STAT 440 Module 2: Sassafras

Team 1: Justin Heer, LingXiang Zou, Ao Tang, Nathania Santoso

https://github.com/surreybro/STAT440-module2

1 INTRODUCTION

The problem we are tackling is accurately predicting the 14 response variables, Z01 to Z14, in which Z02 is categorical, and others are numerical, on a simulated dataset. Our chosen metric to measure success is the RMSE and misclassification rate of our predicted target variables[1].

The dataset we are working with is simulated by our instructor that has no meaning. It consists of 203287 data items, of which 153287 are provided in the training set, and 50000 are provided in the testing set. Each data item consists of 75 covariates divided into nine groups (columns of X) and one target (Y)[1].

Initially, we explored the dataset by printed out the summary tables of each variable and imputed the missing values to prepare the dataset for modelling. We implemented a variety of linear models that require less computation time and further analyzed the data with Random Forest, Neural network, and XGboost. As a result of our efforts, we conclude "model" is our best model, and the resulting Kaggle RMSE we achieved was **1.01064** on the public leaderboard.

2 PREPROCESSING

Prior to performing any machine learning on the dataset, it was required that we sift through the dataset and cleanse it of impurities and discrepancies. Our preprocessing steps included

(1) NAN Imputation

(2) Outlier Removal

(3) Feature Scaling

2.1 NAN Imputation Via Mean

To speed up our "time to trainable model" we decided to initially just impute missing values using the column mean. This was computationally efficient and simple to implement. Doing so allowed us to start iterating through models quickly, as a consequence of this we were the first team to generate predictions on Kaggle.

2.2 NAN Imputation Via K-Nearest Neighbors

Another approach we tried is using K-Nearest Neighbors to impute missing values. Each sample's missing values are imputed using the mean value from K nearest neighbours found in the training set. Two samples are close if the features that neither is missing are close. The distance function we used is the Euclidean distance that supports missing values to find the nearest neighbours. However, In our result, we found that impute missing values using the column mean has a better RMSE on Kaggle. Figure 1 (overleaf) shows a count of the missing/NAN values for the features that had missing values. It turned out only the "B" variables had missing data.

2.3 Outlier Removal

Basic outlier removal was attempted. We calculated the interquartile range(IQR) and removed outliers using the standard 1.5 times IQR method. However, upon doing so the resulting dataset had only 89825 training examples left of the original 15387. This seems too large of an omission to make, so outliers were simply included in our training dataset.

2.4 Feature Scaling

Training on a large dataset such as this requires an immense amount of time. Several research papers have shown that utilizing feature scaling can improve training times for a variety of models. To take advantage of this speedup we performed feature scaling on all feature columns using the Scikit-Learn Standard Scaler implementation[2]. This resulted in a noticeable speedup in our training times.

3 MODELS TRAINED

In order to maximize the number of models tested per response variable, we split the 14 response variables up amongst ourselves, with each group member responsible for three or four variables. This allowed individuals to focus on tuning their model for specific variables and explore a variety of models. A general paradigm we imposed on all models was using the correlation coefficient to select the best features to use in our models. Some of the models that our team explored are listed.

3.1 Partial Least squares

Partial least squares regression (PLS) is a technique that creates new variables that are linear combinations of the original and performs least squares regression on these components. PLS regression is especially useful when your predictors are highly co-linear or have more predictors than observations. We have tuned the number of components using cross-validation for reliable results. Due to the Sassafras dataset, we have way more observations than predictors, which cannot show the advantage of PLS. PLS is really meant for cases where we want to reduce the dimension of the dataset. Just as we expected, PLS did not perform any better than other linear models.

3.2 Linear Regression: Lasso

Lasso performs two main tasks: regularization and variable selection. Lasso applies a regularization or shrinkage process on the least squares, where it attempts to minimize regressors' coefficients.

