1.DATA DOWLOAD AND LOADING

```
import pandas as pd
path = pd.read_csv("/content/creditcard.csv")
```

```
print("Path to dataset files:", path)
```

```
ath to dataset files:         Time        V1        V2        V3        V4        V5        V6
          0 -1.359807 -0.072781  2.536347  1.378155 -0.338321  0.462388
          0  1.191857  0.266151  0.166480  0.448154  0.060018 -0.082361
          1 -1.358354 -1.340163  1.773209  0.379780 -0.503198  1.800499
          1 -0.966272 -0.185226  1.792993 -0.863291 -0.010309  1.247203
          2 -1.158233  0.877737  1.548718  0.403034 -0.407193  0.095921
..      ...       ...       ...       ...       ...       ...       ...
7913  29027 -0.422159  0.231118  1.666711  0.451976 -0.203598  0.097244
7914  29030  1.177387 -0.215585  0.202972  0.215323 -0.029312  0.601788
7915  29030 -0.553746  0.880858  1.644821 -0.132657  0.120940 -0.267411
7916  29030 -2.844632  3.717960 -7.165428  4.120419 -2.991039 -2.942326
7917  29031  1.050204  0.078269  0.484733  1.349623       NaN       NaN

            V7        V8        V9  ...       V21       V22       V23  \
      0.239599  0.098698  0.363787  ... -0.018307  0.277838 -0.110474
     -0.078803  0.085102 -0.255425  ... -0.225775 -0.638672  0.101288
      0.791461  0.247676 -1.514654  ...  0.247998  0.771679  0.909412
      0.237609  0.377436 -1.387024  ... -0.108300  0.005274 -0.190321
      0.592941 -0.270533  0.817739  ... -0.009431  0.798278 -0.137458
..         ...       ...       ...  ...       ...       ...       ...
7913 -0.039666  0.354218  0.062463  ...  0.110909  0.435121 -0.056658
7914 -0.297021  0.188082  0.436370  ... -0.055842  0.075903 -0.187120
7915  0.466892  0.222443 -0.639624  ... -0.133339 -0.348662  0.029947
7916 -4.925187  2.204337 -2.663613  ...  0.894495 -0.340246  0.012222
7917       NaN       NaN       NaN  ...       NaN       NaN       NaN

           V24       V25       V26       V27       V28  Amount  Class
      0.066928  0.128539 -0.189115  0.133558 -0.021053  149.62    0.0
     -0.339846  0.167170  0.125895 -0.008983  0.014724    2.69    0.0
     -0.689281 -0.327642 -0.139097 -0.055353 -0.059752  378.66    0.0
     -1.175575  0.647376 -0.221929  0.062723  0.061458  123.50    0.0
      0.141267 -0.206010  0.502292  0.219422  0.215153   69.99    0.0
..         ...       ...       ...       ...       ...     ...    ...
7913  0.265867 -0.548204  0.734013  0.117023  0.130972    9.00    0.0
7914 -0.717798  0.555294  0.731531 -0.022112 -0.010929   25.00    0.0
7915  0.199962 -0.328384  0.071511  0.275487  0.110195    0.89    0.0
7916 -0.059679 -0.104338 -0.295884  1.326228  0.322688   89.99    0.0
7917       NaN       NaN       NaN       NaN       NaN     NaN    NaN

17918 rows x 31 columns]
```

```
path.describe()
```

	Time	V1	V2	V3	V4	V5	V6
count	17918.000000	17918.000000	17918.000000	17918.000000	17918.000000	17917.000000	17917.000000
mean	13905.276259	-0.244970	0.258166	0.777804	0.291614	-0.146329	0.099878
std	9867.916251	1.893161	1.508296	1.766872	1.479519	1.423917	1.327756
min	0.000000	-30.552380	-40.978852	-31.103685	-5.172595	-32.092129	-23.496714
25%	3781.250000	-0.959806	-0.305367	0.338327	-0.629972	-0.729796	-0.651820
50%	12347.500000	-0.306803	0.235061	0.924255	0.230058	-0.192681	-0.169764
75%	23775.000000	1.164015	0.876538	1.557391	1.155770	0.347812	0.493661
max	29031.000000	1.960497	16.713389	4.101716	11.927512	34.099309	21.393069

8 rows × 31 columns

```
print(path.head())
```

```
   Time        V1        V2        V3        V4        V5        V6        V7  \
0     0 -1.359807 -0.072781  2.536347  1.378155 -0.338321  0.462388  0.239599
1     0  1.191857  0.266151  0.166480  0.448154  0.060018 -0.082361 -0.078803
2     1 -1.358354 -1.340163  1.773209  0.379780 -0.503198  1.800499  0.791461
3     1 -0.966272 -0.185226  1.792993 -0.863291 -0.010309  1.247203  0.237609
4     2 -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    0.0
1  0.125895 -0.008983  0.014724    2.69    0.0
2 -0.139097 -0.055353 -0.059752  378.66    0.0
3 -0.221929  0.062723  0.061458  123.50    0.0
4  0.502292  0.219422  0.215153   69.99    0.0

[5 rows x 31 columns]
```

```
path.isnull().sum().max()
```

```
1
```

```
path.columns
```

```
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')
```

```
# The classes are heavily skewed we need to solve this issue later.
print('No Frauds', round(path['Class'].value_counts()[0]/len(path) * 100,2), '% of the dataset')
print('Frauds', round(path['Class'].value_counts()[1]/len(path) * 100,2), '% of the dataset')
```

```
No Frauds 99.54 % of the dataset
Frauds 0.45 % of the dataset
```

