Tree Models Analysis Report

Dataset

We worked with the Recruitment Dataset, which contains candidate records for a company and whether they were hired or not. The dataset includes attributes such as experience, education, recruitment strategy, personality, skills, interview scores, and distance from the company.

Target Variable: HiringDecision → whether a candidate is hired (1) or not (0).

Features used:
Categorical: EducationLevel, RecruitmentStrategy
Numerical: ExperienceYears, DistanceFromCompany, InterviewScore, SkillScore, PersonalityScore, PreviousCompanies

Preprocessing

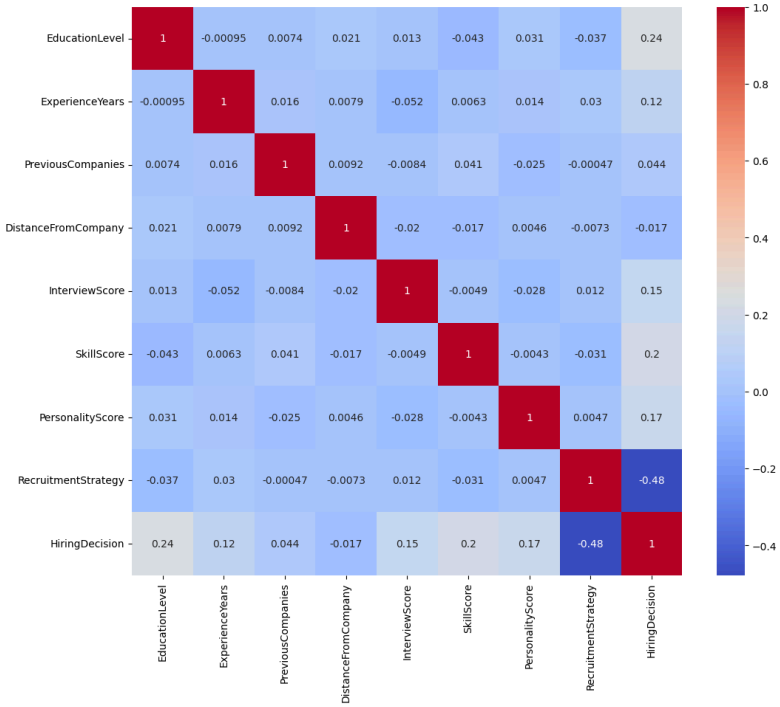
Before training models, the dataset was preprocessed to ensure consistency and suitability for tree-based classification methods.

- Data Cleaning
    - Dropped irrelevant features Gender and Age since they showed minimal correlation with the target.
    - Checked for null values → no missing data detected.
- Categorical Variables
    - Converted categorical variables to numeric using one-hot encoding:
        - EducationLevel
        - RecruitmentStrategy
- Feature Scaling
    - Tree-based models like Decision Trees, Random Forest, AdaBoost, and XGBoost are scale-invariant, so no scaling was required.
- Train-Test Split
    - Dataset split into 80/20 for training and testing.
- Correlation Analysis
    - We generated a correlation heatmap to analyze the relationship between the features and the target variable HiringDecision.

    - The correlation matrix is shown below

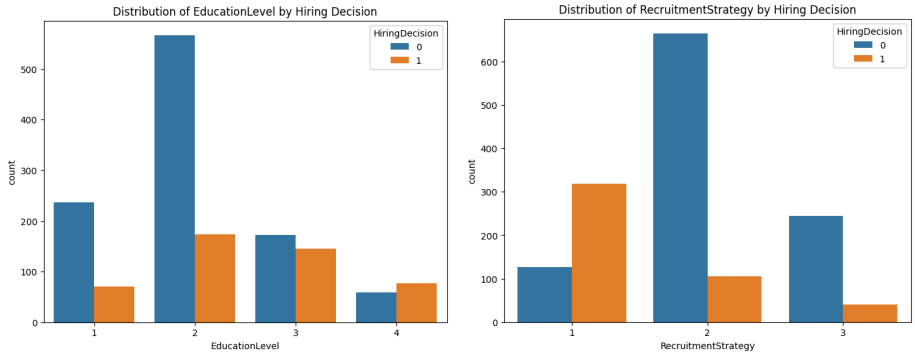
	HiringDecision
HiringDecision	1.00
EducationLevel	0.24
SkillScore	0.20
PersonalityScore	0.17
InterviewScore	0.15
ExperienceYears	0.12
PreviousCompanies	0.04
Gender	-0.00
Age	0.00
DistanceFromCompany	-0.02
RecruitmentStrategy	-0.48

