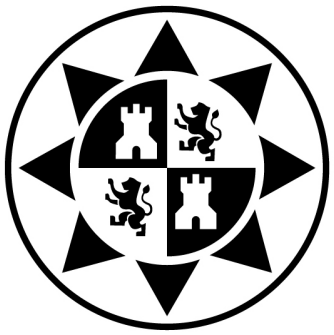


Universidad Politécnica de Cartagena

Trabajo Fin de Estudios

Aplicación de análisis de pacientes odontológicos mediante algoritmos de Machine Learning

Escuela Técnica Superior Ingeniería de Telecomunicación

Autor

Diego Ismael Antolinos García

Director

D. Esteban Egea López

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO

1

INTRODUCCIÓN

En esta sección abordaremos los diferentes objetivos marcados en este Trabajo Fin de Estudios y cuál es el planteamiento propuesto para solucionar los problemas que se comentarán a continuación.

Quizás cuando escuchamos a alguien hablar de *Machine Learning* se nos venga a la cabeza la imagen de un robot, o cualquier fantasía futurística. Pero no, de hecho el *Machine Learning* lleva décadas implementado en algunas aplicaciones como puede ser un filtro de spam, que lleva entre nosotros desde la década de los noventa, aplicaciones de reconocimiento de voz o el reconocimiento óptico de caracteres (OCR).

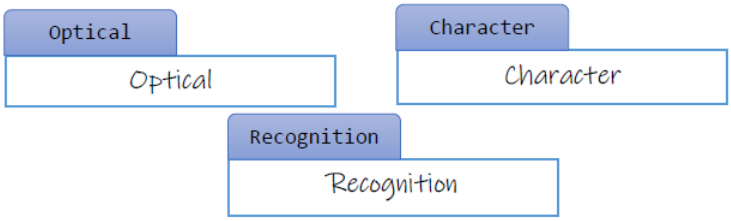


Figura 1.1: OCR

Podemos decir que el *Machine Learning* es una disciplina de la inteligencia artificial en el que un dispositivo es dotado con la capacidad de aprender, para ello hace uso de una serie de algoritmos, siendo capaz de realizar predicciones y clasificaciones de diferentes elementos de forma automática[1].

Machine Learning es una buena solución/herramienta para problemas que requieren grandes listas de reglas o una gran cantidad de ajustes, ya que un algoritmo de aprendizaje puede simplificar el código y obtener mejores resultados que los que puede proporcionar un enfoque tradicional. También es capaz de trabajar con problemas complejos que utilizan grandes cantidades de datos y adaptarse a estos.

Esta disciplina se utiliza en una gran diversidad de sectores, como pueden ser en el económico, realizando un *forecast* de ventas para una empresa o un *clustering* de clientes. En el industrial, con la automatización de vehículos y robots e incluso en el sanitario, que es en el que nos vamos centrar, concretamente en el campo de la odontología.

1.1. Descripción del problema

Para este proyecto se dispone de un conjunto de datos de pacientes odontológicos que han sido recopilados en una clínica desde el año 2006 y que están siendo utilizados para un estudio adicional en la Universidad de Murcia (UMU).

Dos de los grandes problemas a los que se enfrenta la sanidad es la dectección tardía de enfermedades y los diagnósticos confusos, debido a esto se pretende aprovechar este conjunto de datos para comprobar si se pueden utilizar técnicas de *Machine Learning* para predecir ciertas características de los pacientes. Esto permitiría clasificar pacientes en riesgo y desarrollar programas de salud preventivos, además de ayudar al propio especialista en el plan de tratamiento.

1.2. Objetivos

El objetivo principal es conseguir solventar los problemas anteriormente mencionados. Esto genera la necesidad de llegar a una solución que nos proporcione una predicción capaz de facilitar la labor del especialista para realizar un diagnóstico, o que incluso el propio paciente pueda ser capaz de realizar un autodiagnóstico de forma sencilla. Por tanto, nuestro objetivo principal se desglosa en los siguientes puntos:

- Realizar una predicción de una propiedad a partir de un conjunto de características de un paciente.
- Permitir al especialista la exploración y evaluación de diferentes enfoques de predicción de forma de sencilla, de manera que este pueda aplicar diferentes clasificadores y combinar las características que desee.

- Mostrar información de los diferentes pacientes que pueda ser de utilidad para elaborar un diagnóstico.

1.3. Planteamiento de solución

La solución propuesta consiste en utilizar la librería sckit-learn de Python, para realizar diferentes clasificadores, aprovechando que esta presenta una interfaz común que permite usar diferentes algoritmos. Además, desarrollar una aplicación web que facilite el uso de estos algoritmos a usuarios sin conocimientos en *Machine Learning*, de forma que la aplicación sea sencilla e intuitiva.

La aplicación web con la que se elaborarán los análisis de los pacientes contiene un formulario para seleccionar el atributo que hay que clasificar, cuales son las características del paciente que están ligadas al elemento a diagnosticar, y diferentes parámetros como el clasificador, método de imputación a usar, etc. Para el desarrollo web utilizamos HTML, CSS y JavaScript, y además, se hace uso Flask para poder integrar el código Python en nuestra apliación. Además, existe un apartado que contendrá la metainformación del dataset, y otra en la que se muestra la distribución de cada atributo de los pacientes.

CAPÍTULO

2

TECNOLOGÍAS UTILIZADAS

En este capítulo hablaremos sobre las tecnologías que se utilizan en la aplicación web, diferenciando la parte del cliente y la del servidor. Como se comenta en el capítulo 1, las tecnologías usadas son:

- Python
- HTML
- Flask
- CSS
- JavaScript

2.1. Servidor

2.1.1. Python

Python es un lenguaje de programación interpretado, es decir, presenta la necesidad de ser procesado previamente por un compilador. Además, utiliza librerías de código abierto, y presenta una creciente popularidad y aceptación en la actualidad. Se

caracteriza por ser un lenguaje sencillo y potente, ya que se consiguen realizar programas de alto nivel con pocas líneas de código, lo que supone una gran ventaja respecto a otros lenguajes.[2]

Dentro del lenguaje Python se han utilizado una serie de librerías que son las que permiten que se pueda realizar la clasificación de forma eficiente y rápida. Estas librerías son:

- Numpy
- Pandas
- Scikit-learn
- Flask

Numpy

Esta librería es una de las liberías más populares en la comunidad de programadores Python. Numpy es una librería que aporta arrays multidimensionales, objetos derivados (como matrices y arrays enmascarados), y un surtido de rutinas para operaciones rápidas con arrays, incluyendo matemáticas, lógica, manipulación de dimensiones, transformaciones de Fourier, algebra lineal básica, etc.[3]

Por tanto, permite trabajar de forma sencilla y versatil con arrays, facilitando el tratado de los datos.

Pandas

Pandas es otra de las librerías más conocidas y utilizadas en Python, y está orientada al tratado y análisis de la estructura de datos. Permite cargar nuestros datos de forma sencilla desde archivos Excel, CSV o bases de datos. También permite combinar conjuntos de datos diferentes y se basa en los arrays de la librería Numpy para generar nuevas estructuras de datos como son los dataframes. [4]

Scikit-learn

Construida sobre Numpy, Matplotlib y SciPy, Scikit-learn es la librería que permite dotar con esa 'inteligencia' a nuestro dispositivo gracias a la gran cantidad de algoritmos de machine learning que contiene (clasificación, regresión, clusterin, preprocesamiento, etc).[5]

Concretamente en la aplicación web se hace uso de algunos de los diferentes modelos de clasificación que hay desarrollados en esta librería (Kneighbors, Gaussian NB, LogisticRegression, Decision Tree y Random Forest).

K-Neighbors La clasificación de vecinos más cercanos (*Nearest Neighbors Classification*) es un tipo de aprendizaje que no intenta construir un modelo interno general, sino que guarda instancias de los datos de entrenamiento. La clasificación se obtiene por un voto de mayoría de los vecinos más cercanos a cada punto.[6]

K-Neighbors implementa el aprendizaje basado en los K vecinos más cercanos de cada punto, donde K es un valor entero especificado por el propio usuario. El valor óptimo de K depende, mayormente, del conjunto de datos. Una K mayor hace que haya una mayor supresión de ruido pero que los límites de la clasificación sean más parecidos.