MODEL INTERPRETATION WITH SHAP OR LIME

1. LIME (Local Interpretable Model-Agnostic Explanations): For individual predictions, LIME perturbs the input data slightly and retrains simpler models (like linear regressions) to approximate the black-box model locally around that specific instance. Use LIME to explain a few individual predictions. For example, explain why a loan was approved or denied based on feature weights for that instance.
2. SHAP (SHapley Additive exPlanations): SHAP values are based on cooperative game theory and provide a way to distribute the “contribution” of each feature for individual predictions. SHAP can also provide global interpretability, showing the overall feature importance across the model. Use SHAP’s summary plots to show the importance of features and their impact on predictions. With SHAP, you can generate force plots for individual predictions, showing how each feature contributed to the final prediction. !pip install shap lime scikit-learn

```
!pip install shap lime scikit-learn
```

```
Requirement already satisfied: shap in /usr/local/lib/python3.10/dist-packages (0.46.0)
Collecting lime
  Downloading lime-0.2.0.1.tar.gz (275 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 275.7/275.7 kB 5.4 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (1.5.2)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from shap) (1
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (from shap) (1
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (from shap) (
Requirement already satisfied: tqdm>=4.27.0 in /usr/local/lib/python3.10/dist-packages (from sl
Requirement already satisfied: packaging>20.9 in /usr/local/lib/python3.10/dist-packages (from
Requirement already satisfied: slicer==0.0.8 in /usr/local/lib/python3.10/dist-packages (from
Requirement already satisfied: numba in /usr/local/lib/python3.10/dist-packages (from shap) (0
Requirement already satisfied: cloudpickle in /usr/local/lib/python3.10/dist-packages (from sh
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (from lim
Requirement already satisfied: scikit-image>=0.12 in /usr/local/lib/python3.10/dist-packages (
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.10/dist-packages (from
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: networkx>=2.8 in /usr/local/lib/python3.10/dist-packages (from
Requirement already satisfied: pillow>=9.1 in /usr/local/lib/python3.10/dist-packages (from sc
Requirement already satisfied: imageio>=2.33 in /usr/local/lib/python3.10/dist-packages (from
Requirement already satisfied: tifffile>=2022.8.12 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: lazy-loader>=0.4 in /usr/local/lib/python3.10/dist-packages (fr
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (fr
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from m
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (f
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (f
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (fr
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: llvmlite<0.44,>=0.43.0dev0 in /usr/local/lib/python3.10/dist-pa
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from p
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/dist-packages (from
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from pytho
Building wheels for collected packages: lime
  Building wheel for lime (setup.py) ... done
  Created wheel for lime: filename=lime-0.2.0.1-py3-none-any.whl size=283834 sha256=bf8976aad2
  Stored in directory: /root/.cache/pip/wheels/fd/a2/af/9ac0a1a85a27f314a06b39e1f492bee1547d52
Successfully built lime
Installing collected packages: lime
Successfully installed lime-0.2.0.1
```

2.Step-by-Step code for Data Preprocessing

1. Load the Data (from previous steps)
2. Handle Imbalance with SMOTE
3. Scale Features for Amount and Time

```
!pip install --upgrade scikit-learn==1.3.0 imbalanced-learn==0.10.1
```

```
Collecting scikit-learn==1.3.0
  Downloading scikit_learn-1.3.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.me
Collecting imbalanced-learn==0.10.1
  Downloading imbalanced_learn-0.10.1-py3-none-any.whl.metadata (8.2 kB)
Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.10/dist-packages (from
Requirement already satisfied: scipy>=1.5.0 in /usr/local/lib/python3.10/dist-packages (from s
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages
Downloading scikit_learn-1.3.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (10.
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 10.8/10.8 MB 78.0 MB/s eta 0:00:00
Downloading imbalanced_learn-0.10.1-py3-none-any.whl (226 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 226.0/226.0 kB 18.6 MB/s eta 0:00:00
Installing collected packages: scikit-learn, imbalanced-learn
  Attempting uninstall: scikit-learn
    Found existing installation: scikit-learn 1.5.2
    Uninstalling scikit-learn-1.5.2:
      Successfully uninstalled scikit-learn-1.5.2
  Attempting uninstall: imbalanced-learn
    Found existing installation: imbalanced-learn 0.12.4
    Uninstalling imbalanced-learn-0.12.4:
      Successfully uninstalled imbalanced-learn-0.12.4
ERROR: pip's dependency resolver does not currently take into account all the packages that ar
mlxtend 0.23.3 requires scikit-learn>=1.3.1, but you have scikit-learn 1.3.0 which is incompat
Successfully installed imbalanced-learn-0.10.1 scikit-learn-1.3.0
```

```
pip install --upgrade imbalanced-learn
```

```
Requirement already satisfied: imbalanced-learn in /usr/local/lib/python3.10/dist-packages (0.
Collecting imbalanced-learn
  Downloading imbalanced_learn-0.12.4-py3-none-any.whl.metadata (8.3 kB)
Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.10/dist-packages (from
Requirement already satisfied: scipy>=1.5.0 in /usr/local/lib/python3.10/dist-packages (from i
Requirement already satisfied: scikit-learn>=1.0.2 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages
Downloading imbalanced_learn-0.12.4-py3-none-any.whl (258 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 258.3/258.3 kB 4.1 MB/s eta 0:00:00
Installing collected packages: imbalanced-learn
  Attempting uninstall: imbalanced-learn
    Found existing installation: imbalanced-learn 0.10.1
    Uninstalling imbalanced-learn-0.10.1:
      Successfully uninstalled imbalanced-learn-0.10.1
Successfully installed imbalanced-learn-0.12.4
```

```
!pip install imblearn
```

```
Collecting imblearn
  Downloading imblearn-0.0-py2.py3-none-any.whl.metadata (355 bytes)
Requirement already satisfied: imbalanced-learn in /usr/local/lib/python3.10/dist-packages (fr
Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.10/dist-packages (from
Requirement already satisfied: scipy>=1.5.0 in /usr/local/lib/python3.10/dist-packages (from i
Requirement already satisfied: scikit-learn>=1.0.2 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages
Downloading imblearn-0.0-py2.py3-none-any.whl (1.9 kB)
Installing collected packages: imblearn
Successfully installed imblearn-0.0
```

```
from imblearn.over_sampling import SMOTE
```

