Untitled10

May 27, 2025

[1]:
```
# pandas
try:
    import pandas
    print("  pandas is installed")
except ImportError:
    print("  pandas is NOT installed")
```

```
 pandas is installed
```

[2]:
```
# numpy
try:
    import numpy
    print("  numpy is installed")
except ImportError:
    print("  numpy is NOT installed")
```

```
 numpy is installed
```

[3]:
```
# matplotlib
try:
    import matplotlib
    print("  matplotlib is installed")
except ImportError:
    print("  matplotlib is NOT installed")
```

```
 matplotlib is installed
```

[4]:
```
# seaborn
try:
    import seaborn
    print("  seaborn is installed")
except ImportError:
    print("  seaborn is NOT installed")
```

```
 seaborn is installed
```

[5]:
```
# scikit-learn (as sklearn)
try:
    import sklearn
    print("  scikit-learn is installed")
```

```
except ImportError:
    print("  scikit-learn is NOT installed")
```

```
 scikit-learn is installed
```

[6]:
```
# imbalanced-learn (SMOTE is from here)
try:
    import imblearn
    print("  imbalanced-learn is installed")
except ImportError:
    print("  imbalanced-learn is NOT installed")
```

```
 imbalanced-learn is installed
```

[8]:
```
# tensorflow (optional - only if using autoencoders)
try:
    import tensorflow
    print("  tensorflow is installed")
except ImportError:
    print("  tensorflow is NOT installed")
```

```
 tensorflow is installed
```

[9]:
```
import sys
print("  Python is installed")
print("Python version:", sys.version)
```

```
 Python is installed
Python version: 3.11.7 | packaged by Anaconda, Inc. | (main, Dec 15 2023,
18:05:47) [MSC v.1916 64 bit (AMD64)]
```

[11]:
```
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
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier, IsolationForest
from sklearn.metrics import classification_report, roc_auc_score,␣
 ↪confusion_matrix, f1_score
from imblearn.over_sampling import SMOTE
```

[12]:
```
df = pd.read_csv(r"C:
 ↪\Users\saipa\OneDrive\Desktop\satsM\codectechnologies\creditcard.csv")
print(df.shape)
df.head()
```

```
(284807, 31)
```

[12]:
```
   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]
```

[13]:
```python
print(df['Class'].value_counts())

scaler = StandardScaler()
df['scaled_amount'] = scaler.fit_transform(df[['Amount']])
df['scaled_time'] = scaler.fit_transform(df[['Time']])
df.drop(['Time', 'Amount'], axis=1, inplace=True)
scaled_df = df[['scaled_time', 'scaled_amount'] + [col for col in df.columns if␣
 ↪col not in ['scaled_time', 'scaled_amount']]]
```

```
Class
0    284315
1       492
Name: count, dtype: int64
```

[14]:
```python
X = scaled_df.drop('Class', axis=1)
y = scaled_df['Class']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)
```

[15]:
```python
sm = SMOTE(random_state=42)
X_res, y_res = sm.fit_resample(X_train, y_train)

print("Original dataset shape:", y_train.value_counts())
print("After SMOTE:", y_res.value_counts())
```

```
Original dataset shape: Class
0    227451
1       394
Name: count, dtype: int64
After SMOTE: Class
0    227451
1    227451
Name: count, dtype: int64
```

[ ]:
```
clf = RandomForestClassifier(random_state=42)
clf.fit(X_res, y_res)

y_pred = clf.predict(X_test)
y_proba = clf.predict_proba(X_test)[:, 1]

print("Classification Report:\n", classification_report(y_test, y_pred))
print("F1 Score:", f1_score(y_test, y_pred))
print("AUC-ROC:", roc_auc_score(y_test, y_proba))
```

[ ]:
```
iso_forest = IsolationForest(contamination=0.001, random_state=42)
iso_forest.fit(X_train)

y_pred_iso = iso_forest.predict(X_test)
y_pred_iso = [1 if x == -1 else 0 for x in y_pred_iso]  # Anomalies are labeled␣
 ↪-1

print("Isolation Forest Classification Report:\n",␣
 ↪classification_report(y_test, y_pred_iso))
```

[ ]:
```
```