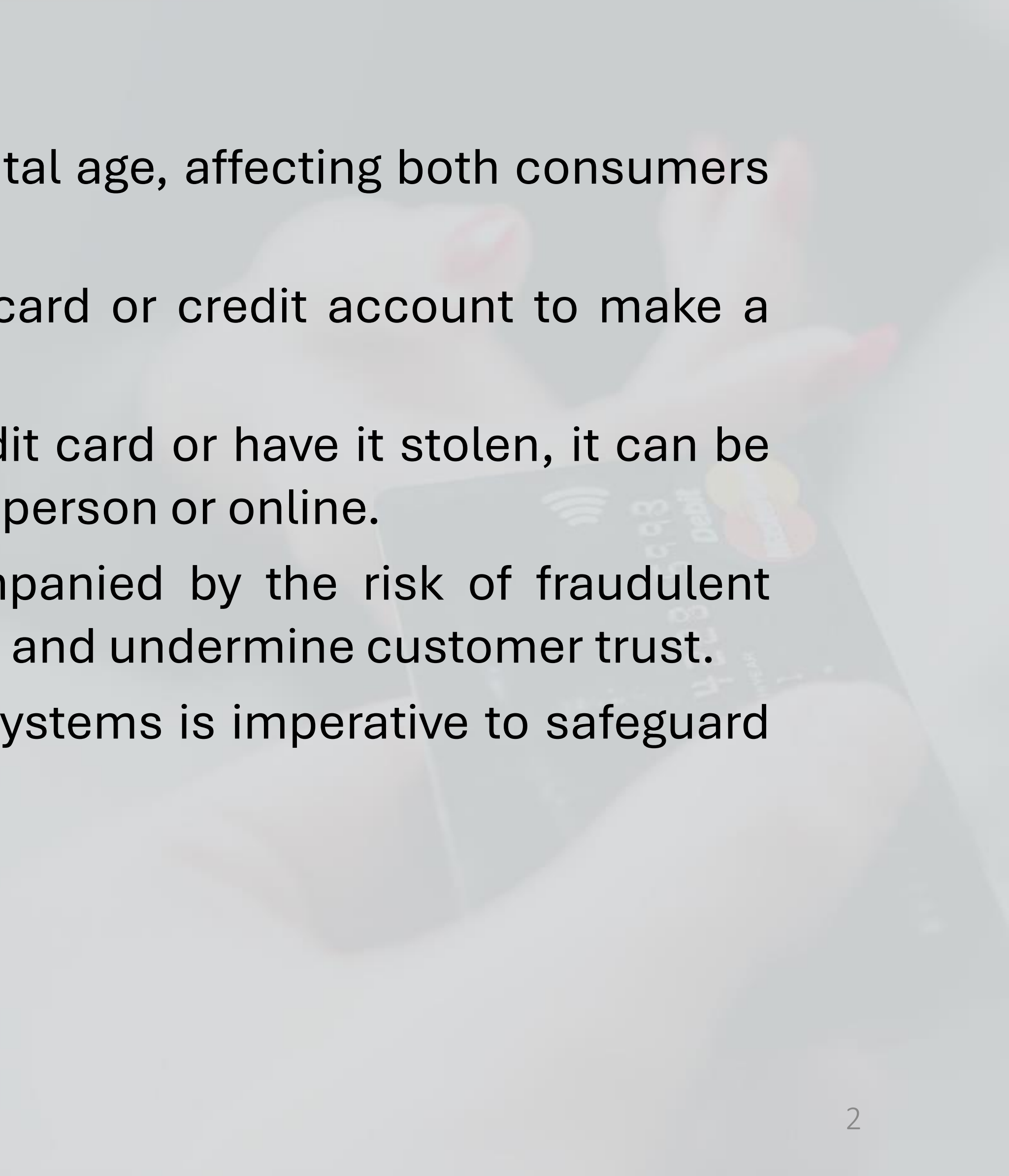
CREDIT CARD FRAUD DETECTION

By Team Members:

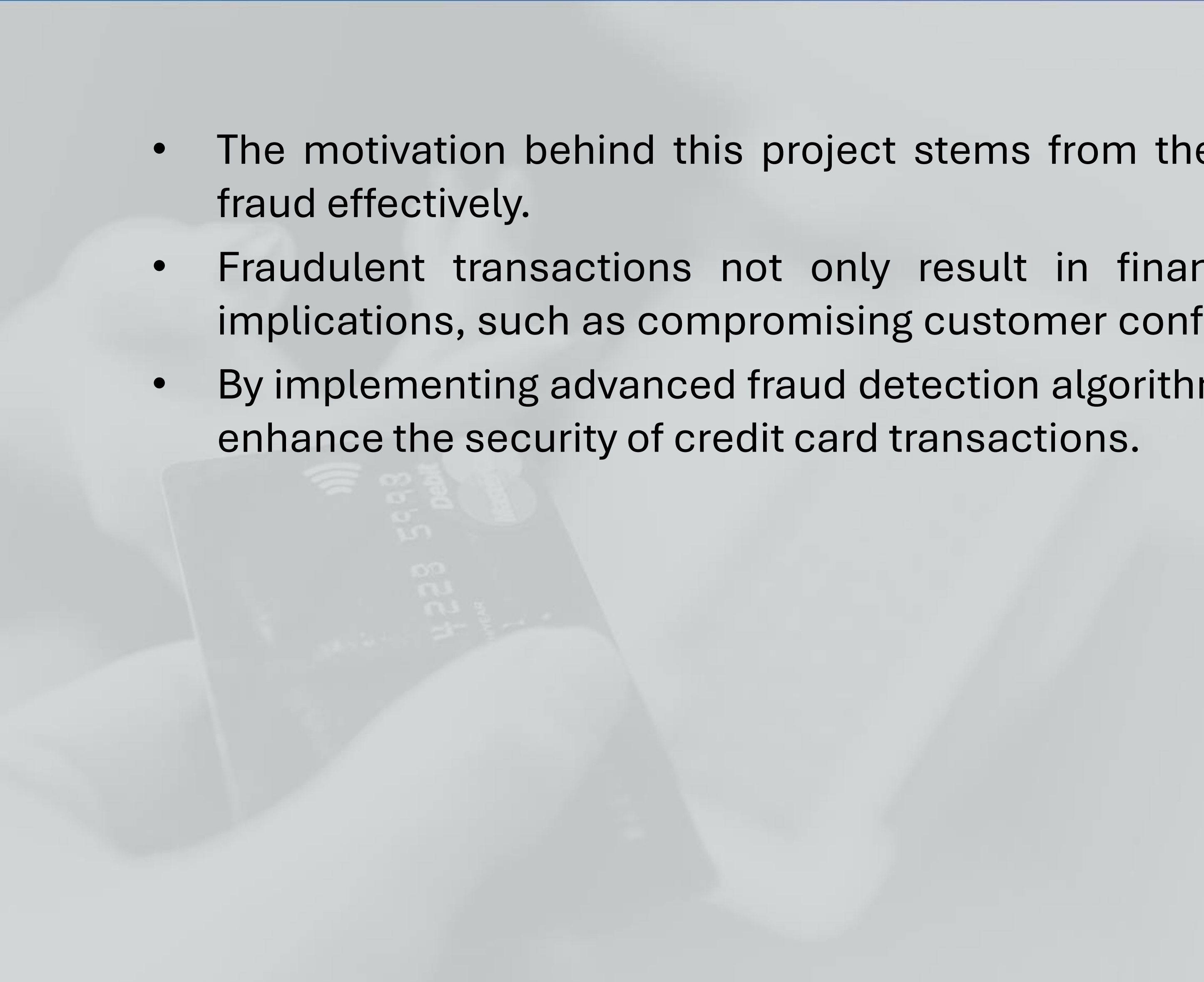
- Brandon Sharp
- Manpreet Kaur
- Shivani Shrivastav
- Sarshaar Mohammed
- Tyler Keating

Introduction



- Credit card fraud is a significant concern in today's digital age, affecting both consumers and financial institutions.
- Credit card fraud is when someone uses your credit card or credit account to make a purchase you did not authorize.
- This can happen in different ways: If you lose your credit card or have it stolen, it can be used to make purchases or other transactions, either in person or online.
- The convenience of electronic transactions is accompanied by the risk of fraudulent activities, which can lead to substantial financial losses and undermine customer trust.
- Therefore, the development of robust fraud detection systems is imperative to safeguard the integrity of electronic payment systems.