GaussianNB Los métodos de Naive Bayes son un conjunto de algoritmos de aprendizaje supervisado que se basan en aplicar el teorema de Bayes. GaussianNB implementa el algoritmo Gaussian Naive Bayes para realizar la clasificación. En este caso se supone que la probabilidad de las características es gaussiana.[7]

LogisticRegression Regresión Logística (Logistic Regression) es un modelo de clasificación lineal, y se conoce también como clasificación de máxima entropía (*Maximum-entropy classification*). Puede ajustarse a regresión logística binaria, OnevsRest y regresión logística multinomial.[8]

Decision Tree Decision Tree es un método de aprendizaje supervisado no paramétrico usado para clasificación y regresión. La finalidad es crear un modelo que pueda predecir el valor de una variable objetivo aprendiendo reglas de decisión simples deducidas de las propias características de los datos.[9]

Random Forest En los arboles de decisión (Random Forest) cada árbol se extrae de una muestra que es extraída del conjunto de entrenamiento. Al dividirse cada nodo durante la construcción de un árbol, la mejor solución es encontrada entre todas las características de entrada o todos los subconjuntos aleatorios de tamaño *max_features*. Con esto se pretende conseguir disminuir la varianza del *forest estimator*.[10]

Flask

La librería Flask es la que permite renderizar nuestro código Python y poder integrarlo de forma sencilla en nuestras hojas HTML.

Otras

Caben destacar otras librerías como Bokeh, con la que se realizan gráficas interactivas de forma sencilla, incluyendo estas un panel de control que permite al propio usuario habilitar y deshabilitar funcionalidades [11]. Joblib, con la que se realiza la descarga y carga de elementos como un modelo de clasificación entrenado[12], por ejemplo. O IPython, con la que se convierte un dataframe a una tabla HTML de forma automática[13].

2.1.2. HTML

HTML (HyperText Markup Language) es un lenguaje de programación de etiquetas como bien nos indica su nombre y es otra de las tecnologías utilizadas en la apliación web. Con está tecnología se estructura toda la aplicación y gran parte de las funcionalidades que esta va a presentar.

2.1.3. Flask

Flask es un framework que incorpora su propio servidor HTTP, depende del motor de plantillas Jinja y se encuentra escrito en Python. Esto permite el desarrollo de aplicaciones web y la integración de código Python en ellas de forma sencilla. Además posee un debugger integrado, lo que facilita la detección de errores mientras nos encontramos desarrollando la aplicación. [14]

2.1.4. CSS

CSS (Cascading Style Sheets) es la tecnología que se utiliza para modificar la forma en la que se visualiza la aplicación en todo momento. Es decir, se encarga de la presentacion de las páginas Webs, incluyendo colores, tipo de fuente que se emplea y el diseño gráfico. También se encarga de la presentación en los diferentes tipos de dispositivos. Por tanto, el código CSS se encuentra ligado al código HTML que se ha desarrollado.

2.2. Cliente

2.2.1. JavaScript

JavaScript es un lenguaje de programación el cual se utiliza para proporcionarle dinamismo a una página web, por ello posee una integración completa con HTML y CSS. Los programas se escriben mediante scripts y es capaz de calcular, manipular y validar datos.[15]

CAPÍTULO

3

DESARROLLO DEL PROYECTO

En este capitulo se realiza una descripción de alto nivel del desarrollo del proyecto. Explicando el funcionamiento de las diferentes partes de la aplicación web desarrollada y como se relacionan las tecnologías utilizadas entre sí.

En la aplicación se aparece por defecto en la página Home, con la opción de cambiar entre las opciones situadas en la barra de navegación (Resumen estadístico, Documentación, Distribuciones y Clasificación de pacientes)

3.1. Flujo de las tecnologías utilizadas

El usuario realiza una petición al servidor desde un cliente haciendo uso de un navegador o interactuando con partes de aplicación web que tengan implementadas funcionalidades con JavaScript. El servidor está formado por un servidor HTTP, que recibe la petición del usuario y procede a la ejecución de las funciones y algoritmos correspondientes que toman los datos del dataset. Para esto se utilizan diferentes librerías, destacando scikit-learn, en la cual todos los clasificadores siguen la misma interfaz, lo que permite que se pueda seleccionar el modelo a aplicar. Una vez que se ha terminado de ejecutar las funciones y algoritmos se obtienen una serie de resultados que son renderizados en plantillas Jinja e implementados en HTML gracias a Flask. Finalmente, se le proporciona la respuesta deseada al cliente.

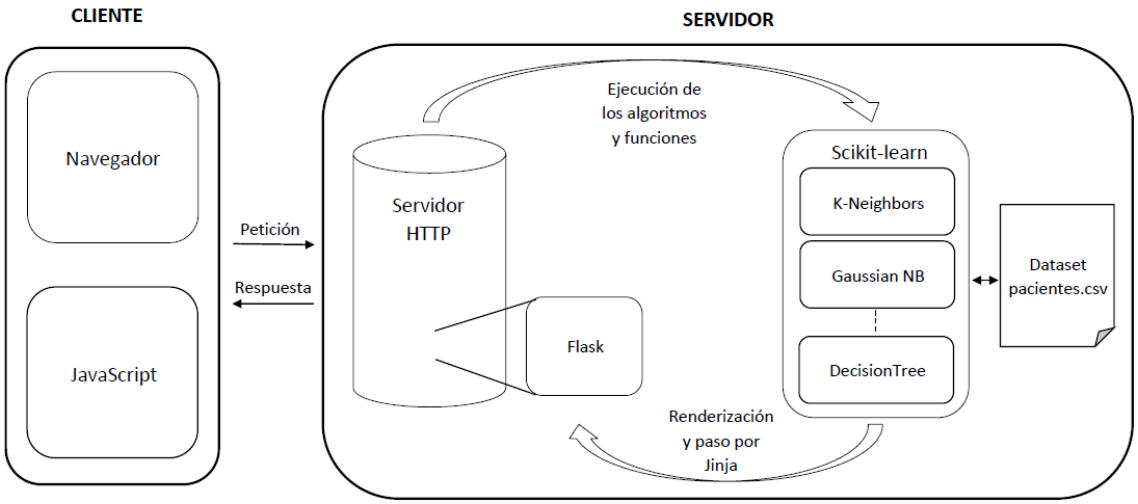


Figura 3.1: Representación del flujo entre las tecnologías implementadas

Un ejemplo de uso sería:

El usuario abre la aplicación en un navegador cualquiera (Google Chrome, Mozilla Firefox, etc) y quiere entrenar un clasificador para la característica *y* utilizando una lista de características X que ya ha introducido en un formulario inicial de la sección *Clasificación de pacientes*, entonces se le realiza una petición al servidor por parte del cliente, pidiendo entrar en la ruta '/classification/', Flask (que posee un servidor HTTP integrado) llama a todas las funciones y algoritmos, que se encuentran asociados a la ruta a la que quiere acceder el usuario, haciendo uso de las diferentes librerías de Python, entre ellas sckit-learn que contiene los diferentes algoritmos implementados, y el dataset *pacientes.csv*. Todos los modelos de clasificación de scikit-learn utilizan la misma interfaz, lo que permite la selección del clasificador que se desee aplicar. Una vez que se han terminado de ejecutar dichas funciones se renderizan en plantillas Jinja los resultados del entrenamiento, las tablas correspondientes y las diferentes variables que se necesitan para la visualización y el correcto funcionamiento de la página, en este caso para la página *Home*. Finalmente, esto pasa a implementarse en HTML gracias a Flask y se le muestra al usuario la página solicitada.

3.2. Datasets

Para el proyecto se utiliza un dataset llamado *pacientes.csv*. Este dataset contiene los datos de pacientes de una clínica odontológica desde el año 2006, con un total de 230 pacientes. En el dataset se recogen diferentes características, estas son:

ID del paciente, sexo del paciente, año de la primera sedación, edad de la primera sedación, remitido, diagnóstico principal del paciente, otras patologías de interés, número de sedaciones, tiempo desde la primera sedación a la última, motivo de la pri-

mera sedación, hábitos de higiene, presencia de caries, número de dientes con caries, si presenta placa, si presenta sarro, afectación pulpar, número de dientes con afectación pulpar, restos radiculares, número de dientes extraídos, si presenta ausencias, número de dientes ausentes, uso de antibiótico por patología oral, nombre del antibiótico, uso de analgésico por patología oral, nombré del analgésico, si presenta dificultad para comer por patología oral, obturaciones realizadas, dientes obturados, materiales utilizados para la obturación, si presenta una PPD, dientes con PPD, materiales utilizados parar PPD, pulpectomías, dientes con pulpectomías, materiales utilizados para pulpectomías, pulpotomías, dientes con pulpotomías, materiales utilizados para pulpotomías, endodoncias, dientes endodonciados, materiales utilizados para endodoncias, apicoformación, dientes tratados con apicoformación, materiales utilizados para la apicoformación, limpieza, RAR, selladores, dientes tratados con selladores, materiales utilizados para selladores, aplicación de flúor, extracciones, dientes extraídos, revisión post-sedación, seguimiento de prevención, necesidad de medicación por patología oral, si presenta mejoría a la hora de comer, presencia de placa, tratamientos realizados posteriormente sin sedación, comportamiento del paciente en las citas, motivación, año de la última revisión, causa de la segunda sedación, fracasos en segunda intervención, nueva patología/tratamientos, causa de tercera sedación, fracasos en tercera intervención, nueva patología/tratamientos, causa de cuarta sedación, fracasos en cuarta intervención, nueva patología/tratamientos, sedaciones reales, tratamientos realizados posteriormente sin sedación.1, tratamientos realizados posteriormente sin sedación.2 y el año de la primera visita a la consulta.

