UCLV
Universidad Central
"Marta Abreu" de Las Villas


MFC
Facultad de Matemática
Física y Computación

**Departamento de Computación**

**Ingeniería Informática**

## TRABAJO DE DIPLOMA

# API REST PARA EL RECONOCIMIENTO FACIAL DE EMOCIONES (FER REST API)

**Autor**: Dairo López Mollinedo

**Tutores**: Ing. Roberto Vicente Rodríguez

Lic. Claudia Rodríguez Rodríguez

Santa Clara, Cuba, junio 2019

UCLV
Universidad Central
"Marta Abreu" de Las Villas

MFC
Facultad de Matemática
Física y Computación

**Informatics Department**

**Informatics Engineering**

## DIPLOMA THESIS

# FACIAL EMOTION RECOGNITION REST API (FER REST API)

**Author**: Dairo López Mollinedo

**Mentors**: Ing. Roberto Vicente Rodríguez

Lic. Claudia Rodríguez Rodríguez

Santa Clara, Cuba, junio 2019


Hago constar que el presente trabajo fue realizado en la Universidad Central “Marta Abreu” de Las Villas como parte de la culminación de los estudios de la especialidad de **Ingeniería Informática**, autorizando a que el mismo sea utilizado por la institución, para los fines que estime conveniente, tanto de forma parcial como total y que además no podrá ser presentado en eventos ni publicado sin la autorización de la Universidad.

_____

Firma del autor

Los abajo firmantes, certificamos que el presente trabajo ha sido realizado según acuerdos de la dirección de nuestro centro y el mismo cumple con los requisitos que debe tener un trabajo de esta envergadura referido a la temática señalada.

| _____ | _____ | _____ |
|---|---|---|
| Firma del tutor | Firma del tutor | Firma del jefe del Dpto |

"*El mayor enemigo del conocimiento no es la ignorancia, sino la ilusión del conocimiento.*"

***Stephen Hawking***

"*Nuestra recompensa se encuentra en el esfuerzo y no en el resultado. Un esfuerzo total es una victoria completa.*"

***Mahatma Gandhi***

"*Todos nuestros sueños se pueden volver realidad si tenemos el coraje de perseguirlos.*"

***Walt Disney***

*A mis padres Danay y Deivis por confiar siempre en mí y apoyarme incondicionalmente en mi sueño de convertirme en un profesional. Por todos sus consejos, preocupación, esfuerzos, sacrificios realizados durante mis estudios y darme aliento cuando más lo necesitaba en el transcurso de la carrera, durante la realización de este trabajo de diploma y en la vida. A mi padre por ser un gran ejemplo como profesional y estar siempre pendiente de mis estudios y hacer realidad su anhelo de que siguiera sus pasos y me formara como ingeniero. A mi madre por su ternura, dedicación y preocupación por mí, la carrera y estudios. A ambos por estar siempre presentes y ser unos padres maravillosos.*

*A mi hermano Dariel por su gran apoyo y aliento dado para que cumpliera la meta de convertirme en ingeniero.*

*A mi esposa Beatriz por estar siempre a mi lado, por su apoyo, consejos, alentarme en los momentos más difíciles, preocupación en mis estudios, en mi formación como ingeniero, por su amor, dulzura y ser una constante inspiración. Por su optimismo siempre en la realización del trabajo de diploma, ser una gran compañera y una persona única para mí.*

*A mis abuelos Marcos y María Elena por todo su amor, dedicación, apoyo durante todos mis estudios y en mi formación universitaria y por sus consejos en la voz de la experiencia y ser ejemplos para mí.*

*A mis abuelos Aida y Raúl que, aunque no estén presentes en este mundo estarían muy orgullosos de mí. De ellos fueron muchas las enseñanzas y consejos que adquirí las cuales me han hecho superar obstáculos y ser más fuerte. Por sus preocupaciones durante mi vida de estudiante, su amor brindado y por las experiencias adquiridas y vividas junto a ellos.*

*A mis tíos Idalberto y Mayelín por toda su preocupación en mi formación como profesional y su apoyo incondicional en todo este trabajo de diploma.*

*A toda mi familia, por su esfuerzo, aliento y preocupación durante toda la carrera, en especial en la realización de este trabajo de diploma que nunca dejaron de apoyarme.*

*A mis tutores el Ing. Roberto Vicente Rodríguez y la Lic. Claudia Rodríguez Rodríguez por toda su magnífica conducción científica e incondicional apoyo durante la investigación. Por la constante dedicación, esfuerzos, alientos y sus consejos en la realización de este trabajo de diploma. Ambos son ejemplos de excelentes profesionales, las experiencias brindadas, así como sus críticas en esta investigación, fueron de gran enseñanza. Muchas gracias a los dos por dedicar parte de su tiempo libre a la investigación, siempre estar presentes cuando más lo necesitaba y mantener su confianza en mí para desarrollar este trabajo de diploma.*

*Al Lic. Roberto Cabrera Álvarez por sus excelentes consejos, incondicional apoyo y críticas bien recibidas sobre la investigación científica que fueron de mucha ayuda.*

*A la Dra. Martha Beatriz Boggiano Castillo que nunca dijo que no y me brindó su ayuda cuando lo necesitaba en la investigación, dedicando horas de su trabajo. Por sus consejos y apoyo brindados durante el transcurso de la carrera, al ser mi profesora y en el tiempo que trabajé como su alumno ayudante. Es un gran ejemplo de educadora y una profesional muy preparada.*

*A todo el claustro de profesores de la Facultad de Matemática, Física y Computación de la Universidad Central "Marta Abreu" de Las Villas que me impartieron docencia y contribuyeron a mi formación como profesional con su alta preparación académica.*

*A todos mis compañeros de la carrera que juntos pasamos estos cincos años y que vivimos una gran cantidad de experiencias, que entre todos nos ayudamos mutuamente en los momentos más complejos para llegar hasta el final de la meta.*

*A todas aquellas personas que me han apoyado, dado aliento, colaborado y han permanecido junto a mí durante toda la carrera y en la realización del trabajo de diploma, que de una forma u otra han contribuido a mi formación profesional.*

*A todos muchas gracias.*

## RESUMEN

A nivel global existen muchas investigaciones que avanzan en la comprensión de las emociones para mejorar la experiencia de las personas al interactuar con una aplicación. Estos trabajos abarcan áreas muy diferentes entre sí, desde seguridad, salud, educación y robótica. En la Universidad Central "Marta Abreu" de Las Villas se creó una biblioteca para el reconocimiento facial de emociones llamada Feel UCLV. Para acceder desde otras aplicaciones a dicha biblioteca se necesita desarrollar un entorno de comunicación. Con el fin de simplificar la programación de métodos y funciones que permitan reconocer las emociones expresadas en el rostro humano para ser empleados en diferentes aplicaciones se decidió la creación de una API REST. Esta permite la detección automática de emociones faciales en tiempo real en aplicaciones de Computación Afectiva. Reconoce las seis emociones universales alegría, tristeza, miedo, ira, sorpresa y asco. Facilita el uso sin que el desarrollador tenga dominio de este tema. Está implementado en el lenguaje de programación Python mediante el framework FastAPI. Facial Emotions Recognition REST API incluye para la clasificación Red Neuronal Perceptrón Multicapa, Máquina de Soporte Vectorial, K Vecinos más Cercanos, Bosque Aleatorio, Árbol de Decisión, y Naive Bayes. Para evaluar los clasificadores utiliza los conjuntos de datos Conh-Kanade y KDEF alcanzando resultados satisfactorios.

**Palabras claves**: clasificadores, Computación Afectiva, detección automática, emociones faciales, API.

**ABSTRACT**

Globally there is a lot of research that advances the understanding of the emotions to improve people's experience when interacting with an app. Those works cover very different areas of each other, from safety, health, education and robotics. At the Universidad Central "Marta Abreu" de Las Villas a library was created for the facial recognition of emotions called Feel UCLV. To access such library from other applications a communication environment needs to be developed. In order to simplify the programming methods and functions that allow to recognize emotions expressed in the human face to be used in different applications it was decided to create a REST API. This allows automatic detection of facial emotions in real time in Affective Computing applications. It recognizes the six universal emotions happiness, sadness, fear, anger, surprise and disgust. Makes it easier to use without the developer having mastery of this topic. It is implemented in the Python programming language using the FastAPI framework. Facial Emotions Recognition REST API includes for the classification Multilayer Perceptron, Support Vectorial Machine, K Nearest Neighbor, Random Forest, Decision Tree, and Naïve Bayes. To evaluate the classifiers, use the Conh-Kanade and KDEF databases, achieving satisfactory results.

**Keywords**: Affective Computing, automatic detection, API, classifiers, facial emotions.

## Tabla de Contenidos

## Lista de Figuras

## Lista de Tablas

## INTRODUCCIÓN

El ser humano es complejo por naturaleza, una de las variantes más confusas y complejas de estudiar han sido sus emociones. El origen que las provoca es tan diverso como incierto por lo que ha levantado un sinfín de discusiones a lo largo de la historia. En el amplio espectro de la conducta del humano, las emociones han representado un tema interesante para el análisis. Primero, debido a que controlan conductas complejas en el humano como la motivación y el aprendizaje. Segundo, porque la mayoría de las enfermedades psiquiátricas más devastadoras involucran desórdenes emocionales. Durante el desarrollo emocional, la cultura y la sociedad tienen gran influencia en las emociones, pues regulan su expresión. Cada vez se conoce más la influencia que las emociones tienen sobre los procesos de pensamiento racional. Estas influyen de manera directa sobre la toma de decisiones, la comunicación, el aprendizaje, pueden mejorar la percepción y otros procesos de racionalidad que el ser humano genera.

En los últimos años se ha desarrollado un creciente interés en mejorar todos los aspectos de la interacción entre humanos y computadoras. Muchos estudios para distinguir emociones faciales han favorecido la creación de entornos para esta interacción. Especialmente en el área del reconocimiento de emociones humanas al observar expresiones faciales. Se ha estado investigando mucho sobre cómo desarrollar tecnología y dispositivos capaces de reconocer, interpretar y simular las emociones humanas. Esto con la finalidad de diseñar tecnología que simule y realice las funciones propias de una persona, brindando al usuario un ambiente personalizado y adaptado a sus necesidades cognitivas y emocionales.

El rol del reconocimiento automático de emociones está creciendo de forma continua actualmente. Esto se debe a que se ha aceptado la importancia que tiene la reacción de la computadora a los estados afectivos del usuario en la interacción persona-computadora. A medida que las computadoras se vuelven más y más sofisticadas, ya sea a nivel profesional o social, se torna más importante que estas sean capaces de interactuar de forma natural, similar a como se interactúa con otros agentes humanos. La característica más importante de la interacción humana que garantiza la realización del proceso de forma natural, es el medio por el cual se puede inferir el estado

emocional de otros. Esto permite ajustar los patrones de comportamiento y respuestas, optimizando el proceso interactivo.

La interacción de la persona con la computadora puede ser mejorada en gran medida si se toma en cuenta el estado emocional del ser humano, haciéndola más cercana y natural. Estos elementos están ausentes en la gran mayoría de los sistemas actuales. Diversos autores tratan este tema, y la investigación en este campo ha sido numerosa en la última década. La autora Baldasarri (2016) plantea que la Dra. Rosalind Picard, en su libro "Affective Computing" publicado en 1997, argumenta la necesidad de tener en cuenta los factores emocionales en el diseño del software y plantea que el reconocimiento de los estados afectivos del usuario es un problema muy importante en el ámbito de la interacción humano-computadora . Es por este motivo, que se considera que las máquinas deben incluir este tipo de inteligencia de reconocer el estado afectivo dado ciertas señales psicológicas.

El término Computación Afectiva se atribuye a Rosalind Picard, profesora e investigadora del Instituto Tecnológico de Massachusetts, quién en 1997 lo definió como una disciplina dentro del campo de la Inteligencia Artificial que intenta desarrollar métodos de computación focalizados en reconocer las emociones humanas y generar emociones sintéticas, así lo figuran los autores Arboleda Rodríguez, Gallar Pérez and Barrios Queipo (2017). La computación afectiva es el área que se centra en investigar cómo se usan las emociones en los sistemas informáticos. Consiste en el desarrollo de sistemas inteligentes capaces de dotar a las computadoras la habilidad de reconocer, interpretar, procesar y expresar las emociones humanas (Baldassarri, 2012).

Se han desarrollado hasta la fecha sistemas que reconocen las emociones a través de diferentes canales humanos, como expresiones faciales, señales de habla, señales fisiológicas. Dentro de estas últimas se encuentran la excitación automática, la frecuencia cardíaca y el sudor, diversos gestos, voz, pulsaciones, presión sanguínea, resistencia de la piel, algunas actividades de electromiografía y postura. Todas con la finalidad de interpretar estas emociones para múltiples fines, entre ellos la educación, salud, seguridad y más.

La computación afectiva es actualmente una de las investigaciones más activas y con una atención cada vez más intensa. Este fuerte interés es impulsado por un amplio espectro de aplicaciones prometedoras en muchas áreas tales como realidad virtual, vigilancia inteligente, interfaz

perceptiva plasmado en Tao and Tan (2014), además de ser muy utilizada en ramas como el marketing y la industria del entretenimiento (Rouse, 2019). Esta disciplina se refiere al conocimiento multidisciplinar, como la psicología, cognitiva, fisiología y ciencias de la computación (Banafa, 2016). Está tratando de asignar a las computadoras las capacidades humanas de observación, interpretación y generación de rasgos afectivos (Tao and Tan, 2014). Es un tema importante para la interacción armoniosa entre el hombre y la computadora.

El rostro es uno de los más ricos canales de expresión en los seres humanos, el cual comunica emociones y señales sociales. Dentro de la literatura se suelen utilizar dos modelos que describen, explican y clasifican las expresiones faciales y sus emociones. Por un lado, el modelo categórico y por el otro, el modelo continuo o dimensional. El primero usa etiquetas o clasificadores para establecer la emoción dentro de categorías específicas como ira, asco, miedo, alegría, tristeza y sorpresa. El segundo, en cambio, explica cómo las emociones pueden ser vistas en un espacio continuo donde se pueden representar infinidad de puntos e intensidades (Bosquez Barcenes *et al.*, 2018).

Uno de los trabajos pioneros en este campo es el de Paul Ekman. Este sugiere que existen seis expresiones faciales prototípicas básicas reconocidas universalmente. Estas son: ira, asco, miedo, alegría, tristeza y sorpresa (Ekman and Rosenberg, 2012). El reconocimiento automatizado de estos seis estados básicos es el primer paso para implementar una solución, que, a diferencia de otras, es universal, ya que es común a todos los seres humanos.

En este sentido existen un conjunto de sistemas, bibliotecas (paquetes) e módulos que implementan métodos y funciones que sirven para el proceso de reconocimiento de emociones, a partir del rostro humano. Ejemplo de las mismas son OpenCV (team OpenCV, 2019) , Dlib (Dlib, 2019), Scikit-Learn (*scikit-learn: machine learning in Python scikit-learn 0.20.3 documentation*, 2019), etc. Sin embargo, el empleo de sus recursos requiere de un conocimiento profundo de las mismas y de los lenguajes en que están implementadas (C++, Python). Resultaría conveniente contar con un entorno de trabajo que facilite el trabajo de los desarrolladores de esta rama y sirva para los diferentes lenguajes de programación para que puedan enfocarse en cuestiones superiores. De esta forma les sería más fácil acceder al reconocimiento de emociones faciales desde cualquier lenguaje

de programación y en cualquier sistema operativo sin necesidad de conocer detalles de implementación.

Específicamente en este trabajo se necesita desarrollar el entorno sugerido anteriormente para acceder desde cualquier lenguaje de programación a la biblioteca Feel UCLV. Implementada en Python, con el fin de simplificar la programación de métodos y funciones que permitan reconocer las emociones expresadas en el rostro humano para ser empleados en diferentes aplicaciones. Dicho entorno podrá ser utilizado, por ejemplo, en sistemas de vigilancia para la seguridad de las empresas que lo utilicen en el país, en comercio electrónico, educación y la medicina.

Teniendo en cuenta las consideraciones descritas anteriormente se plantea el siguiente **Problema de Investigación**:

¿Cómo acceder con facilidad a la biblioteca Feel UCLV desde aplicaciones implementadas en cualquier lenguaje de programación para la detección de emociones reflejadas en el rostro humano?

A partir del problema expuesto se propone el **Objetivo General** que le dará solución:

Implementar una Interfaz de Programación de Aplicaciones (API) utilizando un framework de Python para la comunicación entre la biblioteca Feel UCLV y aplicaciones implementadas en cualquier lenguaje de programación.

Para contribuir con el desarrollo del propósito general este queda desglosado en los siguientes **Objetivos Específicos**:

1. Identificar los fundamentos teóricos y metodológicos relacionados con la detección de emociones reflejadas en el rostro humano.
2. Determinar el tipo de API conveniente para comunicar la biblioteca Feel UCLV y aplicaciones implementadas en cualquier lenguaje de programación.
3. Determinar framework de Python con el que se implementará la API para consumir los métodos de la biblioteca Feel UCLV.
4. Implementar la API para la comunicación entre la biblioteca Feel UCLV y aplicaciones implementadas en cualquier lenguaje de programación.

Como guías para el desarrollo del trabajo se desglosan las siguientes **Preguntas de Investigación**:

1. ¿Cuáles son las tendencias actuales en la detección y clasificación de las emociones del rostro humano, así como las tecnologías que permiten su reconocimiento?
2. ¿Cuáles son los tipos de API que permiten consumir los métodos de la biblioteca Feel UCLV desde otras aplicaciones?
3. ¿Qué frameworks ofrece Python para implementar la API que servirá para la comunicación entre la biblioteca Feel UCLV y otras aplicaciones?
4. ¿Cómo se puede establecer el acceso a la biblioteca Feel UCLV desde aplicaciones implementadas en cualquier lenguaje de programación?

**Justificación de la Investigación**

A nivel global existen muchas investigaciones que avanzan en la comprensión de las emociones para mejorar la experiencia de las personas al interactuar con una aplicación informática. Estos trabajos abarcan áreas muy diferentes entre sí, desde seguridad, salud, educación, entretenimiento, robótica o marketing. La evaluación en tiempo real de emociones como el estrés, el aburrimiento o la distracción puede ser de gran valor en trabajos en los que se realizan tareas repetitivas, pero en los cuales la atención es crucial, como por ejemplo en control de tráfico aéreo o la supervisión de una planta nuclear.

En este sentido la Facial Emotions Recognition REST API permite consumir los métodos de una biblioteca implementada para el reconocimiento de expresiones faciales. Mediante esta API se puede detectar las emociones faciales en el rostro humano en diferentes aplicaciones que lo necesiten, independientemente del lenguaje de programación y el sistema operativo. Permite ser usada sin que el desarrollador tenga dominio de este tema. Por ello el **aporte práctico** de esta investigación lo constituye la implementación de una API REST para el reconocimiento y clasificación de emociones faciales.

La API se puede utilizar en el país en softwares referentes al campo de la psicología para saber el estado afectivo de pacientes con autismo, Parkinson, síndrome Down, con problemas mentales y afectaciones cerebrales. En sistemas de vigilancia de salas de terapia intensiva de los hospitales para detectar el dolor en pacientes con problemas de comunicación. En aplicaciones destinadas a la seguridad vial en sistemas de vigilancia a los conductores de empresas como Ómnibus Nacionales, Transgaviota, Transtur, CubaTaxi y choferes particulares, entre otros, para saber si se

encuentran estresados o presentan síntomas de cansancio y evitar accidentes de tránsito. En el campo de la educación, en el llamado aprendizaje electrónico, se puede incluir en la plataforma Moodle para conocer el estado emocional de los estudiantes ante una asignatura impartida por el profesor. En el área del comercio electrónico para verificar si productos ofertados en tiendas electrónicas cubanas tienen o no la aceptación requerida por el cliente y en el mercado.

De ahí el **aporte económico** de una solución de este tipo, pues disminuye los costos de producción al requerirse menos horas-hombre para realizar el reconocimiento de emociones faciales en las aplicaciones mencionadas anteriormente. Contribuye al aumento de la productividad reduciendo el tiempo de implementación y el mantenimiento de los sistemas donde se use. Gracias a esta aplicación el desarrollador puede centrarse en las funciones principales de su sistema, olvidándose del reconocimiento de emociones. Además, es una herramienta libre de costo, a diferencia de la mayoría de las API y bibliotecas relacionadas a la computación afectiva existentes en el mercado que necesitan una clave o licencia costosa para ser usadas. No necesita la presencia de ambientes profesionales para detectar el rostro humano, con el uso de una cámara de baja calidad se puede realizar esta acción con gran exactitud. Le permite al país implementar aplicaciones como las antes mencionadas con gran facilidad y acceso, contribuyendo al aumento del valor agregado de la misma. Es de código libre, con facilidades de uso y entendimiento, al igual que las bibliotecas utilizadas, por lo que permite ser mejorada cómodamente, aumentando el **aporte social** de la misma.

En cuanto a la estructura de esta investigación consta con introducción, tres capítulos, conclusiones, recomendaciones, referencias bibliográficas y por último los anexos.

En el capítulo 1 se definen los conceptos básicos relacionados con los sistemas de detección automática de emociones. Se analizan las vías que existen para consumir los servicios que brinda una biblioteca para el reconocimiento de emociones faciales de la mejor forma posible y la implementación.

En el capítulo 2 se describen la arquitectura y diseño de la aplicación, las herramientas y funciones que se utilizan para su implementación. Además, se detallan los pasos para la instalación y ejecución de la misma.

En el capítulo 3 se presentan las pruebas realizadas a la aplicación mediante el programa Weka. Se ajustan los valores de los clasificadores para obtener mejores resultados en la detección de emociones. Además, se realiza una comparación entre dos bases de datos para saber la que mejores resultados arroja.

Luego se exponen las conclusiones y las recomendaciones de la investigación con las referencias bibliográficas consultadas y los anexos.

# 1

# EL USO DEL RECONOCIMIENTO DE EMOCIONES FACIALES MEDIANTE UNA API REST

# 1. EL USO DEL RECONOCIMIENTO DE EMOCIONES FACIALES MEDIANTE UNA API REST

En este capítulo se realiza una revisión bibliográfica sobre las emociones humanas y la detección automática de ellas. Así como las aplicaciones de la computación afectiva. Se analiza el funcionamiento de la biblioteca de reconocimiento de emociones faciales Feel UCLV. Se hace referencia a los clasificadores de aprendizaje automático utilizados en la detección de emociones del rostro humano. Se presentan los diferentes protocolos y tipos de API para consumir los métodos de la biblioteca. Además, los frameworks de Python para la creación de las API.

## *1.1 Las emociones humanas*

Las expresiones faciales de emociones resultan de gran importancia cuando se trata de la interacción con otras personas. Pues revelan estados mentales complejos que son transmitidos a los demás (Ekman and Oster, 1979; Arango de Montis *et al.*, 2013). El rostro humano es considerado el principal sistema de señales para mostrar las emociones, además es el área más importante y compleja de la comunicación no verbal. Las emociones mejoran la percepción, la toma de decisiones e influyen en el pensamiento racional (Bosquez Barcenes *et al.*, 2018).

Desde la segunda mitad del siglo XX hasta la actualidad, los estudios de numerosos psicólogos han determinado que todos los seres humanos, independientemente del entorno sociocultural y de las características fisiológicas, manifiestan de forma similar las emociones básicas.

Uno de los más famosos investigadores que obtuvo excelentes resultados y marcó una línea de investigación en las emociones fue Charles Darwin en su libro *"La expresión de las emociones en hombres y animales (1872)"* (Darwin, 1872). A partir de la relación que estableció entre emoción y conducta, aludió a la existencia de ocho emociones básicas (alegría, miedo, malestar, sorpresa, interés, rabia, asco y vergüenza). Señaló que la ira, tristeza o el asco no son solo humanas, son compartidas con otros animales (Rueda Extremera, 2017).

Otra investigación relevante la realizó James Papez en el año 1937. Sugirió un esquema anatómico para el circuito neuronal de la emoción, conocido como el circuito de Papez (véase Anexo 1) (Rueda Extremera, 2017). Años después la teoría fue reformulada por Paul MacLean.

### 1.1.1 Concepto y clasificación

Las diferentes concepciones de las emociones se destacan por los acentos puestos en las distintas cualidades de las mismas. No siempre se excluyen explícitamente entre sí, pero tampoco existe una definición unívoca aceptada.

Según la Real Academia Española define la emoción como una "alteraciones del ánimo intensas y pasajeras, agradables o penosas que van acompañadas de cierta conmoción somática" (Arboleda Rodríguez, Gallar Pérez and Barrios Queipo, 2017).

La definición psicológica plantea "las emociones son reacciones psicológicas que representan modos de adaptación a ciertos estímulos del individuo" (Peinador Yubero, 2016).

El psicólogo Paul Ekman enuncia un concepto moderno y abarcador acerca de la emoción definiéndola como: "Un proceso de tipo particular de valoración automática influida por el pasado evolutivo y personal, en el que se siente que está ocurriendo algo importante para el bienestar, produciendo un conjunto de cambios físicos y comportamentales para hacerse cargo de una situación" (Ekman and Oster, 1979).

Después de la revisión exhaustiva de la literatura y de acuerdo con las definiciones anteriores se define la emoción como un fenómeno complejo multidimensional que afecta a los sistemas fisiológicos, conductuales y cognitivos, de origen innato o influenciado por la experiencia y que contribuye a dirigir y organizar la conducta.

Uno de los primeros científicos que se atrevió a retomar el estudio de las emociones y comprobar las hipótesis de Darwin sobre la expresión de las emociones fue el psicólogo y antropólogo estadounidense Dr. Paul Ekman. Este publicó con su colega Wallace Friesen en 1978 el Facial Action Coding System (Ekman and Friesen, 1978). En este postulan la existencia de seis emociones básicas en todos los seres humanos independiente de las raíces culturales cuya expresión se proyecta en el área facial, según Ekman and Friesen (1975). Las emociones básicas según ellos serían alegría, tristeza, enojo, sorpresa, miedo y asco, expuesto por Arboleda Rodríguez, Gallar Pérez and Barrios Queipo (2017).

Estas seis emociones universales definidas por Paul Ekman proporcionan la base de todos los estudios que se desarrollan hoy en día en el reconocimiento automático de expresiones faciales y en toda el área de trabajos realizados en la Computación Afectiva.

Diversos estudios sobre la universalidad de las emociones han demostrado que los sujetos reconocen las emociones de manera adecuada sin ser resultado del azar, sin embargo, ningún estudio ha reportado niveles perfectos de identificación (Iglesias Hoyos, del Castillo Arreola and Muñoz Delgado, 2016).

Entre las emociones primarias se encuentran las básicas o elementales, definidas por Ekman y Darwin principalmente, el miedo, sorpresa, ira o rabia, asco, tristeza y alegría. Entre las secundarias que son influenciadas por el pensamiento y derivadas de las primarias se encuentran la culpa, celos, placer, preocupación, amor, diversión, soledad, rencor entre otras.

### 1.1.2 **Computación Afectiva**

La Computación Afectiva (Affective Computing) hace referencia al diseño de sistemas y dispositivos que pueden reconocer, interpretar, procesar y generar emociones humanas para mejorar la interacción entre el usuario y la computadora (Baldasarri, 2016). La Dra. Rosalind W. Picard profesora e investigadora del Instituto Tecnológico de Massachusetts (MIT) fue la primera en acuñar el término. Lo definió como una disciplina dentro del campo de la Inteligencia Artificial que intenta desarrollar métodos de computación focalizados en reconocer las emociones humanas y generar emociones sintéticas (Arboleda Rodríguez, Gallar Pérez and Barrios Queipo, 2017).

Esta área se ocupa del reconocimiento de emociones (y de expresiones emotivas) humanas por parte de una computadora y la simulación (o generación) de estados y expresiones emocionales con computadoras.

Esta área permite la interpretación de la emoción, el estado de ánimo, la actitud y es capaz de suplir las preferencias y necesidades del usuario. Es una parte esencial dentro de la interacción Humano-Computadora (HCI) (Sokolova and Fernández Caballero, 2015).

Los seres humanos son capaces de expresar el afecto a través de una serie de canales como la expresión facial, los movimientos corporales, comportamiento de la voz y otras señales fisiológicas, como la frecuencia cardíaca y el sudor, etc (Tao and Tan, 2014). Hasta la actualidad se han realizado trabajos satisfactorios con sistemas unimodales como el procesamiento emocional del habla, en el reconocimiento de expresiones faciales, gestos corporales y movimiento y se ha analizado texto en busca de estados afectivos (Poria *et al.*, 2017). Además, se ha realizado la detección de emociones de forma multimodal, es decir, combinando dos o más medios. Se han

obtenido resultados muy superiores, a los sistemas unimodales, pero con un mayor costo y complejidad, así lo afirma Bosquez Barcenes *et al.* (2018).

El rostro es uno de los más ricos canales de expresión que comunica emociones y señales sociales. Dentro de la literatura se suelen utilizar dos modelos que describen, explican y clasifican las expresiones faciales y sus emociones. Por un lado, el modelo categórico y por el otro, el modelo continuo o dimensional. El primero usa etiquetas o clasificadores para establecer la emoción dentro de categorías específicas como alegría, sorpresa, miedo. El segundo en cambio explica cómo las emociones pueden ser vistas en un espacio continuo, en el cual se pueden representar infinidad de puntos e intensidades (Bosquez Barcenes *et al.*, 2018).

Existen, básicamente, dos aplicaciones fundamentales para esta tecnología:

- Ayudar a las personas a tomar mejores decisiones.
- Lograr un mejor relacionamiento entre personas y máquinas, haciendo que ambos trabajen conjuntamente de manera eficiente.

Algunos ejemplos de aplicaciones de la Computación Afectiva es su uso en la industria robótica, con la creación de sistemas robóticos capaces de procesar información afectiva, como las mascotas virtuales de compañía. Además, en control de tráfico aéreo o la supervisión de una planta nuclear. En este ámbito, hay herramientas automáticas que permiten analizar si un conductor está enfadado, estresado o se está durmiendo. Dependiendo el caso, pone música, le habla para despertarlo o incluso emite alarmas (Baldasarri, 2016).

En la industria automovilística donde un coche puede controlar las emociones de todos los ocupantes y aplicar medidas de seguridad adicionales. Casos exitosos son las compañías Toyota con la colaboración del Grupo de Computación Afectiva del Instituto Tecnológico de Massachusetts (MIT).

Es ampliamente utilizada en áreas como la educación con los tutores afectivos y en el llamado aprendizaje electrónico para saber el estado emocional de los estudiantes si se encuentran aburridos, interesados, frustrados o contentos ante una actividad impartida por el profesor. En la medicina se han realizado aplicaciones adaptables a personas con autismo. En el área del entretenimiento, pueden crearse videojuegos que al detectar un determinado nivel de estrés disminuyan la intensidad de la partida, o dar mayor realismo en juegos como los SIMS.

Hay avances muy relevantes en los sensores en la actualidad, que capturan información de las emociones de las personas, como los *eye-trackers,* que permiten obtener información de la dilatación y el seguimiento de las pupilas; los micrófonos permiten capturar el lenguaje y las variaciones en entonación, tono o volumen de la voz; los sensores que registran medidas fisiológicas como la respiración, el pulso, la resistencia galvánica de la piel, la temperatura corporal; y los electrodos detectan la actividad cerebral (Paz Pellat, 2017).

También se aplica en el Marketing, creando anuncios que generen determinada emoción en su público receptor. En reproductores de música que seleccionan las pistas según el estado emocional del usuario.

## *1.2 Biblioteca para el reconocimiento facial de emociones: Feel UCLV*

En el año 2018 en la Universidad Central "Marta Abreu" de Las Villas se realizó un trabajo de diploma de la Facultad de Matemática, Física y Computación titulado "Desarrollo de una biblioteca para el reconocimiento de emociones faciales" de los autores Guillermo Soto Gómez y Gerardo Martínez Rodríguez (Soto Gómez and Martínez Rodríguez, 2018), en el que se implementó la biblioteca Feel UCLV. A continuación, se abordarán las características y aspectos más relevantes de la biblioteca de reconocimiento de emociones faciales y otros sistemas de detección automática de emociones que existen, las etapas presentes en el proceso.

### 1.2.1 **Finalidad**

La biblioteca Feel UCLV fue desarrollada con el propósito de que los desarrolladores la emplearan en la programación de aplicaciones para el reconocimiento de emociones faciales. Además, se puede utilizar en las implementaciones de programas que empleen la Computación Afectiva, donde el programador no tiene que preocuparse por detectar, interpretar, procesar y clasificar la imagen. Con Feel UCLV el desarrollador no tiene que tener conocimientos específicos sobre Computación Afectiva ni de otros temas. El desarrollador puede centrarse en cuestiones superiores de su implementación, pueden realizar otras funcionalidades. La biblioteca está implementada en el lenguaje de programación Python.

### 1.2.2 Etapas de los sistemas automáticos de detección de emociones

La biblioteca Feel UCLV como la mayoría de los sistemas automáticos de reconocimiento de emociones faciales se dividen en varias etapas (véase Figura 1.1).

Primeramente, se realza la imagen para aumentar la efectividad en la detección del rostro. Luego se detecta el rostro humano, se extraen de este las características más significativas y, por último, se realiza la clasificación. El clasificador entrenado detecta la emoción en función de las particularidades de los rostros contenidos en cada imagen.

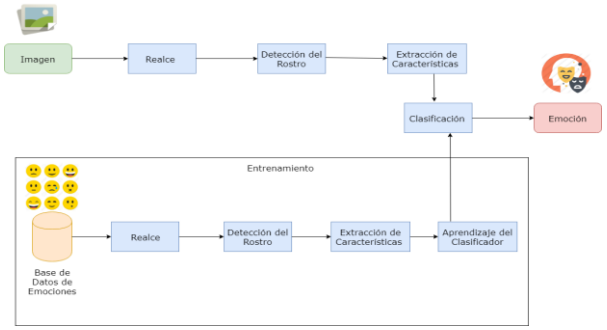



**Figura 1.1 Etapas de un sistema automático de detección de emociones faciales**

El entrenamiento del clasificador es parte de la primera fase. Consiste en entrenar el clasificador mediante un conjunto de imágenes, de las que se conoce su emoción, y con ellas aprende las características asociadas a cada una de las emociones. Para ello se toma cada imagen y se realizan los pasos: realce de la imagen, detección del rostro, extracción de características y asociación de la emoción representada. Una vez que el clasificador ha sido entrenado el sistema ya se encuentra preparado para realizar en reconocimiento de emociones.

El realce y normalización de imágenes tiene como objetivo principal realizar un procesamiento que permita obtener una imagen más apropiada para detectar el rostro y obtener el vector de características. En la mayoría de los sistemas se realiza una ecualización del histograma utilizado para lograr una distribución más uniforme entre el número de píxeles asociado a cada nivel de intensidad dentro de la escala de grises.

En la etapa de detección de rostros se reconoce la cara en la imagen y se devuelve las coordenadas con el rostro detectado. Existen dos tipos de algoritmos según Hatem, Beiji and Majeed (2015), los basados en características y basados en imágenes. El algoritmo de detección facial utilizado en Feel

UCLV se encuentra implementado en la biblioteca Dlib. Es capaz de reconocer con rapidez y exactitud varios rostros humanos frontales en una imagen o video, devolviendo como resultado las coordenadas del rostro detectado.

En la etapa de extracción de características la imagen facial se procesa para extraer un vector que describe las características más significativas del rostro. Resulta imposible determinar cuáles son las características más importantes para construir el vector para obtener mejores resultados en la clasificación, por tanto, los métodos de extracción de características utilizan diferentes enfoques. Otro factor importante a tener en cuenta en esta etapa son las técnicas de clasificación de inteligencia artificial que son sensibles a los diferentes rasgos extraídos.

Los métodos de extracción de características en imágenes de rostros utilizados para la detección de emociones faciales varían tanto en las características que extraen como en el procedimiento utilizado para extraerlos. Existen varios métodos como los basados en apariencia (Holísticos), basados en características (Analíticos) que se dividen en dos categorías: basados en geometría y en modelos. Por último, los híbridos que son basados en apariencia y características.

En la última etapa de clasificación, el vector de características es utilizado para entrenar un clasificador. El vector de características se compara con los vectores de rasgos extraídos de los rostros de la base de datos de emociones con que se hizo el aprendizaje del clasificador. De esta manera, mediante la semejanza de ambos vectores se logra aproximar la clasificación de la emoción correspondiente al rostro de la imagen. En la biblioteca Feel UCLV en la etapa de clasificación se implementa el aprendizaje automático de tipo supervisado, por ser el más utilizado en la literatura para la detección automática de emociones.

Un buen resultado en la detección automática de emociones depende de un satisfactorio realce y normalización de la imagen, escoger algoritmos de detección de rostro y de extracción de características con resultados satisfactorios, y un clasificador con probabilidades altas de acierto sobre cada emoción.

### 1.2.3 Funcionalidades

La biblioteca Feel UCLV está implementada en el lenguaje de programación Python y depende de las bibliotecas NumPy, OpenCV, Dlib y Scikit-learn para su funcionamiento. Mediante el método `describe` se realiza el realce y normalización de la imagen. Para normalizar la iluminación en la

imagen se realiza una ecualización adaptativa del histograma. Este es un método que mejora la ecualización ordinaria al transformar cada píxel por el resultado de una función de transformación derivada de su región vecina (Magudeeswaran and Singh, 2017). Para evitar la propagación del ruido, se aplica el método de ecualización adaptativa del histograma con contraste limitado. Este se aplica mediante un método de la biblioteca OpenCV a las imágenes después de llevarla a escala de grises mediante otro método de la misma biblioteca. Básicamente la ecualización adaptativa del histograma con contraste limitado consiste en distribuir los píxeles de un nivel de intensidad que sobrepase un límite dado, en otros niveles de intensidad de manera uniforme antes de aplicar la ecualización del histograma.

La biblioteca Feel UCLV en la etapa de detección del rostro hace uso del método `get_frontal_face_detector` de la biblioteca Dlib. Este método utiliza el algoritmo de histograma de orientación de gradientes (siglas en inglés HOG), combinado con un clasificador lineal y el método de detección de imagen piramidal y ventana deslizante. Mediante este método se obtiene el rostro humano contenido en una imagen. La biblioteca está implementada para procesar imágenes frontales de una sola persona.

En la fase de extracción de características la biblioteca utiliza un algoritmo de Marcas Faciales expuesto en Kazemi and Sullivan (2014). Para esto hace uso de la biblioteca Dlib. El método correspondiente en Dlib es `shape_predictor` que carga el archivo shape_predictor_68_face_landmarks.dat y permite detectar 68 marcas faciales en el rostro humano (véase Figura 1.2).

**Figura 1.2 Representación de las 68 marcas faciales en el rostro humano (Korshunov and Marcel, 2018)**

El vector características se obtiene tanto de las imágenes del conjunto de entrenamiento como de la imagen que se desea clasificar. Para ello en la biblioteca Feel UCLV se utiliza el método `describe`. Este es el de mayor relevancia en la biblioteca. Recibe la dirección de una imagen y devuelve el vector de características que la representa con las 68 coordenadas de los puntos de

referencia facial. Además, que él realiza el realce, la detección de rostro y la extracción de rasgos. Tanto para el aprendizaje del clasificador como para reconocer una emoción es aplicable este método.

Otra de las funcionalidades principales que permite la biblioteca es crear un clasificador que será entrenado para detectar la emoción de la imagen. Para ello se necesita un conjunto de datos de entrenamiento. La base de datos de emociones debe estar estructurada por carpetas. Estas corresponden a las diferentes emociones que existen (clases). Dentro de estos directorios deben estar las imágenes relacionadas con la emoción adecuada. Esta composición (véase Figura 1.3) permite realizar un aprendizaje supervisado.

