



Credit Card Fraud Detection

By: Afreen, Charmiece, Prachi, Princy

Introduction

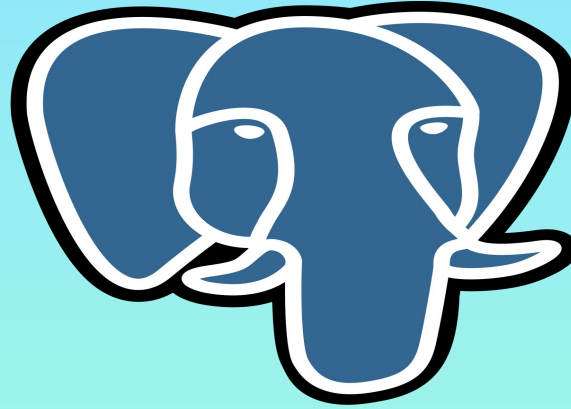
- Fraudulent transactions can occur at anytime whether it is online or offline.
- Online fraud has widespread business impacts and requires an effective end-to-end strategy to prevent account takeover (ATO), deter new account fraud, and stop suspicious payment transactions.
- Through machine learning we are trying to increase the accuracy of detecting fraudulent 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.

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?

Description of the Data Exploration

```
# function to convert dob to years
def age_years(born):
    return 2022 - int(born[0:4])

# replacing the dob column with age column in our data set for test and train
data_train['age'] = data_train['dob'].apply(lambda x: age_years(x))
data_train.head()
```

y	amt	first	last	gender	street	...	long	city_pop	job	dob	trans_num	unix_time	merch_lat	merch_long	is_fraud	age
it	4.97	Jennifer	Banks	F	561 Perry Cove	...	-81.1781	3495	Psychologist, counselling	1988-03-09	0b242abb623afc578575680df90655b9	1325376018	36.011293	-82.048315	0	34
s	107.23	Stephanie	Gill	F	41019 Riley Greens Suite 393	...	-118.2105	149	Special educational needs teacher	1978-06-21	1f76529f8574734946361c461b024d99	1325376044	49.159047	-118.186462	0	44
it	220.11	Edward	Sanchez	M	394 White Dale Suite 530	...	-112.2620	4154	Nature conservation officer	1962-01-19	a1a22d70485983eac12b5b88dad1cf95	1325376051	43.150704	-112.154481	0	60
it	45.00	Jeremy	White	M	9443 Cynthia Court Apt. 038	...	-112.1138	1939	Patent attorney	1967-01-12	6b849c168bdad6f867558c3793159a81	1325376076	47.034331	-112.561071	0	55
s	41.96	Tyler	Garcia	M	408 Bradley Rest	...	-79.4629	99	Dance movement psychotherapist	1986-03-28	a41d7549acf90789359a9aa5346dcb46	1325376186	38.674999	-78.632459	0	36

```
clean_train = data_train[['Unnamed: 0', 'merchant', 'category', 'amt', 'gender', 'unix_time', 'merch_lat', 'merch_long', 'is_fraud', 'age']]
clean_train.head()
```

	Unnamed: 0	merchant	category	amt	gender	unix_time	merch_lat	merch_long	is_fraud	age
0	0	fraud_Rippin, Kub and Mann	misc_net	4.97	F	1325376018	36.011293	-82.048315	0	34
1	1	fraud_Heller, Gutmann and Zieme	grocery_pos	107.23	F	1325376044	49.159047	-118.186462	0	44
2	2	fraud_Lind-Buckridge	entertainment	220.11	M	1325376051	43.150704	-112.154481	0	60
3	3	fraud_Kutch, Hermiston and Farrell	gas_transport	45.00	M	1325376076	47.034331	-112.561071	0	55
4	4	fraud_Keeling-Crist	misc_pos	41.96	M	1325376186	38.674999	-78.632459	0	36

```
# 5. Export the Dataframe as a new CSV file without the index.
clean_train.to_csv("Resources/clean_data_train.csv", index=False)
```

ERD Diagram

```
1  "Age Fraud"
2  -
3  index int pk
4  age int
5  fraud int
6
7
8
9  "Gender Fraud"
10 -
11 index int pk
12 gender obj
13 fraud int
14
15 "Location"
16 -
17 index int pk
18 fraud int
19 merchant obj
20 merch_lat int
21 merch_long int
22
23
24 "Fraud"
25 -
26 index int pk
27 fraud int
28 category obj
29 amt int
30
```

Age Fraud

index	int
age	int
fraud	int

Location

index	int
fraud	int
merchant	obj
merch_lat	int
merch_long	int

Fraud

index	int
fraud	int
category	obj
amt	int

Gender Fraud

index	int
gender	obj
fraud	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

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=1, stratify=y)
X_train.shape
```

(97251, 715)

```
# Resample the training data with the RandomOverSampler
from imblearn.over_sampling import RandomOverSampler
ros = RandomOverSampler(random_state=1)
X_resampled, y_resampled = ros.fit_resample(X, y)
Counter(y_resampled)
```

Counter({0: 128963, 1: 128963})

```
# Train the model using the data
from sklearn.linear_model import LogisticRegression
model = LogisticRegression(solver='lbfgs', random_state=1)
model.fit(X_resampled, y_resampled)
```

LogisticRegression(random_state=1)

The accuracy score was 87%.

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 Preprocessing: SMOTE Oversampling

