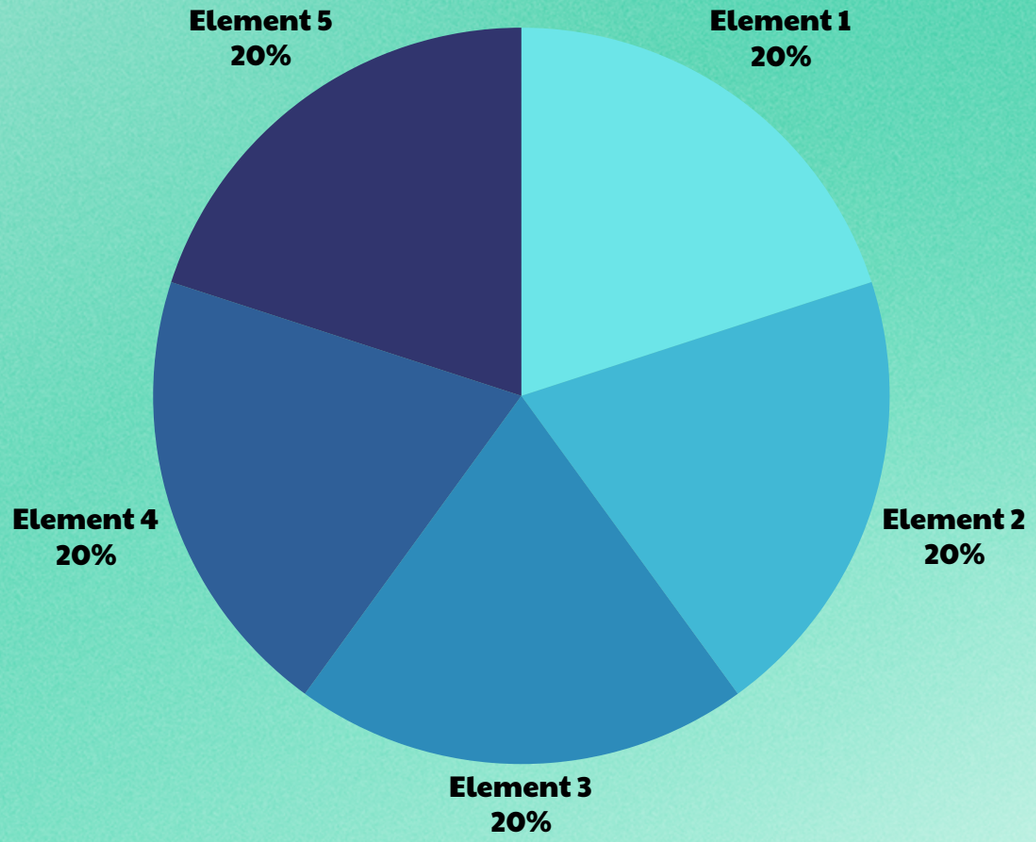
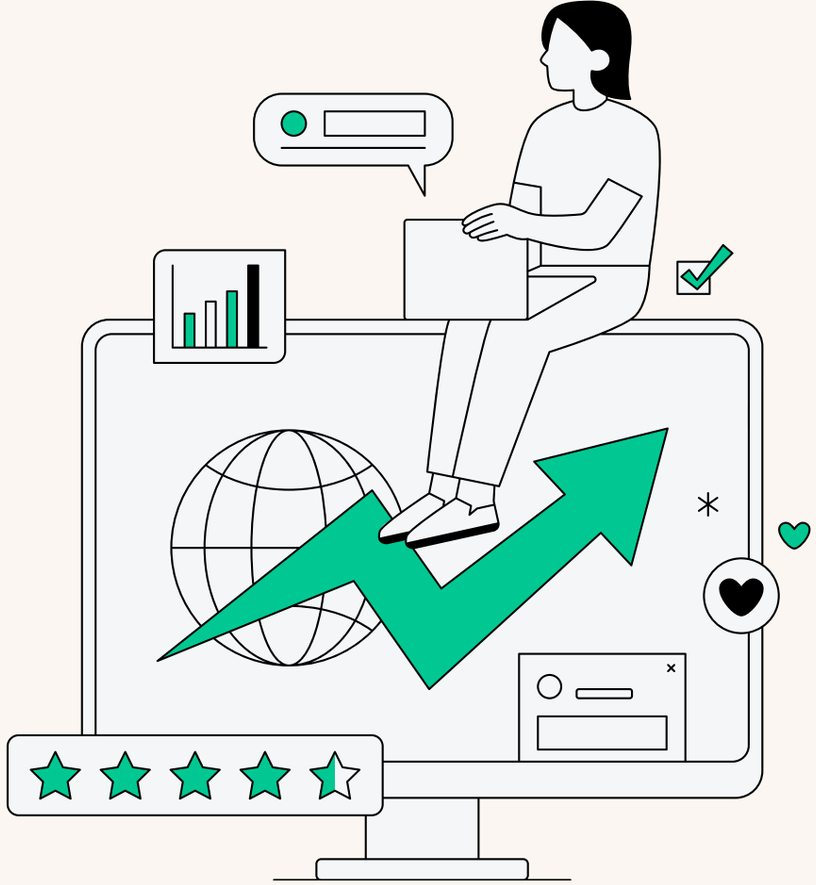
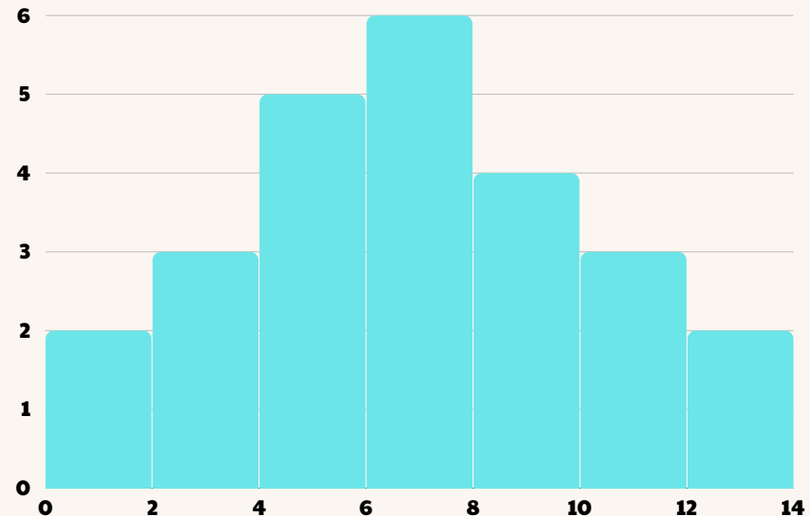
Credit Card Fraud Detection

