

# **PRUEBA TECNICA DATA SCIENCE**

## **Modelos ML**

**Presentado por:**

**Mg Cristian Camilo Ordoñez Quintero**

**Noviembre 2024**

## **CONTENIDO**

PROYECTO 1 .....	3
EVALUACIÓN .....	5
ARTEFACTOS .....	7
PROYECTO 2 .....	7
CONCLUSIÓN. ....	9

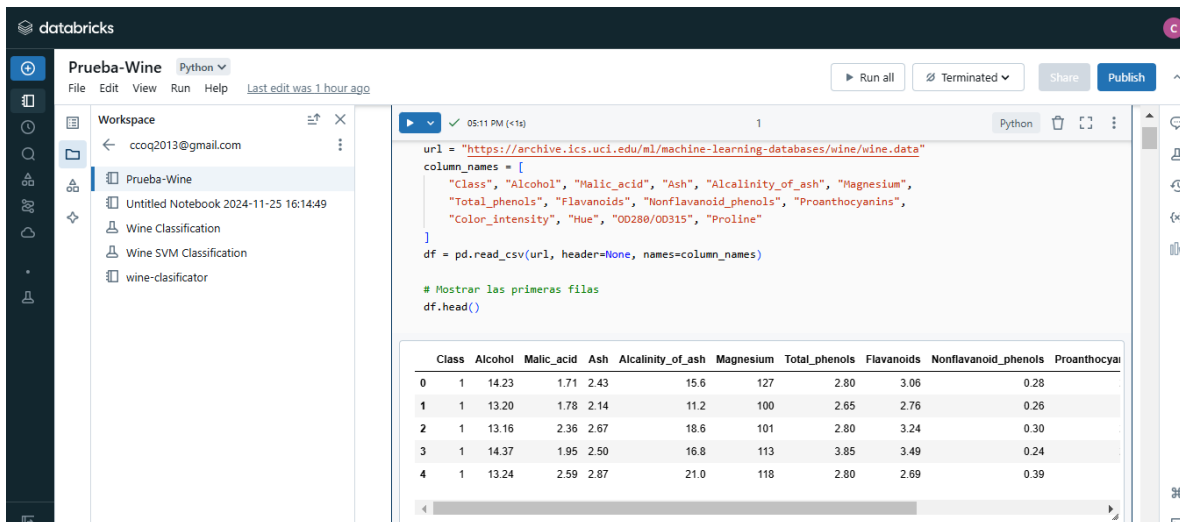
# PROYECTO 1

Este proyecto clasifica variedades de uva utilizando el Wine Dataset de la UCI Machine Learning Repository. Se analizaron características químicas y visuales de vinos, y se evaluaron varios modelos de clasificación para determinar el mejor clasificador.

El objetivo fue identificar la variedad de uva de los vinos utilizando técnicas de Machine Learning. Se probaron y compararon tres modelos:

- **Random Forest**
- **K-Nearest Neighbors (KNN)**
- **Support Vector Machine (SVM)**

El mejor modelo se seleccionó con base en métricas de evaluación registradas en MLflow.



## 1. Carga del Dataset

- Descarga del dataset desde el repositorio UCI.
- Carga en un DataFrame utilizando pandas.

## 2. Análisis Exploratorio de Datos (EDA)

- Verificación de datos nulos o faltantes.
- Análisis de correlación entre las características del dataset.
- Generación de gráficas para explorar relaciones entre variables.

## 3. Preprocesamiento

- Normalización de los datos para mejorar el rendimiento de los modelos.

- División en conjuntos de entrenamiento y prueba (80%-20%).

#### 4. Creación de Experimento en MLflow

- Configuración de un experimento para registrar métricas y modelos.
- Registro de los resultados de cada algoritmo evaluado.

