Predicting Wine Quality Using Regression Trees and Linear Regression

SENG 474 - Data Mining Project

Geoffrey Harper - V00866723

Nicholas Roethel - V00868351

Mark Xu - V00825497

Table of Contents

Abstract

Wine quality can be predicted as a result of various factors. In the dataset we used for this project, it consists of aspects such as different kinds of acidities and chemical component indexes. We are able to find the relationship between these factors and the final quality, by running both wine datasets in two data analysis models followed by a prediction for quality based on the given features. This report covers the preprocessing of the data, two prediction models, and a comparison between the results. A regression decision tree and a linear regression were implemented to make predictions. The differences between the two and the results were used to determine which was superior in predicting the quality of wine based on the dataset. Our research found that the regression decision tree model obtained both a higher prediction accuracy and a lower mean average error, when compared to the linear regression model.

Introduction

The purpose of this report is to compare how our team's implementation of a canonical linear regression model performs in comparison to a regression tree model for the given dataset. First we will give a brief introduction to the background of the project. This will be followed by an explanation of the implementation for each algorithm. Next the findings will be presented, and the performance of the two algorithms will be prepared. Finally ideas for future work will be presented.

For our teams project we chose to look at the The UCI Machine Learning Repository's Wine Quality dataset [1]. This data set was chosen due to it's high quality (not much data cleaning was necessary and it is large enough for most prediction algorithms), and because our team was personally interested in what features made a high quality wine. The purpose of this data set is to predict wine quality based on physicochemical attributes, such as alcohol content and density. The dataset contains 4898 entries and

is a very common dataset for machine learning. Previous work that has been done with the dataset includes using it for the k-nearest neighbour algorithm, the random forest algorithm, the support vector machine algorithm [2] and also various regression techniques [3].

For this project we decided that we wanted to implement two different algorithms and compare their performance in predicting the wine quality. The implementation of these prediction algorithms was chosen to be the primary focus of the project, and the algorithms were implemented using python rather than using built in library features from sklearn. For the classification algorithms we chose to do a linear regression model and a regression tree model because they are both very suitable for our dataset, since it is a multidimensional non-binary numerical classification. However, for feature reduction, data visualization and metric calculation, some premade library functions were used. In order to run the project, it is expected that the user has python3 installed with common data science libraries (specifically pandas, numpy, sklearn, seasborn, matplotlib).

The metrics chosen to compare the two algorithms were accuracy and the mean absolute error. This allowed for us to compare both how often each algorithm was correct, and how close the algorithms were to predicting the right value on average.

Rather than using a Jupyter Notebook, our team chose to use GitHub as it made it easier for us to all work on the project simultaneously. Our GitHub repo is public can be found at: https://github.com/nicholasroethel/data-mining-project/.

Data Preprocessing

Because our datasets had no missing values anywhere not much data cleaning was needed. Furthermore, because both of our models were regression models, no data

standardization was required. However, in order to increase the performance of the chosen algorithms and to create data that was easier to visualize we decided to reduce the dimensions of our data. This also provided an insight as to which attributes were most important in predicting the quality of wines.

When we originally downloaded the dataset it had been split into two parts, entries for red wines and white wines. Initially, we intended on combining the two datasets, as we thought it would be easier to deal with as one single dataset rather than two. However, after completing the feature selection, it became obvious that the quality of red and white wines depend on different factors.

The dataset started with 11 attributes:

- Fixed acidity
- Volatile acidity
- Citric acid
- Residual sugar
- Chlorides
- Free sulfur dioxide
- Total sulfur dioxide
- Density
- pH
- Sulphates
- Alcohol
- and the target attribute: Quality

For our feature reduction we decided to implement recursive feature elimination. This algorithm checks to what extent each attribute is capable of predicting to target on it's own, and ranks them based on their capability. This algorithm was implemented using

Sklearn's RFE [4], LinearRegression and Preprocessing libraries to create an algorithm that recursively eliminated features through the use of a linear regression model. At the end of our algorithm, a condensed csv file with the remaining attributes was produced for both the red and white wine datasets. These condensed 3 feature (and 1 target) datasets are what was used for the remainder of the project.

