STAT 440 Module 2: Sassafras

The Unfittables: *Candy*, *Jade*, *Ryker*, *Neil*, *Leo*

1 INTRODUCTION

This module, "Sassafras", is a machine learning project that focuses on simulated data. The simulated data was done through X to Y mapping and contains 14 dependent variables (Y) and 61 independent variables (X). The dimensions of the data is 203,287 x 75, and is split into training and test sets. The goal of this module is to **predict the hidden targets for the Y-test data by analyzing the simulated training data with different machine learning techniques in an attempt to minimize the Mean Absolute Error (MAE)**.

Before conducting any prediction modeling, pre-processing of the data was done to identify outliers and missing-data. This is an important step in data exploration and likely had a major impact on accuracy. With our new pre-processed datasets, we were able to use machine learning techniques including multiple linear regression, k-nearest neighbors, random forest, support vector machine, and neural networks and chose the one that had the lowest MAE score. All of these methods were done either in R or Python.

The method that provided our team with the lowest MAE score, equal to **1.16114**, is **Random Forest**.

2 DATA EXPLORATION

Even though this module used simulated data so there are no real-life meaning behind the data, it is still important to explore the data-set in attempt to find patterns and peculiarities.

Some of the early exploratory methods we used include:

- dataframe.info(): datatype for each column and see if any column has non-null values
- dataframe.describe(): basic statistics on the dataframe
- dataframe['column'].unique(): all unique values in a given column

Through the above exploratory methods, we found some interesting patterns in the data-set:

- One column of data has the "object" datatype, and we found it contains "?"
- For each column in x_train, the maximum value is 7
- For some of the columns in x_train, the minimum value is -9
- Most columns have continuous data while a few have discrete data

Using what we found above as a guide, we then looked at the distribution of each x_train columns, and discovered some more interesting patterns:

- Many columns exhibit normal distribution, but have difference variances
- There are some unique distributions (e.g. left truncated normal, two different uniform distribution joined together)
- The max 7s and minimum -9s are "masking" some of the distributions since they are outliers and not part of the main distribution

We conducted similar exploratory steps with the y_train data-set and found similar patterns as x_train. This process helped us understand the data-sets better and also helped informing us of what we might want to do in our next step, data-preprocessing.

3 PREPROCESSING

Maximum 7s and minimum -9s: We decided to convert these into categorical variables because they are outliers within their respective column of data. Once they are converted, we replace the 7s and -9s with the column mean. Our idea was to separate the 7s and -9s from the rest of the data, but still keep as much information as possible in case they're informative to our models.

NaN values and "?"s: We first convered the question marks to NaN, then replaced all NaN values with the column mean.

The only exception to taking the column mean is for any column that has discrete values. For those columns, we randomly selected one of the discrete values within its set.

4 MODELS TRAINED

4.1 Multiple Linear Regression

First, based on the boxplot, we construct a simple linear model to do the prediction. We are trying to use multiple linear regression with 64 predictors. To achieve this, we created a loop function to construct 14 models that contain all 64 predictors. Then by checking the variable importance, we found none of the variables had zero value. The MAE and RMSE are not very large for the first 13 models, except for model 14.

Then the stepwise selection was used find out the subset of variables in the data set resulting in the best performing model. The AIC value was our criteria to to select model variable importance. Using this subset of variables, we reconstructed the model, but we did not observe any significant improvement in the MAE score.

If an explanatory variable is directly a linear combination of some other explanatory variables, then this situation is called perfect multicollinearity, which makes the parameters impossible to estimate. Our result does not show high multicollinearity since all variance inflation factors(VIF) were less than 5. We also tried ridge

regression, and it shows a similar result to the general multiple linear models.

4.2 K-Nearest Neighbors

The K-nearest neighbors is a non-parametric method used for classification and regression. It approximates the association between independent variables and continuous outcomes by averaging the observations in the same neighborhood.

We evaluated the best k value through both MAE and RMSE scores. We tested all k values from 50 to 800 and chose the one that gave us the lowest MAE/RMSE score. Based on our record, MAE/RMSE scores for Z01 - Z13 are all quite similar with different k values, except for Z14. When we uploaded to Kaggle, our public leaderboard scores were also similar independent from the k values we tried. For the Z02 column, its values are 0 or 1 except 7 or -9. We attempted a classification KNN for Z02 column but the result did not improve significantly.

While KNN method is appealing, we think it becomes impractical when the dimension increase. In our data set, we have 61 independent variables. Also, KNN is slow in real-time as it has to keep track of all training data and find the neighbor nodes. Although KNN is not our best choice, its result is acceptable.

4.3 Random Forest

Random forest is a machine learning technique that averages multiple decision trees with the goal of overcoming over-fitting of any individual decision tree. This is important for our module since overfitting would explain our training set, but not allow us to find patterns that generalize the entire dataset. Random forest is an "ensemble learning method" for classification and regression, meaning it outputs the class that is the mode of the classes output by individual trees.

The algorithm associated with random forest selects roughly 2/3s of the training, which is used to grow the tree. This is done with a random-sample-with-replacement sampling method. Then, the remaining 1/3 of the data is used to calculate a misclassification rate. Each tree gives a classification on the leftover data and the forest chooses the classification having the most "votes" over the entire set of trees. (Note: For binary variables, the tree with the most votes is selected, and for continuous variables, it is the average)

To tune the model, we first trained the data to obtain the *mtry* value, which is the number of variables selected at each split. Next, variable selection was done using the VSURF package/function in R, which uses three steps to obtain the best predicting variables for each dependent variable. Then, the random forest model was run on the 14 dependent variables individually.

