DATA MINING PROJECT

CREDIT CARD FRAUD DETECTION

Team -Mubeena and River

Abstract: Survey shows 2.2 million fraud reports, of which 34% incurred a loss, up from 22% in 2019. Research shows consumers have experienced a surge of credit card fraud in 2020, and much of it can be attributed to scams related to COVID-19. For the companies, the customer's trust is more important to achieve some position in the business marketplace. Due to these fraudulent activities, companies can easily lose customer's trust, and suffer from customer churn (Jyoti Gaikwad, 2014).

Before companies and consumers alike have to face the consequences of fraudulent activities, credit card fraud classification programs are used to identify the fraud transactions and take measures to stop them before they go any further. The processes of fraud activities keep changing and as such we need to be prepared to detect them using different techniques. Hence, our group will be using different algorithms like classification algorithms, artificial neural network algorithms, anomaly detection algorithms, and etcetera to improve the solution.

Introduction

Given that crimes like scamming, fraud, and identity theft are on the rise, companies are invested in making sure the spending habits and credit card information of their customers are protected, for both their sake and for the sake of preventing financial loss for all parties involved. That is why we as data scientists who would likely work at such a company in the future are interested in learning how we could prevent credit card fraud, and what the best method or program would be for preventing fraud. In this report and in our program, we will explore and build various algorithms made for preventing credit card fraud, including neural networks, classification algorithms, and if time permits, anomaly detection algorithms. Our goal for doing this is to measure their performance and compare them to each other so that we may learn which algorithm would be best for this task, and of course provide ourselves with a chance to improve our coding capabilities by approaching a programming problem from multiple angles.

Why do we need to find fraud transactions?

Fraud activities happen in all industries. Many companies detect fraud after suffering heavy loss from these fraudulent activities or transactions. When it comes to financial-related firms, fraud activities can cause significant damage, so these companies need to detect fraud transactions before fraud activities cause problems.

Survey shows 2.2 million fraud reports, of which 34% incurred a loss, up from 22% in 2019. Research shows consumers have experienced a surge of credit card fraud in 2020, and much of it can be attributed to scams related to COVID-19.

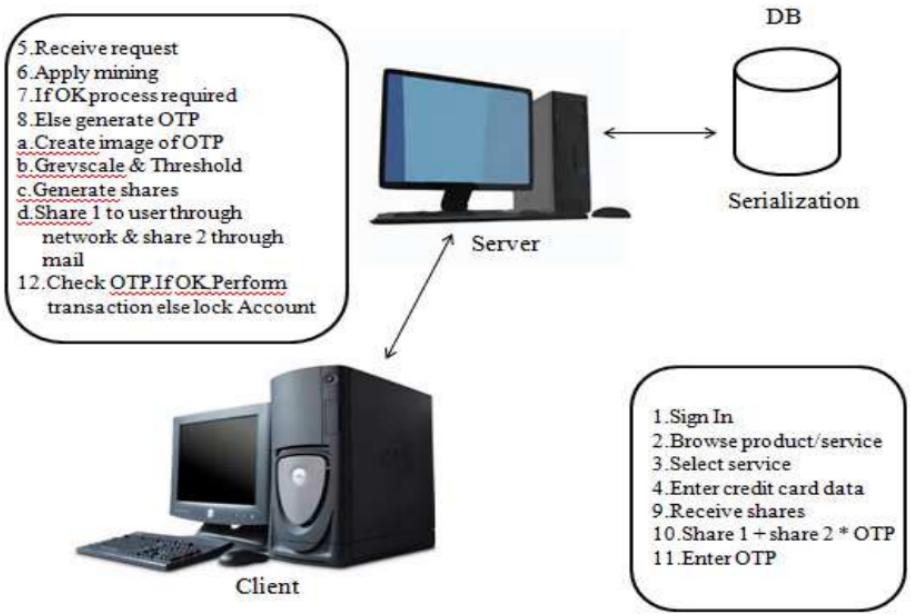
For the companies, the customer's trust is more important to achieve or reach some position in the business marketplace. Due to these fraudulent activities, companies can easily lose customer's trust, and suffer from customer churn.