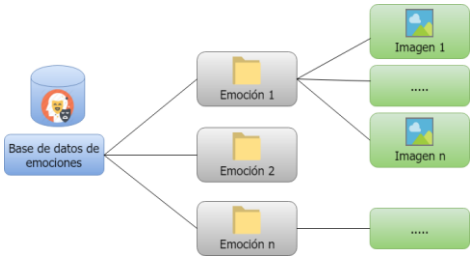



**Figura 1.3 Estructura del conjunto de entrenamiento**

El clasificador que se entrena en la biblioteca Feel UCLV es Máquina de Soporte Vectorial (SVM) con kernel polinomial mediante el método `getEmotionDetector`. La efectividad depende del conjunto de datos de entrenamiento. Se implementa el clasificador mediante la biblioteca de aprendizaje automático Scikit-learn. Por defecto este clasificador está previamente entrenado con 18 imágenes para cada una de las clases: miedo, tristeza, sorpresa, alegría, asco, desprecio, neutral e ira, del conjunto de datos Cohn-Kanade. La biblioteca posee el método `predictEmotion` que recibe como parámetros la dirección de la imagen a clasificar y un clasificador que tiene por defecto entrenado. Este devuelve la emoción detectada una como cadena de texto.

A la imagen que se desea clasificar la emoción se le realiza el realce y normalización para aumentar su efectividad en la detección del rostro. La biblioteca Feel UCLV detecta el rostro humano contenido en una imagen, obteniendo como resultado las coordenadas de la cara. Luego, se realiza la extracción de rasgos y obtención del vector de características, para ser utilizado por el clasificador tanto en la clasificación de la emoción como en la etapa de entrenamiento. Para reconocer la emoción se compara las características extraídas del rostro con las reconocidas en la etapa de entrenamiento del clasificador y se encuentran similitudes. La biblioteca posee una

interfaz visual implementada en PyQt5, para la demostración del funcionamiento general de detección de emociones mediante video utilizando la webcam.

## *1.3 Clasificadores de Aprendizaje Automático*

El Aprendizaje Automático (machine learning) es una rama de la Inteligencia Artificial que se encarga del diseño y desarrollo de algoritmos. Permiten a una computadora mejorar un comportamiento automáticamente a través de la experiencia.

Los algoritmos de aprendizaje automático pueden aprender de cuatro formas distintas: mediante un aprendizaje supervisado, con aprendizaje no supervisado, con aprendizaje semisupervisado o con aprendizaje por refuerzo (Sancho Caparrini, 2017). En la revisión bibliográfica la mayoría de los trabajos realizados sobre Computación Afectiva se implementan mediante aprendizaje supervisado. A continuación, se presentarán algoritmos de clasificación supervisada que son los más utilizados en la detección automática de emociones.

### 1.3.1 K Vecinos más Cercanos (KNN)

K Vecinos más Cercanos (k-nearest neighbors) (Liu, 2017) es un método de clasificación supervisada. Este clasificador es muy simple e intuitivo, y se basa en tomar la información que se desea clasificar y calcular las distancias de entre todos los objetos del conjunto de entrenamiento (véase Anexo 3).

Las predicciones se realizan basándose en los ejemplos más parecidos al que hay que predecir.

La mejor elección de k depende fundamentalmente de los datos; generalmente, valores grandes de k reducen el efecto de ruido en la clasificación, pero crean límites entre clases parecidas. La elección del K a utilizar es muy importante, ya que los resultados pueden variar mucho, por este motivo, es importante conocer bien la información y cómo se calculará la distancia.

Ventajas del algoritmo KNN:

- El coste del aprendizaje es nulo.
- No necesita hacer ninguna suposición sobre los conceptos a aprender.
- Puede aprender conceptos complejos usando funciones sencillas.

Desventajas del algoritmo KNN:

- El coste de encontrar los k mejores vecinos es grande.

- No hay un mecanismo para decidir el valor óptimo para k.
- Lento, si hay muchos datos de entrenamiento.

KNN se usa en la minería de datos (data mining), para el uso de sistemas de recomendación, plataformas de contenido digital y procesos de venta cruzada en comercio electrónico.

### 1.3.2 Máquina de Soporte Vectorial (SVM)

Las máquinas de vectores soporte (SVM, Support Vector Machines) tienen su origen en los trabajos sobre la teoría del aprendizaje estadístico. Aunque originariamente las SVM fueron pensadas para resolver problemas de clasificación binaria, actualmente se utilizan para resolver otros tipos de problemas (regresión, agrupamiento, multiclasificación). También son diversos los campos en los que han sido utilizadas con éxito, tales como visión artificial, reconocimiento de caracteres, categorización de texto e hipertexto, análisis de datos, clasificación de proteínas, procesamiento de lenguaje natural y análisis de series temporales.

Dentro de la tarea de clasificación, las SVMs (Alpaydin, 2014) pertenecen a la categoría de los clasificadores lineales. Utilizan *separadores lineales o hiperplanos*. La búsqueda del hiperplano de separación en estos espacios transformados se hace de forma implícita utilizando las denominadas funciones *kernel*, cuando los datos no son linealmente separables. El sesgo inductivo asociado a las SVMs radica en la minimización del denominado riesgo estructural.

La idea es seleccionar un *hiperplano óptimo* de separación que equidista de los ejemplos más cercanos de cada clase para, de esta forma, conseguir lo que se denomina un *margen máximo* a cada lado del hiperplano. Además, a la hora de definir el hiperplano, sólo se consideran los ejemplos de entrenamiento de cada clase que caen justo en la frontera de dichos márgenes. Estos ejemplos reciben el nombre de *vectores soporte*. Desde un punto de vista práctico, el hiperplano separador de margen máximo ha demostrado tener una buena capacidad de generalización, evitando en gran medida el problema del sobreajuste a los ejemplos de entrenamiento. Para la clasificación basta con determinar a que subespacio pertenece la instancia y asignarle la clase consecuentemente. Un ejemplo gráfico de las características de la SVM mencionados se ilustra en el Anexo 4.

En la mayoría casos reales no pueden ser separados linealmente, incluso con la introducción de las variables de holgura, por lo que no resulta fácil definir el hiperplano óptimo. En el Anexo 5 se

muestra un ejemplo donde los datos de entrada no se pueden separar por un hiperplano en $\Re^n$ , pero sí en un espacio de mayor dimensión.

Las SVMs no lineales mapean los datos de entrada a un espacio de dimensión mayor (espacio de características) $\Re^n \rightarrow \Re^h$ mediante la selección y uso del kernel (González *et al.*, 2017).

La función kernel es el factor principal para lograr la separación lineal en este espacio sin necesidad de realizar cálculos en $\Re^h$. Para que una función sea considerada como kernel debe cumplir las condiciones de Mercer que impone que K debe corresponder a una función simétrica positiva en el espacio de entrada. Entre las funciones kernel más utilizadas en SVM se encuentra las plasmadas en las ecuaciones 1.1 y 1.2 (González *et al.*, 2017).

Kernel lineal:

$$K(\vec{X}_i, \vec{X}_j) = \vec{X}_i \times \vec{X}_j \qquad (1.1)$$

Kernel polinomial:

$$K(\vec{X}_i, \vec{X}_j) = (p + \vec{X}_i \times \vec{X}_j)^d \quad p \in R, \ d \in \aleph \qquad (1.2)$$

Una de las ventajas de utilizar SVM es que no requiere una cantidad enorme de datos para funcionar bien, lo que hace que no sea tan problemático tener las limitaciones del conjunto de datos. Es robusto a ruido (variables de holgura) y la solución se define solo por un subconjunto pequeño de puntos (vectores de soporte) (Morales, González and Escalante, 2017).

### 1.3.3 Red Neuronal Perceptrón Multicapa (MLP)

Las redes neuronales han sido entrenadas para realizar funciones complejas en diversos campos. Estas incluyen procesamiento de imágenes y de voz, reconocimiento de patrones, planeamiento, interfaces adaptivas para sistemas hombre/máquina y predicción. Entre sus aplicaciones más relevantes se encuentran en la industria aeroespacial y militar, automotor, en los sistemas bancarios, electrónica y entretenimiento. Además, en campos como la biología, la psicología, recursos humanos y finanzas. Reciben un conjunto de señales de entrada procedentes de otras neuronas o de un sistema exterior.

Las redes neuronales se ajustan en base a una comparación de la salida de la misma con el de objetivo, hasta la salida de la red sea igual al objetivo. El entrenamiento por lotes de una red neuronal se realiza haciendo cambios en los pesos y basados en el sesgo de un conjunto completo

de los vectores de entrada (Juárez, 2018). En el Anexo 7 se ilustra de forma esquemática, la estructura de una neurona.

Existen tres tipos de neuronas artificiales las cuales se agrupan en unidades funcionales llamadas capas (Shalev Shwartz and Ben David, 2014):

**Neuronas de entrada:** reciben la información directamente del exterior y se agrupan en la capa de entrada.

**Neuronas ocultas:** reciben información de otras neuronas artificiales y cuyas señales de entrada y salida permanecen dentro de la red. Se agrupan en las capas ocultas. La cantidad de neuronas por capas, se escogen mediante la *prueba y el error*.

**Neuronas de salida:** reciben la información procesada y la devuelven al exterior. Forman la capa de salida.

Una red de neuronas artificial puede pensarse como un grafo formado por neuronas organizadas por capas y relacionadas por conexiones que determinan la dirección del grafo. En función de esta dirección, se pueden clasificar en:

- Redes de propagación hacia delante (feedforward): No tienen bucles. Algunos ejemplos pueden ser el perceptrón simple y el perceptrón multicapa.
- Redes recurrentes (feedback): Se producen bucles de retroalimentación.

Las redes neuronales poseen muchas ventajas:

- **Aprendizaje:** Tienen la capacidad de aprender a realizar tareas basadas en un entrenamiento o una experiencia inicial.
- **Auto organización:** Crea su propia organización o representación de la información en su interior que recibe mediante una etapa de aprendizaje.
- **Tolerancia a fallos:** Si sufre algún daño puede seguir respondiendo de manera aceptable pues almacena la información de forma redundante.
- **Flexibilidad:** Puede manejar cambios no importantes en la información de entrada, como señales con ruido u otros cambios en la entrada.
- **Tiempo real:** Por su estructura paralela se pueden obtener respuestas en tiempo real.

El perceptrón multicapa es una extensión del perceptrón simple. La topología está definida por un conjunto de capas ocultas, una capa de entrada y una de salida (véase Anexo 8). No existen restricciones sobre la función de activación, aunque en general se suelen utilizar funciones sigmoideas (Juárez, 2018).

Este modelo es el más utilizado en la actualidad. Se aplica en áreas como la codificación de información, traducción de texto en lenguaje hablado y Reconocimiento Óptico de Caracteres (OCR).

El algoritmo de retropropagación del error se basa en la regla del descenso del gradiente (Morera Munt, 2018). Es un método versátil y potente, el funcionamiento de este algoritmo. Dado un patrón de entrada que se aplica a la primera capa de neuronas de la red, se va propagando por las siguientes capas hasta que llega a la capa de salida, donde se compara la salida obtenida con la deseada. A continuación, se calcula el error para cada neurona de salida y éste es transmitido hacia atrás, por todas las capas intermedias, y se van modificando sus pesos sinápticos según dicho error y los valores de las salidas de las unidades precedentes ponderadas por sus pesos hasta llegar a la capa de las unidades de entrada.

### 1.3.4 Árbol de Decisión (Decision Tree)

Los árboles de decisión (Decision Tree) (Shalev Shwartz and Ben David, 2014; Rokach and Maimon, 2015) son un método de aprendizaje supervisado no paramétrico utilizado para la clasificación y la regresión. El objetivo es crear un modelo que predice el valor de una variable objetivo mediante el aprendizaje de reglas de decisión simples deducidas de las características de los datos. Cuanto más profundo es el árbol, más complejas son las reglas de decisión.

Como clasificador consiste en encontrar delimitadores dentro de los datos para ir tomando decisiones. Son modelos estadísticos que funcionan con bloques *si-sino-entonces,* donde si se cumple una cierta condición será más probable que se acabe clasificando una u otra categoría.

Un árbol cuando presenta sobreajuste que posee muchas ramas, es necesario realizar una poda. Este consiste en cortar sucesivamente las ramas y nodos terminales hasta lograr un tamaño adecuado para dicho árbol. Existen dos tipos de poda: la *prepoda* y la *pospoda* (Medina Merino and Ñique Chacón, 2017).

Es rápido clasificando nuevos elementos y la creación del modelo tiene un bajo costo computacional. A pesar de sus ventajas, también tiene desventajas, tales como poca tolerancia al ruido de los datos. Los árboles de decisión son muy utilizados en el campo de la medicina para diagnóstico médico.

El conocimiento obtenido por este clasificador se representa con un árbol que gráficamente posee un conjunto de nodos, hojas y ramas. El nodo principal o raíz es el atributo a partir del cual se inicia el proceso de clasificación. Los nodos internos corresponden a cada una de las preguntas acerca del atributo en particular del problema. Cada posible respuesta a los cuestionamientos se representa mediante un nodo hijo. Las ramas que salen de cada uno de estos nodos se encuentran etiquetados con los posibles valores del atributo. Los nodos finales o nodos hojas corresponden a una decisión, la cual coincide con una de las variables clase del problema a resolver. Una representación general se muestra en el Anexo 10.

Un algoritmo de generación de árboles de decisión consta de dos etapas importantes inducción del árbol y clasificación. Usualmente se construyen utilizando fórmulas que permiten indicar si un atributo debe ser usado o no. Se suele utilizar la *entropía* y el índice de *Gini*. Para la entropía, o medida de incertidumbre, se suele usar su forma binaria, mientras el índice de Gini representa la medida de desigualdad del sistema. Sin embargo, en general se suele tener en cuenta otro parámetro más preciso para seleccionar el mejor atributo la ganancia de información. Esta evalúa la calidad con la que un atributo es capaz de discriminar los ejemplos. A mayor ganancia de información, más importante será dicho atributo. El algoritmo utiliza esta ganancia calculada para ir eligiendo los mejores atributos en cada paso, crear subconjuntos e ir construyendo el árbol descendentemente.

Por tanto, a mayor entropía mayor incertidumbre, y es más difícil para el árbol tomar una decisión. Un atributo puede ayudar a discriminar más ejemplos, tiende a reducir la entropía, y por ese motivo, debe ser seleccionado como un nodo de selección.

El método C4.5 es uno de los más utilizados en la literatura por los árboles de decisión, también es conocido como J48 en la herramienta Weka. Es una extensión del árbol ID3.

Algunas ventajas de los árboles de decisión según (Scikit-learn developers, 2018) son:

- Simple de entender e interpretar. Los árboles se pueden visualizar.

- Requiere poca preparación de datos.
- Capaz de manejar datos tanto numéricos como categóricos y problemas de salida múltiple.
- Es posible validar un modelo utilizando pruebas estadísticas. Eso permite ver la fiabilidad del modelo.
- Funciona bien incluso si sus suposiciones se violan por el verdadero modelo a partir del cual se generaron los datos.

Algunas de las desventajas de los árboles de decisión según (Scikit-learn developers, 2018) son el fenómeno de sobreajuste y la inestabilidad.

### 1.3.5 **Bosque Aleatorio (Random Forest)**

Bosques Aleatorios (Random Forests) (Murphy, 2012; Louppe, 2015) es un algoritmo de aprendizaje supervisado y capaz de realizar tareas de regresión y clasificación. Se considera un meta-clasificador por ser una combinación de árboles predictores. Es una modificación sustancial de bagging que construye una larga colección de árboles no correlacionados y luego los promedia.

La idea esencial del bagging es promediar muchos modelos ruidosos, pero aproximadamente imparciales, y por tanto reducir la variación. Los árboles son los candidatos ideales para el bagging, dado que ellos pueden registrar estructuras de interacción compleja en los datos. Producto de que los árboles son notoriamente ruidosos, ellos se benefician enormemente al promediar. El clasificador es popular y ampliamente utilizado en el sector bancario, en la medicina, en la bolsa de valores y en el comercio electrónico.

Para la predicción un nuevo caso es analizado por un árbol. Luego se le asigna la etiqueta del nodo terminal donde termina. Este proceso es iterado por todos los árboles en el ensamblado, y la etiqueta que obtenga la mayor cantidad de incidencias es reportada como la predicción de los datos. En el Anexo 12 se muestra el funcionamiento general.

Cada árbol depende de los valores de un vector aleatorio de la muestra de manera independiente y con la misma distribución de todos los árboles en el bosque. La generalización de error para los bosques converge a un límite en cuanto el número de árboles en el bosque sea grande. El error de generalización de un bosque de árboles de clasificación depende de la fuerza de los árboles

individuales en el bosque y la correlación entre ellos (Medina Merino and Ñique Chacón, 2017). Un clasificador Bosques Aleatorios se ilustra gráficamente en el Anexo 13.

Dentro de sus ventajas se puede decir que para un set de datos lo suficientemente grande es muy certero y eficiente. Evita el sobreajuste. Además, maneja cientos de variables de entrada sin excluir ninguna. Estima las variables que son importantes en la clasificación. Posee un método eficaz para estimar datos perdidos y mantener la exactitud cuando una gran proporción de los datos está perdida.

Algunas desventajas que presenta son que lleva más tiempo para entrenarse. Es computacionalmente costoso en comparación con un Árbol de Decisión. A diferencia de los árboles de decisión, la clasificación es difícil de interpretar.

### 1.3.6 **Naïve Bayes**

El algoritmo de Naïve Bayes (Murphy, 2012) está basado en el teorema de Bayes y en la premisa de independencia de los atributos dada una clase. El teorema de Bayes se refiere al cálculo de la probabilidad condicional del evento A dado que ha ocurrido el evento B. Su forma general es: Si $A_1, A_2, \dots, A_n$ son eventos exhaustivos y exclusivos tales que $P(A_i) > 0, \forall\, i = 1,2, \dots, n$ . Sea B un evento cualquiera del que se conocen las probabilidades condicionales $P\ (B|A_i)$, la probabilidad $P(A_i|B)$ viene dada por la expresión plasmada en la ecuación 1.3 (Chaparro, Giraldo and Rodón, 2013).

$$P(A_i|B) = \frac{P(A_i,B)}{P(B)} = \frac{P(A_i)P(B|A_i)}{P(B)} = \frac{P(A_i)P(B|A_i)}{\sum_{k=1}^{n} P(B|A_k)P(A_k)} \qquad (1.3)$$

El clasificador Naive Bayes es un modelo particular de las redes bayesianas que se aplican en la tarea de clasificación supervisada. Las ventajas incluyen menor costo computacional, fácil de entender al ser un clasificador basado en probabilidades, y requiere un solo paso de procesamiento si los datos son discretos.

Comparado con las redes bayesianas el clasificador Naive Bayes es mucho más rápido y si los atributos son independientes puede tener la mayor tasa de acierto comparado con otros clasificadores. Un modelo gráfico del clasificador Naive Bayes se muestra en el Anexo 9.

Se aplica a grandes bases de datos y es especialmente útil en diagnósticos médicos y en el campo de la medicina en general. Asimismo, es uno de los métodos de aprendizaje supervisado más utilizados para el análisis de sentimientos (Tan *et al.*, 2009; Kaur and Singla, 2016). Se utiliza,

además, en NLP (Natural Language Processing), en la detección de "Spam", en reconocimiento de idioma o detectar la categoría apropiada de un artículo de texto. También se usa en la detección de intrusiones o anomalías en redes informáticas. Entre sus ventajas se pueden mencionar el bajo tiempo de clasificación y de aprendizaje, bajos requerimientos de memoria, sencillez.

Este modelo de clasificación cabe destacar que presenta dos problemas subyacentes, que las fronteras de decisión de este modelo con variables binarias son *hiperplanos*. Esto supone que el clasificador no es capaz de generar fronteras que se ajusten perfectamente al conjunto de datos y que es posible que no clasifique correctamente.

Algunas de las distribuciones más utilizadas del clasificador Naive Bayes es la Multinomial para datos discretos, Bernoulli para datos discretos y booleanos, la Complementaria para conjuntos de datos desbalanceados, y la Gaussiana que presenta los mejores resultados, es una de las más ampliamente difundidas (ecuación 1.4) (Scikit-learn developers, 2018).

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} exp\left(-\frac{(x_i-\mu_y)^2}{2\sigma_y^2}\right) \qquad (1.4)$$

### *1.4 Protocolos y tipos de API para el intercambio de recursos*

Una Interfaz de Programación de Aplicaciones (Application Programming Interface (API)) es un conjunto de subrutinas, funciones o procedimientos o métodos, estructuras de datos, clases y variables que ofrece cierta biblioteca para ser utilizados por otro software como una capa de abstracción. Son usadas generalmente para las bibliotecas de programación (Pandorafms Community, 2019).

Existen API de código fuente. Ofrecen bibliotecas de objetos, clases, funciones, variables etc. Este tipo de API es de los más antiguos. La mayoría de las veces se encuentran dentro de los Kits de Desarrollo de Software (SDK). Pueden verse como una abstracción del funcionamiento interno del entorno sobre el que se va trabajar. Su ventaja principal es que son utilizadas por desarrolladores al emplear API de bibliotecas necesarias en su código.

Otro tipo de API se basa en estándares de servicios web de hipertexto. Las API de servicios se usan comúnmente como parte de un enfoque de arquitectura orientada a servicios (Service Oriented Architecture (SOA). Esta es una arquitectura de aplicación en la cual todas las funciones están

definidas como servicios independientes con interfaces invocables que pueden ser llamados en secuencias bien definidas (Newman, 2015).

Un servicio web es un tipo de API que proporciona acceso a un servicio mediante una dirección URL en una red y transfiere información entre aplicaciones. Es importante que la información proporcionada por una API de servicio se presente en un formato comprensible para otras aplicaciones como JSON o XML, si no se perdería la funcionalidad de la API.

Algunas ventajas de las API de servicios son:

- Aportan interoperabilidad entre aplicaciones de software independientemente de los lenguajes de programación o de las plataformas sobre las que se instalen.
- Fomentan los estándares y protocolos basados en texto, que hacen más fácil acceder a su contenido y entender su funcionamiento.
- Permiten que servicios y softwares de diferentes procedencias puedan ser combinados fácilmente para proveer servicios integrados.

Para exponer y consumir los recursos desde otras aplicaciones es recomendado la utilización de servicios mediante los protocolos XML-RPC, SOAP y el estilo arquitectónico REST. Estos son útiles para que los desarrolladores puedan hacer uso de funciones ya existentes accesibles mediante una API, sin tener en cuenta el lenguaje de programación o sistema operativo. La API de servicio es lo más indicado y ampliamente difundido en la última década. Además, es la más útil por su sencilla implementación, soportada por la mayoría de los lenguajes de programación existentes y sistemas operativos.

Una API de servicio para consumir los métodos de la biblioteca Feel UCLV permitirá a los desarrolladores emplear el reconocimiento de emociones en la programación de aplicaciones independiente de los detalles de implementación, del lenguaje y del sistema operativo.

### 1.4.1 **XML-RPC**

XML-RPC (UserLand Software, 2019) es un protocolo de llamada a procedimiento remoto que usa el formato XML para codificar los datos y HTTP para la transmisión de mensajes (véase Figura 1.4).

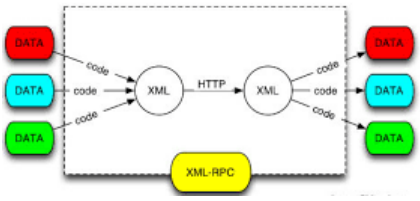



**Figura 1.4 Estructura de XML-RPC (UserLand Software, 2019)**

Dentro de sus características más significativas se puede mencionar que es de código abierto. Está diseñado para ser lo más simple posible permitiendo la transmisión, el procesamiento y la devolución de estructuras de datos complejas. Es una especificación y un conjunto de implementaciones que permiten que el software se ejecute en diferentes sistemas operativos y lenguajes de programación para realizar llamadas de procedimientos.

Las principales ventajas de XML-RPC es que es muy rápido en su ejecución, define algunos tipos de datos y comandos útiles, además de una descripción completa de corta extensión para el desarrollador. Tiene una documentación extensa y requiere considerable soporte de software para su uso. Está ampliamente soportado para varios lenguajes de programación.

El protocolo XML-RPC presenta varias desventajas. Es muy antiguo, por lo que se considera muy sencillo para los servicios web que se realizan en la actualidad. No contiene seguridad ni soporta formato de texto sencillo JSON para el intercambio de datos. Para ello se han realizado algunas modificaciones al protocolo como es JSON-RPC. Este es utilizado solamente para servicios web sencillos y de rápida implementación. Está más destinado a la transferencia de valores. Tiene algunos problemas con los caracteres que no son ASCII.

### 1.4.2 SOAP

Simple Object Access Protocol (SOAP) es un protocolo estándar de comunicación. Define cómo dos objetos en diferentes procesos pueden comunicarse mediante intercambio de datos XML. Es utilizado por los servicios web. Facilita la comunicación entre aplicaciones desarrolladas en distintos lenguajes de programación e incluso, en diferentes sistemas operativos. Es derivado del protocolo XML-RPC (GlosarioIT, 2019).

SOAP se apoya en tecnologías probadas y confiables como XML y HTTP (véase Figura 1.5). La vinculación que se hace entre SOAP y HTTP se deriva de la masiva utilización de ambas tecnologías por separado y en conjunto.

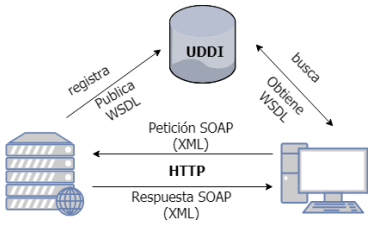



**Figura 1.5 Arquitectura de SOAP**

Se ubica en la capa de empaquetamiento de mensajes entre aplicaciones. Un mensaje o paquete SOAP es un documento XML estructurado. Según detallan los autores Ortiz Vivar and Segarra Flores (2015), los mensajes en SOAP son documentos estandarizados en XML que constan de un Header y un Body, englobados por el elemento raíz envelope (véase Figura 1.6):

**Header:** Contiene información adicional sobre el mensaje, es opcional.

**Body:** Contiene el mecanismo para el intercambio de información con el destinatario. Puede contener un documento XML o una llamada RPC con datos estructurados e informe de fallos.

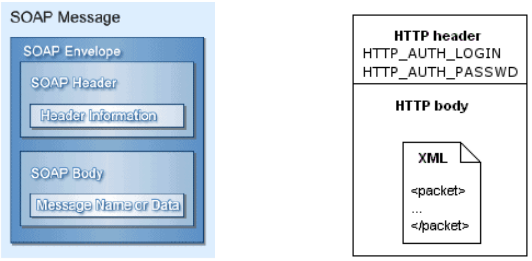



**Figura 1.6 elementos y estructura de un mensaje SOAP (Vergara Arroyo, 2019)**

SOAP posee algunas ventajas importantes. Presenta gran interoperabilidad, pues permite invocar procedimientos remotos, debido al uso de XML. Es fácilmente escalable al utilizar una comunicación vía HTTP, casi siempre es permitido por los cortafuegos. Puede ser implementado en la mayoría de los lenguajes y ejecutado en cualquier plataforma. Es posible utilizarlo mediante usuario anónimo y mediante autentificación. Es posible transmitirlo mediante cualquier protocolo capaz de transmitir texto, típicamente HTTP o SMTP.

Asimismo, el protocolo presenta varias desventajas. Debido al uso de XML para el paso de mensajes, SOAP es considerablemente más lento que otros middlewares, ya que los datos binarios se codifican como texto. Depende del Web Services Description Language (WSDL). Es un protocolo complejo para los desarrolladores y complicado de implementar, posee una documentación extensa y compleja para su entendimiento. Al contrario de Java, PHP o Python

ciertos lenguajes no ofrecen un apoyo adecuado para su uso, ya sea a nivel de integración o de soporte IDE. En algunos lenguajes de programación como Python no está bien soportado.

En la última década, las empresas han empezado a exponer sus APIs para permitir a terceros construir funcionalidades nuevas. Las tecnologías tradicionales como SOAP han ido evolucionando para reducir la interdependencia entre los elementos que la usan, terminando en la gran adopción de la arquitectura REST para diseñar servicios web.

### 1.4.3 **API REST**

El término REST fue introducido por Roy Fielding en su tesis doctoral (Fielding, 2000). REST (Representational State Transfer) es un estilo de arquitectura de software que define la interacción entre componentes dentro de un sistema hipermedia distribuido (véase Figura 1.7). REST ignora los detalles de implementación del componente y la sintaxis de protocolo con el fin de centrarse en las funciones de los componentes, las limitaciones a su interacción con otros componentes y su interpretación de elementos de datos significativos. Estos servicios han venido ganando popularidad actualmente, superando incluso a servicios como SOAP.

Bravo del Río (2016) expone que este estilo se compone de cinco elementos, los cuales tienen por objetivo aumentar la interoperabilidad, el desempeño, la escalabilidad y la fiabilidad de los sistemas. Lo elementos son: funciona sobre una arquitectura cliente-servidor, los servidores no almacenan el estado de los clientes, la comunicación se queda en la caché (puede ser repetida cuando no hay cambios), los clientes y los servidores se comunican utilizando una interfaz uniforme, los servidores se pueden montar en capas, de manera transparente para los clientes.

La API REST es un tipo de API en particular que se basa en REST. No es un estándar, pero se basa en algunos estándares como HTTP y Uniform Resource Locator (URL) (Stowe, 2015; Patni, 2017). Junto a esta arquitectura, se asocian tecnologías como JSON (JavaScript Object Notation) que es un formato ligero de intercambio de datos. De esta forma el desarrollo y uso de APIs se ha acelerado de forma considerable con REST y el formato JSON, soportado por todos los lenguajes de programación existentes.

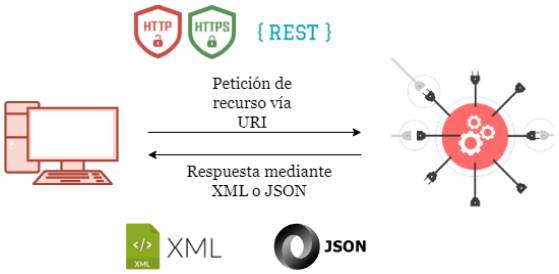



**Figura 1.7 Arquitectura REST**

Dentro de las características de API REST, según BBVAOpen4U (2016), se destacan algunas como:

- **Protocolo cliente/servidor sin estado**: cada petición HTTP contiene la información necesaria para ejecutarla. Se configura el protocolo cliente-caché-servidor sin estado.
- **Operaciones importantes**: POST (crear un recurso nuevo), GET (leer y consultar información de un recurso), PUT (modificar un recurso existente), DELETE (eliminar un recurso determinado) y PATCH (modificar solamente un atributo de un recurso).
- **URI el identificador único de cada recurso**: los objetos en REST siempre se manipulan a partir de la URI. Esta facilita el acceso a la información.
- **Interfaz uniforme**: para la transferencia de datos se aplican acciones concretas sobre los recursos, siempre que estén identificados con una URI.
- **Sistema de capas**: arquitectura jerárquica entre los componentes. Cada una de estas capas lleva a cabo una funcionalidad dentro del sistema REST.
- **Uso de hipermedios**: la capacidad de una interfaz de desarrollo proporciona al cliente y al usuario los enlaces adecuados para ejecutar acciones concretas sobre los datos.

Una de las razones por las que REST ha sido tan ampliamente aceptado es porque permite a los desarrolladores la habilidad de construir funcionalidades modulares con interfaces muy livianas que no requieren una compleja integración.

Algunas de las ventajas por las que se distingue la API REST son:

- **Escalabilidad**: Gracias a la separación entre el cliente y el servidor, el producto se puede escalar con un equipo de desarrollo sin que ello represente muchas dificultades.

- **Flexibilidad y portabilidad**: Es posible realizar una migración de un servidor a otro o practicar cambios en la base de datos. De esta forma el fronted y el backend se pueden alojar en servidores diferentes, lo que supone una enorme ventaja de manejo.
- **Independencia**: El cliente y el servidor solo se comunican con un lenguaje de intercambio como JSON. El protocolo facilita que los desarrollos de las partes de un proyecto se puedan dar de manera independiente. Esto brinda la oportunidad de probar varios entornos dentro del desarrollo.
- **Independencia de tecnologías / lenguajes**: Se puede desarrollar una API REST en cualquier tipo de tecnología o lenguaje de programación.
- **No requiere memoria:** REST requiere menos recursos del servidor, al no mantener el estado se pueden atender más peticiones.

Los autores Lorenzo Gil, Braña Ferreiro and Nieto Caramés (2015) visionan a REST como el modo más flexible en la construcción de servicios de interoperabilidad o integración. La interface REST es una manera efectiva de acceder, reusar y gestionar contenidos almacenados por repositorios desde otros sistemas.

Mediante el estudio profundizado sobre API REST se está de acuerdo con lo planteado por los autores Lorenzo Gil, Braña Ferreiro and Nieto Caramés (2015). Donde se clasifica a REST como el modo más flexible para lograr integraciones. Lograr consumir los recursos de la biblioteca Feel UCLV utilizando una API de tipo REST garantiza seguridad, escalabilidad, simplicidad y portabilidad. Además, permite a los programadores la habilidad de implementar nuevas funcionalidades modulares mediante interfaces muy ligeras sin la necesidad de una integración compleja, con independencia de tecnologías y lenguajes de programación.

La tendencia en los últimos años para brindar y consumir servicios ha sido mediante la utilización de la API de tipo REST.

### 1.4.4 Comparaciones

Después del estudio detallado de los protocolos y tipos de API que existen para el intercambio de recursos se hizo necesario establecer una comparación entre ellos para determinar cuál es el mejor para consumir los métodos y recursos que almacena la biblioteca Feel UCLV desde cualquier tipo de aplicación que los necesite.

A nivel mundial el uso de los servicios web implementados con los protocolos REST, SOAP, XML-RPC y JavaScript como formato implementado recae en el de tipo REST, según Xamarin (2017) publicado en 27 mayo 2017 (véase Figura 1.8).

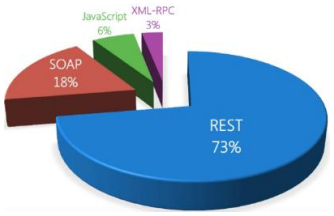



**Figura 1.8 Gráfico del uso de los protocolos REST, SOAP, JavaScript y XML-RPC (Xamarin, 2017)**

El protocolo XML-RPC es muy sencillo y antiguo, no cumple con las exigencias del desarrollo moderno de API. SOAP y API REST son los más indicados para la implementación de la API. REST supera las desventajas que presenta SOAP, como la necesidad de que los clientes conozcan la semántica de las operaciones como requisito previo a su uso, o la necesidad de puertos para distintos tipos de notificaciones. Mientras SOAP se centra en el diseño de aplicaciones distribuidas, REST lo hace en la escalabilidad y en el rendimiento a gran escala para sistemas distribuidos hipermedia. A continuación, se presenta una comparación entre ambos protocolos (véase Tabla 1.1).

Una de las ventajas más relevantes de REST sobre SOAP en cuanto a diseño, es que REST es un estilo arquitectónico y SOAP un protocolo de servicio. Otras de las grandes ventajas respecto a formatos de mensajes, es que permite varios formatos de datos y SOAP solo permite el formato XML. En cuanto a la usabilidad, REST es fácil de usar, solo depende de parámetros; al contrario de SOAP que requiere archivos complejos como WSDL. REST es asequible de utilizar e implementar tanto por desarrolladores de APIs o como clientes. Sin embargo, SOAP es un protocolo sumamente complicado de implementar. El estilo arquitectónico REST es soportado por todos los lenguajes de programación tanto para servidor como cliente y muy ampliamente difundido en frameworks y bibliotecas para su implementación. Por el contrario, SOAP se dificulta en algunos lenguajes y su implementación es más complicada por falta de bibliotecas para desarrollarlo. Se puede concluir que la API REST es la mayor usada a nivel mundial, debido principalmente a la capacidad de funcionamiento en ambientes heterogéneos con bajo acoplamiento, facilidad de uso y más ligero comparado con SOAP.

**Tabla 1.1 Comparación entre los protocolos SOAP y REST**

| Aspectos | SOAP | REST |
|---|---|---|
| **Sentido** | Simple Object Access Protocol | Transferencia de estado representacional |
| **Diseño** | Protocolo estandarizado con reglas predefinidas a seguir. | Estilo arquitectónico con pautas libres y recomendaciones. |
| **Enfoque** | Basado en funciones (datos disponibles servicios). | Controlado por datos (datos disponibles como recursos). |
| **Estado** | Sin estado de forma predeterminada, pero es posible hacer que una API SOAP sea completa. | Sin estado (no hay sesiones del lado del servidor). |
| **Almacenamiento en caché** | Las llamadas a la API no se pueden almacenar en caché. | Las llamadas a la API se pueden almacenar en caché. |
| **Seguridad** | WS-Security con soporte SSL. Cumplimiento de ACID incorporado. Comunicación origen a destino y segura. | Soporta HTTPS y SSL. Comunicación punto a punto y segura. |
| **Rendimiento** | Requiere más ancho de banda y potencia de cálculo. | Requiere menos recursos. |
| **Formato de mensaje** | Solo XML. | Texto sin formato, HTML, XML, JSON, YAML y otros. |
| **Protocolos de transferencia** | HTTP, SMTP, POST, MQ, UDP y otros. Síncrono y Asíncrono. | Solo HTTP (GET, DEL, POST, PUT). Síncrono. |
| **Ventajas** | Alta seguridad, estandarizada, extensibilidad. | Escalabilidad, mejor rendimiento, facilidad de navegación, flexibilidad. |
| **Desventajas** | Desempeño más pobre, más complejidad, menos flexibilidad. Si se desea realizar una modificación en el servidor esto impacta en los clientes ya que ellos tendrán que realizar varias modificaciones a su código. Los clientes necesitan puertos dedicados. | Menos seguridad puesto que no está fuertemente tipado, es un problema y puede llegar a ser una tarea muy difícil de implementar, no apto para entornos distribuidos. No hay un estándar en mensajería y sus respuestas por lo que no se definen tipos de datos y es difícil lidiar con la comunicación de fallos. |

Se decidió que la API REST es la indicada para implementar la API de la biblioteca Feel UCLV. Esta puede permitir la comunicación entre la biblioteca y cualquier tipo de aplicación. Se adapta en todo momento al tipo de sintaxis, sistema operativo o plataformas de trabajo. Se puede desarrollar una API REST en cualquier tipo de tecnología o lenguaje de programación, siempre y cuando siga teniendo las mismas operaciones. Esto evidencia la mayor fortaleza de la API REST.

Además, es el tipo de API más moderno y más difundido a nivel mundial por desarrolladores y respaldada por grandes compañías. Otra de las razones es por ser muy familiar para los desarrolladores y la mejor y más viable entre los protocolos de su tipo. Asimismo, según Chakray (2017), suele ser sencillo de construir y de adaptar con escaso consumo de recursos.

## *1.5 Frameworks de Python para implementar API REST*

La biblioteca de reconocimiento de emociones faciales Feel UCLV se encuentra implementada en el lenguaje de programación Python. En la literatura consultada se evidencia una gran variedad de frameworks para realizar implementaciones de API en este lenguaje de programación. A continuación, se explican los más importantes y más utilizados en la bibliografía sobre el tema.

### 1.5.1 **Django**

Django es un framework de desarrollo web de código abierto y multiplataforma. Respeta el patrón de diseño conocido como Modelo Vista Controlador. Es un framework de alto nivel que fomenta el desarrollo rápido y el diseño limpio y pragmático. Es muy seguro y ayuda a los desarrolladores a evitar muchos errores comunes de seguridad. Posee una documentación muy extensa y una comunidad numerosa (Rubio, 2017; Django Software Foundation, 2019).

El propósito de Django es facilitar la creación de sitios web complejos. Este pone énfasis en el reúso, la conectividad y extensibilidad de componentes, el desarrollo rápido y el principio "No te repitas" (DRY, *Don't Repeat Yourself*). Python es usado en todas las partes del framework, incluso en configuraciones, archivos y en los modelos de datos.

