📊 ML Classification Project: PHDs Produced by Pakistani Universities (2010–2014)

1st step: Loading important .py libraries & Loading Dataset

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix, ConfusionMatrixDisplay
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from xgboost import XGBClassifier

# 🔷 Load Dataset
df = pd.read_csv("/content/PHDs Produced by Pakistani Universities (2010-2014).csv")
df.dropna(inplace=True)
df.head(3)
```

	S#	Institute	2010	2011	2012	2013	2014	Sector
0	1	Abdul Wali Khan University, Mardan	0	0	0	1	1	Public
1	2	Allama Iqbal Open University, Islamabad	12	13	4	4	12	Public
2	3	Air University	0	0	0	1	0	Public

Next steps: Generate code with df | View recommended plots | New interactive sheet

2nd Step: Preprocessing (Dataset)

```
# 📦 Step 1: Import Required Libraries
import pandas as pd
from sklearn.preprocessing import LabelEncoder, StandardScaler

# 📥 Step 2: Load Dataset
df = pd.read_csv("PHDs Produced by Pakistani Universities (2010-2014).csv")
print("✅ Loaded Dataset (First 5 Rows):\n", df.head())

# 🧹 Step 3: Drop Unnecessary Columns
df.drop(columns=["S#", "Institute"], inplace=True)
print("\n✅ After Dropping Columns ['S#', 'Institute']:\n", df.head())

# 🕵 Step 4: Handle Missing Values
missing = df.isnull().sum()
print("\n🔍 Missing Values:\n", missing)

# 🧠 Step 5: Label Encoding for Target Column
label_encoder = LabelEncoder()
df['Sector'] = label_encoder.fit_transform(df['Sector'])  # Public = 1, Private = 0
print("\n✅ Encoded 'Sector' Column:\n", df['Sector'].head())

# 📊 Step 6: Feature Scaling
scaler = StandardScaler()
X = df.drop(columns=['Sector'])        # Independent variables
X_scaled = scaler.fit_transform(X)
X_scaled = pd.DataFrame(X_scaled, columns=X.columns)

y = df['Sector']                       # Target variable

# 📈 Output Preview
print("\n✅ Scaled Features (First 5 Rows):\n", X_scaled.head())
print("\n✅ Encoded Target (First 5 Rows):\n", y.head())
print("\n✅ Missing Values After Preprocessing:\n", df.isnull().sum())
```

Sector

```
  Sector
0  Public
1  Public
2  Public
3  Public
4  Public
```

✅ After Dropping Columns ['S#', 'Institute']:

```
   2010  2011  2012  2013  2014  Sector
0     0     0     0     1     1  Public
1    12    13     4     4    12  Public
2     0     0     0     1     0  Public
3    16    21    27    35    33  Public
4     0     0     1     3     0  Public
```

🔍 Missing Values:

```
 2010      0
2011      0
2012      0
2013      0
2014      0
Sector    0
dtype: int64
```

✅ Encoded 'Sector' Column:

```
 0    1
1    1
2    1
3    1
4    1
Name: Sector, dtype: int64
```

✅ Scaled Features (First 5 Rows):

```
       2010      2011      2012      2013      2014
0 -0.419636 -0.438951 -0.467637 -0.425359 -0.456756
1  0.052454 -0.004287 -0.341859 -0.337741 -0.137622
2 -0.419636 -0.438951 -0.467637 -0.425359 -0.485768
3  0.209818  0.263199  0.381366  0.567644  0.471634
4 -0.419636 -0.438951 -0.436192 -0.366947 -0.485768
```

✅ Encoded Target (First 5 Rows):

