ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERIA EN SISTEMAS

MODELO DE PREDICCIÓN DE PARÁMETROS DE CALIDAD DE PRODUCTOS VÍNICOS UTILIZANDO MACHINE LEARNING

MODELO DE PREDICCIÓN DE CALIDAD DE VINO UTILIZANDO APRENDIZAJE AUTOMÁTICO

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN CIENCIAS DE LA COMPUTACION

ALVARO ISRAEL VALVERDE MUÑOZ

alvaro.valverde@epn.edu.ec

DIRECTOR: IVÁN MARCELO CARRERA IZURIETA

ivan.carrera@epn.edu.ec

Febrero 2024

CERTIFICACIONES

Yo, Alvaro Israel Valverde Muñoz declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

ALVARO ISRAEL VALVERDE MUÑOZ

Certifico que el presente trabajo de integración curricular fue desarrollado por Alvaro Israel Valverde Muñoz, bajo mi supervisión.

IVAN MARCELO CARRERA IZURIETA

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el producto resultante del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

Alvaro Israel Valverde Muñoz

Iván Marcelo Carrera Izurieta

DEDICATORIA

A mis padres. Por el apoyo y el amor incondicional

AGRADECIMIENTO

A todos los profesores que me guiaron para tener una vocación.

ÍNDICE DE CONTENIDO

RESUMEN

El presente proyecto de titulación busca determinar si es posible predecir la calidad del vino a partir de sus propiedades fisicoquímicas utilizando un modelo de aprendizaje automático. Inicialmente, se realizó una investigación bibliográfica utilizando términos clave relacionados con el tema. Se recopilaron modelos de aprendizaje automático y se obtuvo una base de datos para el entrenamiento y prueba del modelo. Tras comparar los modelos, se decidió utilizar redes neuronales y se ajustaron sus hiperparámetros utilizando el proceso de grilla de parámetros para mejorar su exactitud.

Posteriormente, se analizaron los parámetros de la base de datos para identificar las características más influyentes en el modelo. Se emplearon los métodos LIME y SHAP para explicar el modelo y determinar las propiedades fisicoquímicas más importantes para cada posible valor de calidad del vino.

Finalmente, se integró el modelo entrenado en un sistema de arquitectura vista-controlador, con componentes backend y frontend. El backend se desarrolló con Fastapi y el frontend con JavaScript y Angular. Este sistema permitirá a los emprendedores evaluar la calidad del vino producido y obtener predicciones sobre su calidad.

PALABRAS CLAVE: Aprendizaje Automático, Redes Neuronales, Exactitud, LIME, SHAP, Calidad de vino

ABSTRACT

The titling project seeks to determine whether it is possible to predict the quality of wine from its physical and chemical properties using an automatic learning model. Initially, bibliographic research was carried out using key terms related to the topic. Automatic learning models were collected, and a database was obtained for model training and testing. After comparing the models, it was decided to use neural networks and adjust their hyper parameters using the parameter grid process to improve their accuracy.

Subsequently, database parameters were analyzed to identify the most influential characteristics in the model. Lime and Shape methods were used to explain the model and determine the most important physical and chemical properties for each possible quality value of the wine.

Finally, the trained model was integrated into a view-control architecture system, with backend and frontend components. The backend was developed with Fastapi and frontend with JavaScript and Angular. This system will enable entrepreneurs to assess the quality of the wine produced and to obtain predictions about its quality.

KEYWORDS: Machine Learning, Neural Networks, Accuracy, LIME, SHAP, Wine Quality

1 DESCRIPCIÓN DEL COMPONENTE DESARROLLADO

Este proyecto de titulación tiene como objetivo responder la pregunta *¿se puede predecir la calidad del vino, basándose en sus propiedades fisicoquímicas, utilizando un modelo de aprendizaje automático?* Para cumplir con este propósito, en un inicio se recopiló documentación sobre el tema, y se recolectaron publicaciones científicas con objetivos similares. Para la búsqueda de bibliografía se utilizó términos clave relacionados a la finalidad del proyecto. Dichos términos fueron: *"wine"*, *"quality"*, *"prediction"* y *"machine learning"*. De la investigación se obtuvieron dos resultados. El primero es una lista modelos de aprendizaje automático. Con esto se adquirió información como el número de veces que se implementó cada modelo y sus métricas de validación. El segundo es una base de datos que se utilizó como datos de prueba y entrenamiento.

Con la información obtenida se realizó una tabla comparativa de todos los modelos y sus métricas de validación. Como consecuencia se decidió utilizar el modelo de redes neuronales en nuestra investigación. Para la implementación de este modelo, se utilizó el lenguaje de programación Python. Para entrenar este modelo, se necesita definir valores adecuados para sus hiper parámetros. La exactitud de un modelo depende de su capacidad de predecir un valor a partir de un conjunto de valores de entrada. Dicha capacidad varía dependiendo de la combinación de los hiper parámetros. Para obtener la combinación optima, se utilizó el proceso de grilla de parámetros, el cual entrena al modelo con las combinaciones posibles de los hiper parámetros. Como resultado se obtiene el modelo ya entrenado con la exactitud más alta.

Posteriormente, se analizaron los parámetros existentes en la base de datos. Para identificar las características con mayor influencia en el modelo. Para esto se implementó dos métodos de explicabilidad de modelos. Los cuales fueron LIME y SHAP. Con esto se obtuvo las propiedades fisicoquímicas más importantes para cada posible valor de calidad que puede tomar una muestra de vino. Con este resultado se determina qué factores son los determinantes para tener un vino de calidad.

Finalmente, con el modelo entrenado, se implementó en un sistema de arquitectura vista controlador. El cual posee dos componentes backend y frontend. El primero se construyó utilizando el framework Fastapi. Para el segundo componente se utilizó el lenguaje de programación JavaScript con el framework angular. El cual permitirá a los emprendedores evaluar su producción de vino y obtener una predicción de la calidad del vino que produjeron.

1.1 Objetivo general

Generar un modelo computacional preciso y eficaz que pueda predecir y mejorar la calidad del vino, brindando una herramienta práctica para la toma de decisiones en la producción vinícola.

1.2 Objetivos específicos

- Generar una base de datos de las propiedades fisicoquímicas del vino.
- Entrenar un modelo de machine learning que prediga la calidad del vino.
- Implementar ese modelo en datos propios recopilados.
- Crear una interfaz de usuario para el uso del modelo.

1.3 Alcance

El presente componente busca desarrollar modelos de aprendizaje automático para determinar la calidad del vino, teniendo en consideración los métodos más utilizados e identificando las variables más influyentes en la calidad del vino.

En primer lugar, se realizará una revisión de literatura, en donde se analizará los métodos o técnicas de aprendizaje automático existentes en el sector de la vinicultura y los resultados, que serán un insumo para identificar el algoritmo más utilizado e identificar las variables fisicoquímicos influyen en la calidad de vino de forma predominante.

A partir de un conjunto de datos públicos abiertos relacionada con la calidad de algunas variedades de vino, se filtrará la información, se integrarán bases de datos, se correlacionarán datos y se identificarán al menos tres variables químicas más relevantes que permitan obtener resultados confiables sobre la calidad del vino.

1.4 Marco teórico

Red Neuronal

Una red neuronal artificial (RNA) es un modelo computacional que imita la estructura y el funcionamiento del sistema nervioso biológico. Se compone de una serie de unidades de procesamiento, llamadas neuronas artificiales, que están interconectadas entre sí. Las neuronas artificiales reciben entradas, procesan la información y producen salidas. (Michael I. Jordan, Stuart Russell y Peter Norvig, 2016)

Regresión Softmax

La regresión softmax es una función de activación que se utiliza en las redes neuronales para clasificar datos. La función softmax toma como entrada un vector de valores y devuelve un vector de probabilidades. Las probabilidades se distribuyen de forma que la suma de todas ellas es 1. Se utiliza comúnmente en los clasificadores multiclase. Por ejemplo, un clasificador de imágenes puede clasificar imágenes en las categorías "perro", "gato", "coche", etc. La función softmax se puede utilizar en cualquier tipo de red neuronal. Sin embargo, es más común utilizarla en redes neuronales profundas, donde se pueden aprender patrones complejos en los datos. (Michael I. Jordan, Stuart Russell y Peter Norvig, 2016)

Investigación bibliográfica

(Cortez et al., 2009) presenta un modelo de machine learning que relaciona variables físico químicas de vino y predice lo que la llama como preferencias. En este artículo se usa como variables de entrada las propiedades fisicoquímicas, tales como: Acidez fija, Acidez volátil, Ácido cítrico, Azúcar residual, Cloruros, Dióxido de azufre libre, Dióxido de azufre total, Densidad, pH, Sulfatos, Alcohol. En este trabajo se evalúan los modelos regresión múltiple, redes neuronales, máquinas de vectores de soporte. En este trabajo se tiene los siguientes resultados:

	Vino rojo			Vino Blanco		
	Regresión múltiple	Redes neuronales	Máquinas de soporte de vectores	Regresión múltiple	Redes neuronales	Máquinas de soporte de vectores
Exactitud	88.6	88.8	89	20.9	23.5	43.9

Tabla 1 Resultados Cortez et al., 2009

(Yogesh Gupta, 2017) presenta un modelo de machine learning que relaciona variables físico químicas de vino y predice lo que la llama como preferencias. En este artículo se usa como variables de entrada las propiedades fisicoquímicas, tales como: Acidez fija, Acidez volátil, Ácido cítrico, Azúcar residual, Cloruros, Dióxido de azufre libre, Dióxido de azufre total, Densidad, pH, Sulfatos, Alcohol. En este trabajo se evalúan los modelos redes neuronales, regresión lineal. En este trabajo se tiene los siguientes resultados:

	Vino rojo		Vino Blanco	
	Regresión lineal	Redes neuronales	Regresión lineal	Redes neuronales
Error		0.187312		0.234133
R^2	0.36055		0.306753	

Tabla 2 Resultados Yogesh Gupta, 2017

(Seunghan Lee, Juyoung Park and Kyungtae Kang, s. f.) presenta un modelo de machine learning que relaciona variables físico químicas de vino y predice lo que la llama como preferencias. En este artículo se usa como variables de entrada las propiedades fisicoquímicas, tales como: Acidez fija, Acidez volátil, Ácido cítrico, Azúcar residual, Cloruros, Dióxido de azufre libre, Dióxido de azufre total, Densidad, pH, Sulfatos, Alcohol. En este trabajo se evalúa el modelo árbol de decisión. En este trabajo se tiene los siguientes resultados:

	Vino rojo	Vino Blanco
	Arboles de decisión	Arboles de decisión
Exactitud	60.7	58.7

Tabla 3 Resultados Seunghan Lee, Juyoung Park and Kyungtae Kang, s. f.

(Nuriel S, Tigabo Asras, Eli Gal, Tesfahon Demasia, Ezra Tarab, Nathaniel Ezekiel, Osher Nikapros, and Oshri Semimufar, 2022) presenta un modelo de machine learning que relaciona variables físico químicas de vino y predice el tipo de vino al que corresponde puede ser blanco o rojo. En este artículo se usa como variables de entrada las propiedades fisicoquímicas, tales como: Acidez fija, Acidez volátil, Ácido cítrico, Azúcar residual, Cloruros, Dióxido de azufre libre, Dióxido de azufre total, Densidad, pH, Sulfatos, Alcohol. En este trabajo se evalúa el modelo de clasificación con redes neuronales. En este trabajo se tiene los siguientes resultados: exactitud igual a 99.69.

(K. R. Dahal, J. N. Dahal, H. Banjade, S. Gaire, 2021) presenta un modelo de machine learning que relaciona variables físico químicas de vino y predice lo que la llama como preferencias. En este artículo se usa como variables de entrada las propiedades fisicoquímicas, tales como: acidez fija, acidez volátil, ácido cítrico, azúcar residual, cloruros, dióxido de azufre libre, dióxido de azufre total, densidad, pH, sulfatos, alcohol. Se evalúa los modelos regresión de cresta, máquina de vectores de soporte, regresor de aumento de gradiente, redes neuronales, y se obtienen los siguientes resultados:

Modelo	R	Error medio cuadrado	Error porcentual absoluto medio
Regresión de cresta	0.6029	0.4281	0.0934
Máquina de vectores de soporte	0.7797	0.267	0.1426
Regresor de aumento de gradiente	0.7255	0.3286	0.0826
Redes neuronales	0.66	0.37	0.14

Tabla 4 Resultados K. R. Dahal, J. N. Dahal, H. Banjade, S. Gaire, 2021

(Mahima, Ujjawal Gupta, Yatindra Patidar, Abhishek Agarwal,and Kushall Pal Singh, 2020) presenta un modelo de machine learning que relaciona variables físico químicas de vino y predice lo que la llama como preferencias. En este artículo se usa como variables de entrada las propiedades fisicoquímicas, tales como: acidez fija, acidez volátil, ácido cítrico, azúcar residual, cloruros, dióxido de azufre libre, dióxido de azufre total, densidad, pH, sulfatos, alcohol. En este trabajo se evalúa los modelos K-nearest neighbours y Random forest. En este trabajo se obtienen los siguientes resultados:

	Vino rojo	Vino Blanco
Modelo	Error cuadrático medio	Error cuadrático medio
K-nearest neighbour	0.584	0.541
Random forest	0.584	0.541

Tabla 5 Resultados Mahima, Ujjawal Gupta, Yatindra Patidar, Abhishek Agarwal,and Kushall Pal Singh, 2020

(Satyabrata Aich, Ahmed Abdulhakim Al-Absi, Kueh Lee Hui, Mangal Sain, 2018) presenta un modelo de machine learning que relaciona variables físico químicas de vino y predice lo que la llama como preferencias. En este artículo se usa como variables de entrada las propiedades fisicoquímicas, tales como: Acidez fija, Acidez volátil, Ácido cítrico, Azúcar residual, Cloruros, Dióxido de azufre libre, Dióxido de azufre total, Densidad, pH, Sulfatos, Alcohol. En este trabajo se evalúa el modelo de máquinas de vectores de soporte. En este trabajo se obtienen los siguientes resultados: exactitud de para vino rojo igual a 95.23 y exactitud de con vino banco 98.1.

(Piyush Bhardwaj a,b et al., 2022)presenta un modelo de machine learning que relaciona variables físico químicas de vino y predice lo que la llama como preferencias. En este artículo se usa como variables de entrada las propiedades fisicoquímicas, tales como: Acidez fija, Acidez volátil, Ácido cítrico, Azúcar residual, Cloruros, Dióxido de azufre libre, Dióxido de azufre total, Densidad, pH, Sulfatos, Alcohol. En este trabajo se evalúa los modelos random forest, gausian naive bayes, adaBoost, descenso de gradiente

estocastico, máquina de vectores de soporte, árbol de decisión de clasificación, K-nearest neighbour. En este trabajo se obtienen los siguientes resultados:

Modelo	Exactitud
Random forest	83
Gausian naive bayes	33
AdaBoost	100
Descenso de gradiente estocastico	83
Máquina de vectores de soporte	83
Arbol de decisión de clasificación	50
K-nearest neighbour	83

Tabla 6 Resultados Piyush Bhardwaj a,b et al., 2022

(Mohit Gupta, Vanmathi C, s. f.) presenta un modelo de machine learning que relaciona variables físico químicas de vino y predice lo que la llama como preferencias. En este artículo se usa como variables de entrada las propiedades fisicoquímicas, tales como: Acidez fija, Acidez volátil, Ácido cítrico, Azúcar residual, Cloruros, Dióxido de azufre libre, Dióxido de azufre total, Densidad, pH, Sulfatos, Alcohol. En este trabajo se evalúa los modelos random forest, K-nearest neighbour, máquina de vectores de soporte, J48, Cart, M5P. En este trabajo se obtienen los siguientes resultados:

	Vino rojo	Vino Blanco
Modelo	Exactitud	Exactitud
J48	56	69.36
K-nearest neighbour	61.39	60.52
Máquinas de vector de soporte	62.34	63.72
CART	70.75	78.19
Random forest	73.25	76.39
M5P	81.81	83.27

Tabla 7 Resultados Mohit Gupta, Vanmathi C

(Akanksha Trivedi, Ruchi Sehrawat, 2020) presenta un modelo de machine learning que relaciona variables físico químicas de vino y predice lo que la llama como preferencias. En este artículo se usa como variables de entrada las propiedades fisicoquímicas, tales como: Acidez fija, Acidez volátil, Ácido cítrico, Azúcar residual, Cloruros, Dióxido de azufre libre, Dióxido de azufre total, Densidad, pH, Sulfatos, Alcohol. En este trabajo se evalúa los modelos regresión logística, random forest. En este trabajo se obtienen los siguientes resultados:

	Vino rojo
Modelo	Exactitud
Regresión logística	76
Random forest	84

Tabla 8 Resultados Akanksha Trivedi, Ruchi Sehrawat, 2020

(Xianghui Jiang , Xuanyu Liu,, Yutong Wu, Dehuai Yang, 2023) presenta un modelo de machine learning que relaciona variables físico químicas de vino y predice lo que la llama como preferencias. En este artículo se usa como variables de entrada las propiedades fisicoquímicas, tales como: Acidez fija, Acidez volátil, Ácido cítrico, Azúcar residual, Cloruros, Dióxido de azufre libre, Dióxido de azufre total, Densidad, pH, Sulfatos, Alcohol. En este trabajo se evalúa los modelos regresión logística multinomial, random forest. En este trabajo se obtienen los siguientes resultados:

	Vino blanco
Modelo	Exactitud
Regresión logística multinomial	52
Random forest	90

Tabla 9 Resultados Xianghui Jiang , Xuanyu Liu,, Yutong Wu, Dehuai Yang, 2023

(Chao Ye, Ke Li,b, Guo-zhu Jia, 2020) presenta un modelo de machine learning que relaciona variables físico químicas de vino y predice lo que la llama como preferencias. En este artículo se usa como variables de entrada las propiedades fisicoquímicas, tales como: Acidez fija, Acidez volátil, Ácido cítrico, Azúcar residual, Cloruros, Dióxido de azufre libre, Dióxido de azufre total, Densidad, pH, Sulfatos, Alcohol. En este trabajo se evalúa los modelos LightGBM, XGBoost. En este trabajo se obtienen los siguientes resultados:

	Vino rojo
Modelo	Exactitud
LightGBM	91.04
XGBoost	91.04

Tabla 10 Resultados Chao Ye, Ke Li,b, Guo-zhu Jia, 2020

A continuación, se mostrará la tabla comparativa de los modelos realizados en la bibliografía realizada:

Vino Rojo							
Autor	***Modelo***	***Exactitud***	***Error***	R^2	***R***	***Error cuadrático medio***	***Error porcentual absoluto medio***
Cortez et al., 2009	Regresión múltiple	88.6					
	Redes neuronales	88.8					
	Máquinas de soporte de vectores	89					
Yogesh Gupta, 2017	Regresión lineal			0.36055			
	Redes neuronales		0.187312				
Seunghan Lee, Juyoung Park and Kyungtae Kang, s. f.	Arboles de decisión	60.7					
K. R. Dahal, J. N. Dahal, H. Banjade, S. Gaire, 2021)	Regresión de cresta				0.6029	0.4281	0.0934
	Máquina de vectores de soporte				0.7797	0.267	0.1426
	Regresor de aumento de gradiente				0.7255	0.3286	0.0826
	Redes neuronales				0.66	0.37	0.14
Mahima, Ujjawal Gupta, Yatindra Patidar, Abhishek Agarwal,and Kushall Pal Singh, 2020	K-nearest neighbour					0.584	
	Random forest					0.584	
Piyush Bhardwaj a,b et al., 2022	Random forest	83					
	Gausian naive bayes	33					
	AdaBoost	100					
	Descenso de gradiente estocastico	83					
	Máquina de vectores de soporte	83					
	Arbol de decisión de clasificación	50					
	K-nearest neighbour	83					
Mohit Gupta, Vanmathi C	J48	56					
	K-nearest neighbour	61.39					
	Máquinas de vector de soporte	62.34					
	CART	70.75					
	Random forest	73.25					
	M5P	81.81					
Akanksha Trivedi, Ruchi Sehrawat, 2020	Regresión logística	76					
	Random forest	84					
Chao Ye, Ke Li,b, Guo-zhu Jia, 2020	LightGBM	91.04					
	XGBoost	91.04					

Tabla 11 Comparación de resultados con vino rojo

Vino Blanco					
Autor	***Modelo***	***Exactitud***	***Error***	R^2	***Error cuadrático medio***
Cortez et al., 2009	**Regresión múltiple**	20.9			
	Redes neuronales	23.5			
	Máquinas de soporte de vectores	43.9			
Yogesh Gupta, 2017	**Regresión lineal**			0.306753	
	Redes neuronales		0.234133		
Seunghan Lee, Juyoung Park and Kyungtae Kang, s. f.	**Arboles de decisión**	58.7			
Mahima, Ujjawal Gupta, Yatindra Patidar, Abhishek Agarwal,and Kushall Pal Singh, 2020	**K-nearest neighbour**				0.541
	Random forest				0.541
Mohit Gupta, Vanmathi C	**J48**	69.36			
	K-nearest neighbour	60.52			
	Máquinas de vector de soporte	63.72			
	CART	78.19			
	Random forest	76.39			
	M5P	83.27			
Xianghui Jiang , Xuanyu Liu,, Yutong Wu, Dehuai Yang, 2023	**Regresión logística multinomial**	52			
	Random forest	90			

Tabla 12 Comparación de resultados con vino blanco

2 METODOLOGÍA

En este trabajo se utilizó el modelo CRISP-DM, que tiene las siguientes fases:

2.1 Entendimiento del negocio.

Se reviso la literatura, se concluyó que si es posible predecir la calidad del vino utilizando las propiedades fisicoquímicas. Por lo tanto, se va a utilizar la base de datos de wine quality.

2.2 Entendimiento de los datos.

El conjunto de datos Calidad del vino (wine quality) posee información sobre dos tipos de vino tinto (rojo) y blanco. Que fueron producidos en la región portuguesa de Vinho Verde. Se creo con el fin de predecir la calidad del vino basándose en las propiedades fisicoquímicas que posee. La cual tiene doce variables once de entrada y una de salida. Las variables de entrada son:

- Acidez fija: cantidad de ácido tartárico, el ácido más abundante en el vino.
- Acidez volátil: cantidad de ácido acético, un ácido orgánico que puede causar sabores desagradables en el vino.
- Ácido cítrico: cantidad de ácido cítrico, un ácido orgánico que puede proporcionar un sabor fresco al vino.
- Azúcar residual: cantidad de azúcar que queda en el vino después de la fermentación.
- Cloruros: cantidad de cloruro en el vino.
- Dióxido de azufre libre: cantidad de dióxido de azufre libre en el vino.
- Dióxido de azufre total: cantidad total de dióxido de azufre en el vino.
- Densidad: densidad del vino.
- pH: valor de pH del vino.
- Sulfatos: cantidad de sulfatos en el vino.

La variable de salida es la calidad del vino, que toma valores discretos en el rango de 0 a 10.

El conjunto de datos posee 1599 muestras de vino tinto (rojo) y 4898 de vino blanco. Las clases no están equilibradas, con muchos más vinos de calidad normal que vinos de calidad

excelente o mala. Puede utilizarse para tareas de clasificación o regresión. Para la clasificación, el objetivo sería predecir la clase de calidad del vino. Para la regresión, el objetivo sería predecir la puntuación de calidad del vino. Algunos desafíos potenciales al trabajar con este conjunto de datos incluyen:

- La presencia de valores atípicos, que pueden dificultar el aprendizaje del modelo.
- La incertidumbre sobre la relevancia de todas las variables de entrada.

El conjunto de datos ES útil para la investigación sobre la calidad del vino. Este es de tamaño razonable y contiene una variedad de variables fisicoquímicas que pueden ser relevantes para la calidad del vino. (Markelle Kelly, Rachel Longjohn, Kolby Nottingham, 2009)

2.3 Preparación de los datos.

Inicialmente se importan todas las librerías de Python que se usaron. Luego se carga el conjunto de datos de wine quality desde un archivo CSV. El archivo se encuentra en un directorio especifico. Este posee 1599 registros de vino tinto (rojo). A continuación, extraemos las características fisicoquímicas que corresponden a las variables de entrada. Y la calidad del vino que corresponde a la variable de salida. Posteriormente se normaliza las características. Esto ayuda a garantizar que todos los datos tengan el mismo peso al entrenar el modelo. En este caso, se utiliza un escalador estándar para normalizar los datos. Después convertimos los valores de la calidad a categóricas. Mediante codificación one-hot, el cual es un método que representa las posibles clases como vectores binarios. Esto permite que el modelo aprenda las relaciones entre las diferentes clases de calidad del vino. Para representar las 10 clases de calidad del vino. Finalmente dividimos los datos en conjuntos de entrenamiento y prueba. El conjunto de entrenamiento contiene el 80% de los datos y el conjunto de prueba contiene el 20%.

```
import itertools
from keras.models import Sequential
from keras.layers import Dense, Dropout
from keras.regularizers import l1, l2, l1_l2
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from tensorflow.keras.utils import to_categorical
from sklearn.model_selection import GridSearchCV
from keras.models import load_model
import shap
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import lime
import lime.lime_tabular
import re
```

```
# Cargar el conjunto de datos
wine_data = pd.read_csv( 'C:/Users/israv/OneDrive/Escritorio/winequality/winequality-red_point.csv', delimiter=';')

# Extraer características y etiquetas
y = wine_data['quality']
X = wine_data.drop('quality', axis=1)

# Normalizar los datos de la característica
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Convertir etiquetas a categóricas (codificación one-hot)
y_categorical = to_categorical(y)

# Divida los datos en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_categorical, test_size=0.2, random_state=42)
```

2.4 Modelado

Empezamos definiendo la función `build_model()`.Los parámetros de entrada definidos como las siguientes variables:

- `hidden_layers`: El número de capas ocultas en la red neuronal.
- `neurons`: El número de neuronas en cada capa oculta.
- `activation`: La función de activación utilizada en cada capa oculta.
- `dropout_rate`: La tasa de abandono utilizada después de cada capa oculta.
- `l1_reg`: El coeficiente de regularización L1.
- `l2_reg`: El coeficiente de regularización L2.