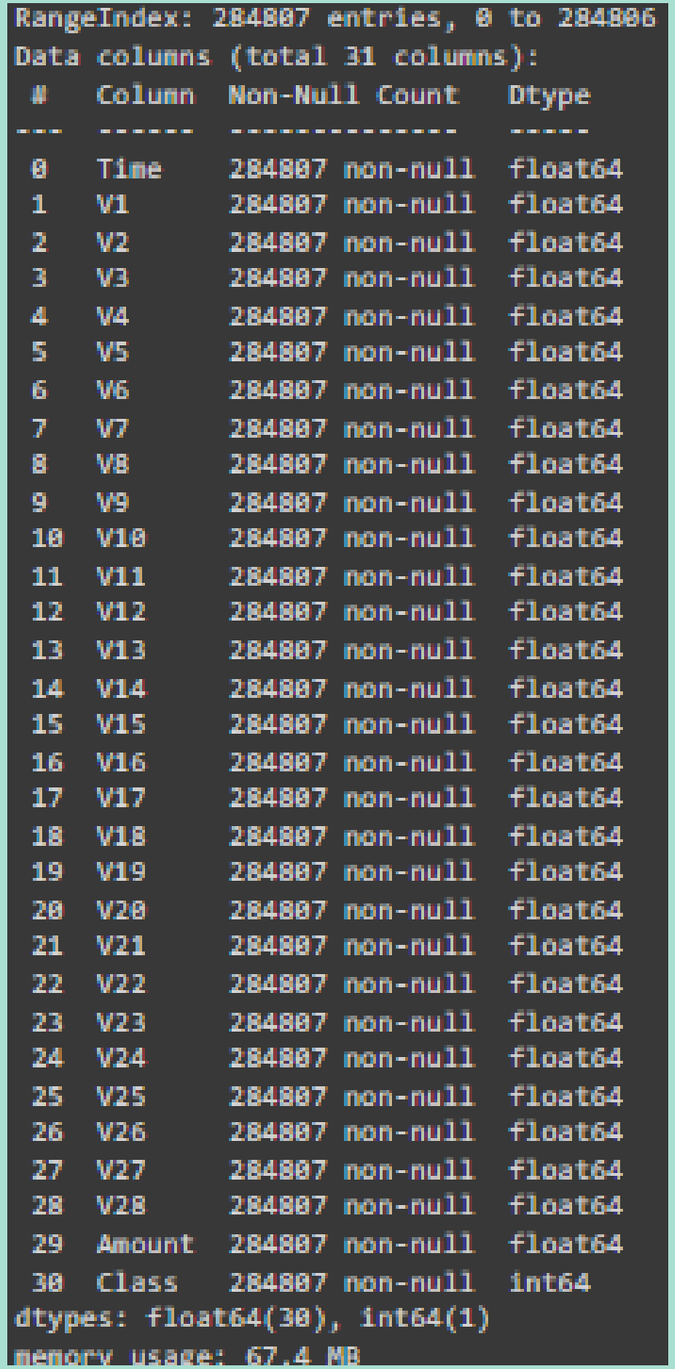






Credit-card dataset

The dataset consists of 284807 rows and 31 columns Dataset Checks credit card transactions for fraud. There are no missing values in the dataset

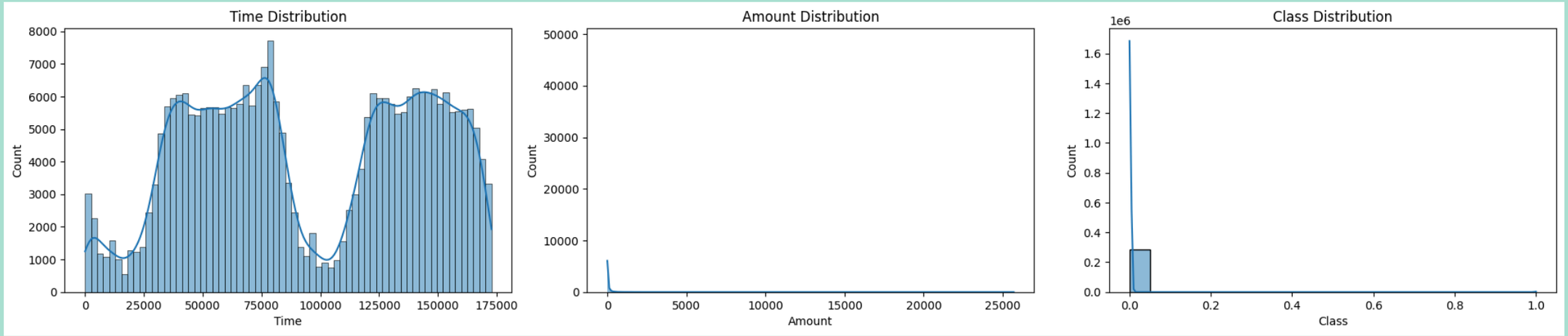


```
RangeIndex: 284807 entries, 0 to 284806
Data columns (total 31 columns):
 #   Column  Non-Null Count   Dtype  
---  ------  --------------   -----  
 0   Time    284807 non-null  float64
 1   V1      284807 non-null  float64
 2   V2      284807 non-null  float64
 3   V3      284807 non-null  float64
 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
```

EDA

Exploratory data analysis

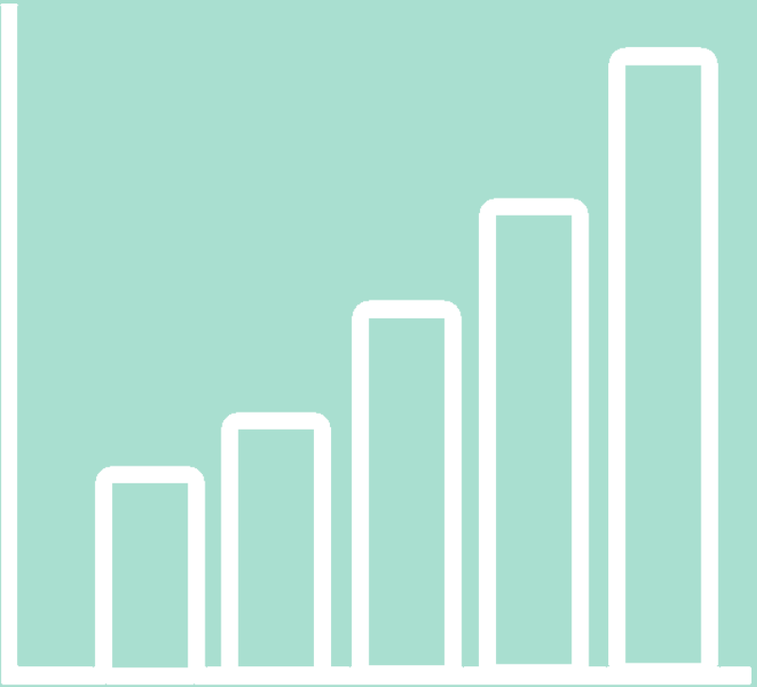
First we look at the distribution of the variables (V1-V28 is not meaningful because it is changed by PCA). As you can see here, the Time column shows a lot of variance and the Class column shows a lot of non-fraud



We know that the class column is our target column.

What about Time and Amount?

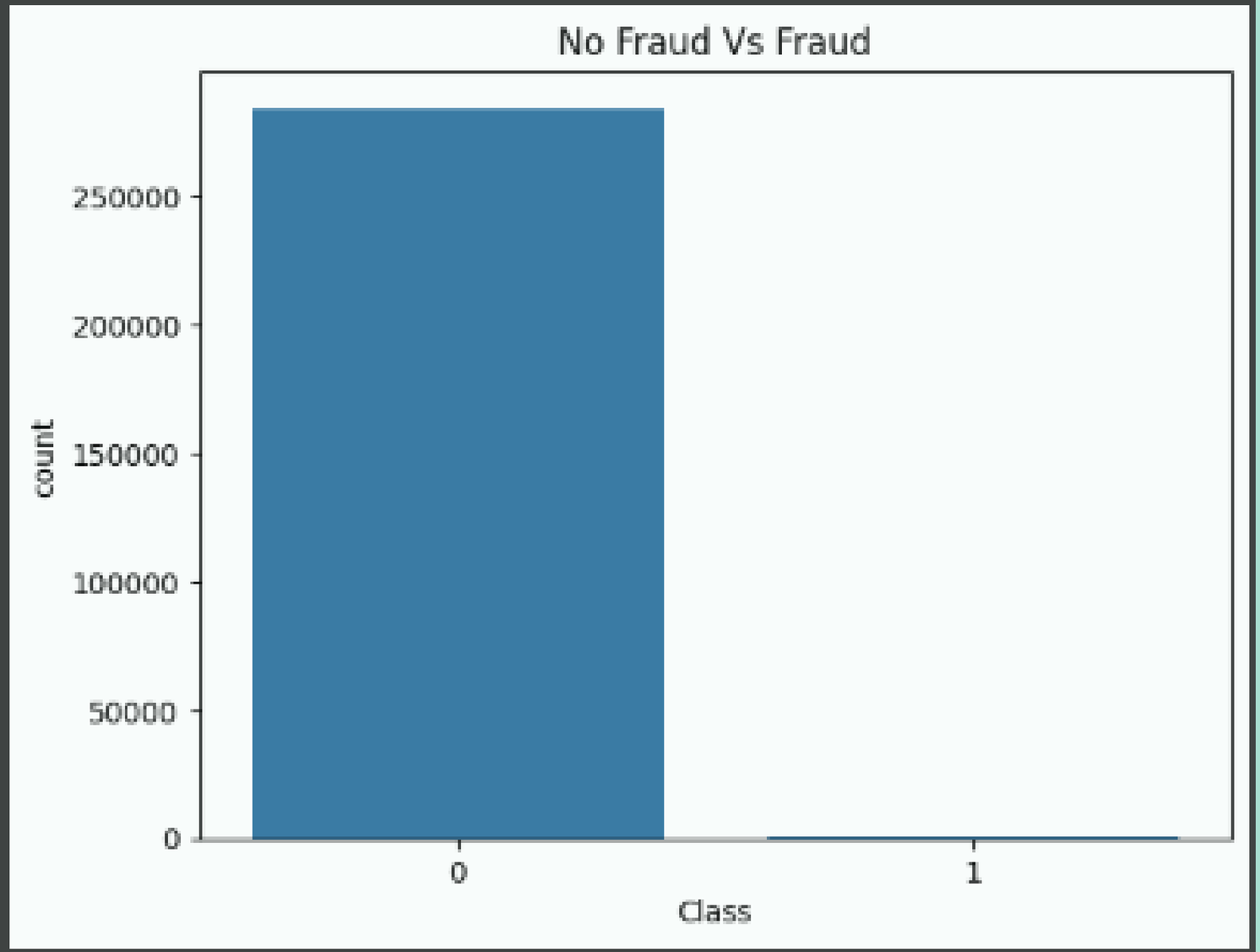
Amount column shows mean 88, median 22, std 250 and outlier



