Machine Learning Engineer Nanodegree

Capstone Project

Brian Wozniak
January 24th, 2019

I. Definition

Project Overview

Granting credit is a process that stimulates the economy, enables individuals to live life without having to wait to pay cash for everything, and allows credit-granting institutions to create profit and jobs. Because of these things, it makes sense to extend credit to all those who will respect that credit and will make payments according to plan. How can credit be extended to such people? How can one decide whether someone will be willing and able to pay back a loan?

This is a problem to which machine learning can be applied. Using information about an individual, especially historical, financial information, and a proper model can be used to decide (perhaps imperfectly, but very well) who will be willing and able to honor payment of a loan. As someone who works for an institution that grants credit, this is what I decided to investigate for the Capstone Project.

The information for this Project comes from the past Kaggle competition, "Give Me Some Credit." The following is the link to the competition: https://www.kaggle.com/c/GiveMeSomeCredit. This competition was held by a company that solicited help from the Kaggle community to build a credit model that would most accurately determine who would experience financial distress in the following two years. The company provided a training and test dataset with financial, age, and dependent information on 250,000 borrowers. In this project I will use the 150,000 observations from the training dataset to build three machine learning models. The goal of the project is to build at least one model that can more strongly predict the probability of a person experiencing financial distress in the next two years as measured by the area under the ROC curve.

Problem Statement

The problem to be solved is to build a machine learning model that will predict if an individual will experience financial distress in the next two years better than a benchmark model. Quantitatively, the problem is to build a model that will have a higher ROC area under the curve metric than the logistic benchmark model.

The strategy to solve this problem will be as follows:

1. Explore the data (Visualizations, distributions, basic statistics)
2. Preprocess the data (Impute missing data, transform variables, identify and remove outliers, and balance classes using SMOTE)
3. Build three machine learning models: Random Forest, Support Vector Machine (SVM), and Gradient Boosting Machine (GBM)
4. Use grid to optimize parameters
5. Predict on test dataset (subset of training dataset from competition)
6. Predict on test dataset from competition and compare area under the curve to previous test dataset

The intended solution to the problem is to use machine learning models to get a ROC area under the curve that is higher than the logistic benchmark model.

As a note, I inadvertently used a Gradient Boosting Machine instead of doing the xgboost. At this point in the project, it is too late to begin building the xgboost so I will stick with the GBM.

Metrics

The metric that I will use to measure the performance of the models that I will build is the ROC area under the curve. According to Wikipedia, this "curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings." (https://en.wikipedia.org/wiki/Receiver_operating_characteristic). The greater the area under the curve of the ROC, the better the model is at predicting the outcome.

The main reason I am using this metric is because the Kaggle competition used this metric as measurement of how well models predicted the outcome. This will make it possible to compare my model performance to that of other solutions submitted to the contest.

II. Analysis

Data Exploration

The features are all right skewed heavily. Several of the features are zero inflated, meaning that the distribution of the feature is dominated by values of zero. In order to introduce some variance, I will transform these variables. I will do a Yeo-Johnson transformation on the features to try to smooth out the distributions. This will be useful since many of the values of the features are zeros.