Esta comienza inicializando un modelo secuencial usando `Sequential()`. Un modelo secuencial es un tipo de modelo de red neuronal en el que las capas se conectan en secuencia. A continuación, agregamos la primera capa densa al modelo. Una capa densa es un tipo de capa de red neuronal la cual usa una función de activación no lineal para procesar los datos. La función de activación especificada se define como la variable `activation`. Se define la propiedad `kernel_regularizer=l1_l2(l1=l1_reg, l2=l2_reg)` con el fin de aplicar regularización L1 y L2 en la capa. La regularización ayuda a evitar el sobreajuste. Este es un problema que puede ocurrir cuando un modelo se ajusta demasiado a los datos de entrenamiento y no es capaz de generalizar a nuevos datos. Luego la función agrega un bucle `for` para crear `hidden_layers - 1` capas ocultas. Cada capa oculta tiene la misma estructura que la capa de entrada.

Opcionalmente, la función puede aplicar el abandono con una tasa `dropout_rate` después de cada capa oculta. El abandono ayuda a evitar el sobreajuste al eliminar aleatoriamente algunas neuronas de la red durante el entrenamiento. Después la función agrega la capa de salida al final del modelo. La capa de salida tiene un número de neuronas igual a `y_train.shape[1]`, que coincide con el número de clases en el conjunto de datos de entrenamiento. La función de activación especificada para la capa de salida es '`softmax`'. Antes de terminar se compila el modelo. La función de pérdida escogida es '`categorical_crossentropy`', la cual es adecuada para la clasificación multiclase. El optimizador especificado es '`Adam`'. La métrica especificada es '`accuracy`', que mide la exactitud del modelo en la predicción de la clase correcta. Finalmente, la función devuelve el modelo compilado listo para el entrenamiento.

```
def build_model(hidden_layers, neurons, activation, dropout_rate, l1_reg, l2_reg):
    model = Sequential()
    model.add(Dense(neurons, input_dim=X_train.shape[1], activation=activation,
kernel_regularizer=l1_l2(l1=l1_reg, l2=l2_reg)))
    for _ in range(hidden_layers - 1):
        model.add(Dense(neurons, input_dim=X_train.shape[1], activation=activation,
                kernel_regularizer=l1_l2(l1=l1_reg, l2=l2_reg)))
        if dropout_rate > 0:
            model.add(Dropout(dropout_rate))
    model.add(Dense(y_train.shape[1], activation='softmax'))
    model.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])
    return model
```

2.5 Evaluación

A continuación, se realizó una búsqueda de grilla de parámetros para encontrar la mejor configuración de hiper parámetros posible en una red neuronal para una tarea de clasificación. Inicialmente se definió la grilla de parámetros. Para esto se crea un diccionario llamado `param_grid` que contiene los diferentes parámetros a explorar y sus posibles valores. Estos parámetros son:

- `hidden_layers`: número de capas ocultas (5 o 6).
- `neurons`: número de neuronas en cada capa oculta (512 o 1024).
- `activation`: función de activación (relu, tanh o sigmoid).
- `dropout_rate`: tasa de abandono (0.0 o 0.1).
- `l1_reg`: coeficiente de regularización L1 (0.0 o 0.1).
- `l2_reg`: coeficiente de regularización L2 (0.0 o 0.1).

Des puesta se realizó la iteración sobre combinaciones posibles de parámetros definidos anteriormente. Para esto se utilizó la función `itertools.product` y un bucle `for` .Que permitirán evaluar todas las posibles combinaciones de parámetros de la grilla. Para cada una de estas desempaquetamos los parámetros. Después creamos un modelo usando la función `build_model` con los valores anteriormente desenpaquetados. Entonces se procedió a entrenar el modelo con los datos `(X_train, y_train)`. Finalmente evaluamos el rendimiento del modelo en el conjunto de prueba `(X_test, y_test)`. Mediante la exactitud de este. Si la exactitud actual es mejor que la mejor previamente registrada, se actualizan la mejor exactitud, los mejores parámetros y el mejor modelo. Como resultado se imprimen la mejor puntuación y los mejores parámetros encontrados.

```
# Definir la grilla de parámetros a buscar
param_grid = {
    'hidden_layers': [3,4,5],
    'neurons': [ 512,256,128],
    'activation': ['relu', 'tanh','sigmoid'],
    'dropout_rate': [0.0,0.1],
    'l1_reg': [0.0,0.1],
    'l2_reg': [0.0,0.1]
}

# Iterar sobre todas las combinaciones.
best_score = 0
best_params = None
best_model = None

for params in itertools.product(*param_grid.values()):
    # Unpack parameters
    hidden_layers, neurons, activation, dropout_rate, l1_reg, l2_reg = params
    print(f"Parameters: {params}")

    # Build and train the model
    model = build_model(hidden_layers, neurons, activation, dropout_rate,
l1_reg, l2_reg)
    model.fit(X_train, y_train, epochs=200, batch_size=64, verbose=0) # Set
verbose to 0 to reduce output

    # Evaluate the model
    score = model.evaluate(X_test, y_test, verbose=0)[1]  # Assuming [1] is
accuracy
    print(f"Score: {score}")

    # Update best score
    if score > best_score:
        best_score = score
        best_params = params
        best_model = model

# Print the best configuration
print(f"Best Score: {best_score}")
print(f"Best Parameters: {best_params}")
```

2.6 Explicabilidad

2.6.1 Proceso SHAP

Para el siguiente proceso de explicabilidad se utilizó la biblioteca SHAP (SHapley Additive exPlanations) para explicar las predicciones de un modelo de aprendizaje automático entrenado, llamado `best_model`. Con objetivo es entender cómo las diferentes características del conjunto de datos influyen en las predicciones del modelo para diferentes clases de salida. Se inicio con la definición de los nombres de las características de los datos de entrada. Para esto se creó una lista llamada `_feature_names` . Luego se creó de un objeto `masker`, a partir de la clase `shap.maskers.Independent`. Este se utiliza para manejar la independencia entre las características del conjunto de datos. Posterior mente se realizó la creación de un objeto `explainer`, a partir de la clase `shap.Explainer`. Él se encarga de calcular los valores SHAP, que explican la contribución de cada característica a las predicciones del modelo. Este se inicializa con el modelo entrenado `(loaded_model)` y el objeto `masker`. A continuación, se calculan los valores SHAP para el conjunto de datos de entrenamiento `(X_train)` utilizando el método `explainer(X_train)`. Los cuales se almacenaron en la variable `shap_values`. Finalmente se utiliza un bucle `for` para iterar sobre las clases de salida del modelo, desde la clase 3 hasta la clase 8. Para cada clase se crea un gráfico de resumen SHAP utilizando la función `shap.summary_plot`. El gráfico muestra la influencia de cada característica en las predicciones del modelo para la clase actual. Se imprime un mensaje indicando la clase actual. Subsiguientemente se muestra el gráfico utilizando `plt.show()`. Para terminar, guardando el gráfico en un archivo PNG utilizando `plt.savefig()`.

```
_feature_names = ['fixed acidity', 'volatile acidity', 'citric acid',
'residual sugar', 'chlorides',
                  'free sulfur dioxide', 'total sulfur dioxide',
'density', 'pH', 'sulphates', 'alcohol']
masker = shap.maskers.Independent(data=X_train)
explainer = shap.Explainer(best_model, masker=masker)
shap_values = explainer(X_train)
for i in range(3, 9, 1):
    shap.summary_plot(shap_values.values[:, :, i], X_train,
feature_names=_feature_names,
                      title='SHAP values for'+str(i))
    print("Calidad igual a "+str(i))
    plt.show()
    plt.savefig('Shap_summary_plot_'+str(i)+'.png')
```