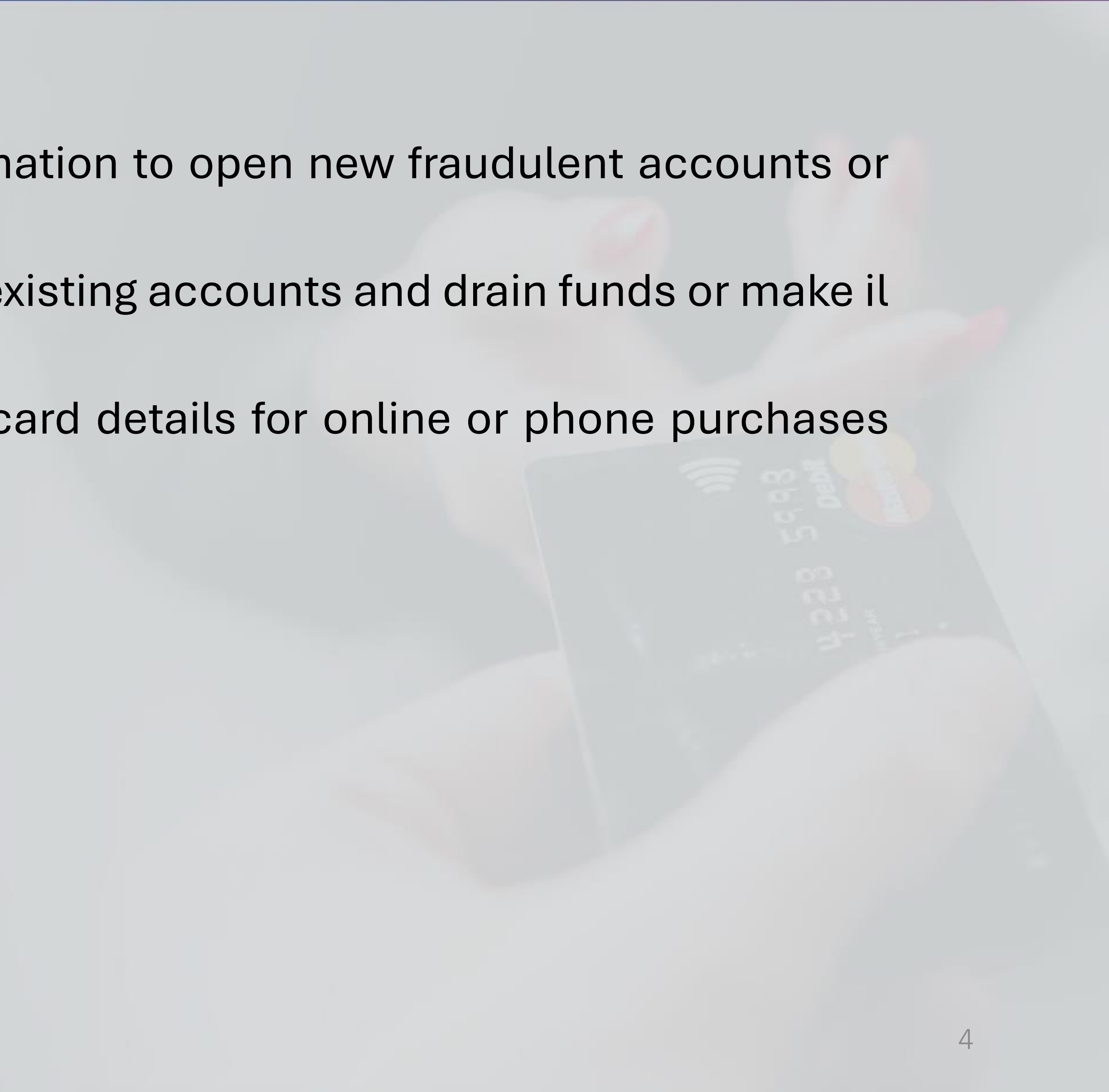
Motivation/Background



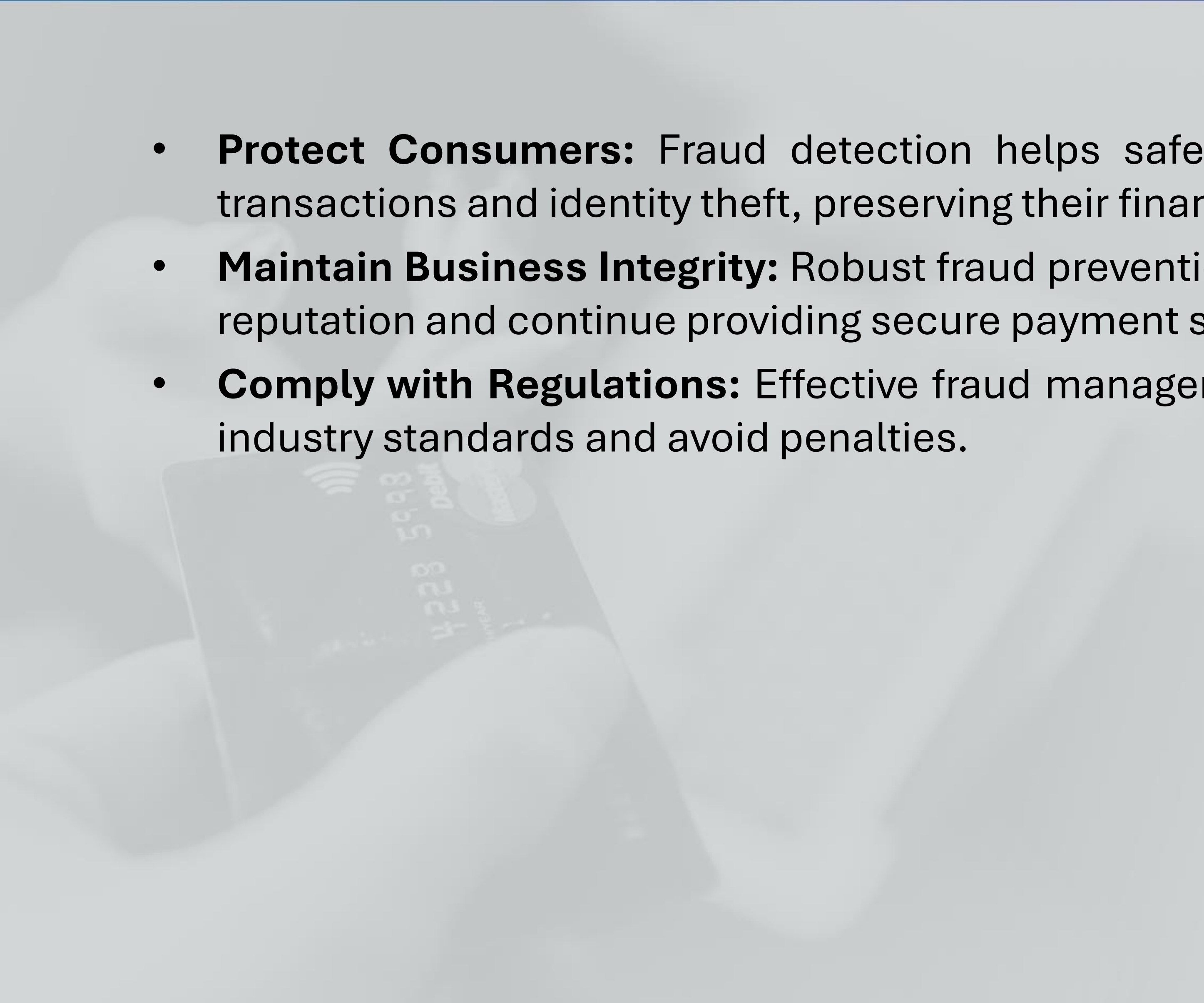
- The motivation behind this project stems from the critical need to combat credit card fraud effectively.
- Fraudulent transactions not only result in financial losses but also have broader implications, such as compromising customer confidence in electronic payment systems.
- By implementing advanced fraud detection algorithms, we aim to mitigate these risks and enhance the security of credit card transactions.

Common Fraud Techniques/Trends

- **Identity Theft:** Criminals steal personal information to open new fraudulent accounts or make unauthorized purchases.
- **Account Takeover:** Fraudsters gain access to existing accounts and drain funds or make il legal transactions.
- **Card-Not-Present Fraud:** Thieves use stolen card details for online or phone purchases without the physical card.

