Maharshi Karve Stree Shikshan Samstha's
Cummins College of Engineering for Women, Pune
(An autonomous Institute affiliated to Savitribai Phule Pune University)
Department of Computer Engineering

MKSSS's Cummins COLLEGE OF ENGINEERING FOR WOMEN

23PCCE501L ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING LABORATORY

CREDIT CARD FRAUD DETECTION

OUR TEAM

UCE2023435 - ANUSHKA KONDKAR
UCE2023445 - BHARGAVEE MUJBAILE
UCE2023450 - PARNAVI PATE

PROBLEMS

To build a classification model that accurately identifies fraudulent transactions from normal transactions, despite the class imbalance.

01. **Massive Financial Impact**

Credit card fraud disrupts financial systems

02. **Inefficient Old Systems**

Traditional rule-based systems (e.g., "block transaction if > $1000")

03. **The Need for AI**

adaptive systems that can learn complex patterns

04. **Project Goal**

machine learning model that can accurately detect and flag fraudulent

DATASET

Data Source

Combined data from `fraudTrain.csv` and `fraudTest.csv`, simulating a large, real-world transaction history.

Volume

Contains approximately 1.85 million individual transaction records, providing a robust dataset for model training.

Key Features

Includes transaction amount, merchant, category, customer demographics (gender, job, DOB), location, and time.

DATA PREPREPROCESSING

1. Removing Sensitive & Unnecessary Columns

We removed columns like name, address, credit card number, transaction ID etc. because:

2. Converting Date–Time Columns

We converted:

- trans_date_trans_time
- dob

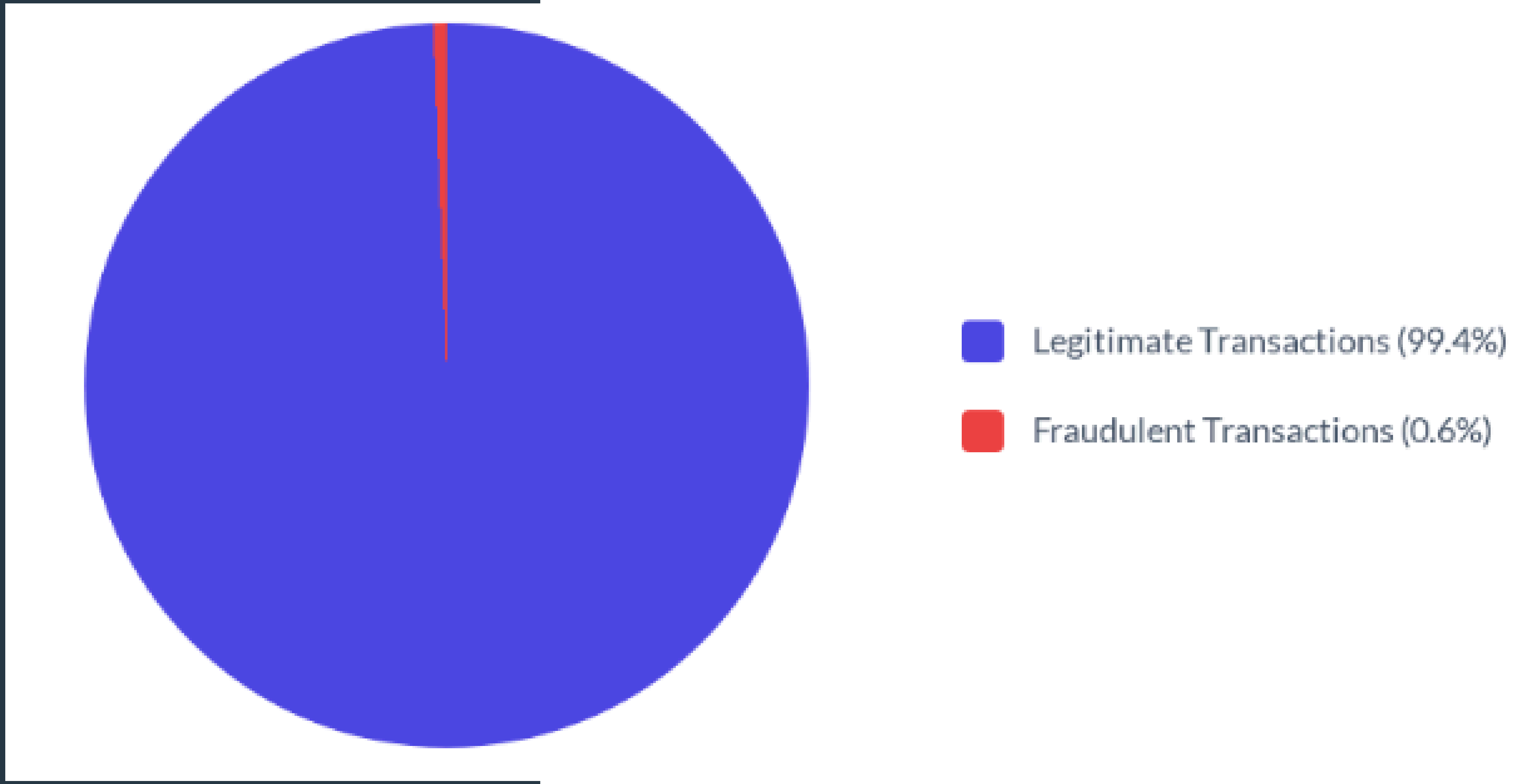
to proper datetime format using pd.to_datetime()

