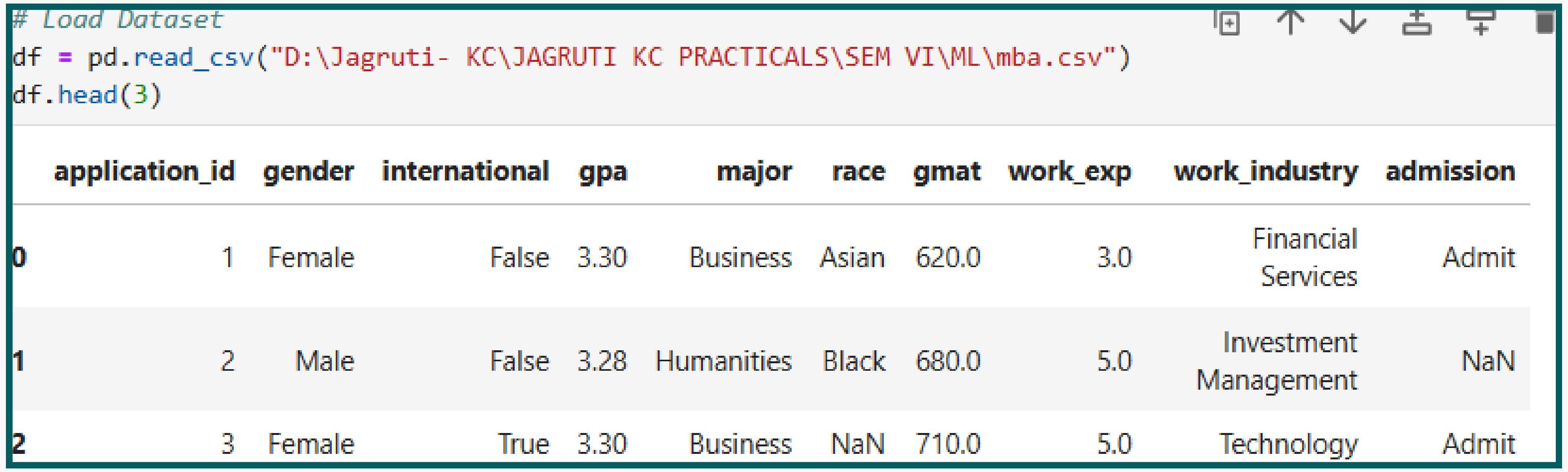
MBA ADMISSION PREDICTION USING RANDOM FOREST

Objective:

- To analyze MBA admission data and predict admissions using a Random Forest Classifier.
- Understand key patterns through Exploratory Data Analysis (EDA).
- Improve model accuracy with proper data preprocessing.

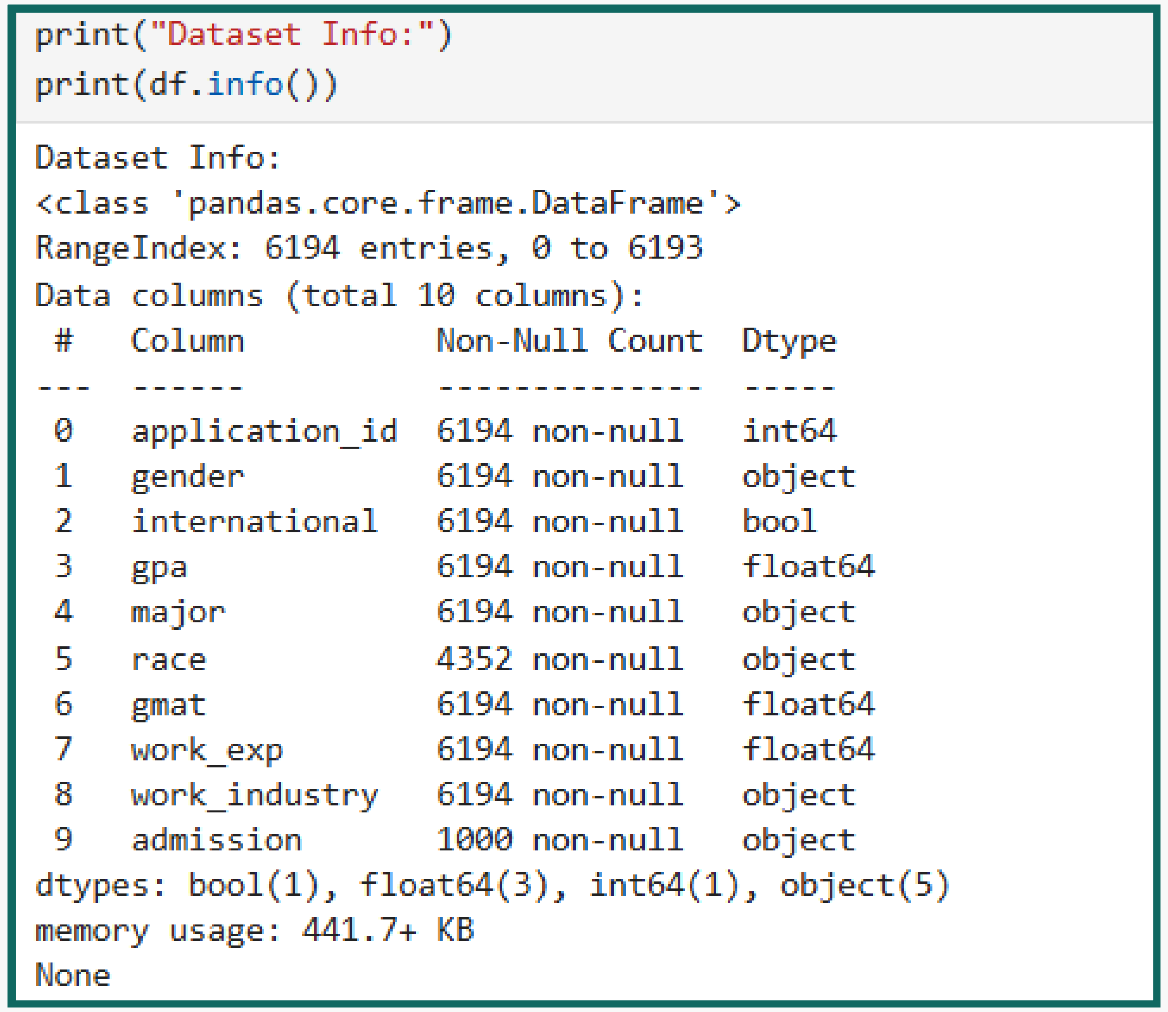
Dataset Overview

- Total Entries: 6,194
- Target Variable: admission (Categorical, with many missing values)
- Notable Issues:
  - ***race*** column has many missing values (Only 4,352 non-null).
  - ***admission*** column has mostly missing values (Only 1,000 non-null).
  - ***work_exp*** is numerical but may need scaling.
  - ***international*** is a boolean (can be converted to 0/1).



