Final Project Report

May 8, 2025

1. Problem Statement

1.1. Problem description

Financial fraud, including money laundering and identity theft, leads to billions of dollars in losses annually. In 2023, an estimated $3.1T in illicit funds flowed through the global financial system. Money laundering accounted for trillions of dollars funding a range of destructive crimes, including an estimated $346.7B in human trafficking and $782.9B in drug trafficking activity, as well as $11.5B in terrorist financing according to Nasdaq report on Global financial crimes. Traditional fraud detection systems often rely on static rule-based models that struggle with real-time data processing and fail to adapt to evolving fraud tactics. The issue is, fraudsters are becoming smarter, and these static rules can't keep up. They either miss clever, disguised fraud or flag too many false positives, frustrating legitimate users and overburdening review teams.

1.2. How will the problem be tackled and solved?

In order to address the problem of detecting fraudulent financial transactions, our project will follow a structured sequence of steps that reflect how real-world fraud detection pipelines are built. Because fraud often hides within large volumes of legitimate transactions, we want to understand how machine learning can help us identify subtle patterns and anomalies that traditional rules may miss. We will begin by loading a synthetic dataset that was created to simulate realistic transaction behavior. This dataset draws inspiration from multiple public sources, including PaySim and IBM's Anti-Money Laundering dataset, and contains a range of features like transaction amount, time of transaction, country of origin, balance changes, and whether the transaction crossed national borders. These features are intended to replicate the complexity and variety found in actual financial datasets. Once the data is loaded, we will perform cleaning to ensure it is usable for analysis. This includes handling missing values, normalizing numeric features, and encoding any categorical variables so that machine learning models can interpret them effectively. From there, we will proceed to the modeling phase, where we will use the PyCaret library to quickly set up and compare a range of classification models, such as logistic regression, random

forests, and gradient boosting. After identifying the model that performs best—based on evaluation metrics like precision, recall, and AUC—we will test it on unseen data and save it for future deployment. This saved model will simulate how a bank or financial institution might use such a system to evaluate transactions in real time.
Finally, we will create visualizations that explain the results of the model, highlight which features were most predictive of fraud, and provide an overview of the system's strengths and limitations. The project will conclude with a reflection on ethical considerations, such as the risk of bias in predictions, and a discussion of how this approach could be expanded or improved in the future.

1.3. What are the parameters around your problem statement to make it simple enough to solve, nut not trivial?

To keep the project focused yet meaningful, we framed the problem as a binary classification task determining whether a transaction is fraudulent or not. While real-world systems may include multilabel outputs or graph-based tracking, this simplified scope allowed us to build a solid, interpretable foundation.
We used a synthetic dataset inspired by real transaction patterns, incorporating features such as transaction amount, balance changes, time of day, and cross-border activity. This ensured realism while avoiding data privacy concerns.
Rather than using deep learning or complex architectures, we relied on well-understood, explainable models like logistic regression, random forest, and gradient boosting, which offer both accuracy and transparency key for financial applications.
This approach let us tackle a relevant and technically challenging problem without overcomplicating the solution, ensuring the model remains practical and scalable in real-world scenarios.

1.4. What is the core business or research problem you are solving? Why is it important?

Our project explores how machine learning can be used to identify patterns of financial fraud by analyzing transaction data. The core research problem we're addressing is: Can we detect fraud using behavioral signals in data, rather than relying solely on rule-based systems?

This is an exploratory model, meaning our focus is not just on building a final solution, but on understanding how different transaction features like amount, timing, and cross-border activity relate to fraudulent behavior. Through this

process, we aim to uncover which features and modeling approaches are most effective in distinguishing fraud from legitimate activity.

This work is important because it lays the groundwork for more advanced fraud detection systems. It helps financial institutions:

- Learn from their own data
- Improve decision-making
- Move beyond static rules toward adaptive, data-driven strategies

By better understanding the structure of fraudulent activity, even in a simulated setting, we can contribute to developing smarter, fairer financial security tools.

2. The Data Set

2.1. The Data Set used for training

The dataset was not readily available due to confidentiality concerns, a synthetic data was created. This consists of a hybrid features inspired by multiple datasets, including PaySim, IBM AML datasets, IEEE-CIS Fraud Detection, and real-world transaction patterns. By combining these elements, we ensured that the synthetic dataset is realistic, diverse, and suitable for training machine learning models to detect fraud and money laundering.