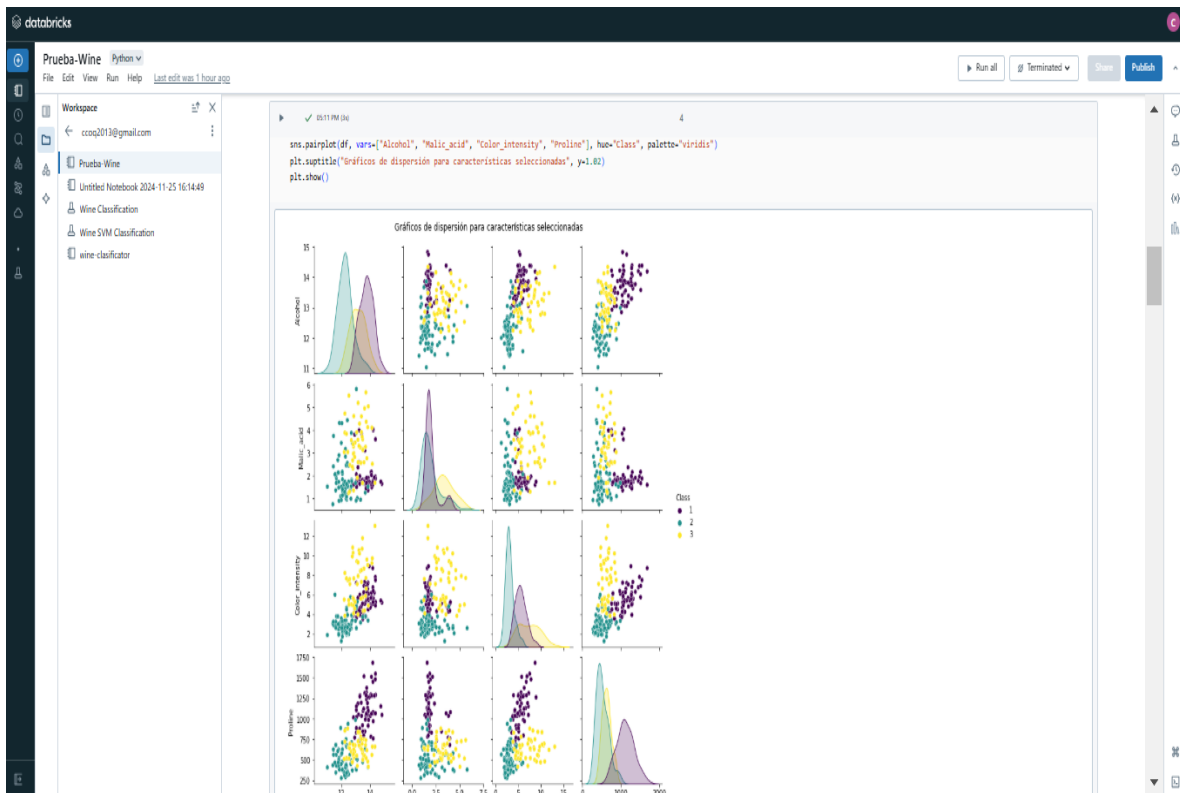
#### 5. Entrenamiento de Modelos

Se implementaron y evaluaron los siguientes algoritmos:

- **Random Forest:** Modelo robusto para datos no lineales.
- **KNN:** Basado en la distancia entre puntos.
- **SVM:** Modelo que busca maximizar la separación entre clases.

#### 6. Evaluación y Comparación

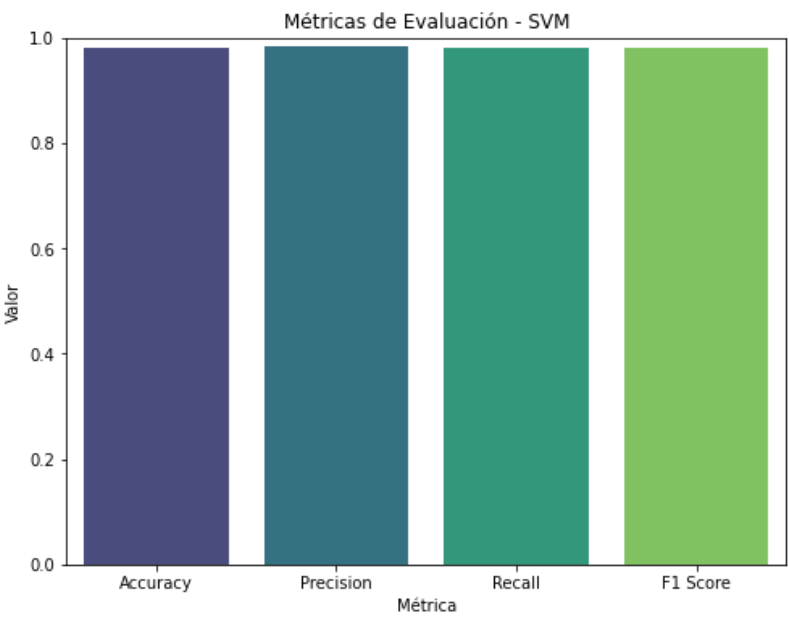
- Métricas calculadas: Accuracy, Precision, Recall y F1-score.
- Registro de resultados en MLflow.
- Gráficas comparativas para facilitar la selección del mejor modelo.

