Credit Card Fraud Detection

Research paper for
CS F415 Data Mining Project

BITS Pilani Hyderabad Campus

By

Dhairya Luthra	**2022A7PS1377H**
Animish Agrahari	**2022A7PS1367H**
B. Vaishnavi	**2022A7PS1357H**

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE,
PILANI,HYDERABAD CAMPUS

April,2024

TABLE OF CONTENTS

Team Members and Emails

DHAIRYA LUTHRA	f2022177@hyderabad.bits-pilani.ac.in
ANIMISH AGRAHARI	f20221367@hyderabad.bits-pilani.ac.in
B. VAISHNAVI	f20221357@hyderabad.bits-pilani.ac.in

1) Abstract

This paper explores various data mining techniques for fraud detection in credit card transactions, focusing on the challenge posed by highly imbalanced datasets where legitimate transactions vastly outnumber fraudulent ones. The detection of fraud is crucial in financial systems to prevent monetary losses and maintain trust among customers and businesses. This problem is particularly challenging due to the small number of fraudulent cases, which can lead to skewed model performance and inaccurate predictions.

In this study, several machine learning and anomaly detection methods were implemented and evaluated using a real-world credit card transaction dataset. Techniques such as logistic regression with undersampling, support vector machines (SVM), random forest, isolation forest, local outlier factor (LOF), and XGBoost were employed to identify fraudulent transactions accurately. The experiments focused on assessing each method's performance in terms of precision, recall, and overall accuracy.

The outcomes of the experimental results highlighted the effectiveness of ensemble techniques like random forest and XGBoost, as well as anomaly detection methods such as isolation forest, in accurately identifying fraudulent transactions despite class imbalance. These methods demonstrated strong performance in distinguishing between legitimate and fraudulent transactions, showcasing their potential for practical fraud detection applications.

Keywords: fraud detection, data mining, machine learning, imbalance dataset, credit card transactions

2) Related Works

2.1)) Design and Implementation of Different Machine Learning Algorithms for Credit Card Fraud Detection

CITATION:

A. Singh, A. Singh, A. Aggarwal and A. Chauhan, "Design and Implementation of Different Machine Learning Algorithms for Credit Card Fraud Detection," 2022 International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME), Maldives, Maldives, 2022, pp. 1-6, doi: 10.1109/ICECCME55909.2022.9988588.

The research paper explores the use of machine learning algorithms for credit card fraud detection, emphasizing the increasing need for effective solutions as online transactions and credit card frauds continue to rise. The study compares four machine learning algorithms **- logistic regression, decision tree, random forest, and Catboost** - based on their accuracies.

The dataset for credit card fraud detection was obtained from Kaggle, and the **Catboost algorithm was found to be the most effective with an accuracy of 99.87%.** The paper also discusses the importance of data resampling and preprocessing, as well as the significance of features such as amount spent, average amount spent by the user in the last 24 hours, and transaction hour in fraud detection.

Furthermore, the paper presents a detailed methodology, including data preprocessing, resampling techniques, and the implementation of the algorithms using Python. The authors highlight the challenges posed by highly unbalanced datasets in credit card transaction data and discuss the importance of resampling to address this issue. The conclusion emphasizes the development of a more accurate fraud detection system for credit card transactions and the superior performance of the Catboost algorithm compared to other methods. Overall, the research provides valuable insights into the application of machine learning in credit card fraud detection and underscores the significance of algorithm selection and data preprocessing in achieving high accuracy.

3) Approach/Methodology

3.1)Challenges in Credit Card Fraud Detection

The detection of credit card forgery poses a significant challenge for banking systems and financial markets. Various precise and reliable techniques have been developed in recent years to identify fraudulent credit card activities. However, several constraints, limitations, problems, and challenges still persist in this domain.

1. Fraudster Dynamic Behavior

- With increasing fraud ratios, detection tools and methods need to evolve to address the changing behaviors of fraudsters. As hackers continuously update and sophisticate their techniques, it becomes challenging for investigation teams to keep up with these dynamic behaviors.

2. Absence of Real Dataset

- One of the primary challenges faced by investigation teams and researchers is the reluctance of credit card companies, banks, and financial institutes to share customer's confidential data. The unavailability of real datasets hinders further investigation into credit card fraud detection.

3. Unstable Dataset

- Credit card fraud data is often convoluted, comprising a mix of legitimate and fraudulent transactions. This complexity makes it difficult to accurately identify instances of fraud, especially when customers fail to register complaints against fraudsters.

4. Size of Dataset

- The sheer volume of daily credit card transactions makes it challenging to collect and analyze data effectively. The vast amount of data presents concentration issues for research and investigation teams, leading to difficulties in comprehensive analysis.

5. Absence of Proper Technique and Parameters

- While there are numerous detection and prevention techniques available, no single technique is comprehensive. Furthermore, the instability and limitations of parameters used for investigation contribute to errors in credit card fraud detection and prevention.

3.2)Requirements

Obtaining an appropriate dataset is crucial for effectively training a model for credit card fraud detection. The dataset must encompass all relevant attributes that contribute to accurate fraud identification. Furthermore, it is imperative that the dataset contains minimal inconsistent values to ensure smooth model functioning and optimal performance.