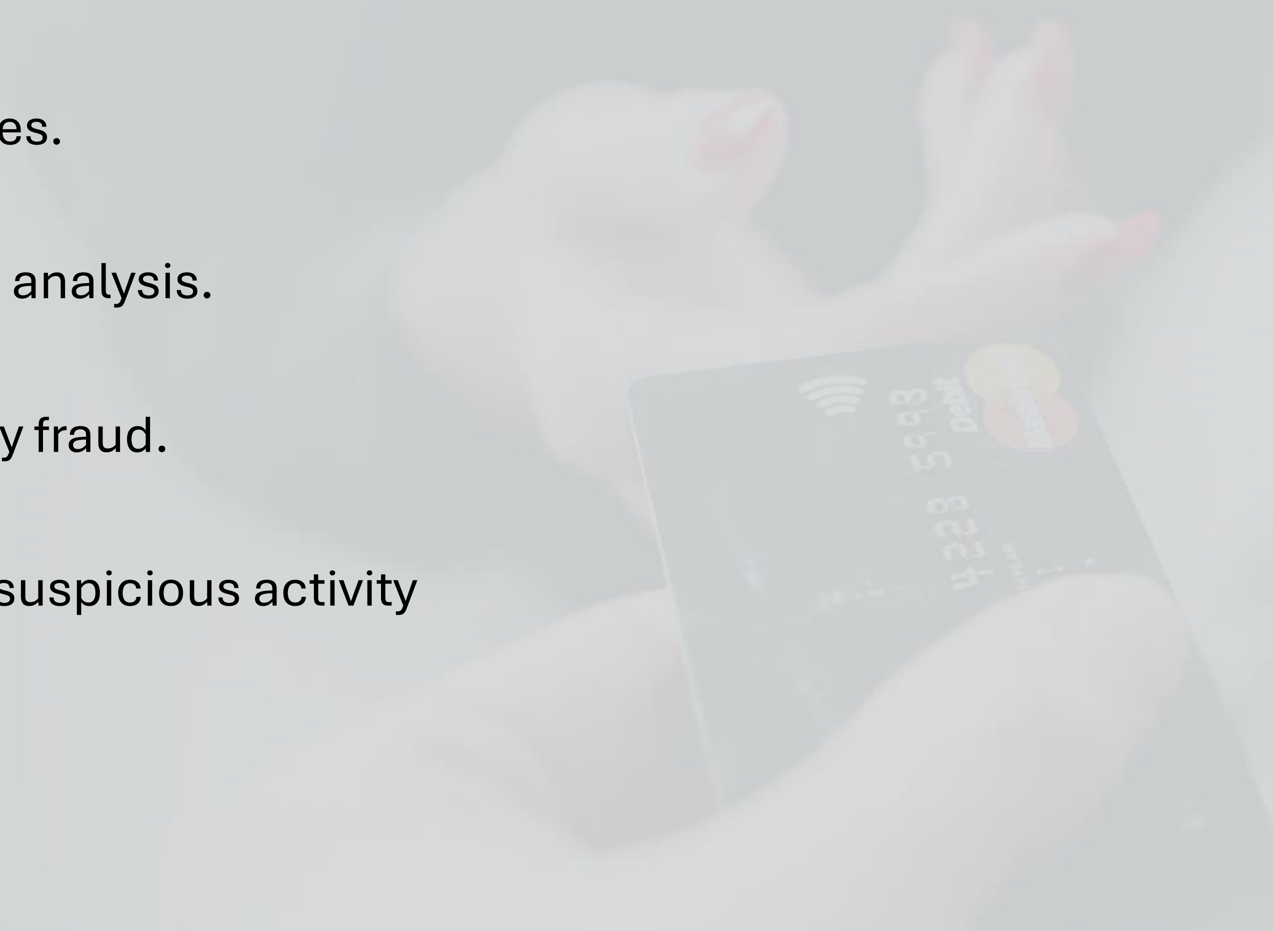


Importance of Fraud Detection



- **Protect Consumers:** Fraud detection helps safeguard customers from unauthorized transactions and identity theft, preserving their financial well-being.
- **Maintain Business Integrity:** Robust fraud prevention enables companies to uphold their reputation and continue providing secure payment services.
- **Comply with Regulations:** Effective fraud management is often required by law to meet industry standards and avoid penalties.

Implementing a Fraud Detection System



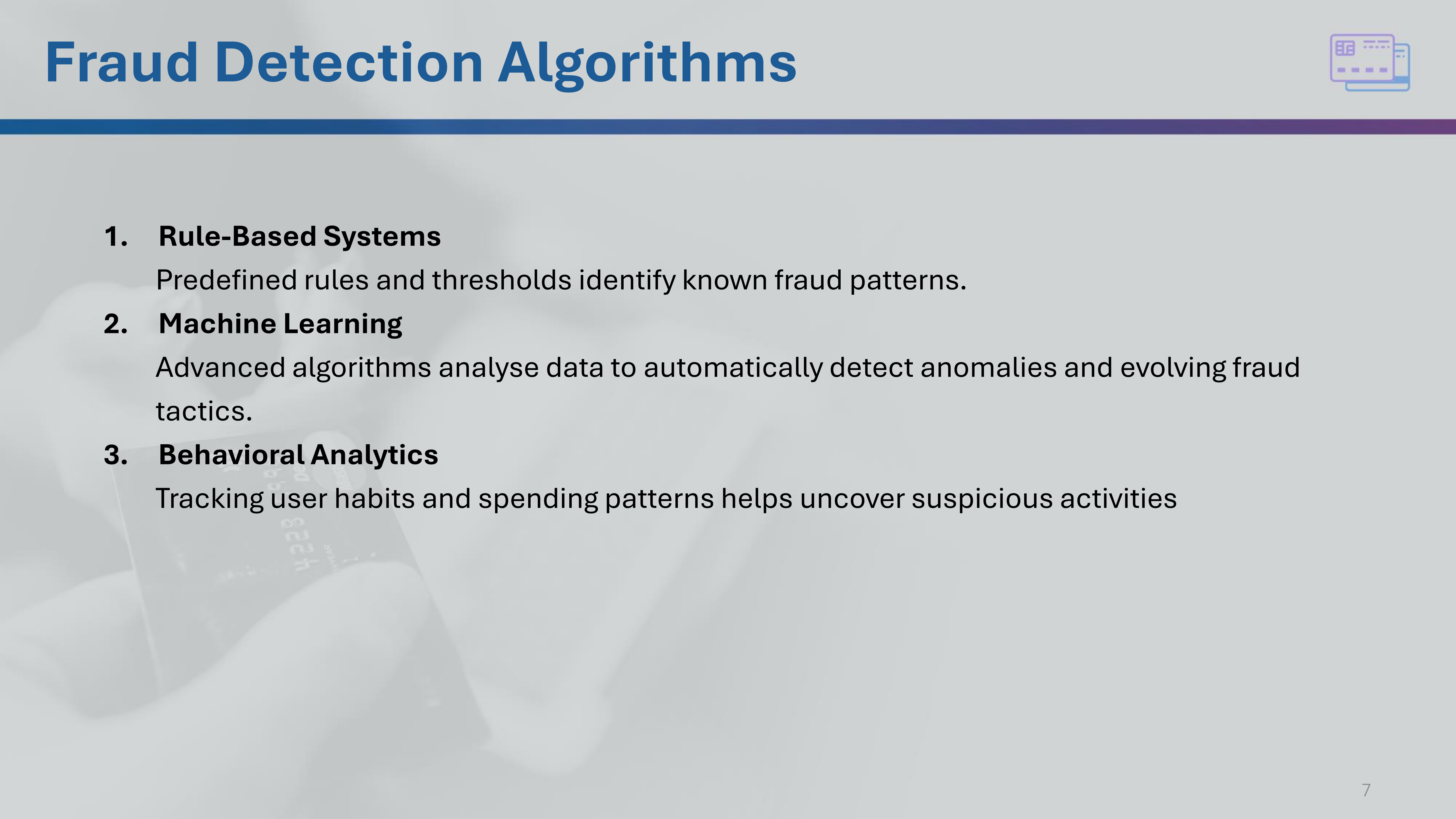
- **Data Collection**

  Gather transaction data from multiple sources.
- **Data Processing**

  Clean, normalize, and transform the data for analysis.
- **Machine Learning**

  Train and deploy predictive models to identify fraud.
- **Real-Time Monitoring**

  Continuously monitor transactions and flag suspicious activity



Fraud Detection Algorithms

1. **Rule-Based Systems**

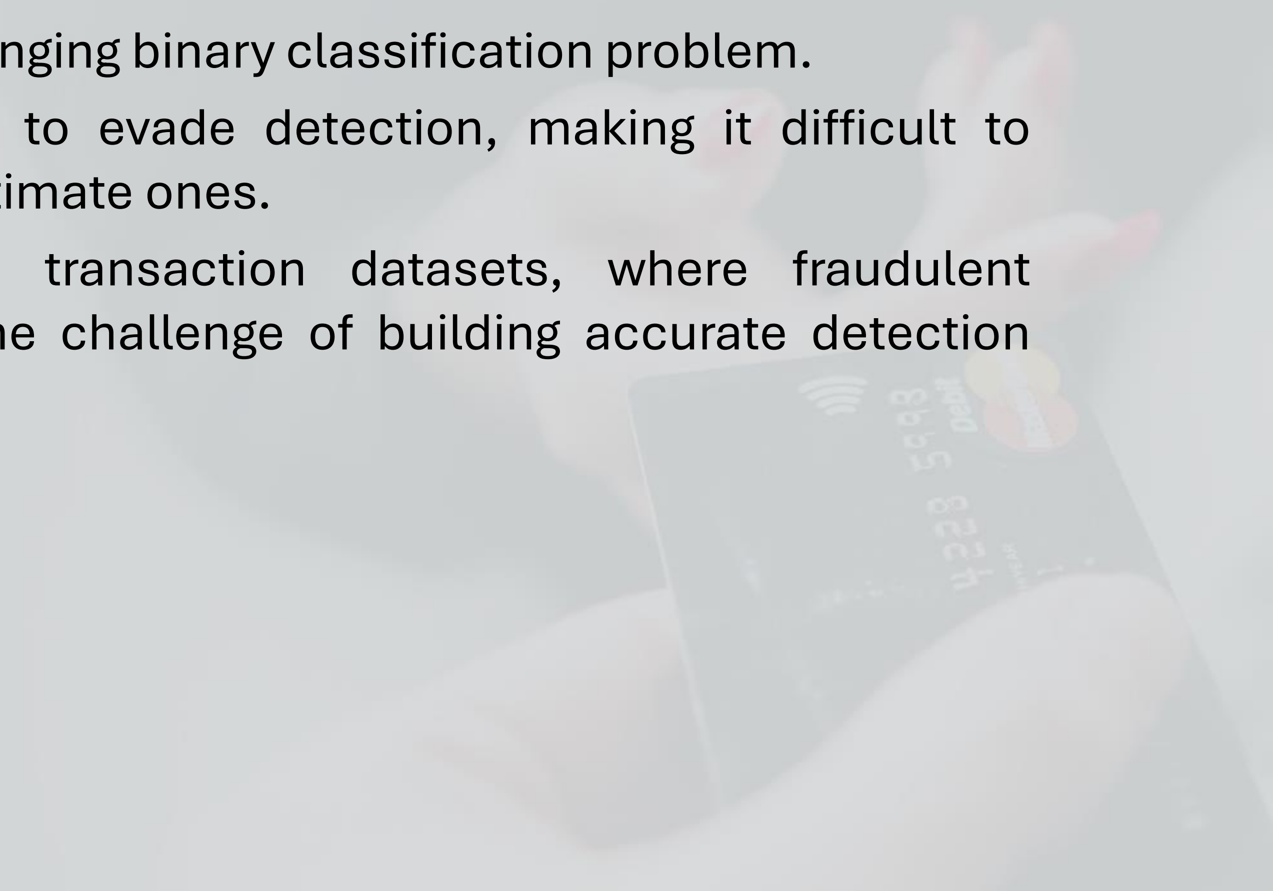
   Predefined rules and thresholds identify known fraud patterns.
2. **Machine Learning**

