King Saud University
College of Computer and Information Sciences
Information Technology Department

IT 461 Course Project
Semester 1, 1446

IT461
Practical Machine Learning
Project Proposal

Credit Card Fraud Detection

Prepared by :
Raseel Alrawdhan, 443200592
Raghad Alharkan, 443200477
Marya Asaad, 443200794
Alia Alrassan, 443202519
Sara alzayed, 443203107

Supervised by:
Dr. Yousra Saud Almathami

Table of Contents

Table of Figures

Table of Tables

1- Introduction

The problem being addressed is credit card fraud detection, which is of significant concern to financial institutions and customers due to the rapid increase in online transactions. Fraudulent transactions result in billions of dollars in losses annually, highlighting the importance of addressing this issue. The task of identifying and preventing fraud in real-time is crucial for protecting sensitive information and maintaining the financial integrity of the system. As transaction volumes continue to grow, traditional manual fraud detection methods are no longer sufficient, making the need for automated, real-time solutions even more critical.

Given the complexity and evolving nature of fraud, advanced machine learning techniques can significantly enhance the detection rate, reducing false positives while ensuring a secure transaction environment. This project is driven by the need to implement a robust system that can adapt to new fraud patterns effectively.

The task is to develop a machine learning model that can classify credit card transactions as "fraudulent" (1) or "legitimate" (0).

- **Input**: A dataset containing transaction features like amount, time, and anonymized variables.
- **Output**: A binary output indicating whether a transaction is fraudulent.

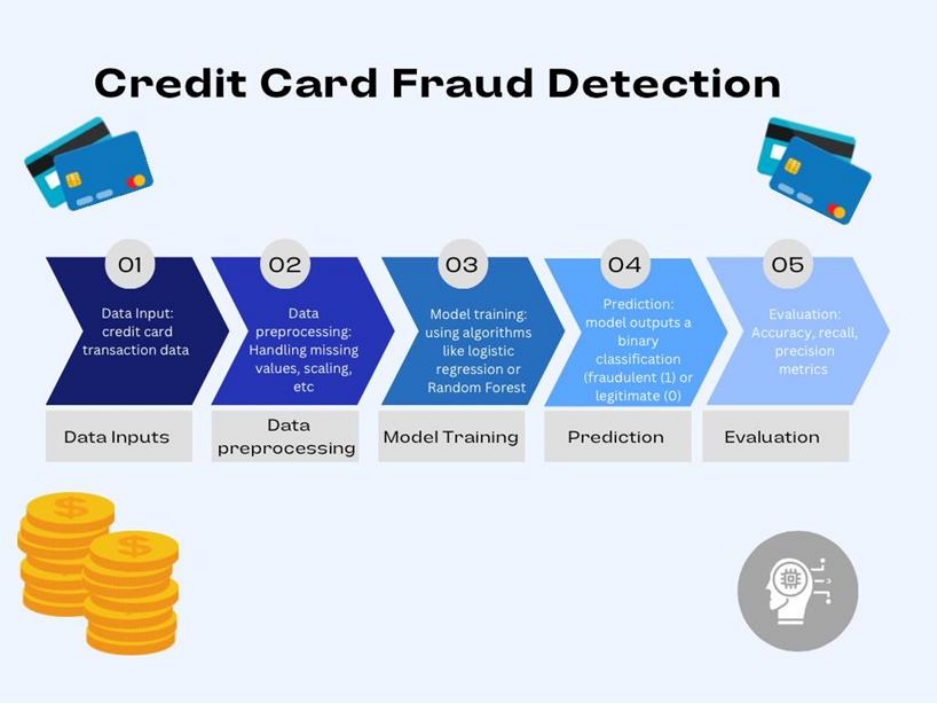

Figure 1- Input output illustration figure

2- Related Works

[1] The paper titled "Supervised Machine Learning Algorithm for Detecting and Predicting Fraud in Credit Card Transactions" is one of the previous attempts of solving similar problem, addresses the growing problem of credit card fraud in online transactions. The authors test and compare the performance of three algorithms Logistic Regression, Decision Tree, and Random Forest, to find the most suitable model for classifying fraudulent and legitimate transactions. Their goal was to classify transactions as fraudulent or not using a dataset of simulated credit card transactions. The authors found that the Random Forest model performed best, achieving 96% accuracy and an AUC (Area Under the Curve) value of 98.9%, making it the most suitable for detecting fraudulent transactions. Logistic Regression and Decision Tree models also performed well but were outperformed by Random Forest. Key findings included that credit card holders above 60 years old were more frequently targeted, and most fraudulent activities occurred between 10:00 PM and 5:00 AM, when banks are not operational, and cardholders may be less vigilant.