After thorough evaluation, **we acquired a dataset from Kaggle.com** that aligns with the majority of our requirements for building a robust fraud detection model.

3.3)Dataset and its properties

The dataset under consideration was gathered and analyzed through a collaborative research effort between **Worldline and the Machine Learning Group of ULB (Université Libre de Bruxelles)**, focusing on big data mining and fraud detection.

It encompasses transactions conducted by European credit card holders in **September 2013.** Notably, the dataset comprises transactions that occurred over two days, with **492 identified as fraudulent out of a total of 284,807 transactions**.

This dataset is characterized by a high level of imbalance, where the occurrences of fraud, constituting the positive class, only represent **0.172% of all transactions**. The dataset solely consists of numerical input variables resulting from a PCA transformation. Regrettably, owing to confidentiality constraints, the original features and additional contextual details regarding the data cannot be disclosed.

The principal components obtained through PCA are represented by features V1 through V28, whereas the attributes 'Time' and 'Amount' have not undergone PCA transformation. The 'Time' attribute denotes the elapsed seconds between each transaction and the initial transaction recorded in the dataset, while the 'Amount' attribute indicates the transaction amount, potentially serving in example-dependent cost-sensitive learning. Furthermore, the 'Class' feature serves as the response variable, assuming a value of 1 in the event of fraud and 0 otherwise.

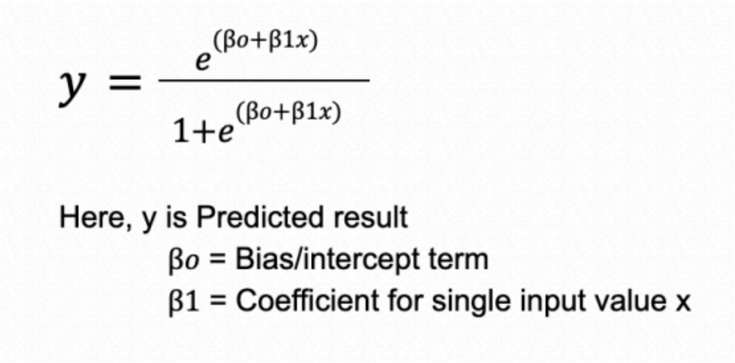
3.4) Techniques used

3.4.1)Classification Algorithms

3.4.1.1)Logistic Regression

Logistic regression is a statistical modeling technique employed for binary classification problems. It is a form of supervised learning that utilizes a linear model to estimate the probability of an instance belonging to one of two distinct classes. The target variable in logistic regression is dichotomous, meaning it can take on only two possible values, typically represented as 0 and 1, or true and false.

Logistic regression is particularly useful when the relationship between the predictor variables and the target variable is not necessarily linear, as it models the log odds of the target variable rather than the target variable itself. By estimating these log odds, logistic regression effectively captures the non-linear relationships present in the data.



$$y = \frac{e^{(\beta o+\beta 1x)}}{1+e^{(\beta o+\beta 1x)}}$$

Here, y is Predicted result

βo = Bias/intercept term

$\beta 1$ = Coefficient for single input value x

3.4.1.2)Support Vector Machines

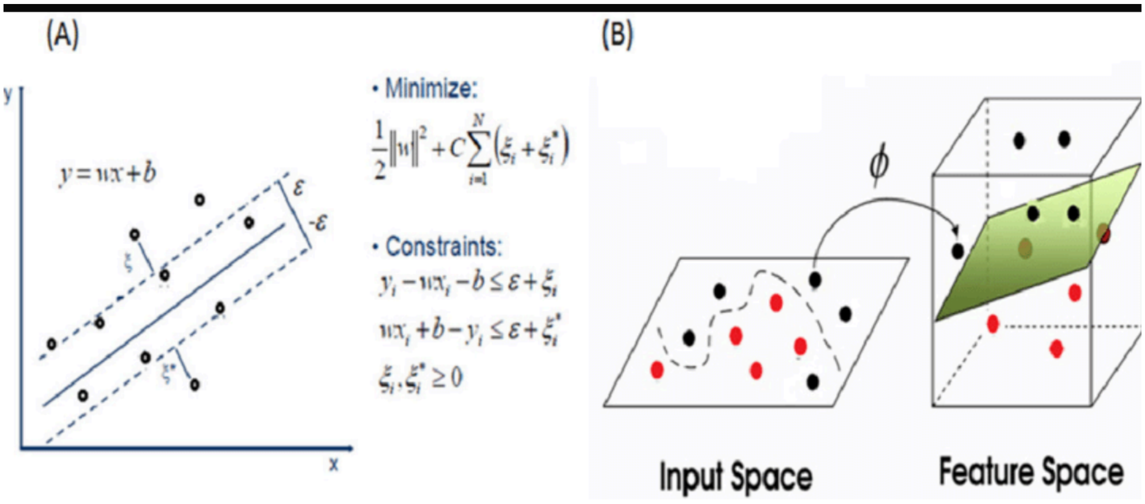
In the context of outlier detection, Support Vector Machines (SVMs) can be employed as a powerful technique for identifying anomalous data points or observations that deviate significantly from the normal patterns within a dataset.

