

```
In [10]: print ('بسم الله الرحمن الرحيم')
print ('الاختبار النهائي لمادة ')
print ('Practical Machine Learning')
print ('تقديم الطالب / حسن محمد احمد الامير خليفة')
print ('الرقم الجامعي', '253002941')
print ('اعداد الدكتور / هاجر صالح')
print ('الداتا ست التي تم استخدامها https://www.kaggle.com/datasets/brendan45774/test-file')
```

بسم الله الرحمن الرحيم
الاختبار النهائي لمادة
Practical Machine Learning
تقديم الطالب / حسن محمد احمد الامير خليفة
الرقم الجامعي 253002941
اعداد الدكتور / هاجر صالح
الداتا ست التي تم استخدامها https://www.kaggle.com/datasets/brendan45774/test-file

```
In [4]: import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
In [6]: print ('لعرض اول خمسة صفوف من الداتا ست')
df = pd.read_csv('tested.csv')
df.head()
```

لعرض اول خمسة صفوف من الداتا ست

Out[6]:	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
0	892	0	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292
1	893	1	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000
2	894	0	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875
3	895	0	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625
4	896	1	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875

```
In [11]: print ('لمعرفة عدد التكرار للذكور والاناث في الداتا ست')
df['Sex'].value_counts()
```

لمعرفة عدد التكرار للذكور والاناث في الداتا ست

```
Out[11]: Sex
male      266
female    152
Name: count, dtype: int64
```

```
In [15]: print ('لعرض عدد ال Survived')
df['Survived'].value_counts()
```

لعرض عدد ال Survived

```
Out[15]: Survived
0      266
1      152
Name: count, dtype: int64
```

```
In [12]: print ('لمعرفة نوع البيانات في الداتا ست')
df.dtypes
```

لمعرفة نوع البيانات في الداتا ست

```
Out[12]: PassengerId      int64
Survived      int64
Pclass        int64
Name          object
Sex           object
Age           float64
SibSp         int64
Parch         int64
Ticket        object
Fare          float64
Cabin         object
Embarked      object
dtype: object
```

```
In [19]: print ('لتحويل السمات الفئوية إلى بيانات رقمية')
label_encoders = {}
for column in df.select_dtypes(include=['object']).columns:
    le = LabelEncoder()
    df[column] = le.fit_transform(df[column])
    label_encoders[column] = le
```

لتحويل السمات الفئوية إلى بيانات رقمية

```
In [21]: print ('لتحديد الميزات والمتغير المستهدف')
X = df.drop('Survived', axis=1).values
y = df['Survived'].values
```

لتحديد الميزات والمتغير المستهدف

```
In [22]: print ('لتنقسم البيانات إلى مجموعات تدريب واختبار')
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_st
```

لتنقسم البيانات إلى مجموعات تدريب واختبار

```
In [24]: def evaluate_model(model_name, y_true, y_pred):
    accuracy = accuracy_score(y_true, y_pred)
    precision = precision_score(y_true, y_pred, average='weighted')
    recall = recall_score(y_true, y_pred, average='weighted')
    f1 = f1_score(y_true, y_pred, average='weighted')
    cm = confusion_matrix(y_true, y_pred)
    report = classification_report(y_true, y_pred)

    metrics = {
        'Model Name': model_name,
        'Accuracy': accuracy,
        'Precision': precision,
        'Recall': recall,
        'F1 Score': f1,
        'Classification Report': report
    }

    plt.figure(figsize=(4, 4))
    sns.heatmap(
        cm,
        annot=True,
        fmt='d',
        cmap='Blues',
```

```

        cbar=False,
        xticklabels=np.unique(y_true),
        yticklabels=np.unique(y_true)
    )
    plt.title(f'Confusion Matrix for {model_name}')
    plt.xlabel('Predicted Label')
    plt.ylabel('True Label')
    plt.show()

    return metrics

```

```

In [25]: from sklearn.ensemble import RandomForestClassifier

# 1 إنشاء النموذج
rf_classifier = RandomForestClassifier(n_estimators=95, random_state=40)

# 2 تدريب النموذج
rf_classifier.fit(X_train, y_train)

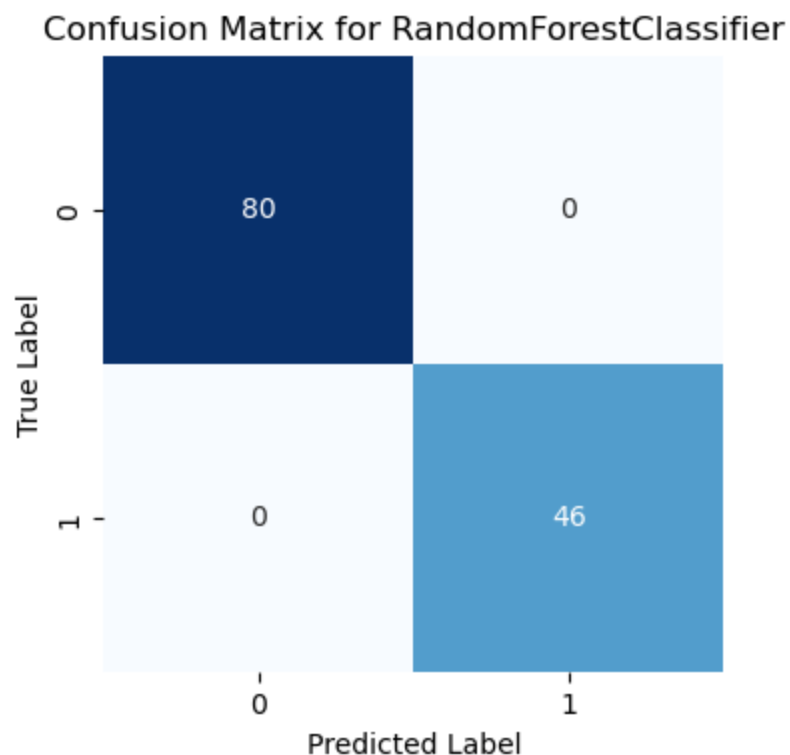
# 3 التنبؤ على مجموعة الاختبار
y_pred = rf_classifier.predict(X_test)

# 4 تقييم النموذج
evaluation_results = evaluate_model('RandomForestClassifier', y_test, y_pred)

# 5 طباعة النتائج بطريقة منظمة
for key, value in evaluation_results.items():
    if key == 'Classification Report':
        print(value)
    else:
        print(f"{key}: {value:.4f}" if isinstance(value, float) else f"{key}: {value}")

# 6 طباعة النتائج مع ألوان
print("\033[94mAccuracy:", evaluation_results['Accuracy'], "\033[0m") # أزرق
print("\033[92mPrecision:", evaluation_results['Precision'], "\033[0m") # أخضر
print("\033[91mRecall:", evaluation_results['Recall'], "\033[0m") # أحمر

