Statistical Modeling III
Project 2020

James Fricker - a1706059

Contents

Executive Summary

Mathematical modelling is often used for prediction of unknown quantities. In this report, mathematical methods are used to predict the quality of different Portuguese "Vinho Verde" wines using a set of predictor variables.

In order to do so, the XGBoost (eXtreme Gradient Boosting) machine learning algorithm is used. This package uses a variant of a Gradient Boosting Decision Tree algorithm to model complex data.

When attempting to predict wine quality, the model returned 0.254 and 0.393 mean squared error for two test sets. This outperformed both a simple linear regression model and a simple random forest.

1 Introduction

In social circles, types of wine and their quality can receive extensive debate. In order to settle this debate, mathematical methods can be used to provide an objective view of wine quality and wine rankings. In this report, machine learning models will investigate various dependent variables in order to find the best way to predict wine quality.

First, the data will be investigated, then method used to model the data will be outlined and the code used will be explained.

The aim of the report is to produce the model that can most accurately predict the quality of wine given a set of predictors.

2 Background

In order to produce the best model, a first step is to gain insight into the data being analysed. The data-set contains red and white variants of the Portuguese "Vinho Verde" wine. Included in the data set are the following variables.

- type
- fixed.acidity
- volatile.acidity
- citric.acid
- residual.sugar
- chlorides
- free.sulfur.dioxide
- total.sulfur.dioxide
- density
- pH
- sulphates
- alcohol

The final model will use these twelve predictors to predict values for wine quality, which is also contained in the set.

Plots of each variable are shown in Figure 1. Important aspects to note are that the quality is between 3 and 9, with the majority of points in the 5-7 range. Other variables seem to be fairly kindly distributed with no noticeable outliers.