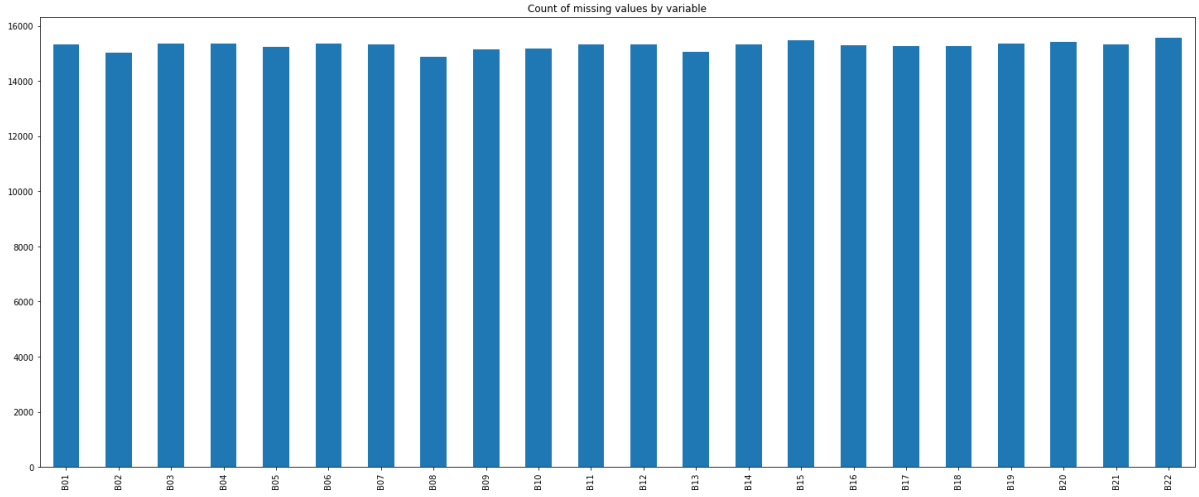

Figure 1: Count of missing values for variables that have missing values

The strength of regularization is controlled by the tuning parameter λ. Larger λ value implies more coefficients will shrink to zero. Non-zero coefficients after variable selection will be selected to be included in the model.

3.3 Linear Regression: Ridge

Ridge Regression is similar to linear regression with an added L2 regularization component. It penalizes predictions by using the square of the residuals plus a regularization term that uses the square of the variable coefficients. If tuned correctly this can reduce overfitting and allow Ridge Regression to generalize better. We used the glmnet package in R to perform this method. In order to train the model, we performed hyperparameter tuning to find the optimal regularization coefficient that minimizes the mean squared error. In order to find the best parameter value we used cross validation with 3 folds to tune the model.

3.4 ElasticNet

The idea of Elastic Net Regression is to combine the best features of both Lasso and Ridge Regression methods. In this method we have find the optimal value of two parameters to minimize the MSE. Those two parameters, λ and α, have to be tuned in order for us to have the best prediction with the lowest MSE. We once again used cross validation to tune the model and find the best optimal λ and α values. Having predicted our test set with Elastic Net, the mean squared error is slightly higher than the predictions of test set using Ridge Regression.

3.5 Neural Network

A Neural Network is a powerful tool used for regression and classification problems, they are exceptionally good at learning complex non-linear relationships between variables. The basic implementation in neural net function requires a lot of tuning, the three parameters we tuned are the number of layers, number of nodes, decay parameter. Due to time constrains we were only able to tune using a single layer, for the other parameters we used a grid of (2,4,6,8,12,17) and (0.01,0.1,1) for nodes and decay, respectfully.

3.6 XGBoost

XGBoost is a decision tree based ensemble machine learning algorithm, which uses a gradient boosting framework. Gradient boosting is an approach in which new models are created that predict the residuals or errors of prior models and then added together to make the final prediction[3]. This approach supports both regression and classification predictive modelling problems. In our dataset only 'Z02' is a classification problem, the rest are regression problems. So we trained an XGBoost classifier for 'Z02' and for the rest of the variables we use the XGBoost regressor to predict them. The main advantage to use XGBoost is it has builtin Lasso Regression (L1) and Ridge Regression (L2) regularization which prevents the model from overfitting. In order to tune the hyper-parameters, we use cross-validation (CV) to find the best parameter. We run cross-validation on our training dataset and use the RMSE to evaluate each parameter. Since our dataset is large we only set 3 folds to use for cross-validation. We tuned our parameters to minimize the MAE on the validation set, and then check the performance of our model on the test dataset.