What is Credit Card Fraud Detection?

Before facing the consequences of fraudulent activities by the companies, the credit card fraud classification problem is used to find fraudulent transactions. We have datasets that we've used to train and test our algorithms. All are unbalanced and contain credit card information. We ended up using only two out of three datasets because one of the datasets did not make much sense as their values were changed to meaningless symbols to protect the confidentiality of the sensitive information in the dataset, unfortunately to the point that the classification of which were fraudulent could not be identified anymore.

Experimental Setting

To understand the setting of the credit card activities, the following figure illustrates the system's step by step processes. These include client signing in to get a service or product. Credit card details will be entered by the client, the request will be received on the server side. This is where data mining algorithms are applied to process the authentication process. This may further require sharing of One Time Passwords between client and server for authentication purposes.



Tasks to monitor fraudulent activities on two different data sets (Jyoti Gaikwad, 2014)

1. Exploring and visualizing the Dataset
   - Track time and amount of transactions during Exploratory data analysis.
   - Current datasets (obtained from sets that exist in the public domain):
     - Data set 1: https://www.kaggle.com/mendozav/credit-card-fraud-detection-project
     - Dataset 2: https://www.kaggle.com/kartik2112/fraud-detection?select=fraudTrain.csv

- Data set 3: https://archive.ics.uci.edu/ml/datasets/Statlog+%28Australian+Credit+Approval%29

2. Determining number of fraud and non-fraud cases
   - Correlation between predictors
   - Training imbalanced data set
   - Extreme outlier detection and removal

3. Algorithms to be used and compared
   - Data Mining - Classification algorithms
   - Deep Learning - Artificial neural network algorithm
   - If time permits, Anomaly detection algorithms (LOF, DBSCAN, Isolation Forest) will be applied.

Modelling the Algorithms

Reconciling the datasets

Before jumping into modeling the algorithm, we had to make sure the datasets were okay to use. To make things more convenient for us and to keep a record, we created a GitHub repository. Two out of three of our datasets turned out to be ok to use. Initially we did not include dataset 2, the fraudTrain.csv and fraudTest.csv files, because the size of the files were so large that they exceeded the allotted storage on GitHub. However, the 3rd dataset called Australian.dat first took a lot of work on our part to convert a .dat file, a file format we are unfamiliar with, into a .csv file. We managed to do this and add headers to every column, but that was when we noticed that the file did not indicate which column would be the class file. We dug through the UCI machine learning repository archive that provided this dataset but unfortunately found that there is no definitive key to indicate which column should be the class column that would indicate whether the entry is fraudulent or not. If there was a fraud case, it would be a 1, if there is not a fraud case, it would be a 0, but there were multiple columns that showed they could be this class column because it was comprised of 0s and 1s. Therefore, the inconclusiveness of the dataset prompted us to throw it out as useless.

The good news was that we still had another dataset to rely on along with the first dataset, creditcard.csv, being perfectly fine to rely on to begin with. The only issue with dataset 2 was that it was too large to reasonably work with, so we broke it up into parts by using the fraudTest.csv dataset instead of both the fraudTrain.csv dataset and fraudTest.csv dataset. Even just the fraudTrain.csv dataset file size was so absurdly large that it stands to reason we could not use it in any program without breaking it up. However, the fraudTest.csv in its compressed state was small enough to upload to GitHub and to run through the algorithm on account of being the same file size as creditcard.csv. The best part of this dataset was that it came labelled already with the person who created the simulated dataset on Kaggle clearly explaining what each column represents. The dataset is explained to have two years' worth of credit card transactions derived from separate credit card reports of different people, all of which were simulated with a program the creator got from github. Here, the class column is labelled by is_fraud, and it shares the desired format of fraud cases being designated by 0s and 1s just like creditcard.csv. It was very easy to read so we

decided to use this algorithm for further testing. The dataset did, however, contain quite a few instances of categorical string data that needed to be converted to categorical numerical data and scaled with the Standard Scaler package. Still, this proved to not be as much trouble compared to having an unorganized unlabeled dataset like Australian.dat, and the converted and scaled data worked fairly well for our purpose.