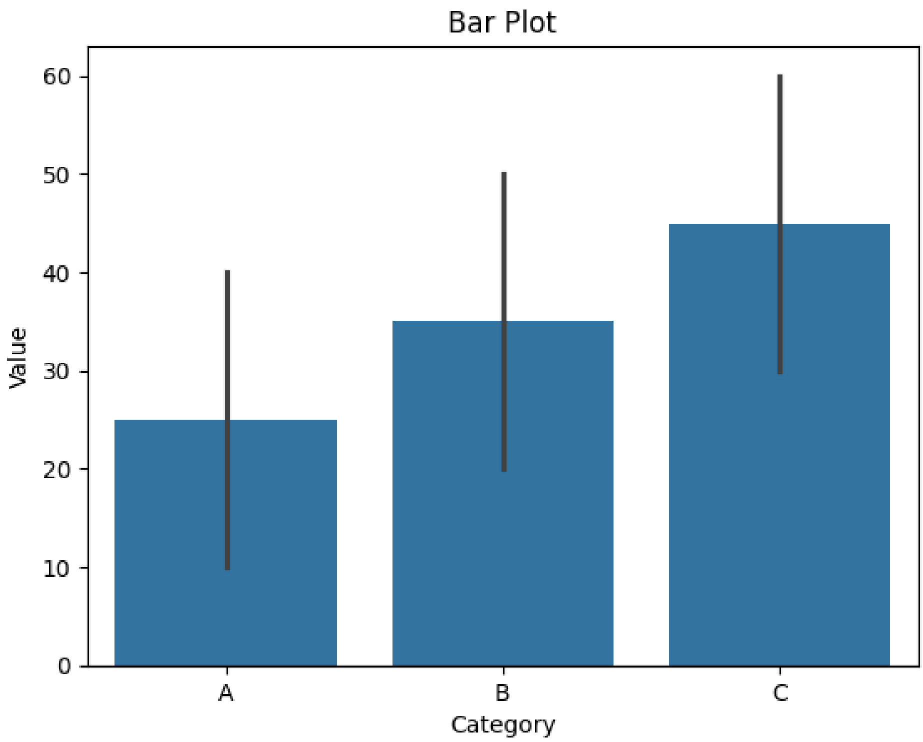
```
import sklearn
print(sklearn.__version__)
```

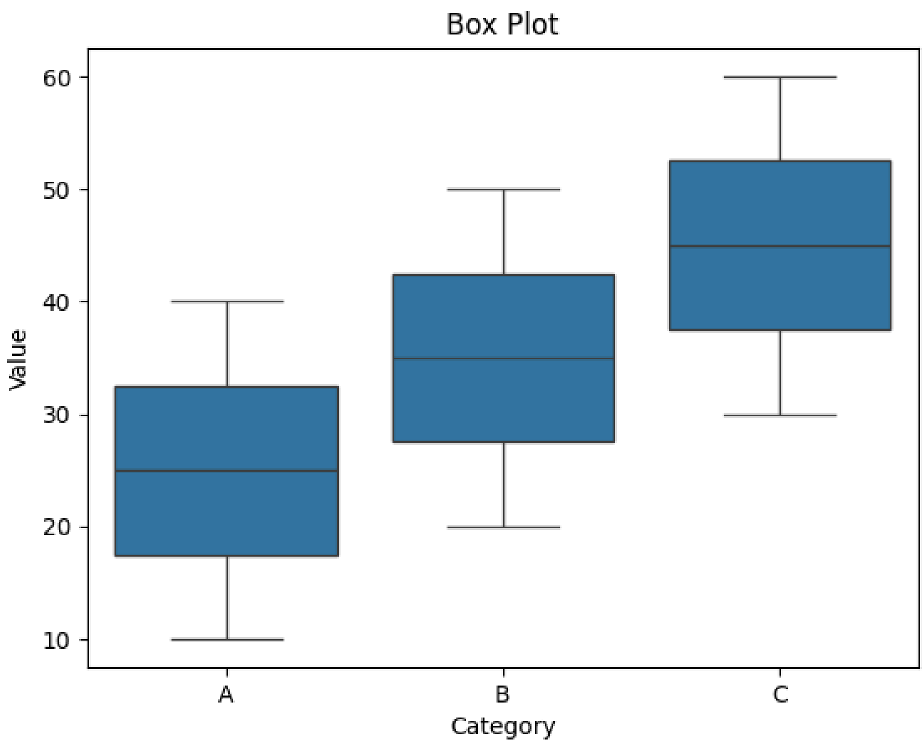
```
1.3.0
```

```
pip install imbalanced-learn --upgrade
```

```
Requirement already satisfied: imbalanced-learn in /usr/local/lib/python3.10/dist-packages (0.
Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.10/dist-packages (from
Requirement already satisfied: scipy>=1.5.0 in /usr/local/lib/python3.10/dist-packages (from i
Requirement already satisfied: scikit-learn>=1.0.2 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages
```

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Sample DataFrame
data = {'Category': ['A', 'B', 'C', 'A', 'B', 'C'],
        'Value': [10, 20, 30, 40, 50, 60]}
df = pd.DataFrame(data)

# Create a bar plot
sns.barplot(x='Category', y='Value', data=df)
plt.title('Bar Plot')
plt.xlabel('Category')
plt.ylabel('Value')
plt.show()

# Create a box plot
sns.boxplot(x='Category', y='Value', data=df)
plt.title('Box Plot')
plt.xlabel('Category')
plt.ylabel('Value')
plt.show()

# Create a scatter plot
sns.scatterplot(x='Category', y='Value', data=df, hue='Category')
plt.title('Scatter Plot')
plt.xlabel('Category')
plt.ylabel('Value')
plt.show()
```





2.1 Checking Class Imbalance

Check class distribution

```
# Check class distribution
print("Class distribution:\n", path['Class'].value_counts())
```

```
Class distribution:
 Class
0.0    17836
1.0       81
Name: count, dtype: int64
```

Category

2.2 Data Preprocessing

```
# Separate features and target variable
X = path.drop(columns=['Class'])
y = path['Class']

# Handle missing values in the target variable (y) before splitting
# Option 1: Drop rows with missing values in 'Class'
path = path.dropna(subset=['Class'])
X = path.drop(columns=['Class'])
y = path['Class']

# Option 2: Impute missing values (e.g., with the most frequent class) - Use with caution
# from sklearn.impute import SimpleImputer
# imputer = SimpleImputer(strategy='most_frequent')
# y = imputer.fit_transform(y.values.reshape(-1, 1))  # Reshape for SimpleImputer
# y = y.ravel() # Flatten back to 1D array