2.6.2 Proceso LIME

Para la implementación de este proceso de explicabilidad se utiliza la biblioteca LIME (Local Interpretable Model-Agnostic Explanations) con el objetivo de explicar las predicciones de un modelo de aprendizaje automático entrenado, llamado `best_model`. Con esto se comprendió cómo las diferentes características del conjunto de datos influyen en las predicciones del modelo para diferentes clases de salida. Para lo cola se procedió a crear de un objeto `explainer` LIME. A partir de la clase `lime.lime_tabular.LimeTabularExplainer`. El cual se encarga de generar explicaciones locales para las predicciones del modelo. Se inicializaron con los datos de entrenamiento `(X_train)`, el modo de clasificación, los nombres de las características y los nombres de las clases. Después se realizó la generación de explicaciones para cada instancia. Se utilizo un bucle `for` para iterar sobre cada instancia en el conjunto de datos de entrenamiento `(X_train)`.En cada iteración se creó una función de predicción para LIME `(predict_fn)` que utiliza el modelo entrenado. Seguido se generó una explicación local para la instancia utilizando el método `explainer.explain_instance()`.Con esto se obtuvo la predicción del modelo para la iteración, se extrajo la clase predicha de la predicción, luego se convirtió la explicación en un DataFrame de Pandas. Con esto se limpiaron los nombres de las características en el DataFrame. Enseguida se tomaron los valores absolutos de la importancia de las características. Para continuar se almaceno el DataFrame en un diccionario `all_explanations` según la clase predicha. Posteriormente se calculó de la importancia media de las características por clase. Entonces se creó un diccionario `mean_importance_dict` para almacenar la importancia media de las características por clase. Con este se itero sobre cada clase en el diccionario `all_explanations`. Para cada iteración se concatenaron todos los DataFrames de la clase actual. Inmediatamente se calculó la importancia media de cada característica. Ordenamos las características por importancia descendente. Finalmente se guardó a el resultado en un archivo CSV.

```
explainer = lime.lime_tabular.LimeTabularExplainer(X_train,
mode='classification',feature_names=_feature_names,class_names=['0', '1', '2',
'3', '4', '5', '6', '7', '8'])
all_explanations = {'1':[],'2':[],'3':[],'4':[],'5':[],'6':[],'7':[],'8':[]}
# Generar explicaciones con LIME para cada instancia en X_test
for instance in X_train:
    predict_fn = lambda x: best_model.predict(x)  # Función de predicción para
LIME
    explanation = explainer.explain_instance(instance, predict_fn,
num_features=11)
    array_result = best_model.predict(np.array([instance]))

    quality_prediction = float(np.argmax(array_result, axis=1)[0])

    quality_prediction = int(round(quality_prediction))
    explanation_data = explanation.as_list()
    # Crear un DataFrame con los datos de la explicación
    explanation_df = pd.DataFrame(explanation_data, columns=['Feature',
'Importance'])
    explanation_df['Feature'] = explanation_df['Feature'].apply(lambda x:
re.sub(r'[^a-zA-Z\s]+', '', x).strip())
    explanation_df['Importance'] = explanation_df['Importance'].abs()
    #all_explanations.append(explanation_df)  # Solo guardamos los valores de
importancia
    all_explanations[str(quality_prediction)].append(explanation_df)

    mean_importance_dict = {}

# Iterar sobre cada índice en el diccionario
for index, dfs_list in all_explanations.items():
    if(len(dfs_list)!=0):
        # Concatenar todos los dataframes para el índice actual
        combined_df = pd.concat(dfs_list, ignore_index=True)
        # Calcular el promedio de 'Importance' para cada 'Feature'
        mean_importance =
combined_df.groupby('Feature')['Importance'].mean().reset_index()
        mean_importance = mean_importance.sort_values(by='Importance',
ascending=False).reset_index(drop=True)
        mean_importance.to_csv('archivo_lime_'+str(index)+'.txt', index=False,
sep='\t')
```

2.7 Despliegue

Una vez el modelo entrenado. Se procedió a generar dos componentes. Un componente back-end realizado con Python y un componente front-ent con Javascript. Esto con la finalidad de poder consumir el modelo y mostrar las predicciones que este puede generar de una forma amigable para los productores de vino que quieran evaluar su producción de vino.

2.7.1 Back-end

Inicialmente se importaron las bibliotecas necesarias: `fastapi, pydantic, keras.models, numpy y fastapi.middleware.cors`. Se creo una instancia de la clase `FastAPI` llamada `app` para crear la aplicación web. A continuación, cargo un modelo de red neuronal previamente entrenado desde un archivo `.h5` utilizando la función `load_model` de `Keras`. Posteriormente definimos dos clases `Pydantic`:

- `WineFeatures`: Representa las características del vino que se utilizarán como entrada para la predicción.
- `PredictionResult`: Representa el resultado de la predicción, que consiste en la calidad predicha del vino.

Luego se añade el middleware `CORSMiddleware` para permitir el acceso a la API desde otros dominios. Después definimos un endpoint POST en la ruta `/predict/` utilizando el decorador `@app.post("/predict/")`. Este endpoint recibe un objeto `WineFeatures` que contiene las características del vino. Con esto se desencadenan los siguientes procesos:

- Imprime los datos del vino recibidos para depuración.
- Convierte los datos del vino en un array NumPy con el formato adecuado para la predicción.
- Realiza la predicción utilizando el modelo cargado.
- Extrae la calidad predicha del resultado de la predicción.
- Devuelve un objeto `PredictionResult` con la calidad predicha.

```python
from fastapi import FastAPI
from pydantic import BaseModel
from keras.models import load_model
import numpy as np
from fastapi.middleware.cors import CORSMiddleware

app = FastAPI()

# Cargar el modelo de red neuronal desde el archivo .h5
model = load_model('C:/Users/israv/Downloads/Untitled/693.h5')


class WineFeatures(BaseModel):
    Fixed_acidity: float
    Volatile_acidity: float
    Citric_acid: float
    Residual_sugar: float
    Chlorides: float
    Free_sulfur_dioxide: float
    Total_sulfur_dioxide: float
    Density: float
    PH: float
    Sulphates: float
    Alcohol: float


class PredictionResult(BaseModel):
    calidad: float


app.add_middleware(
    CORSMiddleware,
    allow_origins=['http://localhost:4200'],
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)
@app.post("/predict/")
def predict(wine_feature: WineFeatures):  # Cambio aquí a WineFeatures en
lugar de List[WineFeatures]
    # Convertir los datos del vino a un formato adecuado para la predicción

    print(wine_feature)
    wine_data = np.array([
        wine_feature.Fixed_acidity, wine_feature.Volatile_acidity,
wine_feature.Citric_acid,
        wine_feature.Residual_sugar, wine_feature.Chlorides,
wine_feature.Free_sulfur_dioxide,
        wine_feature.Total_sulfur_dioxide, wine_feature.Density,
wine_feature.PH,
        wine_feature.Sulphates, wine_feature.Alcohol
    ]).reshape(1, -1)

    # Realizar la predicción usando el modelo cargado
    quality_prediction = model.predict(wine_data)
    quality_prediction = np.argmax(quality_prediction, axis=1)[0]

    prediction = {"calidad": float(quality_prediction)}

    return prediction
```

2.7.2 Front-end

Para consumir el api que se creó anteriormente. Se creo un proyecto con TypeScript. Utilizando el framework Angular. Este mediante un formulario, que recolecta las propiedades fisicoquímicas, genera una estructura Json. Esta se envió al api generada (Back-end) anteriormente, mediante una petición Post. Recibe una estructura Json que tiene el valor de la predicción. Toma ese valor y lo muestra en pantalla.

Componente HTML del formulario:

```
<div class="full-width-section">
    <div class="container">
      <div class="imagen">
        <img src="assets/img/logo-epn-vertical.png" alt="logo" class="logo">
      </div>

      <div class="title-section">
        <h1 class="title">Predicción de la calidad del vino</h1>
      </div>
    </div>
  </div>

  <div class="content">
    <!-- Contenido debajo del título -->
    <form #form="ngForm" (ngSubmit)="onPredict(form)">
        <div class="form">
          <div class="input-row" *ngFor="let param of parameters">
            <label class="input-label">{{ param }}</label>
            <input type="number" name="{{ param }}" [(ngModel)]="formData[param]"
required class="input-field">
          </div>
        </div>
```

```
        <button type="submit">Predecir calidad</button>
      </form>

      <div *ngIf="resultadoCalidad !== undefined" class="resultado-container">
        <p class="resultado-text">Resultado de calidad:</p>
        <div class="resultado-valor">{{ resultadoCalidad }}</div>
      </div>

  </div>
```

Componente TypeScript del formulario

```
import { FormsModule } from '@angular/forms';
import { HttpClientModule } from '@angular/common/http';
import { HttpClient } from '@angular/common/http';
import { CommonModule } from '@angular/common';
import { NgForm } from '@angular/forms';
import { Component, ElementRef } from '@angular/core';

@Component({
  selector: 'app-prediction-form',
  standalone: true,
  imports: [FormsModule,
    HttpClientModule,
    CommonModule],
  templateUrl: './prediction-form.component.html',
  styleUrl: './prediction-form.component.css'
})
export class PredictionFormComponent {
  resultadoCalidad: number | undefined;
  formData: { [key: string]: number } = {};
  parameters: string[] = [
    'Fixed_acidity',
    'Volatile_acidity',
    'Citric_acid',
    'Residual_sugar',
    'Chlorides',
    'Free_sulfur_dioxide',
    'Total_sulfur_dioxide',
    'Density',
    'PH',
    'Sulphates',
    'Alcohol'
  ];
  constructor(private http: HttpClient, private elementRef: ElementRef) {}
  onPredict(form: NgForm): void {
    if (form.valid) {
      const values = Object.values(this.formData);
      if (values.some(value => isNaN(value))) {
        alert('Por favor ingrese solo valores numéricos.');
      } else {
        console.log(this.formData)
        this.http.post<any>('http://localhost:8000/predict/', this.formData)
          .subscribe(response => {
            //alert(`El valor predicho es: ${response.calidad}`);
            //this.resetForm();
            this.resultadoCalidad = response.calidad;
            this.scrollDownToResult();
          });
      }
    } else {
      alert('Por favor ingrese todos los parámetros.');
    }
  }
  private scrollDownToResult(): void {
    const resultadoContainer =
this.elementRef.nativeElement.querySelector('.resultado-container');
    if (resultadoContainer) {
      resultadoContainer.scrollIntoView({ behavior: 'smooth' });
    }
  }

}
```