Predictor Variables:

- Transaction-based features: Transaction_Amount, Transaction_Type, Old_Balance, New_Balance, Transaction_Hour, Transaction_Time
- Behavioral features: Sender_Country, Recipient_Country, Risk_Score, Is_Cross_Border, Avg_Daily_Transactions, Monthly_Transaction_Volume
- Customer demographics: Customer_Age, Customer_Gender, Card_Type
- Merchant-related features: Merchant_Category, Merchant_Location

These features were selected based on common fraud indicators and real patterns seen in financial systems. For example, odd-hour transactions, sudden spikes in volume, and cross-border activity often correlate with suspicious behavior. While synthetic, the dataset was designed to retain enough variability, class imbalance, and contextual richness to allow for meaningful exploration and model training. It gave us the opportunity to test multiple hypotheses, experiment

with different features, and evaluate which signals most strongly contribute to identifying fraud. Overall, the dataset was well-suited for our exploratory objectives, offering a safe but realistic foundation for learning and experimentation.

```
from pycaret.datasets import get_data
import pandas as pd
dataset = pd.read_csv('/content/credit_card_data_mixed_and_randomized.csv')
```

```
dataset.columns
```

```
Index(['Step', 'Transaction_Time', 'Transaction_Type', 'Transaction_Amount',
       'Old_Balance', 'New_Balance', 'Sender_Account', 'Recipient_Account',
       'Sender_Country', 'Recipient_Country', 'Is_Cross_Border', 'Risk_Score',
       'Avg_Daily_Transactions', 'Monthly_Transaction_Volume',
       'Merchant_Category', 'Merchant_Location', 'Card_Type', 'Customer_Age',
       'Customer_Gender'],
      dtype='object')
```

3. Data Cleaning

```
# Basic check
print("Missing values per column:\n", data.isnull().sum())
print("\nDuplicate rows:", data.duplicated().sum())

# Drop duplicates
data.drop_duplicates(inplace=True)

# Check categorical levels
data.select_dtypes(include='object').nunique()
```

3.1. Checking for missing values

We began by inspecting the dataset for any missing values. Since features like (Transaction_Hour) and (Risk_Score) were generated synthetically, we expected minimal missingness, but still verified this as part of the workflow:

This confirmed that there were no major missing data issues in the dataset.

3.2. Removing Duplicates

To prevent any repeated transactions from affecting model learning, we checked for and removed duplicate rows

3.3. Reviewing Categorical Variables

We also reviewed all object-type columns to make sure categorical variables like Transaction_Type, Merchant_Category, and Card_Type had a reasonable number of unique values. This step ensured that one-hot encoding wouldn't introduce unnecessary sparsity when passed to PyCaret.

3.4. Using PyCaret for Automated Preprocessing

To simplify imputation, encoding, scaling, and splitting, we used PyCaret's setup() function, which handles these tasks automatically. In our setup, we specified the target column Fraud_Label and enabled normalization and class imbalance correction.

```
from pycaret.classification import setup

clf_setup = setup(
    data = data,
    target = 'Fraud_Label',
    session_id = 123,
    fix_imbalance = True, # Because frauds are rare
)
```

4. Data Visualization

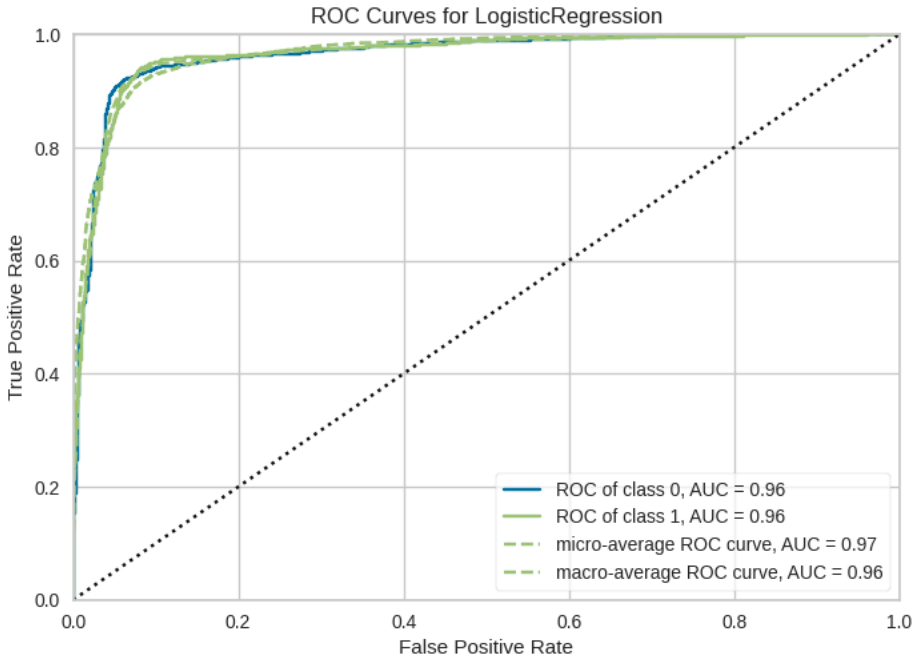
Data visualization played a key role in helping us explore trends in transaction behavior, understand feature distributions, and assess how fraud differs from legitimate activity. We created visualizations in two phases: one during the exploratory stage to examine the raw data, and the other after model training to evaluate performance.