```



Model Name: RandomForestClassifier

Accuracy: 1.0000

Precision: 1.0000

Recall: 1.0000

F1 Score: 1.0000

	precision	recall	f1-score	support
0	1.00	1.00	1.00	80
1	1.00	1.00	1.00	46
accuracy			1.00	126
macro avg	1.00	1.00	1.00	126
weighted avg	1.00	1.00	1.00	126

Accuracy: 1.0

Precision: 1.0

Recall: 1.0

In [28]: `print ('الانحدار اللوجستي')`

الانحدار اللوجستي

In [34]: `from sklearn.impute import SimpleImputer`
`imputer = SimpleImputer(strategy='mean')`
`X_train_imputed = imputer.fit_transform(X_train)`
`X_test_imputed = imputer.transform(X_test)`

In [35]: `from sklearn.impute import SimpleImputer`
`imputer = SimpleImputer(strategy='mean')`

```
X_train_imputed = imputer.fit_transform(X_train)
X_test_imputed = imputer.transform(X_test)
```

```
In [37]: for key, value in evaluation_results.items():
         if key == 'Classification Report':
             print(value)
         else:
             print(f"{key}: {value:.4f}" if isinstance(value, float) else f"{key}: {valu
```

Model Name: RandomForestClassifier

Accuracy: 1.0000

Precision: 1.0000

Recall: 1.0000

F1 Score: 1.0000

	precision	recall	f1-score	support
0	1.00	1.00	1.00	80
1	1.00	1.00	1.00	46
accuracy			1.00	126
macro avg	1.00	1.00	1.00	126
weighted avg	1.00	1.00	1.00	126

```
In [38]: print ('لعمل مصنف شجرة القرار')
```

لعمل مصنف شجرة القرار

```
In [39]: from sklearn.tree import DecisionTreeClassifier

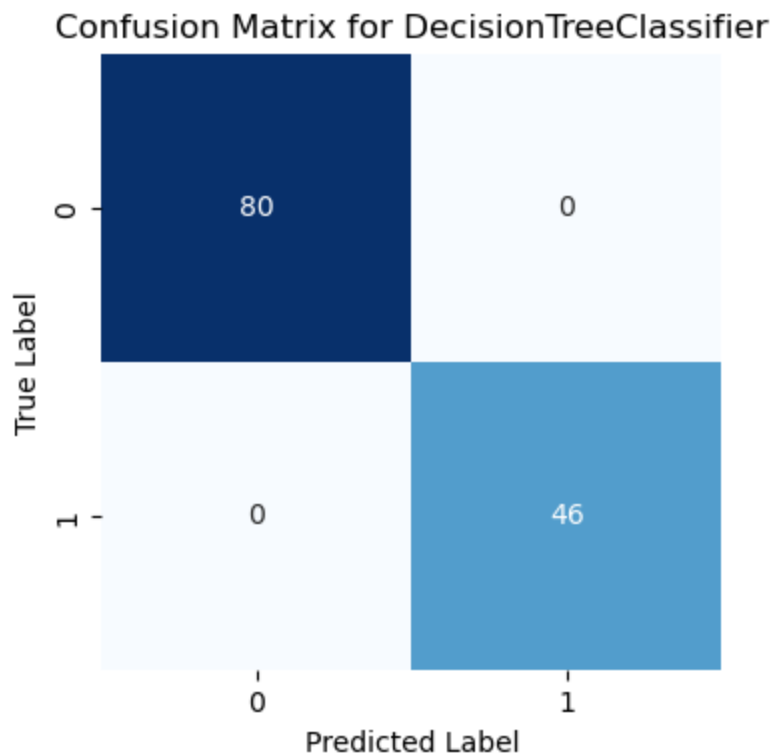
# 1 إنشاء النموذج
DT = DecisionTreeClassifier(random_state=42)

# 2 تدريب النموذج
DT.fit(X_train, y_train)

# 3 التنبؤ
y_pred = DT.predict(X_test)

# 4 تقييم النموذج
evaluation_results = evaluate_model('DecisionTreeClassifier', y_test, y_pred)

# 5 طباعة النتائج
for key, value in evaluation_results.items():
    if key == 'Classification Report':
        print(value) # طباعة التقرير بشكل منفصل
    else:
        print(f"{key}: {value:.4f}" if isinstance(value, float) else f"{key}: {valu
```



Model Name: DecisionTreeClassifier

Accuracy: 1.0000

Precision: 1.0000

Recall: 1.0000

F1 Score: 1.0000

	precision	recall	f1-score	support
0	1.00	1.00	1.00	80
1	1.00	1.00	1.00	46
accuracy			1.00	126
macro avg	1.00	1.00	1.00	126
weighted avg	1.00	1.00	1.00	126

```
In [40]: print ('عمل GaussianNB')
```