   Advanced algorithms analyse data to automatically detect anomalies and evolving fraud tactics.
3. **Behavioral Analytics**

   Tracking user habits and spending patterns helps uncover suspicious activities

Problem Setting

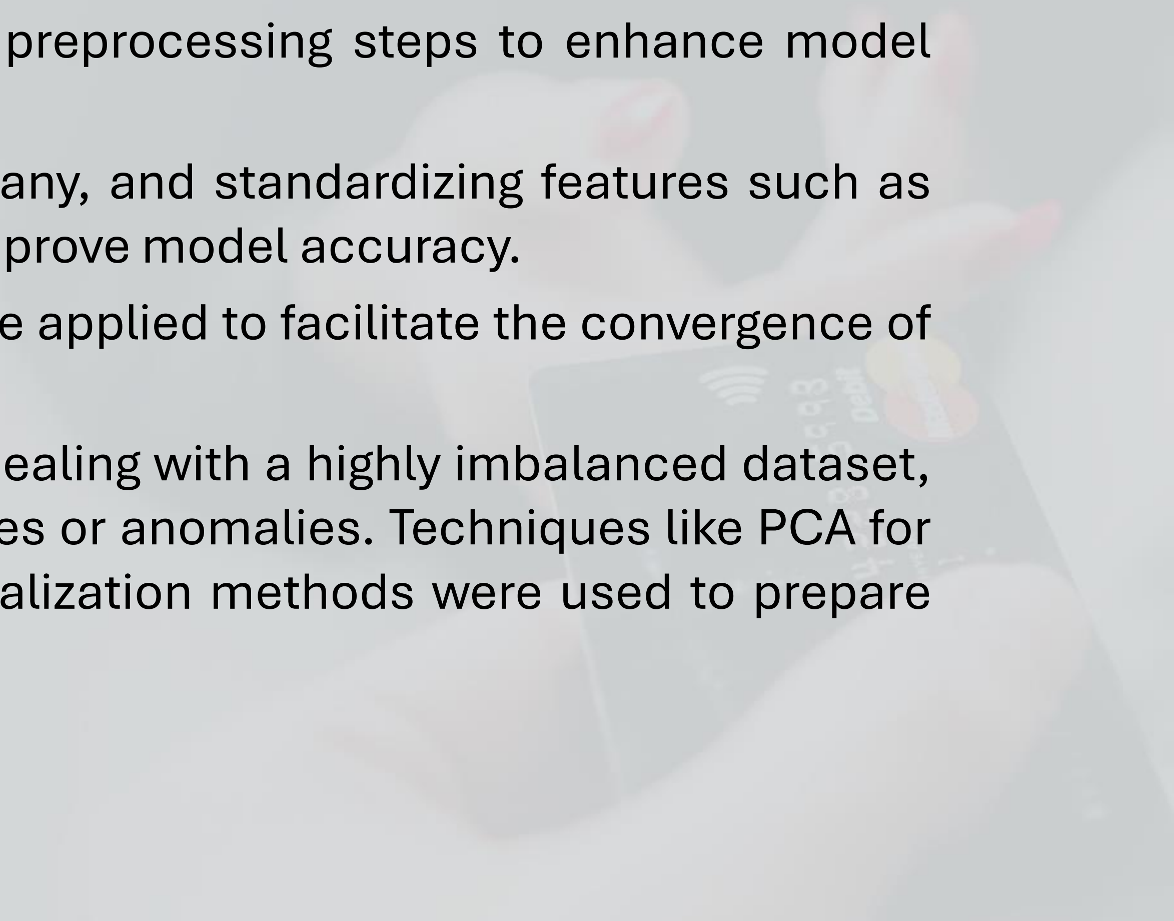
- Credit card fraud detection presents a challenging binary classification problem.
- Fraudsters continually evolve their tactics to evade detection, making it difficult to distinguish fraudulent transactions from legitimate ones.
- Furthermore, the inherent imbalance in transaction datasets, where fraudulent transactions are a minority, exacerbates the challenge of building accurate detection models.



Dataset Description

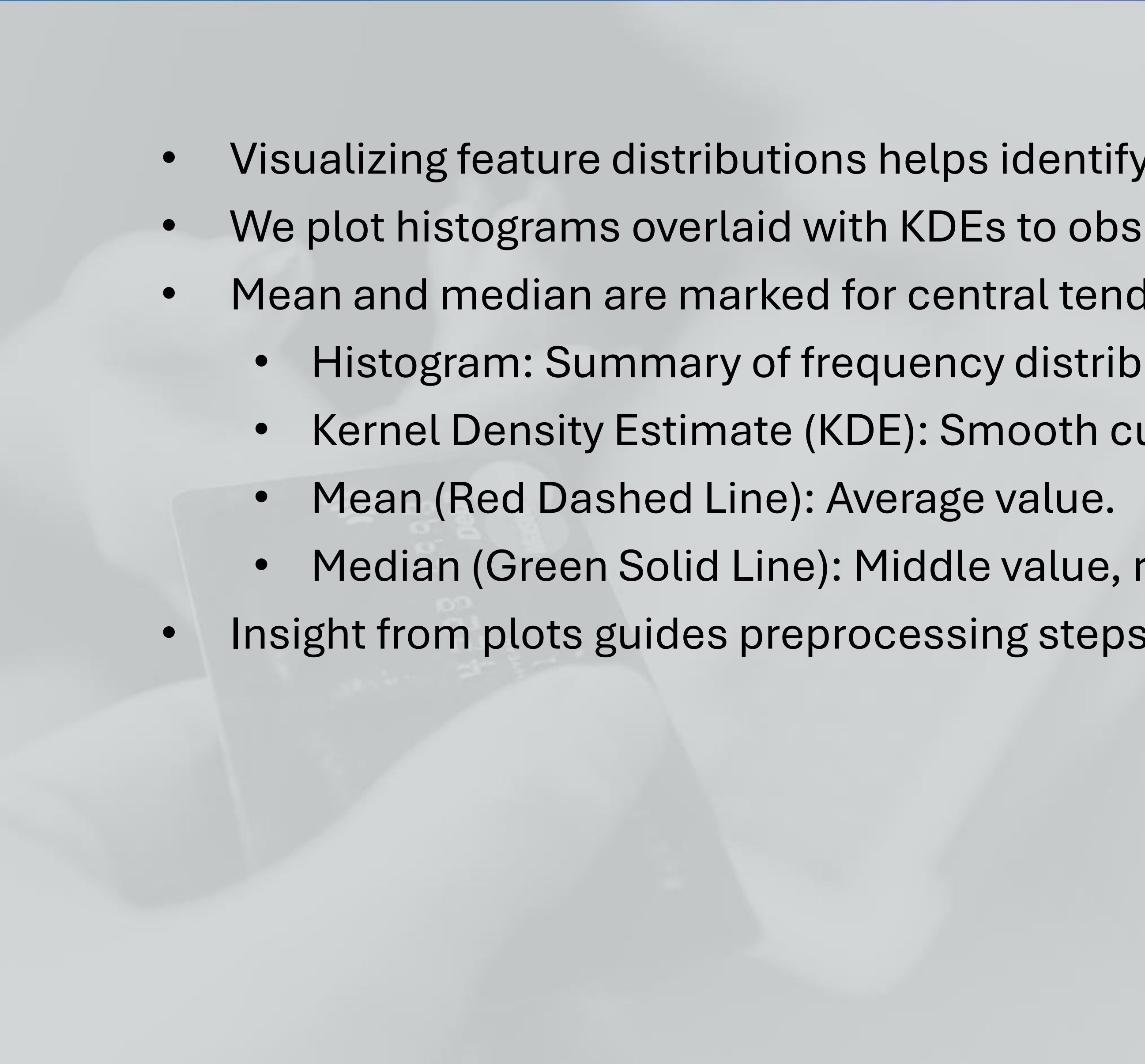
- The dataset utilized in this project consists of credit card transactions made by European cardholders during September 2013.
- It comprises a total of 284,807 transactions, out of which only 492 are identified as fraudulent, representing a highly imbalanced dataset.
- The dataset includes numerical features derived from principal component analysis (PCA), along with 'Time' and 'Amount' variables, and a binary 'Class' variable indicating transaction status.
- The principal components (V1 - V28) are obtained through PCA transformation, preserving the anonymity of sensitive information.
- 'Time’ variable represents the elapsed time between transactions, while 'Amount' denotes the transaction value.
- The 'Class' variable serves as the response variable, with value of 1 indicating a fraudulent transaction and 0 representing a legitimate one.