Classification Algorithms

We have chosen to model the four effective classification algorithms to compare and conclude better results for detecting credit card fraudulent activities.

1. Decision Tree: provides highly effective structure to layout options between several courses of actions to investigate the possible outcomes. This algorithm also helps form a balanced picture of rewards and risks associated with all possible actions.

2. K-Nearest Neighbor: is faster as an algorithm as it stores the training dataset and learns from it while making real time predictions only.

3. Logistic Regression: is easier to implement and can be effectively trained and interpreted.

4. SVM: is effective when given a higher number of dimensions than the number of samples.

Now to talk about the classification algorithms themselves. In examination of the creditcard.csv file, we found that there were a total number of 284807 cases, with 284315 non-fraud cases and 492 fraud cases, making the percentage of fraud cases 17% with an outlier chance of 0.17%. In examination of the fraudTest.csv file, there are 555719 total cases, with 553574 non-fraud cases, 2145 fraud cases, a 39% chance of fraud, and an outlier chance of 0.38%. All this is to say that fraud is typically not that common, but nonetheless important to detect.

We began by splitting the datasets each into their own individual training and test sets using the sklearn module. The algorithms we utilized for this included decision trees, logistic regression, knn nearest neighbors, and SVM. We ran the creditcard.csv dataset and the fraudTest.csv dataset through these algorithms to make predictions for fraud cases and then we used the four different confusion matrix measures to evaluate how well they did. It should be noted that for precision, recall, and f-measure accuracy was weighted in calculations to account for the presence of multiple classes and avoid the program throwing warnings or assigning 0s to their scores, which would be inaccurate. Their scores regarding their predictions were are as follows:

creditcard.csv	Decision Tree	K-Nearest Neighbor	Logistic Regression	SVM
Accuracy	0.999367	0.999455	0.99919	0.999315
Precision	0.999339	0.999431	0.999132	0.999277
Recall	0.999367	0.99945	0.99919	0.999315

F-measure	0.999348	0.999427	0.999127	0.999262
Average score	0.9993553	0.999441	0.99916	0.999292

According to this table, it would appear as if K-nearest neighbor is the best classification algorithm for evaluating and predicting credit card fraud for this dataset with Decision Tree in second. To validate this, let's look at the fraudTest.csv:

fraudTest.csv	Decision Tree	K-Nearest Neighbor	Logistic Regression	SVM
Accuracy	0.9968	0.9964	0.9957	0.9965
Precision	0.99675	0.9955	0.99207	0.995844
Recall	0.996814	0.996428	0.995735	0.996572
F-measure	0.995769	0.995476	0.9939	0.9956
Average score	0.9965333	0.995951	0.994351	0.996129

It appears that fraudTest.csv produced different results using the same python packages, signifying that Decision Trees are better this time with SVM close behind.

All of these measures were obtained via the metrics package in python. Please check our program to verify the numbers if you wish to do so.

Deep Neural Network algorithm

Deep Neural Network is multiple layers deep Artificial Neural Network. This algorithm is known to model complex non- linear relationships. Hence, we have modelled the two datasets with deep neural network algorithm, to check if the results are much better than the algorithms which we have used earlier. Upon applying this algorithm on the credit card data set, we see that the accuracy, precision, recall and f-measure seems to be fairly close to the values of other algorithms, though still a bit lower than KNN. However, we feel it's worth noting that in all of our test runs Deep Neural Network was the only algorithm that managed to get 1.0 on recall a few times, meaning that it detected 100% of the true positive cases. It also occasionally outscored the other algorithms, though this seems to vary among all the algorithms from test to test. This is all to say it is a pretty close call.

