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.

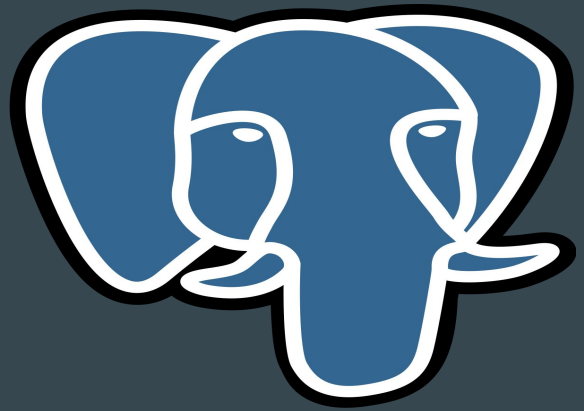
Description of the source of data

- 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.

Description of the data exploration

- We used fraud train csv to make a clean data csv to perform machine learning models.
- For the clean csv, we made an age column from the date of birth column using lambda function and then we took the index column, merchant column, merchant latitude and merchant longitude columns, amount column, category, unix time, is_fraud and gender columns.
- From this we performed used multiple machine learning preprocessing models for Logistic Regression and Balanced Random Forest Classifier model.
- We also Extracted, Transformed and Loaded the data from AWS to pgadmin.

Tools we used

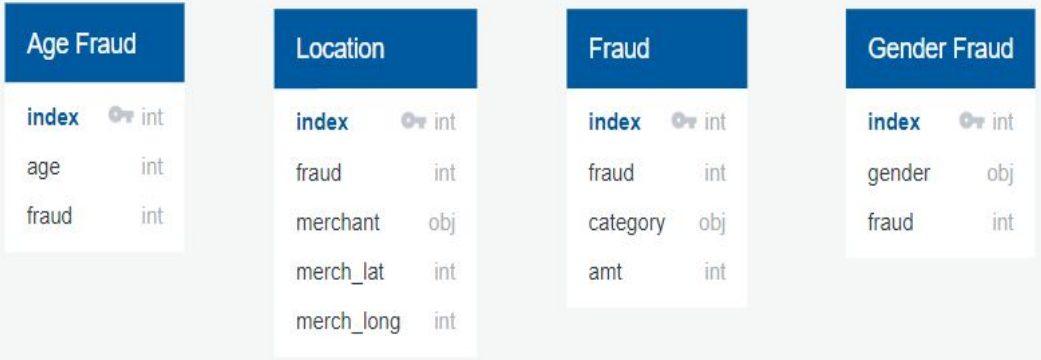


Questions we hope to answer with the data

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. Which age group are targeted by credit card fraud?
4. Are women targeted more than men?
5. What locations do 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?

ERD Diagram

```
"Age Fraud"
-
index int pk
age int
fraud int



"Gender Fraud"
-
index int pk
gender obj
fraud int

"Location"
-
index int pk
fraud int
merchant obj
merch_lat int
merch_long int


"Fraud"
-
index int pk
fraud int
category obj
amt int

```



ETL

Connect to the AWS RDS instance and write each DataFrame to its table.

In [ ]:
```
# Configure settings for RDS
mode = "append"
jdbc_url="jdbc:postgresql://finalproject.cylfu7wfg2zs.us-east-1.rds.amazonaws.com:5432/final_DB"
config = {"user":"postgres",
          "password": "fraudcc1",
          "driver":"org.postgresql.Driver"}
```

In [ ]:
```
# Write age_fraud to table in RDS
age_fraud.write.jdbc(url=jdbc_url, table='age_fraud', mode=mode, properties=config)
```

In [ ]:
```
# Write gender_fraud to table in RDS
gender_fraud.write.jdbc(url=jdbc_url, table='gender_fraud', mode=mode, properties=config)
```

In [ ]:
```
# Write location_info to table in RDS
location_info.write.jdbc(url=jdbc_url, table='location_info', mode=mode, properties=config)
```

In [ ]:
```
# Write fraud_df to table in RDS
fraud_df.write.jdbc(url=jdbc_url, table='fraud_df', mode=mode, properties=config)
```

Logistic Regression Preprocessing : Random Oversampling

- In random oversampling, instances of the minority class are randomly selected and added to the training set until the majority and minority classes are balanced.
- 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

Logistic Regression Pre-Processing :SMOTE Oversampling

- The synthetic minority oversampling technique (SMOTE) is another oversampling approach to deal with unbalanced datasets. In SMOTE, like random oversampling, the size of the minority is increased.
- 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

Logistic Regression Preprocessing :Undersampling

- Undersampling is another technique to address class imbalance. Undersampling takes the opposite approach of oversampling. Instead of increasing the number of the minority class, the size of the majority class is decreased.
- 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

Logistic Regression Preprocessing :Combination (Over and Under) Sampling

- SMOTEENN combines the SMOTE and Edited Nearest Neighbors (ENN) algorithms. SMOTEENN is a two-step process:

1. Oversample the minority class with SMOTE.
2. Clean the resulting data with an undersampling strategy. If the two nearest neighbors of a data point belong to two different classes, that data point is dropped.