Posee, además funcionalidades como el mapeador objeto-relacional. Plugins que pueden instalarse en cualquier página gestionada con Django. Una API de base de datos robusta que soporta PostgreSQL, MySQL, Oracle, SQLite. Un sistema incorporado de vistas genéricas que ahorra tener que escribir la lógica de ciertas tareas comunes. Un sistema extensible de plantillas basado en etiquetas, con herencia de plantillas. Documentación incorporada accesible. Un despachador de URLs basado en expresiones regulares. Soporte de internacionalización, incluyendo traducciones incorporadas de la interfaz de administración.

Django se extiende mediante Django REST Framework. Este es un kit de herramientas potente y flexible para crear API REST. Por lo que convierte a Django en un framework muy completo.

Algunas razones por las que Django REST Framework es muy potente y esencial para la realización de API REST, según Christie (2019) son listadas:

- La API de navegación web es una ganancia de usabilidad para sus desarrolladores.
- Contiene políticas de autenticación que incluyen paquetes para OAuth1a y OAuth2.
- Posee serialización compatible con orígenes de datos ORM y no ORM.
- Es personalizable, solo permite usar las vistas regulares basadas en funciones.
- Contiene amplia documentación y gran apoyo de la comunidad.
- Es ampliamente utilizado y en el que confían empresas de renombre internacional.

### 1.5.2 **Flask**

Flask es un microframework implementado en Python, de código abierto, desarrollado por Armin Ronacher. Permite crear aplicaciones web rápidamente y con un mínimo número de líneas de código (Copperwaite and Leifer, 2015; Ronacher, 2019a). Está basado en la especificación WSGI de Werkzeug (2019) y el motor de plantillas (templates) Jinja2 (Ronacher, 2019b), tiene una licencia BSD y es multiplataforma.

Está inspirado en el sistema de plantillas de Django, pero lo amplía con un lenguaje expresivo que brinda a los autores de plantillas un conjunto de herramientas más poderoso. Además, agrega ejecución de espacio aislado y escape automático opcional para aplicaciones donde la seguridad es importante.

Flask incluye un servidor web de desarrollo para que se pueda probar e ir viendo los resultados que se van obteniendo sin la necesidad de tener que instalar ningún otro servidor web. También tiene un depurador y soporte integrado para pruebas unitarias. Contiene un excelente soporte para Unicode y es totalmente compatible con WSGI 1.0 y con Python en su versión 3.

El framework posee una comunidad numerosa de desarrolladores y una buena documentación. Unas de sus características más relevantes es su esquema de rutas. El microframework soporta el uso de cookies seguras. Flask no tiene ORM, pero brinda la facilidad de poder integrarlo con casi cualquier herramienta. Además, presenta gran cantidad de plugins desarrollados por la comunidad de este. La no existencia de wrappers o configuraciones complejas, lo convierte en un candidato

ideal para aplicaciones ágiles y rápidas o que no necesiten manejar ninguna dependencia (Domingo Muñoz, 2017).

Este framework resulta ideal para construir API REST o aplicaciones de contenido estático. Tiene un plugin Flask-RESTful que agrega soporte para la creación de este tipo de API. Flask-RESTful fomenta las mejores prácticas con una configuración mínima.

### 1.5.3 **FastAPI**

FastAPI (FastAPI, 2019) es un framework web moderno y rápido de alto rendimiento bajo la licencia MIT. Sirve para crear API en el lenguaje de programación Python, basado en sugerencias estándares de tipo Python. Como requisito necesita una versión superior a la 3.6 del lenguaje de programación. FastAPI se encuentra desarrollado sobre Starlette para la parte web y Pydantic para los datos.

Es uno de los frameworks de Python más rápidos disponibles actualmente, pues presenta un rendimiento muy alto. Se encuentra a la par con Nodejs y Go (gracias a Starlette y Pydantic). Su código es rápido y sencillo de implementar, la velocidad para desarrollar funciones es sumamente mayor a la de otros frameworks Python. Reduce aproximadamente el 40% de los errores inducidos por desarrolladores. Minimiza la duplicación de código. Permite convertir desde el código fuente diferentes tipos de datos de Python como str, int, float, bool, list, datetime, UUID, diccionarios, modelos de base de datos entre otros, a formato JSON de manera automática, mediante el framework.

Está diseñado para ser fácil de usar y aprender. Se emplea menos tiempo estudiando la documentación. Es un framework intuitivo. Posee gran soporte de editor. Es robusto, permite obtener código listo para la producción con documentación interactiva automática.

Basado y totalmente compatible con los estándares abiertos para API: OpenAPI (anteriormente conocido como Swagger) y esquema JSON. El esquema JSON es una herramienta poderosa para validar la estructura de datos JSON (Douglas Crockford, 2019).

FastAPI utiliza la iniciativa OpenAPI (OpenAPI Initiative, 2019). Este es un estándar para crear, administrar y consumir una API REST. OpenAPI Specification (OAS) es un estándar para crear un manual de uso para las API REST. El diseño de las API con OpenAPI se definen mediante un

JSON o YAML, dos formatos de ficheros estándares. La versión actual de la especificación es la 3 y orientada a YAML (YAML, 2019).

El framework permite documentación automática del modelo de datos con el esquema JSON. Permite utilizar la generación automática de código de cliente en muchos lenguajes. Además, permite realizar documentos automáticos para los desarrolladores mediante OpenAPI con la herramienta Swagger UI, con exploración interactiva probando su API directamente desde el navegador web y mediante ReDoc.

OpenAPI permite descubrir y comprender las capacidades de un servicio o una API, sin necesidad de acceder al código fuente, sin documentación adicional o inspección de las peticiones. Cuando se diseña correctamente una API a través de esta especificación, un desarrollador puede comprender e interactuar con el servicio mucho más fácil, ya que se eliminan las suposiciones al llamar a un servicio.

La empresa TechEmpower (TechEmpower, 2019a) está dedicada al desarrollo de software, mayormente, orientados a la web. Realiza una comparación de rendimiento entre todos frameworks Python de aplicaciones web que ejecutan tareas fundamentales como la serialización JSON, el acceso a la base de datos y la composición de la plantilla del lado del servidor, entre otras. Estos están organizados por lenguaje de programación. Los resultados se capturan en instancias de nube y en hardware físico. Esto se puede observar en la lista de comparaciones referentes a los frameworks de Python más utilizados de acuerdo a su velocidad y rendimiento (véase Anexo 2) (TechEmpower, 2019b).

Se considera FastAPI como uno de los frameworks Python más rápidos, disponibles. En la lista de comparación entre frameworks Python se encuentra en el tercer puesto, solo por debajo de Starlette y Uvicorn (utilizados internamente por FastAPI). Por tanto, es un microframework muy útil para la implementación de API REST.

FastAPI brinda validación para la mayoría de los tipos de datos de Python: Objetos JSON (diccionarios), Matriz JSON (lista) que define los tipos de elementos, Campos de cadena (str), definiendo longitudes mínimas y máximas, Números (int, float) con valores mínimos y máximos, etc. También permite validar otros tipos de datos como: URL, Email, UUID. Toda la validación es manejada por el sólido y robusto Pydantic. FastAPI posee seguridad y autenticación integradas, sin ningún compromiso con bases de datos o modelos de datos. Todos los esquemas de seguridad

definidos están incluidos en OpenAPI: HTTP básico, OAuth2 (también con tokens JWT). Además, todas las funciones de seguridad de Starlette.

FastAPI no existiría si no fuera por trabajos anteriores, sus desarrolladores se basaron en muchas herramientas creadas anteriormente que inspiraron su creación. De estas fueron extraídas la lógica y lo necesario. Tuvieron en cuenta el framework Django, el más popular en Python y su biblioteca Django REST Framework. Fue inspirado, también, en Flask por ser un microframework que hace más rápido el trabajo en algunas cuestiones y es la competencia más relevante de Django entre los frameworks web. Utilizaron Requests, una biblioteca para interactuar con las API en Python como cliente, se basaron en su lógica de las rutas para incluirlo en FastAPI. Se inspiraron en la documentación automática del esquema OpenAPI. Starlette fue utilizado por ser el microframewok más rápido disponible en Python y Uvicorn como servidor web porque se basan en ASGI.

Se decidió utilizar FastAPI como framework Python para implementar la API REST. Dentro de sus características, por ser muy potente, moderno y muy intuitivo. Uno de los frameworks de Python más rápidos disponibles actualmente y creado únicamente con el propósito de implementar API REST. Su código es rápido y sencillo de programar. Permite implementar la API REST con calidad. La velocidad para desarrollar funciones es sumamente mayor a la de otros framework Python. Reduce aproximadamente el 40% de los errores inducidos por desarrolladores. Minimiza la duplicación de código. Brinda validación para la mayoría de los tipos de datos de Python. Está basado y es totalmente compatible con los estándares abiertos para API como OpenAPI, que es respaldado por grandes empresas y utilizado en sus APIs y soporta el esquema JSON. Además, permite la generación de un portal de documentación que describe la API y es capaz de generar SDKs. FastAPI también, posee un servidor ASGI muy moderno y rápido, es de fácil uso y manejo, sin la necesidad de instalar uno externo. Este framework tiene seguridad y autenticación integradas, sin ningún compromiso con bases de datos o modelos de datos. Además, permite realizar documentos automáticos para los desarrolladores mediante el estándar OpenAPI.

Se seleccionó FastAPI ante Django, pues este framework web se desarrolla con un patrón Modelo-Vista-Controlador y el trabajo con base de datos es bien riguroso. Razones que no se correspondían con la API a implementar para la biblioteca Feel UCLV. Además, que Django necesita una biblioteca de terceros para lograr la implementación de una API REST. Con respecto al microframework Flask, FastAPI presenta gran similitud por ser más ligeros y minimalistas. Uno

de sus incovenientes es que, también necesita una biblioteca de terceros para que los desarrolladores implementen una API REST. Se descartaron Django y Flask, mayormente, porque son framework dedicados a la programación de aplicaciones web, con la particularidad de brindar herramientas para el desarrollo de API de servicios mediante bibliotecas de terceros.

Por tanto, FastAPI es el ideal dentro de los frameworks consultados. Además de lo mencionado anteriormente, es un framework recomendado por la iniciativa OpenAPI y una herramienta útil y novedosa para la creación de API REST en el lenguaje de programación Python. A pesar de ser un microframework muy sencillo, fácil de código, es a la vez muy potente y completo. Pues, está basado en las mejores características de los framework web del lenguaje de programación Python y otras bibliotecas. FastAPI puede marcar el desarrollo venidero de las implementaciones de API REST con calidad en Python, haciéndole frente a dos de los framework web más populares en este lenguaje de programación y establecidos desde hace tiempo.

## *1.6 Herramientas y lenguaje de programación útiles*

En este epígrafe se presenta una revisión de la literatura acerca de las herramientas necesarias y útiles para la confección y mejoramiento de la biblioteca Feel UCLV. Así como el lenguaje que se utilizó en la programación de esta y que será usado en el perfeccionamiento de la misma. Además, este lenguaje será utilizado para implementar la REST API que permitirá consumir los métodos de dicha biblioteca.

### 1.6.1 **Python**

Python (Browning and Alchin, 2019; Python Software Foundation, 2019) es un lenguaje de programación creado por Guido van Rossum a principios de los años 90. El nombre está inspirado en el grupo de cómicos ingleses "Monty Python". Es un lenguaje de código abierto, con una sintaxis muy limpia y sencilla, que favorece un código legible. Se trata de un lenguaje interpretado o de script, con tipado dinámico, fuertemente tipado, multiplataforma y orientado a objetos. Además, permite la programación imperativa, funcional y orientada a aspectos.

El intérprete de Python está disponible en una multitud de plataformas como UNIX, Solaris, Linux, DOS, Windows, OS/2, Mac OS, entre otros. Es ampliamente utilizado en la industria de los videojuegos, en el desarrollo web y sistemas distribuidos, para el desarrollo de tareas científicas y aplicaciones clientes.

En los últimos años Python ha sido catalogado como uno de los mejores lenguajes para el procesamiento de imágenes y el trabajo con algoritmos de aprendizaje automático. También en el campo de la Inteligencia Artificial en general, existen una gran variedad de bibliotecas que poseen interfaces para este lenguaje y que aprovechan la potencia de Python en la creación de aplicaciones referentes a estos temas.

### 1.6.2 **OpenCV**

OpenCV (Open Source Computer Vision Library) (Kaehler and Bradski, 2017; team OpenCV, 2019) es una biblioteca de código abierto. Utilizada para el procesamiento de imágenes y el aprendizaje automático. Tiene interfaces C ++, Python, Java y MATLAB y es compatible con los sistemas operativos Windows, Linux, Mac OS, iOS y Android. OpenCV fue diseñado para la eficiencia computacional y con un fuerte enfoque en las aplicaciones en tiempo real. La biblioteca está escrita en C / C ++ optimizado y hace uso del procesamiento multi-núcleo.

La biblioteca cuenta con más de 2500 algoritmos optimizados y un gran número de funciones que incluyen un conjunto completo de algoritmos de aprendizaje automático y de visión artificial tanto clásicos como de última generación. Mediante dichos algoritmos se pueden detectar y reconocer rostros, identificar objetos, clasificar acciones humanas en videos, Además, se puede rastrear movimientos de cámara, objetos en movimiento, extraer modelos 3D de objetos, producir nubes de puntos 3D desde cámaras estéreo, unir imágenes para producir una alta resolución. También permite encontrar imágenes similares de una base de datos de imágenes, eliminar los ojos rojos de las imágenes tomadas con flash, seguir los movimientos de los ojos, reconocer escenarios y establecer marcadores para superponerlos con realidad aumentada, etc.

### 1.6.3 **Dlib**

Dlib (Dlib, 2019) es un moderno kit de herramientas de C ++ que contiene algoritmos de aprendizaje automático. Así como herramientas para crear software complejo y resolver problemas del mundo real. Se usa tanto en la industria como en el mundo académico en una amplia gama de dominios que incluyen robótica, dispositivos integrados, teléfonos móviles y grandes entornos informáticos de alto rendimiento. Dlib permite mediante su licencia de código abierto que pueda utilizarse en cualquier aplicación sin costo alguno.

En las versiones más recientes se le ha adicionado a Dlib una interfaz para Python que cuenta con varios algoritmos, no tan completa como la interfaz para crear aplicaciones en C++. La biblioteca es compatible con sistemas Windows, Linux y Mac OS X. Además, que funciona en cualquier sistema POSIX y se ha utilizado en Solaris, HPUX y BSD. Se ha incluido en sus algoritmos de aprendizaje automático las técnicas de aprendizaje profundo (deep learning) por ser un amplio uso en trabajos recientes.

### 1.6.4 **Scikit-learn**

Sicikit-learn (Scikit-learn developers, 2018; Scikit-learn, 2019) es una biblioteca de aprendizaje automático para Python y escrita en ese propio lenguaje de programación. Es de código abierto porque se encuentra bajo la licencia BSD. Sciki-learn tiene como bases las bibliotecas de Python NumPy, SciPy y matplotlib. Posee una serie de herramientas simples y eficientes para la minería de datos y el análisis de datos.

Muchas empresas hacen uso de la biblioteca con diferentes fines como para: hacer recomendaciones de música, la construcción de clasificadores de aprendizaje automático, recomendar hoteles y destinos a los clientes, detectar reservas fraudulentas o programar a los agentes de servicio al cliente, la comprensión del comportamiento del usuario, la mejora de la calidad de los datos y la detección de fraudes.

### 1.6.5 **NumPy**

NumPy (Johansson, 2019; NumPy developers, 2019) es el paquete fundamental para el trabajo de computación científica con Python. Permite el trabajo con arreglos, principalmente el objeto ndarray multidimensional. Es de código abierto y multiplataforma sobre los sistemas operativos Windows, Linux y Mac OS X. NumPy. Posee una perfecta integración con Python interactuando con los tipos de datos y estructuras propios del lenguaje. En NumPy se puede realizar una serie de operaciones como:

- Operaciones matemáticas y lógicas en arreglos.
- Transformadas de Fourier y rutinas para la manipulación de la forma de los arreglos.
- Operaciones de álgebra lineal.
- Generación aleatoria de números.

### 1.6.6 Bases de Datos de emociones

Las bases de datos de emociones juegan un papel importante en la detección automática de emociones faciales. Son esenciales en la etapa de clasificación. De esta manera la precisión del clasificador para reconocer las emociones depende del conjunto de datos principalmente. Se recomienda que la base de datos sea amplia, con personas de diferentes razas, géneros, edades y nacionalidades, con imágenes de poca calidad. Además, los conjuntos de datos definen las emociones que se van a detectar y sistema reconoce esas mismas emociones. La gran mayoría de las bases de datos de emociones que existen para la Computación Afectiva utilizan las emociones universales.

***Cohn-Kanade***

Cohn-Kanade (CK) (Kanade, Cohn and Tian, 2000; Lucey *et al.*, 2010) fue lanzada en el año 2000 para promover la investigación de la Computación Afectiva y para el reconocimiento de emociones faciales en el rostro humano compuesta por 210 participantes. En 2010 se lanzó una versión mejorada llamada Extended Cohn-Kanade (CK+), logrando una base de datos más robusta. Es uno de los conjuntos de datos más ampliamente utilizados en los sistemas de detección automática de expresiones faciales y es gratuita. Es muy utilizada en el entrenamiento de clasificadores y reconocimiento de emociones. Está conformada por 123 sujetos que se encuentran de 18 a 50 años de edad. De ellos 69% mujeres, 31% hombres, 81% euro-americanos, el 13% afro-americanos y el 6% de otros grupos. Se realizó a los participantes una serie de muestras faciales donde evidencia la transformación de la expresión facial de un individuo a partir de la neutral hasta llegar a cada una de las seis emociones universales: ira, alegría, tristeza, sorpresa, asco, miedo y también se incluye desprecio.

***Karolinska Directed Emotional Faces (KDEF)***

KDEF fue creada en 1998 por Daniel Lundqvist, Anders Flykt y el profesor Arne Öhman en el Instituto de Karolinska, Departamento de Neurociencia Clínica, Sección de Psicología, en Estocolmo, Suecia (Lundqvist, Flykt and Ohman, 1998; Department of Clinical Neuroscience Psychology Karolinska Institutet, 2019). El material cuenta con un conjunto de 490 imágenes de expresiones faciales humanas. Contiene 70 individuos, de ellos 35 mujeres y 35 hombres, los cuales posan en siete expresiones emocionales diferentes, las seis universales: ira, alegría, tristeza,

sorpresa, asco, miedo, y la neutral durante dos sesiones de fotos. La edad de los participantes es entre 20 y 30, todos de nacionalidad sueca, de raza blanca. Fue desarrollada originalmente para ser utilizado con fines de investigación psicológica y médica, para experimentos de percepción, atención, emoción y memoria. Esta es compartida libremente dentro de la comunidad de investigación, es libre su descarga. KDEF se ha utilizado en más de 1500 publicaciones de investigación y múltiples trabajos en la detección automática de emociones faciales, sin embargo, no es más usada que el conjunto de datos Cohn Kanade.

## *1.7 Conclusiones Parciales*

Mediante este capítulo se estudiaron las emociones como elemento fundamental utilizadas en la mayoría de los sistemas de detección automática de emociones en la actualidad. Las expresiones faciales de emociones resultan importantes en la interacción entre los humanos. El rostro es el principal sistema de señales para mostrar las emociones. La Computación Afectiva es una rama de la Inteligencia Artificial que hace referencia al diseño de sistemas y dispositivos para reconocer, interpretar, procesar y generar emociones humanas mejorando la interacción entre el usuario y la computadora.

La clasificación supervisada es la más utilizada en la clasificación de emociones. Los clasificadores más importantes y más utilizados para la clasificación automática de emociones son K Vecinos más Cercanos, Máquina de Soporte Vectorial, Red Neuronal Perceptrón Multicapa, Árbol de Decisión, Bosques Aleatorios y Naive Bayes.

Una API es un conjunto de funciones, estructuras de datos y clases que ofrece cierta biblioteca para ser utilizados por otro software como una capa de abstracción. Principalmente la API de tipo REST es la más utilizada a nivel mundial porque aporta independencia de tecnologías y lenguajes de programación a los desarrolladores. Dicho tipo es el ideal para implementar una API, porque permite integrar la detección automática de emociones faciales en diferentes aplicaciones independientemente del lenguaje de programación y el sistema operativo.

El framework Python más indicado para implementar una API REST es FastAPI. Debido a que está orientado solamente a la creación de API REST con calidad, minimalista, muy potente, moderno e intuitivo, es uno de los frameworks más rápidos disponibles en Python actualmente.

Python resulta ser uno de los mejores lenguajes de programación más potente y popular en la actualidad, utilizado para el procesamiento de imágenes, implementación de algoritmos de extracción de rasgos, detección de rostros y aprendizaje automático. Entre las diversas bibliotecas existentes, OpenCV, Dlib y Scikit-learn y NumPy resultan las más recomendadas para los trabajos relacionados con la clasificación de emociones.

Las bases de datos de emociones Cohn Kanade y KDEF, son las más difundidas en investigaciones y trabajos realizados en el campo de la Computación Afectiva. Ambas contienen las seis emociones universales y la neutral. Cohn Kanade es la más utilizada y variada por presentar imágenes con poca calidad de personas de diferentes razas, géneros y sexos.

# 2

# DISEÑO E IMPLEMENTACIÓN DE FER REST API

# 2. DISEÑO E IMPLEMENTACIÓN DE FER REST API

En el presente capítulo se aborda el diseño e implementación de la API de reconocimiento de emociones faciales a través de imágenes FER REST API. Se presentan las herramientas de software útiles en la creación de la API. Se aborda la arquitectura y funcionamiento de la API. Así como los diagramas UML utilizados para el modelado. Se exponen las funciones importantes en la implementación de la FER REST API. Finalmente se describe el proceso de instalación y ejecución de la API y la documentación como ventaja para los desarrolladores.

## *2.1 Ambiente de desarrollo*

Para el diseño e implementación de la API REST se utilizaron una serie de recursos de softwares necesarios en cada una de las etapas. Logrando crear un ambiente de desarrollo capaz de cumplir el objetivo principal del trabajo de diploma. A continuación, se presentarán cada una de las herramientas de softwares con sus características más relevantes que resultaron útiles en la realización del trabajo.

### 2.1.1 **Visual Paradigm**

Visual Paradigm (Visual Paradigm, 2019b) es una herramienta CASE (Ingeniería de Software Asistida por Computación) que ayuda al desarrollo de aplicaciones desde la planificación, pasando por las etapas de análisis y diseño. Proporciona capacidades de generación de informes e ingeniería de códigos, incluida la generación de códigos y documentación. Puede aplicar ingeniería inversa a los diagramas del código y proporcionar ingeniería de ida y vuelta para varios lenguajes de programación. Admite UML en su versión 2. Permite diseñar y generar API ORM y REST mediante las especificaciones Swagger (OpenAPI). Facilita la colaboración en equipo para realizar modelos de forma colaborativa y concurrente utilizando control de versiones. En el diseño de la API REST fue utilizado Visual Paradigm en su versión 15.2, en el modelado mediante UML. Se crearon los diagramas de recursos, componentes, despliegue, actividad y el modelo de implementación. La herramienta tiene un buen entorno de trabajo. Facilita la visualización y la manipulación del proyecto de modelado.

### 2.1.2 **PyCharm**

PyCharm (Jetbrains, 2019) es un entorno de desarrollo integrado (IDE) utilizado en la programación, específicamente para el lenguaje Python. Proporciona análisis de código, un depurador gráfico, un comprobador de unidades integrado. Es multiplataforma.

Se utilizó PyCharm en su versión Community 2018.3.4 para el desarrollo de FER REST API, por ser de código abierto y muy potente. Es el IDE más popular y poderoso para el lenguaje de programación Python en la actualidad. Brinda una interfaz amigable para el desarrollador, y con excelentes funcionalidades muy útiles y se encuentra en desarrollo activo. Presenta asistencia y análisis de codificación, con finalización de código, sintaxis y resaltado de errores e integración. Navegación de proyectos y códigos y refactorización del código. Soporte para frameworks web como: Django, FastAPI, Flask y Pyramid. Posee gran cantidad de plugins. Permite ejecutar, debuguear, testear. Soporta el trabajo con bases de datos y con múltiples paquetes incluidos y científicos.

### 2.1.3 **Postman**

El Postman (Postman Inc, 2019) es una herramienta que principalmente permite crear peticiones sobre APIs de una forma muy sencilla. Permite al desarrollador comprobar el funcionamiento de una API. Ofrece un conjunto de utilidades adicionales para poder gestionar las APIs de una forma más fácil. Proporciona herramientas para documentar y monitorización las APIs, crear equipos sobre una API para que trabajen de forma colaborativa.

Esta herramienta se utilizó en su versión 6.7.4 para comprobar el funcionamiento de la FER REST API, cumpliendo la función de cliente. Además, es una de las aplicaciones más completas para probar y desarrollar las API REST. Es muy utilizada por desarrolladores de API a nivel mundial y grandes compañías. Se encuentra bajo la especificación de OpenAPI. Postman para la realización de FER REST API proporciona un entorno de desarrollo que ayuda a la construcción, prueba, documentación, monitoreo y publicación de la documentación para la API. Permite crear y enviar peticiones HTTP a servicios REST mediante una interfaz gráfica. Permite importar API existentes desde Swagger. Se puede generar una documentación basada en las API.

### 2.1.4 **Weka**

Weka (Bouckaert *et al.*, 2018; Weka, 2019) está escrito en el lenguaje de programación Java. Es una colección de algoritmos de aprendizaje automático para tareas de minería de datos. Contiene herramientas para la preparación de datos, clasificación, regresión, agrupación, extracción de reglas de asociación y visualización, así como facilidades para su aplicación y análisis de prestaciones cuando son aplicadas a los datos de entrada seleccionados. Weka es un software de código abierto emitido bajo la Licencia Pública General de GNU. En las últimas versiones es posible aplicar Weka para procesar Big Data y realizar aprendizaje profundo. El software se utilizó para ajustar los parámetros de cada clasificador de aprendizaje automático. De esta manera validarlos y aumentar la efectividad para la correcta detección de las emociones humanas. Este proceso se realiza antes de la implementación de los clasificadores en el código fuente de la API, es un proceso de prueba y análisis de los clasificadores.

## *2.2 Arquitectura de FER REST API*

El fin de Facial Emotions Recognition REST API consiste en lograr la independencia de tecnologías y lenguajes de programación para la comunicación con las diferentes aplicaciones, un aspecto esencial de las API REST. De esta forma se cumple con el objetivo general del presente trabajo de diploma, en el que softwares y aplicaciones en los diferentes lenguajes de programación, sistemas operativos e implementados con distintas tecnologías puedan realizar la comunicación mediante la API REST con los recursos de la biblioteca Feel UCLV, e incluir la detección automática de emociones en su funcionamiento.

Desde cualquier tipo de software y aplicación independiente del sistema operativo y el lenguaje de programación que necesite reconocer las emociones faciales del humano se puede hacer uso de la FER REST API. El desarrollador, sin tener conocimiento profundo de la implementación de los sistemas automáticos de reconocimiento facial de emociones, puede vincular en su sistema algunos procesos de la computación afectiva. Para clasificar la emoción que experimenta una persona se envía vía URI la dirección de la imagen que se desea clasificar a la FER REST API. La API es quien se encarga de hacer la comunicación con la biblioteca de reconocimiento facial de emociones Feel UCLV. Esta biblioteca es la encargada de procesar la imagen y clasificar la emoción. Para ello hace uso de las bibliotecas OpenCV, Dlib, Scikit-learn y NumPy implementadas en Python. Incluye los clasificadores supervisados Red Neuronal Perceptrón Multicapa (MLP), Árbol de

Decisión (Decision Tree), Bosque Aleatorio (Random Forest), K Vecinos más Cercanos (KNN), Naive Bayes (Naive Bayes) y Máquinas de Soporte Vectorial (Support Vectorial Machine) con kernel lineal y polinomial. Según lo que se desee conocer (la emoción, todas las probabilidades por emociones) la API devuelve el resultado codificado en el formato JSON (véase Figura 2.1).

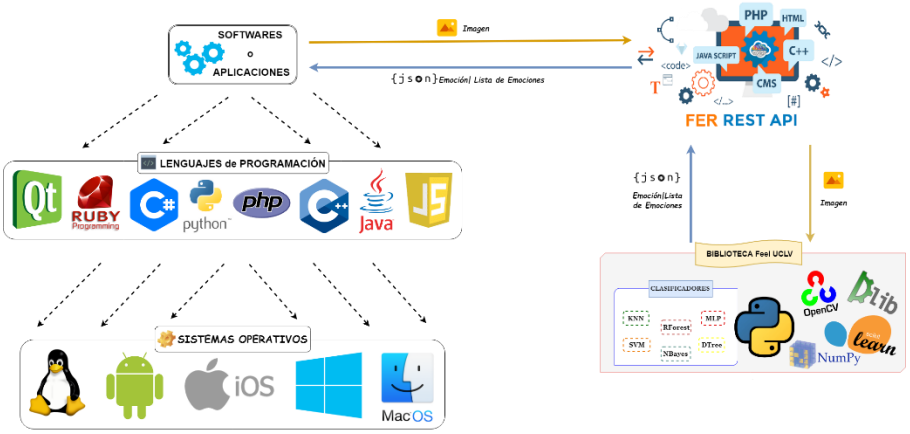



**Figura 2.1 Arquitectura de FER REST API**

## *2.3 Diseño UML FER REST API*

El Lenguaje Unificado de Modelado (UML) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad. Se le puede definir como un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema (Hernández Orallo, 2017).

UML ofrece un estándar para describir un plano del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio, funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y compuestos reciclados. Existen dos tipos de diagramas: los estructurales que entre ellos se encuentran los diagramas de clases, componentes, despliegue y objetos; y los diagramas de comportamiento como son el diagrama de actividades, casos de uso, máquinas de estado y secuencia.

### 2.3.1 Diagrama de Recursos

Un diagrama de clases es un tipo de diagrama de estructura estática que describe la estructura de un sistema mostrando las clases de este. La opción más viable para mostrar la estructura y funcionalidades que tendrá FER REST API es mediante este diagrama. Es posible realizar este diagrama pues en la nueva versión de la herramienta Visual Paradigm se incluye el componente

recurso (Visual Paradigm, 2019a). En el diagrama asociado a la FER REST API todos son recursos de tipo GET (véase Figura 2.2).

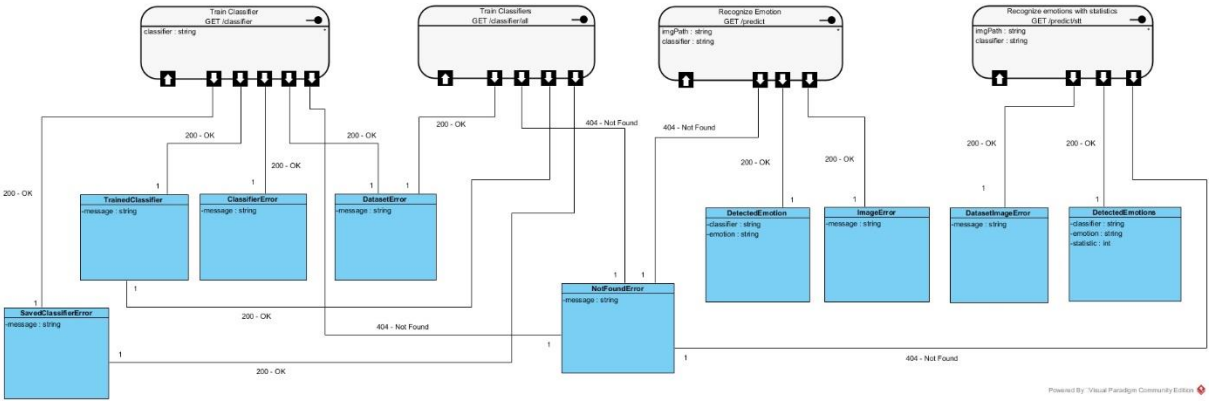



**Figura 2.2 Diagrama de recursos de FER REST API**

Todas las clases y atributos representan las respuestas de cada recurso de FER REST API mediante el formato JSON. Si la respuesta es de código 200 puede representar que se estableció una correcta comunicación con la API o que se le devolvió la respuesta indicada al desarrollador. Si es código 404 significa que existe un problema con la URI del recurso y no se logra una comunicación con los recursos de la API.

El recurso *Train Classifier* con URI */classifier* entrena un clasificador especificado mediante el parámetro de consulta de tipo cadena *classifier*.

El recurso *Train Classifiers* con URI /*classifier/all* entrena todos los clasificadores existentes en FER REST API.

El recurso *Recognize Emotion* con URI */predict* reconoce la emoción contenida en una imagen. El parámetro de consulta *imgPath* de tipo cadena es la ruta de la imagen y *classifier* de tipo cadena, es el clasificador especificado que se va utilizar para reconocer la emoción.

El recurso *Recognize emotion with statistics* con URI */predict/stt* reconoce las emociones con probabilidades a través de una imagen. El parámetro de consulta *imgPath* de tipo cadena es la ruta de la imagen, y *classifier* de tipo cadena es el clasificador especificado que se va utilizar para reconocer las emociones con probabilidades.

La clase *TrainedClassifier* posee el atributo *message* de tipo cadena que representa la respuesta de los recursos *Train Classifier* y *Train Classifiers* cuando se entrena correctamente un clasificador o todos los clasificadores de FER REST API.

La clase *ClassifierError* posee el atributo *message* que hace referencia a la respuesta del recurso *Train Classifier* cuando la API no contiene el clasificador especificado o es nulo ese parámetro de consulta del recurso. La clase *SavedClassifierError* y su atributo *message* de tipo cadena indica la respuesta de los recursos *Train Classifier* y *Train Classifiers* cuando el clasificador o los clasificadores entrenados no son guardados correctamente en archivos.

La clase *DatasetError* y su atributo *message* de tipo cadena se refiere a la respuesta de los recursos *Train Classifier* y *Train Classifiers* cuando no es encontrado el conjunto de datos de entrenamiento o no está debidamente estructurado para entrenar un clasificador o todos los clasificadores existentes.

La clase *DetectedEmotion* con los atributos *classifier* y *emotion* ambos de tipo cadena, representa la respuesta del recurso *Recognize Emotion*, *classifier* es el clasificador utilizado para el reconocimiento de la emoción representada en la imagen y *emotion*, la emoción detectada.

La clase *ImageError* y su atributo *message* de tipo cadena, representa la respuesta del recurso *Recognize Emotion* cuando no se puede reconocer la emoción en la imagen, porque no existe la imagen en la ruta especificada o no posee el formato correcto.

La clase *DetectedEmotions* y sus atributos *classifier*, *emotion*, ambos de tipo cadena y *statistic* de tipo entero representan la respuesta del recurso *Recognize emotion with statistics, classifier* es el clasificador utilizado para el reconocimiento de las emociones por probabilidades representadas en la imagen a través del rostro humano, *emotion* las emociones detectadas y *statistic* las probabilidades de cada emoción respectivamente.

La clase *DatasetImageError* y su atributo *message* de tipo cadena representa la respuesta del recurso *Recognize emotion with statistics* cuando no se puede reconocer las emociones por probabilidades en la imagen, porque no existe la imagen en la ruta especificada o no posee el formato correcto. Además, hace referencia al conjunto de datos con que se entrenó el clasificador utilizado, que no se encuentra o no está correctamente estructurado para obtener los nombres de cada emoción para asociarla a su probabilidad.

La clase *NotFoundError* y el atributo *message* de tipo cadena representan la respuesta de todos los recursos de Facial Emotions Recognition REST API cuando existe un error en su URI y no se puede acceder a ellos.

### 2.3.2 Diagrama de Componentes

Según Pressman (2010) los diagramas de componentes describen los elementos físicos del sistema y sus relaciones. Un diagrama de componentes representa las dependencias entre componentes software, incluyendo componentes de código fuente, componentes del código binario, y componentes ejecutables.

Los diagramas de componentes según Bailón Delgado (2015) muestran los componentes del software y los artilugios de que está compuesto como los archivos de código fuente, las librerías o las tablas de una base de datos. Los componentes pueden tener interfaces (es decir clases abstractas con operaciones) que permiten asociaciones entre componentes.

El diagrama de componentes representa cómo un sistema de software es dividido en componentes y muestra la organización y dependencias entre estos componentes. Los componentes físicos incluyen archivos, tablas, bibliotecas, paquetes, bibliotecas compartidas (DLLs), bases de datos, módulos, documentos, ejecutables, o paquetes. Uno de los usos principales es que puede servir para ver qué componentes pueden compartirse entre sistemas o entre diferentes partes de un sistema.

En el diagrama de componentes se representa el subsistema de implementación *Facial Emotions Recognition REST API* que hace referencia a la API de detección automática de emociones faciales con los componentes usados para su funcionamiento (véase Figura 2.3).

El subsistema contiene un componente principal, el módulo de Python llamado *api.py*. Este contiene el código fuente de la API. Depende del componente de tipo base de datos que representa el conjunto de datos llamado *Database of emotions*. También se encuentra el archivo *shape_predictor_68_face_landmarks.dat* perteneciente a la biblioteca Dlib y utilizado para el reconocimiento de las 68 marcas faciales para la extracción de características del rostro humano.

El componente *api.py* depende además de *classifier.dat* que es del tipo estereotipo archivo. Este representa el clasificador entrenado que se guardará en un fichero. A su vez, él dependerá de la base de datos de emociones para realizar el entrenamiento. Por último, el módulo Python va a hacer uso también de un componente físico llamado *Image.* Este es de tipo archivo, representa así la imagen para el reconocimiento de la emoción.

El subsistema de implementación *Programming Language, Framework and Libraries* hace referencia al lenguaje de programación, framework utilizado para la implementación de la API y las bibliotecas necesarias. Por tanto, el subsistema *Facial Emotions Recognition REST API* depende

de este subsistema. El segundo subsistema de implementación está compuesto por los componentes de estereotipo biblioteca. Estas representan a OpenCV, Dlib, NumPy y Scikit-learn. Todas estas dependen internamente de la biblioteca NumPy para su funcionamiento.

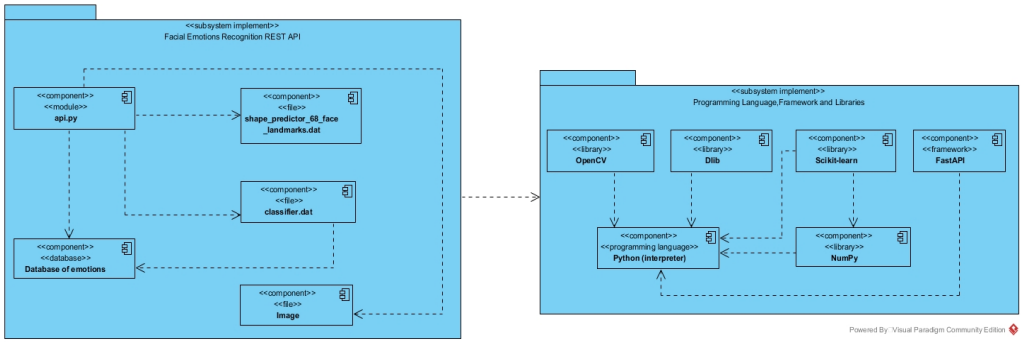



**Figura 2.3 Diagrama de componentes de FER REST API**

El componente de tipo framework FastAPI se utiliza para la creación de FER REST API. Por tanto, todos estos componentes dependen del componente Python. Este es de tipo lenguaje de programación y hace referencia al intérprete del lenguaje. Es sumamente importante e imprescindible para el funcionamiento de las bibliotecas, el framework y la API REST.

### 2.3.3 Diagrama de Despliegue y Modelo de implementación

Según Pressman (2010) los diagramas de despliegue muestran las relaciones físicas de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos. El diagrama de despliegue muestra el ambiente de computación, pero no indica de manera explícita los detalles de la configuración.

Los diagramas de despliegue UML según Sommerville (2011) muestran cómo los componentes de software se despliegan físicamente en los procesadores; es decir, el diagrama de despliegue muestra el hardware y el software en el sistema, así como el middleware usado para conectar los diferentes componentes en el sistema. En esencia, los diagramas de despliegue se pueden considerar como una forma de definir y documentar el entorno objetivo.