Comparison of suitable metrics for two unbalanced data sets, modelled with different effective algorithms sets is shown below:

creditcard.csv	Decision Tree	K-Nearest Neighbor	Logistic Regression	SVM	Deep Neural Network
Accuracy	0.999367	0.999455	0.99919	0.999315	0.99942

Precision	0.999339	0.999431	0.999132	0.999277	0.999417
Recall	0.999367	0.99945	0.99919	0.999315	0.99942
F-measure	0.999348	0.999427	0.999127	0.999262	0.999419
Average score	0.9993553	0.999441	0.99916	0.999292	0.999419

As we see with credticard.csv, K-nearest neighbor won out in performance in this one, with Deep Neural Network close behind. We will validate this with fraudTest.csv:

fraudTest.csv	Decision Tree	K-Nearest Neighbor	Logistic Regression	SVM	Deep Neural Network
Accuracy	0.9968	0.9964	0.9957	0.9965	0.9965
Precision	0.99675	0.9955	0.99207	0.995844	0.99635
Recall	0.996814	0.996428	0.995735	0.996572	0.996509
F-measure	0.995769	0.995476	0.9939	0.9956	0.996426
Average score	0.9965333	0.995951	0.994351	0.996129	0.996446

It seems that in this dataset Decision tree won out with the highest values in every category, with Deep Neural Network as a close second and SVM in third.

Local Outlier Factor algorithm (Anomaly detection)

Local Outlier Factor is an algorithm designed to compare the local density of a point to that of its k-nearest neighbors, similar to that of other density-based clustering methods like KNN. According to towardsdatascience.com, which was a source we referred to a lot in researching the algorithms we should use for credit card fraud detection, “Local Outlier Factor is one of the most popular among outlier detection methods for its relatively simply intuition and effectiveness” (Will Arliss, 2020). That same article, however, notes that its ability to identify and locate outliers is “good but not great,” as the article showed from its own experiment that LOF was able to identify 69% of all outliers or “exoplanets” as they called it. Judging from our other data, that is probably not a good score, though we should look at the data first to make that call:

creditcard.csv	Decision Tree	K-Nearest Neighbor	Logistic Regression	SVM	Deep Neural Network	Local Outlier Factor
Accuracy	0.999367	0.999455	0.99919	0.999315	0.99942	0.00158
Precision	0.999339	0.999431	0.999132	0.999277	0.999417	0.00167

Recall	0.999367	0.99945	0.99919	0.999315	0.99942	0.89108
F-measure	0.999348	0.999427	0.999127	0.999262	0.999419	0.003339
Average score	0.9993553	0.999441	0.99916	0.999292	0.999419	0.224417

For creditcard.csv, the total runtime of the LOF algorithm, including training the model and making the prediction, was 23 minutes long. I mention this because it is in stark contrast with the other algorithms which never took more than 5 minutes to train and predict their values, and that is including the neural networks algorithm. Considering the amount of time it took and the overall very poor scores LOF is by far the worst algorithm in the set. As before KNN is in the lead and Neural Networks is second best.

We will examine the fraudTest data to see if this holds true:

fraudTest.csv	Decision Tree	K-Nearest Neighbor	Logistic Regression	SVM	Deep Neural Network	Local Outlier Factor
Accuracy	0.9968	0.9964	0.9957	0.9965	0.9965	0.00358
Precision	0.99675	0.9955	0.99207	0.995844	0.99635	0.003607
Recall	0.996814	0.996428	0.995735	0.996572	0.996509	0.9047
F-measure	0.995769	0.995476	0.9939	0.9956	0.996426	0.0071854
Average score	0.9965333	0.995951	0.994351	0.996129	0.996446	0.229768

This time, LOF took an absurdly longer amount of time to train and predict even just once. The total time for a single run was 1 hour and 15 minutes, which is absolutely horrendous given that its results were once again not great. The judgement for this dataset still stands that Decision Tree is first, Deep Neural Network is a close second and SVM is third best.

To be frank we were quite disappointed with this algorithm. Many sources recommended this as one of the best options for outlier detection but it seems that this algorithm was just not cut out for our task. The settings were adjusted a few times to see if there would be any significant difference if the n_nearest neighbors were increased or decreased, in a range from as little as 10 to as big as 50, and then executed on the creditcard.csv file. The results were still disappointing as the biggest increase or decrease was by .0002, still not enough to catch up with the other algorithms. To keep consistency with the other algorithms that required the nearest neighbors setting, the algorithm was set back to n_neighbors = 10.