عمل GaussianNB

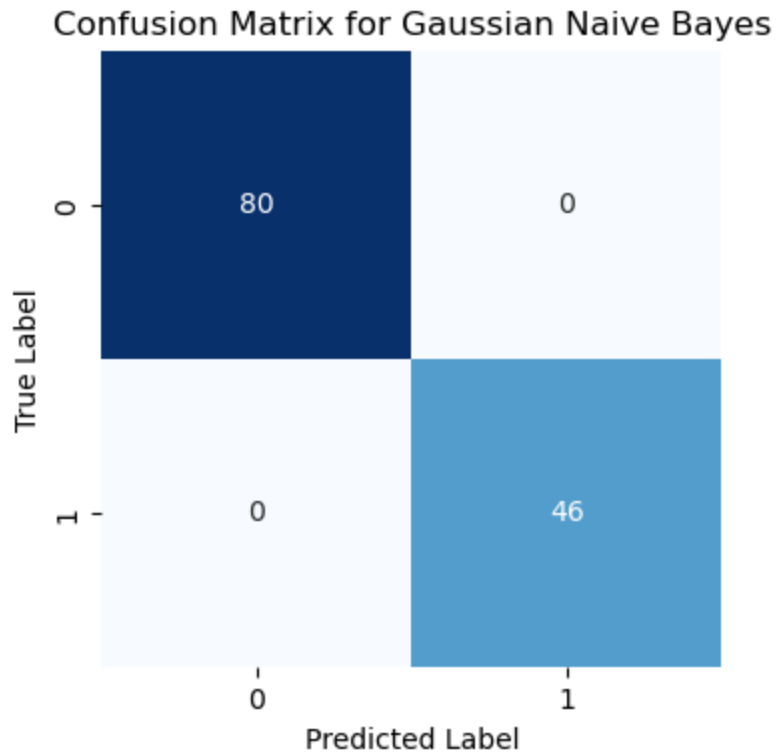
```
In [43]: from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.naive_bayes import GaussianNB

pipeline_gnb = Pipeline([
    ('imputer', SimpleImputer(strategy='mean')),
    ('model', GaussianNB())
])

pipeline_gnb.fit(X_train, y_train)

y_pred = pipeline_gnb.predict(X_test)

evaluation_results = evaluate_model('Gaussian Naive Bayes', y_test, y_pred)
```



```
In [44]: from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.impute import SimpleImputer

# 1 إذا كان لديك معالجة القيم المفقودة (NaN)
imputer = SimpleImputer(strategy='mean')
X_train_imputed = imputer.fit_transform(X_train)
X_test_imputed = imputer.transform(X_test)

# 2 إنشاء النموذج
knn = KNeighborsClassifier()

# 3 تحديد الـ hyperparameters للـ Grid Search
param_grid = {
    'n_neighbors': [1, 3, 5, 7, 9],
    'weights': ['uniform', 'distance'],
    'metric': ['euclidean', 'manhattan'],
    'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute']
}

# 4 إعداد GridSearchCV
grid_search = GridSearchCV(
    estimator=knn,
    param_grid=param_grid,
    cv=5,
    scoring='accuracy',
    n_jobs=-1
)

# 5 تدريب Grid Search
grid_search.fit(X_train_imputed, y_train)
```

```

# 6 اختيار أفضل نموذج
best_knn = grid_search.best_estimator_

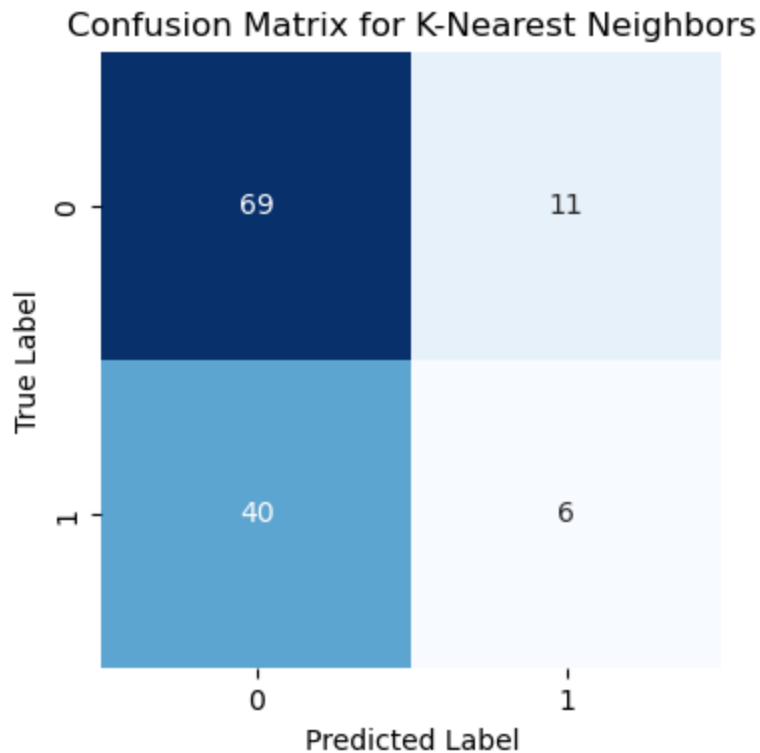
# 7 التنبؤ باستخدام أفضل نموذج
y_pred = best_knn.predict(X_test_imputed)

# 8 تقييم النموذج
evaluation_results = evaluate_model('K-Nearest Neighbors', y_test, y_pred)

# 9 طباعة النتائج
for key, value in evaluation_results.items():
    if key == 'Classification Report':
        print(value)
    else:
        print(f"{key}: {value:.4f}" if isinstance(value, float) else f"{key}: {value}")

# 10 أفضل ال hyperparameters
print("\nBest hyperparameters found by GridSearchCV:")
print(grid_search.best_params_)

```



Model Name: K-Nearest Neighbors

Accuracy: 0.5952

Precision: 0.5308

Recall: 0.5952

F1 Score: 0.5331

	precision	recall	f1-score	support
0	0.63	0.86	0.73	80
1	0.35	0.13	0.19	46
accuracy			0.60	126
macro avg	0.49	0.50	0.46	126
weighted avg	0.53	0.60	0.53	126

Best hyperparameters found by GridSearchCV:

```
{'algorithm': 'auto', 'metric': 'manhattan', 'n_neighbors': 9, 'weights': 'distance'}
```

In [45]: `print ('لعمل مهمة الانحدار')`

لعمل مهمة الانحدار

In [48]: `print ('لعرض تفاصيل الداتا ست')`
`df.describe()`

لعرض تفاصيل الداتا ست

Out[48]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp
count	418.000000	418.000000	418.000000	418.000000	418.000000	332.000000	418.000000
mean	1100.500000	0.363636	2.265550	208.500000	0.636364	30.272590	0.447368
std	120.810458	0.481622	0.841838	120.810458	0.481622	14.181209	0.896760
min	892.000000	0.000000	1.000000	0.000000	0.000000	0.170000	0.000000
25%	996.250000	0.000000	1.000000	104.250000	0.000000	21.000000	0.000000
50%	1100.500000	0.000000	3.000000	208.500000	1.000000	27.000000	0.000000
75%	1204.750000	1.000000	3.000000	312.750000	1.000000	39.000000	1.000000
max	1309.000000	1.000000	3.000000	417.000000	1.000000	76.000000	8.000000

◀ ▶

In [49]: `print ('لعرض الاعمدة المفقودة')`

لعرض الاعمدة المفقودة

In [50]: `df.isnull().sum()`

```
Out[50]: PassengerId    0
Survived    0
Pclass      0
Name        0
Sex          0
Age         86
SibSp       0
Parch       0
Ticket      0
Fare        1
Cabin       0
Embarked    0
dtype: int64
```

```
In [55]: X = df.drop(['Sex', 'Cabin'], axis=1)
y = df['Parch']
X
```

```
Out[55]:
```

	PassengerId	Survived	Pclass	Name	Age	SibSp	Parch	Ticket	Fare	Embarked
0	892	0	3	206	34.5	0	0	152	7.8292	
1	893	1	3	403	47.0	1	0	221	7.0000	
2	894	0	2	269	62.0	0	0	73	9.6875	
3	895	0	3	408	27.0	0	0	147	8.6625	
4	896	1	3	178	22.0	1	1	138	12.2875	
...	...	...	...	...	...	...	...	...	...	...
413	1305	0	3	353	NaN	0	0	267	8.0500	
414	1306	1	1	283	39.0	0	0	324	108.9000	
415	1307	0	3	332	38.5	0	0	346	7.2500	
416	1308	0	3	384	NaN	0	0	220	8.0500	
417	1309	0	3	302	NaN	1	1	105	22.3583	

