Clasificación de vinos

Adrián Yared Armas de la Nuez

Contenido

1. Objetivo

El objeto de esta actividad es poner en práctica los conocimientos adquiridos hasta el momento para ellos vamos a utilizar el siguiente dataset que contiene una serie de características físico-químicas que determina la calidad del vino en una escala de valores del 1 al 10.
El enlace donde se encuentran los dataset es el siguiente:
https://archive.ics.uci.edu/ml/datasets/Wine+Quality
Como proyecto de partida se puede utilizar el ejemplo:
Título: Ejemplo_3_3_Clasificación_con_Naive_Bayes_(Heart_Diseases)
Url:
https://colab.research.google.com/drive/1hwri6X-N_cHmpZs31-zyK2XwRyfA4EGN?usp=sharing

2.1 Librerías

2.1.1 Código

```
import pandas as pd  # Data handling
import numpy as np  # Numerical calculations
import matplotlib.pyplot as plt  # Data visualization
import seaborn as sns  # Statistical graphics
from sklearn.model_selection import train_test_split, cross_val_score
# Data splitting and validation
from sklearn.naive_bayes import GaussianNB  # Naïve Bayes model
from sklearn.neighbors import KNeighborsClassifier  # KNN model
from sklearn.metrics import confusion_matrix, accuracy_score  #
Evaluation metrics
from sklearn.feature_selection import SelectKBest, f_classif  # Feature
selection
import joblib  # Save and load models
import seaborn as sns  # Confusion matrix visualization
from sklearn.preprocessing import StandardScaler  # Data scaling
```

3 Vino tinto

3.1 Carga del dataset

3.1.1 Código

```
# Load datasets
```

```
red_wine =
pd.read_csv('https://archive.ics.uci.edu/ml/machine-learning-databases/
wine-quality/winequality-red.csv', sep=';') # Red Wine
```

3.1.2 Separado de características

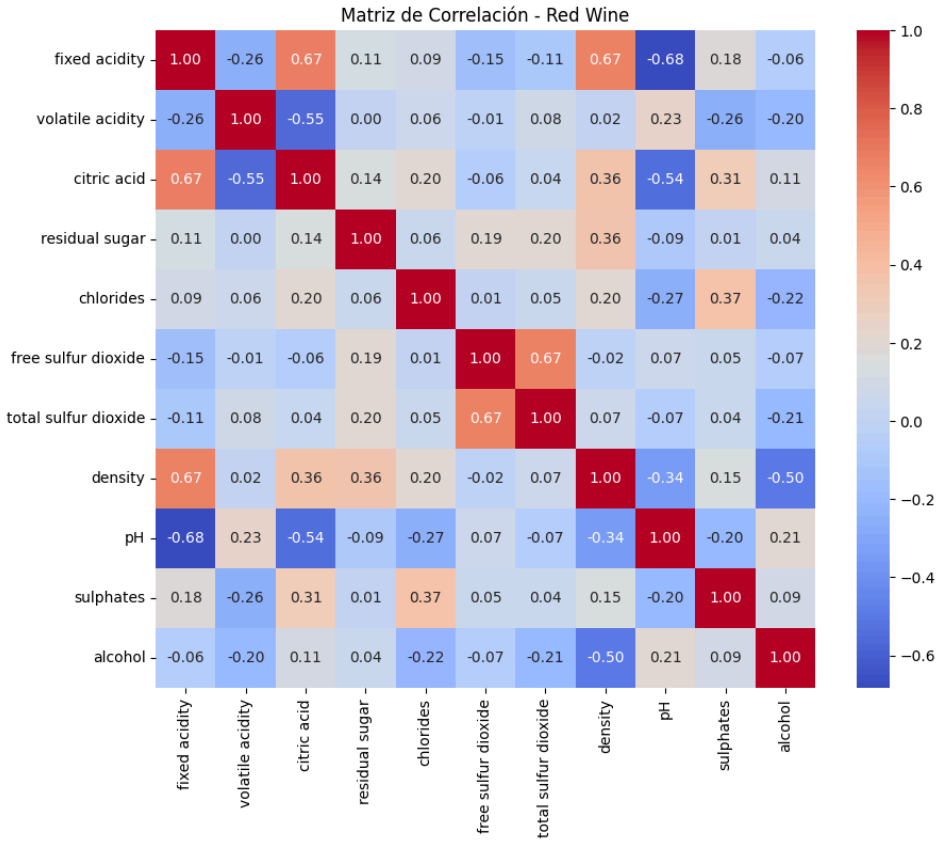
```
X_red = red_wine.drop(columns=['quality'])
y_red = red_wine['quality']
```