[2] another paper titled "Credit Card Fraud Detection using Machine Learning Algorithms." which is another attempt to solve a similar problem, they focused on tackling the problem of credit card fraud with machine learning. The goal was to build a model that can accurately choose between fraudulent and legitimate transactions using features like transaction time and amount. They tested several algorithms to know which one is better such as Logistic Regression, Naive Bayes, Decision Tree, and Artificial Neural Networks (ANN). ANN performed the best, reaching an accuracy of 98.69% with few false positives. They faced some challenges such as data imbalance as fraudulent transactions were much fewer than legitimate ones, and to solve this they used oversampling techniques to balance the data. The study concludes that ANN is very effective for fraud detection.

[3] This project "Credit Card Fraud Detection Using Machine Learning" aimed to identify fraudulent credit card transactions using various machine learning models due to the growing incidence of fraud worldwide. The research employed three machine learning techniques—K-Nearest Neighbor (KNN), Support Vector Machine (SVM), and Logistic Regression—on a dataset sourced from Kaggle, containing credit card transactions from Europe in 2013. The objective was to compare these models to determine which one performed best in detecting fraudulent activity. Among the models, the Support Vector Machine achieved the highest accuracy at 99.94%, followed by Logistic Regression at 99.92% and KNN at 99.88%. In contrast, the Naïve Bayes model performed the least effectively with an accuracy of 97.76%. The conclusion drawn from the study was that SVM is the most suitable model for detecting fraudulent transactions due to its superior accuracy. The researcher also recommended exploring additional datasets and alternative algorithms for further improvements.

3- Data

The dataset used for the project contains a large set of simulated credit card transactions, including both legitimate and fraudulent ones. It covers transactions made by 1,000 customers at 800 different merchants. Each transaction is described by 22 features, capturing various transaction details such as the amount, merchant category, location of the transaction, and customer information. The target variable, `is_fraud`, indicates whether the transaction is fraudulent (1) or legitimate (0). The dataset mirrors a real-world fraud detection problem, where fraudulent transactions make up a very small proportion of the data. This class imbalance reflects the typical challenge faced in fraud detection, where the majority of transactions are legitimate, but identifying the small fraction of fraudulent ones is critical.

3.1- Source:

The dataset was obtained from the "Credit Card Transactions Fraud Detection Dataset" on Kaggle, a popular platform for data science competitions and datasets, generated using the Sparkov synthetic data generator.[4]

3.2- Why This Dataset Was Chosen:

The dataset contains detailed transaction features like transaction amount, time, location, and customer details, making it suitable for building a machine learning model to identify fraudulent transactions. Furthermore, the dataset's size and variety of features make it an excellent candidate for testing both traditional and advanced machine learning algorithms, ensuring that the solution is scalable and adaptable to real-world fraud detection scenarios.

Summary Statistics:

Table 1- Summary Statistics

Attribute	Description
Number of Examples	1,296,675
Number of Features	22 (including both transaction and customer details)
Number of Classes	2 (Legitimate: 0, Fraudulent: 1)

Data Characteristics:

- Imbalance: Fraudulent transactions constitute only about 0.17% of the dataset, highlighting the class imbalance challenge.
- Diversity of Features: The dataset includes transaction-specific features such as the merchant's category and location, and customer-specific features like job, city, and gender. This diversity is crucial for detecting complex fraud patterns.

Dataset sample:

trans_date_trans_time	cc_num	merchant	category	amt	first	last	gender	street	city	state	zip	lat	long	city_pop	job	dob	trans_num	unix_time	merch_lat	merch_long	is_fraud
2019-01-01 00:00:18	2703186189652095	fraud_Rippin, Kub and Mann	misc_net	4.97	Jennifer	Banks	F	561 Perry Cove	Moravian Falls	NC	28654	36.0788	-81.1781	3495	Psychologist, counselling	1988-03-09	0b242abb623afc578575680df30655b9	1325376018	36.011293	-82.048315	0
2019-01-01 00:00:44	6304233337322	fraud_Heller, Gutmann and Zieme	grocery_pos	107.23	Stephanie	Gill	F	43039 Riley Greens Suite 393	Orient	WA	99160	48.8878	-118.2105	149	Special educational needs teacher	1978-06-21	1f76529f8574734946361c461b024d99	1325376044	49.159047000000000	-118.186462	0
2019-01-01 00:00:51	38859492057661	fraud_Lind-Buckridge	entertainment	220.11	Edward	Sanchez	M	594 White Dale Suite 530	Malad City	ID	83252	42.1808	-112.262	4154	Nature conservation officer	1962-01-19	a1a22d70485983eac12b5b88dad1cf95	1325376051	43.150704	-112.154481	0
2019-01-01 00:01:16	3534093764340240	fraud_Kutch, Hermiston and Farrell	gas_transport	45.0	Jeremy	White	M	9443 Cynthia Court Apt. 038	Boulder	MT	59632	46.2306	-112.1138	1939	Patent attorney	1967-01-12	6b849c168bdad6f867558c3793159a81	1325376076	47.034331	-112.561071	0

Figure 2- Data sample

3.3- Preprocess Steps:

- Handling Missing Data: Null values were removed using data.dropna()
- Feature Encoding: Categorical variables such as merchant category, state, and gender were encoded into numerical values using LabelEncoder
- Dropping redundant columns (e.g., "Unnamed: 0").
- Detecting and removing outliers using the IQR method.
- Addressing class imbalance by undersampling the majority class (not_fraud).

Some graphs that represent the data:

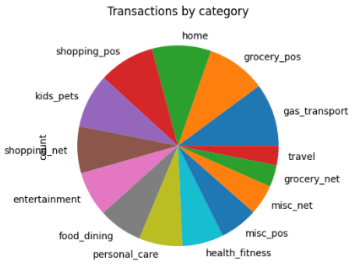

Figure 4 - Transaction by category pie plot

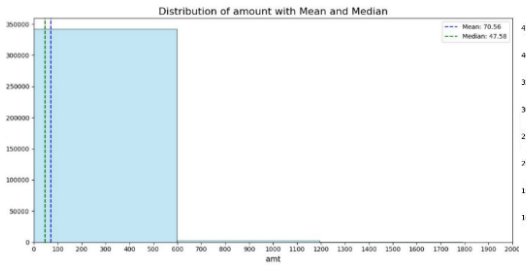

Figure 3- Distribution of transactions amount

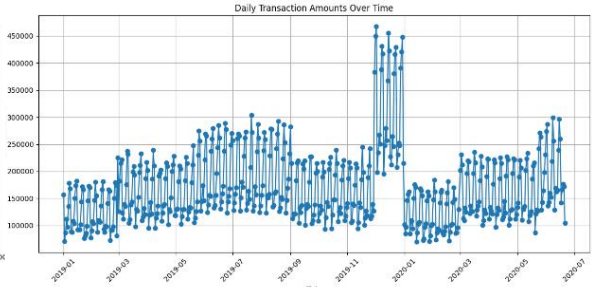

Figure 5- Transaction amount over time

The daily transaction trends (Figure 3) show seasonal spikes, particularly at the start of 2020. Category analysis (Figure 5) highlights significant spending on home, groceries, and transport. The distribution of transaction amounts (Figure 4) reveals most transactions are small, with mean and median amounts close together, indicating consistent spending behavior.

4- Methods

This was conducted in two phases to systematically address the problem of fraud detection, progressing from baseline models to advanced techniques.

4.1- Phase 1: Baseline Models

In the first phase, we employed Random Forest, Logistic Regression and Support Vector Machine (SVM) as initial models to establish a performance baseline and compare their effectiveness.

- Random Forest was used for its inherent ability to handle high-dimensional data, assess feature importance, and deal with class imbalance. The default configuration of Random Forest provided an initial understanding of its potential.
- Logistic Regression was chosen for its simplicity, interpretability, and ability to model linear relationships. It served as a straightforward benchmark for performance evaluation.
- SVM (Support Vector Machine) was included for its capability to handle non-linear relationships and create robust decision boundaries. SVM with hyperparameter tuning was optimized using RandomizedSearchCV, exploring parameters such as:
    - Regularization (C)
    - Kernel coefficient (gamma)
    - Kernel types (rbf, linear, sigmoid).

  Additionally, StandardScaler was applied to preprocess the data for better performance.