	Time	Amount
count	284807.000000	284807.000000
mean	94813.859575	88.349619
std	47488.145955	250.120109
min	0.000000	0.000000
25%	54201.500000	5.600000
50%	84692.000000	22.000000
75%	139320.500000	77.165000
max	172792.000000	25691.160000

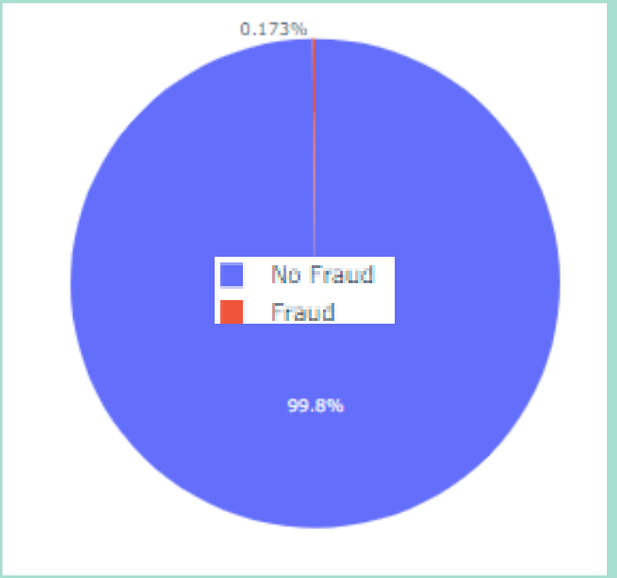
Could our Target column be imbalanced?

```
492 Of Frauds
284315 No Frauds
```



In the Class column, we have 492 Frauds, 284315 No Fraud transactions, which indicates that we have an imbalance dataset.

Non-fraud transaction percentage = 99.827%
Fraud Transaction Rate = 0.173%

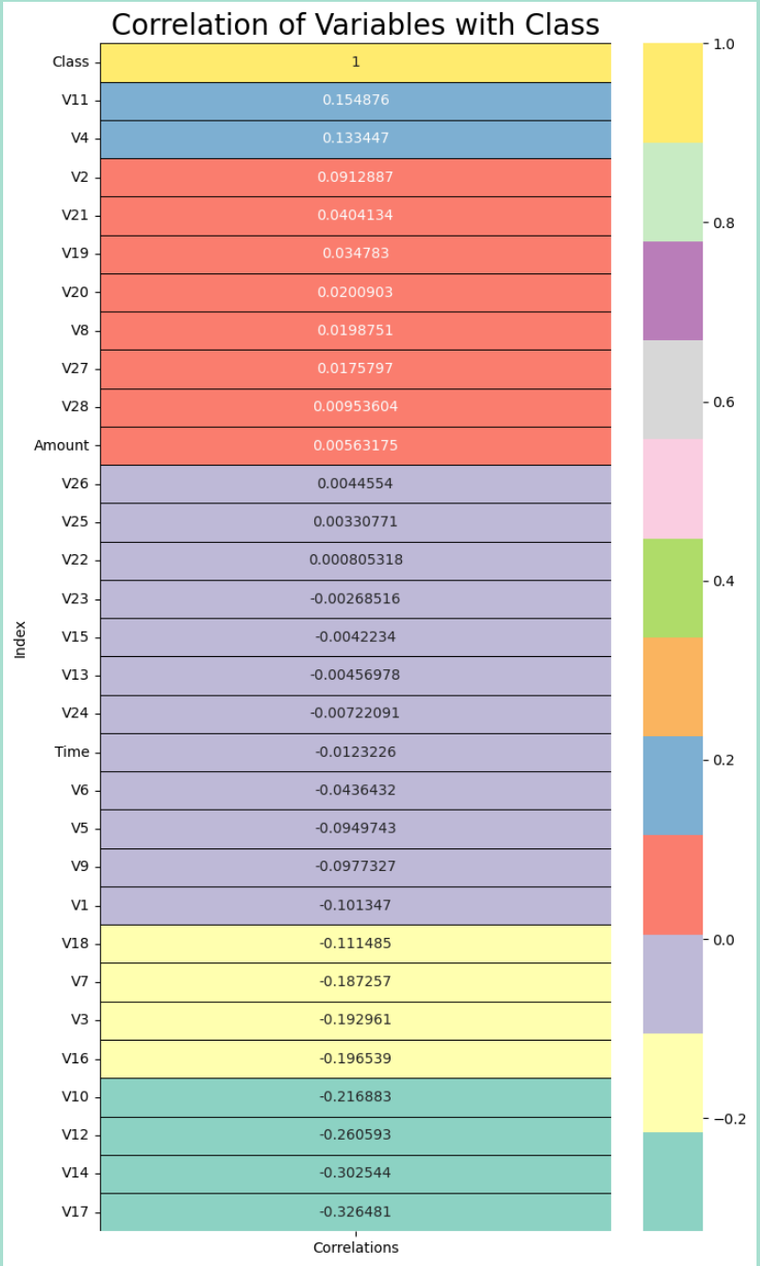


What is the transaction amount?

Here you can see that 113 transactions are 1.27 transactions for $99.99 and there are also 27 transactions for $0.