# Split the dataset into training and testing sets
from sklearn.model_selection import train_test_split # Importing the required module
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y, random_state=

# Standardize the features (training data only)
from sklearn.preprocessing import StandardScaler # Importing the required module
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
# Split the dataset into training and testing sets (after scaling)
X_test, y_test = X_test, y_test  # No need to scale the testing data


from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report # Import the classification_report function

# ... (other code, including data splitting and scaling)

# Create a logistic regression model with class weighting
model = LogisticRegression(class_weight='balanced')

# Train the model
model.fit(X_train_scaled, y_train)

# Make predictions
y_pred = model.predict(X_test_scaled)

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

```
              precision    recall  f1-score   support

         0.0       1.00      0.99      0.99      3568
         1.0       0.26      1.00      0.41        16

    accuracy                           0.99      3584
   macro avg       0.63      0.99      0.70      3584
weighted avg       1.00      0.99      0.99      3584

/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:460: ConvergenceWarn
```

```
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
from sklearn.preprocessing import StandardScaler

# ... (other code, including data splitting)

# Initialize the scaler
scaler = StandardScaler()

# Fit the scaler to the training data and transform both training and testing data
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

3.Feature Scaling

```
print("Processed training data shape:", X_train_scaled.shape)
print("Processed test data shape:", X_test_scaled.shape)
```

```
Processed training data shape: (14333, 30)
Processed test data shape: (3584, 30)
```

4. Training the RandomForestClassifier and Model Evaluation

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, average_precision_score
from tqdm import tqdm
import numpy as np

# Assuming you already have split your data into training and testing sets (X_train, X_test, y_tra

# Initialize the RandomForest model with warm_start to enable incremental training
model = RandomForestClassifier(n_estimators=1, warm_start=True, class_weight="balanced", random_st

# Set total estimators
total_estimators = 100

# Fit each tree incrementally with tqdm progress bar
for i in tqdm(range(1, total_estimators + 1), desc="Training Random Forest", unit="tree"):
    model.n_estimators = i  # Increment the number of trees
    model.fit(X_train, y_train)  # Use X_train since you're not using SMOTE

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

# Evaluate model performance
print("Classification Report:\n", classification_report(y_test, y_pred))
print("AUPRC:", average_precision_score(y_test, y_pred))
```

```
Training Random Forest:   0%|          | 0/100 [00:00<?, ?tree/s]/usr/local/lib/python3.10/dis
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:780: UserWarning: class_we
  warn(
Training Random Forest:   2%|▏         | 2/100 [00:00<00:07, 13.72tree/s]/usr/local/lib/python
  warn(
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:780: UserWarning: class_we
  warn(
Training Random Forest:   4%|▍         | 4/100 [00:00<00:06, 15.26tree/s]/usr/local/lib/python
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:780: UserWarning: class_we
  warn(
Training Random Forest:   6%|▌         | 6/100 [00:00<00:05, 16.02tree/s]/usr/local/lib/python
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:780: UserWarning: class_we
  warn(
Training Random Forest:   8%|▊         | 8/100 [00:00<00:05, 17.08tree/s]/usr/local/lib/python
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:780: UserWarning: class_we
  warn(
Training Random Forest:  10%|█         | 10/100 [00:00<00:05, 17.87tree/s]/usr/local/lib/pytho
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:780: UserWarning: class_we
  warn(
Training Random Forest:  12%|█▏        | 12/100 [00:00<00:05, 15.69tree/s]/usr/local/lib/pytho
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:780: UserWarning: class_we
  warn(
Training Random Forest:  14%|█▍        | 14/100 [00:00<00:05, 16.46tree/s]/usr/local/lib/pytho
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:780: UserWarning: class_we
  warn(
Training Random Forest:  16%|█▌        | 16/100 [00:00<00:05, 16.37tree/s]/usr/local/lib/pytho
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:780: UserWarning: class_we
  warn(
Training Random Forest:  18%|█▊        | 18/100 [00:01<00:05, 15.95tree/s]/usr/local/lib/pytho
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:780: UserWarning: class_we
  warn(
Training Random Forest:  20%|██        | 20/100 [00:01<00:04, 16.14tree/s]/usr/local/lib/pytho
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:780: UserWarning: class_we
  warn(
Training Random Forest:  22%|██▏       | 22/100 [00:01<00:05, 15.56tree/s]/usr/local/lib/pytho
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:780: UserWarning: class_we
  warn(
Training Random Forest:  24%|██▍       | 24/100 [00:01<00:04, 15.80tree/s]/usr/local/lib/pytho
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:780: UserWarning: class_we
  warn(
Training Random Forest:  26%|██▌       | 26/100 [00:01<00:04, 15.78tree/s]/usr/local/lib/pytho
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:780: UserWarning: class_we
  warn(
Training Random Forest:  28%|██▊       | 28/100 [00:01<00:04, 14.67tree/s]/usr/local/lib/pytho
  warn(
```

5. Improving the model

to improving model performance and adding interpretability to understand which features are driving predictions:

5.1 Hyperparameter Tuning with RandomizedSearchCV

Improve the model by tuning hyperparameters for the RandomForestClassifier or exploring other algorithms (e.g., XGBoost, LightGBM), which often perform well on structured, tabular data.

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, RandomizedSearchCV
from sklearn.preprocessing import StandardScaler
```

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, average_precision_score
from sklearn.utils.class_weight import compute_class_weight
from imblearn.over_sampling import SMOTE
from tqdm import tqdm
```

```
!pip install xgboost
```

```
Requirement already satisfied: xgboost in /usr/local/lib/python3.10/dist-packages (2.1.2)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from xgboost)
Requirement already satisfied: nvidia-nccl-cu12 in /usr/local/lib/python3.10/dist-packages (fr
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (from xgboost)
```

Hyperparameters Hyperparameters to be tuned for the RandomForestClassifier:

1. n_estimators: Number of trees in the forest.
2. max_depth: Maximum depth of each tree, controlling model complexity.
3. min_samples_split: Minimum samples required to split an internal node.
4. min_samples_leaf: Minimum samples required to be at a leaf node.

