Using ML Techniques to Detect Credit Card Fraud - Midterm Report

1st Ryan Peruski
Department of EECS
University of Tennessee, Knoxville
Knoxville, TN
yhg461@vols.utk.edu

2nd Jake Marlow
Department of EECS
University of Tennessee, Knoxville
Knoxville, TN
amarlow6@vols.utk.edu

3rd Sam Lavey
Department of EECS
University of Tennessee, Knoxville
Knoxville, TN
slavey@vols.utk.edu

***Abstract*—In this midterm report, we explore the application of various machine learning techniques to detect credit card fraud in electronic payment systems. Our goal is to build models capable of identifying fraudulent transactions in real-time, ensuring the safety of consumers and financial institutions. Using a dataset containing transaction data, we apply methods like K-Nearest Neighbors (KNN), Support Vector Machines (SVM), Ridge Regularization, and Logistic Regression. Each model is evaluated using multiple performance metrics, including R-Squared Score, F1-Score, ROC-AUC Score, and accuracy, to identify the most effective approach. Preliminary results show that the KNN algorithm performs best with an accuracy of 0.95 and a ROC-AUC score of 0.91. Future work will focus on integrating categorical variables and employing ensemble methods to further enhance model performance.**

***Index Terms*—machine learning, svm, knn, ridge regularization, ensemble**

I. Introduction

Can machine learning be used to make the electronic payments safer? In this project, machine learning models are utilized to detect credit card fraud. Over the past few years, payment fraud has become a major issue that has cost consumers and institutions over 10 billion dollars over the past fiscal year [4]. This trend has grown year-over-year for the past 5 years. As technology becomes more advanced, new techniques have emerged that expose millions of consumers to greater risk of being defrauded. Various social media trends have highlighted the prevalence of card skimmers and other deceptive practices. It is clear that as the threat model changes and new threats emerge, a comprehensive solution is needed to mitigate the damage and prevent fraud from occurring. Developing our model is made possible through the access to historical data containing both fraudulent and non-fraudulent cases. With the right training methods, a model capable of predicting credit card fraud is made possible. Determining which training method is to be used will be done experimentally. In the end, our model can be used to serve as a fraud detection and prevention service. With this, we can detect when a transaction is fraud in real time, meaning that our model will catch the transaction before it goes through; this will, ultimately, make bank transactions safer for consumers.

II. Dataset

A. Overview

The dataset utilized in this project is from Kaggle and has an MIT license [3]. Some of its features include transaction amount, transaction description, location, and merchant-specific data. Data collected from the merchant includes the name, geographic location, type, and address. The last column tells us whether the transaction is fraudulent or not. For our preliminary experiment, we ended up taking all of the non-categorical data to use as independent variables, or ”parameters”.

B. Data Cleaning

Making sure the data is cleaned is essential to prevent bugs, mitigate over-fitting or under-fitting, and lower irreducible error. In order to clean the code, we got rid of all rows that were missing data for any of the features mentioned, and made sure that the ”is_fraud” column only had 1’s or 0’s to make this a binary classification problem. By setting this up as a binary classification problem, we can choose to use models that would be more efficient or suitable to the dataset.

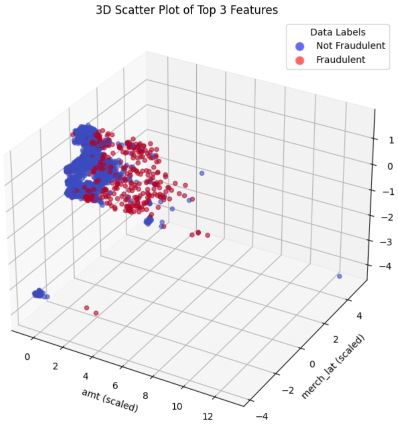

Fig. 1. Top 3 Features Plot [2]

C. *Important Features*

The 3-D graph (Figure 1) shows ”clusters” that should be formed by a model with an optimal setting. To do this, we identified the three most influential features on the dependent variable for the graph. The three most important features were chosen via a simple Random Forest Classifier. This graph displays the optimal way a model should ”cluster” the data appropriately. As can be seen on the graph, it seems that the transaction amount (amt) has the most effect on whether or not a transaction is fraudulent.

III. Baseline Solution

A. *Scores Used*

There are different scores that we ended up testing on, including an R-squared score, an F1-score, a ROC-AUC score, and a simple Accuracy Score.

An R-squared score is a return value generated based on how well the model fits the data, and it measures the proportion of variance in the dependent variable that can be described by the independent variable. Higher R-squared scores mean a better model. The best possible R-squared score is 1, which would mean that all variance is accounted for.

An F1-score is a weighted average of precision and recall, with precision being the proportion of positive classifications that are actually positive and recall being the proportion of relevant data points that are identified. Higher F1 scores mean a better model. An F1-score of 1 is the best F1-score.

