Research Report

Credit Card Fraud Detection - Machine Learning Methods

Presented By
Ibrahim Dayah
Abdullahi

CHAPTER 1: INTRODUCTION

1.1 Background

Fraud committed when scammers open new card accounts or hijack existing ones doubled in 2022 to reach almost £52m ($65m), according to new figures from UK Finance [1]. Losses from transaction fraud frequently represent more than 10% of a company's overall costs. Companies are continuously looking for new ways to prevent, detect, and eliminate fraud due to the worry over these significant losses. One of the most promising technological tools to combat financial fraud is machine learning [2].

1.2 Problem Statement

In this research, we focus on the development and evaluation of two machine learning models, namely Logistic Regression and K-Nearest Neighbors (KNN), for the detection of fraudulent credit card transactions. The dataset, which can be found on Kaggle, is made up of transactions made by Europeans using their credit cards over the course of two days in September 2013. A very unbalanced data collection, it contains only 492 fraud transactions, or 0.172% of the total 284,807 transactions, making it. As a result of a PCA transformation, the input variables are numerical. The original data and any supporting information were not made public due to confidentiality concerns[3].

1.3 Objectives

The primary objective of this project is to develop Logistic Regression and a KNN models that can identify fraud in credit card transactions, and then to assess how well each model performs. Since most transactions are legitimate, it is significantly harder to identify the fraudulent transactions, hence the largest challenge we faced is to develop models that are particularly sensitive to fraudulent transactions.

1.4 Scope

The scope of this research is the classification of credit card transactions as fraud or legitimate using machine learning methods on real world data. To achieve that, we used Logistic Regression and KNN algorithms and compared their performances. This paper will also focus on handling class imbalance through under sampling and feature preprocessing to increase the accuracy of the models. 1.5 Organization of the Paper The rest of this paper is organized as follows: Chapter 2 gives detailed review on credit card fraud, feature selection detection techniques and performance comparison. Chapter 3 describes the experimental setup approach including the data pre-processing, the two classifier methods on credit card fraud detection and reports the experimental results and discussion about the comparative analysis. Chapter 4 concludes the comparative study and suggests future areas of research.

CHAPTER 2: LITERATURE REVIEW

Credit card transaction classification is primarily a binary classification problem. Credit card transactions are classified as either valid (negative class) or fraudulent (positive class). Fraud detection is commonly seen as a data mining classification challenge, with the goal of correctly classifying credit card transactions as legal or fraudulent [4].

Fraudulent actions cause significant loss, motivating academics to develop a way to identify and prevent fraud. Several approaches have been developed and tested. Some of these are discussed briefly below.

Classical algorithms including Gradient Boosting (GB), Support Vector Machines (SVM), Decision Tree (DT), LR, and RF have proven to be beneficial. In study [5] GB, LR, RD, SVM, and a combination of classifiers were utilized, resulting in a high recall of over 91% on a European dataset. Only after balancing the dataset by under sampling the data were high precision and recall achieved. In publication [6], a European dataset was also employed, and models based on LR, DT, and RF were compared. RF showed to be the most accurate of the three models, with an accuracy of 95.5%, followed by DT with 94.3% and LR with 90%.

According to [7] and [8], k-Nearest neighbors (KNN) and outlier detection approaches can also be effective in detecting fraud. They have been shown to be effective in reducing false alarm rates and raising fraud detection rates. The KNN algorithm also fared well in an experiment for a study [9], in which the authors tested and compared it to other traditional algorithms.

Several publications have been published on the detection of fraudulent transactions using deep neural networks. These models, on the other hand, are computationally expensive and perform better on larger datasets [16]. This strategy may produce excellent results, as evidenced by certain studies, but what if the same, or even better, results can be obtained with fewer resources? Our main goal is to demonstrate that with proper preprocessing, several machine learning algorithms may produce acceptable results.

Given those facts, we chose to examine the suitability of Logistic Regression and KNN models for detecting credit card fraud. An experiment was carried out in an attempt to accomplish this.

2.1 Model Selection Factors

1. Variables: Both the LR and KNN models' input variables contained columns such as transaction features, amount, time, and a plethora of other anonymized numerical features (e.g., V1, V2,..., V28). The "Class" column, which indicates whether a transaction is fraudulent (Class=1) or not (Class=0), is the target variable.
2. Parameters: For the Logistic Regression model, model selection involved choosing a regularization parameter (lambda) that helps limit overfitting. Also, the learning rate (0.01) and number of iterations (1000) were important parameters we had to choose when training the model using gradient descent.
3. Training Function: For Linear Regressions, the training process consisted of using the gradient descent algorithm to update the models’ weights iteratively to minimize the difference between the predicted probability and true class. On the other hand, KNN is a lazy learning algorithm that doesn’t need actual training, it just stores the entire dataset in

memory and calculates the k-nearest neighbors for each data point in the testing dataset using Euclidean distance and classifies it according to the majority class of the k-nearest neighbors. However, for KNN, the most important thing is choosing the number of nearest neighbors, K, to consider for each testing data point to achieve accurate classification.

4. Training Goal: Our general training goal was to achieve an accuracy of 0.9 or above for our models when performing on unseen data. The goal for the Logistic Regression model was to come up with the optimal weights that can classify the data correctly, while for KNN it was to identify the closest k-neighbors for each sample in the dataset and predict its class according to the majority class of its k-nearest neighbors. The difficulty we faced was that the dataset was highly unbalanced with so little fraud and so many non-frauds, so before unbalancing the data our LR model achieved an overall accuracy of 0.998 but wrongly classified about 40% of fraudulent unseen test data. But when we used under sampling to balance the dataset the overall accuracy decreased to 0.96 but the model performed much better on unseen fraudulent data.

2.2 Feature (Variables) Selection

The examination of cardholder spending behavior is the foundation of credit card fraud detection. This expenditure profile is analyzed using an optimal variable selection that captures the unique behavior of a credit card. Both the profile of a valid and fraudulent transaction is continually evolving. To ensure effective credit card transaction classification, an optimal selection of variables that significantly separates both profiles is required [11].

The variables that comprise the card usage profile, as well as the approaches employed, have an impact on the performance of credit card fraud detection systems. These characteristics are derived from a credit card's transaction and historical transaction history. All transactions statistics, regional statistics, merchant type statistics, time-based amount statistics, and time-based number of transactions statistics are the five main variable kinds [12].

The variables in all transactions statistics category provide the card's overall usage profile. The variables under regional statistics type show the card's spending habits while taking geographical regions into account. The variables under merchant statistics type display the card's usage in various merchant categories. The variables of time-based statistic types identify the cards' usage profile in terms of usage amounts vs time ranges or usage frequencies versus time ranges [13].

Much of the literature concentrated on cardholder profiles rather than card profiles. It goes without saying that a person can use two or more credit cards for different purposes. As a result, multiple spending profiles can be displayed on such cards. The focus of this study is on the card rather than the cardholder because one credit card can only have one spending profile, but a cardholder can have several spending profiles on different cards. [14] employs 30 variables, [15] employs 27 variables, while [16] reduces 20 variables to only 16 variables that are relevant.