The basic principle behind using SVMs for outlier detection is to treat the problem as a semi-supervised learning task, where the majority of the data is assumed to be normal, and the goal is to identify the few outlying instances. SVMs achieve this by constructing a decision boundary that separates the inliers (normal data points) from the outliers, maximizing the margin between these two groups.

The process involves training an SVM model on the available data, treating all instances as belonging to the normal class. By leveraging the maximum margin principle, the SVM aims to find the hyperplane that best separates the majority of the data from the origin or a predetermined outlier region in the feature space.

Once the SVM model is trained, new instances can be evaluated based on their position relative to the decision boundary. Data points that fall within the learned boundary are classified as inliers, while those that lie beyond the boundary are identified as potential outliers.

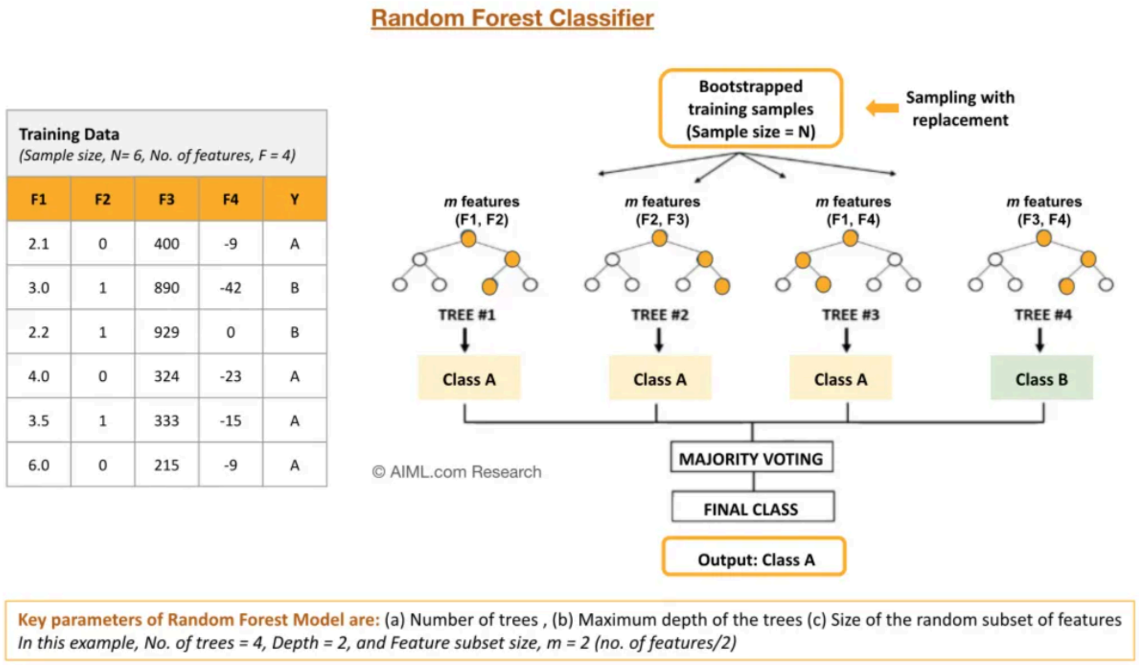
One of the advantages of using SVMs for outlier detection is their ability to handle high-dimensional and non-linear data. By employing appropriate kernel functions, SVMs can effectively map the data into a higher-dimensional space, where the separation between inliers and outliers becomes more apparent and linear


(A)
(B)
y
x
$y = wx + b$
ε
$-\varepsilon$
ξ
ξ^*
• Minimize:
$\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}(\xi_i + \xi_i^*)$
• Constraints:
$y_i - wx_i - b \leq \varepsilon + \xi_i$
$wx_i + b - y_i \leq \varepsilon + \xi_i^*$
$\xi_i, \xi_i^* \geq 0$
ϕ
Input Space
Feature Space

3.4.1.3)Random Forest Classifier

The Random Forest algorithm is a supervised machine learning technique that combines multiple decision trees, each trained on a random subset of the features and data instances. This ensemble approach helps mitigate the high variance and overfitting issues commonly encountered with individual decision trees, resulting in improved accuracy and robustness.

When applied to outlier detection, the Random Forest algorithm leverages its ability to capture complex patterns and interactions within the data. Each decision tree in the ensemble contributes to the identification of potential outliers by analyzing the data from different perspectives, considering various combinations of features and samples.



Training Data
(Sample size, N= 6, No. of features, F = 4)

F1	F2	F3	F4	Y
2.1	0	400	-9	A
3.0	1	890	-42	B
2.2	1	929	0	B
4.0	0	324	-23	A
3.5	1	333	-15	A
6.0	0	215	-9	A

Key parameters of Random Forest Model are: (a) Number of trees , (b) Maximum depth of the trees (c) Size of the random subset of features
In this example, No. of trees = 4, Depth = 2, and Feature subset size, m = 2 (no. of features/2)

3.4.1.4) XGBoost - Extreme Gradient Boosting

XGBoost is a powerful ensemble learning algorithm that combines multiple weak learners, typically decision trees, in a boosting framework. It has gained widespread popularity due to its exceptional performance, scalability, and ability to handle various data types and distributions.

