Fraud Detection Model for Businesses

Business problem

- Credit card fraud occurs when an unauthorized person gains access to your information and uses it to make purchases. Frauds happen in various settings: Lost or stolen credit cards, Hacking your computer, Phishing attempts, etc. Credit card fraud poses danger to not only consumers, banks, credit card companies but also retailers.
- Credit card fraud detection is a set of methods and techniques designed to block fraudulent purchases, both online and in-store.
- **Motivations**: By initiating this project, we want to explore methods to identify the right cardholder and the legitimate purchases, thus reducing risks for both businesses and individuals.
- **Objectives**: How the usage of machine learning model can help detect patterns associated with Credit card fraud transactions.

https://www.teradata.com/insights/ai-and-machine-learning/fraud-detection-machine-learning

https://seon.io/resources/credit-card-fraud-detection/

https://www.cnbc.com/select/credit-card-fraud/#

https://spd.tech/machine-learning/credit-card-fraud-detection/

Dataset

- To reach the objective we deploy a simulated dataset from Kaggle. The dataset contains legitimate and **fraud credit card transactions** from the duration 1st Jan 2019 - 31st Dec 2020.
- The original dataset consists of over **500,000 observations**, with a significant class imbalance, as only 0.4% of the observations are fraudulent. From our initial assessment, the dataset would require intensive handling of variables, as well as a suitable approach to rebalance the class ratios.

Column Name	Description	Data Type
Unnamed: 0	Index row of the dataset	int64
trans_date_trans_time	Timestamp of the transaction.	object
cc_num	Credit card number (anonymized).	float64
merchant	Name of the merchant where the transaction occurred.	object
category	Category of the merchant (e.g., groceries, electronics).	object
amt	Amount spent in the transaction.	float64
first	First name of the cardholder.	object
last	Last name of the cardholder.	object
gender	Gender of the cardholder.	object
street	Street address of the cardholder.	object
city	City of the cardholder's residence.	object
state	State of the cardholder's residence.	object
zip	Postal code of the cardholder's residence.	int64
lat	Latitude of the cardholder's residence.	float64
long	Longitude of the cardholder's residence.	float64
city_pop	Population of the cardholder's city.	int64
job	Profession of the cardholder.	object
dob	Date of birth of the cardholder.	object
trans_num	Unique transaction identifier.	object
unix_time	Unix timestamp of the transaction.	int64
merch_lat	Latitude of the merchant's location.	float64
merch_long	Longitude of the merchant's location.	float64
is_fraud	Binary indicator of fraud (1 for fraud, 0 for legitimate transactions).	int64

https://www.kaggle.com/datasets/kartik2112/fraud-detection

Questions:

- What are the ML methods to detect credit fraud transactions?
- What is the most suitable methods with the data given?

Dataset Processing

No missing values were found. High data quality.
We addressed class imbalance and reduce dataset size for computational efficiency.

Separate Classes:

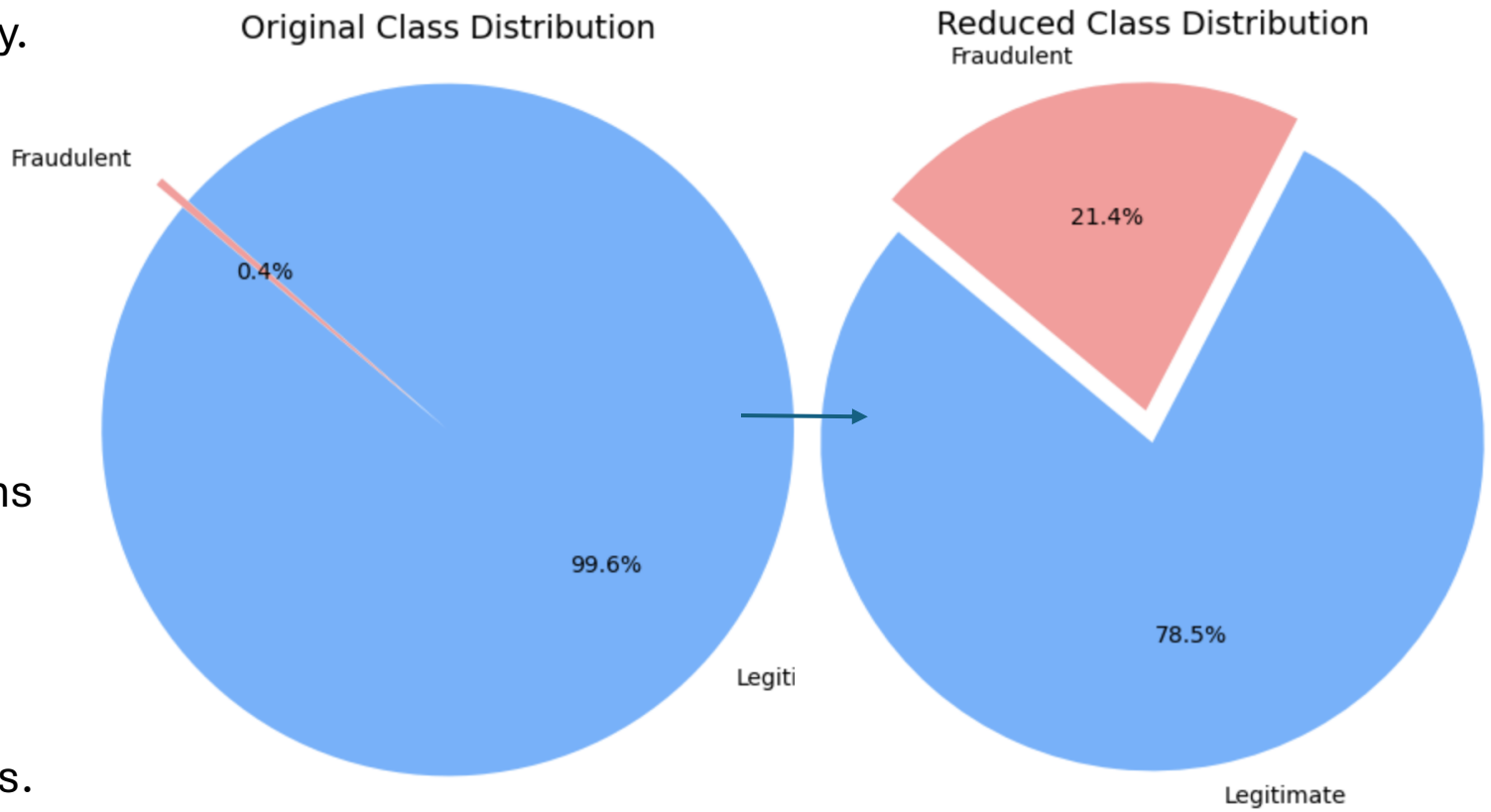
- **Fraud Cases:** All 0.4% (≈2,000)
- **Non-Fraud Cases:** 99.6% (≈498,000)