- Features such as EducationLevel, SkillScore, PersonalityScore, InterviewScore, and ExperienceYears are positively correlated with being hired. This indicates that candidates with higher scores and education levels are more likely to be selected.
- RecruitmentStrategy is negatively correlated (-0.48), suggesting certain strategies are associated with lower hiring in this dataset.
- Features like DistanceFromCompany, Gender, and Age show barely any or no correlation and are not useful for us
- Correlation Heatmap:

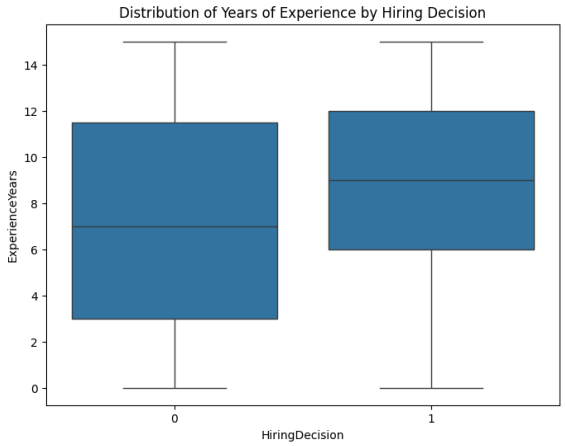


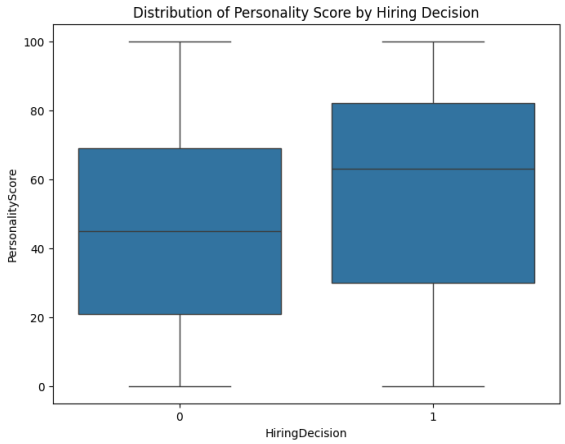
- Distribution of features

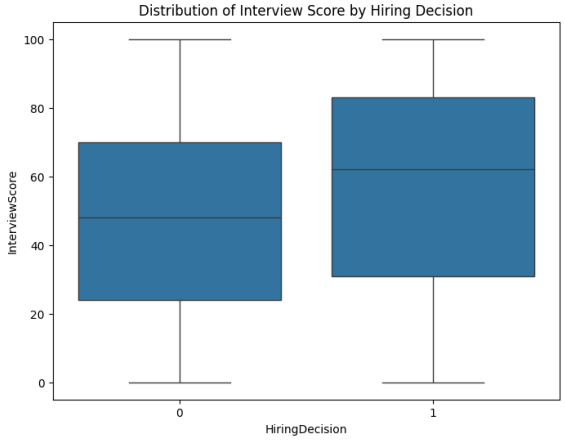
- Distribution of Categorical Features based on plots below
  - Certain education levels led to higher hiring rates.
  - Recruitment strategies showed variance in success, aligning with the negative correlation observed.

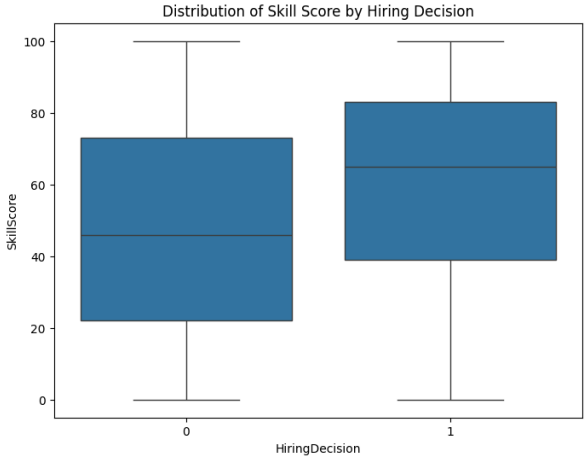


- Numerical Features based on box plots below
  - Hired candidates had higher PersonalityScore, ExperienceYears, and InterviewScore on average.
  - For skill score, most hired candidates scored higher, and their scores are more concentrated in the upper range but both groups have candidates from 0 → 100, so SkillScore alone doesn't guarantee hiring.