El diagrama de despliegue, según Bailón Delgado (2015) muestra la arquitectura física de un sistema informático. Puede representar a los equipos y a los dispositivos, y también mostrar sus interconexiones y el software que se encontrará en cada computadora.

El diagrama de despliegue según Gómez Palomo and Moraleda Gil (2014) muestra cómo el sistema se asentará físicamente en el entorno hardware que lo acompaña. Su propósito es mostrar dónde los componentes del sistema se ejecutarán y cómo se comunicarán entre ellos.

Port tanto, los diagramas de despliegue son los complementos de los diagramas de componentes que, unidos, proveen la vista de implementación del sistema. Describen la topología del sistema, la estructura de los elementos de hardware y software que ejecuta cada uno, las conexiones físicas entre el hardware y las relaciones entre componentes.

Los elementos usados por este tipo de diagrama son los nodos que representan un recurso físico que pueden ser computadoras, impresoras, servidores o dispositivos externos, además están los componentes y las asociaciones. Los nodos son conectados por asociaciones de comunicación tales como enlaces de red, conexiones TCP/IP. Principalmente los diagramas de despliegue se utilizan para modelar sistemas cliente-servidor y sistemas completamente distribuidos. El diagrama de despliegue correspondiente a FER REST API muestra estos elementos y sus asociaciones (véase Figura 2.4).

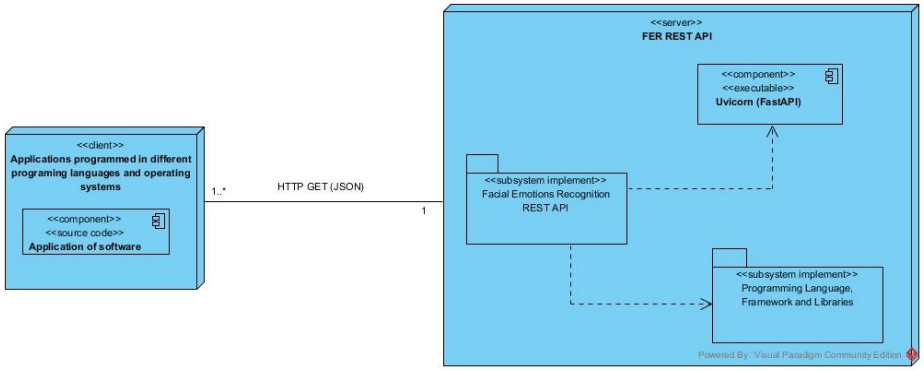



**Figura 2.4 Diagrama de despliegue de FER REST API**

En el diagrama de despliegue se muestra el nodo cliente que representa las múltiples aplicaciones implementadas en diferentes lenguajes de programación y sistemas operativos que van a hacer uso del nodo servidor *FER REST API.* El nodo cliente tiene un componente de tipo código fuente. Desde su código fuente se necesita hacer uso de la API. La comunicación se hará mediante el protocolo HTTP a través de los recursos de tipo GET de la API REST y el intercambio de datos se realizará mediante JSON.

En el nodo servidor es donde se alojará la FER REST API y se ejecutará. Está compuesto por el subsistema de implementación *Facial Emotions Recognition REST API*. Esta es la implementación

real de la API. Depende del subsistema *Programming Language, Framework and Libraries*. Además, en el nodo servidor se encuentra el componente ejecutable *Uvicorn*. Este es el servidor perteneciente al framework FastAPI que ejecutará la API REST. A su vez el subsistema *Facial Emotions Recognition REST API* depende del servidor para su funcionamiento y ejecución.

***Modelo de implementación***

Según Pantaleo and Rinaudo (2015) el modelo de implementación describe cómo los elementos del modelo de diseño y las clases se implementan en términos de componentes, como ficheros de código fuente, ejecutables, etc. Describe, también, cómo se organizan los componentes de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación y en los lenguajes de programación utilizados.

Los diagramas de implementación según Bailón Delgado (2015) muestran las instancias existentes al ejecutarse, así como sus relaciones. También se representan los nodos que identifican recursos físicos, típicamente un ordenador, así como interfaces y objetos (instancias de las clases).

Por tanto, es constituido por un conjunto de componentes y subsistemas de implementación que constituyen la composición física de la implementación del sistema. Entre los componentes (elementos de implementación) se encuentran datos, archivos, ejecutables, código fuente y directorios, entre otros. Fundamentalmente, se describe la relación que existe desde los paquetes y clases del modelo de diseño a subsistemas y componentes físicos. El modelo de implementación muestra las dependencias entre las partes de código del sistema (diagramas de componentes) y la estructura del sistema en ejecución (diagrama de despliegue). Así lo representa el Modelo de Implementación de FER REST API (véase Figura 2.5).

El modelo de implementación es similar al diagrama de despliegue combinado con el de componentes. El nodo cliente que representa las múltiples aplicaciones implementadas en diferentes lenguajes de programación y sistemas operativos que van a hacer uso del nodo servidor *FER REST API* donde se alojará la API.

En el nodo cliente se tiene un componente de tipo código fuente que representa la aplicación de software que necesita hacer uso de la API desde su código. La comunicación se hará mediante el protocolo HTTP a través de los recursos de tipo GET de la API REST y el intercambio de datos se realizará mediante JSON.

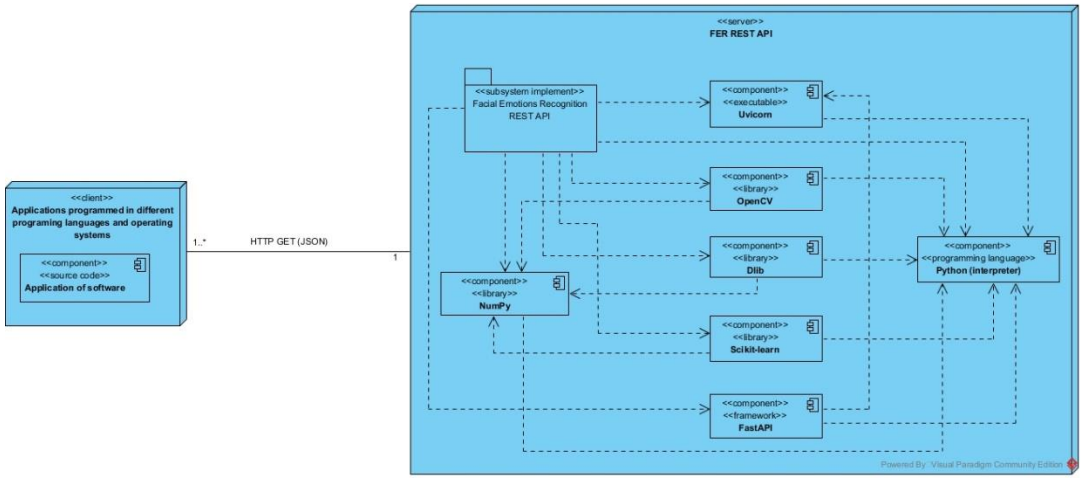



**Figura 2.5 Modelo de implementación de FER REST API**

El servidor representa el nodo donde se instalará la FER REST API y se ejecutará. Está compuesto por el subsistema de implementación *Facial Emotions Recognition REST API* que es la implementación de la API de reconocimiento de emociones faciales. En este nodo se encuentran los componentes de tipo biblioteca OpenCV, Dlib y Scikit-learn que representan las bibliotecas utilizadas y que van a depender internamente para su funcionamiento del componente biblioteca NumPy. Además, se encuentra el componente de tipo framework FastAPI utilizado en la implementación de la API REST. En el nodo servidor también va a estar desplegado el componente ejecutable *Uvicorn* que es el servidor perteneciente al framework FastAPI y que ejecuta la API REST. A su vez el subsistema *Facial Emotions Recognition REST API* depende del servidor ejecutable *Uvicorn* y de todos los componentes bibliotecas para su funcionamiento y ejecución. Todos los componentes y el subsistema de implementación van a depender del lenguaje de programación Python y su intérprete para su correcto desempeño.

## *2.4 Implementación de FER REST API*

La API de Reconocimiento de Emociones Faciales (FER REST API) está implementada con FastAPI. Hace uso de las bibliotecas OpenCV, Dlib, Scikit-learn y NumPy. Está desarrollada para el reconocimiento facial de imágenes frontales de una persona y estructurada de una forma sencilla (véase Figura 2.6).

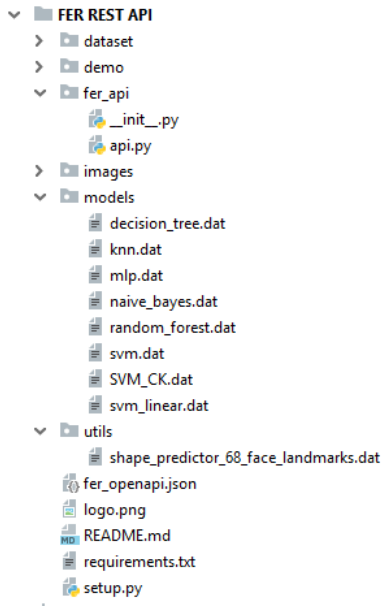



**Figura 2.6 Estructura de FER REST API**

En el nivel más alto de la estructura jerárquica existen seis carpetas: "dataset", "demo", "fer_api", "images", "models" y "utils".

La carpeta "dataset" contiene el conjunto de datos de entrenamiento.

En la carpeta "demo" se encuentra almacenado el código fuente de la interfaz visual de ejemplo de la API (véase Anexo 14), implementada por la biblioteca PyQt. Este ejemplo permite observar el funcionamiento de los algoritmos implementados. Se puede visualizar la detección de rostros y la extracción de las 68 marcas faciales. Además, se puede visualizar todo el preprocesamiento de las imágenes que se realiza en la API. También incluye la clasificación por probabilidades de las emociones utilizando el clasificador Máquina de Soporte Vectorial (SVM) con kernel polinomial entrenada con la base de datos de emociones Cohn Kanade.

La carpeta "fer_api" contiene el código fuente de la API. En el módulo *api.py* se encuentran las funciones elementales. Además, contiene el módulo *_init_.py*, en el que se encuentran los métodos de FER REST API. Este permite que sea instalada directamente en el intérprete de Python como una API de código fuente para el lenguaje.

En la carpeta "images" se almacenan las imágenes que serán procesadas.

En la carpeta "models" se encuentran los archivos de los clasificadores entrenados.

En la carpeta "utils" se encuentra el archivo utilizado por la biblioteca Dlib para la extracción de las marcas faciales del rostro humano.

Externos a las carpetas antes mencionadas se encuentran los ficheros adicionales necesarios para el correcto funcionamiento y la configuración de la API. Está presente la imagen *logo.png* que representa el logotipo de la API. Asimismo, el fichero *README.md* que describe una pequeña documentación para el desarrollador. El archivo *requirements.txt* que contiene los requisitos para el correcto funcionamiento de la API. Además, el archivo *fer_openapi.json* que representa la especificación OpenAPI de FER REST API para el desarrollador (véase Anexo 37). Finalmente contiene el módulo *setup.py* que sirve para la instalación de la FER REST API como dependencia en el intérprete de Python y su posterior uso como una biblioteca en el lenguaje. La instalación se realiza utilizando el módulo *__init__.py* de la carpeta "fer_api".

En la implementación de FER REST API se utiliza la biblioteca OpenCV para el procesamiento y manipulación de las imágenes. Dlib para la extracción de las marcas faciales en el rostro humano y obtener el vector de características. NumPy por el tipo de arreglo multidimensional utilizado y Scikit-learn para la implementación de los clasificadores de aprendizaje automático.

### 2.4.1 Función para la extracción de características

La función `describe` del módulo *api.py* realiza la extracción de características de una imagen que contiene el rostro humano devolviendo el vector de rasgos. Dicha función carga la imagen a partir de su dirección física, haciendo uso de un método de la biblioteca OpenCV. Este método devuelve, en formato BGR, un arreglo con la información del nivel de intensidad de los píxeles en cada canal. Los formatos de imágenes aceptados son los siguiente: Windows bitmaps (*.bmp, *.dib), JPEG (*.jpeg, *.jpg, *.jpe), JPEG 2000 (*.jp2), Portable Network Graphics (*.png), WebP (*. webp), Formato portable de imagen (*.pbm, *.pgm, *.ppm *.pxm, *.pnm), PFM (*.pfm), tramas Sun (*.sr, *.ras), TIFF (*.tiff, *.tif), archivos de imagen OpenEXR (*.exr), Radiance HDR (*.hdr, *.pic), tramas y vectores geoespaciales sustentados por GDAL. Este método devuelve nulo si el formato no es correcto, no se encuentra el archivo físico o no se tienen los permisos para acceder a este.

Posteriormente la imagen se convierte a escala de grises usando un método de la biblioteca OpenCV (véase Anexo 16). En esta etapa de realce y normalización, se tiene en cuenta que las variaciones en la dirección y la intensidad de la iluminación son factores que modifican

significativamente la apariencia de los objetos en una imagen digital (Singh Negi and Singh Bhandari, 2014). La iluminación ambiental puede variar a lo largo del día e influir en la imagen dificultando la detección de rostros, ya sea en interiores o exteriores. Debido a la forma tridimensional de los objetos, una fuente de iluminación puede generar sombras que acentúan o disminuyen ciertos rasgos de la imagen. Distintas condiciones de iluminación pueden producir representaciones desiguales de un mismo objeto. Dichas variaciones son indeseables pues dificultan el proceso de reconocimiento de patrones (Hussain Shah *et al.*, 2015).

Para normalizar la iluminación en la imagen un ecualizador adaptativo de histograma con contraste limitado se aplica a la imagen. Este es creado con una función de OpenCV que recibe dos parámetros. El primero especifica el mayor número de píxeles que puede tener una barra del histograma (umbral para limitar el contraste). El otro, el tamaño de la ventana que se calculará para la ecualización del histograma. Se utiliza para el primer parámetro valor 2 y para el segundo 8,8. Estos valores son recomendados en la literatura consultada. El histograma de una imagen es la cantidad de píxeles que tienen la misma intensidad dentro de la escala de niveles de gris. En Zuiderveld (1994); Magudeeswaran and Singh (2017) se resume la ecualización adaptativa del histograma con contraste limitado (CLAHE). Para evitar amplificar el ruido se aplica el método de ecualización adaptativa del histograma con contraste limitado.

Para evitar crear un nuevo ecualizador por cada imagen, este es creado una sola vez y luego se le aplica a cada imagen (véase Anexo 17).

Luego se pasa a la etapa de detección del rostro haciendo uso de un método de la biblioteca Dlib (véase Anexo 18). Esta función se usa una sola vez para crear un detector de rostros. Seguidamente, al detector se le pasa la imagen y se le indica cuántas veces se debe remuestrear la imagen. En el método, la imagen se remuestrea una vez. Este método es el propuesto en Dalal and Triggs (2005) y mejorado según Felzenszwalb *et al.* (2010). Utiliza un algoritmo de histograma de orientación de gradientes (HOG), combinado con un clasificador lineal Máquina de Soporte Vectorial (SVM) y el método de detección de imagen piramidal y ventana deslizante. Este método es muy eficaz y rápido para imágenes frontales y de una sola persona. Este devuelve el rectángulo en el que está contenido el rostro.

Si ningún rostro es detectado se informa y la imagen es ignorada. Si es detectado más de un rostro se informa y se procesa solo la primera detección.

Luego la imagen es recortada a la zona del rostro mediante un método de la biblioteca OpenCV, que posee entre sus parámetros la imagen con las coordenadas del rostro detectado y el tamaño de la imagen de salida. La imagen es llevada a un tamaño de 350x350 para agregar invariancia ante la proximidad del rostro a la cámara.

Para detectar las marcas faciales se hace uso de un método de la biblioteca Dlib (véase Anexo 20) y se carga un archivo que permite detectar las 68 marcas faciales. El detector de marcas faciales se crea una sola vez. Después, en cada imagen se especifica el rectángulo donde debe buscar el rostro y devuelve una lista con las coordenadas de los 68 puntos (véase Anexo 21).

El detector de marcas faciales está basado en la propuesta de los autores Kazemi and Sullivan (2014). Fue entrenado con la base de datos iBUG 300-W. Este es capaz de extraer las 68 marcas faciales en tiempo real con una calidad alta.

Para evitar la rotación de la cabeza en el reconocimiento de las marcas faciales, se realiza una rotación a cada marca. Los valores rotados, son almacenados en un arreglo y son ordenados de acuerdo a la marca facial que representan. Se utiliza un método de NumPy para devolver el arreglo de forma contigua con las diferentes marcas faciales encontradas.

Se realizó un diagrama de actividad para el método `describe` que resume su funcionamiento (véase Figura 2.7). Los diagramas de actividad según Bailón Delgado (2015) describen la secuencia de las actividades en un sistema. Son una forma especial de los diagramas de estado, que únicamente (o mayormente) contienen actividades.

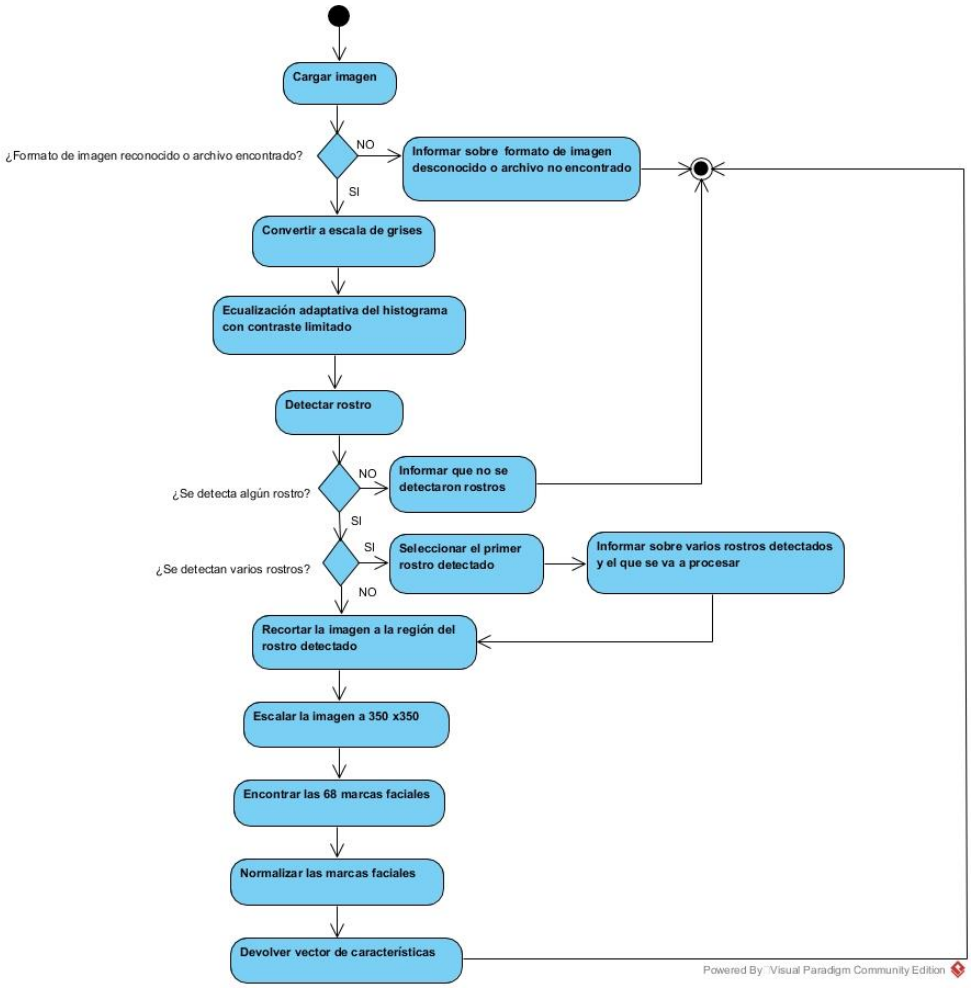



**Figura 2.7 Diagrama de actividad de la función *describe***

### 2.4.2 Función para Entrenar un solo Clasificador

La función `saveEmotionDetector` del módulo *api.py* entrena un clasificador de los existentes en FER REST API recibido por parámetro. Cuando se entrena se guarda en un archivo.

Un ejemplo de URL sería la siguiente: http://localhost:8000/classifier?classifier=<string>. Donde la URL base sería localhost:8000 que es el servidor por defecto, utilizando el protocolo HTTP y la URI para acceder al recurso. Se tiene además un parámetro que indica al método, el clasificador a entrenar. En dependencia del valor recibido, se entrena un clasificador. Si el valor es SVM entrena una Máquina de Soporte Vectorial con kernel polinomial, SVMLinear una Máquina de Soporte Vectorial con kernel lineal, KNN el clasificador K Vecino más Cercanos, NaiveBayes el clasificador Naive Bayes, DecisionTree un Árbol de Decisión, RandomForest el clasificador Bosque Aleatorio y MLP entrena una Red Neuronal Artificial Perceptrón Multicapa. De esta forma

se brinda la posibilidad de escoger el clasificador a entrenar al desarrollador. Si se escoge algún clasificador que no existe en la API o es nulo el parámetro, se notifica al desarrollador que debe especificar uno de los clasificadores.

La cantidad de emociones a detectar y la forma de expresarlas, pueden variar de una aplicación a otra en dependencia de la necesidad. Estas características dependen de la estructura del conjunto de datos de entrenamiento (base de datos de emociones) y de las emociones que estén representadas para la clasificación supervisada. La API permite al desarrollador crear clasificadores personalizados con las emociones de interés. Además, puede escoger su conjunto de datos con las imágenes que sean de su preferencia. El conjunto de datos tiene que estar alojado en la carpeta "dataset".

La función `saveEmotionDetector` verifica que la base de datos esté bien estructurada y notifica al desarrollador si no lo está.

A cada imagen dentro del directorio de la emoción se le aplica el método `describe`. Si se obtiene correctamente el vector de rasgos, se guarda.

Después de guardar todos los vectores de características del conjunto de entrenamiento y las clases (emociones), se crean los modelos del clasificador correspondiente.

FER REST API permite crear y entrenar los clasificadores correspondientes haciendo uso de la biblioteca Scikit-learn. Utilizando el método `SVC`, se crea una Máquina de Soporte Vectorial (SVM) (véase Anexo 23) con varios parámetros. El primero es el kernel polinomial con grado 1,6. El segundo, verdadero para la probabilidad (representa la estimación de las probabilidades para la clasificación). Para los restantes parámetros del método se tomaron los valores por defecto. Estos valores son: la penalización del error con valor 1, la tolerancia para el criterio de parada con valor 0,0001, el coeficiente del kernel con valor 1/68 y la cantidad máxima de iteraciones con valor -1 que representa que no va a tener límites.

Con el método `SVC` se crea, además, una Máquina de Soporte Vectorial (SVM) (véase Anexo 24) con los siguientes parámetros: kernel lineal y verdadero para la probabilidad. Para los restantes parámetros del método se tomaron los valores por defecto. Estos valores son: la penalización del error con valor 1, la tolerancia para el criterio de parada con valor 0,0001, el coeficiente del kernel con valor 1/68 y la cantidad máxima de iteraciones con valor -1.

Haciendo uso del método `KNeighborsClassifier` se crea el clasificador K Vecinos más Cercanos (véase Anexo 25). Recibe como parámetros: k prototipos más cercanos al patrón a clasificar dentro del conjunto de datos igual a 10. La inclusión de varios procesadores en el proceso de búsqueda de los vecinos con valor -1, que representa la inclusión de todos los procesadores que se encuentren disponibles. Para los restantes parámetros del método se tomaron los valores por defecto. Estos parámetros son: los pesos de las funciones usadas para la predicción que son constantes y uniformes. La función de distancia utilizada para medir la proximidad es la Distancia Euclidiana.

Utilizando el método `GaussianNB` se crea el clasificador Gaussian Naive Bayes con distribución Gaussiana (véase Anexo 26).

Con el método `MLPClassifier` se crea una Red Neuronal Artificial Perceptrón Multicapa (véase Anexo 27) con los siguientes parámetros: 65 neuronas con una capa oculta, que representa la cantidad de neuronas por capas ocultas. Para los restantes parámetros del método se tomaron los valores por defecto. Estos parámetros son: la función de activación `relu` utilizada para las capas ocultas. El parámetro que representa la solución para la optimización de los pesos con valor `adam`, que es un optimizador basado en gradientes estocástico propuesto en Kingma and Lei Ba (2015), es recomendado y obtiene resultados excelentes en conjuntos de entrenamientos grandes. Otros valores por defecto de los parámetros son: la tasa de aprendizaje inicial usada con valor 0,001, la tolerancia para la optimización con valor 0,00001 y el valor numérico para la estabilidad en `adam` es $1e-8$.

Empleando el método `DecisionTreeClassifier` se crea el clasificador Árbol de Decisión (véase Anexo 28) con los siguientes parámetros: el criterio que se utiliza para realizar la división (split), es la entropía y obtener la ganancia de información. La máxima cantidad de nodos hojas con valor 5. Para los restantes parámetros del método se tomaron los valores por defecto. Estos valores son: la estrategia para la división en cada nodo con el mejor (best) valor. La máxima profundidad del árbol con valor nulo. El número mínimo de pruebas requeridas para un nodo de hoja con valor 1, y el mínimo número para separar un nodo si su impureza está por encima del umbral, de otra manera es una hoja con valor $1e-7$.

Haciendo uso del método `RandomForestClassifier` se crea el meta clasificador Bosque Aleatorio (véase Anexo 29). Recibe los siguientes parámetros: la cantidad de árboles que contiene

el bosque es 30. La función para medir la calidad de la división (split) de los árboles es la entropía y para obtener la ganancia de información. La inclusión de varios procesadores en el proceso de búsqueda de los vecinos con valor -1. Para los restantes parámetros del método se tomaron los valores por defecto. Estos valores son: la máxima profundidad de los árboles con valor nulo. El número mínimo de pruebas requeridas para un nodo hoja con valor 1, y la máxima cantidad de nodos hojas de los árboles con valor nulo, representa que no va a tener restricción.

Después de creado el modelo del clasificador determinado se realiza el entrenamiento (véase Anexo 30). Para ello es necesario pasarle todos los vectores de características del conjunto de entrenamiento y las clases (emociones).

Para el uso posterior, se crea un archivo binario con extensión *dat*. Estos archivos son creados en el directorio "models" con el nombre de cada clasificador. Por ejemplo *svm.dat* representa el fichero del clasificador SVM con kernel polinomial, *svm_linear.dat* una SVM con kernel lineal, *mlp.dat* para MLP, *knn.dat* para el KNN, *naive_bayes.dat* para Naive Bayes, *decision_tree.dat* para el clasificador de Árbol de Decisión y *random_forest.dat* para Bosque Aleatorio.

Si se logró guardar el clasificador entrenado en un fichero o dio error, se notifica al desarrollador.

El diagrama de actividad asociado al método `saveEmotionDetector` despliega su funcionamiento (véase Figura 2.8).

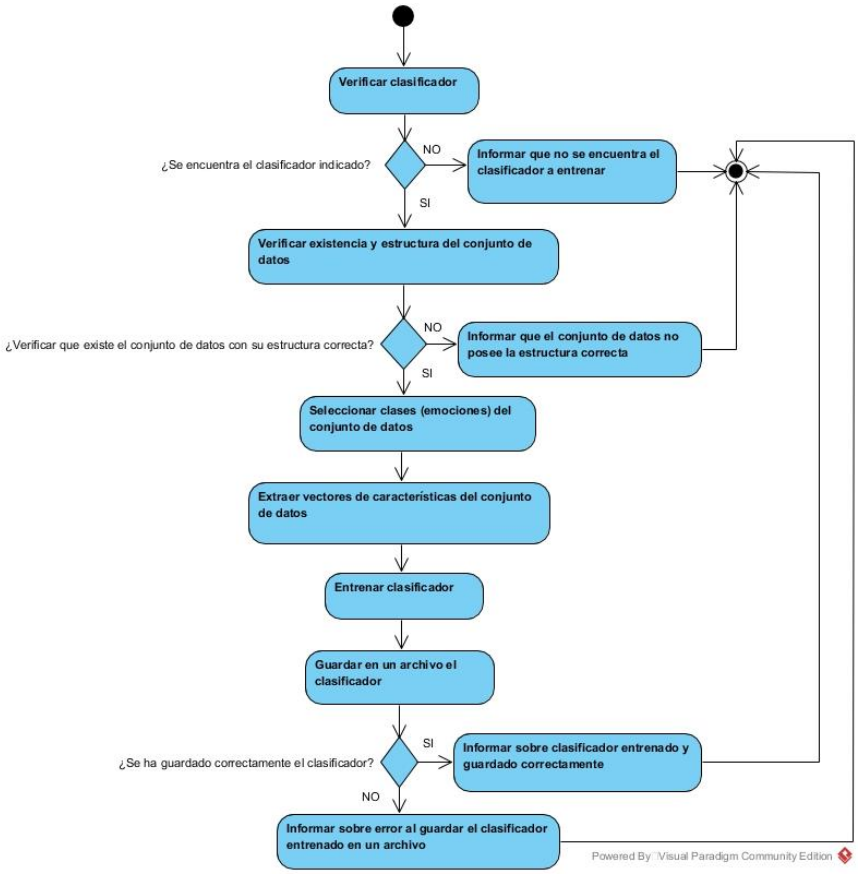



**Figura 2.8 Diagrama de actividad de la función *saveEmotionDetector***

### 2.4.3 Función para entrenar varios clasificadores

La función `saveAllEmotionDetector` del módulo *api.py* entrena todos los clasificadores y los guarda en archivos.

Un ejemplo de URL sería la siguiente http://localhost:8000/classifier/all donde la URL base sería localhost:8000 que es el servidor por defecto, utilizando el protocolo HTTP y la URI para acceder al recurso. En el método `saveAllEmotionDetector`, como ocurre en `saveEmotionDetector`, las emociones a detectar pueden variar de una aplicación a otra en dependencia de la necesidad del desarrollador. La función verifica que la base de datos esté bien estructurada y notifica al desarrollador si no lo está. Se realizan los mismos pasos descritos en el método `saveEmotionDetector`. Si se logró guardar todos los clasificadores entrenados en ficheros o dio error, se notifica al desarrollador.

Para este método se realizó un diagrama de actividad que representa su comportamiento (véase Figura 2.9).

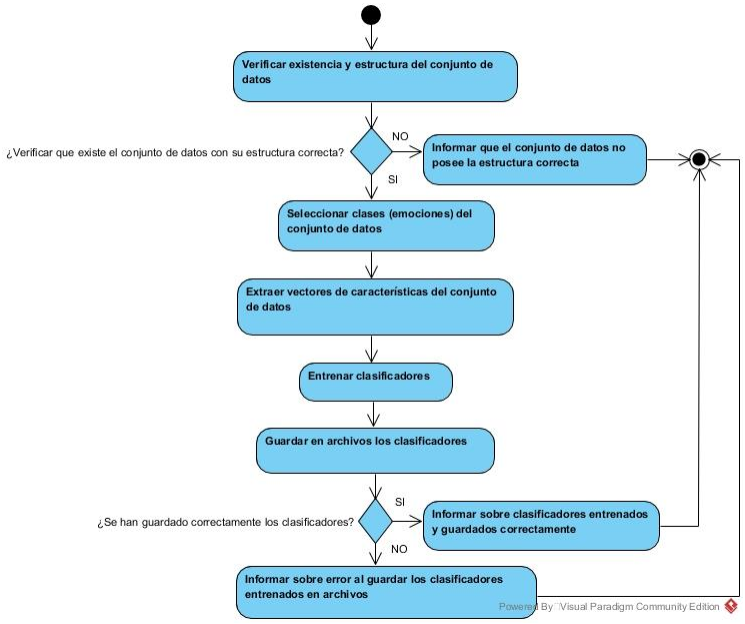



**Figura 2.9 Diagrama de actividad de la función *saveAllEmotionDetector***

### 2.4.4 Función para el reconocimiento de emociones

La función `predictEmotion` del módulo *api.py* reconoce una emoción en una imagen. El método recibe la dirección de la imagen y el clasificador con el cual se reconocerá la emoción.

Un ejemplo de URL sería la siguiente http://localhost:8000/predict?imgPath=<string>&classifier=<string> donde la URL base sería localhost:8000 que es el servidor por defecto, utilizando el protocolo HTTP y la URI para acceder al recurso. Se tienen los párametros *imgPath*, que indica la ruta de la imagen y *classifier*, que indica el clasificador a utilizar en la detección de la emoción.

Si el clasificador especificado está previamente entrenado se carga el archivo del clasificador desde el directorio "models" (véase Anexo 31).

Si el clasificador pasado como parámetro no hace referencia a ninguno de los clasificadores entrenados por FER REST API, tiene valor nulo, no se encuentra previamente entrenado o el archivo no puede ser cargado, la API usa un clasificador por defecto. Este clasificador es una Máquina de Soporte Vectorial (SVM) entrenado con el conjunto de datos Cohn Kanade.

Luego de cargar el archivo del clasificador, a la imagen se le aplica la función `describe` la cual devuelve el vector de rasgos. Al modelo cargado del archivo correspondiente del clasificador entrenado se le aplica el método `predict` que posee cada modelo de clasificador de la biblioteca

Scikit-learn. Con este método se detecta la emoción correspondiente, la cual devuelve la emoción en dependencia de las clases (emociones) que fueron entrenadas. El método `predict` recibe el vector de características de la imagen, y verifica la similitud del mismo con los vectores de características utilizados en la etapa de entrenamiento del clasificador (véase Anexo 32). El clasificador, al encontrar una semejanza con el vector de características de la imagen, devuelve la emoción. La respuesta que le da el sistema al desarrollador es en formato JSON del tipo clave-valor. En el primer par la clave es *Classifier* que representa el nombre del clasificador utilizado en el reconocimiento de la emoción. En el segundo es *Emotions* que representa la emoción detectada que depende de cuales hayan sido las entrenadas por el clasificador y el conjunto de datos de entrenamiento utilizado para ello. En el caso de que el clasificador utilizado sea el por defecto las emociones que reconoce son: *anger* (ira), *contempt* (desprecio), *disgust* (asco), *fear* (miedo), *happiness* (alegría), *neutral* (neutral), *sadness* (tristeza) y *surprise* (sorpresa).

En el caso que no se pueda realizar la extracción de rasgos de la imagen porque su formato sea incorrecto, o no se encuentre en la dirección física**,** se notifica al desarrollador del error.

Para este método se realizó un diagrama de actividad que representa su comportamiento (véase Figura 2.10).

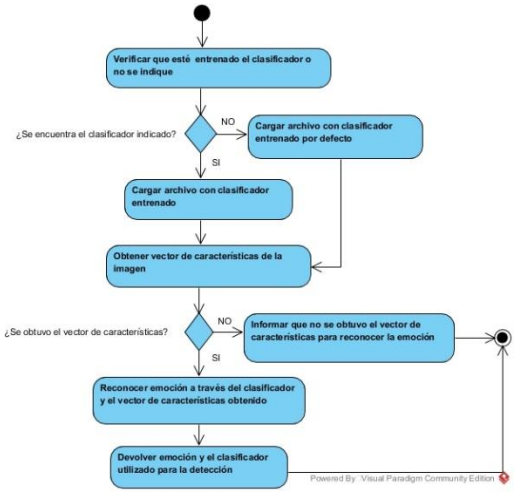



**Figura 2.10 Diagrama de actividad de la función *predictEmotion***

### 2.4.5 Función para el reconocimiento de emociones con probabilidades

La función `predictEmotionStt` del módulo *api.py* reconoce las probabilidades por emociones en una imagen. Utiliza dos parámetros, el primero es la dirección de la imagen y el segundo, que

por defecto es nulo, el clasificador con el cual se reconocerán las probabilidades por emociones en la imagen.

Un ejemplo de URL es http://localhost:8000/predict/stt?imgPath=<string>&classifier=<string>. Los valores que puede tomar el párametro *classifier* son los mismo que la función `predictEmotion`.

La función necesita obtener los nombres de las emociones (clases) de manera automática del conjunto de datos de entrenamiento con que se entrenó el clasificador, y de esta manera asociar a cada emoción su probabilidad. Para ello escanea la base de datos almacenada en el directorio "dataset". Si no está estructurado el conjunto de datos se notifica del error al programador.

Luego de cargar el archivo del clasificador, a la imagen se le aplica la función `describe`. Al modelo cargado del archivo correspondiente del clasificador entrenado se le aplica el método `predict_proba` que posee cada modelo de clasificador de la biblioteca Scikit-learn. Con este método se estiman las probabilidades de cada clase (emoción) del clasificador entrenado. El método `predict_proba` recibe el vector de características de la imagen a analizar para verificar la similitud del mismo con los vectores de características utilizados en la etapa de entrenamiento del clasificador (véase Anexo 33). El formato que se notifica al desarrollador es un JSON de tipo clave-valor. El primer par con clave *Classifier* representa el nombre del clasificador utilizado en el reconocimiento. Los demás pares son cada emoción con su nombre y su probabilidad en formato entero, las cuales varían en dependencia del conjunto de datos de entrenamiento y del clasificador.

En el caso que no se pueda realizar la extracción de características de la imagen porque su formato sea incorrecto, o no se encuentre en la ruta física**,** se notifica al desarrollador del error.

Para este método se realizó un diagrama de actividad que representa su comportamiento (véase Figura 2.11).

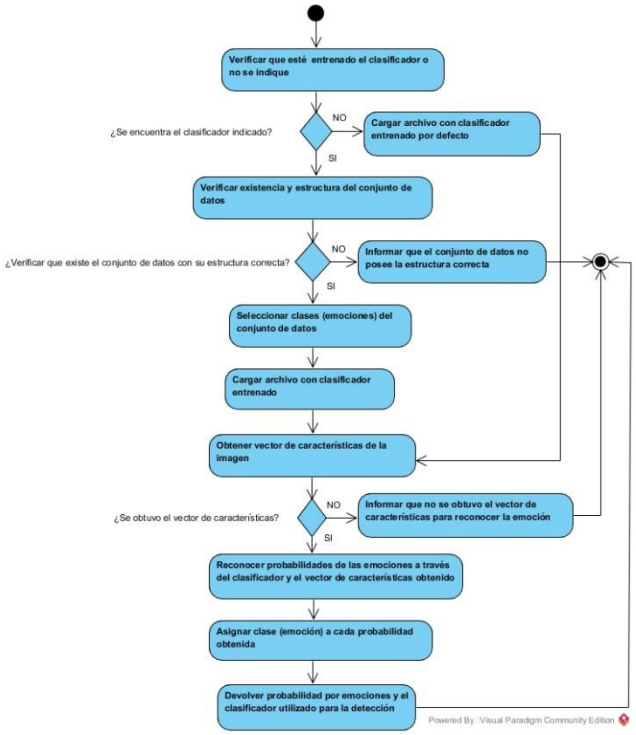



**Figura 2.11 Diagrama de actividad de la función *predictEmotionStt***

## *2.1 Instalación de FER REST API*

En este epígrafe se explica el proceso de instalación de FER REST API desde el lenguaje de programación Python con sus dependencias. Además, las distintas documentaciones que posee la API REST para auxiliar a los desarrolladores. Se describe como es el proceso de ejecución de FER REST API para su uso e inclusión de los recursos de la API en las aplicaciones.

### 2.1.1 Python y dependencias

Para la correcta instalación de FER REST API se debe descargar la versión de Python 3.6.x desde su sitio oficial http://www.python.org/downloads/, en específico la versión 3.6.8. Se utilizó en la implementación la versión de 64 bit para el sistema operativo Microsoft Windows 10, aunque puede utilizarse la versión 32 bit también. Existen instaladores de Python para Mac OS y Linux.

FER REST API necesita tener instaladas las dependencias:

- NumPy versión 1.16.0.
- OpenCV con sus módulos principales y extras versión 4.0.0.21.
- Dlib versión 19.8.1.

