CODE FOR THE GIVEN PROBLEM STATEMENT

```
# Importing necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split,
GridSearchCV
from sklearn.preprocessing import StandardScaler,
LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report,
confusion_matrix, roc_auc_score, roc_curve

# Step 1: Load the data
# Load the data from the uploaded CSV file
data
=pd.read_csv('/mnt/data/customer_purchase_data.csv')

# Step 2: Explore the dataset
print("First few rows of the dataset:")
print(data.head())  # Display first few rows of the dataset

print("\nDataset information:")
print(data.info())  # Summary of the dataset

print("\nStatistical summary of the dataset:")
print(data.describe())  # Statistical summary of numerical
columns

print("\nChecking for missing values:")
```

```
print(data.isnull().sum())  # Checking for missing values

# Step 3: Data Preprocessing
# Handling missing values (example: filling missing
numerical values with mean)
data.fillna(data.mean(), inplace=True)

# Encoding categorical variables
label_encoders = {}
for column in
data.select_dtypes(include=['object']).columns:
    le = LabelEncoder()
    data[column] = le.fit_transform(data[column])
    label_encoders[column] = le

# Replace 'target_column' with the actual target column
name from your dataset
PurchaseStatus= 'PurchaseStatus'  # Adjust this with the
actual column name
X = data.drop(PurchaseStatus, axis=1)
y = data[PurchaseStatus]

# Train-test split
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)

# Step 4: Model Development
# Using RandomForestClassifier as an example
```

```
model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)

# Step 5: Model Evaluation
y_pred = model.predict(X_test)
print("\nConfusion Matrix:")
print(confusion_matrix(y_test, y_pred))

print("\nClassification Report:")
print(classification_report(y_test, y_pred))

# ROC-AUC Score
roc_auc = roc_auc_score(y_test,
model.predict_proba(X_test)[:, 1])
print(f'\nROC-AUC Score: {roc_auc}')

# Plotting the ROC Curve
fpr, tpr, thresholds = roc_curve(y_test,
model.predict_proba(X_test)[:, 1])
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, label=f'AUC = {roc_auc:.2f}')
plt.plot([0, 1], [0, 1], linestyle='--', color='gray')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend()
plt.grid()
plt.show()

# Step 6: Hyperparameter Tuning
# Example: Tuning Random Forest hyperparameters
param_grid = {
    'n_estimators': [100, 200, 300],
```

```python
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10]
}

grid_search = GridSearchCV(estimator=model,
param_grid=param_grid, cv=5, scoring='roc_auc',
n_jobs=-1)
grid_search.fit(X_train, y_train)

# Best Parameters from Grid Search
print("\nBest Parameters found by Grid Search:")
print(grid_search.best_params_)

# Best Model Evaluation
best_model = grid_search.best_estimator_
y_pred_best = best_model.predict(X_test)
print("\nBest Model Confusion Matrix:")
print(confusion_matrix(y_test, y_pred_best))

print("\nBest Model Classification Report:")
print(classification_report(y_test, y_pred_best))

# Best Model ROC-AUC Score
roc_auc_best = roc_auc_score(y_test,
best_model.predict_proba(X_test)[:, 1])
print(f'\nBest Model ROC-AUC Score: {roc_auc_best}')
```