Decision Tree Classifier

Model Construction:

We tuned a wide range of hyperparameters using GridSearchCV. The following image shows the various combinations of hyperparameters used in the current model.

```
param_grid = {
    'criterion': ['gini', 'entropy', 'log_loss'],
    'max_depth': [None, 3, 5, 7, 9, 12],
    'min_samples_split': [2, 5, 10, 20],
    'min_samples_leaf': [1, 2, 4, 8],
    'max_features': [None, 'sqrt', 'log2'],
    'class_weight': [None, 'balanced']  #for class imbalance - which we have because more than twice as much were not hired
}

dt = DecisionTreeClassifier()

gs1 = GridSearchCV(
    estimator=dt,
    param_grid=param_grid,
    scoring='f1',
    cv=5,
    n_jobs=-1, #use all processors
    verbose=0
)
```

The best parameters had a f1 score of **0.8546159586846749** and came out to be:

```
{'class_weight': 'balanced',
 'criterion': 'log_loss',
 'max_depth': 7,
 'max_features': None,
 'min_samples_leaf': 4,
 'min_samples_split': 2}
```

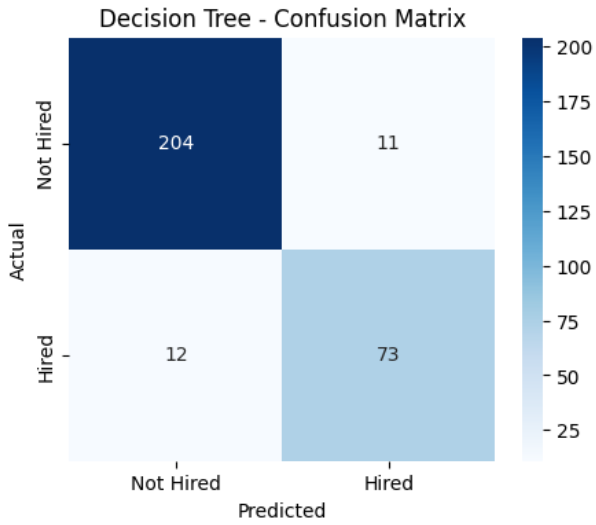
Model results:

Accuracy & Classification report:

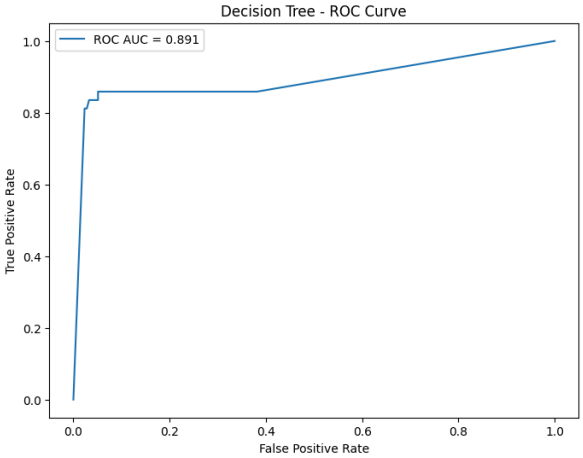
Test Accuracy : 0.9233333333333333

```
Classification report:
              precision    recall  f1-score   support

           0      0.944     0.949     0.947       215
           1      0.869     0.859     0.864        85

    accuracy                          0.923       300
   macro avg      0.907     0.904     0.905       300
weighted avg      0.923     0.923     0.923       300
```