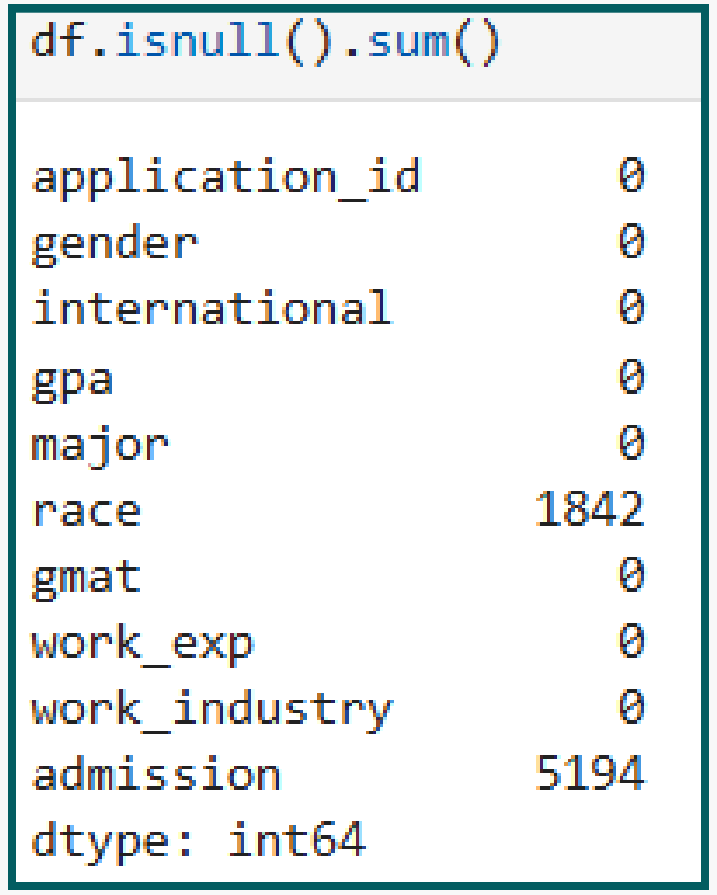
```
# Load Dataset
df = pd.read_csv("D:\Jagruti- KC\JAGRUTI KC PRACTICALS\SEM VI\ML\mba.csv")
df.head(3)
```

	application_id	gender	international	gpa	major	race	gmat	work_exp	work_industry	admission
0	1	Female	False	3.30	Business	Asian	620.0	3.0	Financial Services	Admit
1	2	Male	False	3.28	Humanities	Black	680.0	5.0	Investment Management	NaN
2	3	Female	True	3.30	Business	NaN	710.0	5.0	Technology	Admit



```
print("Dataset Info:")
print(df.info())
```

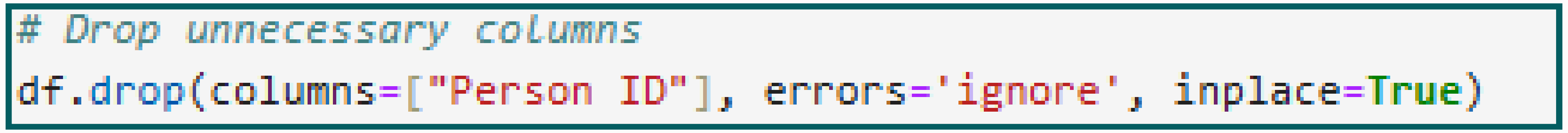
```
Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6194 entries, 0 to 6193
Data columns (total 10 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   application_id  6194 non-null   int64
 1   gender          6194 non-null   object
 2   international   6194 non-null   bool
 3   gpa             6194 non-null   float64
 4   major           6194 non-null   object
 5   race            4352 non-null   object
 6   gmat            6194 non-null   float64
 7   work_exp        6194 non-null   float64
 8   work_industry   6194 non-null   object
 9   admission       1000 non-null   object
dtypes: bool(1), float64(3), int64(1), object(5)
memory usage: 441.7+ KB
None
```



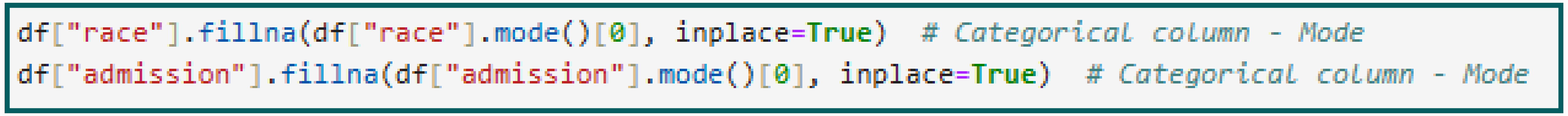
```
df.isnull().sum()
```

```
application_id       0
gender               0
international         0
gpa                  0
major                0
race              1842
gmat                 0
work_exp             0
work_industry        0
admission         5194
dtype: int64
```