418 rows × 10 columns



```
In [56]: print ('لعمل ميزات التحجيم (X)')
```

لعمل ميزات التحجيم (X)

```
In [67]: from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

print("Shape:", X_scaled.shape)
print("First 5 rows:\n", X_scaled[:5])
```

Shape: (418, 10)

First 5 rows:

```
[[-1.72791209 -0.75592895  0.87348191 -0.02071837  0.29854934 -0.49947002
 -0.4002477  -0.26949361 -0.49781052 -0.47091535]
 [-1.71962474  1.32287566  0.87348191  1.61188922  1.18132793  0.61699237
 -0.4002477   0.37293425 -0.51265996  0.70076689]
 [-1.71133739 -0.75592895 -0.31581919  0.50138456  2.24066224 -0.49947002
 -0.4002477  -1.00502694 -0.46453181 -0.47091535]
 [-1.70305005 -0.75592895  0.87348191  1.65332596 -0.23111782 -0.49947002
 -0.4002477  -0.31604635 -0.48288766  0.70076689]
 [-1.6947627   1.32287566  0.87348191 -0.25276412 -0.58422925  0.61699237
  0.61989583 -0.39984129 -0.41797062  0.70076689]]
```

```
In [68]: # Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_st
```

```
In [77]: import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

def evaluate_model(model_name, y_true, y_pred):
    # حساب المقاييس
    mae = mean_absolute_error(y_true, y_pred)
    mse = mean_squared_error(y_true, y_pred)
    rmse = np.sqrt(mse)
    r2 = r2_score(y_true, y_pred)

    results = {
        "Mean Absolute Error (MAE)": mae,
        "Mean Squared Error (MSE)": mse,
        "Root Mean Squared Error (RMSE)": rmse,
        "R^2 Score": r2
    }

    # رسم القيم الحقيقية مقابل المتوقعة
    plt.figure(figsize=(8, 6))
    plt.scatter(y_true, y_pred, alpha=0.6, color='blue', label='Predicted vs Actual')
    plt.plot([min(y_true), max(y_true)], [min(y_true), max(y_true)], 'r--', label='True Values')
    plt.xlabel('True Values')
    plt.ylabel('Predicted Values')
    plt.title(f'Regression Model: {model_name} - True vs Predicted Values')
    plt.text(0.05, 0.9, f'R^2 Score: {r2:.2f}', transform=plt.gca().transAxes, font
    plt.legend()
    plt.grid(True)
    plt.show()

    return results
```

```
In [ ]:
```

```
In [78]: from sklearn.ensemble import RandomForestRegressor

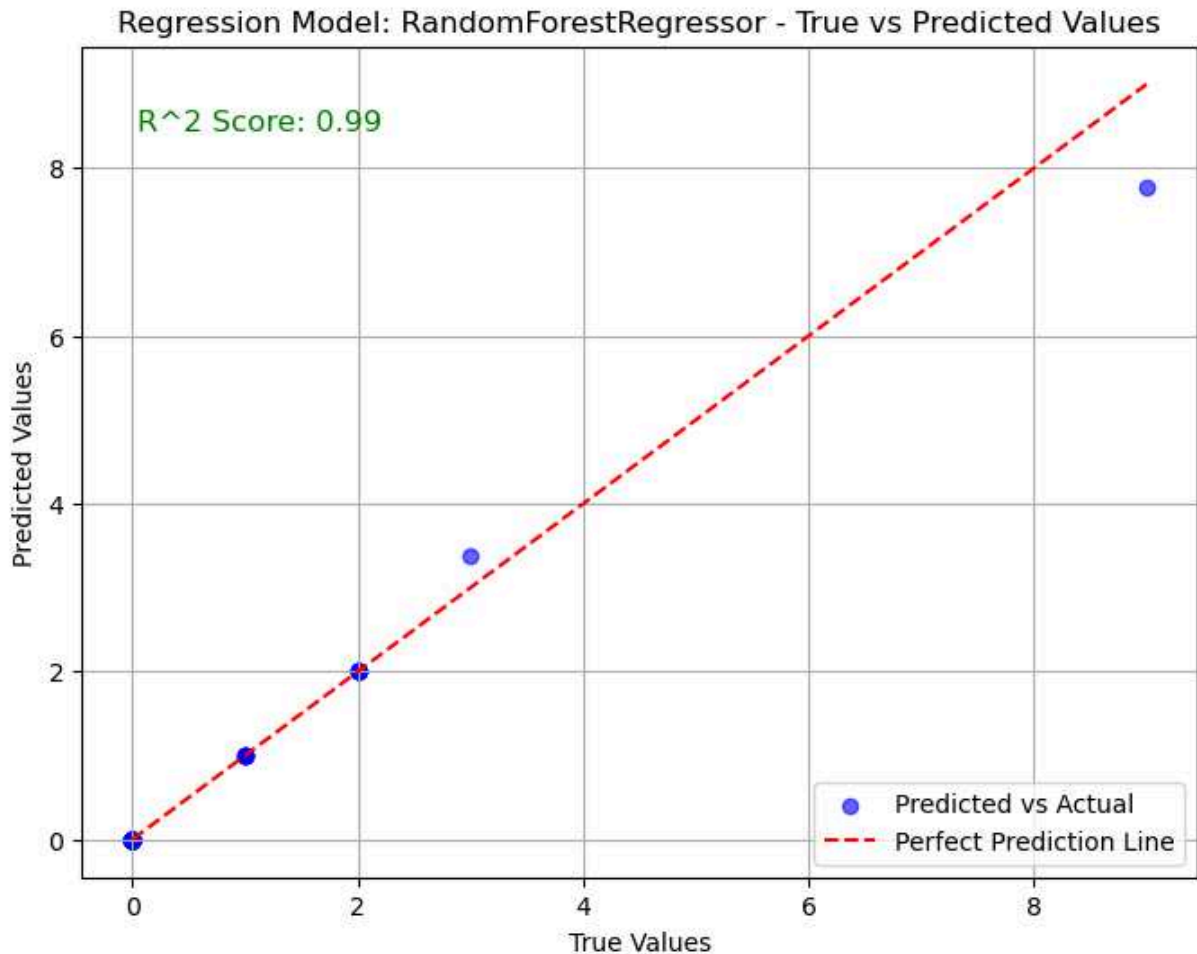
# إنشاء النموذج
rf_regressor = RandomForestRegressor(n_estimators=100, random_state=42)
```

```
# 2 تدريب النموذج
rf_regressor.fit(X_train, y_train)

# 3 التنبؤ على مجموعة الاختبار
y_pred = rf_regressor.predict(X_test)

# 4 تقييم النموذج باستخدام الدالة السابقة
evaluation_results = evaluate_model('RandomForestRegressor', y_test, y_pred)

# 5 طباعة النتائج
evaluation_results
```



```
Out[78]: {'Mean Absolute Error (MAE)': 0.015333333333333336,
'Mean Squared Error (MSE)': 0.015783809523809533,
'Root Mean Squared Error (RMSE)': np.float64(0.1256336321365005),
'R^2 Score': 0.9854937895965321}
```

```
In [79]: print ('العمل نموذج الانحدار لشجرة القرار')
```