```
# 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)
```

```
Counter({0: 96722, 1: 96722})
```

```
# 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)
```

```
# Calculated the balanced accuracy score
from sklearn.metrics import balanced_accuracy_score
balanced_accuracy_score(y_test, y_pred)
```

```
0.868922722620266
```

The accuracy score was 87%.

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

```
# Resample the data using the ClusterCentroids resampler
# Warning: This is a large dataset, and this step may take some time to complete
from imblearn.under_sampling import ClusterCentroids
cc = ClusterCentroids(random_state=1)
X_resampled, y_resampled = cc.fit_resample(X_train, y_train)
```

```
# Train the Logistic Regression model using the resampled data
from sklearn.linear_model import LogisticRegression
model = LogisticRegression(solver='lbfgs', random_state=1)
model.fit(X_resampled, y_resampled)
```

```
LogisticRegression(random_state=1)
```

```
# Calculated the balanced accuracy score
from sklearn.metrics import balanced_accuracy_score
balanced_accuracy_score(y_test, y_pred)
```

```
0.868922722620266
```

The accuracy score was 87%.

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

```
# Resample the training data with SMOTEENN
# Warning: This is a large dataset, and this step may take some time to complete
from imblearn.combine import SMOTEENN
smote_enn = SMOTEENN(random_state=0)
X_resampled, y_resampled = smote_enn.fit_resample(X, y)
```

```
# Train the Logistic Regression model using the resampled data
from sklearn.linear_model import LogisticRegression
model = LogisticRegression(solver='lbfgs', random_state=1)
model.fit(X_resampled, y_resampled)
```

```
LogisticRegression(random_state=1)
```

```
# Calculated the balanced accuracy score
from sklearn.metrics import balanced_accuracy_score
balanced_accuracy_score(y_test, y_pred)
```

```
0.5
```

The accuracy score was 50%.

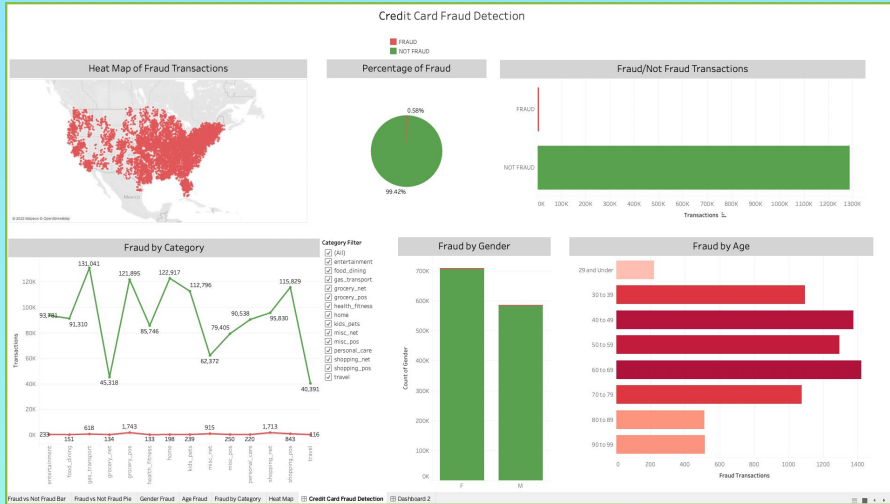
Confusion Matrix

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

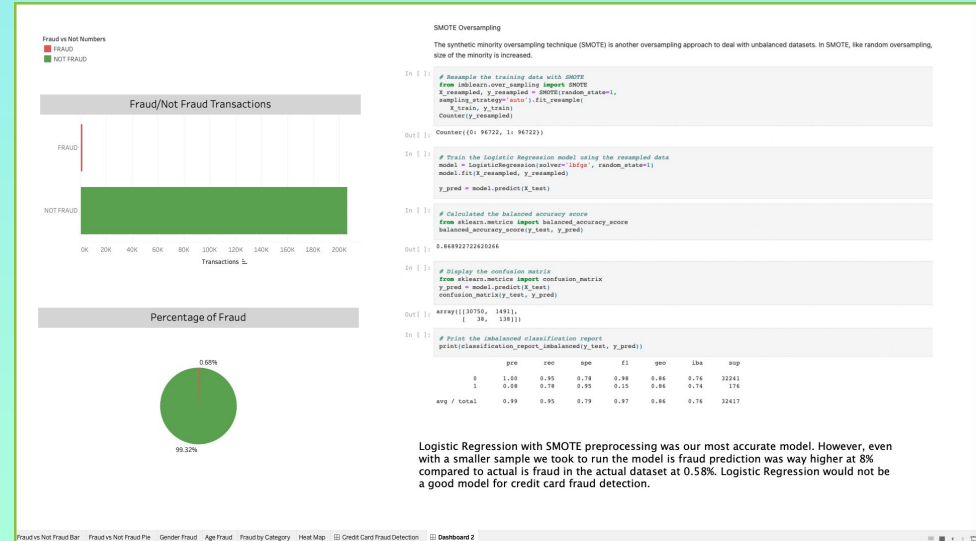
Classification Report

	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

Dashboard



[Link to Dashboard](#)



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 helps detect when credit card companies should shut off (decline) credit cards due to fraudulent activity. We used 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 performed SMOTE, Combination (SMOTEEN), Undersampling, and Random Oversampler. Logistic Regression with SMOTE preprocessing was our most accurate model. However, even with the smaller sample we took to run the program, the model “Is Fraud” prediction was much higher at 8% compared to the actual “Is Fraud” percentage in the given dataset at 0.58%.
- 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. 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.