Sample Non-Fraud Transactions:

- **Target Size:** 10,000 total transactions
- **Non-Fraud Sampled:** 8,000 transactions (assuming 2,000 fraud cases)

Combine and Shuffle:

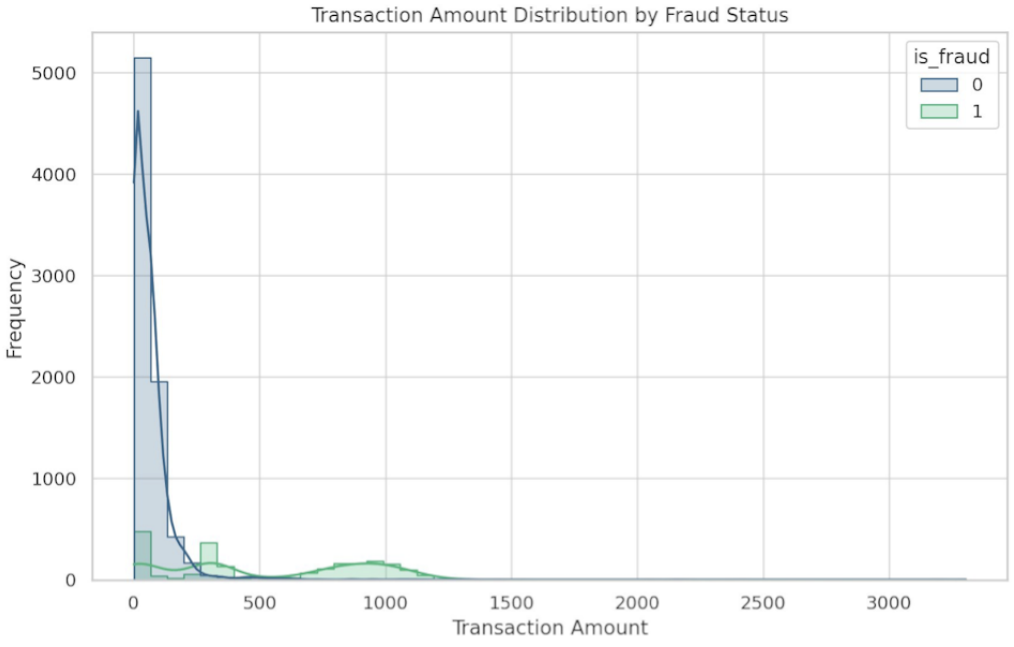
- **Total Reduced Dataset:** 10,000 transactions
- **Class Distribution:** Shown in the figures.