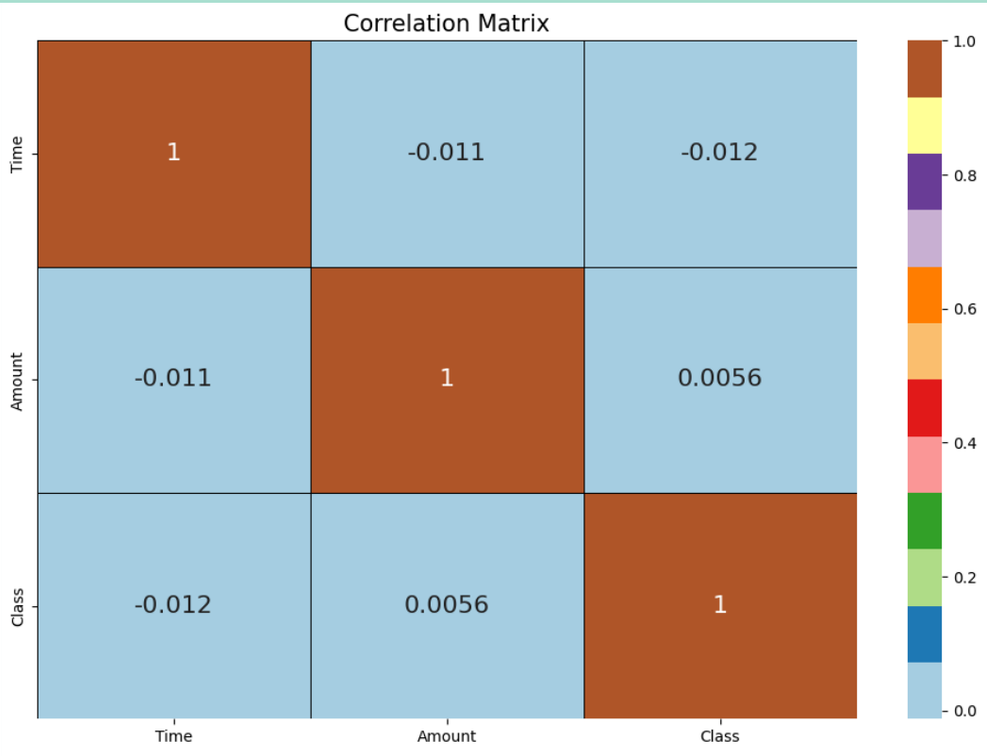
```
Amount
1.00      113
0.00       27
99.99      27
0.76       17
0.77       10
Name: count, dtype: int64
```

Correlation matrix

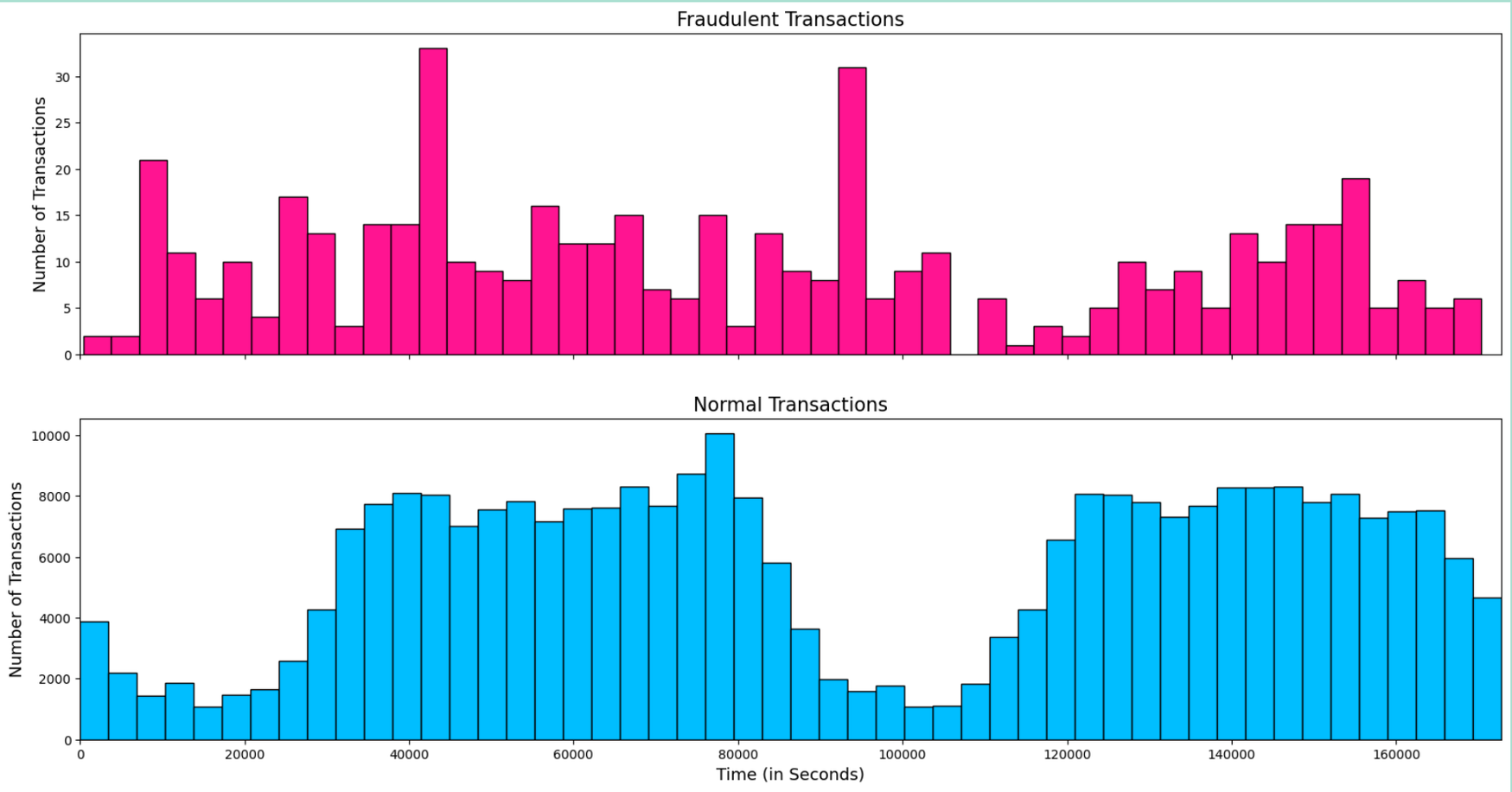


As seen here
V10,V12,V14,V17 are negative correlation
V11,V2,V4,V21 are positively correlated

Here, only the correlation of Time, Amount and Class columns is shown. There is a slight correlation between Class and Amount, but not with Time.

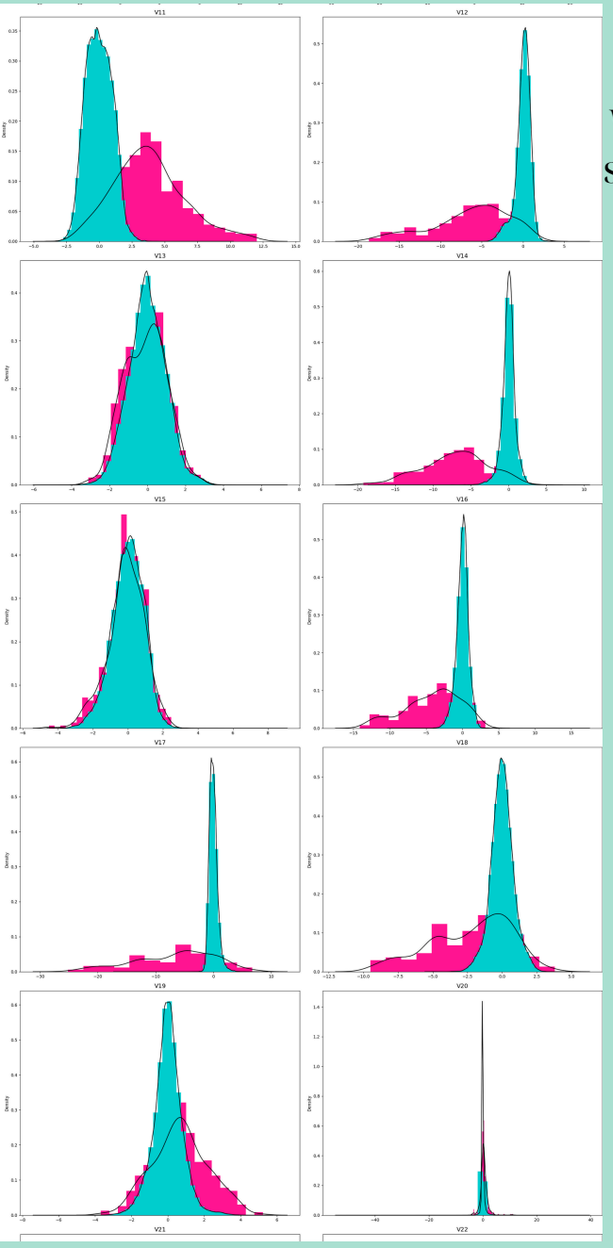


What is the distribution between Time and Class?