In this paper we first use all the variables for both our models, then we remove some of the columns/variables for the Logistic Regression model and compare its performance to before we removed them.

CHAPTER 3: METHODOLOGY

This chapter covers the dataset utilized in the experiments as well as the two classifiers under consideration: k-Nearest Neighbor, and Logistic Regression. The steps involved in developing classifiers are as follows: data collection, preprocessing of data, data analysis, training of the classifier algorithm, and testing (evaluation).

3.1 Dataset

The dataset utilized in this study was obtained from Kaggle and consists of credit card transactions performed by European cardholders over the course of two days in September 2013. The dataset is severely skewed, with only 492 fraudulent transactions out of a total of 284,807 transactions, accounting for only 0.172% of the total as can be seen from Figure 1.

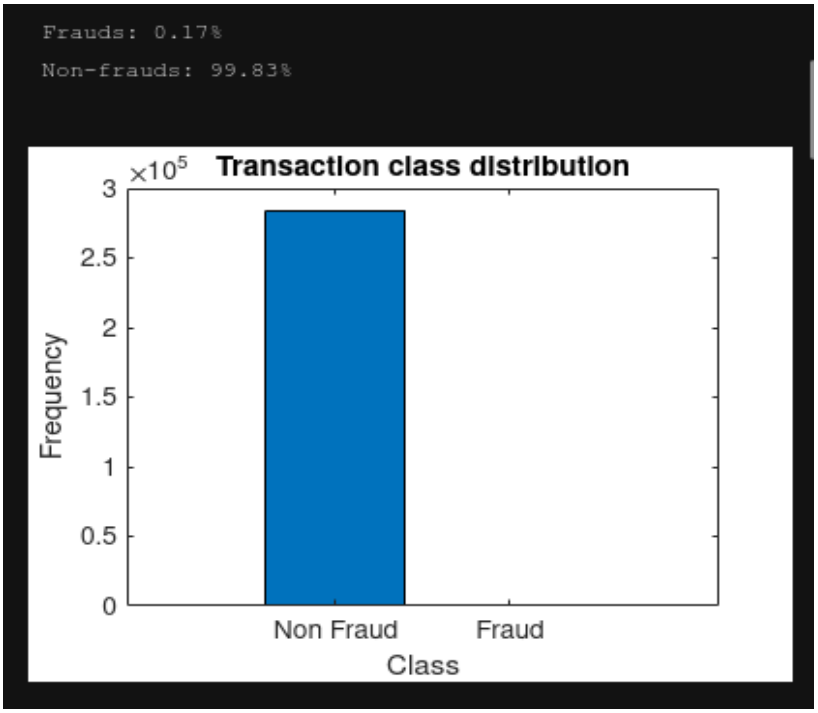

Figure 1: *Dataset distribution*

There are 31 numerical features in the dataset. Because some of the input variables contain financial information, the PCA transformation of these input variables was conducted to ensure that the data remained anonymous. Three of the specified characteristics were not transformed. The "Time" feature displays the time between the first transaction and each subsequent transaction in the dataset. The "Amount" feature displays the total amount of credit card transactions. The label is represented by the feature "Class," which has only two values: 1 in the case of a fraudulent transaction and 0 otherwise [17].

Because the classes are severely unbalanced and the distribution ratio of classes plays a key role in model accuracy and precision, data preprocessing is critical.

3.2 Data Preprocessing

When classification categories are not relatively equally distributed, machine learning algorithms struggle to learn. Given that the presented data is severely unbalanced, some sort of balancing is required so that the model may be trained efficiently. Under sampling the dominant class, oversampling the minority class, or a combination of the two approaches are often used for modifying the class distribution. Accuracy in a highly unbalanced data set does not represent a correct value for the efficiency of a model. Initially, a method should be applied to balance the data before considering any performance evaluation metrics [18]. So, we decided to use under sampling to balance the data and obtain a more accurate model.

Under sampling is the method of removing many of the majority class records from the sample. In our case, we had to remove random records from the majority non-fraud class so that we can obtain a number of records close to that of the minority fraud class which would then enable us to train the model on balanced data as can be seen in Figure 2.

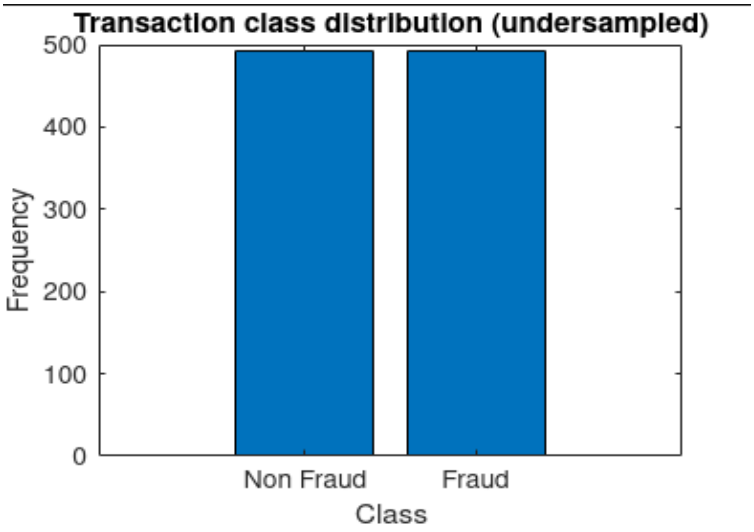

Figure 2: *Dataset distribution after under sampling*

3.3 Logistic Regression Classifier

Logistic Regression is a statistical model that employs a functional approach to estimate the probability of a binary response based on one or more variables/features. It determines the parameters that best suit a nonlinear function known as the sigmoid. Figure 3 depicts the sigmoid function's implementation in MATLAB.

Define the sigmoid function:

```
function output = sigmoid(x)
    output = 1 ./ (1 + exp(-x));
end
```

Figure 3: *Sigmoid function in MATLAB code*

To build our Logistic Regression model using MATLAB we followed the steps below:

1. **Data Splitting:** After under sampling the dataset was split into training and testing sets. 70% of the data was set aside for training while 30% was used for testing as can be seen in Figure 4.

Applying the undersampling technique and then splitting the data into training and testing datasets

```
% The function "sum" counts the number of classes = 1 and saves it as an object "fraud_records"
fraud_records = sum(data.Class == 1);

% Defines the index for fraud and non-fraud in the lines:
fraud_indices = find(data.Class == 1);
not_fraud_indices = find(data.Class == 0);

% Randomly collect equal samples of each type:
under_sample_indices = randsample(not_fraud_indices, fraud_records, false);
data_undersampled = data([fraud_indices; under_sample_indices], :);
X_undersampled = data_undersampled(:, 2:30);
Y_undersampled = data_undersampled.Class;

% Split the undersampled data into training and testing sets
[X_undersampled_train, X_undersampled_test, Y_undersampled_train, Y_undersampled_test] = splitdata( ...
    X_undersampled, Y_undersampled, 0.3);
```

Figure 4: Undersampling and splitting the data into training and testing sets

We used the training set to train the model, and the testing set was used to evaluate its performance on unseen data.