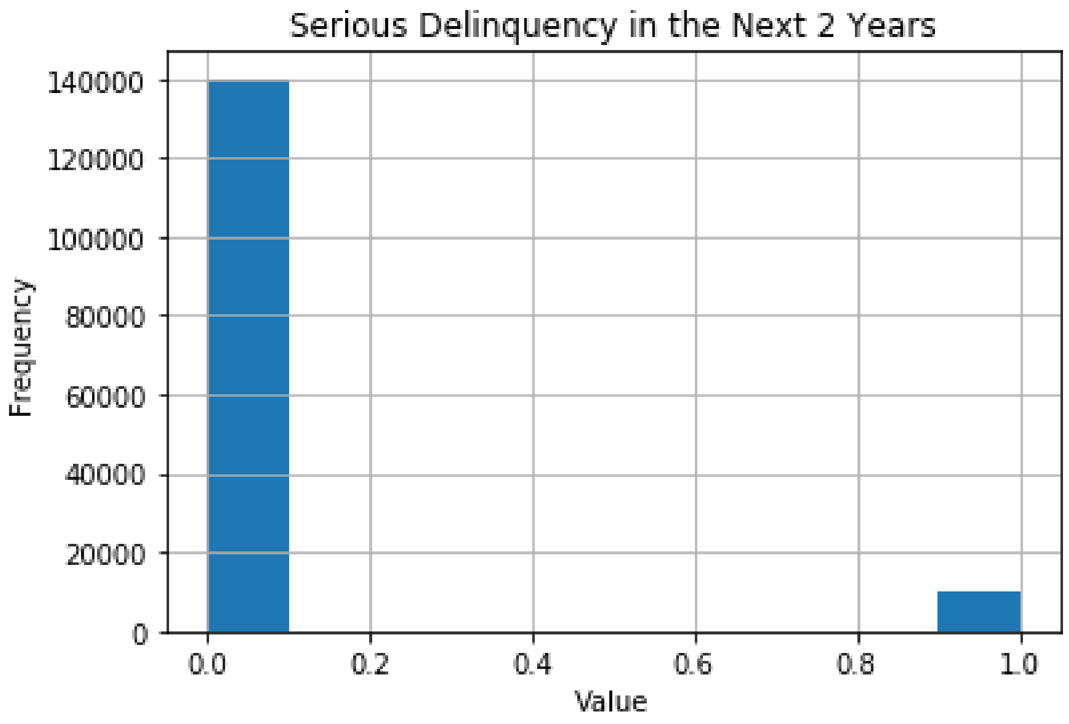
All eight features are quantitative, no categorical. The only variable with missing values is the Monthly Income variable. Since this variable is pretty heavily skewed right, I will use the median to impute missing values. Using the mean to impute values would insert values that are probably much higher than what the actual missing values are.

Because SVM is affected by the scale of each feature and Decision Trees are not, I will be using MinMaxScalar on all of the features. This will normalize the features in relation to each other.

I will use the Interquartile Range to determine if points are outliers.

Exploratory Visualization

The visualization below is a histogram of the values for the target variable, Serious Delinquency in the Next 2 Years. The reason I chose this visualization is to highlight the class imbalance in the target variable. Out of the 150,000 records given, only around 10,000 are positive, or about 7% of the examples. This could present a real problem in training models on this data, as I do not want my models to just predict a zero value all of the time. As discussed previously, I will use SMOTE to counteract this attribute of the target variable.



Algorithms and Techniques

For this problem I will use Random Forest, Support Vector Machine, and Gradient Boosting Machine models to solve the problem. These methods should be appropriate for the problem as these models are able to do classification and predict probability of classes.

Random Forest and Gradient Boosting Machines are both ensemble methods. This means that they build multiple models and unify the results of the many models to come to a decision about the outcome.

A Support Vector Machine works by maximizing the distance between margins and the data.

Benchmark

The benchmark that will be used as the baseline for this problem is a logistic model. This is a suitable benchmark as the logistic model is fairly simple, can be run quickly, and it can do classification. The model was submitted as part of the Capstone Proposal. For this model, the only preprocessing of data that I did was remove the 'age' and 'dependents' variables, as I thought it unfair to use those variables to assess credit worthiness, and I imputed missing values using the mean of each column.

When I built the logistic model, I calculated the ROC area under the curve using the model predictions and the test dataset. The model scored a value of 0.654. This will be the value that I will attempt to beat using the Random Forest, Support Vector Machine, and Gradient Boosting Machine models.

III. Methodology

Data Preprocessing

I started off the preprocessing by imputing missing values based on the median. I used the median since the distribution of the eight financial variables I used was heavily right skewed for all. The next step was normalizing the data with MinMaxScalar. This was done because Support Vector Machines can be affected by the scaling of variables and not all of the variables had the same scaling (for example: Debt Ratio vs Monthly Income). Trees are generally not affected by changing the scaling of variables so it is reasonable to use this data for all of the models.

To address outliers, I identified outliers using the Inter Quartile Range (IQR) for each variable. I decided that any point that was an outlier in at least half (four) of the variables was a candidate to be removed from the dataset. This resulted in 518 points being removed from the data.

The last step of preprocessing was balancing the class imbalance of the target variable using the SMOTE method.

