Data Science

Project Protocol

Credit Card Fraud Detection in Washington DC

By

Rachel Shriki

Introduction

Credit card fraud is one of the most common types of financial crime in today's digital world. With the expansion of e-commerce, the increase in online transfers and the high accessibility of digital payment methods – new opportunities have also arisen for fraudsters.

In essence, fraud involves an unauthorized party using another person's credit card, without their permission, to make purchases or transfer money. These actions may be carried out in various ways – from stealing card details (such as through phishing sites), using a stolen physical card, to sophisticated attacks that involve analyzing transaction data and identifying weaknesses in payment systems.

The extent of the economic damage caused requires credit companies and financial institutions to invest significant resources in fraud prevention. These systems are based on analyzing behavioral patterns, identifying anomalies, and machine learning that attempts to detect unusual transactions in real time.

The goal of this project was to examine transaction data, perform preliminary analysis (EDA), identify imbalances between fraudulent and legitimate examples, and build predictive models capable of classifying whether a transaction is fraudulent or legitimate, using imbalance management techniques, machine learning, and model performance evaluation.

Data

The data contains 2 csv files:
Transactions dataset that contained approximately 34 million records
and Customers dataset with 20,000 records.

In order to enable in-depth analysis within the scope of the work, while maintaining reasonable processing times and computational efficiency, the data was reduced and a representative subset was selected.

After examining the rate of fraud by year and geographic location, it was found that in 2019 the Washington region received the highest percentage of fraud compared to other regions.

Therefore, a focus was placed on analyzing transactions from this period and region, with the aim of examining distinct fraud patterns and improving the performance of models for identifying suspicious transactions.

This approach allows for:

- Focusing on areas where there is significant fraudulent activity.
- Improving the accuracy of models in identifying fraud patterns.
- Maintaining computational efficiency during data processing and analysis.

```
select YEAR(trans_date),
sum(case when is_fraud=1 then 1 else 0 end) as fraud,
sum(case when is_fraud=0 then 1 else 0 end) as not_fraud,
count(*) as cnt,
100.0*sum(case when is_fraud=1 then 1 else 0 end)/count(*) as fraud_rate,
state
from Transactions_Customers
group by  YEAR(trans_date), state
order by fraud_rate desc
```