SOLUTION OR OUTPUT FOR THE PROBLEM STATEMENT

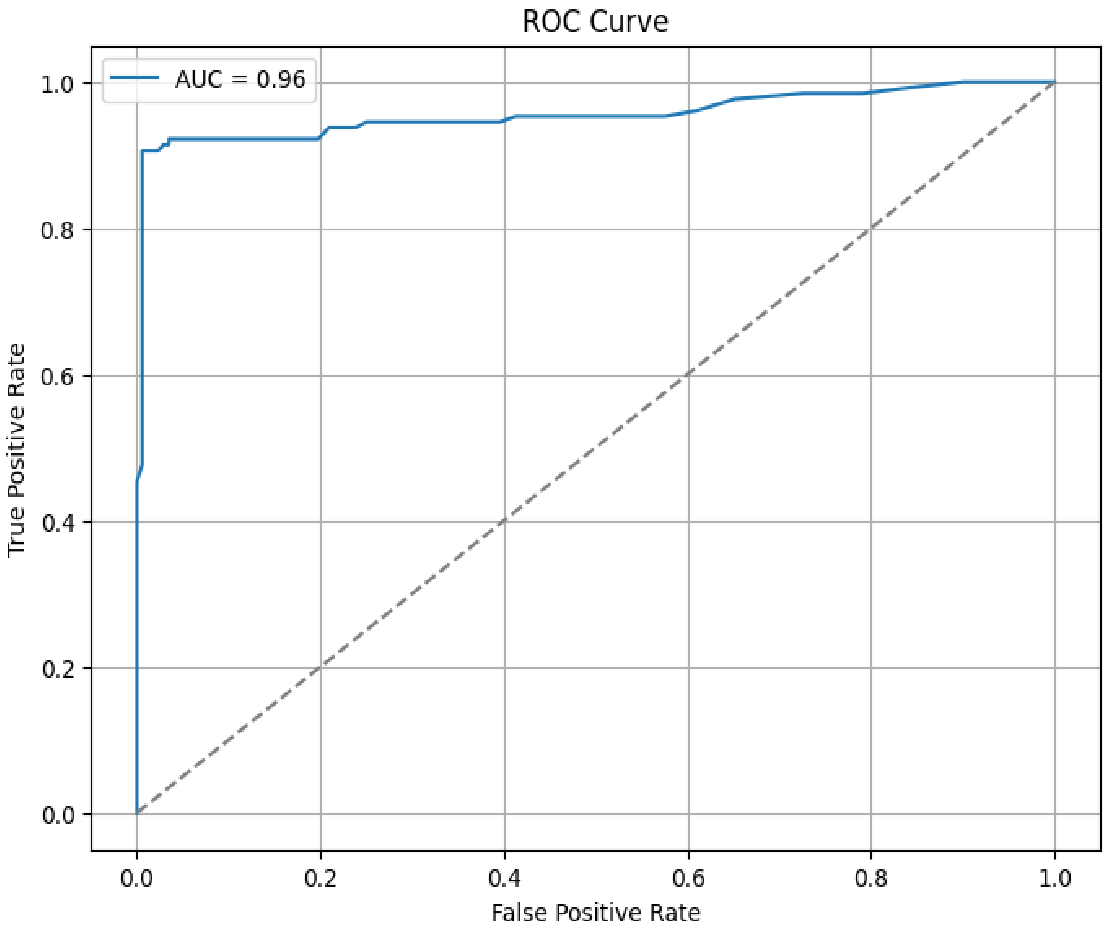
```
First few rows of the dataset:
   Age  Gender   AnnualIncome  NumberOfPurchases  ProductCategory  \
0   40       1   66120.267939                  8                0
```

```
1   20       1   23579.773583                4              2
2   27       1  127821.306432               11              2
3   24       1  137798.623120               19              3
4   31       1   99300.964220               19              1

   TimeSpentOnWebsite  LoyaltyProgram  DiscountsAvailed
PurchaseStatus
0           30.568601               0                 5               1
1           38.240097               0                 5               0
2           31.633212               1                 0               1
3           46.167059               0                 4               1
4           19.823592               0                 0               1

Dataset information:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1500 entries, 0 to 1499
Data columns (total 9 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Age                 1500 non-null   int64
 1   Gender              1500 non-null   int64
 2   AnnualIncome        1500 non-null   float64
 3   NumberOfPurchases   1500 non-null   int64
 4   ProductCategory     1500 non-null   int64
 5   TimeSpentOnWebsite  1500 non-null   float64
 6   LoyaltyProgram      1500 non-null   int64
 7   DiscountsAvailed    1500 non-null   int64
 8   PurchaseStatus      1500 non-null   int64
dtypes: float64(2), int64(7)
memory usage: 105.6 KB
None

Statistical summary of the dataset:
               Age       Gender   AnnualIncome  NumberOfPurchases  \
count  1500.000000  1500.000000    1500.000000        1500.000000
mean     44.298667     0.504667   84249.164338          10.420000
std      15.537259     0.500145   37629.493078           5.887391
min      18.000000     0.000000   20001.512518           0.000000
25%      31.000000     0.000000   53028.979155           5.000000
50%      45.000000     1.000000   83699.581476          11.000000
75%      57.000000     1.000000  117167.772858          15.000000
max      70.000000     1.000000  149785.176481          20.000000
```

```
       ProductCategory  TimeSpentOnWebsite  LoyaltyProgram
DiscountsAvailed  \
count      1500.000000        1500.000000     1500.000000
1500.000000
mean          2.012667          30.469040       0.326667         2.555333
std           1.428005          16.984392       0.469151         1.705152
min           0.000000           1.037023       0.000000         0.000000
25%           1.000000          16.156700       0.000000         1.000000
50%           2.000000          30.939516       0.000000         3.000000
75%           3.000000          44.369863       1.000000         4.000000
max           4.000000          59.991105       1.000000         5.000000

       PurchaseStatus
count      1500.00000
mean          0.43200
std           0.49552
min           0.00000
25%           0.00000
50%           0.00000
75%           1.00000
max           1.00000

Checking for missing values:
Age                  0
Gender               0
AnnualIncome         0
NumberOfPurchases    0
ProductCategory      0
TimeSpentOnWebsite   0
LoyaltyProgram       0
DiscountsAvailed     0
PurchaseStatus       0
dtype: int64

Confusion Matrix:
[[171   1]
 [ 13 115]]

Classification Report:
              precision    recall  f1-score   support

           0       0.93      0.99      0.96       172
           1       0.99      0.90      0.94       128
```

accuracy			0.95	300
macro avg	0.96	0.95	0.95	300
weighted avg	0.96	0.95	0.95	300

ROC-AUC Score: 0.9556458938953489



Best Parameters found by Grid Search:
{'max_depth': None, 'min_samples_split': 5, 'n_estimators': 200}

Best Model Confusion Matrix:
[[169   3]
 [13 115]]

Best Model Classification Report:

	precision	recall	f1-score	support
0	0.93	0.98	0.95	172
1	0.97	0.90	0.93	128

accuracy			0.95	300
macro avg	0.95	0.94	0.94	300
weighted avg	0.95	0.95	0.95	300

Best Model ROC-AUC Score: 0.9543059593023255