Final Report
Evaluation of Credit Card Fraud Detection

CS 375 - 001 Final Group Project
Adonis Pujols, ap2933
Feng Chen, fc43
Neo Maralit, nm775
Siva Anand Sivakumar, ss546

Keywords: Exploratory Data Analysis, Bagging, K-Nearest Neighbor, Convolutional Neural Network, Deep Learning, Model Ensembling, Data Pre-Processing, XgBoost, Credit Card Fraud

Abstract: With increased digital financial transactions, credit fraud has surged, causing significant monetary losses. Our aim for this project is to tackle this issue by developing a machine learning-based solution that can properly handle the accurate detection of fraudulent credit card activities. Utilizing a given dataset, we explored various models including, K-Nearest Neighbor (KNN), Convolutional Neural Networks (CNN), and an ensemble approach combining Bagging with XGBoost. This report underscores the potential of advanced machine learning techniques in enhancing the security of financial transactions, offering promising ways to mitigate the impact of credit card fraud.

Dataset: https://www.kaggle.com/datasets/nelgiriyewithana/credit-card-fraud-detection-dataset-2023

Links: Code:
https://drive.google.com/file/d/1SR8Kec1Nx8J7r8SHKMilxP5gGIF5GtGk/view?usp=sharing

https://colab.research.google.com/drive/1OIHmxfBkNlXrrUZI4gxVu5a3cjhE4IJ4?usp=sharing

https://colab.research.google.com/drive/1FM2pYHNeJzbNS07_YJrHok6-MlvV4i89?authuser=1

I. INTRODUCTION

In this report, we will explain the tasks we performed for our project in different sections of this report.

Related work: In this section, we will discuss the approaches that were taken by different people who have conducted experiments using this dataset. By knowing what others have done, we can try various different approaches that they haven't done before and uncover new avenues for exploration.

Exploratory Data Analysis: In this section, we will discuss the data preprocessing and preparation steps that we performed before applying the dataset to any model. This step is important because you gain insight about the characteristics of the dataset, patterns, and any potential issue with the dataset.

Methods: In this section, we discussed the models that we apply to this dataset. Things that were discussed include the model's overall performance and the detailed steps involved in the application.

Discussion: In this section, we discussed all the tasks that have been performed in our project. Discussing the result, process involved and reflections about our findings.

Conclusion: In this section, we summarized all our findings and discussed any improvement that we can make in the future and potential limitations in our project.

II. RELATED WORKS

Before we began to dive into what we did for our project, we discussed the different approaches that were taken by others and Looked at what they did compared to ours. Upon looking at these approaches, we realized that, while approaches from one individual may differ from another, there are several techniques They used that are similar. All the projects start with exploratory data analysis that will check what dataset they are working with, then train and test the model after they figure out what dataset they are dealing with and make adjustments for it if needed. In milestone 3, we picked several approaches from others to view. In this report, we will mainly focus on one of the approaches or analysis that are conducted among them. In the notebook [1.1], the author could achieve nearly 100 percent accuracy with the approach of a decision tree, random forest, and XGBRF classifier. The author first did some exploratory data analysis, like checking if the dataset has any duplicate rows or not, looking for missing or null values, observing the correlation between features, determining whether the dataset is normally distributed, etc. Our work differs from the previous approach because of the following reasons: For our neural network model, instead of using a classical stochastic gradient descent, RMSprop or AdaGrad, we chose Adam (Adaptive Moment Estimation). This choice gave efficient convergence while requiring less hyperparameter testing, since Adam uses “momentum” and avoids training decay around local extrema. [2.1]. In contrast to some more naive implementations we saw, we chose to use a validation dataset so that our model's parameters are not overfitting to the test set. So we expect better results from this model in similar datasets than those. [2.2]