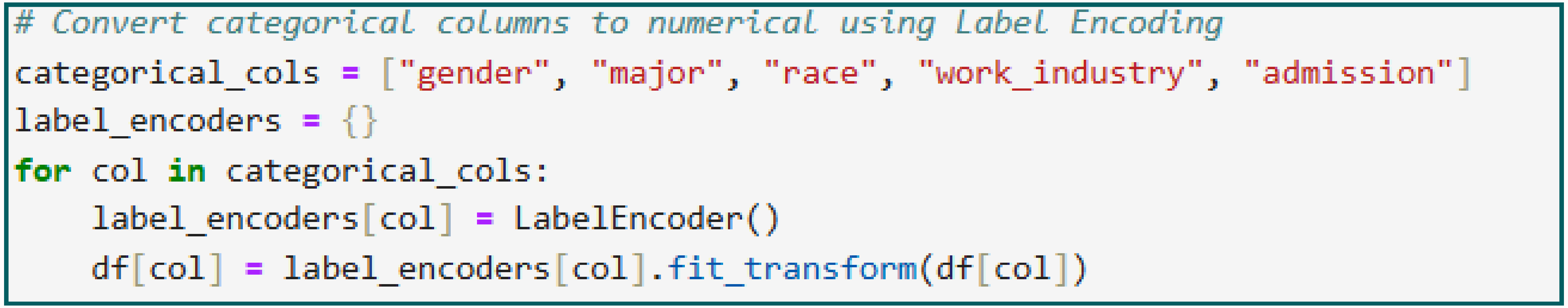
DATA PRE-PROCESSING



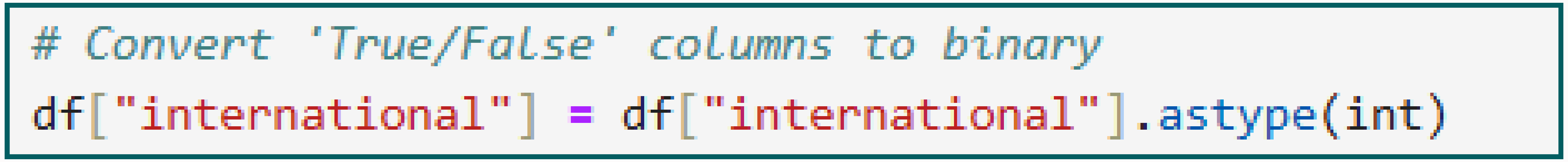
```python
# Drop unnecessary columns
df.drop(columns=["Person ID"], errors='ignore', inplace=True)
```



```python
df["race"].fillna(df["race"].mode()[0], inplace=True)  # Categorical column - Mode
df["admission"].fillna(df["admission"].mode()[0], inplace=True)  # Categorical column - Mode
```

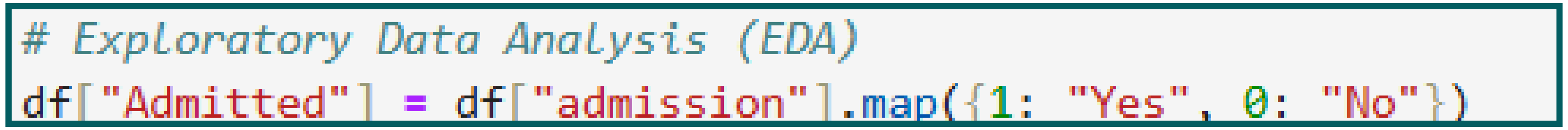


```python
# Convert categorical columns to numerical using Label Encoding
categorical_cols = ["gender", "major", "race", "work_industry", "admission"]
label_encoders = {}
for col in categorical_cols:
    label_encoders[col] = LabelEncoder()
    df[col] = label_encoders[col].fit_transform(df[col])
```

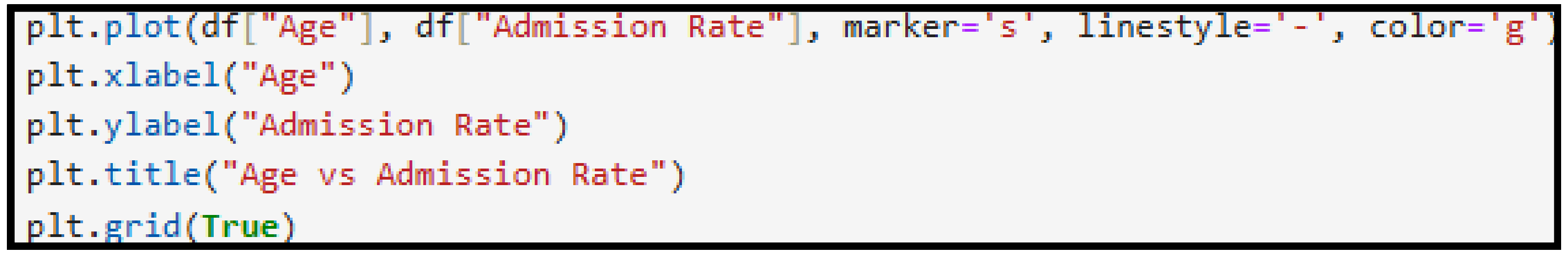


```python
# Convert 'True/False' columns to binary
df["international"] = df["international"].astype(int)
```

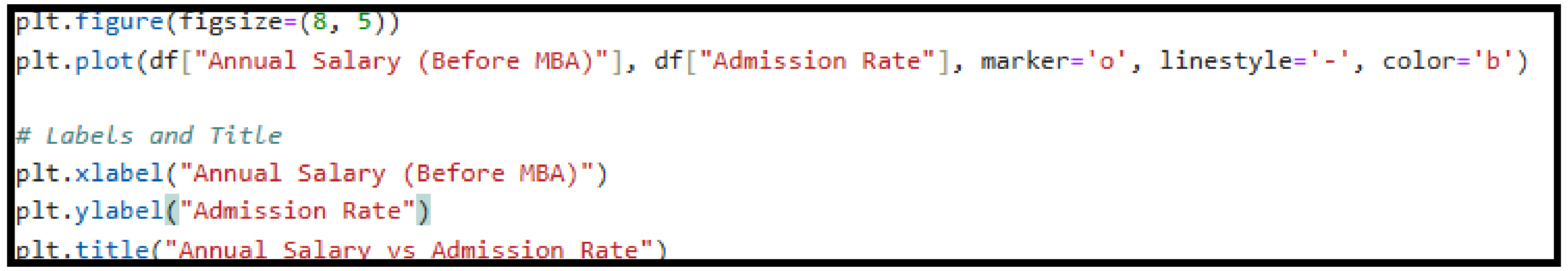
EXPLORATORY DATA ANALYSIS (EDA)



```python
# Exploratory Data Analysis (EDA)
df["Admitted"] = df["admission"].map({1: "Yes", 0: "No"})
```