Además, se ha realizado otro dataset (*leyenda.csv*) que contiene aquellas características que son categóricas y la etiqueta correspondiente de los diferentes valores numéricos que puede poseer cada categoría. Este dataset se usa principalmente para la visualización de estas categorías en las diferentes tablas, sustituyendo el valor numérico que aparece en el primer dataset por su valor categórico.

3.3. Home

Se trata de la página principal de la aplicación de diagnóstico, en está página nos encontraremos con la información que se encuentra en el dataset de los pacientes odontológicos en forma de tabla. Existe la opción de mostrar 10, 25, 50 o 100 filas de la tabla, la opción de buscar contenido de esta y ordenar los datos en función de los valores de la columna que se seleccione como referencia.

Para mostrar correctamente los datos de los pacientes en la tabla utilizo una función desarrollada en Python y que se llama *leyenda_pacientes()*, la cual se encarga de sustituir los valores de determinadas columnas por sus valores correspondientes que se encuentran en un dataset denominado leyenda. Cada vez que el usuario entra en la ruta '/ ' , se ejecuta una función denominada inicio() que se encarga de llamar a *leyenda_pacientes()*, convertir dicha tabla ya a formato HTML y renderizarla para poder

incluirla en nuestra hoja HTML. Además, se utiliza un script de JavaScript para habilitar las funcionalidades de número de filas a mostrar, ordenar en función de los valores de una columna y búsqueda de datos de la tabla.

3.4. Resumen estadístico

Cada vez que el usuario pinche en Resumen estadístico, se le redireccionará a la ruta '/describe/', y cada vez que se llame a esta ruta se ejecutará la función *ejec_describe()*, la cual se encargará de aplicar otra función, llamada *describe()* de la librería pandas sobre nuestro dataset pacientes. Esta funcion *describe()* se encarga de obtener toda la metainformación (media, desviación tipica, mediana, valor máximo y mínimo de cada columna, cantidad de datos por columna, etc) del dataframe al que ha sido aplicado. Una vez que ha sido generado el nuevo dataframe con la metainformacion, se convierte a una tabla en formato HTML y se renderiza, tal y como ocurría en la página Home.

3.5. Documentación

Documentación redirecciona al usuario a la página en la que se encuentra a este documento que contiene toda la información necesaria sobre la aplicación.

3.6. Distribuciones

Distribuciones es la página en la que el usuario puede ver cual es la distribución de cada atributo de los pacientes. Para ello, existe un desplegable inicial en el que se selecciona cual es el atributo del que se quiere visualizar su distribución, posteriormente se hace uso de un botón inferior de Mostrar, seguidamente se muestra la distribución deseada.

3.6.1. Selección de atributo

Primero, cuando el usuario selecciona Resumen estadístico en la barra de opciones se le redirecciona a la ruta '/distribution_condition/', en la cual se ejecuta la función distribution_condition(), que a su vez ejecuta la función *leyenda_pacientes()* antes descrita, y obtendrá únicamente el valor de la cabecera de las columnas del dataframe resultante de dicha función. Estas corresponden con el nombre de los diferentes atributos que hay registrados en el dataset pacientes. Esto se convierte a una lista y se

ordenan los valores por orden alfabético para su correcta visualización, se guarda bajo el nombre de atributos. Finalmente, se realiza la renderización de atributos.

3.6.2. Gráfico de distribución

Una vez que se le da al botón Mostrar se redirecciona al usuario a la ruta '/hist_distribution/', en la que se ejecuta la función *ejec_distribution_feature()* que se encarga de obtener, a través del método GET, el valor que ha seleccionado el usuario en el desplegable anterior. Toma dicho valor y se usa como único parámetro de entrada de la función *hist_distribution()* que es la que se encarga de construir la gráfica de distribución, haciendo uso de la librería Bokeh, mencionada en el capítulo 2. El resultado se guarda en la variable graph, y se utiliza la función components(), que devuelve un *<script>* que contiene los datos para nuestra gráfica y proporciona un * * para la visualización de esta.

Además, se vuelve a construir el formulario de seleccion de atributo. Finalmente, se renderizan las componentes de la gráfica y la variable atributos.

3.7. Clasificación de pacientes

Esta sección es la que se encarga de realizar el diagnóstico del paciente utilizando algoritmos de clasificación. Cuando el usuario selecciona la opción Clasificación de pacientes en la aplicación web se le redirecciona a la ruta '/classification_condition/' en la que aparece un formulario inicial a rellenar para comenzar con el proceso de diagnóstico del paciente. Una vez rellenado y enviada la información del formulario, se entrena el modelo de clasificación y se muestran los resultados de dicho entrenamiento. Además, aparece otro formulario a rellenar con los atributos específicos del paciente y así poder clasificarlo. Finalmente, se envía la información del paciente y se muestran los resultados de la clasificación.

3.7.1. Formulario inicial

Se encuentra en la ruta '/classification_condition/', y la mayor parte de su contenido es producto de la ejecución de la función *classification_condition()*. El formulario consta de las siguientes partes:

- Clasificador
- Características

- Atributo a predecir
- Solución para problema de imputación

Clasificador

El apartado Clasificador está implementado directamente en HTML, y en este apartado el usuario se encarga de seleccionar el clasificador que quiere utilizar para realizar el diagnóstico. Todos los modelos de clasificadores de scikit-learn siguen la misma interfaz lo que permite que se pueda seleccionar el modelo deseado y que se aplique de forma sencilla.

Características

El apartado de características se trata de un *<select>* con opción de selección múltiple. Para que el usuario sea capaz de seleccionar todas las características (*features*) que vea convenientes para realizar la clasificación. Para obtener el valor de los diferentes atributos se vuelve a hacer uso de la funcion *leyenda_pacientes()* obteniendo el valor de todos los atributos posibles ordenados alfabéticamente.

Atributo a predecir

En esta sección del formulario se escoge cual es el atributo o característica del paciente que se pretende diagnosticar. Para ello, se guarda el valor del nombre de las columnas del dataset *leyenda*, que serán los únicos atributos diagnosticables por la aplicación.

Problema de imputación

En el dataset *pacientes* encontramos un problema de imputación, es decir, existen columnas que contienen valores faltantes (NaN) y el clasificador no acepta un NaN como valor de entrada (input). Por tanto, se debe realizar una modificación de esos valores, es decir, se utiliza una técnica de imputación. Existen 4 posibilidades y el usuario puede escoger entre usar una de ellas. Esas opciones son sustituir los valores faltantes por la media, la mediana o el valor más frecuente de esa columna, o bien, eliminar todos los valores faltantes. Se encuentra implementada completamente en HTML sin necesidad de hacer uso de ninguna función desarrollada en Python, al igual que ocurría con las opciones de *Clasificador*

3.7.2. Entrenamiento clasificador

Una vez rellenado el formulario inicial y que se ha pulsado el botón *Entrenar modelo* el usuario es redireccionado a la ruta '/classification/' ejecutándose la función *ejec_classification()*. Esta función se encarga de obtener los resultados del formulario antes rellenado por el usuario. Guardando los resultados en las siguientes variables:

- **feat_list**: lista que contiene todas la *features* del paciente que se han seleccionado como input.
- **feat_pred**: *feature* de la que se va a realizar el diagnóstico.
- **strategy**: solución escogida para el problema de imputación.
- **classificator**: clasificador escogido para realizar el diagnóstico.

Posteriormente se hace uso de la funcion *classification_model()* que usa como parametros de entrada el dataset *pacientes*, y las variables *feat_list*, *feat_pred*, *strategy* y *classificator*. Esta función es la que se va a encargar de implementar el clasificador elegido, utilizar el imputador seleccionado, entrenar el modelo y obtener varios parámetros que resultan de gran utilidad por la información que aportan.