3. Label Encoding

4. Feature Scaling (Standardization)

5. Handling Imbalanced Data (Undersampling)

CHALLENGE: CLASS IMBALANCE



Fraud is rare. A model that only predicts "Legit" would be 99.4% accurate but completely useless. Our main challenge is to correctly identify the tiny 0.6% of fraudulent cases.

MODELS USED

Model 1: K-Nearest Neighbors (KNN)

We first trained a KNN model (n_neighbors=7) on an undersampled dataset (where we randomly removed 'Legit' transactions to equal the 'Fraud' count). This helped the model learn the fraud pattern, showing good initial recall but potential for false positives.

Model 2: Logistic Regression

We also trained a Logistic Regression model on the same undersampled data. This model is fast and highly interpretable. We evaluated it using the ROC-AUC curve, which showed a strong ability to distinguish between fraud and legit transactions (a high AUC score).

MODELS USED

3. Support Vector Machine (SVM)

It is a powerful kernel-based method that finds an optimal hyperplane for classification. Its distance-based mechanism means it is highly sensitive to feature magnitude, and thus requires feature scaling

4. Random Forest Classifier

It is an ensemble learning model built from many decision trees. Because its logic is based on conditional splits (not distance), it is inherently robust and does not require feature scaling, simplifying the data pipeline significantly.

