credit

January 27, 2024

1 Credit Card Fraud Detection

Credit Card Fraud Detection is a major problem that financial world is facing now a days. Even greater problem is to detect the fraudsters and fraud transactions. To tackle this we need AI to automate the process, because we, a humans, its merely impossible to search the fraud transactions in a short period of time.

Now, we use Machine learning algorithms here to find the the number of fraud transactions. It's number is much lesser than the number of legitimate transaction for any bank. Most approaches involve building model on such imbalanced data, and thus fails to produce results on real-time new data because of overfitting on training data which is bias towards the class of legitimate transactions (major class). Thus, we can see this as an anomaly detection problem.

1.1 Here, I found out the answer to 2 of the major problems in anomaly.

Question.1: At what time does the Credit Card Frauds takes place usually?

Question.2: On legitimate transactions, how do we balance the data and dont let it overfit?

[2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import SGDClassifier

from mlxtend.plotting import plot_learning_curves
from sklearn.model_selection import train_test_split
from imblearn.over_sampling import SMOTE
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import make_scorer, matthews_corrcoef
```

```
from sklearn.metrics import precision_score, recall_score, f1_score,␣
 ↪roc_auc_score, accuracy_score, classification_report
from sklearn.model_selection import KFold, StratifiedKFold
```

[3]:
```
df = pd.read_csv('credit.csv')
df.head()
```

[3]:
```
   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]
```

2 About Data

The Data has 32 features which are unknown for Time, Amount and Class(From V1-V28).

Input features are V1-V28, Time and Amount

Target variable is Class

[4]:
```
df.shape
```

[4]:
```
(284807, 31)
```

[5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 284807 entries, 0 to 284806
Data columns (total 31 columns):
 #   Column  Non-Null Count   Dtype
```

```
---  ------  --------------   -----
 0   Time    284807 non-null  float64
 1   V1      284807 non-null  float64
 2   V2      284807 non-null  float64
 3   V3      284807 non-null  float64
 4   V4      284807 non-null  float64
 5   V5      284807 non-null  float64
 6   V6      284807 non-null  float64
 7   V7      284807 non-null  float64
 8   V8      284807 non-null  float64
 9   V9      284807 non-null  float64
 10  V10     284807 non-null  float64
 11  V11     284807 non-null  float64
 12  V12     284807 non-null  float64
 13  V13     284807 non-null  float64
 14  V14     284807 non-null  float64
 15  V15     284807 non-null  float64
 16  V16     284807 non-null  float64
 17  V17     284807 non-null  float64
 18  V18     284807 non-null  float64
 19  V19     284807 non-null  float64
 20  V20     284807 non-null  float64
 21  V21     284807 non-null  float64
 22  V22     284807 non-null  float64
 23  V23     284807 non-null  float64
 24  V24     284807 non-null  float64
 25  V25     284807 non-null  float64
 26  V26     284807 non-null  float64
 27  V27     284807 non-null  float64
 28  V28     284807 non-null  float64
 29  Amount  284807 non-null  float64
 30  Class   284807 non-null  int64
dtypes: float64(30), int64(1)
memory usage: 67.4 MB
```

[6]:
```python
# Check for missing values
print(df.isnull().sum())
```

```
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
```

3 About Data

The Data doesn't have any missing values, so, no need not be handled. The Data has all numerical features except Target Variable Class which is a categorical feature.

Class 0: Legitimate Transaction Class 1: Fraud Transaction

[7]:
```
df.describe()
```

[7]:
```
                Time            V1            V2            V3            V4  \
count  284807.000000  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05
mean    94813.859575  1.759061e-12 -8.251130e-13 -9.654937e-13  8.321385e-13
std     47488.145955  1.958696e+00  1.651309e+00  1.516255e+00  1.415869e+00
min         0.000000 -5.640751e+01 -7.271573e+01 -4.832559e+01 -5.683171e+00
25%     54201.500000 -9.203734e-01 -5.985499e-01 -8.903648e-01 -8.486401e-01
50%     84692.000000  1.810880e-02  6.548556e-02  1.798463e-01 -1.984653e-02
75%    139320.500000  1.315642e+00  8.037239e-01  1.027196e+00  7.433413e-01
max    172792.000000  2.454930e+00  2.205773e+01  9.382558e+00  1.687534e+01

                 V5            V6            V7            V8            V9  \
count  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05
mean   1.649999e-13  4.248366e-13 -3.054600e-13  8.777971e-14 -1.179749e-12
std    1.380247e+00  1.332271e+00  1.237094e+00  1.194353e+00  1.098632e+00
min   -1.137433e+02 -2.616051e+01 -4.355724e+01 -7.321672e+01 -1.343407e+01
25%   -6.915971e-01 -7.682956e-01 -5.540759e-01 -2.086297e-01 -6.430976e-01
```

```
50%   -5.433583e-02 -2.741871e-01  4.010308e-02  2.235804e-02 -5.142873e-02
75%    6.119264e-01  3.985649e-01  5.704361e-01  3.273459e-01  5.971390e-01
max    3.480167e+01  7.330163e+01  1.205895e+02  2.000721e+01  1.559499e+01

       …            V21           V22           V23           V24  \
count  …   2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05
mean   …  -3.405756e-13 -5.723197e-13 -9.725856e-13  1.464150e-12
std    …   7.345240e-01  7.257016e-01  6.244603e-01  6.056471e-01
min    …  -3.483038e+01 -1.093314e+01 -4.480774e+01 -2.836627e+00
25%    …  -2.283949e-01 -5.423504e-01 -1.618463e-01 -3.545861e-01
50%    …  -2.945017e-02  6.781943e-03 -1.119293e-02  4.097606e-02
75%    …   1.863772e-01  5.285536e-01  1.476421e-01  4.395266e-01
max    …   2.720284e+01  1.050309e+01  2.252841e+01  4.584549e+00

                V25           V26           V27           V28         Amount  \
count  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05  284807.000000
mean  -6.987102e-13 -5.617874e-13  3.332082e-12 -3.518874e-12      88.349619
std    5.212781e-01  4.822270e-01  4.036325e-01  3.300833e-01     250.120109
min   -1.029540e+01 -2.604551e+00 -2.256568e+01 -1.543008e+01       0.000000
25%   -3.171451e-01 -3.269839e-01 -7.083953e-02 -5.295979e-02       5.600000
50%    1.659350e-02 -5.213911e-02  1.342146e-03  1.124383e-02      22.000000
75%    3.507156e-01  2.409522e-01  9.104512e-02  7.827995e-02      77.165000
max    7.519589e+00  3.517346e+00  3.161220e+01  3.384781e+01   25691.160000

               Class
count  284807.000000
mean        0.001727
std         0.041527
min         0.000000
25%         0.000000
50%         0.000000
75%         0.000000
max         1.000000

[8 rows x 31 columns]
```