La función primero comprueba cual es la solución para el problema de imputación que se quiere utilizar, si se escoge las opciones de Media, Mediana o Valor más frecuente se procede de la siguiente manera:

Primero, se crea un dataframe llamado *paciente_df* que contiene las columnas del dataframe *pacientes* correspondientes a las *features* seleccionadas en el formulario inicial y se comprueba de que tipo son datos que contiene cada columna ya que si la columna es de tipo *object* el clasificador no puede trabajar con ella y tampoco se puede solventar el problema de imputación. Se realiza una codificación de las columnas que sean de dicho tipo, usando *LabelEncoder()* para ello. Esta función se encarga de codificar las features categóricas a numéricas.

Una vez que todas las features correspondientes han sido codificadas se procede a solventar el problema de imputación mediante la función *SimpleImputer()* que usa la variable *strategy* como parámetro de entrada, cambiando los valores faltantes de las columnas de *paciente_df* según la estrategia que haya seleccionado el usuario anteriormente (valor de *strategy*).

Los datos ya se encuentran tratados correctamente para poder separarlos en un set de entrenamiento (*X_train* , *y_train*) y un set de testeo (*X_test* , *y_test*) utilizando la función *train_test_split()* que realiza esto de forma automática.

Por el contrario, si la estrategia de imputación escogida es la de eliminar los datos faltantes, el mecanismo sería muy similar. Se crea el dataframe *pacientes_df* al igual que se hace si se escoge cualquiera de las otras tres estrategias de imputación, se crea un nuevo dataframe llamado *X* que contiene únicamente las columnas cuyos nombres se encuentran en *feat_list* de *paciente_df* y se codifican aquellas columnas que poseen valores categóricos *LabelEncoder()*. Cada columna codificada se guarda en una nueva columna del dataframe *X*, y al terminar de codificar se crea *y* que contiene la columna de *paciente_df* cuyo nombre es igual a *feat_pred* . Posteriormente eliminamos todos aquellos valores faltantes utilizando la función *dropna()*, y finamente se separa en set de entrenamiento y de testeo.

Una vez solventado el problema de imputación se procede a realizar el entrenamiento del clasificador, para esto se comprueba cual es el valor de la variable *classificator* que contiene el modelo escogido por el usuario, y se guarda bajo el nombre de *model*. Se utiliza la función *fit()*, a la que se le pasa *X_train* e *y_train* como parámetros de entrada para entrenar el modelo, se guarda en local mediante la función *dump()*. También se obtienen valores que indican la precisión con la que el modelo esta clasificando, estos valores son *acc_train* y *acc_test*(precisión de la clasificación realizada con el conjunto de datos *X_train* y *X_test* respectivamente).

Posteriormente, se realiza una clasificación con *X_test* como input y se obtienen las probabilidades de las clasificaciones de todo el conjunto. Una vez que se tienen estos datos se procede a realizar un dataframe, llamado *converter*, que contiene como se encuentra clasificado el paciente originalmente, la clasificación que realiza el modelo, la probabilidad, y los valores de las características seleccionadas como entrada que posee el paciente.

También se genera una matriz denominada matriz de confusión (*conf_matrix*), la cual se utiliza para saber que cantidad de aciertos ha tenido nuestro modelo a la hora de clasificar los diferentes pacientes del conjunto de datos.

Por último, esta función también se encarga de generar otro dataframe llamado *classifi_report* que contiene diferentes métricas del entrenamiento como son *f1_score*, *precision*, *recall* y *support*

- **Precision**: El valor de esta métrica nos indica la precisión de las predicciones positivas que ha realizado el clasificador.[1]

$$Precision = \frac{TP}{TP + FP} \tag{3.1}$$

  Donde *TP* es el numer de verdaderos positivos y *FP* es el número de falsos positivos.

- **Recall**: La métrica *recall*, también llamada sensibilidad o tasa de verdaderos positivos (*True Positive Rate*, TPR), es el ratio de instancias positivas que son detec-

tadas correctamente por el clasificador. [1]

$$Recall = \frac{TP}{TP + FN} \tag{3.2}$$

Donde FN es el número de falsos negativos.

- **F1_score**: Esta métrica corresponde a la media armónica de las métricas *precision* y *recall*. Mientras que la media regular trata todos los valores de forma equitativa, la media armónica le da un mayor peso a los valores más bajos. El valor de *f1_score* será alto si los valores de *precision* y *recall* son altos.[1]

$$F1\_score = \frac{2}{\frac{1}{precision} + \frac{1}{recall}} = 2 \times \frac{precision \times recall}{precision + recall} = \frac{TP}{TP + \frac{FN+FP}{2}} \tag{3.3}$$

- **Support**: La métrica *support* nos indica el número de apariciones de cada etiqueta en el conjunto que se ha utilizado.[1]

Los dataframes *converter*, *conf_matrix* y *classifi_report* también se guardan en local.

Una vez que se ha realizado todo esto, estos tres últimos dataframes se convierten en tablas HTML. Posteriormente y como último paso, se renderizan las diferentes variables utilizadas y dependiendo de si el clasificador escogido ha sido K-Neighbors o no, si este ha sido el seleccionado por el usuario también se renderizan las componentes de una gráfica que nos indica los valores de la precisión que tendría el modelo con diferentes valores de K posibles . Además, destacar también que se renderizan las variables *feat_list* y *leters_values* (esta contiene los valores posibles que el usuario puede escoger para rellenar el campo) que permiten crear el formulario de características del paciente de forma dinámica.

3.7.3. Clasificación del paciente

Una vez que el usuario rellena el formulario de características del paciente y pulsa el botón *Clasificar* se redirecciona a la ruta '/*prediction_classificator*/' en la que se ejecuta la función *predict_classification*. Esta función se encarga de obtener las características introducidas por el usuario en el formulario anterior y pasarlas como parámetro de entrada a la función *create_X* la cual se encarga de tratar los datos y convertirlos en un dataframe, denominado *X*, que sirve como parámetro de entrada de la función *prediction_class* junto con las variables *y*, *feat_list*, *first_nan*.

La función *prediction_class* se encarga de codificar aquellas características del paciente que sean categóricas, cargar el modelo entrenado utilizando la función *load()*, utilizar la función *predict()* para clasificar al paciente según los valores recogidos en el dataframe *X* y obtener también la probabilidad asociada a cada clasificación posible

del paciente con la función *predict_proba()*, estas probabilidades se guardan en la variable *prediction* (la probabilidad asociada a la clasificación del paciente se guarda en la variable *best_prediction*).

Posteriormente, se carga la matriz de confusión, que se había guardado en local anteriormente, y se utiliza junto con las variables *prediction* y *first_nan* para realizar la tabla de clasificación que contiene cual es el valor numérico de cada característica y la probabilidad asociada de cada una. Se carga también de forma local la tabla de las clasificaciones del conjunto de testeo (*converter*) y la tabla de las métricas obtenidas en el entrenamiento (*classifi_report*).

Finalmente, se convierten las tablas que se van a mostrar a formato HTML y se renderizan las diferentes variables que se utilizan para la creación de formularios, en este caso se quedan habilitados tanto el formulario inicial (selección de características, atributo a clasificar, clasificador, etc) como el formulario de características que presenta el paciente, con el fin de poder clasificar diferentes pacientes con otras características usando el clasificador que se ha entrenado anteriormente. Si se usa el primer formulario el clasificador ya será diferente, aunque se use el mismo tipo de clasificador, debido a que el entrenamiento de este cada vez será diferente al anterior.

Por tanto, el flujo general que abarca estas dos últimas secciones consiste en preprocesar si son necesarias las características seleccionadas por el usuario, si hay que crear, entrenar y guardar el modelo, y si hay que reutilizarlo con nuevos datos del usuario.

CAPÍTULO

4

RESULTADOS

Se ha conseguido desarrollar una aplicación web intuitiva, cuya apariencia es como se puede observar en las figuras 4.1, 4.2, 4.3 y 4.4 .