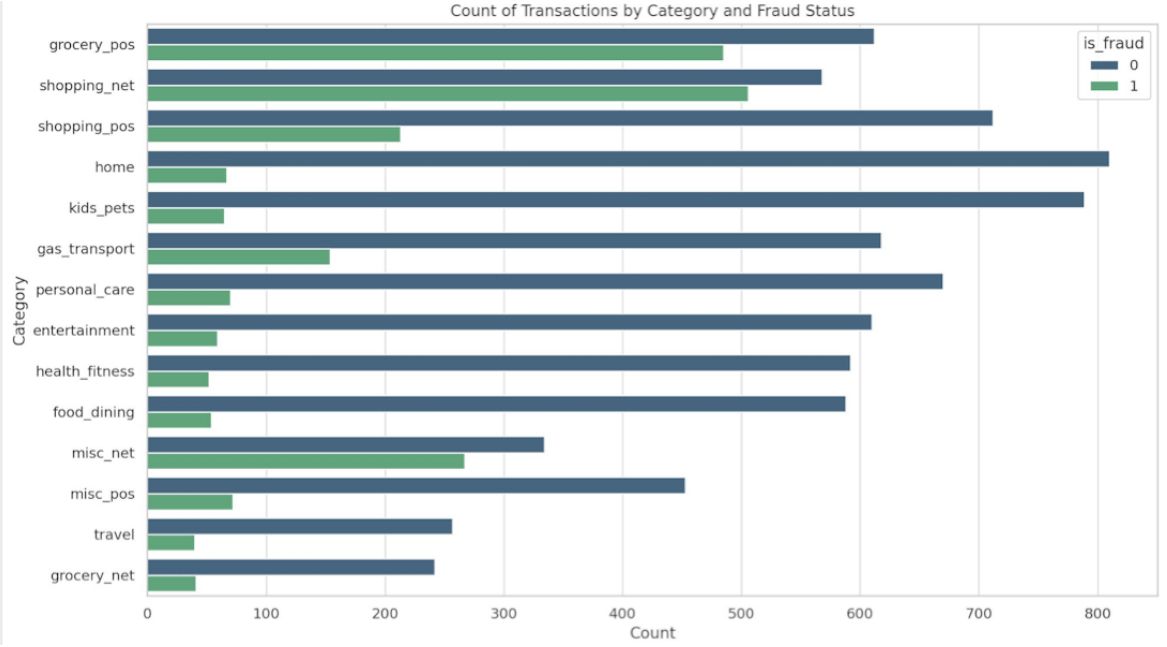


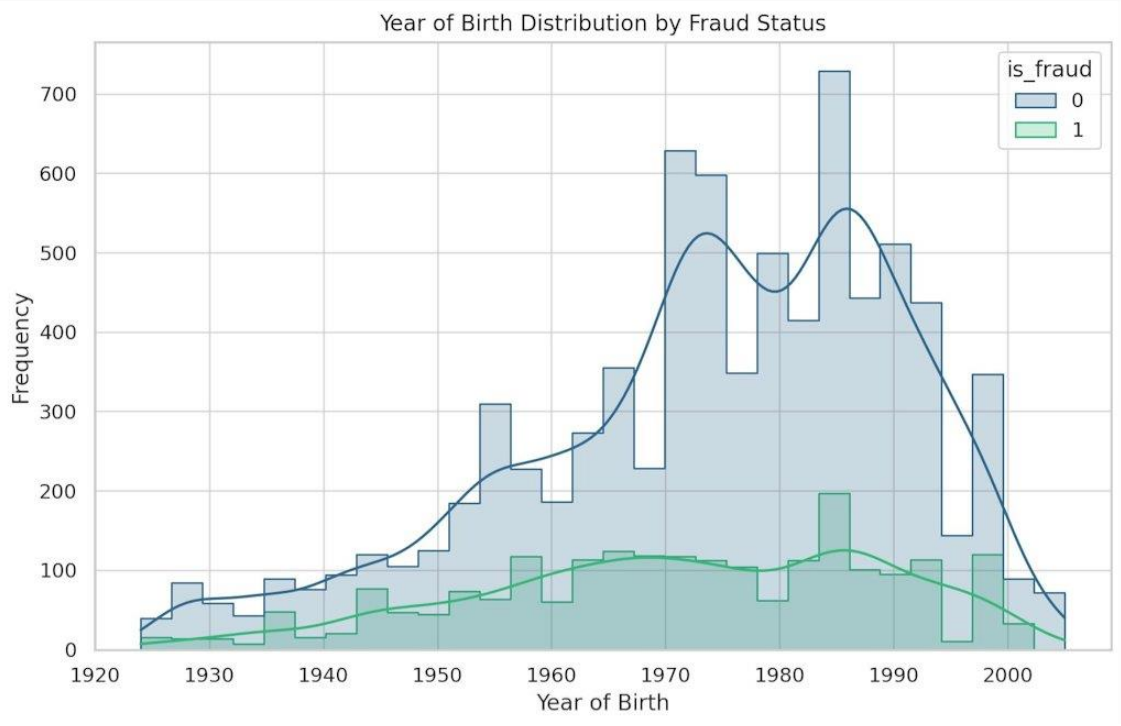
Technique used to handle class imbalance: Random Under-Sampling

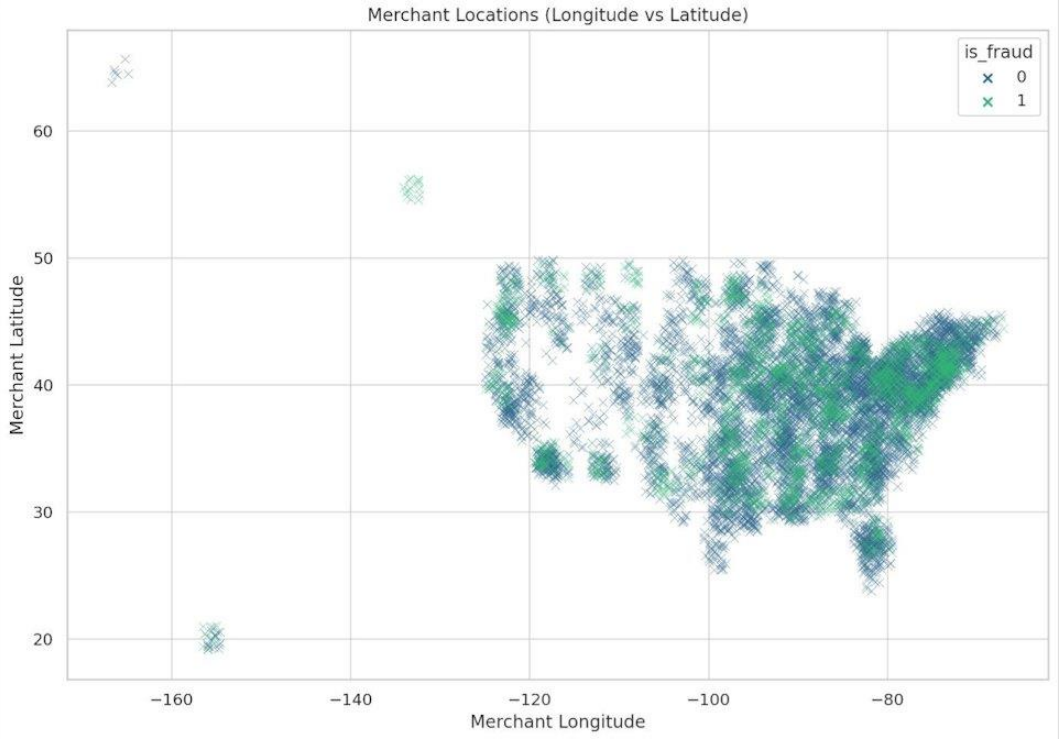
Random Under-Sampling is a method used to address class imbalance by reducing the number of instances in the majority class. In the context of the fraud detection dataset, this technique involves decreasing the number of legitimate (non-fraudulent) transactions to balance the dataset with fraudulent transactions.

