

```
# Project 2 - Practical Machine Learning
# Name: Asraa Alwattar

from google.colab import files
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

plt.style.use("seaborn-v0_8")

uploaded = files.upload()

filename = list(uploaded.keys())[0]
print("تم رفع الملف:", filename)

df = pd.read_csv(filename)

print("Shape of data:", df.shape)
print("\nFirst 5 rows:")
print(df.head())

print("\nMissing values per column (before handling):")
print(df.isna().sum())
```

 WineQT.csv

WineQT.csv(text/csv) - 78057 bytes, last modified: 12/7/2025 - 100% done

Saving WineQT.csv to WineQT (1).csv

تم رفع الملف: WineQT (1).csv

Shape of data: (1143, 13)

First 5 rows:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides \
0	7.4	0.70	0.00	1.9	0.076
1	7.8	0.88	0.00	2.6	0.098
2	7.8	0.76	0.04	2.3	0.092
3	11.2	0.28	0.56	1.9	0.075
4	7.4	0.70	0.00	1.9	0.076

	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates \
0	11.0	34.0	0.9978	3.51	0.56
1	25.0	67.0	0.9968	3.20	0.68
2	15.0	54.0	0.9970	3.26	0.65
3	17.0	60.0	0.9980	3.16	0.58
4	11.0	34.0	0.9978	3.51	0.56

	alcohol	quality	Id
0	9.4	5	0
1	9.8	5	1
2	9.8	5	2
3	9.8	6	3
4	9.4	5	4

Missing values per column (before handling):

fixed acidity	0
volatile acidity	0
citric acid	0
residual sugar	0
chlorides	0
free sulfur dioxide	0
total sulfur dioxide	0
density	0
pH	0
sulphates	0
alcohol	0
quality	0
Id	0

dtype: int64

```
target_col = "quality"
```

```
if target_col not in df.columns:
```

```
    raise ValueError("البيانات quality عمود الهدف.")
```

```

y = df[target_col]
X = df.drop(columns=[target_col])

numeric_cols = X.select_dtypes(include=["float64", "int64"]).columns
X = X[numeric_cols]

print("Features shape:", X.shape)
print("Target shape:", y.shape)

y_binary = (y >= 6).astype(int)

print("\nClass distribution (after binarization):")
print(y_binary.value_counts())

X = X.fillna(X.median(numeric_only=True))

print("\nMissing values in X after imputation:")
print(X.isna().sum())

```

```

Features shape: (1143, 12)
Target shape: (1143,)

Class distribution (after binarization):
quality
1    621
0    522
Name: count, dtype: int64

```

```

Missing values in X after imputation:
fixed acidity      0
volatile acidity   0
citric acid        0
residual sugar     0
chlorides          0
free sulfur dioxide 0
total sulfur dioxide 0
density            0
pH                 0
sulphates          0
alcohol            0
Id                 0
dtype: int64

```

```

from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report

X_train, X_test, y_train, y_test = train_test_split(
    X, y_binary, test_size=0.2, random_state=42, stratify=y_binary
)

print("Train shape:", X_train.shape)
print("Test shape:", X_test.shape)

pipe_lr = Pipeline([
    ("scaler", StandardScaler()),
    ("clf", LogisticRegression(max_iter=1000))
])

param_grid_lr = {
    "clf__C": [0.01, 0.1, 1, 10],
    "clf__penalty": ["l2"],
    "clf__solver": ["lbfgs"]
}

grid_lr = GridSearchCV(
    estimator=pipe_lr,
    param_grid=param_grid_lr,
    cv=5,
    scoring="accuracy",
    n_jobs=-1
)

```

```

)

grid_lr.fit(X_train, y_train)

print("\nBest Logistic Regression params:", grid_lr.best_params_)
print("Best CV accuracy (LogReg):", grid_lr.best_score_)

y_pred_lr = grid_lr.predict(X_test)
print("\nTest Accuracy (LogReg):", accuracy_score(y_test, y_pred_lr))
print("\nClassification Report (LogReg):")
print(classification_report(y_test, y_pred_lr))

rf = RandomForestClassifier(random_state=42)

param_grid_rf = {
    "n_estimators": [100, 200],
    "max_depth": [None, 5, 10],
    "min_samples_split": [2, 5]
}

grid_rf = GridSearchCV(
    estimator=rf,
    param_grid=param_grid_rf,
    cv=5,
    scoring="accuracy",
    n_jobs=-1
)

grid_rf.fit(X_train, y_train)

print("\nBest RandomForest params:", grid_rf.best_params_)
print("Best CV accuracy (RF):", grid_rf.best_score_)

y_pred_rf = grid_rf.predict(X_test)
print("\nTest Accuracy (RF):", accuracy_score(y_test, y_pred_rf))
print("\nClassification Report (RF):")
print(classification_report(y_test, y_pred_rf))

```

Train shape: (914, 12)

Test shape: (229, 12)

Best Logistic Regression params: {'clf__C': 1, 'clf__penalty': 'l2', 'clf__solver': 'lbfgs'}

