Machine Learning Engineer Nanodegree

Credit Card Fraud Detection
Capstone Project

Ricky J. Sparks

April 30th, 2019

Project Overview

Fraud detection is a serious problem in the modern world. The news reports on countless attacks of credit card information being stolen annually. The transactions of fraudulent cards might seem minimal in occurrence but, on a larger scale, these small

occurrences can cost companies millions in losses. This is where machine learning comes into play learning from the patterns and adapting to new possible schemes. Machine learning can give credit card companies the ability to recognize fraudulent credit card transactions so that customers are not charged for items that they did not purchase.

Problem Statement

The problem presented is analyzing historical data of credit card transactions that were fraudulent and cards that weren't fraudulent. The goal of this model is to predict future transactions as fraud. The model will implement machine learning algorithms on the dataset that will ultimately lead us to an optimal algorithm that performs superior to the others. It is important that this model identifies most if not all of the fraud transactions with as much accuracy as possible. Thus, the nature of this problem is binary in nature as 0 represents valid transactions and 1 represents fraud transactions.

Metrics

This model should be expected as a high recall model since the nature of the problem is to catch more importantly fraudulent transactions even if the model has to sacrifice a small amount of valid transactions being misclassified as fraud. All in all, the metric recall should be represented in this model as a main metric.

Analysis

Data Exploration

The datasets contains transactions made by credit cards in September 2013 by european cardholders. This dataset presents transactions that occurred in two days, where we have 492 frauds out of 284,807 transactions. The dataset is highly unbalanced, the positive class (frauds) account for 0.172% of all transactions.

It contains only numerical input variables which are the result of a PCA transformation. Unfortunately, due to confidentiality issues, we cannot provide the original features and more background information about the data. Features V1, V2, ... V28 are the principal components obtained with PCA, the only features which have not been transformed with PCA are 'Time' and 'Amount'. Feature 'Time' contains the seconds elapsed between each transaction and the first transaction in the dataset. The feature 'Amount' is the transaction Amount, this feature can be used for example-dependant cost-senstive learning. Feature 'Class' is the response variable and it takes value 1 in case of fraud and 0 otherwise.

Dataset Source: https://www.kaggle.com/mlg-ulb/creditcardfraud

Dataset Stats below

```
In [4]: data.head()
```

Out[4]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838	-0.110474	0.066928	0.128
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672	0.101288	-0.339846	0.167
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679	0.909412	-0.689281	-0.327
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274	-0.190321	-1.175575	0.647
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278	-0.137458	0.141267	-0.206

5 rows × 31 columns

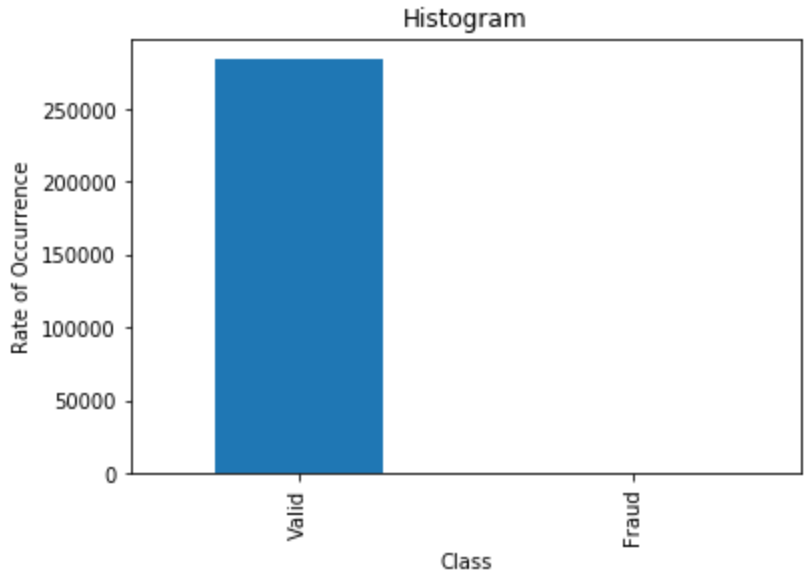
```
In [18]: #print(data.describe())
         data.describe()
```