4 Data Declaration

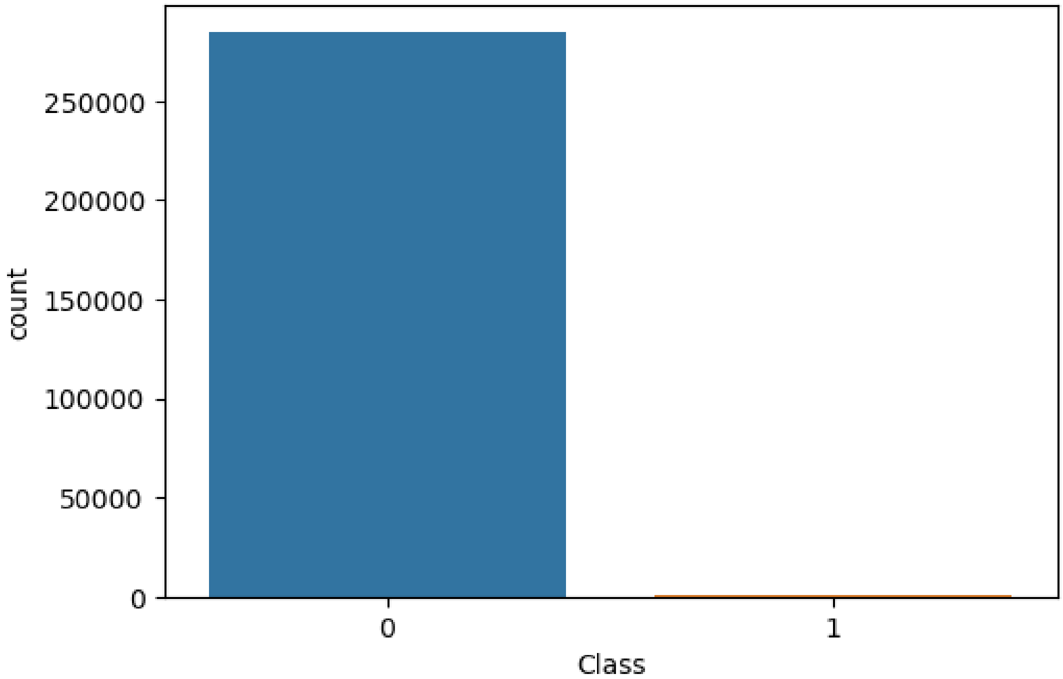
- Mean and StdDev of Amount of Data is shown above.
- In this case, I will not delete or drop any data.

[13]:
```python
def countplot_data(data, feature):
    plt.figure(figsize=(6,4))
    sns.countplot(x=feature, data=data)
    plt.show()

def pairplot_data_grid(data, feature1, feature2, target):
```

```
    sns.FacetGrid(data, hue=target, size=6).map(plt.scatter, feature1,␣
↪feature2).add_legend()
    plt.show()
```

[14]:
```
countplot_data(df, df.Class)
```



5 Insights:

- The target variable is Class and rest are are input features.
- 0 = Legitimate Transactions
- 1 = Fraud Transactions

As we can see Dataset is highly imbalanced Major class label = 0 and minor class label 1. Now, we will perform Synthetic Minority Oversampling on the data to balance it out

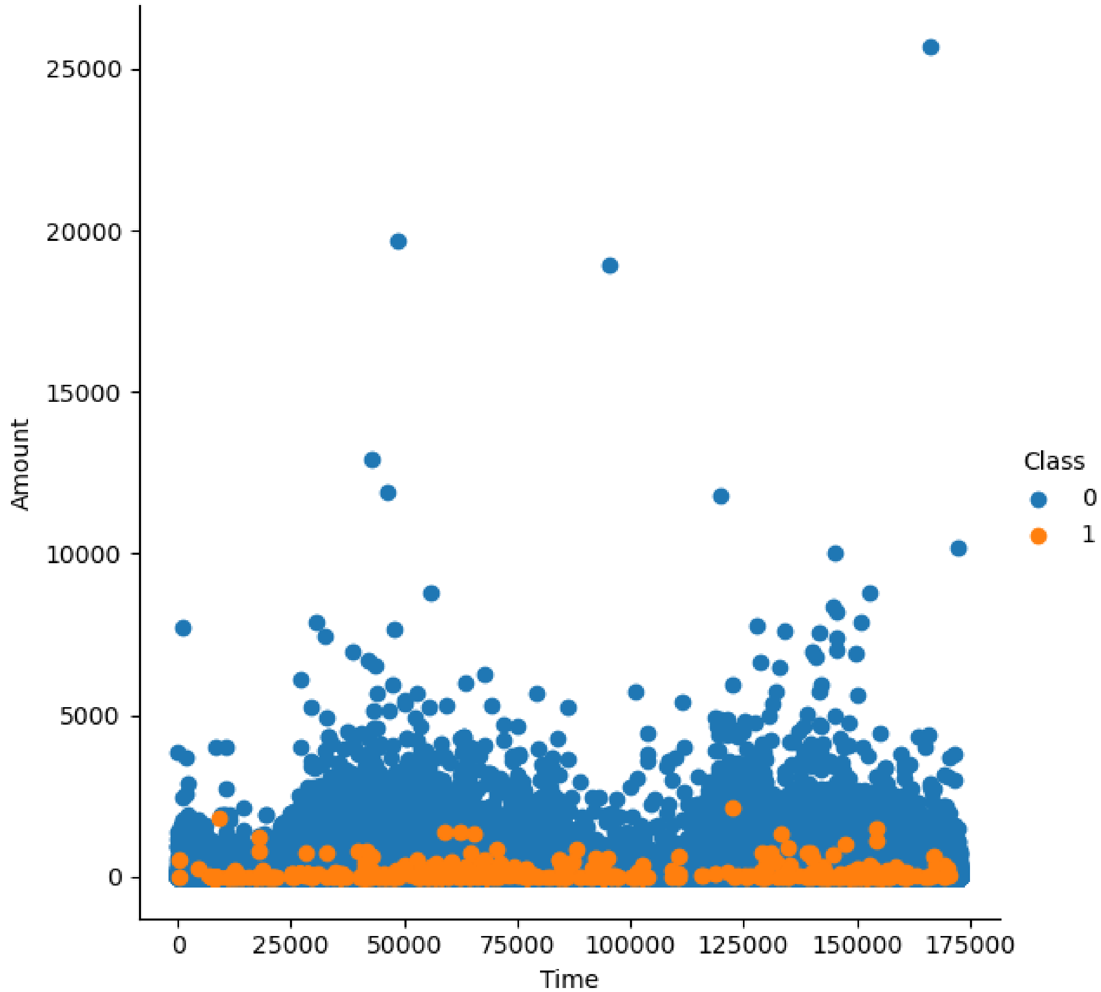
What is relationship of fraud transactions with amount of money? Let us try to determine the nature of transactions which are fraud and obtain a relevant set of the same with respect to their amount.