2. **Training the Model:**

   The logistic regression model was initialized using random weights and biases. During the training process, the model's weights were iteratively updated using the gradient descent algorithm to minimize the gap between the predicted probability and the true class labels (fraud or not fraud) as can be seen in Figure 5.

Building and training the logistic regression model simultaneously

```
% Prepend a column of ones to the training data table for the bias term
X_train_augmented = [ones(size(X_train, 1), 1), table2array(X_train)];

% Initialize the weights randomly
num_features = size(X_train_augmented, 2);
weights = rand(num_features, 1);

% Set the learning rate and number of iterations
learning_rate = 0.01;
num_iterations = 1000;

% Perform gradient descent
for iter = 1:num_iterations
    % Calculate the predicted probabilities using the current weights
    z = X_train_augmented * weights;
    predicted_probs = sigmoid(z);

    % Calculate the error between the predicted probabilities and the true labels
    error = predicted_probs - table2array(y_train);

    % Update the weights using gradient descent
    gradient = X_train_augmented' * error;
    weights = weights - learning_rate * gradient;
end

% Store the trained model in a structure
model.weights = weights;
model.threshold = 0.5;
model.predict = @(X) double(sigmoid([ones(size(X, 1), 1), table2array(X)] * model.weights) >= model.threshold);
```

Figure 5: Building and training the LR model simultaneously

The sigmoid function was used to calculate predicted probabilities, and the weights were updated based on the gradient of the error. After completing the iterations, the trained model was stored in a structure containing the learned weights and a threshold of 0.5 for class predictions. A predict function was then defined using the model, enabling the model to make predictions on new data by applying the trained weights and threshold after augmenting the data with a column of ones for the bias term.

To avoid overfitting, we chose a regularization parameter (lambda) and modified the learning rate (0.01) and number of iterations (1000) throughout the training phase. The logistic regression model's training goal was to discover the best weights that could correctly classify the data as fraudulent or real.

3. Making Predictions/Classifications:

After the completion of the model training, we used the trained Logistic Regression model to predict the class labels for the testing dataset as can be seen in Figure 6.

```
% Convert the test data table to a numeric matrix
X_test_numeric = table2array(X_test);

% Calculate predictions
y_pred = model.predict(X_test);
```

Figure 6: *Using the LR model to make classifications on test data*

Each transaction was then assigned a probability, indicating whether it was fraudulent or legitimate.

4. **Model Performance Evaluation:**

   To assess the model's performance, we compared the predicted class labels to the true class labels in the testing set. We generated its confusion matrix and evaluated different performance measures, such as accuracy, precision, recall, and F1-score, to assess the model's capacity to detect fraudulent transactions and overall performance as can be seen in Figure 7.

```
% Calculate confusion matrix
cmat = confusionmat(table2array(y_test), y_pred);

% Display confusion matrix
confusionchart(cmat, {'Non-Fraud', 'Fraud'});

% Calculate performance measures
accuracy = sum(diag(cmat)) / sum(cmat(:));
precision = cmat(1, 1) / sum(cmat(:, 1));
recall = cmat(1, 1) / sum(cmat(1, :));
f1_score = 2 * (precision * recall) / (precision + recall);

% Display performance measures
fprintf('Accuracy: %.16f\n', accuracy);
fprintf('Precision: %.16f\n', precision);
fprintf('Recall: %.16f\n', recall);
fprintf('F1 Score: %.16f\n', f1_score);
```

Figure 7: *LR model performance evaluations*

3.4 KNN Classifier

K-Nearest Neighbors (KNN) is a straightforward classification algorithm for pattern recognition and binary classification. KNN assigns a class label to each data point based on the majority class of its K-nearest neighbors in the feature space. The algorithm computes distances between the new data point and all training data points, then identifies the K data points with the shortest distances (i.e., nearest neighbors). The majority class among these K neighbors determines the

class of the new data point. Because it does not require an explicit training phase, KNN is termed a lazy learning algorithm; it memorizes the full training dataset and executes the classification task during the prediction step. However, choosing an acceptable number for K is critical, since a small K value may result in noisy conclusions, whilst a big K value may include irrelevant data points in the decision boundary. Figure 8 illustrates how the KNN algorithm works.

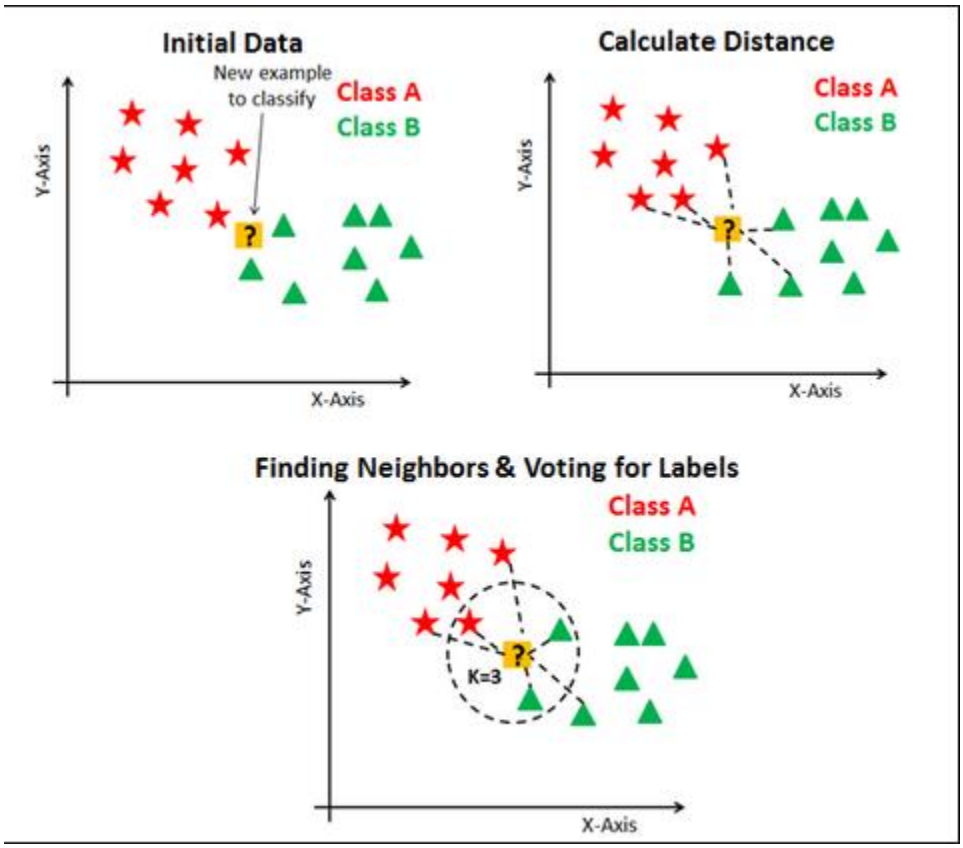

Figure 8: *How KNN works*

To build and test our KNN model using MATLAB we followed the steps below:

1. **Data Splitting:** After the data was loaded from the csv file, balanced using under sampling and preprocessed, it was split into training and testing sets (80% for training) and normalized using z-score normalization as can be seen in Figure 9.

```
% Prepare the data
features = new_data(:, 1:end-1);
labels = new_data(:, end);

% Split the data into training and testing sets (80% for training)
[train_indices, test_indices] = crossvalind('HoldOut', size(new_data, 1), 0.2);
X_train = features(train_indices, :);
X_test = features(test_indices, :);
y_train = labels(train_indices, :);
y_test = labels(test_indices, :);

% Normalize the data using z-score normalization from Statistics and Machine Learning Toolbox
X_train = normalize(X_train, 'zscore');
X_test = normalize(X_test, 'zscore');
```

Figure 9: *Splitting the data into training and testing sets for KNN*

2. **Finding the optimal K value:** The optimal value of K for KNN was found using k-fold cross-validation by evaluating accuracy for different K values as can be seen in Figure 10.

```
% Find the optimal K using k-fold cross-validation
k_range = 1:25;
knn_accuracy_train = zeros(length(k_range), 1);
knn_accuracy_test = zeros(length(k_range), 1);

for k_idx = 1:length(k_range)
    knn_model = customKNN(X_train, y_train, k_range(k_idx));
    y_pred_train = predict(knn_model, X_train);
    knn_accuracy_train(k_idx) = sum(y_train == y_pred_train) / numel(y_train);

    y_pred_test = predict(knn_model, X_test);
    knn_accuracy_test(k_idx) = sum(y_test == y_pred_test) / numel(y_test);
end

% Find the optimal K based on test accuracy
[~, optimal_k_idx] = max(knn_accuracy_test);
optimal_k = k_range(optimal_k_idx);
```

Figure 10: *Finding the optimal K value using cross-validation*

The accuracy versus K was then plotted to visualize the relationship.

3. **Training the KNN model and predicting labels:** After determining the appropriate K, the KNN model is trained on the training data using our customKNN function that maintains both the training data and the specified K. Because KNN is a lazy learning algorithm, the customKNN function does not perform any training. Figure 11 shows how we defined our customKNN function.

```
% Define the custom KNN function
function knn_model = customKNN(X_train, y_train, k)
    knn_model = struct();
    knn_model.X_train = X_train;
    knn_model.y_train = y_train;
    knn_model.k = k;
    % No training is needed for KNN as it's a lazy learning algorithm
end
```

Figure 11: *Defining the customKNN function*

To make classifications on new data, the predict function for custom KNN was defined as can be seen in Figure 12.

```
% Define the predict function for custom KNN
function y_pred = predict(knn_model, X_test)
    X_train = knn_model.X_train;
    y_train = knn_model.y_train;
    k = knn_model.k;

    num_test = size(X_test, 1);
    y_pred = zeros(num_test, 1);

    for i = 1:num_test
        % Calculate distances between the test data point and all training data points
        distances = sqrt(sum((X_train - X_test(i, :)).^2, 2));

        % Get the indices of the k-nearest neighbors
        [~, indices] = mink(distances, k);

        % Get the classes of the k-nearest neighbors
        k_nearest_classes = y_train(indices);

        % Predict the majority class among the k-nearest neighbors
        y_pred(i) = mode(k_nearest_classes);
    end
end
```

Figure 12: *Predict function for customKNN*

The distances to all training data points were calculated for each data point in the test set, the K-nearest neighbors were found and the projected class for the data point was then assigned to the majority class among the K-nearest neighbors using those two functions as can be seen in Figure 13.

```
% Train the k-NN model with the optimal K using customKNN
knn_model = customKNN(X_train, y_train, optimal_k);

% Predict labels for the testing set using customKNN
knn_predicted_test_labels = predict(knn_model, X_test);
```

Figure 13: *Training the KNN model and making predictions with it*

4. **Evaluating the performance of our KNN model:** The accuracy, precision, recall, F1 score, and confusion matrix were used to assess the performance of the KNN model as can be seen in Figure 14.

```
% Scoring k-NN
knn_accuracy_score = sum(y_test == knn_predicted_test_labels) / numel(y_test);
knn_precision_score = sum(y_test & knn_predicted_test_labels) / sum(knn_predicted_test_labels);
knn_recall_score = sum(y_test & knn_predicted_test_labels) / sum(y_test);
knn_f1_score = 2 * knn_precision_score * knn_recall_score / (knn_precision_score + knn_recall_score);
knn_MCC = corrcoef(y_test, knn_predicted_test_labels);

% Printing scores
fprintf('KNN Model Performance:\n');
%fprintf('MCC --> %.2f\n', knn_MCC);
fprintf('Accuracy:  %.2f\n', knn_accuracy_score);
fprintf('Precision: %.2f\n', knn_precision_score);
fprintf('Recall:    %.2f\n', knn_recall_score);
fprintf('F1:        %.2f\n', knn_f1_score);

% Display the confusion matrix
conf_matrix = confusionmat(y_test, knn_predicted_test_labels);
LABELS = {'Non-Fraud', 'Fraud'};
plotConfusionMatrix(conf_matrix, LABELS);
```

Figure 14: *Evaluating the performance of the KNN model*

The true positive, true negative, false positive, and false negative counts for the model's predictions were shown by the confusion matrix generated by the code above.

3.5 Results & Discussion

In this study, we designed and tested two classifier models, k-Nearest Neighbor (KNN) and Logistic Regression, on a binary classification task. The dataset was divided into 70% for model training and 30% for validation and testing. Both classifiers learned patterns and relationships within the features throughout the training phase in order to make predictions. The models were evaluated using a variety of performance indicators, including accuracy, precision, recall (sensitivity), F1 Score, and Confusion Matrix.

Accuracy showed the proportion of correctly classified transactions, precision showed the fraction of true positive predictions among positive predictions, recall showed the right identification of positive cases, and F1 Score provided a balanced measure that considered false positives and false negatives. The performance was represented by the Confusion Matrix in terms of true positives, true negatives, false positives, and false negatives. We gained significant insights into the strengths and shortcomings of both our models by using these measures.

3.5.1 Logistic Regression Model Performance Results & Discussion

Our Logistic Regression model got an amazing accuracy of 0.998 when it was trained and tested on the original unbalanced dataset. However, the model's accuracy concealed a critical flaw - its performance on fraud data. It wrongly classified more than seven out of every ten fraud transactions, showing weakness in its capacity to detect fraud reliably as can be seen in Figure 15.