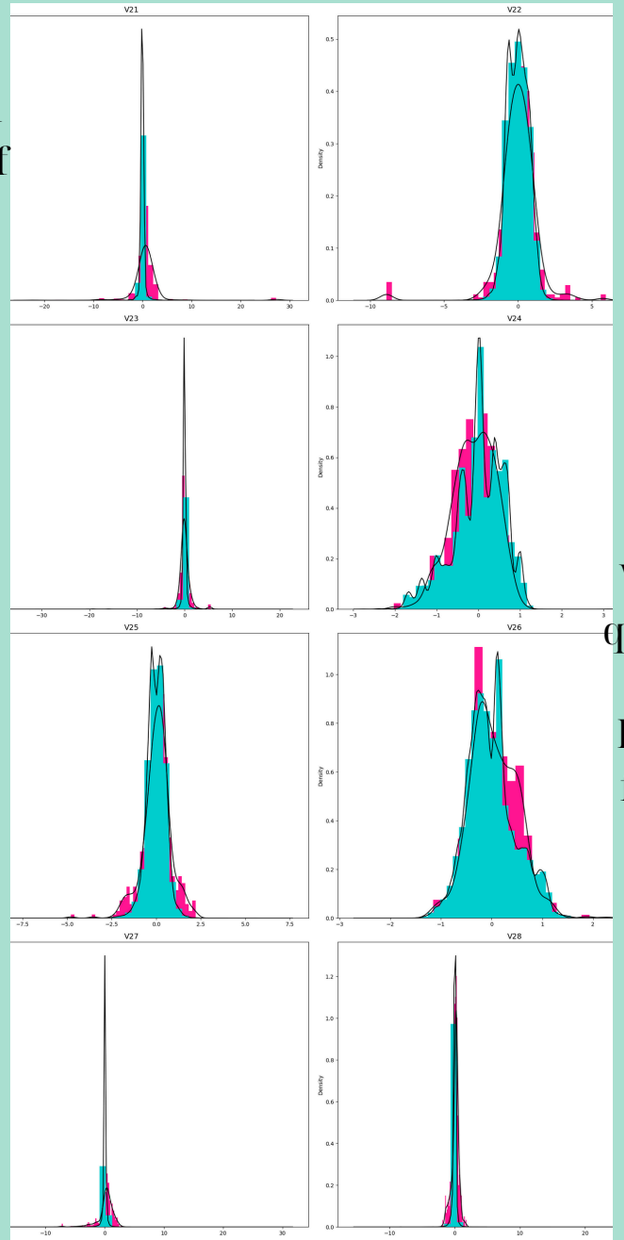


Here, in the distribution of normal and fraud transactions by Time, according to Class, it can be seen that Fraud increases between 4000-100000, and the normal transaction also constitutes the majority of transactions at the same time as the fraud transaction.

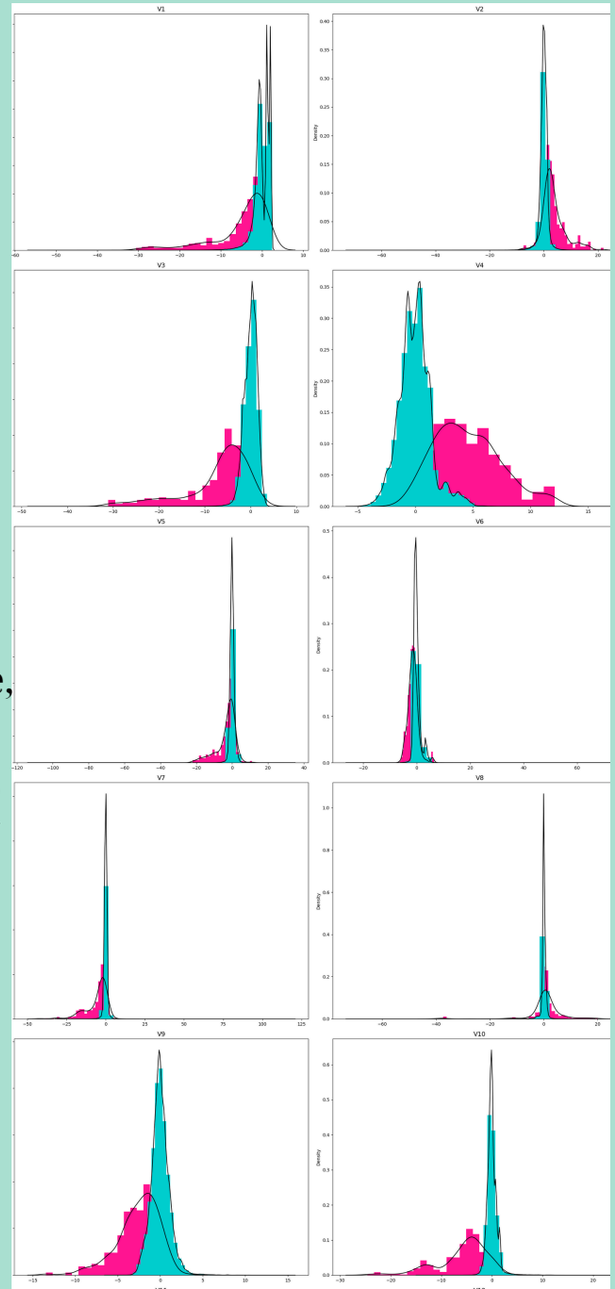
Now let's look at the distribution of PCA Vs with Class



For some features, we can observe good selectivity in terms of distribution for the two values of Class: V4, V11 have clearly separated distributions for class values 0 and 1,



V12, V14, V18 are partially separated, V1, V2, V3 , V10 have quite different profile, whereas V20-V28 have similar profiles for the two values of Class and therefore not very useful in differentiating between the two classes.



Outlier handling

As we have already seen, there are too many outliers in the amount column, so they need to be removed as they can affect the performance of the model. We will use the interquartile range to detect outliers, eliminating anything above the lower bound (25 percent) and upper bound (75 percent).

```
Q3 = np.percentile(data['Amount'], 75)
Q1 = np.percentile(data['Amount'], 25)

cutoff = 5.0

IQR = (Q3 - Q1)

lower_bound = Q1 - (IQR * cutoff)
upper_bound = Q3 + (IQR * cutoff)

filter_data = (data['Amount'] < lower_bound) | (data['Amount'] > upper_bound)

outliers = data[filter_data]['Amount']
fraud_outliers = data[(data['Class'] == 1) & filter_data]['Amount']
normal_outliers = data[(data['Class'] == 0) & filter_data]['Amount']

print(f"Total Number of Outliers : {outliers.count()}")
print(f"Number of Outliers in Fraudulent Class : {fraud_outliers.count()}")
print(f"No of Outliers in Normal Class : {normal_outliers.count()}")
print(f"Percentage of Fraud amount outliers : {round((fraud_outliers.count()/outliers.count())*100,2)}%")
```

```
Total Number of Outliers : 11366
Number of Outliers in Fraudulent Class : 41
No of Outliers in Normal Class : 11325
Percentage of Fraud amount outliers : 0.36%
```

```
data = data.drop(outliers.index)
data.reset_index(inplace=True, drop=True)
```

```
data.shape
```

```
(273441, 31)
```

Note that the data we have for fraud cases is very low, so we want to keep the cutoff a bit high to avoid removing too many cases of fraud. Here, since the data is skewed (sort of exponential), a high cutoff will help us. Let's take the cutoff value as 5.0 instead of the commonly used 1.5.