- Sickit-learn versión 0.20.2.
- Framework FastAPI versión 0.7.1.
- El servidor Uvicorn versión 0.5.2.
- PyQt5 versión 5.11.3.

FER REST API posee un archivo *requirements.txt* en el cual se encuentran todas las dependencias necesarias para instalar. El comando `pip` permite gestionar listas de paquetes a través de un archivo de requisitos con los números de versión correspondientes. Y mediante el comando siguiente son instaladas los paquetes de manera automática:

```
pip install -r requirements.txt
```

Para instalar FER REST API, se copia su código fuente para cualquier ubicación de la computadora en la que se implemente la aplicación que va a hacer uso de la misma. Luego el programador podrá ejecutar la API y consumir sus métodos.

FER REST API también permite su instalación desde el intérprete de Python, para ello se necesita estar ubicado en el directorio de la API y ejecutar el siguiente comando en un terminal:

```
python3 setup.py install
```

De esta forma se permite utilizar la API desde el código fuente de Python como una dependencia instalada en el intérprete para aplicaciones en ese lenguaje de programación. Luego se puede importar el paquete *fer_api* en el código fuente del proyecto.

### 2.1.2 **Documentación de la API**

FER REST API posee una documentación interactiva brindada por el framework FastAPI mediante la dirección URL http://localhost:8000/docs basada en la herramienta Swagger UI. Permite al desarrollador, visualizar e interactuar con los recursos de la API sin tener implementada la lógica de su aplicación (véase Anexo 35). Además, la API posee otra documentación (véase Anexo 36) mediante la dirección http://localhost:8000/redoc basada en la herramienta ReDoc. Ambas herramientas pertenecen a la especificación de la Fundación Linux OpenAPI.

La API posee un archivo llamado *README.md* (véase Anexo 41) que representa una documentación para los desarrolladores desde el código fuente. Este permite conocer brevemente

el funcionamiento, características y aspectos principales de FER REST API. Puede leerse con diversos editores.

La API está especificada mediante el estándar OpenAPI versión 3 en un archivo de tipo JSON (véase Anexo 37). Este representa una documentación muy completa que le brinda al desarrollador la forma de consumir la API REST e información sobre la misma. Además, brinda una descripción de sus recursos, cual es el formato de respuesta, entre otras características. Esta especificación OpenAPI puede ser importada mediante herramientas como Postman que hace función de cliente y permite cargar el estándar en su versión 2 y 3. Esto facilita al desarrollador una vía de probar y consumir los recursos de la API previa a una inclusión en el código de aplicaciones. Además, existen herramientas como Swagger que a través de este estándar de OpenAPI, permite generar código de la parte del cliente para consumir los recursos de la API e incluirlo en varios lenguajes de programación como C#, C++, Go, Java, Kotlin, Node.js, Objective-C, Perl, PHP, Python, Ruby, Scala, Swift, Typescript entre otros.

FER REST API cuenta con un manual de usuario de muy fácil entendimiento para interactuar con la API en los formatos: eBook de Windows (.exe), pdf y en un archivo de ayuda de HTML compilado (.chm). Fue realizado por la herramienta Help & Manual versión 7.5.0 (EC Software GmbH, 2019). Esta herramienta es líder para escribir y publicar documentación en varios formatos como Ayuda HTML, basada en el navegador de ayuda, Adobe PDF, Winhelp, Help 2.0, ayuda Visual Studio, Microsoft Word, eBook de ejecutable de Windows, ePUB y ebook Amazon Kindle entre otros. Es un editor WYSIWYG XML que no requiere conocimientos de XML o técnicas de codificación.

### 2.1.3 **Ejecución**

Para la ejecución de FER REST API se debe hacer desde la terminal de sistemas operativos como Windows accediendo a la línea de comandos cmd (Símbolos del Sistema), Linux o Mac con el siguiente comando:

```
uvicorn api:fer_rest_api
```

Este comando se ejecuta al estar ubicado dentro de la carpeta de FER REST API específicamente dentro del directorio “fer_api”. El comando, inicia el servidor ASGI Uvicorn del framework FastAPI donde se ejecuta la API (véase Figura 2.12).

```
C:\Users\Betty\PycharmProjects\FER REST API\fer_api>uvicorn api:fer_rest_api
INFO: Started server process [668]
INFO: Waiting for application startup.
INFO: Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
```

**Figura 2.12 Ejecución de FER REST API desde la terminal del IDE PyCharm en el sistema operativo Microsoft Windows 10**

## *2.2 Conclusiones parciales*

En este capítulo se definieron los elementos principales en el proceso de diseño e implementación de Facial Emotions Recognition REST API. El Visual Paradigm es una herramienta muy completa, potente y útil para la modelación de diagramas UML. PyCharm como IDE para el lenguaje de programación Python es muy útil por las características y ventajas que ofrece al programador.

La arquitectura FER REST API permite la independencia de tecnologías y lenguajes de programación para la comunicación con las distintas aplicaciones.

Entre los diagramas UML más relativos a las API REST, el de recursos permite ver las funcionalidades y respuestas de la API. El de componentes permite plasmar las dependencias entre los componentes de software. El de despliegue muestra las relaciones físicas entre los componentes. El de implementación expresa la composición física de la implementación del sistema.

El framework FastAPI es ligero, minimalista de código, intuitivo y muy rápido. Se obtiene una API REST potente y de calidad implementada en el lenguaje de programación Python.

Las bibliotecas OpenCV, Dlib, Scikit-learn y NumPy resultan útiles para el procesamiento de imágenes, la detección de rostros, la extracción de características y la clasificación de emociones. Los clasificadores utilizados se ajustan a los conjuntos de datos para el entrenamiento.

FER REST API permite que las emociones a detectar y la forma de presentarlas varíen en dependencia de la aplicación, con la posibilidad de crear clasificadores personalizados en dependencia del conjunto de datos de entrenamiento.

FER REST API es asequible de instalar, de fácil ejecución y manejo. Posee una documentación buena y extensa para el desarrollador de software. Está implementada con el fin de que los programadores que no tengan dominio específico sobre el tema la puedan utilizar fácilmente.

# 3

# ANÁLISIS DE LOS RESULTADOS OBTENIDOS CON FER REST API

# 3. ANÁLISIS DE LOS RESULTADOS OBTENIDOS CON FER REST API

En el presente capítulo se realizan pruebas y analizan los clasificadores de aprendizaje automático con la herramienta Weka. Para las pruebas se utilizan las bases de datos de emociones Cohn Kanade y KDEF que contienen las seis emociones universales. Se comprueba el funcionamiento de FER REST API consumiendo sus recursos desde la herramienta Postman como cliente.

## *3.1 Evaluación de clasificadores de aprendizaje automático con la base de datos de emociones Cohn Kanade*

Para realizar las pruebas se utilizan los clasificadores implementados en la herramienta Weka 3.9.3. Se extrajeron los vectores de características de todas las imágenes del conjunto de datos Cohn Kanade (Lucey *et al.*, 2010) y se guardan en un archivo ARFF (Attribute Relation File Format). Este archivo es cargado por Weka para realizar las pruebas. Desde Weka son ajustados los parámetros de los clasificadores, para optimizar los algoritmos implementados con Scikit-learn en la implementación de la API.

La base de datos Cohn Kanade se escogió por ser una de las más variada en cuanto a la raza y género de sus participantes y ser más utilizada (véase Anexo 38). Sus imágenes no poseen tanta calidad y no está equilibrada la cantidad de imágenes por cada emoción, ni el género, las razas y edades de los participantes. Todas las imágenes se encuentran en formato PNG con una calidad no tan alta y poseen cambios en la luminosidad y colores. Estas condiciones son ideales para probar el desempeño de los algoritmos de procesamiento de imágenes, detección de rostros, extracción de rasgos y de clasificación de la API.

En la Figura 3.1 se ilustra la distribución de la cantidad de imágenes por emociones del conjunto de datos, para un total de 445 imágenes en ocho emociones, las seis universales, la neutral más la de desprecio.

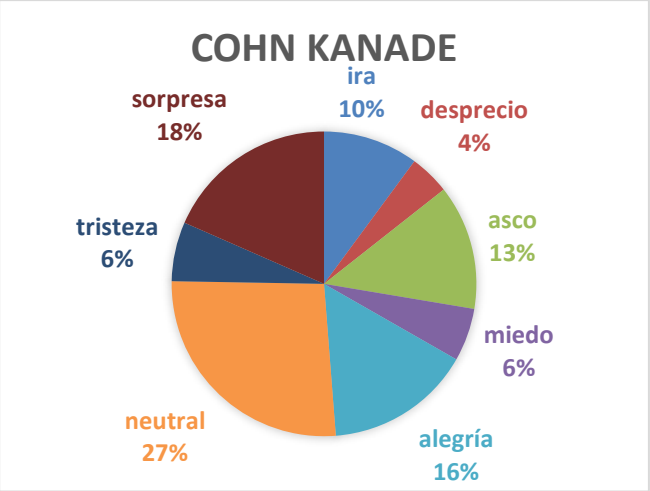



**Figura 3.1 Composición del conjunto de datos Cohn Kanade según cantidad de imágenes por emociones**

### Validación cruzada de K iteraciones

Para cada clasificador, en Weka se realiza una validación cruzada de diez iteraciones pues esto es lo recomendado en la literatura. La validación cruzada (cross-validation) (Refaeilzadeh, Tang and Liu, 2008; Elkan, 2011) es una técnica utilizada para evaluar los resultados de un análisis estadístico. Además, garantiza la independencia de la partición entre datos de entrenamiento y prueba (véase Anexo 39). Se utiliza en entornos donde el objetivo principal es la predicción y se quiere estimar la precisión de un modelo que se llevará a cabo a la práctica.

### Matriz de confusión

Para comprobar si el clasificador proporciona buenos resultados se emplea la matriz de confusión comúnmente llamada *tabla de contingencia*. La matriz de confusión permite visualizar el funcionamiento del clasificador, resumiendo la concordancia entre la clase real y clase pronosticada. Las columnas representan el número de predicciones de cada clase y las filas los elementos de la clase real. Estas matrices tienen la estructura como se muestra en la Tabla 3.1.

**Tabla 3.1 Estructura de la matriz de confusión**

| | | **Predicción** | |
|---|---|---|---|
| | | **Positivos** | **Negativos** |
| **Observación** | **Positivos** | Verdaderos Positivos (VP) | Falsos Negativos (FN) |

| | | | |
|---|---|---|---|
| | **Negativos** | Falsos Positivos (FP) | Verdaderos Negativos (VN) |

Los Verdaderos Positivos (VP) es la cantidad de positivos que fueron clasificados correctamente por el modelo. Los Verdaderos Negativos (VN) es la cantidad de negativos que fueron clasificados correctamente. Los Falsos Negativos (FN) es la cantidad de positivos que fueron clasificados incorrectamente. Los Falsos Positivos (FP) es la cantidad de negativos que fueron clasificados incorrectamente.

A partir de la matriz de confusión, existen diversos conceptos que ayudan a valorar la efectividad del clasificador.

### 3.1.1 Máquina de Soporte Vectorial (SVM) con Kernel Polinomial

Se entrenó y evalúo el clasificador SVM con kernel polinomial de grado 1,6, con penalización del error igual 1, coeficiente del kernel 1/68 y tolerancia para el criterio de parada 0,0001.

Se clasificaron correctamente 369 (82,92 %) imágenes e incorrectamente 76 (17,08 %), con un coeficiente de Kappa de 0,79.

En la Tabla 3.2 se muestran los resultados por cada clase (emoción) para el clasificador con los diferentes parámetros que miden su efectividad. La clase alegría, sorpresa, asco, miedo, neutral e ira fueron las de mejores resultados, entre ellas, la mejor fue alegría. Las de peores resultados fueron tristeza y desprecio, entre ellas, la peor fue tristeza.

**Tabla 3.2 Estadísticas por clase de SVM con kernel polinomial utilizando Cohn-Kanade**

| Razón de verdaderos positivos | Razón de falsos positivos | Precisión | Medida-F | Clase |
|---|---|---|---|---|
| 0,62 | 0,02 | 0,74 | 0,67 | ira |
| 0,50 | 0,02 | 0,56 | 0,53 | desprecio |
| 0,88 | 0,02 | 0,90 | 0,89 | asco |
| 0,88 | 0,02 | 0,76 | 0,81 | miedo |
| 0,99 | 0,00 | 0,99 | 0,99 | alegría |
| 0,87 | 0,09 | 0,77 | 0,82 | neutral |

| | | | | |
|---|---|---|---|---|
| 0,89 | 0,01 | 0,96 | 0,92 | sorpresa |
| 0,46 | 0,03 | 0,54 | 0,50 | tristeza |
| **0,83** | **0,03** | **0,83** | **0,83** | **Promedio** |

La Tabla 3.3 muestra la matriz de confusión para este clasificador. Podemos ver la clasificación de las imágenes por clases. Se muestran los resultados satisfactorios de las clases alegría, asco, sorpresa, miedo, neutral e ira. Así como, los malos resultados obtenidos en las clases desprecio y tristeza que fueron, principalmente, a imágenes mal clasificadas como neutral.

**Tabla 3.3 Matriz de confusión de SVM con kernel polinomial utilizando Cohn-Kanade**

| ira | desprecio | asco | miedo | alegría | neutral | sorpresa | tristeza | ←Clasificado como |
|---|---|---|---|---|---|---|---|---|
| **28** | 0 | 3 | 0 | 1 | 7 | 0 | 6 | **ira** |
| 1 | **9** | 0 | 0 | 0 | 7 | 0 | 1 | **desprecio** |
| 2 | 0 | **52** | 0 | 0 | 5 | 0 | 0 | **asco** |
| 0 | 0 | 0 | **22** | 0 | 0 | 2 | 1 | **miedo** |
| 1 | 0 | 0 | 0 | **68** | 0 | 0 | 0 | **alegría** |
| 4 | 3 | 3 | 1 | 0 | **103** | 1 | 3 | **neutral** |
| 0 | 1 | 0 | 5 | 0 | 3 | **74** | 0 | **sorpresa** |
| 2 | 3 | 0 | 1 | 0 | 9 | 0 | **13** | **tristeza** |

### 3.1.2 Evaluación de Máquina de Soporte Vectorial (SVM) con Kernel Lineal

Se entrenó y evalúo el clasificador Máquina de Soporte Vectorial (SVM) con kernel lineal, penalización del error 1, coeficiente del kernel 1/68 y tolerancia para el criterio de parada 0,0001.

Se clasificaron correctamente 361 (81,12 %) imágenes e incorrectamente 84 (18,88 %), con un coeficiente de Kappa de 0,77.

En la Tabla 3.4 se muestran los resultados por cada clase (emoción) para el clasificador con los diferentes parámetros que miden su efectividad. La clase alegría, sorpresa, asco, miedo y neutral fueron las de mejores resultados, entre ellas, la de mejor resultado fue alegría. Las de peores resultados fueron tristeza, ira y desprecio, entre ellas, la peor fue desprecio.

**Tabla 3.4 Estadísticas por clase de SVM con kernel lineal utilizando Cohn-Kanade**

| Razón de verdaderos positivos | Razón de falsos positivos | Precisión | Medida-F | Clase |
|---|---|---|---|---|
| 0,58 | 0,02 | 0,74 | 0,65 | ira |
| 0,11 | 0,00 | 1,00 | 0,20 | desprecio |
| 0,85 | 0,01 | 0,91 | 0,88 | asco |
| 0,80 | 0,01 | 0,87 | 0,83 | miedo |
| 0,99 | 0,00 | 0,97 | 0,98 | alegría |
| 0,95 | 0,19 | 0,64 | 0,77 | neutral |
| 0,94 | 0,01 | 0,96 | 0,95 | sorpresa |
| 0,18 | 0,00 | 1,00 | 0,30 | tristeza |
| **0,81** | **0,06** | **0,85** | **0,79** | **Promedio** |

La Tabla 3.5 muestra la matriz de confusión para este clasificador. Se muestran los resultados satisfactorios de las clases alegría, asco, sorpresa, miedo y neutral. Además, los malos resultados obtenidos en las clases ira, desprecio y tristeza que fueron, principalmente, a imágenes mal clasificadas como neutral.

**Tabla 3.5 Matriz de confusión de SVM con kernel lineal utilizando Cohn-Kanade**

| ira | desprecio | asco | miedo | alegría | neutral | sorpresa | tristeza | ←Clasificado como |
|---|---|---|---|---|---|---|---|---|
| **26** | 0 | 3 | 0 | 0 | 16 | 0 | 0 | **ira** |
| 0 | **2** | 0 | 0 | 0 | 16 | 0 | 0 | **desprecio** |
| 4 | 0 | **50** | 1 | 0 | 4 | 0 | 0 | **asco** |
| 0 | 0 | 0 | **20** | 2 | 0 | 3 | 0 | **miedo** |
| 0 | 0 | 0 | 0 | **68** | 1 | 0 | 0 | **alegría** |
| 3 | 0 | 2 | 1 | 0 | **112** | 0 | 0 | **neutral** |
| 0 | 0 | 0 | 1 | 0 | 4 | **78** | 0 | **sorpresa** |
| 2 | 0 | 0 | 0 | 0 | 21 | 0 | **5** | **tristeza** |

### 3.1.3 Evaluación de Red Neuronal Perceptrón Multicapa (MLP)

Se entrenó y evalúo el clasificador Red Neuronal Perceptrón Multicapa (MLP) con 65 neuronas y una capa oculta, con la función de activación para las capas ocultas relu. Con una planificación de la tasa de aprendizaje para la actualización de pesos de 0,25 y el momento (momentum) aplicado a la actualización de los pesos con valor 0,2.

Se clasificaron correctamente 379 (85,17 %) imágenes e incorrectamente 66 (14,83 %), con un coeficiente de Kappa de 0,82.

En la Tabla 3.6 se muestran los resultados por cada clase (emoción) para el clasificador con los diferentes parámetros que miden su efectividad. La clase alegría, sorpresa, asco, miedo, ira y neutral fueron las de mejores resultados, entre ellas, la mejor fue alegría. Las de peores resultados fueron tristeza y desprecio, entre ellas, la peor fue tristeza.

**Tabla 3.6 Estadísticas por clase de MLP utilizando Cohn-Kanade**

| Razón de verdaderos positivos | Razón de falsos positivos | Precisión | Medida-F | Clase |
|---|---|---|---|---|
| 0,73 | 0,03 | 0,72 | 0,72 | ira |
| 0,50 | 0,01 | 0,69 | 0,58 | desprecio |
| 0,91 | 0,01 | 0,93 | 0,92 | asco |
| 0,88 | 0,00 | 0,92 | 0,90 | miedo |
| 0,99 | 0,00 | 0,99 | 0,99 | alegría |
| 0,86 | 0,10 | 0,76 | 0,80 | neutral |
| 0,95 | 0,01 | 0,96 | 0,96 | sorpresa |
| 0,46 | 0,02 | 0,65 | 0,54 | tristeza |
| **0,85** | **0,03** | **0,85** | **0,85** | **Promedio** |

La Tabla 3.7 muestra la matriz de confusión para este clasificador. Se muestran los resultados satisfactorios de las clases alegría, ira, asco, sorpresa, miedo y neutral. Además, los malos resultados obtenidos en las clases desprecio y tristeza que fueron, principalmente, a imágenes mal clasificadas como neutral.

**Tabla 3.7 Matriz de confusión de MLP utilizando Cohn-Kanade**

| ira | desprecio | asco | miedo | alegría | neutral | sorpresa | tristeza | ←Clasificado como |
|---|---|---|---|---|---|---|---|---|
| **33** | 0 | 2 | 0 | 0 | 8 | 0 | 2 | **ira** |
| 0 | **9** | 0 | 0 | 0 | 8 | 0 | 1 | **desprecio** |
| 3 | 0 | **54** | 0 | 0 | 2 | 0 | 0 | **asco** |
| 0 | 1 | 0 | **22** | 1 | 0 | 1 | 0 | **miedo** |
| 1 | 0 | 0 | 0 | **68** | 0 | 0 | 0 | **alegría** |
| 7 | 3 | 2 | 0 | 0 | **101** | 1 | 4 | **neutral** |
| 0 | 0 | 0 | 1 | 0 | 3 | **79** | 0 | **sorpresa** |
| 2 | 0 | 0 | 1 | 0 | 11 | 1 | **13** | **tristeza** |

### 3.1.4 Evaluación de K Vecinos más Cercanos (KNN)

Se entrenó y evalúo el clasificador K Vecinos más Cercanos (KNN) con un valor de K igual a 10, con el algoritmo de fuerza bruta y función de distancia la Euclidiana.

Se clasificaron correctamente 297 (66,74 %) imágenes e incorrectamente 148 (33,26 %), con un coeficiente de Kappa de 0,58.

En la Tabla 3.8 se muestran los resultados por cada clase (emoción) para el clasificador con los diferentes parámetros que miden su efectividad. La clase alegría, sorpresa, asco y neutral fueron las de mejores resultados, entre ellas, la mejor fue alegría. Las de peores resultados fueron tristeza, ira, miedo y desprecio, entre ellas, las peores fueron desprecio y tristeza.

**Tabla 3.8 Estadísticas por clase KNN utilizando Cohn-Kanade**

| Razón de verdaderos positivos | Razón de falsos positivos | Precisión | Medida-F | Clase |
|---|---|---|---|---|
| 0,18 | 0,04 | 0,35 | 0,23 | ira |
| 0,00 | 0,00 | 0,00 | 0,00 | desprecio |
| 0,63 | 0,02 | 0,84 | 0,72 | asco |
| 0,20 | 0,01 | 0,56 | 0,29 | miedo |
| 0,94 | 0,03 | 0,84 | 0,89 | alegría |
| 0,91 | 0,32 | 0,51 | 0,65 | neutral |

| | | | | |
|---|---|---|---|---|
| 0,89 | 0,00 | 0,99 | 0,94 | sorpresa |
| 0,00 | 0,00 | 0,00 | 0,00 | tristeza |
| **0,67** | **0,10** | **0,63** | **0,62** | **Promedio** |

La Tabla 3.9 muestra la matriz de confusión para este clasificador. Se muestran los resultados satisfactorios de las clases alegría, sorpresa, asco y neutral. Además, los malos resultados obtenidos en las clases ira, desprecio y tristeza que fueron, principalmente, a imágenes mal clasificadas como neutral.

**Tabla 3.9 Matriz de confusión de KNN utilizando Cohn-Kanade**

| ira | desprecio | asco | miedo | alegría | neutral | sorpresa | tristeza | ←Clasificado como |
|---|---|---|---|---|---|---|---|---|
| **8** | 0 | 3 | 0 | 1 | 32 | 0 | 1 | **ira** |
| 2 | **0** | 1 | 0 | 0 | 15 | 0 | 0 | **desprecio** |
| 2 | 0 | **37** | 0 | 2 | 18 | 0 | 0 | **asco** |
| 0 | 0 | 0 | **5** | 9 | 10 | 1 | 0 | **miedo** |
| 1 | 1 | 0 | 1 | **65** | 1 | 0 | 0 | **alegría** |
| 5 | 1 | 3 | 0 | 0 | **108** | 0 | 1 | **neutral** |
| 1 | 0 | 0 | 2 | 0 | 6 | **74** | 0 | **sorpresa** |
| 4 | 0 | 0 | 1 | 0 | 23 | 0 | **0** | **tristeza** |

### 3.1.5 Evaluación de Naive Bayes

Se entrenó y evalúo el clasificador Naive Bayes.

Se clasificaron correctamente 307 (68,99 %) imágenes e incorrectamente 138 (31,01 %), con un coeficiente de Kappa de 0,63.

En la Tabla 3.10 se muestran los resultados por cada clase (emoción) para el clasificador con los diferentes parámetros que miden su efectividad. La clase alegría, sorpresa, asco y neutral fueron las de mejores resultados, entre ellas, la mejor fue alegría. Las de peores resultados fueron tristeza, ira, miedo y desprecio, entre ellas, la peor fue tristeza.

**Tabla 3.10 Estadísticas por clase de Naive Bayes utilizando Cohn-Kanade**

| Razón de verdaderos positivos | Razón de falsos positivos | Precisión | Medida-F | Clase |
|---|---|---|---|---|
| 0,42 | 0,04 | 0,53 | 0,47 | ira |
| 0,33 | 0,05 | 0,21 | 0,25 | desprecio |
| 0,81 | 0,04 | 0,75 | 0,78 | asco |
| 0,56 | 0,03 | 0,56 | 0,56 | miedo |
| 0,90 | 0,01 | 0,92 | 0,91 | alegría |
| 0,64 | 0,12 | 0,66 | 0,65 | neutral |
| 0,95 | 0,03 | 0,89 | 0,92 | sorpresa |
| 0,14 | 0,04 | 0,18 | 0,16 | tristeza |
| **0,69** | **0,05** | **0,69** | **0,69** | **Promedio** |

La Tabla 3.11 muestra la matriz de confusión para este clasificador. Se muestran los resultados satisfactorios de las clases alegría, sorpresa, asco y neutral. Además, los malos resultados obtenidos en las clases ira, miedo, desprecio y tristeza que fueron debido, principalmente, a imágenes mal clasificadas como neutral y de la clase miedo con alegría.

**Tabla 3.11 Matriz de confusión de Naive Bayes utilizando Cohn-Kanade**

| ira | desprecio | asco | miedo | alegría | neutral | sorpresa | tristeza | ←Clasificado como |
|---|---|---|---|---|---|---|---|---|
| **19** | 5 | 7 | 0 | 0 | 8 | 1 | 5 | **ira** |
| 0 | **6** | 0 | 1 | 0 | 8 | 0 | 3 | **desprecio** |
| 5 | 1 | **48** | 1 | 1 | 3 | 0 | 0 | **asco** |
| 0 | 1 | 1 | **14** | 4 | 1 | 3 | 1 | **miedo** |
| 0 | 0 | 3 | 3 | **62** | 1 | 0 | 0 | **alegría** |
| 9 | 13 | 4 | 4 | 0 | **75** | 4 | 9 | **neutral** |
| 0 | 0 | 0 | 1 | 0 | 3 | **79** | 0 | **sorpresa** |
| 3 | 3 | 1 | 1 | 0 | 14 | 2 | **4** | **tristeza** |

### 3.1.6 Evaluación de Árbol de Decisión (Decision Tree)

Se entrenó y evalúo el clasificador Árbol de Decisión implementado el J48 basado en el modelo C4.5. Es utilizado el criterio de entropía para la división, con 5 nodos como número mínimo de hojas y el factor de confianza destinado para la poda con valor 0,05.

Se clasificaron correctamente 299 (67,19 %) imágenes e incorrectamente 146 (32,81 %), con un coeficiente de Kappa de 0,60.

En la Tabla 3.12 se muestran los resultados por cada clase (emoción) para el clasificador con los diferentes parámetros que miden su efectividad. La clase alegría, sorpresa, asco y neutral fueron las de mejores resultados, entre ellas, la mejor fue sorpresa. Las de peores resultados fueron tristeza, ira, miedo y desprecio, entre ellas, las peores desprecio y tristeza.

**Tabla 3.12 Estadísticas por clase de Árbol de Decisión utilizando Cohn-Kanade**

| Razón de verdaderos positivos | Razón de falsos positivos | Precisión | Medida-F | Clase |
|---|---|---|---|---|
| 0,29 | 0,05 | 0,38 | 0,33 | ira |
| 0,17 | 0,03 | 0,21 | 0,19 | desprecio |
| 0,76 | 0,04 | 0,75 | 0,76 | asco |
| 0,56 | 0,06 | 0,35 | 0,43 | miedo |
| 0,84 | 0,02 | 0,88 | 0,86 | alegría |
| 0,71 | 0,15 | 0,63 | 0,67 | neutral |
| 0,93 | 0,01 | 0,94 | 0,93 | sorpresa |
| 0,18 | 0,03 | 0,31 | 0,23 | tristeza |
| **0,67** | **0,06** | **0,66** | **0,66** | **Promedio** |

La Tabla 3.13 muestra la matriz de confusión para este clasificador. Se muestran los resultados satisfactorios de las clases alegría, sorpresa, asco y neutral. Además, los malos resultados obtenidos en las clases ira, miedo, desprecio y tristeza que fueron, principalmente, a imágenes mal clasificadas como neutral, y de la clase miedo con alegría.

**Tabla 3.13 Matriz de confusión de Árbol de Decisión utilizando Cohn-Kanade**

| ira | desprecio | asco | miedo | alegría | neutral | sorpresa | tristeza | ←Clasificado como |
|---|---|---|---|---|---|---|---|---|
| **13** | 1 | 6 | 4 | 0 | 18 | 0 | 3 | **ira** |
| 0 | **3** | 2 | 0 | 2 | 9 | 0 | 2 | **desprecio** |
| 4 | 2 | **45** | 3 | 1 | 4 | 0 | 0 | **asco** |
| 1 | 0 | 1 | **14** | 4 | 2 | 3 | 0 | **miedo** |
| 1 | 0 | 0 | 9 | **58** | 1 | 0 | 0 | **alegría** |
| 10 | 6 | 5 | 6 | 1 | **84** | 1 | 5 | **neutral** |
| 1 | 0 | 0 | 3 | 0 | 1 | **77** | 1 | **sorpresa** |
| 4 | 2 | 1 | 1 | 0 | 14 | 1 | **5** | **tristeza** |

### 3.1.7 Evaluación de Bosque Aleatorio (Random Forest)

Se entrenó y evalúo el clasificador Bosque Aleatorio con 30 árboles de decisión. Es utilizado el criterio de entropía como la función para medir la calidad de la división. La profundidad de cada árbol es ilimitada.

Se clasificaron correctamente 316 (71,01 %) imágenes e incorrectamente 129 (28,99 %), con un coeficiente de Kappa de 0,64.

En la Tabla 3.14 se muestran los resultados por cada clase (emoción) para el clasificador con los diferentes parámetros que miden su efectividad. La clase alegría, sorpresa, asco y neutral fueron las de mejores resultados, entre ellas, las mejores fueron sorpresa y alegría. Las de peores resultados fueron tristeza, ira, miedo y desprecio, entre ellas, las peores fueron desprecio y tristeza.

**Tabla 3.14 Estadísticas por clase de Bosque Aleatorio utilizando Cohn-Kanade**

| Razón de verdaderos positivos | Razón de falsos positivos | Precisión | Medida-F | Clase |
|---|---|---|---|---|
| 0,24 | 0,02 | 0,61 | 0,35 | ira |
| 0,11 | 0,00 | 0,67 | 0,19 | desprecio |
| 0,75 | 0,03 | 0,80 | 0,77 | asco |
| 0,40 | 0,02 | 0,59 | 0,48 | miedo |
| 0,94 | 0,03 | 0,87 | 0,90 | alegría |
| 0,88 | 0,26 | 0,55 | 0,68 | neutral |

| | | | | |
|---|---|---|---|---|
| 0.94 | 0,02 | 0,92 | 0,93 | sorpresa |
| 0,07 | 0,00 | 0,67 | 0,13 | tristeza |
| **0,71** | **0,08** | **0,72** | **0,67** | **Promedio** |

La Tabla 3.15 muestra la matriz de confusión para este clasificador. Se muestran los resultados satisfactorios de las clases alegría, sorpresa, asco y neutral. Además, los malos resultados obtenidos en las clases ira, miedo, desprecio y tristeza que fueron, principalmente, a imágenes mal clasificadas como neutral.

**Tabla 3.15 Matriz de confusión de Bosque Aleatorio utilizando Cohn-Kanade**

| ira | desprecio | asco | miedo | alegría | neutral | sorpresa | tristeza | ←Clasificado como |
|---|---|---|---|---|---|---|---|---|
| **11** | 0 | 5 | 0 | 0 | 29 | 0 | 0 | **ira** |
| 0 | **2** | 0 | 0 | 0 | 16 | 0 | 0 | **desprecio** |
| 1 | 0 | **44** | 1 | 4 | 8 | 1 | 0 | **asco** |
| 0 | 0 | 1 | **10** | 6 | 5 | 3 | 0 | **miedo** |
| 0 | 0 | 1 | 2 | **65** | 1 | 0 | 0 | **alegría** |
| 5 | 0 | 3 | 4 | 0 | **104** | 1 | 1 | **neutral** |
| 0 | 0 | 0 | 0 | 0 | 5 | **78** | 0 | **sorpresa** |
| 1 | 1 | 1 | 0 | 0 | 21 | 2 | **2** | **tristeza** |

Es importante señalar que, tal como determinó Ekman and Friesen (1975) ni siquiera el ser humano es capaz de clasificar las emociones básicas a partir de expresiones faciales con un 100 % de precisión.

## *3.2 Evaluación de clasificadores de aprendizaje automático con la base de datos de emociones KDEF*

Para hacer la evaluación de los clasificadores utilizando la base de datos de emociones KDEF, se extrajeron todos los vectores de características de las imágenes y fueron almacenados en un fichero ARFF y cargado por Weka.

La base de datos KDEF (Department of Clinical Neuroscience Psychology Karolinska Institutet, 2019) se escogió por ser utilizada en una gran cantidad de trabajos. Además, los participantes son de la misma raza y nacionalidad (véase Anexo 40). Sus imágenes tienen calidad y se encuentran

bajo la misma iluminación y en un mismo color todas. La edad de los participantes es menor con respecto a los participantes de Cohn Kanade, su media es de 25 años de edad. Todas las imágenes se encuentran en formato JPEG. Se encuentra equilibrada la cantidad de imágenes por emoción y la cantidad de personas por género. Estas características son ideales para probar el desempeño de los algoritmos de procesamiento de imágenes, detección de rostros, extracción de rasgos y de clasificación de la API REST.

Mediante las pruebas se realiza una comparación entre los resultados de los clasificadores en ambos conjuntos de datos Cohn Kanade y KDEF. Ambos poseen características de composición diferentes, pero similitud en las emociones que pueden detectar.

En la Figura 3.2 se ilustra la distribución de la cantidad de imágenes por emociones del conjunto de datos, para un total de 490 imágenes en siete emociones, las seis universalmente y la neutral.

Se realiza para cada clasificador una validación cruzada de diez vueltas. Los clasificadores utilizados son los mismos y con la misma configuración de parámetros que se utilizaron en el conjunto de datos Cohn Kanade.

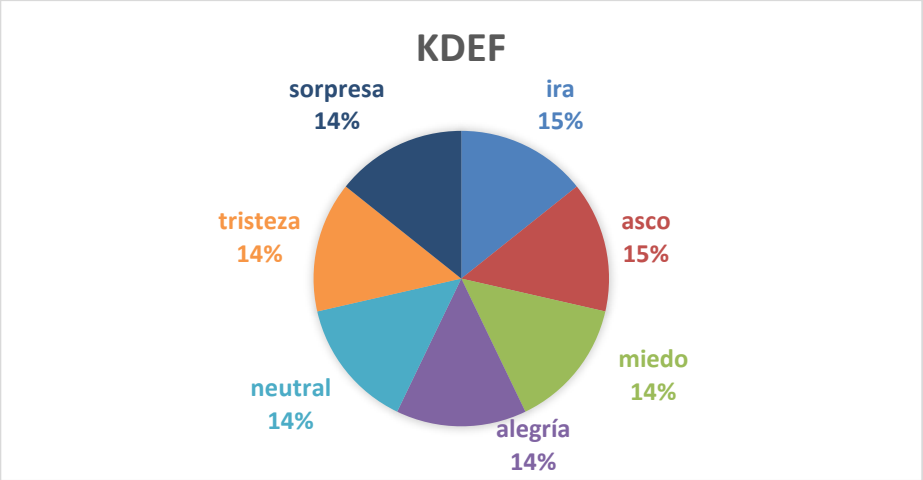



**Figura 3.2 Composición del conjunto de datos KDEF según cantidad de imágenes por emociones**

### 3.2.1 Evaluación de Máquina de Soporte Vectorial (SVM) con Kernel Polinomial

Se clasificaron correctamente 378 (77,14 %) imágenes e incorrectamente 112 (22,86 %), con un coeficiente de Kappa de 0,73.

En la Tabla 3.16 se muestran los resultados por cada clase (emoción) para el clasificador con los diferentes parámetros que miden su efectividad. La clase alegría, sorpresa, asco, ira, neutral fueron

las de mejores resultados, entre ellas, las mejores fueron alegría y sorpresa. Las de peores resultados fueron tristeza y miedo, entre ellas, la peor fue tristeza.

**Tabla 3.16 Estadísticas por clase de SVM con kernel polinomial utilizando KDEF**

| Razón de verdaderos positivos | Razón de falsos positivos | Precisión | Medida-F | Clase |
|---|---|---|---|---|
| 0,59 | 0,06 | 0,61 | 0,60 | miedo |
| 0,86 | 0,04 | 0,77 | 0,81 | ira |
| 0,80 | 0,02 | 0,87 | 0,84 | asco |
| 0,89 | 0,01 | 0,92 | 0,90 | alegría |
| 0,83 | 0,06 | 0,70 | 0,76 | neutral |
| 0,57 | 0,04 | 0,71 | 0,63 | tristeza |
| 0.87 | 0,03 | 0,81 | 0,84 | sorpresa |
| **0.77** | **0,04** | **0,77** | **0,77** | **Promedio** |

La Tabla 3.17 muestra la matriz de confusión para este clasificador. Se muestran los resultados satisfactorios de las clases alegría, sorpresa, asco, ira y neutral. Además, los malos resultados obtenidos en las clases miedo y tristeza que fueron, principalmente, a imágenes mal clasificadas de la clase miedo con sorpresa y tristeza, y de tristeza con neutral y miedo.

**Tabla 3.17 Matriz de confusión de SVM con kernel polinomial utilizando KDEF**

| miedo | ira | asco | alegría | neutral | tristeza | sorpresa | ←Clasificado como |
|---|---|---|---|---|---|---|---|
| **41** | 2 | 3 | 3 | 4 | 5 | 12 | **miedo** |
| 2 | **60** | 2 | 1 | 4 | 1 | 0 | **ira** |
| 2 | 8 | **56** | 0 | 0 | 4 | 0 | **asco** |
| 5 | 1 | 0 | **62** | 2 | 0 | 0 | **alegría** |
| 2 | 2 | 0 | 1 | **58** | 6 | 1 | **neutral** |
| 8 | 5 | 3 | 0 | 13 | **40** | 1 | **tristeza** |
| 7 | 0 | 0 | 0 | 2 | 0 | **61** | **sorpresa** |

### 3.2.2 Evaluación de Máquina de Soporte Vectorial (SVM) con Kernel Lineal

Se clasificaron correctamente 372 (75,92 %) imágenes e incorrectamente 118 (24,08 %), con un coeficiente de Kappa de 0,72.

En la Tabla 3.18 se muestran los resultados por cada clase (emoción) para el clasificador con los diferentes parámetros que miden su efectividad. La clase alegría, sorpresa, asco, ira, neutral fueron las de mejores resultados, entre ellas, la mejor fue alegría. Las de peores resultados fueron tristeza y miedo.

**Tabla 3.18 Estadísticas por clase de SVM con kernel lineal utilizando KDEF**

| Razón de verdaderos positivos | Razón de falsos positivos | Precisión | Medida-F | Clase |
|---|---|---|---|---|
| 0,63 | 0,05 | 0,66 | 0,64 | miedo |
| 0,76 | 0,04 | 0,75 | 0,75 | ira |
| 0,80 | 0,02 | 0,85 | 0,82 | asco |
| 0,93 | 0,01 | 0,91 | 0,92 | alegría |
| 0,76 | 0,06 | 0,68 | 0,72 | neutral |
| 0,63 | 0,06 | 0,64 | 0,63 | tristeza |
| 0,81 | 0,03 | 0,84 | 0,83 | sorpresa |
| **0,76** | **0,04** | **0,76** | **0,76** | **Promedio** |

La Tabla 3.19 muestra la matriz de confusión para este clasificador. Se muestran los resultados satisfactorios de las clases alegría, sorpresa, asco, ira y neutral. Además, los malos resultados obtenidos en las clases miedo y tristeza que fueron, principalmente, a imágenes mal clasificadas de la clase miedo con sorpresa y alegría, y tristeza con neutral e ira.