Preprocessing



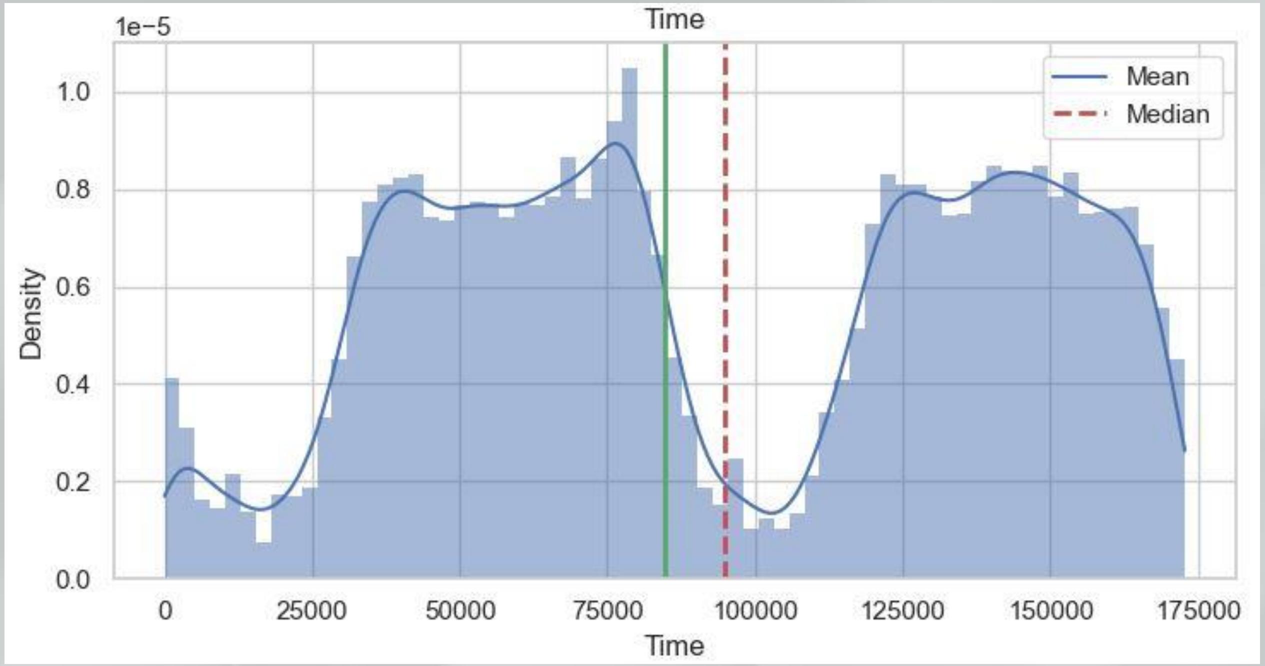
- Prior to model building, the dataset undergoes preprocessing steps to enhance model performance.
- These steps include handling missing values, if any, and standardizing features such as 'Time' and 'Amount' to ensure consistency and improve model accuracy.
- Additionally, feature scaling and normalization are applied to facilitate the convergence of machine learning algorithms.
- The main challenges in preprocessing included dealing with a highly imbalanced dataset, ensuring data privacy, and handling missing values or anomalies. Techniques like PCA for dimensionality reduction and various data normalization methods were used to prepare the data for modeling.

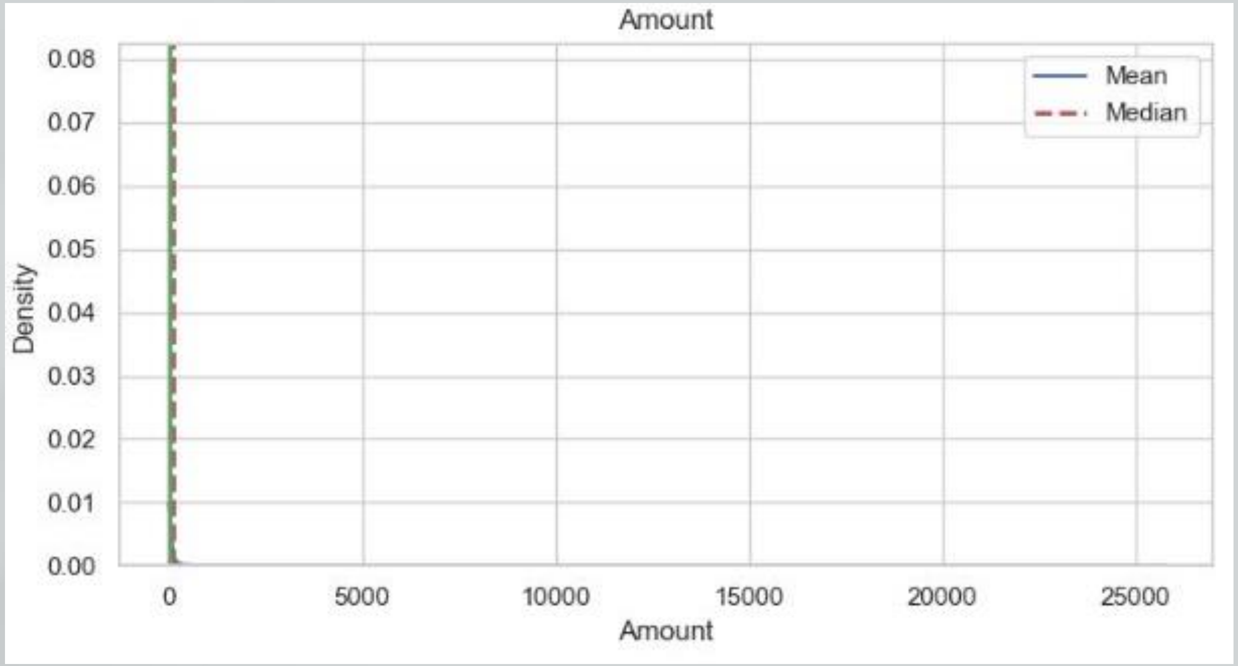
Exploratory Data Analysis (EDA)



- Visualizing feature distributions helps identify irregularities like skewness or outliers.
- We plot histograms overlaid with KDEs to observe distribution shapes.
- Mean and median are marked for central tendency:
  - Histogram: Summary of frequency distribution.
  - Kernel Density Estimate (KDE): Smooth curve estimation
  - Mean (Red Dashed Line): Average value.
  - Median (Green Solid Line): Middle value, robust to outliers.
- Insight from plots guides preprocessing steps and feature engineering for model accuracy

Exploratory Data Analysis (EDA)





Random Forest Classifier (Base Model)

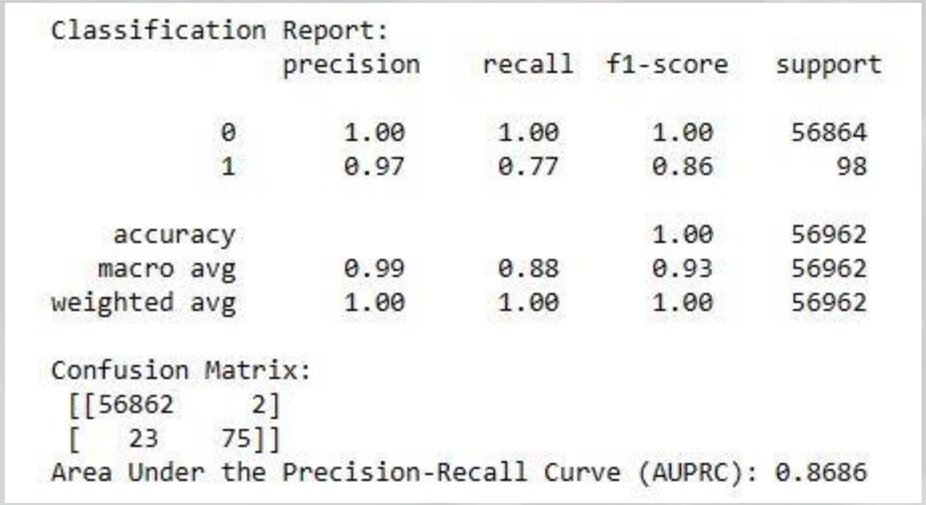
Model Explanation and Implementation:

- The Random Forest algorithm was introduced as an ensemble learning method primarily used for classification and regression tasks.
- It was operated by constructing multiple decision trees during the training phase and outputted the mode of the classes or mean prediction of the individual trees.
- Random Forests correct for decision trees' habit of overfitting to their training set, providing a more generalized model.
- Random Forest was chosen because of its robustness to noise and overfitting, and its ability to handle unbalanced data effectively. It’s also beneficial for handling high dimensionality and complex feature relationships without extensive hyperparameter tuning.
- For our base model, the Random Forest classifier with default parameters provided by the sklearn library was utilized.
- The model was trained using all features in our dataset to predict whether a transaction was fraudulent.
- The dataset was split into training and testing sets to evaluate the model's performance realistically.

Random Forest Classifier (Base Model)

Model Evaluation:

- Given the dataset's imbalance, traditional accuracy metrics might not have provided a true representation of performance. Instead, the focus was on the Area Under the Precision-Recall Curve (AUPRC) as a more indicative measure.
- Additionally, the confusion matrix was examined to understand the trade-offs between false positives and false negatives.
- The base Random Forest model was implemented, and its performance was evaluated.