العمل نموذج الانحدار لشجرة القرار

```
In [80]: from sklearn.tree import DecisionTreeRegressor
```

```
# 1 إنشاء النموذج
DT = DecisionTreeRegressor(random_state=42)

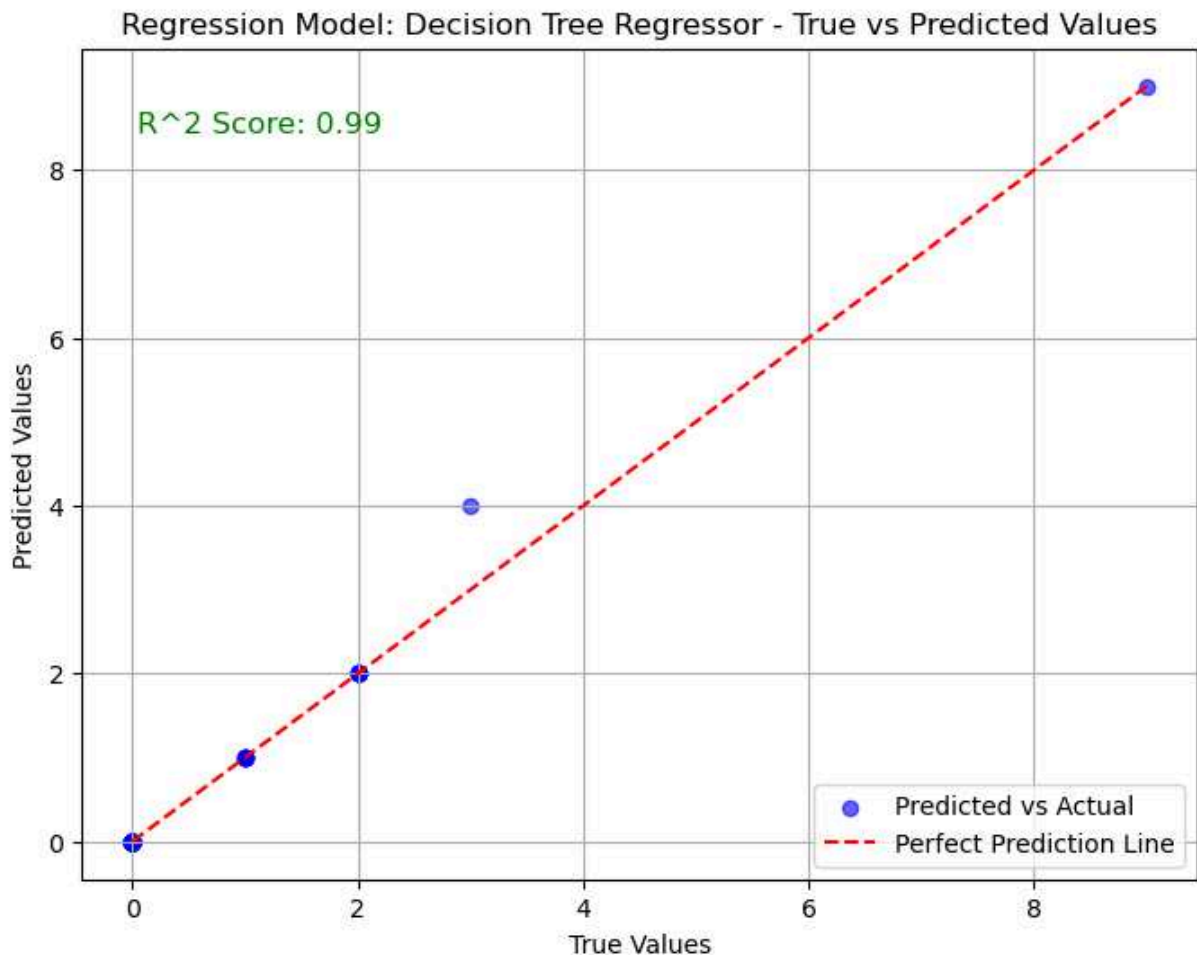
# 2 تدريب النموذج
```

```
DT.fit(X_train, y_train)

# 3 التنبؤ على مجموعة الاختبار
y_pred = DT.predict(X_test)

# 4 تقييم النموذج
evaluation_results = evaluate_model('Decision Tree Regressor', y_test, y_pred)

# 5 طباعة النتائج بشكل منسق
for key, value in evaluation_results.items():
    if isinstance(value, float):
        print(f"{key}: {value:.4f}")
    else:
        print(f"{key}: \n{value}")
```



Mean Absolute Error (MAE): 0.0095
Mean Squared Error (MSE): 0.0095
Root Mean Squared Error (RMSE): 0.0976
R^2 Score: 0.9912

```
In [81]: print ('KNeighborsRegressor')
```

