Analysis of Predictive Algorithms in Use of Determining Credit Card Fraud

Ralph J. Gleaton
ralphg@email.sc.edu

May 03, 2022

1 Introduction

Credit cards are one of the most popular methods of paying for goods and services around the globe. However, credit cards are often susceptible to fraud, with credit card companies losing an estimated $24.26 Billion in 2018. Credit card fraud rates have risen every year and are predicted to continue rising faster than ever as time progresses [1]. There is obviously a substantial need for a reliable predictive tools to properly detect and stop the fraud before it can cause significant damage.

The goal of this project is to test the ability of several machine learning techniques to detect credit card fraud. The techniques I will be testing include a decision tree, random forest (both with and without hyperparameter tuning), and a deep neural network. Having an efficient and reliable way to detect fraud could potentially save credit card companies billions of dollars every year.

To train the models I will be processing several categorical and numerical features of a specific transaction. Given these data points the models will output either a 1 or a 0 with 1 corresponding to a fraudulent transaction and 0 corresponding to a legitimate transaction.

2 Related Work

Zareapoor and Shamsolmoali have an excellent paper where they also analyze and design several data mining techniques to evaluate their performance in detecting credit card fraud. They also go over several common techniques currently employed in this use. They end by introducing a bagging classifier which is based on a decision tree architecture. They evaluate performance on real life credit card transactions unlike this paper which uses a simulated data set. [8]

3 Dataset and Features

The models were trained using a Kaggle dataset – Credit Card Transactions Fraud Detection Dataset [2]. The dataset is in the form of two csv files, one with training data and the second with testing data. Both datasets contain 23 features for each transaction, these features being: row number of transaction, date and time of transaction, credit card number, merchant where transaction was made, category of transaction, amount of currency transferred, first and last name of credit card holder, gender of card holder, address, city, state, zip code, longitude and

latitude, population of city, job of credit card holder, birthdate of credit card holder, transaction number, Unix time, and longitude and latitude of merchant to determine whether the transaction was fraudulent.
To processes this data for the decision tree and random forest I passed the data into a pandas data frame. The last column was dropped as it's the predictive variable and a separate data frame was created for it. I then used a one-hot encoder to convert the categorical variables into numeric variables [3] [4]. For the deep neural network, the process is very similar, however, for reasons discussed in section 5, instead of encoding the categorical features they are dropped from the data frame. This leaves us with only 9 features from which to train our network.
The datasets contain significant amounts of data with the training set containing 1,296,675 entries and the test set containing 555,719 entries. For the DNN however, I only trained it on 100,000 entries and tested on 15,000 due to time and memory limitations. However, the training set only contains 7,514 instances (0.5% of the dataset) where the transaction is fraudulent. This, while similar to real world statistics, can present challenges for training machine learning models.

4 Methods

I created several models to perform classification. That is, to classify a transaction as either fraudulent (1) or legitimate (0). For this task we have 3 methods:

4.1 Decision Tree

A decision tree is a flow chart like tree structure where each node represents a feature and each branch/edge off that node represents a feature value. These edges either lead to new nodes, which, in turn, have their own edges or a classification that represents the end state of the path. Decision trees are constructed in a top-down recursive divide-and-conquer manner [5]. We begin at the root node and follow the tree down based on the value of each feature in our instance. Once we reach the bottom, we return the category predicted by the tree.
When constructing a decision tree, nodes are chosen based on which features provides the greatest information gain [5], where information gain is calculated as:

$$G(S,A) = E(S) - \sum_{v \in Values(A)} \frac{|S_v|}{S} E(S_v)$$

With $E(S)$ in this instance representing the entropy of a certain state S. Entropy of a discrete random variable x is a measure of the uncertainty associated with the value of x [5]. Entropy is calculated via:

$$E(S) = -\sum_{x \in S} p(x) \log\, p(x)$$

4.2 Random Forest

A random forest is a classification algorithm that consists of many decisions tree and outputs the class that is the mode of all classes outputted by the individual decision trees. This can often produce a highly accurate classifier for many data sets. To build a random forest we build a number of decision trees on bootstrapped training samples for each split we consider. A random sample of m predictors are chosen to be split candidates out of the full set of p predictors [5]. For this project the default value of m shall be $\sqrt{p}$ as recommended by the inventors of the random forest for classification problems [5].