Results: All fraud transactions occur for an amount less than 2500.

[15]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

```
def pairplot_data_grid(data, feature1, feature2, target):
    sns.FacetGrid(data, hue=target, height=6).map(plt.scatter, feature1,␣
 ↪feature2).add_legend()
    plt.show()

# Example usage
pairplot_data_grid(df, "Time", "Amount", "Class")
```



5.1 Insights:

- The fraud transactions are evenly distributed about time.
- The fraud transactions are generally not above an amount of 2500.

[16]:
```
df_refine = pd.DataFrame(df)
amount_more = 0
amount_less = 0
```

```
for i in range(df_refine.shape[0]):
    if(df_refine.iloc[i]["Amount"] < 2500):
        amount_less += 1
    else:
        amount_more += 1
print(amount_more)
print(amount_less)
```

```
449
284358
```

[17]:
```
percentage_less = (amount_less/df.shape[0])*100
percentage_less
```

[17]: 99.84234938045763

Hence, we observe that the 99.85% of transactions are amount to less than 2500. Let us see how many of these are fraud and others legitimate.

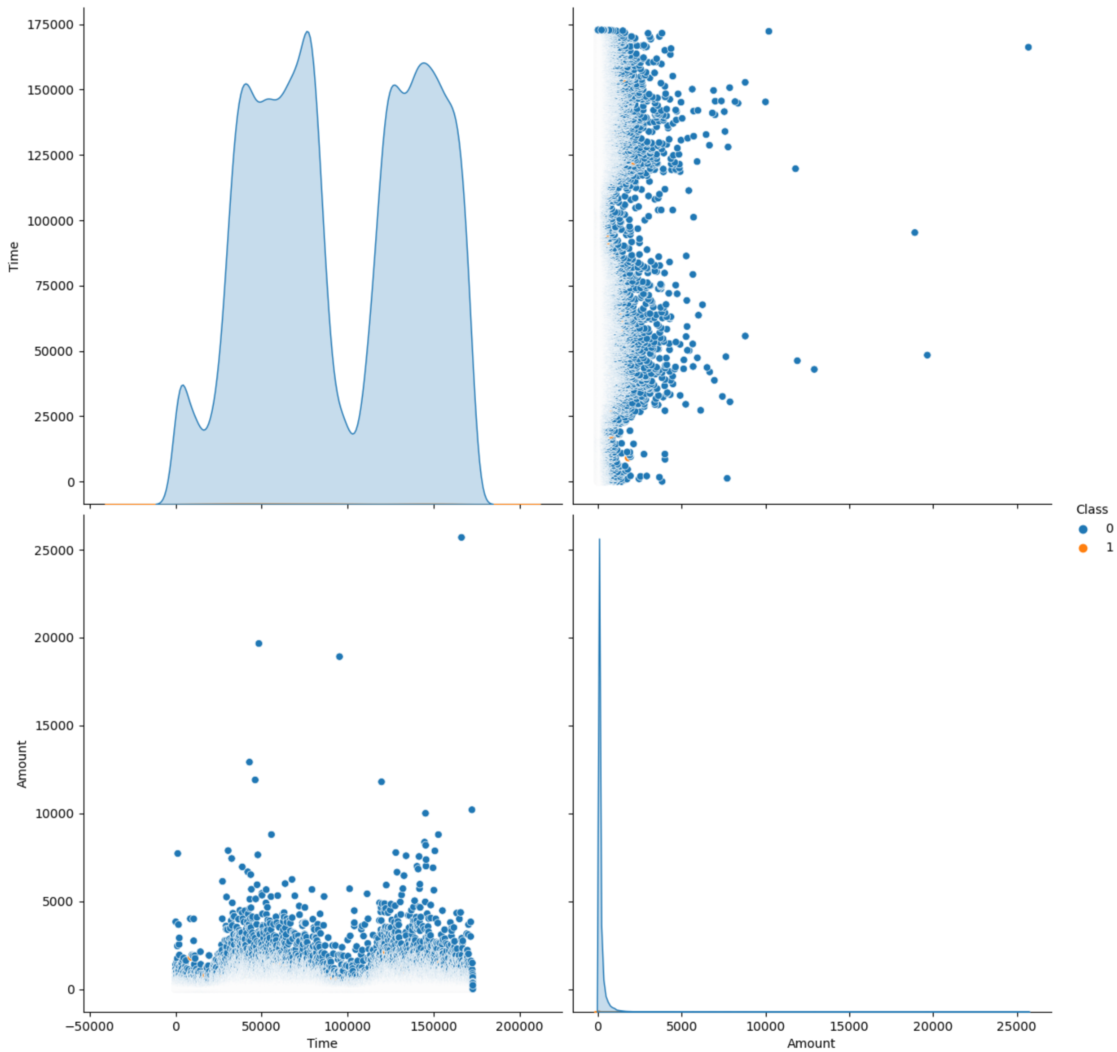
[18]:
```
fraud = 0
legitimate = 1
for i in range(df_refine.shape[0]):
    if(df_refine.iloc[i]["Amount"]<2500):
        if(df_refine.iloc[i]["Class"] == 0):
            legitimate += 1
        else:
            fraud+=1
print(fraud)
print(legitimate)
```

```
492
283867
```

