```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
```

```
df = pd.read_csv("/content/placementdata.csv")
df.head()
```

	StudentID	CGPA	Internships	Projects	Workshops/Certifications	AptitudeTestScore	SoftSkillsRatin
0	1	7.5	1	1	1	65	4.
1	2	8.9	0	3	2	90	4.
2	3	7.3	1	2	2	82	4.
3	4	7.5	1	1	2	85	4.
4	5	8.3	1	2	2	86	4.

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

```
# Convert categorical variables to numerical using Label Encoding
le = LabelEncoder()
df['PlacementStatus'] = le.fit_transform(df['PlacementStatus'])
df['ExtracurricularActivities'] = le.fit_transform(df['ExtracurricularActivities'])
df['PlacementTraining'] = le.fit_transform(df['PlacementTraining'])
```

```
# Define features (X) and target (y)
X = df[['CGPA', 'Internships', 'Projects', 'Workshops/Certifications',
        'AptitudeTestScore', 'SoftSkillsRating', 'ExtracurricularActivities',
        'PlacementTraining', 'SSC_Marks', 'HSC_Marks']]
y = df['PlacementStatus']
```

```
# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
# Feature scaling
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```
# Initialize and train a RandomForestClassifier (you can try other classifiers)
model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)
```

```
▾ RandomForestClassifier ⓘ ?
RandomForestClassifier(random_state=42)
```

```
# Make predictions on the test set
y_pred = model.predict(X_test)
```

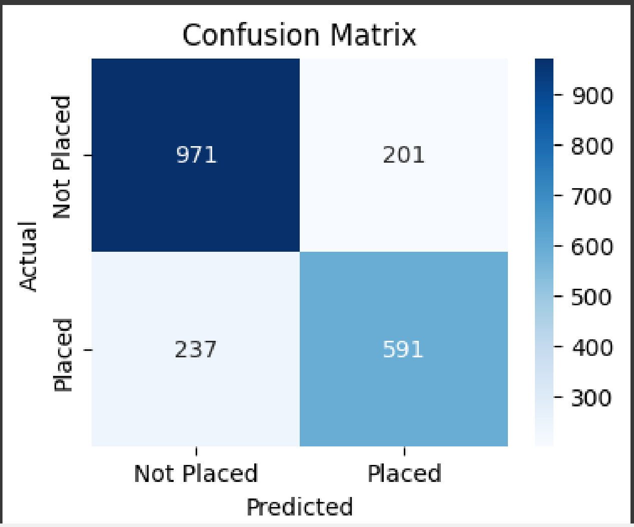
```
# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")
```

```
Accuracy: 0.781
```

