Fraud Detection Pipeline

Mohamad Hijazi - 213011026

Overview:

This project addresses the problem of credit card fraud detection using both traditional machine learning models and a GPU-accelerated deep learning approach.
The dataset contains transactional data with numeric and categorical features, as well as a binary target variable indicating fraudulent transactions.
The workflow involves data preprocessing, model training, evaluation, and temporal analysis of fraud patterns.

Dataset:

Credit Card Transactions Fraud Detection Dataset – (Kaggle link)
I selected this dataset because it provides a realistic simulation of credit card transactions, including fraudulent and legitimate activities.
The dataset is large and diverse (over 1.29 million transactions, 23 features), which makes it suitable for:

- Fraud detection model development
- Exploratory data analysis (EDA)
- Handling real-world issues like class imbalance

Additionally, it contains both customer information and transaction details, which allows for rich and varied analysis.
In addition, The dataset is imbalanced, it reflects real-world fraud detection challenges, making this dataset particularly interesting and valuable.

File System:

*Dataset Files:

Property	Description
File Name	fraudTrain.csv
Size	~227.54 MB in memory
Format	CSV (Comma-Separated Values)
Source	Downloaded from Kaggle

*Protocol/Format:
Standard CSV file – easy to handle with Python and other libraries.

*Version Management:
There is one version of the file: fraudTrain.csv.

*Data Types Summary:

Data Type	Number of Columns
Object (Text)	12
Integer (int64)	6
Float (float64)	5

*Examples of Each Type:

- Textual Features:
  merchant, category, first, last, job, gender
- Numerical Features:
  amt, city_pop, merch_lat, merch_long, etc.
- Datetime Features:
  trans_date_trans_time (transaction time)
  dob (date of birth)

Statistical Analysis:

We performed descriptive statistical analysis on the numeric columns of the dataset (excluding 'is_fraud' column).
The dataset includes 10 numeric features, and here is a summary of key statistics:

Feature	Mean	Std Dev	Min	25%	Median	75%	Max
amt (transaction amount)	70.35	160.32	1.00	9.65	47.52	83.14	28,948.90
city_pop (city population)	88,824	301,956	23	743	2,456	20,328	2,906,700
lat (latitude)	38.54	5.08	20.03	34.62	39.35	41.94	66.69
long (longitude)	-90.23	13.76	-165.67	-96.80	-87.48	-80.16	-67.95
merch_lat (merchant latitude)	38.54	5.11	19.03	34.73	39.37	41.96	67.51
merch_long (merchant longitude)	-90.23	13.77	-166.67	-96.90	-87.44	-80.24	-66.95
unix_time (transaction time as UNIX timestamp)	1.349e+09	1.28e+07	1.325e+09	1.338e+09	1.349e+09	1.359e+09	1.371e+09

Transaction amount (amt) has the strongest correlation with fraud (0.219). This makes sense as fraudulent transactions often involve unusual amounts.

All other features have very low correlations (< 0.01) with fraud, indicating they are not linearly related to the target.

Is PCA recommended?

- Since the features are not strongly correlated with each other or with the target (except for amt), PCA may not bring significant benefit for fraud prediction in this case.

- However, if the goal is visualization or feature compression, applying PCA could help reduce redundancy in geolocation features (latitude/longitude) or time-based features.

Anomaly Detection:

We applied both univariate and multivariate anomaly detection techniques to identify abnormal transactions in the dataset. Using the Interquartile Range (IQR) method, we detected significant outliers in transaction amounts (amt), highlighting large transactions that are typically associated with fraudulent behavior. Additionally, geographic anomalies in 'lat' and 'merch_lat' suggest transactions occurring in unusual or unexpected locations. To capture complex, multidimensional anomalies, we employed the Isolation Forest algorithm, which identified 64,834 anomalous transactions (5% of the dataset). Notably, the fraud rate within these anomalies was 5.30%, compared to the overall fraud rate of 0.58%, resulting in a 9.16x improvement in fraud detection focus. These findings align with domain knowledge, indicating that anomaly detection can be a valuable tool for prioritizing suspicious transactions and supporting fraud detection systems.

Clustering:

We used K-Means clustering to group transactions into distinct segments based on 'amt', 'lat', 'long', and 'city_pop'. The algorithm successfully divided the data into five clusters, each representing different transaction behaviors. Clusters 0, 1, and 2 covered most of the data (about 96% combined) and had similar characteristics with low average amounts (~$61-$62) and low fraud rates (~0.3%), likely representing regular daily transactions. Cluster 3, though smaller (3.1% of the data), showed a slightly higher average amount ($66.03) and a fraud rate of 0.42%, indicating moderately riskier transactions. The most distinct group was Cluster 4, containing only 0.8% of the transactions, but with an exceptionally high fraud rate of 34.5% and an average amount of $1,188.23. This suggests Cluster 4 represents high-value, high-risk transactions, potentially making it a priority segment for fraud

monitoring. The clustering analysis revealed clear customer and transaction segmentation, allowing better focus on suspicious behaviors in specific clusters.