```
✅ Missing Values After Preprocessing:
 2010      0
2011      0
2012      0
2013      0
2014      0
Sector    0
dtype: int64
```

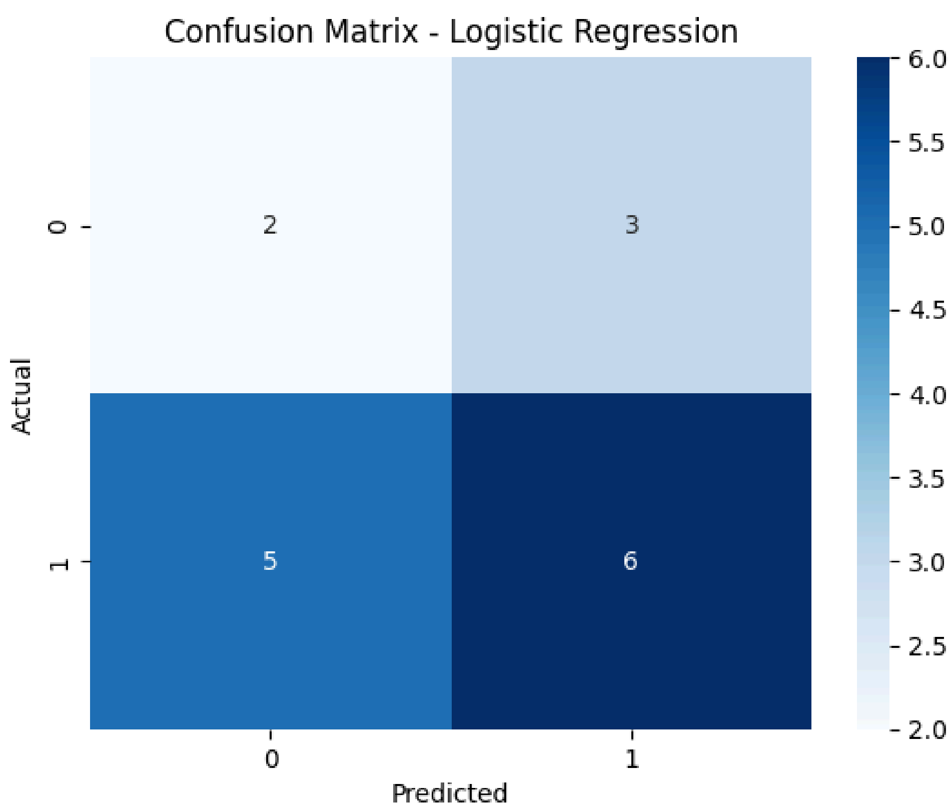
3rd Step: Apply ML Models Separately with there accuracy_score, classification_report, confusion_matrix.

(1) Logistic Regression

```
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
print(classification_report(y_test, y_pred_lr))


#(1) Logistic Regression
lr_model = LogisticRegression()
lr_model.fit(X_train, y_train)
y_pred_lr = lr_model.predict(X_test)
print("\n🔍 Logistic Regression Results")
print("Accuracy:", accuracy_score(y_test, y_pred_lr))
print(classification_report(y_test, y_pred_lr))
sns.heatmap(confusion_matrix(y_test, y_pred_lr), annot=True, fmt='d', cmap='Blues')
plt.title("Confusion Matrix - Logistic Regression")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```

```
              precision    recall  f1-score   support

           0       0.29      0.40      0.33         5
           1       0.67      0.55      0.60        11

    accuracy                           0.50        16
   macro avg       0.48      0.47      0.47        16
weighted avg       0.55      0.50      0.52        16
```

```
🔍 Logistic Regression Results
Accuracy: 0.5
              precision    recall  f1-score   support

           0       0.29      0.40      0.33         5
           1       0.67      0.55      0.60        11

    accuracy                           0.50        16
   macro avg       0.48      0.47      0.47        16
weighted avg       0.55      0.50      0.52        16
```