**Tabla 3.19 Matriz de confusión de SVM con kernel lineal utilizando KDEF**

| miedo | ira | asco | alegría | neutral | tristeza | sorpresa | ←Clasificado como |
|---|---|---|---|---|---|---|---|
| **44** | 2 | 2 | 5 | 4 | 4 | 9 | **miedo** |
| 3 | **53** | 5 | 0 | 5 | 4 | 0 | **ira** |
| 3 | 5 | **56** | 0 | 0 | 5 | 1 | **asco** |
| 3 | 1 | 0 | **65** | 1 | 0 | 0 | **alegría** |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 4 | 0 | 1 | **53** | 12 | 0 | **neutral** |
| 3 | 6 | 3 | 0 | 13 | **44** | 1 | **tristeza** |
| 11 | 0 | 0 | 0 | 2 | 0 | **57** | **sorpresa** |

### 3.2.3 Evaluación de Red Neuronal Perceptrón Multicapa (MLP)

Se clasificaron correctamente 380 (77,55 %) imágenes e incorrectamente 110 (22,45 %), con un coeficiente de Kappa de 0,74.

En la Tabla 3.20 se muestran los resultados por cada clase (emoción) para el clasificador con los diferentes parámetros que miden su efectividad. La clase alegría, sorpresa, asco, ira, neutral fueron las de mejores resultados, entre ellas, la mejor fue alegría. Las de peores resultados fueron tristeza y miedo, entre ellas, la peor fue miedo.

**Tabla 3.20 Estadísticas por clase de MLP utilizando KDEF**

| Razón de verdaderos positivos | Razón de falsos positivos | Precisión | Medida-F | Clase |
|---|---|---|---|---|
| 0,59 | 0,07 | 0,59 | 0,59 | miedo |
| 0,80 | 0,02 | 0,86 | 0,83 | ira |
| 0,87 | 0,03 | 0,82 | 0,85 | asco |
| 0,94 | 0,02 | 0,90 | 0,92 | alegría |
| 0,84 | 0,04 | 0,77 | 0,80 | neutral |
| 0,63 | 0,05 | 0,70 | 0,66 | tristeza |
| 0,76 | 0,04 | 0,78 | 0,77 | sorpresa |
| **0,78** | **0,04** | **0,77** | **0,77** | **Promedio** |

La Tabla 3.21 muestra la matriz de confusión para este clasificador. Se muestran los buenos resultados de las clases alegría, sorpresa, asco, ira y neutral. Además, los malos resultados obtenidos en las clases miedo y tristeza que fueron, principalmente, a imágenes mal clasificadas de la clase miedo con sorpresa y tristeza, y de tristeza con neutral y miedo.

**Tabla 3.21 Matriz de confusión de MLP utilizando KDEF**

| miedo | ira | asco | alegría | neutral | tristeza | sorpresa | ←Clasificado como |
|---|---|---|---|---|---|---|---|

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **41** | 2 | 2 | 4 | 2 | 6 | 13 | **miedo** |
| 4 | **56** | 6 | 0 | 3 | 1 | 0 | **ira** |
| 1 | 2 | **61** | 0 | 0 | 5 | 1 | **asco** |
| 3 | 0 | 0 | **66** | 1 | 0 | 0 | **alegría** |
| 1 | 2 | 0 | 1 | **59** | 7 | 0 | **neutral** |
| 6 | 3 | 5 | 2 | 9 | **44** | 1 | **tristeza** |
| 13 | 0 | 0 | 0 | 3 | 1 | **53** | **sorpresa** |

### 3.2.4 Evaluación de K Vecinos más Cercanos (KNN)

Se clasificaron correctamente 279 (56,94 %) imágenes e incorrectamente 211 (43,06 %), con un coeficiente de Kappa de 0,50.

En la Tabla 3.22 se muestran los resultados por cada clase (emoción) para el clasificador con los diferentes parámetros que miden su efectividad. La clase alegría, sorpresa, neutral fueron las de mejores resultados, entre ellas, la mejor fue alegría. Las de peores resultados fueron ira, asco, tristeza y miedo, entre ellas, las peores fueron miedo y tristeza.

**Tabla 3.22 Estadísticas por clase de KNN utilizando KDEF**

| Razón de verdaderos positivos | Razón de falsos positivos | Precisión | Medida-F | Clase |
|---|---|---|---|---|
| 0,27 | 0,09 | 0,33 | 0,30 | miedo |
| 0,47 | 0,06 | 0,55 | 0,51 | ira |
| 0,60 | 0,03 | 0,76 | 0,67 | asco |
| 0,93 | 0,04 | 0,81 | 0,87 | alegría |
| 0,76 | 0,18 | 0,42 | 0,54 | neutral |
| 0,23 | 0,07 | 0,36 | 0,28 | tristeza |
| 0,73 | 0,04 | 0,76 | 0,74 | sorpresa |
| **0,57** | **0,07** | **0,57** | **0,56** | **Promedio** |

La Tabla 3.23 muestra la matriz de confusión para este clasificador. Se muestran los resultados satisfactorios de las clases alegría, sorpresa y neutral. Además, los resultados de las clases miedo, asco, ira y tristeza que fueron mal clasificadas como neutral, principalmente.

**Tabla 3.23 Matriz de confusión de MLP utilizando KDEF**

| miedo | ira | asco | alegría | neutral | tristeza | sorpresa | ←Clasificado como |
|---|---|---|---|---|---|---|---|
| **19** | 5 | 3 | 7 | 15 | 6 | 15 | **miedo** |
| 5 | **33** | 7 | 1 | 16 | 8 | 0 | **ira** |
| 6 | 6 | **42** | 5 | 6 | 4 | 1 | **asco** |
| 3 | 0 | 1 | **65** | 1 | 0 | 0 | **alegría** |
| 5 | 4 | 0 | 0 | **53** | 8 | 0 | **neutral** |
| 6 | 12 | 2 | 2 | 32 | **16** | 0 | **tristeza** |
| 13 | 0 | 0 | 0 | 4 | 2 | **51** | **sorpresa** |

### 3.2.5 Evaluación de Naive Bayes

Se clasificaron correctamente 299 (61,02 %) imágenes e incorrectamente 191 (38,98 %), con un coeficiente de Kappa de 0,54.

En la Tabla 3.24 se muestran los resultados por cada clase (emoción) para el clasificador con los diferentes parámetros que miden su efectividad. La clase alegría, sorpresa, neutral, asco fueron las de mejores resultados, entre ellas, la mejor fue alegría. Las de peores resultados fueron ira, tristeza y miedo, entre ellas, la peor fue miedo.

**Tabla 3.24 Estadísticas por clase de Naive Bayes utilizando KDEF**

| Razón de verdaderos positivos | Razón de falsos positivos | Precisión | Medida-F | Clase |
|---|---|---|---|---|
| 0,31 | 0,08 | 0,39 | 0,35 | miedo |
| 0,60 | 0,05 | 0,66 | 0,63 | ira |
| 0,66 | 0,06 | 0,65 | 0,65 | asco |
| 0,84 | 0,03 | 0,84 | 0,84 | alegría |
| 0,67 | 0,10 | 0,53 | 0,59 | neutral |
| 0,46 | 0,10 | 0,43 | 0,44 | tristeza |
| 0,73 | 0,04 | 0,76 | 0,74 | sorpresa |
| **0,61** | **0,06** | **0,61** | **0,61** | **Promedio** |

La Tabla 3.25 muestra la matriz de confusión para este clasificador. Se muestran los resultados satisfactorios de las clases alegría, asco, sorpresa y neutral. Además, los malos resultados obtenidos en las clases miedo, ira y tristeza que fueron, principalmente, a imágenes mal clasificadas como neutral, y de la clase miedo con sorpresa.

**Tabla 3.25 Tabla 3.25. Matriz de confusión de Naive Bayes utilizando KDEF**

| miedo | ira | asco | alegría | neutral | tristeza | sorpresa | ←Clasificado como |
|---|---|---|---|---|---|---|---|
| **22** | 8 | 9 | 4 | 9 | 7 | 11 | **miedo** |
| 3 | **42** | 7 | 1 | 10 | 5 | 2 | **ira** |
| 7 | 6 | **46** | 3 | 3 | 4 | 1 | **asco** |
| 5 | 0 | 4 | **59** | 1 | 1 | 0 | **alegría** |
| 3 | 1 | 0 | 0 | **47** | 19 | 0 | **neutral** |
| 3 | 7 | 5 | 3 | 18 | **32** | 2 | **tristeza** |
| 13 | 0 | 0 | 0 | 0 | 6 | **51** | **sorpresa** |

### 3.2.6 Evaluación de Árbol de Decisión (Decision Tree)

Se clasificaron correctamente 262 (53,47 %) imágenes e incorrectamente 228 (46,53 %), con un coeficiente de Kappa de 0,46.

En la Tabla 3.26 se muestran los resultados por cada clase (emoción) para el clasificador con los diferentes parámetros que miden su efectividad. La clase alegría, sorpresa y asco fueron las de mejores resultados, entre ellas, la mejor fue alegría. Las de peores resultados fueron ira, neutral, tristeza y miedo, entre ellas, la peor fue tristeza.

**Tabla 3.26 Estadísticas por clase de Árbol de Decisión utilizando KDEF**

| Razón de verdaderos positivos | Razón de falsos positivos | Precisión | Medida-F | Clase |
|---|---|---|---|---|
| 0,43 | 0,10 | 0,41 | 0,42 | miedo |
| 0,39 | 0,10 | 0,39 | 0,39 | ira |
| 0,60 | 0,06 | 0,62 | 0,61 | asco |
| 0,87 | 0,02 | 0,87 | 0,87 | alegría |

| | | | | |
|---|---|---|---|---|
| 0,47 | 0,09 | 0,45 | 0,46 | neutral |
| 0,26 | 0,10 | 0,30 | 0,28 | tristeza |
| 0,73 | 0,06 | 0,66 | 0,69 | sorpresa |
| **0,53** | **0,08** | **0,53** | **0,53** | **Promedio** |

La Tabla 3.27 muestra la matriz de confusión para este clasificador. Se muestran los resultados satisfactorios de las clases alegría, asco y sorpresa. Además, los malos resultados obtenidos en las clases miedo, ira, neutral y tristeza que fueron debido principalmente a imágenes mal clasificadas de la clase miedo con sorpresa, asco y tristeza, de ira con neutral y asco, de neutral con tristeza e ira, y de tristeza con neutral e ira.

**Tabla 3.27 Matriz de confusión de Árbol de Decisión utilizando KDEF**

| **miedo** | **ira** | **asco** | **alegría** | **neutral** | **tristeza** | **sorpresa** | ←Clasificado como |
|---|---|---|---|---|---|---|---|
| **30** | 5 | 8 | 2 | 3 | 8 | 14 | **miedo** |
| 8 | **27** | 10 | 5 | 11 | 8 | 1 | **ira** |
| 6 | 10 | **42** | 1 | 2 | 7 | 2 | **asco** |
| 2 | 0 | 4 | **61** | 3 | 0 | 0 | **alegría** |
| 4 | 13 | 1 | 0 | **33** | 17 | 2 | **neutral** |
| 10 | 12 | 2 | 1 | 20 | **18** | 7 | **tristeza** |
| 13 | 3 | 1 | 0 | 1 | 1 | **51** | **sorpresa** |

### 3.2.7 Evaluación de Bosque Aleatorio (Random Forest)

Se clasificaron correctamente 331 (67,55 %) imágenes e incorrectamente 159 (32,45 %), con un coeficiente de Kappa de 0,62.

En la Tabla 3.28 se muestran los resultados por cada clase (emoción) para el clasificador con los diferentes parámetros que miden su efectividad. La clase alegría, sorpresa, ira, neutral y asco fueron las de mejores resultados, entre ellas, la mejor fue alegría. Las de peores resultados fueron tristeza y miedo, entre ellas, la peor fue tristeza.

**Tabla 3.28 Estadísticas por clase de Bosque Aleatorio utilizando KDEF**

| Razón de verdaderos positivos | Razón de falsos positivos | Precisión | Medida-F | Clase |
|---|---|---|---|---|
| 0,50 | 0,07 | 0,54 | 0,52 | miedo |
| 0,61 | 0,04 | 0,70 | 0,66 | ira |
| 0,79 | 0,04 | 0,77 | 0,78 | asco |
| 0,91 | 0,02 | 0,88 | 0,89 | alegría |
| 0,69 | 0,09 | 0,54 | 0,61 | neutral |
| 0,39 | 0,07 | 0,49 | 0,43 | tristeza |
| 0,84 | 0,04 | 0,77 | 0,80 | sorpresa |
| **0,68** | **0,05** | **0,67** | **0,67** | **Promedio** |

La Tabla 3.29 muestra la matriz de confusión para este clasificador. Se muestran los resultados satisfactorios de las clases alegría, sorpresa, ira, neutral y asco. Además, los malos resultados obtenidos en las clases miedo y tristeza que fueron, principalmente, a imágenes mal clasificadas de la clase miedo con sorpresa y neutral, y de tristeza con neutral y miedo.

**Tabla 3.29 Matriz de confusión de Bosque Aleatorio utilizando KDEF**

| miedo | ira | asco | alegría | neutral | tristeza | sorpresa | ←Clasificado como |
|---|---|---|---|---|---|---|---|
| **35** | 4 | 3 | 3 | 6 | 4 | 15 | **miedo** |
| 8 | **43** | 6 | 2 | 6 | 4 | 1 | **ira** |
| 4 | 5 | **55** | 0 | 2 | 3 | 1 | **asco** |
| 2 | 0 | 3 | **64** | 1 | 0 | 0 | **alegría** |
| 2 | 3 | 1 | 1 | **48** | 15 | 0 | **neutral** |
| 8 | 5 | 3 | 3 | 23 | **27** | 1 | **tristeza** |
| 6 | 1 | 0 | 0 | 2 | 2 | **59** | **sorpresa** |

## *3.3 Análisis del funcionamiento de FER REST API*

En el presente epígrafe, mediante la herramienta Postman, se analiza el funcionamiento general de FER REST API. Se muestran ejemplos del funcionamiento probando sus funcionalidades. Se analizan tres imágenes con emociones distintas y comprobar si la emoción detectada es la correcta.

Los clasificadores son entrenados mediante la base de datos de emociones Cohn Kanade por los excelentes resultados que se obtuvieron.

### 3.3.1 Entrenar clasificador Máquina de Soporte Vectorial y Red Neuronal Perceptrón Multicapa

Se entrenan los clasificadores SVM con kernel polinomial y MLP.

Se realiza la petición mediante la URL http://localhost:8000/classifier?classifier=SVM, para entrenar el clasificador SVM. Se muestra la petición para el entrenamiento y la respuesta en formato JSON (véase Figura 3.3).

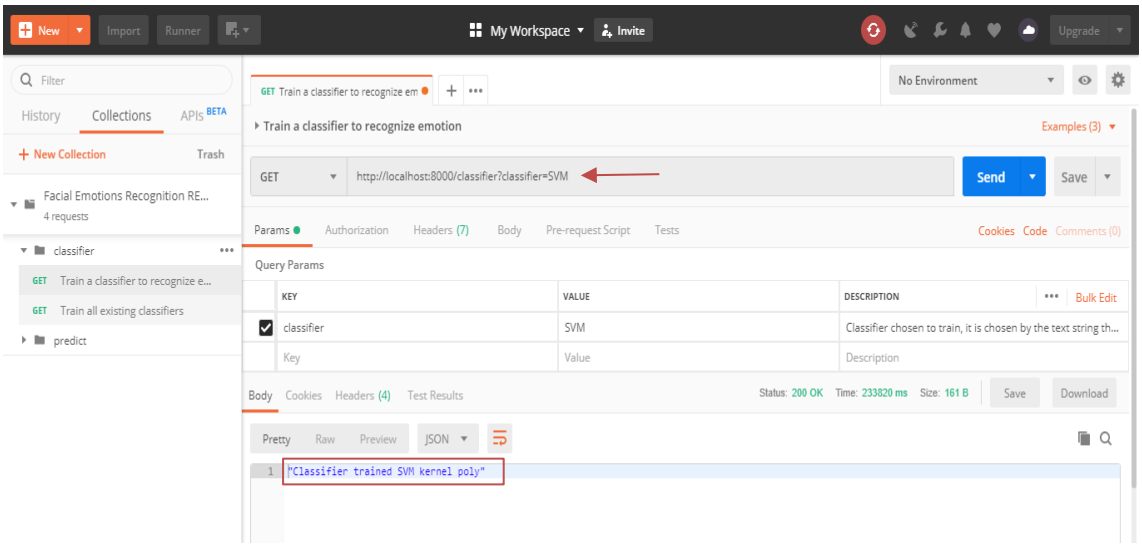



**Figura 3.3 Petición con Postman a FER REST API para entrenar el clasificador SVM**

Se realiza la petición mediante la URL http://localhost:8000/classifier?classifier=MLP, para entrenar el clasificador MLP. Se muestra la petición para realizar el entrenamiento y la respuesta en formato JSON (véase Figura 3.4).

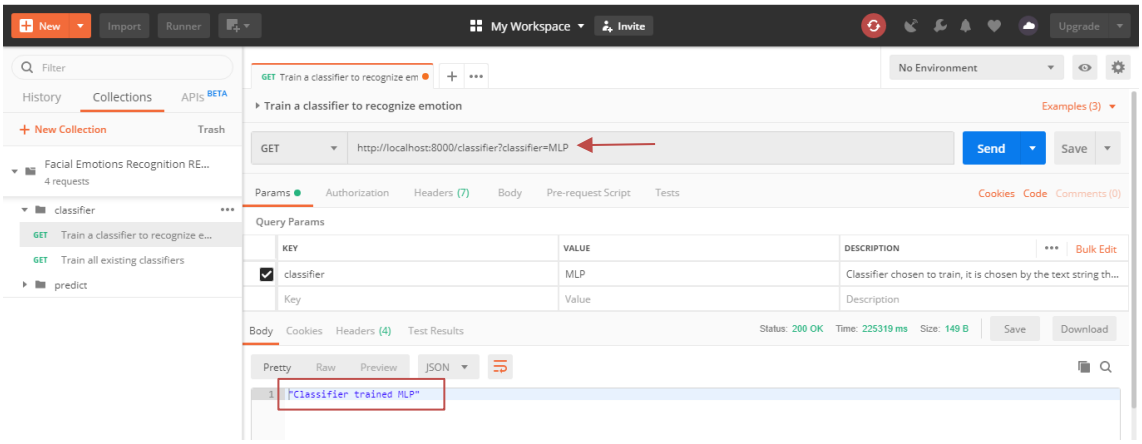



**Figura 3.4 Petición con Postman a FER REST API para entrenar el clasificador MLP**

Los clasificadores entrenados se guardan en sus archivos correspondientes para su uso posterior.

### 3.3.2 Entrenar varios clasificadores

Se entrenan todos los clasificadores de la API para comprobar el funcionamiento en la etapa de entrenamiento. Se realiza la petición mediante la URL http://localhost:8000/classifier/all. Se muestra la petición para realizar el entrenamiento de todos los clasificadores y la respuesta en formato JSON (véase Figura 3.5).

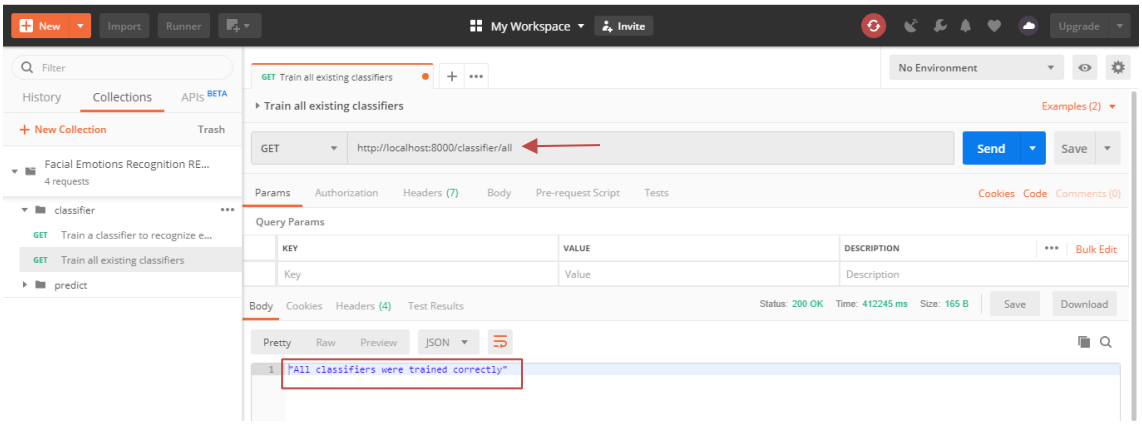



**Figura 3.5 Petición con Postman a FER REST API para entrenar todos los clasificadores**

### 3.3.3 Reconocimiento de emoción

Para probar la detección automática de emociones faciales se intenta reconocer la emoción correspondiente a dos imágenes propias con las emociones alegría y sorpresa. Las imágenes fueron capturadas mediante la webcam integrada de una laptop marca Dell Inspiron 15 3537. Se encuentran en formato JPG, con una resolución de 1280x720 píxeles. La captura se realizó bajo un

ambiente controlado y no profesional. Son imágenes frontales y las emociones contenidas en las imágenes son espontáneas. Las respuestas de la API se realizan en JSON.

Se reconoce la emoción presente en la imagen que contiene la emoción alegría (véase Figura 3.6) mediante el clasificador SVM con kernel polinomial.

**Figura 3.6 Imagen con emoción alegría**

Se realiza la petición mediante la URL http://localhost:8000/predict?imgPath=E:\alegria.jpg&classifier=SVM, la ruta de la imagen y el clasificador previamente entrenado.

En la Figura 3.7 se muestra la petición para realizar el reconocimiento de la imagen con el clasificador especificado y la respuesta de la detección de la emoción.

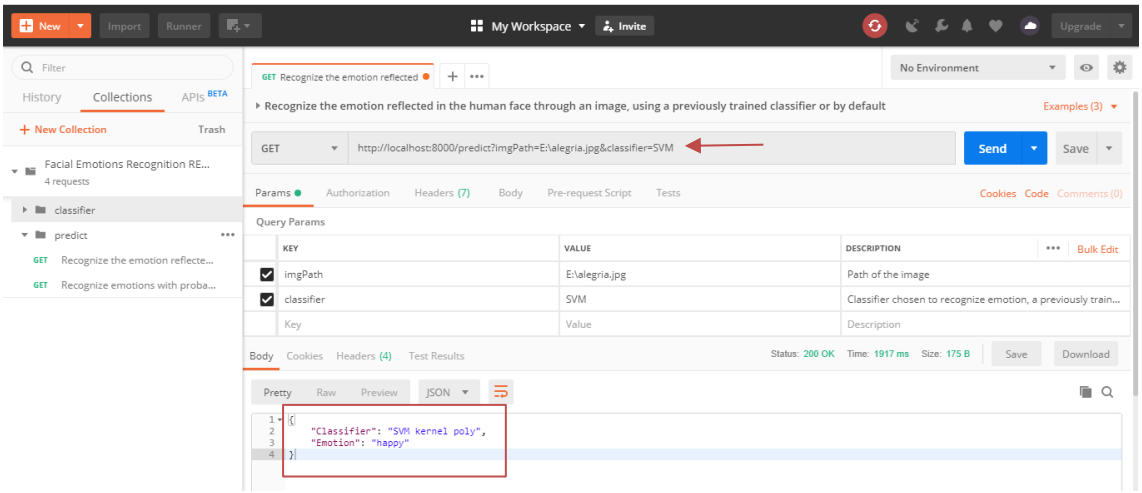



**Figura 3.7 Petición con Postman a FER REST API para reconocer la emoción alegría de la imagen**

Como se puede observar, se reconoce correctamente la emoción presente en la imagen.

Se reconoce la emoción presente en la imagen que contiene la emoción alegría (véase Figura 3.6) mediante el clasificador MLP.

Se realiza la petición mediante la URL http://localhost:8000/predict?imgPath=E:\alegria.jpg&classifier=MLP, la ruta de la imagen y el clasificador previamente entrenado.

En la Figura 3.8 se muestra la petición para realizar el reconocimiento de la imagen con el clasificador especificado y la respuesta de la detección de la emoción.

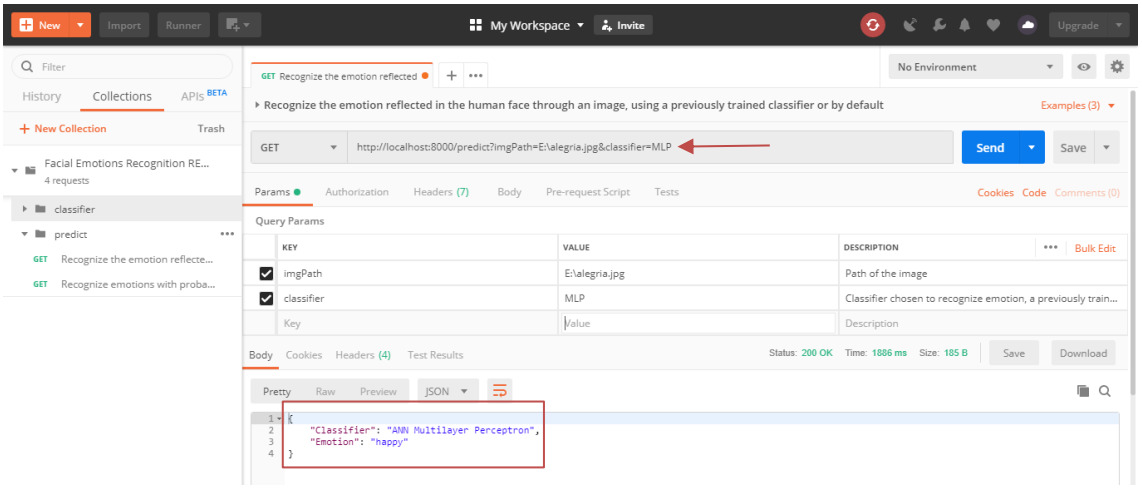



**Figura 3.8 Petición con Postman a FER REST API para reconocer la emoción alegría de la imagen con el clasificador MLP**

Como se puede observar, se reconoce correctamente la emoción presente en la imagen.

Se reconoce la emoción presente en la imagen que contiene la emoción sorpresa (véase Figura 3.9) mediante el clasificador SVM con kernel polinomial.

**Figura 3.9 Imagen con emoción sorpresa**

Se realiza la petición mediante la URL http://localhost:8000/predict?imgPath=E:\sorpresa.jpg&classifier=SVM, la ruta de la imagen y el clasificador previamente entrenado.

En la Figura 3.10 se muestra la petición para realizar el reconocimiento de la imagen con el clasificador especificado y la respuesta de la detección de la emoción.

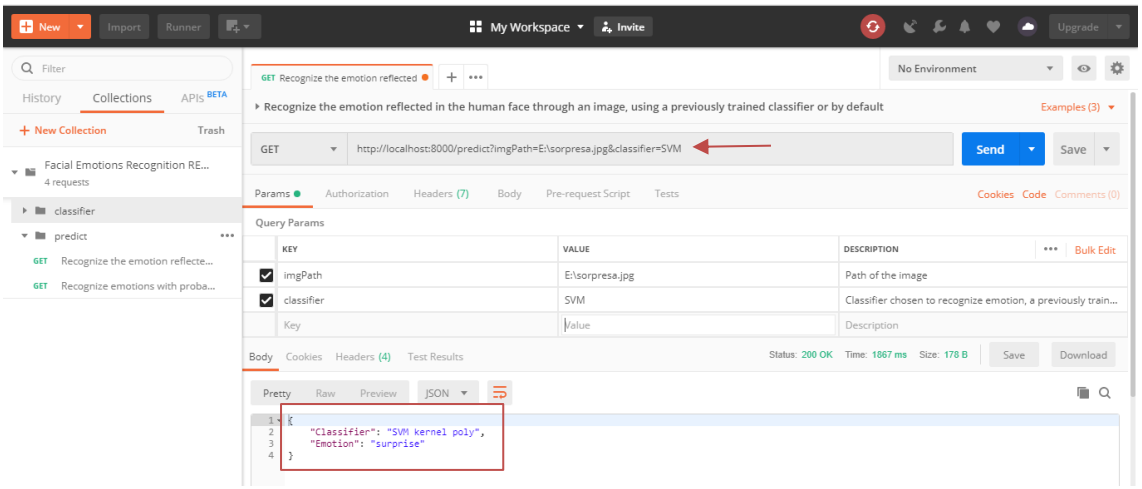



**Figura 3.10 Petición con Postman a FER REST API para reconocer la emoción sorpresa de la imagen con el clasificador SVM**

Como se puede observar, se reconoce correctamente la emoción presente en la imagen.

Se reconoce la emoción presente en la imagen que contiene la emoción sorpresa (véase Figura 3.9) mediante el clasificador MLP.

Se realiza la petición mediante la URL http://localhost:8000/predict?imgPath=E:\sorpresa.jpg&classifier=MLP, la ruta de la imagen y el clasificador previamente entrenado.

En la Figura 3.11 se muestra la petición para realizar el reconocimiento de la imagen con el clasificador especificado y la respuesta de la detección de la emoción.

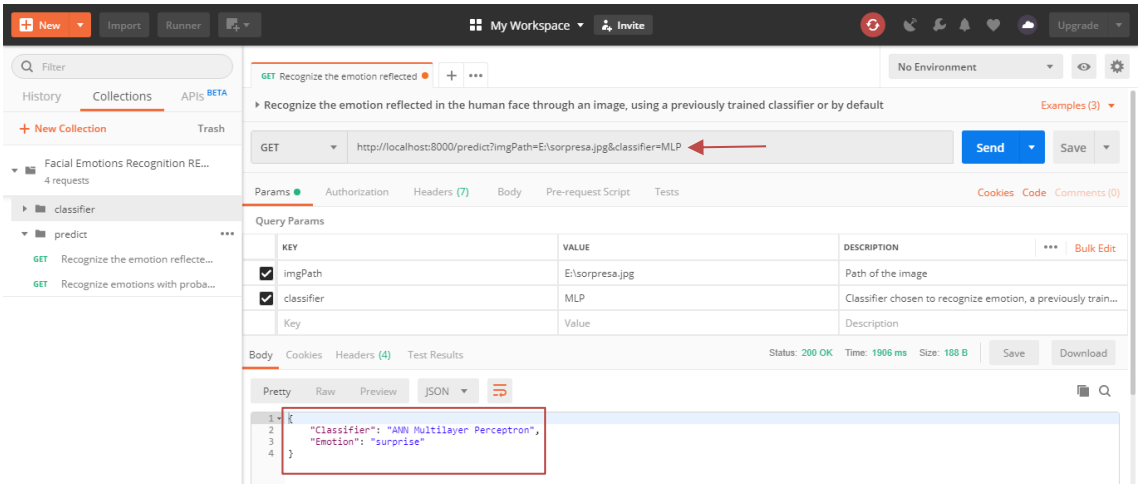



**Figura 3.11 Petición con Postman a FER REST API para reconocer la emoción sorpresa de la imagen con el clasificador MLP**

Como se puede observar, se reconoce correctamente la emoción presente en la imagen.

El reconocimiento de las emociones contenidas en las imágenes fue satisfactorio utilizando los clasificadores supervisados propuestos.

### 3.3.4 Reconocimiento de las emociones con probabilidades

Se reconocen las probabilidades por emociones correspondientes en la imagen que contiene la emoción alegría (véase Figura 3.6), mediante el clasificador SVM con kernel polinomial.

Se realiza la petición mediante la URL http://localhost:8000/predict/stt?imgPath=E:\alegria.jpg&classifier=SVM, la ruta de la imagen y el clasificador previamente entrenado.

En la Figura 3.12 se muestra la petición para realizar el reconocimiento de la imagen con el clasificador especificado y la respuesta de la detección de las emociones por probabilidades, siendo la de mayor probabilidad la alegría.

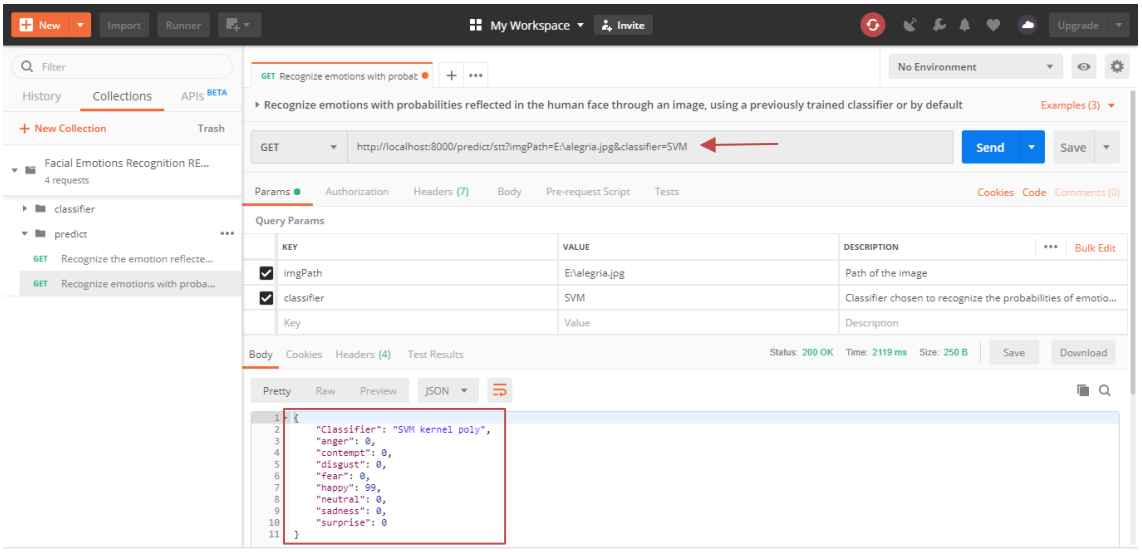



**Figura 3.12 Petición con Postman a FER REST API para reconocer la emoción alegría de la imagen con el clasificador SVM**

Como se puede observar, se reconoce correctamente las emociones por probabilidades presente en la imagen, y se detecta que la de mayor probabilidad es la alegría.

Se reconocen las probabilidades por emociones correspondientes en la imagen que contiene la emoción alegría (véase Figura 3.6), mediante el clasificador MLP.

Se realiza la petición mediante la URL http://localhost:8000/predict/stt?imgPath=E:\alegria.jpg&classifier=MLP, la ruta de la imagen y el clasificador previamente entrenado.

En la Figura 3.13 se muestra la petición para realizar el reconocimiento de la imagen con el clasificador especificado y la respuesta de la detección de las emociones por probabilidades, siendo la de mayor probabilidad la alegría.

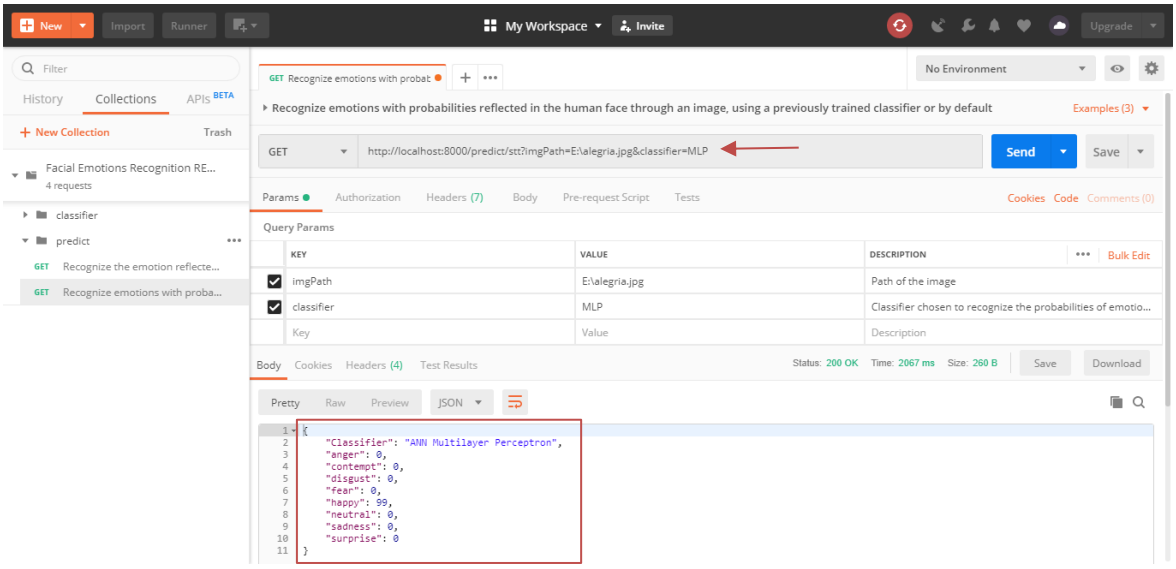



**Figura 3.13 Petición con Postman a FER REST API para reconocer la emoción alegría de la imagen con el clasificador MLP**

Como se puede observar, se reconoce correctamente las emociones por probabilidades presente en la imagen, y se detecta que la de mayor probabilidad es la alegría.

Se reconocen las probabilidades por emociones correspondientes en la imagen que contiene la emoción sorpresa (véase Figura 3.9), mediante el clasificador SVM con kernel polinomial.

Se realiza la petición mediante la URL http://localhost:8000/predict/stt?imgPath=E:\sorpresa.jpg&classifier=SVM, la ruta de la imagen y el previamente entrenado.

En la Figura 3.14 se muestra la petición para realizar el reconocimiento de la imagen con el clasificador especificado y la respuesta de la detección de las emociones por probabilidades, siendo la de mayor probabilidad la sorpresa.

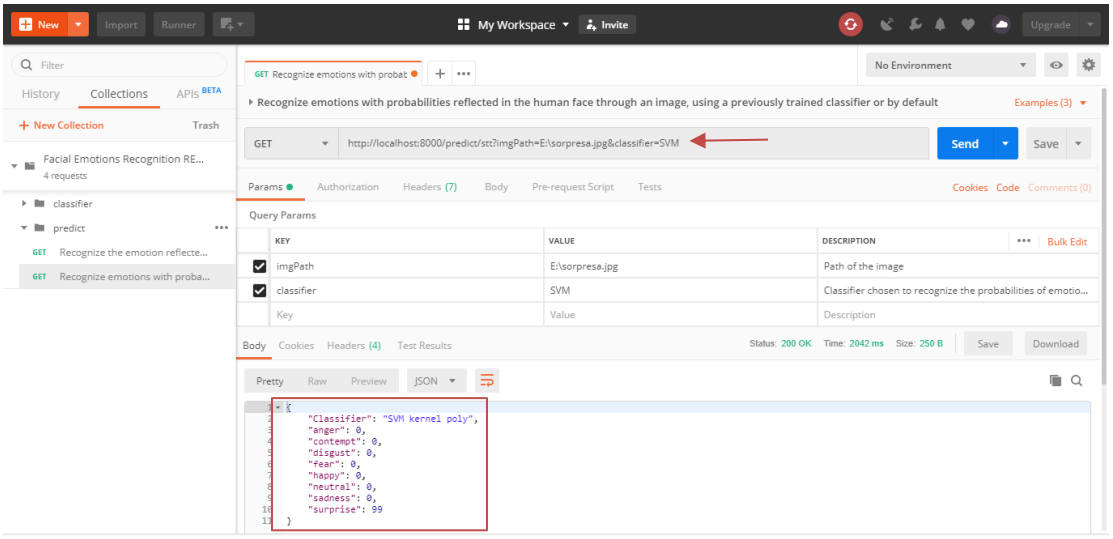



**Figura 3.14 Petición con Postman a FER REST API para reconocer la emoción sorpresa de la imagen con el clasificador SVM**

Como se puede observar, se reconoce correctamente las emociones por probabilidades presente en la imagen, y se detecta que la de mayor probabilidad es la sorpresa.

Se reconocen las probabilidades por emociones correspondientes en la imagen que contiene la emoción sorpresa (véase Figura 3.9), mediante el clasificador MLP.