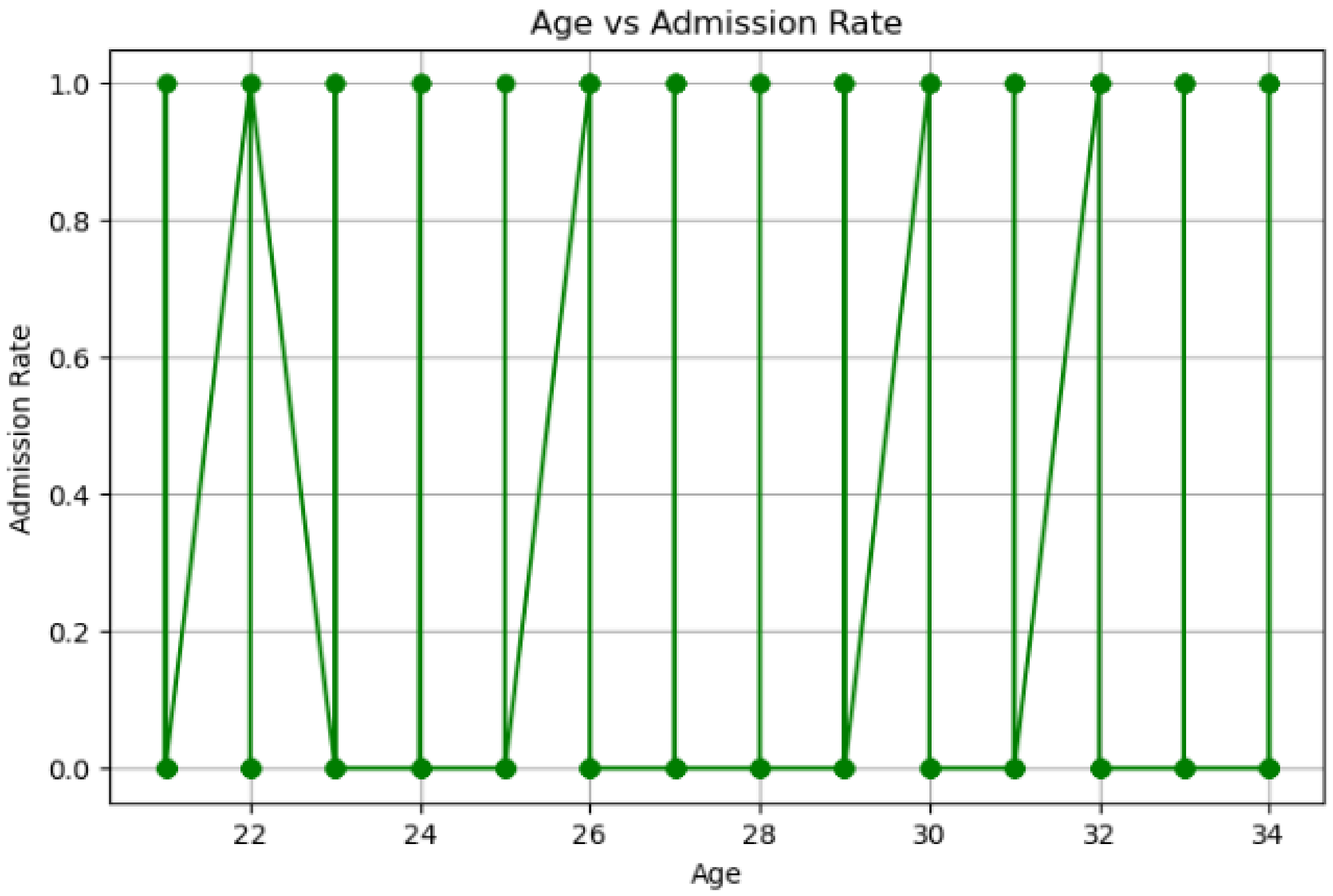


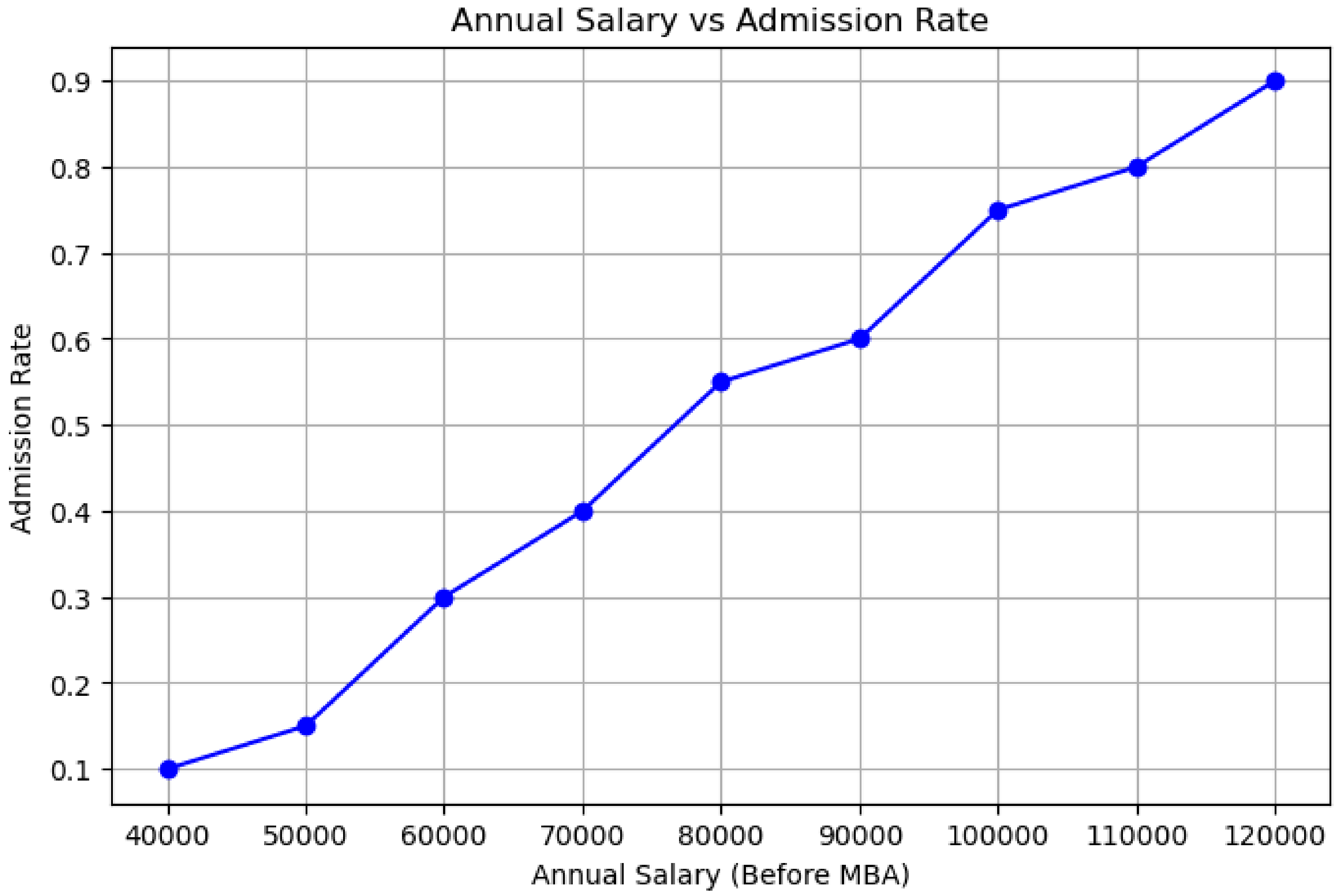
```python
plt.plot(df["Age"], df["Admission Rate"], marker='s', linestyle='-', color='g')
plt.xlabel("Age")
plt.ylabel("Admission Rate")
plt.title("Age vs Admission Rate")
plt.grid(True)
```