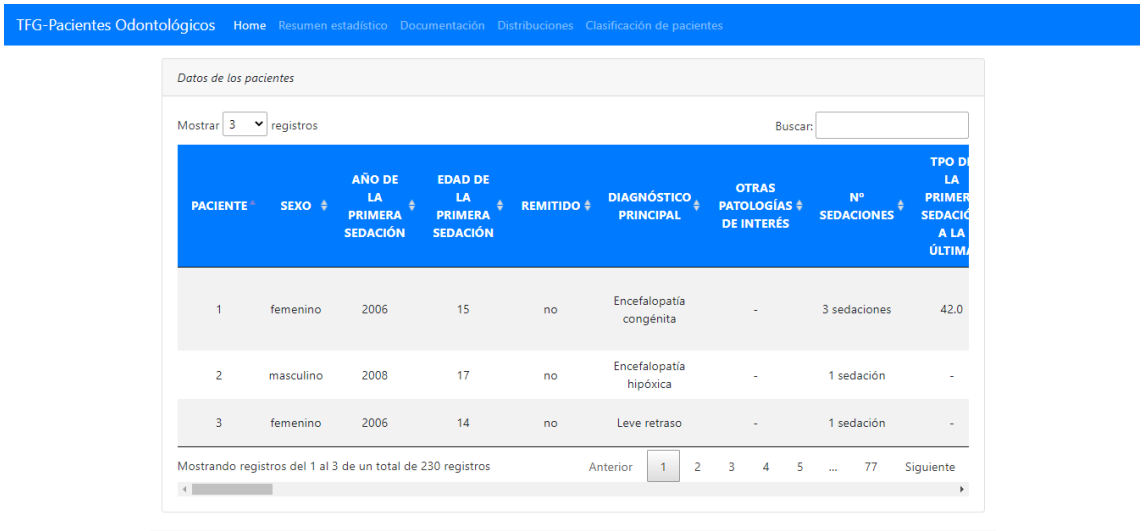


Figura 4.1: Home

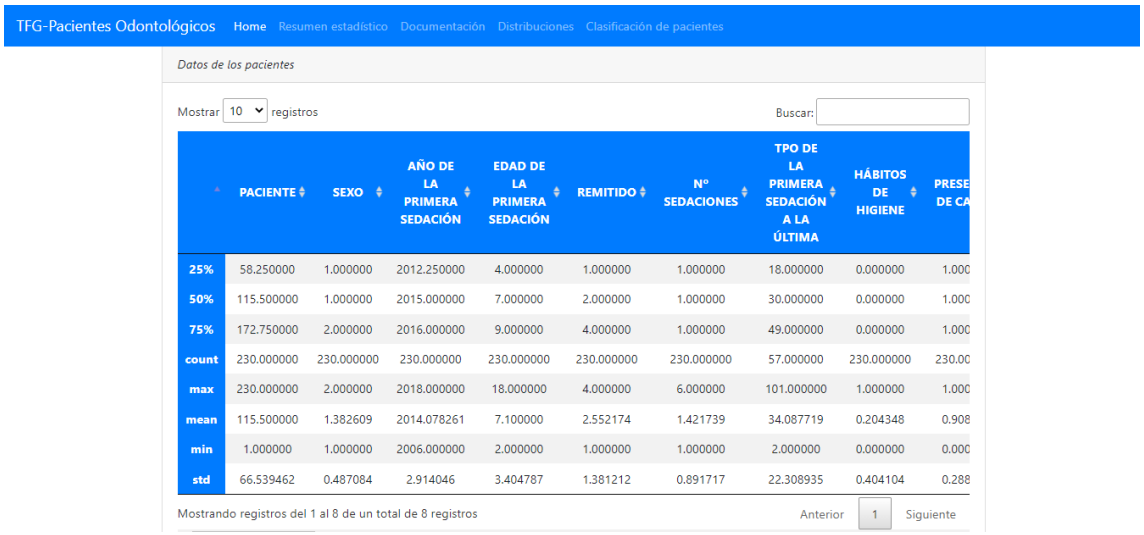


Figura 4.2: Resumen estadístico

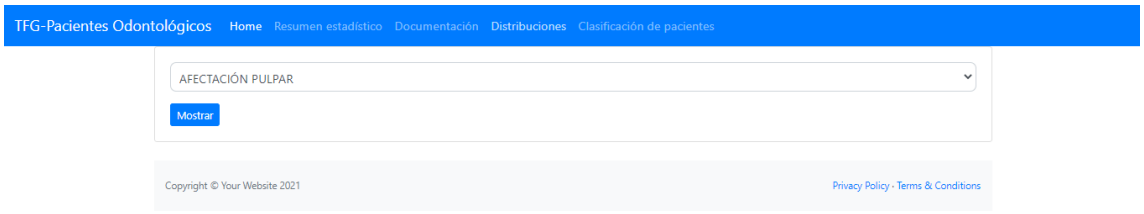


Figura 4.3: Distribuciones

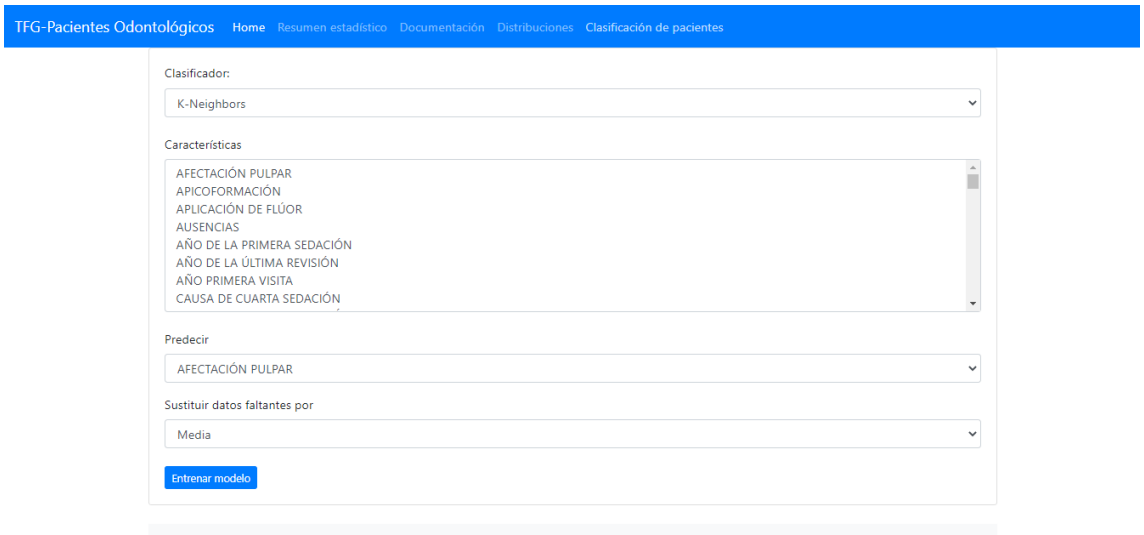


Figura 4.4: Formulario inicial de Clasificación de pacientes

4.1. Pasos para realizar una clasificación

Si se quiere realizar la clasificación de una característica de un paciente la forma de proceder es la siguiente:

1. Entrar en la página *Clasificación de pacientes*.
2. Escoger el clasificador deseado. Por ejemplo: K-Neighbors

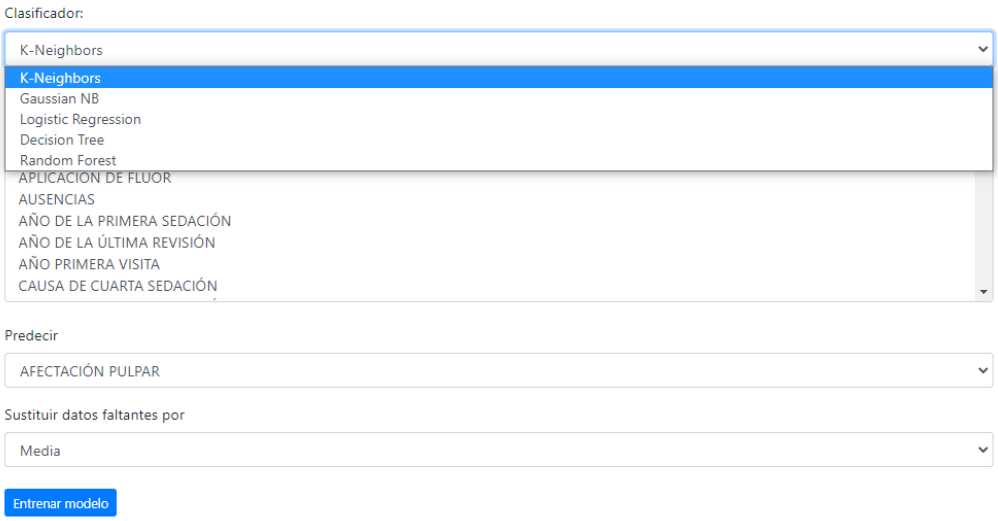


Figura 4.5: Selección de clasificador

3. Seleccionar las características del paciente que influyen en el atributo a predecir. Por ejemplo: DIAGNÓSTICO PRINCIPAL, EDAD DE LA PRIMERA SEDACIÓN, PLACA y NECESIDAD DE MEDICACIÓN POR PATOLOGÍA ORAL.