After running the RFE algorithm on both the red and the white dataset, we were surprised to see that the remaining features for the two differed. For the red wines the 3 features selected were:

- Volatile Acidity
- Sulphur
- Alcohol

Whereas for the white wines the 3 features selected were:

- Volatile Acidity
- Residual Sugar
- Density

As mentioned previously in the report, since red and white wines only shared one attribute between their chosen attributes, it was chosen to keep them seperate.

In order to run the RFE algorithm, one must simply clone our git repository and run "python3 featureSelection.py" to generate the reduced datasets.

Data visualizations after preprocessing (4 attributes ONLY):

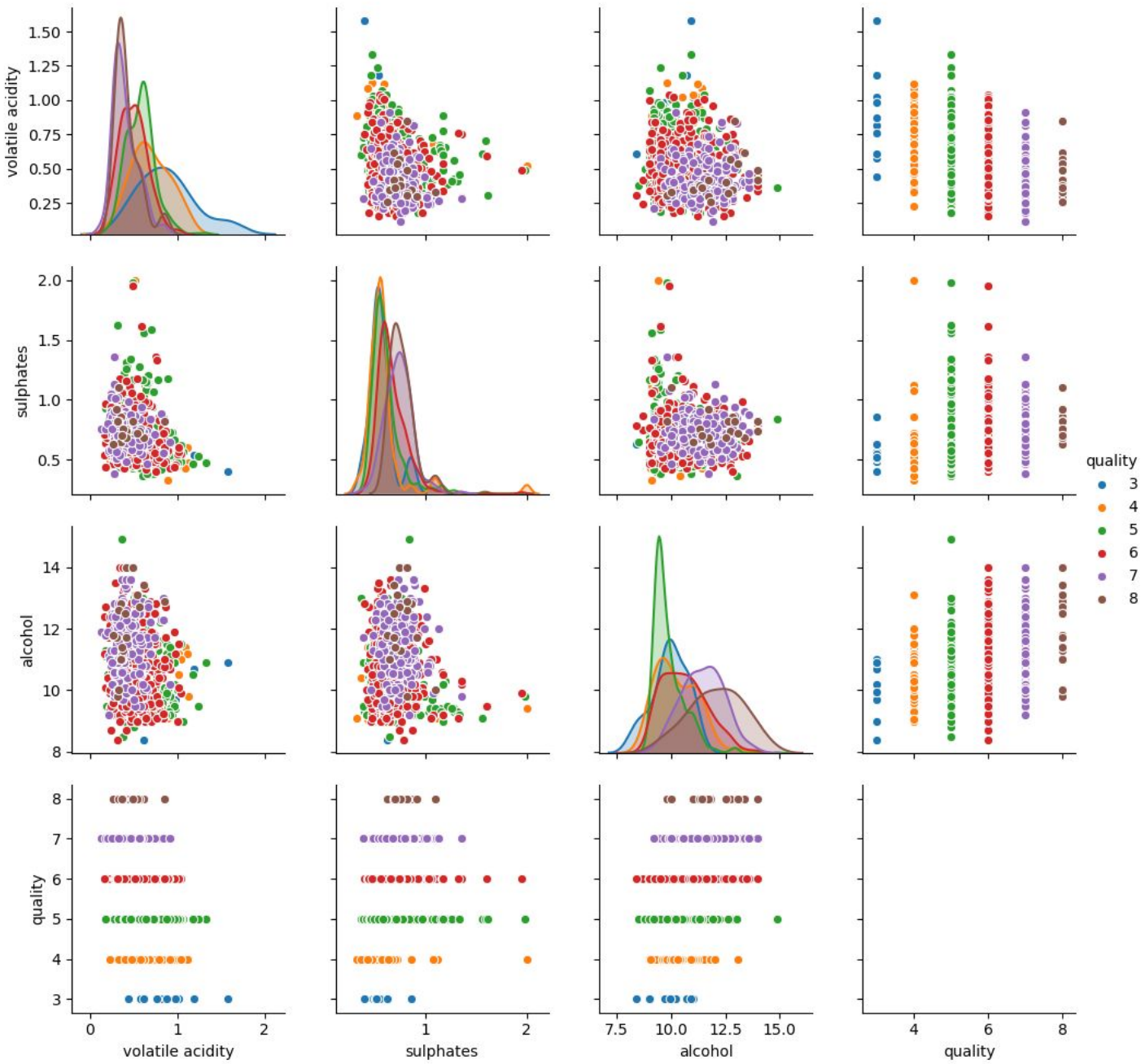

Figure 1: red wine Pair plot

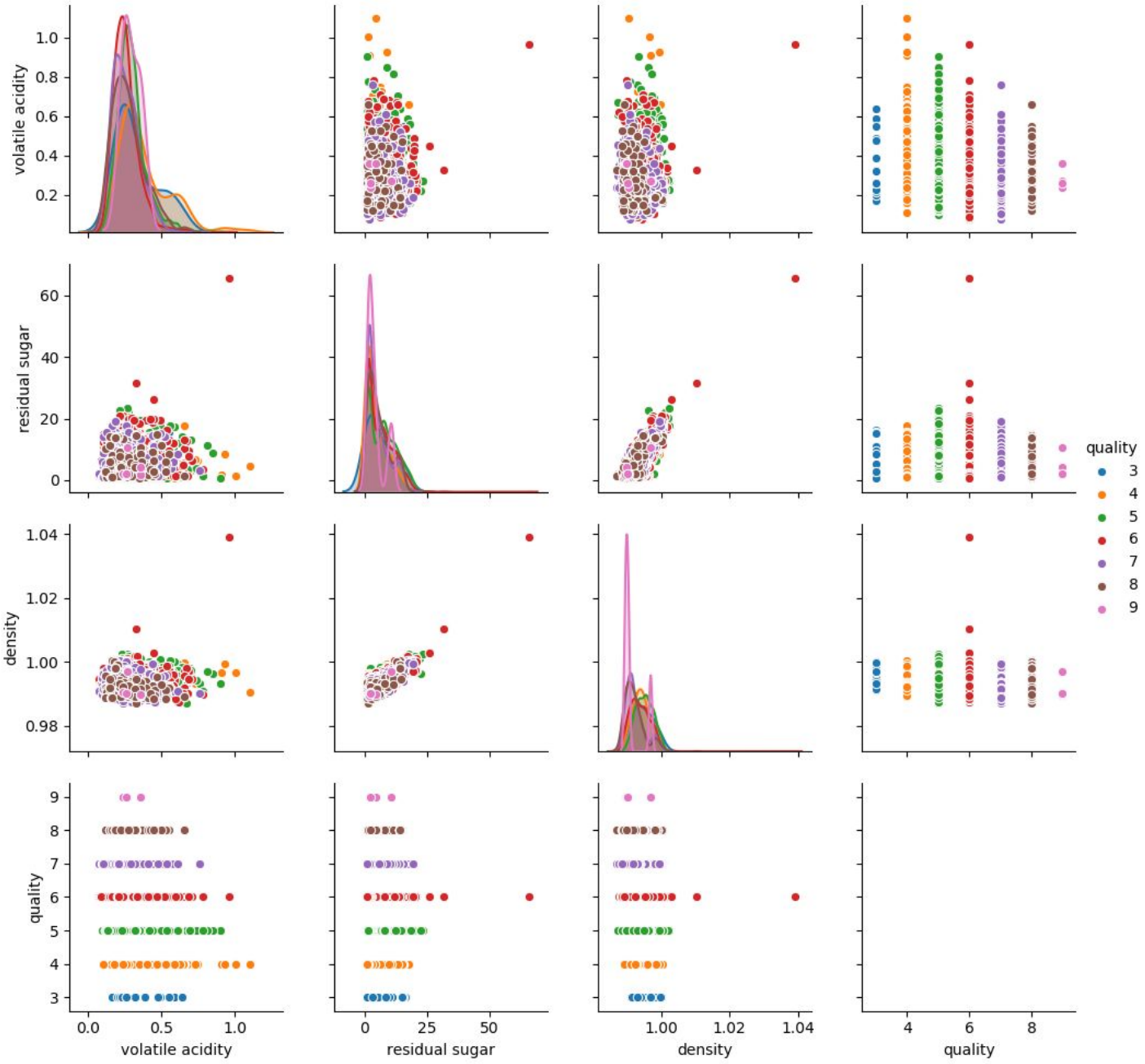

Figure 2: White wine Pair Plot

Regression Tree Model Implementation

A decision tree (DT) can be interpreted as a decision-making tool that uses a tree structure to visualize conditional statement algorithms. Tree models can make very stable predictions and are particularly effective when mapping non-linear relationships [8]. However, a standard DT usually targets on a binary value. In order to apply the model to our dataset, a variation of DT was used: the continuous variable decision tree.

As we have learnt in this course so far, the strategy in tree-based models is always to choose the attribute that produces the purest nodes. One of the impurity criteria we have used in categorical DTs is entropy value. In this regression tree model we used a similar metric for splitting nodes – GINI:

$$GINI(P_1, ..., P_n) = 1 - \sum_{i=1}^{n} p_i^{\,2}, \text{ where } p_i \text{ is the portion of class } i \text{ in the node} \quad [7]$$

Both entropy and GINI can be understood in the same manner. The lower the value, the higher purity this node contains (i.e. less classes are distributed within). In this algorithm we recursively split the node in terms of its GINI impurity until there are no two children can be purer than their parent, or when the maximum depth of the tree reached.

At this point, it is clear how crucial finding an adequate threshold is and how much it influences the impurities of nodes. *"Decision tree from scratch" by Joachim Valente [7]* provides an efficient way: sort the feature values first, then iterate them as potential thresholds. Meanwhile keep track of the number of classes on the right and left, add or minus them by 1 after each threshold comparison. After the i-th threshold iteration there should be i values on the left and m-i on the right, m being the size of node, m[k] is the portion of class k within the node [7]:

$$G_i^{left} = 1 - \sum_{k=1}^{n} \left(\frac{m_k^{left}}{i}\right)^2 \qquad G_i^{right} = 1 - \sum_{k=1}^{n} \left(\frac{m_k^{right}}{m-i}\right)^2$$

Then we take the weighted average as the resulted GINI value.

```
gini_left = 1.0 - sum((num_left[x]/i)**2 for x in range(self.n_classes_))
gini_right = 1.0 - sum((num_right[x]/(m-i))**2 for x in range(self.n_classes_))
gini = (i*gini_left + (m-i)*gini_right)/m
```

The rest is to recursively split the nodes in this manner until maximum depth is reached.