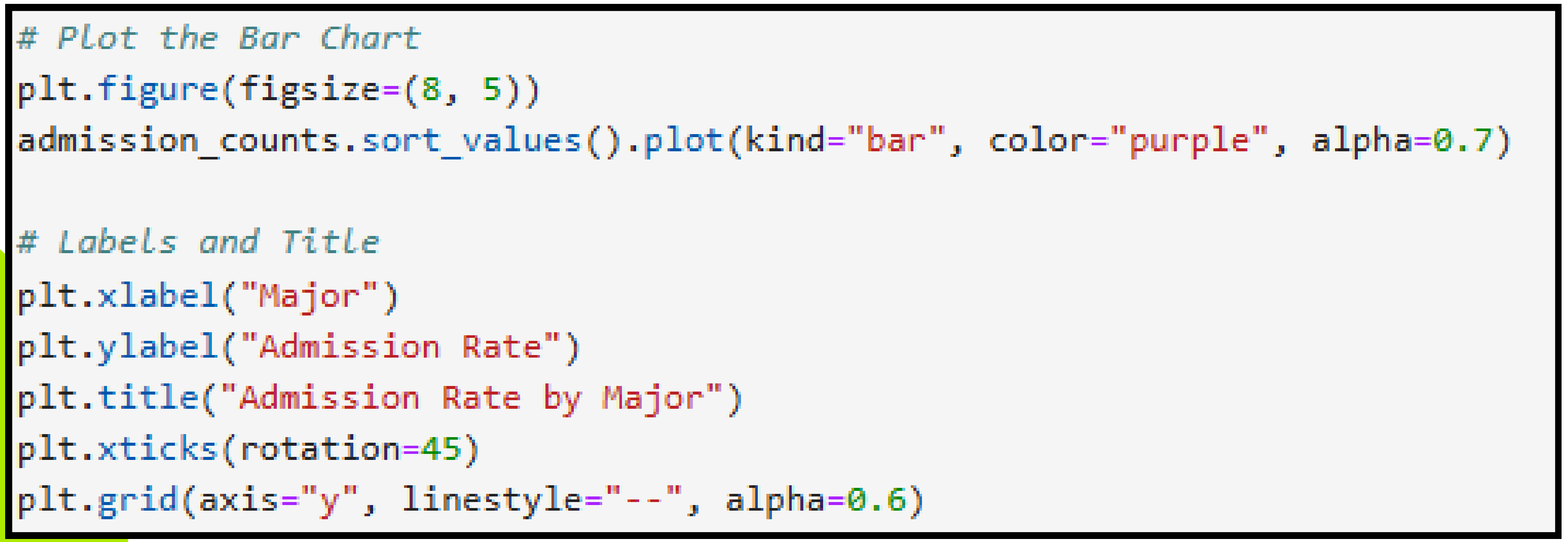
```python
plt.figure(figsize=(8, 5))
plt.plot(df["Annual Salary (Before MBA)"], df["Admission Rate"], marker='o', linestyle='-', color='b')

# Labels and Title
plt.xlabel("Annual Salary (Before MBA)")
plt.ylabel("Admission Rate")
plt.title("Annual Salary vs Admission Rate")
```



