Credit Card Fraud Detection

By Charmiece, Princy, Afreen, Prachi

Reason for selecting the topic

Using Python and Machine learning, we plan to detect credit card fraud. As eCommerce sales rise, payment fraud continues to plague customers and merchants. We all have been targeted or known someone whose been targeted. It'll be interesting to know how the companies detect when this happens. Also to try and see how credit card companies detect fraudulent websites. Online fraud has widespread business impacts and requires an effective end-to-end strategy to prevent account takeover (ATO), deter new account fraud, and stop suspicious payment transactions.

Description of the source of data

- fraudTest.csv
- fraudTrain.csv
- https://www.kaggle.com/code/chethanbr86/credit-card-fraud-capstone/data
- This is a simulated credit card transaction dataset containing legitimate and fraud transactions from the duration 1st Jan 2019 - 31st Dec 2020. It covers credit cards of 1000 customers doing transactions with a pool of 800 merchants. This was generated using Sparkov Data Generation | Github tool created by Brandon Harris. This simulation was run for the duration - 1 Jan 2019 to 31 Dec 2020. The files were combined and converted into a standard format.

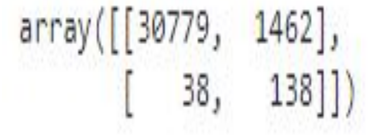
Questions we hope to answer with the data

1. When should the credit card companies shut off a card when it detects fraud?
2. What we are trying to accomplish through this data?
3. Whiich age group are targeted by credit card fraud?
4. Is women targeted more than man?
5. What locations frauds occur?
6. What areas do the company need to pay attention to in order to catch the detection?
7. what machine learning works the best?

Random Oversampling

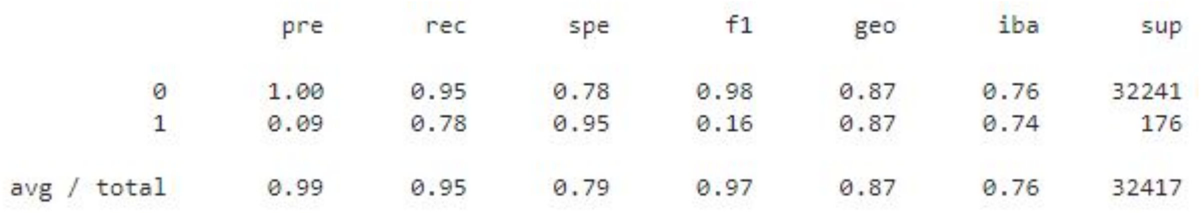
The accuracy score was 87%. It matched with SMOTE and undersampling. It was a good model to run.

Confusion Matrix



```
array([[30779,  1462],
       [   38,   138]])
```

Classification Report

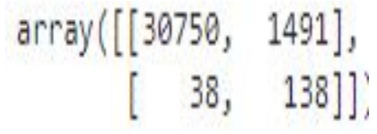


```
                   pre       rec       spe        f1       geo       iba       sup

          0       1.00      0.95      0.78      0.98      0.87      0.76     32241
          1       0.09      0.78      0.95      0.16      0.87      0.74       176

avg / total       0.99      0.95      0.79      0.97      0.87      0.76     32417
```

SMOTE Oversampling

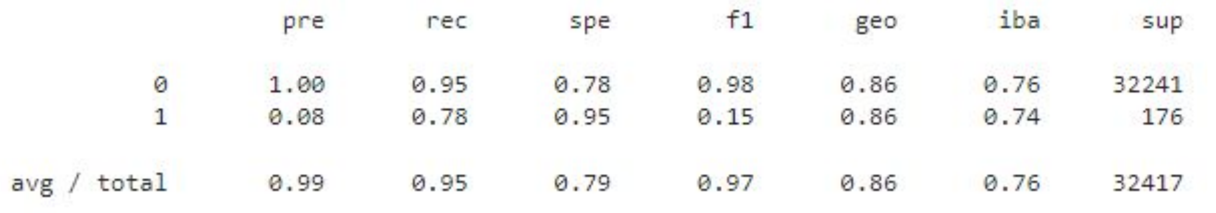
The accuracy score was 87%. It matched with Random oversampling and undersampling. It was a good model to run.

Confusion Matrix



```
array([[30750,  1491],
       [   38,   138]])
```

Classification Report

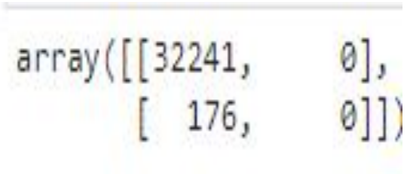


```
                   pre       rec       spe        f1       geo       iba       sup

          0       1.00      0.95      0.78      0.98      0.86      0.76     32241
          1       0.08      0.78      0.95      0.15      0.86      0.74       176

avg / total       0.99      0.95      0.79      0.97      0.86      0.76     32417
```