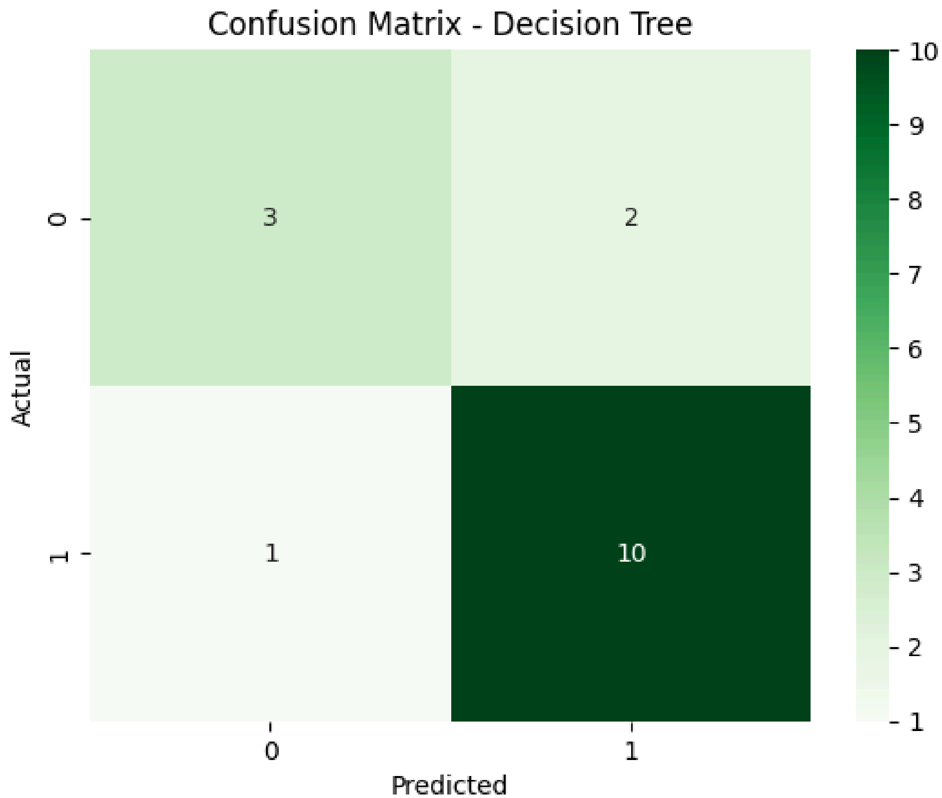


(2) Decision Tree

```
dt_model = DecisionTreeClassifier()
dt_model.fit(X_train, y_train)
y_pred_dt = dt_model.predict(X_test)
print("\n🔍 Decision Tree Results")
print("Accuracy:", accuracy_score(y_test, y_pred_dt))
print(classification_report(y_test, y_pred_dt))
sns.heatmap(confusion_matrix(y_test, y_pred_dt), annot=True, fmt='d', cmap='Greens')
plt.title("Confusion Matrix - Decision Tree")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```

🔍 Decision Tree Results
Accuracy: 0.8125

	precision	recall	f1-score	support
0	0.75	0.60	0.67	5
1	0.83	0.91	0.87	11
accuracy			0.81	16
macro avg	0.79	0.75	0.77	16
weighted avg	0.81	0.81	0.81	16

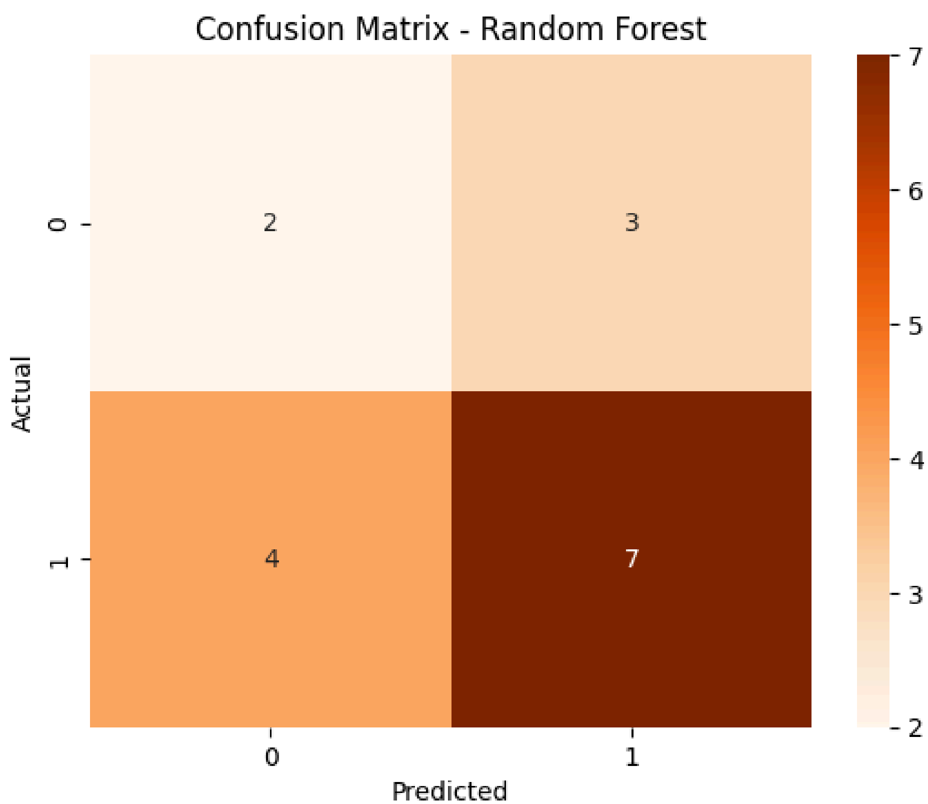


(3) Random Forest