Text analysis:

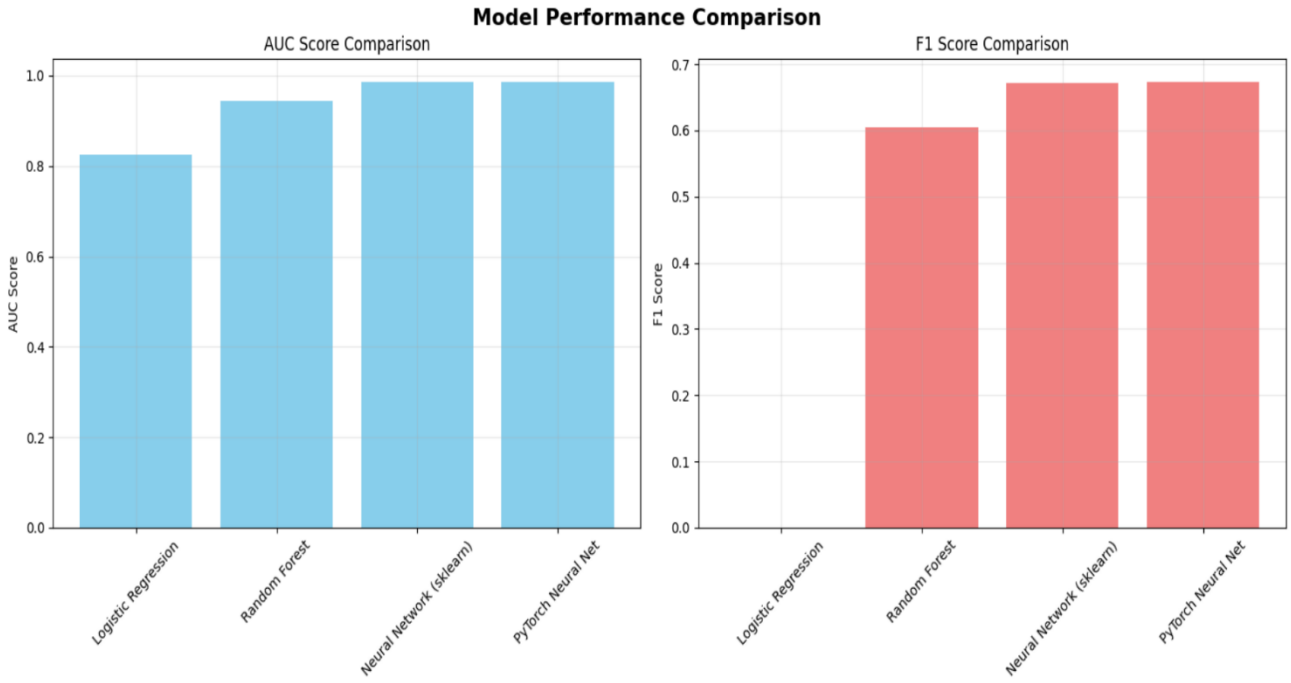
We focused on categorical and textual columns such as merchant, category, first, last, gender, and others. The merchant column contains 693 unique merchants, with specific merchants labeled as "fraud_*", likely synthetic fraud patterns. The category field, with 14 distinct values, revealed meaningful fraud insights—online shopping categories (shopping_net, misc_net, grocery_pos) showed significantly higher fraud rates, suggesting fraud is more prevalent in online transactions compared to physical purchases. The first and last name fields have many unique values (352 and 481, respectively), mostly representing customer demographics, while gender showed a slight difference in fraud rates between males (0.64%) and females (0.53%).

Models:

We implemented and compared several machine learning models using both classical and deep learning approaches. Each model was trained on a large, imbalanced dataset where fraudulent transactions were rare compared to legitimate ones.

Model	AUC Score	F1 Score	Cross-Validation AUC (Mean ± Std)	Explainability
Logistic Regression	0.8241	0.0000	0.8295 ± 0.0122	Fully Explainable
Random Forest	0.9442	0.6048	0.7851 ± 0.1644	Partial (Feature Importances)
Neural Network (sklearn)	0.9870	0.6717	0.9699 ± 0.0222	Black Box (Low Explainability)

PyTorch Neural Network	0.9864	0.6738	Not Applicable (No CV Implemented)	Black Box (Low Explainability)



*AUC Score: Measures the model's ability to distinguish between fraud and non-fraud transactions.

- Neural Network models achieved the highest AUC (~0.99), showing excellent classification ability.
- Random Forest performed well (AUC ~0.94) while Logistic Regression had the lowest AUC (0.82) due to its simplicity.

*F1 Score: Measures the balance between precision and recall for fraud detection.

- Neural Networks (both sklearn and PyTorch) obtained the best F1 scores (~0.67), successfully handling data imbalance.
- Random Forest also performed reasonably well (F1 ~0.60).
- Logistic Regression failed to detect fraud in practice (F1 = 0.0000) because of extreme class imbalance.

*Cross-Validation:

- CV was used to evaluate model stability. The Neural Network (sklearn) showed consistent high performance across folds (CV AUC = 0.9699 ± 0.0222).
- Random Forest had higher variance (CV AUC = 0.7851 ± 0.1644), possibly due to sensitivity to different training splits.

*Explainability:

- Logistic Regression is fully explainable but performed poorly.
- Random Forest provides some interpretability via feature importance.
- Neural Networks (sklearn & PyTorch) are black-box models, performing best but lacking in direct explainability.

The models demonstrated the best performance in both AUC and F1 scores, suggesting a good fit to the data. However, they are not easily explainable, which is a trade-off when compared to simpler models like Logistic Regression. Random Forest serves as a middle ground, offering reasonable performance with partial explainability.

For real-world deployment, combining the high performance of neural networks with explainability tools (like SHAP or LIME) is recommended to ensure trust and compliance in fraud detection systems.

Future Work and Improvement Suggestions:

To further enhance the system, several steps can be considered:

- Model Enhancement:
  Test advanced models or deep learning approaches to improve fraud detection performance.
- Better Segmentation:
  Refine the clustering process to identify specific fraud-prone customer segments or behavioral patterns.

- Explainability:
  Incorporate explainable AI tools to better understand the model's decisions.
- Data Enrichment:
  Integrate additional data sources to improve model accuracy.
- Continuous Monitoring:
  Implement a system to monitor model performance over time to detect concept drift and retrain as needed.