Se realiza la petición mediante la URL http://localhost:8000/predict/stt?imgPath=E:\sorpresa.jpg&classifier=MLP, la ruta de la imagen y el clasificador previamente entrenado.

En la Figura 3.15 se muestra la petición para realizar el reconocimiento de la imagen con el clasificador especificado y la respuesta de la detección de las emociones por probabilidades, siendo la de mayor probabilidad la sorpresa.

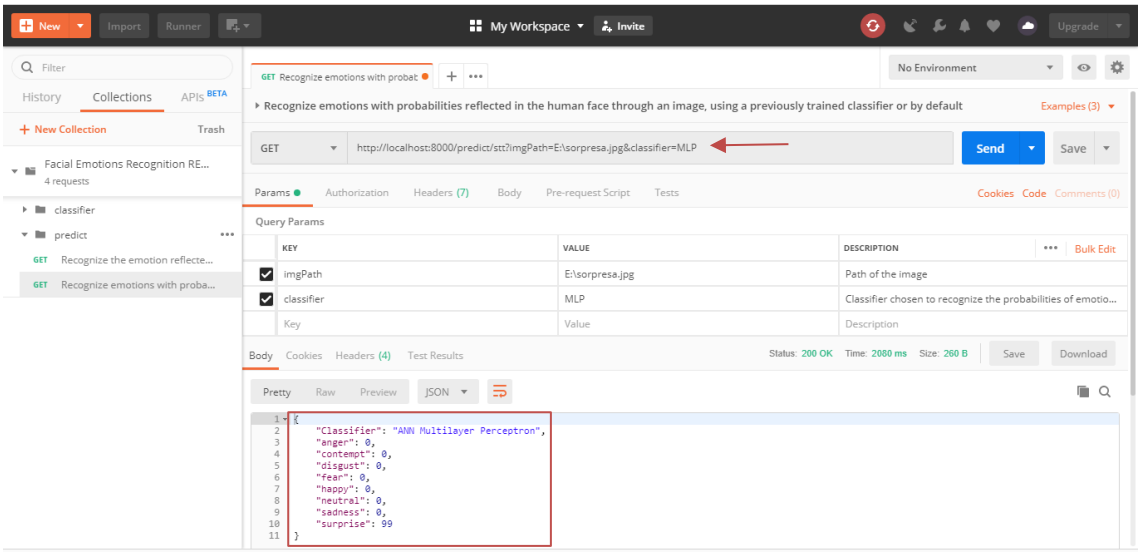



**Figura 3.15 Petición con Postman a FER REST API para reconocer la emoción sorpresa de la imagen con el clasificador MLP**

Como se puede observar, se reconoce correctamente las emociones por probabilidades presente en la imagen, y se detecta que la de mayor probabilidad es la sorpresa.

El reconocimiento de las emociones por probabilidades contenidas en las imágenes fue satisfactorio utilizando los clasificadores supervisados propuestos.

## *3.4 Conclusiones parciales*

En este capítulo se realiza una descripción de los resultados obtenidos con FER REST API.

La base de datos de emociones Cohn Kanade posee mejores características que KDEF para evaluar e implementar sistemas de detección automática de emociones faciales. Cohn Kanade posee una excelente composición, es más variada en cuanto a razas, géneros, edades de sus participantes, iluminación, colores de las imágenes, a pesar de no está equilibrada la cantidad de imágenes por emoción.

Weka es una herramienta de software muy útil para evaluar los clasificadores de aprendizaje automático, ajustar sus parámetros y aumentar la efectividad en la clasificación.

Se utiliza la técnica de validación cruzada por ser la recomendada en la literatura.

La matriz de confusión permite visualizar el funcionamiento del clasificador y las distintas estadísticas obtenidas permiten logran un mayor entendimiento del mismo.

Los clasificadores Máquina de Soporte Vectorial con kernel polinomial y lineal, y la Red Neuronal Perceptrón Multicapa fueron los de mejor precisión en la clasificación de emociones.

Mediante Postman se realizaron peticiones de prueba a la API. Todas las funcionalidades dieron respuestas satisfactorias. FER REST API realiza la correcta detección de las emociones en imágenes propias, evidenciando el correcto funcionamiento y alta precisión en la clasificación.

Los resultados obtenidos en las validaciones y el funcionamiento en general de FER REST API son similares a otros sistemas de reconocimiento automático de emociones faciales, APIs y SDK revisados en la literatura.

## CONCLUSIONES

1. La Computación Afectiva es una rama de la Inteligencia Artificial que hace referencia al diseño de sistemas y dispositivos para reconocer, interpretar, procesar y generar emociones humanas mejorando la interacción entre el usuario y la computadora.
2. Las expresiones faciales de emociones resultan importantes en la interacción entre los humanos. El rostro es el principal sistema de señales para mostrar las emociones. La clasificación de las mismas es una tarea complicada para los seres humanos y los sistemas de detección automática de emociones.
3. FastAPI es el framework indicado para la creación de API REST, por ser minimalista, muy potente, moderno e intuitivo y uno de los más rápidos disponibles en Python.
4. La API REST garantiza seguridad, escalabilidad, simplicidad y portabilidad. Además, permite a los programadores la habilidad de implementar nuevas funcionalidades modulares mediante interfaces muy ligeras sin la necesidad de una integración compleja, con independencia de tecnologías y lenguajes de programación.
5. FER REST API logra integrar el reconocimiento automático de emociones faciales en tiempo real desde aplicaciones implementadas en diferentes lenguajes de programación y sistemas operativos. Es de fácil instalación, ejecución y manejo. Está implementada con el fin de que los programadores que no tengan conocimientos específicos sobre el tema la puedan utilizar fácilmente.
6. FER REST API en su funcionamiento y resultados obtenidos, tiene gran similitud con APIs, SDK y sistemas de detección automática de emociones faciales revisados en la literatura.
7. La API obtenida es un acercamiento al empleo de la Computación Afectiva en aplicaciones de diferentes índoles. Los resultados alcanzados brindan la posibilidad de nuevas investigaciones futuras sobre este campo tan amplio.

## RECOMENDACIONES

- Probar la FER REST API para medir su funcionamiento y mejorar su acierto, con otras bases de datos de emociones que sean amplias y recomendadas por sus características.
- Generalizar la FER REST API en el país para aplicaciones que necesiten incluir la Computación Afectiva y el reconocimiento de emociones faciales.
- Incluir a FER REST API el aprendizaje profundo (deep learning) como subcategoría del aprendizaje automático, utilizando el clasificador una Red Neuronal Convolucional (Convolutional Neural Network, CNN).
- Incluir el reconocimiento de emociones por otros canales del cuerpo humano como la voz, los movimientos corporales a la API logrando un sistema más completo.

## REFERENCIAS BIBLIOGRÁFICAS

Alpaydin, E. (2014) *Introduction to machine learning*. Third. Edited by MIT Press. Cambridge, Massachusetts London, England. Available at: https://mitpress.mit.edu/books/introduction-machine-learning-third-edition (Accessed: 10 June 2019).

Arango de Montis, I. *et al.* (2013) 'Recognition of facial expression of the emotions and their relation to attachment styles and psychiatric symptoms. Preliminary study on Psychiatric Residents'. Instituto Nacional de Psiquiatría Ramón de la Fuente Muñiz, Calz. México-Xochimilco 101, Col. San Lorenzo Huipulco, Tlalpan, México, D.F. Tel. 4160-5000., 36, p. 95–100 p. Available at: http://repositorio.inprf.gob.mx/handle/123456789/6114 (Accessed: 10 June 2019).

Arboleda Rodríguez, V., Gallar Pérez, Y. and Barrios Queipo, E. A. (2017) 'Consideraciones teóricas acerca de la Computación Afectiva en el proceso de enseñanza aprendizaje de la Educación Superior', *CienciAmérica: Revista de divulgación científica de la Universidad Tecnológica Indoamérica, ISSN-e 1390-9592, Vol. 6, Nº. 3, 2017 (Ejemplar dedicado a: Revista CienciAmérica), págs. 170-175*, 6(3), pp. 170–175. Available at: https://dialnet.unirioja.es/servlet/articulo?codigo=6163703.

Bailón Delgado, D. H. (2015) 'Ingeniería del software: Lenguaje Unificado de Modelado UML', p. 11.

Baldasarri, S. (2016) 'Computación Afectiva : tecnología y emociones para mejorar la experiencia de usuario', *Revista Institucional de la Facultad de Informática. Universidad Nacional de La Plata*, 3(3), pp. 1–2.

Baldassarri, D. S. (2012) 'Computación Afectiva. Aplicación educativa para TVDi'.

Banafa, A. (2016) *¿Qué es la computación afectiva? - OpenMind*. Available at: https://www.bbvaopenmind.com/tecnologia/mundo-digital/que-es-la-computacion-afectiva/ (Accessed: 10 June 2019).

BBVAOpen4U (2016) *API REST: qué es y cuáles son sus ventajas en el desarrollo de proyectos*. Available at: https://bbvaopen4u.com/es/actualidad/api-rest-que-es-y-cuales-son-sus-ventajas-en-el-desarrollo-de-proyectos (Accessed: 10 June 2019).

Bosquez Barcenes, V. A. *et al.* (2018) 'La Computación Afectiva : emociones , tecnologías y su

relación con la educación virtual', *Revista de Investigación Talentos Volumen V. (1)*, 1(1), pp. 94–103.

Bouckaert, R. R. *et al.* (2018) *WEKA Manual for Version 3-9-3*. Hamilton, New Zealand: University of Waikato.

Bravo del Río, A. (2016) *Módulo para la integración de la plataforma Moodle con el repositorio institucional DSpace*. Universidad Central 'Marta Abreu' de Las Villas.

Browning, J. B. and Alchin, M. (2019) *Pro Python 3 Features and Tools for Professional Development*. Third. Apress. doi: https://doi.org/10.1007/978-1-4842-4385-5.

Chakray (2017) *¿Cuáles son las ventajas de una API REST?* Available at: https://www.chakray.com/es/cuales-son-las-ventajas-de-una-api-rest/ (Accessed: 10 June 2019).

Chaparro, J. A., Giraldo, B. and Rodón, S. (2013) 'Evaluación del clasificador Naïve Bayes como herramienta de diagnóstico en Unidades de Cuidado Intensivo'.

Christie, T. (2019) *Home - Django REST framework*. Available at: https://www.django-rest-framework.org/ (Accessed: 10 June 2019).

Copperwaite, M. and Leifer, C. (2015) *Learning Flask framework : build dynamic, data-driven websites and modern web applications with Flask*. Packt Publishing Ltd. Available at: http://www.allitebooks.org/learning-flask-framework/ (Accessed: 10 June 2019).

Dalal, N. and Triggs, B. (2005) 'Histograms of Oriented Gradients for Human Detection', *International Conference on Computer Vision*.

Darwin, C. (1872) *The expression of the emotions in man and animals*. Edited by J. Murray. London.

Department of Clinical Neuroscience Psychology Karolinska Institutet (2019) *About KDEF*. Available at: http://kdef.se/home/aboutKDEF.html (Accessed: 13 June 2019).

Django Software Foundation (2019) *Django: The Web framework for perfectionists with deadlines*. Available at: https://www.djangoproject.com/ (Accessed: 15 June 2019).

Dlib (2019) *dlib C++ Library*. Available at: http://dlib.net/ (Accessed: 10 June 2019).

Domingo Muñoz, J. (2017) *¿Qué es Flask?* Available at: https://openwebinars.net/blog/que-es-flask/ (Accessed: 15 June 2019).

Douglas Crockford (2019) *JSON*. Available at: https://www.json.org/ (Accessed: 13 June 2019).

EC Software GmbH (2019) *Help & Manual*. Available at: https://www.helpandmanual.com (Accessed: 15 June 2019).

Ekman, P. and Friesen, W. (1978) 'Manual for the facial action coding system', *Consulting Psychologists Press*.

Ekman, P. and Friesen, W. V. (1975) *Unmasking the face: A guide to recognizing emotions from facial clues*. Oxford,England: Prentice-Hall. Available at: https://books.google.es/books?hl=es&lr=&id=TukNoJDgMTUC&oi=fnd&pg=PR3&dq=Ekman,+P.,+%26+Friesen,+W.+V.+(1975).+Unmasking+the+face:+A+guide+to+recognizing+emotions+from+facial+clues.+Oxford,+England:+Prentice-Hall.&ots=GVImak82g8&sig=Ip_gmex7hzWVYix9WT0Kba (Accessed: 13 June 2019).

Ekman, P. and Oster, H. (1979) 'Facial Expressions of Emotion', *Annual Review of Psychology*. Annual Reviews 4139 El Camino Way, P.O. Box 10139, Palo Alto, CA 94303-0139, USA , 30(1), pp. 527–554. doi: 10.1146/annurev.ps.30.020179.002523.

Ekman, P. and Rosenberg, E. L. (2012) *What the Face Reveals: Basic and Applied Studies of Spontaneous Expression Using the Facial Action Coding System (FACS)*. Second. New York: Oxford University Press. doi: 10.1093/acprof:oso/9780195179644.001.0001.

Elkan, C. (2011) 'Evaluating Classifiers'.

FastAPI (2019) *FastAPI*. Available at: https://fastapi.tiangolo.com/ (Accessed: 13 June 2019).

Felzenszwalb, P. F. *et al.* (2010) 'Object Detection with Discriminatively Trained Part Based Model Object localization', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 32(No. 9).

Fielding, R. T. (2000) *Architectural Styles and the Design of Network-based Software Architectures*. University of California, Irvine. Available at: https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf.

GlosarioIT (2019) *SOAP - GlosarioIT: Glosario Informático*. Available at: https://www.glosarioit.com/SOAP (Accessed: 10 June 2019).

Gómez Palomo, S. R. and Moraleda Gil, E. (2014) *Aproximación a la Ingeniería del Software*. Edited by C. de E. R. Areces.

González, R. *et al.* (2017) 'Aplicación de las Máquinas de Soporte Vectorial (SVM) al diagnóstico clínico de la Enfermedad de Párkinson y el Temblor Esencial', *Revista Iberoamericana de Automática e Informática Industrial*. Elsevier Srl, 14(4), pp. 394–405. doi: 10.1016/j.riai.2017.07.005.

Hatem, H., Beiji, Z. and Majeed, R. (2015) 'A survey of feature base methods for human face detection', *International Journal of Control and Automation*, 8(5), p. pp.61-78. doi: http://dx.doi.org/10.14257/ijca.2015.8.5.07.

Hernández Orallo, E. (2017) *El Lenguaje Unificado de Modelado (UML)*.

Hussain Shah, J. *et al.* (2015) 'Robust Face Recognition Technique under Varying Illumination', *Journal of Applied Research and Technology*. Elsevier, 13(1), pp. 97–105. doi: 10.1016/S1665-6423(15)30008-0.

Iglesias Hoyos, S., del Castillo Arreola, A. and Muñoz Delgado, J. I. (2016) 'Reconocimiento facial de expresión emocional : diferencias por licenciaturas', *Acta de Investigación Psicológica - Psychological Research Records*, 6(núm. 3), pp. 2494–1499.

Jetbrains (2019) *PyCharm*. Available at: https://www.jetbrains.com/pycharm/ (Accessed: 15 June 2019).

Johansson, R. (2019) *Numerical Python: Scientific Computing and Data Science Applications with Numpy,SciPy and Matplotlib*. Second. Apress. doi: https://doi.org/10.1007/978-1-4842-4246-9.

Juárez, G. E. (2018) *Inteligencia Artificial: Redes Neuronales*.

Kaehler, A. and Bradski, G. (2017) *Learning OpenCV 3 Computer Vision in C++ with the OpenCV Library*. O'reilly.

Kanade, T., Cohn, J. F. and Tian, Y. (2000) 'Comprehensive Database for Facial Expression Analysis The Robotics Institute', *In Proceedings of the International Conference on Automatic Face and Gesture Recognition*, pp. 46–53.

Kaur, G. and Singla, A. (2016) 'Sentimental Analysis of Flipkart reviews using Naïve Bayes and Decision Tree algorithm', *International Journal of Advanced Research in Computer Engineering &Technology (IJARCE)*, 5(1), pp. 148–153.

Kazemi, V. and Sullivan, J. (2014) 'One millisecond face alignment with an ensemble of regression trees'. Available at:

http://openaccess.thecvf.com/content_cvpr_2014/html/Kazemi_One_Millisecond_Face_2014_CVPR_paper.html (Accessed: 10 June 2019).

Kingma, D. P. and Lei Ba, J. (2015) 'Adam: A method for stochastic optimization', *ICLR 2015*, pp. 1–15.

Korshunov, P. and Marcel, S. (2018) 'Speaker Inconsistency Detection in Tampered Video', in *2018 26th European Signal Processing Conference (EUSIPCO)*. IEEE, pp. 2375–2379. doi: 10.23919/EUSIPCO.2018.8553270.

Liu, B. (2017) *Web data mining : exploring hyperlinks, contents, and usage data*. Edited by Springer.

Lorenzo Gil, E., Braña Ferreiro, E. and Nieto Caramés, S. (2015) 'Estudio de la integración de repositorios en el sistema científico-investigador: alternativas y estado actual', *VI Jornadas de OS-Repositorios / XIV Workshop de REBIUN de proyectos digitales, 'Los horizontes de los repositorios' (Universidad de Córdoba, 11-13 marzo 2015)*, pp. 11–13.

Louppe, G. (2015) *Understanding Random Forests: From Theory to Practice*. Available at: http://arxiv.org/abs/1407.7502v3 (Accessed: 10 June 2019).

Lucey, P. *et al.* (2010) 'The Extended Cohn-Kanade Dataset ( CK + ): A complete dataset for action unit and emotion-specified expression', p. 8.

Lundqvist, D., Flykt, A. and Ohman, A. (1998) *The Karolinska Directed Emotional Faces – KDEF. CD-ROM*. Stockholm, Sweden.

Magudeeswaran, V. and Singh, J. F. (2017) 'Contrast limited fuzzy adaptive histogram equalization for enhancement of brain images', *International Journal of Imaging Systems and Technology*, 27(1), pp. 98–103. doi: 10.1002/ima.22214.

Medina Merino, R. F. and Ñique Chacón, C. I. (2017) 'Bosques aleatorios como extensión de los árboles de clasificación con los programas R y Python', pp. 165–190.

Morales, E., González, J. and Escalante, H. J. (2017) *Máquinas de Soporte Vectorial*.

Morera Munt, A. (2018) *Introducción a los modelos de redes neuronales artificiales El Perceptrón simple y multicapa*. Universidad Zaragoza.

Murphy, K. P. (2012) *Machine learning : a probabilistic perspective*. Cambridge,Massachusetts London, England: MIT Press. Available at: https://mitpress.mit.edu/books/machine-learning-1

(Accessed: 10 June 2019).

Newman, S. (2015) *Building microservices: designing fine-grained systems*. First Edition. Edited by M. Loukides and B. MacDonald. O´reilly. Available at: https://www.google.com/books?hl=es&lr=&id=jjl4BgAAQBAJ&oi=fnd&pg=PP1&dq=Building+microservices.+Designing+fine-grained+systems.+Sam+Newman.+O%E2%80%99Reilly&ots=_AMMUs7ViM&sig=laRCOUhog-_wuygNiUIFhYQ_RUc (Accessed: 10 June 2019).

NumPy developers (2019) *NumPy*. Available at: https://www.numpy.org/ (Accessed: 15 June 2019).

OpenAPI Initiative (2019) *Home - OpenAPI Initiative*. Available at: https://www.openapis.org/ (Accessed: 13 June 2019).

Ortiz Vivar, J. E. and Segarra Flores, J. L. (2015) *Plataforma para la Anotación Semántica Automática de Servicios Web RESTful sobre un Bus de Servicios*. Universidad de Cuenca.

Pandorafms Community (2019) *¿Qué es y para qué sirve una API?* Available at: https://pandorafms.com/blog/es/para-que-sirve-una-api/ (Accessed: 10 June 2019).

Pantaleo, G. and Rinaudo, L. (2015) *Ingeniería de Software*. Alfaomega.

Patni, S. (2017) *Pro RESTful APIs: Design, Build and Integrate with REST,JSON,XML and JAX-RS*. Santa Clara,California: Apress. doi: 10.1007/978-1-4842-2665-0.

Paz Pellat, M. (2017) *Computación afectiva*. Available at: http://marcopaz.mx/2017/10/26/computacion-afectiva/ (Accessed: 10 June 2019).

Peinador Yubero, A. (2016) *Smartphone como soporte para Interacción Afectiva: Expresión Afectiva*. Universidad Autónoma de Madrid.

Poria, S. *et al.* (2017) 'A Review of Affective Computing : From Unimodal Analysis to Multimodal Fusion', *Information Fusion*, pp. 98–125. doi: 10.1016/j.inffus.2017.02.003.

Postman Inc (2019) *Postman*. Available at: https://www.getpostman.com/ (Accessed: 15 June 2019).

Pressman, R. S. (2010) *Ingeniería del software: Un enfoque práctico*. Séptima. McGraw-Hill.

Python Software Foundation (2019) *Python*. Available at: https://www.python.org/ (Accessed: 15 June 2019).

Refaeilzadeh, P., Tang, L. and Liu, H. (2008) 'Cross-Validation'.

Rokach, L. and Maimon, O. (2015) *Data mining with decision trees: theory and applications*. 2nd Editio. World Scientific Publishing Co. Pte. Ltd. Available at: https://books.google.com/books?hl=es&lr=&id=OVYCCwAAQBAJ&oi=fnd&pg=PR6&dq=Rokach,+Lior+y+Oded+Maimon+(2014)+Data+mining+with+decision+tres:+theory+and+applications.+World+scientific&ots=tJm5b368TJ&sig=6TSrGLAc2XNwWCTimkt1DD3jIm8 (Accessed: 10 June 2019).

Ronacher, A. (2019a) *Flask*. Available at: http://flask.pocoo.org/ (Accessed: 15 June 2019).

Ronacher, A. (2019b) *Jinja*. Available at: http://jinja.pocoo.org/ (Accessed: 15 June 2019).

Rouse, M. (2019) *What is affective computing (emotion AI)? - Definition from WhatIs.com*. Available at: https://whatis.techtarget.com/definition/affective-computing (Accessed: 10 June 2019).

Rubio, D. (2017) *Beginning Django: Web Application Development and Deployment with Python*. F. Bahia, Ensenada, Baja California, Mexico: Apress. doi: https://doi.org/10.1007/978-1-4842-2787-9.

Rueda Extremera, M. (2017) *Reconocimiento emocional a partir de expresiones faciales y piezas musicales en niños y adultos*. Universidad Autónoma de Madrid.

Sancho Caparrini, F. (2017) *Introducción al Aprendizaje Automático*. Available at: http://www.cs.us.es/~fsancho/?e=75 (Accessed: 10 June 2019).

Scikit-learn (2019) *scikit-learn: machine learning in Python — scikit-learn 0.20.3 documentation*. Available at: https://scikit-learn.org/stable/ (Accessed: 10 June 2019).

Scikit-learn developers (2018) *Scikit-learn user guide Release 0.20.2*.

Shalev Shwartz, S. and Ben David, S. (2014) *Understanding machine learning: From theory to algorithms*. Edited by C. U. Press. doi: 10.1017/CBO9781107298019.

Singh Negi, S. and Singh Bhandari, Y. (2014) 'A hybrid approach to Image Enhancement using Contrast Stretching on Image Sharpening and the analysis of various cases arising using histogram', in *International Conference on Recent Advances and Innovations in Engineering, ICRAIE 2014*, pp. 1–6. doi: 10.1109/ICRAIE.2014.6909232.

Sokolova, M. V. and Fernández Caballero, A. (2015) 'A Review on the Role of Color and Light in

Affective Computing', (5), pp. 275–293. doi: 10.3390/app5030275.

Sommerville, I. (2011) *Ingeniería de software*. Novena. Addison-Wesley.

Soto Gómez, G. and Martínez Rodríguez, G. (2018) *Desarrollo de una biblioteca para el reconocimiento de emociones faciales.* Universidad Central 'Marta Abreu' de Las Villas.

Stowe, M. (2015) *Undisturbed REST: A guide to designing the perfect API*.

Tan, S. *et al.* (2009) 'Adapting naive bayes to domain adaptation for sentiment analysis', *Springer*. Available at: https://link.springer.com/chapter/10.1007/978-3-642-00958-7_31 (Accessed: 10 June 2019).

Tao, J. and Tan, T. (2014) 'Affective Computing : A Review', (March). doi: 10.1007/11573548.

team OpenCV (2019) *OpenCV library*. Available at: https://opencv.org/ (Accessed: 10 June 2019).

TechEmpower (2019a) *TechEmpower*. Available at: https://www.techempower.com/ (Accessed: 13 June 2019).

TechEmpower (2019b) *TechEmpower Framework Benchmarks*. Available at: https://www.techempower.com/benchmarks/#section=test&runid=7464e520-0dc2-473d-bd34-dbdfd7e85911&hw=ph&test=query&l=zijzen-7 (Accessed: 13 June 2019).

UserLand Software (2019) *XML-RPC Home*. Available at: http://xmlrpc.scripting.com/ (Accessed: 10 June 2019).

Vergara Arroyo, J. F. (2019) *Protocolo simple de acceso a objetos (SOAP)*. Available at: https://monografias.com/trabajos29/protocolo-acceso/protocolo-acceso.shtml/ (Accessed: 15 June 2019).

Visual Paradigm (2019a) *How to Design REST API with UML?* Available at: https://www.visual-paradigm.com/support/documents/vpuserguide/276/3420/85154_modelingrest.html (Accessed: 13 June 2019).

Visual Paradigm (2019b) *Visual Paradigm*. Available at: https://www.visual-paradigm.com/ (Accessed: 15 June 2019).

Weka (2019) *Weka 3 - Data Mining with Open Source Machine Learning Software in Java*. Available at: https://www.cs.waikato.ac.nz/ml/weka/ (Accessed: 13 June 2019).

Werkzeug (2019) *Werkzeug*. Available at: https://www.palletsprojects.com/p/werkzeug/)

(Accessed: 15 June 2019).

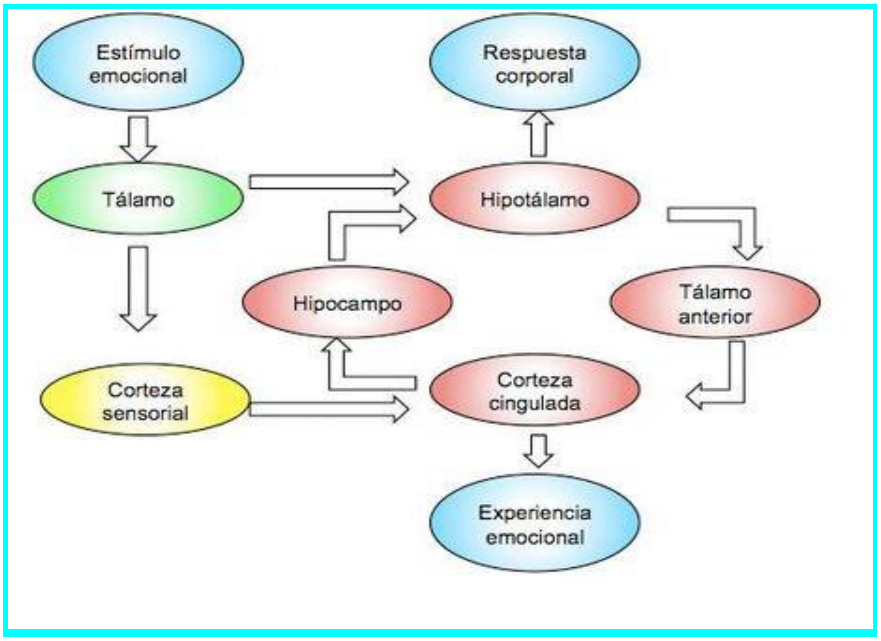
Xamarin (2017) *Introduction to Rest Web Services in Xamarin*. Available at: https://social.technet.microsoft.com/wiki/contents/articles/37978.introduction-to-rest-web-services-in-xamarin.aspx.

YAML (2019) *The Official YAML Web Site*. Available at: https://yaml.org/ (Accessed: 13 June 2019).

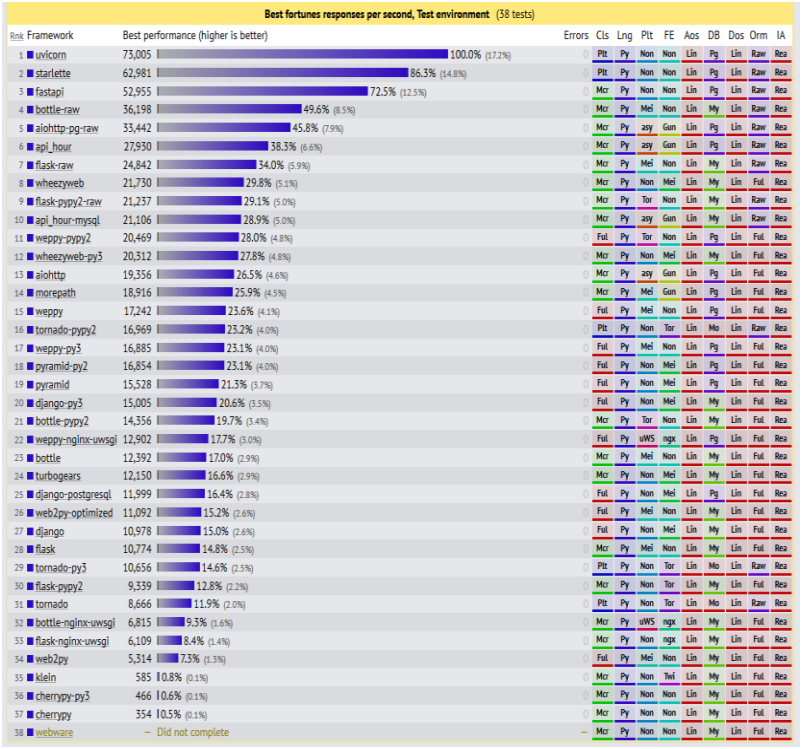
Zuiderveld, K. (1994) 'Contrast limited adaptive histogram equalization', *Graphics gems IV, Academic Press Professional, Inc.*, pp. 474–485.

# ANEXOS

**Anexo 1**. Circuito de Papez



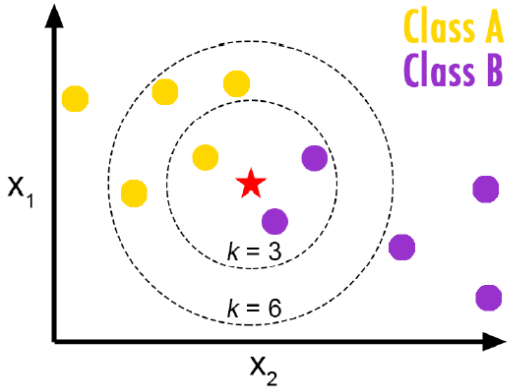
**Anexo 2**. Lista de comparaciones según rendimiento y velocidad de los framework de Python más usados a nivel mundial



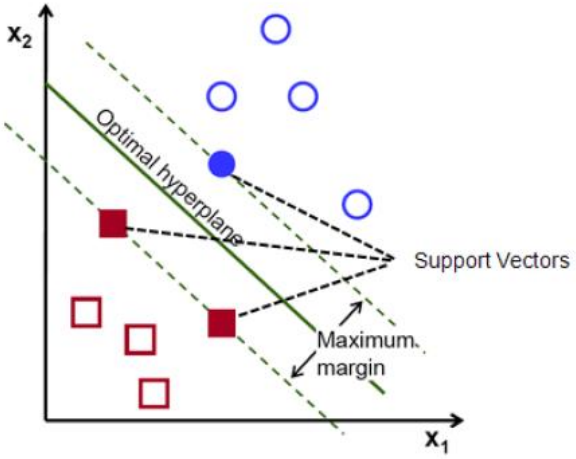
**Best fortunes responses per second, Test environment** (38 tests)

| Rnk | Framework | Best performance (higher is better) | Errors | Cls | Lng | Plt | FE | Aos | DB | Dos | Orm | IA |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | uvicorn | 73,005 100.0% (17.2%) | 0 | Plt | Py | Non | Non | Lin | Pg | Lin | Raw | Rea |
| 2 | starlette | 62,981 86.3% (14.8%) | 0 | Plt | Py | Non | Non | Lin | Pg | Lin | Raw | Rea |
| 3 | fastapi | 52,955 72.5% (12.5%) | 0 | Mcr | Py | Non | Non | Lin | Pg | Lin | Raw | Rea |
| 4 | bottle-raw | 36,198 49.6% (8.5%) | 0 | Mcr | Py | Mei | Non | Lin | My | Lin | Raw | Rea |
| 5 | aiohttp-pg-raw | 33,442 45.8% (7.9%) | 0 | Mcr | Py | asy | Gun | Lin | Pg | Lin | Raw | Rea |
| 6 | api_hour | 27,930 38.3% (6.6%) | 0 | Mcr | Py | asy | Gun | Lin | Pg | Lin | Raw | Rea |
| 7 | flask-raw | 24,842 34.0% (5.9%) | 0 | Mcr | Py | Mei | Non | Lin | My | Lin | Raw | Rea |
| 8 | wheezyweb | 21,730 29.8% (5.1%) | 0 | Mcr | Py | Non | Mei | Lin | My | Lin | Ful | Rea |
| 9 | flask-pypy2-raw | 21,237 29.1% (5.0%) | 0 | Mcr | Py | Tor | Non | Lin | My | Lin | Raw | Rea |
| 10 | api_hour-mysql | 21,106 28.9% (5.0%) | 0 | Mcr | Py | asy | Gun | Lin | My | Lin | Raw | Rea |
| 11 | weppy-pypy2 | 20,469 28.0% (4.8%) | 0 | Ful | Py | Tor | Non | Lin | Pg | Lin | Ful | Rea |
| 12 | wheezyweb-py3 | 20,312 27.8% (4.8%) | 0 | Mcr | Py | Non | Mei | Lin | My | Lin | Ful | Rea |
| 13 | aiohttp | 19,356 26.5% (4.6%) | 0 | Mcr | Py | asy | Gun | Lin | Pg | Lin | Ful | Rea |
| 14 | morepath | 18,916 25.9% (4.5%) | 0 | Mcr | Py | Mei | Gun | Lin | Pg | Lin | Ful | Rea |
| 15 | weppy | 17,242 23.6% (4.1%) | 0 | Ful | Py | Mei | Non | Lin | Pg | Lin | Ful | Rea |
| 16 | tornado-pypy2 | 16,969 23.2% (4.0%) | 0 | Plt | Py | Non | Tor | Lin | Mo | Lin | Raw | Rea |
| 17 | weppy-py3 | 16,885 23.1% (4.0%) | 0 | Ful | Py | Mei | Non | Lin | Pg | Lin | Ful | Rea |
| 18 | pyramid-py2 | 16,854 23.1% (4.0%) | 0 | Ful | Py | Non | Mei | Lin | Pg | Lin | Ful | Rea |
| 19 | pyramid | 15,528 21.3% (3.7%) | 0 | Ful | Py | Non | Mei | Lin | Pg | Lin | Ful | Rea |
| 20 | django-py3 | 15,005 20.6% (3.5%) | 0 | Ful | Py | Non | Mei | Lin | My | Lin | Ful | Rea |
| 21 | bottle-pypy2 | 14,356 19.7% (3.4%) | 0 | Mcr | Py | Tor | Non | Lin | My | Lin | Ful | Rea |
| 22 | weppy-nginx-uwsgi | 12,902 17.7% (3.0%) | 0 | Ful | Py | uWS | ngx | Lin | Pg | Lin | Ful | Rea |
| 23 | bottle | 12,392 17.0% (2.9%) | 0 | Mcr | Py | Mei | Non | Lin | My | Lin | Ful | Rea |
| 24 | turbogears | 12,150 16.6% (2.9%) | 0 | Mcr | Py | Non | Mei | Lin | My | Lin | Ful | Rea |
| 25 | django-postgresql | 11,999 16.4% (2.8%) | 0 | Ful | Py | Non | Mei | Lin | Pg | Lin | Ful | Rea |
| 26 | web2py-optimized | 11,092 15.2% (2.6%) | 0 | Ful | Py | Mei | Non | Lin | My | Lin | Ful | Rea |
| 27 | django | 10,978 15.0% (2.6%) | 0 | Ful | Py | Non | Mei | Lin | My | Lin | Ful | Rea |
| 28 | flask | 10,774 14.8% (2.5%) | 0 | Mcr | Py | Mei | Non | Lin | My | Lin | Ful | Rea |
| 29 | tornado-py3 | 10,656 14.6% (2.5%) | 0 | Plt | Py | Non | Tor | Lin | Mo | Lin | Raw | Rea |
| 30 | flask-pypy2 | 9,339 12.8% (2.2%) | 0 | Mcr | Py | Non | Tor | Lin | My | Lin | Ful | Rea |
| 31 | tornado | 8,666 11.9% (2.0%) | 0 | Plt | Py | Non | Tor | Lin | Mo | Lin | Raw | Rea |
| 32 | bottle-nginx-uwsgi | 6,815 9.3% (1.6%) | 0 | Mcr | Py | uWS | ngx | Lin | My | Lin | Ful | Rea |
| 33 | flask-nginx-uwsgi | 6,109 8.4% (1.4%) | 0 | Mcr | Py | Non | ngx | Lin | My | Lin | Ful | Rea |
| 34 | web2py | 5,314 7.3% (1.3%) | 0 | Ful | Py | Mei | Non | Lin | My | Lin | Ful | Rea |
| 35 | klein | 585 0.8% (0.1%) | 0 | Mcr | Py | Non | Twi | Lin | My | Lin | Ful | Rea |
| 36 | cherrypy-py3 | 466 0.6% (0.1%) | 0 | Mcr | Py | Non | Non | Lin | My | Lin | Ful | Rea |
| 37 | cherrypy | 354 0.5% (0.1%) | 0 | Mcr | Py | Non | Non | Lin | My | Lin | Ful | Rea |
| 38 | webware | – Did not complete | – | Mcr | Py | Non | Non | Lin | My | Lin | Ful | Rea |