3.2 Matriz de correlación

3.2.1 Código

```
# Display the correlation matrix
plt.figure(figsize=(10, 8))
sns.heatmap(X_red.corr(), annot=True, cmap='coolwarm', fmt='.2f')
plt.title("Matriz de Correlación - Red Wine")
plt.show()
```

3.2.2 Resultado



3.3 Selección de características (SelectKBest)

3.3.1 Código

```
# Feature selection using SelectKBest
k = 8  # Select the top 8 features
selector = SelectKBest(score_func=f_classif, k=k)
X_red_selected = selector.fit_transform(X_red, y_red)
selected_features = X_red.columns[selector.get_support()]
print(f"Selected features: {selected_features.tolist()}")
```

3.3.2 Resultado

```
Selected features: ['fixed acidity', 'volatile acidity', 'citric acid', 'chlorides', 'total sulfur dioxide', 'density', 'sulphates', 'alcohol']
```

3.4 comparativa de la precisión KNN y Naive Bayes

3.4.1 Código

```
# Comparison of Naïve Bayes and KNN using cross-validation
X_train, X_test, y_train, y_test = train_test_split(X_red_selected,
y_red, test_size=0.2, random_state=42)
# Data scale for a better presition
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

3.4.2 Naive Bayes

```
nb = GaussianNB()
nb_scores = cross_val_score(nb, X_train, y_train, cv=5,
scoring='accuracy')
print(f"Naïve Bayes - Average Accuracy: {nb_scores.mean():.4f}")
```

```
Naïve Bayes - Average Accuracy: 0.5614
```

3.4.3 Knn

```
knn = KNeighborsClassifier(n_neighbors=5)
knn_scores = cross_val_score(knn, X_train, y_train, cv=5,
scoring='accuracy')
print(f"KNN (k=5) - Average Accuracy: {knn_scores.mean():.4f}")
```

```
KNN (k=5) - Average Accuracy: 0.5950
```

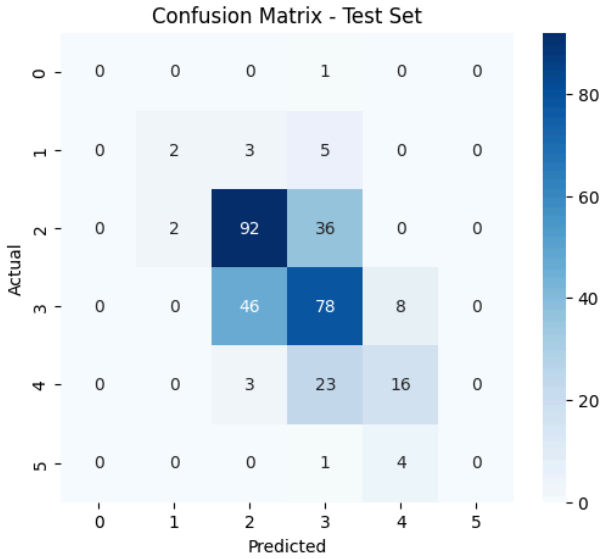
3.4.4 Selección del mejor modelo

```
best_model = nb if nb_scores.mean() > knn_scores.mean() else knn
best_model.fit(X_train, y_train)
```

3.5 Tras elegir el mejor modelo

3.5.1 Entrenamiento y matriz de confusión

```
# Train the best model and obtain the confusion matrix
y_pred = best_model.predict(X_test)
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6,5))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix - Test Set")
plt.show()
```



3.5.2 Exportar el modelo