```
rf_model = RandomForestClassifier()
rf_model.fit(X_train, y_train)
y_pred_rf = rf_model.predict(X_test)
print("\n🔍 Random Forest Results")
print("Accuracy:", accuracy_score(y_test, y_pred_rf))
print(classification_report(y_test, y_pred_rf))
sns.heatmap(confusion_matrix(y_test, y_pred_rf), annot=True, fmt='d', cmap='Oranges')
plt.title("Confusion Matrix - Random Forest")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```

🔍 Random Forest Results
Accuracy: 0.5625

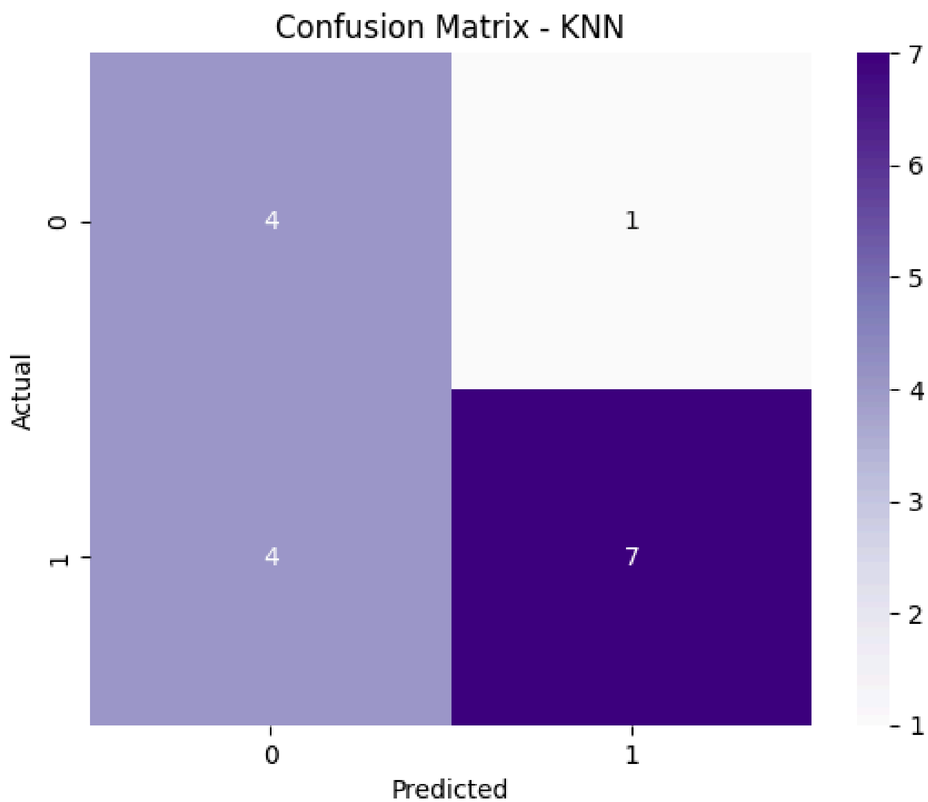
	precision	recall	f1-score	support
0	0.33	0.40	0.36	5
1	0.70	0.64	0.67	11
accuracy			0.56	16
macro avg	0.52	0.52	0.52	16
weighted avg	0.59	0.56	0.57	16