Variance İnflation Factor

	features	vif_value
0	Time	1.000000
1	V1	1.016583
2	V2	1.050082
3	V3	1.020613
4	V4	1.000685
5	V5	1.000697
6	V6	1.025602
7	V7	1.031733
8	V8	1.003093
9	V9	1.000014
10	V10	1.000030
11	V11	1.000000
12	V12	1.000020
13	V13	1.000002
14	V14	1.000018
15	V15	1.000001
16	V16	1.000015
17	V17	1.000055
18	V18	1.000044
19	V19	1.000064
20	V20	1.005710
21	V21	1.044297
22	V22	1.000568
23	V23	1.006878
24	V24	1.001550
25	V25	1.000001
26	V26	1.000052
27	V27	1.000018
28	V28	1.026239
29	Amount	1.000031
30	Class	1.000629

Scaling

```
from sklearn.preprocessing import RobustScaler
robust_scaler = RobustScaler()
data['Time'] = robust_scaler.fit_transform(data['Time'].values.reshape(-1,1))
```

Train ,Test split

```
X = data.drop(['Class'],axis=1)
Y = data['Class']
```

Let's balance with the SMOTE method

```
from collections import Counter
from imblearn.over_sampling import SMOTE


print(f'Original dataset shape : {Counter(Y)}')

smote = SMOTE(random_state=42)
X_res, y_res = smote.fit_resample(X, Y)

print(f'Resampled dataset shape {Counter(y_res)}')
```

```
Original dataset shape : Counter({0: 272990, 1: 451})
Resampled dataset shape Counter({0: 272990, 1: 272990})
```

Cross Validation

```
from sklearn.model_selection import StratifiedKFold
strat = StratifiedKFold(n_splits=5, random_state=None, shuffle=False)

for train_index, test_index in strat.split(X_res, y_res):
    print("Train:", train_index, "Test:", test_index)
    original_Xtrain, original_Xtest = X_res.iloc[train_index], X_res.iloc[test_index]
    original_ytrain, original_ytest = y_res.iloc[train_index], y_res.iloc[test_index]
```

```
Train: [ 54745  54746  54747 ... 545977 545978 545979] Test: [     0      1      2 ... 327585 327586 327587]
Train: [     0      1      2 ... 545977 545978 545979] Test: [ 54745  54746  54747 ... 382183 382184 382185]
Train: [     0      1      2 ... 545977 545978 545979] Test: [109418 109419 109420 ... 436781 436782 436783]
Train: [     0      1      2 ... 545977 545978 545979] Test: [164127 164128 164129 ... 491379 491380 491381]
Train: [     0      1      2 ... 491379 491380 491381] Test: [218772 218773 218774 ... 545977 545978 545979]
```

```
original_Xtrain = original_Xtrain.values
original_Xtest = original_Xtest.values
original_ytrain = original_ytrain.values
original_ytest = original_ytest.values
```

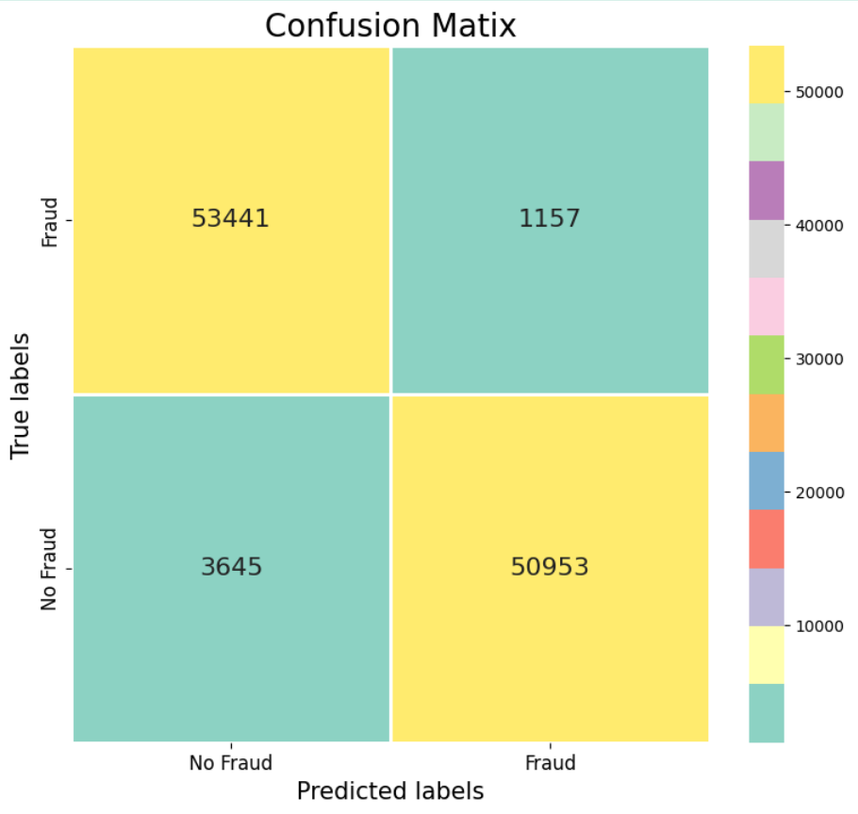
Train again, Test split

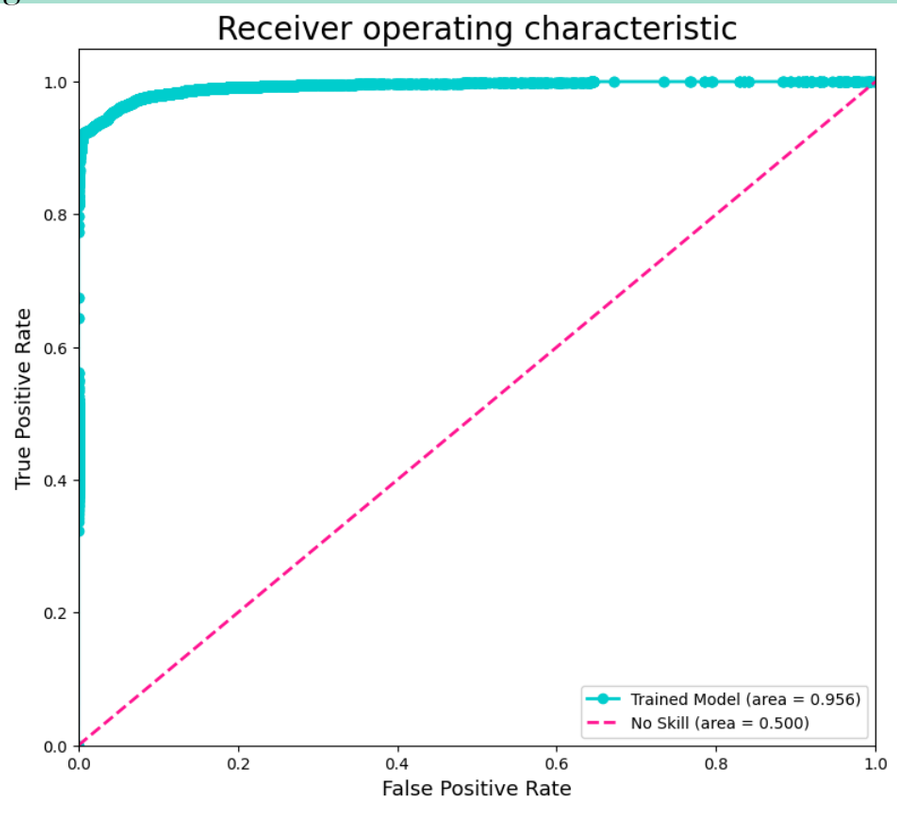
MODELLING

Logistic Regression

```
Train_accuracy: 93.59%
Test_accuracy: 93.32%
accuracy_score: 95.60%
roc_auc_score: 99.23%
precision_score: 93.32%
recall_score: 97.78%
f1_score: 95.50%
```

ROC, AUC score showed accuracy of 93.32 percent and it shows that the performance of the model is good