KNeighborsRegressor

```
In [84]: from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsRegressor
```

```

# 1 ملء القيم المفقودة بالمتوسط
imputer = SimpleImputer(strategy='mean')
X_train_imputed = imputer.fit_transform(X_train)
X_test_imputed = imputer.transform(X_test)

# 2 مقياس البيانات (Scaling)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train_imputed)
X_test_scaled = scaler.transform(X_test_imputed)

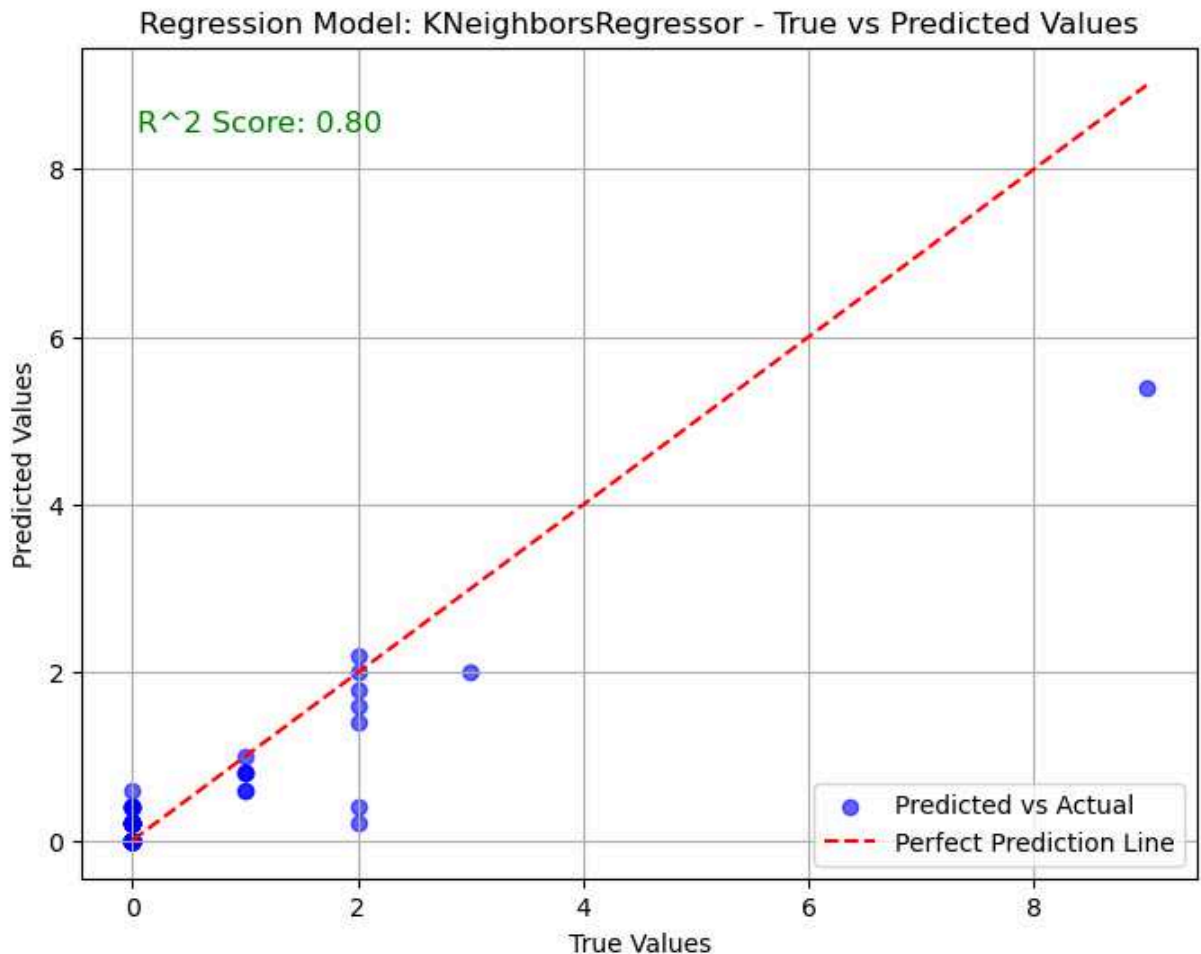
# 3 تدريب KNN Regressor
knn = KNeighborsRegressor()
knn.fit(X_train_scaled, y_train)

# 4 التنبؤ
y_pred = knn.predict(X_test_scaled)

# 5 تقييم النموذج
evaluation_results = evaluate_model('KNeighborsRegressor', y_test, y_pred)

# 6 طباعة النتائج
for key, value in evaluation_results.items():
    if isinstance(value, float):
        print(f"{key}: {value:.4f}")
    else:
        print(f"{key}: \n{value}")

```



Mean Absolute Error (MAE): 0.1619
Mean Squared Error (MSE): 0.2168
Root Mean Squared Error (RMSE): 0.4656
R^2 Score: 0.8008

```
In [86]: import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.multioutput import MultiOutputRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import numpy as np
import matplotlib.pyplot as plt
df=pd.read_csv("tested.csv")
df.isnull().sum()
```