(4) K-Nearest Neighbors

```
knn_model = KNeighborsClassifier()
knn_model.fit(X_train, y_train)
y_pred_knn = knn_model.predict(X_test)
print("\n🔍 KNN Results")
print("Accuracy:", accuracy_score(y_test, y_pred_knn))
print(classification_report(y_test, y_pred_knn))
sns.heatmap(confusion_matrix(y_test, y_pred_knn), annot=True, fmt='d', cmap='Purples')
plt.title("Confusion Matrix - KNN")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```

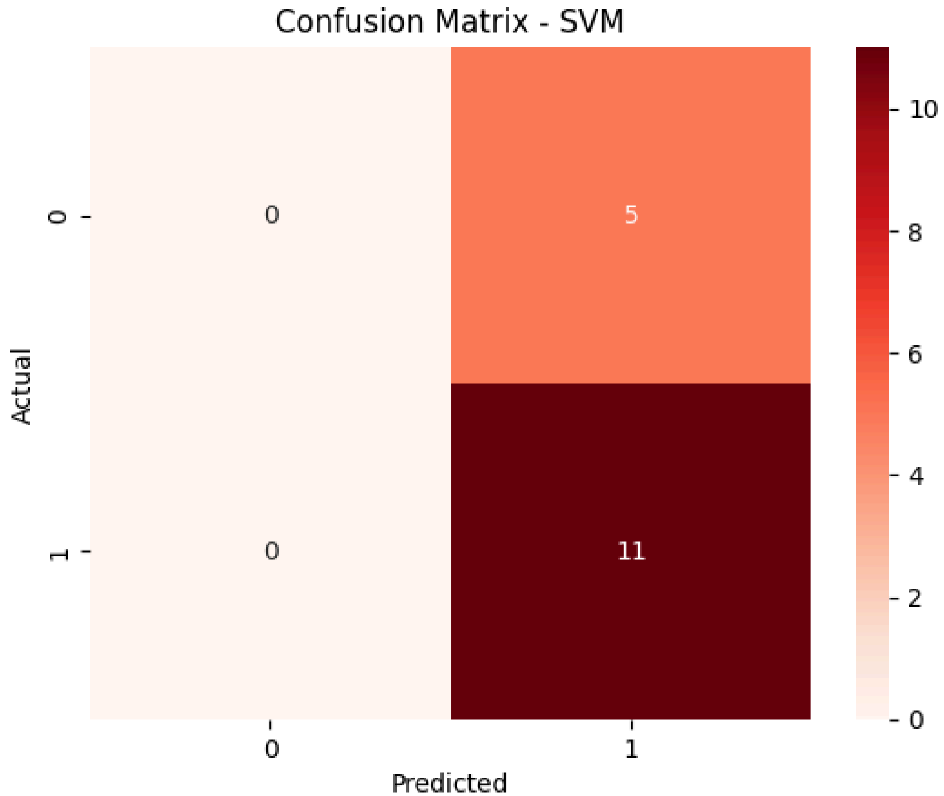
```
🔍 KNN Results
Accuracy: 0.6875
              precision    recall  f1-score   support

           0       0.50      0.80      0.62         5
           1       0.88      0.64      0.74        11

    accuracy                           0.69        16
   macro avg       0.69      0.72      0.68        16
weighted avg       0.76      0.69      0.70        16
```



(5) Support Vector Machine

```
svm_model = SVC(probability=True)
svm_model.fit(X_train, y_train)