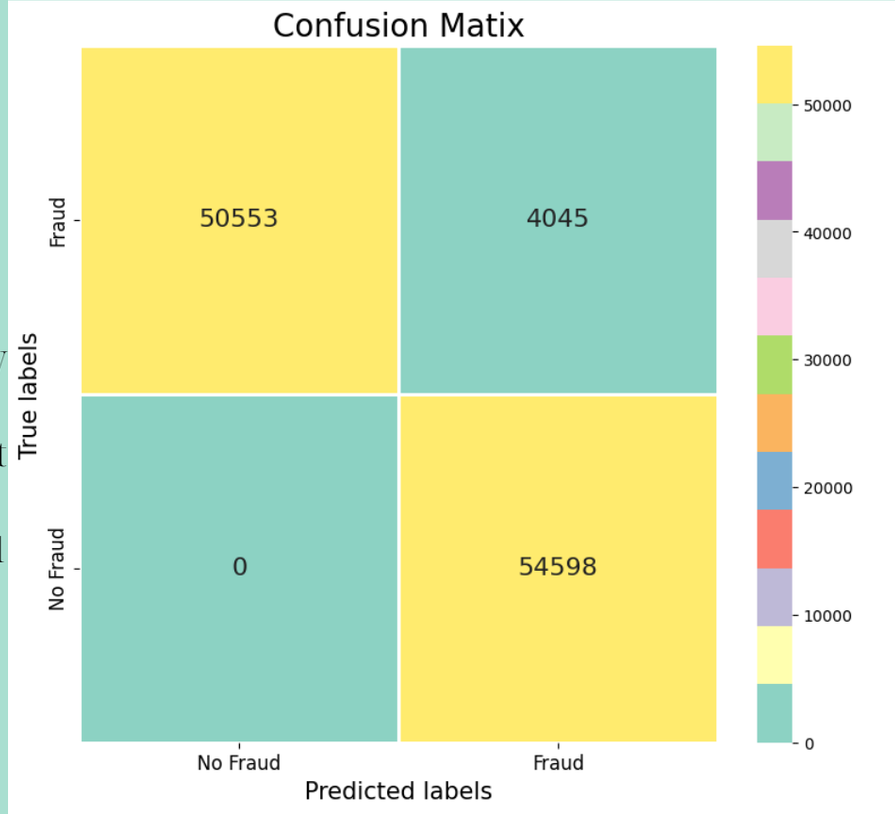
The real truth is 53441, that is, those who are frauds of my model are really frauds
False negative 1137 was not fraud, but the model saw it as fraud
False negative 3645 saw true value as no fraud but model fraud
A true negative of 50953 saw the true value as no fraud in the no fraud model as well

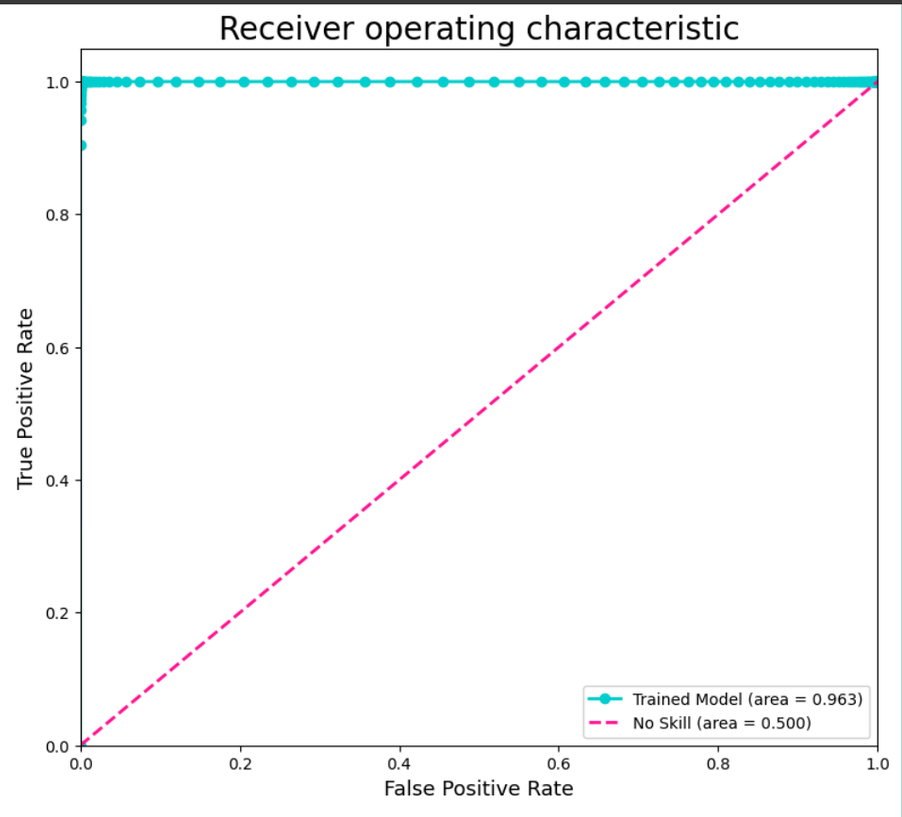
Random Forest Classifier

Results of 100% in this RFC model indicate that the model is experiencing overfitting

```
Train_accuracy: 100.00%
Test_accuracy: 96.30%
accuracy_score: 96.30%
roc_auc_score: 100.00%
precision_score: 100.00%
recall_score: 93.10%
f1_score: 96.43%
```

The real truth is 50553, that is, those who are frauds of my model are really frauds
False negative 4045 was not fraud, but the model saw it as fraud
False negative 0 true value is no fraud but the model saw it as fraud
A true negative of 54508 saw the true value as no fraud in the no fraud model as well