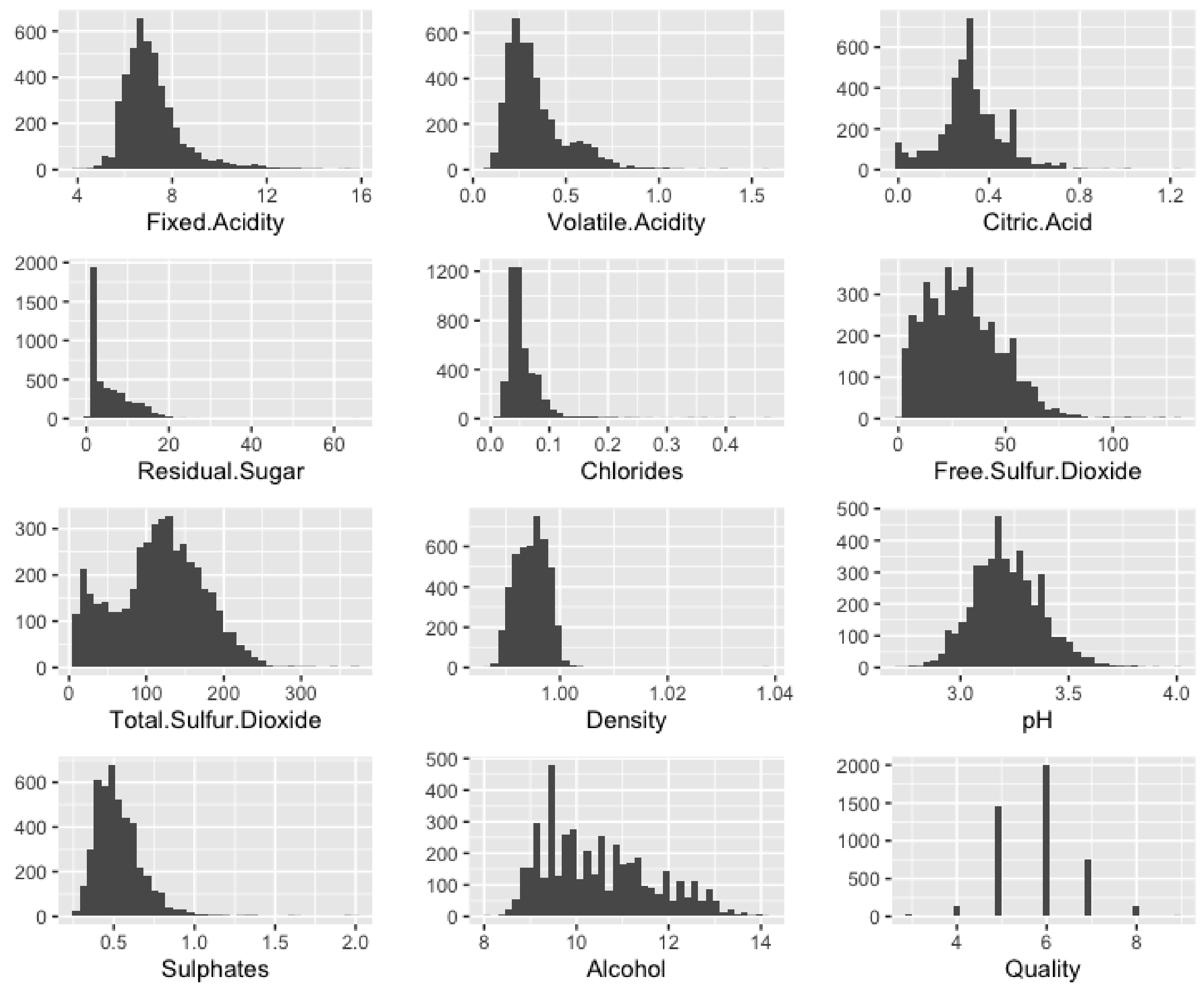

Figure 1: Variable Plots

Typically, before a model is selected, there will be a number of processes undertaken. These are things like data cleaning and data normalisation. Fortunately, in this data set, there are no null values, and the data has been normalised to a satisfactory level. This also means that there are no changes that need to be made to the data before the models can be applied.

3 Methods

3.1 XGBoost Background

The selected method for modelling the data was the machine learning algorithm called "XGBoost"[3]. XGBoost is an eXtreme Gradient Boosting algorithm. Since being officially released in 2016, on the popular machine learning site "Kaggle", XGBoost has been the al-