y_pred_svm = svm_model.predict(X_test)
print("\n🔍 SVM Results")
print("Accuracy:", accuracy_score(y_test, y_pred_svm))
print(classification_report(y_test, y_pred_svm))
sns.heatmap(confusion_matrix(y_test, y_pred_svm), annot=True, fmt='d', cmap='Reds')
plt.title("Confusion Matrix - SVM")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```

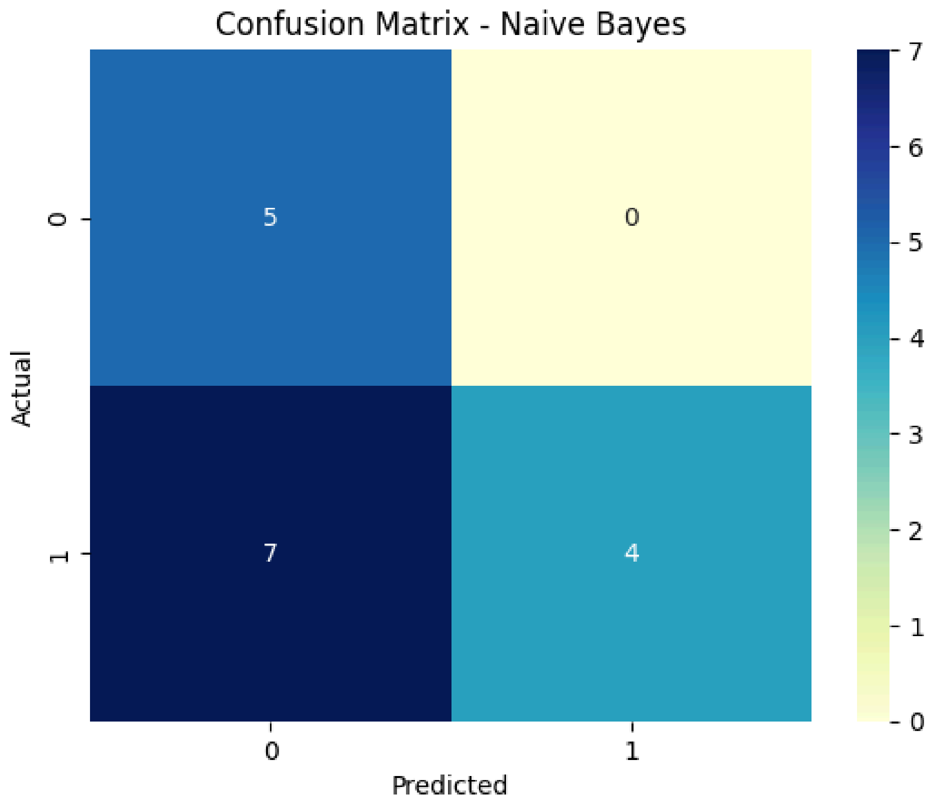
```
🔍 SVM Results
Accuracy: 0.6875
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         5
           1       0.69      1.00      0.81        11

    accuracy                           0.69        16
   macro avg       0.34      0.50      0.41        16
weighted avg       0.47      0.69      0.56        16

/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined ar
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined ar
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined ar
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```



(6) Naive Bayes