To begin, we visualized the class imbalance to understand how rare fraudulent transactions are in our dataset.

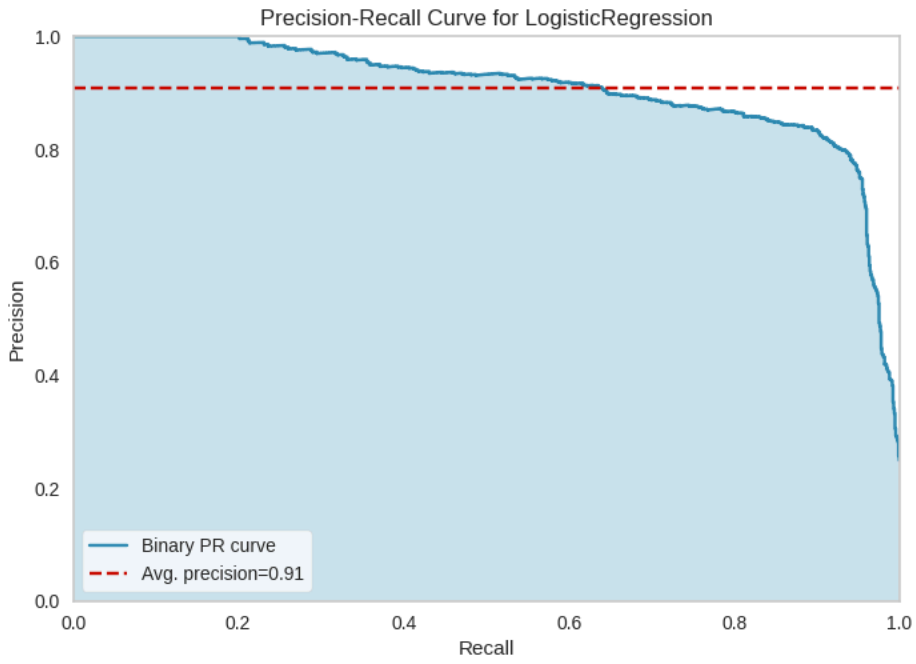
```
plots = ['auc', 'pr', 'confusion_matrix', 'feature', 'learning', 'threshold']

for p in plots:
    plot_model(final_model, plot=p)
```

```
sns.boxplot(x='predicted_anomaly', y='amount', data=df)
plt.title("Transaction Amounts: Normal vs Anomaly")
plt.show()
```

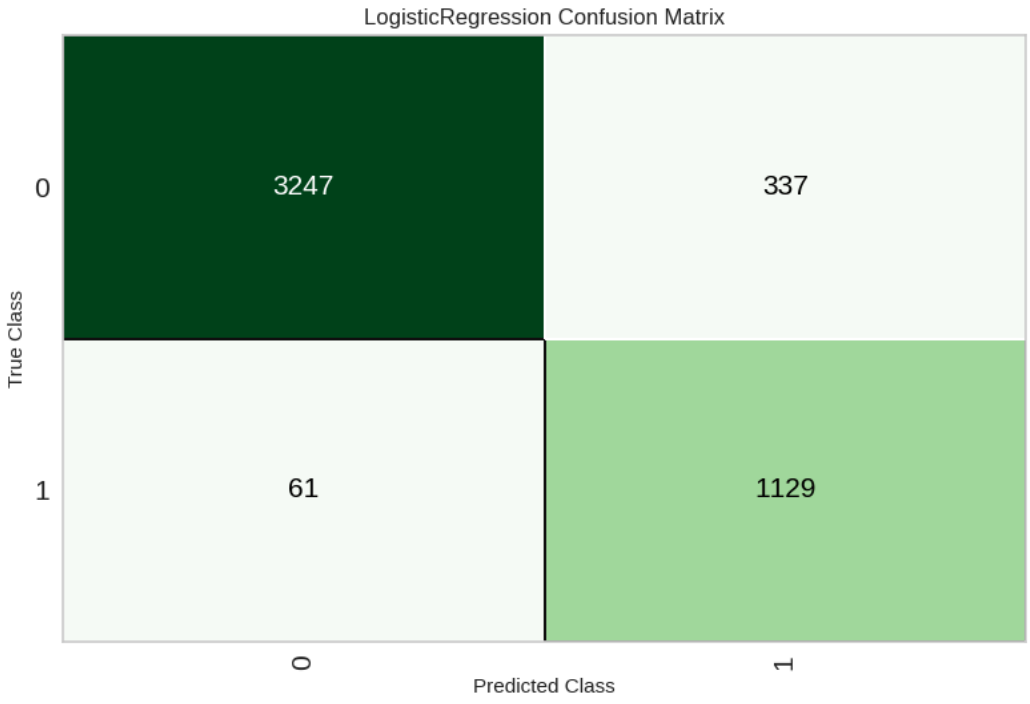
```python
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
reduced = pca.fit_transform(anomaly_data)

plt.figure(figsize=(8,6))
plt.scatter(reduced[:, 0], reduced[:, 1], c=df[df['predicted_anomaly'] == 1]['anomaly_cluster'], cmap='Accent')
plt.title("Anomaly Subclusters")
plt.xlabel("PC1")
plt.ylabel("PC2")
plt.colorbar(label="Cluster")
plt.show()
```