We attempted two different Random Forest methods. One of them is with sklearn's RandomForestRegressor function. The best result we got from this method is **MAE of 1.22**. The next method was from the randomForest package in R, which provided an **MAE of 1.16**.

The downside to this method was not being able to utilize the entire training dataset. The parameters that needed to be tuned, along with the variable selection algorithm took a substantial amount of time to run. The way around this issue was to take a random subset of the training data which sped up computation time significantly.

4.4 Support Vector Machine

Support Vector Machine (SVM) learns the importance of each training data points in relation to representing the decision boundary between different classes. Only those training points that lie on the decision boundary matter, and they are called the support vectors.[1]

Scaling was applied to prepossessed data, as SVM is very sensitive to feature scaling. Next, we find the best parameters for C, gamma, and epsilon. In addition, we also tried various kernel functions such as "linear" and "radial". We couldn't get the results as the model took extremely long time to run, and due to time constraints we decided to not to use it eventually.

4.5 Neural Network

Neural Network (NN) is an algorithm that creates nonlinear function out of linear combination of variables. NN contains hidden layers and hidden nodes. The input of Xs are fitted into each hidden node with linear functions, while NN fits the component simultaneously. The aim of tuning the parameter of NN is to minimize the chosen accuracy score. The weight decay in NN is to prevent over-parameterized caused by nodes and layers.

In our module, we scaled the range of variables from 0 to 1 at the beginning. Since the data set was huge, we randomly selected 5000 rows of data for this process. We randomly selected some numbers as the hidden node and weighted decay to compare the results from different parameters and performed 5-fold cross-validation. We found that hidden nodes of 2 and weighted decay of 0.1 provided the smallest mean of MAEs. We repeated this process with hidden nodes of 1,2,3 layers and weighted decay of 0.1, 0.5, 0.05. The result still shows that hidden nodes of 2 and weighted decay of 0.1 was the best combination. When we fitted the NN model, we applied the full set of data and repeated it five times to get the smallest MAE score. In other words, we created five models for each fold, with each set of parameters. Out of all those models, we used the one with the smallest MAE score to predict the Y values. We achieved a MAE score of 1.17 on Kaggle.

We ran into some issues when we tried to tune our NN models. Due to our limited technique, running NN in R was extremely time-consuming. Hence, we were not able to try multiple layers and larger node sizes. This might be one of the reasons why NN is not the best predictive model for us in this module.

5 RESULTS

5.1 Best model

As seen above, many different methods were attempted to minimize MAE from the Sassafras dataset. Pre-processing the data to deal with outliers and missing data and getting together as a team to discuss results were crucial steps in this module. Our best model was done with a **Random Forest** algorithm, providing us an **MAE of 1.16114** on Kaggle.

5.2 Lessons learned

5.2.1 Teammate: Leo. From this module, I gained very good practical experience in doing data exploration and data preprocessing. I learned that data exploration is an ongoing process, not just at the beginning of a project. A good example from this module is that after doing data preprocessing, we went back and looked at all the column distributions again to see if the processed data present different distributions versus preprocessed data.

5.2.2 Teammate: Ryker. This module tested my machine learning, data manipulation, and patience. I have dealt with large datasets in the past, especially in my co-op position as a data analyst, however, since this dataset was unique in the fact that it had no real world implications, meaning you had to strictly worry about the computation and not about intuition. I learned a lot from my teammates, who completed a wide range of models in different languages. During our meetings, we were able to explain our thought processes, which undoubtedly will make me a better data scientist. I also learned a lot about computation time and how such a large dataset affects computation time between different models. This was definitely the biggest hurdle with this module.

5.2.3 Teammate: Candy. I have never imagined that the code we wrote would directly affect the time of execution until I got involved in this module. I obtained the experience of how to deal with such huge data set. Furthermore, I realized that different techniques and programming languages provide different outcomes to the process of model building. We should not be limited to our familiar languages but step out of the comfort zone and try to learn new knowledge.

5.2.4 Teammate: Jade. I have found out that R deals with large dataset a lot slower than Python. Maybe the reason is that I didn't find a more efficient way or package to do the tasks. Also the high dimension dataset provided extra complexity, as I may need to choose a different method for each variable prediction. Also, for the data processing part, I tried several new ways to deal with it. This dataset provided many opportunities to practice what we learned and how to deal with a large dataset, and it is quite different from Module 1. I need to find more precise direction to predict it.

5.2.5 Teammate: Neil. I learned how to train models in R, as I had never done it before. R took a lot of time to run models and I honestly think that python is preferred choice to train and test models since it takes much less time. I learned about neural nets from my teammates as they were really helpful in explaining the model. At first, I didn't understand the data as it was not a real world dataset so I couldn't make much sense of it but after spending time during prepossessing and team meeting I understood how to go about with this data and how to train models. This was an unique experience for me as I have never dealt with this kind of data before and I learned a lot about how to train different models.

REFERENCES

[1] Aurelien Geron. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems.* O'Reilly Media, Incorporated, 2019.