Practical Machine Learning: ML Algorithm Selection

Russell Todd[1]

[1]University of Wyoming

Abstract

1 Introduction

In this exercise, I will be looking at the simplest version of machine learning and choosing the best type of machine learning model for my task. I will familiarize myself with the process and report on how I progressed through the exercise. I will strive to explain well enough that reproducing my work would be doable even if I did not share my exact code. The dataset I will be looking at is the White Wine Quality dataset.

2 Dataset Description

The famous White Wine Quality dataset is comprised of 12 features. The first 11 are measurements of various physical qualities of each wine (fixed acidity, citric acid, residual sugar, pH, alcohol, etc.) and the 12th is the output variable which is a score (0-10) denoting the quality of each wine. There are 4,897 entries (wines) in the dataset and no entries are missing data for any feature. All features are positive numeric values.

After exploring the White Wine Quality dataset, it can be seen that it is quite imbalanced with relatively few entries for low quality and high quality wines which can be seen in figure 1. Unfortunately, in my process I have not included an oversampling or undersampling step to address this.

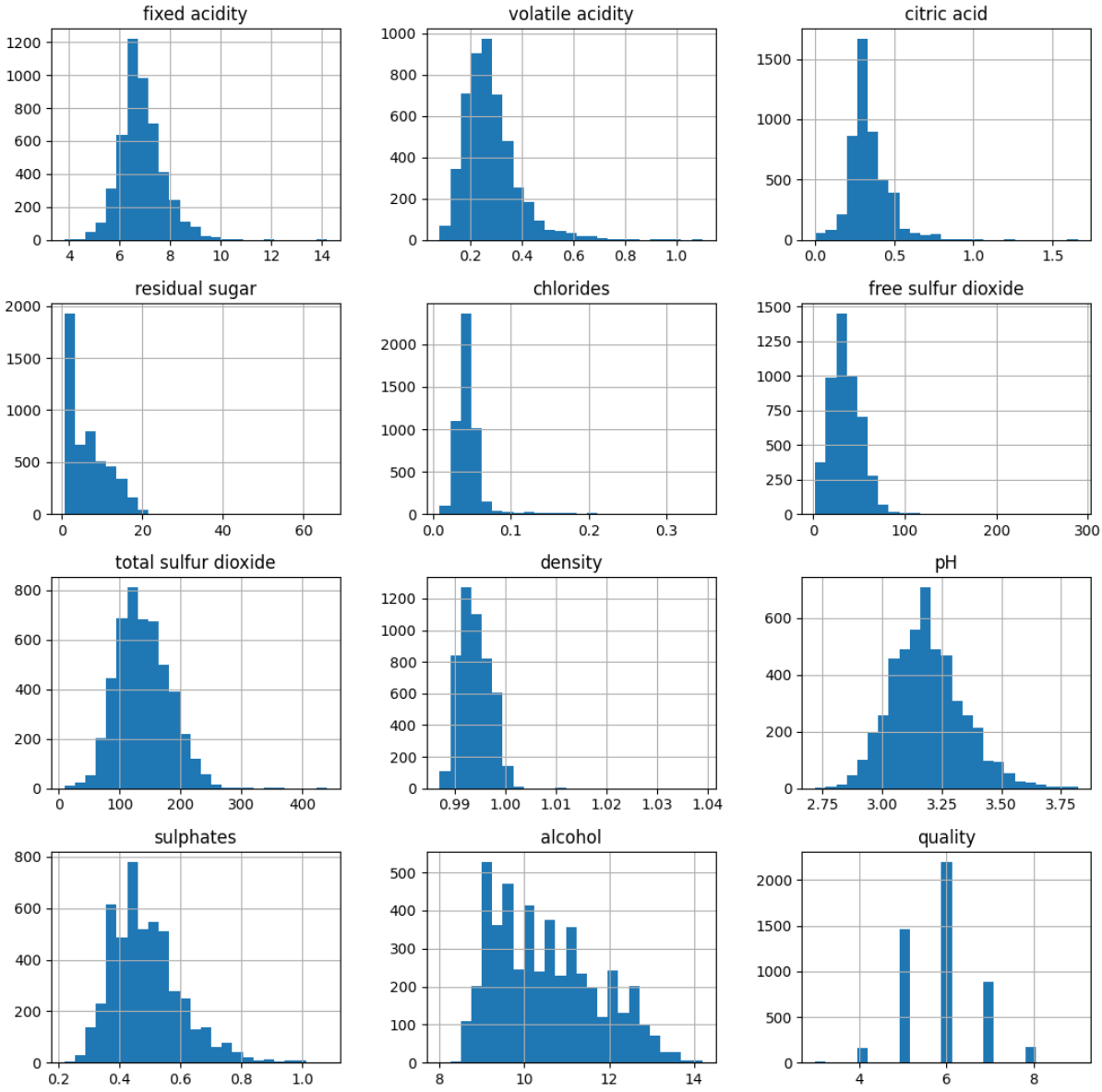

Figure 1: Histogram of Wine dataset features

It also has two variables, density and residual sugar, that are highly correlated as can be seen in figure 2. I decided it might be worth dropping density before training models as it is less correlated to the score than residual sugar.

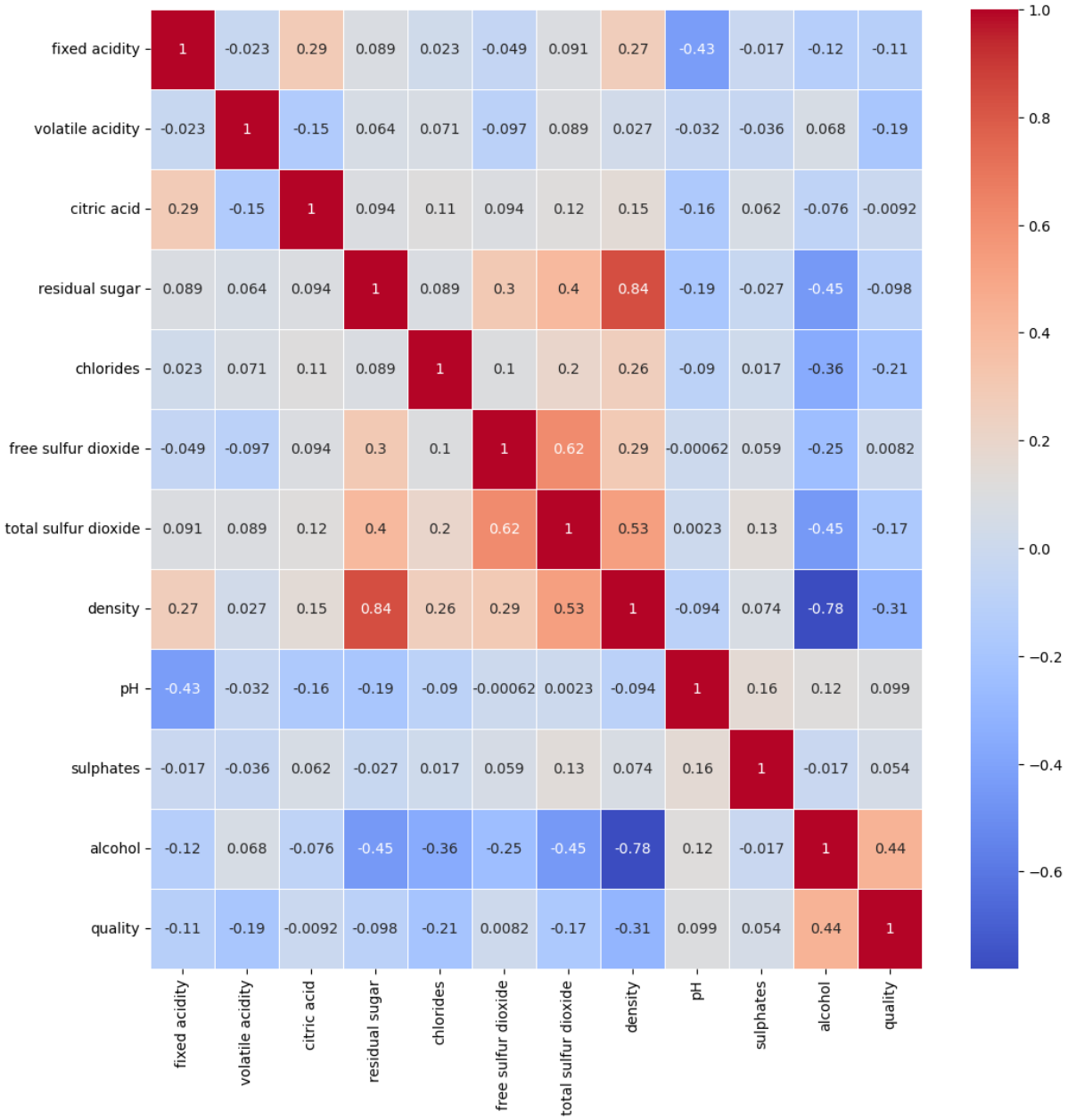

Figure 2: Correlation Heatmap of Wine dataset features

Additionally, most of the features have significant outliers present in their data as can be seen from a selection of plots in figure 3.

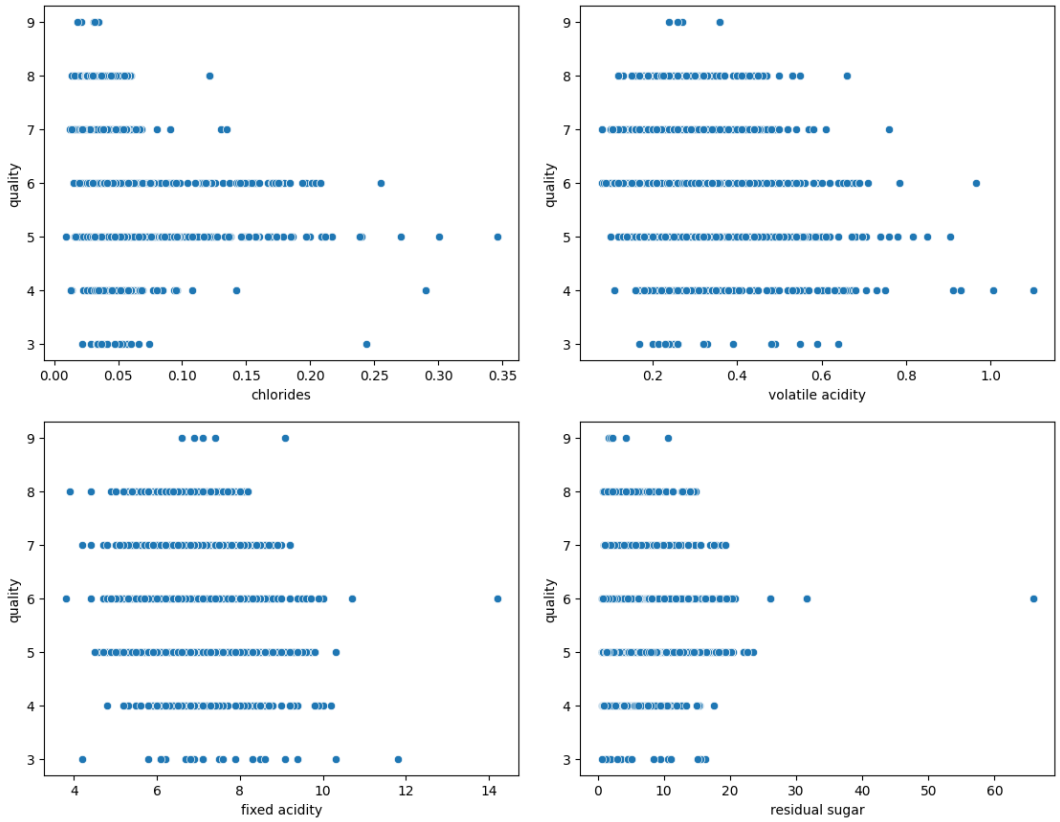

Figure 3: Chlorides, Volatile Acidity, Fixed Acidity, and Residual Sugar Plotted against Quality

2.1 Preprocessing

Based on my findings while exploring the data, I believed that I should apply a feature transformation preprocess like scaling. I then split the data into a an 80:20 training test split before applying the scalers so that the test data would not be included when fitting the scalers. I used the Standard Scaler and the Quantile Transformer from the sklearn preprocessing package. I fit them to the training data and then applied that fit to both the training data and the testing data.

To see the impact the scaling had on the data, I produced a few graphs of the data after both scalers (histogram, correlation heatmap, and pairplot). The most impacted graph was the quantile transformer histogram which is shown below. All of the other scaler plots contained changes that were more subtle or at least indistinguishable in these plot formats.

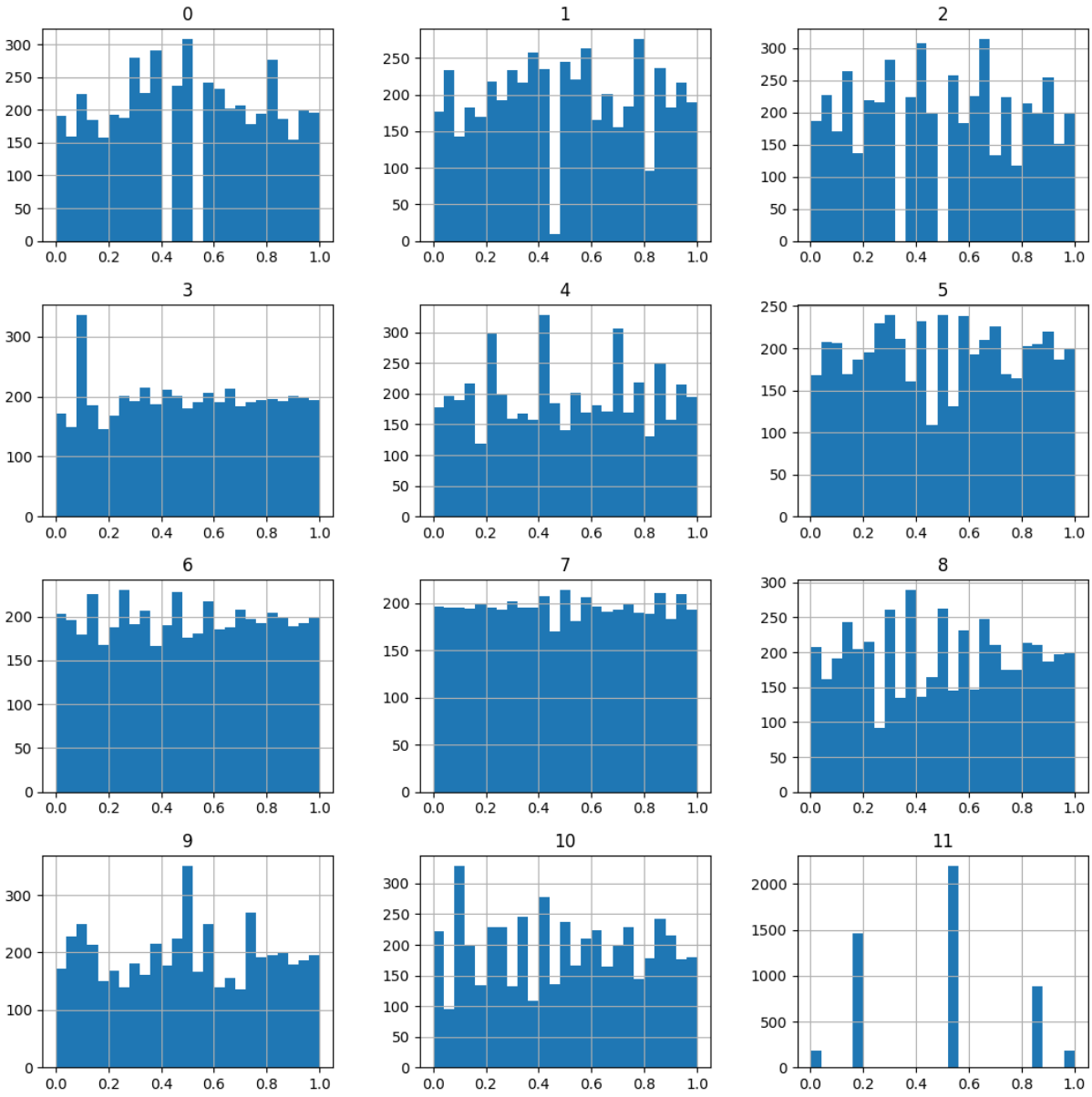

Figure 4: Wine Histogram after Quantile Transformer

3 Experimental Setup

The first step was to get the data properly loaded into a pandas package dataframe and ensure that the datatypes of each feature are appropriate. Afterwards, several summary statistics panes are produced using built-in functions of the pandas dataframe class like info(), dtypes, and describe(). I then use these to double check that the data loading process was done correctly. If not, I then corrected them manually.

I then applied the data preprocessing steps as outlined in the data description section above.

Because I had data from two different scalers, I decided to train and test the machine learning algorithms on both of the datasets for a selection of both Regression and Classification algorithms.

The selected regression algorithms were LinearRegression, DecisionTreeRegressor, and RandomForestRegressor. The selected classification algorithms were SupportVectorMachine, LogisticRegression, DecisionTreeClassifier, KNearestNeighborsClassifier, and RandomForestClassifier.

All algorithms were run with parameters as close to default as possible, with the exception of LogisticRegression which required me to raise the max iterations parameter to run it properly.

To assess the models performance I used the accuracy score function built into each model as well as the mean squared error function from the sklearn metrics package. For the Regression models, I additionally calculated the absolute difference between the predictions and correct answers. I placed them into a dataframe and used the describe function to see some statistics about them. This gave me a better idea of how far off the predictions were from the correct answer and improved my perception of the performance of the regression models. In my code I also included an option to run the classifiers on the dataset as a binary classification problem, sorting the wines into good or bad wines based on whether their quality score was greater than or less than 5.

4 Results

From the graphs below it can be seen that the RandomForest Algorithm noticeably outperformed the other competing algorithms. Additionally we can see that the quantile transformer was beneficial for the LinearRegression, DecisionTreeRegressor, LogisticRegressionClassifier, and DecisionTreeClassifier but not for the others.

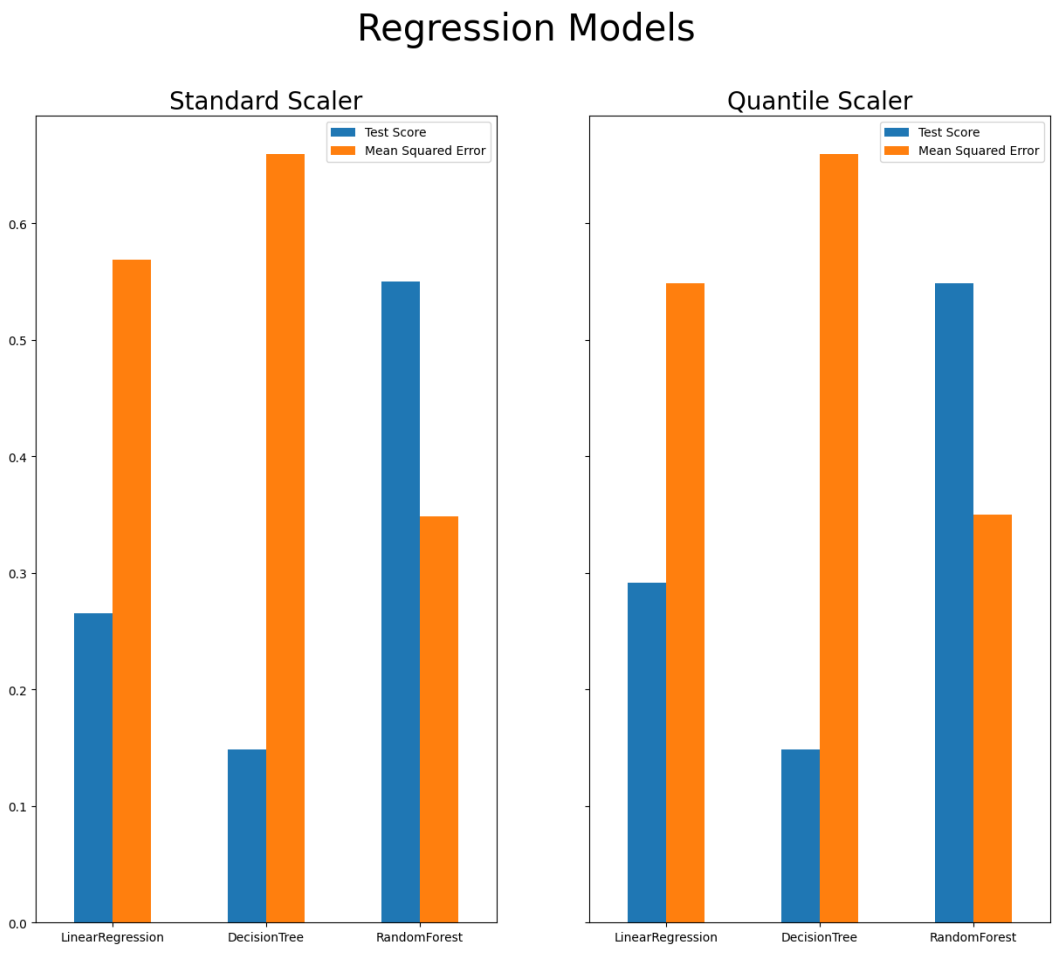

Figure 5: Regression and Classification Model Scores

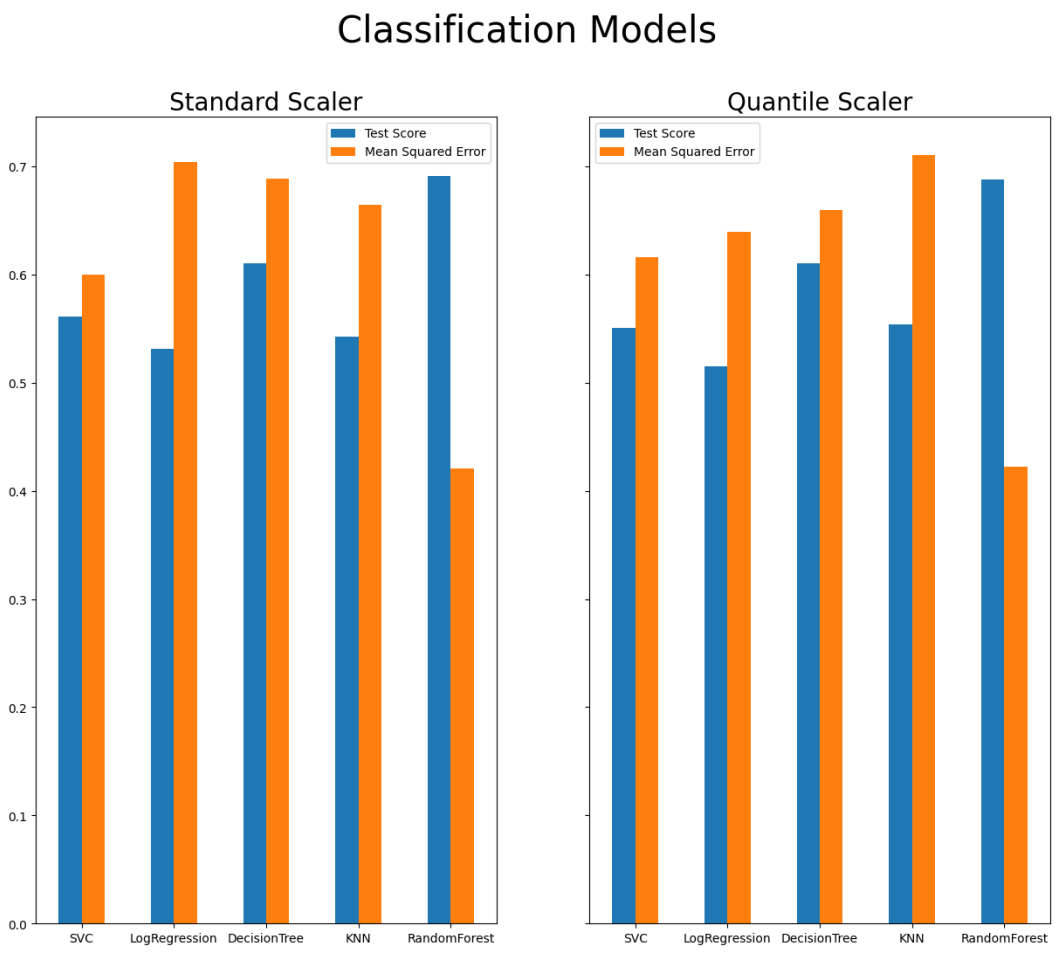

Figure 6: Classification Model Scores

This satisfies my curiosity about whether the fancier scalers should be chosen over the simpler ones like the standard scaler. Now I can see for myself that selecting the correct scaler for the data is not as simple as picking the most complicated one I can find.

If I were to repeat this exercise (I really hope not), I would add an undersampling/oversampling step to address the dataset imbalance and I would add some cross fold validation while training the models to improve their performance.