Out[18]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9
count	284807.000000	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05
mean	94813.859575	3.919560e-15	5.688174e-16	-8.769071e-15	2.782312e-15	-1.552563e-15	2.010663e-15	-1.694249e-15	-1.927028e-16	-3.137024e-15
std	47488.145955	1.958696e+00	1.651309e+00	1.516255e+00	1.415869e+00	1.380247e+00	1.332271e+00	1.237094e+00	1.194353e+00	1.098632e+00
min	0.000000	-5.640751e+01	-7.271573e+01	-4.832559e+01	-5.683171e+00	-1.137433e+02	-2.616051e+01	-4.355724e+01	-7.321672e+01	-1.343407e+01
25%	54201.500000	-9.203734e-01	-5.985499e-01	-8.903648e-01	-8.486401e-01	-6.915971e-01	-7.682956e-01	-5.540759e-01	-2.086297e-01	-6.430976e-01
50%	84692.000000	1.810880e-02	6.548556e-02	1.798463e-01	-1.984653e-02	-5.433583e-02	-2.741871e-01	4.010308e-02	2.235804e-02	-5.142873e-02
75%	139320.500000	1.315642e+00	8.037239e-01	1.027196e+00	7.433413e-01	6.119264e-01	3.985649e-01	5.704361e-01	3.273459e-01	5.971390e-01
max	172792.000000	2.454930e+00	2.205773e+01	9.382558e+00	1.687534e+01	3.480167e+01	7.330163e+01	1.205895e+02	2.000721e+01	1.559499e+01

8 rows × 31 columns

Exploratory Visualization



The histogram graph above as one can see shows how much the data is greatly skewed toward valid transactions versus fraudulent ones. One key takeaway from analyzing this dataset based on the histogram is that it is very unbalanced.

Algorithms and Techniques

Unsupervised Outlier Detection using Local Outlier Factor (LOF)

The anomaly score of each sample is called Local Outlier Factor. It measures the local deviation of density of a given sample with respect to its neighbors. It is local in that the anomaly score depends on how isolated the object is with respect to the surrounding neighborhood. More precisely, locality is given by k-nearest neighbors, whose distance is used to estimate the local density. By comparing the local density of a sample to the local densities of its neighbors, one can identify samples that have a substantially lower density than their neighbors. These are considered outliers.

Source: https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.LocalOutlierFactor.html

Isolation Forest Algorithm

The IsolationForest 'isolates' observations by randomly selecting a feature and then randomly selecting a split value between the maximum and minimum values of the selected feature.

Since recursive partitioning can be represented by a tree structure, the number of splittings required to isolate a sample is equivalent to the path length from the root node to the terminating node.

This path length averaged over a forest of such random trees, is a measure of normality and our decision function.

Random partitioning produces noticeably shorter paths for anomalies. Hence, when a forest of random trees collectively produces shorter path lengths for particular samples, they are highly likely to be anomalies.

Source: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.IsolationForest.html

Logistic Regression Algorithm

In the multiclass case, the training algorithm uses the one-vs-rest (OvR) scheme if the 'multi_class' option is set to 'ovr', and uses the cross- entropy loss if the 'multi_class' option is set to 'multinomial'. (Currently the 'multinomial' option is supported only by the 'lbfgs', 'sag' and 'newton-cg' solvers.) This class implements regularized logistic regression using the 'liblinear' library, 'newton-cg', 'sag' and 'lbfgs' solvers. It can handle both dense and sparse input. Use C-ordered arrays or CSR matrices containing 64-bit floats for optimal performance; any other input format will be converted (and copied). The 'newton-cg', 'sag', and 'lbfgs' solvers support only L2 regularization with primal formulation. The 'liblinear' solver supports both L1 and L2 regularization, with a dual formulation only for the L2 penalty.

Source:https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

Benchmark

This dataset was obtained from Kaggle and based upon my observation the accuracy, f1-score, and recall shouldn't lesser than 75%. Otherwise, this would be a poor model falling short of other credit card fraud detection models in the data science space.

Methodology

Data Preprocessing

The first data preprocessing tactic I performed was normalizing the Amount Column. Then, I performed a shuffling technique on the unbalanced dataset to combat unbalanced classes within the dataset.

Implementation

The chosen tactic used on the data distribution needed to split evenly across the valid and fraud classes to a ratio of 50% for each class. Thus, the chosen tactic was under-sampling which included shuffling to ensure minimal information loss occurred. Thus, I went on to perform the algorithms on the dataset.

Refinement

The model fitting was used to create a simplified representation of the data. The prediction values were reshaped to 0 for valid and 1 for Fraud. Thus, the classifications are processed and generate a simplified representation of both algorithms in the outputted code cells.