```
# Export the model
joblib.dump(best_model, 'best_wine_model.pkl')
```

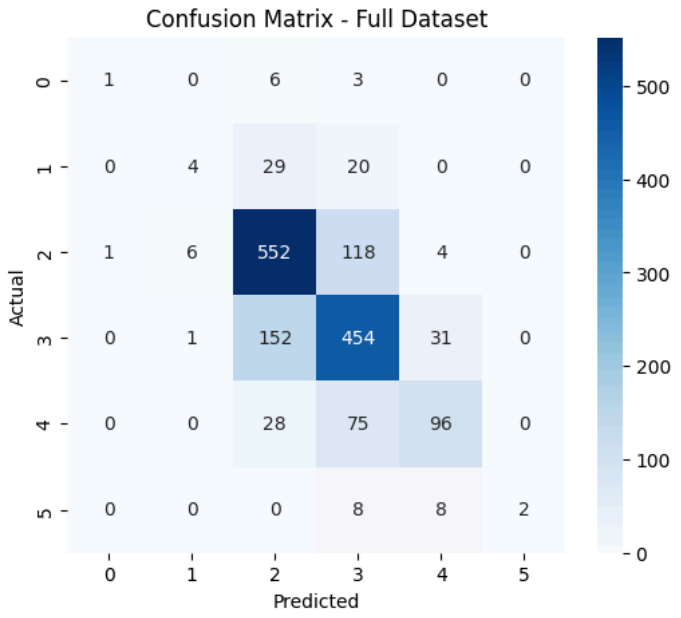
```
['best_wine_model.pkl']
```

3.5.3 Importar el modelo

```
# Import the model
loaded_model = joblib.load('best_wine_model.pkl')
```

3.5.4 Predicción y matriz de confusión

```
# Apply the model to the entire dataset and obtain the confusion matrix
y_all_pred = loaded_model.predict(scaler.transform(X_red_selected))
overall_cm = confusion_matrix(y_red, y_all_pred)
plt.figure(figsize=(6,5))
sns.heatmap(overall_cm, annot=True, fmt='d', cmap='Blues')
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix - Full Dataset")
plt.show()
```



3.6 Comparar el resultado

```
# Compare accuracy
accuracy = accuracy_score(y_red, y_all_pred)
print(f"Final Accuracy: {accuracy:.4f}")
```