After making several predictions with the completed model, we observed that there is a relationship between accuracy and maximum depth. We further explored this observation in the following method: Since we have randomly split the dataset 80/20 for training and testing purposes, we can generate two pool of predictions from training and testing features, respectively. A prediction in each pool is corresponding to a certain max_depth value we pass into the model, in this case it is a size-15 natural number set from range 1 to 30 in incremental manner. Then we plot each prediction pool in terms of depths for both datasets. The model also outputs the the highest accuracy after all 15 depth inputs, along with the mean absolute error for testing data predictions. All results are listed and discussed in the evaluation section.

Linear Regression Model Implementation

The Linear Regression Model (LRM) was implemented using the gradient descent closed canonical form (GDCF) to find the weights used to predict.

$(X' * X)^{-1} * X' * y = w$ (1) [5]

Where y represents the predetermined wine quality values from the data set.

The main portion of the algorithm is solving for the weights (w). The weights are then used to predict the quality values by using the formula

$y = wx$ (2)

Where x represents the reduced wine attributes.

The algorithm is very simple to implement. First we isolate the quality scores from the wine data. Then we let the scores be y and the wine data be x and pass them into equation (1) in order to solve for the weights. The weights are then used along with the wine data using equation (2) to create predictions. Because of the closed form solution there is no need to split data for training we can just uses the entire data set to compute the weights.

