

```

# Student Name: Mohammed Aljohani
# Assignment: Machine Learning Pipeline & Feature Selection

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.impute import SimpleImputer
from sklearn.feature_selection import SelectKBest, f_classif, RFE
from sklearn.metrics import classification_report, accuracy_score

# 1. Download/Load Dataset (Using Titanic dataset as a Kaggle example)
url = "https://raw.githubusercontent.com/datasciencedojo/datasets/master/titanic.csv"
df = pd.read_csv(url)

# 2. Handle missing values
# Imputing numerical with median and categorical with mode
imputer_num = SimpleImputer(strategy='median')
df['Age'] = imputer_num.fit_transform(df[['Age']])

imputer_cat = SimpleImputer(strategy='most_frequent')
df['Embarked'] = imputer_cat.fit_transform(df[['Embarked']]).ravel()

# Drop unnecessary columns
df.drop(['Cabin', 'Name', 'Ticket', 'PassengerId'], axis=1, inplace=True)

# Convert categorical to numeric (One-Hot Encoding)
df = pd.get_dummies(df, columns=['Sex', 'Embarked'], drop_first=True)

# Prepare Features (X) and Target (y)
X = df.drop('Survived', axis=1)
y = df['Survived']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# 3. Apply Machine Learning with Grid Search Optimizer
param_grid = {
    'n_estimators': [50, 100],
    'max_depth': [None, 10, 20],
    'criterion': ['gini', 'entropy']
}

grid_search = GridSearchCV(RandomForestClassifier(random_state=42), param_grid, cv=5)
grid_search.fit(X_train, y_train)
best_model = grid_search.best_estimator_

# Evaluation before feature selection
y_pred_all = best_model.predict(X_test)
acc_all = accuracy_score(y_test, y_pred_all)

# 4. Feature Selection Methods
# Method 1: SelectKBest (Statistical)
selector_kb = SelectKBest(score_func=f_classif, k=4)
X_train_kb = selector_kb.fit_transform(X_train, y_train)

# Method 2: RFE (Recursive Feature Elimination)
selector_rfe = RFE(estimator=RandomForestClassifier(random_state=42), n_features_to_select=4)
selector_rfe.fit(X_train, y_train)
X_train_rfe = selector_rfe.transform(X_train)
X_test_rfe = selector_rfe.transform(X_test)

# 5. Apply Model to Selected Features & Evaluate
best_model.fit(X_train_rfe, y_train)
y_pred_rfe = best_model.predict(X_test_rfe)
acc_rfe = accuracy_score(y_test, y_pred_rfe)

```

```

acc_rfe = accuracy_score(y_test, y_pred_rfe)

# Final Reports
print(f"--- Results for Mohammed Aljohani ---")
print(f"Best GridSearch Parameters: {grid_search.best_params_}")
print(f"Accuracy (All Features): {acc_all:.4f}")
print(f"Accuracy (Selected Features - RFE): {acc_rfe:.4f}")
print("\nDetailed Classification Report (Final Model):")
print(classification_report(y_test, y_pred_rfe))

# Visualizations
plt.figure(figsize=(14, 5))

# Plot 1: Feature Importance
plt.subplot(1, 2, 1)
feat_importances = pd.Series(best_model.feature_importances_, index=X.columns[selector_rfe.support])
feat_importances.sort_values().plot(kind='barh', color='teal')
plt.title('Top 4 Important Features (RFE)')
plt.xlabel('Importance Score')

# Plot 2: Model Performance Comparison
plt.subplot(1, 2, 2)
sns.barplot(x=['All Features', 'Selected Features'], y=[acc_all, acc_rfe], hue=['All Features', 'Selected Features'])
plt.ylim(0, 1)
plt.title('Accuracy Comparison')
plt.ylabel('Score')

plt.tight_layout()
plt.show()