Results

Model Evaluation and Validation

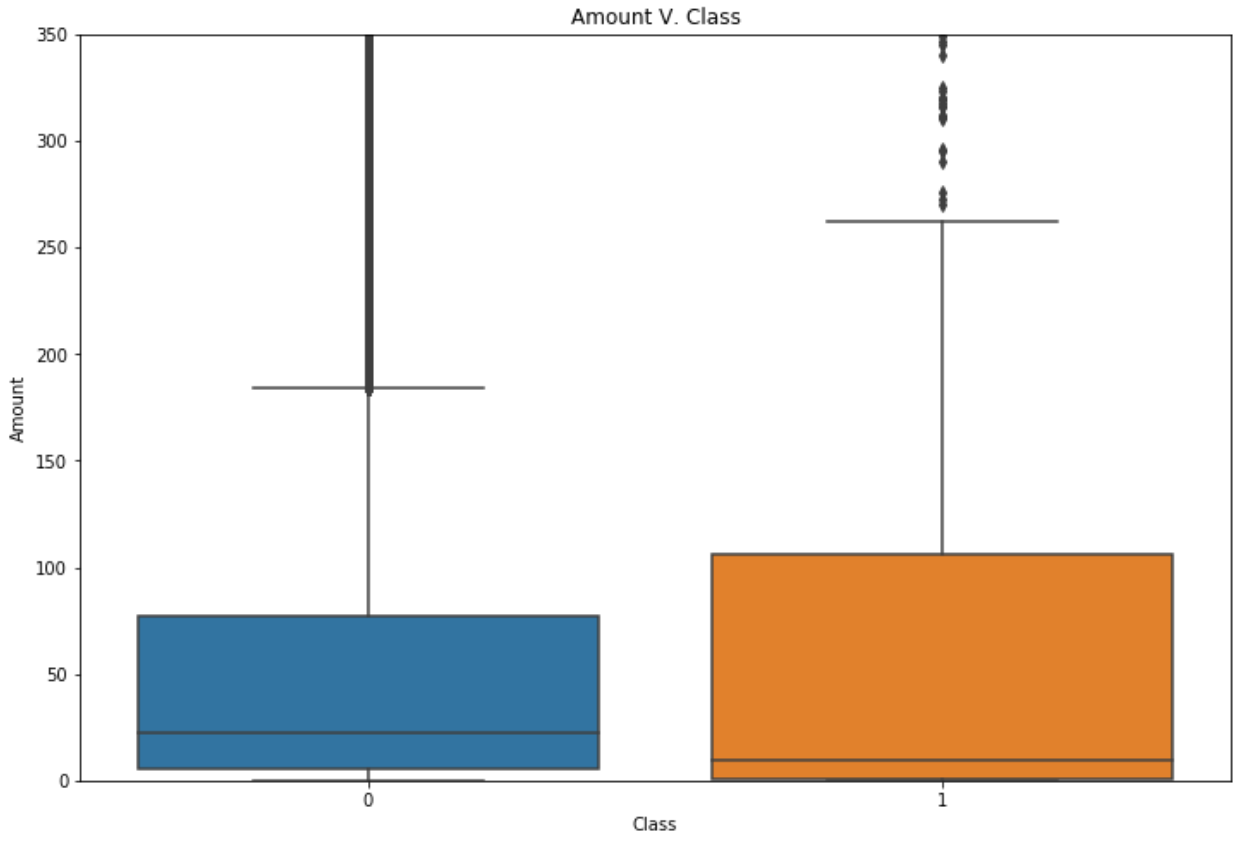
The model that won was Isolation Forest algorithm. Why so? This selected model obtained 0.34 in precision, recall, and F1-score across. While Local Outlier Factor Algorithm obtained 0.05 in precision, recall, and F1-score across. Logistic Regression obtained 0% in precision, recall, and F1-score across. Thus, Logistic Regression Algorithm was selected for having the best results. This model is not ready to detect credit card fraud just yet.

Justification

The model that was designed with the Isolation Forest Algorithm obtained 0.35% in precision, recall, and F1-score. On the other hand, Local Outlier Factor Algorithm obtained 0.05% in precision, recall, and F1-score. Logistic Regression performed even worse obtaining 0% in precision, recall, and F1-score across. All in all, none of these models are quite yet ready to detect credit card fraud.

Conclusion

Free-Form Visualization



The visualization above shows how fraud transaction is in greater amounts than valid transactions. The two classes are 0 for valid transactions and 1 for fraud transactions.

Reflection

The flow of this project started with installing the dependencies. Then the dataset was loaded into the Ipython Notebook(also known as Jupyter Notebooks). From here some quick data analysis was performed using pandas on the data to display to some visualizations. Then the data was preprocessed with normalization and Under-sampling was also used on the dataset with shuffling. Finally, the algorithms were applied to the dataset. The hardest thing about this model was figuring out what parameters to add and tune. I learned that this could also be solved in the future hopefully with a software that can aid in determining what algorithms to use for the dataset.

Improvement

Looking back on the whole process of this project I would say one improvement that could be added is additional complex algorithms. Secondly, adding more data to this model could also improve it greatly.