FindDefault (Prediction of Credit Card fraud)

Problem Statement:

A credit card is one of the most used financial products to make online purchases and payments. Though the Credit cards can be a convenient way to manage your finances, they can also be risky. Credit card fraud is the unauthorized use of someone else's credit card or credit card information to make purchases or withdraw cash.

It is important that credit card companies are able to recognize fraudulent credit card transactions so that customers are not charged for items that they did not purchase.

The dataset contains transactions made by credit cards in September 2013 by European cardholders. This dataset presents transactions that occurred in two days, where we have 492 frauds out of 284,807 transactions. The dataset is highly unbalanced, the positive class (frauds) account for 0.172% of all transactions.

We have to build a classification model to predict whether a transaction is fraudulent or not.

Data Exploration

```
df
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431
...	...	...	...	...	...	...	...	...	...	...	...	...
284802	172786.0	-11.881118	10.071785	-9.834783	-2.066656	-5.364473	-2.606837	-4.918215	7.305334	1.914428	...	0.213454
284803	172787.0	-0.732789	-0.055080	2.035030	-0.738589	0.868229	1.058415	0.024330	0.294869	0.584800	...	0.214205
284804	172788.0	1.919565	-0.301254	-3.249640	-0.557828	2.630515	3.031260	-0.296827	0.708417	0.432454	...	0.232045
284805	172788.0	-0.240440	0.530483	0.702510	0.689799	-0.377961	0.623708	-0.686180	0.679145	0.392087	...	0.265245
284806	172792.0	-0.533413	-0.189733	0.703337	-0.506271	-0.012546	-0.649617	1.577006	-0.414650	0.486180	...	0.261057

284807 rows × 31 columns

To protect the user's identity and the security of their confidential information, the dataset provider has applied Principal Component Analysis transformation on the original numerical features and compressed it into 28 principal's components.

Only two features have not been transformed i.e. 1) Time and 2) Amount

The feature class will be targeting column with user labels as:

0: non-fraudulent

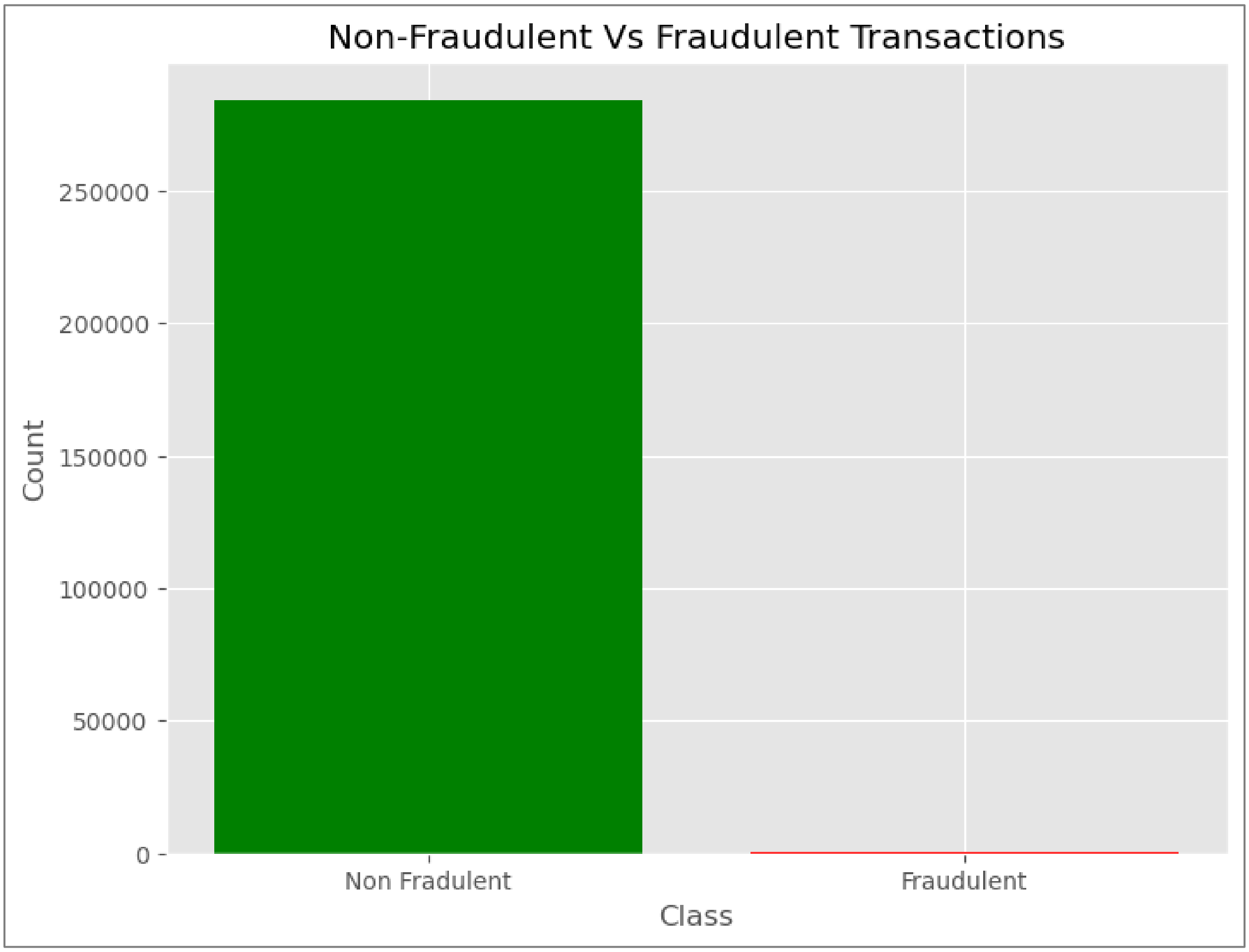
1: fraudulent

The dataset exclusively comprises numerical features, and notably, there are no instances of missing values.