XGBoost is an implementation of the gradient boosting algorithm, which is an ensemble learning technique that combines multiple weak learners, typically decision trees, to create a strong predictive model. The algorithm works in an iterative manner, where each successive tree in the ensemble is trained to correct the errors made by the previous trees. In the context of outlier detection, XGBoost treats the problem as a binary classification task, where the goal is to separate the normal instances from the outliers. The algorithm follows these general steps:

1. Initialization: XGBoost starts with a base model, which can be a single decision tree or a constant value, depending on the configuration.

2. Residual Calculation: For each instance in the training data, the algorithm calculates the residual, which is the difference between the actual label (normal or outlier) and the prediction made by the current model.

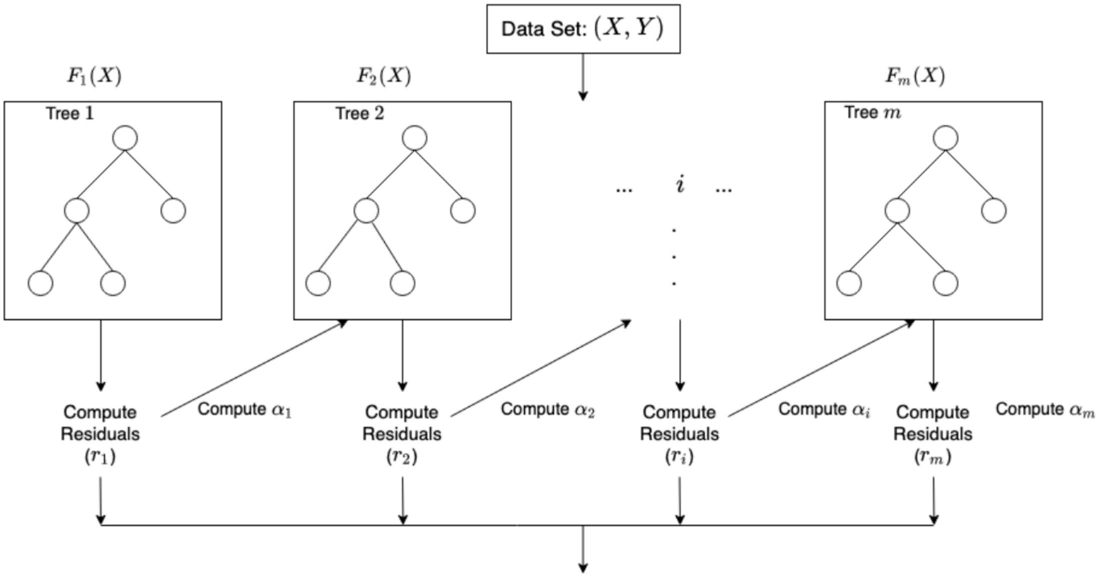
3. Fitting New Trees: A new decision tree is constructed to fit the residuals from the previous step. The tree aims to capture the patterns in the residuals, effectively learning the errors made by the current model.

4. Additive Boosting: The new decision tree is added to the ensemble with a weight determined by a learning rate parameter. This ensures that the new tree contributes to the overall model without completely overriding the previous trees.

5. Iteration: Steps 2-4 are repeated for a predetermined number of iterations or until a stopping criterion is met (e.g., maximum depth of trees, early stopping based on a validation set).

During the training process, XGBoost employs several techniques to improve its performance and prevent overfitting:

1. Regularization: XGBoost incorporates L1 and L2 regularization terms in the objective function, which helps to control the complexity of the individual trees and prevent overfitting.
2. Tree Pruning: XGBoost can automatically prune the decision trees during the training process, removing splits that do not contribute significantly to the objective function. This helps to reduce the complexity of the trees and improve generalization.
3. Feature Subsampling and Column Subsampling: XGBoost can randomly subsample the features and columns during the tree construction process, which introduces randomness and reduces the correlation between the trees in the ensemble.



$$F_m(X) \ = \ F_{m-1}(X) + \alpha_m h_m(X, r_{m-1}),$$

where α_i, and r_i are the regularization parameters and residuals computed with the i^{th} tree respectfully, and h_i is a function that is trained to predict residuals, r_i using X for the i^{th} tree. To compute α_i we use the residuals computed, r_i and compute the following: $arg \ \min\limits_{\alpha} \ = \sum_{i=1}^{m} L(Y_i, F_{i-1}(X_i) + \alpha h_i(X_i, r_{i-1}))$ where $L(Y, F(X))$ is a differentiable loss function.

3.4.2) Clustering

3.4.2.1) DBSCAN - Density Based Spatial Clustering of Applications with Noise

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a widely used unsupervised machine learning algorithm for clustering and outlier detection. Unlike traditional clustering techniques, DBSCAN does not require the number of clusters to be specified a priori and can effectively handle datasets with arbitrary shapes and noise.
In the context of outlier detection, DBSCAN leverages the concept of density-based clustering to identify data points that are isolated or belong to low-density regions, which are considered as potential outliers. The algorithm works as follows:

1. Density Estimation: DBSCAN begins by estimating the density of each data point based on its neighborhood. Two key parameters are used for this purpose:

    - Epsilon (ε): Specifies the radius of the neighborhood around a data point.