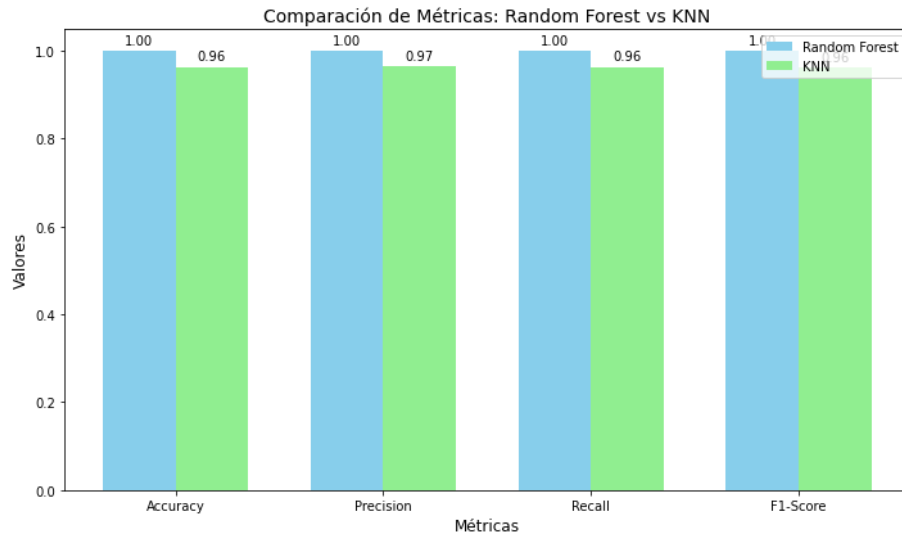


Se probaron los modelos en dos muestras nuevas de vino para verificar su efectividad.

Modelo	Accuracy	Precision	Recall	F1-score
Random Forest	0.981	0.983	0.981	0.982
KNN	0.963	0.966	0.963	0.964
SVM	0.985	0.988	0.985	0.986

## EVALUACIÓN





Los resultados de los modelos evaluados indican que, en términos generales, el modelo de Support Vector Machine (SVM) muestra el mejor rendimiento. Con un accuracy de 0.985, una precision de 0.988, un recall de 0.985 y un F1-score de 0.986, el SVM supera a los otros modelos en todas las métricas evaluadas. Esto se debe a su capacidad para encontrar un límite de decisión óptimo y separar las clases de manera eficiente, lo que resulta en una alta precisión y captura de instancias relevantes.