MERCHANT CLUSTERING

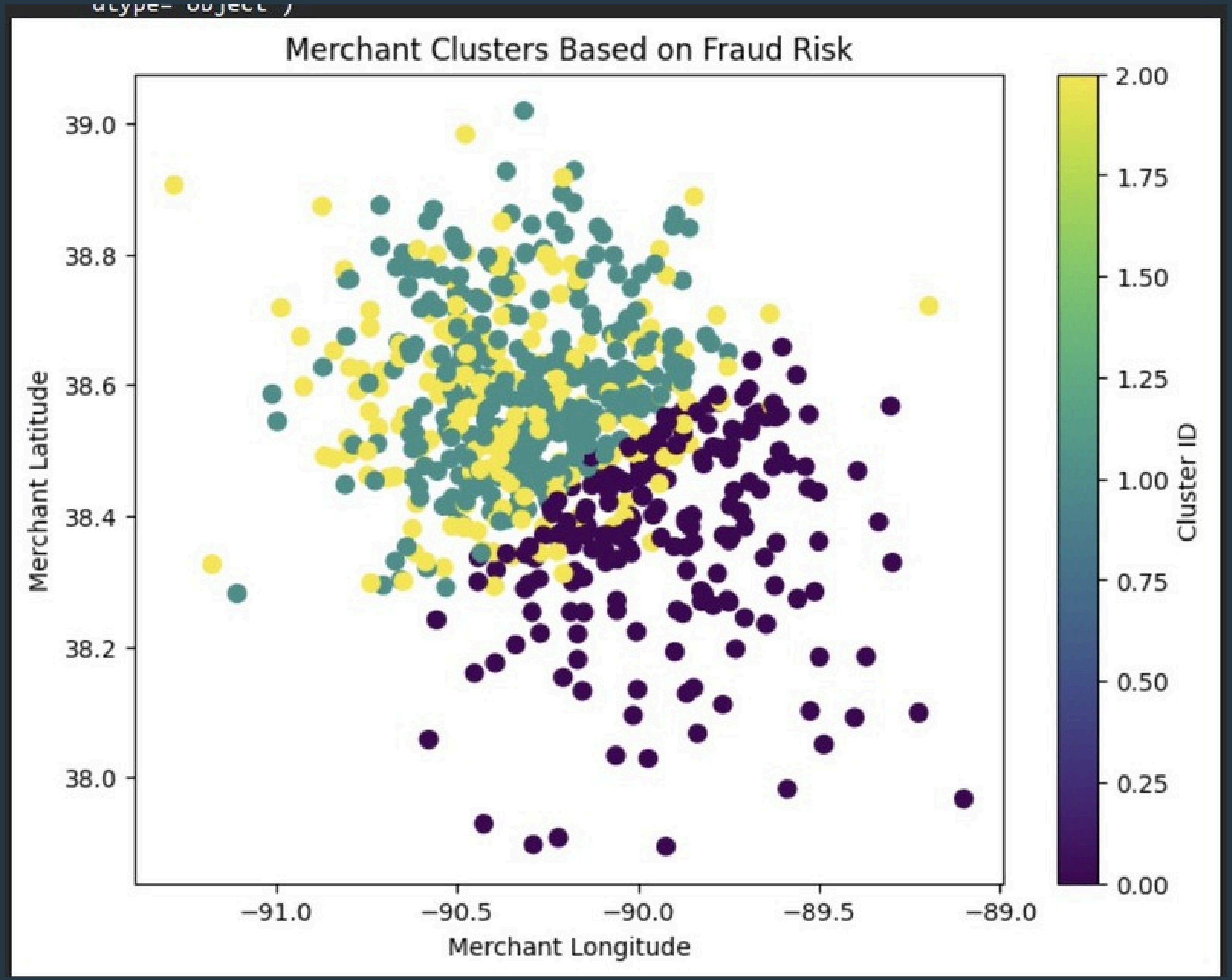
The Insight

Not all merchants are equal. Some merchants might be inherently riskier due to their location, business type, or because they are a frequent target for fraudsters.

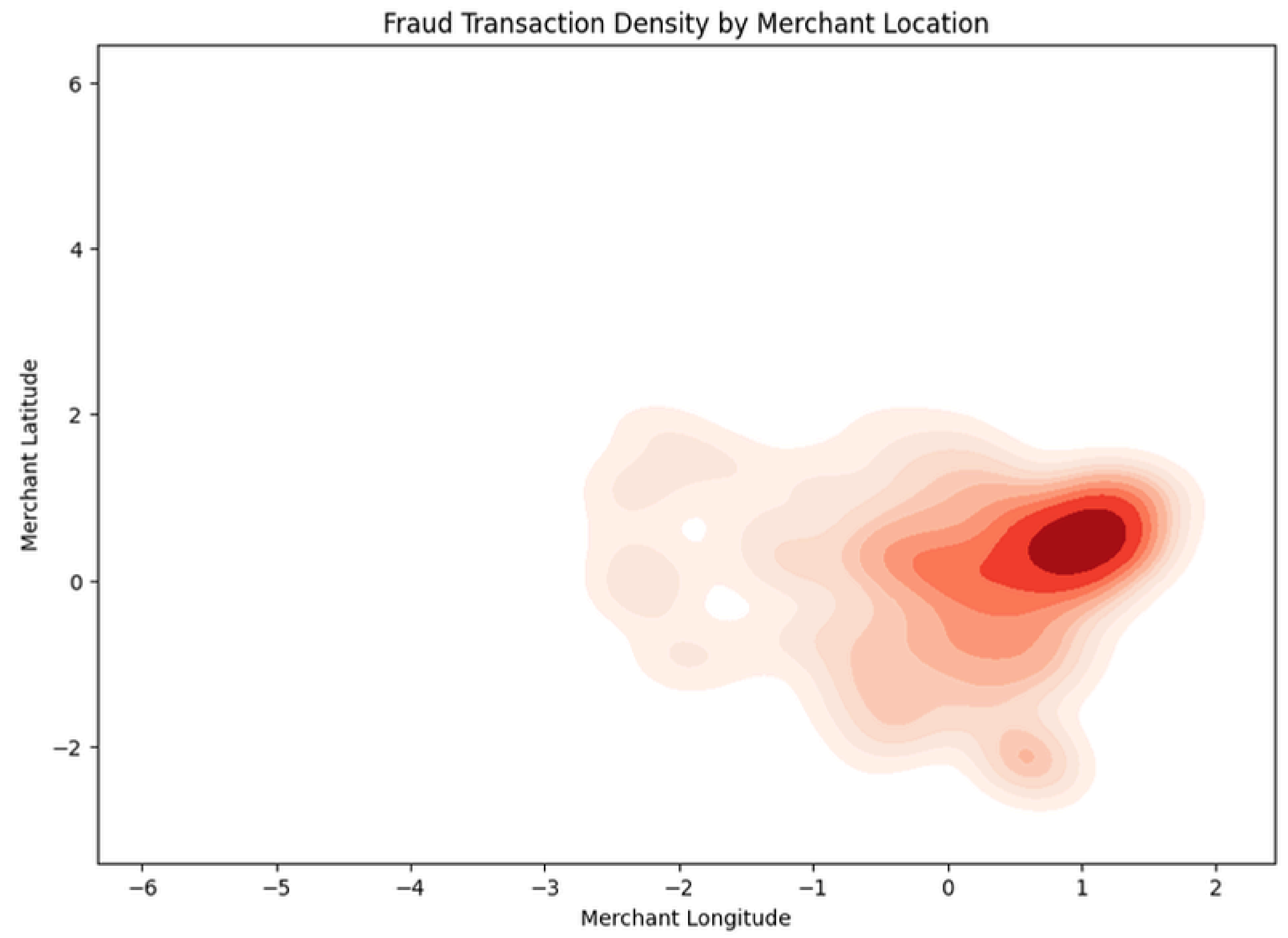
The Method (K-Means)