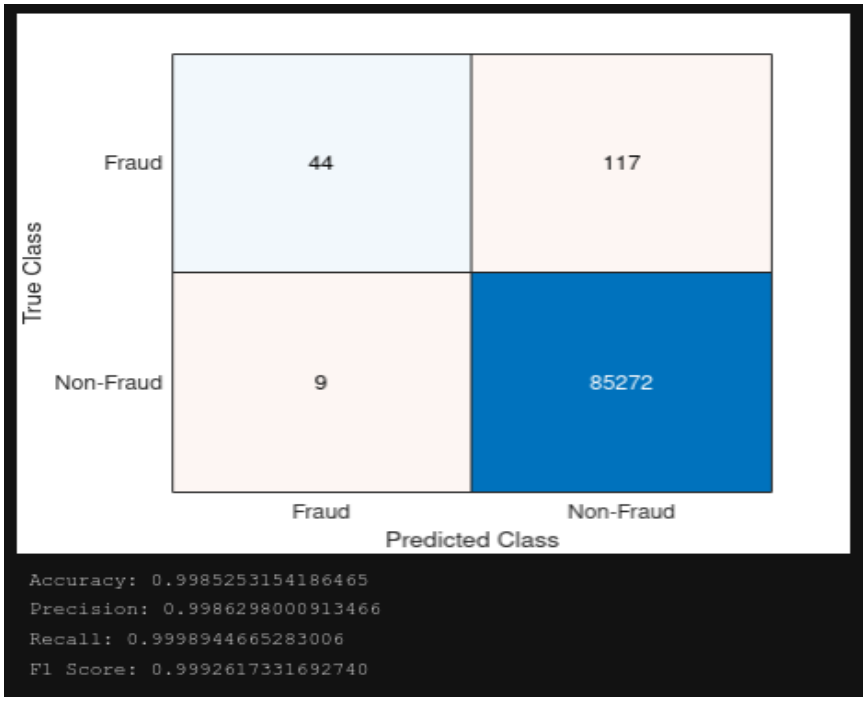

Figure 15: *Performance of LR model on unbalanced dataset*

This misclassification of fraudulent transactions is especially concerning because detecting fraudulent activity is critical for a credit card fraud detection application. In this case, the high accuracy can be attributed to the dataset's large majority of non-fraud transactions, which eclipsed the relatively tiny number of fraud incidents. Further changes and assessment procedures were required to address this imbalance and improve the model's performance on fraud detection, leading us to investigate techniques such as under-sampling and feature preprocessing to improve the model's sensitivity to fraudulent transactions.

We then tested the performance of the Logistic Regression model on the original unbalanced dataset after balancing it using fraud detection techniques and training it on the balanced data. The accuracy fell to 0.96, showing that the model's overall accuracy in classifying both fraudulent and non-fraudulent transactions was reduced marginally. The trade-off, however, resulted in a big improvement in detecting fraudulent transactions as can be seen in Figure 16.

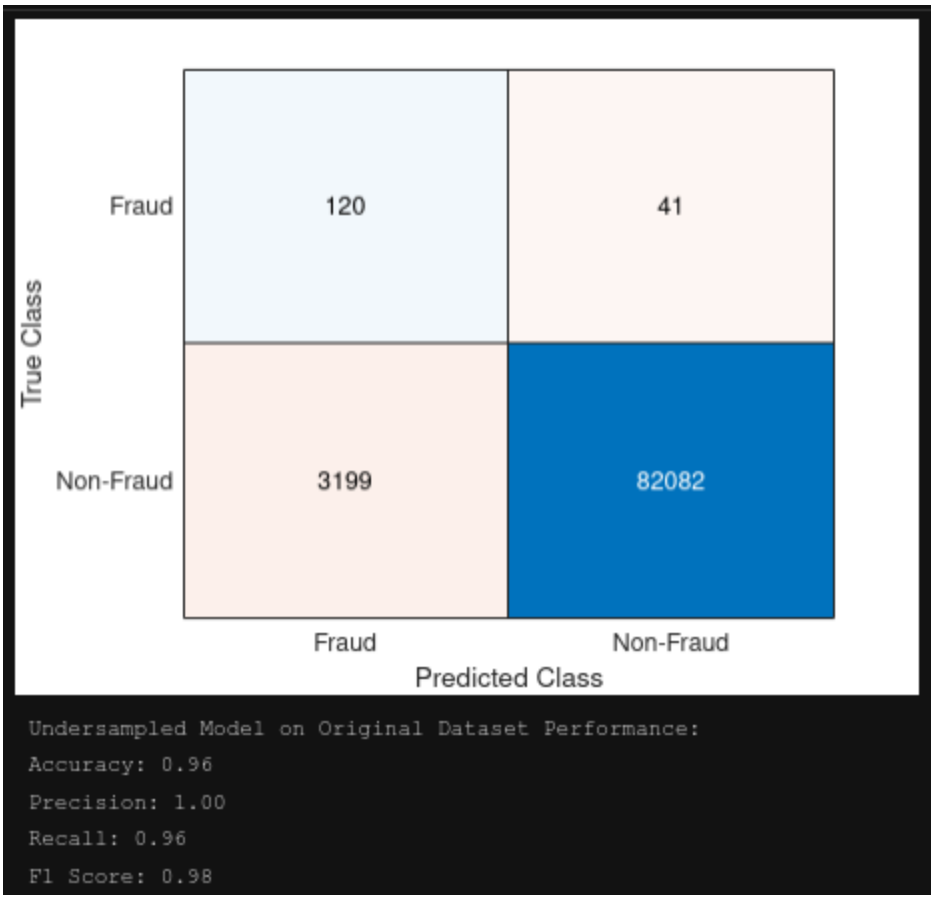

Figure 16: *Performance of LR model on balanced dataset for training*

As can be seen above, the model correctly identified 75% of the fraudulent transactions in the test set, a significant improvement over the previous model's 27% detection rate.

The model's precision obtained a perfect score of 1.00, meaning that all fraudulent transactions were actually fraudulent. Despite the model's improvement in fraud detection, the accuracy of categorizing non-fraudulent transactions fell slightly. In comparison to its earlier almost perfect accuracy on the original unbalanced data, the model only successfully classified 96% of the non-fraud test data. Despite this trade-off, the balanced model's overall performance was superior because it generated considerable gains in detecting fraudulent transactions, which is a main goal in fraud detection scenarios. This balanced Logistic Regression model exhibits its applicability and efficacy in solving credit card fraud detection tasks by correctly categorizing the majority of both fraud and non-fraud transactions.

Next, we trained and tested our model on the under sampled dataset. In comparison to the prior logistic regression models that were tested on the original unbalanced dataset, this model performed worse with an overall accuracy of 0.88 as can be seen in Figure 17.

