STAD68 Project Proposal: Comparison of Random Forests and Support Vector Machines (SVM)

Guan Yu Chen
1001591967

Due: November 2, 2018

The Dataset

For this project, I went on Kaggle and looked at datasets that will be suitable for doing supervised machine learning. The dataset I chose is a red wine quality dataset. This dataset contains 1599 observations (rows) and 12 columns. 11 of those columns are features that are used to predict the quality of the red wine and 1 of them is the quality of the red wine. The dataset rates different red wines on a scale of 0 to 10 where 0 would be bad red wine and 10 would be the best red wine. These ratings are based on physiochemical tests of red wine that includes:

- Fixed Acidity This is a numeric value that quantifies most acids involved with red wine or fixed or non-volatile (do not readily evaporate).
- Volatile Acidity This is a numeric value that quantifies the amount of acetic acid in red wine. If this value is too high, it can lead to an unpleasant taste (vinegar taste).
- Citric Acid This is a numeric value that quantifies the amount of citric acid in red wine. This is normally found in small quantities and adds freshness and flavour.
- Residual Sugar This is a numeric value that quantifies the amount of sugar remaining after fermentation stops. Note, it is rare to find wine with less than 1 gram/litre and wines with greater than 45 grams/litre.
- Chlorides This is a numeric value that quantifies the amount of salt in red wine.
- Free sulfur dioxide This is a numeric value that quantifies the free form of SO_2 exists in equilibrium between molecular SO_2 (dissolved gas) and bisulfite ion. This prevents microbial growth and the oxidation of red wine.
- Total sulfur dioxide (PPM) This is a numeric value that quantifies the amount of free and bound forms of SO_2. Low concentration of SO_2 is mostly undetectable but free SO_2 levels over 50 ppm (parts per million), it becomes evident in the nose and the taste of red wine.
- Density This is a numeric value that quantifies the density of red wine. It is close to the density of water depending on percentage of alcohol and sugar contents.
- pH This is a numeric value that describes how acidic or basic the red wine is on a scale from 0 (very acidic) to 14 (very basic). Note, most wines are between 3-4 on the pH scale.
- Sulphates This is a numeric value that quantifies the amount of sulphate in red wine. This is a wine additive which can contribute to sulfur dioxide gas (SO_2) levels, which acts as an antimicrobial and antioxidant.
- Alcohol (%) This is a numeric value that gives the percent of alcohol content in red wine.

Given the features above, there doesn't seem to be any missing value in the dataset. As well as the quality of the red wine for every observation is given. The dataset can be found at `https://www.kaggle.com/uciml/red-wine-quality-cortez-et-al-2009`.

The Analysis

Introduction

Given the description of the dataset above, it would be more logical to model the features and perform classification. In other words, train a model on the set of 11 features given and the quality of the red wine to predict a new red wine given the set of 11 features.

Preliminary - Simplifying the classification problem

Lets start with simplifying the problem a bit, in the dataset, we have 11 classes (wine quality ratings from 0 to 10). Upon careful inspection of the dataset, most of the wine quality ratings seem to fall between 3 and 8 and within that interval, most of it falls between 5 and 8. With this in mind, let q be the quality rating of the red wine and consider the following 2 classes:

1. If $q \leq 5$, classify as *poor* quality wine
2. If $q \geq 6$, classify as *good* quality wine

To make classification easier and more convenient to compute the loss, there will be 2 columns added to the dataset for Poor and Good respectively. Each column will only take TRUE (1) or FALSE (0). Therefore when classifying, we can use a 0-1 loss function to measure accuracy/performance .

Classifying Using Random Forest

With the configuration of the data above, the machine learning algorithm to try is Random Forest. A Random Forest is a collection of decision trees constructed from a subsample of the training data, S sampled with replacement and a vector of features sampled without replacement. The classification of a certain observation is done by majority vote over the predictions of the individual trees.

After constructing and training the Random Forest on the training set, the testing set is used to measure the performance of the model. Here, we use K-Fold Cross Validation to compute the accuracy of the model. To compute the accuracy of the model on the test set, we use a confusion matrix to show true positives, false negatives, false positives and true negative. These metrics are very useful as they will be used to adjust the hyperparameters and training-testing set split percentages. These metrics will also be used to compare Random Forests and SVM performance on this dataset.

Classifying Using Support Vector Machines (SVM)

Before beginning with classifying with SVMs, the labels for the 2 classes are changed to ± 1 since we are still in binary classification case. A SVM works by plotting all data points in a n-dimensional space (in this case 11-dimensional space), and then it classifies by finding the best hyperplane that differentiate the 2 classes very well. In the process of finding the best hyperplane, it uses the margin (distance between the hyperplane and closest data point(s)) to find adjust the hyperplane such that it finds the best hyperplane to differentiate the 2 classes.

The SVM case is very similar to the Random Forest case since both are only dealing with binary classification. So to measure the performance/accuracy of the SVM, we will use a confusion matrix to show true positives, false negatives, false positives and true negative on the test set. SVMs also have hyperparameters that need to be adjusted in order to for it to learn the best model, therefore these matrics will be useful when adjusting the hyperparameters.

Classifying Multi-Classes

In the above learning, the classification problem was simplified into 2 classes, this is like the minimum requirement for this project. Now that we have Random Forest and SVM working with binary classification, we can expand into multi-class. A starting step for this will be just to add 1 more class. The 3 classes will be the following:

1. If $q \leq 4$, classify as *poor* quality wine
2. If $q = 5$ or $q = 6$, classify as *okay* quality wine
3. If $q \geq 7$, classify as *good* quality wine

This part will be done as an experiment to compare if in the case of this dataset, if increasing the number of classes would in crease or decrease the performance/accuracy of the Random Forest and SVM. So, I will run it with the above 3 classes first and compare the metrics, then I will involve all 11 classes (wine quality rating 0 to 10). In both cases, the metrics will be compared to each other and the binary classification cases.

Conclusion & Next Step

To wrap up the project, a comparison between Random Forest and SVM will be made based on Red Wine Quality. One advantage of this dataset is that it doesn't have any missing data. If all implementation goes well for the task mentioned above, there is an algorithm that uses Random Forests to imputation algorithm on missing data. The way training and testing will work for this is the training will still be done on a training set, but for the testing, some data will be removed at random from the testing set and the imputed value from the algorithm will be compared against the original value. This is somewhat like a regression with Random Forest so the appropriate metric to measure this would be mean squared-error.

In conclusion, this project is to analyze the Red Wine Quality dataset using 2 machine algorithms, namely Random Forest and SVM and compare both performance/accuracy.