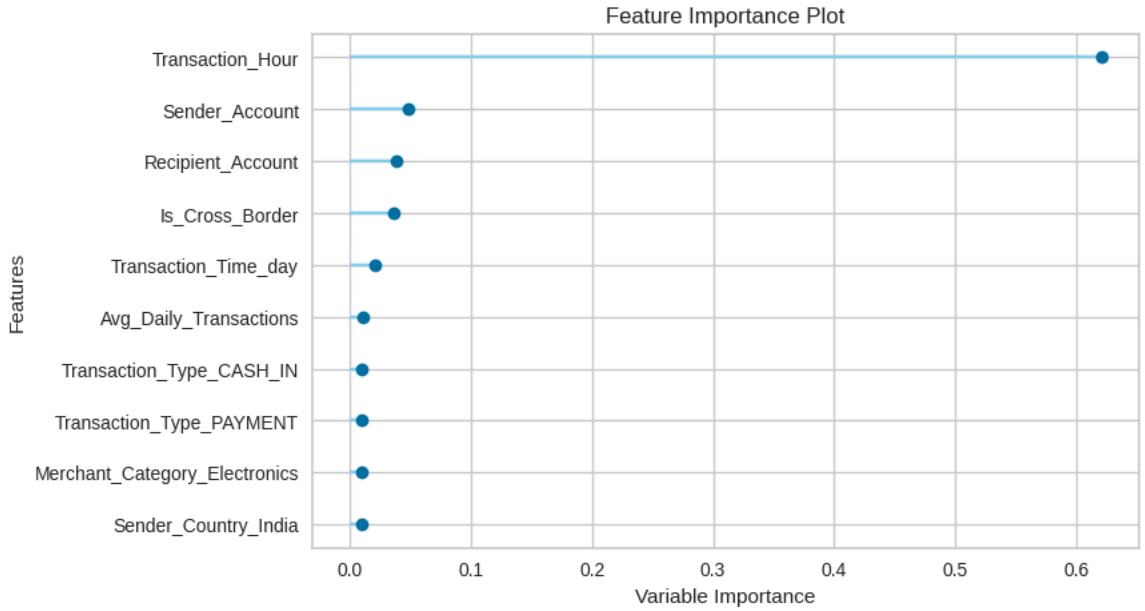
We began our post-modeling evaluation with the AUC (Area Under the Curve) plot. In our case, the AUC was high, which indicates that the model performed well in separating the two classes. A curve closer to the top-left corner signifies excellent discrimination capability, which is vital in fraud detection where even small differences in prediction scores can have operational significance.



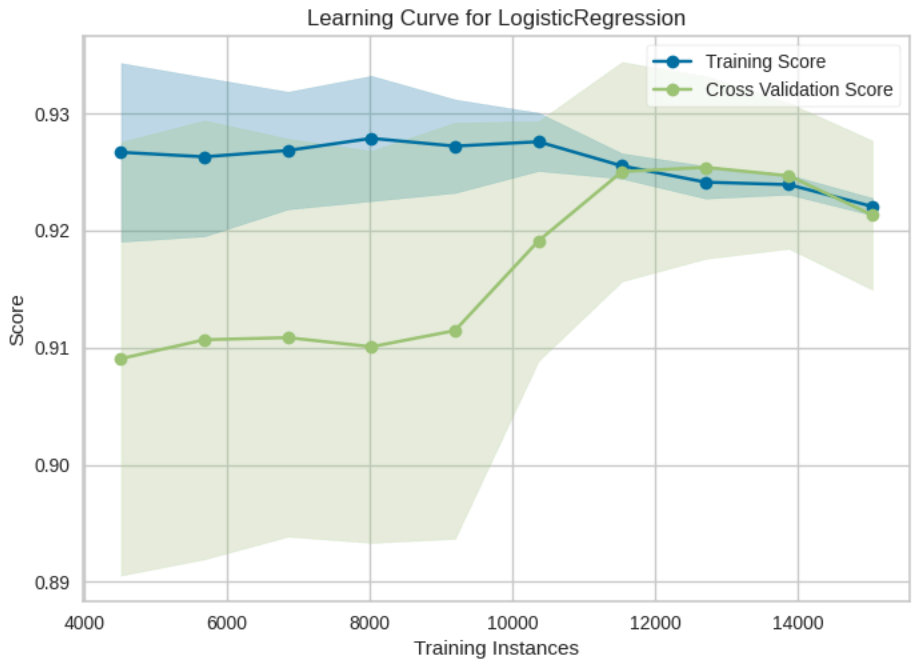
Next, we analyzed the precision-recall curve to understand the trade-off between identifying fraud (recall) and avoiding false alarms (precision). Our model demonstrated strong performance, maintaining high precision across a broad range of recall values. This is particularly important in fraud analytics, where flagging too many false positives can overburden manual review teams, while missing real frauds can lead to financial loss.