RandomForest Initialization Parameters:

1. class_weight="balanced": Adjusts weights inversely proportional to class frequencies, crucial for imbalanced datasets.
2. random_state=42: Sets a seed for reproducibility.
3. verbose=1: Outputs training information for monitoring.

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, RandomizedSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, average_precision_score
from tqdm import tqdm

# Since you're not using imblearn, remove the import:
# # from imblearn.over_sampling import SMOTE

# Define a simplified parameter grid for RandomForestClassifier
rf_param_grid = {
    'n_estimators': [50, 100],
    'max_depth': [10, 20, None],
    'min_samples_split': [2, 5],
    'min_samples_leaf': [1, 2],
}

# Initialize RandomForest model with verbose output
rf_model = RandomForestClassifier(
    class_weight="balanced", random_state=42, verbose=1
)

# Initialize RandomizedSearchCV with verbose output
rf_search = RandomizedSearchCV(
    estimator=rf_model,
    param_distributions=rf_param_grid,
    n_iter=10,
    cv=3,
    scoring='average_precision',
    n_jobs=-1,
    random_state=42,
    verbose=2
```

```
)

# Ensure you have XGBoost installed (if desired)
# If not installed, run: pip install xgboost
try:
  from xgboost import XGBClassifier

  # Define a parameter grid for XGBoost
  xgb_param_grid = {
      'n_estimators': [50, 100],
      'max_depth': [3, 6],
      'learning_rate': [0.01, 0.1],
      'subsample': [0.8, 1.0],
      'colsample_bytree': [0.8, 1.0]
  }

  # Initialize XGBoost model with verbosity
  xgb_model = XGBClassifier(
      random_state=42,
      eval_metric='logloss',
      verbosity=1  # Set verbosity to print progress
  )

  # Initialize RandomizedSearchCV with verbose output for XGBoost
  xgb_search = RandomizedSearchCV(
      estimator=xgb_model,
      param_distributions=xgb_param_grid,
      n_iter=10,
      cv=3,
      scoring='average_precision',
      n_jobs=-1,
      random_state=42,
      verbose=2
  )
except ModuleNotFoundError:
  print("XGBoost not found. Skipping XGBoost hyperparameter tuning.")

# Use original training data (X_train and y_train)
rf_search.fit(X_train, y_train)
print("Best RandomForest Parameters:", rf_search.best_params_)

# Use the best model
best_rf_model = rf_search.best_estimator_

# Make predictions on the test set with RandomForest
y_rf_pred = best_rf_model.predict(X_test)
print("RandomForest Classification Report:\n", classification_report(y_test, y_rf_pred))
print("RandomForest AUPRC:", average_precision_score(y_test, y_rf_pred))

if 'XGBClassifier' in globals():  # Check if XGBoost was imported
  xgb_search.fit(X_train, y_train)
  print("Best XGBoost Parameters:", xgb_search.best_params_)

  # Use the best model
  best_xgb_model = xgb_search.best_estimator_

  # Make predictions on the test set with XGBoost
  y_xgb_pred = best_xgb_model.predict(X_test)
  print("XGBoost Classification Report:\n", classification_report(y_test, y_xgb_pred))
  print("XGBoost AUPRC:", average_precision_score(y_test, y_xgb_pred))
```

```
Fitting 3 folds for each of 10 candidates, totalling 30 fits
[Parallel(n_jobs=1)]: Done  49 tasks      | elapsed:    2.1s
[Parallel(n_jobs=1)]: Done  49 tasks      | elapsed:    0.0s
Best RandomForest Parameters: {'n_estimators': 50, 'min_samples_split': 5, 'min_samples_leaf':
RandomForest Classification Report:
```

```
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00      3568
         1.0       0.88      0.94      0.91        16

    accuracy                           1.00      3584
   macro avg       0.94      0.97      0.95      3584
weighted avg       1.00      1.00      1.00      3584

RandomForest AUPRC: 0.827484900210084
Fitting 3 folds for each of 10 candidates, totalling 30 fits
Best XGBoost Parameters: {'subsample': 1.0, 'n_estimators': 100, 'max_depth': 6, 'learning_rat
XGBoost Classification Report:
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00      3568
         1.0       0.88      0.94      0.91        16

    accuracy                           1.00      3584
   macro avg       0.94      0.97      0.95      3584
weighted avg       1.00      1.00      1.00      3584

XGBoost AUPRC: 0.827484900210084
```

```
!pip install shap
```

```
Requirement already satisfied: shap in /usr/local/lib/python3.10/dist-packages (0.46.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from shap) (1
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (from shap) (1
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (from sl
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (from shap) (
Requirement already satisfied: tqdm>=4.27.0 in /usr/local/lib/python3.10/dist-packages (from sl
Requirement already satisfied: packaging>20.9 in /usr/local/lib/python3.10/dist-packages (from
Requirement already satisfied: slicer==0.0.8 in /usr/local/lib/python3.10/dist-packages (from
Requirement already satisfied: numba in /usr/local/lib/python3.10/dist-packages (from shap) (0
Requirement already satisfied: cloudpickle in /usr/local/lib/python3.10/dist-packages (from sha
Requirement already satisfied: llvmlite<0.44,>=0.43.0dev0 in /usr/local/lib/python3.10/dist-pa
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packag
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pa
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/dist-packages (from
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from pytho
```