[19]:
```
df_refine = df[["Time", "Amount", "Class"]]
sns.pairplot(df_refine, hue= "Class", size=6)
plt.show()
```

```
C:\Users\ADhiRAJ\Downloads\Music\Lib\site-packages\seaborn\axisgrid.py:2095:
UserWarning: The `size` parameter has been renamed to `height`; please update
your code.
  warnings.warn(msg, UserWarning)
C:\Users\ADhiRAJ\Downloads\Music\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed in a
future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\ADhiRAJ\Downloads\Music\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed in a
future version. Convert inf values to NaN before operating instead.
```

```
with pd.option_context('mode.use_inf_as_na', True):
```



[20]:
```
sns.FacetGrid(df_refine, hue="Class").map(sns.distplot,"Time").add_legend()
plt.show()
```

```
C:\Users\ADhiRAJ\Downloads\Music\Lib\site-packages\seaborn\axisgrid.py:848:
UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751
```

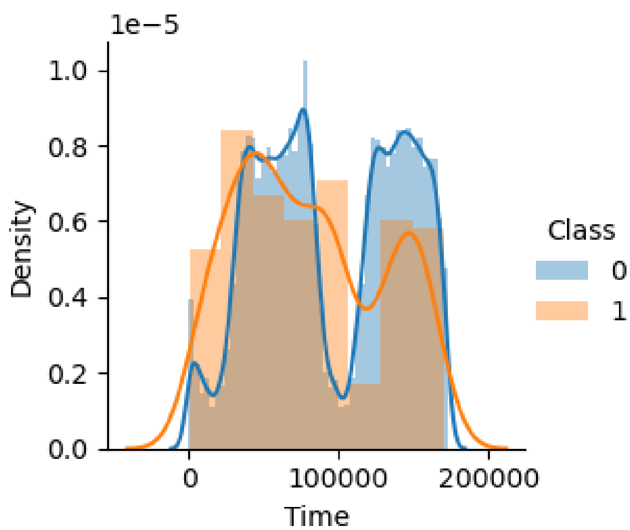
```
  func(*plot_args, **plot_kwargs)
C:\Users\ADhiRAJ\Downloads\Music\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed in a
future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\ADhiRAJ\Downloads\Music\Lib\site-packages\seaborn\axisgrid.py:848:
UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  func(*plot_args, **plot_kwargs)
C:\Users\ADhiRAJ\Downloads\Music\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed in a
future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



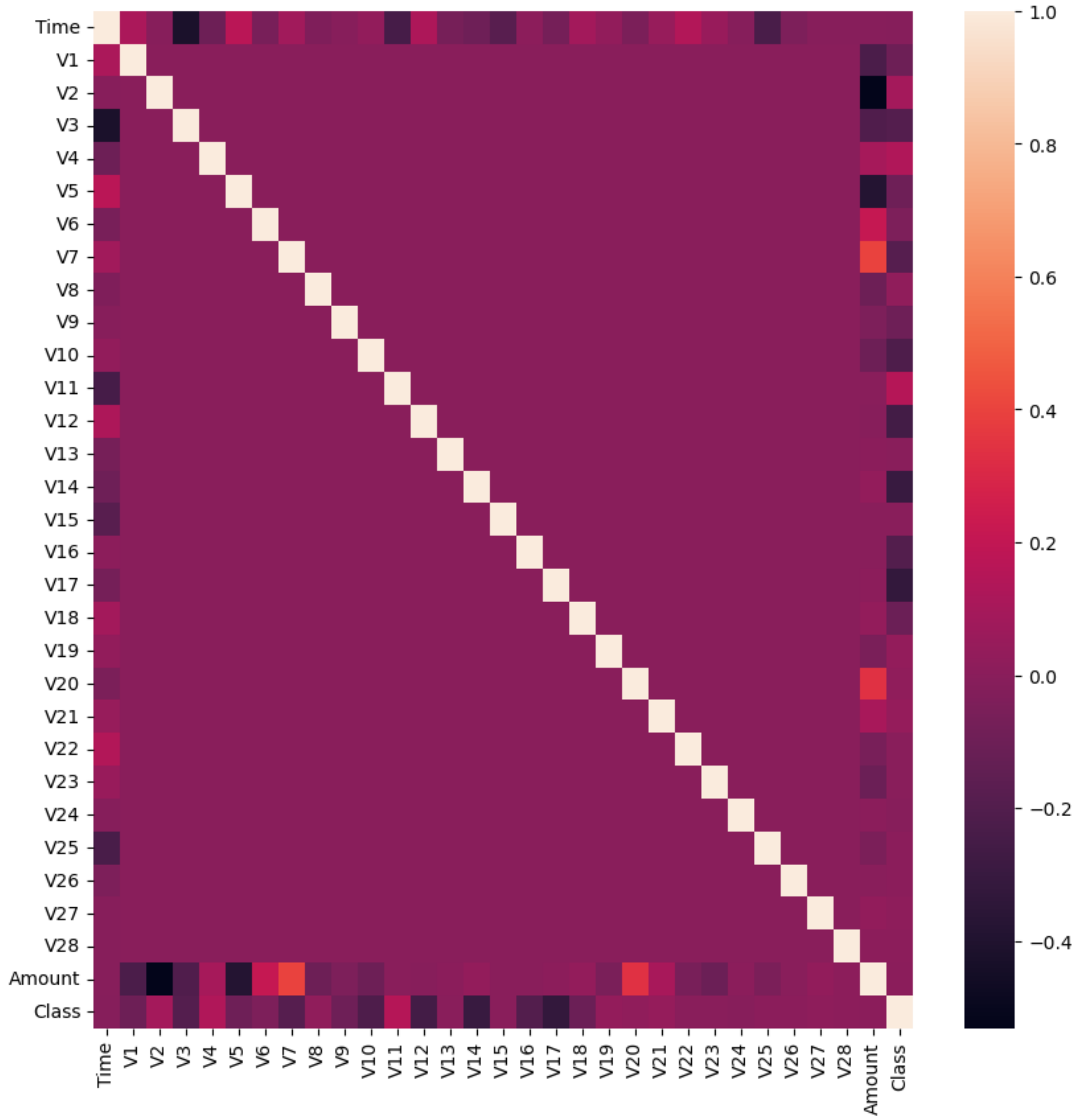
6 From the above distribution plot, it is clear that the fraudulent transactions are spread throughout the time period

- Modelling
- Study the Feature Correlations of the given data

- Plot a Heatmap
- Run GridSearch on the Data
- Fine Tune the Classifiers
- Create Pipelines for evaluation

[21]:
```
plt.figure(figsize=(10,10))
df_corr = df.corr()
sns.heatmap(df_corr)
```

[21]: <Axes: >



[22]:
```python
# Create Train and Test Data in ratio 70:30
X = df.drop(labels='Class', axis=1) # Features
y = df.loc[:,'Class']               # Target Variable