Best CV accuracy (LogReg): 0.7548910106287157

Test Accuracy (LogReg): 0.777292576419214

Classification Report (LogReg):

	precision	recall	f1-score	support
0	0.75	0.77	0.76	105
1	0.80	0.78	0.79	124
accuracy			0.78	229
macro avg	0.78	0.78	0.78	229
weighted avg	0.78	0.78	0.78	229

Best RandomForest params: {'max_depth': 10, 'min_samples_split': 2, 'n_estimators': 200}

Best CV accuracy (RF): 0.780057647270762

Test Accuracy (RF): 0.7947598253275109

Classification Report (RF):

	precision	recall	f1-score	support
0	0.81	0.72	0.76	105
1	0.79	0.85	0.82	124
accuracy			0.79	229
macro avg	0.80	0.79	0.79	229
weighted avg	0.80	0.79	0.79	229

```
from sklearn.feature_selection import SelectKBest, f_classif
```

```
k = 5
```

```
# --- Method 1: SelectKBest (ANOVA F-test) ---
selector_kbest = SelectKBest(score_func=f_classif, k=k)
selector_kbest.fit(X_train, y_train)

mask_kbest = selector_kbest.get_support()
selected_features_kbest = X.columns[mask_kbest]

print("\nTop", k, "features by SelectKBest (ANOVA):")
print(selected_features_kbest)

# --- Method 2: RandomForest feature importances ---
rf_for_fs = RandomForestClassifier(random_state=42)
rf_for_fs.fit(X_train, y_train)

importances = rf_for_fs.feature_importances_
feature_importances = pd.Series(importances, index=X.columns).sort_values(ascending=False)

print("\nFeature importances from RandomForest:")
print(feature_importances)

top_k_rf = feature_importances.head(k).index
print("\nTop", k, "features by RandomForest importance:")
print(top_k_rf)
```

```
Top 5 features by SelectKBest (ANOVA):
Index(['volatile acidity', 'total sulfur dioxide', 'density', 'sulphates',
       'alcohol'],
      dtype='object')

Feature importances from RandomForest:
alcohol      0.176020
sulphates    0.132226
total sulfur dioxide  0.101001
volatile acidity  0.095830
density      0.073835
Id           0.073781
chlorides    0.065930
pH           0.060795
fixed acidity 0.059409
citric acid  0.058387
free sulfur dioxide 0.052094
residual sugar 0.050693
dtype: float64

Top 5 features by RandomForest importance:
Index(['alcohol', 'sulphates', 'total sulfur dioxide', 'volatile acidity',
       'density'],
      dtype='object')
```

```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

# ===== Using SelectKBest features =====
X_train_kbest = selector_kbest.transform(X_train)
X_test_kbest = selector_kbest.transform(X_test)

model_kbest = Pipeline([
    ("scaler", StandardScaler()),
    ("clf", LogisticRegression(max_iter=1000))
])

model_kbest.fit(X_train_kbest, y_train)
y_pred_kbest = model_kbest.predict(X_test_kbest)

print("\n=== Performance using SelectKBest features ===")
print("Accuracy:", accuracy_score(y_test, y_pred_kbest))
print(classification_report(y_test, y_pred_kbest))

# ===== Using RandomForest top-k features =====
X_train_rf_fs = X_train[top_k_rf]
X_test_rf_fs = X_test[top_k_rf]

model_rf_fs = Pipeline([
    ("scaler", StandardScaler()),
    ("clf", LogisticRegression(max_iter=1000))
])
```

```
    ])  
  
    model_rf_fs.fit(X_train_rf_fs, y_train)  
    y_pred_rf_fs = model_rf_fs.predict(X_test_rf_fs)  
  
    print("\n=== Performance using RandomForest top-k features ===")  
    print("Accuracy:", accuracy_score(y_test, y_pred_rf_fs))  
    print(classification_report(y_test, y_pred_rf_fs))  
  
    # ===== Visualization 1: Confusion Matrix for RandomForest (all features) =====  
    cm = confusion_matrix(y_test, y_pred_rf)  
    disp = ConfusionMatrixDisplay(confusion_matrix=cm)  
    disp.plot()  
    plt.title("Random Forest Confusion Matrix (All Features)")  
    plt.show()  
  
    # ===== Visualization 2: Feature Importances bar plot =====  
    plt.figure(figsize=(8, 5))  
    feature_importances.sort_values(ascending=False).plot(kind='bar')  
    plt.title("Feature Importances (RandomForest)")  
    plt.ylabel("Importance")  
    plt.tight_layout()  
    plt.show()
```

```
=== Performance using SelectKBest features ===
Accuracy: 0.777292576419214
      precision    recall  f1-score   support

     0       0.75       0.78       0.76       105
     1       0.81       0.77       0.79       124

 accuracy
macro avg       0.78       0.78       0.78       229
weighted avg       0.78       0.78       0.78       229
```

```
=== Performance using RandomForest top-k features ===
Accuracy: 0.777292576419214
      precision    recall  f1-score   support

     0       0.75       0.78       0.76       105
     1       0.81       0.77       0.79       124

 accuracy
macro avg       0.78       0.78       0.78       229
weighted avg       0.78       0.78       0.78       229
```