4.3 Deep Neural Network

I attempted to use a deep neural network for classification to predict if a given transaction was fraudulent or legitimate. Deep neural networks consist of multiple layers, each containing any number of nodes which are interconnected via edges. The first layer is the input and the last layer is the output, the layers in between are called hidden layers. These networks, once trained, are extremely efficient at data processing. Any given node in the network past the input is just the weighted sum of its inputs passed into an activation function. To train a neural network we compute its loss then perform back propagation to compute the gradient of the loss relative to the weights of the previous nodes [6].
My network consisted of 4 layers. An input layer with 9 nodes, the first hidden layer with 20 nodes, the second hidden layer with 10 nodes, and the output layer with a single node. At each hidden layer a batch normalization function was applied due to the low amount of variance in the outputs of the samples. Batch normalization normalizes each batch (50 transactions in our case) passed into the network via:

$$y = \frac{x - E(x)}{\sqrt{Var(x)} + e} * \gamma + \beta$$

Where γ and β are learnable parameters.
After the batch normalization function is applied, we use a sigmoid activation function:

$$\sigma(x) = \frac{1}{1 + e^{x}}$$

Where x is calculated as:

$$x = (W^{i}a^{i} + b^{1}) \in \mathbb{R}^{3}$$

Where $W^{i}$is the weight matrix of the previous layer at position i and a^{i} is the output of the ith node in the previous layer. To compute backpropagation for each node we make use of the chain rule of differentiation [7]. For example, to compute the back propagation of a single node C with respect to weight W_1 we'd have:

$$\frac{dC}{dW_1} = \frac{dC}{dy} \cdot \frac{dy}{dz} \cdot \frac{dz}{dW_1} = -d\frac{\partial \log(y)}{\partial W_1} - (1-d)\frac{\partial \log(1-y)}{\partial W_1}$$

$$z = \sum_{n=1}^{i} W_n X_n$$

4.4 Evaluation

To evaluate each model, we will use 4 different indicators:

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

$$F1\ Score = \frac{TP}{TP + \frac{1}{2}(FP + FN)}$$

$$R2\ Score = 1 - \frac{\sum_i (y_i - \breve{y}_i)^2}{\sum_i (y_i - \mu)^2}$$

$$Mean\ Absolute\ Eror\ (MAE) = \frac{1}{n}\sum_{i=1}^{n} |y_i - \breve{y}_i|$$

Where TP is the true positive rate, TN is the true negative rate, FP is the false positive rate, and FN is the false negative rate.

5 Results and Discussion

5.1 Decision Tree

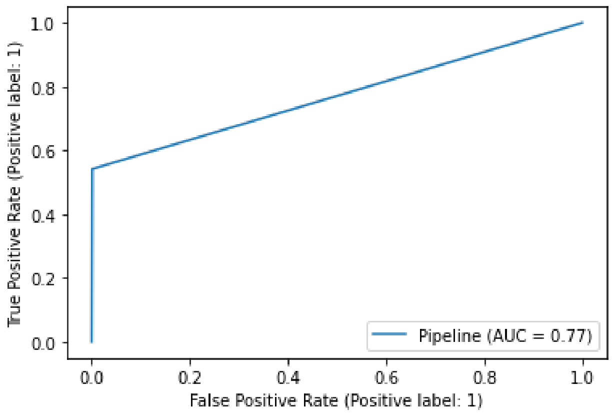
The decision tree was trained on the entire training set and tested with the entire testing set. The confusion matrix it produced is:

552927	602
947	1198

In this confusion matrix a positive result is considered to be ‘not fraud’ or a legitimate transaction. The scores of decision tree are:

Accuracy: 0.9972
F1 Score: 0.9972
R2 Score: 0.275
MAE: 0.0028

Overall, the model has a very high accuracy and a low error. Both F1 and R2 scores are reasonable. However, the model still isn’t perfect for predicting fraud with 602 false positive and 947 false negatives.


True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
Pipeline (AUC = 0.77)
0.0
0.2
0.4
0.6
0.8
1.0

On the left we have a visualization of the decision tree produced by the training data and on the right is the ROC curve of the model.