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
 ↪random_state=1, stratify=y)
```

[23]:
```python
# Use Synthetic Minority Oversampling
sm = SMOTE(random_state=42)
X_res, y_res = sm.fit_resample(X_train, y_train)
```

[24]:
```python
from sklearn.feature_selection import mutual_info_classif
mutual_infos = pd.Series(data=mutual_info_classif(X_res, y_res,␣
 ↪discrete_features=False, random_state=1), index=X_train.columns)
```

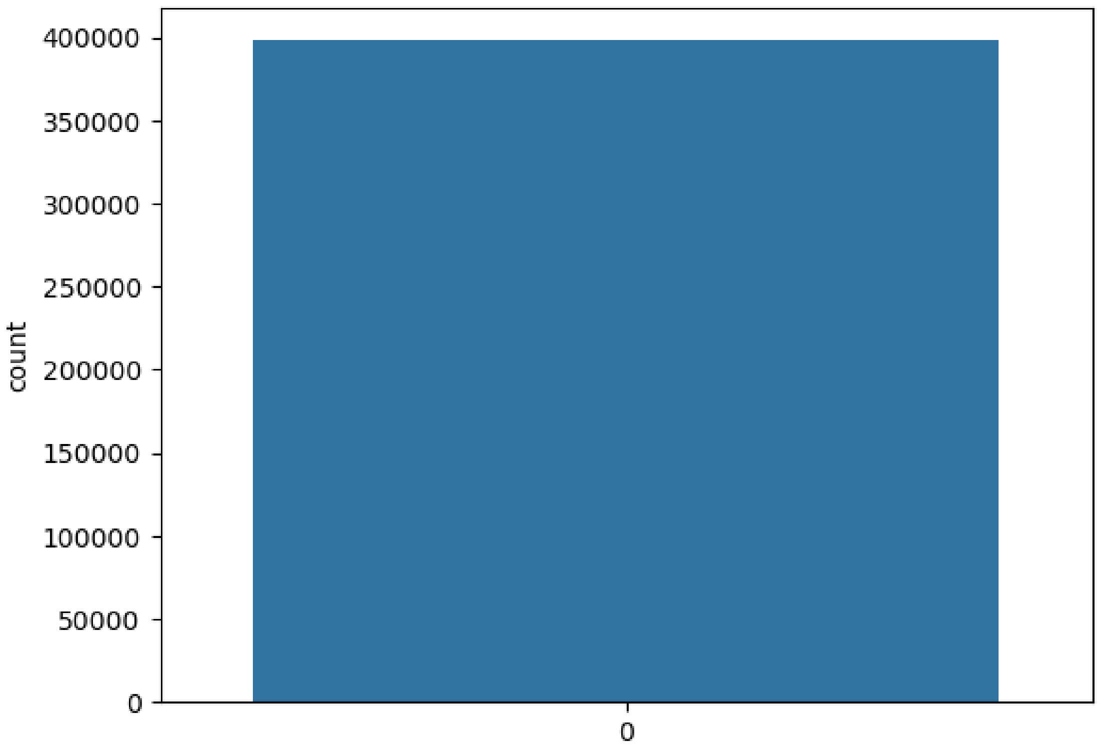
[25]:
```python
mutual_infos.sort_values(ascending=False)
```

[25]:
```
V14       0.535044
V10       0.464777
V12       0.456053
V17       0.438194
V4        0.427426
V11       0.404040
Amount    0.392909
V3        0.387187
V16       0.335316
V7        0.304169
V2        0.291487
V9        0.256678
Time      0.247989
V21       0.235011
V27       0.229915
V1        0.220733
V18       0.198264
V8        0.174371
V6        0.171969
V28       0.170488
V5        0.157363
V20       0.107487
V19       0.099838
V23       0.067337
V24       0.063567
V26       0.046977
V25       0.031609
V22       0.031540
V13       0.024934
```

```
V15       0.022443
dtype: float64
```

[30]:
```
sns.countplot(y_res)
```

[30]: <Axes: ylabel='count'>



Hence, we can say that the most correlated features after resolving class imbalance using Synthetic Minority Oversampling are V14, V10, V4, V12 and V17.

7 Evaluation

We make use of AUC-ROC Score, Classification Report, Accuracy and F1-Score to evaluate the performance of the classifiers

Method to compute the following:

1. Classification Report
2. F1-score
3. AUC-ROC score
4. Accuracy ##### Parameters:

- y_test: The target variable test set
- grid_clf: Grid classifier selected

- X_test: Input Feature Test Set

[31]:
```
# Evaluation of Classifiers
def grid_eval(grid_clf):
    print("Best Score", grid_clf.best_score_)
    print("Best Parameter", grid_clf.best_params_)

def evaluation(y_test, grid_clf, X_test):
    y_pred = grid_clf.predict(X_test)
    print('CLASSIFICATION REPORT')
    print(classification_report(y_test, y_pred))

    print('AUC-ROC')
    print(roc_auc_score(y_test, y_pred))

    print('F1-Score')
    print(f1_score(y_test, y_pred))

    print('Accuracy')
    print(accuracy_score(y_test, y_pred))
```

[32]:
```
# The parameters of each classifier are different
# Hence, we do not make use of a single method and this is not to violate DRY␣
↪Principles
# We set pipelines for each classifier unique with parameters
param_grid_sgd = [{
    'model__loss': ['log'],
    'model__penalty': ['l1', 'l2'],
    'model__alpha': np.logspace(start=-3, stop=3, num=20)
}, {
    'model__loss': ['hinge'],
    'model__alpha': np.logspace(start=-3, stop=3, num=20),
    'model__class_weight': [None, 'balanced']
}]

pipeline_sgd = Pipeline([
    ('scaler', StandardScaler(copy=False)),
    ('model', SGDClassifier(max_iter=1000, tol=1e-3, random_state=1,␣
↪warm_start=True))
])