```
<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
```

Exploratory Data Analysis

- For the subsequent step, we will conduct fundamental Exploratory Data Analysis (EDA) on the dataset to enhance our understanding and extract valuable insights.



- The bar plot reveals a significant imbalance between classes (0: Non-Fraudulent) and (1: Fraudulent).
- Majority of features are in PCA (Principal component analysis) form, with the exceptions being Time and Amount, a more in-depth examination of these two features is required.

```
df['Time'].describe()

count    284807.000000
mean      94813.859575
std       47488.145955
min           0.000000
25%       54201.500000
50%       84692.000000
75%      139320.500000
max      172792.000000
Name: Time, dtype: float64
```

```
df['Amount'].describe()

count    284807.000000
mean         88.349619
std         250.120109
min           0.000000
25%           5.600000
50%          22.000000
75%          77.165000
max       25691.160000
Name: Amount, dtype: float64
```

- Now we will check the number of occurances of each class label and we will plot the information using matplotlib.

```
Number of Non-Fraudulent Transactions:  284315
Number of Fraudulent Transactions:  492
Percentage of Fraudulent Transactions:  0.17
```

- Our Non-Fraudulent transactions are over 99%.
- We will apply scaling techniques on the **"Amount"** feature to transform the range of values.
- We will drop the original **"Amount"** column and add a new column with the scaled values. We will also drop the **"Time"** columns as it is irrelevant.

- Now, we will split the credit card data with a split of 70-30 using train_test_split().
- train_test_split() function in scikit-learn is a useful utility for splitting a dataset into training and testing sets.
    - Parameters
    - X: Feature matrix
    - Y: Target variable
    - test_size: Proportion of the dataset to include in the test split. Here we have set the test_size as 0.3 means 30% of the data we take as testing data set.

- random_state: We have set the seed for random number generation, to ensure the reproducibility

```
Shape of the training dataset train_X:  (199364, 29)
Shape of the testing dataset test_X:  (85443, 29)
```

Applying Machine Learning Algorithm to Credit Card Dataset

- We will explore various machine learning algorithms to find the most effective model for our binary classification problem.
- The task involves predicting one of the two class labels. We plan to access the performance of different algorithms, such as Random Forest and Decision Tree, to identify the most suitable solution for our specific problem.