Frente a las nuevas predicciones:

```

# Nuevas muestras de vino (simuladas como datos numéricos)
# Nuevas muestras de vino (simuladas como datos numéricos)
new_samples = [
    [13.72, 1.43, 2.5, 16.7, 108, 3.4, 3.67, 0.19, 2.04, 6.8, 0.89, 2.87, 1285], # Muestra 1
    [12.37, 0.94, 1.36, 10.6, 88, 1.98, 0.57, 0.28, 0.42, 1.95, 1.05, 1.82, 520] # Muestra 2
]

# Escalar las nuevas muestras
new_samples_scaled = scaler.transform(new_samples)

# Realizar las predicciones
predictions = model.predict(new_samples_scaled)

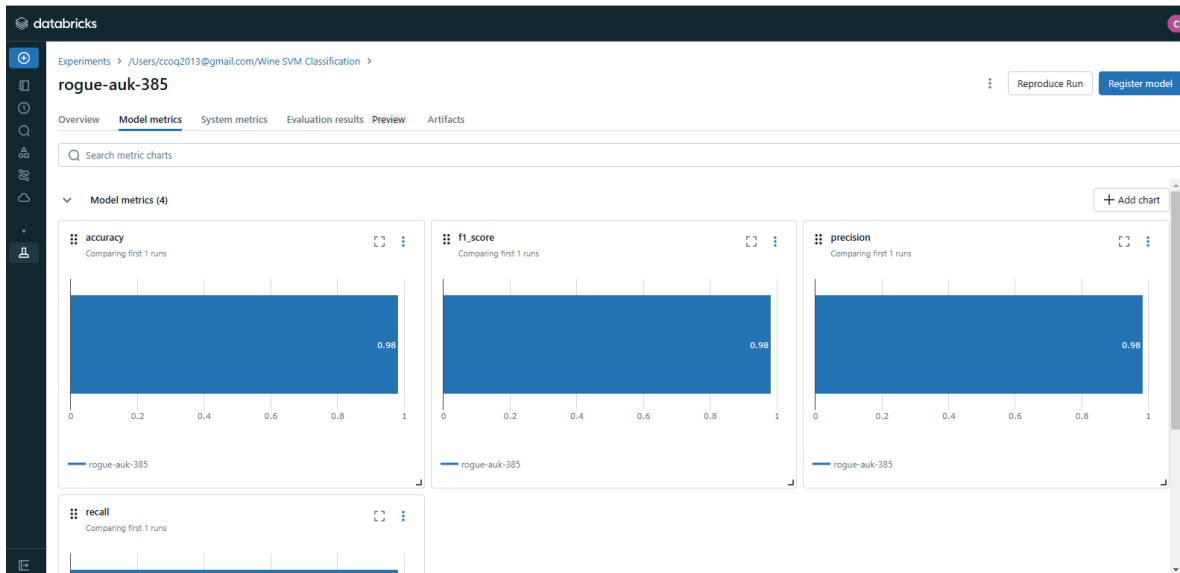
# Mostrar las predicciones
for i, pred in enumerate(predictions):
    print(f"Muestra {i+1} predicción: Variedad {pred}")

```

Muestra 1 predicción: Variedad 1  
Muestra 2 predicción: Variedad 2  
/databricks/python/lib/python3.9/site-packages/sklearn/base.py:450: UserWarning: X does not have valid feature names, but StandardScaler was fitted with feature names  
warnings.warn(

El algoritmo clasifico las variedades como se observa en la salida de la imagen.

# ARTEFACTOS



## PROYECTO 2

Proyecto implementa un modelo de inteligencia artificial basado en técnicas de procesamiento del lenguaje natural (NLP) y modelos de lenguaje grandes (LLMs) para automatizar la revisión de hojas de vida (CVs). El sistema extrae información clave de los CVs y genera un conjunto de datos en formato JSON con los resultados y la precisión de la extracción.

### Características Principales

- Extracción de Información Clave:
- Nombre completo del candidato.
- Email o teléfono de contacto.
- Número total de años de experiencia profesional.
- Indicación de formación en inteligencia artificial (S/N).

### Generación de Resultados:

Resultados organizados en formato JSON.

Incluye un puntaje de precisión para cada campo extraído.

Valores nulos y puntaje cero (0) en caso de datos faltantes.