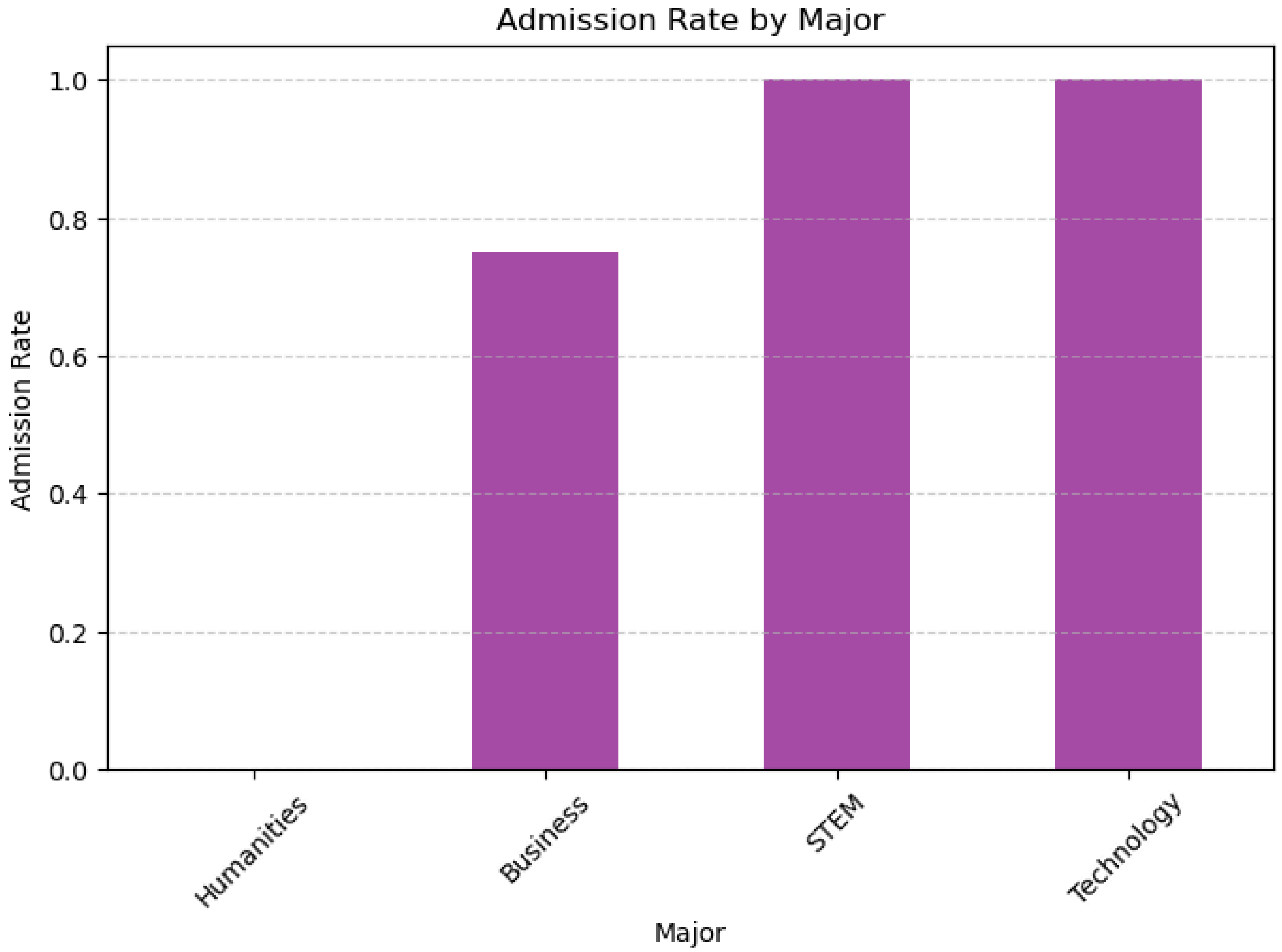


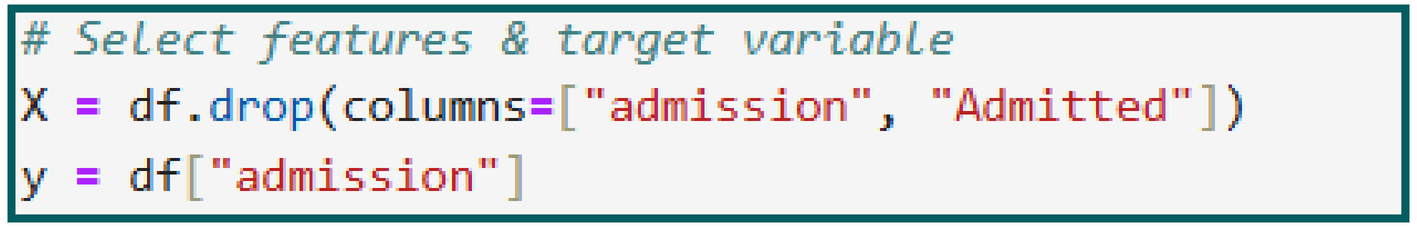
EXPLORATORY DATA ANALYSIS (EDA)



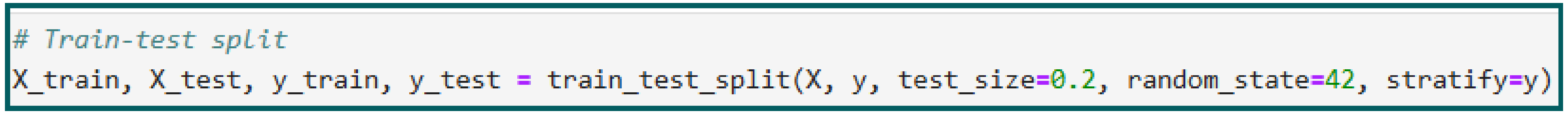
```
# Plot the Bar Chart
plt.figure(figsize=(8, 5))
admission_counts.sort_values().plot(kind="bar", color="purple", alpha=0.7)

# Labels and Title
plt.xlabel("Major")
plt.ylabel("Admission Rate")
plt.title("Admission Rate by Major")
plt.xticks(rotation=45)
plt.grid(axis="y", linestyle="--", alpha=0.6)
```

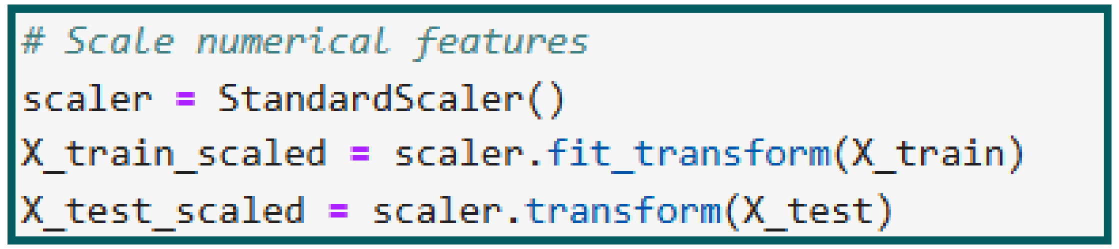




```
# Select features & target variable
X = df.drop(columns=["admission", "Admitted"])
y = df["admission"]
```

