Machine Learning (ECE5XX)

Assignment 2

Aniket Pradhan 2017133

Question 1:

Part 1

```
SVM used: svm.SVC
Kernel: linear
Input Parameters: Cs = [0.001, 0.01, 0.1] ; Best Param: C = 0.001
Training accuracy: 0.31802
Testing accuracy: 0.3172
Time took to train the model: 5054.775724887848 seconds
```

Refer to the appendix for more information regarding the gridsearch optimization

To reduce the feature space of the data, I had first normalized the images (divided all pixels by 2^8, i.e. 255) later on I converted all the images to grayscale, which reduced the size of an image by a factor of 3.

Since I used a linear kernel, gamma was not required. Otherwise, the penalty tradeoff term: C, was set to the lowest value of the 3 provided. The C value controls the trade-off between the decision boundary and the data points. Increasing the C value would lead to overfitting, and hence a lower value of C is preferred.

Part 2

```
Training accuracy: 0.3014217815291322
Testing accuracy: 0.3172
The number of Support Vectors: 48812
Time took: 4904.812579154968 seconds
```

Refer to the appendix for more information regarding the training

Part 3

From part 2, we can see that the number of support vectors is: 48812 out of the 50000 training vectors. This means that 97% of members of the training set are support vectors. This means that svm.SVC uses almost every single training vector to train the model, highlighting a lack of regularity in the data. This can be resolved by using a bigger dataset or using some other features apart from the raw pixels.

The training and the testing accuracies are almost similar (testing accuracies being the exact same). This can be explained by the simple fact that the support vectors are mainly responsible for training the model as they define the hyperplane separating the classes.

Question 2

Part 1

Because of the higher number of features, the pair-plot was very big in size, which could not fit on this report. Therefore, I have uploaded an image of the plot here [1] so that you can see the image un-compressed and with more detail.

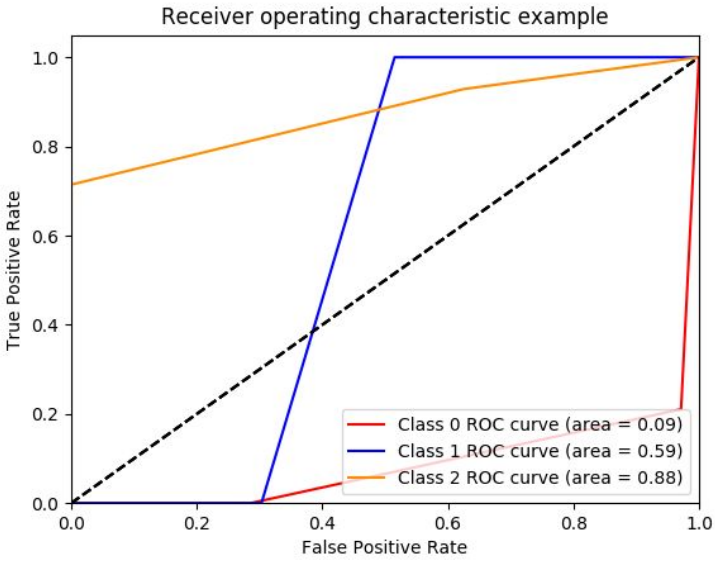
The pair-plot basically shows the data points with respect to pairs of features, say alcohol-malic_acid. Using these plots one can estimate the relationship between two features. The histograms show the distribution of a single feature.