Exploratory Data Visualization









Exploratory Data Analysis

Key Statistics

- Median **fraudulent** transaction amount: **€398.75**
- Maximum **fraudulent** transaction amount: **€3,304.44**
- Median non-fraudulent transaction amount: **€50.05**
- Maximum non-fraudulent transaction amount: **€2,545.85**
- Median year of birth for fraudulent vs. otherwise: **1976 vs 1975**
- Top Fraud Categories: **shopping_net, grocery_pos, misc_net**.
- Fraudulent transactions are fairly evenly distributed across age groups, with a slight concentration among younger customers (aged 30-45).
- Merchants located across wide geographical regions, with no distinct clustering. Fraudulent activity occurs both in major cities and rural areas.

Categories that involve online shopping (shopping_net and misc_net) and Point of Sale (POS) transactions (grocery_pos) have the highest rates of fraud, reflecting vulnerability in these sectors.

Key Predictors (deduced from Exploratory Data Analysis):

- **Transaction Amount**: High correlation with fraud; fraudulent transactions typically involve larger amounts.
- **Transaction Category**: Online and POS transactions show higher fraud risks.
- **Geographical Spread**: Fraud is widely distributed across locations, suggesting that location-based features may not be strong standalone predictors but should be part of the feature set.

Based on the EDA, we used **SVM, Decision Tree, Logistic Regression, Random Forest, and XGBoost** to capture both linear and non-linear relationships in the dataset, addressing the complexity of fraud patterns. Models like **Random Forest and XGBoost** are effective in handling key predictors (e.g., high transaction amounts, specific categories) and the class imbalance between fraud and non-fraud. **Logistic Regression and Decision Tree** provide interpretability, while **Random Forest and XGBoost** offer high accuracy and robustness.