We used unsupervised learning to group merchants:

- Aggregated data for each merchant (avg. amount, fraud rate, location).
- Used K-Means Clustering to group them into 3 distinct clusters: **(0: Low Risk, 1: Medium Risk, 2: High Risk)**.
- This new `cluster` ID became a powerful new feature for our final model.



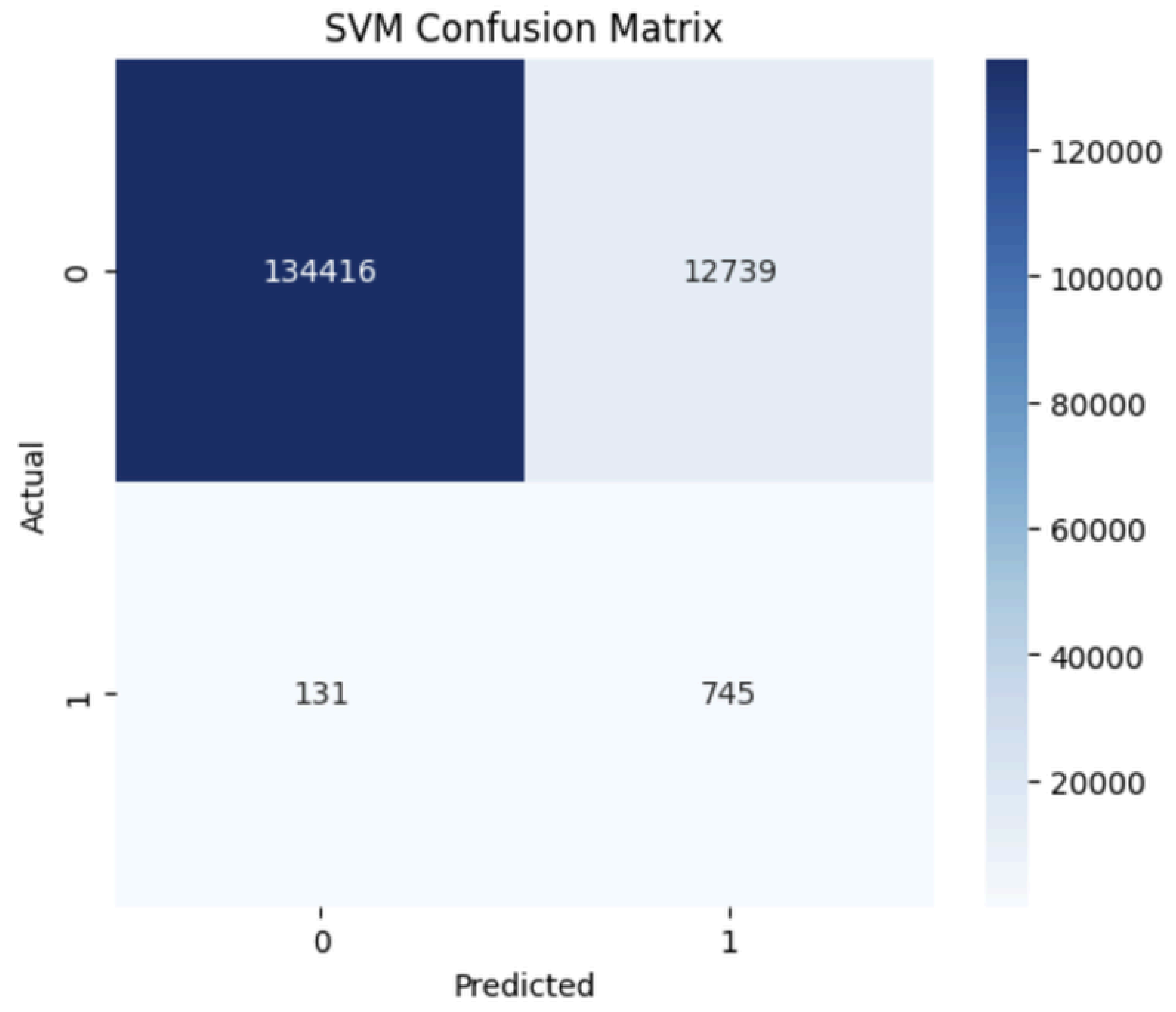
HEATMAP



