Capstone Project 1: In-depth Analysis (Machine Learning)

In my first Capstone Project, I am trying to build a model that would predict “Payment in Full” on leases provided through the Lending Club platform. From the loan\_status variable I generated a binary response variable called Perf (short for Performance) that takes one of two values: either a 1 for Good Performance (Status indicating Paid In Full) or a 0 for Bad Performance (Status indicating ChargeOff or Number of days Late). Since the response variable is a binary variable, I decided to build Classification models, and as such I decided to build a Logistic Regression model, a Random Forest model, an XGBoost model and a Gradient Boosting model. My hypothesis is that the XGBoost model would outperform the other 3 out of the box, or maybe with some hyperparameter tuning.

In a first step, I looked at the variables in the Lending Club data set to try to get an idea as to potential correlations between the independent and dependent variables.

While in my “Applying Statistics to Capstone Project 1.ipynb” Jupyter notebook I generated a quick bare bones model using a Random Forest Classifier to look at feature importance and feature correlations in an attempt to find the most relevant features for future model building, here, in my “Capstone Project 1 – In-Depth Analysis.ipynb” I instead went through the process of thoroughly preprocessing variables to be able to do Recursive Feature Elimination and finally model building where I built several Classification models and tuned a couple for model accuracy improvement.

Building on the previous work I did, after loading the three input files and merging them properly, generating a response variable, removing merge keys and other variables identified through Pandas Profiling as categorical columns that are unique, entirely missing, rejected, unsupported, with high cardinality, date fields, I imputed missing values for categorical variables with ‘Unknown’ and then Label Encoded them.

I then got rid of records with no value for my response variable and generated a list of continuous numerical columns that I imputed with the mean, and a list of discrete numerical columns that I imputed with the median.

I then checked for potential removal of variables that should not be in the model. I found a few variables related to performance on the lease, which would not have been available at the time of application, so I removed them.

Now I built a function to automatically remove outliers based on three standard deviations away from the mean on both sides. I used the function by feeding it a list of variables to work on. This list consisted of the numeric variables. The function identifies the lower and upper limit to use to remove outliers for each variable that is fed to it from the list. It also provides a box plot with whiskers and a histogram for the before removal of outliers, as well as the after removal of outliers.

Following the removal of outliers, I removed correlated variables from the modeling data set.

I then used Recursive Feature Elimination with Cross Validation (RFECV) to be able to remove correlated variables.

After all the preprocessing above, I was then able to run the modeling process where I split my data between training and testing data where I used 20% of the data for testing.

1. I first built a bare bones LogisticRegression model for which I checked the accuracy and

roc\_auc\_score which were respectively 0.8989022798802487 and 0.8427205591748035.

1. Next, I built a quick RandomForestClassifier model for which I checked the accuracy and

roc\_auc\_score which were respectively 0.9037383894987334 and 0.8499914546449344.

1. Next, I built an XGBClassifier model for which I checked the accuracy and roc\_auc\_score which

were respectively 0.9048130805250634 and 0.8566023697931192.

1. Next, I built a GradientBoostingClassifier model for which I checked the accuracy and

roc\_auc\_score which were respectively 0.9048130805250634 and 0.8566023697931192.

Based on the results from these 4 classification models, I decided to tune the hyperparameters for the

two best performing: XGBClassifier and GradientBoostingClassifier. I went through a couple rounds of

tuning on each of these models.

On the first round of tuning for the XGBClassifier model, I tuned the learning\_rate and n\_estimators and

the results were:

* Tuned XGBClassifier parameters: {'learning\_rate': 0.1, 'n\_estimators': 200}
* The best score is: 0.9134681363456659

This first round improved the score from 0.9048130805250634 to 0.9134681363456659.

On the second round of tuning for the XGBClassifier model, I tuned the max\_depth and the results were:

* Tuned XGBClassifier parameters: {'max\_depth': 4}
* The best score is: 0.9141398329705319

This second round improved the score from 0.9134681363456659 to 0.9141398329705319.

On the first round of tuning for the GradientBoostingClassifier model, I tuned the learning\_rate and

n\_estimators and the results were:

* Tuned GradientBoostingClassifier parameters: {'learning\_rate': 0.1, 'n\_estimators': 200}
* The best score is: 0.9125085907843223

This first round improved the score from 0.904506025946112 to 0.9125085907843223.

On the second round of tuning for the GradientBoostingClassifier model, I tuned the max\_depth and the results were:

* Tuned GradientBoostingClassifier parameters: {'max\_depth': 5}
* The best score is: 0.9130843218041098

This second round improved the score from 0.9125085907843223 to 0.9130843218041098.

As shown above, the best performing model after hyperparameter tuning was the

XGBClassifier with a final score of 0.9141398329705319, which confirmed my hypothesis.