Componente CSS del formulario

```
/* Estilos para el contenedor principal */
.full-width-section {
    background-color: #f8f9fa; /* Color de fondo */
    padding: 20px 0; /* Espacio interno */
  }

  .container {
    display: flex;
    justify-content: space-between; /* Distribuye los elementos a lo largo del
contenedor */
    align-items: center; /* Centra los elementos verticalmente */
  }

  /* Estilos para la imagen */
  .imagen {
    /* Elimina la propiedad flex y establece un margen derecho */
    margin-right: auto; /* Mueve la imagen a la izquierda */
  }

  .logo {
    width: 100px; /* Tamaño del logo */
  }

/* Estilos para el título */
.title-section {
    text-align: center; /* Centra el texto */
    flex: 1; /* El título ocupará todo el espacio disponible */
    margin-right: 10px; /* Agrega un pequeño margen a la derecha */
    margin-left: -5%; /* Ajusta el margen a la izquierda */
  }
```

```css
.title {
  margin: 5;
}

/* Estilos para el formulario */
.content {
  display: flex;
  flex-direction: column;
  align-items: center;
}

.form {
  width: 300px; /* Ancho del formulario */
  margin-bottom: 20px; /* Espacio inferior */
}

.input-row {
  margin-bottom: 10px; /* Espacio entre filas */
}

.input-label {
  margin-right: 10px; /* Espacio a la derecha de las etiquetas */
}

.input-field {
  width: 100%; /* Ancho completo */
  padding: 5px; /* Espaciado interno */
  border: 1px solid #ced4da; /* Borde */
  border-radius: 4px; /* Borde redondeado */
}

button[type="submit"] {
  width: 100%; /* Ancho completo */
  padding: 10px; /* Espaciado interno */
  background-color: #007bff; /* Color de fondo del botón */
  color: #fff; /* Color del texto del botón */
  border: none; /* Sin borde */
  border-radius: 4px; /* Borde redondeado */
  cursor: pointer; /* Cursor apuntador */
  transition: background-color 0.3s; /* Transición suave */
}

button[type="submit"]:hover {
  background-color: #0056b3; /* Cambio de color al pasar el mouse */
}
```

```css
  .resultado-container {
    margin-top: 20px; /* Espacio superior */
    text-align: center; /* Centrar el contenido */
}

.resultado-text {
    margin-bottom: 5px; /* Espacio inferior */
    font-size: 1.5em; /* Tamaño de fuente más grande */
    font-weight: bold; /* Texto en negrita */
}

.resultado-valor {
    font-size: 2em; /* Tamaño de fuente más grande */
    font-weight: bold; /* Texto en negrita */
}
```

3 RESULTADOS, CONCLUSIONES Y RECOMENDACIONES

3.1 Resultados

3.1.1 Evaluación

En la siguiente tabla presentamos los diez resultados más relevantes de la evaluación del modelo utilizando el proceso grilla de parámetros.

hidden_layers	neurons	activation	dropout_rate	l1_reg	l1_reg	Exactitud
4	512	tanh	0.0	0.0	0.0	64.6875023841
3	512	relu	0.0	0.0	0.0	65.3124988079
5	256	tanh	0.0	0.0	0.0	65.9375011920
5	256	relu	0.0	0.0	0.0	65.9375011920
3	256	tanh	0.0	0.0	0.0	66.2500023841
5	256	tanh	0.0	0.0	0.0	66.2500023841
3	256	relu	0.0	0.0	0.0	66.8749988079
4	512	relu	0.0	0.0	0.0	67.5000011920
4	256	relu	0.0	0.0	0.0	68.1249976158
3	512	tanh	0.0	0.0	0.0	69.6874976158

Tabla 13 Resultados de evaluación del modelo

Con esto se obtuvo el modelo con la exactitud más alta en base a la mejor combinación de los valores para los hiper parámetros. Este tiene una exactitud de **69.68%.**

3.2 Explicabilidad

3.2.1 SHAP

Tras realizar el proceso SHAP, con el modelo obtenido en la fase de evaluación, se tiene un gráfico para cada clase que puede tomar el valor de la calidad. El cual resume la influencia de las características en las predicciones del modelo.