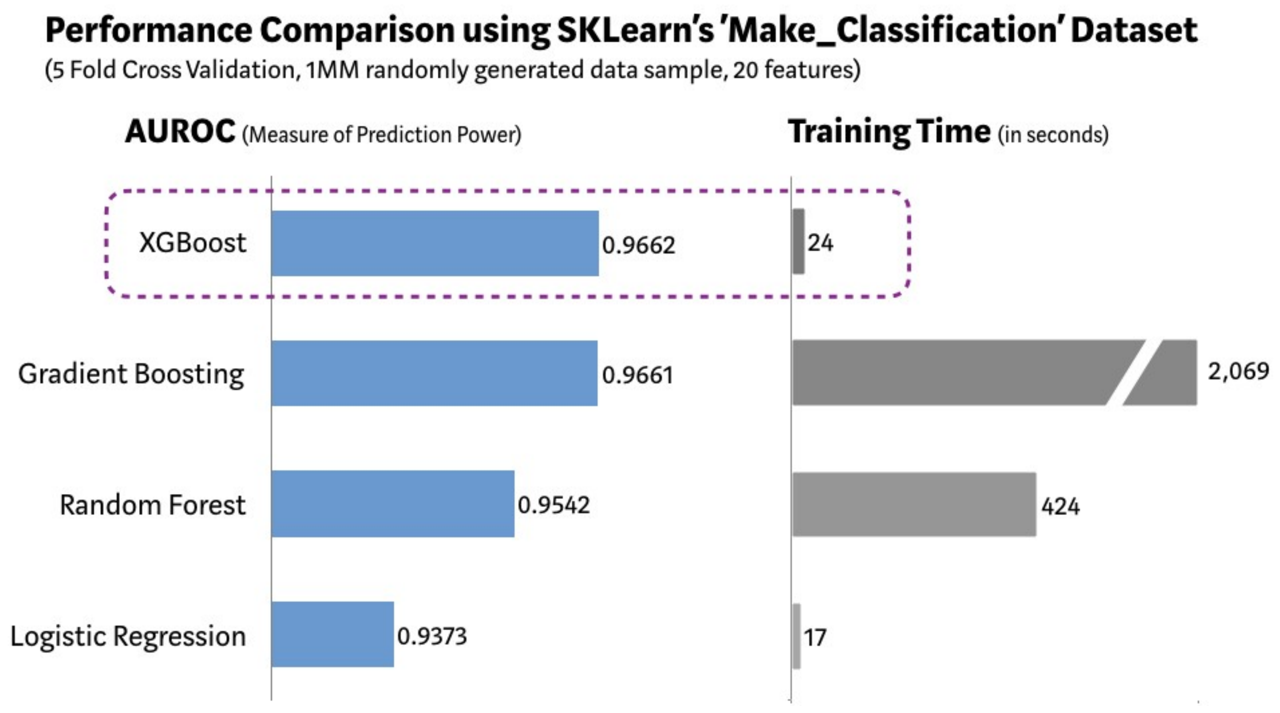

Figure 2: XGBoost vs. Other ML Algorithms[4]

gorithm of choice for a number of victories[2]. Among the 29 challenge winning solutions published at Kaggle's blog during 2015, 17 solutions used XGBoost in some capacity.

XGBoost has two major advantages, execution speed and model performance.

These can both be seen in Figure 2. XGBoost performs much better than other similar algorithms. The reasons for this include both modelling and software optimisations.

The explanation of XGBoost starts with an ensemble model. An ensemble model works by assigning the final score based on the sum of basic models. An example of this is seen in Figure 3. In the figure, each person receives a score based on the two basic decision trees. It's important to note that the trees aren't fully independent, in fact, the trees attempt to complement each other to provide a better estimate.

The next step after creating the tree ensemble is to understand that there are different ways for each tree to be trained. In the case of Gradient Boosting, each additional tree is based on the current set of trees and seeks to minimise an objective function.

The objective function contains information on the current error, as well as the model complexity. This trade-off between error and complexity is known as the bias-variance trade-off. A model may fit the data well but also be too complex and show signs of over-fitting. The best models are those that fit well with as little complexity as needed.

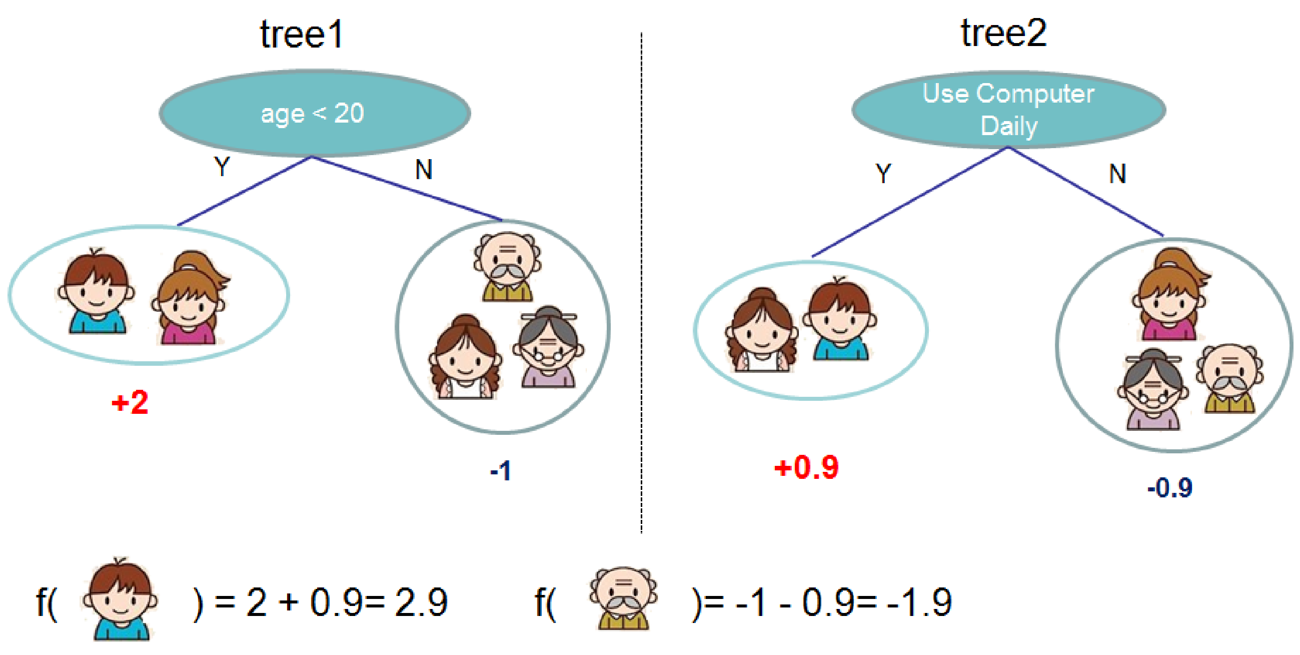

Figure 3: Ensemble Model[1]