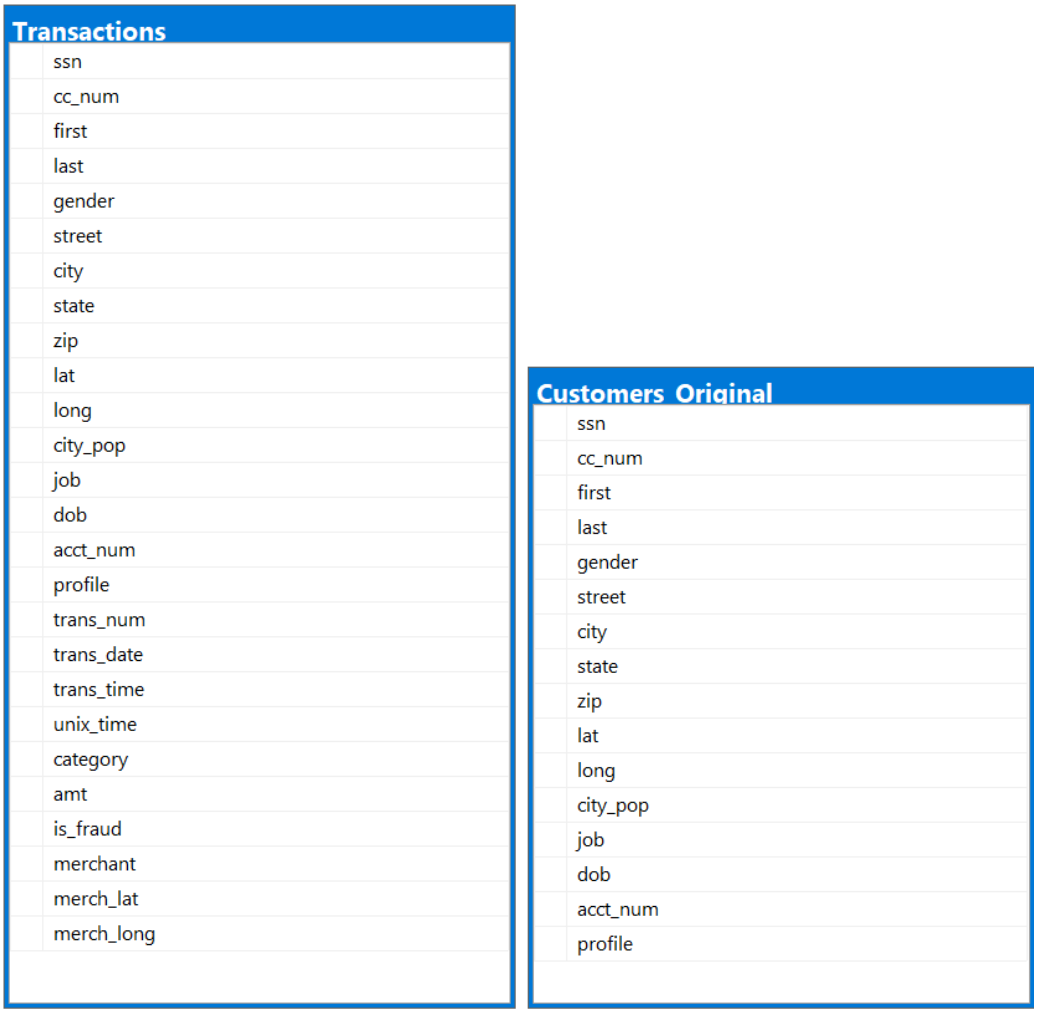
00 % No issues found

Results Messages

	(No column name)	fraud	not_fraud	cnt	fraud_rate	state
1	2019	224	28986	29210	0.766860664156	DC
2	2020	387	53103	53490	0.723499719573	RI
3	2019	193	26576	26769	0.720983226866	WY
4	2019	968	137699	138667	0.698075245011	KS
5	2020	1322	203380	204702	0.645816845951	OR
6	2019	931	143612	144543	0.644098987844	MS
7	2020	250	38793	39043	0.640319647568	ND
8	2019	250	39103	39353	0.635275582547	ND
9	2019	1912	303992	305904	0.625032689994	MN
10	2020	1074	171752	172826	0.621434274935	IA
11	2019	1883	302697	304580	0.618228380064	WI
12	2019	380	61964	62344	0.609521365327	DE
13	2020	1453	240197	241650	0.601282847092	AL
14	2019	2410	398638	401048	0.600925574993	AZ
15	2020	2140	355084	357224	0.599063892683	IN
16	2019	562	93361	93923	0.598362488421	WV
17	2020	610	101389	101999	0.598045078873	NE
18	2019	3734	622420	626154	0.596338919818	OH

Description of the database structure

The relasionship between the tables using fields: ssn, cc_num.
The data from the customer table is contained in the transactions table and is therefore not used.



Description of the columns that exist in both tables:

- **ssn** – Social Security Number - American identity number
- **cc_num** – Credit card number
- **first, last** – First name and last name.
- **gender** – (Male/Female)
- **street, city, state, zip** – Geographical area information
- **lat, long** – (Latitude/Longitude)
- **city_pop** – The number of residents in the city
- **job** – The client's profession
- **dob** – Date of Birth
- **acct_num** – Bank account number
- **profile** – A description of the customer including age, gender, and more

Description of the columns that exist in Transaction table:

- **trans_num** – Unique transaction identifier
- **trans_date** – Date of transaction
- **trans_time** – Time of transaction
- **unix_time** – Timestamp in unix format (seconds since 1970)
- **category** – Purchase category
- **amt** – Amount
- **is_fraud** – Target value: 1-fruad, 0-non-fraud
- **merchant** – The name of the business where the transaction was made
- **merch_lat, merch_long** – Geographic location of the business