Among these models, Random Forest emerged as the best-performing model in terms of accuracy, recall, and handling class imbalance, demonstrating its suitability for the problem at hand. This result motivated its further exploration and refinement in Phase 2.

4.2- Phase 2: Advanced Models

Building on the insights from Phase 1, the second phase focused on refining and expanding the modeling techniques:

- Random Forest with Hyperparameter Tuning: To optimize the Random Forest model, hyperparameters such as the number of estimators, maximum depth, minimum samples split, and class weighting were fine-tuned using RandomizedSearchCV. This refinement aimed to enhance the model's precision and recall while addressing class imbalance.
- Stacking Classifier: To combine the strengths of multiple algorithms, a stacking ensemble was created with Random Forest, Gradient Boosting, XGBoost, and LightGBM as base learners, and Logistic Regression as the meta-learner. This ensemble approach leveraged the complementary strengths of different models and significantly improved overall performance. Cross-validation was used to train the meta-learner on unbiased predictions from the base models.

The two-phase approach was designed to systematically tackle the challenges of fraud detection, such as class imbalance, non-linear relationships, and the need for high generalization. Phase 1 provided a baseline understanding of model performance, with Random Forest standing out as the most effective model. Phase 2 capitalized on these insights by refining Random Forest through hyperparameter tuning and combining it with other models in an ensemble for improved robustness.

4.3- Feature Engineering:

- Scaling: Features were scaled using StandardScaler to standardize the data. This step was particularly important for models sensitive to feature magnitudes, such as SVM and ensemble techniques.
- Feature Creation: New features, such as transaction day, transaction hour, and normalized age, were derived to provide additional predictive power. These features helped capture patterns related to temporal aspects and user demographics.

4.4- Dimensionality Reduction:

To simplify the predictive model and improve computational efficiency, a feature selection technique based on variable importance derived from a Random Forest classifier was employed. Random Forest inherently provides a ranking of feature importance, which we used to identify the most significant input features for predicting the target variable, "is_fraud." By focusing on these critical features, we reduced noise and ensured the model concentrated on the variables with the most predictive power.

While we initially experimented with backward and forward feature selection methods, the limited size of the dataset and computational constraints made these approaches impractical. Consequently, we opted for the Random Forest-based feature importance approach, which is both efficient and robust for datasets of this size.

5- Experiment

5.1- Feature Engineering

- Handling Missing Data: Null values were removed using data.dropna() to ensure complete records for processing.
- Feature Encoding: Categorical variables such as merchant category, state, and gender were encoded into numerical values using LabelEncoder.
- Datetime Conversion: The transaction timestamp was converted to datetime format and split into new features, including trans_day, trans_month, trans_year, trans_hour, and trans_minute, for analysis.
- Outlier Removal: Numeric columns (cc_num, amt, city_pop, lat, long) were cleaned by detecting and removing outliers using the Interquartile Range (IQR) method.
- Data Balancing: Class imbalance in the target class (is_fraud) was addressed by undersampling the majority class to match the number of fraudulent cases.
- Variable Importance Ranking: A Random Forest classifier ranked features based on their contribution to model performance, highlighting the most significant predictors.
- Drop Unused Columns: Useless columns (e.g., first, last, trans_num) were removed to simplify the dataset.
- Standardization: Numeric features were standardized with StandardScaler to have a mean of 0 and unit variance
- Age Calculation and Normalization: The dob column was used to calculate age in years, which was normalized to a range of 0–1 using Min-Max scaling.

5.2- Dataset Splitting

The dataset was split into two parts: 80% was allocated to the training set, which was used for model learning, and 20% was reserved as the testing set to evaluate the model's performance on unseen data.

5.3- Machine Learning Models

- Random Forest

We employed the Random Forest algorithm as one of our primary models for fraud detection. Initially, the model used default parameters, including 100 decision trees and Gini impurity as the splitting criterion. While this approach delivered strong performance by balancing bias and variance, we enhanced its predictive capabilities by incorporating hyperparameter tuning.

In the updated model, we utilized a pipeline with StandardScaler for feature scaling and a Random Forest classifier. Hyperparameters were optimized using Random Search with 5-fold cross-validation and 50 iterations. We tested the number of decision trees (500 to 800), maximum features (auto and sqrt), tree depth (None, 50, 70, 100), and the minimum samples for splits and leaves (1 and 2). Class balancing was achieved using class_weight='balanced' to address dataset imbalance.