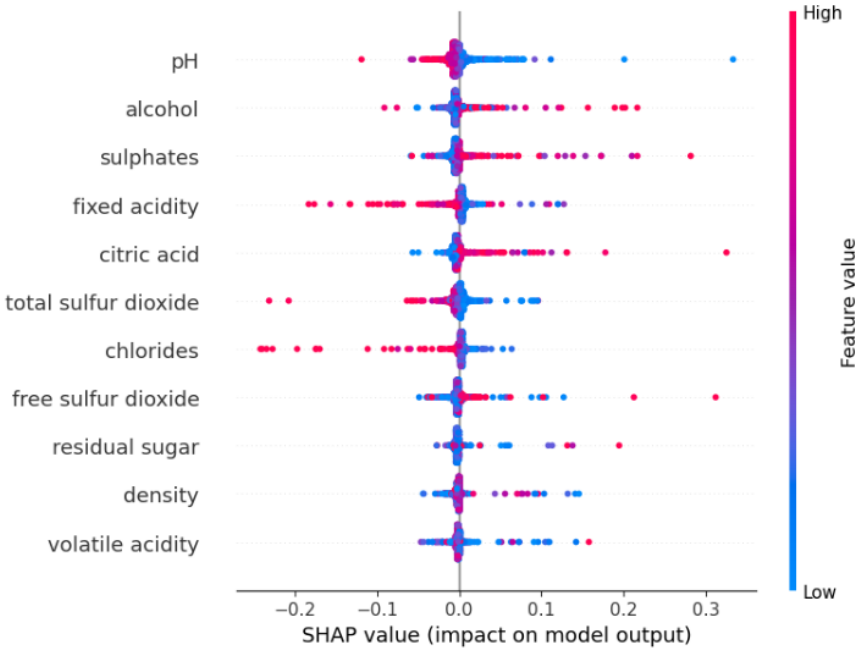

Ilustración 1 grafico Shap para la calidad igual a 3

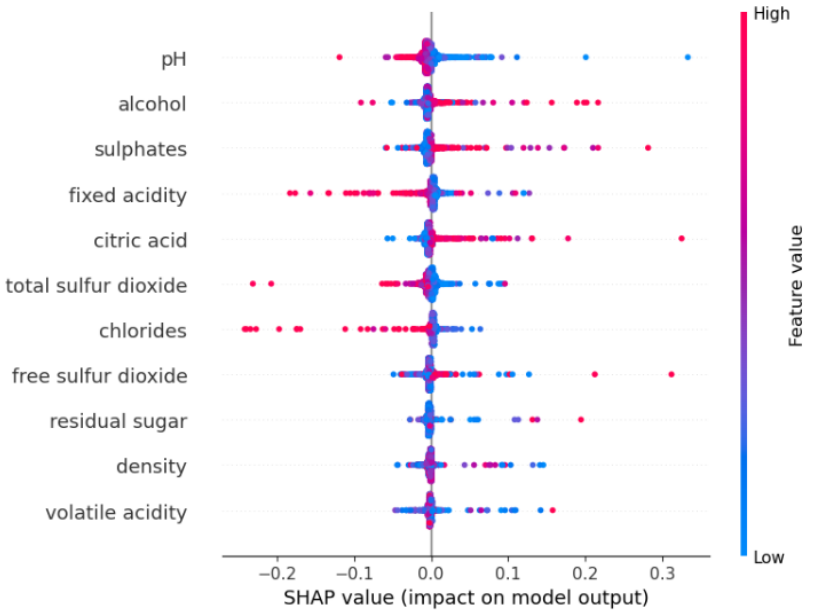

Ilustración 2 grafico Shap para la calidad igual a 4

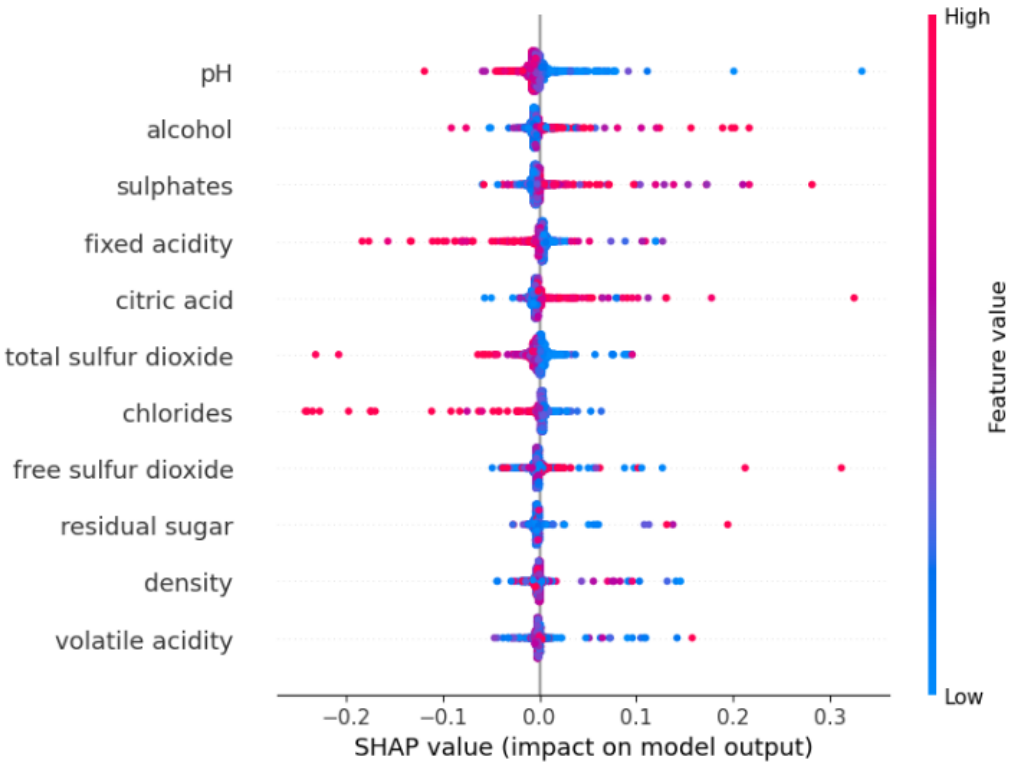

Ilustración 3 grafico Shap para la calidad igual a 5

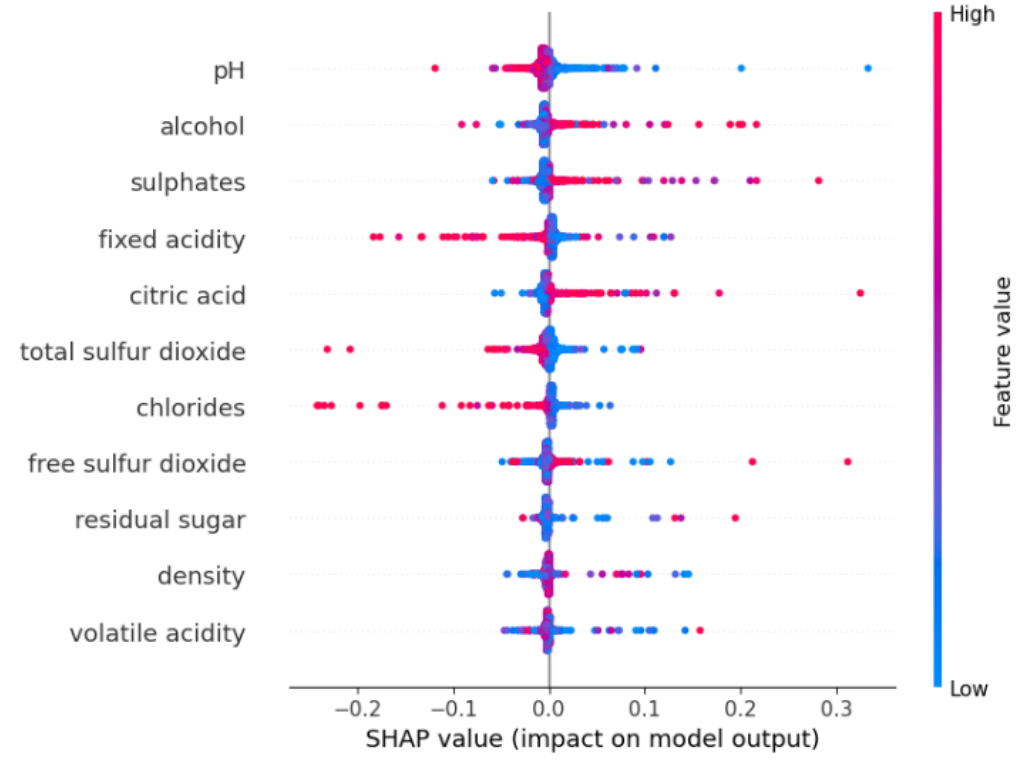

Ilustración 4 grafico Shap para la calidad igual a 6

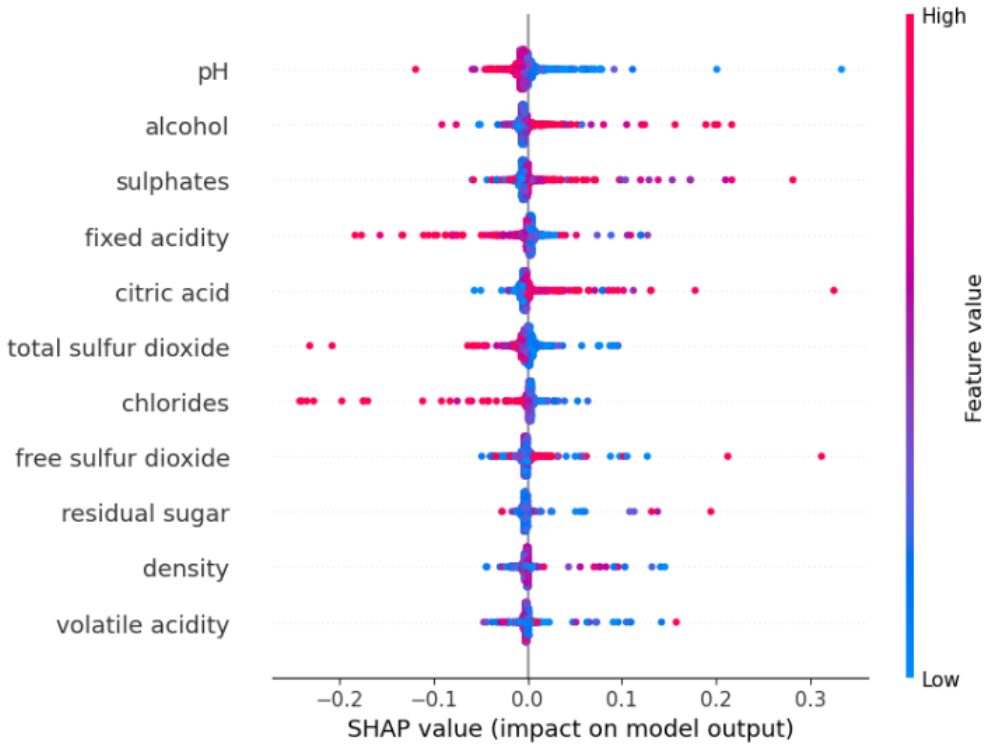

Ilustración 5 grafico Shap para la calidad igual a 7

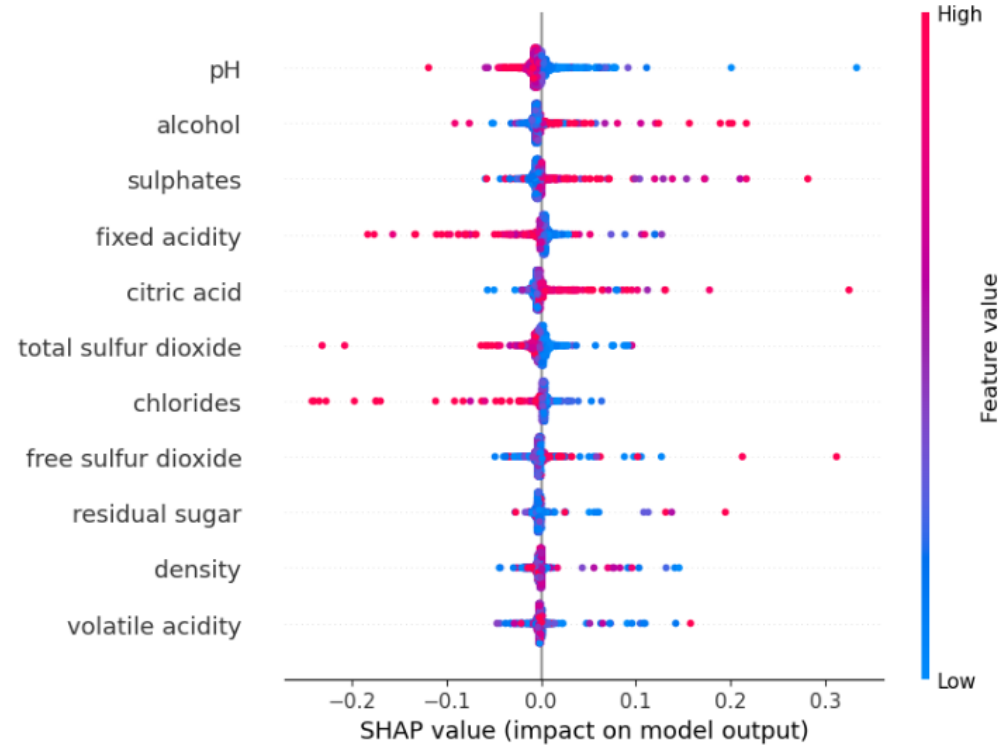

Ilustración 6 grafico Shap para la calidad igual a 8

3.2.2 Lime

Como resultado del proceso Lime se obtiene archivos CSV contienen la importancia de cada característica en las predicciones del modelo de aprendizaje automático para cada clase de salida. Cada fila del archivo representa una instancia de datos, y las columnas representan las características utilizadas para hacer la predicción.

