Project B Sassafras: XGBoost on simulated Data

Ivan Cao: 301316978
Team Name: Team Ivan

05 December, 2021

Contents

Introduction

The Sassafras data set contained simulated random data with no meaning, rather the project is gauging how well a machine algorithm can pick up its randomness and predict for the response variables. There are 75 covariates that are used to predict for one dependent variable. In total, the data set contains 203,287 entries, with 153,287 provided. Given this, I have deployed the XGBoost algorithm, as well as tuning it's parameters in hopes of obtaining the lowest possible mean absolute error possible.

Data Description

The information for our analysis was drawn from three datasets:

1. **XYtr (training data set):** contains 153,287 rows with 75 predictors.
2. **Ytr (training data set):** contains the response variable Y.
3. **Xte:** contains 50,000 rows with 75 predictors which is predicted for.

The project will consist of building a model that can learn from the Xtr data set and the Ytr data set before applying it to the testing data set. The prediction outcomes of Xte is then to be submitted on Kaggle where they will be ranked based on the lowest Mean Absolute Error (MAE).

Variable Selection and Data Preprocessing

In the training and testing data set, I replaced missing entries and 0's with the median of each column, this allowed for the data set to be fed through XGBoost as it can not accept missing values. Also, I have reduced the dimensions of the data set being fed through the model, originally the columns 67-75 contains all 9's, this has been removed completely from the model.

Data Preprocessing

These six variables will be used to predict the total price of the NFT. Before building our model, we must first preprocess the variables. Since the data sets also contain empty entries, we will fill the empty entries with 0. Also, the data types of the variables vary from dates, integers, floats, and strings, we will have to convert them to the same data type for it to be fed through the model. Since cdates, X.sales, and the fees can be viewed as quantitative variables, they can be changed to the float data type, whereas the version and the extension type are categorical variables are converted to dummy variables using one hot encoding. One hot encoding converts each categorical value into a new categorical column where it assigns a binary value depending on whether or not it exists. After converting the categorical variable, we avoid the dummy variable trap by dropping the last dummy column of each variable that was one hot encoded to avoid multicollinearity.

Considerations

Before deciding on using XGBoost, I have also considered these algorithms:

K-Nearest Neighbors Regressor (KNN): K-nearest neighbors was the first method I considered and implemented. K-nearest neighbors regressor is a non-parametric method that predicts for the response variables using the average of the neighboring results. I was able to get the MAE to peak at a low point of 13.05, but this was the lowest it could go down.

Multivariate Adaptive Regression Splines (MARS): Multivariate adaptive regression splines (MARS), which is commonly used to tackle non-linear data sets. The regression splines will account for different variables within a high dimensional data set. After running the algorithm, the MAE was the highest out of all the methods that I have tried. This is most likely due to the randomness of the simulated data since there are no patterns, making it difficult for MARS to assosicate a prediction with the predictor variables.

AdaBoost: Lastly, I implemented Adaboost, short for adaptive boosting which uses an ensemble method that reassigns weights of each instance according to the error of current prediction. With Adaboost, I was able to bring MAE down to 10.09

Scaling the Data: Usually when dealing with a large data set, I test whether scaling the data in the preprocessing step will improve the model's performance. Scaling the data is used to normalize the range of independent variables and/or features of the data. However, with this data set, MAE tends to increase if I scale it, so I decided not to for the final model.

Methodology and Results

We will conduct our analysis using the Python Programming language. For libraries, I used XGBoost, XGBoostRegressor, Scikit-learn, and Pandas to assist on data transformation and manipulation.

For the final model, I used XGBoost, it is a decision tree ensemble method that uses gradient boosting to improve scores. It is a form of supervised learning that predicts for the values by combining estimates of weaker models.

Before applying XGBoostregressor onto the data, I first split the Xtr and Ytr data sets into its own training and test data sets, by doing so, I am able gauge the performance of the model before implementing the test data. I want to achieve the lowest mean absolute error on the training data set. By adding parameters to XGBoost, the model can better adjust for the data, I used the package GridSearch and the gridsearchcv function to run tests on how to improve the tuning. Gridsearch scans the data to configure for optimal parameters given the XGBoost algorithm. With XGBoostregressor on the training data set, I was able to achieve a low of 4.98 MAE, saving that model, I will be applying it to the test data set. After submitting onto the public leaderboard, this cut the original XGBoost with no added parameters from 9.96 to 8.59.

Conclusion and Remarks

In my analysis of the simulated data, XGBoost performed the best out of all the models given my methods of preprocessing the data. I believe that the Mean Absolute Error can be furthered lower if I preprocessed the text more and tuned the parameters further. I suspect that the XGBoost algorithm performed well on this set of data because it was able to convert the weaker learners by combining them, developing them into stronger learners for the algorithm, the core of boosting algorithm.