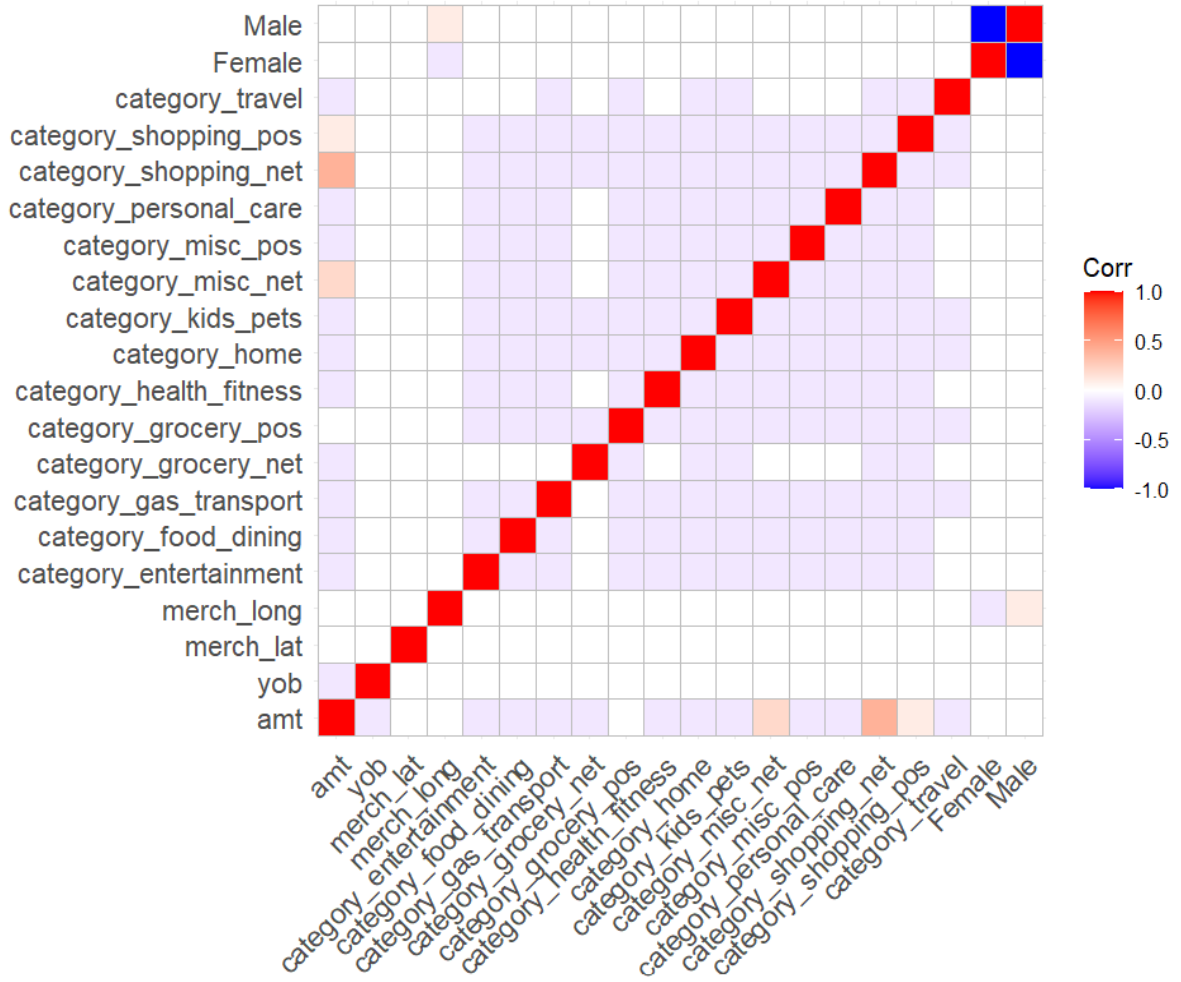
SVM Classifier – Data & Training

Due to the size of the original data set, a set of **4 continuous and 16 (after OHE)** binary variables were selected as the starting point. The features were ”amt”, ”yob”, ”merch_lat”, ”merch_long”, ”category” and ”gender.” **Linear correlation among the variables is low**, however different types of dependencies were not tested for at this stage.

Redundancies were reduced by omitting features containing overlapping information. These were mostly features representing geographical location. The features corresponding to latitude and longitude were left representing location. Features corresponding to time were omitted as well.

There are **obvious limitations** to this type of heuristic variable selection. This will be discussed further later.

The data was normalized before hyperparameter optimization and training. **The data was split 80/20** into training and testing, with around 1/5 of the observations positive.




```
param_grid = [
    {"kernel" : ["rbf"], "C" : [0.1, 1, 10, 100, 1000], "gamma" : np.logspace(-4,0,16)},
    {"kernel" : ["poly"], "C" : [0.1, 1, 10, 100, 1000]},
    {"kernel" : ["linear"], "C" : [0.1, 1, 10, 100, 1000]}
]
```

Grid search was used to optimize hyperparameters. Optimal model in grid: **Radial basis function –kernel with C=1000, gamma ≈ 0.0251.**

- Danger of overfitting as C is large, will be examined

SVM Classifier – Performance and Limitations

Confusion Matrix

1566	23
61	350

The model seems to **work fairly well** for this type of classification problem, and generalize decently to new data:

Precision	Accuracy	Recall	F1
0.94	0.96	0.85	0.89

However, depending on the cost of false negatives, **models with higher recall might be better.**

As mentioned, the feature selection process most likely left a lot of relevant information unused. Going through the entirety of the dataset would most likely been unfeasible with the current tools and computing power due to the sheer number of categories among the categorical variables, but we still could have **started with a wider selection of features**. Being aware of the possibility of overfitting, including some perhaps cyclical time variables could be beneficial.

However, with the increasing number of features the computational cost of hyperparameter tuning can increase quickly. When adding more features, more sophisticated, **suitable hyperparameter tuning methods should be considered.**

Decision Tree

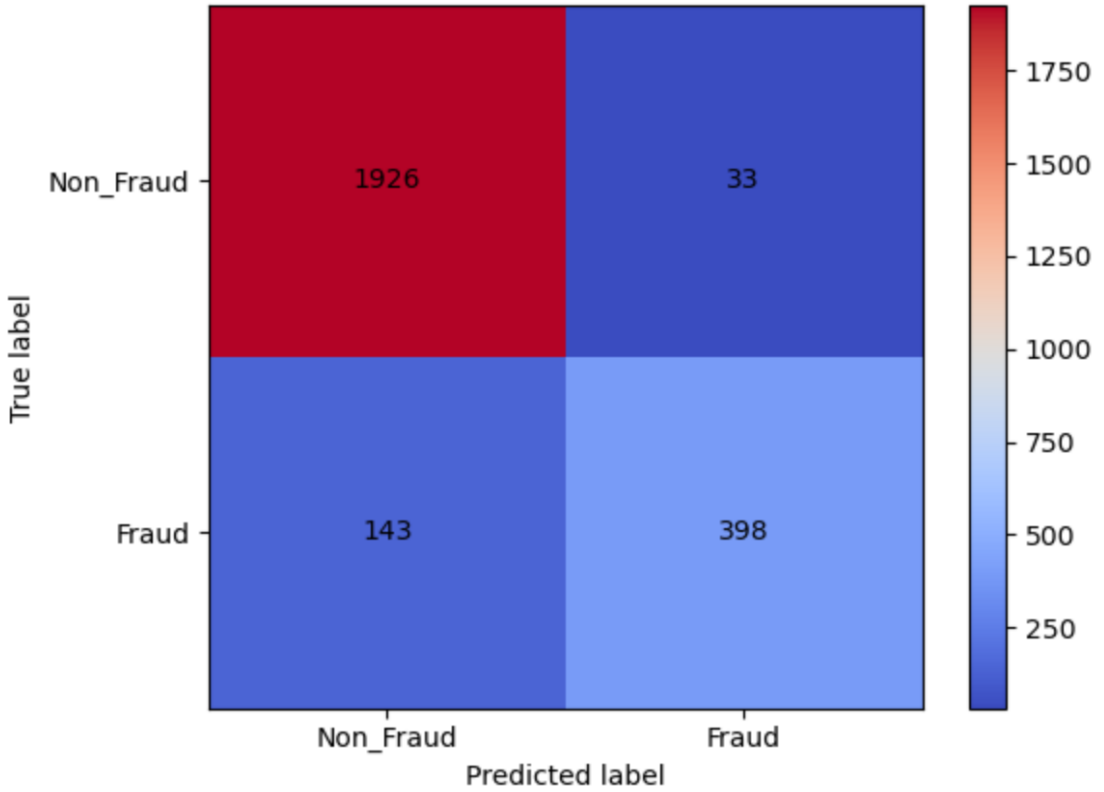
Feature Engineering for Logistic Regression:

- Remove very specific data – merchant / street names/ id's/ even coordinates – resulting in a relatively concise sample – 10.000 x 88 dimension

- Created age groups, city population groups, change genders into binary format

- Polished the data with replacing identical - ages, population, states, and sectors etc. with binary dummies – 0/1

Train/Test Split – .75/.25
(same total sample of fraud cases)



Precision	Accuracy	Recall	F1
0.9234	0.9296	0.7357	0.8189

- The **accuracy** and **precision** are surprisingly still high, which means that decision tree classifier quite successfully distinguishes fraud and non-fraud cases
- **Precision of 92.35%** means overwhelming majority of the predicted fraud cases are actually fraud.
- Again, value of **recall** is lower which indicates that model still misses some fraud cases counting them as non-fraud
- Despite lower number of features, the model still generalized quite well, meaning there is general corellation between independent variables and corresponding fraud cases.

Logistic Regression

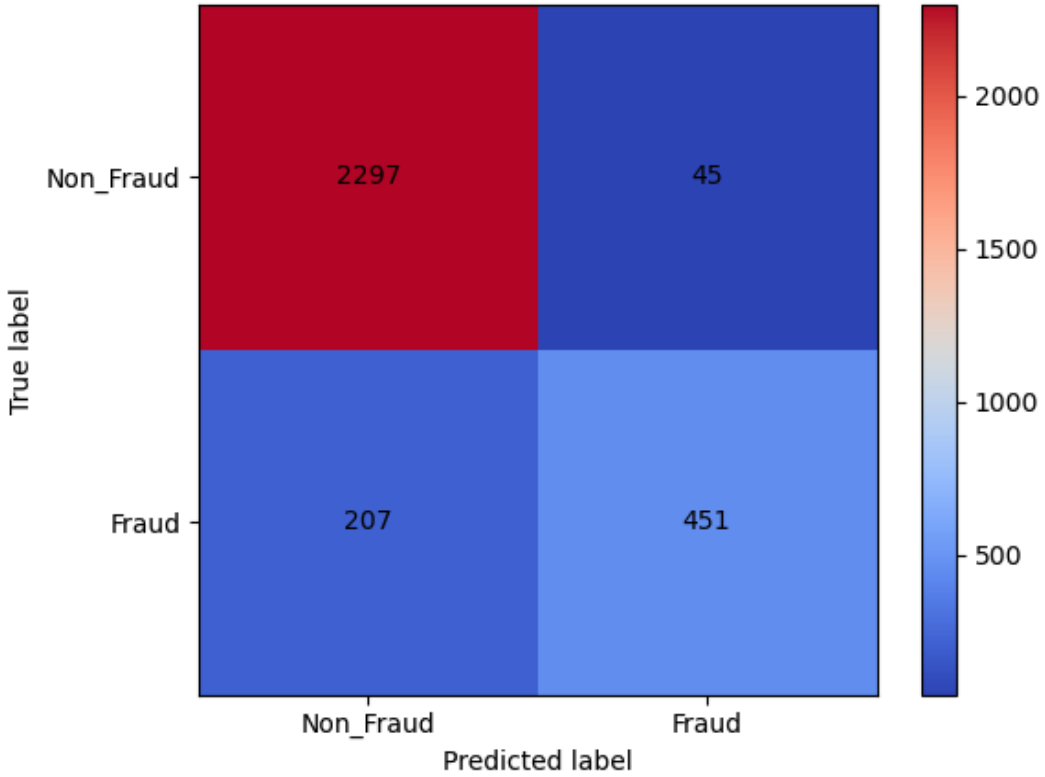
Feature Engineering for Logistic Regression:

- Drop unnecessary variables about transaction numbers, personal information, etc

- Feature engineering: convert time variables to usable features, group job into larger sector, label gender variable

- Categorical variables: using one-hot encoding to convert into binary columns

Train/Test Split – .70/.30
(same total sample of fraud cases)



Precision	Accuracy	Recall	F1
0.90	0.91	0.68	0.77

- The **accuracy** and **precision** are quite high, meaning the model is good at predicting non-fraud cases and identifying fraud when predicted.
- **Precision of 90%** means most of the predicted fraud cases are actually fraud.
- The **recall** for fraud detection is lower (68%). This indicates that a notable portion of actual fraud cases (about 32%) are not being caught by the model. This could be a concern in a fraud detection context where missing fraudulent transactions might have serious consequences.

Feature Engineering for Random Forest and XGBoost Model

_ Classify the transaction time into **"day_period," "trans_month," and "is_weekend" categories** to optimize the analysis and reduce the amount of data being processed.

_ Use the date of birth data to calculate the age and group individuals into **age categories** such as "14-18," "18-35," "35-60," and "60+," to facilitate demographic analysis.