When implementing LRM there were two choices when it came to the type of gradient descent to use. Either the iterative gradient descent algorithm which computes the weights over a series of iterations:

$$w = w + k(1/n)Sum(y - wx)x \quad (3)$$

Or, using closed from conacial equation (1). Although both equations (1) and (2) help provide estimated weight values, the closed form equation was chosen because of the size of the data set. Calculating the inverse of a matrix is very costly on large data sets[6]. However, the wine data sets are very small with the largest amount of data worked with is only 4898 rows of data making the equation (1) a viable solution to estimating the weights.

Evaluation

Linear Regression

Overall the LRM performed well. We tested two metrics, accuracy and mean absolute error (MAE) when evaluating the success of the LRM algorithm. The prediction accuracy was found to be 38% and 42% for whites and reds respectively. This is considered to be relatively high accuracy since just guessing on the correct quality value with no machine learning would be 10% (range of quality metric goes from 1-10).

The MAE is 0.82 and 0.66 for whites and reds respectively. Therefore, on average the quality prediction for wines is off by less than one which is considered to be a successful prediction of qualities because the algorithm is not miss predicting by a whole point.

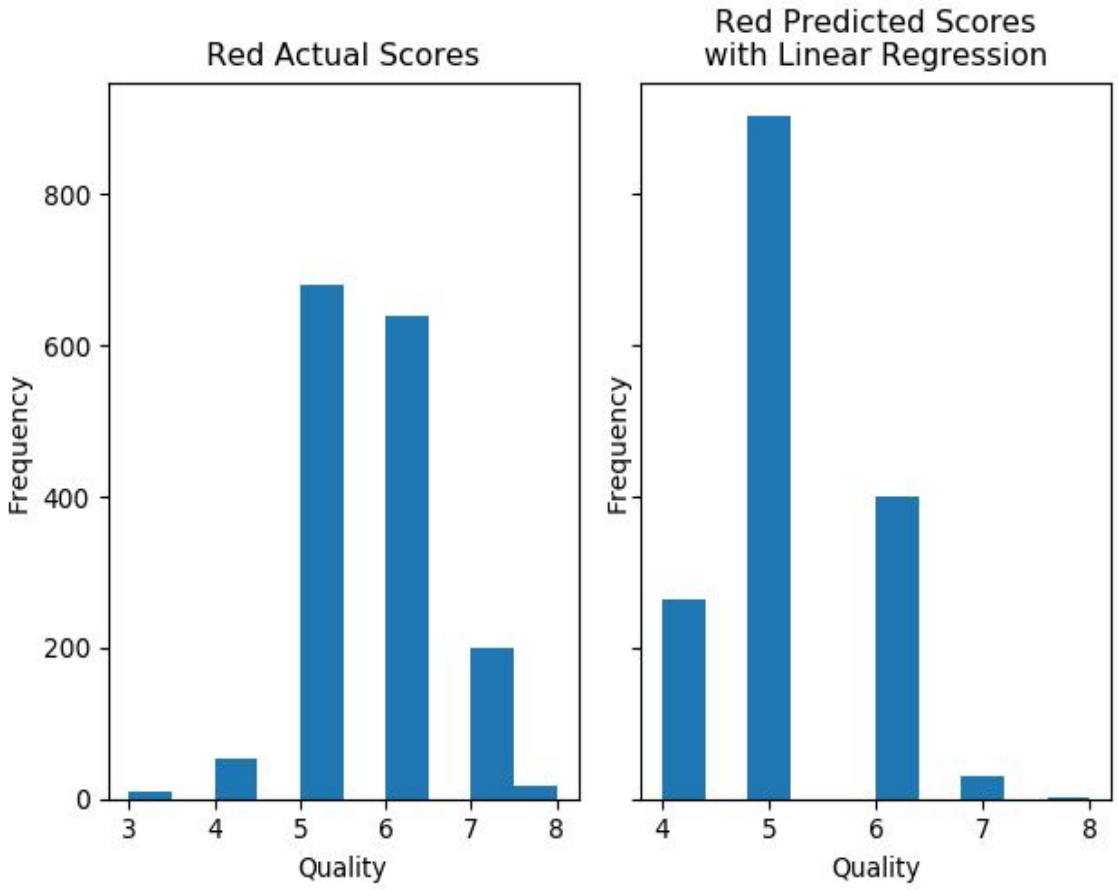

Figure 3: Frequency of red wine quality for actual and predicted scores

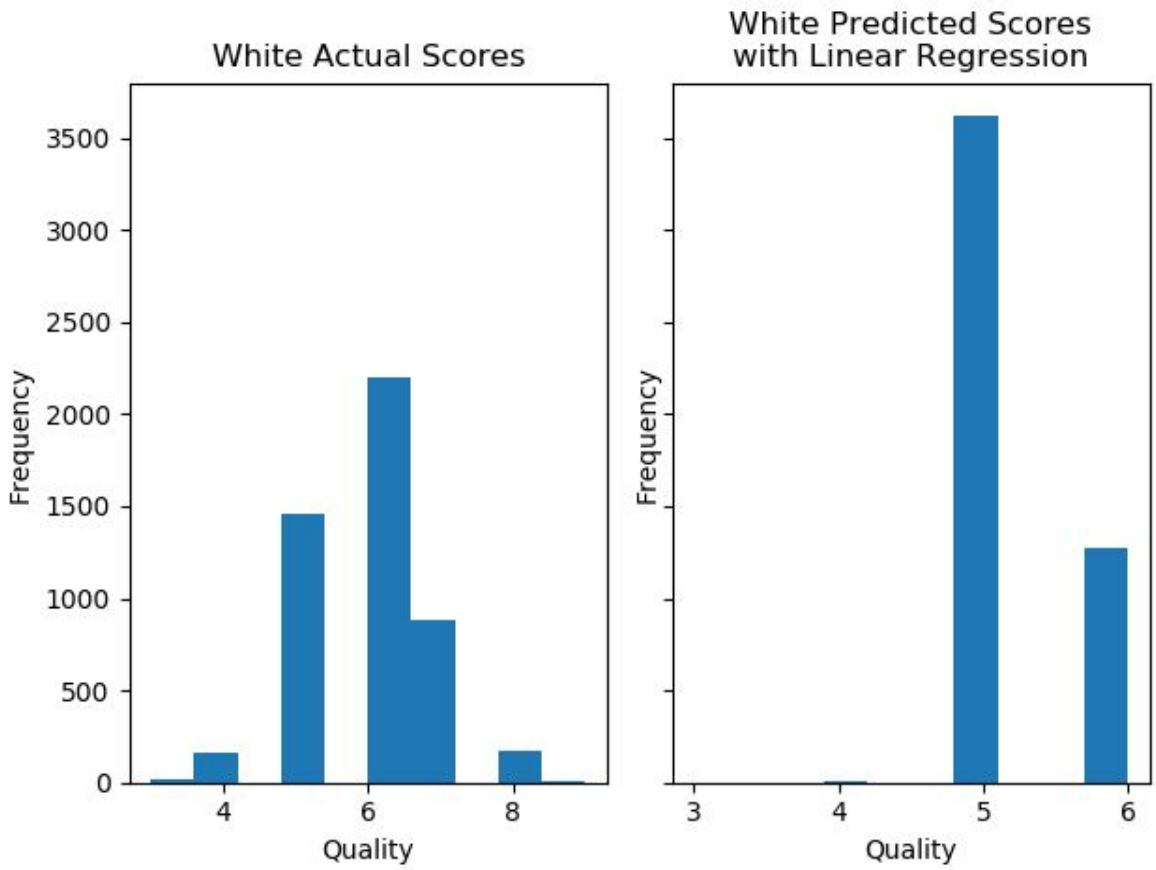

Figure 3: Frequency of red wine quality for actual and predicted scores

As shown in both figures 3 and 4 the highest frequency of predictive scores is 5.0 which happens to be the mean score for the actual quality scores. Thus is should be noted that one drawback of the LRM is that it seems to have a bias to predict the average score instead of predicted the correct score. This is one of the reasons for the accuracy not being higher than it is.