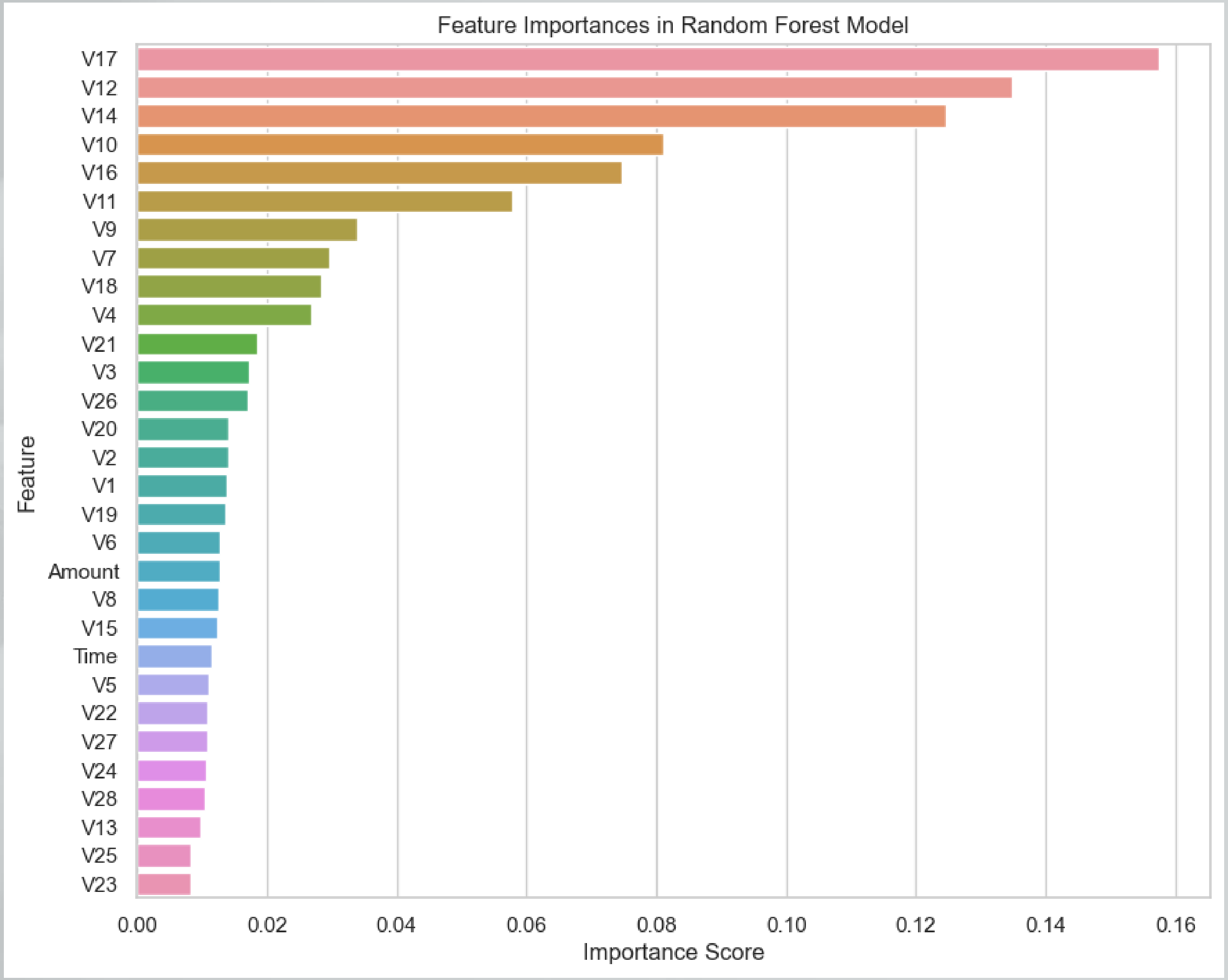


```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56864
           1       0.97      0.77      0.86        98

    accuracy                           1.00     56962
   macro avg       0.99      0.88      0.93     56962
weighted avg       1.00      1.00      1.00     56962

Confusion Matrix:
 [[56862     2]
 [   23    75]]
Area Under the Precision-Recall Curve (AUPRC): 0.8686
```

Feature Importance Analysis

- Feature importance assessment provides insights into model dynamics.
- In Random Forest, features are ranked based on their contribution to predictive accuracy.
- Removal of less important features can enhance model efficiency and interpretability.
- However, careful evaluation is crucial to avoid compromising model performance.
- Visual representation, such as bar charts or heatmaps, facilitate understanding of feature importance.
- By refining our model based on this analysis, complexity and accuracy of model was bala nced.
- Optimization in this case, involves iterative feature selection and model evaluation.
- Balancing model complexity with predictive power ensures robust performance.
- Continuous monitoring and adjustment adapt to evolving data patterns.

Feature Importance Analysis



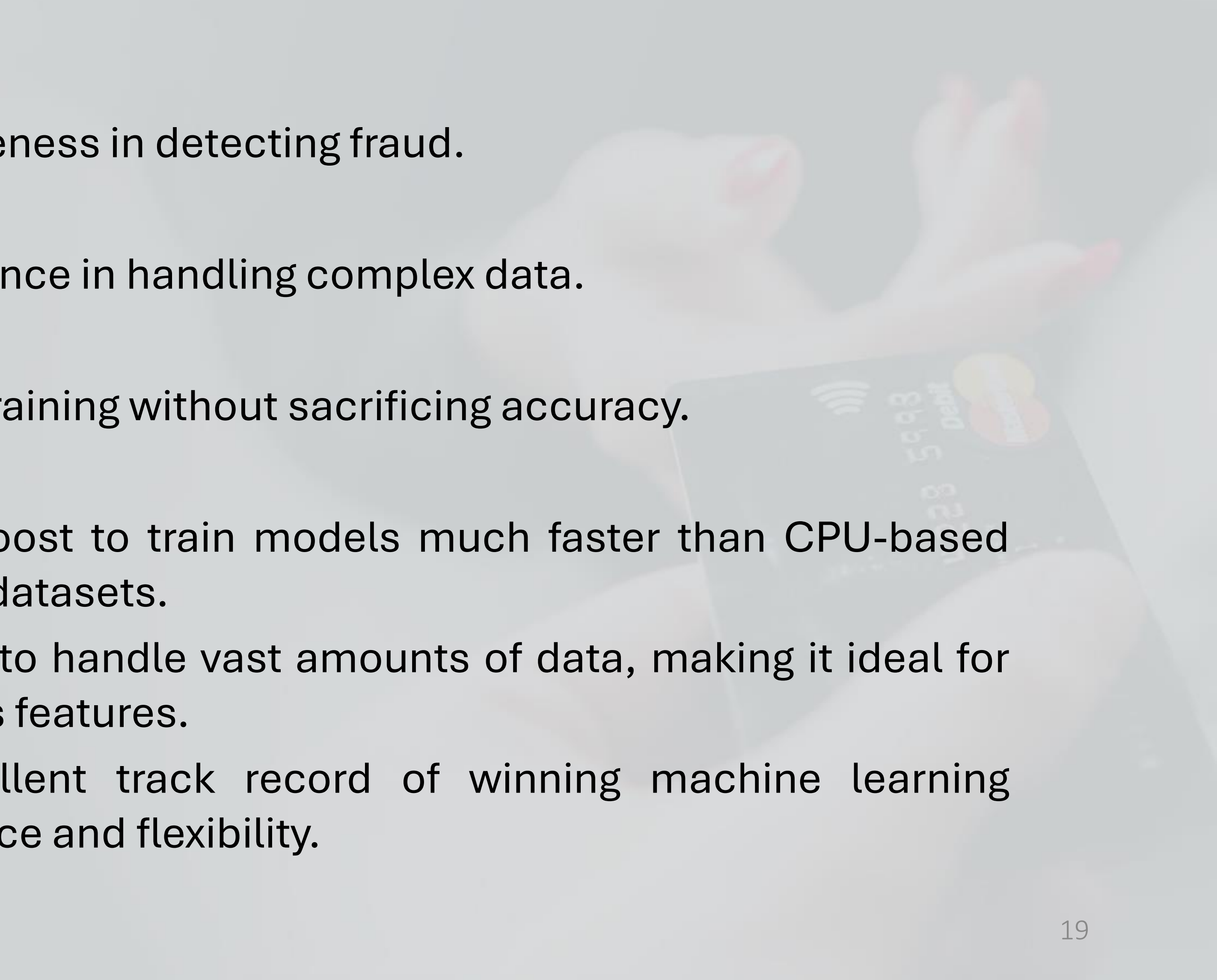
XGBoost Classifier

- Powerful tool used for fraud detection, like a toolbox full of features.
- XGBoost (Extreme Gradient Boosting) is an optimized distributed gradient boosting library).
- It provides parallel computing, regularization, enabled cross verification, missing values, flexibility, availability, save and reload, tree pruning.
- XGBoost belongs to a family of boosting algorithms that convert weak learners into strong learners. A weak learner is one which is slightly better than random guessing.
- Boosting is a sequential process i.e. trees are grown using the information from a previously grown tree one after the other. This process slowly learns from data and tries to improve its prediction in subsequent iterations.

Parameters of XGBoost

- **nrounds[default=100]**
  It controls the maximum number of iterations of trees.
- **eta[default=0.3][range:(0,1)]**
  It controls the learning rate.
- **gamma[default=0][range:(0,Inf)]**
  It controls regularization and prevents overfitting.
- **max_depth[default=6][range:(0,Inf)]**
  It controls the depth of the tree.
- **min_child_weight[default=1][range:(0,Inf)]**
  If the leaf node has a minimum sum of instance weight lower than min_child_weight, the tree splitting stops.
- **sumsample[default=1][range:(0,1)]**
  It controls the number of samples supplied to a tree.
- **colsample_bytree[default=1][range:(0,1)]**
  It control the number of variables supplied to a tree.

Optimizing with GPU-Enabled XGBoost Classifier



Enhancing Fraud Detection Model:

- Sought to improve the model's effectiveness in detecting fraud.

Leveraging XGBoost:

- Chose XGBoost for its proven performance in handling complex data.