```
Final Accuracy: 0.6936
```

4 Vino blanco

4.1 Carga del dataset

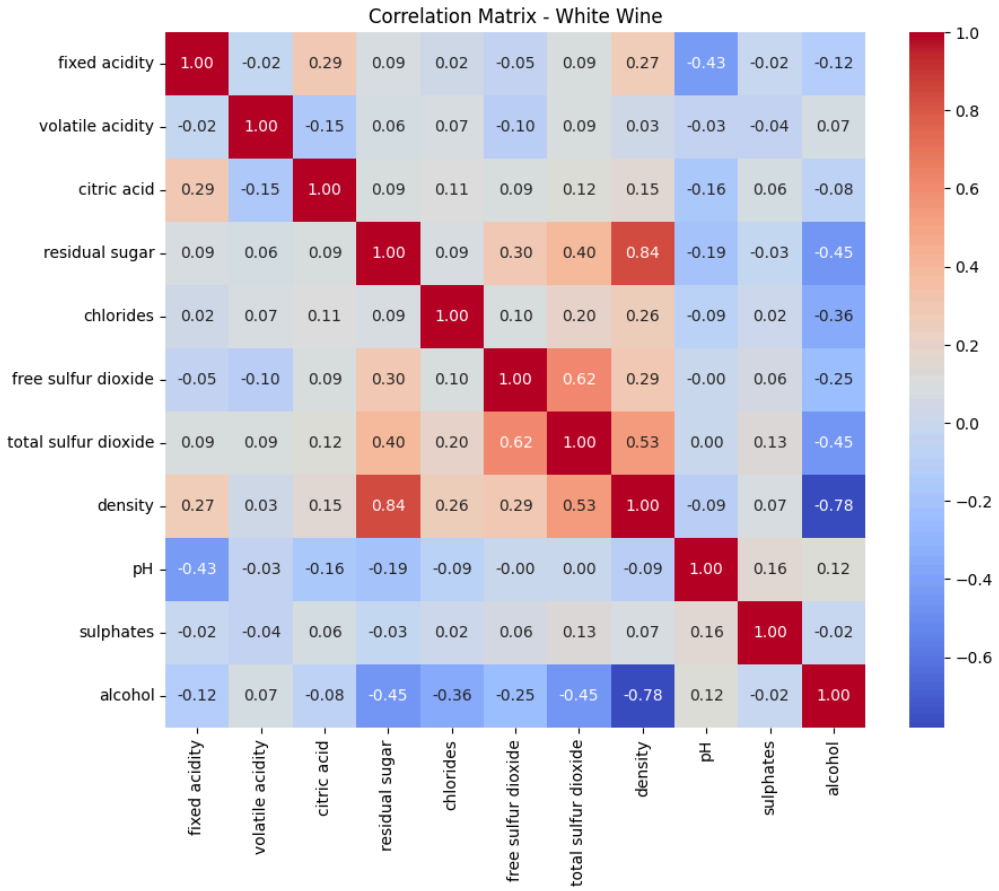
```
# Fetch dataset
white_wine =
pd.read_csv('https://archive.ics.uci.edu/ml/machine-learning-databases/
wine-quality/winequality-white.csv', sep=';')
```

4.1.1 Separado de características

```
# data (as pandas dataframes) (white wine)
X_white = white_wine.drop(columns=['quality'])
y_white = white_wine['quality']
```

4.2 Matriz de correlación

```
# Display the correlation matrix
plt.figure(figsize=(10, 8))
sns.heatmap(X_white.corr(), annot=True, cmap='coolwarm', fmt='.2f')
plt.title("Correlation Matrix - White Wine")
plt.show()
```



4.3 Selección de características

```
# Feature selection using SelectKBest
k = 8  # Select the top 8 features
selector = SelectKBest(score_func=f_classif, k=k)
X_white_selected = selector.fit_transform(X_white, y_white)
selected_features = X_white.columns[selector.get_support()]
print(f"Selected features: {selected_features.tolist()}")
```

4.4 Comparación de precisión

```
# Comparison of Naïve Bayes and KNN using cross-validation
X_train, X_test, y_train, y_test = train_test_split(X_white_selected,
y_white, test_size=0.2, random_state=42)
# Data scale for a better presition
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

4.4.1 Naive bayes

```
nb = GaussianNB()
nb_scores = cross_val_score(nb, X_train, y_train, cv=5,
scoring='accuracy')
print(f"Naïve Bayes - Average Accuracy: {nb_scores.mean():.4f}")
```

```
Naïve Bayes - Average Accuracy: 0.4400
```

4.4.2 Knn

```
knn = KNeighborsClassifier(n_neighbors=5)
knn_scores = cross_val_score(knn, X_train, y_train, cv=5,
scoring='accuracy')
print(f"KNN (k=5) - Average Accuracy: {knn_scores.mean():.4f}")
```

```
KNN (k=5) - Average Accuracy: 0.5350
```