III. EXPLORATORY DATA ANALYSIS

Upon looking into all the variables in the data set, we observe that “class” is the target variable of our dataset. The data points in this dataset are either identified as fraudulent (class 0) or non-fraudulent (class 1). All the other features, like v1 to v28, are the features that we will use to define each data point and whether it belongs to a certain class. The data type for all the columns is either a float or an integer, which is good for our model. If the dataset contains categorical values, then a technique like one-hot encoding will be performed to convert the categorical feature to a numerical value. One thing that we want to check for a dataset is whether it is balanced or not. An imbalanced dataset is a problem for us, as an imbalanced dataset will affect the accuracy of the model when classifying a specific data point. Imagine that 90% of the dataset belongs to class 0, and the other 10% belongs to class 1. The model in this case will have an easier time classifying a data point to be class 0 than a data point to be class 1, as the model doesn’t have enough information to be trained with. Based on our observation using a bar graph, the dataset is balanced as the frequency between the classes is close to or identical to each other. Thus, there is a lesser chance of the data being overfitted or misclassified, and we can say that the dataset is a balanced dataset. When going through the steps of cleaning the dataset, we found that the dataset doesn’t contain duplicate rows or any null values. This means that the dataset is very well organized and clean. We obtained this information by counting the total number of times a “NaN” appeared in each column, and the result came out to be 0 for each column. For the duplicated row, it is the same concept, except you look at the whole dataset instead. For a sizable dataset like this one, one of the cases that we considered is if any data point is considered an outlier according to certain criteria for a specific column of the dataset. Upon going through the dataset, the result shows that no column contains outliers. This result proved the point of this dataset being a very clean dataset. We also implemented a correlation matrix to check the correlation between features. Mainly, we want to look at the correlation between all the features and the target variable. We observe that one feature named “ID” has a strong correlation with the target variable, as the correlation coefficient is 0.86. With a strong correlation like this, it is possible that this feature could lead to overfitting of the model, and this feature doesn’t give the model extra information that can help the model make predictions. Therefore, this feature is not considered in our project. With the correlation matrix, we also observed that V17 and V16 are strongly correlated with a correlation coefficient of 0.85, and V17 and V18 are also strongly correlated with the same correlation coefficient. Amount has no correlation according to the coefficient matrix; however, a relationship between the feature and the target variable doesn’t have to be linear, which is what the correlation matrix is based on. The feature amount provides important information in classifying whether a transaction is fraudulent or not, since if the transaction amount is incorrect, then the transaction has a high chance of being classified as fraudulent. There are many features that have a decently strong correlation with the target variable, but all of them are considered useful for this project in helping classify the data point.

IV. METHODS

This section can describe any models that you develop for solving the problem.

Group 1 KNN:

For our group 1 model, we chose K Nearest Neighbors. From the beginning, we know that optimizing for the correct k hyperparameter would be critical. A low k can lead to a model that doesn’t generalize well for future data points, leading to overfitting. On the other hand, a larger k can lead to a model that underfits. As a result, we set up an experiment to choose the optimal k parameter by training the model with k from 1 to 39. For each value of k, we stored the trained model itself so that we could access it later. Furthermore, we also recorded the accuracy, precision, recall, f1 score, and the ROC AUC score for each k in lists. After recording the models and metrics, we plotted the metrics to see how the increasing values of k affected the metrics. In our evaluation of the different k values, we decided to focus on the precision and recall metrics. We decided on these metrics because there are two main areas of concern with credit card fraud: catching as many fraudulent transactions as possible (optimizing for recall will enable this) and minimizing incorrectly labeling non-fraudulent transactions as fraudulent (optimizing for precision will enable this).

Group 2 Neural Network:

We created a fully connected neural network using Keras on top of Tensorflow. We defined early stopping as no longer improving when min_delta is 1e-2 for at least 2 epochs. Given the relatively low amount of features in our dataset, the neural network was simply constructed using 4 layers. A 64 neuron ‘ReLU’ activation input layer (size chosen to fit the entire feature set), another 32 and 16 neuron ‘ReLU’ activation layers, with a final 1 neuron output “sigmoid” layer. We measured loss using binary cross entropy, as this is a simple classification problem. We trained the model for max 10 epochs (early stopping occurred at epoch 4, anyway), batch size of 512 (found after experimentation), and validation dataset of 20% of the training data. Our final model had 5415 parameters with a total size of 21.15KB. In addition, we went one step beyond to create a rudimentary structure of a potential convolutional neural network approach to this problem. Future fraud detection will require use of CNN in image or text processing when the entire product image and/or description is necessary to identify fraud. Consider the case where a thief buys posters off the credit card of a victim that also buys posters, but by analyzing the image of the thief’s purchase, we determine it is unlikely to have been bought by the victim [2.4]. A Fully Connected Deep Neural Network may not be efficient at handling the necessary number of parameters. So we tested a CNN approach. Our CNN is made of a 1D convolution and max pooling layer, with filter size 5, kernel size 10, and ‘ReLU’ activation, and strides equals 2. It is then flattened and added to a dense 32 neuron hidden layer with a 1 neuron Sigmoid activation output classification layer. It also uses the Adam optimizer and a validation dataset of 20% of the training set.