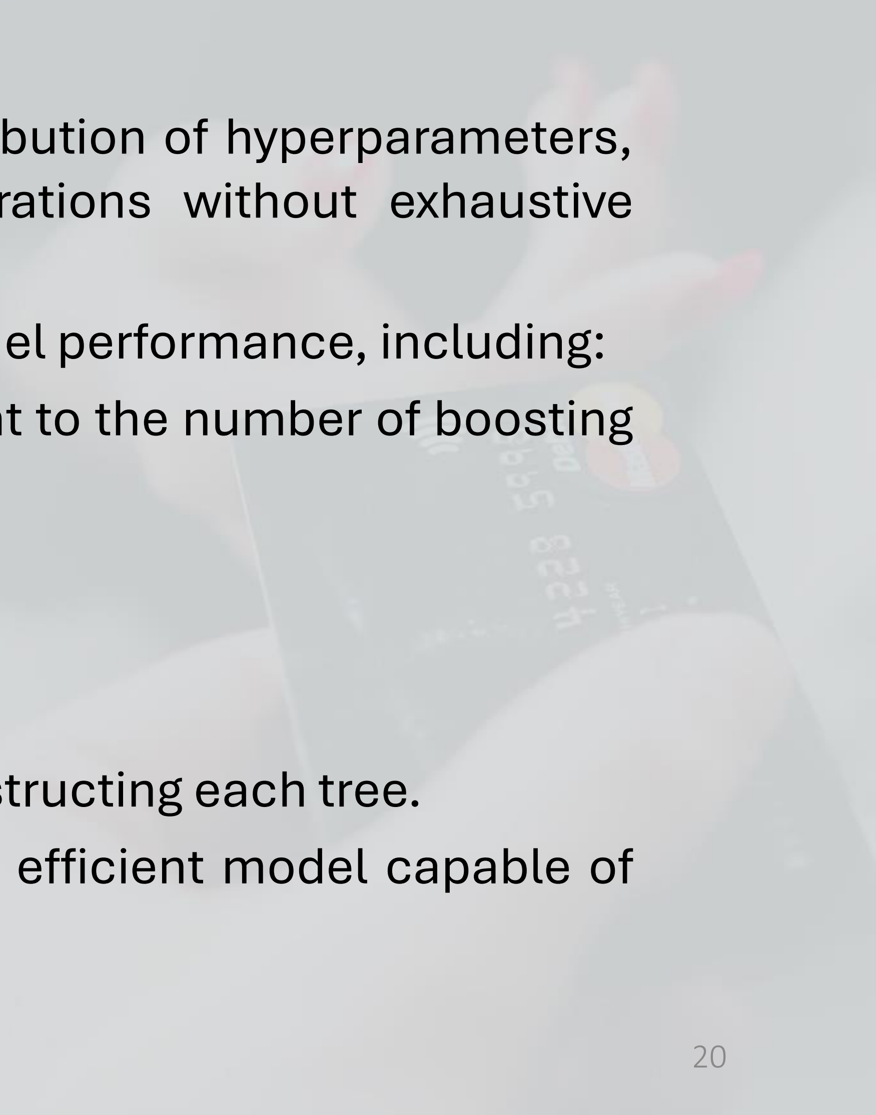
Efficiency with GPU Acceleration:

- Utilized GPU acceleration to expedite training without sacrificing accuracy.

Advantages of GPU-Enabled XGBoost:

- Speed: GPU acceleration allows XGBoost to train models much faster than CPU-based training, especially beneficial for large datasets.
- Scalability: XGBoost efficiently scales to handle vast amounts of data, making it ideal for our fraud detection task with numerous features.
- Performance: XGBoost has an excellent track record of winning machine learning competitions due to its high performance and flexibility.

Optimizing with GPU-Enabled XGBoost Classifier

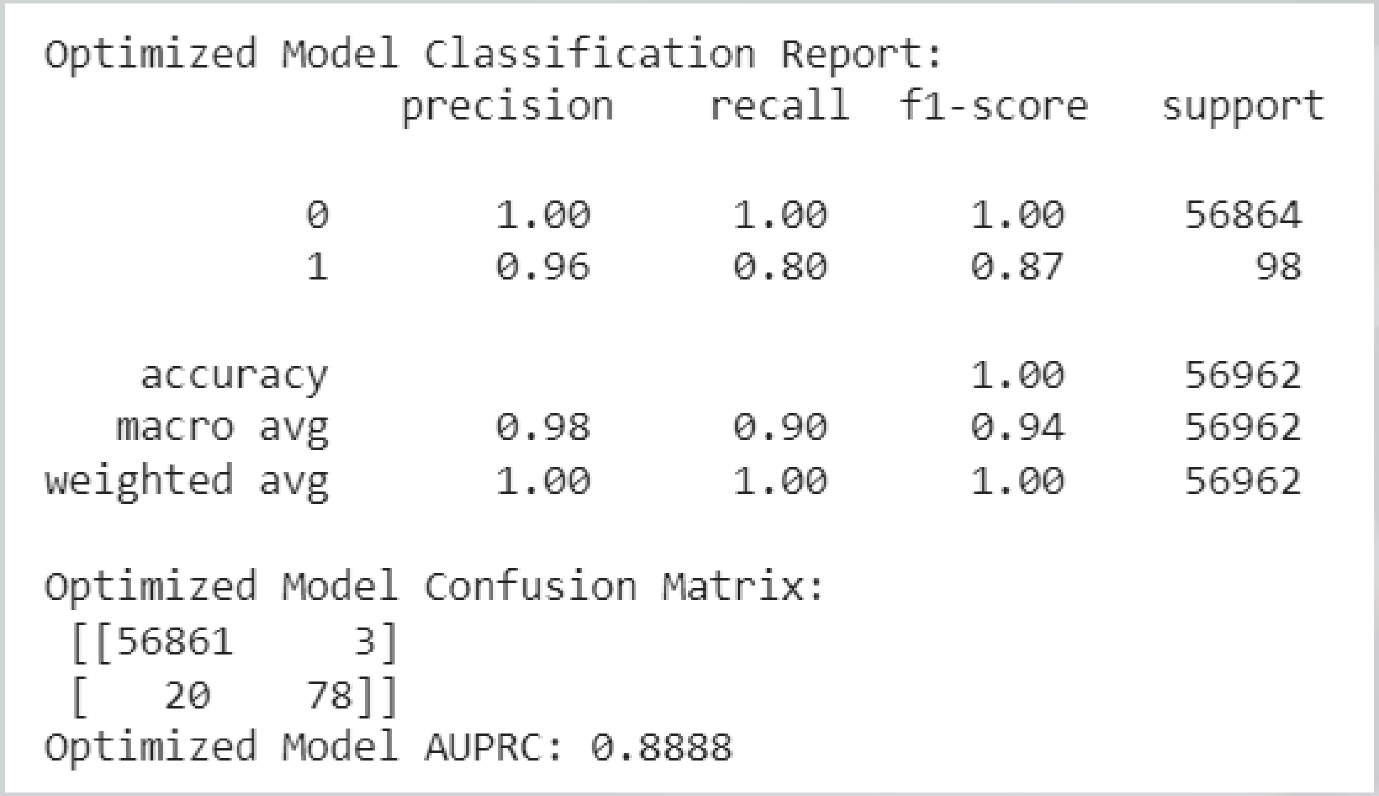


Hyperparameter Optimization with Random Grid Search:

- Random Grid Search method randomly samples from a distribution of hyperparameters, enabling us to identify the most effective model configurations without exhaustive searching.
- We focused on tuning key hyperparameters that influence model performance, including:
    - n_estimators: Number of gradient boosted trees equivalent to the number of boosting rounds.
    - max_depth: Maximum tree depth for base learners.
    - learning_rate: Boosting learning rate (also known as eta).
    - subsample: Subsample ratio of the training instances.
    - colsample_bytree: Subsample ratio of columns when constructing each tree.

By fine-tuning these parameters, we build a highly accurate and efficient model capable of detecting fraudulent transactions with greater precision.

Optimizing with GPU-Enabled XGBoost Classifier



```
Optimized Model Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56864
           1       0.96      0.80      0.87        98

    accuracy                           1.00     56962
   macro avg       0.98      0.90      0.94     56962
weighted avg       1.00      1.00      1.00     56962

Optimized Model Confusion Matrix:
 [[56861     3]
 [   20    78]]
Optimized Model AUPRC: 0.8888
```

Hyperparameter Optimization with Random Grid Search and Out-of-Bag Error

Out-of-Bag Error Estimation:

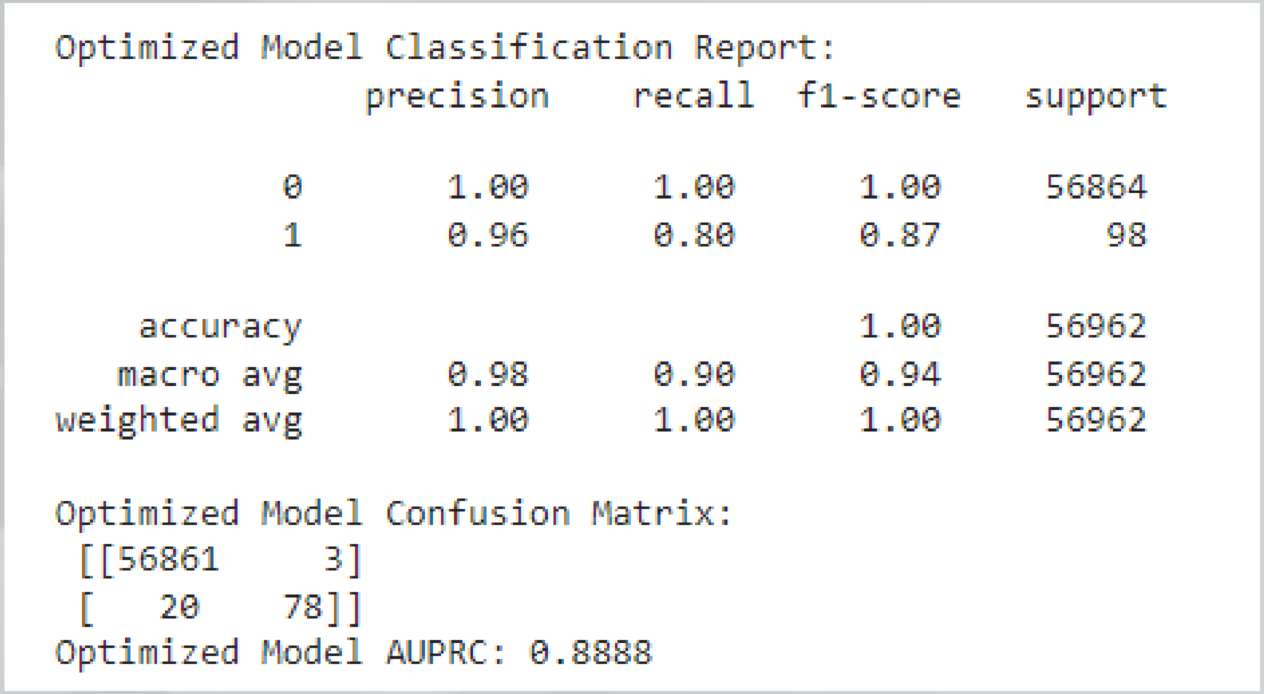
- Utilized the Out-of-Bag (OOB) error estimate in our Random Forest model. Each tree is trained on a different bootstrap sample, and the OOB error is calculated using only the cases not inclu ded in the bootstrap sample for each tree, serving as an internal cross-validation mechanism.