MCC_scorer = make_scorer(matthews_corrcoef)
grid_sgd = GridSearchCV(estimator=pipeline_sgd, param_grid=param_grid_sgd,␣
↪scoring=MCC_scorer, n_jobs=-1, pre_dispatch='2*n_jobs', cv=5, verbose=1,␣
↪return_train_score=False)
```

```
grid_sgd.fit(X_res, y_res)
```

```
Fitting 5 folds for each of 80 candidates, totalling 400 fits
```

```
C:\Users\ADhiRAJ\Downloads\Music\Lib\site-
packages\sklearn\model_selection\_validation.py:425: FitFailedWarning:
200 fits failed out of a total of 400.
The score on these train-test partitions for these parameters will be set to
nan.
If these failures are not expected, you can try to debug them by setting
error_score='raise'.

Below are more details about the failures:
--------------------------------------------------------------------------------
25 fits failed with the following error:
Traceback (most recent call last):
  File "C:\Users\ADhiRAJ\Downloads\Music\Lib\site-
packages\sklearn\model_selection\_validation.py", line 729, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\ADhiRAJ\Downloads\Music\Lib\site-packages\sklearn\base.py",
line 1152, in wrapper
    return fit_method(estimator, *args, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\ADhiRAJ\Downloads\Music\Lib\site-packages\sklearn\pipeline.py",
line 427, in fit
    self._final_estimator.fit(Xt, y, **fit_params_last_step)
  File "C:\Users\ADhiRAJ\Downloads\Music\Lib\site-packages\sklearn\base.py",
line 1145, in wrapper
    estimator._validate_params()
  File "C:\Users\ADhiRAJ\Downloads\Music\Lib\site-packages\sklearn\base.py",
line 638, in _validate_params
    validate_parameter_constraints(
  File "C:\Users\ADhiRAJ\Downloads\Music\Lib\site-
packages\sklearn\utils\_param_validation.py", line 96, in
validate_parameter_constraints
    raise InvalidParameterError(
sklearn.utils._param_validation.InvalidParameterError: The 'loss' parameter of
SGDClassifier must be a str among {'hinge', 'log_loss', 'huber',
'squared_error', 'squared_epsilon_insensitive', 'squared_hinge', 'perceptron',
'modified_huber', 'epsilon_insensitive'}. Got 'log' instead.

--------------------------------------------------------------------------------
25 fits failed with the following error:
Traceback (most recent call last):
  File "C:\Users\ADhiRAJ\Downloads\Music\Lib\site-
packages\sklearn\model_selection\_validation.py", line 729, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\ADhiRAJ\Downloads\Music\Lib\site-packages\sklearn\base.py",
```

```
line 1152, in wrapper
    return fit_method(estimator, *args, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\ADhiRAJ\Downloads\Music\Lib\site-packages\sklearn\pipeline.py",
line 427, in fit
    self._final_estimator.fit(Xt, y, **fit_params_last_step)
  File "C:\Users\ADhiRAJ\Downloads\Music\Lib\site-packages\sklearn\base.py",
line 1145, in wrapper
    estimator._validate_params()
  File "C:\Users\ADhiRAJ\Downloads\Music\Lib\site-packages\sklearn\base.py",
line 638, in _validate_params
    validate_parameter_constraints(
  File "C:\Users\ADhiRAJ\Downloads\Music\Lib\site-
packages\sklearn\utils\_param_validation.py", line 96, in
validate_parameter_constraints
    raise InvalidParameterError(
sklearn.utils._param_validation.InvalidParameterError: The 'loss' parameter of
SGDClassifier must be a str among {'squared_hinge', 'squared_error',
'modified_huber', 'huber', 'epsilon_insensitive', 'log_loss', 'hinge',
'squared_epsilon_insensitive', 'perceptron'}. Got 'log' instead.

--------------------------------------------------------------------------------
25 fits failed with the following error:
Traceback (most recent call last):
  File "C:\Users\ADhiRAJ\Downloads\Music\Lib\site-
packages\sklearn\model_selection\_validation.py", line 729, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\ADhiRAJ\Downloads\Music\Lib\site-packages\sklearn\base.py",
line 1152, in wrapper
    return fit_method(estimator, *args, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\ADhiRAJ\Downloads\Music\Lib\site-packages\sklearn\pipeline.py",
line 427, in fit
    self._final_estimator.fit(Xt, y, **fit_params_last_step)
  File "C:\Users\ADhiRAJ\Downloads\Music\Lib\site-packages\sklearn\base.py",
line 1145, in wrapper
    estimator._validate_params()
  File "C:\Users\ADhiRAJ\Downloads\Music\Lib\site-packages\sklearn\base.py",
line 638, in _validate_params
    validate_parameter_constraints(
  File "C:\Users\ADhiRAJ\Downloads\Music\Lib\site-
packages\sklearn\utils\_param_validation.py", line 96, in
validate_parameter_constraints
    raise InvalidParameterError(
sklearn.utils._param_validation.InvalidParameterError: The 'loss' parameter of
SGDClassifier must be a str among {'squared_hinge', 'log_loss', 'hinge',
'squared_error', 'huber', 'epsilon_insensitive', 'perceptron', 'modified_huber',
'squared_epsilon_insensitive'}. Got 'log' instead.
```