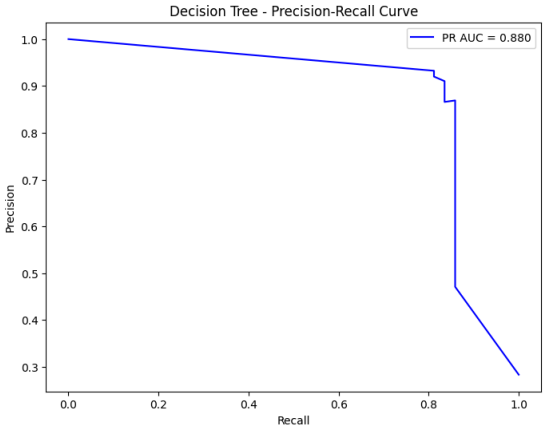
Confusion matrix:



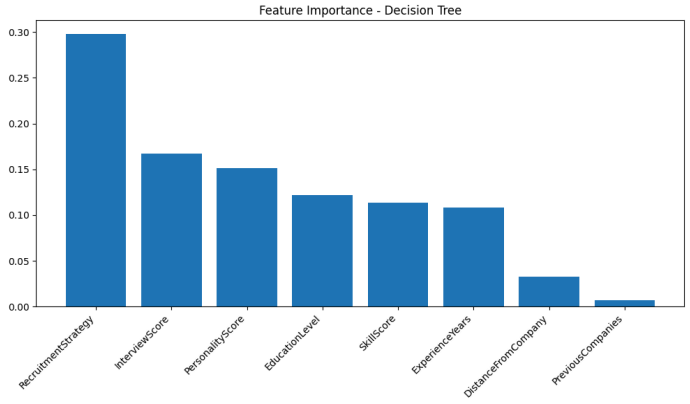
ROC curve:



Precision recall curve:



Feature importance:



Random Forest Classifier

Model Construction:

We tuned a wide range of hyperparameters using GridSearchCV. The following image shows the various combinations of hyperparameters used in the current model.

```
param_grid = {
    'n_estimators': [100, 300, 600], #number of trees
    'max_depth': [None, 5, 8, 12], #to control overfitting
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'max_features': ['sqrt', 'log2', None],
    'bootstrap': [True],
    'class_weight': ['balanced']
    }
```

```
gs_rf = GridSearchCV(
    rf,
    param_grid=param_grid,
    scoring='f1',
    cv=5,
    n_jobs=-1,
    refit=True,
    verbose=0
)
```

The best parameters had a f1 score of **0.8854** and came out to be:

```
gs_rf.best_params_
```

```
{'bootstrap': True,
 'class_weight': 'balanced',
 'max_depth': None,
 'max_features': 'log2',
 'min_samples_leaf': 2,
 'min_samples_split': 5,
 'n_estimators': 300}
```

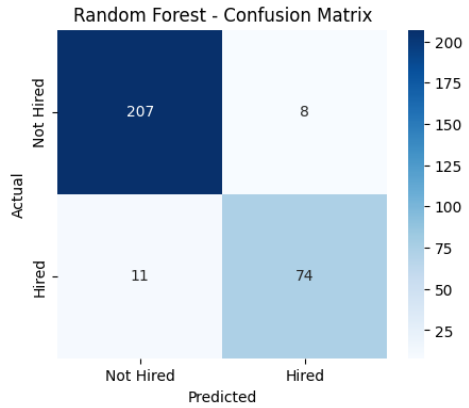
Model results:

Accuracy & Classification report:

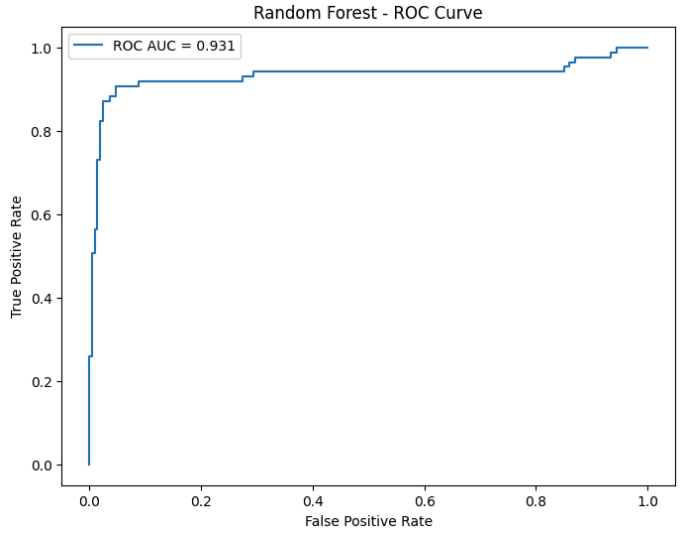
Test Accuracy: 0.9366666666666666

```
RF Classification report:
              precision    recall  f1-score   support

           0      0.950     0.963     0.956       215
           1      0.902     0.871     0.886        85

    accuracy                          0.937       300
   macro avg      0.926     0.917     0.921       300
weighted avg      0.936     0.937     0.936       300
```