The model was evaluated using accuracy, precision, recall, and F1-score, which measured its effectiveness in detecting fraud. Accuracy reflected overall correctness, precision assessed correctly identified frauds, recall measured sensitivity, and the F1-score balanced precision and recall. A confusion matrix further provided insights into prediction performance.

- Logistic Regression

Logistic Regression was implemented as a baseline model due to its simplicity, interpretability, and effectiveness in binary classification tasks. This linear classifier predicts probabilities to separate classes based on a decision boundary. As part of feature engineering, we standardized features using StandardScaler to ensure consistent scaling across all variables. The model's stability and generalization were evaluated through 5-fold cross-validation, providing insights into its performance across multiple data splits.

- Support Vector Machine (SVM)

We selected Support Vector Machine (SVM) for its ability to handle both linear and non-linear classification tasks effectively. To ensure consistent feature scaling, we implemented a pipeline combining StandardScaler for normalization with the SVM classifier. Hyperparameter tuning was conducted using Random Search, optimizing key parameters such as C (0.001 to 1000), gamma (0.0001 to 100), and kernel types (RBF, Linear, Sigmoid). The optimal configuration, identified based on cross-validation accuracy, was used to train the model, which demonstrated strong classification capabilities in identifying fraudulent transactions.

The model's performance was evaluated on the test set using accuracy, precision, recall, and F1-score to assess its effectiveness in classifying fraudulent and non-fraudulent transactions. Additionally, ROC curves compared SVM with Random Forest, with SVM achieving a high Area Under the Curve (AUC) score, reflecting its ability to distinguish between classes effectively.

- Stacking Classifier Using Multiple Base Models

To enhance predictive performance, we implemented a stacking classifier that combines multiple base models: Random Forest, Gradient Boosting, XGBoost, and LightGBM. Logistic Regression was used as the meta-model to aggregate predictions, leveraging the diverse strengths of the individual algorithms for a more robust fraud detection solution.

Hyperparameter tuning was performed for each base model. For example, the number of estimators was varied between 100 and 500, and the maximum tree depth for Random Forest was explored across values such as None, 10, and 20. Optimization was conducted using GridSearchCV within a pipeline that included StandardScaler for feature scaling and the stacking classifier.

A 5-fold cross-validation strategy was employed to ensure optimal parameter configuration and generalization to unseen data. The optimized stacking model was then evaluated on a test set, achieving strong performance. Metrics such as accuracy, the confusion matrix, and the classification report confirmed its ability to outperform standalone classifiers, demonstrating its effectiveness in detecting fraudulent transactions.

5.4- libraries

The implementation utilizes several libraries, including pandas, numpy, matplotlib, seaborn, sklearn, geopandas, scipy, xgboost, and lightgbm, for data preprocessing, visualization, and model training. All models were trained and evaluated using CPU resources.

6- Results and Discussion

6.1- Model Performance Overview

The performance of multiple machine learning models, including Logistic Regression, Support Vector Machines (SVM), Random Forest, and a Stacking Classifier, was evaluated on the dataset. The detailed metrics for each model, such as accuracy, precision, recall, F1-score, and Area Under the Curve (AUC), are summarized in Table 2. Additionally, confusion matrices and Receiver Operating Characteristic (ROC) curves were used to better understand and compare model performance.

Table 2-Performance Matrics of the models

Model	Accuracy	Precision	Recall	F1-Score	AUC
Logistic Regression	0.6929	-	-	-	-
SVM (RBF Kernel)	0.8539	0.8539	0.8539	0.8539	0.90
Random Forest (Phase 1)	0.9469	0.9186	0.9713	0.9442	0.98
Random Forest (Phase 2)*	0.9772	0.9775	0.9772	0.9772	0.99
Stacking Classifier	0.9753	0.9750	0.9755	0.9753	0.99

**Phase 2 refers to models optimized with hyperparameter tuning.*

6.2- Performance Analysis

- Logistic Regression

The Logistic Regression model showed moderate performance, with an accuracy of approximately 69.3%. While this baseline model is computationally efficient, it struggled to capture the complexity of the data.

- Support Vector Machine (SVM)

The SVM with an RBF kernel demonstrated improved performance, achieving an accuracy of 85.4%. The model achieved balanced precision, recall, and F1-score values, showcasing its ability to generalize well to unseen data. The ROC curve (Figure 6) highlights the AUC of 0.90, confirming its robustness in separating the classes. The confusion matrix shows that there are 218 true negatives, 40 false positives, 37 false negatives, and 232 true positives.