Minimising the objective function is done by finding the second order Taylor series expansion, and then adding the tree that minimises the objective function estimate to the model. By adding more trees in this way, the error is being gradually reduced and the model is effectively 'descending' to a solution.

3.2 XGBoost in R

The XGBoost library in R provides access to the XGBoost modelling tools using the function xgboost. An example function is below. One important aspect of the function is that the input data must be numerical. Categorical variables must be converted to a numerical equivalent before training can occur.

```
xgboost(data = NULL, label = NULL, missing = NULL,
params =list(),nrounds, verbose = 1,
 print.every.n = 1L, early.stop.round = NULL,
maximize = NULL, ...)
```

The process of selecting the best variables for the model is known as model tuning. In the case of XGBoost, there are a very large number of parameters, which makes it difficult to find the optimal set of parameters for the data. In order to find a good set of parameters, a initial selection was made, and the improvements on each individual parameters were made separately. The parameter description and it's final selection are found below.

max.depth=18

Indicates how many levels there can be in a tree. Increasing this means increasing the complexity of the model and therefore also the risk of over-fitting.

colsamplebytree = 0.6

This variable indicates how many variables should be considered for each tree. The selection of 0.6 indicates that for each tree, a subset of 60% of the possible variables will be used.

subsample = 0.8

Subsample refers to the amount of data to be used in each boosting iteration. In the case of 0.8, it means that for each round, 80% of the data is being used to train the tree. This feature also removes some need of cross-validation since not all of the data is being used at one time to train the model.

eta = 0.01

ETA is the step size shrinkage used in update to prevents over-fitting. A larger value reduces the probability of over-fitting, and a smaller value means a more accurate fit to the data. The value of 0.01 was found to perform similarly to smaller values and thus is preferred.

nround = 2200

This is the number of decision trees in the final model.

evalmetric = "rmse"

This is the method for measuring the objective function. It is the default for linear prediction models.

objective = "reg:linear"

Indicates that the model will be a linear predictor. Another possible choice for this could be the classification method, however since the model was not extremely accurate, the classification method led to higher MSE results. Therefore the linear prediction method is used.

booster = "gbtree"

Determines what the model is made from. gbtree means that a gradient boosted tree