```
import shap
```

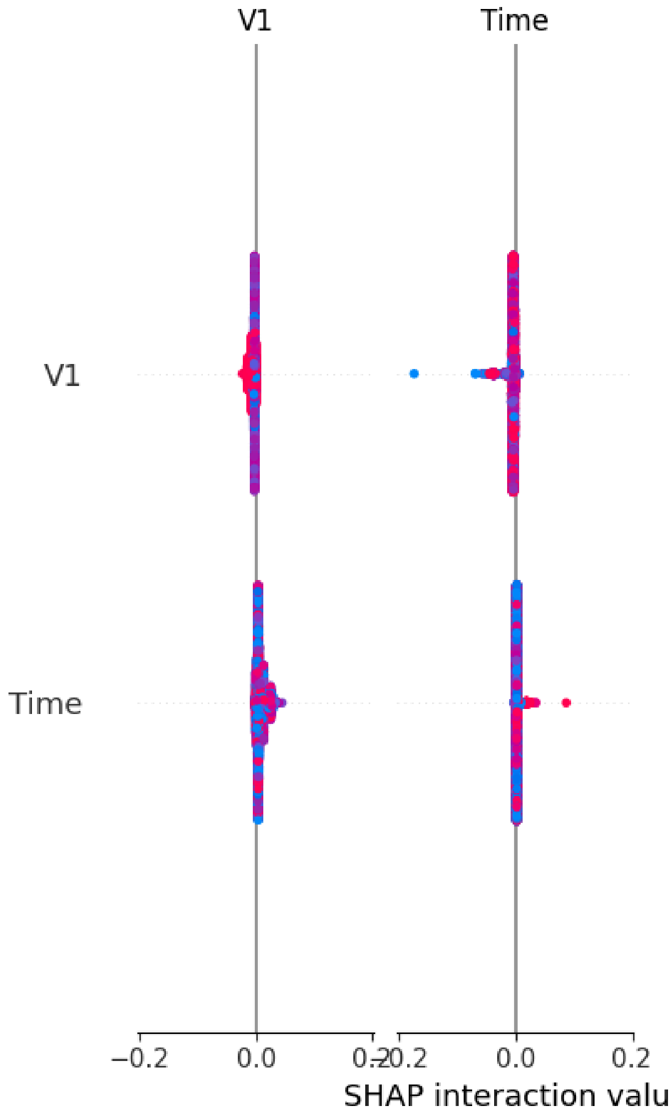
```
# Downgrade OpenCV to a stable version
!pip install opencv-python==4.5.5.64
!pip install opencv-python-headless==4.5.5.64  # for headless environments
```

```
Collecting opencv-python==4.5.5.64
  Downloading opencv_python-4.5.5.64-cp36-abi3-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.
Requirement already satisfied: numpy>=1.21.2 in /usr/local/lib/python3.10/dist-packages (from
Downloading opencv_python-4.5.5.64-cp36-abi3-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (6
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 60.5/60.5 MB 11.4 MB/s eta 0:00:00
Installing collected packages: opencv-python
  Attempting uninstall: opencv-python
    Found existing installation: opencv-python 4.10.0.84
    Uninstalling opencv-python-4.10.0.84:
      Successfully uninstalled opencv-python-4.10.0.84
Successfully installed opencv-python-4.5.5.64
Collecting opencv-python-headless==4.5.5.64
  Downloading opencv_python_headless-4.5.5.64-cp36-abi3-manylinux_2_17_x86_64.manylinux2014_x8
Requirement already satisfied: numpy>=1.21.2 in /usr/local/lib/python3.10/dist-packages (from
Downloading opencv_python_headless-4.5.5.64-cp36-abi3-manylinux_2_17_x86_64.manylinux2014_x86_
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 47.8/47.8 MB 15.8 MB/s eta 0:00:00
Installing collected packages: opencv-python-headless
  Attempting uninstall: opencv-python-headless
    Found existing installation: opencv-python-headless 4.10.0.84
    Uninstalling opencv-python-headless-4.10.0.84:
      Successfully uninstalled opencv-python-headless-4.10.0.84
ERROR: pip's dependency resolver does not currently take into account all the packages that ar
albucore 0.0.19 requires opencv-python-headless>=4.9.0.80, but you have opencv-python-headless
albumentations 1.4.20 requires opencv-python-headless>=4.9.0.80, but you have opencv-python-he
Successfully installed opencv-python-headless-4.5.5.64
```

```python
import shap
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

# Ensure X_test is a DataFrame
if isinstance(X_test, np.ndarray):
    X_test = pd.DataFrame(X_test, columns=X.columns)

# 1. Apply SHAP to RandomForest model
print("Applying SHAP to RandomForest Model")

# Initialize the SHAP explainer for the RandomForest model
rf_explainer = shap.TreeExplainer(best_rf_model)

# Calculate SHAP values for the test set
rf_shap_values = rf_explainer.shap_values(X_test)
# Select SHAP values for the positive class
if isinstance(rf_shap_values, list) and len(rf_shap_values) == 2:
    rf_shap_values = rf_shap_values[1]  # Take SHAP values for positive class
else:
    rf_shap_values = rf_shap_values  # Use directly if it's already for binary classification

