**Anomaly Detection: How to Identify Credit Card Fraud**

Introduction

One of the more intriguing application of machine learning is anomaly detection. An anomaly can be an outlier, rare occurrence, or a data point that could raise suspicions about the data. (Flovik, 2018) There are multiple real-world applications of this, some example includes insurance fraud, identification of health problems, and malfunction machinery. The application that will be examined with this project is credit card fraud. What will be investigate here is how do these models that help protect us from fraudulent charges are created and which classification algorithms work the best.

Abstract

As the world becomes more entwined with the digital world, the potential for credit card fraud grows and grows. Credit card companies have to try and stay one step ahead of the criminals that try to take advantage of unsuspecting victims. One of the ways to try to identify fraudulent transaction is by the use of a machine learning model. Since credit card companies store a tone of transaction data, they are able to train, and continual train, models to identify fraudulent transaction while they are happening and stop them. By doing this the individual that had their information stolen will have peace of mind, and the credit card companies can save money. This project will look over how models and created and determine which kind of classification algorithm works best for this kind of data.

Research Questions

The two main research questions for this project are:

1. How is a machine learning model created from various anonymized variables (i.e. how is it processed before training of the model can begin)?
2. Which classification algorithms work the best for credit card data?

Methods

The dataset that will be used here is the *Credit Card Fraud* dataset from Kaggle. (Machine Learning Group - ULB, 2018) This data set contains 31 variables and over 280,000 transactions (rows). One of the biggest red flags in this dataset is how it is balanced. Only 492 transactions are classified as fraudulent (this translates to about 0.001% of the dataset being classified as fraudulent). This is a scary figure going into modeling, if every row in the test set was classified as a legitimate transaction, the accuracy could be around 99%. This makes it appear the model is accurate, but is it really?

Since this application requires the ability to handle large amount of data with ease, Apache Spark along with PySpark will be used to process the data and create the models. The data will be split up with 70% of the rows dedicated to the training set and 30% for the test set. There are three different classifiers that will be investigated: logistic regression, gradient-boosted tree, and linear support vector machine. Logistic regression is the simplest classifier of the three, it is primarily used for binary classification problems, however, it assumes that variables are independent from one another. (Brownlee, 2016) This might be a problem for this scenario since we are not 100% if that is the case. Since information was anonymized it’s possible the variables may not be independent. Gradient-boosted trees take decision trees and create an ensemble of smaller trees. The goal of using all of these trees is to try and reduce the amount of errors from the previous one. (Yıldırım, 2020) Lastly, the linear support vector machine is a classifier that determines a hyperplane in n-space that divides the data points well enough to classifier them into two group. (Ray, 2017)

Even though three different classifiers will be used, the data processing steps for each remains the same. There are 29 feature variables that need to be processed before a model is created. The first step is to use the VectorAssembler feature. This takes all of the features that will be used in the model and assembles them into a single vector inside a single column. Next, the StandardScaler feature is used to standardize the values from the VectorAssembler so they are within the same variance. From here the results are passed to the classifier for training. To make the coding process easier, a pipeline is used to create an array of tasks and transformations the training data must undergo. Figure 1 (below) shows the Python code used to define the various processing and classification functions as well as how the Pipelines are created.

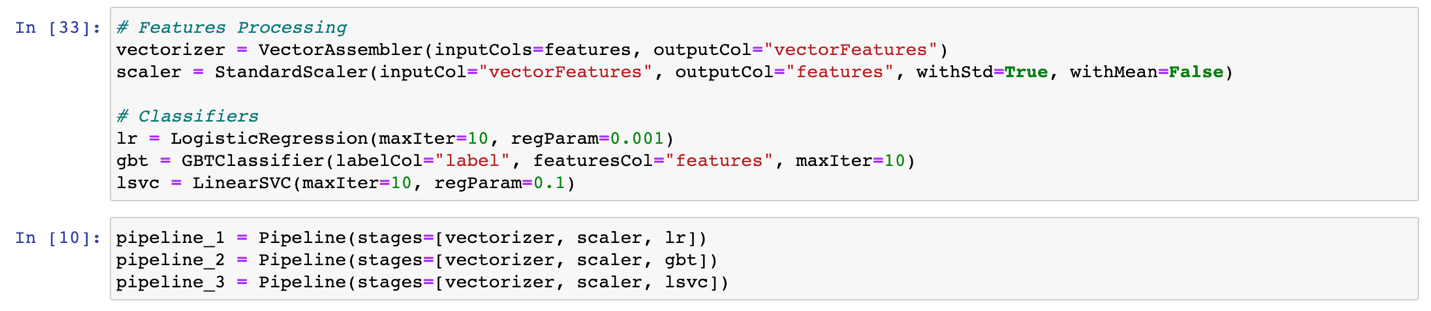


Figure 1 - Pipeline that is used to process and classify fraudulent Credit Card Transactions in Spark.

From here predictions are made on the test set and an evaluator is used to determine how well the model performs. For this instance, the BinaryClassificationEvaluator is used in PySpark to evaluate the predictions looking at the area under the ROC curve. This metric is important because it’s the best indicator of the models performance. With all of the methods outlined and described, it’s time to find out which model performed the best!

Results

After training the classifiers with the training set and evaluating them with the test set, the area under the ROC curve metric was computed for each. The results for each classifier are outline in Figure 2 below.

|  |  |
| --- | --- |
| **Classifier** | **Area Under ROC Curve** |
| Logistic Regression | 0.80 |
| Gradient-Boosted Tree | 0.89 |
| Linear Support Vector Machine | 0.65 |

Figure 2 - Area under the ROC curve for three different classifiers.

Looking at these results the two best classifiers for this application are logistic regression and gradient-boosted tree. Both of these classifiers have at or above a 0.8 area under the ROC curve. Of these two, gradient-boosted tree is the clear winner!

Conclusion

When handling a dataset that is heavily unbalanced, it is a delicate balancing act to find a classifier that is accurate. In this scenario three different classifiers were used to assign a label of legitimate or fraudulent based on 29 different features. What was discovered is that for this particular domain, the gradient-boosted tree classifier was the best. Using an ensemble of trees proved to outperform logistic regression and linear single vector machine (LSVM). However, logistic regression did not perform badly, it produced an area under the ROC curve of 0.8 while the gradient-boosted tree had a value of almost 0.9. This was surprising since logistic regression one of the simpler classifiers, yet outperformed a more complicate one (LSVM).

Bibliography

Brownlee, J. (2016, April 1). *Logistic Regression for Machine Learning*. Retrieved from Machine Learning Mastery: https://machinelearningmastery.com/logistic-regression-for-machine-learning/

Flovik, V. (2018, December 21). *How to use machine learning for anomaly detection and condition monitoring*. Retrieved from Towards Data Science: https://towardsdatascience.com/how-to-use-machine-learning-for-anomaly-detection-and-condition-monitoring-6742f82900d7

Machine Learning Group - ULB. (2018, March 22). *Credit Card Fraud Detection*. Retrieved from Kaggle: https://www.kaggle.com/mlg-ulb/creditcardfraud

Ray, S. (2017, September 13). *Understanding Support Vector Machine(SVM) algorithm from examples (along with code)*. Retrieved from Analytics Vidhya: https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/

Yıldırım, S. (2020, February 17). *Gradient Boosted Decision Trees-Explained*. Retrieved from Towards Data Science: https://towardsdatascience.com/gradient-boosted-decision-trees-explained-9259bd8205af