10wmwwd7c

January 27, 2025

```
[123]: import numpy as np
       import pandas as pd
       import matplotlib.pyplot as plt
       import seaborn as sns
       from sklearn.model_selection import train_test_split
       import xgboost as xgb
       from sklearn.preprocessing import LabelEncoder
       from sklearn.preprocessing import MinMaxScaler
       from sklearn.metrics import accuracy_score, classification_report,␣
        ↪confusion_matrix,roc_auc_score, roc_curve
```

```
[73]: #loading the dataset
      data = pd.read_csv(r"C:\Users\91703\Downloads\credit card.csv")
```

```
[74]: data.head()
```

```
[74]:    Time        V1        V2        V3        V4        V5        V6        V7  \
      0   0.0 -1.359807 -0.072781  2.536347  1.378155 -0.338321  0.462388  0.239599
      1   0.0  1.191857  0.266151  0.166480  0.448154  0.060018 -0.082361 -0.078803
      2   1.0 -1.358354 -1.340163  1.773209  0.379780 -0.503198  1.800499  0.791461
      3   1.0 -0.966272 -0.185226  1.792993 -0.863291 -0.010309  1.247203  0.237609
      4   2.0 -1.158233  0.877737  1.548718  0.403034 -0.407193  0.095921  0.592941

               V8        V9  …       V21       V22       V23       V24       V25  \
      0  0.098698  0.363787  … -0.018307  0.277838 -0.110474  0.066928  0.128539
      1  0.085102 -0.255425  … -0.225775 -0.638672  0.101288 -0.339846  0.167170
      2  0.247676 -1.514654  …  0.247998  0.771679  0.909412 -0.689281 -0.327642
      3  0.377436 -1.387024  … -0.108300  0.005274 -0.190321 -1.175575  0.647376
      4 -0.270533  0.817739  … -0.009431  0.798278 -0.137458  0.141267 -0.206010

              V26       V27       V28  Amount  Class
      0 -0.189115  0.133558 -0.021053  149.62      0
      1  0.125895 -0.008983  0.014724    2.69      0
      2 -0.139097 -0.055353 -0.059752  378.66      0
      3 -0.221929  0.062723  0.061458  123.50      0
      4  0.502292  0.219422  0.215153   69.99      0
```

```
[5 rows x 31 columns]
```

[75]:
```
data.describe()
```

[75]:
```
                Time             V1             V2             V3  \
count  283726.000000  283726.000000  283726.000000  283726.000000
mean    94811.077600       0.005917      -0.004135       0.001613
std     47481.047891       1.948026       1.646703       1.508682
min         0.000000     -56.407510     -72.715728     -48.325589
25%     54204.750000      -0.915951      -0.600321      -0.889682
50%     84692.500000       0.020384       0.063949       0.179963
75%    139298.000000       1.316068       0.800283       1.026960
max    172792.000000       2.454930      22.057729       9.382558

                  V4             V5             V6             V7  \
count  283726.000000  283726.000000  283726.000000  283726.000000
mean       -0.002966       0.001828      -0.001139       0.001801
std         1.414184       1.377008       1.331931       1.227664
min        -5.683171    -113.743307     -26.160506     -43.557242
25%        -0.850134      -0.689830      -0.769031      -0.552509
50%        -0.022248      -0.053468      -0.275168       0.040859
75%         0.739647       0.612218       0.396792       0.570474
max        16.875344      34.801666      73.301626     120.589494

                  V8             V9  …            V21            V22  \
count  283726.000000  283726.000000  …  283726.000000  283726.000000
mean       -0.000854      -0.001596  …      -0.000371      -0.000015
std         1.179054       1.095492  …       0.723909       0.724550
min       -73.216718     -13.434066  …     -34.830382     -10.933144
25%        -0.208828      -0.644221  …      -0.228305      -0.542700
50%         0.021898      -0.052596  …      -0.029441       0.006675
75%         0.325704       0.595977  …       0.186194       0.528245
max        20.007208      15.594995  …      27.202839      10.503090

                 V23            V24            V25            V26  \
count  283726.000000  283726.000000  283726.000000  283726.000000
mean        0.000198       0.000214      -0.000232       0.000149
std         0.623702       0.605627       0.521220       0.482053
min       -44.807735      -2.836627     -10.295397      -2.604551
25%        -0.161703      -0.354453      -0.317485      -0.326763
50%        -0.011159       0.041016       0.016278      -0.052172
75%         0.147748       0.439738       0.350667       0.240261
max        22.528412       4.584549       7.519589       3.517346

                 V27            V28         Amount          Class
count  283726.000000  283726.000000  283726.000000  283726.000000
mean        0.001763       0.000547      88.472687       0.001667
```

```
std         0.395744       0.328027     250.399437       0.040796
min       -22.565679     -15.430084       0.000000       0.000000
25%        -0.070641      -0.052818       5.600000       0.000000
50%         0.001479       0.011288      22.000000       0.000000
75%         0.091208       0.078276      77.510000       0.000000
max        31.612198      33.847808   25691.160000       1.000000

[8 rows x 31 columns]
```

[76]:

```
data.isnull().sum()
```

[76]:

```
Time      0
V1        0
V2        0
V3        0
V4        0
V5        0
V6        0
V7        0
V8        0
V9        0
V10       0
V11       0
V12       0
V13       0
V14       0
V15       0
V16       0
V17       0
V18       0
V19       0
V20       0
V21       0
V22       0
V23       0
V24       0
V25       0
V26       0
V27       0
V28       0
Amount    0
Class     0
dtype: int64
```

[77]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 283726 entries, 0 to 283725
Data columns (total 31 columns):
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   Time    283726 non-null  float64
 1   V1      283726 non-null  float64
 2   V2      283726 non-null  float64
 3   V3      283726 non-null  float64
 4   V4      283726 non-null  float64
 5   V5      283726 non-null  float64
 6   V6      283726 non-null  float64
 7   V7      283726 non-null  float64
 8   V8      283726 non-null  float64
 9   V9      283726 non-null  float64
 10  V10     283726 non-null  float64
 11  V11     283726 non-null  float64
 12  V12     283726 non-null  float64
 13  V13     283726 non-null  float64
 14  V14     283726 non-null  float64
 15  V15     283726 non-null  float64
 16  V16     283726 non-null  float64
 17  V17     283726 non-null  float64
 18  V18     283726 non-null  float64
 19  V19     283726 non-null  float64
 20  V20     283726 non-null  float64
 21  V21     283726 non-null  float64
 22  V22     283726 non-null  float64
 23  V23     283726 non-null  float64
 24  V24     283726 non-null  float64
 25  V25     283726 non-null  float64
 26  V26     283726 non-null  float64
 27  V27     283726 non-null  float64
 28  V28     283726 non-null  float64
 29  Amount  283726 non-null  float64
 30  Class   283726 non-null  int64
dtypes: float64(30), int64(1)
memory usage: 67.1 MB
```

[82]:

```python
data = data.applymap(int)

# Verify data types
print(data.dtypes)
import warnings
warnings.filterwarnings('ignore')
```

```
Time      int64
V1        int64
V2        int64
```

```
V3        int64
V4        int64
V5        int64
V6        int64
V7        int64
V8        int64
V9        int64
V10       int64
V11       int64
V12       int64
V13       int64
V14       int64
V15       int64
V16       int64
V17       int64
V18       int64
V19       int64
V20       int64
V21       int64
V22       int64
V23       int64
V24       int64
V25       int64
V26       int64
V27       int64
V28       int64
Amount    int64
Class     int64
dtype: object
```

[83]:

```python
#Checking outliers using Z-Score:
from scipy import stats
data[(np.abs(stats.zscore(data)) < 3).all(axis=1)]
#Outliers handled by not removing any data as z score is less than 3.
```

[83]:

```
          Time  V1  V2  V3  V4  V5  V6  V7  V8  V9  …  V21  V22  V23  V24  \
0            0  -1   0   2   1   0   0   0   0   0  …    0    0    0    0
1            0   1   0   0   0   0   0   0   0   0  …    0    0    0    0
3            1   0   0   1   0   0   1   0   0  -1  …    0    0    0   -1
4            2  -1   0   1   0   0   0   0   0   0  …    0    0    0    0
5            2   0   0   1   0   0   0   0   0   0  …    0    0    0    0
…          …  ..  ..  ..  ..  ..  ..  ..  ..  ..  …  …  …  …  …
283719  172784   2   0  -1   0   0   0   0   0   0  …    0    0    0    0
283720  172785   0   0   0   0   1   0   0   0   0  …    0    0    0    0
283722  172787   0   0   2   0   0   1   0   0   0  …    0    0    0   -1
283723  172788   1   0  -3   0   2   3   0   0   0  …    0    0    0    0
283725  172792   0   0   0   0   0   0   1   0   0  …    0    0    0    0
```

```
        V25  V26  V27  V28  Amount  Class
0         0    0    0    0     149      0
1         0    0    0    0       2      0
3         0    0    0    0     123      0
4         0    0    0    0      69      0
5         0    0    0    0       3      0
…     …  …  …  …     …    …
283719    0    0    0    0       2      0
283720    0    0    0    0       2      0
283722    0    0    0    0      24      0
283723    0    0    0    0      67      0
283725    0    0    0    0     217      0

[206274 rows x 31 columns]
```

[84]:

```
data.head()
```

[84]:

```
   Time  V1  V2  V3  V4  V5  V6  V7  V8  V9  …  V21  V22  V23  V24  V25  \
0     0  -1   0   2   1   0   0   0   0   0  …    0    0    0    0    0
1     0   1   0   0   0   0   0   0   0   0  …    0    0    0    0    0
2     1  -1  -1   1   0   0   1   0   0  -1  …    0    0    0    0    0
3     1   0   0   1   0   0   1   0   0  -1  …    0    0    0   -1    0
4     2  -1   0   1   0   0   0   0   0   0  …    0    0    0    0    0

   V26  V27  V28  Amount  Class
0    0    0    0     149      0
1    0    0    0       2      0
2    0    0    0     378      0
3    0    0    0     123      0
4    0    0    0      69      0

[5 rows x 31 columns]
```

[89]:

```
#Scaling data using MinMaxScaler:
scaler = MinMaxScaler()
data['Amount'] = scaler.fit_transform(data[['Amount']])
```

[90]:

```
data.head()
```

[90]:

```
   Time  V1  V2  V3  V4  V5  V6  V7  V8  V9  …  V21  V22  V23  V24  V25  \
0     0  -1   0   2   1   0   0   0   0   0  …    0    0    0    0    0
1     0   1   0   0   0   0   0   0   0   0  …    0    0    0    0    0
2     1  -1  -1   1   0   0   1   0   0  -1  …    0    0    0    0    0
3     1   0   0   1   0   0   1   0   0  -1  …    0    0    0   -1    0
4     2  -1   0   1   0   0   0   0   0   0  …    0    0    0    0    0
```

```
   V26  V27  V28    Amount  Class
0    0    0    0  0.005800      0
1    0    0    0  0.000078      0
2    0    0    0  0.014713      0
3    0    0    0  0.004788      0
4    0    0    0  0.002686      0

[5 rows x 31 columns]
```

[91]:

```
data.columns
```

[91]:

```
Index(['Time', 'V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7', 'V8', 'V9', 'V10',
       'V11', 'V12', 'V13', 'V14', 'V15', 'V16', 'V17', 'V18', 'V19', 'V20',
       'V21', 'V22', 'V23', 'V24', 'V25', 'V26', 'V27', 'V28', 'Amount',
       'Class'],
      dtype='object')
```

[112]:

```
# Define target variable (y) and feature variables (X)
y = data['Class']
X = data[['Time', 'V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7', 'V8', 'V9', 'V10',
       'V11', 'V12', 'V13', 'V14', 'V15', 'V16', 'V17', 'V18', 'V19', 'V20',
       'V21', 'V22', 'V23', 'V24', 'V25', 'V26', 'V27', 'V28','Amount']]
```

[113]:

```
#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)
```

[114]:

```
# Verify unique values in y_train and y_test
print("Unique values in y_train:", np.unique(y_train))
print("Unique values in y_test:", np.unique(y_test))
```

```
Unique values in y_train: [0 1]
Unique values in y_test: [0 1]
```

[115]:

```
# Create and train XGBoost Classifier model
model = xgb.XGBClassifier()
model.fit(X_train, y_train)
```

[115]:

```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              multi_strategy=None, n_estimators=None, n_jobs=None,
              num_parallel_tree=None, random_state=None, …)
```

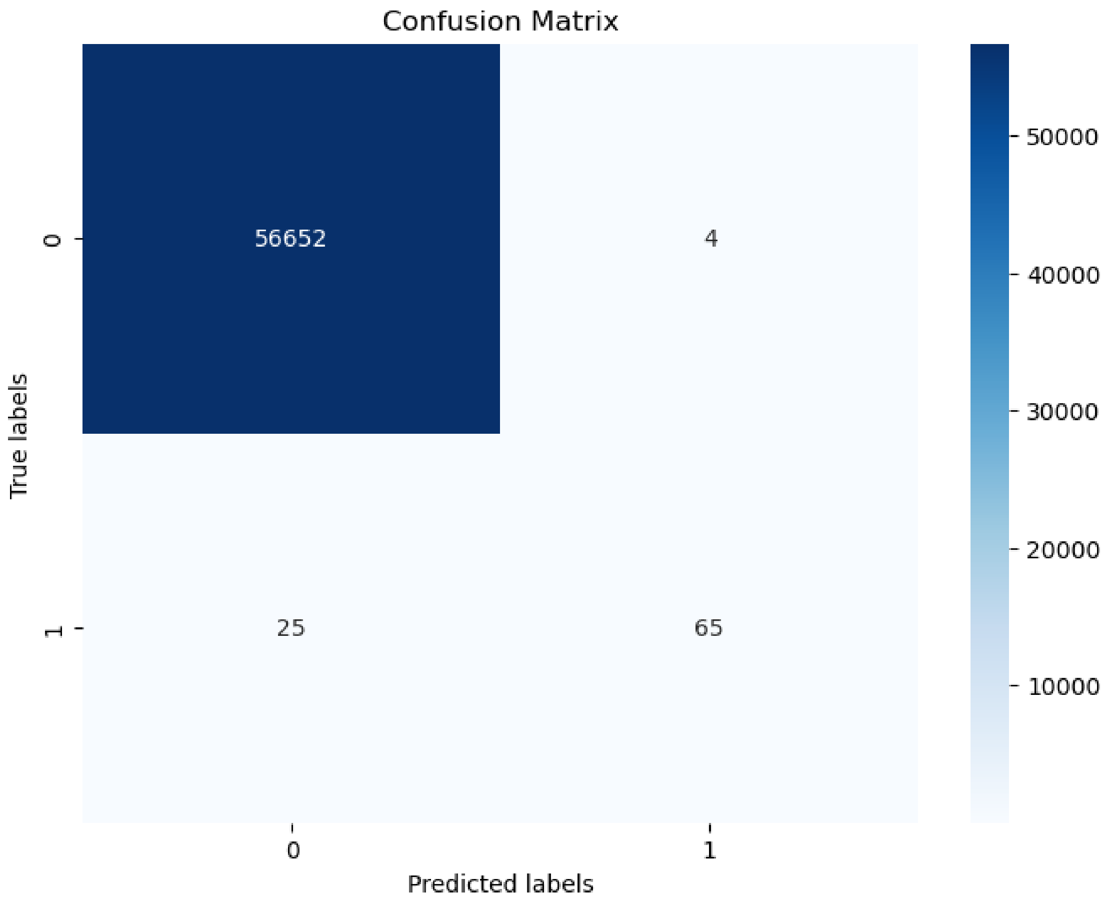
[116]:
```
# Make predictions on test data
y_pred = model.predict(X_test)
print(y_pred)
```

```
[0 0 0 … 0 0 0]
```

[117]:
```
# Evaluate model performance
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
print("Classification Report:", classification_report(y_test, y_pred))
print("Confusion Matrix:", confusion_matrix(y_test, y_pred))
```

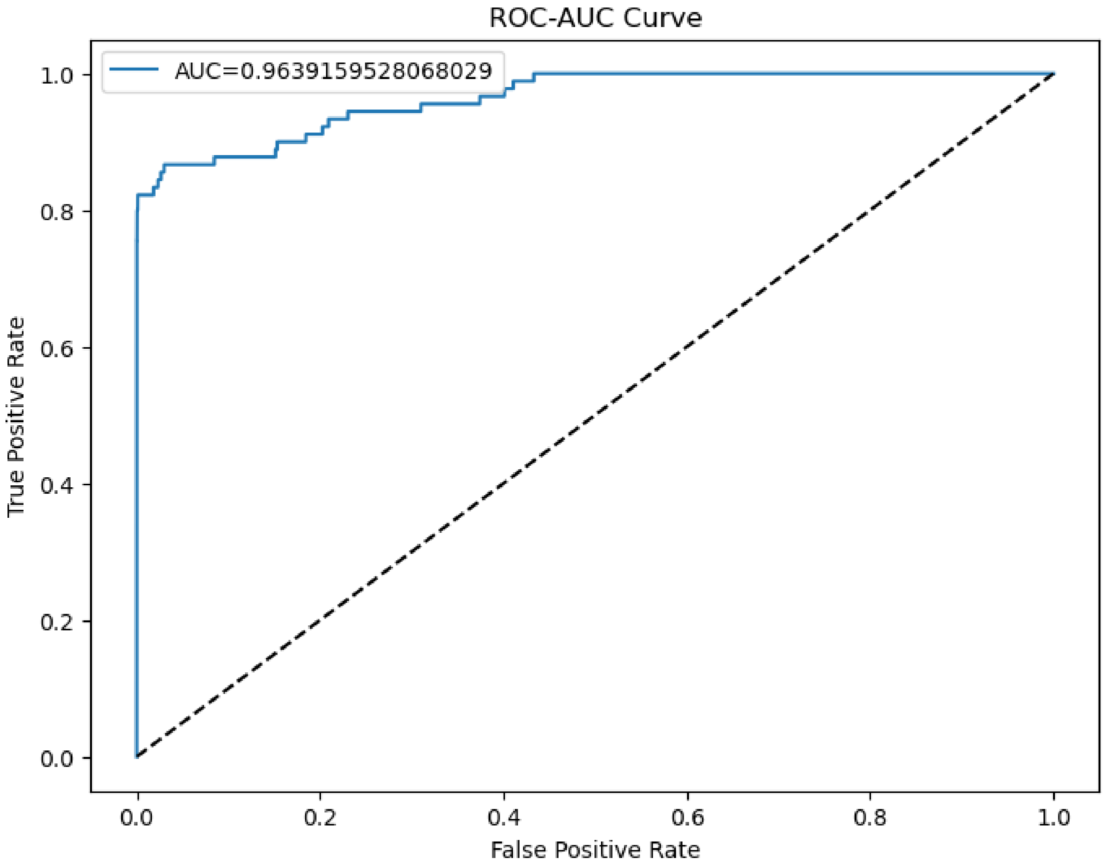
```
Accuracy: 0.9994889507630493
Classification Report:               precision    recall  f1-score   support

           0       1.00      1.00      1.00     56656
           1       0.94      0.72      0.82        90

    accuracy                           1.00     56746
   macro avg       0.97      0.86      0.91     56746
weighted avg       1.00      1.00      1.00     56746

Confusion Matrix: [[56652     4]
 [   25    65]]
```

[120]:
```
# Plot Confusion Matrix
plt.figure(figsize=(8,6))
sns.heatmap(confusion_matrix(y_test, model.predict(X_test)), annot=True,␣
 ↪cmap='Blues', fmt='g')
plt.xlabel("Predicted labels")
plt.ylabel("True labels")
plt.title('Confusion Matrix')
plt.show()
```

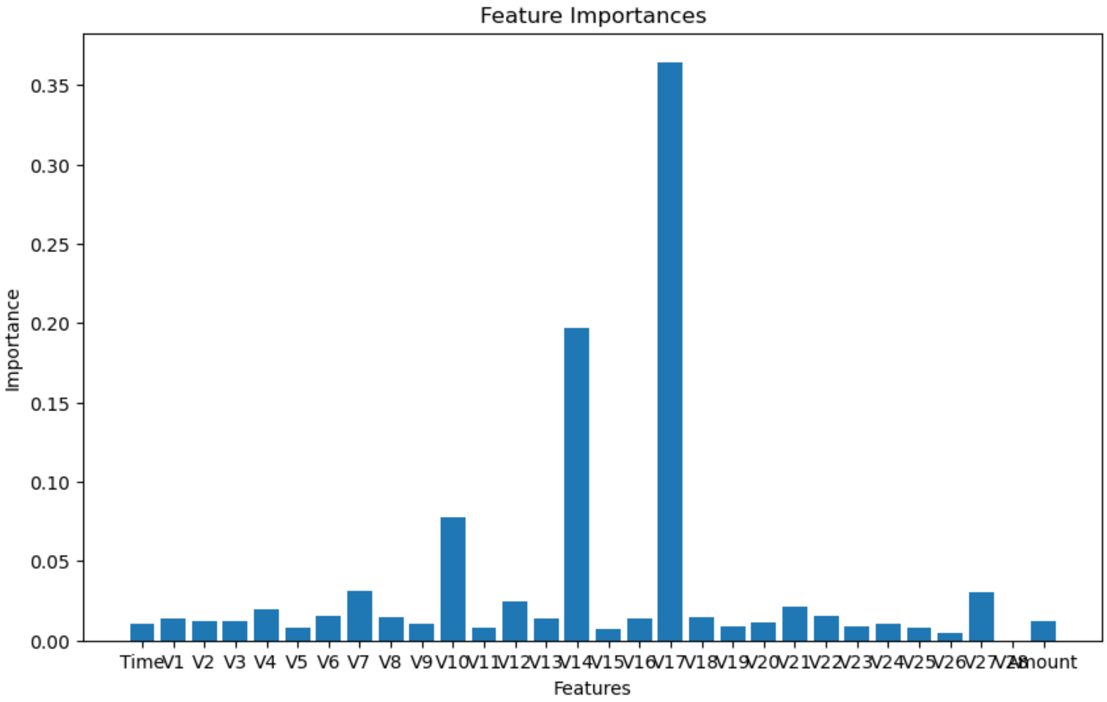


[124]:

```
#Plot ROC-AUC Curve
y_pred_proba = model.predict_proba(X_test)[:,1]
fpr, tpr, _ = roc_curve(y_test, y_pred_proba)
auc = roc_auc_score(y_test, y_pred_proba)
plt.figure(figsize=(8,6))
plt.plot(fpr,tpr,label="AUC="+str(auc))
plt.plot([0,1],[0,1],'k--')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title('ROC-AUC Curve')
plt.legend()
plt.show()
```



```
[119]: #Plot Feature Importances
feature_importances = model.feature_importances_
plt.figure(figsize=(10, 6))
plt.bar(X.columns, feature_importances)
plt.xlabel("Features")
plt.ylabel("Importance")
plt.title("Feature Importances")
plt.show()
```


Feature Importances
Importance
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
Features
Time
V1
V2
V3
V4
V5
V6
V7
V8
V9
V10
V11
V12
V13
V14
V15
V16
V17
V18
V19
V20
V21
V22
V23
V24
V25
V26
V27
V28
Amount