```

```

--- Results for Mohammed Aljohani ---
Best GridSearch Parameters: {'criterion': 'gini', 'max_depth': 10, 'n_estimators': 100}
Accuracy (All Features): 0.8156
Accuracy (Selected Features - RFE): 0.8268

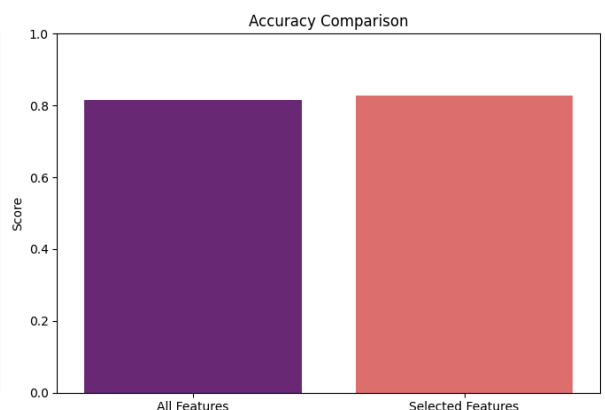
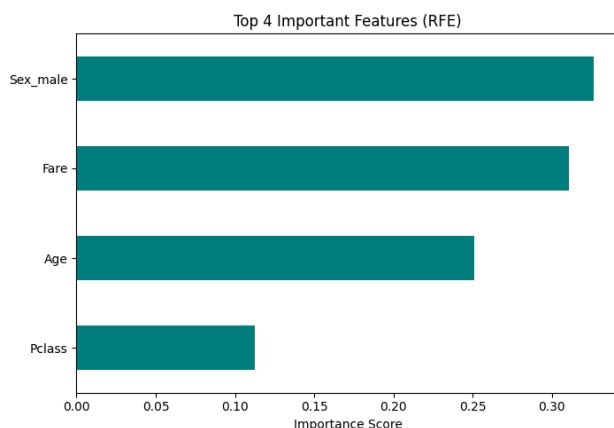
```

```

Detailed Classification Report (Final Model):

```

|              | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| 0            | 0.82      | 0.90   | 0.86     | 105     |
| 1            | 0.84      | 0.72   | 0.77     | 74      |
| accuracy     |           |        | 0.83     | 179     |
| macro avg    | 0.83      | 0.81   | 0.82     | 179     |
| weighted avg | 0.83      | 0.83   | 0.82     | 179     |



```

print("\nDetailed Classification Report (Selected Features - SelectKBest):")
print(classification_report(y_test, y_pred_kb))

```

```

Detailed Classification Report (Selected Features - SelectKBest):

```

|   | precision | recall | f1-score | support |
|---|-----------|--------|----------|---------|
| 0 | 0.83      | 0.86   | 0.85     | 105     |

|              | 1 | 0.79 | 0.76 | 0.77 | 74  |
|--------------|---|------|------|------|-----|
| accuracy     |   |      |      | 0.82 | 179 |
| macro avg    |   | 0.81 | 0.81 | 0.81 | 179 |
| weighted avg |   | 0.81 | 0.82 | 0.82 | 179 |

```
# Transform X_test using the SelectKBest selector
X_test_kb = selector_kb.transform(X_test)

# Apply the best model to the SelectKBest selected features and evaluate
best_model_kb = grid_search.best_estimator_
best_model_kb.fit(X_train_kb, y_train)
y_pred_kb = best_model_kb.predict(X_test_kb)
acc_kb = accuracy_score(y_test, y_pred_kb)

print(f"Accuracy (Selected Features - SelectKBest): {acc_kb:.4f}")
```

Accuracy (Selected Features - SelectKBest): 0.8156

```
# Get the names of the features selected by RFE
selected_features_rfe = X.columns[selector_rfe.support_]
print("Features selected by RFE:")
print(selected_features_rfe.tolist())
```

Features selected by RFE:  
['Pclass', 'Age', 'Fare', 'Sex\_male']

```
# Get the names of the features selected by SelectKBest
selected_features_kb = X_train.columns[selector_kb.get_support()]
print("Features selected by SelectKBest:")
print(selected_features_kb.tolist())
```

Features selected by SelectKBest:  
['Pclass', 'Fare', 'Sex\_male', 'Embarked\_S']