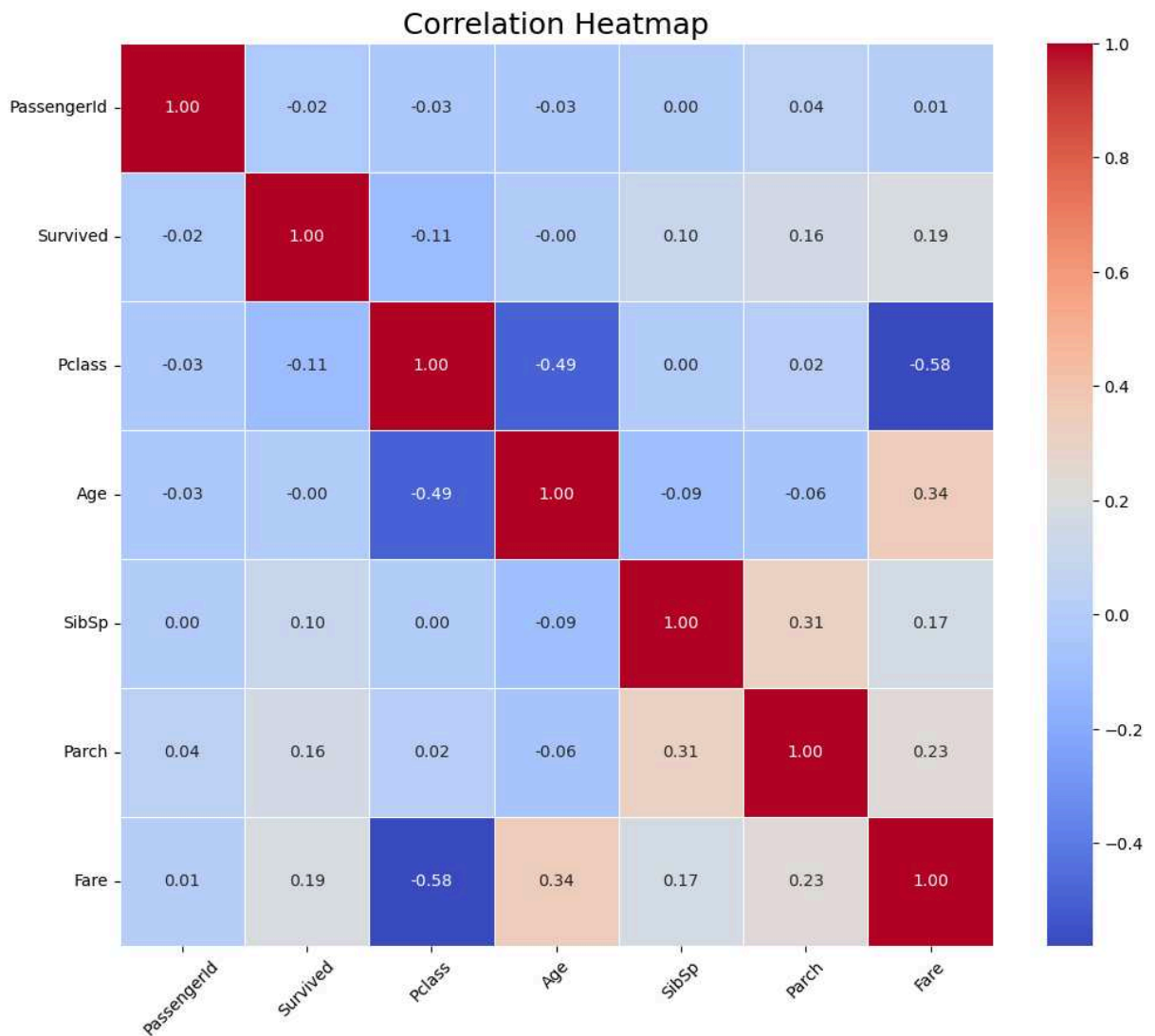
```
Out[86]: PassengerId      0
Survived      0
Pclass      0
Name      0
Sex      0
Age      86
SibSp      0
Parch      0
Ticket      0
Fare      1
Cabin      327
Embarked      0
dtype: int64
```

```
In [91]: import seaborn as sns
import matplotlib.pyplot as plt

# اختيار الأعمدة الرقمية فقط
numeric_df = df.select_dtypes(include=['int64', 'float64'])

# حساب مصفوفة الارتباط
corr = numeric_df.corr().round(2)

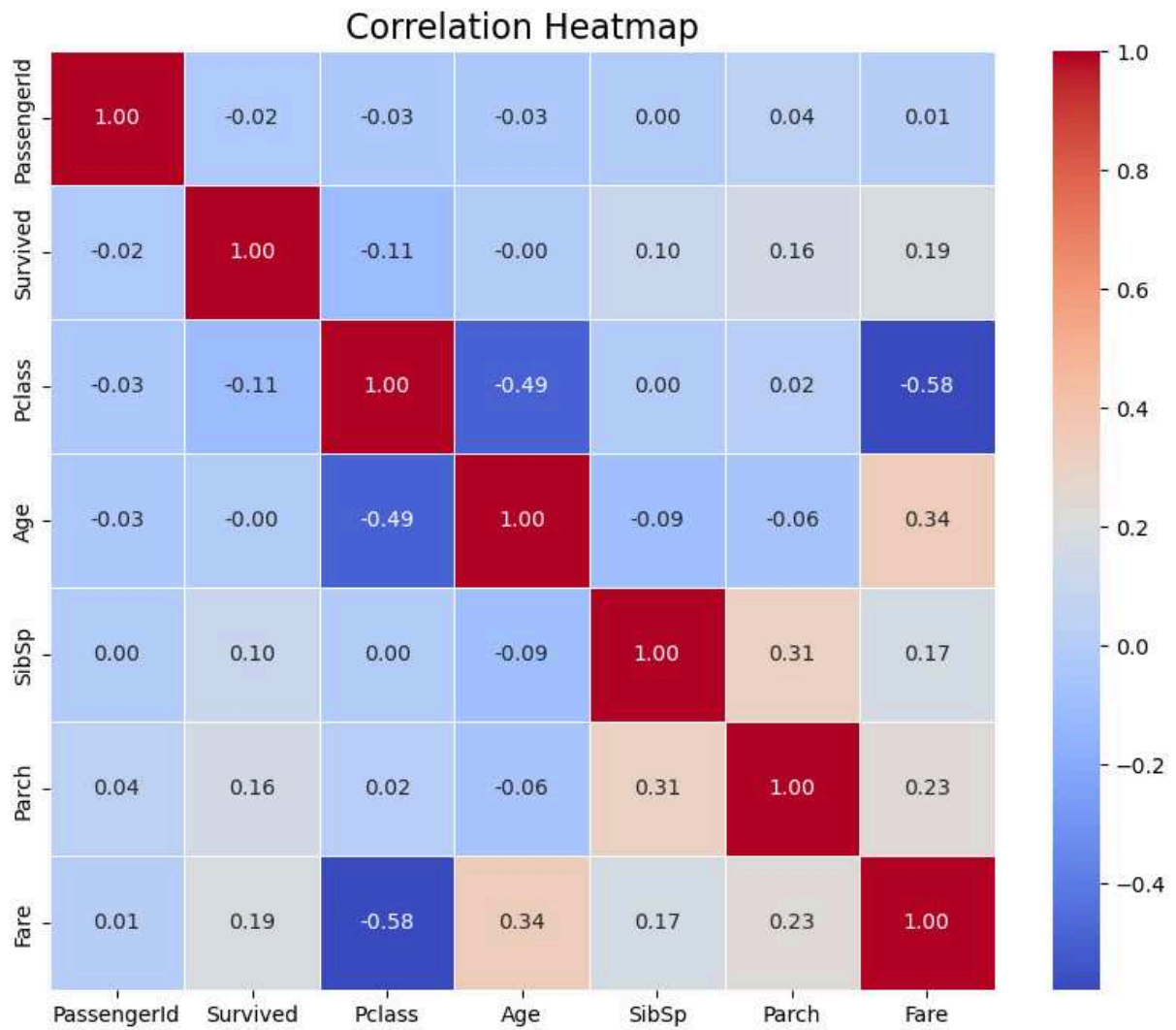
# رسم خريطة الحرارة
plt.figure(figsize=(12, 10))
sns.heatmap(corr, cmap='coolwarm', annot=True, fmt=".2f", linewidths=0.5)
plt.title('Correlation Heatmap', fontsize=18)
plt.xticks(rotation=45)
plt.yticks(rotation=0)
plt.show()
```



```
In [93]: numeric_df = df.select_dtypes(include=['int64', 'float64'])
corr = numeric_df.corr().round(2)

import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 8))
sns.heatmap(corr, cmap='coolwarm', annot=True, fmt=".2f", linewidths=0.5)
plt.title('Correlation Heatmap', fontsize=16)
plt.show()
```



```
In [97]: print ('لمعرفة عدد الصفوف في الدانا ست')
          len(df)
```

لمعرفة عدد الصفوف في الدانا ست

```
Out[97]: 418
```

```
In [99]: df.isnull().sum()
```

```
Out[99]: PassengerId    0
          Survived      0
          Pclass       0
          Name         0
          Sex          0
          Age         86
          SibSp        0
          Parch        0
          Ticket       0
          Fare         1
          Cabin      327
          Embarked     0
          dtype: int64
```

```
In [101... X = df.drop(['PassengerId', 'Sex', 'Age', 'Pclass'], axis=1).values
y = df['Embarked'].values
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