```python
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
```

```python
# Decision Tree
decision_tree = DecisionTreeClassifier()

# Random Forest
random_forest = RandomForestClassifier(n_estimators=100)
```

- Here we are creating a **RandomForestClassifier** with 100 trees in the forest.
- The large number of trees will generally lead to better performance but also increase the training time.

- Now we will check the score of the Decision Tree model.
- The Random Forest classifier has slightly a better result over the Decision Tree Classifier.

```
# Printing the scores of the both classifiers
print("Decision Tree: ", round((decision_tree_score),4))
print("Random Forest: ", round((random_forest_score),4))
```

```
Decision Tree:  99.9263
Random Forest:  99.9625
```

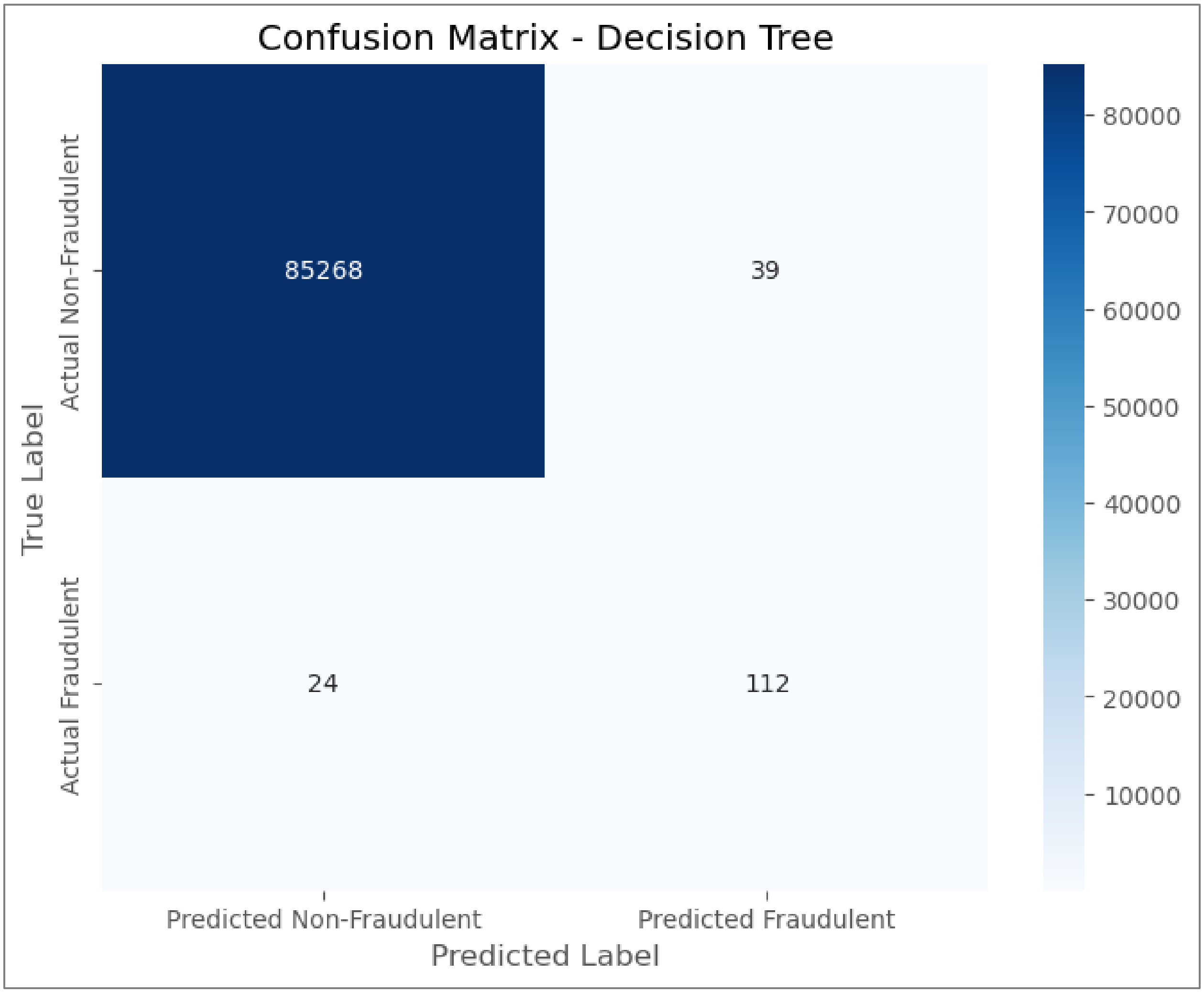
- Evaluation of Decision Tree Model