Advantages of OOB Error:

- Internal Validation: OOB error provides an unbiased estimate of the model’s error rate with-out the need for a separate validation set.
- Efficiency: It makes efficient use of data, because no data is set aside for validation; all data is used for both training and validating the model.
- Performance Indicator: It offers a quick glimpse into the model's potential performance on uns een data, helping in assessing the model's stability and robustness.

By evaluating the OOB error, we gained insights into how our model performs in a real-world scena rio without further splitting the dataset or performing external cross-validation. In this version, we used the same random search as before but with OOB error.

Hyperparameter Optimization with Random Grid Search and Out-of-Bag Error



```
Optimized Model Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56864
           1       0.96      0.80      0.87        98

    accuracy                           1.00     56962
   macro avg       0.98      0.90      0.94     56962
weighted avg       1.00      1.00      1.00     56962

Optimized Model Confusion Matrix:
 [[56861     3]
 [   20    78]]
Optimized Model AUPRC: 0.8888
```

Model Interpretability and Visualization: Partial Dependence Plots (PDP)

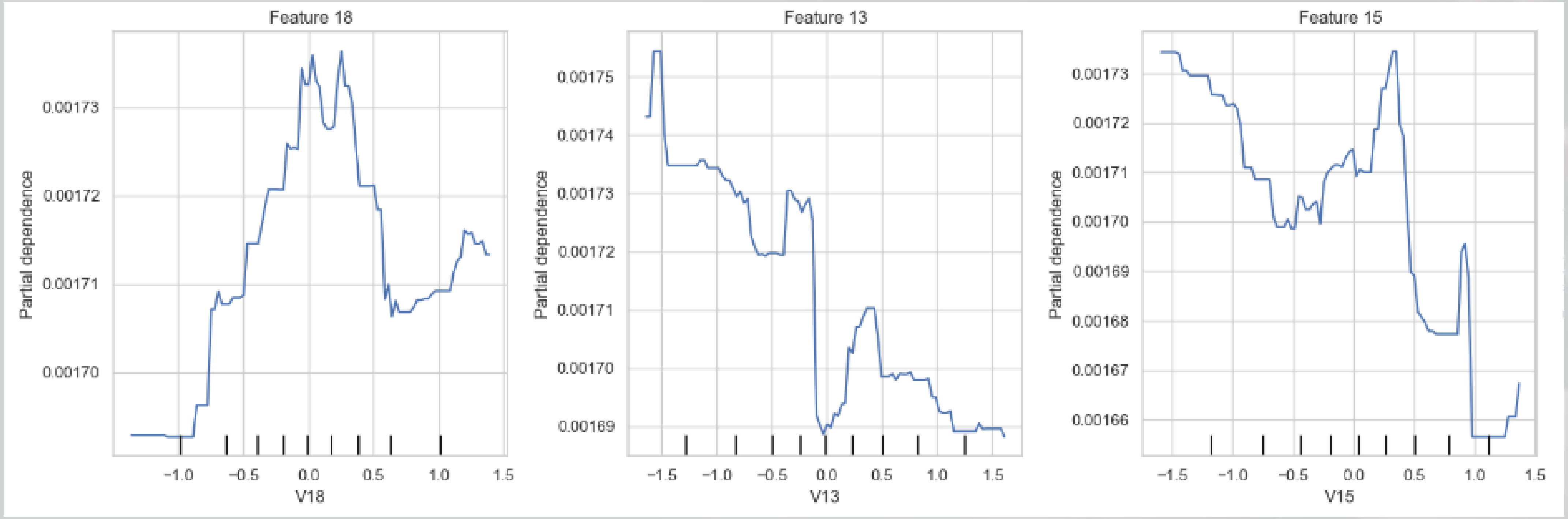
Partial Dependence Plots (PDP) provide a graphical depiction of the marginal effect of a variable on the predicted outcome of a model, holding all other variables constant. By applying PDPs to the PCA-transformed features in our model, we can explore how changes in these features affect the probability of detecting fraud.

Benefits of PDPs:

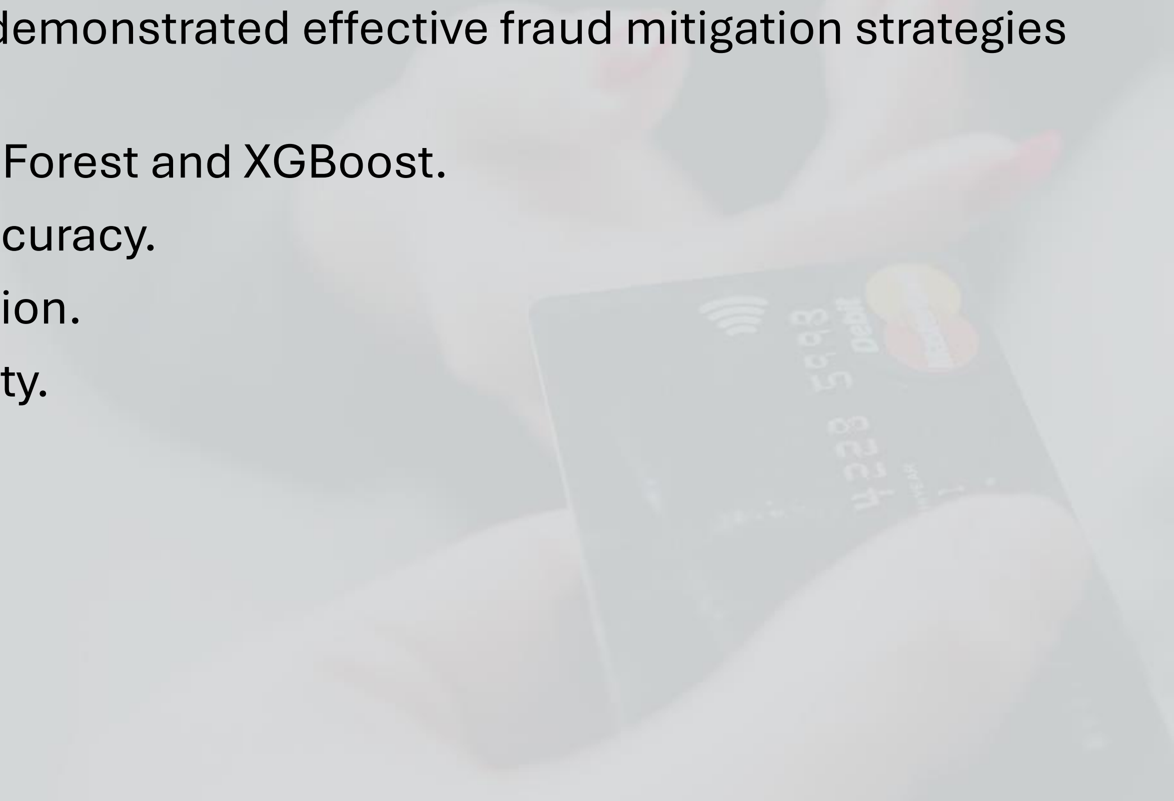
- Insightful: Shows the influence of single or pairs of features on the predicted outcome.
- Non-linear Patterns: Helps in capturing non-linear dependencies between features and the target.
- Broad Applicability: Useful for any model as they depend only on the model outputs and feature values.

PDPs are instrumental in demystifying the black-box nature of complex models, especially those involving PCA features.

Model Interpretability and Visualization: Partial Dependence Plots (PDP)



Conclusion and Future Trends



Conclusion & Key Achievements:

This project underscores the vital role of fraud detection in today's digital economy. By using the machine lea
rning algorithms like Random Forest and XGBoost, we have demonstrated effective fraud mitigation strategies
, bolstering consumer and financial institution security.

- Developed robust fraud detection models using Random Forest and XGBoost.
- Conducted hyperparameter optimization for improved accuracy.
- Leveraged Out-of-Bag error estimation for internal validation.
- Utilized Partial Dependence Plots for model interpretability.

Future Scope: To further enhance fraud detection:

- Integrate real-time data for continuous monitoring.
- Explore deep learning for complex pattern recognition.
- Collaborate with industry for practical deployment.
- Investigate novel feature engineering methods.
- Address ethical and regulatory considerations.

THANK YOU