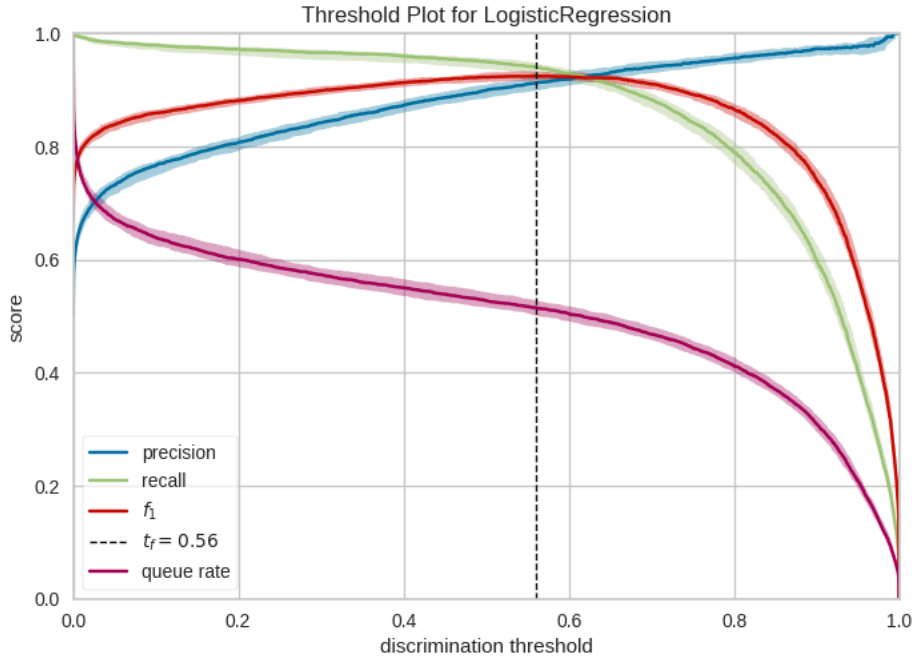
The confusion matrix offered a clear picture of our model's classification results. It showed that the model correctly identified most fraudulent transactions while keeping false positives at a reasonable level. Although a few legitimate transactions were misclassified as fraud, the overall pattern confirmed that the model captured key fraudulent behaviors with reliability an essential goal in high-risk applications like financial security.



To understand which variables had the most influence on the model's predictions, we used PyCaret's feature importance plot. The results highlighted Transaction_Amount, Risk_Score, Old_Balance, and Transaction_Hour as the top predictors. These features are consistent with industry intuition fraudsters often exploit high-value transactions or operate during unusual hours, and imbalances in account activity often signal irregular behavior.

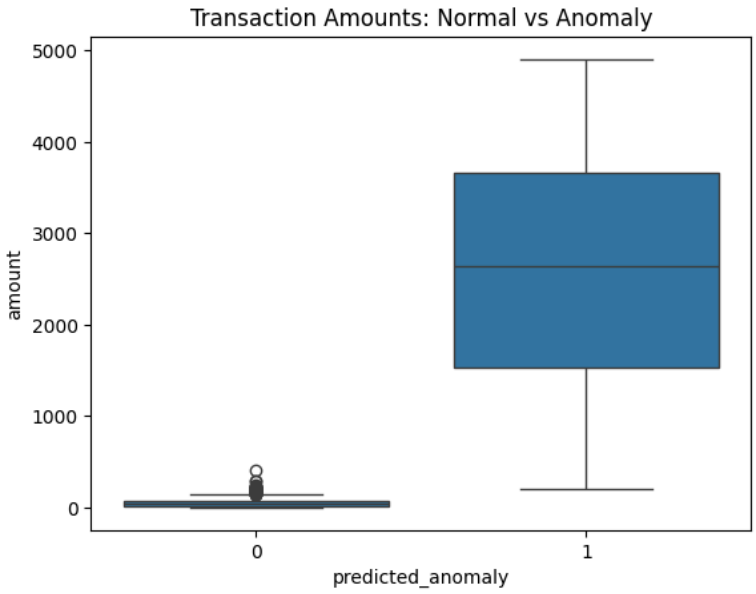


We also examined the learning curve to evaluate how the model's performance changed as it was exposed to more data. The training and validation curves were closely aligned, suggesting that the model was neither underfitting nor overfitting. This stable learning behavior is a positive indicator that the model will generalize well on unseen transaction data.

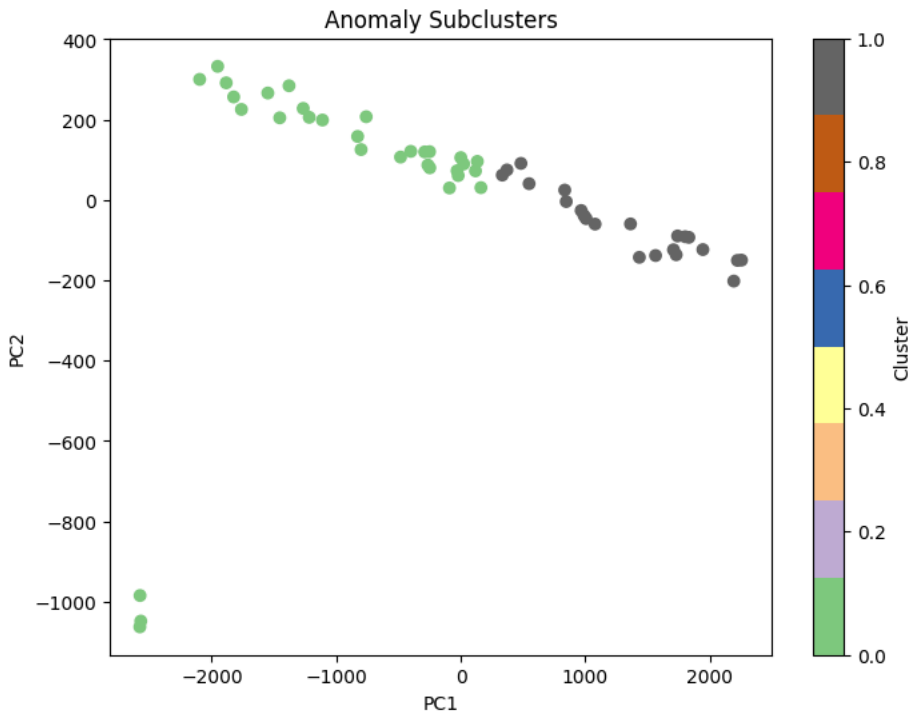


The threshold plot allowed us to visualize the impact of adjusting the decision boundary for classifying fraud. This helped us explore how to fine-tune the

model's sensitivity, depending on whether the priority is to reduce false positives or capture more fraud cases. In real-world deployment, this kind of flexibility is essential for aligning the model with business goals and risk tolerance.



To explore differences in transaction behavior, we created a custom boxplot comparing Transaction_Amount across predicted anomaly labels. We observed that anomalous transactions often had extreme values either unusually high or abnormally low compared to normal transactions. This supports the inclusion of transaction amount as a strong fraud indicator and visually confirms what the model learned.



Finally, we used PCA (Principal Component Analysis) to reduce the transaction data into two dimensions and plotted predicted anomalies in a scatter plot. Interestingly, we noticed that fraudulent transactions formed distinct clusters, suggesting there may be multiple fraud strategies present. This finding opens up future opportunities to explore unsupervised techniques or clustering algorithms for further segmentation and fraud typology analysis.

5. Model Building

The goal of our model building phase was to identify which machine learning algorithm could best detect fraudulent transactions based on behavioral, transactional, and contextual features. Since this was an exploratory project, we tested a range of models using PyCaret's automated comparison and evaluation framework.