Regression Tree

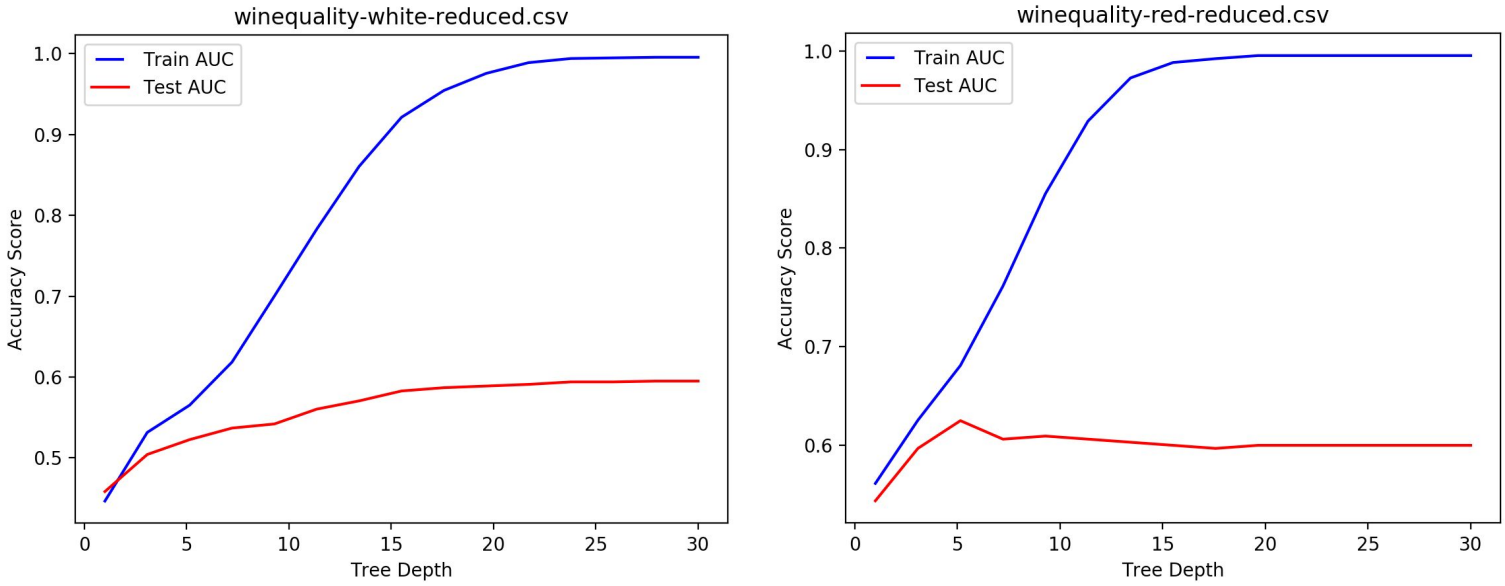

Figure 5: Accuracy vs. tree depth plots for both datasets

As we can notice from figure 5, training data accuracy seems to grow up nearly 100% accuracy as depth increases. Which is justified because the model itself is trained by the same set of data. However we do not see the same behavior for the testing accuracy curve: A ceiling for accuracy seems to exist. For the white wine dataset the accuracy ceases to grow once the depth reaches 15, same for red wine dataset around 6. The plotting have been conducted several rounds and although some changes do occur, the general behaviour follows. We can see that for the red dataset, increasing the depth past 5, caused overfitting.

```
markyBook:data-mining-project markxu$ python3 regressiontree.py winequality-red-reduced.csv
Higest accuracy score in 15-step increasing depths is 0.640625
Mean Absolute Error is: 0.44
markyBook:data-mining-project markxu$ python3 regressiontree.py winequality-white-reduced.csv
Higest accuracy score in 15-step increasing depths is 0.571429
Mean Absolute Error is: 0.56
```

Figure 6: regressiontree.py command line outputs

The highest prediction accuracy for the 15 depths and the mean absolute error for both datasets are shown above. Similar to the graphs, these outputs fluctuate as well. However with rounds of testing the average accuracies are 0.56 and 0.48 for red and white wine dataset, respectively. Mean absolute errors are also computed in the same manner, but had smaller variation.