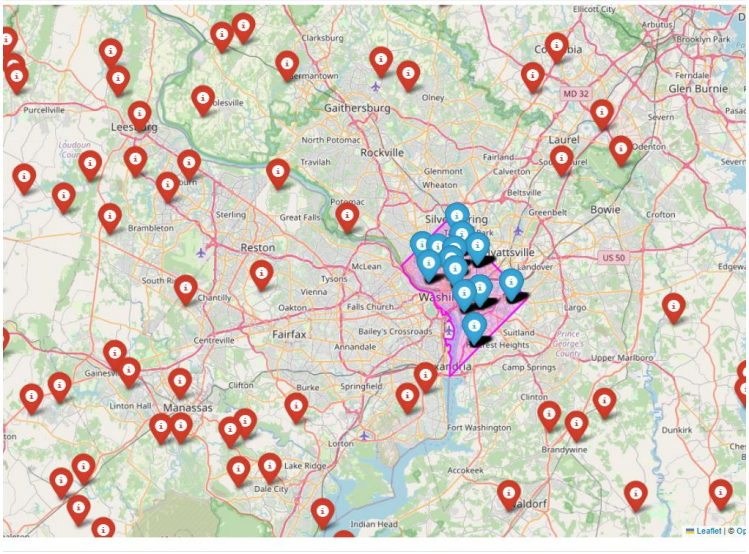
Missing values:
Ther is no missing value in the dataset.

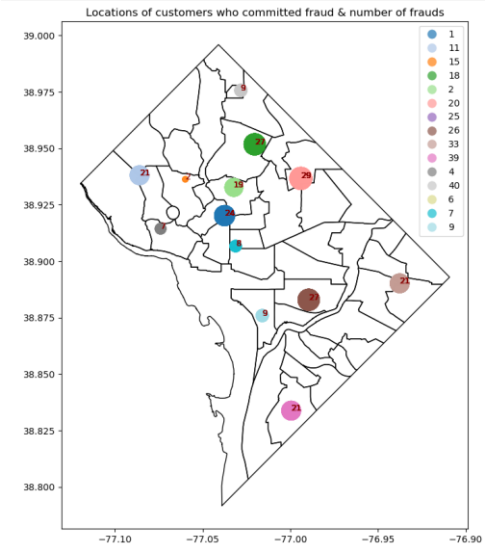
Data Enrichment

I enriched the data with new fields from the data itself, for example
creating a new field for age from the fields: date of birth and the transaction date.
In addition, I enriched the data from an external open government source, in order to add a new field of neighborhood cluster. For this purpose, the Open Data DC database was used, which provides official information.
https://opendata.dc.gov/datasets/neighborhood-clusters/explore

The downloaded file contains geographic coordinates (Latitude / Longitude) organized by neighborhood.
By combining the location data found in the transaction system, the information was loaded into the database, and a match was created between the customer's location and the relevant neighborhood in the city.
During the work, it became clear that while customer addresses could be successfully mapped to neighborhoods within Washington, DC, a similar match could not be made for a significant portion of merchant addresses. Manual inspection using Google Maps showed that many of the merchant landmarks were located outside the city limits, and sometimes even in neighboring states such as Maryland and Pennsylvania.
Therefore, neighborhood classification for merchants was not possible.

A map showing the frauds In Red – merchant location In Blue – customer location	A map showing number of frauds in neighborhood clusters



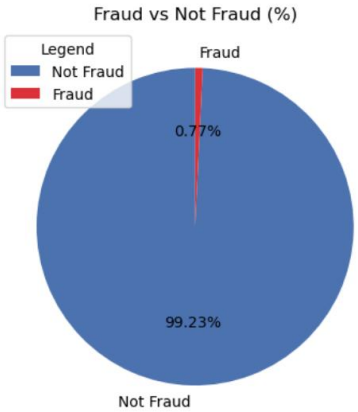


Imbalanced

It was found that the value is_fraud is unbalanced in the dataset.
The fraud rate of all transactions is significantly low (0.77%), so it is necessary to perform class balancing in order to prevent a situation in which the model "learns" to predict only a majority.

