

```

import pandas as pd
import numpy as np

from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.impute import SimpleImputer
from sklearn.pipeline import Pipeline

from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report

from sklearn.feature_selection import SelectKBest, f_classif, RFE

# =====
# 1. Load Dataset
# =====
df = pd.read_csv("WineQT.csv")

print("First 5 rows:")
print(df.head())

print("\nDataset Info:")
print(df.info())

# =====
# 2. Handle Missing Values
# =====
print("\nMissing Values:")
print(df.isnull().sum())

# Separate features and target
X = df.drop("quality", axis=1)
y = df["quality"]

# Impute missing values
imputer = SimpleImputer(strategy='mean')
X_imputed = pd.DataFrame(imputer.fit_transform(X), columns=X.columns)

# =====
# 3. Apply ML Model with Grid Search
# =====
X_train, X_test, y_train, y_test = train_test_split(
X_imputed, y, test_size=0.2, random_state=42
)

```

```

pipeline = Pipeline([
('scaler', StandardScaler()),
('model', RandomForestClassifier())
])

param_grid = {
'model__n_estimators': [50, 100],
'model__max_depth': [None, 10, 20]
}

grid = GridSearchCV(pipeline, param_grid, cv=5, scoring='accuracy')
grid.fit(X_train, y_train)

y_pred = grid.predict(X_test)

print("\n=== Model Performance (All Features) ===")
print("Best Parameters:", grid.best_params_)
print("Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))

# 4. Feature Selection
# =====

# ---- Method 1: SelectKBest ----
selector_kbest = SelectKBest(score_func=f_classif, k=5)
X_kbest = selector_kbest.fit_transform(X_imputed, y)

features_kbest = X.columns[selector_kbest.get_support()]
print("\nTop Features (SelectKBest):", list(features_kbest))

# ---- Method 2: RFE ----
model = RandomForestClassifier()
rfe = RFE(model, n_features_to_select=5)
X_rfe = rfe.fit_transform(X_imputed, y)

features_rfe = X.columns[rfe.support_]
print("\nTop Features (RFE):", list(features_rfe))

# =====
# 5. Train Models on Selected Features

# ---- Using SelectKBest ----
X_train_k, X_test_k, y_train_k, y_test_k = train_test_split(

```

```

X_kbest, y, test_size=0.2, random_state=42
)

model_k = RandomForestClassifier()
model_k.fit(X_train_k, y_train_k)

y_pred_k = model_k.predict(X_test_k)

print("\n=== Performance (SelectKBest Features) ===")
print("Accuracy:", accuracy_score(y_test_k, y_pred_k))
print(classification_report(y_test_k, y_pred_k))

# ---- Using RFE ----
X_train_r, X_test_r, y_train_r, y_test_r = train_test_split(
X_rfe, y, test_size=0.2, random_state=42
)

model_r = RandomForestClassifier()
model_r.fit(X_train_r, y_train_r)

y_pred_r = model_r.predict(X_test_r)

print("\n=== Performance (RFE Features) ===")
print("Accuracy:", accuracy_score(y_test_r, y_pred_r))
print(classification_report(y_test_r, y_pred_r))

# 6. End of Code

print("\nProcess Completed Successfully!")

```