Comparisons

- The Regression Tree was found to make more accurate predictions for the wine dataset than Linear Regression Model
- The Mean Absolute Error was also lower in the Regression Tree Model than the Linear Regression Model for the given dataset
- Our project has confirmed continuous variable decision tree models, are advantages over linear-based models when predicting continuous real-number values. [8]

Suggestions for Future Work

The LMR algorithm at this at the moment is limited by the lack of scalability due to using the slow closed from gradient descent calculation and no ability to tune the prediction accuracy. For future work changing the gradient descent calculation to a stochastic gradient descent calculation would help algorithm work with large data sets. This is because instead of iteratively updating with weights with a normal gradient descent calculation the stochastic gradient descent algorithm updates the weights after each training tuple [6]. Stochastic gradient descent also allows the user to tune the number of iterations that the algorithm runs which helps tune the accuracy because more iterations the higher accuracy will be produced [6]. As well, in order to improve accuracy the stochastic gradient descent algorithm could be implemented using an adaptive learning rate using the formula:

$$\eta(t+1) := \eta(t)/(1 + t \times d)$$

Where d represents a decrease constant that shrinks the learning rate.

The adaptive learning rate helps control the models ability to over-fit or under-fit and the human error that can be involved with picking a suitable learning rate.

For the regression tree model to further improve, we can suggest the concept of random forests. We would also recommend to increase the size of the max depth number input and decrease the increment gap between numbers. This would allow the model to capture the ceiling of accuracy more precisely. Then instead of only making one round of predictions, one could extract the max_depth that produces the highest accuracy, then use this parameter to re-train our model. This refinement process is to be done in repetitive manner until a stable, consistent increase is observed in accuracy each time.

References

1. UCI Machine Learning Repository: Wine Quality Data Set. [Online]. Available: http://archive.ics.uci.edu/ml/datasets/Wine Quality. [Accessed: 01-Dec-2019].
2. rstudio: “Machine Learning with the UCI Wine Quality Dataset,” 25-Apr-2016. [Online]. Available: https://rstudio-pubs-static.s3.amazonaws.com/175762_83cf2d7b322c4c63bf9ba2487b79e77e.html. [Accessed: 01-Dec-2019].
3. A. Dave, “Regression from scratch - Wine quality prediction,” Medium, 23-Feb-2019. [Online]. Available: https://medium.com/datadriveninvestor/regression-from-scratch-wine-quality-prediction-d61195cb91c8. [Accessed: 01-Dec-2019].
4. “sklearn.feature_selection.RFE,” *scikit*. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFE.html. [Accessed: 01-Dec-2019].
5. “Linear Regression,” SEng 474, https://connex.csc.uvic.ca/access/content/attachment/2d9da65d-1810-4838-ab68-de89fa9a07ab/Syllabus/012d4ba2-7cbf-4486-83a2-3e2b98ead747/regression.pdf[Accessed: 01-Dec-2019]
6. “Machine Learning FAQ Fitting a model via closed-form equations vs. Gradient Descent vs Stochastic Gradient Descent vs Mini-Batch Learning. What is the difference?”, *sebastianraschka*. [Online]. Available: https://sebastianraschka.com/faq/docs/closed-form-vs-gd.html. [Accessed: 01-Dec-2019].
7. ”Decision Tree from Scratch in Python”,[online] Joachim Valente - https://towardsdatascience.com/decision-tree-from-scratch-in-python-46e99dfea775 [Accessed: 27-Nov 2019]
8. R. S. Brid, “Decision Trees - A simple way to visualize a decision,” *Medium*, 26-Oct-2018. [Online]. Available:https://medium.com/greyatom/decision-trees-a-simple-way-to-visualize-a-decision-dc506a403aeb. [Accessed: 15-Nov-2019].