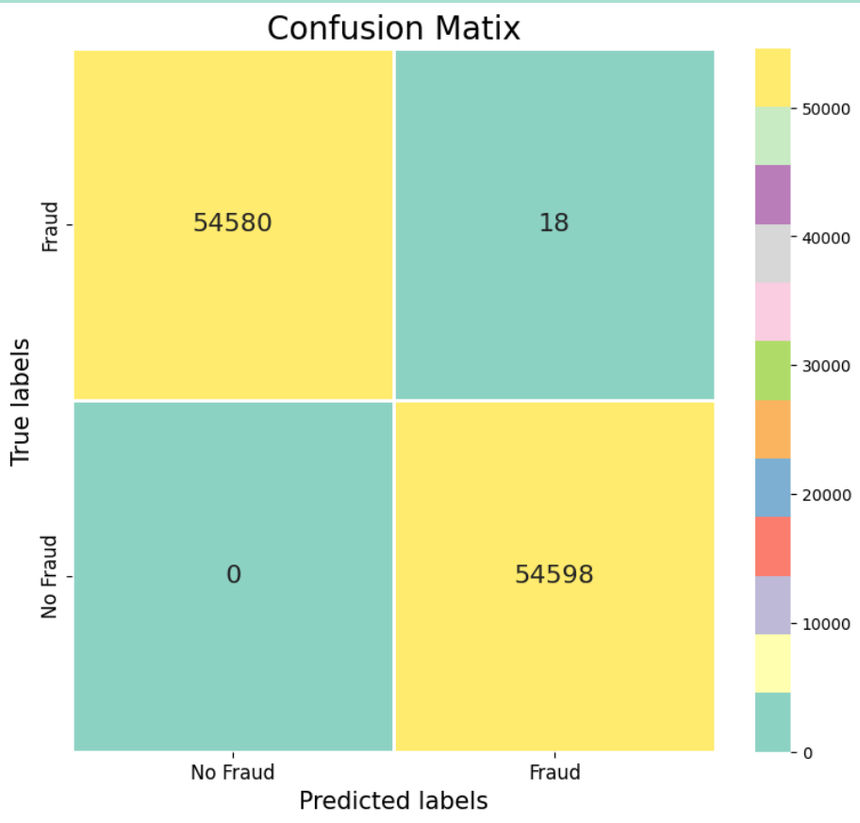


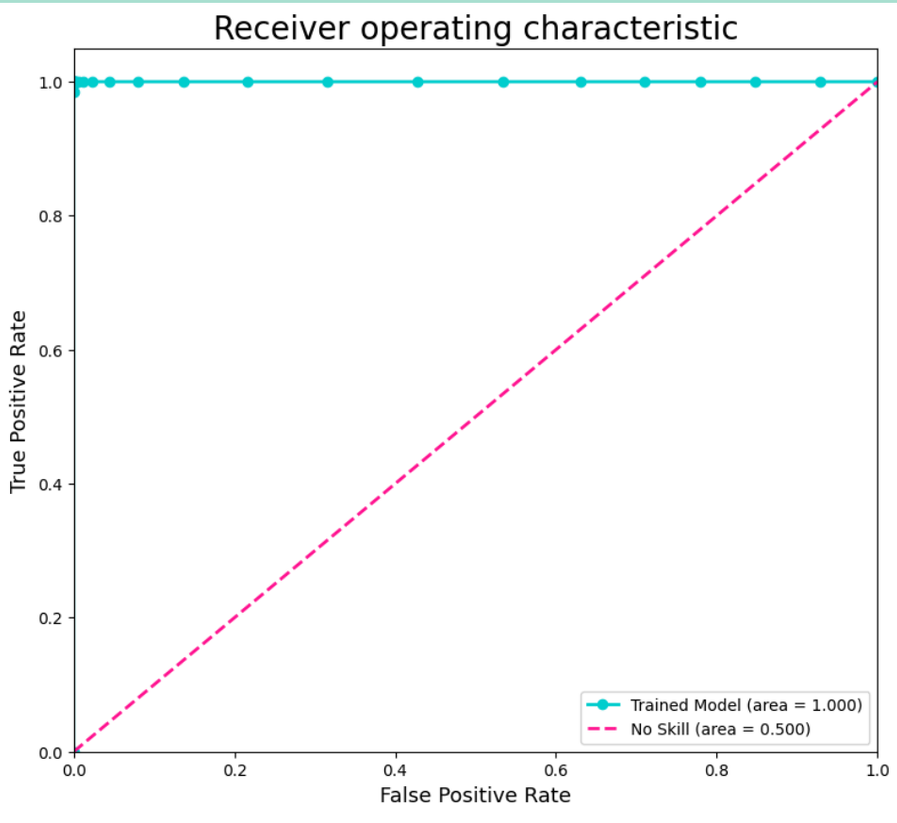


Extra Trees Classifier

```
Train_accuracy: 100.0 %
Test_accuracy: 99.98351587970255 %
accuracy_score: 99.98351587970255 %
precision_score: 100.0 %
recall_score: 99.96704262487182 %
roc_auc_score: 99.99999396616396 %
f1_score: 99.98351859651693 %
```

Overfitting is seen here, just like the RFC model

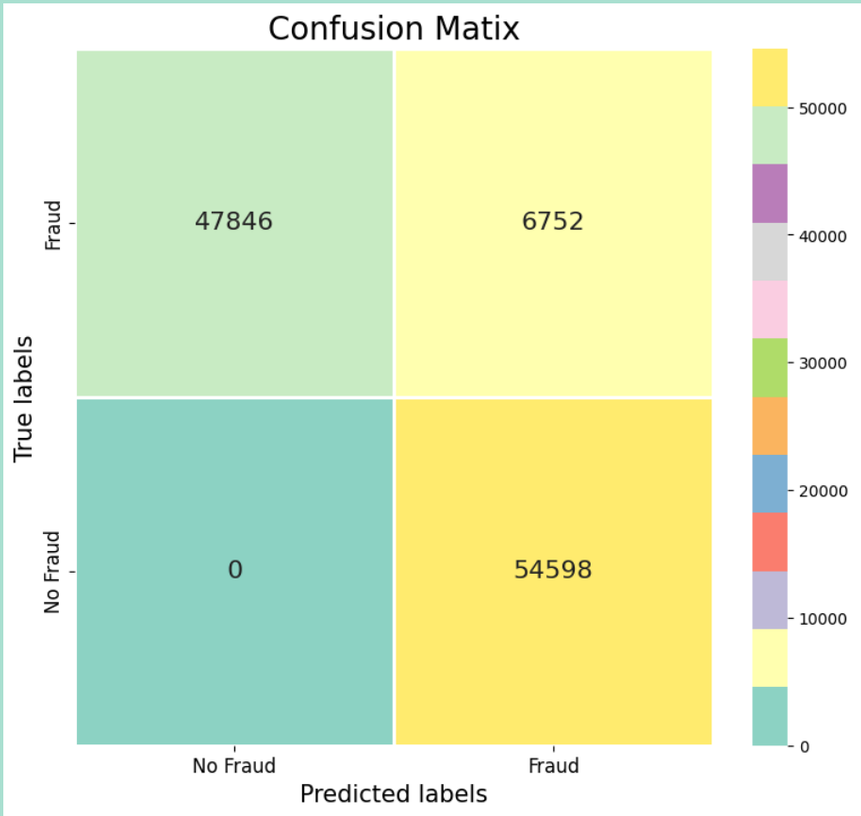


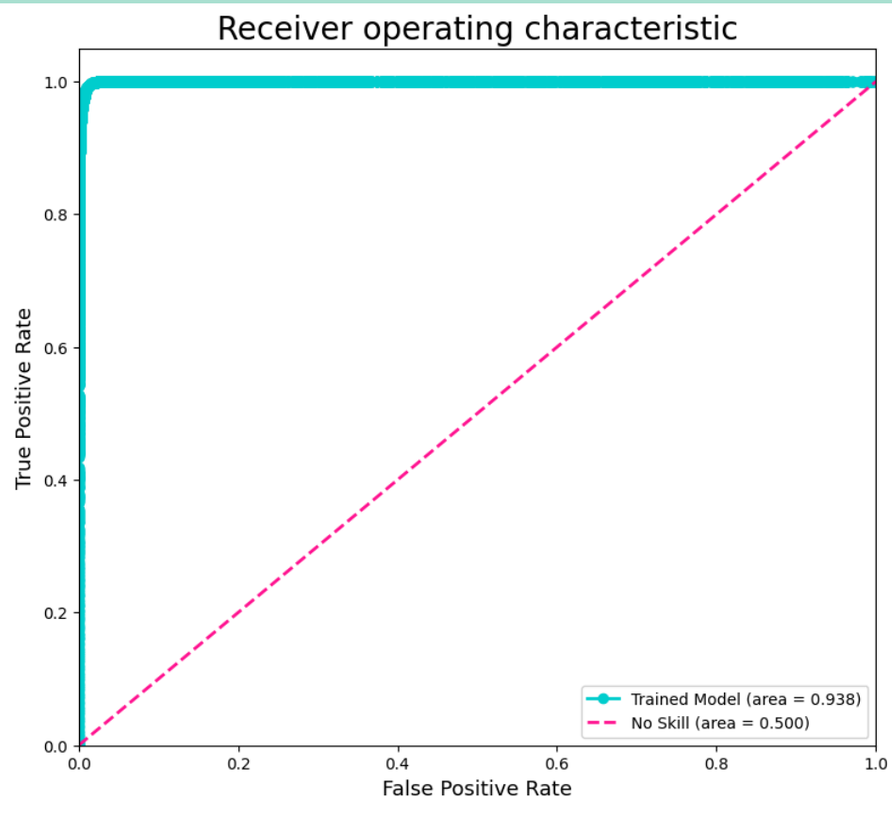


XGB Classifier

As shown in the XGB model, the model scores show us that the model will give correct predictions

```
Train_accuracy: 99.99450529323418 %
Test_accuracy: 93.81662331953552 %
accuracy_score: 93.81662331953552 %
precision_score: 100.0 %
recall_score: 88.99429502852486 %
roc_auc_score: 99.94360481608415 %
f1_score: 94.17669989995515 %
```





The Result

After dataset EDA and modeling, 2 models Logistic Regression and XGB classifier models are the models that we can choose our correct predict. Since the better result is in XGB model, this model will show our main predict