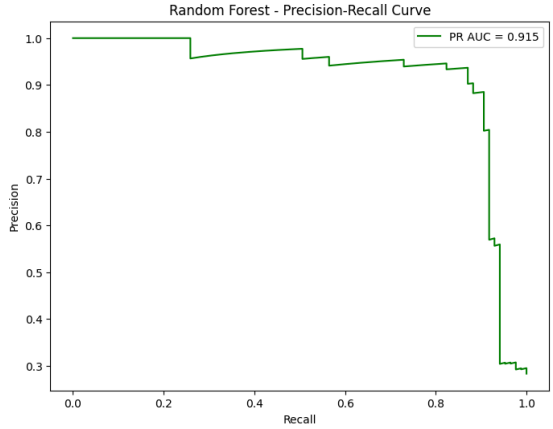
Confusion matrix:



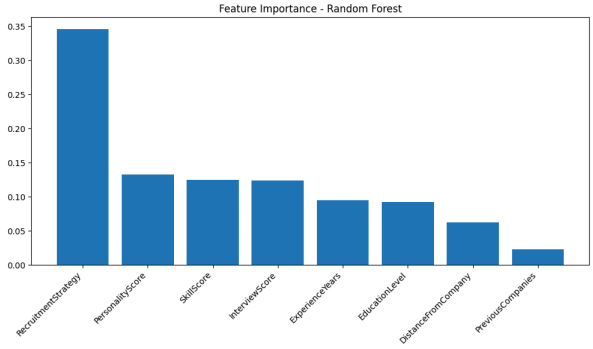
ROC curve:



Precision recall curve:



Feature importance:



Adaboot Classifier

Model Construction:

We tuned a wide range of hyperparameters using GridSearchCV. The following image shows the various combinations of hyperparameters used in the current model.

```
param_grid_ada = {
    'n_estimators': [50, 100, 200, 300],
    'learning_rate': [0.01, 0.1, 0.5, 1.0]
}
```

```
gs_ada = GridSearchCV(
    ada,
    param_grid=param_grid_ada,
    scoring='f1',
    cv=5,
    n_jobs=-1,
    refit=True,
    verbose=0
)
```

The best parameters had a f1 score of **0.8669** and came out to be:

```
AdaBoost best params: {'learning_rate': 1.0, 'n_estimators': 100}
AdaBoost best CV F1: 0.866918007197809
```

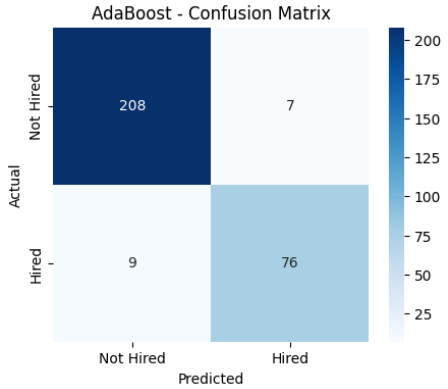
Model results:

Accuracy & Classification report:

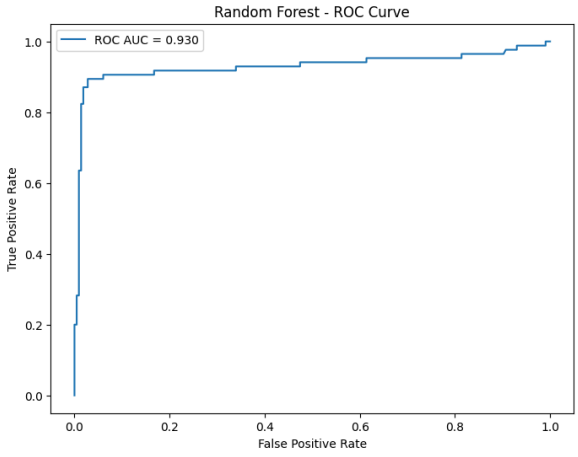
Test Accuracy: 0.947

```
AdaBoost Classification report:
              precision    recall  f1-score   support

           0      0.959     0.967     0.963       215
           1      0.916     0.894     0.905        85

    accuracy                          0.947       300
   macro avg      0.937     0.931     0.934       300
weighted avg      0.946     0.947     0.946       300
```