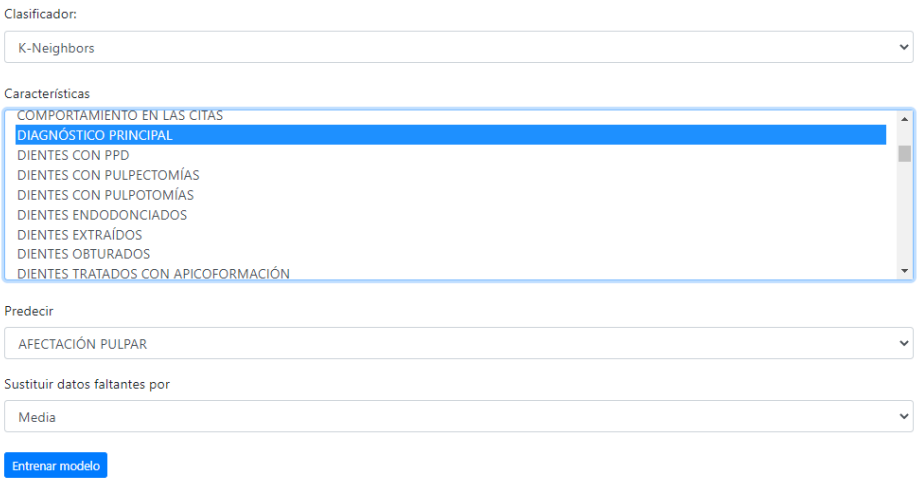


Figura 4.6: Selección de características del paciente

4. Elegir el atributo a predecir. Por ejemplo: PRESENCIA DE CARIES

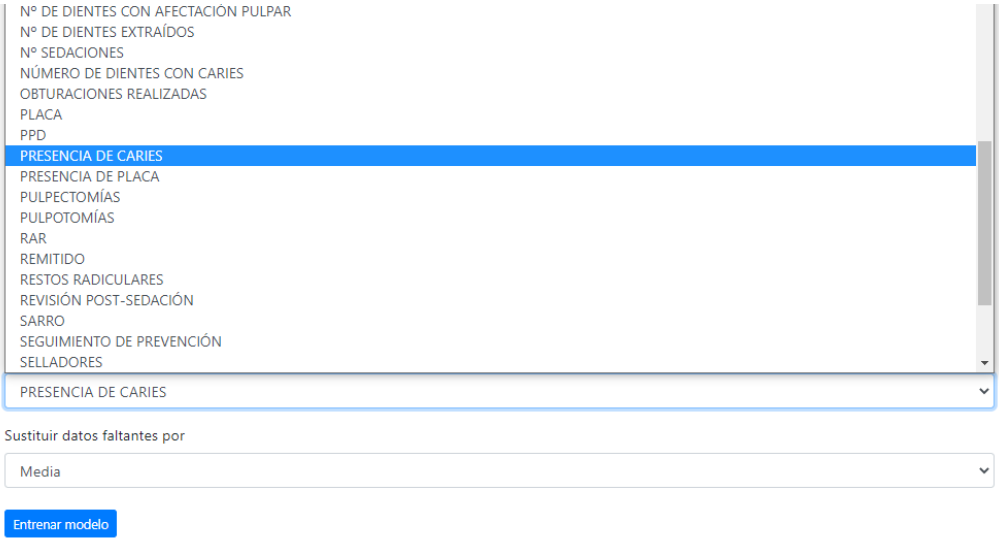


Figura 4.7: Selección de atributo a predecir

5. Elegir la solución que se quiere para el problema de imputación. Por ejemplo: Valor más frecuente.

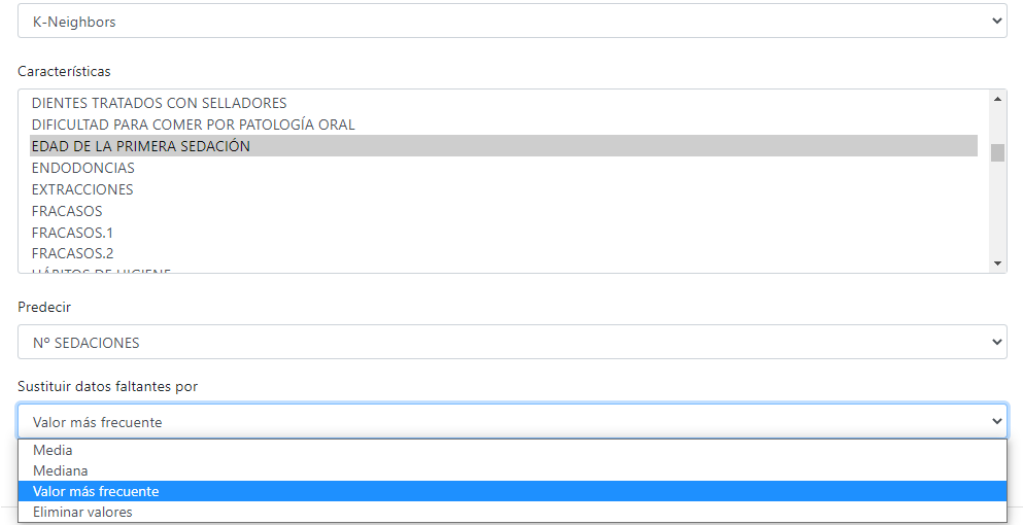
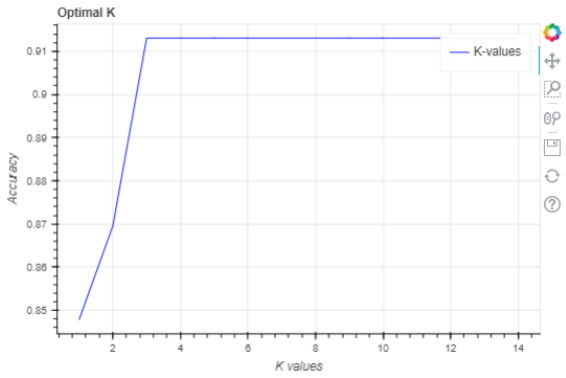


Figura 4.8: Selección de método para solución al problema de imputación

6. Entrenar el modelo pulsando en el botón *Entrenar modelo*. Apareciendo los datos del entrenamiento del modelo. Como son la precisión del clasificador para el set de entrenamiento y testeo, las métricas obtenidas para la *feature* a predecir, la matriz de confusión y las clasificaciones para comprobar el funcionamiento del clasificador con el conjunto de datos de testeo.

Datos del entrenamiento



Precisión del clasificador para el set de entrenamiento: 0.9402173913043478

Precisión del clasificador para el set de testeo: 0.9130434782608695

Figura 4.9: Gráfica de valores de K

Métricas obtenidas del entrenamiento para PRESENCIA DE CARIES

Mostrar 5 registros Buscar:

index	no	sí
f1-score	0.333333	0.953488
precision	0.500000	0.931818
recall	0.250000	0.976190
support	4.000000	42.000000

Mostrando registros del 1 al 4 de un total de 4 registros Anterior 1 Siguiente

Figura 4.10: Métricas obtenidas en el entrenamiento

Tabla de confusión

Mostrar 3 registros Buscar:

	CLASIFICADO COMO: no	CLASIFICADO COMO: sí
no	1	3
sí	1	41

Mostrando registros del 1 al 2 de un total de 2 registros Anterior 1 Siguiente

Figura 4.11: Matriz de confusión obtenida en el entrenamiento

Clasificaciones conjunto de testeo

Mostrar 10 registros Buscar:

	ORIGINAL	CLASIFICADO	PROBABILIDAD	EDAD DE LA PRIMERA SEDACIÓN	NECESIDAD DE MEDICACIÓN POR PATOLOGÍA ORAL	PLACA	DIAGNÓSTICO PRINCIPAL
0	sí	sí	1.000000	6.0	no registrado	sí	sano
1	sí	sí	1.000000	5.0	no registrado	sí	autista
2	sí	sí	1.000000	5.0	no	sí	sano
3	sí	sí	1.000000	7.0	no	sí	sano
4	sí	sí	1.000000	9.0	no registrado	sí	autista
5	sí	sí	1.000000	5.0	sí	sí	sano
6	sí	sí	1.000000	12.0	no	sí	Cromosopatía autogénica congénita
7	sí	sí	1.000000	9.0	no registrado	sí	autista
8	sí	sí	1.000000	8.0	no registrado	sí	autista
9	sí	sí	1.000000	3.0	sí	sí	sano

Mostrando registros del 1 al 10 de un total de 46 registros Anterior 1 2 3 4 5 Siguiente

Figura 4.12: Clasificaciones del modelo realizadas con el conjunto de testeo

7. Realizar la clasificación del paciente rellenando el formulario de características del paciente. Y vamos a suponer, para este caso, que el hipotético paciente es sano, la primera sedación se le realizó a los 9 años, no necesita medicación por patología oral y no presenta placa.