_ Group the "State" data into **regions** and categorize the "Job" data into **job sectors.**

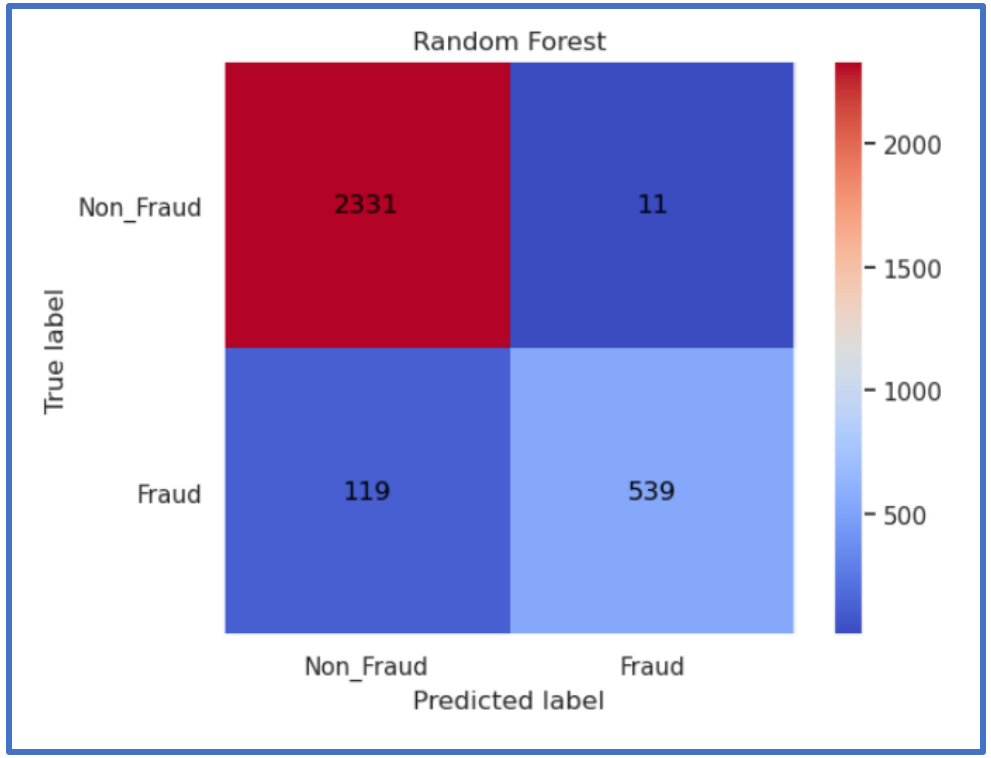
_ **Drop data features** such as "merch_lat," "merch_long," "first," "last," "street," "lat," and "long" as they do not show significant relevance to the "is_fraud" feature, which is used for predicting credit card fraud.

The final features, as can be seen below, are the ones retained for model prediction, excluding irrelevant ones that do not contribute significantly to the prediction of credit card fraud. Afterward, the remaining categorical features will undergo one-hot encoding to prepare them for use in the model prediction process.

```
df.head()
```

	cc_num	merchant	category	amt	gender	city	is_fraud	trans_year	is_weekend	day_period	trans_month	age_group	Region	Job Sector
0	6564459919350820	fraud_Jenkins, Hauck and Friesen	gas_transport	75.79	M	Benton	0	2020	0	Night	December	60+	Midwest	Marketing
1	4451952084362894	fraud_Schuppe LLC	entertainment	81.57	M	Nazareth	0	2020	1	Night	July	35-60	South	IT
2	378006354724784	fraud_Lockman Ltd	grocery_pos	311.92	M	Etlan	1	2020	1	Night	November	35-60	South	Manufacturing
3	370818583810103	fraud_Johns Inc	entertainment	8.21	M	Rockwood	0	2020	1	Afternoon	July	35-60	Northeast	Healthcare
4	180067784565096	fraud_Frami Group	entertainment	2.32	F	North Prairie	0	2020	0	Afternoon	December	60+	Midwest	IT

Random Forest Classifier



The confusion matrix further illustrates that the model correctly identified 539 fraudulent transactions, while only misclassifying 119 as non-fraudulent. This is visualized in the heatmap.

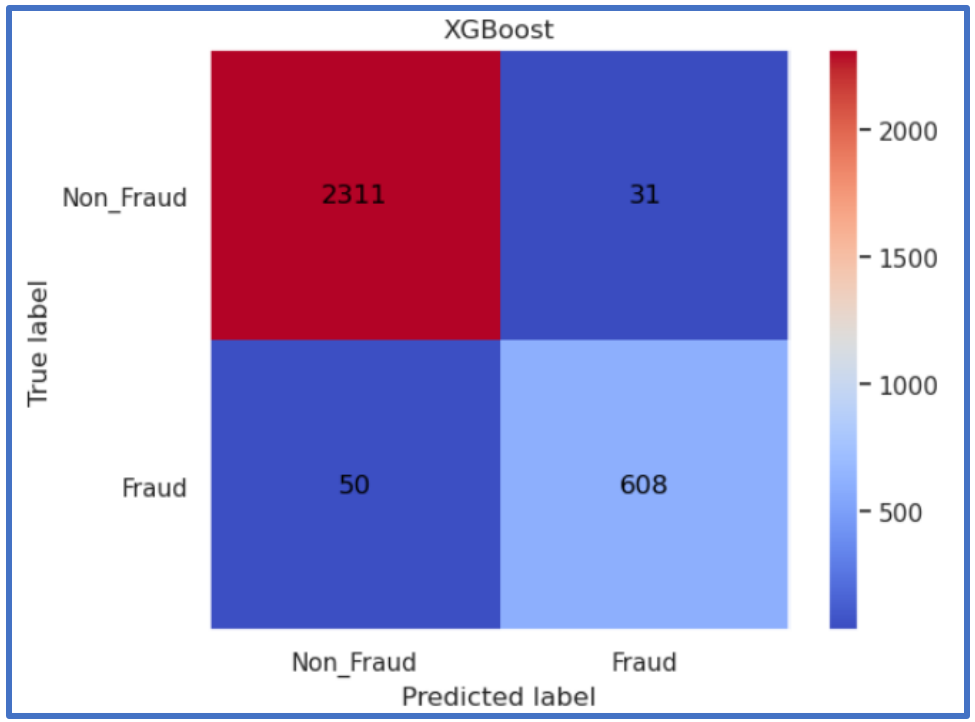
The dataset contains significantly more non-fraudulent transactions than fraudulent ones, making class imbalance an important factor to consider. To address this, we used the **class_weight='balanced'** parameter in the Random Forest model, which adjusts the weights of classes inversely proportional to their frequencies in the dataset. This ensures that the model gives equal importance to the minority class (fraudulent transactions) during training.