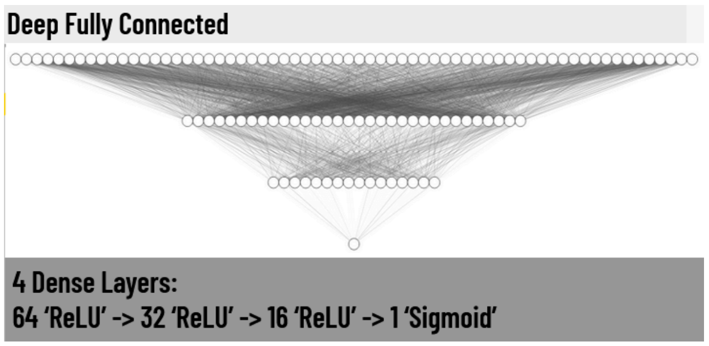

Fig. 4.2.1 Model of Deep Fully Connected Network

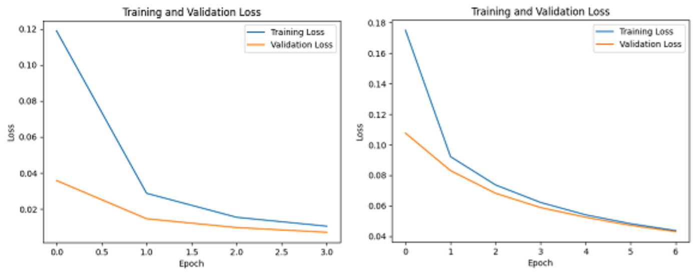

Fig. 4.2.1 Training and Validation Loss of Fully Connected (left) vs CNN Model (right)

Group 3 Bagging:

For the group 3 model, we chose the bagging ensemble method. We utilized the XgBoost framework in order to trivially model the bagging process. We first defined the XGB Classifier [4.3.1] to run using 0.8 as the subsample and 100 n_estimators. Then using the bagging classifier method we used the XGB Classification method as the estimator and defined n_estimators as 10 which becomes the 10 parallel trees for the bagging process. Once these parameters were defined, we could run K-Fold cross validation. Initially, 5 splits were tested, having an accuracy of 0.99 across all 5 splits. Further testing on 10 splits showed the same results, concluding with an average cross validation (CV) score of 0.995 on 10 splits. This reinforces the hypothesis that the data being used is cleaned and highly efficient to use with the model, and is not a result of overfitting. Furthermore, we tested the use of Principal Component Analysis (PCA) on the model to test further for any overfitting of the model on the test data. Initializing the PCA from scikit-learn decomposition [4.3.2] using n_components = 0.85 we were able to transform the data to preserve 85% of its variance from the original data. This is done to both the test values and the train values and resulted in a continued accuracy of 0.99. This means that even while trained on a variance of the original dataset, the model can continue to perform well on both training and testing. Further analysis of the model can be seen in the confusion matrix and classification report. The confusion matrix saw 15 false negatives, 52961 true negatives, 152 false positives, and 56598 true positives. The ratio of these values are good for the total amount of data we used. Likewise, the precision, recall, and f1-score all returned values of 1 meaning the model performs extremely well despite the more simplified model through PCA. Disregarding PCA, the model demonstrates higher performance on the given data, retaining the precision, recall, and f1-score of 1. Moreover, it shows a reduction to 27 false positives from 152 and 0 false negatives from 15.

V. DISCUSSIONS

In our analysis of the KNN models, the highest recall (0.999) and precision (0.999) were found to be k = 2. Increasing values of k led to the metrics decreasing but was still very good with accuracy = 0.9928, precision = 0.9929, recall = 0.993, f1 score = 0.9928. The ROC AUC scores increased, but the range was less than 0.001. This was also reflected for the other metrics: the variation in range between our lowest and highest metrics was less than 0.01. Overall, the KNN model was highly effective.

For the deep, fully connected neural network, it took 23.21 seconds of training time on Google Collab's CPU before stopping at 4 epochs to achieve 99% accuracy. Our CNN model took 29.61 seconds to train on Collab's CPU before stopping at seven epochs to achieve 98% accuracy. As we can see, the deep neural network achieved higher accuracy, but at the cost of over 5x the memory footprint. If the features of this dataset were far more numerous, and if we could get the image or description of the products or services purchased on the credit cards, then a CNN model would be far more efficient and accurate at detecting fraudulent card use. The next step for our project then is to choose a dataset with over 5 million or 5 billion transactions, with an anonymization of data that does not destroy crucial data.