```
nb_model = GaussianNB()
nb_model.fit(X_train, y_train)
y_pred_nb = nb_model.predict(X_test)
print("\n🔍 Naive Bayes Results")
print("Accuracy:", accuracy_score(y_test, y_pred_nb))
print(classification_report(y_test, y_pred_nb))
sns.heatmap(confusion_matrix(y_test, y_pred_nb), annot=True, fmt='d', cmap='YlGnBu')
plt.title("Confusion Matrix - Naive Bayes")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```

```
🔍 Naive Bayes Results
Accuracy: 0.5625
              precision    recall  f1-score   support

           0       0.42      1.00      0.59         5
           1       1.00      0.36      0.53        11

    accuracy                           0.56        16
   macro avg       0.71      0.68      0.56        16
weighted avg       0.82      0.56      0.55        16
```



7th Step: installing xgboost pkg and importing requred libraries then performing and Evaluating Xgboost..

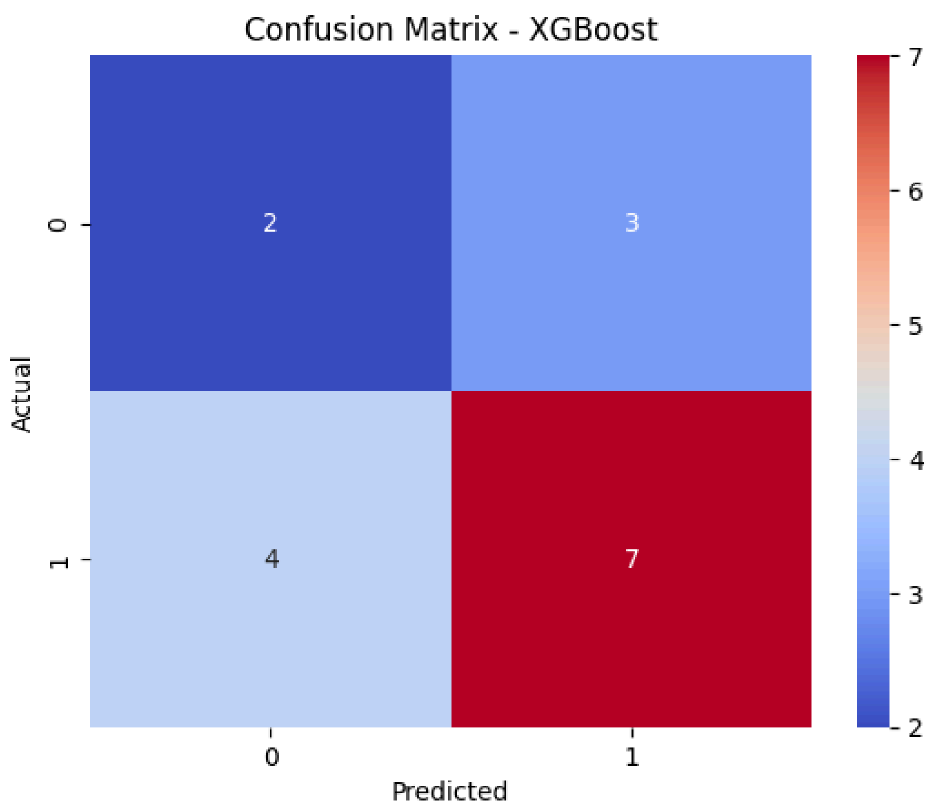
```
pip install xgboost
```

```
Requirement already satisfied: xgboost in /usr/local/lib/python3.11/dist-packages (3.0.2)
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages (from xgboost) (2.0.2)
Requirement already satisfied: nvidia-nccl-cu12 in /usr/local/lib/python3.11/dist-packages (from xgboost) (2.21.5)
Requirement already satisfied: scipy in /usr/local/lib/python3.11/dist-packages (from xgboost) (1.16.0)
```

```
# 📦 Required Libraries
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from xgboost import XGBClassifier

# 🟡 Example Data (replace with your own dataset)
# df = pd.read_csv("your_dataset.csv")
# X = df.drop("target", axis=1)
# y = df["target"]

# ✅ Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# (7) ✅ XGBoost Classifier
xgb_model = XGBClassifier(use_label_encoder=False, eval_metric='mlogloss')
xgb_model.fit(X_train, y_train)
y_pred_xgb = xgb_model.predict(X_test)

# 📊 Evaluation
print("\n🔍 XGBoost Results")
print("Accuracy:", accuracy_score(y_test, y_pred_xgb))
print(classification_report(y_test, y_pred_xgb))

# 📉 Confusion Matrix
sns.heatmap(confusion_matrix(y_test, y_pred_xgb), annot=True, fmt='d', cmap='coolwarm')
plt.title("Confusion Matrix - XGBoost")
plt.xlabel("Predicted")
plt.ylabel("Actual")
```

```
plt.show()
```

```
🔍 XGBoost Results
Accuracy: 0.5625
              precision    recall  f1-score   support

           0       0.33      0.40      0.36         5
           1       0.70      0.64      0.67        11

    accuracy                           0.56        16
   macro avg       0.52      0.52      0.52        16
weighted avg       0.59      0.56      0.57        16

/usr/local/lib/python3.11/dist-packages/xgboost/training.py:183: UserWarning: [08:14:39] WARNING: /workspace/src/learner.cc:738:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
```



```
#8th Step: Comparing all above applyed ML-Models Performance..
```