# Check shapes
print("Shape of rf_shap_values:", rf_shap_values.shape)
print("Shape of X_test:", X_test.shape)

# 2. Plot SHAP summary for RandomForest
```

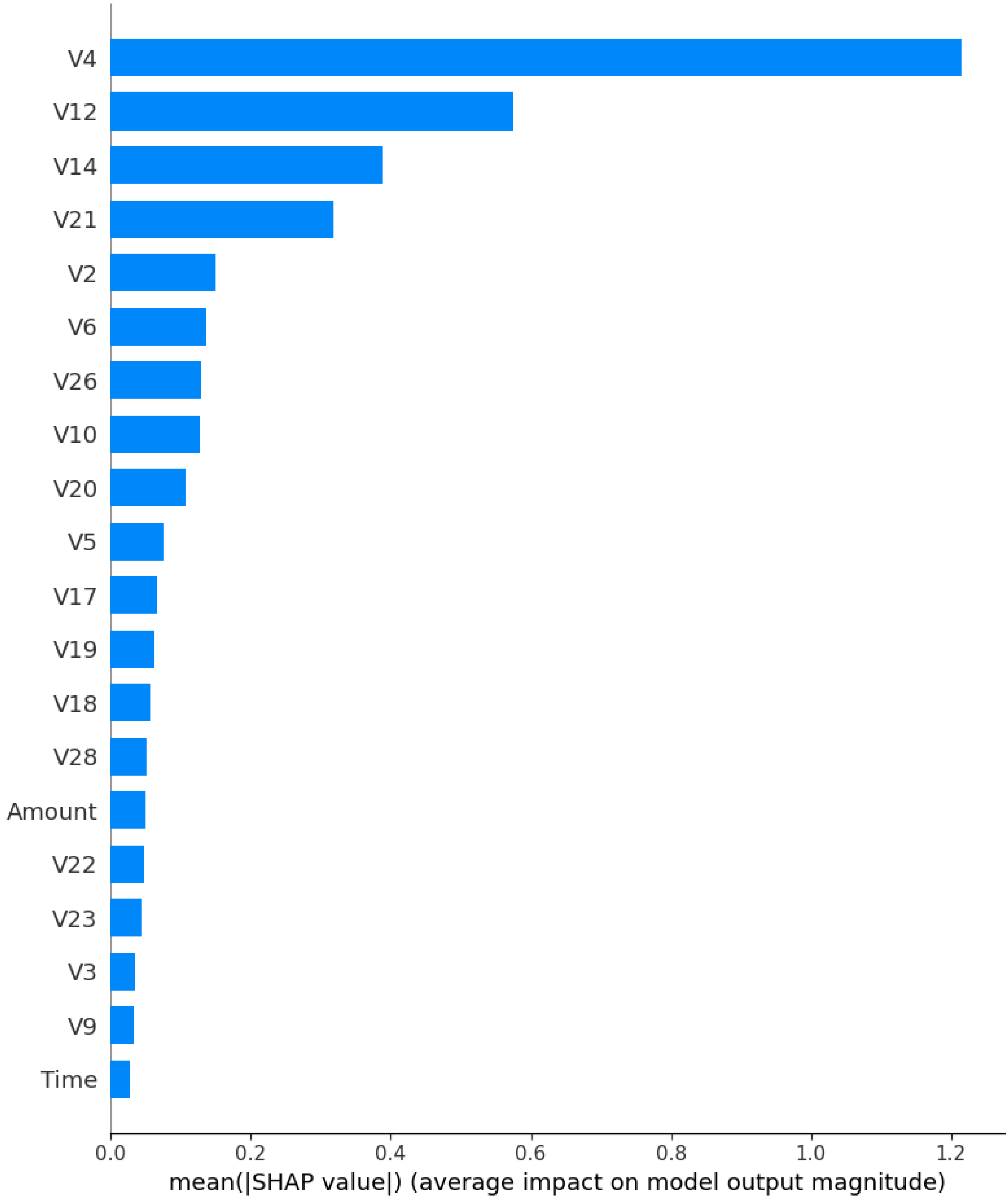
```
print("RandomForest SHAP Summary Plot")
shap.summary_plot(rf_shap_values, X_test, plot_type="bar")
```

```
Applying SHAP to RandomForest Model
Shape of rf_shap_values: (3584, 30, 2)
Shape of X_test: (3584, 30)
RandomForest SHAP Summary Plot
```



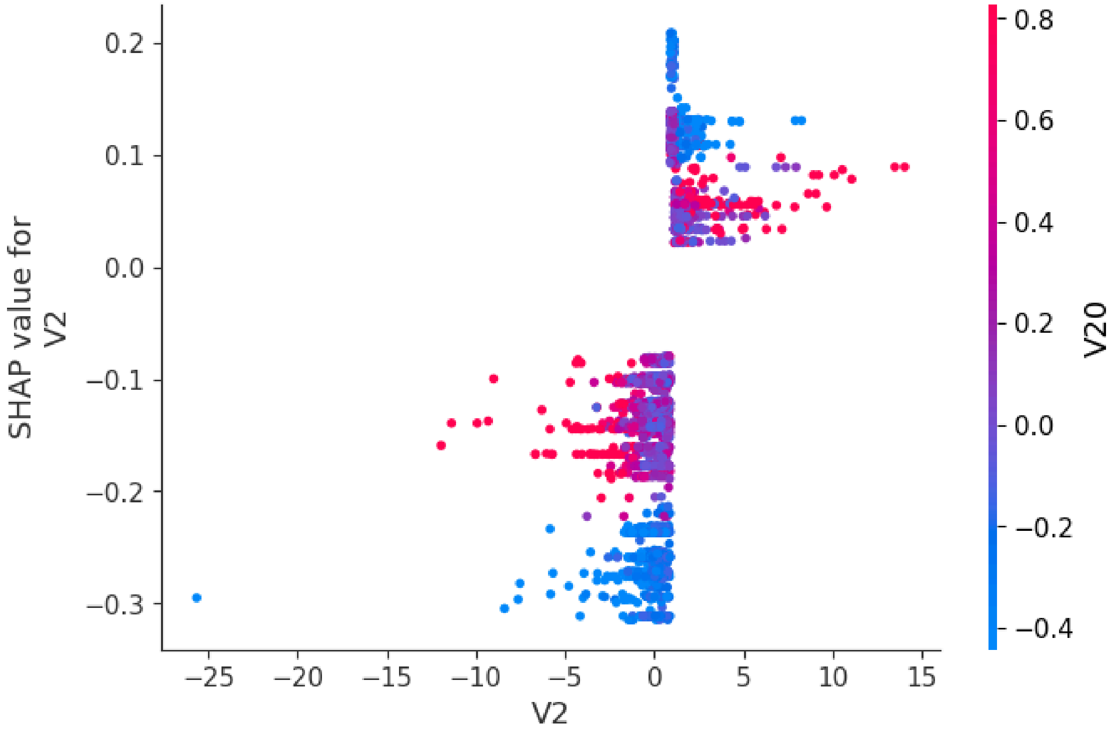

```
# 2. Apply SHAP to XGBoost model
print("\nApplying SHAP to XGBoost Model")

# Initialize the SHAP explainer for the XGBoost model
xgb_explainer = shap.TreeExplainer(best_xgb_model)

# Calculate SHAP values for the test set
xgb_shap_values = xgb_explainer.shap_values(X_test)

# Plot SHAP summary for XGBoost
print("XGBoost SHAP Summary Plot")
shap.summary_plot(xgb_shap_values, X_test, plot_type="bar")
```