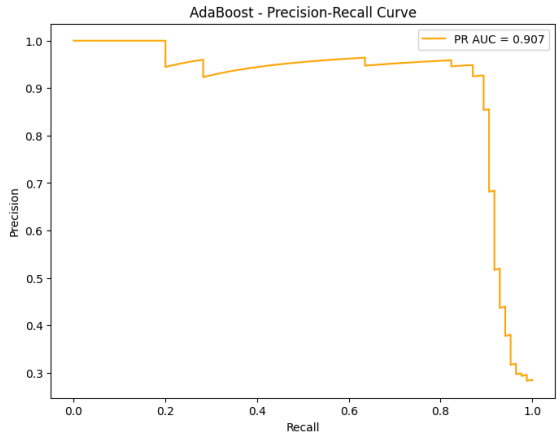
Confusion matrix:



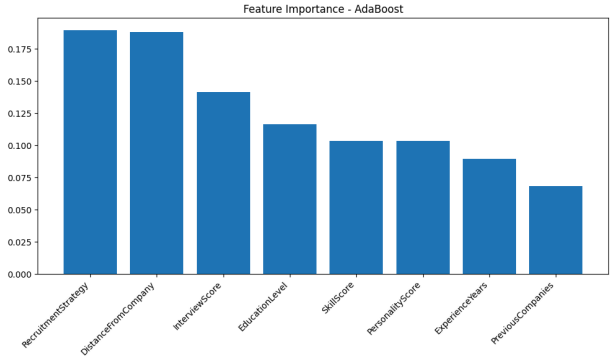
ROC curve:



Precision recall curve:



Feature importance:



XGBoost Classifier

Model Construction:

We tuned a wide range of hyperparameters using GridSearchCV. The following image shows the various combinations of hyperparameters used in the current model.

```
param_grid_xgb = {
    'n_estimators': [100, 200, 300],
    'max_depth': [3, 5, 7, 9],
    'learning_rate': [0.01, 0.05, 0.1, 0.3],
    'subsample': [0.8, 1.0],
    'colsample_bytree': [0.8, 1.0],
    'gamma': [0, 0.1, 0.2],
    'min_child_weight': [1, 3, 5]
}
```

```
gs_xgb = GridSearchCV(
    xgb,
    param_grid=param_grid_xgb,
    scoring='f1',
    cv=5,
    n_jobs=-1,
    refit=True,
    verbose=0
)
```

The best parameters had a f1 score of **0.893** and came out to be:

```
gs_xgb.best_params_
```

```
{'colsample_bytree': 1.0,
 'gamma': 0.1,
 'learning_rate': 0.1,
 'max_depth': 7,
 'min_child_weight': 1,
 'n_estimators': 100,
 'subsample': 0.8}
```

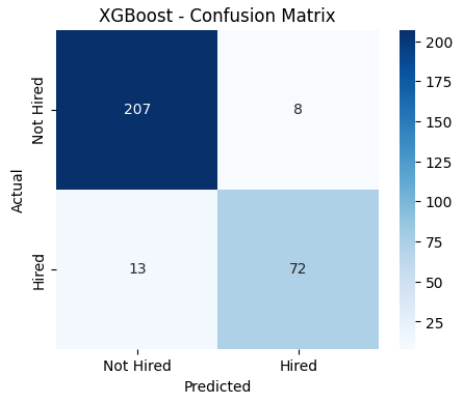
Model results:

Accuracy & Classification report:

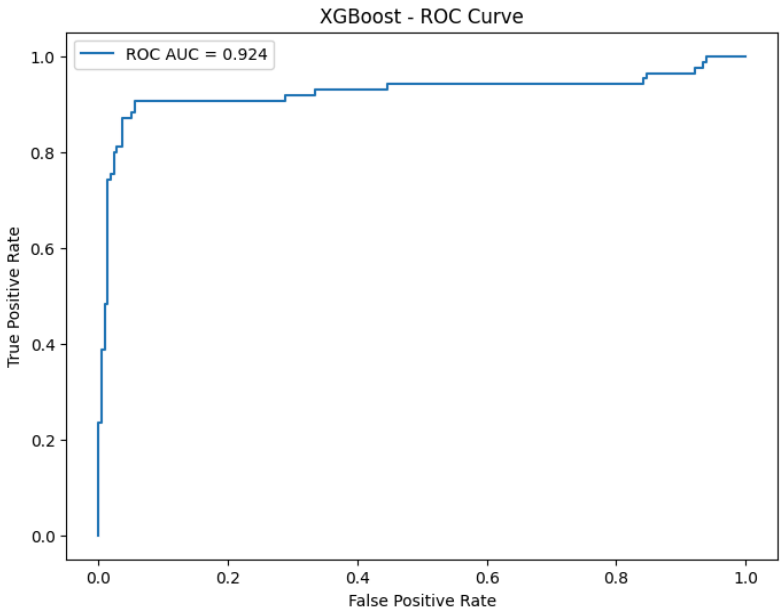
Test Accuracy : 0.93

```
XGBoost Classification report:
              precision    recall  f1-score   support

           0      0.941     0.963     0.952       215
           1      0.900     0.847     0.873        85

    accuracy                          0.930       300
   macro avg      0.920     0.905     0.912       300
weighted avg      0.929     0.930     0.929       300
```

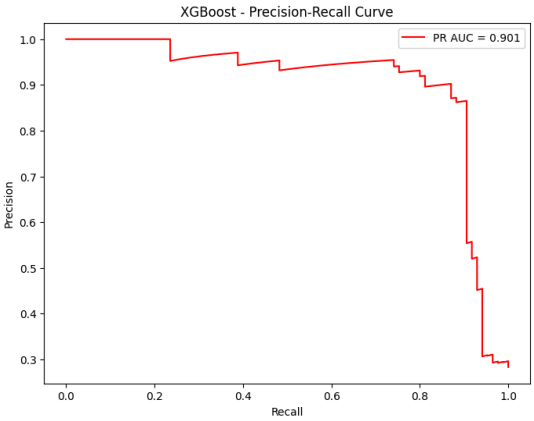
Confusion matrix:



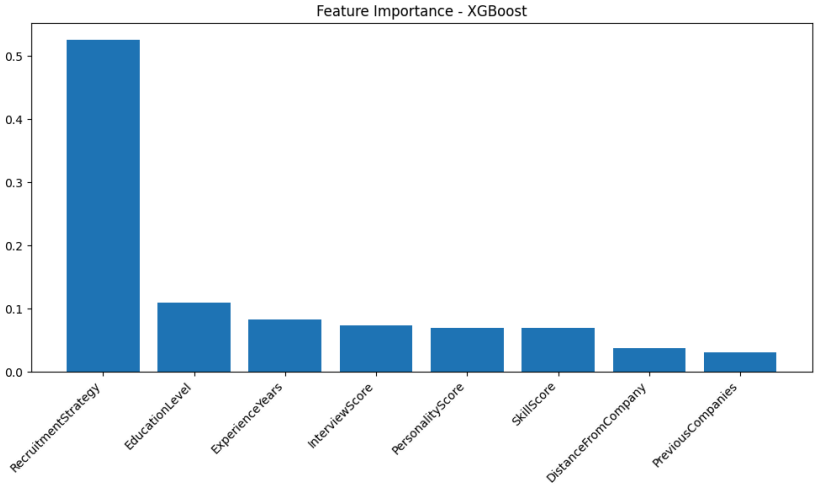
ROC curve:



Precision recall curve:



Feature importance:



Model Comparison

	Model	Accuracy	F1 Score	ROC AUC
2	AdaBoost	0.946667	0.904762	0.929877
1	Random Forest	0.936667	0.886228	0.931436
0	Decision Tree	0.936667	0.884848	0.891436
3	XGBoost	0.930000	0.872727	0.923721

How to run

The .ipynb notebook file is very easy to run. You just need to download the file and upload to Google Colab and everything should be populated and be able to run with the “Run all” button.