- Random Forest

The Random Forest model initially achieved an accuracy of 94.7% with an AUC of 0.98 (Phase 1), which improved to 97.7% after hyperparameter optimization (Phase 2). The confusion matrices revealed a

significant reduction in both false positives and false negatives after tuning, with the F1-score also improving to 0.977. This indicates excellent performance in generalizing to new data.

Confusion Matrixes of RF

[262 21]	[260 9]
[7 237]	[3 255]
Random forest	Random Forest Classifier with hyperparameter

- Stacking Classifier

The Stacking Classifier, which combines predictions from Random Forest, Gradient Boosting, XGBoost, and LightGBM, achieved comparable results to the optimized Random Forest, with an accuracy of 97.5% and an AUC of 0.99. This ensemble approach effectively leveraged the strengths of multiple models, offering robust and reliable predictions. The confusion matrix shows that there are 262 true negatives, 7 false positives, 6 false negatives, and 252 true positives.

6.3- Visual Comparisons

- ROC Curves

(Figure 7) illustrates the ROC curves for the Random Forest (Phase 1) and SVM models, demonstrating the superior AUC of the Random Forest (0.98) compared to SVM (0.90). Similarly, (Figure 6) compares the optimized Random Forest (Phase 2) with the Stacking Classifier, both achieving AUC values of 0.99. These visualizations highlight the ability of ensemble methods to achieve near-perfect classification performance.

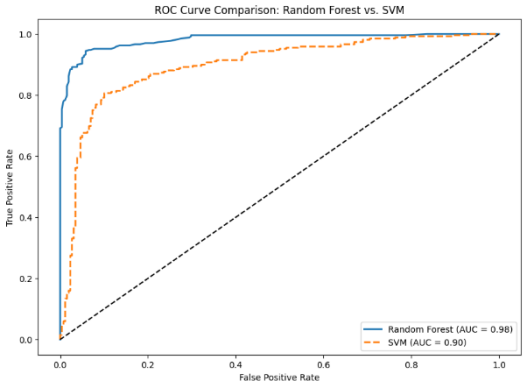

Figure 7- ROC curves for the Random Forest (Phase 1) and SVM models

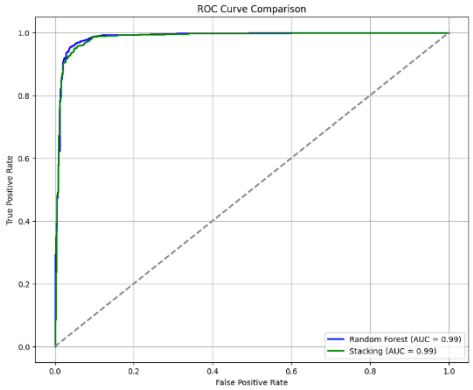

Figure 6- ROC curves for the optimized Random Forest (Phase 2) with the Stacking Classifier

- Learning Curve

In Figure 7, the learning curve for the Random Forest model with hyperparameter tuning shows that the model has not yet reached optimal performance. Both training and validation losses continue to decrease, indicating that the model can still benefit from additional data, improving generalization and accuracy.

By adding more data, we can aim to achieve accuracy and performance beyond 97.7%.

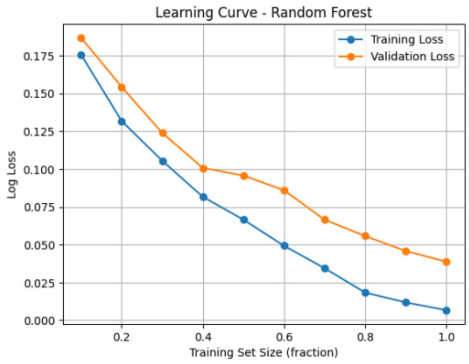

Figure 8- Random forest with HP Learning Curve

- Confusion Matrices

The confusion matrices (mentioned above) further confirmed the effectiveness of the Random Forest and Stacking models, showing high true positive and true negative rates. For instance, in Phase 2, the Random Forest model misclassified only 12 samples (9 false positives and 3 false negatives), significantly fewer than the Phase 1 results.