If time permitted we would have liked to investigate further why this algorithm scored so poorly as it seems suspicious that an algorithm that typically seems to do okay detecting outliers according to our sources detected little to none compared to the rest. It was capable of detecting 90% of true positives as

indicated by recall, but it was neither accurate nor precise and its f-measure is very much a joke. We wondered if perhaps this was not the correct way to measure the performance of this algorithm, or if there are other settings we should have been aware of, or if we simply were not using it the way it was intended to be used. One reason for its poor performance I can say for sure based on the materials used in researching LOF is perhaps that our data is too complex/has too much dimensionality and has too much noise. If time permitted for us to run more tests despite its 1 hour and 15 minutes runtime and do more research on this algorithm, we may be able to conclude for certain what contributed to its poor performance.

Conclusion

We can conclude from our understanding that a combination of Decision Tree, K-Nearest Neighbor and Deep Neural Networks would be efficient for credit card fraud detection in this instance with these datasets. In our research of this problem, Decision Tree was the best algorithm for the first set, K-Nearest Neighbor was the best algorithm for the second set, and Neural Networks was very close behind both of them, sometimes even producing better results than the top two in some areas like recall. Please note that while we know the task we have chosen for this project is relatively simple, the main goal was to give ourselves experience applying tons of algorithms to a real world task that we may likely be asked to do in the future at whatever company we work at. By doing this exercise we gained both experience and a deeper understanding of data mining algorithms and how they work. We came to understand that the simplicity of the classification algorithms is something to be admired and appreciated, while neural networks, though difficult to implement if you are a novice, demonstrate thoroughness, adaptability, and a kind of dependability that we came to love when working with it, especially with the consistency of the high scores it displayed. We came to be disappointed with the anomaly detection algorithm we implemented, Local Outlier Factor, but we still must wonder if other more complex anomaly detection algorithms like DBSCAN or Isolation Forests would give us much better results. Perhaps in future work we would better be able to explore this.

Future work

Anomaly Detection assumes rare anomaly occurrences in the data, and the difference in features from significantly normal instances. Hence, with the anomaly detection algorithms of DBSCAN and Isolation Forest we hope to focus on detecting a different criteria for determining the right choices for achieving better results. We would hope to find out if, since our problem of credit card fraud is defined as an anomaly detection problem, other anomaly detection algorithms would outperform the classification algorithms, the deep neural networks algorithm, and the Local Outlier Factor algorithm. We would also hope to research and investigate more why LOF did not live up to our expectations like we expected.

Reference:

Boaz Shmueli. 2020. Multi-class metrics made simple, part I: Precision and recall. (June 2020). Retrieved November 24, 2021 from https://towardsdatascience.com/multi-class-metrics-made-simple-part-i-precision-and-recall-9250280bddc2

Boaz Shmueli. 2020. Multi-class metrics made simple, part II: The F1-score. (July 2020). Retrieved November 24, 2021 from https://towardsdatascience.com/multi-class-metrics-made-simple-part-ii-the-f1-score-ebe8b2c2ca1

Jason Brownlee. 2021. Your first deep learning project in python with keras step-by-step. (October 2021). Retrieved November 24, 2021 from https://machinelearningmastery.com/tutorial-first-neural-network-python-keras/

Jyoti Gaikwad, Amruta Deshmane, Harshada Somavanshi, Snehal Patil, and Rinku Badgujar. 2014. Credit Card Fraud Detection using Decision Tree Induction Algorithm. (2014). Retrieved November 24, 2021 from https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.675.408&rep=rep1&type=pdf

Keras Team. Keras Documentation: Keras API reference. Retrieved November 24, 2021 from https://keras.io/api/

Will Arliss. 2020. Local outlier factor for imbalanced classification. (October 2020). Retrieved November 24, 2021 from https://towardsdatascience.com/local-outlier-factor-for-imbalanced-classification-cbced8f84baf