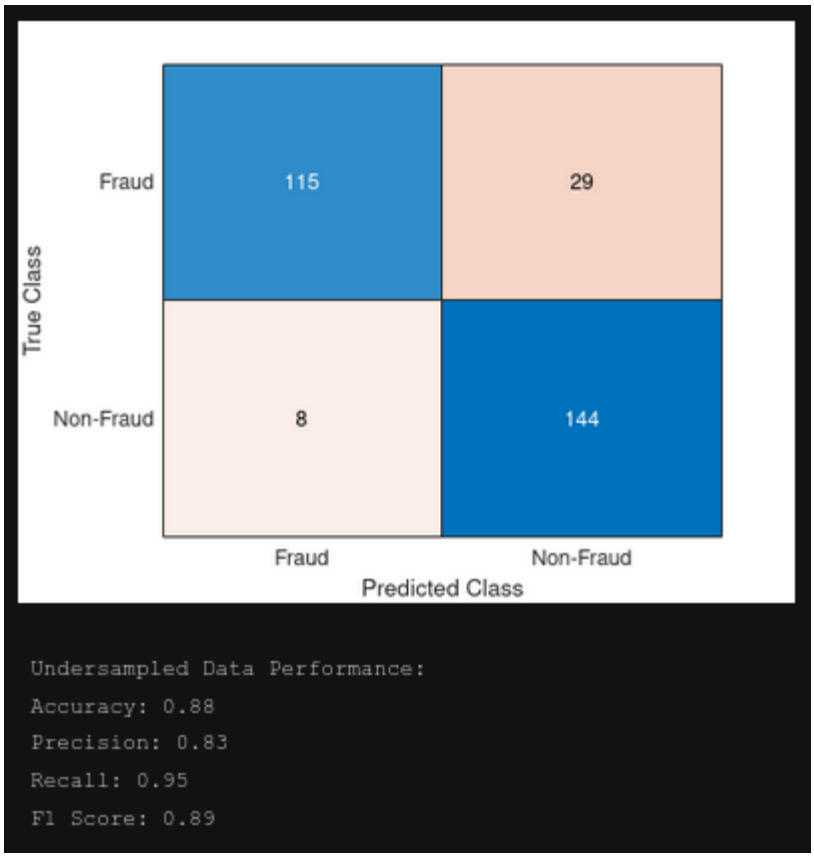

Figure 17: *Performance of LR model on balanced dataset*

However, it displayed a huge improvement in correctly classifying fraudulent transactions, with an accuracy of 80% compared to the model that was trained and tested on the unbalanced datasets' 27% accuracy, and the model that was trained on the original unbalanced dataset and trained on the under sampled dataset's 75% accuracy. In addition, the model maintained a high accuracy of 95% for non-fraudulent transactions, which is slightly lower than the first model's near-perfect performance on non-fraudulent data.

The "AUC" (Area Under the Curve) metric is generated from the "ROC" (Receiver Operating Characteristic) curve and is used in classification jobs to quantify separability. It assesses a model's capacity to differentiate between various classes by aggregating the ROC curve into a single value. The AUC, which ranges from 0.0 to 1.0, reflects the area under the ROC curve. Higher AUC values imply that the model is performing better at predicting 0s as 0s and 1s as 1s. In the case of credit card fraud detection, a higher AUC indicates that the model is better at differentiating between legitimate and fraudulent transactions.

An excellent model has an AUC near to one, indicating a high level of separability and predictive accuracy. A poor model, on the other hand, would have an AUC near to zero, indicating that it performs poorly in distinguishing between the two groups, essentially

forecasting 0s as 1s and 1s as 0s. When the AUC is 0.5, the model is incapable of class separation, implying random guessing. The AUC for the Logistic Regression model trained on the balanced dataset was 0.85, showing a reasonable measure of separability as well as a good performance in detecting fraudulent transactions as can be seen in Figure 18.

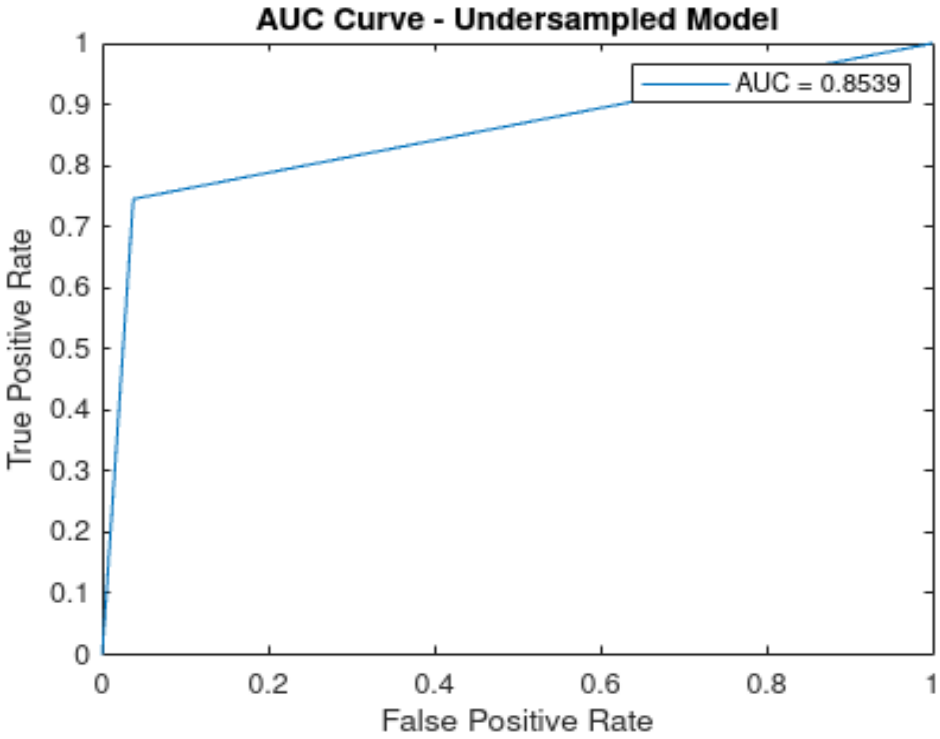

Figure 18: *AUC for LR on balanced dataset*

3.5.2 KNN Model Performance Results & Discussion

When building our KNN (k-nearest neighbors) model, we plotted accuracy versus the number of neighbors (K) to see how both testing and training accuracies changed as K changed. The plot helped us find the "sweet spot", the value of K that produced the best overall accuracy for the model. In our case, we observed that the ideal K value was 5, with the model achieving the highest accuracy as can be seen in Figure 19.

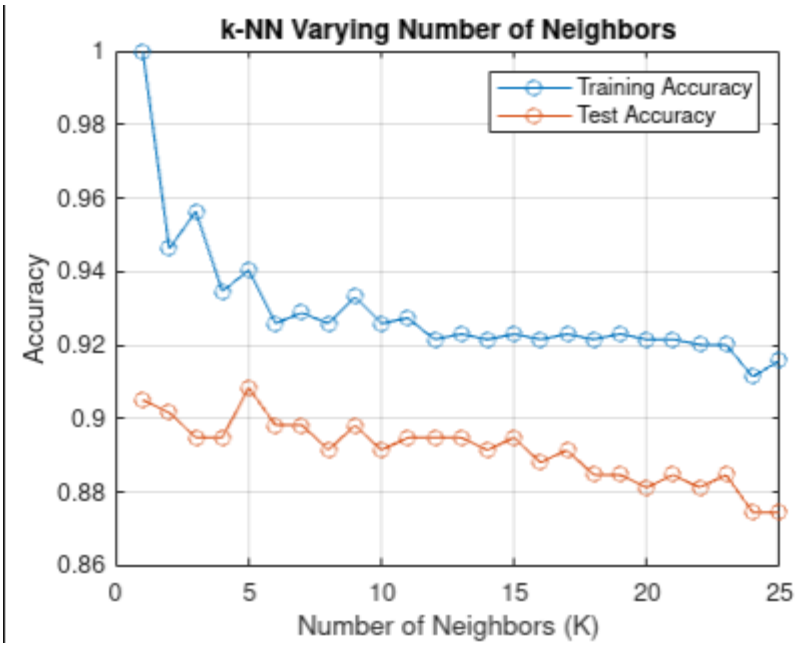

Figure 19: *Accuracy vs K plot*

This means that taking into account the five closest neighbors for each data point in the testing dataset resulted in the highest accurate classification of credit card transactions as fraudulent or non-fraudulent. This careful selection of neighbors increased the performance of our KNN model, making it more successful in recognizing fraudulent transactions while maintaining a decent mix of accuracy and generalization.