Implementation

Three machine learning models were implemented in this project: a random forest classifier, a support vector classifier, and a gradient boosting classifier. For each method I started off by building the model with the default hyperparameters and then finding the ROC area under the curve. Then, for Random Forest and Gradient Boosting, I did variable importance and shortened the dataset down to the top four most important variables. I then adjusted one hyperparameter for each method three times to come up with three more models. I will select the best model for each method based off of the four fitted models.

I had originally started off doing grid search with Gradient Boosting to find the best model but it took much longer than I had time for. All of the code for this process is included with the submission.

Refinement

In order to refine the models, I originally decided to do a grid search using cross validation. However, when implementing this method for the GBM it took much more time than anticipated. I switched to building an initial model by using the method with the default hyperparameters. I then built three more models by adjusting one hyperparameter. Also, for random forest and GBM I removed half of the variables based on variable importance.

For GBM my initial model had a ROC area under the curve of 0.767. The hyperparameter that I adjusted was max depth. The following is the result of each max depth value:

Max depth= 5, ROCAUC = 0.707. Max depth= 10, ROCAUC= 0.672. Max depth= 20, ROCAUC= 0.651

For the random forest my initial model had a ROC area under the curve of 0.635. The hyperparameter that I adjusted was max depth. The following is the result of each max depth value:

Max depth= 5, ROCAUC = 0.726. Max depth= 10, ROCAUC= 0.727. Max depth= 20, ROCAUC= 0.680

For the support vector machine my initial model had a ROC area under the curve of 0.783. The hyperparameter that I adjusted was the kernel. The following is the result of each kernel value:

Kernel= Linear, ROCAUC = 0.751. Kernel= Poly, ROCAUC= 0.770. Kernel= Sigmoid, ROCAUC= 0.675

Unfortunately, because I randomly guessed which hyperparameter value is best, I am guilty of using information from the test set to inform the hyperparameters, and thus, the model. I understand that the remedy to this is cross validation. As stated previously I would have liked to have used cross validation with grid search but there was simply not enough time to do this for all three algorithms. This is something I will comment on later when I discuss project improvements.

IV. Results

Model Evaluation and Validation

The final model that performed the best was the Support Vector Machine with the rbf kernel. This model was selected because it had the highest ROC area under the curve: 0.783. This was higher than any of the other models I fit and it also beat the benchmark ROC area under the curve of 0.654. I was surprised that the SVM performed better than the GBM, as GBM seems to be a popular choice for Kaggle competitions in the form of xgboost.

I performed a sensitivity analysis by running the SVM final model on five different input datasets. These datasets were generated using the same train_test_split command as before, but the random state was changed (in this case it was 0, 1, 2, 3, and 4). As will be shown visually later in the paper, the model held up well under multiple trials, with a range in ROC area under the curve of 0.012. With this small difference, I consider this model to be robust enough for the problem and that results from the model can be trusted.