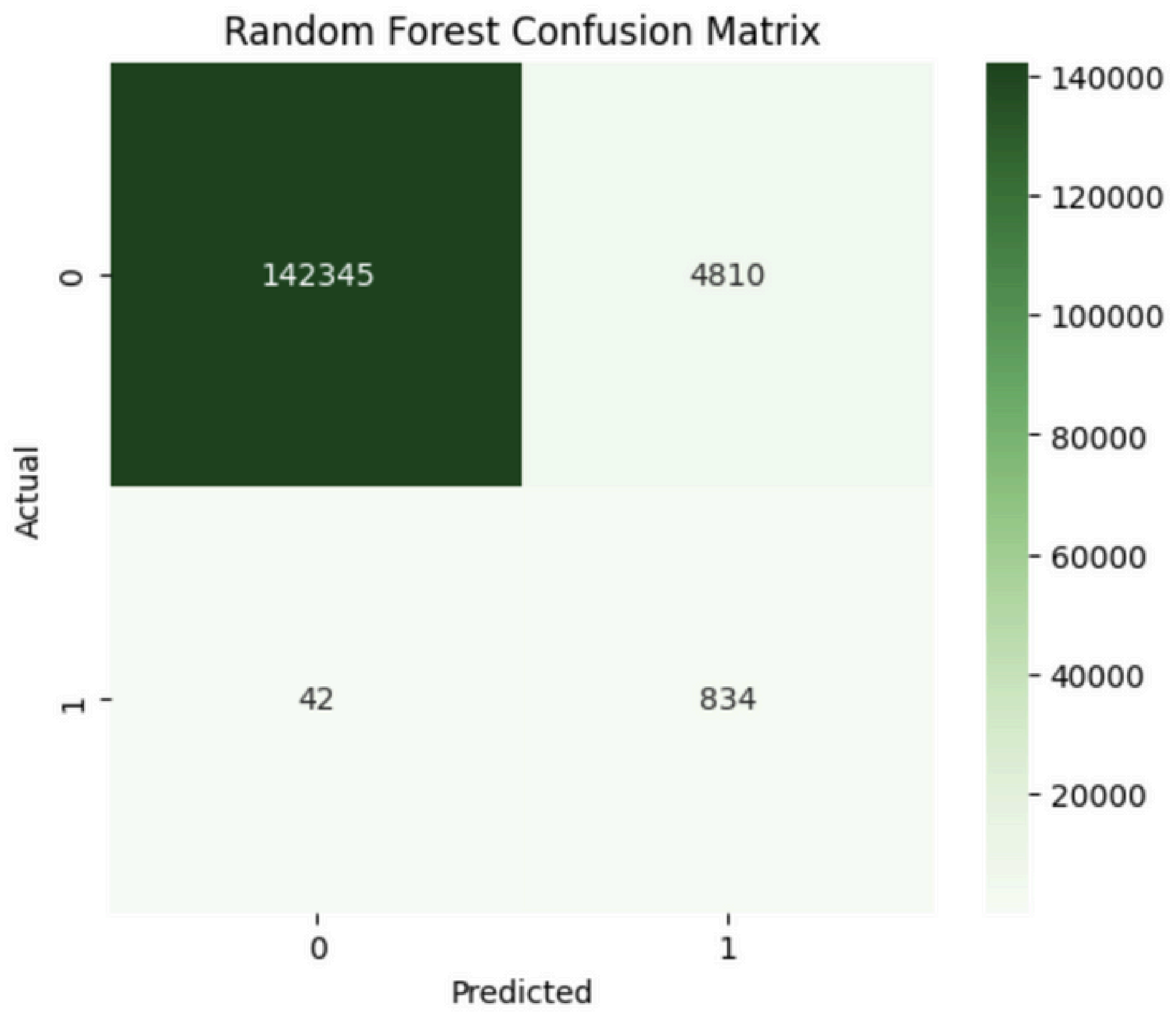
This heatmap illustrates the spatial concentration of fraudulent transactions based on the normalized merchant latitude and longitude. The darker, red areas indicate high-density fraud hotspots, suggesting that the geographical location (and likely the merchant type associated with it) is a critical feature for identifying fraudulent activity.