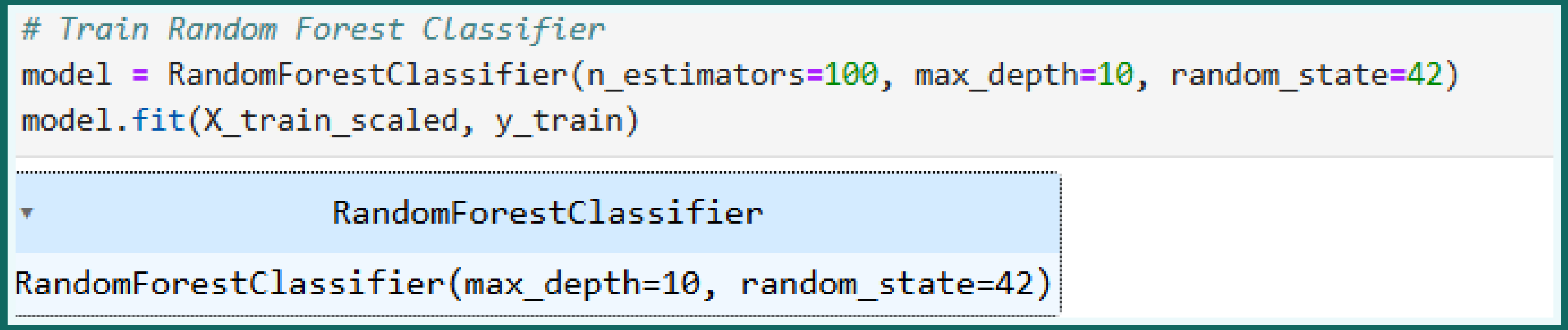


```
# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)
```



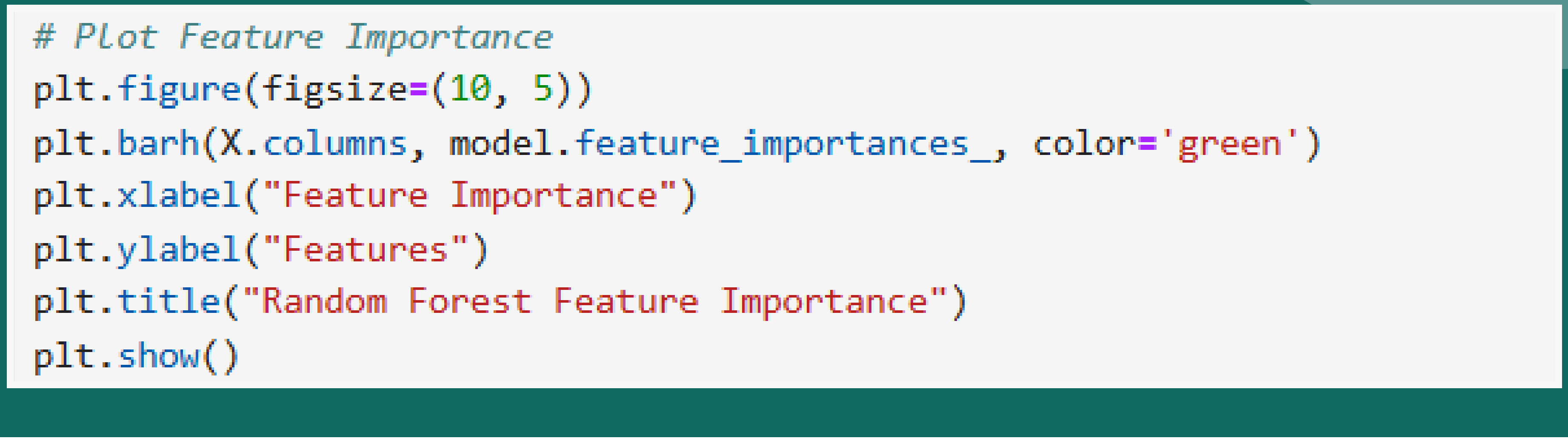
```
# Scale numerical features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

MODEL TRAINING & ANALYSIS

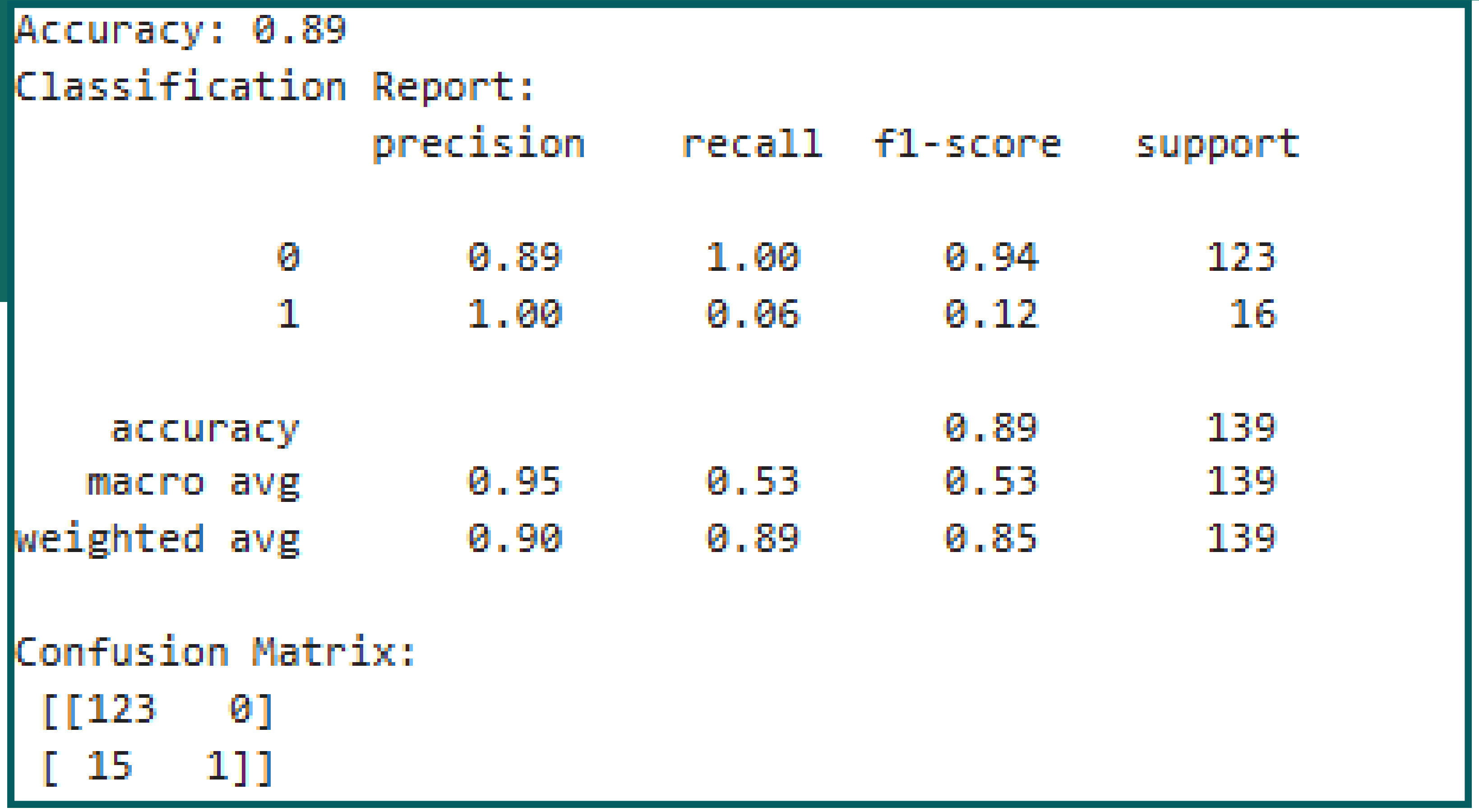