A ROC-AUC score is measured using the Area Under the Curve in a ROC curve, which is plotted precision over recall. The best ROC-AUC score is 1, which is when the ROC curve shows a perfect 1 by 1 square, while a score of 0.5, showing a right triangle, would indicate random guessing.

Finally, the accuracy score is simply the number of correct predictions over all predictions, so an accuracy of 1.0 or 100 percent means a better model. Accuracy percentages are returned as a sliding scale from 0-1.

With all of these scores, achieving a score of 1 is best; however, this will not happen in real situations. Thus it is necessary to establish a threshold by which we can judge a ”good” model from a ”bad” one. It is our determination that having scores of around 0.9 is ideal, and if this can be achieved, then this project can be considered a success.

B. *Previous Solution*

For our preliminary experiment, we looked into an RIT paper that had a very similar goal [1]. From this paper, we observed their table of accuracies, giving an idea of what methods to use. From here, we decided on using K Nearest Neighbors (KNN), Support Vector Machine (SVM), and Logistic Regression models, as well as a new model, Ridge (L2) Regression, as it may be able to suppress independent variables we do not use. From here, we had some ideas of what we should be using for our models.

C. *KNN*

For the first attempt at building a model, we tried the K-Nearest Neighbors (KNN) approach. In this approach, data points are classified by the state of their K nearest neighbors. After scaling and fitting the data, we ran a loop to determine which K was the best to use for the model. This was judged by choosing the K with the highest R-Squared score. Our results showed that 10 was the best K to use and resulted in an R-Squared score of 0.60, ROC-AUC score of 0.91, F1-score of 0.79, and an Accuracy score of 0.95 [2].

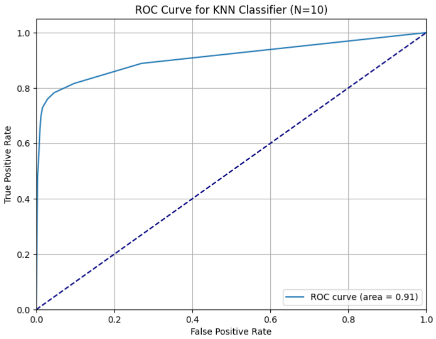

Fig. 2. ROC curve for K-Nearest Neighbors with K=10 [2]

D. *SVM*

The second method we tried utilized what is known as the Support Vector Machine (SVM) method, or support vector machine method. Since its primary goal is to find a hyperplane, or in this case a line, in between two classes, its most common use case is binary classification problems. After fitting the model to the training data and using the test data to make predictions, it was discovered that the SVM method had an R-Squared score of 0.31, ROC-AUC score of 0.90, F1 score of 0.62, and Accuracy score of 0.92 [2].

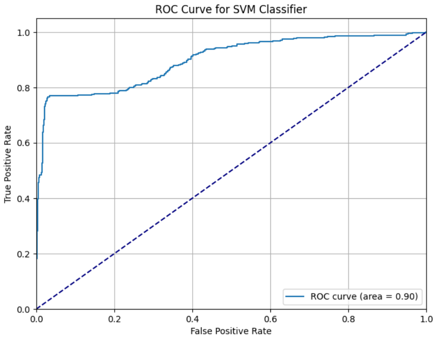

Fig. 3. ROC curve for SVM Model [2]

E. *Ridge Regularization*

One method we ended up using was a simple Ridge Regression model. We wanted to test how Regularization would affect results, and, from Dr. Santos' Lecture, we determined that L2 (Ridge) Regularization was the best method to use for this. As usual, we fitted a Ridge Regressor to the dataset and obtained an R-squared score of 0.52, ROC-AUC score of 0.84, F1 score of 0.61, and Accuracy score of 0.92 [2].

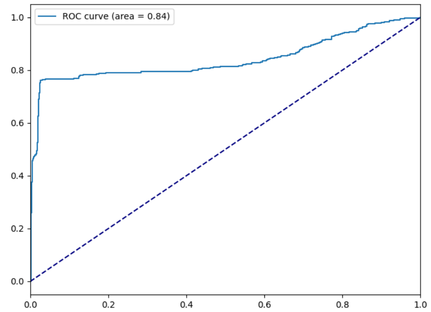

Fig. 4. ROC Curve for Ridge Regularization [2]

F. Logistic Regression

The final method we ended up using was the "default" for a binary classifier: Logistic Regression. As usual, we fitted the model to the dataset and obtained an R-squared score of 0.52, ROC-AUC score of 0.73, F1 Score of 0.61, and Accuracy score of 0.92 [2].