```
Evaluation of Decision Tree Model:
Accuracy: 0.9993
Precision: 0.7417
recall_score: 0.8235
F1-Score: 0.7805
```

- Evaluation of Random Forest Model

```
Evaluation of Random Forest Model:
Accuracy: 0.9996
Precision: 0.9262
recall_score: 0.8309
F1-Score: 0.876
```

Confusion Matrix - Decision Tree



We understand from the confusion matrix (Decision Tree):

Non-Fraudulent transactions:

1. Correctly predicted as non-fraudulent (True Negative): 85268 transactions.
2. Incorrectly predicted as fraudulent (False Positive): 39 transactions.

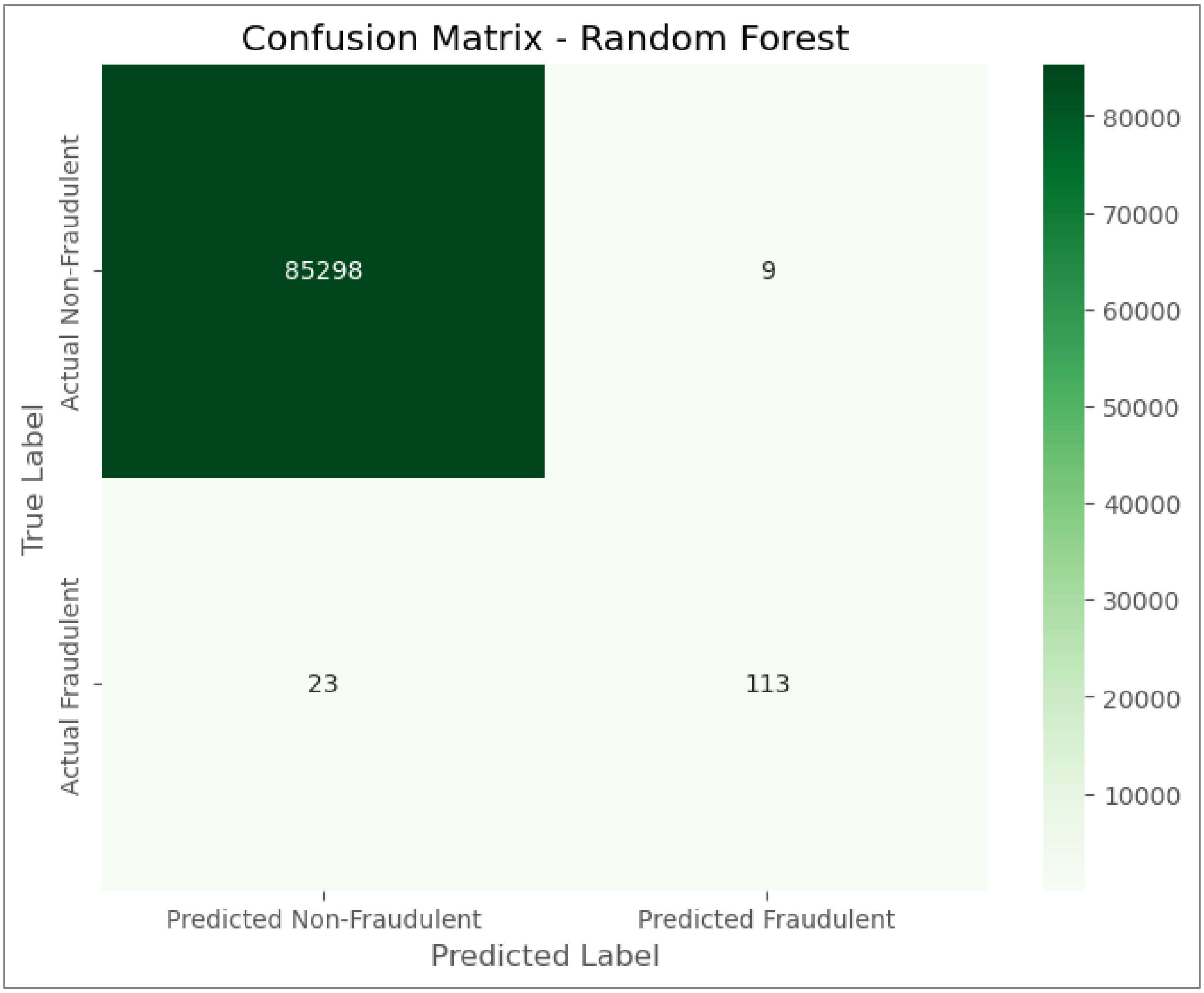
Fraudulent Transactions:

1. Incorrectly predicted as non-fraudulent (False Negative): 24 transactions
2. Correctly predicted as fraudulent (True Positive): 112 transactions

In-short summary:

- The model correctly identified 112 fraudulent transactions.
- It incorrectly identified 24 transactions as non-fraudulent.
- It correctly identified 85268 non-fraudulent transactions.
- It incorrectly identified 39 non-fraudulent transactions as fraudulent.

Confusion Matrix - Random Forest



We understand from the confusion matrix (Random Forest):

Non-Fraudulent transactions:

1. Correctly predicted as non-fraudulent (True Negative): 85298 transactions.
2. Incorrectly predicted as fraudulent (False Positive): 9 transactions.

Fraudulent Transactions:

1. Incorrectly predicted as non-fraudulent (False Negative): 23 transactions
2. Correctly predicted as fraudulent (True Positive): 113 transactions

In-short summary:

- The model correctly identified 113 fraudulent transactions.
- It incorrectly identified 23 transactions as non-fraudulent.
- It correctly identified 85298 non-fraudulent transactions.
- It incorrectly identified only 9 non-fraudulent transactions as fraudulent.

- We will use the SMOT (Synthetic Minority Oversampling Technique, or SMOTE)
- It is the method of data augumentation for the minority class.