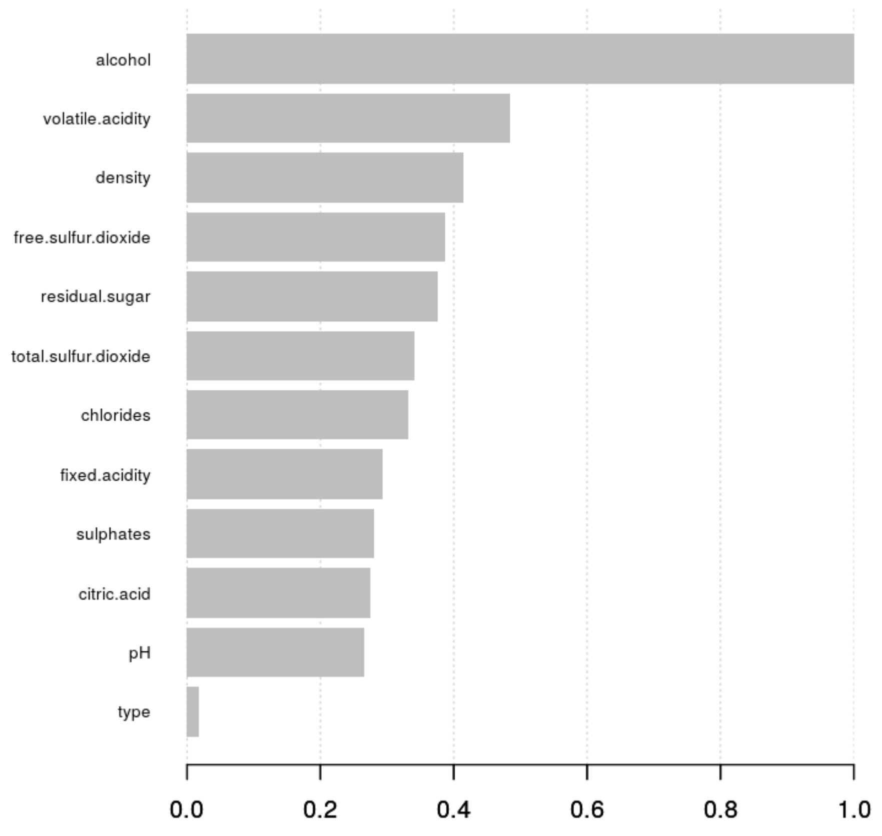

Figure 4: Variable Relative Importance

will be added at each round.

nfold=5

Completes a 5 fold cross validation as part of the training. The best model from the cross validation will be taken as the final model.

There are even more parameters available to be modified[1], however these were not considered due to the added complexity. The final code used to create the model can be found in the appendix.

Figure 4 shows the relative importance of each variable in the model. Type is the least important variable by a significant margin. This is likely because features of a certain type are also reflected in other variables which makes the type variable itself less relevant. Alcohol is the most important criteria for determining the wine quality.

4 Results

As part of the project, training and practice data sets are provided. In order to test the performance of the model, an additional test set was created by partitioning the training set. This was also necessary due to the size of the practice set. The training set has 4500 rows, while the practice has only 100. This ratio of testing to training is not suitable, it should be closer to 80% training and 20% testing to make sure that the model fits correctly. To do this, the training set was split into 80% training and 20% testing. The practice set was used as a further check for the model quality.

Cross validation was not used in a separate function as it has been included both as part of the xgboost function, as well as being a property of each tree in the model.

The metric that the final model will be graded with is the Mean Squared Error (MSE). This is the metric that will be used to validate the success of the model. The MSE is given by the formula $mean((y-p)^2)$ where y is the correct wine quality and p is the models prediction of the quality.

When bench-marking the scores of the model, a simple linear model can be used, as well as a simple random forest. This will provide more insight into how well the model is performing.

When testing on the practice set provided, the XGBoost algorithm achieved a MSE of **0.254**. This is in comparison to the simple linear model which achieved 0.496 and the random forest with 0.290.

When testing on the larger test set, the XGBoost model achieved a MSE of **0.393**. This, again, in contrast to the linear model of 0.526 and the random forest at 0.406.

5 Conclusions

XGBoost was found to have lower MSE than both the simple linear model and a simple random forest. This indicates that the selection of XGBoost in modelling the data was a good choice.

The model could be further improved by considering transformations of the data that allow for more effective modelling such as Box-Cox transformations. Other improvements could be using a neural network with the XGBoost algorithm as an input.

6 Appendix

6.1 XGBoost Training Code

Final function to train xgboost algorithm.

```
gb_dt <- xgboost(data = as.matrix(train_x), label = train_y,
                 max.depth = 18,
                 colsample_bytree = 0.6,
                 subsample = 0.8,
                 eta = 0.01,
                 nround = 2200,
                 eval_metric = "rmse",
                 objective = "reg:linear",
                 booster = "gbtree",
                 gamma=0,
                 nfold=5,
                 verbose = FALSE
                 )
```

6.2 Predict.Quality Function

```
# load library xgboost
library(xgboost)
# install.packages('xgboost')

# load xgboost model from file
bst <- xgb.load('xgb.model')

predict.quality <- function(x){
  # convert file to matrix (changes type to numeric)
  dt <- data.matrix(x)
  dt <- data.frame(dt)
  # predict data using model
  p <- predict(bst,xgb.DMatrix(as.matrix(dt)))
  return(p)
}
```

References

[1] URL https://xgboost.readthedocs.io/en/latest/parameter.html.

[2] URL https://github.com/dmlc/xgboost/tree/master/demo\#machine-learning-challenge-winning-so

[3] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. *CoRR*, abs/1603.02754, 2016. URL http://arxiv.org/abs/1603.02754.

[4] Vishal Morde. Xgboost algorithm: Long may she reign!, Apr 2019. URL https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d.