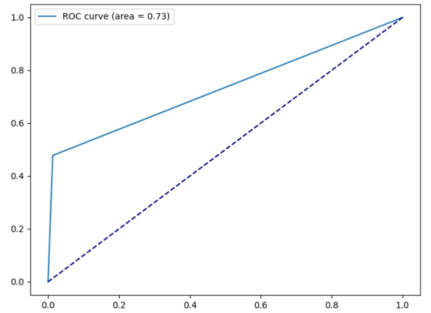

Fig. 5. ROC Curve for Logistic Regression [2]

Model	R-Squared	ROC-AUC	F1-Score	Accuracy
KNN (K=10)	0.60	0.91	0.79	0.95
SVM	0.31	0.90	0.62	0.92
Ridge	0.52	0.84	0.61	0.92
Logistic Reg.	0.52	0.73	0.61	0.92

TABLE I
COMPARISON OF MODEL PERFORMANCE [2]

G. Discussion

Each of these models performed in a fairly similar manner to each other, especially when comparing the Accuracy and F1 scores. However, the KNN and SVM models have significantly better ROC-AUC scores, meaning these models are overall better at distinguishing the classes of "fraud" and "not fraud" from one another. However, the SVM model in particular had a worse R-squared score than the others, which implies a high variance in the model, and thus, perhaps, overfitting. The reason why we used multiple performance metrics rather than just an R-squared score is that it is now obvious that the R-Squared Score does not tell the whole story of model performance. As one can see from viewing Table 1, each model has strengths and weaknesses that are represented in the metrics returned by testing.

Overall, the KNN model with an N-value of 10 provided the best results. The KNN algorithm is less sensitive than other models to outliers, which may explain its success. If credit card fraud generally occurs within certain parameters with some extreme outliers, this could explain the success of this algorithm. A larger, more diverse dataset may yield a different result.

IV. PROPOSED EXTENSION

A. Using More of the Dataset

One thing that we neglected to do was use all of the variables in the dataset. In the future, we will use some kind of encoding for categorical variables. This way, every single part of the dataset is being used.

B. Ensembling

In the future, since we have tried out and gotten similar results in each classifier, we want to Ensemble each of these models to try and extract the "best of all worlds". We are planning on using a simple majority voting ensemble technique to start, and, if we do not get desired results, we can switch to different methods. For example, since we know that some models have better scores than others, we could apply weights to better-performing models. We would like to conduct a test of how these models perform individually versus ensembled.

V. CONCLUSION

Although the KNN model performed the best from our current methods of training and testing the dataset on our models, we do not believe that KNN is the end-all, be-all for this dataset. The primary reason for this has to do with not including some of the categorical data in training, and this categorical data could be the difference between KNN being the best and KNN being the worst. Therefore, we will stick to our current future plans of using categorical data and ensembling each of these models.

VI. DISTRIBUTION OF WORK

The contributions to the project were divided as follows:

Team Member	Contribution
Ryan Peruski	• Primary Code Leader • Data Preprocessing and cleaning • Implementation of KNN, Ridge Regularization, and Logistic Regression model • Drafting the Midterm Report and Code Documentation
Jake Marlow	• Primary Data Visualization Leader • Implementation of SVM model • Visualization of results and performance metrics • Final review of content
Sam Lavey	• Primary Recorder for this report and results • Literature review and background research • Code cleanup and final review • Finalizing report structure and edits

TABLE II
DISTRIBUTION OF WORK AMONG TEAM MEMBERS

ACKNOWLEDGMENT

We would like to express our sincere gratitude to the Department of Electrical Engineering and Computer Science at the University of Tennessee, Knoxville, for their continuous support and resources throughout this project. A special thanks to Dr. Hector Santos for his invaluable guidance, mentorship, and encouragement, which greatly contributed to the success of this work.

REFERENCES

[1] Cardillo, J. (2020). Wearable computing: A system that adapts based on the user's location (Master's thesis). Rochester Institute of Technology. https://repository.rit.edu/cgi/viewcontent.cgi?article=12455&context=theses

[2] Peruski, R., Marlow, J, Lavey, S. (2024). Intro to ML Project (Version 1.0) [Computer software]. GitHub. https://github.com/Silverasdf/introtomlproject

[3] Roychoudhury, N. (2021). *Credit card fraud data* [Data set]. Kaggle. https://www.kaggle.com/datasets/neharoychoudhury/credit-card-fraud-data/data

[4] Federal Trade Commission. (2024, February 8). *Nationwide fraud losses top $10 billion in 2023 as FTC steps up efforts to protect the public*. https://www.ftc.gov/news-events/news/press-releases/2024/02/nationwide-fraud-losses-top-10-billion-2023-ftc-steps-efforts-protect-public

[5] Santos, H. (2024). Lecture 10: Bias, Variance, and Regularization [Lecture]. University of Tennessee, Knoxville.