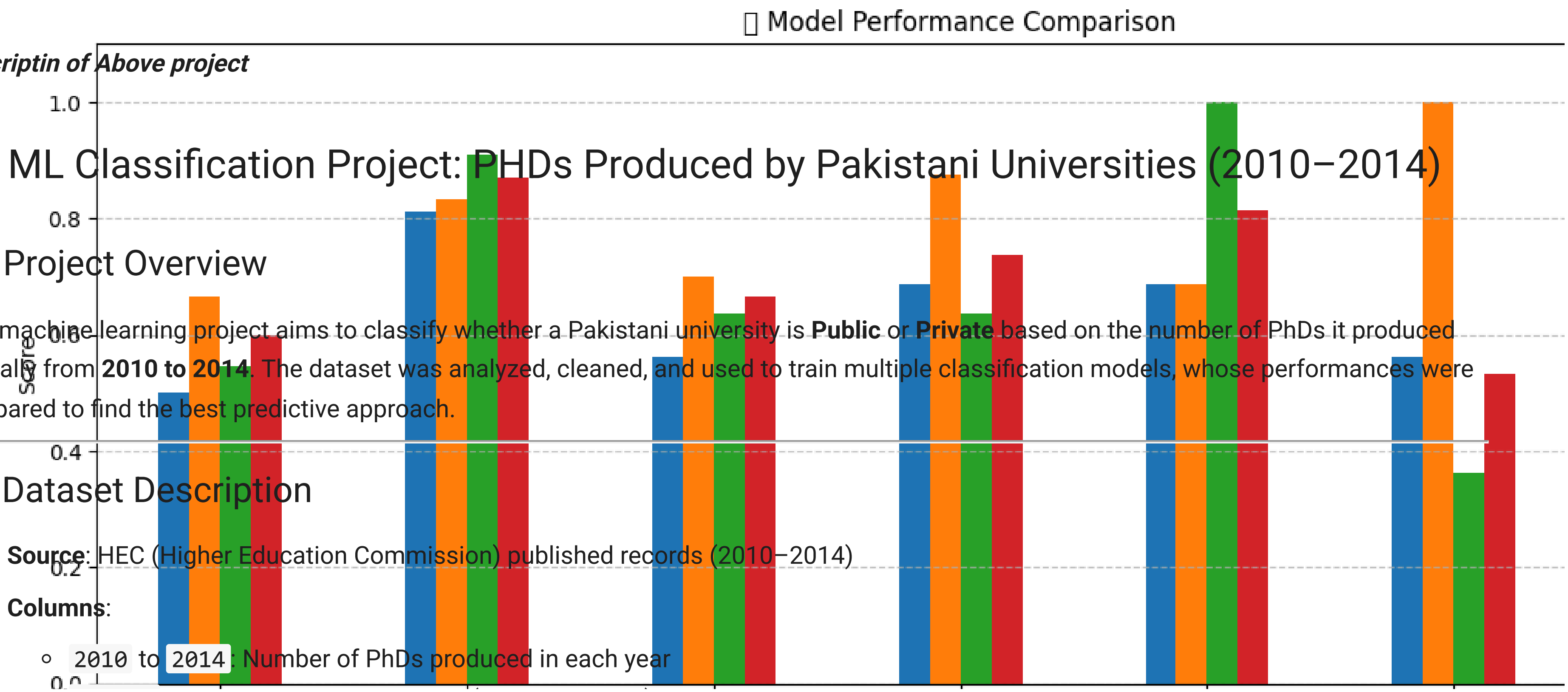
8th Step: Comparing all above applyed ML-Models Performance..

```
# 📊 Compare Models

from sklearn.metrics import precision_score, recall_score, f1_score

# 📝 Create a results list
results = []

# 🔁 Loop through all model predictions
models = {
    "Logistic Regression": y_pred_lr,
    "Decision Tree": y_pred_dt,
    "Random Forest": y_pred_rf,
    "KNN": y_pred_knn,
    "SVM": y_pred_svm,
    "Naive Bayes": y_pred_nb,
    "XGBoost": y_pred_xgb,
}

for name, preds in models.items():
    results.append({
        "Model": name,
        "Accuracy": accuracy_score(y_test, preds),
        "Precision": precision_score(y_test, preds),
        "Recall": recall_score(y_test, preds),
        "F1 Score": f1_score(y_test, preds),
    })

# 📑 Create DataFrame from results
results_df = pd.DataFrame(results)
print("✅ Model Performance Comparison:\n", results_df)

# 📊 Plot Bar Chart
plt.figure(figsize=(12, 6))
```

```
results_df.set_index("Model")[["Accuracy", "Precision", "Recall", "F1 Score"]].plot(kind="bar", figsize=(12, 6))
plt.title("📊 Model Performance Comparison")
plt.ylabel("Score")
plt.ylim(0, 1.1)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.xticks(rotation=45)
plt.legend(loc="lower right")
plt.tight_layout()
plt.show()
```

✅ Model Performance Comparison:

```
                 Model  Accuracy  Precision    Recall  F1 Score
0  Logistic Regression    0.5000   0.666667  0.545455  0.600000
1        Decision Tree    0.8125   0.833333  0.909091  0.869565
2        Random Forest    0.5625   0.700000  0.636364  0.666667
3                  KNN    0.6875   0.875000  0.636364  0.736842
4                  SVM    0.6875   0.687500  1.000000  0.814815
5          Naive Bayes    0.5625   1.000000  0.363636  0.533333
6              XGBoost    0.5625   0.700000  0.636364  0.666667
/tmp/ipython-input-1134567460.py:41: UserWarning: Glyph 128202 (\N{BAR CHART}) missing from font(s) DejaVu Sans.
  plt.tight_layout()
<Figure size 1200x600 with 0 Axes>
/usr/local/lib/python3.11/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 128202 (\N{BAR CHART}) missing from font(
  fig.canvas.print_figure(bytes_io, **kw)
```



Descriptin of Above project

📊 ML Classification Project: PHDs Produced by Pakistani Universities (2010–2014)

🧠 Project Overview

This machine learning project aims to classify whether a Pakistani university is **Public** or **Private** based on the number of PhDs it produced annually from **2010 to 2014**. The dataset was analyzed, cleaned, and used to train multiple classification models, whose performances were compared to find the best predictive approach.

📁 Dataset Description

- **Source**: HEC (Higher Education Commission) published records (2010–2014)
- **Columns**:
  - `2010` to `2014`: Number of PhDs produced in each year