```
# We will use the SMOT (Synthetic Minority Oversampling Technique, or SMOTE)
# It is the method of data augumentation for the minority class.

from imblearn.over_sampling import SMOTE

X_resampled, Y_resampled = SMOTE().fit_resample(X,Y)

print("Resampled shape of X: ",X_resampled.shape)
print("Resampled shape of Y: ",Y_resampled.shape)
```

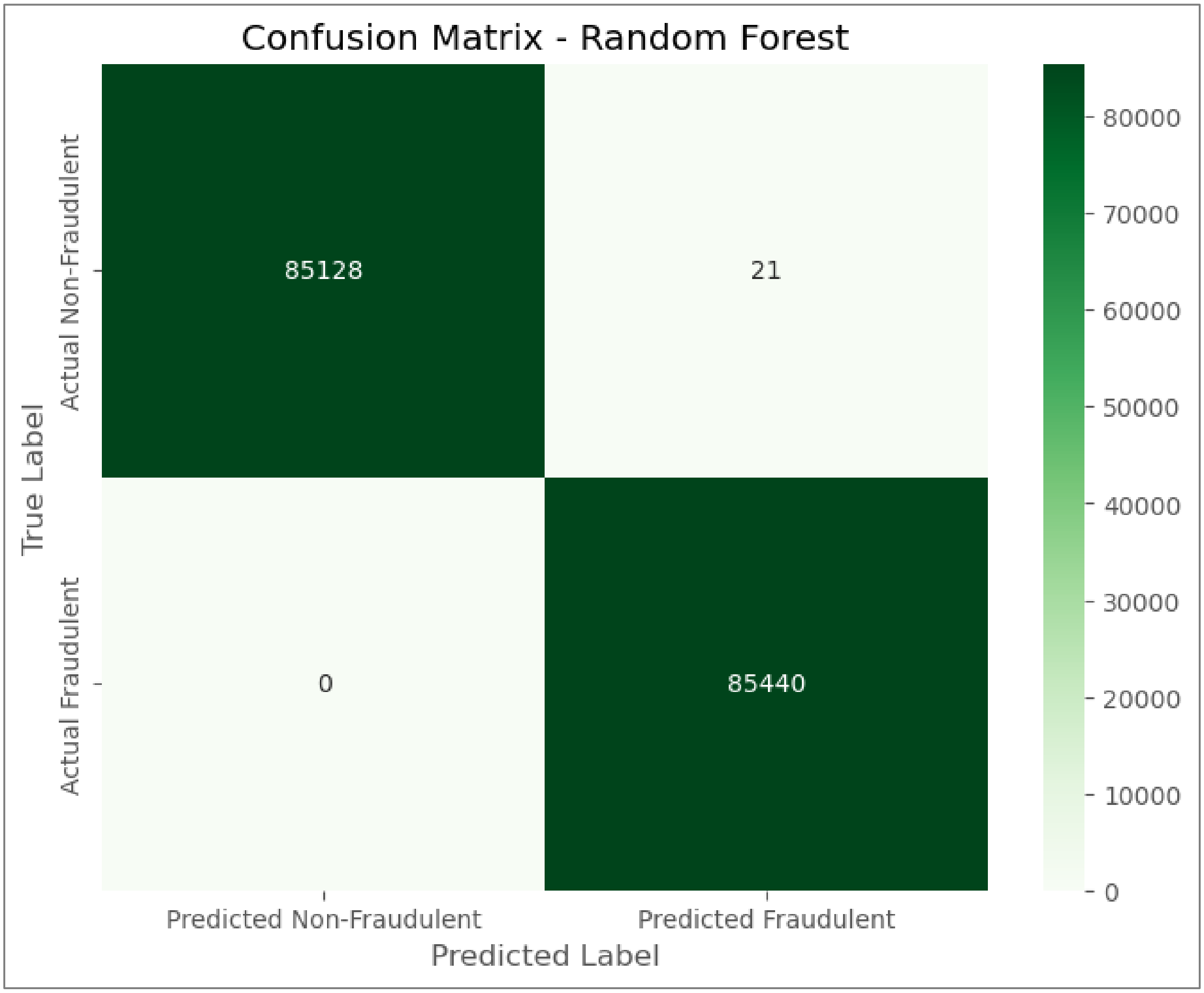
```
Resampled shape of X:  (568630, 29)
Resampled shape of Y:  (568630,)
```

```
predictions_resampled = rf_resampled.predict(test_X)
random_forest_score_resampled = rf_resampled.score(test_X, test_Y) * 100
```

```
print(random_forest_score_resampled)
```

```
99.98768971035648
```

Confusion matrix - Random Forest for resampled data



We understand from the confusion matrix (Random Forest):

Non-Fraudulent transactions:

1. Correctly predicted as non-fraudulent (True Negative): 85128 transactions.
2. Incorrectly predicted as fraudulent (False Positive): 21 transactions.

Fraudulent Transactions:

1. Incorrectly predicted as non-fraudulent (False Negative): 0 transaction.
2. Correctly predicted as fraudulent (True Positive): 85440 transactions.

We understand from the confusion Matrix is:

- The model correctly identified 85440 fraudulent transactions.
- It incorrectly identified 0 transactions as non-fraudulent.
- It correctly identified 85128 non-fraudulent transactions.
- It incorrectly identified only 21 non-fraudulent transactions as fraudulent.

```
Evaluation of Resampled Random Forest Model:
Accuracy: 0.9999
Precision: 0.9998
recall_score: 1.0
F1-Score: 0.9999
```

- We can see that our model performed much better than the previous Random Forest classifier without oversampling.
- We have applied the techniques to address the class imbalance issues and achieved an accuracy of more than 99%.

- We will import the pickle to dump the data frame and the model for the model deployment as the future scope.

```
import pickle
pickle.dump(df,open('df.pkl','wb'))
pickle.dump(rf_resampled,open('rf_resampled.pkl','wb'))
```