We evaluated the model using key performance metrics such as accuracy, precision, recall, F1-score, and the AUC (Area Under the Curve). The model achieved a high accuracy of 95.67% and an impressive AUC score of 0.99, demonstrating its ability to **distinguish between fraudulent and non-fraudulent transactions effectively**.

The precision and recall scores highlight the model's reliability, balancing both minimizing false positives and ensuring a good capture rate of fraudulent cases.

These results validate the correctness of the chosen predictive method and its performance, proving that the **Random Forest model is well-suited** for the task of credit card fraud detection.

Extreme Gradient Boosting



The confusion matrix further illustrates that the model correctly identified 608 fraudulent transactions, while only 50 fraudulent transactions that the model failed to identify.

The high performance of XGBoost on unseen test data shows that it generalizes well and is not overfitting

XGBoost is inherently robust to class imbalance due to its weighted voting mechanism in the boosting process. The parameter **eval_metric='logloss'** optimizes the log loss during training, which is a proper choice for a binary classification problem.

The XGBoost model demonstrates excellent performance in predicting credit card fraud, with a 97.3% accuracy and a perfect AUC of 1. The model **effectively balances precision and recall**, minimizing false positives while capturing most fraudulent transactions. Its scalability and efficiency make it ideal for real-time applications, and its ability to handle imbalanced data sets it apart from simpler models.

By fine-tuning the decision threshold and exploring additional fraud-specific features, the model can potentially achieve even better recall, ensuring more fraudulent transactions are caught without sacrificing too much precision.

XGBoost has **higher accuracy and a perfect AUC** compared to models that rely on linear assumptions or single trees, which may underperform in complex datasets with imbalanced classes.

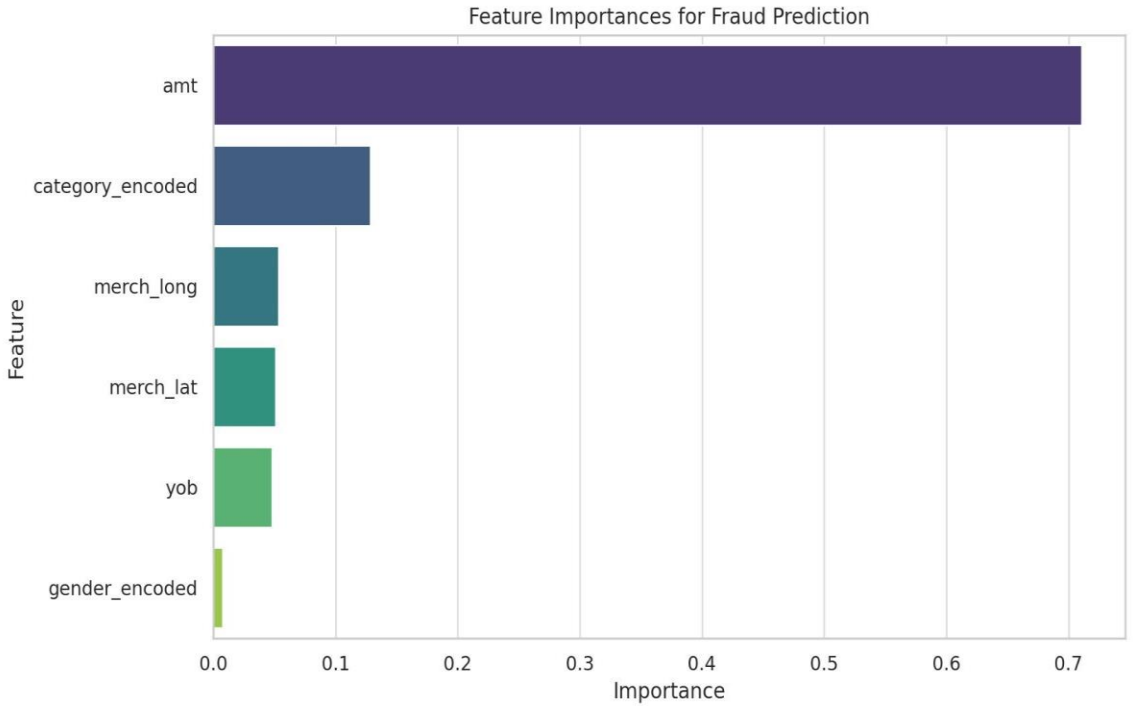
Conclusions

1. All models were trained well on the dataset meaning there is good corelation between the variables and fraud cases – age/ gender/ sector/ amount paid/ and state location.

2. As expected, ensemble learning models – **XGBoost and Random Forest** – performed better compared to others – higher Recall and F-1 score values.

3. Decision tree still had good generalization even though it had significantly less features – excluded extra names/ cities/ and grouped city population.

4. **For the future:** with better computational power and more time – the business should test models on all 550 thousand data points. Even though our maximum Recall score is 92% (**XGBoost**) , we should try to maximize Recall to focus on detecting fraud transactions.



Model	Precision	Accuracy	Recall	F1 - Score
Logistic Regression	0.90	0.91	0.68	0.77
Decision Tree	0.92	0.93	0.76	0.82
SVM	0.94	0.96	0.85	0.89
Random Forest	0.95	0.98	0.81	0.89
XGBoost	0.97	0.95	0.92	0.93