When we tested our KNN model on the under sampled test data we got an accuracy of 0.91, suggesting that the model accurately identified 91% of the total transactions as fraudulent or non-fraudulent. The KNN model performed well in correctly detecting non-fraudulent transactions, obtaining an accuracy of 98%, which means that it correctly identified 98% of the non-fraudulent transactions. Also, the model performed well in detecting fraudulent transactions, with an accuracy of 84%, showing that it successfully detected 84% of the fraudulent transactions as can be seen in Figure 20.

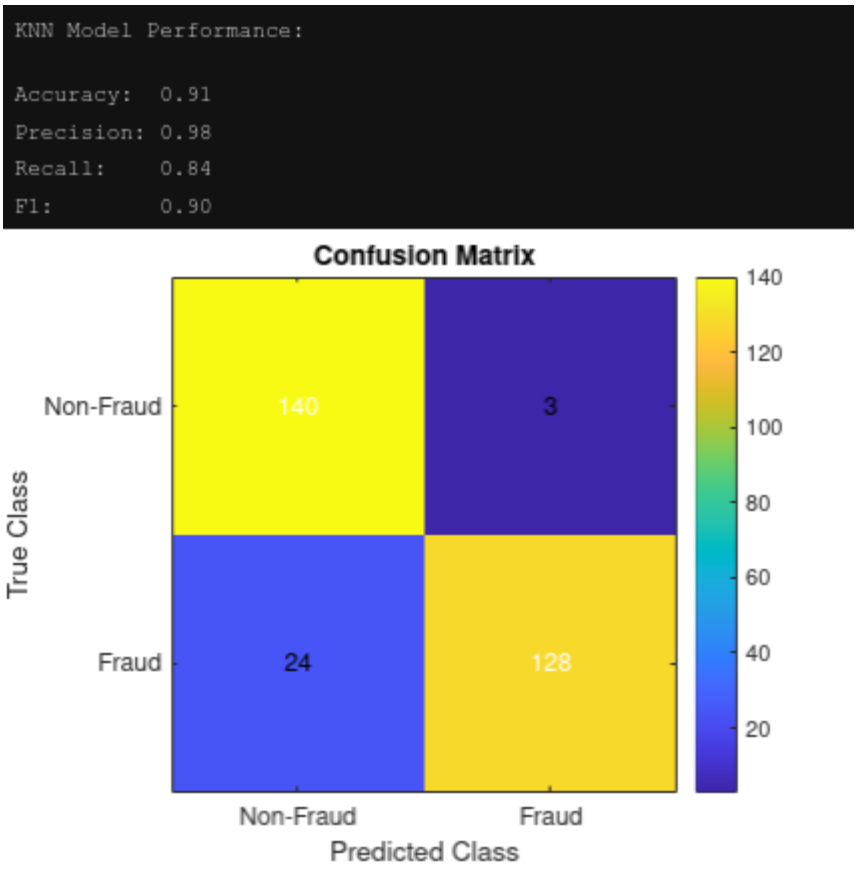

Figure 20: *KNN model performance on under sampled test data*

We also evaluated the precision, recall, and F1 score to acquire a better understanding of the model's performance. The KNN model's precision was 0.98, meaning that 98% of the transactions predicted as fraudulent were in fact fraudulent. The recall, also known as sensitivity or true positive rate, was 0.84, suggesting that the model successfully classified 84% of all fraudulent transactions. Finally, the F1 score, which takes into account both precision and recall, was 0.9, indicating that the model was accurate in categorizing both fraudulent and non-fraudulent transactions.

3.6 Logistic Regression vs KNN Comparison

When comparing the performance of our Logistic Regression (LR) and K-Nearest Neighbors (KNN) models, both trained and tested on the balanced under-sampled dataset, several key metrics that provide valuable insights into the models' respective strengths and weaknesses as can be seen from the table below.

Table 1: *Model Performance Comparisons*

Performance Metric Used	**Logisitc Regression**	**K-Nearest Neighbors**
Accuracy	0.88	0.91
Precision	0.83	0.98
Recall	0.95	0.84
F1 Score	0.89	0.90

We can take the following insights from the performance table:

- The accuracy of the Logistic Regression model was 0.88, while the accuracy of the KNN model was 0.91, indicating that the KNN model had a better overall correct classification rate.
- The KNN model excelled in terms of precision, which evaluates the ability to properly identify positive predictions, with a precision of 0.98, substantially greater than the Logistic Regression model's precision of 0.83. This means that the KNN model was better at detecting real fraudulent transactions.
- However, when it comes to recall, or the true positive rate, the Logistic Regression model outperformed the KNN model with a recall score of 0.95, beating out the KNN model's recall score of 0.84. This suggests that the Logistic Regression model captured a higher share of genuine fraudulent transactions.
- The F1 score, which is a harmonic mean of precision and recall, was 0.90 for the KNN model, which was slightly higher than the F1 score of 0.89 for the Logistic Regression model. The F1 score is a balanced measure of precision and recall, demonstrating that the KNN model successfully identified positive cases while avoiding false negatives.

These comparisons show that the KNN model outperformed the LR model in terms of accuracy and precision, while the LR model outperformed in terms of recall rate. The KNN model excelling at accurately detecting fraudulent transactions and the LR model succeeding at recognizing a higher proportion of genuine fraud cases.

Which model is better for Credit card fraud detection?

If the primary goal is to properly identify fraudulent transactions, the KNN model may be preferable because it has a greater precision (0.98) than the LR model (0.83). With high

precision, the KNN model is superior at minimizing false positives, guaranteeing that the majority of anticipated fraudulent transactions are true fraud occurrences.

The LR model, on the other hand, may be more suited if the primary objective is to capture a higher proportion of actual fraudulent transactions at the expense of more false positives. It outperformed the KNN model in terms of recall (0.95), indicating that the LR model may detect a bigger proportion of total fraudulent transactions.

The final decision of which model to use comes down to another factor: the cost associated with false positives and false negatives. In our case since the cost of missing a fraudulent transaction is significantly higher than the cost of a false positive, then the model with higher recall, the Linear Regression model, would be the preferred option. However, if for some reason the primary objective is to minimize the cases of false positives, the KNN model would be more suitable because of its much higher precision.

3.7 Bias-Variance Tradeoff Analysis, Strengths & Limitations

In the bias-variance tradeoff analysis, the LR model exhibited lower variance compared to the KNN model, indicating more stable predictions. However, the KNN model was fine-tuned to find an optimal number of neighbors (K=5), leading to a "sweet spot" of highest overall accuracy.