RESULTS: SVM VS. RANDOM FOREST

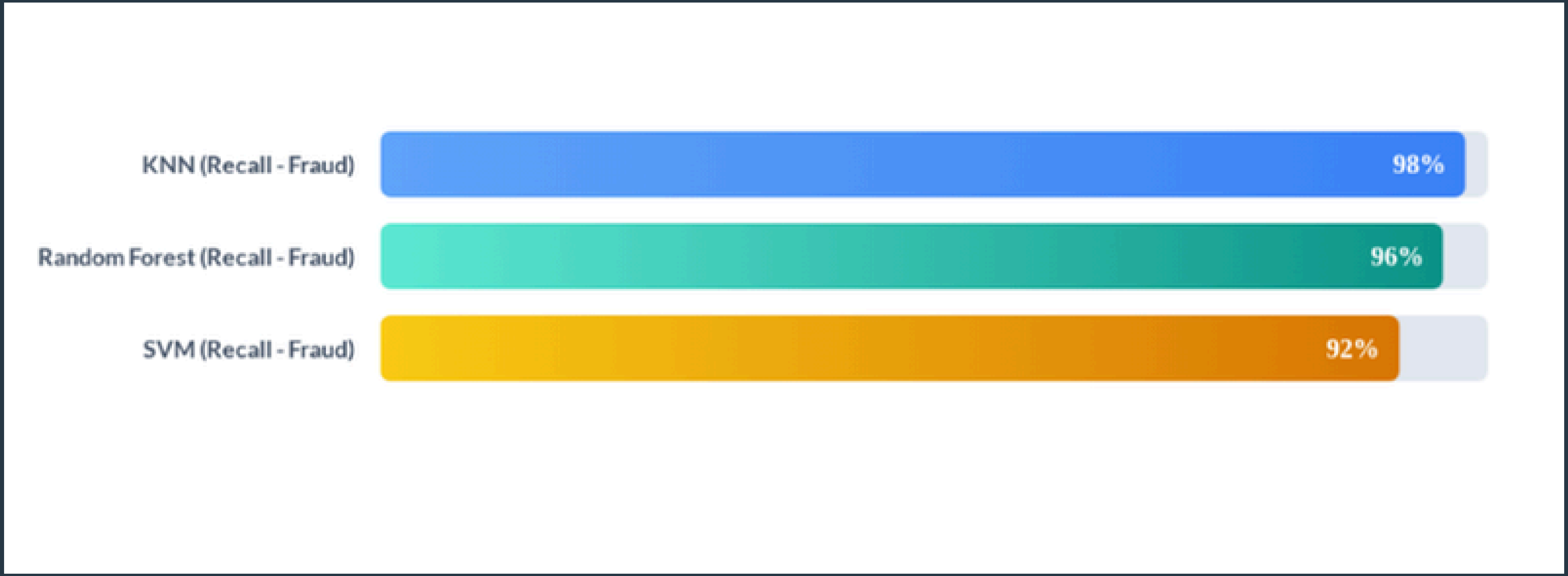
Support Vector Machine (SVM)



Random Forest (RF)



MODEL COMPARISON: FINAL RESULTS



SYSTEM DEMO (GRADIO UI)