Formulario del paciente

DIAGNÓSTICO PRINCIPAL:

sano

EDAD DE LA PRIMERA SEDACIÓN:

9

NECESIDAD DE MEDICACIÓN POR PATOLOGÍA ORAL:

no

PLACA:

no

Clasificar

Figura 4.13: Formulario de características del paciente

8. Finalmente, clasificar al paciente pulsando sobre el botón *Clasificar*. Mostrándose el resultado de la clasificación, una tabla de clasificación y las tablas antes mostradas.

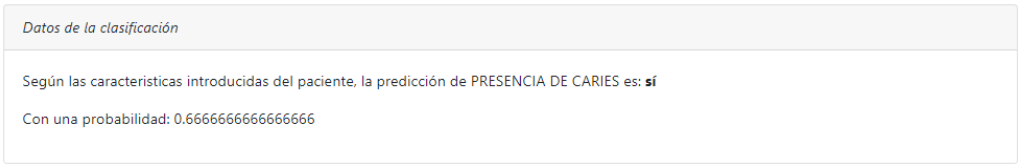


Figura 4.14: Clasificación obtenida del paciente

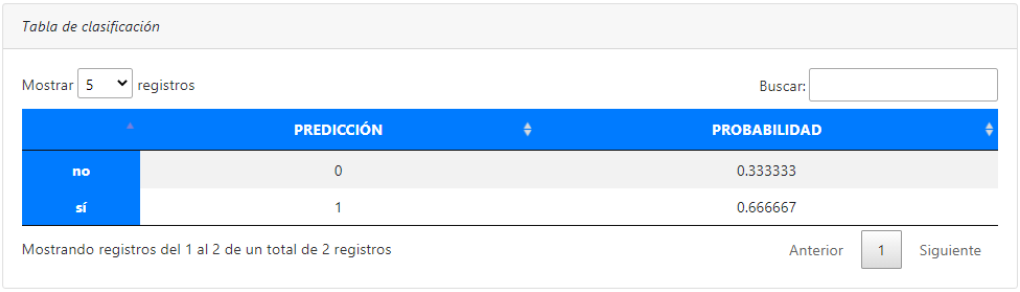


Figura 4.15: Tabla de clasificaciones posibles del paciente

Como se puede observar, el paciente tiene un 66,66 % de posibilidades de presentar caries.

4.2. Resultados clasificaciones

En este punto se muestra el resultado de diferentes clasificadores de interés con unas características de pacientes ya determinadas. Los clasificadores son los que se muestran en la tabla 4.1.

	Características				
Presencia de caries	Diagnóstico principal	Edad de la primera sedación	Placa	Necesidad de medicación por patología oral	
Seguimiento de prevención	Diagnóstico principal	Edad de la primera sedación	Remitido	Necesidad de medicación por patología oral	
Comporta-miento en citas	Diagnóstico principal	Edad de la primera sedación	Remitido	Necesidad de medicación por patología oral	
Más de una sedación	Diagnóstico principal	Edad de la primera sedación	Motiva-ción	Comporta-miento en citas	Seguimiento de prevención

Tabla 4.1: Tabla de clasificaciones

4.2.1. Presencia de caries

K-Neighbors - Media

- Métricas

	no	sí
F1-score	0	0,955
Precision	0	0,913
Recall	0	1
Support	4	42

Tabla 4.2: Tabla métricas K-Neighbors usando Media

- Matriz de confusión

	clasificado como: no	clasificado como: sí
no	0	4
sí	0	42

Tabla 4.3: Matriz de confusión K-Neighbors usando Media

- Clasificación de un paciente sano, sedado por primera vez a los 9 años, que no necesita medicación por patología oral y no presenta placa.

  Resultado: Sí con una probabilidad de un 66,67 %

	Probabilidad (%)
no	33,33
sí	66,67

Tabla 4.4: Tabla de probabilidad de clasificación

Logistic Regression - Media

- Métricas

	no	sí
F1-score	0	0,930
Precision	0	0,889
Recall	0	0.976
Support	5	41

Tabla 4.5: Tabla métricas Logistic Regression usando Media

- Matriz de confusión

	clasificado como: no	clasificado como: sí
no	0	5
sí	1	40

Tabla 4.6: Matriz de confusión Logistic Regression usando Media

- Clasificación de un paciente sano, sedado por primera vez a los 9 años, que no necesita medicación por patología oral y no presenta placa.

  Resultado: Sí con una probabilidad de un 98.86 %

	Probabilidad (%)
no	1,14
sí	98,86

Tabla 4.7: Tabla de probabilidad de clasificación

Como se puede observar, en ambos modelos no se consigue clasificar correctamente el atributo no, clasificándose la totalidad de los casos del conjunto de testeo como sí. Esto nos indica que ambos modelos presentarán una tendencia a clasificar al paciente con un sí aunque esta no sea la clasificación adecuada, *Logistic Regression* lo hará con menos precisión aún. También se observa que el modelo *K-Neighbors* presenta un *F1_score* mayor que *LogisticRegression*, por tanto, se puede decir que en este caso el primer modelo funciona mejor.

4.2.2. Seguimiento de prevención

Gaussian NB - Mediana

- Métricas

	no acude	acude	no registrado porque el paciente es de otra consulta
F1_score	0,516	0,769	0,545
Precision	0,533	0,789	0,5
Recall	0,5	0,75	0,6
Support	16	20	10

Tabla 4.8: Tabla métricas Gaussian NB usando Mediana

- Matriz de confusión

	Clasificado como: no acude	Clasificado como: acude	Clasificado como: no registrado porque el paciente es de otra consulta
acude	4	15	1
no acude	8	3	5
no registrado porque el paciente es de otra consulta	3	1	6

Tabla 4.9: Matriz de confusión Gaussian NB usando Mediana

- Clasificación de un paciente autista, sedado por primera vez a los 7 años, que necesita medicación por patología oral y no remitido.

  Resultado: No acude con una probabilidad de un 99.55 %

	Probabilidad (%)
acude	0,45
no acude	99,55
no registrado porque el paciente es de otra consulta	0

Tabla 4.10: Tabla de probabilidad de clasificación

DecisionTree - Mediana

- Métricas

	no acude	acude	no registrado porque el paciente es de otra consulta
F1_score	0,471	0,757	0,476
Precision	0,471	0,737	0,5
Recall	0,471	0,778	0,455
Support	17	18	11

Tabla 4.11: Tabla métricas DecisionTree usando Mediana

- Matriz de confusión

	Clasificado como: no acude	Clasificado como: acude	Clasificado como: no registrado porque el paciente es de otra consulta
acude	3	14	1
no acude	8	5	4
no registrado porque el paciente es de otra consulta	6	0	5

Tabla 4.12: Matriz de confusión DecisionTree usando Mediana

- Clasificación de un paciente autista, sedado por primera vez a los 7 años, que necesita medicación por patología oral y no remitido

  Resultado: No registrado porque el paciente es de otra consulta con una probabilidad de un 100 %

	Probabilidad (%)
acude	0
no acude	0
no registrado porque el paciente es de otra consulta	100

Tabla 4.13: Tabla de probabilidad de clasificación

En este caso se observa que la matriz de confusión, en los dos modelos utilizados,

presenta valores más repartidos. Si comparamos las métricas obtenidas (misma precisión para ambos modelos, pero mejor recall en *Gaussian NB*) vemos que los valores en ambos modelos son bajos, lo que es indicador de que los modelos no son especialmente buenos para esta *feature* concreta. Finalmente, se llega a la conclusión de que el modelo *Gaussian Naive Bayes* realizará una mejor clasificación para los pacientes de la *feature Seguimiento de prevención*.

4.2.3. Comportamiento en citas

Random Forest - Valor más frecuente

- Métricas

	no colaborador	colaborador
F1-score	0,516	0,754
Precision	0,471	0,793
Recall	0,571	0.719
Support	14	32

Tabla 4.14: Tabla métricas Logistic Regression usando Media

- Matriz de confusión

	clasificado como: no colaborador	clasificado como: colaborador
colaborador	9	23
no colaborador	8	6

Tabla 4.15: Matriz de confusión RandomForest usando Valor más frecuente

- Clasificación de un paciente con encefalopatía, sedado por primera vez a los 16 años, que no necesita medicación por patología oral y remitido desde centro de salud. **Resultado:** Colaborador con una probabilidad de un 57 % .

	Probabilidad (%)
colaborador	57
no colaborador	43

Tabla 4.16: Tabla de probabilidad de clasificación *Comportamiento en citas*

K-Neighbors

- Métricas

	no colaborador	colaborador
F1-score	0,471	0,88
Precision	0,5	0,868
Recall	0,44	0.891
Support	9	37

Tabla 4.17: Tabla métricas K-Neighbors usando Valor más frecuente

- Matriz de confusión

	clasificado como: no colaborador	clasificado como: colaborador
colaborador	4	33
no colaborador	4	5

Tabla 4.18: Matriz de confusión K-Neighbors usando Valor más frecuente

- Clasificación de un paciente con encefalopatía, sedado por primera vez a los 16 años, que no necesita medicación por patología oral y remitido desde centro de salud. **Resultado:** Colaborador con una probabilidad de un 66,67 %.