Both models had their strengths and limitations. The LR model showed good overall performance but struggled with correctly classifying fraudulent transactions in the imbalanced dataset. The KNN model, while providing promising results on the under sampled data, was computationally expensive on large datasets.

3.8 Further Improvements

To further improve our models, we suggested several potential avenues for future research. Employing advanced data handling techniques, such as oversampling or using synthetic data generation, could help mitigate class imbalance issues. Feature engineering and inclusion of additional transaction-related information might also enhance the models' ability to capture fraud patterns effectively. Ensemble methods like Random Forest or Gradient Boosting could be employed to combine the strengths of different models and improve overall performance.

CHAPTER 4: CONCLUSION

In this study, we explored and compared the performance of two machine learning models, Logistic Regression (LR) and K-Nearest Neighbors (KNN), for credit card fraud detection. Our primary goal was to develop accurate and reliable models capable of distinguishing fraudulent transactions from legitimate ones.

We first trained and evaluated the models on the original unbalanced dataset, and the Logistic Regression model performed admirably, with an accuracy of 0.998. However, due to the large class imbalance, this accuracy was deceptive, and the model struggled to correctly categorize fraudulent transactions. The KNN model, on the other hand, demonstrated more balanced accuracy, but it, too, encountered difficulties with the unbalanced dataset.

We used under-sampling strategies to construct a balanced training dataset for both models to solve the issue of class imbalance. Reevaluation reduced the accuracy of the Logistic Regression model to 0.88, while the KNN model retained a comparatively high accuracy of 0.91. However, the Logistic Regression model outperformed KNN in detecting fraudulent transactions, with a recall of 0.95 on the under-sampled data compared to 0.84 for the KNN.

Our analysis showed the significance of dealing with imbalanced datasets in fraud detection settings, as precision alone can be misleading. Furthermore, the precision and recall measures were critical in determining the models' genuine effectiveness in successfully identifying fraudulent transactions.

In conclusion, the K-Nearest Neighbors (KNN) model, when trained and tested on a balanced under-sampled dataset, emerged as the better option for credit card fraud detection. Its higher recall and relatively balanced accuracy demonstrated its capability to accurately identify fraudulent transactions, making it a valuable tool in mitigating financial losses due to fraud. However, there is still room for further refinement and exploration of other machine learning approaches to create even more robust and accurate fraud detection systems for future contributions to the field.

REFERENCES

1. Muncaster, P. (2023) *Card 'ID theft' fraud doubles in 2022*, *Infosecurity Magazine*. Available at: https://www.infosecurity-magazine.com/news/card-id-theft-fraud-doubles-in-2022/ (Accessed: 21 July 2023).
2. Marcelotc. (2020). Creditcard Fraud - Logistic Regression example. *Kaggle*. https://www.kaggle.com/code/marcelotc/creditcard-fraud-logistic-regression-example#Credit-card-transaction-fraud-detection---Logistic-Regression-example

3. Marcelotc. (2020). Creditcard Fraud - Logistic Regression example. *Kaggle*. https://www.kaggle.com/code/marcelotc/creditcard-fraud-logistic-regression-example#Credit-card-transaction-fraud-detection---Logistic-Regression-example
4. Seeja, K. R., and Zareapoor, M., (2014). FraudMiner: A Novel Credit Card Fraud Detection Model Based on Frequent Itemset Mining, The Scientific World Journal, Hindawi Publishing Corporation, Volume 2014, Article ID 252797, pp. 1 – 10, http://dx.doi.org/10.1155/2014/252797
5. A. Mishra, C. Ghorpade, "Credit Card Fraud Detection on the Skewed Data Using Various Classification and Ensemble Techniques" 2018 IEEE International Students' Conference on Electrical, Electronics and Computer Science (SCEECS) pp. 1-5. IEEE.
6. S. V. S. S. Lakshmi, S. D. Kavilla "Machine Learning For Credit Card Fraud Detection System", unpublished
7. N. Malini, Dr. M. Pushpa, "Analysis on Credit Card Fraud Identification Techniques based on KNN and Outlier Detection", Advances in Electrical, Electronics, Information, Communication and Bio-Informatics (AEEICB), 2017 Third International Conference on pp. 255-258. IEEE.
8. Mrs. C. Navamani, M. Phil, S. Krishnan, "Credit Card Nearest Neighbor Based Outlier Detection Techniques"
9. J. O. Awoyemi, A. O. Adentumbi, S. A. Oluwadare, "Credit card fraud detection using Machine Learning Techniques: A Comparative Analysis", Computing Networking and Informatics (ICCNI), 2017 International Conference on pp. 1-9. IEEE.
10. Learning – Towards Data Science. [online] Available at: https://towardsdatascience.com/deep-learning-vs-classical-machine-earning-9a42c6d48aa [Accessed 19 Jan. 2019].
11. Awoyemi, J. O., Adetunmbi, A. O., & Oluwadare, S. A. (2017). Credit card fraud detection using machine learning techniques: A comparative analysis. *2017 International Conference on Computing Networking and Informatics (ICCNI)*. https://doi.org/10.1109/iccni.2017.8123782
12. Bahnsen, A. C., Stojanovic, A., Aouada, D., & Ottersten, B. (2013). Cost sensitive credit card fraud detection using Bayes minimum risk. In Machine Learning and Applications (ICMLA), 2013 12th International Conference on (Vol. 1, pp. 333-338). IEEE.
13. Awoyemi, J. O., Adetunmbi, A. O., & Oluwadare, S. A. (2017). Credit card fraud detection using machine learning techniques: A comparative analysis. *2017 International Conference on Computing Networking and Informatics (ICCNI)*. https://doi.org/10.1109/iccni.2017.8123782
14. Stolfo, S., Fan, D. W., Lee, W., Prodromidis, A., & Chan, P. (1997). Credit card fraud detection using meta-learning: Issues and initial results. In AAAI-97 Workshop on Fraud Detection and Risk Management.
15. Bahnsen, A. C., Stojanovic, A., Aouada, D., & Ottersten, B. (2013). Cost sensitive credit card fraud detection using Bayes minimum risk. In Machine Learning and Applications (ICMLA), 2013 12th International Conference on (Vol. 1, pp. 333-338). IEEE.
16. Seeja, K. R., and Zareapoor, M., (2014). FraudMiner: A Novel Credit Card Fraud Detection Model Based on Frequent Itemset Mining, The Scientific World Journal, Hindawi Publishing Corporation, Volume 2014, Article ID 252797, pp. 1 – 10, http://dx.doi.org/10.1155/2014/252797

17. Varmedja, D., Karanovic, M., Sladojevic, S., Arsenovic, M., & Anderla, A. (2019). Credit Card Fraud Detection - machine learning methods. *2019 18th International Symposium INFOTEH-JAHORINA (INFOTEH)*. https://doi.org/10.1109/infoteh.2019.8717766
18. Marcelotc. (2020, March 4). *Creditcard fraud - logistic regression example*. Kaggle. https://www.kaggle.com/code/marcelotc/creditcard-fraud-logistic-regression-example#Credit-card-transaction-fraud-detection---Logistic-Regression-example