```
Applying SHAP to XGBoost Model
XGBoost SHAP Summary Plot
```

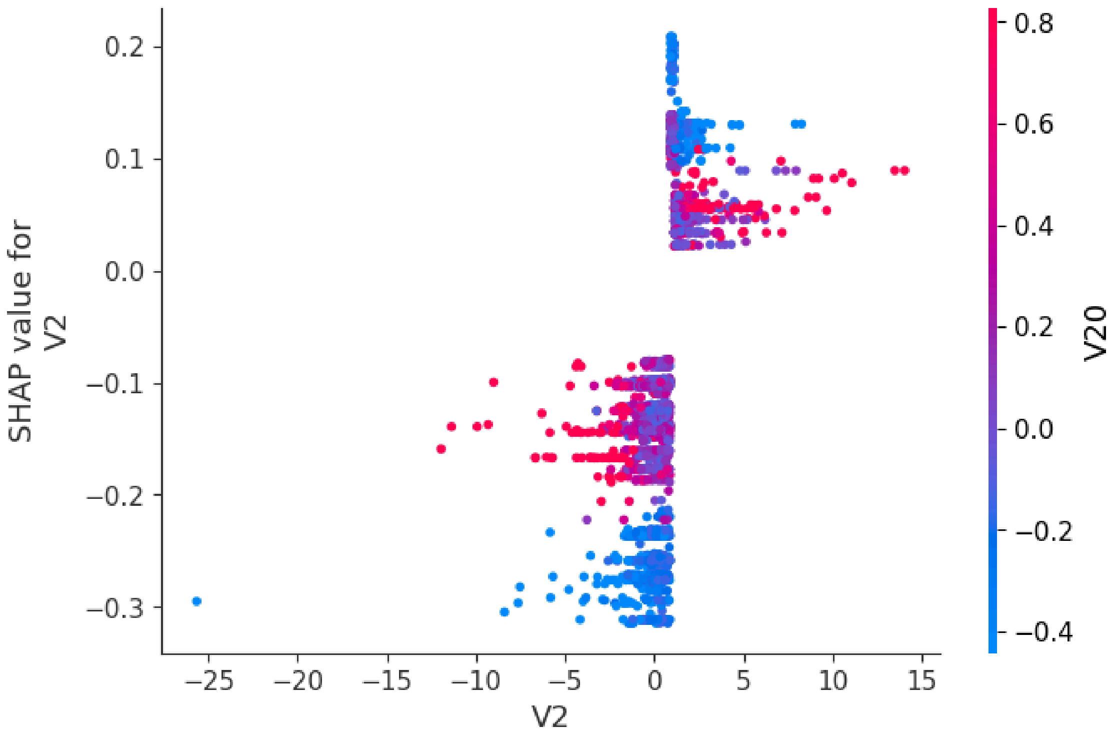



```
# This shows the relationship between the SHAP values and the feature values of V2.
print("XGBoost SHAP Dependence Plot for Feature 'V2'")
shap.dependence_plot("V2", xgb_shap_values, X_test)# This shows the relationship between the SHAP
print("XGBoost SHAP Dependence Plot for Feature 'V2'")
shap.dependence_plot("V2", xgb_shap_values, X_test)
```

XGBoost SHAP Dependence Plot for Feature 'V2'



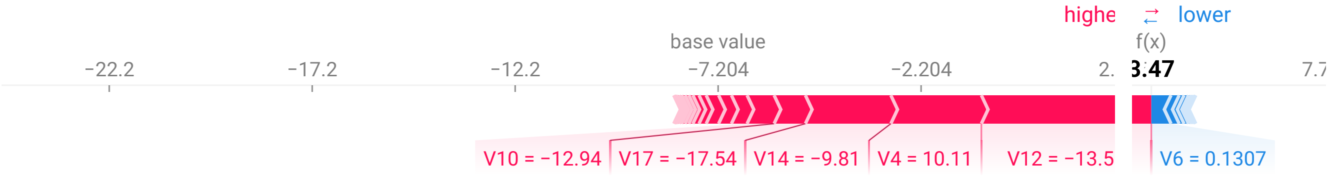
XGBoost SHAP Dependence Plot for Feature 'V2'

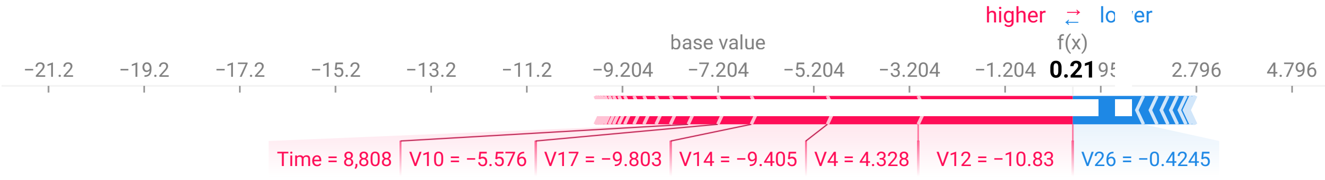



```
# Initialize JavaScript for SHAP plots
shap.initjs()

# Assuming 'y_pred' contains anomaly predictions (0 or 1) from the XGBoost model
# and 'y_test' contains the true labels:
y_pred = best_xgb_model.predict(X_test)  # Get predictions from the XGBoost model
# You may need to adjust the threshold (0.5 in this example)
anomalous_indices = X_test.index[y_pred == 1] # Get indices where the prediction is

# Generate force plots for the first 3 anomalies, if any
if len(anomalous_indices) > 0:
    print("XGBoost SHAP Force Plot for Anomalous Instances")
    for idx in anomalous_indices[:3]:
        position = X_test.index.get_loc(idx)
```

```
        # Generate force plot for each anomaly and display it
        display(shap.force_plot(
            xgb_explainer.expected_value,
            xgb_shap_values[position, :],
            X_test.iloc[position, :],
            feature_names=X_test.columns
        ))
else:
    print("No anomalies detected based on the XGBoost predictions.")
```

```
XGBoost SHAP Force Plot for Anomalous Instances
```






```
# Initialize JavaScript for SHAP plots
shap.initjs()

# SHAP Summary Plot with Beeswarm for identifying patterns in anomalies
print("\nXGBoost SHAP Summary Beeswarm Plot for Anomalous Predictions")

# Get the indices of anomalous predictions within the test set
anomalous_positions_in_test = [X_test.index.get_loc(i) for i in anomalous_indices if i in X_test.i

# Use the filtered indices to access SHAP values and data for the test set
shap.summary_plot(
```