Undersampling

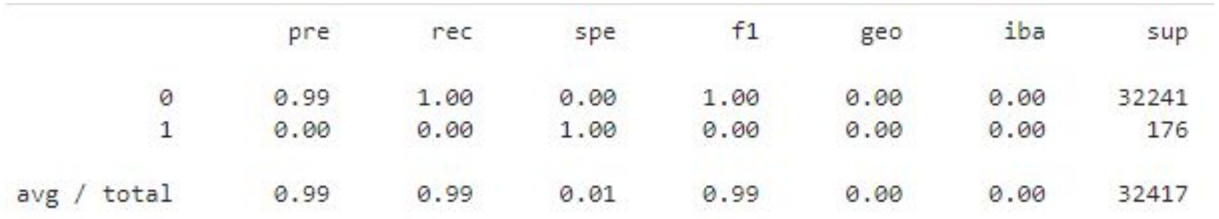
The accuracy score was 87%. It matched with SMOTE and Random Oversampling. It was a good model to run.

Confusion Matrix



```
array([[32241,     0],
       [  176,     0]])
```

Classification Report

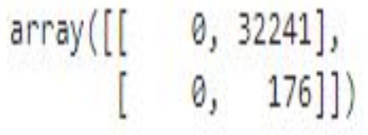


```
                   pre       rec       spe        f1       geo       iba       sup

          0       0.99      1.00      0.00      1.00      0.00      0.00     32241
          1       0.00      0.00      1.00      0.00      0.00      0.00       176

avg / total       0.99      0.99      0.01      0.99      0.00      0.00     32417
```

Combination (Over and Under) Sampling

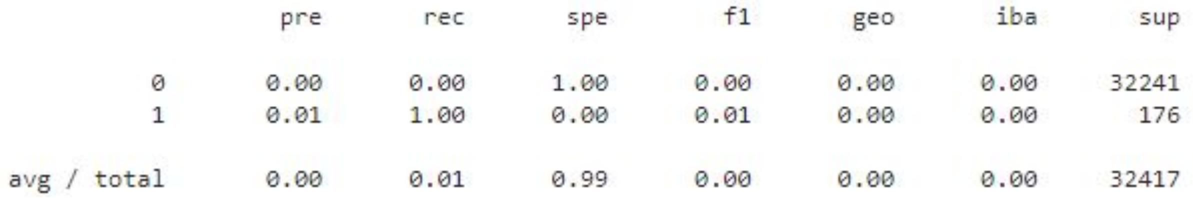
The accuracy score was 50%.

Confusion Matrix



```
array([[    0, 32241],
       [    0,   176]])
```

Classification Report

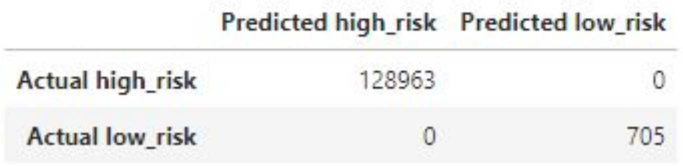


```
                   pre       rec       spe        f1       geo       iba       sup

          0       0.00      0.00      1.00      0.00      0.00      0.00     32241
          1       0.01      1.00      0.00      0.01      0.00      0.00       176

avg / total       0.00      0.01      0.99      0.00      0.00      0.00     32417
```

Balanced Random Forest Classifier

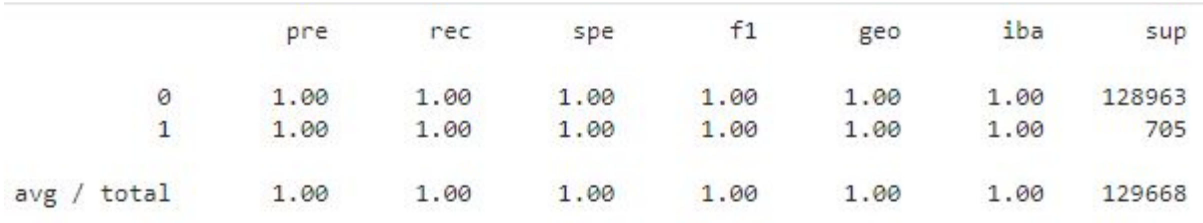
After cleaning the data we put the data to aws and connected it to google colab notebook and performed balanced random forest classifier machine learning model. The accuracy score was 100%.

Confusion matrix



	Predicted high_risk	Predicted low_risk
Actual high_risk	128963	0
Actual low_risk	0	705

Classification Report



```
                   pre       rec       spe        f1       geo       iba       sup

          0       1.00      1.00      1.00      1.00      1.00      1.00    128963
          1       1.00      1.00      1.00      1.00      1.00      1.00       705

avg / total       1.00      1.00      1.00      1.00      1.00      1.00    129668
```

Summary

The Balanced Forest and the Combination were not good models because of the accuracy scores (100% and 50% respectively). Even though the SMOTE, Undersampling and Random Oversampling all had 87% accuracy scores, we would recommend SMOTE because it predicts both high and low risks as opposed to undersmapling and combination.