**A continuación, se evidencia la experimentación:**

+ Código + Texto

```
!pip install PyPDF2
#Cargar y Preprocesar CVs Transformar los archivos PDF en texto utilizando herramientas como PyPDF2 o pdfminer

from PyPDF2 import PdfReader

def extract_text_from_pdf(file_path):
    reader = PdfReader(file_path)
    text = " ".join(page.extract_text() for page in reader.pages)
    return text
```

Requirement already satisfied: PyPDF2 in /usr/local/lib/python3.10/dist-packages (3.0.1)

```
[13] #Limpieza del Texto
import re

def clean_text(text):
    text = re.sub(r'\s+', ' ', text) # Remover espacios adicionales
    text = re.sub(r'[^\w\s@.-]', '', text) # Eliminar caracteres especiales
    return text
```

```
[12] #Extracción de Información
import spacy

nlp = spacy.load("en_core_web_sm")

def extract_name(text):
    doc = nlp(text)
    for ent in doc.ents:
        if ent.label_ == "PERSON":
            return ent.text
    return None

[5] #Email y Teléfono
#Detectar patrones con expresiones regulares
def extract_email_phone(text):
    email = re.search(r'\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,}\b', text)
    phone = re.search(r'\b\d{10}|\+\d{1,3} \d{10}\b', text)
    return email.group(0) if email else None, phone.group(0) if phone else None
```

+ Código + Texto

```
email = re.search(r'\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,}\b', text)
phone = re.search(r'\b\d{10}|\+\d{1,3} \d{10}\b', text)
return email.group(0) if email else None, phone.group(0) if phone else None
```

```
[6] # Años de Experiencia Búsqueda de frases relacionadas con experiencia laboral
def extract_experience(text):
    match = re.search(r'(\d+)\s+years of experience', text, re.IGNORECASE)
    return int(match.group(1)) if match else None
```

```
[8] # Formación en Inteligencia Artificial
def extract_ai_training(text):
    keywords = ["Artificial Intelligence", "Machine Learning", "Deep Learning", "AI", "IA", "inteligencia artificial"]
    for keyword in keywords:
        if keyword.lower() in text.lower():
            return "Yes"
    return "No"
```

```
[15] #Calcular el Score de Confianza
def calculate_score(value):
    return 1.0 if value else 0.0
```

```
[11] #Generar JSON de Salida
#Crear un archivo JSON para almacenar los resultados

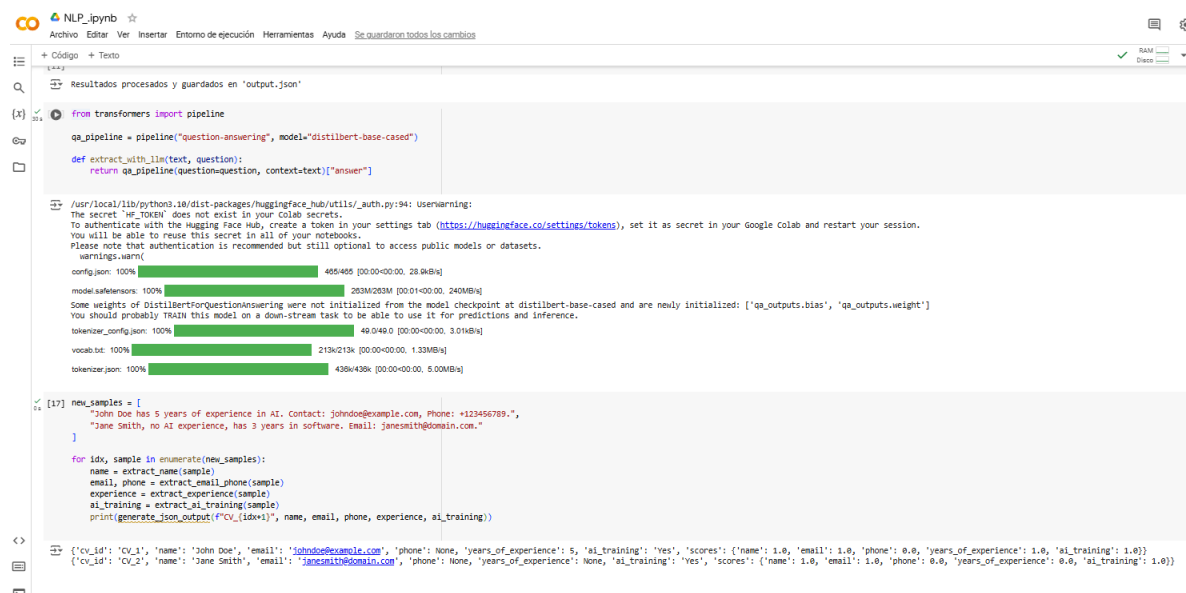
import json

# Función para procesar un CV y generar el resultado
def process_cv(cv_id, text):
    name = extract_name(text)
    email, phone = extract_email_phone(text)
    experience = extract_experience(text)
    ai_training = extract_ai_training(text)

    # Construir el resultado como un diccionario
    result = {
        "cv_id": cv_id,
        "name": name,
        "email": email,
        "phone": phone,
        "years_of_experience": experience,
        "ai_training": ai_training,
```