5.2 Random Forest

For the random forest I used scikit-learns random forest module to create and fit a random forest to the data. I also performed hyperparameter tuning on the random forest. For the standard random forest, we have the confusion matrix:

55368	6
2136	9

The scores for the random forest model are:

Accuracy: 0.996
F1 Score: 0.996
R2 Score: -0.0025
MAE: 0.0039

This model seems to default to nearly every instance being considered legitimate. The scores here still seem decent on the surface, though it's clear if that's due to the low number of fraudulent cases in the test set.

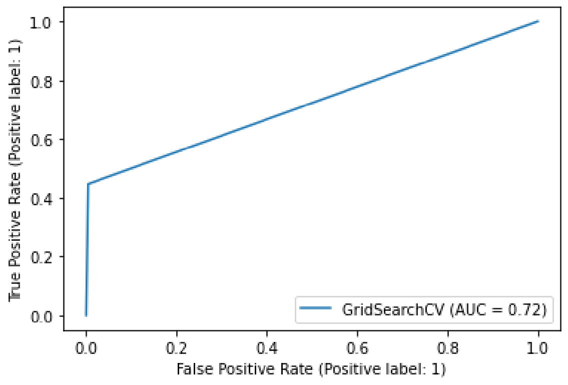
To see if I could remedy this, I performed hyperparameter tuning using grid search (GridSearchCV) on the model. That process resulted in the confusion matrix:

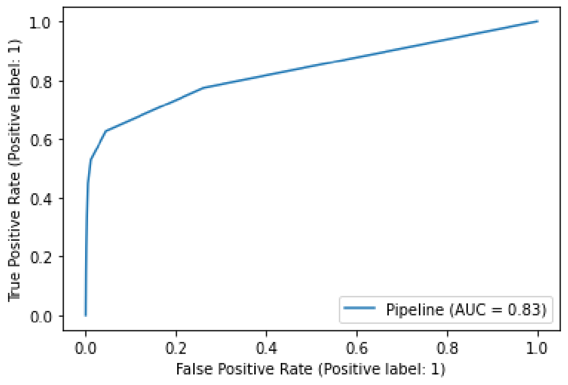
553574	0
2145	0

With scores:

Accuracy: 0.996
F1 Score: 0.996
R2 Score: -0.0038
MAE: 0.003

The model post-hyperparameter tuning actually scored lower than previously. The model has completely defaulted to predicting the transaction is legitimate every time. It's not presently clear why this is happening.





On the left we have the ROC curve for the random forest post-hyperparameter tunning and on the right we have the ROC curve for the generic random forest before tunning.

5.3 Deep Neural Network

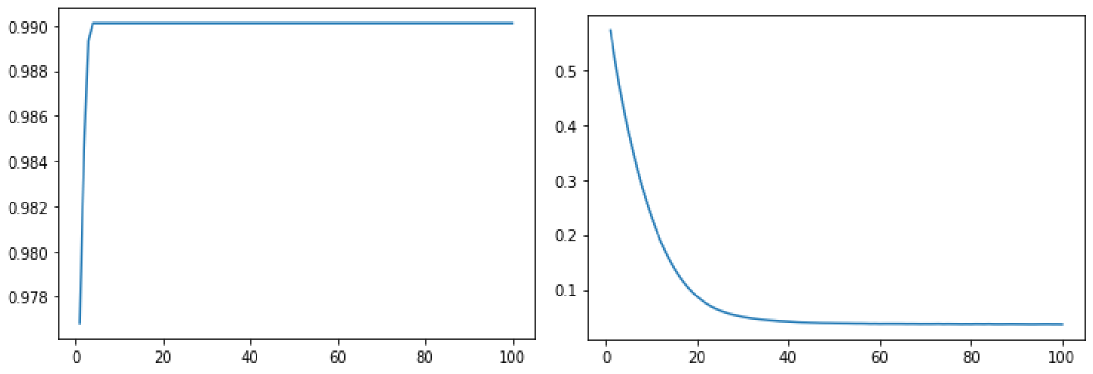
The deep neural network (DNN) was extremely finicky to get working. Generally speaking, categorical networks will feature SoftMax layers at the output to convert the output to a probability distribution of the various options. However, because this network consists of only one output that feature is not present here. Instead, the output is rounded to its closest whole value.
Encoding the data for this network was challenging as the one-hot encoder couldn't just be passed into a pipeline like the previous models and the systems I had access to didn't have enough memory to store the new table if done without pipelining. I experimented with in place replacing the categorical features with numeric values, however, that resulted in a less accurate models. In the end I just ended up dropping the categorical features and only passing in the numeric features. This gave me the confusion matrix:

14954	0
46	0

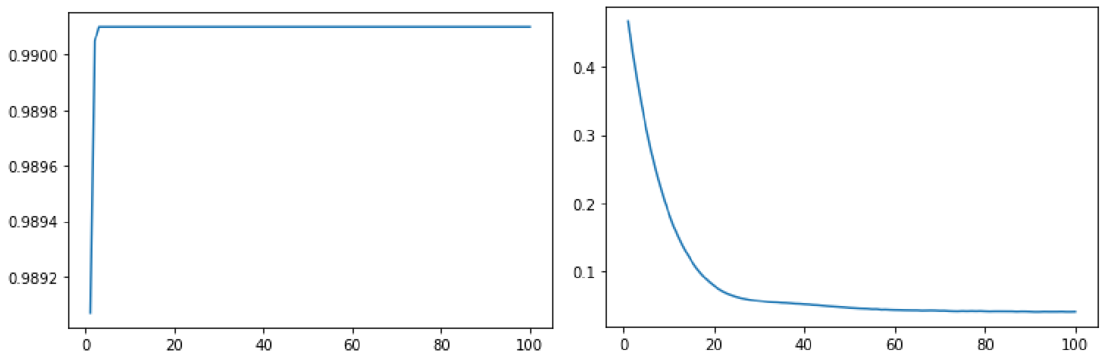
With scores:

Accuracy: 0.9969
F1 Score: 0.9969
R2 Score: -0.003
MAE: 0.0226

Just like in the random forest with hyperparameter tuning the model defaults to predicting every transaction to be legitimate. This is likely due to the lower number of fraudulent transactions in the training data despite my attempts to remedy that issue via batch normalization. I also attempted to add a dropout layer (p=0.2) after the first hidden layer, however, the results were the same.



Pictured left if the accuracy vs number of epochs graph and pictured right is the loss vs number of epochs graph for the DNN.



Pictured left is the accuracy vs number of epochs graph and pictured right is the loss vs number of epochs graph for the DNN with a dropout layer.

6 Conclusion and Future Work

This project aimed to analyze different machine learning methods for predicting credit card fraud. Of the methods I analyzed a simple decision tree performed the best but realistically was the only model that made predictions other than a legitimate transaction. Most of these issues likely come from the lack of variety in the training data. Given a training set with more fraudulent transactions I believe some of the other methods could have ended up outperforming the decision tree. In the future it would also be interesting to test a logistic regression model out on the data which is also included in the scikit-learn library. This project as a whole, while not completely inconclusive, appears to have been hampered by the lack of fraudulent transactions present in the training data. All models in the end did end with low errors and high accuracies, just not because they made meaningful predictions

No data set balancing techniques were employed in this project as to not accidentally implement any inherent bias into the project. However, this may have been useful in this case and in future projects. Though, with only seven thousand fraudulent examples in the training set it's possible that data set balancing techniques would have been difficult to implement regardless.

7 References

[1] Credit Card Fraud Statistics. Shift Credit Card Processing. (2022). Retrieved 3 May 2022, from https://shiftprocessing.com/credit-card-fraud-statistics/
[2] https://www.kaggle.com/datasets/kartik2112/fraud-detection/code
[3] https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html
[4] Seger, C. (2018). *An investigation of categorical variable encoding techniques in machine learning: binary versus one-hot and feature hashing* (Dissertation). Retrieved from http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-237426
[5] Dr. Jianjun Hu. (2022, January 10-16). *Lecture 3-ML* [PowerPoint presentation]. UofSC: Columbia, South Carolina, USA.
[6] Dr. Jianjun Hu. (2022, January 10-16). *Lecture 7-deeplearning-v1* [PowerPoint presentation]. UofSC: Columbia, South Carolina, USA.
[7] Dr. Jianjun Hu. (2022, January 10-16). *Lecture 6-Nural Networks* [PowerPoint presentation]. UofSC: Columbia, South Carolina, USA.
[8] Zareapoor, M., & Shamsolmoali, P. (2015). *Application of Credit Card Fraud Detection: Based on Bagging Ensemble Classifier*. Procedia Computer Science, 48, 679-685. https://doi.org/10.1016/j.procs.2015.04.201