Feature	Importance
total sulfur dioxide	0.0004081493656037607
sulphates	0.000399625125408986
free sulfur dioxide	0.00036830891342115493
citric acid	0.0002833976882269526
volatile acidity	0.00018313714074264697
residual sugar	0.00018007441023501285
fixed acidity	0.00016737878800438726
density	0.00010210878177463393
pH	0.00010173272428625573
chlorides	8.446064655647 34e-05
alcohol	8.367820699474562e-05

Ilustración 7 resultados de Lime para calidad igual a 3

Feature	Importance
free sulfur dioxide	0.00034389691369844774
total sulfur dioxide	0.00029354465172243396
sulphates	0.00025566085357446117
citric acid	0.00018260227193660266
fixed acidity	0.00015232670067792352
volatile acidity	0.00014676434782642927
residual sugar	0.00011791394145149861
pH	9.751339308927282e-05
density	8.199333869417705e-05
alcohol	6.89711463294471e-05
chlorides	6.780860039567827e-05

Ilustración 8 resultados Lime para calidad igual a 4

Feature	Importance
free sulfur dioxide	0.0003978847640294834
sulphates	0.0002778215662583271
total sulfur dioxide	0.0002742494015995776
citric acid	0.00017486268344974246
volatile acidity	0.00014049610924107565
fixed acidity	0.00013095699455705997
residual sugar	0.00010325825231887674
pH	0.00010082497116537755
chlorides	8.781590705353695e-05
alcohol	7.30337627060733e-05
density	6.791076095486031e-05

Ilustración 9 resultados Lime para calidad igual a 5

Feature	Importance
free sulfur dioxide	0.00039999525270034556
total sulfur dioxide	0.00026077798480357524
sulphates	0.00022828243513962237
citric acid	0.00018698344830279657
fixed acidity	0.0001542580720615091
volatile acidity	0.00014721889755363768
residual sugar	0.00010760531496543143
pH	0.00010278255437664032
chlorides	8.856625622072428e-05
density	7.646602021597552e-05
alcohol	7.564884300809552e-05

Ilustración 10 resultados Lime para calidad igual a 6

Feature	Importance
free sulfur dioxide	0.000355840791322126
total sulfur dioxide	0.00027874829730851065
sulphates	0.0002291280854220887
citric acid	0.00018972342036882199
volatile acidity	0.00017351415042345686
fixed acidity	0.00014677432388689247
residual sugar	0.00010613394106051026
pH	0.0001058512400777745
chlorides	8.589016756365965e-05
alcohol	7.410729341047712e-05
density	7.21264341568251e-05

Ilustración 11 resultados Lime para calidad igual a 7

Feature	Importance
free sulfur dioxide	0.00035989977580170045
total sulfur dioxide	0.00034501247663515865
sulphates	0.00022615969158815738
citric acid	0.00022442057681097168
volatile acidity	0.00015402414071337753
fixed acidity	0.0001434259398961416
residual sugar	9.675391317849044e-05
pH	8.086505374526886e-05
chlorides	7.089042120314453e-05
alcohol	6.906106968898961e-05
density	5.764645004490011e-05

Ilustración 12 resultados Lime para calidad igual a 8

3.3 3.3 Despliegue

Con un modelo entrenado se creó una aplicación web la cual tiene como objetivo brindar el servicio de predicción de calidad del vino, para los fabricantes de vino que necesiten evaluar su producción.

Ilustración 13 interfaz aplicación web

3.4 Conclusiones

- Se desarrolló un modelo de aprendizaje automático que predice la calidad del vino con una exactitud del 69.68%. El modelo desarrollado en este proyecto tiene un gran potencial para impactar la industria vinícola. Este puede ayudar a los productores de vino a mejorar la calidad de su vino y a tomar decisiones más

informadas sobre el proceso de producción. Además, el modelo puede ayudar a los consumidores a elegir vinos de mejor calidad.

- El modelo se implementó en un sistema de arquitectura vista controlador que permite a los emprendedores evaluar su producción de vino y obtener una predicción de la calidad del vino que produjeron.
- Las propiedades fisicoquímicas más importantes para la calidad del vino son el pH, la acidez total y el alcohol.

3.5 Recomendaciones

- Se recomienda utilizar el modelo para evaluar la calidad del vino durante el proceso de producción para tomar decisiones que mejoren la calidad del vino.
- Se recomienda recopilar datos de diferentes variedades de vino para mejorar la precisión del modelo.
- Se recomienda investigar otras propiedades fisicoquímicas que puedan influir en la calidad del vino.

4 REFERENCIAS BIBLIOGRÁFICAS

Bibliografía

Carvajal, L. (2006). *Metodología de la Investgación Científica. Curso general y aplicado* (28 ed.). Santiago de Cali: U.S.C.

Markelle Kelly, Rachel Longjohn, Kolby Nottingham. (6 de 10 de 2009). *UC Irvine*. Obtenido de https://archive.ics.uci.edu/dataset/186/wine+quality

Michael I. Jordan, Stuart Russell y Peter Norvig. (2016). *Aprendizaje automático: Una introducción.* Addison-Wesley.

Akanksha Trivedi, Ruchi Sehrawat. (2020). Wine Quality Detection through Machine Learning Algorithms. *IEEE*, 5. https://doi.org/10.1109/ICRIEECE44171.2018.9009111

Chao Ye, Ke Li,b, Guo-zhu Jia. (2020). A new red wine prediction framework using machine learning. *IOPSCEINCE*, 16. https://doi.org/10.1088/1742-6596/1684/1/012067

Cortez, P., Cerdeira, A., Almeida, F., Matos, T., & Reis, J. (2009). Modeling wine preferences by data mining from physicochemical properties. *Decision Support Systems*, *47*(4), 547-553. https://doi.org/10.1016/j.dss.2009.05.016

K. R. Dahal, J. N. Dahal, H. Banjade, S. Gaire. (2021). Prediction of Wine Quality Using Machine Learning Algorithms. *Scientific Research*, 12. https://doi.org/10.1007/978-3-642-04747-3_8

Mahima, Ujjawal Gupta, Yatindra Patidar, Abhishek Agarwal,and Kushall Pal Singh. (2020). Wine Quality Analysis Using Machine Learning Algorithms. *Springer LINK*, 8. https://doi.org/10.1007/978-981-15-2329-8_2

Mohit Gupta, Vanmathi C. (s. f.). A Study and Analysis of Machine Learning Techniques in Predicting Wine Quality. *ResearchGate*, 9. https://doi.org/10.35940/ijrte.A5854.0510121

Nuriel S, Tigabo Asras, Eli Gal, Tesfahon Demasia, Ezra Tarab, Nathaniel Ezekiel, Osher Nikapros, and Oshri Semimufar. (2022). Wine Quality and Type Prediction from Physicochemical Properties Using Neural Networks for Machine Learning A Free Software for Winemakers and Customers. *CABI digital library*, 13. https://doi.org/10.31220/agriRxiv.2022.00125

Piyush Bhardwaj a,b, , Parul Tiwari a,b, , Kenneth Olejar Jr c, , Wendy Parr d, , Don Kulasiri, & Piyush. (2022). A machine learning application in wine quality prediction. *sciencedirect*, 11. https://doi.org/10.1016/j.mlwa.2022.100261

Satyabrata Aich, Ahmed Abdulhakim Al-Absi, Kueh Lee Hui, Mangal Sain. (2018). Prediction of Quality for Different Type of Wine based on Different Feature Sets Using Supervised Machine Learning Techniques. *IEEE*, 6. https://doi.org/10.23919/ICACT.2019.8702017

Seunghan Lee, Juyoung Park and Kyungtae Kang. (s. f.). Assessing wine quality using a decision tree. *IEEE*, 3. https://doi.org/10.1109/SysEng.2015.7302752

Xianghui Jiang , Xuanyu Liu,, Yutong Wu, Dehuai Yang. (2023). White Wine Quality Prediction and Analysis with Machine Learning Techniques. *drpress*, 6. https://doi.org/10.54097/hset.v39i.6548

Yogesh Gupta. (2017). Selection of important features and predicting wine quality using machine learning techniques. *sciencedirect*, 8. https://doi.org/10.1016/j.procs.2017.12.041

5 ANEXOS

ANEXO 1. Repositorio del proyecto:

ivan-carrera/winequality (github.com)