```
In [103... from imblearn.over_sampling import RandomOverSampler
ros = RandomOverSampler()
```

```
In [104... X_resampled, y_resampled = ros.fit_resample(X_train, y_train)
print("Before:")
pd.Series(y_train).value_counts()
```

Before:

```
Out[104... S    208
C     86
Q     40
Name: count, dtype: int64
```

```
In [105... print("After:")
pd.Series(y_resampled).value_counts()
```

After:

```
Out[105... S    208
Q    208
C    208
Name: count, dtype: int64
```

```
In [106... print ('أخذ عينات عشوائية ناقصة')
```

أخذ عينات عشوائية ناقصة

```
In [107... from imblearn.under_sampling import RandomUnderSampler
rus = RandomUnderSampler()
X_rus, y_rus = rus.fit_resample(X_train, y_train)
print("After Undersampling:", pd.Series(y_rus).value_counts())
```

After Undersampling: C 40
Q 40
S 40
Name: count, dtype: int64

```
In [108... print ('SMOTE + Tomek Links (Hybrid)')
```

SMOTE + Tomek Links (Hybrid)

```
In [112... print ('feature selection methods')
```

feature selection methods

```
In [115... import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
```

```

from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.ensemble import RandomForestClassifier
import matplotlib.pyplot as plt

```

```

In [116... df = pd.read_csv("tested.csv")
df.head()

```

Out[116...

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
0	892	0	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292
1	893	1	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000
2	894	0	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875
3	895	0	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625
4	896	1	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875

```

In [117... df.isnull().sum()

```

Out[117...

```

PassengerId    0
Survived        0
Pclass         0
Name           0
Sex            0
Age           86
SibSp          0
Parch          0
Ticket         0
Fare           1
Cabin        327
Embarked       0
dtype: int64

```

```

In [118... df.dtypes

```

```
Out[118... PassengerId      int64
Survived         int64
Pclass           int64
Name             object
Sex              object
Age              float64
SibSp            int64
Parch            int64
Ticket           object
Fare             float64
Cabin            object
Embarked         object
dtype: object
```

```
In [119... df.describe()
```

```
Out[119...      PassengerId  Survived  Pclass     Age  SibSp  Parch    Fare
count  418.000000  418.000000  418.000000  332.000000  418.000000  418.000000  417.000000
mean    1100.500000    0.363636    2.265550   30.272590    0.447368    0.392344   35.627188
std     120.810458    0.481622    0.841838   14.181209    0.896760    0.981429   55.907576
min      892.000000    0.000000    1.000000    0.170000    0.000000    0.000000    0.000000
25%     996.250000    0.000000    1.000000   21.000000    0.000000    0.000000    7.895800
50%    1100.500000    0.000000    3.000000   27.000000    0.000000    0.000000   14.454200
75%    1204.750000    1.000000    3.000000   39.000000    1.000000    0.000000   31.500000
max    1309.000000    1.000000    3.000000   76.000000    8.000000    9.000000  512.329200
```



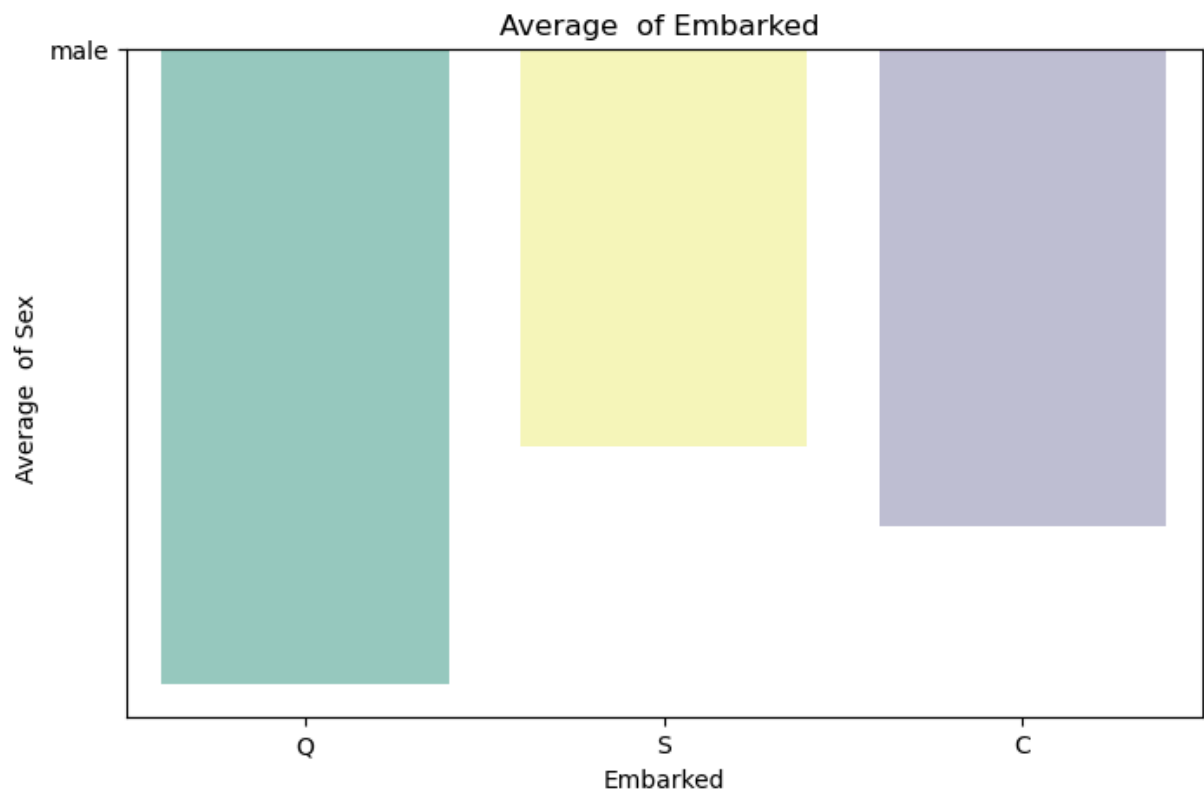
```
In [121... from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['Embarked'] = le.fit_transform(df['Embarked'])
X = df.drop(['Embarked', 'PassengerId'], axis=1)
y = df['Embarked']
```

```
In [39]: plt.figure(figsize=(8,5))
sns.barplot(
    data=df,
    x='Embarked',
    y='Sex',
    estimator='mean',
    palette='Set3',
    errorbar=None
)
plt.title("Average of Embarked ")
plt.ylabel("Average of Sex ")
plt.show()
```

```
C:\Users\user\AppData\Local\Temp\ipykernel_26980\3557372214.py:2: FutureWarning:
```

```
Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.
```

```
sns.barplot(
```



```
In [ ]:
```