3.7 Random Forest Regressor

The random forest regressor fits multiple decision tress on various sub samples of the dataset. It then creates an ensemble of these decision trees to improve the predictive accuracy and control overfitting. Random forests have been shown to not overfit easily (the creator of Random Forests argues that the model will never overfit, regardless of how many estimators/trees one uses), so they are a good contender for most datasets.

4 CONCLUDING REMARKS

After considerable training, we found that we achieved the best performance using XGBoost and a Neural Network, aside from response variable Z04. The main optimal parameter values are listed

in Table 1. Due to the size of the dataset, it was expected the Neural Networks would perform quite well, it was surprising to several team members that XGBoost outperformed Neural Networks for some of the response variables. We think that this could have been because we only used a single layer in our networks, which limited the complexity of the relationships the network could learn. If this project was conducted again, it could be advantageous to setup a GPU cluster on which a neural network could be trained more efficiently.

4.1 Lessons learned

4.1.1 Teammate: Carl. This module presented a challenge that the dataset is so large and complex. It becomes difficult in the process of analyzing using machine learning algorithms, which require an enormous time. To be more effective and efficient on this model, I learned that taking a random sample of 1000 observations works through the syntax problems before fitting a final model on the entire dataset. This is a good practice in general for machine learning to give me quick spot-checks of algorithms. Another lesson that I learned is that parallel processing that makes use of all of the cores in the computer in training. I improve my machine learning model generation's speed by approximately two times, which allows more valid Kaggle submission for the team.

4.1.2 Teammate: Justin. The central challenge for module 2 was the massive dataset; it truly encompassed the idea of working with big data compared to the previous module. This meant that we had to shift our strategies from the previous module and update our workflows to manage the increased training times. We decided to each train 3-4 response variables to split the workload between us. This allowed for each team member to iterate though more models for each of their response variables. Further this required us to write our scripts carefully, else a large amount of time could be wasted .

4.1.3 Teammate: Nathania. I find that the dataset for this module is huge, which I have never experienced before. I learned how to analyze the real definition of 'Big Data' (even though it is just a simulation). Moreover, I encountered several difficulties with the long run time as well, since it is a really big dataset. Overall, it is a great experience learning about analyzing big dataset with different methods of machine learning techniques.

4.1.4 Teammate: Ao. The dataset for this module is very large, and we need to predict 14 variables for this task. We have tried many different models and we found each model has a different performance. I learned that there are no best algorithms in machine learning, we must test all possible algorithms for data to find which one has better performance. Additionally, not only test the right algorithm but also need the right configuration to tune the hyper-parameters. Moreover, We need to consider some other factors for each algorithm such as computational complexity, explainability, and ease of implementation.

REFERENCES

[1] URL: https://www.kaggle.com/c/stat440-20-module2/overview.

[2] URL: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html.

[3] Jason Brownlee. *A Gentle Introduction to XGBoost for Applied Machine Learning*. https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/. Online; August 17. 2016.

Response variables	Models	Tuning parameters
Z01	XGBoost	$\lambda = 5$, $\gamma = 5$, max depth=400
Z02	XGBoost	$\lambda = 100$, $\gamma = 10$, max depth=1200
Z03	XGBoost	$\lambda = 25$, $\gamma = 5$, max depth=400
Z04	Random Forest	n estimators=70, max depth=20
Z05	XGBoost	$\lambda = 55$, $\gamma = 8$, max depth=200
Z06	Neural network	Hidden nodes=2, decay=1
Z07	Neural network	Hidden nodes=8, decay=1
Z08	XGBoost	$\lambda = 20$, $\gamma = 5.5$, max depth=400
Z09	XGBoost	$\lambda = 200$, $\gamma = 8.5$, max depth=200
Z10	Neural network	Hidden nodes=2, decay=0.0106
Z11	Neural network	Hidden nodes=8, decay=0.0109
Z12	Random Forest	n estimators=90, max depth=50
Z13	Neural network	Hidden nodes=17, decay=2
Z14	Neural network	Hidden nodes=16, decay=0.5