Part 2

OneVSOne

```
Training Accuracy: [0.86666666, 0.85135135, 0.95238095] 0.79032258
# Classwise training accuracy, training set accuracy
Testing Accuracy: 0.8518518518518519
Training F1-Score: 0.7903225806451614
Testing F1-Score: 0.8518518518518519
Time Took: 0.304684400558471 ---
Classifier used: svm.LinearSVC
```

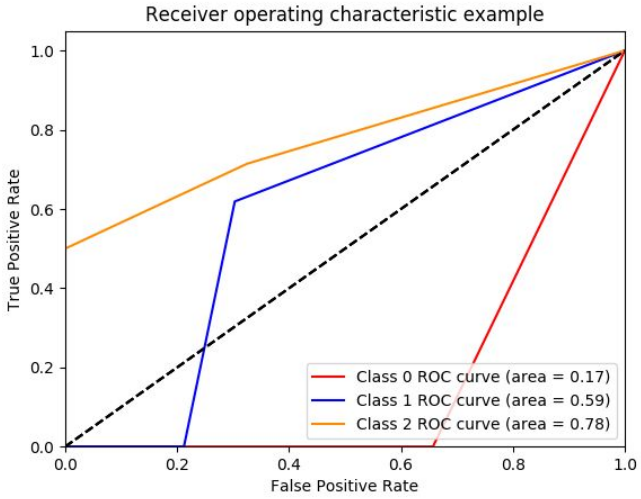
ROC Plot:



OneVSRest

```
Training Accuracy: [0.79838709, 0.77419354, 0.96774193] 0.74193548
# Classwise training accuracy, training set accuracy
Testing Accuracy: 0.7222222222222222
Training F1-Score 0.7419354838709677
Testing F1-Score 0.7222222222222222
Time took: 0.05858802795410156 seconds ---
Classifier used: svm.LinearSVC
```

ROC Plot



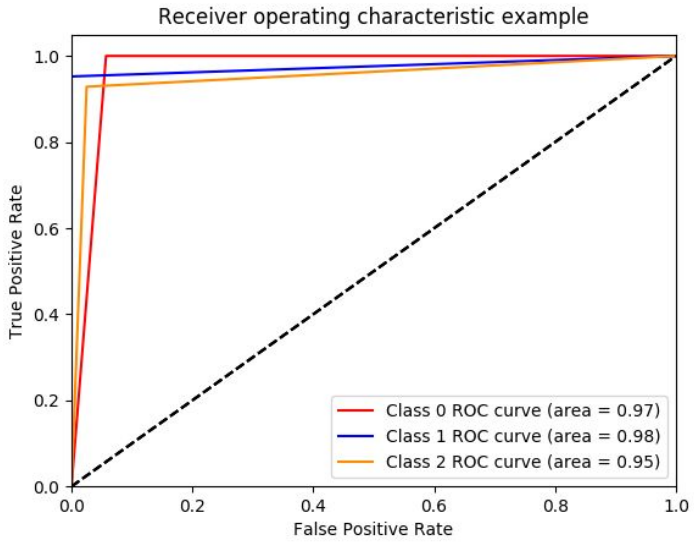
Both OVO and OVR methods report lesser accuracies than Naive Bayes and Decision Trees

(as shown below) because OVO and OVR are beneficial for multinomial classification where the number of classes is relatively higher. The current dataset had only 3 classes (that too with fewer data).

Part 3

```
Training accuracy: 0.9758064516129032
Testing accuracy: 1.0
Training F1-Score 0.9758064516129032
Testing F1-Score 1.0
Time took: 0.010170936584472656 seconds ---
```

ROC Plot

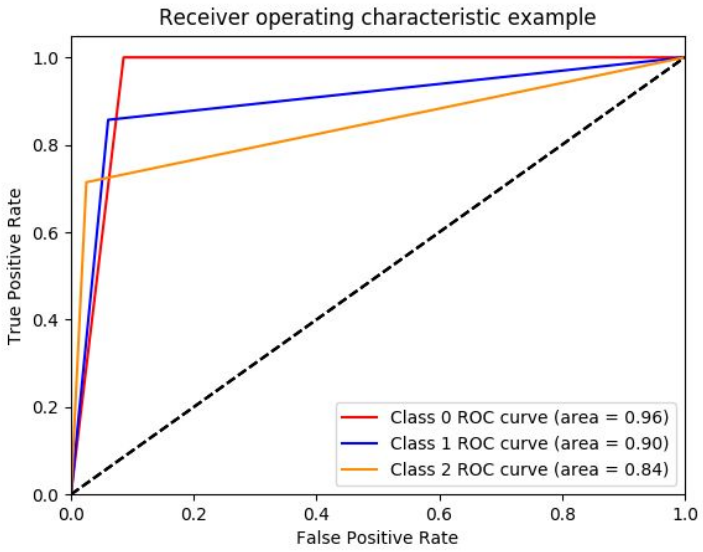


Gaussian Naive Bayes provides a perfect testing accuracy, mainly because of the less testing data. The training accuracy is also high because the data has already been trained once. Given the number of features (13) and the number of classes (3), naive Bayes has performed as per the expectations, because it works on the probabilities of every feature over the selection of the class.

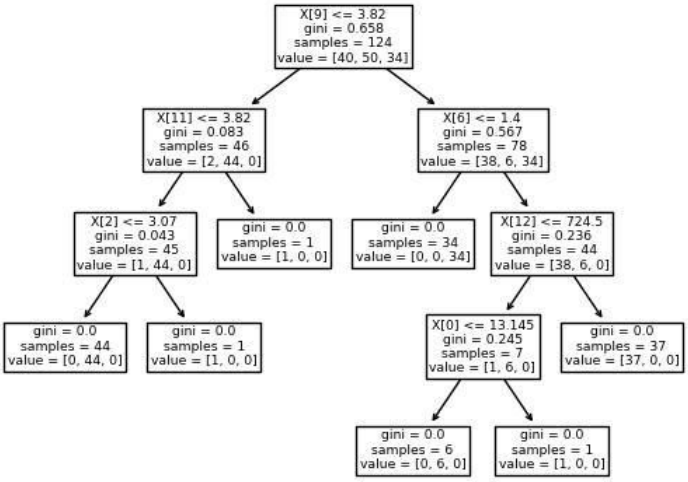
Part 4

```
Training accuracy: 1.0
Testing accuracy: 0.9444444444444444
Training F1-Score 1.0
Testing F1-Score 0.9444444444444444
Time took: 0.0014061927795410156 seconds ---
Params: Default params
```

ROC Curve:



Decision tree:



Decision Trees provide a perfect training accuracy as it is able to create a tree w.r.t. to the given training data. The testing error is a bit low, mainly because of slight overfitting in the decision tree.

Part 5

From the above parts, time taken for different approaches are:

```
OVO Time Took: 0.304684400558471 seconds ---
OVR Time took: 0.05858802795410156 seconds ---
Gaussian NB Time took: 0.010170936584472656 seconds ---
Dec. Tree Time took: 0.0014061927795410156 seconds ---
```

Since the number of classes is low (3), decision trees take the least amount of time to fit the data in the tree. GaussianNB takes a bit longer as it calculates the probabilities of the respective training data to train the model. OVR and OVO methods take the longest time as they train 3 different models simultaneously and have to predict and compare amongst the fitted models to get the final result. The OVR and OVO methods were trained and tested using svm.SVC with a linear kernel and C = 0.001

The testing accuracies and the F1 scores for the OVR and OVO methods are less than the Gaussian NB and decision trees, mainly because OVR and OVO methods are suitable when there are a higher number of classes. Since in our dataset only 3 classes are present these methods do not produce a satisfactory result. GaussianNB produces a good result because of the presence of more number of features and fewer classes. Since the NB method is probabilistic, having more number of features acts as a boost for the method. Dec. trees also give a satisfactory result, yielding a perfect accuracy on the training set and somewhat lesser for the test set. This shows that the model overfits, but not much because the data is not much variable.

In my opinion, GaussianNB is the best model to use for the current dataset. This is because of the higher number of features and fewer classes. The accuracies also show that the data is independent of each other (maybe separable by a linear boundary) which is a perfect fit for the GaussianNB.