```
--------------------------------------------------------------------------------
25 fits failed with the following error:
Traceback (most recent call last):
  File "C:\Users\ADhiRAJ\Downloads\Music\Lib\site-
packages\sklearn\model_selection\_validation.py", line 729, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\ADhiRAJ\Downloads\Music\Lib\site-packages\sklearn\base.py",
line 1152, in wrapper
    return fit_method(estimator, *args, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\ADhiRAJ\Downloads\Music\Lib\site-packages\sklearn\pipeline.py",
line 427, in fit
    self._final_estimator.fit(Xt, y, **fit_params_last_step)
  File "C:\Users\ADhiRAJ\Downloads\Music\Lib\site-packages\sklearn\base.py",
line 1145, in wrapper
    estimator._validate_params()
  File "C:\Users\ADhiRAJ\Downloads\Music\Lib\site-packages\sklearn\base.py",
line 638, in _validate_params
    validate_parameter_constraints(
  File "C:\Users\ADhiRAJ\Downloads\Music\Lib\site-
packages\sklearn\utils\_param_validation.py", line 96, in
validate_parameter_constraints
    raise InvalidParameterError(
sklearn.utils._param_validation.InvalidParameterError: The 'loss' parameter of
SGDClassifier must be a str among {'squared_error', 'log_loss', 'hinge',
'huber', 'modified_huber', 'squared_epsilon_insensitive', 'epsilon_insensitive',
'squared_hinge', 'perceptron'}. Got 'log' instead.

--------------------------------------------------------------------------------
25 fits failed with the following error:
Traceback (most recent call last):
  File "C:\Users\ADhiRAJ\Downloads\Music\Lib\site-
packages\sklearn\model_selection\_validation.py", line 729, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\ADhiRAJ\Downloads\Music\Lib\site-packages\sklearn\base.py",
line 1152, in wrapper
    return fit_method(estimator, *args, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\ADhiRAJ\Downloads\Music\Lib\site-packages\sklearn\pipeline.py",
line 427, in fit
    self._final_estimator.fit(Xt, y, **fit_params_last_step)
  File "C:\Users\ADhiRAJ\Downloads\Music\Lib\site-packages\sklearn\base.py",
line 1145, in wrapper
    estimator._validate_params()
  File "C:\Users\ADhiRAJ\Downloads\Music\Lib\site-packages\sklearn\base.py",
line 638, in _validate_params
    validate_parameter_constraints(
```

```
  File "C:\Users\ADhiRAJ\Downloads\Music\Lib\site-
packages\sklearn\utils\_param_validation.py", line 96, in
validate_parameter_constraints
    raise InvalidParameterError(
sklearn.utils._param_validation.InvalidParameterError: The 'loss' parameter of
SGDClassifier must be a str among {'hinge', 'log_loss', 'huber',
'squared_error', 'squared_hinge', 'perceptron', 'squared_epsilon_insensitive',
'epsilon_insensitive', 'modified_huber'}. Got 'log' instead.

--------------------------------------------------------------------------------
25 fits failed with the following error:
Traceback (most recent call last):
  File "C:\Users\ADhiRAJ\Downloads\Music\Lib\site-
packages\sklearn\model_selection\_validation.py", line 729, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\ADhiRAJ\Downloads\Music\Lib\site-packages\sklearn\base.py",
line 1152, in wrapper
    return fit_method(estimator, *args, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\ADhiRAJ\Downloads\Music\Lib\site-packages\sklearn\pipeline.py",
line 427, in fit
    self._final_estimator.fit(Xt, y, **fit_params_last_step)
  File "C:\Users\ADhiRAJ\Downloads\Music\Lib\site-packages\sklearn\base.py",
line 1145, in wrapper
    estimator._validate_params()
  File "C:\Users\ADhiRAJ\Downloads\Music\Lib\site-packages\sklearn\base.py",
line 638, in _validate_params
    validate_parameter_constraints(
  File "C:\Users\ADhiRAJ\Downloads\Music\Lib\site-
packages\sklearn\utils\_param_validation.py", line 96, in
validate_parameter_constraints
    raise InvalidParameterError(
sklearn.utils._param_validation.InvalidParameterError: The 'loss' parameter of
SGDClassifier must be a str among {'modified_huber', 'log_loss',
'squared_error', 'huber', 'squared_hinge', 'hinge', 'epsilon_insensitive',
'perceptron', 'squared_epsilon_insensitive'}. Got 'log' instead.

--------------------------------------------------------------------------------
25 fits failed with the following error:
Traceback (most recent call last):
  File "C:\Users\ADhiRAJ\Downloads\Music\Lib\site-
packages\sklearn\model_selection\_validation.py", line 729, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\ADhiRAJ\Downloads\Music\Lib\site-packages\sklearn\base.py",
line 1152, in wrapper
    return fit_method(estimator, *args, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\ADhiRAJ\Downloads\Music\Lib\site-packages\sklearn\pipeline.py",
```

```
line 427, in fit
    self._final_estimator.fit(Xt, y, **fit_params_last_step)
  File "C:\Users\ADhiRAJ\Downloads\Music\Lib\site-packages\sklearn\base.py",
line 1145, in wrapper
    estimator._validate_params()
  File "C:\Users\ADhiRAJ\Downloads\Music\Lib\site-packages\sklearn\base.py",
line 638, in _validate_params
    validate_parameter_constraints(
  File "C:\Users\ADhiRAJ\Downloads\Music\Lib\site-
packages\sklearn\utils\_param_validation.py", line 96, in
validate_parameter_constraints
    raise InvalidParameterError(
sklearn.utils._param_validation.InvalidParameterError: The 'loss' parameter of
SGDClassifier must be a str among {'squared_epsilon_insensitive',
'modified_huber', 'epsilon_insensitive', 'huber', 'hinge', 'squared_hinge',
'log_loss', 'squared_error', 'perceptron'}. Got 'log' instead.