6.4- Insights and Interpretation

- Hyperparameter Tuning Impact

The marked improvement in Random Forest performance after hyperparameter tuning demonstrates the critical role of model optimization in achieving higher accuracy and generalization.

- Ensemble Learning

The Stacking Classifier consistently matched or exceeded the performance of individual models. This highlights the benefits of leveraging diverse learning algorithms to create a more generalized model.

- Data Complexity

The poor performance of Logistic Regression suggests that the dataset is not linearly separable, requiring more complex models like Random Forest and SVM for effective classification.

- Model Robustness

The high AUC values and consistent performance across metrics indicate that Random Forest and Stacking Classifiers are well-suited for handling unseen data, offering robust predictive power.

7- Conclusion

This project aimed to develop a reliable fraud detection system using machine learning techniques. By exploring various models and feature engineering methods, we successfully designed a robust predictive framework that addresses the challenges of detecting fraudulent transactions.

Below are the key findings, limitations, and proposed future directions.

Key Findings:

Ensemble methods such as Random Forest and Stacking Classifiers demonstrated superior performance, effectively balancing precision and recall, which is critical in fraud detection. These models handled the class imbalance well and leveraged feature importance to make accurate predictions.

Feature selection based on Random Forest's variable importance was instrumental in identifying and retaining the most influential features for prediction. This approach improved computational efficiency and reduced the risk of overfitting while maintaining model accuracy. Features such as transaction time and user demographics proved particularly impactful.

The methodology implemented—including feature scaling, efficient feature selection, and hyperparameter tuning—ensures that the models are scalable to larger datasets. Furthermore, the interpretability provided by the feature importance analysis enhances trust in the model's predictions.

Challenges and Limitations:

Fraud detection inherently deals with imbalanced datasets, where fraudulent cases are rare compared to legitimate transactions. While techniques like class weighting were employed, addressing this imbalance remains an ongoing challenge. Computational limitations impacted our ability to apply more resource-intensive feature selection methods, such as backward and forward selection. Instead, we relied on Random Forest-based feature importance, which, although effective, might overlook potential feature interactions.

Proposed Improvements and Future Directions:

The proposed improvements and future directions suggest incorporating larger, more diverse datasets to enhance model generalizability. Advanced feature selection techniques, such as recursive feature elimination (RFE) or L1-regularization, could uncover more relevant features while ensuring computational efficiency. Optimizing models for real-time fraud detection would enable deployment in live environments, providing immediate fraud prevention. Additionally, integrating advanced resampling techniques like SMOTE or developing cost-sensitive learning models would help address class imbalance and reduce the financial impact of false positives and negatives. In conclusion, this project demonstrates the potential of machine learning techniques in building effective fraud detection systems. By addressing the identified challenges and implementing the proposed improvements, the system can evolve into a more scalable, interpretable, and efficient solution for combating fraud.

8- Contributions

Tasks	Student Name
introduction	**Alia Alrassan**
Related works	**Raseel Alrawdhan** **Raghad Alharkan** **Sara alzayed**
Data	**Marya Asaad**
Methods	**Raghad Alharkan**
Experiment	**Marya Asaad** **Alia Alrassan**
Results and Discussions	**Sara alzayed** **Raseel Alrawdhan**
Conclusion	**Raghad Alharkan**
Notebook coding	**Raseel Alrawdhan**
Contributions & References	**Raseel Alrawdhan**
Reviewing the document	**Raseel Alrawdhan** **Alia Alrassan** **Marya Asaad**

9- References

[1] J. K. Afriyie et al., “A supervised machine learning algorithm for detecting and predicting fraud in credit card transactions,” Decision Analytics Journal, vol. 6, p. 100163, Mar. 2023, doi: 10.1016/J.DAJOUR.2023.100163.

[2] V. K. Kumar S, V. V Kumar G, and V. A. Shankar, “Credit Card Fraud Detection using Machine Learning Algorithms,” vol. 9, no. 7, Jul. 2020, Accessed: Sep. 19, 2024. [Online]. Available: www.ijert.org

[3] M. Alemad, “Credit Card Fraud Detection Using Machine Learning,” Jul. 2022, Accessed: Sep. 19, 2024. [Online]. Available: https://repository.rit.edu/theses

[4] “Credit Card Transactions Fraud Detection Dataset.” Accessed: Sep. 19, 2024. [Online]. Available: https://www.kaggle.com/datasets/kartik2112/fraud-detection/data