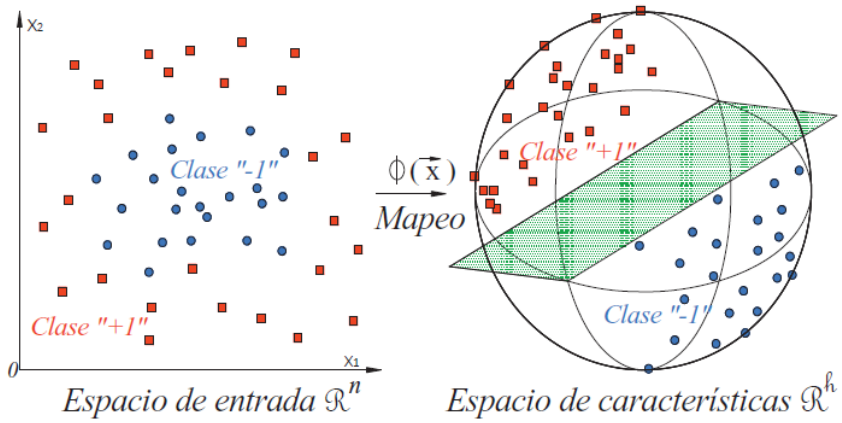
**Anexo 3.** Ejemplo de algoritmo KNN



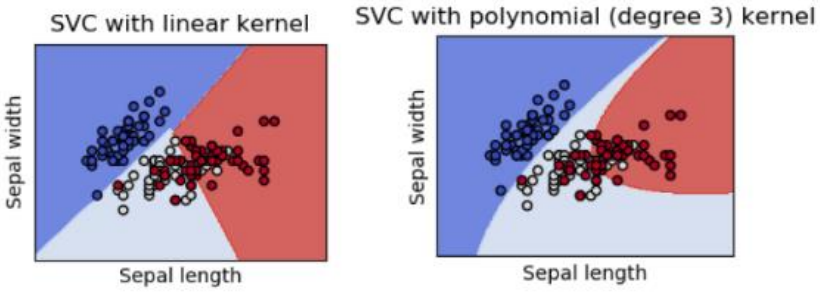

**Anexo 4.** Ejemplo de algoritmo SVM



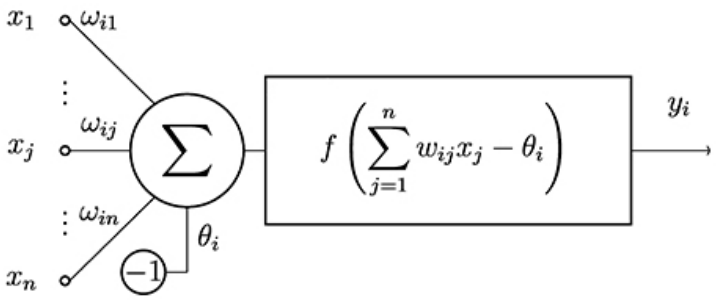

**Anexo 5.** Mapeo de datos de SVM a un espacio de mayor dimensión utilizando la función kernel



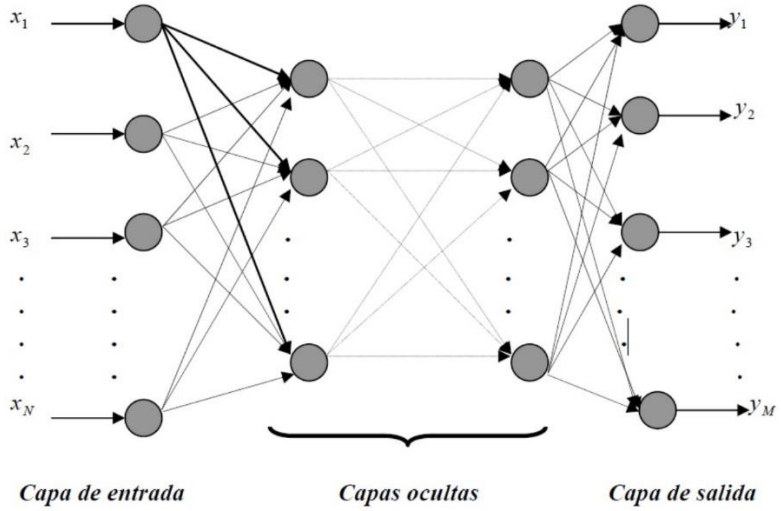
**Anexo 6.** Ejemplo de representación de diferentes tipos de kernel en SVM



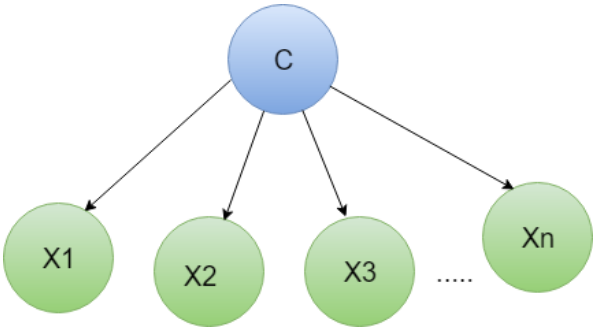

**Anexo 7.** Estructura de una neurona



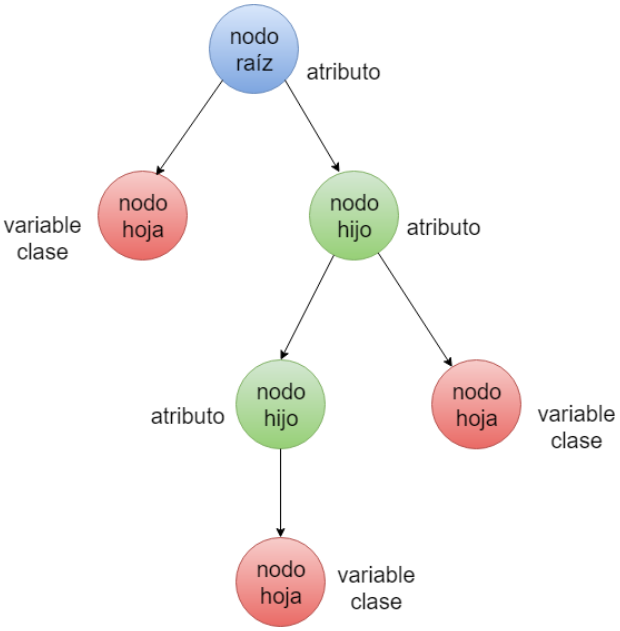

**Anexo 8.** Topología de un perceptrón multicapa



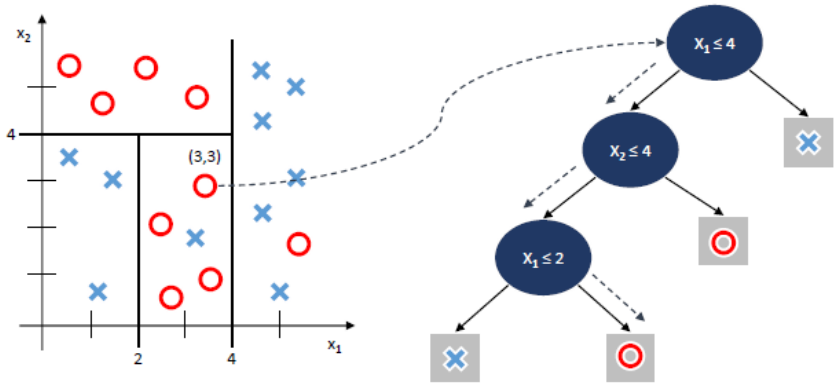
**Anexo 9.** Estructura de un modelo Naive Bayes



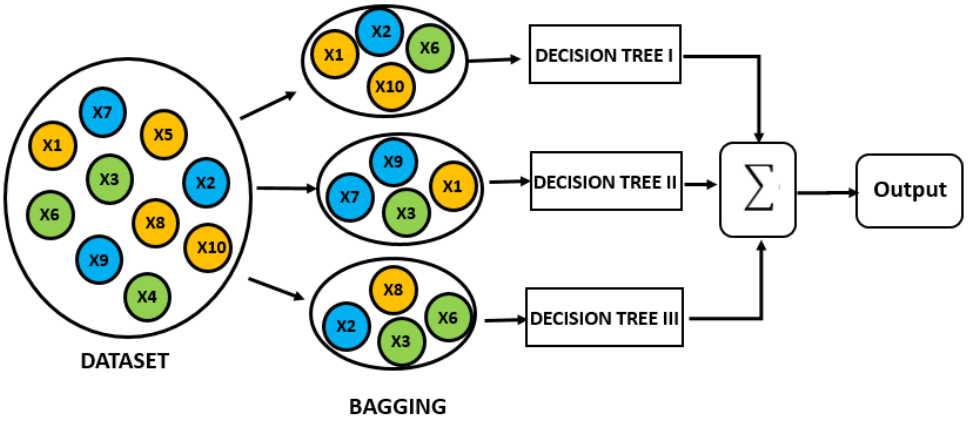

**Anexo 10.** Estructura de un árbol de decisión



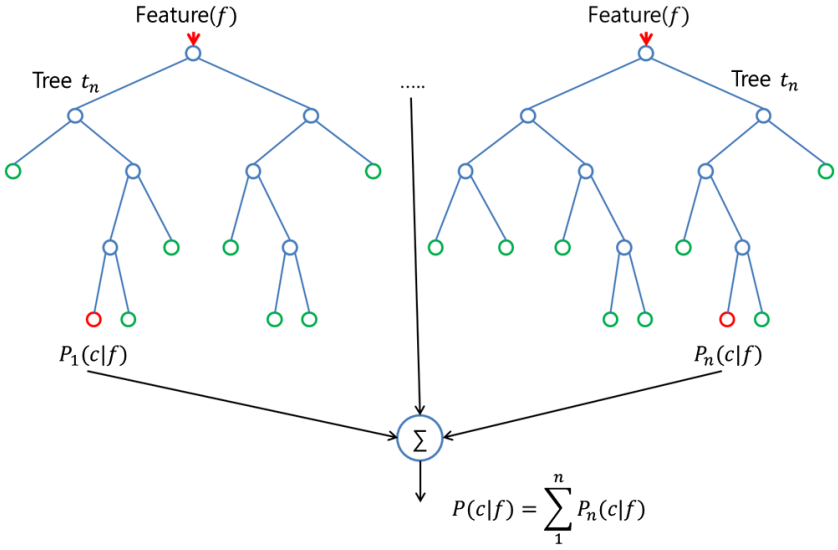

**Anexo 11.** Problema de clasificación con su respectiva representación mediante el clasificador árbol de decisión



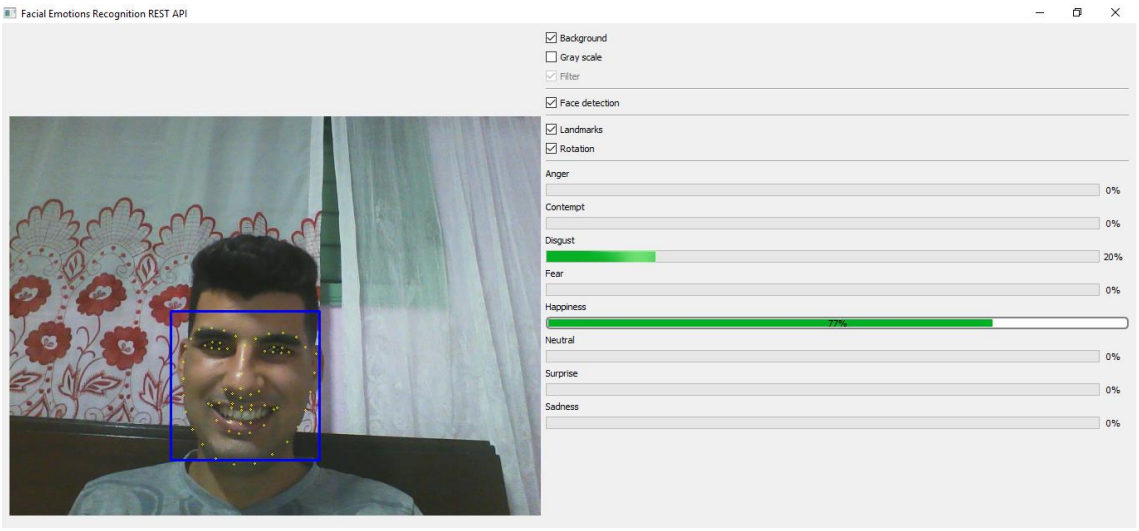
**Anexo 12.** Funcionamiento del clasificador Árbol Aleatorio




**Anexo 13.** Representación del clasificador Bosque Aleatorio



**Anexo 14.** Ejemplo con interfaz visual de FER REST API



**Anexo 15.** Fragmento de código utilizado para cargar la imagen, verificar el formato y existencia física de la misma. Se utiliza el método `imread` de OpenCV

```
def describe(imagePath):
    towork = cv2.imread(imagePath)
    if towork is None:
        print('Unrecognized format or not found file', imagePath)
        return None
```

**Anexo 16.** Utilización de método `cvtColor` de OpenCV que recibe la imagen y la convierte a escala de grises

```
towork = cv2.cvtColor(towork, cv2.COLOR_BGR2GRAY)
```

**Anexo 17.** Creación de un ecualizador adaptativo del histograma con contraste limitado mediante el método `createCLAHE` y su aplicación a una imagen en escala de grises mediante el método `apply`, ambos de la biblioteca OpenCV

```
clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8, 8))

towork = clahe.apply(towork)
```

**Anexo 18.** Detector facial de la biblioteca Dlib

```
detector = dlib.get_frontal_face_detector()
```

**Anexo 19.** Fragmento de código utilizado para realizar la detección del rostro a través de la imagen ecualizada y se escoge la primera detección

```
detections = detector(towork, 1)
if len(detections) == 0:
    if _verbose:
        print("No face detected in", imagePath, 'skipped')
    return None

rect = detections[0]
if len(detections) > 1:
    print(detections)
    print("More that one face detected in", imagePath)
    print('\tusing:', rect)
```

**Anexo 20.** Detector de marcas faciales de Dlib

```
shape = dlib.shape_predictor('../utils/shape_predictor_68_face_landmarks.dat')
```

**Anexo 21**. Detección de las 68 marcas faciales en la imagen procesada

```
s = shape(cut, dlib.rectangle(0, 0, 349, 349))
```

**Anexo 22.** Fragmento de código donde se guardan los vectores de rasgos de las imágenes del conjunto de datos, y se guardan las clases por cada emoción, para entrenar el clasificador.

```
for clase in os.scandir(dataSetPath):
    if not clase.is_dir():
        print('Ignoring no directory', clase.path)
    else:
        print('processing class:', clase.name)
        for sample in os.scandir(clase.path):
            print('processing images', sample.name)
            tmp = describe(sample.path)
            if tmp is not None:
                data.append(tmp)
                labels.append(clase.name)
```

**Anexo 23.** Implementación del modelo del clasificador Máquina de Soporte Vectorial con kernel polinomial implementado con Scikit-learn

```
clf = SVC(kernel='poly', probability=True, degree=1.6)
```

**Anexo 24.** Implementación del modelo del clasificador Máquina de Soporte Vectorial con kernel lineal implementado con Scikit-learn

```
clf = SVC(kernel='linear', probability=True)
```

**Anexo 25.** Implementación del modelo del clasificador K Vecinos más Cercanos implementado con Scikit-learn

```
clf = KNeighborsClassifier(n_neighbors=10, n_jobs=-1)
```

**Anexo 26.** Implementación del modelo del clasificador Naive Bayes implementado con Scikit-learn

```
clf = GaussianNB()
```

**Anexo 27.** Implementación del modelo del clasificador Red Neuronal Perceptrón Multicapa implementado con Scikit-learn

```
clf = MLPClassifier(hidden_layer_sizes=65)
```

**Anexo 28.** Implementación del modelo del clasificador Árbol de Decisión implementado con Scikit-learn

```
clf = DecisionTreeClassifier(criterion='entropy', max_leaf_nodes=5)
```

**Anexo 29.** Implementación del modelo del clasificador Bosque Aleatorio implementado con Scikit-learn

```
clf = RandomForestClassifier(n_estimators=30, criterion='entropy', n_jobs=-1)
```

**Anexo 30.** Método `fit` que posee cada clasificador de la biblioteca Scikit-learn para realizar el entrenamiento mediante los vectores de rasgos del conjunto de datos y las clases (emociones)

```
clf.fit(data, labels)
```

**Anexo 31.** Cargar archivo de clasificador entrenado mediante la biblioteca Pickle de Python.

```
with open('../models/classifier.dat', 'rb') as f:
    clf = pickle.load(f, fix_imports=False)
```

**Anexo 32.** Método `predict` de cada clasificador de la biblioteca Scikit-learn para reconocer la emoción a través del vector de rasgos

```
clf.predict([describe(imgPath)])[0]}
```

**Anexo 33.** Método `predict_proba` de cada clasificador de la biblioteca Scikit-learn para reconocer las probabilidades por emociones a través del vector de rasgos

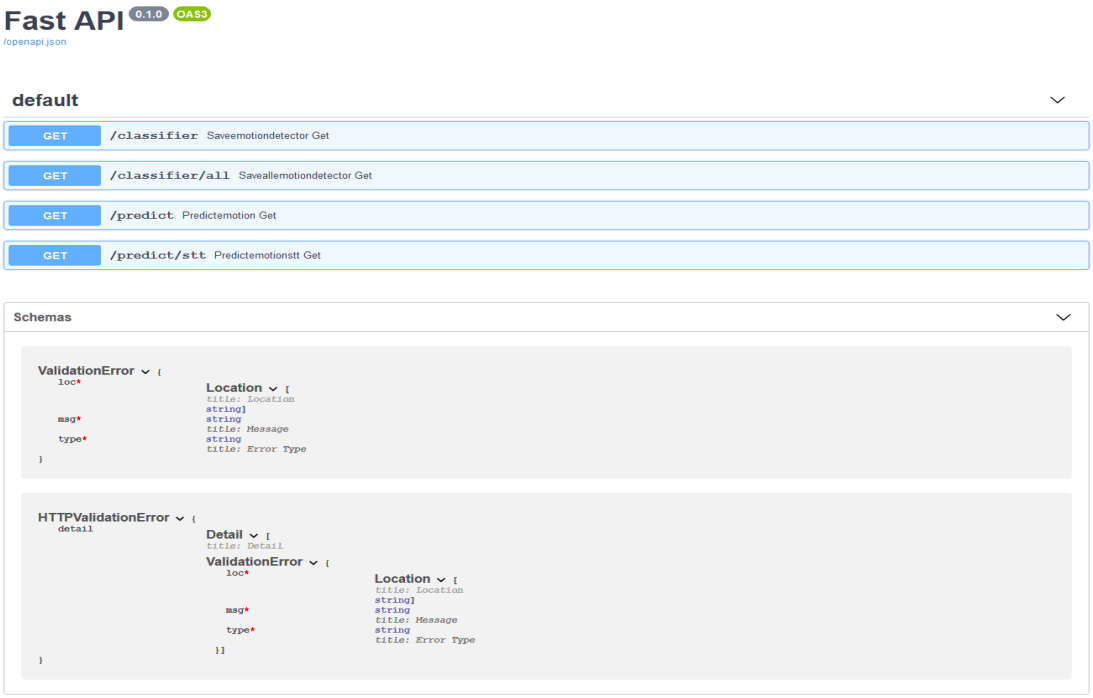
```
values = clf.predict_proba([describe(imgPath)])[0]
```

**Anexo 34.** Código del módulo setup.py para la instalación de FER REST API como una dependencia en el intérprete de Python

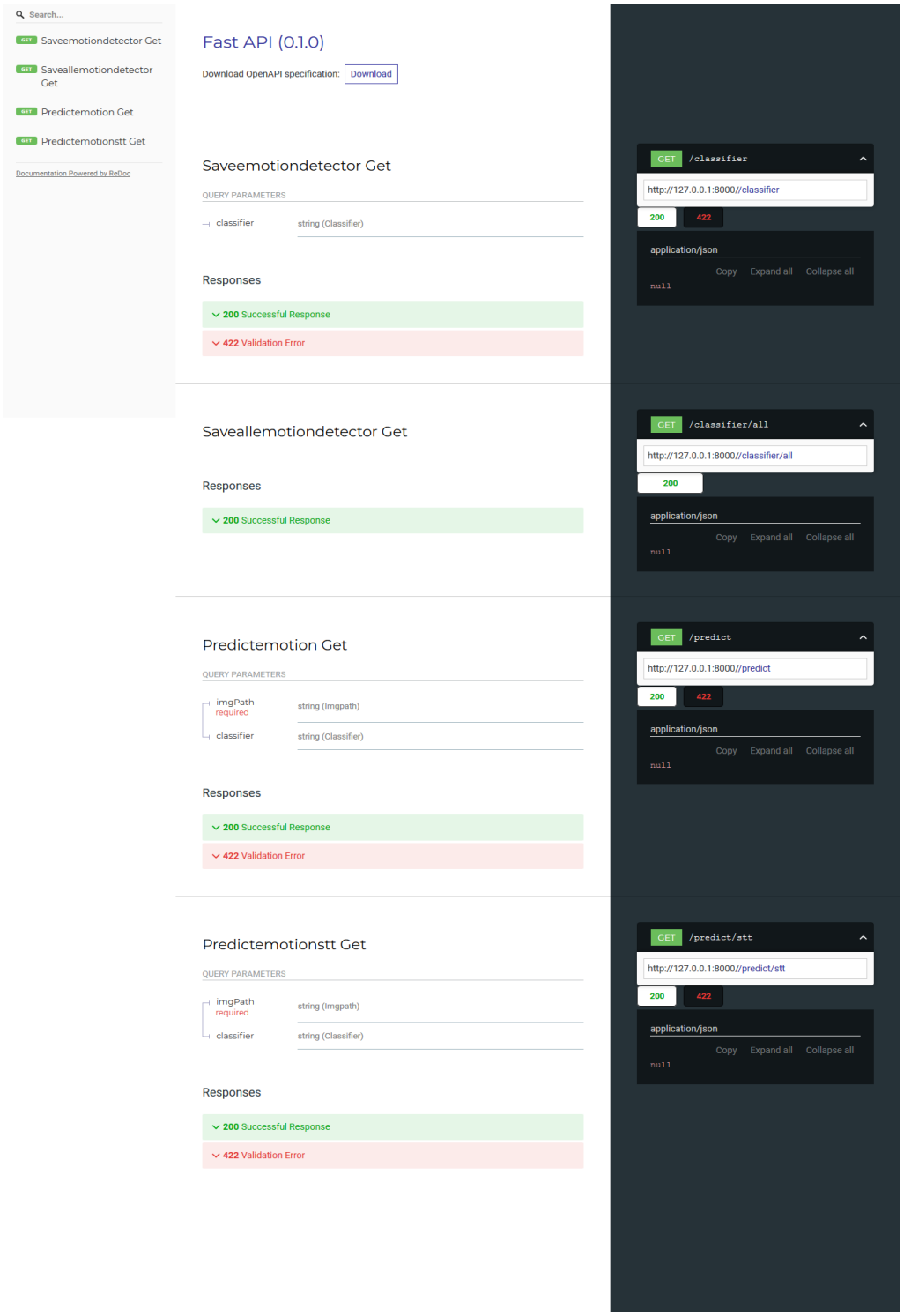
```
from distutils.core import setup

setup(name='Facial Emotions Recognition REST API',
      version='1.0.0',
      description='REST API for recognition of facial emotions contained in images',
      author='Dairo López Mollinedo',
      author_email='dlmollinedo@uclv.cu',
      packages=['fer_api'],
      )
```

**Anexo 35.** Documentación interactiva de FER REST API mediante FastAPI utilizando la herramienta Swagger



**Anexo 36.** Documentación estática de FER REST API mediante FastAPI utilizando la herramienta ReDoc



**Anexo 37.** Archivo JSON con la especificación de OpenAPI versión 3 de FER REST API

```
responses:
  200:
    description:                "Emotion detected correctly"
    content:
      application/json:
        schema:
          title:                "Emotion detected"
          description:          "Emotion detected and classifier used"
          required:
            0:                  "Classifier"
            1:                  "Emotion"
          properties:
            classifier:
              title:            "Classifier"
              description:      "Classifier used to recognize emotion"
              type:             "string"
              example:          "SVM kernel poly"
            Emotion:
              title:            "Emotion"
              description:      "Emotion detected"
              type:             "string"
              example:          "happy"
  404:
    description:                "Not Found"
    content:
      application/json:
        schema:
          title:                "Error Not Found"
          description:          "Error message when the service is not available, there is a problem in your URI"
          required:
            0:                  "detail"
          properties:


            detail:
              title:            "Error Not Found"
              description:      "Error message when the service is not available, there is a problem in your URI"
              type:             "string"
              example:          "Not Found"
  422:
    description:                "Validation Error"
    content:
      application/json:
        schema:
          $ref:                 "#/components/schemas/HTTPValidationError"
summary:                        "Recognize the emotion reflected in the human face through an image, using a previously trained classifier or by default"
operationId:                    "predictEmotion_predict_get"
parameters:
  0:
    required:                   true
    description:                "Path of the image"
    schema:
      title:                    "ImgPath"
      type:                     "string"
    name:                       "imgPath"
    in:                         "query"
  1:
    required:                   false
    description:                "Classifier chosen to recognize emotion, a previously trained one is chosen through the text strings that can be SVM for a Vector Support Machine with polynomial kernel, SVMLinear
                                a Vector Support Machine with linear kernel, NaiveBayes for the Gaussian Naive Bayes algorithm, MLP for a Multilayer Perceptron Neural Network, KNN for the nearest K Neighbor,
                                DecisionTree for a Decision Tree and RandomForest for the Random Forest classifier. If none is chosen, it is classified with an SVM with a trained polynomial kernel with the Cohn
                                Kanade data set"
    schema:
      title:                    "Classifier"
      type:                     "string"
    name:                       "classifier"
    in:                         "query"

/predict/stt:
  get:
    responses:
      200:
        description:            "Emotions correctly detected with their probabilities"
        content:
          application/json:
            schema:
              title:            "Emotions detected with probabilities"
              description:      "Emotions detected with their probabilities and classifier used"
              required:
                0:              "Classifier"
                1:              "Emotion"
              properties:
                classifier:
                  title:        "Classifier"
                  description:  "Classifier used to recognize emotion"
                  type:         "string"
                  example:      "SVM kernel poly"
                Emotion:
                  title:        "Emotion"
                  description:  "Probability corresponding to each emotion"
                  type:         "int"
                  example:      98
      404:
        description:            "Not Found"
        content:
          application/json:
            schema:
              title:            "Error Not Found"
              description:      "Error message when the service is not available, there is a problem in your URI"
              required:
```

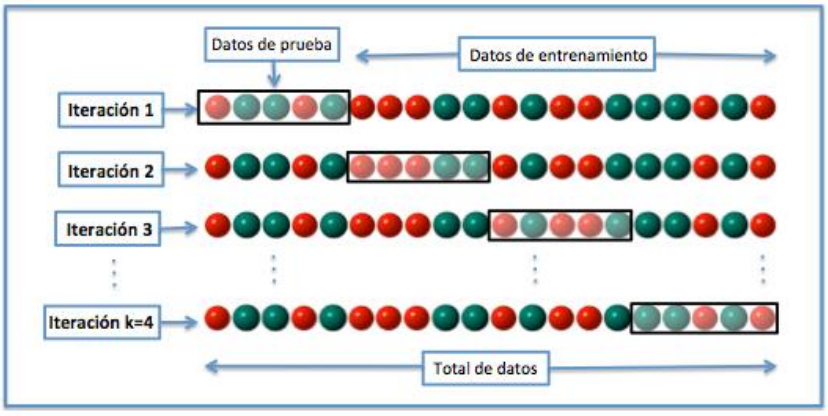
```
                0:                  "detail"
              properties:
                detail:
                  title:            "Error Not Found"
                  description:      "Error message when the service is not available, there is a problem in your URI"
                  type:             "string"
                  example:          "Not Found"
        422:
          description:              "Validation Error"
          content:
            application/json:
              schema:
                $ref:               "#/components/schemas/HTTPValidationError"
      summary:                      "Recognize emotions with probabilities reflected in the human face through an image, using a previously trained classifier or by default"
      operationId:                  "predictEmotionStt_predict_stt_get"
      parameters:
        0:
          required:                 true
          description:              "Path of the image"
          schema:
            title:                  "ImgPath"
            type:                   "string"
          name:                     "imgPath"
          in:                       "query"
        1:
          required:                 false
          description:              "Classifier chosen to recognize the probabilities of emotions, a previously trained one is chosen through the text strings that can be SVM for a Vector Support Machine with
                                    polynomial kernel, SVMLinear a Vector Support Machine with linear kernel, NaiveBayes for the Gaussian Naive algorithm Bayes, MLP for a Multilayer Perceptron Neural Network, KNN
                                    for the nearest K neighbor, DecisionTree for a Decision Tree and RandomForest for the Random Forest classifier. If none is chosen, it is classified with an SVM with a trained
                                    polynomial kernel with the set of Cohn Kanade data"
          schema:
            title:                  "Classifier"

            type:                   "string"
          name:                     "classifier"
          in:                       "query"
  components:
    schemas:
      ClassifierTrainedError:
        title:                      "Classifier error message"
        description:                "Error message if one of the existing classifiers in the API is not chosen"
        required:                   "message"
        properties:
          message:
            title:                  "Classifier error message to train"
            description:            "Error message if one of the existing classifiers in the API is not chosen"
            type:                   "string"
            example:                "Choose an existent classifier"
      ClassifierErrorFile:
        title:                      "Error message when saving classifier in file"
        description:                "Error message when saving classifier trained in file"
        required:                   "messageErrorFile"
        properties:
          messageErrorFile:
            title:                  "Error message when saving classifier trained in file"
            description:            "Error message when saving classifier trained in file when training a classifier or by the resource of all existing classifiers"
            type:                   "string"
            example:                "Error to put classifiers in files"
      ImageError:
        title:                      "Image error message"
        description:                "Error message of the format of the image or that does not exist"
        required:                   "messageErrorImage"
        properties:
          messageErrorImage:
            title:                  "Image error message"

            description:            "Error message of the format of the image or that does not exist"
            type:                   "string"
            example:                "The image does not exist or error with format"
      ValidationError:
        title:                      "ValidationError"
        required:
          0:                        "loc"
          1:                        "msg"
          2:                        "type"
        type:                       "object"
        properties:
          loc:
            title:                  "Location"
            type:                   "array"
            items:
              type:                 "string"
          msg:
            title:                  "Message"
            type:                   "string"
          type:
            title:                  "Error Type"
            type:                   "string"
      HTTPValidationError:
        title:                      "HTTPValidationError"
        type:                       "object"
        properties:
          detail:
            title:                  "Detail"
            type:                   "array"
            items:
              $ref:                 "#/components/schemas/ValidationError"
```

**Anexo 38.** Composición y emociones representadas en conjunto de datos Cohn Kanade

**Anexo 39.** Validación cruzada de K iteraciones




**Anexo 40.** Composición y emociones representadas en conjunto de datos KDEF

**Anexo 41.** Archivo de documentación README.md

# Facial Emotions Recognition REST API

## *FER REST API*

- Autor: Dairo López Mollinedo
- Carrera: Ingeniería Informática
- Año: 2019

---

Tutores:

- Lic.Claudia Rodríguez Rodríguez
- Ing.Roberto Vicente Rodríguez
- Departamento: Analíticas del Aprendizaje
- Facultad de Matemática, Física y Computación
- Universidad Central "Marta Abreu" de Las Villas

---

FER REST API permite a los desarrolladores de softwares incluir la detección automática de emociones faciales a través de imágenes frontales, en aplicaciones implementadas en cualquier lenguaje de programación y sistema operativo. La API reconoce las emociones universales planteadas por Paul Ekman ira, felicidad, tristeza, asco, sorpresa, miedo más la neutral. Está implementada en el lenguaje de programación Python y realiza las respuestas en el formato JSON soportado por todos los lenguajes de programación.

# Bibliotecas y framework requeridos

- OpenCV para el tratamiento, realce y normalización de las imágenes.
- Dlib para la detección del rostro y las marcas faciales para la extracción de características en el rostro humano representado en la imagen.
- Scikit-learn es una biblioteca de aprendizaje automático que posee los clasificadores supervisados de aprendizaje automático implementados en la API para clasificar las emociones.
- Numpy para las matrices científicas ndarray que son requeridas por las distintas bibliotecas OpenCV,Dlib y Scikit-learn y en especial para el vector de características.
- Framework FastAPI utilizado para la implementación de la API.
- Uvicorn servidor ASGI interno del framework FastAPI.s
- PyQt5 que es el enlace (binding) de Python para la biblioteca Qt. Este permite realizar interfaces gráficas de usuario (GUI) desde el lenguaje de programación Python.

# Características fundamentales en el funcionamiento de la API

- Con OpenCV se carga la imagen y se transforma a escala de grises.Se utiliza para normalizar la iluminación en la imagen un ecualizador adaptativo de histograma mediante el método createCLAHE. También se utiliza para llevar a un tamaño de 350X350 para agregar invariancia ante la proximidad del rostro a la cámara.
- Con Dlib se utiliza el método get_frontal_face_detector como detector facial. Este método utiliza el algoritmo de histograma de orientación de gradientes (HOG), combinado con un clasificador lineal, y el método de detección de imagen piramidal y ventana deslizante,y devuelve el rectángulo en el que está contenido el rostro.Además, se utiliza Dlib para detectar las marcas faciales con el método shape_predictor se carga el archivo shape_predictor_68_face_landmarks.dat, que permite detectar las 68 marcas faciales reflejadas en el rostro.
- La API REST posee los clasificadores supervisados Máquina de Soporte Vectorial (SVM) con kernel lineal y polinomial, Redes Neuronales Perceptrón Multicapa (MLP), Bosque Aleatorio (RandomForest),Árbol de Decisión (DecisionTree), Naive Bayes ( NaiveBayes) y K Vecinos más Cercanos (KNN).
- Posee un demostración implementada en PyQt5 y con las facilidades que brinda OpenCV para ver el funcionamiento de la biblioteca desde un entorno visual mediante video.
- La base de datos de emociones debe estar ubicada en el directorio dataset organizadas en directorios con el nombre de cada emoción que representa y sus imágenes correspondientes.
- La base de datos de emociones utilizada para entrenar los clasificadores supervisados por defecto de la API es Cohn Kanade y reconoce las emociones unviersales ira, asco, alegría, tristeza, miedo, sorpresa más la neutral y desprecio. FER REST API

  permite entrenar clasificadores personalizados que reconozcan emociones determinadas, para ello depende de la etapa de entrenamiento y la composición del conjunto de datos que se utilice. De esta forma, la API brinda al programador la posibilidad de escoger que emociones se van a detectar y como presentarlas en las aplicaciones según sus necesidades.
- FER REST API es multiplataforma y de fácil manejo e instalación. Posee una documentación estática e interactiva.

# Formatos de imagen soportados por la API

FER REST API soporta imágenes frontales en los siguientes formatos:

- Windows bitmaps (*.bmp,*.dib).
- JPEG (*.jpeg, *.jpg, *.jpe).
- JPEG 2000 (*.jp2).
- Portable Network Graphics (*.png).
- WebP (*.webp).
- Formato portable de imagen (*.pbm, *.pgm, *.ppm *.pxm, *.pnm).
- PFM (*.pfm).
- Tramas Sun (*.sr, *.ras).
- TIFF (*.tiff,*.tif).
- Archivos de imagen OpenEXR (*.exr).
- Radiance HDR (*.hdr, *.pic).
- Tramas y vectores geoespaciales sustentados por GDAL.

# Funciones

- El método describe devuelve el vector de rasgos del rostro de la imagen.
- El método saveEmotionDetector entrena los diferentes clasificadores al indicarle como parámetro, SVM para entrenar una Máquina de Soporte Vectorial con kernel polinomial,KNN para K Vecinos más Cercanos,NaiveBayes,MLP para entrenar una Red Neuronal Perceptrón Multicapa, RandomForest para Bosque Aleatorio,DecisionTree para un Árbol de Decisión y SVMLinear para una Máquina de Soporte Vectorial con kernel lineal, si no se indica ningún clasificador lanza un mensaje de error.
- El método saveAllEmotionDetector entrena todos los clasificadores existentes.
- El método predictEmotion devuelve la emoción indicando la ruta de la imagen por parámetro y un clasificador previamente entrenado, si no escoge ningún clasificador se clasifica con uno por defecto, una SVM con kernel polinomial entrenado con la base de datos de emociones Cohn Kanade con un acierto de 83 %.
- El método predictEmotionStt devuelve las emociones con su respectivas probabibilidades indicando la ruta de la imagen y un clasificador previamente entrenado, sino se indica se clasifica con el de por defecto.

---

- El recurso GET para entrenar un clasificador tiene URL http://127.0.0.1:8000/classifier?classifier=classifier y hace uso del método saveEmotionDetector, donde /classifier es la dirección del recurso y el párametro classifier indica el clasificador a entrenar, si no se escoge ningún clasificador se lanza un mensaje de error. Ejemplo del formato JSON de respuesta del recurso GET /classifier

{

```
"Training classifier Multilayer Perceptron ..."
```

} Ejemplo del formato JSON de respuesta del recurso GET /classifier mensaje de error {

```
"Choose an existent classifier"
```

}

- El recurso GET para reconocer la emoción correspondiente en la imagen tiene URL http://127.0.0.1:8000/predict?imgPath=img&classifier=classifier y hace uso del método predictEmotion, donde /predict es la dirección del recurso y el párametro classifier indica el clasificador entrenado previamente a utilizar e imgPath la dirección de la imagen. Si no se indica ningún clasificador utiliza el de por defecto.

---

Ejemplo del formato JSON de respuesta del recurso GET /predict

{

```
"Classifier": "SVM kernel poly",
"Emotion": "happy"
```

}

Ejemplo del formato JSON de respuesta del recurso GET /predict cuando existe problema con el formato de la imagen {

```
"The image does not exist or error with format"
```

}

- El recurso GET para entrenar todos los clasificadores existentes tiene URL http://127.0.0.1:8000/classifier/all donde /classifier/all es la dirección del recurso y hace uso del método saveAllEmotionDetector.

---

Ejemplo del formato JSON de respuesta del recurso GET /classifier/all {

```
"All classifiers were trained correctly"
```

}

- El recurso GET para reconocer las emociones con sus probabilidades respectivas en la imagen tiene URL http://127.0.0.1:8000/predict/stt?imgPath=img&classifier=classifier y hace uso del método predictEmotionStt, donde /predict/stt es la dirección del recurso y el párametro classifier indica el clasificador previamente entrenado e imgPath la dirección de la imagen.

Ejemplo del formato JSON de respuesta del recurso GET /predict/stt

{

```
"Classifier": "SVM kernel poly",
"Anger": 3,
"Contempt": 3,
"Disgust": 7,
"Fear": 9,
"Happiness": 4,
"Neutral": 16,
"Sadness": 6,
"Surprise": 48
```

}

Ejemplo del formato JSON de respuesta del recurso GET /predict/stt cuando existe problema con el formato de la imagen {

```
"The image does not exist or error with format"
```

# Documentación API

- La API brinda una documentación utilizando el framework FastAPI, está es proporcionada por la iniciativa OpenAPI con algunas de sus herramientas. Con la dirección URL http://127.0.0.1:8000/docs se brinda la documentación interactiva con Swagger UI y con http://127.0.0.1:8000/redoc se brinda la documentación con ReDoc.Además se brinda el JSON de la API basado en la especificación OpenAPI 3.0.2 para brindarle una documentación a los desarrolladores. Posee un manual de usuario en los formatos eBook de Windows (.exe), pdf y en un archivo de ayuda de HTML compilado (.chm).

## Instalación

Es compatible con la versión de Python 3.6.x es recomendable instalar la versión 3.6.8. Puede ser para arquitectura de 32 o 64 bits y en cualquier sistema operativo. Se instalan las siguientes dependencias con sus respectivas versiones, mediante los comandos siguientes basados en paquetes whl alojados en el repositorio Python Package Index (PyPI), accediendo a la línea de comandos del sistema operativo de la siguiente manera:

- pip install numpy==1.16.0
- pip install scikit-learn==0.20.2
- pip install opencv-python==4.0.0.21s
- pip install opencv-contrib-python==4.0.0.21
- pip install dlib==19.8.1
- pip install PyQt5-sip==4.19.13
- pip install PyQt5==5.11.3
- pip install fastapi==0.7.1
- pip install uvicorn==0.5.2

FER REST API posee un archivo requirements.txt en el cual se encuentran todas las dependencias necesarias para instalar. El comando pip permite gestionar listas de paquetes a través de un archivo de requisitos con los números de versión correspondientes. Y mediante el comando siguiente son instaladas los paquetes de manera automática:

- pip install -r requirements.txt

Esta vía de instalación es homogénea desde los sistemas operativos Windows, Linux y Mac OS.

Para instalar FER REST API, se copia su código fuente para cualquier ubicación de la computadora en la que se implemente la aplicación que va a hacer uso de la misma. Debe estar instalado el intérprete de Python versión 3.6.x y todas las dependencias con sus versiones correspondientes. Luego el programador podrá ejecutar la API y consumir sus métodos. FER REST API también permite su instalación desde el intérprete de Python, para ello se necesita estar ubicado en el directorio de la API y ejecutar el siguiente comando en un terminal:

- python3 setup.py install

De esta forma se permite utilizar la API desde el código fuente de Python como una dependencia instalada en el intérprete para aplicaciones en ese lenguaje de programación. Luego se puede importar el paquete fer_api en el código fuente del proyecto.

# Ejecución

La API se ejecuta en cualquier sistema operativo Linux, Windows o Mac OS cumpliendo con todos los requisitos de instalación del intérprete de Python y las dependencias. FER REST API se ejecuta mediante el siguiente comando desde la línea de comandos del sistema operativo, ubicado en el directorio de la API nombrado fer_api:

- uvicorn api:fer_rest_api