    - MinPts: The minimum number of data points required within the

    - ε-neighborhood for a data point to be considered a core point.

2. Core Point Identification: A data point is classified as a core point if its ε-neighborhood contains at least MinPts data points. Core points are considered to be part of a dense region or cluster.

3. Cluster Formation: DBSCAN iteratively groups core points and their neighbors into clusters. If a non-core point falls within the ε-neighborhood of a core point, it is added to the same cluster as the core point.

4. Outlier Identification: Data points that are not part of any cluster are classified as outliers or noise points. These outliers can be either:

    - Border points: Non-core points that are within the ε-neighborhood of a core point but do not have enough neighbors to be considered core points themselves.
    - Noise points: Data points that are not reachable from any core point within the specified ε-neighborhood.

3.4.3) Outlier Detection

3.4.2.1) Isolation Forest

Isolation Forests is an unsupervised machine learning algorithm specifically designed for anomaly or outlier detection. It is based on the principle of isolating anomalies rather than profiling normal instances. The algorithm works by recursively partitioning the data space into smaller subspaces, where anomalies are expected to be isolated sooner than normal instances.

The key idea behind Isolation Forests is that anomalies are few and different from the majority of the data. Therefore, they are more susceptible to being isolated or separated from the rest of the data points during the partitioning process. The algorithm works as follows:

1. Tree Construction: Isolation Forests build an ensemble of isolation trees, which are binary trees constructed by recursively partitioning the data space. Each tree is built independently on a random subsample of the data.

2. Recursive Partitioning: At each node of the tree, a random feature and a random split value are selected to partition the data space into two subspaces. The partitioning process continues recursively until all instances are isolated or a maximum tree depth is reached.

3. Path Length Calculation: For each instance in the dataset, the path length (the number of edges traversed from the root to the terminating node) is calculated across all trees in the ensemble. Anomalies are expected to have shorter path lengths compared to normal instances because they are isolated earlier in the partitioning process.

4. Anomaly Score Computation: The anomaly score for each instance is computed based on the average path length across all trees in the ensemble. Instances with shorter average path lengths are considered more likely to be anomalies.

5. Outlier Identification: A threshold is set to determine the cutoff for classifying instances as outliers or inliers based on their anomaly scores. Instances with anomaly scores above the threshold are labeled as outliers, while those below the threshold are considered normal.

4) Experiments

4.1)

1. Data Normalization: Scaling the data to a standard range to ensure that all features contribute equally to the analysis.
2. Feature Selection: Identifying and selecting the most relevant features to improve model performance and reduce dimensionality.
3. Data Imputation: Handling missing values in the dataset by imputing or removing them to ensure data completeness.
4. Data Balancing: Addressing class imbalance in the dataset to prevent bias in the model's predictions.
5. Data Splitting: Splitting the dataset into training and testing sets to evaluate model performance.
6. Data Encoding: Converting categorical variables into numerical format using techniques such as one-hot encoding or label encoding.
7. Data Normalization: Scaling the data to a standard range to ensure that all features contribute equally to the analysis.
8. Feature Selection: Identifying and selecting the most relevant features to improve model performance and reduce dimensionality.
9. Data Imputation: Handling missing values in the dataset by imputing or removing them to ensure data completeness.
10. Data Balancing: Addressing class imbalance in the dataset to prevent bias in the model's predictions.
11. These pre-processing methods are essential for preparing the data before training machine learning models to ensure accurate and reliable results

The final processed dataset would likely have the following characteristics after undergoing pre-processing methods:

1. Normalized Features: The features in the dataset would be scaled or normalized to a standard range, such as between 0 and 1, to ensure uniform contribution from all features.
2. Selected Features: After feature selection, the dataset would contain only the most relevant features that have been identified as important for model training and prediction.
3. Imputed Data: Any missing values in the dataset would have been handled through imputation or removal, ensuring that the dataset is complete and ready for analysis.

4. Balanced Classes: If the dataset had class imbalance, techniques such as oversampling, undersampling, or synthetic data generation might have been applied to balance the classes.
5. Encoded Categorical Variables: Categorical variables would have been encoded into numerical format using techniques such as one-hot encoding or label encoding to make them suitable for machine learning algorithms
6. Split into Training and Testing Sets: The dataset would have been split into training and testing sets to facilitate model training and evaluation.

Overall, the final processed dataset would be optimized for machine learning tasks, with standardized features, relevant attributes, and balanced classes, ready for training and testing predictive models.