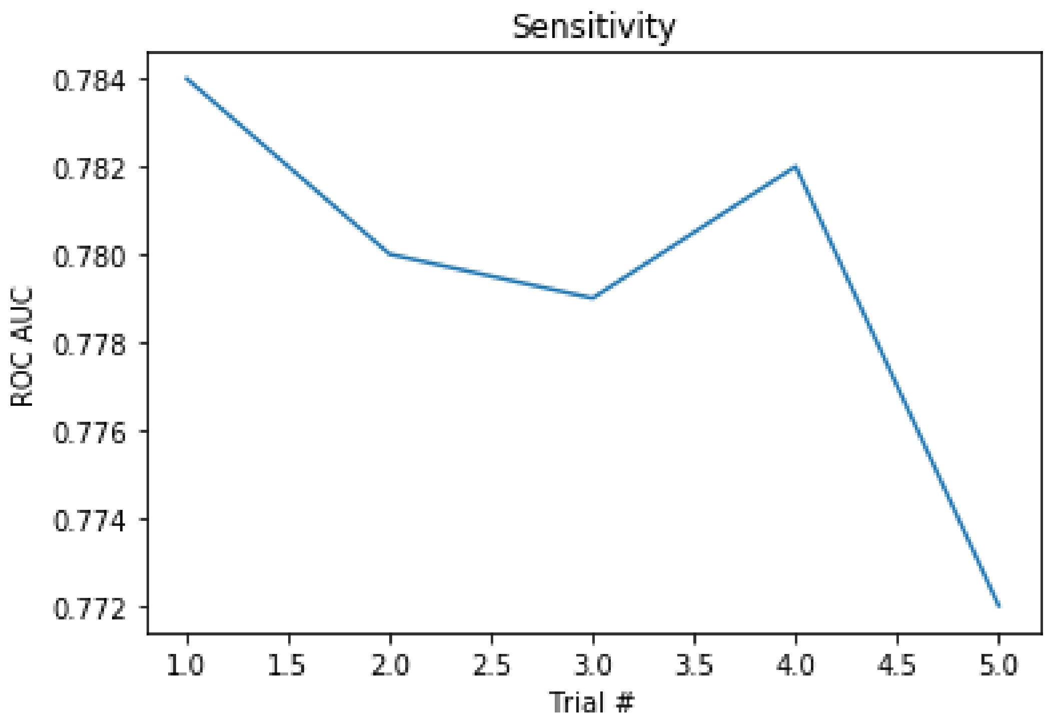
Justification

The support vector machine model with the rbf kernel performed better than the benchmark because it had a higher ROC area under the curve, which was the chosen measure to use to compare models. The SVM had a ROC area under the curve of 0.783 while the logistic benchmark had a ROC area under the curve of 0.654.

V. Conclusion

Free-Form Visualization

I chose to visualize my sensitivity analysis for my final model. I decided to show case this aspect of the project because of how little the ROC area under the curve varied by trial. The range between trials is 0.012, which is small compared to the difference of 0.129 between my final model and the benchmark logistic model. These trials mimicked the process of creating the SVC model the first time, but changing the random state when partitioning the data between training and test datasets. I also included the SMOTE adjustment to make sure the target classes were balanced. As can be seen by the sensitivity analysis, this is a fairly reliable model.



Reflection

This project started off with data exploration and preprocessing. After looking at the data and the basic statistics I decided that some variable transformations needed to happen. Also, after choosing the support vector machine as one of the algorithms I was going to use, I realized that I needed to normalize the variables so that scaling was not an issue.

I decided to do a Yeo-Johnson transformation on the data after filling in the missing values with the median of each variable and normalizing the data using the MinMaxScalar. I identified outliers using the interquartile range and removed 518 outliers based on if they appeared in at least half the variables as outliers. I used SMOTE on the target variable and features to balance the classes of the target.

After the data was preprocessed, I started modeling the data. I started off with the GBM in a grid search using cross validation. However, after running the code I realized I would not have enough time to finish the project if I used this method! I decided to take the less efficient path of guessing hyperparameter values randomly and this took significantly less time. I built an initial model using the default hyperparameters and then I built three more models using the same algorithm while adjusting one of the hyperparameters.

After building all of these models I was surprised to see that my Support Vector Machine with the rbf kernel did the best. This was probably the most interesting aspect of the project. I was expecting the GBM to do the best, especially since it seems to be a favorite for Kaggle competitions. I also expected GBM and random forest to do better than SVM because they are ensemble methods. However, when I did hyperparameter adjusting for the tree methods I did only use half the variables based on variable importance.

This was an insightful project. I learned a lot and I was pushed outside of my comfort zone. The preprocessing was difficult because it required a lot of data manipulation, which I am becoming better at in python. Also, there were some issues with downloading packages, but in the end, everything worked out.

I believe that this model would do well in a real-world setting. However if I were running a business based on lending credit, I would definitely take the time to do a grid search with cross validation to make sure I was using the best model I could to determine if creditors are going to be able to pay back the loans.

Improvement

If I had more time, I would have liked to have used grid search with repeated cross validation so that I could find the most robust, optimized model for each algorithm. Additionally, I would have liked to use the entire dataset for random forest and gradient boosting when refining the models. It would have been nice to use xgboost and to see how that might have done.

If I had more time I would like to implement two more things: first, I would have liked to have built a function that would build the models and tune the hyperparameters the way that I did without having to do copy and paste three times; second, I would like to have compared my results with the test results on Kaggle so that I could see how these models did compared to the general population. I imagine it is something I will do once I have completed this project.