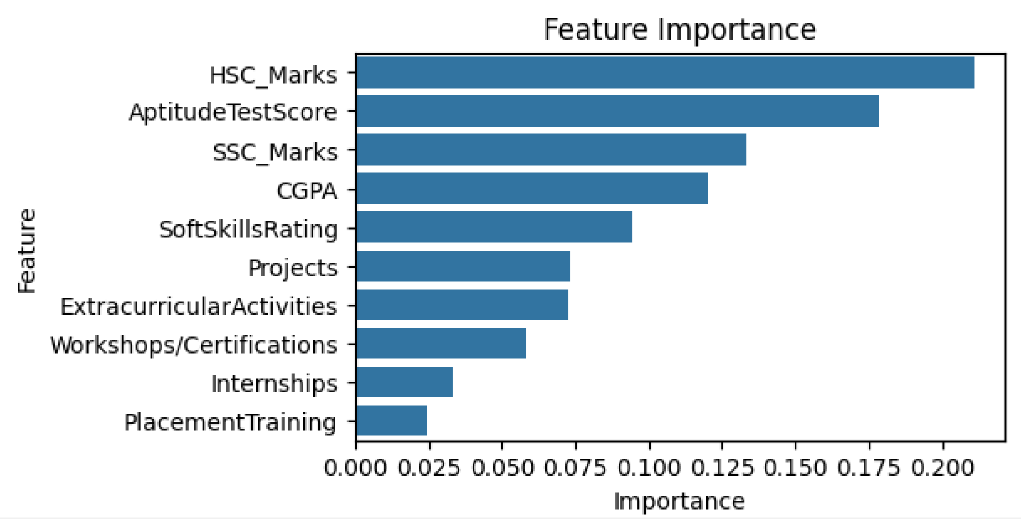
```
# Evaluate the model with additional metrics
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.80      0.83      0.82      1172
           1       0.75      0.71      0.73       828

    accuracy                           0.78      2000
   macro avg       0.78      0.77      0.77      2000
weighted avg       0.78      0.78      0.78      2000
```

```
# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(4, 3))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=['Not Placed', 'Placed'], yticklabels=['Not Placed', 'Placed'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```



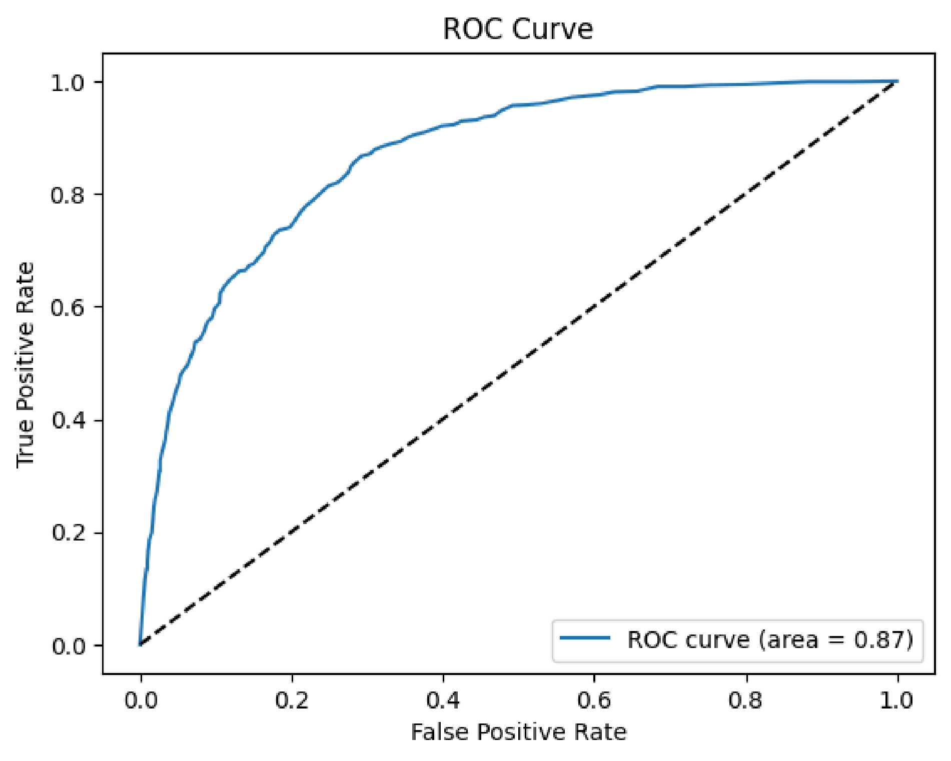
```
# Feature Importance
feature_importances = model.feature_importances_
feature_names = X.columns
importance_df = pd.DataFrame({'Feature': feature_names, 'Importance': feature_importances})
importance_df = importance_df.sort_values(by='Importance', ascending=False)
plt.figure(figsize=(5, 3))
sns.barplot(x='Importance', y='Feature', data=importance_df)
plt.title('Feature Importance')
plt.show()
```



```python
#ROC AUC (requires probability predictions)
from sklearn.metrics import roc_auc_score, roc_curve
y_pred_prob = model.predict_proba(X_test)[:, 1]
roc_auc = roc_auc_score(y_test, y_pred_prob)
print(f"ROC AUC Score: {roc_auc}")
```

```
ROC AUC Score: 0.86730484658126
```

```python
fpr, tpr, thresholds = roc_curve(y_test, y_pred_prob)
plt.plot(fpr, tpr, label=f'ROC curve (area = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], 'k--')  # Diagonal line
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend(loc='lower right')
plt.show()
```


ROC Curve
True Positive Rate
False Positive Rate
ROC curve (area = 0.87)