4.2)

Accuracy: measures the overall correctness of the model across all classes. It's calculated as:
Accuracy = (Number of Correct Predictions Total Number of Predictions Total/Number of Predictions Number of Correct Predictions).
Accuracy is a straightforward metric but can be misleading in cases of imbalanced datasets, where one class dominates the others. For instance, if 90% of your data belongs to one class, a model that simply predicts this majority class for all instances can achieve a high accuracy, but it might not be effective in practical terms.
Precision: Precision measures the accuracy of positive predictions. It's calculated as: Precision=(True Positives)/(True Positives+False Positives)
Precision indicates how many of the predicted positive instances are actually positive. It is useful when the cost of false positives is high.
Recall: also known as sensitivity or true positive rate, measures the ability of the model to identify all relevant instances (true positives). It's calculated as:recall=(True Positives)/(True Positives+False Negatives).
Recall is important when the cost of false negatives (missing positive cases) is high.
F1 Score: It is the harmonic mean of precision and recall. It combines both precision and recall into a single metric. It's calculated as: F1=2*(precision*recall)/(precision+recall).
The F1 score is particularly useful when you want to seek a balance between precision and recall.
Confusion Matrix:A confusion matrix is a table that summarizes the performance of a classification model. It presents a comparison of predicted classes against actual classes. The confusion matrix consists of four main terms:
a)**True Positives (TP)**: Correctly predicted positive instances.
b)**True Negatives (TN)**: Correctly predicted negative instances.
c)**False Positives (FP)**: Incorrectly predicted as positive when the actual class is negative (Type I error).
d)**False Negatives (FN)**: Incorrectly predicted as negative when the actual class is positive (Type II error).

The confusion matrix provides a detailed breakdown of the model's predictions, showing the true positive, false positive, true negative, and false negative values. It helps in understanding the model's performance in terms of correctly and incorrectly classified instances. From the confusion matrix, we can compute metrics like accuracy, precision, recall, and F1 score.

4.3)

Exploratory Data Analysis

Engaging in Exploratory Data Analysis (EDA) facilitates effective data preprocessing by visually exploring attributes and their interrelationships through graphs. This process aids in identifying data quality issues, understanding data distributions, detecting relationships between variables, handling categorical data, guiding feature engineering, selecting appropriate models, and communicating insights to stakeholders. EDA serves as a foundational step towards informed data preprocessing decisions, enhancing overall data understanding and paving the way for robust analysis and modeling.

The exploratory data analysis (EDA) conducted on the transaction dataset focused on fraud detection. Visualizations including a pie chart and histogram provided insights into the distribution of normal versus fraud transactions and the characteristics of dataset attributes. The pie chart highlighted the proportion of fraud transactions, while the histogram showcased attribute distributions and patterns. These EDA techniques are foundational for understanding the dataset and informing subsequent data preprocessing and analysis for fraud detection.

Logistic Regression

Logistic Regression with Undersampling :

- Undersampling Technique :
  - Balances dataset by randomly selecting non-fraudulent transactions to match fraudulent ones.

Model Characteristics :

- Logistic Regression:
  - Binary classifier estimating fraud probability based on transaction attributes.
  - Utilizes logistic function and L2 regularization.
  - Adjustable parameters like C control model complexity.

Evaluation Metrics :

- Confusion Matrix:
  - Details true positives, true negatives, false positives, and false negatives.
- Accuracy, Precision, Recall, F1-score:

- Measures correctness, positive prediction accuracy, sensitivity, and balance of precision and recall.

ROC Curve and AUC :

- ROC Curve:
  - Plots true positive rate against false positive rate at different thresholds.
- AUC (Area Under the Curve):
  - Quantifies model performance across all thresholds.

Support Vector Classifier

The proposed method employs several key data mining techniques to address the problem of fraud detection:

1. Feature Selection Using Correlation Ranking :

- Utilizes correlation coefficients between features and the target variable (fraud class) to rank features by importance.
- Focuses on selecting the top-ranked features based on absolute correlation values to reduce dimensionality and potentially improve model performance.

2. Class Imbalance Handling :

- Addresses the issue of class imbalance (few instances of fraud compared to non-fraud) by creating a balanced dataset for training.
- Achieves balance by sampling an equal number of non-fraud instances to match the number of fraud instances.
- This approach helps prevent the model from being biased towards the majority class (non-fraud) and potentially improves fraud detection accuracy.

3. Support Vector Machine (SVM) Classifier :

- Utilizes SVM with a linear kernel for classification, a popular machine learning algorithm known for its effectiveness in handling high-dimensional data and non-linear relationships.
- SVM can handle both linear and non-linear classification tasks by finding an optimal hyperplane that maximizes the margin between different classes.
- In this case, SVM is used to predict fraud (class 1) versus non-fraud (class 0) based on the selected features.

4. Model Evaluation Metrics :

- Employs confusion matrix analysis to evaluate model performance, particularly focusing on fraud detection.

- Calculates accuracy, which measures the overall correctness of the model predictions, and also considers the precision and recall of fraud detection.
- The approach also uses a custom evaluation criterion that gives weight to both overall accuracy and fraud detection rates, reflecting the importance of correctly identifying fraudulent transactions.

5. Class Weight Adjustment :

- Demonstrates the use of class weights within SVM to address class imbalance directly during model training.
- Adjusts class weights to assign higher importance to the minority class (fraud) during model training, which can improve the model's ability to detect instances of fraud.