Live Prediction Interface

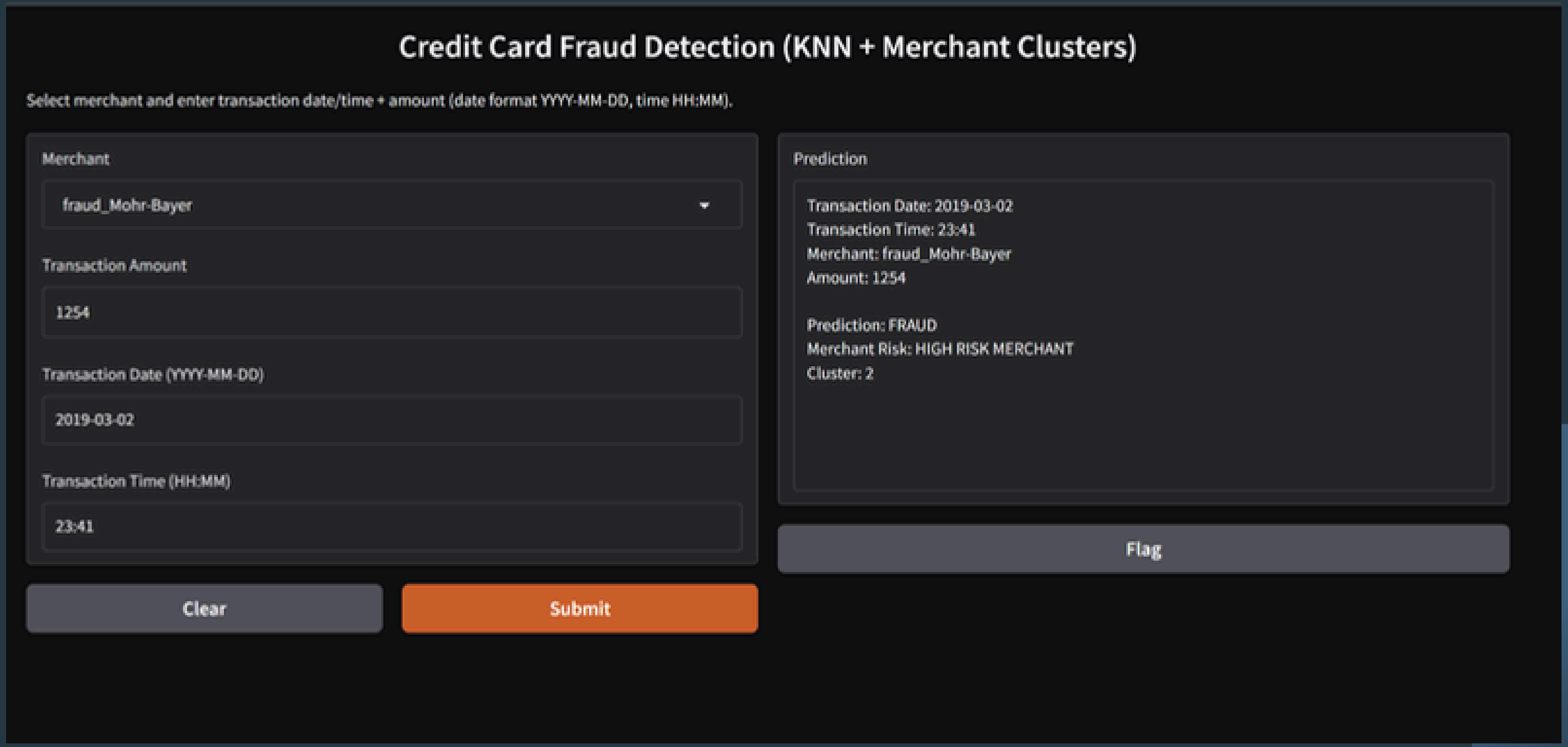
To make the model usable, we built a simple web interface using `Gradio`.

- **Inputs:** The user provides Merchant Name, Amount, Date, and Time.
- Process :

  1. The app looks up the merchant's pre-calculated features (cluster, lat, long, etc.).

  2.It combines these with the user's inputs ('amt', 'hour', 'day').

  3. The complete data row is scaled and fed into the trained KNN model.
- **Output:** The UI returns a real-time prediction ("FRAUD" or "LEGIT"), the merchant's risk level, and the model's confidence probability



Credit Card Fraud Detection (KNN + Merchant Clusters)

Select merchant and enter transaction date/time + amount (date format YYYY-MM-DD, time HH:MM).