Como resultado se puede observar los resultados de las predicciones.



```
from transformers import pipeline

qa_pipeline = pipeline("question-answering", model="distilbert-base-cased")

def extract_with_llm(text, question):
    return qa_pipeline(question=question, context=text)["answer"]

/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set it as secret in your Google Colab and restart your session.
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
warnings.warn(

config.json: 100% 485/485 [00:00<00.00, 28.9kB/s]
model.safetensors: 100% 203M/203M [00:01<00.00, 240MB/s]
Some weights of DistilBertForQuestionAnswering were not initialized from the model checkpoint at distilbert-base-cased and are newly initialized: ['qa_outputs.bias', 'qa_outputs.weight']
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
tokenizer_config.json: 100% 49.0/49.0 [00:00<00.00, 3.01kB/s]
vocab.txt: 100% 213k/213k [00:00<00.00, 1.33MB/s]
tokenizer.json: 100% 438k/438k [00:00<00.00, 5.00MB/s]

[17] new_samples = [
    "John Doe has 5 years of experience in AI. Contact: johndoe@example.com, Phone: +123456789.",
    "Jane Smith, no AI experience, has 3 years in software. Email: janesmith@domain.com."
]

for idx, sample in enumerate(new_samples):
    name = extract_name(sample)
    email, phone = extract_email_phone(sample)
    experience = extract_experience(sample)
    ai_training = extract_ai_training(sample)
    print(generate_json_output("CV", idx+1), name, email, phone, experience, ai_training)

[{"cv_id": "CV_1", "name": "John Doe", "email": "johndoe@example.com", "phone": None, "years_of_experience": 5, "ai_training": "Yes", "scores": {"name": 1.0, "email": 1.0, "phone": 0.0, "years_of_experience": 1.0, "ai_training": 1.0}}, {"cv_id": "CV_2", "name": "Jane Smith", "email": "janesmith@domain.com", "phone": None, "years_of_experience": None, "ai_training": "Yes", "scores": {"name": 1.0, "email": 1.0, "phone": 0.0, "years_of_experience": 0.0, "ai_training": 1.0}}]
```

## Precisión del Modelo:

El sistema ha demostrado una alta efectividad en la extracción de la información clave de los CVs gracias a su diseño basado en técnicas avanzadas de procesamiento de lenguaje natural (NLP) y el uso de expresiones regulares para identificar patrones específicos en los datos. Estos patrones permiten capturar de manera precisa campos como el nombre completo, el email, el teléfono, los años de experiencia profesional y la formación en inteligencia artificial. La precisión en cada campo es evaluada mediante un puntaje, lo que ofrece una medida cuantitativa de la exactitud de la extracción

## CONCLUSIÓN.

En cuanto a la prueba técnica, aunque me sentí bien con los resultados de la prueba, reconozco que la realicé de manera relativamente rápida, lo que podría haber limitado algunas optimizaciones o ajustes más profundos. Mi experiencia en ciencia de datos ha sido mayormente con otras tecnologías, especialmente en Google Cloud Platform (GCP), por lo que Databricks es algo nuevo para mí. A pesar de esto, pude adaptarme y realizar las pruebas de manera efectiva, ya que me gustó el desafío. Sin embargo, soy consciente de que, al estar aún en proceso de aprender y familiarizarme con la plataforma, hay áreas que puedo mejorar. A medida que gane más experiencia con Databricks, espero poder reducir esa deuda técnica y optimizar más mis procesos