```
# Train Random Forest Classifier
model = RandomForestClassifier(n_estimators=100, max_depth=10, random_state=42)
model.fit(X_train_scaled, y_train)
```

```
▾            RandomForestClassifier
RandomForestClassifier(max_depth=10, random_state=42)
```



```
# Plot Feature Importance
plt.figure(figsize=(10, 5))
plt.barh(X.columns, model.feature_importances_, color='green')
plt.xlabel("Feature Importance")
plt.ylabel("Features")
plt.title("Random Forest Feature Importance")
plt.show()
```

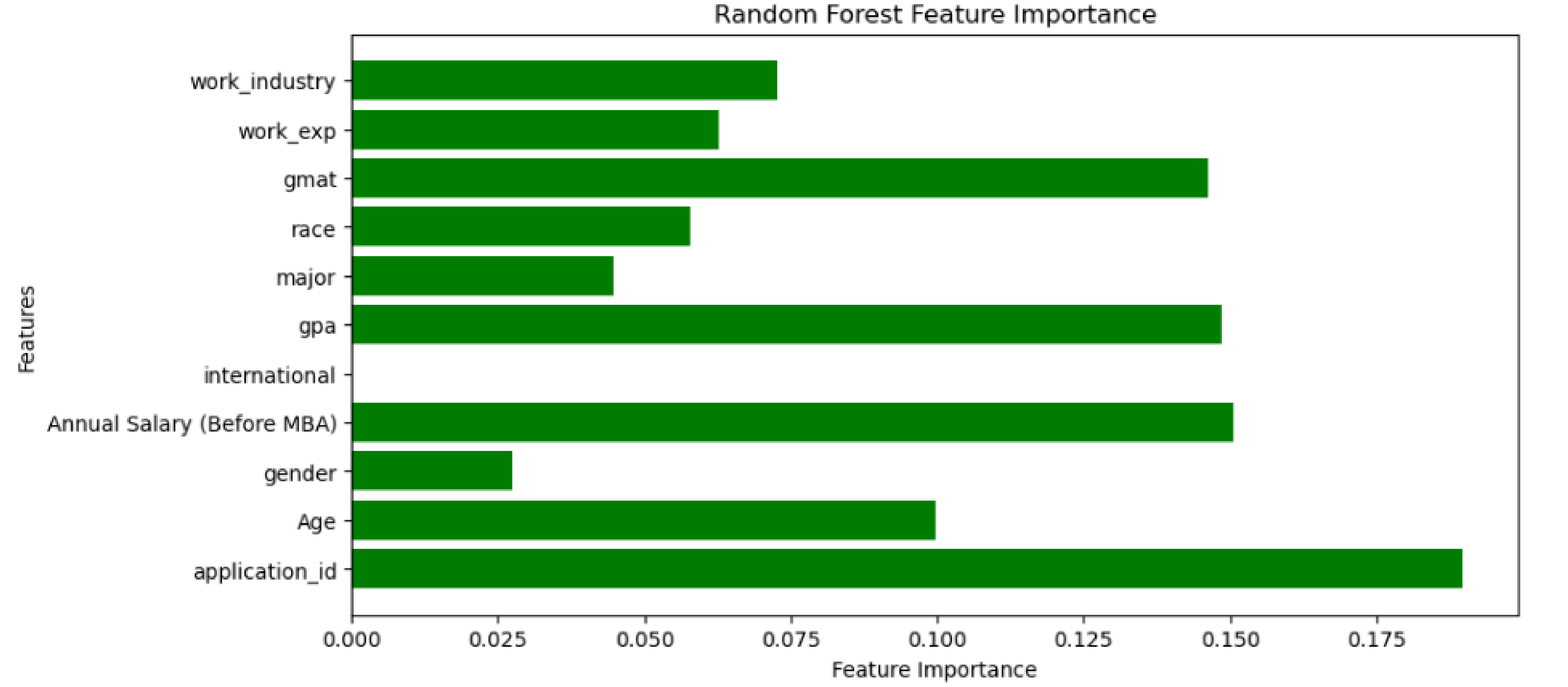


```
Accuracy: 0.89
Classification Report:
              precision    recall  f1-score   support

           0       0.89      1.00      0.94       123
           1       1.00      0.06      0.12        16

    accuracy                           0.89       139
   macro avg       0.95      0.53      0.53       139
weighted avg       0.90      0.89      0.85       139

Confusion Matrix:
 [[123   0]
 [ 15   1]]
```


Random Forest Feature Importance
work_industry
work_exp
gmat
race
major
gpa
international
Annual Salary (Before MBA)
gender
Age
application_id
Features
Feature Importance