Merchant: fraud_Mohr-Bayer

Transaction Amount: 1254

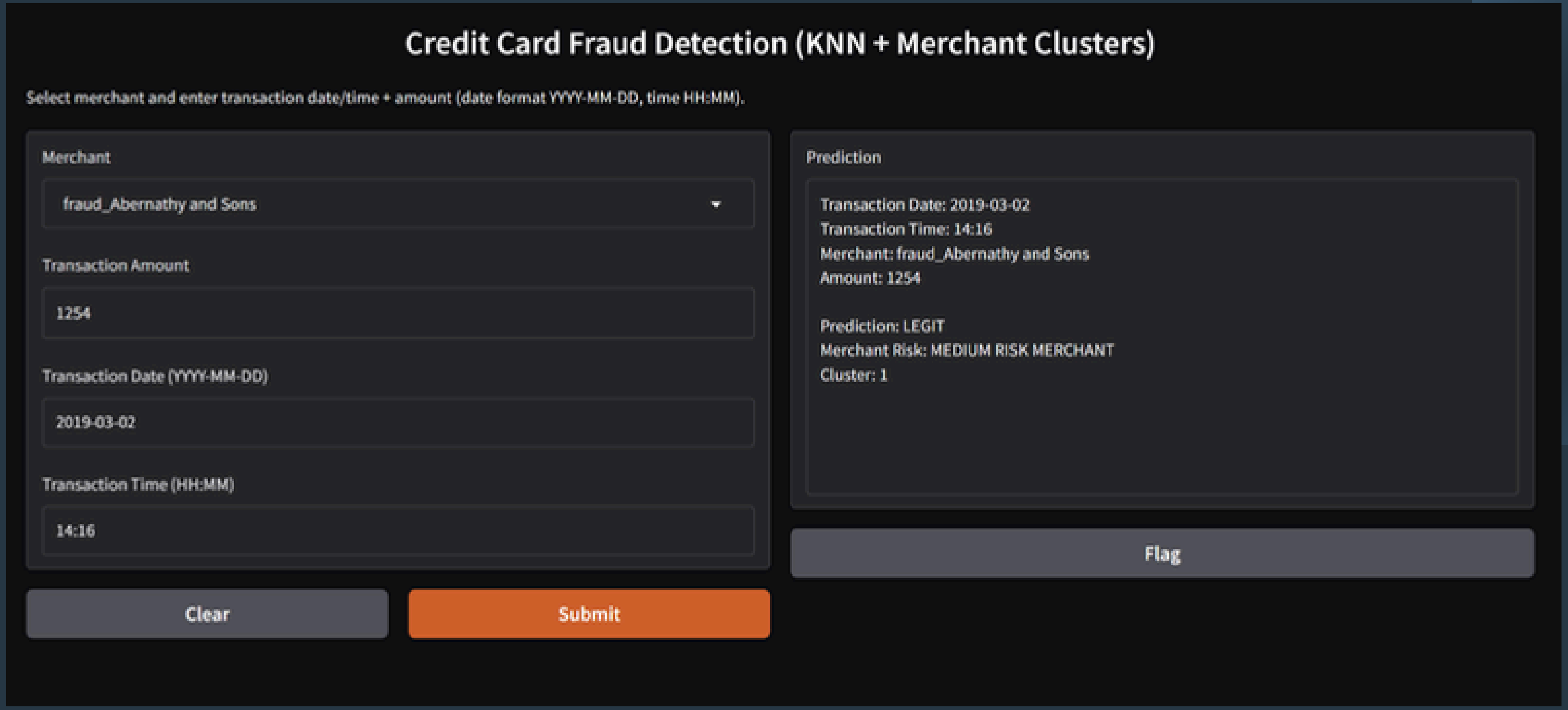
Transaction Date (YYYY-MM-DD): 2019-03-02

Transaction Time (HH:MM): 23:41

Prediction

```
Transaction Date: 2019-03-02
Transaction Time: 23:41
Merchant: fraud_Mohr-Bayer
Amount: 1254

Prediction: FRAUD
Merchant Risk: HIGH RISK MERCHANT
Cluster: 2
```



Credit Card Fraud Detection (KNN + Merchant Clusters)

Select merchant and enter transaction date/time + amount (date format YYYY-MM-DD, time HH:MM).

Merchant: fraud_Abernathy and Sons

Transaction Amount: 1254

Transaction Date (YYYY-MM-DD): 2019-03-02

Transaction Time (HH:MM): 14:16

Prediction

```
Transaction Date: 2019-03-02
Transaction Time: 14:16
Merchant: fraud_Abernathy and Sons
Amount: 1254

Prediction: LEGIT
Merchant Risk: MEDIUM RISK MERCHANT
Cluster: 1
```

CONCLUSION

- We successfully built and benchmarked multiple ML models (KNN, SVM, RF) on a feature-rich, balanced dataset.
- **Key Finding:** Engineering features based on *merchant risk* (using clustering) was the most significant factor in boosting model performance.
- **Future Work:** Experiment with more complex models like Gradient Boosting (XGBoost) to potentially improve accuracy even further.
- Future Work: Add more time-series features (e.g., "time since customer's last transaction") and deploy the model as a scalable microservice API.

THANK YOU