--------------------------------------------------------------------------------
25 fits failed with the following error:
Traceback (most recent call last):
  File "C:\Users\ADhiRAJ\Downloads\Music\Lib\site-
packages\sklearn\model_selection\_validation.py", line 729, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\ADhiRAJ\Downloads\Music\Lib\site-packages\sklearn\base.py",
line 1152, in wrapper
    return fit_method(estimator, *args, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\ADhiRAJ\Downloads\Music\Lib\site-packages\sklearn\pipeline.py",
line 427, in fit
    self._final_estimator.fit(Xt, y, **fit_params_last_step)
  File "C:\Users\ADhiRAJ\Downloads\Music\Lib\site-packages\sklearn\base.py",
line 1145, in wrapper
    estimator._validate_params()
  File "C:\Users\ADhiRAJ\Downloads\Music\Lib\site-packages\sklearn\base.py",
line 638, in _validate_params
    validate_parameter_constraints(
  File "C:\Users\ADhiRAJ\Downloads\Music\Lib\site-
packages\sklearn\utils\_param_validation.py", line 96, in
validate_parameter_constraints
    raise InvalidParameterError(
sklearn.utils._param_validation.InvalidParameterError: The 'loss' parameter of
SGDClassifier must be a str among {'squared_error', 'modified_huber',
'squared_hinge', 'huber', 'perceptron', 'epsilon_insensitive',
'squared_epsilon_insensitive', 'log_loss', 'hinge'}. Got 'log' instead.

  warnings.warn(some_fits_failed_message, FitFailedWarning)
C:\Users\ADhiRAJ\Downloads\Music\Lib\site-
packages\sklearn\model_selection\_search.py:979: UserWarning: One or more of the
```

```
test scores are non-finite: [        nan         nan         nan         nan
nan         nan
        nan         nan         nan         nan         nan         nan
        nan         nan         nan         nan         nan         nan
        nan         nan         nan         nan         nan         nan
        nan         nan         nan         nan         nan         nan
        nan         nan         nan         nan         nan         nan
        nan         nan         nan         nan 0.94997731 0.94997731
 0.94490925 0.94490925 0.94014831 0.94014831 0.93549796 0.93549796
 0.92929771 0.92929771 0.919665   0.919665   0.91054209 0.91054209
 0.88885805 0.88885805 0.86805323 0.86805323 0.84164232 0.84164232
 0.80113477 0.80113477 0.73528779 0.73528779 0.87123537 0.87123537
 0.04920753 0.04920753 0.0014177  0.0014177  0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.        ]
  warnings.warn(
```

[32]:
```
GridSearchCV(cv=5,
             estimator=Pipeline(steps=[('scaler', StandardScaler(copy=False)),
                                       ('model',
                                        SGDClassifier(random_state=1,
                                                      warm_start=True))]),
             n_jobs=-1,
             param_grid=[{'model__alpha': array([1.00000000e-03, 2.06913808e-03,
4.28133240e-03, 8.85866790e-03,
       1.83298071e-02, 3.79269019e-02, 7.84759970e-02, 1.62377674e-01,
       3.35981829e-01, 6.95192796e-01, 1.43844989e+00,…
       1.83298071e-02, 3.79269019e-02, 7.84759970e-02, 1.62377674e-01,
       3.35981829e-01, 6.95192796e-01, 1.43844989e+00, 2.97635144e+00,
       6.15848211e+00, 1.27427499e+01, 2.63665090e+01, 5.45559478e+01,
       1.12883789e+02, 2.33572147e+02, 4.83293024e+02, 1.00000000e+03]),
                          'model__class_weight': [None, 'balanced'],
                          'model__loss': ['hinge']}],
             scoring=make_scorer(matthews_corrcoef), verbose=1)
```

[33]:
```
grid_eval(grid_sgd)
```

```
Best Score 0.9499773085502039
Best Parameter {'model__alpha': 0.001, 'model__class_weight': None,
'model__loss': 'hinge'}
```

[34]:
```
evaluation(y_test, grid_sgd, X_test)
```

```
CLASSIFICATION REPORT
              precision    recall  f1-score   support

           0       1.00      0.99      1.00     85295
           1       0.17      0.89      0.28       148
```

```
    accuracy                            0.99     85443
   macro avg       0.58      0.94      0.64     85443
weighted avg       1.00      0.99      0.99     85443

AUC-ROC
0.942112192502016
F1-Score
0.2826552462526767
Accuracy
0.9921585150334141
```

[ ]:
```
pipeline_rf = Pipeline([
    ('model', RandomForestClassifier(n_jobs=-1, random_state=1))
])
param_grid_rf = {'model__n_estimators': [75]}
grid_rf = GridSearchCV(estimator=pipeline_rf, param_grid=param_grid_rf,␣
 ↪scoring=MCC_scorer, n_jobs=-1, pre_dispatch='2*n_jobs', cv=5, verbose=1,␣
 ↪return_train_score=False)
grid_rf.fit(X_res, y_res)
```

```
Fitting 5 folds for each of 1 candidates, totalling 5 fits
```

[ ]:
```
grid_eval(grid_rf)
```

[ ]:
```
evaluation(y_test, grid_rf, X_test)
```

[ ]:
```
pipeline_knn = Pipeline([
    ('model', KNeighborsClassifier(n_neighbors=5))
])
param_grid_knn = {'model__p': [2]}
grid_knn = GridSearchCV(estimator=pipeline_knn, param_grid=param_grid_knn,␣
 ↪scoring=MCC_scorer, n_jobs=-1, pre_dispatch='2*n_jobs', cv=5, verbose=1,␣
 ↪return_train_score=False)
grid_knn.fit(X_res, y_res)
```

[ ]:
```
grid_eval(grid_knn)
```

[ ]:
```
evaluation(y_test, grid_knn, X_test)
```

8 Conclusion

The K-Nearest Neighbors Classifier tuned with Grid Search with the best parameter being the Euclidean Distance (p=2) outperforms its counterparts to give a test accuracy of nearly 99.8% and a perfect F1-Score with minimal overfitting SMOTE overcomes overfitting by synthetically oversampling minority class labels and is successful to a great degree # Summary All Fraud Transactions occur for an amount below 2500. Thus, the bank can infer clearly that the fraud committers try to commit frauds of smaller amounts to avoid suspicion. The fraud transactions are equitable distributed throughout time and there is no clear relationship of time with commiting of fraud. The number of fraud transactions are very few comparted to legitimate transactions and it has to be balanced in order for a fair comparison to prevent the model from overfitting.

[ ]:
```

```

[ ]:
```

```