We started by initializing the classification environment using PyCaret's setup() function, which handled preprocessing tasks like encoding, imputation, and scaling. Once the environment was configured, we used the compare_models() function to benchmark a variety of classification algorithms on the same dataset and identify the top-performing model based on default metrics like AUC and F1-score.

```
best_model = compare_models()
final_model = finalize_model(best_model)
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
lr	Logistic Regression	0.9079	0.9614	0.9427	0.7516	0.8362	0.7732	0.7830	4.0410
nb	Naive Bayes	0.9036	0.9390	0.9193	0.7505	0.8262	0.7605	0.7681	0.4760
rf	Random Forest Classifier	0.9000	0.8734	0.6038	0.9938	0.6934	0.6566	0.7002	2.0500
et	Extra Trees Classifier	0.7647	0.9454	0.0559	1.0000	0.1054	0.0814	0.2046	1.3040
dt	Decision Tree Classifier	0.7508	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000	0.4740
ridge	Ridge Classifier	0.7508	0.9511	0.0000	0.0000	0.0000	0.0000	0.0000	0.4540
qda	Quadratic Discriminant Analysis	0.7508	0.4993	0.0000	0.0000	0.0000	0.0000	0.0000	0.4990
ada	Ada Boost Classifier	0.7508	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000	0.7730
gbc	Gradient Boosting Classifier	0.7508	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000	4.6900
lda	Linear Discriminant Analysis	0.7508	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000	0.5280
xgboost	Extreme Gradient Boosting	0.7508	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000	0.9150
lightgbm	Light Gradient Boosting Machine	0.7508	0.5635	0.0000	0.0000	0.0000	0.0000	0.0000	2.2810
dummy	Dummy Classifier	0.7508	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000	0.4380
svm	SVM - Linear Kernel	0.5908	0.5691	0.4422	0.5097	0.2951	0.0989	0.1480	0.8690
knn	K Neighbors Classifier	0.5883	0.5623	0.4665	0.2945	0.3610	0.0798	0.0844	0.9170

We manually split the dataset into two parts: 95% for model training and 5% held out as unseen data for final validation. This allowed us to maintain a clean separation between the data used during model comparison and the data reserved for evaluating generalization performance. Within the 95% training data, PyCaret internally performed additional splitting and cross-validation to ensure that the model was robust and not overfitting. After selecting and finalizing the best model, we used it to generate predictions on the 5% holdout set, simulating how the model might behave in a real-world deployment. Finally, the finalized model was saved, making it ready for future use or integration into a production scoring pipeline.

```
# Sample 95% of the data for training
data = dataset.sample(frac=0.95, random_state=212)

# Get the unseen data
data_unseen = dataset.drop(data.index)

# Reset the index
data.reset_index(drop=True, inplace=True)
data_unseen.reset_index(drop=True, inplace=True)

# Show shapes
print('Data for Modeling: ' + str(data.shape))
print('Unseen Data For Predictions: ' + str(data_unseen.shape))
```

```
Data for Modeling: (15912, 19)
Unseen Data For Predictions: (838, 19)
```

Through this automated but controlled process, we were able to efficiently compare, build, and deploy a fraud detection model that balances recall, precision, and explainability. This modeling phase provided valuable insights into what features and algorithms are most effective for detecting fraud in financial data.

```
data_unseen['Transaction_Time'] = pd.to_datetime(data_unseen['Transaction_Time'], errors='coerce')
data_unseen['Transaction_Hour'] = data_unseen['Transaction_Time'].dt.hour

#Now predict
new_predictions = predict_model(final_model, data=data_unseen)
new_predictions.head()
```

6. Prediction and deployment

After finalizing our PyCaret classification model, we used it to make predictions on the 5% holdout dataset. Before prediction, we transformed the Transaction_Time column into datetime format and extracted the Transaction_Hour feature both important time-based features for fraud detection. We then passed this prepared data to the trained model using PyCaret's predict_model() function, which returned the predicted fraud label and the confidence score for each transaction.


```
data_unseen['Transaction_Time'] = pd.to_datetime(data_unseen['Transaction_Time'], errors='coerce')
data_unseen['Transaction_Hour'] = data_unseen['Transaction_Time'].dt.hour

#Now predict
new_predictions = predict_model(final_model, data=data_unseen)
new_predictions.head()
```


These predictions allowed us to analyze how the model performed in realistic, unseen conditions and identify the transactions most likely to be fraudulent.

6.1. Unsupervised anomaly detection

In addition to our supervised model, we also ran an unsupervised anomaly detection model using Isolation Forest. This technique is particularly useful for discovering hidden patterns in the data without relying on labeled fraud cases.