Random Forest:

The provided method utilizes two key data mining techniques: Random Forest and Naive Bayes.

Random Forest:

- Ensemble Learning: Random Forest is an ensemble method that combines multiple decision trees during training. Each tree is trained on a subset of the data, and predictions are aggregated to improve overall accuracy and robustness.
- Feature Importance: Random Forest can assess the importance of different features in making predictions, enabling feature selection and identifying influential variables.
- Handles Large Datasets: It can effectively handle large datasets with many features by randomly sampling subsets of features for each tree and aggregating their predictions.
- Reduces Overfitting: By using multiple trees and random feature subsets, Random Forest reduces overfitting compared to individual decision trees, making it suitable for complex datasets.

Naive Bayes:

- Probabilistic Classifier: Naive Bayes is a probabilistic classifier based on Bayes' theorem, assuming independence among features. It calculates the probability of a class given input features using conditional probabilities.
- Efficient with High-Dimensional Data: Naive Bayes performs well with high-dimensional data and is computationally efficient, making it suitable for datasets with many features.
- Assumption of Feature Independence: Despite its "naive" assumption of feature independence, Naive Bayes often performs surprisingly well in practice and can provide fast predictions.
- Interpretable Results: Naive Bayes provides straightforward probability estimates for each class, making it interpretable and useful for understanding the model's decision-making process.

Isolation Forest and Local Outlier Factor (LOF):

Isolation Forest:

- Ensemble-Based Approach: Isolation Forest is an ensemble method that isolates anomalies by randomly selecting a feature and then randomly selecting a split value between the maximum and minimum values of that feature.
- Efficient with High-Dimensional Data: It efficiently handles high-dimensional data and is less sensitive to outliers in multi-dimensional spaces compared to traditional distance-based methods.
- Scalable and Fast: Isolation Forest is computationally efficient and scalable, making it suitable for large datasets. It constructs a forest of random decision trees to isolate anomalies efficiently.

Local Outlier Factor (LOF):

- Density-Based Approach: LOF is a density-based anomaly detection method that computes the local density deviation of a data point with respect to its neighbors. Anomalies are points with significantly lower density than their neighbors.
- Adaptive to Local Structures: LOF adapts well to local data structures and can identify anomalies based on variations in local densities, making it effective for identifying outliers in complex datasets.
- Sensitive to Neighborhood Size: The performance of LOF depends on the choice of the number of neighbors (k), allowing flexibility in defining what constitutes a "local" neighborhood.

XG BOOST

The proposed method employs two advanced data mining techniques for classification tasks, specifically focusing on fraud detection in credit card transactions: XGBoost (Extreme Gradient Boosting) and hyperparameter tuning using GridSearchCV.

XGBoost (Extreme Gradient Boosting):

- Gradient Boosting Ensemble: XGBoost is an ensemble learning technique that builds a series of decision trees sequentially, with each subsequent tree correcting the errors made by the previous ones. It combines weak learners (individual decision trees) to create a strong predictive model.
- Regularization: XGBoost incorporates L1 (Lasso) and L2 (Ridge) regularization techniques to prevent overfitting and improve generalization performance.
- High Performance: XGBoost is optimized for performance and efficiency, featuring parallel and distributed computing capabilities to handle large datasets efficiently.
- Flexibility: XGBoost supports various objective functions and evaluation criteria for classification tasks, allowing customization based on specific use cases and metrics.

- Feature Importance: XGBoost provides insights into feature importance, allowing users to understand which features contribute most significantly to the model's predictions.

GridSearchCV for Hyperparameter Tuning:

- Parameter Optimization: GridSearchCV systematically searches through a specified grid of hyperparameters to find the best combination that maximizes model performance based on a chosen scoring metric.
- Cross-Validation: GridSearchCV employs cross-validation to evaluate each combination of hyperparameters, ensuring robustness and reducing the risk of overfitting.
- Customizable Grid: Users can define a grid of hyperparameters to search over, including learning rate, maximum depth of trees, and number of estimators (trees) in the ensemble.
- Automated Model Selection: GridSearchCV automates the process of hyperparameter tuning, helping to fine-tune the model without manual intervention and providing insights into the impact of different parameter configurations on model performance.

5)

Based on the outcomes and evaluation metrics presented for each method applied to fraud detection in credit card transactions, here's a summary and comparative analysis:

Logistic Regression with Undersampling:

- Outcome: The logistic regression model with undersampling achieved high recall (sensitivity) for detecting fraud (class 1), indicating that it was effective in identifying a high proportion of actual fraud cases. However, the precision (positive predictive value) was relatively low, suggesting a higher rate of false positives.
- Evaluation Metrics: The model achieved an overall accuracy of 97%, with a significant imbalance in class distribution affecting precision and recall.
- Comparative Analysis: While the recall rate for fraud detection was impressive, the high false positive rate implies a need for further tuning to improve precision without compromising recall.

Support Vector Machine (SVM):

- Outcome: The SVM model achieved a high accuracy of 97% and identified 180 out of 199 total fraud cases, resulting in a recall rate of 90.5%.
- Evaluation Metrics: The model exhibited a trade-off between precision and recall, with a focus on accurately detecting fraud cases.
- Comparative Analysis: SVM performed well in identifying fraudulent transactions with a high recall rate, but its precision could be further optimized.