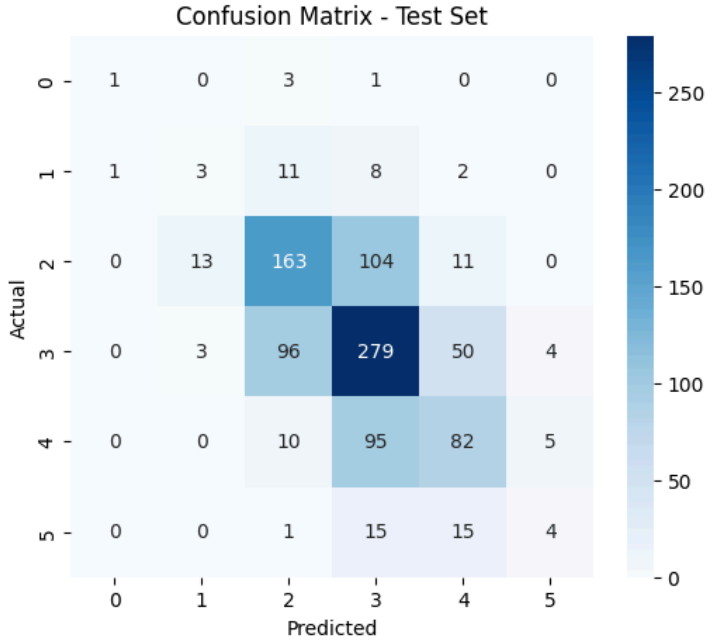
4.4.3 Comparación de modelos

```
best_model = nb if nb_scores.mean() > knn_scores.mean() else knn
best_model.fit(X_train, y_train)
```

```
▾ KNeighborsClassifier
KNeighborsClassifier()
```

4.5 Entrenamiento y matriz de confusión

```
# Train the best model and obtain the confusion matrix
y_pred = best_model.predict(X_test)
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6,5))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix - Test Set")
plt.show()
```



4.5.2 Exportar el modelo

```
# Export the model
joblib.dump(best_model, 'best_wine_model_white.pkl')
```

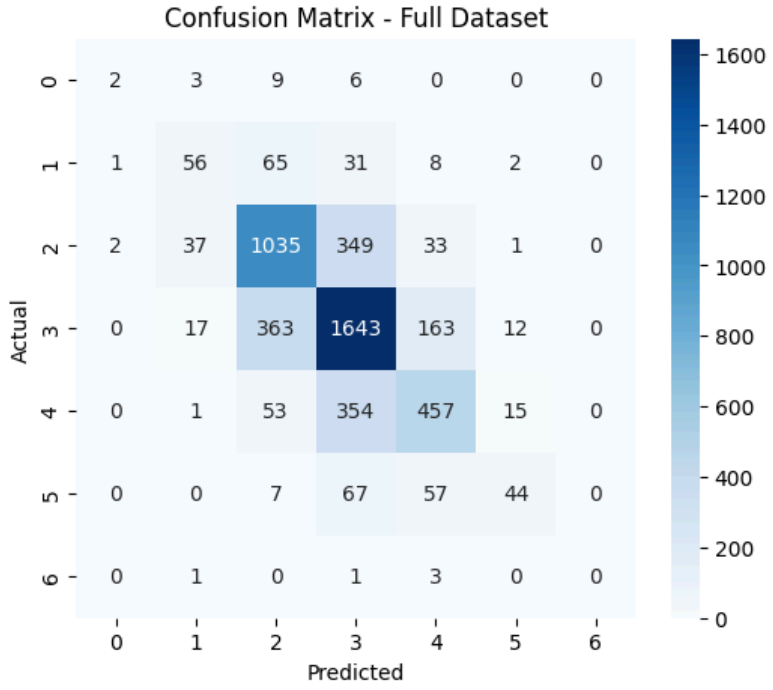
```
['best_wine_model_white.pkl']
```

4.5.3 Importar el modelo

```
# Import the model
loaded_model = joblib.load('best_wine_model_white.pkl')
```

4.5.4 Aplicar la predicción y la matriz de confusión

```
# Apply the model to the entire dataset and obtain the confusion matrix
y_all_pred = loaded_model.predict(scaler.transform(X_white_selected))
overall_cm = confusion_matrix(y_white, y_all_pred)
plt.figure(figsize=(6,5))
sns.heatmap(overall_cm, annot=True, fmt='d', cmap='Blues')
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix - Full Dataset")
plt.show()
```



4.6 Comparación del resultado

```
# Compare accuracy
accuracy = accuracy_score(y_white, y_all_pred)
print(f"Final Accuracy: {accuracy:.4f}")
```

```
Final Accuracy: 0.6609
```

5. Github y Colab