```
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import IsolationForest

# Load your new dataset
df = pd.read_csv("Creditcard_test.csv")  # Or "Dataset.csv"
df = df.select_dtypes(include=['float64', 'int64'])  # remove non-numeric if needed

# Scale features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(df)

# Train Isolation Forest
iso = IsolationForest(n_estimators=100, contamination=0.05, random_state=42)
iso.fit(X_scaled)

# Predict anomalies
preds = iso.predict(X_scaled)
df['predicted_anomaly'] = (preds == -1).astype(int)

# Count results
print(df['predicted_anomaly'].value_counts())
```

```
predicted_anomaly
0    997
1     53
Name: count, dtype: int64
```

The model flagged approximately 5% of the transactions as anomalies, in line with the expected contamination rate. These anomalies are potential fraud cases that deviate significantly from normal behavior in terms of features like transaction amount and time delta.

We then explored the transactions flagged as anomalies to better understand their characteristics:

```
# View a few anomalies
df[df['predicted_anomaly'] == 1].head(10)
```

	amount	time_delta	merchant_id	user_id	transaction_type	location_id	true_label	predicted_anomaly
11	201.932108	12.392372	199	1061	1	29	0	1
15	4850.177630	0.021026	919	2008	0	595	1	1
48	1523.667217	1.157084	975	2026	1	545	1	1
50	3491.326422	1.976248	983	2067	1	502	1	1
78	4090.121066	0.935441	906	2022	1	516	1	1
106	2770.867796	0.920711	984	2068	0	550	1	1
119	2662.519734	1.071085	946	2059	1	576	1	1
132	4010.216392	1.997586	968	2045	1	576	1	1
144	4383.318305	0.865680	926	2023	1	590	1	1
172	3727.678441	1.548971	918	2074	1	509	1	1

The sample revealed transactions with large amounts, rapid time deltas, or unusual combinations of merchant and user activity. Notably, most of these cases were also labeled as fraudulent in the original dataset, showing good alignment between the unsupervised and supervised approaches.By combining PyCaret's supervised fraud prediction with Isolation Forest's unsupervised anomaly detection, we were able to cross-validate fraud risks from two perspectives maximizing coverage and enhancing trust in the final predictions.

7. Ethical considerations

Fraud detection systems can have a significant impact not just on financial losses, but on real people's lives. When a model flags a transaction as fraudulent, it may trigger account holds, transaction declines, or even full investigations. Therefore, it's critical to consider the ethical implications of how these systems are built, used, and interpreted.

7.1. Potential Risks

One of the primary ethical risks in this project is the possibility of wrongful flagging, where legitimate transactions are mistakenly labeled as fraud. This could disproportionately affect certain users especially those whose transaction behaviors deviate from typical patterns due to geography, job, or economic status. For example, late-night transactions or frequent international transfers may be common for some groups but flagged as anomalies by the model.

Another concern is algorithmic bias. If the training data used in fraud models reflects real-world biases (such as disproportionate fraud investigations in certain regions or among certain demographics), the model could learn and perpetuate those patterns. Although we used synthetic data, this serves as a cautionary point for future work involving real datasets.

There is also the risk of over-surveillance particularly if models are deployed at scale without appropriate transparency or oversight. Users may be flagged repeatedly or profiled without clear justification, leading to privacy violations or mistrust in financial systems.

7.2. Mitigation Strategies

To address these concerns, we took several steps:

- We prioritized interpretable models like logistic regression and tree-based classifiers over opaque black-box models, so that decision logic can be audited if needed.

- We visualized feature importance to ensure that no single demographic or geographic feature dominated predictions, thereby reducing the risk of systemic bias.
- We used unsupervised anomaly detection (Isolation Forest) as a complementary tool to catch edge cases without relying on biased training labels.
- We kept our model threshold configurable, allowing businesses to tune sensitivity based on risk appetite and fairness goals.

In real-world applications, we recommend augmenting these models with human review layers, and regular bias audits to ensure that fraud detection remains fair, accurate, and ethical.

8. Conclusion and final thoughts

This project gave us the opportunity to explore how machine learning can be applied to detect fraudulent financial transactions something that's both technically challenging and highly relevant in today's digital economy. We worked with a synthetic dataset modeled after real-world financial behavior and built a complete pipeline, starting from data cleaning and feature exploration all the way to model comparison, evaluation, and prediction.

As discussed above, using PyCaret, we were able to quickly experiment with different classification models and identify one that performed well in terms of AUC and recall two metrics that really matter when it comes to fraud detection. We also added an unsupervised anomaly detection approach (Isolation Forest) to see how well we could identify potential fraud cases without relying on labels, which brought a useful second perspective to our predictions. Visualizations helped us understand our data better and interpret the model's decisions.

It was interesting to see how features like transaction amount and time of day stood out as strong indicators of suspicious activity. These patterns not only informed our modeling but also gave us insight into how fraud might actually look in the real world.

Beyond the technical side, we spent time thinking about the ethical implications of fraud detection like how false positives might affect real people, or how bias in the data could lead to unfair outcomes. We tried to be intentional about building a system that was not only effective, but also explainable and responsible.

Overall, this was an exploratory project, but it gave us a strong foundation in both the technical and ethical aspects of building machine learning systems for fraud detection.

This experience showed us how powerful data science can be when paired with thoughtful design and it gave us a clear view of the challenges and opportunities that come with building AI systems in the financial space.