Random Forest:

- Outcome: The Random Forest model yielded an Area Under the Receiver Operating Characteristic Curve (AUROC) of 0.909, indicating good discrimination ability between classes.
- Evaluation Metrics: The model's performance was notable, with a focus on accurately classifying transactions and achieving a high AUROC.
- Comparative Analysis: Random Forest demonstrated robustness in handling imbalanced data and achieving good overall performance in fraud detection.

Isolation Forest and Local Outlier Factor (LOF):

- Outcome: Isolation Forest outperformed LOF and SVM in terms of detecting fraudulent cases, achieving a higher accuracy of 99.74%.
- Evaluation Metrics: Isolation Forest demonstrated superior precision and recall for fraud detection, indicating its effectiveness in identifying anomalous transactions.
- Comparative Analysis: Isolation Forest's performance highlights the strength of anomaly detection techniques in identifying fraudulent activities with high accuracy and precision.

XGBoost:

- Outcome: XGBoost achieved a high overall accuracy of 99.96% and demonstrated excellent recall (sensitivity) for detecting fraud.
- Evaluation Metrics: The model exhibited strong performance in correctly classifying both normal and fraudulent transactions, with a balanced precision and recall.
- Comparative Analysis: XGBoost's effectiveness in handling imbalanced data and achieving high accuracy underscores its suitability for fraud detection tasks.

Comparative Analysis:

- Model Performance: XGBoost and Random Forest emerged as top performers, demonstrating high accuracy and robustness in handling imbalanced data for fraud detection.
- Precision-Recall Trade-off: SVM and Logistic Regression with Undersampling showed a trade-off between precision and recall, highlighting the challenge of balancing accurate fraud detection with minimizing false positives.
- Efficiency and Complexity: Isolation Forest and LOF excelled in anomaly detection but may require additional computational resources compared to traditional classification algorithms like Random Forest and XGBoost.

Testing with Another Dataset:

- The proposed methods should be tested on diverse datasets to assess generalizability and efficiency across different contexts.
- Efficiency can vary based on dataset characteristics, including the degree of class imbalance, feature distributions, and the nature of fraud patterns.
- Deep learning techniques could be explored for enhanced performance, balancing computational cost with accuracy and scalability.

In summary, while each method showed strengths in detecting fraudulent transactions, XGBoost and Random Forest stood out for their overall accuracy and robustness. Further experimentation and testing with diverse datasets are recommended to validate and refine the proposed methods for real-world fraud detection applications.

6)

CONCLUSION:

In this data mining project, the primary objective was to develop effective fraud detection models using various machine learning and anomaly detection techniques on a highly imbalanced credit card transaction dataset. The dataset consisted of a large number of legitimate transactions (class 0) and a small number of fraudulent transactions (class 1), making it challenging to identify and distinguish the minority class accurately.

To address this problem, several methods were implemented and evaluated:

1. **Logistic Regression with Undersampling:**

   - Undersampled the majority class (legitimate transactions) to balance the dataset.
   - Achieved high recall (ability to detect fraudulent transactions) but at the cost of precision.

2. **Support Vector Machine (SVM):**

   - Trained an SVM classifier with a radial basis function (RBF) kernel.
   - Achieved high accuracy and recall, but lower precision due to class imbalance.

3. **Random Forest:**

   - Employed an ensemble of decision trees using random forest.
   - Achieved excellent performance in distinguishing between classes with high precision and recall.

4. **Isolation Forest and Local Outlier Factor (LOF):**

   - Utilized anomaly detection techniques such as isolation forest and LOF to identify fraudulent transactions based on their distinctiveness from normal transactions.
   - Isolation forest performed particularly well in detecting anomalies with high precision.

5. XGBoost (Extreme Gradient Boosting):

- Implemented XGBoost, a gradient boosting algorithm known for its accuracy and efficiency.
- Achieved outstanding performance with high accuracy, precision, recall, and F1-score.

The comparative analysis of these methods revealed that ensemble techniques like random forest and XGBoost outperformed other approaches in terms of overall accuracy and the ability to detect fraudulent transactions accurately. Isolation forest also demonstrated strong performance in anomaly detection, while logistic regression with undersampling provided a trade-off between precision and recall.

Future work:

- Exploring advanced feature engineering techniques to capture more meaningful information from transactional data.
- Fine-tuning model hyperparameters using techniques like grid search or Bayesian optimization to optimize performance.
- Experimenting with different ensemble methods or deep learning architectures to further enhance fraud detection capabilities.
- Conducting extensive cross-validation and testing on larger datasets to evaluate model robustness and generalizability.

7)

References:

1. *A. Singh, A. Singh, A. Aggarwal and A. Chauhan, "Design and Implementation of Different Machine Learning Algorithms for Credit Card Fraud Detection," 2022 International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME), Maldives, Maldives, 2022, pp. 1-6, doi: 10.1109/ICECCME55909.2022.9988588.*

2. *https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud*