DATA MINING HOMEWORK 2

KNN IMPLEMENTATION ON WINE_QUALITY DATA SET

VEDASRI UPPALA

uppala.1@osu.edu

TABLE OF CONTENTS

1. DATA SET DESCRIPTION

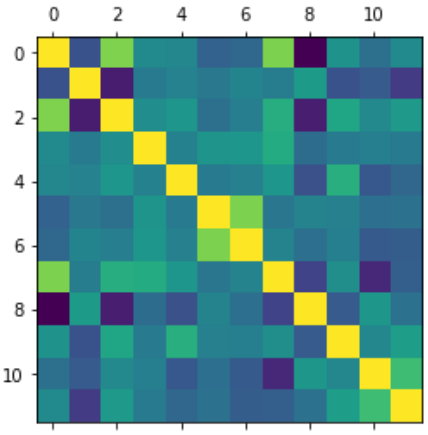
The wine quality-red dataset under consideration gives us information about red variant of Portuguese “Vihno Verde” wine. The data set has 1599 instances and 12 attributes. The ‘quality’ attribute is the output variable and gives information about the quality of the wine. Quality value 0 indicates wine is very bad and 10 indicates that it is excellent.

FEATURES IN THE DATA SET : 'fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar', 'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density', 'pH', 'sulphates', 'alcohol' and ‘quality’.

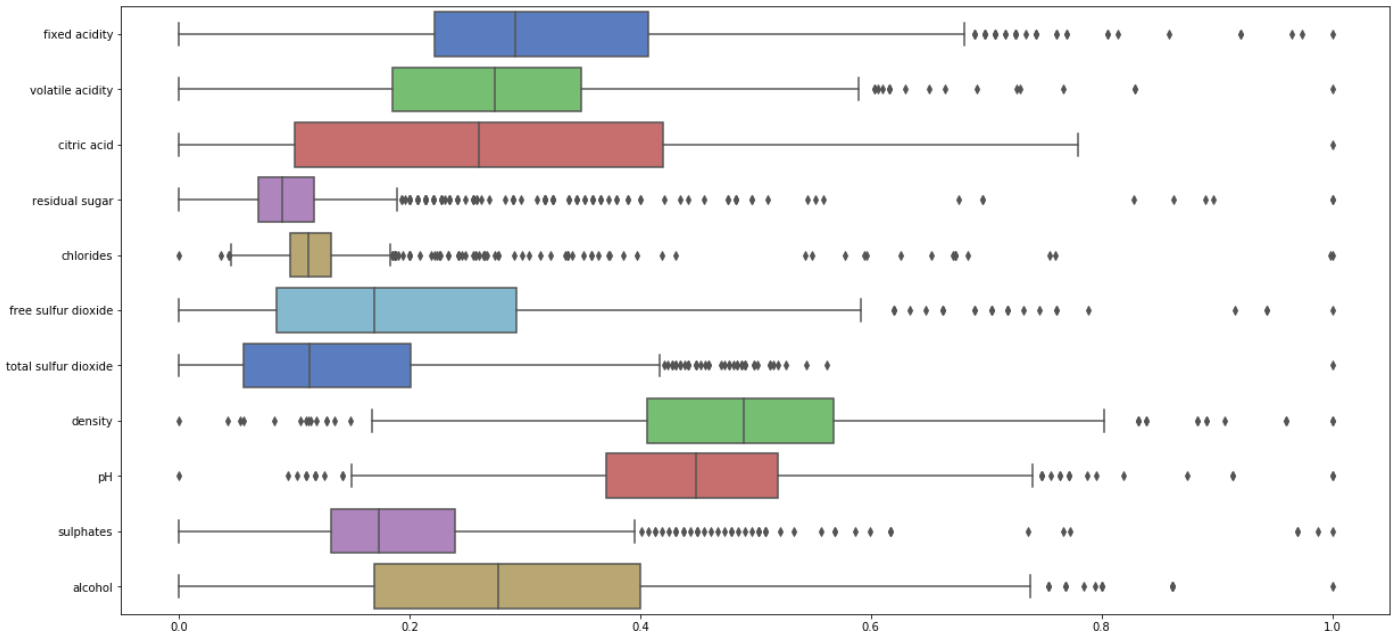
2. DATA ANALYSIS

It has been observed that the given data set has no missing values. To observe correlation between any pair of attributes, the covariance matrix has been calculated with the pearson coefficient.

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
fixed acidity	1.000000	-0.256131	0.671703	0.114777	0.093705	-0.153794	-0.113181	0.668047	-0.682978	0.183006	-0.061668	0.124052
volatile acidity	-0.256131	1.000000	-0.552496	0.001918	0.061298	-0.010504	0.076470	0.022026	0.234937	-0.260987	-0.202288	-0.390558
citric acid	0.671703	-0.552496	1.000000	0.143577	0.203823	-0.060978	0.035533	0.364947	-0.541904	0.312770	0.109903	0.226373
residual sugar	0.114777	0.001918	0.143577	1.000000	0.055610	0.187049	0.203028	0.355283	-0.085652	0.005527	0.042075	0.013732
chlorides	0.093705	0.061298	0.203823	0.055610	1.000000	0.005562	0.047400	0.200632	-0.265026	0.371260	-0.221141	-0.128907
free sulfur dioxide	-0.153794	-0.010504	-0.060978	0.187049	0.005562	1.000000	0.667666	-0.021946	0.070377	0.051658	-0.069408	-0.050656
total sulfur dioxide	-0.113181	0.076470	0.035533	0.203028	0.047400	0.667666	1.000000	0.071269	-0.066495	0.042947	-0.205654	-0.185100
density	0.668047	0.022026	0.364947	0.355283	0.200632	-0.021946	0.071269	1.000000	-0.341699	0.148506	-0.496180	-0.174919
pH	-0.682978	0.234937	-0.541904	-0.085652	-0.265026	0.070377	-0.066495	-0.341699	1.000000	-0.196648	0.205633	-0.057731
sulphates	0.183006	-0.260987	0.312770	0.005527	0.371260	0.051658	0.042947	0.148506	-0.196648	1.000000	0.093595	0.251397
alcohol	-0.061668	-0.202288	0.109903	0.042075	-0.221141	-0.069408	-0.205654	-0.496180	0.205633	0.093595	1.000000	0.476166
quality	0.124052	-0.390558	0.226373	0.013732	-0.128907	-0.050656	-0.185100	-0.174919	-0.057731	0.251397	0.476166	1.000000



In order to observe and visualize the outliers which may be present I have created a box plot for all the attributes. It can be seen from the graph below that the given data set has many outliers. In the next section steps have been taken to handle these outliers.



3. DATA PREPROCESSING

In order to use classification with K Nearest Neighbors Algorithm in the next section, the given data set is first divided into training set and testing set. 75% of the given tuples are randomly sampled as training data and the remaining 25% will be then used for testing.

A new attribute called “class label” has been added based on the quality attribute and the quality attribute has been removed. The value of “class label” attribute will be high if quality > 5 and will be low if quality <= 5. Having a nominal class label will be useful while performing classification

3.1 DATA NORMALIZATION

In order to apply KNN, we need to scale the data. Otherwise, the distance measures will be dominated by attributes which have larger values in magnitude. So I have normalized the data using min-max normalization to fit into 0-1 range. I have created instance of a transformer MinMaxScaler in Python and the instance is first applied on the test data. I have used the same instance of the transformer to normalize the new test data unseen during the fit call. The same scaling and shifting operations will be applied to be consistent with the transformation performed on the train data.

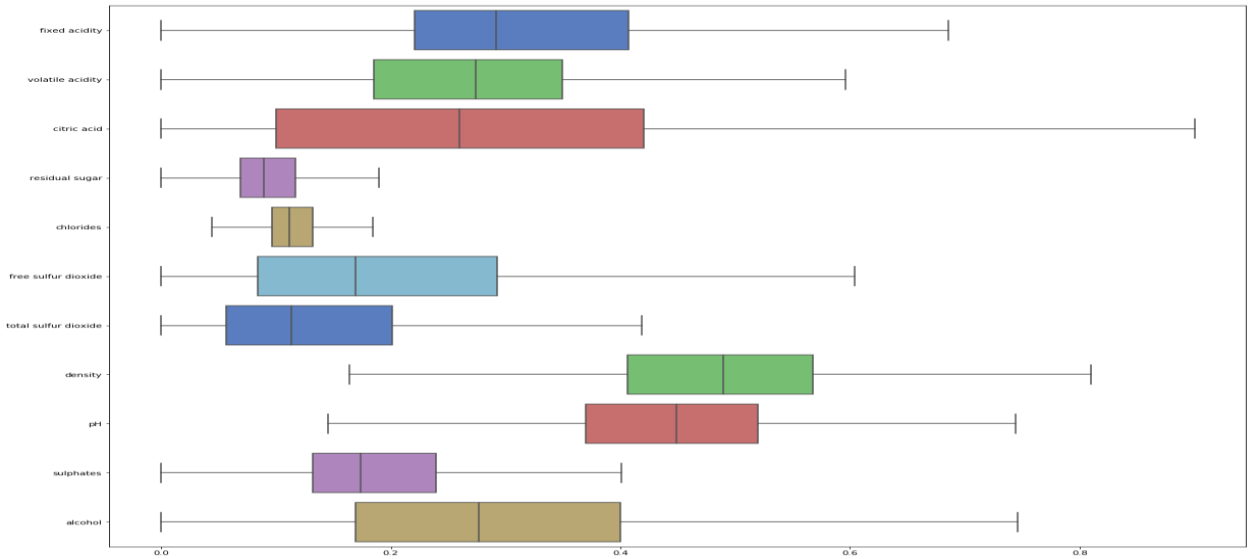
3.2 OUTLIER DETECTION AND HANDLING

As discussed in the previous section, the given data has many outliers. KNN, which we are using as the model for classification later, is susceptible to noise. So I have processed the outliers so that my KNN model gives better results. 1.5IQR rule is used to decide whether a particular point in the training data set is outlier or not. A point will be an outlier if it is falling 1.5 * IQR beyond third quartile or 1.5 * IQR below first quartile.

The data set has only 1599 instances and I felt that it is not a good idea to eliminate tuples in order to deal with the outliers. The following rules have been applied to all the attributes to deal with the outliers.

*IF value > (Q3 + (1.5 * IQR)) then replace the value with (Q3+(1.5 * IQR))*

*IF value < ((Q1 - (1.5 * IQR)) then replace the value with ((Q1-(1.5 * IQR))*



The Box plot of all the attributes after handling the outliers is given and it can be seen clearly that our training data has no more points that are outliers. Since the test data has to be completely unseen I have not done any preprocessing to the test data set apart from normalization.

3.3 FEATURE SUBSET SELECTION

In the previous section, the covariance matrix has been calculated and plotted. It can be observed that no pair of attributes has Pearson Coefficient that is high in magnitude. The maximum magnitude observed is 0.68 and this does not indicate high correlation between those attributes. The feature space is already very small (11 predictors) and reducing it further doesn't seem to be reasonable. Moreover, I have also calculated the variance for all the attributes and observed that none of them have variance which is near to zero. By eliminating features, we might be missing on some important information. For all the above reasons I have not performed feature subset selection to reduce the number of features.

4. MODEL DEVELOPMENT

- Firstly, as mentioned earlier, the given data set is split into training and testing set. Training set has 75% of the given data and the rest is stored as test data.
- For each instance in the testing set, distances have been calculated to all the instances in the training set. I have chosen the value of k.
- Euclidean distance has been chosen as the proximity measure
- The calculated distances have been sorted and the top k tuples having smallest distances to the test tuple under consideration are selected. Sorting has to be done in order to get the most "nearest tuples" to the given tuples.
- The class labels of these k tuples are used to decide the class label of test instance using majority voting. For my model, I have used uniform weights: that is, equal importance is given to all the k most nearest neighbors
- For each test tuple, the posterior probability has been calculated based on the votes obtained for each class label

  Classification Rule : if P(low|x) >= 0.5 then predict "Low" else predict "High"

- All the test tuples have been classified according to the above rule.

	predicted label	actual label	posterior prob
0	Low	'Low'	1.0
1	Low	'Low'	1.0
2	Low	'Low'	0.9090909090909091
3	Low	'Low'	0.9090909090909091
4	Low	'Low'	0.6363636363636364
5	Low	'High'	0.5454545454545454
6	Low	'High'	0.6363636363636364
7	Low	'High'	0.7272727272727273
8	High	'Low'	0.18181818181818177
9	Low	'Low'	0.7272727272727273

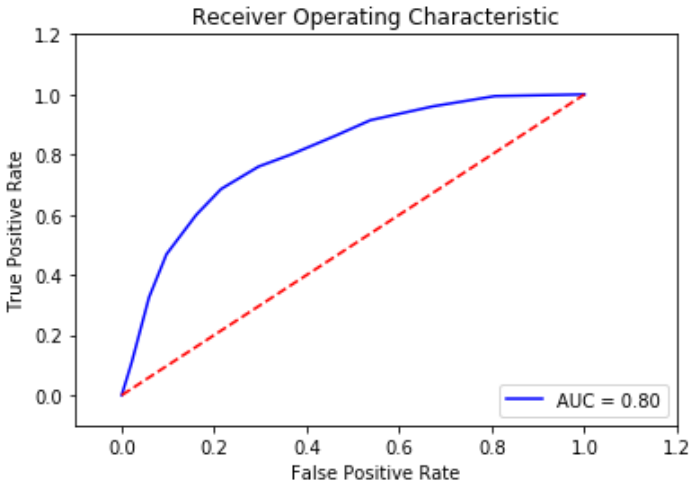
An mx3 table has been created where m is the number of instances in the test data. Some rows in that table are shown on the left side. In the table "posterior prob" is the value of probability that the class label is "Low". So "predicted label" will be "Low" if this value is >= 0.5 and "High" otherwise. Here the value of k is chosen as 11.

5. MODEL EVALUATION

Since the class labels of the test data are known, I have tested the accuracy of my model. A python function called "getAccuracy" has been implemented to do this. This function will iterate through all the samples in the test data and will compare the label predicted by the KNN model with the true label of the test tuple. Accordingly, accuracy of the model is calculated. I have obtained maximum accuracy when the k value is taken as 11. Precision, Recall, FPR, FNR, TNR also have been calculated. The ROC curve has been plotted using the posterior probabilities P(low|X).

The model has been tested with different values of k and sample results for 3 values of k are shown below.

1. K=11




```
Confusion matrix:
Predicted  High  Low  __all__
Actual
High        131   55      186
Low          42  133      175
__all__     173  188      361
```

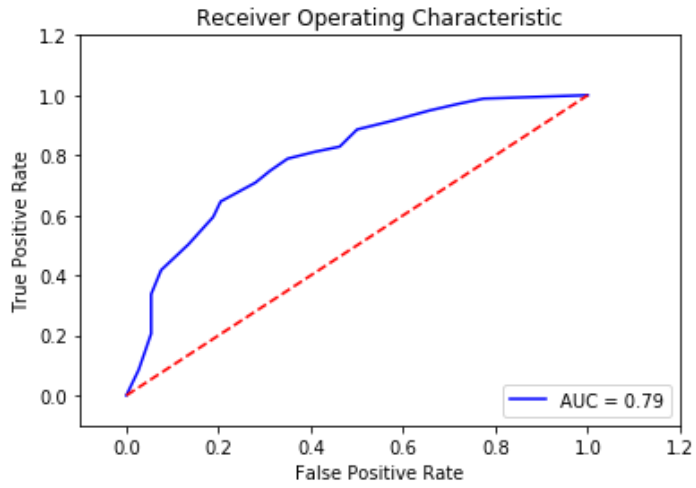
Accuracy = 73.132% *Error rate = 26.868%*

Precision = 70.744% *FNR = 24%*

Recall = 76% *FPR = 29.569%*

Fscore = 73.278% *TNR = 70.43%*

2. K=19




```
Confusion matrix:
Predicted  High  Low  __all__
Actual
High        128   58      186
Low          44  131      175
__all__     172  189      361
```

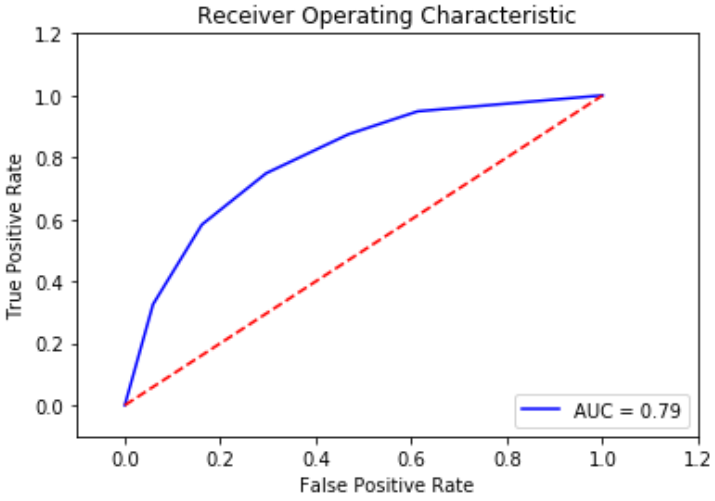
Accuracy = 71.7451% *Error rate = 28.255%*

Precision = 69.31% *FNR = 25.143%*

Recall = 74.8% *FPR = 31.182%*

F score = 71.97% *TNR = 68.81%*

3. K=5




```
Confusion matrix:
Predicted  High  Low  __all__
Actual
High        131   55      186
Low          44  131      175
__all__     175  186      361
```

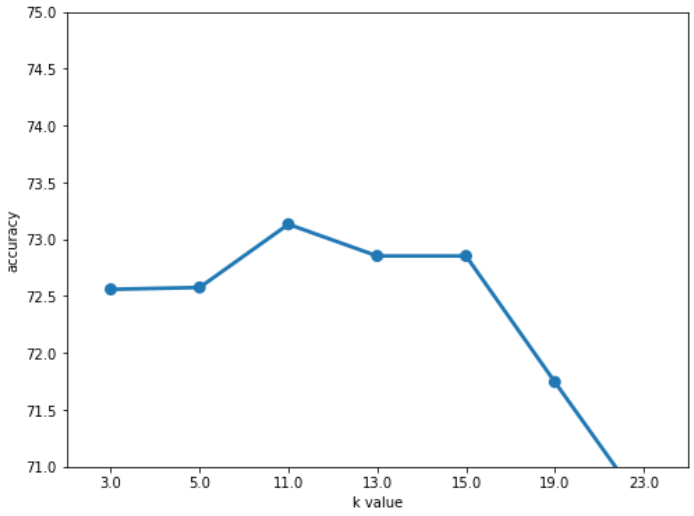
Accuracy = 72.5761% *Error Rate = 27.424%*

Precision = 70.430% *FNR = 25.143%*

Recall = 74.857% *FPR = 29.569%*

Fscore = 72.5761% *TNR = 70.43%*

The accuracy is calculated for a bunch of different values of k and the accuracies are plotted.



	k value	accuracy
0	3.0	72.5600
1	5.0	72.5761
2	11.0	73.1320
3	13.0	72.8531
4	15.0	72.8531
5	23.0	71.4680

It can be seen that, initially as k increased, accuracy also increased. Accuracy reached maximum when k=11. Then accuracy started decreasing. When k = 3, there is under fitting problem and the model did not have enough neighbors to correctly predict the label. When k=23, because of over fitting the accuracy has decreased.

6. COMPARISON

The “scikit” library for Machine Learning in python provides a plugin called “KNeighborsclassifier”. The KNN model implemented is compared with this off-the-shelf implementation. Apart from scaling I have not done any preprocessing to the data fed into the off-the-shelf implementation. The accuracy and F scores have been calculated. The maximum accuracy observed was when k=11 and the accuracy is found to be 74.615%.

K value	Accuracy	Fscore
3	69.74%	70%
5	74.102%	74%
11	74.615%	75%
13	74.312	74%
19	73.333%	73%

The above table indicates the accuracy obtained by off-the-shelf implementation for different values of k. It can be clearly seen that these results conform to the results attained using my implementation. The accuracies and F scores are approximately in agreement with my implementation. The preprocessing I have done to the input of my model might be the reason for getting reasonably good accuracy. In some cases accuracy might not be a good measure of performance. So the F scores have been calculated and it is observed that the F scores of both the implementations differ only by a small value.