	Probabilidad (%)
colaborador	66.67
no colaborador	33.33

Tabla 4.19: Tabla de probabilidad de clasificación *Comportamiento en citas*

Para la *feature Comportamiento en citas*, se han utilizado los modelos *Random Forest* y *K-Neighbors*. Si comparamos las métricas obtenidas, los valores de *K-Neighbors* son mejores que los de *Random Forest*, aunque ambos valores son buenos. Por tanto, el modelo que obtiene mejores resultados es *K-Neighbors*.

4.2.4. Más de 1 sedación

Gaussian Naive Bayes

- Métricas

	1 sedación	2 sedaciones	3 sedaciones	4 sedaciones	5 sedaciones
F1 score	0,44	0	0	0	0
Precision	1	0	0	0	0
Recall	0,286	0	0	0	0
Support	14	7	5	0	0

Tabla 4.20: Tabla métricas Gaussian NB eliminando valores faltantes

- Matriz de confusión

	Clasificado como: 1 sedación	Clasificado como: 2 sedaciones	Clasificado como: 3 sedaciones	Clasificado como: 4 sedaciones	Clasificado como: 5 sedaciones
1 sedación	4	0	0	5	5
2 sedaciones	0	0	0	5	2
3 sedaciones	0	0	0	3	2
4 sedaciones	0	0	0	0	0
5 sedaciones	0	0	0	0	0

Tabla 4.21: Matriz de confusión Gausssian NB eliminando valores faltantes

- Clasificación de un paciente colaborador, sano, sedado por primera vez a los 5 años, que acude al seguimiento de prevención y sin motivación alguna.

  Resultado: 1 sedación con una probabilidad de un 100 %

	Probabilidad (%)
1 sedación	100
2 sedaciones	0
3 sedaciones	0
4 sedaciones	0
5 sedaciones	0

Tabla 4.22: Tabla de probabilidad de clasificación *Más de una sedación*

Logistic Regression

- Métricas

	1 sedación	2 sedaciones	3 sedaciones	4 sedaciones
F1 score	0,857	0,33	0	0
Precision	0,818	0,25	0	0
Recall	0,9	0,5	0	0
Support	20	2	3	1

Tabla 4.23: Tabla métricas Logistic Regression eliminando valores faltantes

- Matriz de confusión

	Clasificado como: 1 sedación	Clasificado como: 2 sedaciones	Clasificado como: 3 sedaciones	Clasificado como: 4 sedaciones
1 sedación	18	2	0	0
2 sedaciones	1	1	0	0
3 sedaciones	2	1	0	0
4 sedaciones	1	0	0	0

Tabla 4.24: Matriz de confusión Logistic Regression eliminando valores faltantes

- Clasificación de un paciente colaborador, sano, sedado por primera vez a los 5 años, que acude al seguimiento de prevención y sin motivación alguna.

  Resultado: 2 sedaciones con una probabilidad de un 99,53 %

	Probabilidad (%)
1 sedación	0,4663
2 sedaciones	99,532
3 sedaciones	0,0005
4 sedaciones	0,0012

Tabla 4.25: Tabla de probabilidad de clasificación *Más de una sedación*

Para el clasificador *Más de una sedación* encontramos que para *Gaussian NB* las métricas obtenidas son malas, ya que solo obtenemos valores por encima de 0 en el caso de *1 sedación*. Con el modelo *Logistic Regression* observamos que pasa prácticamente lo mismo, aunque esta vez se obtienen valores por encima de 0 en los casos de *1 sedación* y *2 sedaciones*, siendo además, las de *1 sedación* valores bastante buenos. Por tanto, observando también las clasificacaciones realizadas en el conjunto de testeo y las probabilidades obtenidas en la tabla de probabilidades **??**, vemos que estos resultados son más creíbles que los obtenidos por el modelo *Guassian NB*. Con lo que el modelo que mejor funciona para este clasificador es *Logistic Regression*.

CAPÍTULO

5

CONCLUSIONES Y LÍNEAS FUTURAS

5.1. Conclusión

Finalmente se ha conseguido desarrollar una aplicación que cumpliese los requisitos mencionados en la sección *Objetivos* del *Capítulo 1*, en la que se pedía que esta fuese capaz de realizar la predicción de una propiedad a partir de un conjunto de características de un paciente y que además fuese capaz de mostrar información de los diferentes pacientes con el fin de facilitar la labor de los especialistas. La aplicación cuenta con diferentes secciones a las que el usuario puede acceder, y estas son Home, Resumen estadístico, Documentación, Distribuciones y Clasificación de pacientes.

Tal y como se ha explicado anteriormente, la sección *Home* muestra toda la información de los pacientes recogida en el dataset *pacientes.csv* . La sección *Resumen estadístico* muestra metainformación de utilidad para la elaboración de un diagnóstico y saber datos generales de las diferentes características que puede presentar un paciente.

La sección *Documentación* lleva al usuario hasta la ubicación de este documento.

La sección *Distribuciones* muestra la distribución de la característica que el usuario

escoja mediante una gráfica interactiva.

Y la sección *Clasificación de pacientes*, la que más peso ha tenido en el desarrollo de la aplicación, se encarga de realizar la clasificación del atributo escogido por el usuario pudiendo escoger el clasificador entre *K-Neigbors*, *Gaussian NB*, *LogisticRegression*, *DecisionTree* y *RandomForest*, y la solución que el usuario quiera utilizar para el problema de imputación. Una vez rellenado el primer formulario se entrena el clasificador escogido, mostrando la precisión del clasificador para los conjuntos de datos de entrenamiento y testeo, la tabla de métricas obtenidas para el entrenamiento, la tabla de confusión y las clasificaciones realizadas con el conjunto de testeo. Además, se muestra el segundo formulario a rellenar por el usuario con las características del paciente. En la parte final aparecen ambos formularios y se produce la clasificación final del paciente mostrándose la clasificación final, y las tablas anteriores.

5.2. Líneas futuras

Como posible extensión a la aplicación web y siempre con el objetivo de facilitar la labor de los especialistas se puede realizar un *clustering* de los diferentes pacientes, de forma que se pueda realizar una representación gráfica de como se agrupan todos los pacientes de los que se tienen datos.

Crear otro apartado que se utilice para realizar predicciones utilizando algoritmos de regresión como *Linear Regression*, los cuales se pueden utilizar para predecir características numéricas, como puede ser el número de sedaciones o el número de caries que va a desarrollar un paciente. Aunque esto ya se ha realizado utilizando los algoritmos de clasificación que se han explicado en los apartados anteriores.

También se puede utilizar una base de datos en vez de un archivo *.csv*, dotando a la aplicación de cierto dinamismo en cuanto a la actualización de los datos.

BIBLIOGRAFÍA

[1] Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow (Second Edition)*. O'Reilly Media, 2019.

[2] Python Software Foundation. *Python*. <https://www.python.org/>. 2021.

[3] NumPy Community. *NumPy*. <https://numpy.org/>. 2021.

[4] Pandas Community. *Pandas*. <https://pandas.pydata.org/>.

[5] Scikit-Learn Community. *Scikit-Learn*. <https://scikit-learn.org/stable/>. 2021.

[6] Scikit-Learn. *Nearest Neighbors*. <https://scikit-learn.org/stable/modules/neighbors.html>.

[7] Scikit-Learn. *Gaussian Naive Bayes*. <https://scikit-learn.org/stable/modules/naive_bayes.html>.

[8] Scikit-Learn. *Logistic Regression*. <https://scikit-learn.org/stable/modules/linear_model.html>.

[9] Scikit-Learn. *Decision Tree*. <https://scikit-learn.org/stable/modules/tree.html>.

[10] Scikit-Learn. *1.11. Ensemble methods, 1.11.2. Forests of randomized trees, 1.11.2.1. Random Forests*. <https://scikit-learn.org/stable/modules/ensemble.html>.

[11] Bokeh Community. *Bokeh 2.3.2 documentation*. <https://docs.bokeh.org/>. 2021.

[12] Joblib developers. *Joblib*. <https://joblib.readthedocs.io/en/latest/>.

[13] IPython development team. *IPython*. <https://ipython.org/>.

[14] Flask Community. *Flask web development, one drop at a time*. <https://flask.palletsprojects.com/>.

[15] JavaScript Community. *JavaScript*. <https://www.javascript.com/>.