- 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

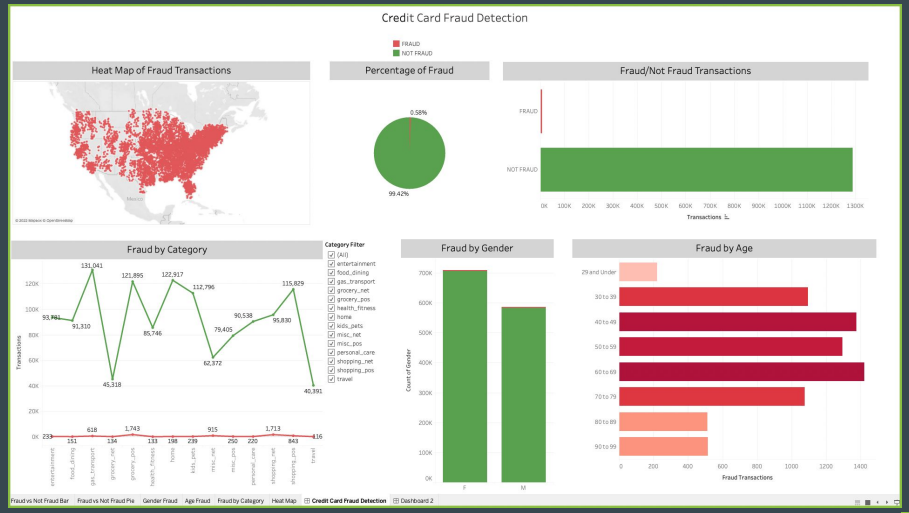
- Random forest algorithm will sample the data and build several smaller, simpler decision trees.
  1. Are robust against overfitting as all of those weak learners are trained on different pieces of the data.
  2. Can be used to rank the importance of input variables in a natural way.
  3. Can handle thousands of input variables without variable deletion.
  4. Are robust to outliers and nonlinear data.
  5. Run efficiently on large datasets.
- 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%.

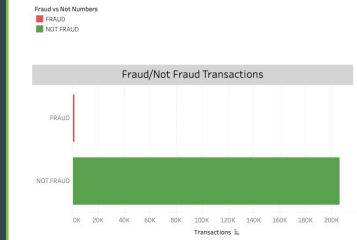
Confusion matrix

	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





SMOTE Oversampling

The synthetic minority oversampling technique (SMOTE) is another oversampling approach to deal with unbalanced datasets. In SMOTE, like random oversampling, size of the minority is increased.

```
In [ ]: # Resample the training data with SMOTE
        from imblearn.over_sampling import SMOTE
        X_resampled, y_resampled = SMOTE(random_state=1,
        sampling_strategy='auto').fit_resample(
            X_train, y_train)
        Counter(y_resampled)

Out[ ]: Counter({0: 96722, 1: 96722})

In [ ]: # Train the Logistic Regression model using the resampled data
        model = LogisticRegression(solver='lbfgs', random_state=1)
        model.fit(X_resampled, y_resampled)

        y_pred = model.predict(X_test)

In [ ]: # Calculated the balanced accuracy score
        from sklearn.metrics import balanced_accuracy_score
        balanced_accuracy_score(y_test, y_pred)

Out[ ]: 0.8689227226202646

In [ ]: # Display the confusion matrix
        from sklearn.metrics import confusion_matrix
        y_pred = model.predict(X_test)
        confusion_matrix(y_test, y_pred)

Out[ ]: array([[30750,  1491],
               [   38,   138]])

In [ ]: # Print the imbalanced classification report
        print(classification_report_imbalanced(y_test, y_pred))
```

	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

Logistic Regression with SMOTE preprocessing was our most accurate model. However, even with a smaller sample we took to run the model is fraud prediction was way higher at 8% compared to actual is fraud in the actual dataset at 0.58%. Logistic Regression would not be a good model for credit card fraud detection.

Summary

- Our goal was to find a machine learning model that helped detect when credit card companies should shut off credit cards due to fraudulent activity. We use Balance Random Forest Classifier which gave us a 50% accuracy. This was not a good model due to the data showing less than 1% of the transactions being fraud. The second model we used was Logistic Regression. We preformed SMOTE, Combination (SMOTEEN), Undersampling and Random Oversampler. Logistic Regression with SMOTE preprocessing was our most accurate model. However, even with a smaller sample we took to run the model is fraud prediction was way higher at 8% compared to actual is fraud in the actual dataset at 0.58%. Logistic Regression would not be a good model for credit card fraud detection.
- From the analysis, ages 40 to 70 were the most targeted for fraud. Based on the number of transactions, men were targeted more than women for fraud. According to our dataset, more fraud transactions occurred on the east coast versus the west coast. The areas that credit card companies should focus on are grocery stores and online shopping.