Examining the bagging ensemble method, we note the following trends and insights in our analysis. The model showed high accuracy despite being tested through K-Fold CV and PCA. Retaining a 99% accuracy while being tested on 10 splits and 85% data variance, respectively. Without the use of PCA, the split of 0.2 training and 0.8 testing performs even better, giving 0 false negatives. For this dataset, this model performed the best out of group 1 and 2 most likely because of its simple technique and ensemble methodology. It is important to consider that the dataset is already cleaned and processed before our further processing, which gives the high accuracies from the model. Furthermore, if using the same train sample on a different test dataset, bagging with the use of PCA may see improvements over the former.

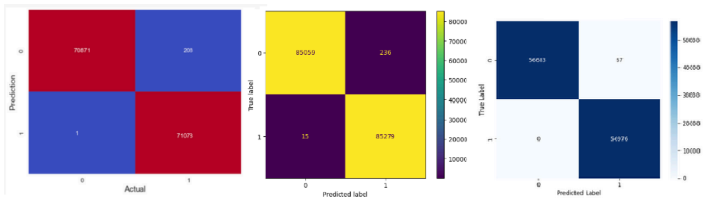

Fig. 5.1 Confusion Matrices of KNN, Neural Network, Bagging

Table 5.1: FP and FN Rates for KNN, Neural Network, Bagging

	FP (%)	FN (%)
KNN	**0.293**	**0.001**
Neural Network	**0.277**	**0.018**
Bagging	**0.048**	**0.000**

VI. CONCLUSION

Our conclusions are based on recall and precision metrics, which are related to FN and FP respectively. Although the KNN resulted in very low percent of FN (which means that almost all fraudulent transactions were labeled as fraudulent) it had the highest FP (which means that more of non-fraudulent data was labeled as fraudulent) so both the neural network and bagging approach are overall better in this sense**.** Our deep fully connected neural network model succeeded in achieving high accuracy with low false positive or negative rates along with fast convergence times using a proper validation set and Adam optimizer. We also left a flexible framework that will adapt to larger datasets with more features. As we reflect on our findings, we recognize the dataset exhibits exceptional quality, which is uncommon. When a dataset exhibits exceptional quality, it leads to misjudgment when evaluating models performance. So moving forward, something we can improve upon is selecting a dataset that closely resembles a real-life scenario rather than a perfect scenario. The bagging ensemble method had the best results with the lowest FP and FN, which makes it the top approach considering our critters of high precision and recall. Given these insights, introducing Principal Component Analysis (PCA) in conjunction with bagging may further enhance the model's efficiency, especially on rougher datasets where feature reduction can play a crucial role in improving model performance.

REFERENCES

[1.1] "Credit Card Fraud Detection with 100% Accuracy," *kaggle.com*. https://www.kaggle.com/code/anmolarora15/credit-card-fraud-detection-with-100-accuracy#Conclusion (accessed Apr. 23, 2024).

[4.2.1] Jiang, L. (2020, September 21). *A visual explanation of gradient descent methods (momentum, AdaGrad, RMSProp, adam)*. Medium. https://towardsdatascience.com/a-visual-explanation-of-gradient-descent-methods-mom entum-adagrad-rmsprop-adam-f898b102325c

[4.2.2] https://www.kaggle.com/code/mamaliesmaeili/credit-card-fraud-modeling-99-99-accuracy

[4.2.3] https://www.kaggle.com/code/zayedupal/credit-card-fraud-detection-using-deep-learning

[4.2.4] M. L. Gambo, A. Zainal and M. N. Kassim, "A Convolutional Neural Network Model for Credit Card Fraud Detection," *2022 International Conference on Data Science and Its Applications (ICoDSA)*, Bandung, Indonesia, 2022, pp. 198-202, doi: 10.1109/ICoDSA55874.2022.9862930.

[4.3.1] https://xgboost.readthedocs.io/en/stable/tutorials/categorical.html

[4.3.2] https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html

VII. WORK ALLOCATION

Objective: To share the workload between us as close to 25% as possible, while harnessing each other's skills and strengths.

Tasks:

1. Research dataset and possible models to pursue (record sources used for report)
   a. Feng and Adonis
2. Explore/Examine dataset and identify perform preliminary analysis
   a. Siva and Neo
3. Discuss on project direction based on tasks 1 & 2 (decide on how our project will be different)
   a. Whole team
4. Perform Feature Engineering and Handle missing data
   a. Feng and Neo
5. Decide on best models (afterward meet with whole team to discuss)
   a. Siva and Adonis
6. Encode categorical data, normalize/standardize features as needed, prep data for models
   a. Feng and Neo
7. Train models and record results
   a. Adonis and Neo
8. Evaluate models and perform hyperparameter tuning
   a. Siva and Feng
9. Report
   Whole team