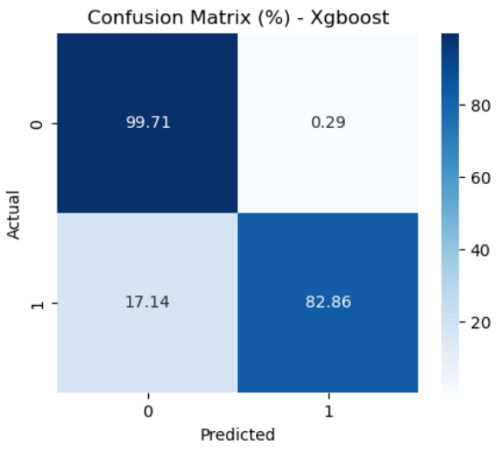
To balance the classes, I used methods from the imblearn library, including:

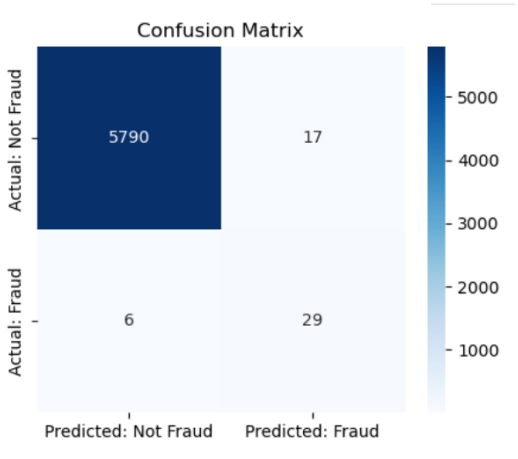
- Random Over Sampling - Increasing minority samples
- Randon Under Sampling - Reducing samples from the majority
- SMOTE - Creating synthetic samples for the minority
- SMOTE_Tomek - Combining SMOTE and removing similar points belonging to different groups



We tested the various methods on the XGBOOST model. After running them, we found that Random Over Sampling (ROS) gave the best score according F1-Score and Recall.

	Model	Balancing	Precision	Recall	F1 Score	Confusion Matrix
1	Xgboost	RandomOverSampler	0.630435	0.828571	0.716049	[[5790, 17], [6, 29]]
0	Xgboost	No Balance	0.718750	0.657143	0.686567	[[5798, 9], [12, 23]]
4	Xgboost	SMOTE_Tomek	0.549020	0.800000	0.651163	[[5784, 23], [7, 28]]
3	Xgboost	SMOTE	0.562500	0.771429	0.650602	[[5786, 21], [8, 27]]
2	Xgboost	RandomUnderSampler	0.084399	0.942857	0.154930	[[5449, 358], [2, 33]]





Model Selection

While modeling, I divided the into Train and Test, a confrontation of:
Train – 80%, Test – 20%.

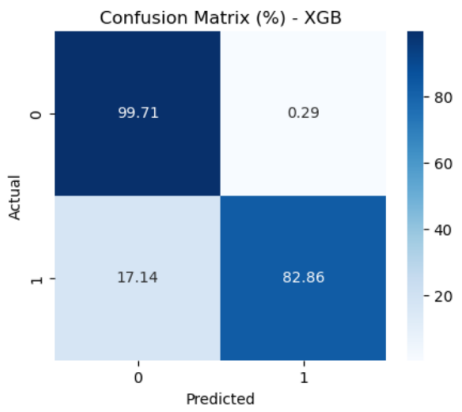
The models that were run, which also included hyperparameter tuning, were:

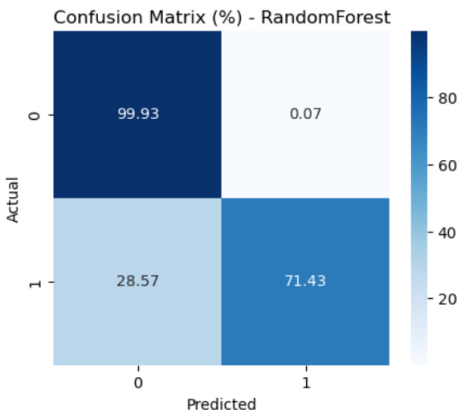
- Decision Tree
- Random Forest
- AdaBoost
- Gradient Boosting
- XGBoost
- SVC (Support Vector Classifier)

This are the results:

	Model	Accuracy	Precision	Recall	F1-Score	Confusion Matrix
5	SVC	0.943170	0.069565	0.685714	0.126316	[[5486, 321], [11, 24]]
2	AdaBoost	0.945567	0.094556	0.942857	0.171875	[[5491, 316], [2, 33]]
3	GradientBoosting	0.970045	0.160194	0.942857	0.273859	[[5634, 173], [2, 33]]
0	DecisionTree	0.994180	0.510638	0.685714	0.585366	[[5784, 23], [11, 24]]
4	XGB	0.996063	0.630435	0.828571	0.716049	[[5790, 17], [6, 29]]
1	RandomForest	0.997604	0.862069	0.714286	0.781250	[[5803, 4], [10, 25]]

According Recall and F1-Score XGBoost is better
Percentage confusion matrix :





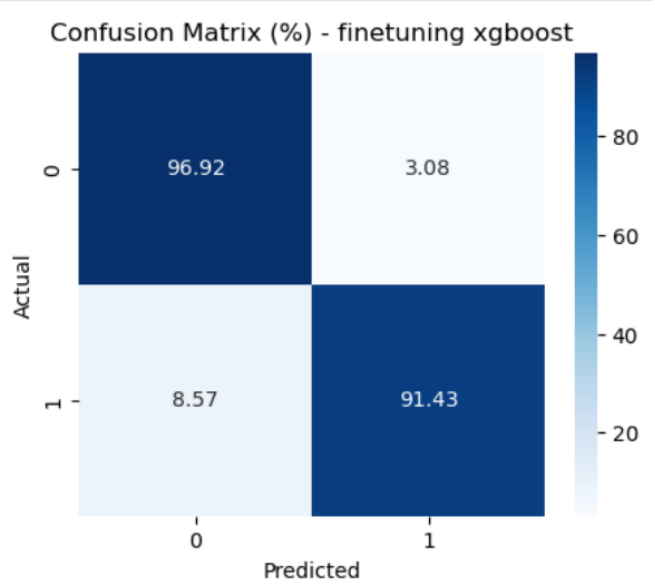
Fine Tuning

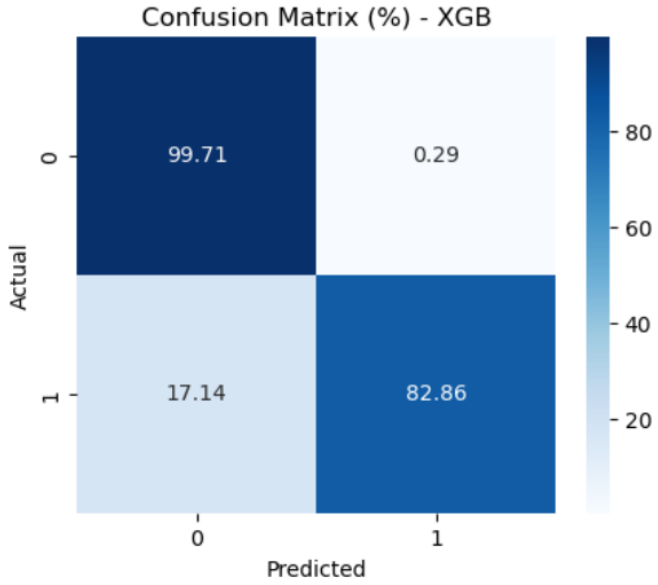
I used GridSearchCV to run a large number of parameter combinations and choose the best one based on performance.

The hyperparamtes I used:

- n_estimators - number of trees
- max_depth - maximum depth
- learning_rate - learning rate
- subsample - percentage of data in each tree
- colsample_bytree - Percentage of features in each tree

After Fine-Tuning	Before Fine-Tuning





Conclusions:

The final model can effectively detect fraudulent behaviour, in real-world transaction environments.

Project Notebooks

The project is written in Python and contain the following notebooks:

1-Data Preparation
2-EDA (Explanatory Data Analysis)
3-Data Cleansing
4-Feature Engineering
5- FeatureSelection
6-Imbalanced data
7-Model Selection

Thank you

Rachel Shriki