Exam 4 Ben Gowaski
EECE5466

Question 1:
File Output Interpretation:

SVM Cross Validation Scores:

```
[0.61878453 0.61878453 0.61878453 0.61666667 0.61666667 0.61666667
 0.61666667 0.62011173 0.62011173 0.62011173]
```

- We can see that as 'cv' approaches 10, we get much better accuracy

Best SVM parameters found for K= 10 :

```
 {'C': 10, 'gamma': 0.1}
```

- These parameters were consistent among the majority of tests

SVM accuracy score: [0.6432160804020101]

- Using the 'svm_score' routine we can get an even more accurate score

MLP:

- For MLP we ran a list of parameters against each other to see which resulted the best:
- # Find the best params out of a set
- parameter_space = {
-     'hidden_layer_sizes': [(50,50,50), (50,100,50), (100,)],
-     'activation': ['tanh', 'relu'],
-     'solver': ['sgd', 'adam'],
-     'alpha': [0.0001, 0.05],
-     'learning_rate': ['constant','adaptive'],
- }

Best MLP parameters found for K= 10 :

```
 {'activation': 'relu', 'alpha': 0.05, 'hidden_layer_sizes': (50, 100, 50), 'learning_rate': 'constant',
'solver': 'adam'}
```

- MLP has a lot more parameters to tune, but by doing this we can consistently get better accuracy than SVM in our tests as seen below

Results on the MLP accuracy test set:

```
MLP mean_test_score:  [0.61277778 0.63111111 0.61611111 0.64444444 0.61722222
0.65277778
 0.61722222 0.65166667 0.60722222 0.61833333 0.60222222 0.61777778
 0.61944444 0.63833333 0.61777778 0.645      0.61833333 0.63777778
 0.61833333 0.64833333 0.6        0.61833333 0.60666667 0.61833333
 0.62277778 0.65944444 0.62       0.66444444 0.62       0.67722222
 0.61611111 0.68277778 0.61888889 0.63388889 0.62222222 0.63444444
 0.61833333 0.66666667 0.61777778 0.66388889 0.61722222 0.68666667
 0.62166667 0.68222222 0.62055556 0.62888889 0.62       0.61888889]
MLP std_test_score:  [0.01448457 0.0766701  0.00575145 0.0845644  0.00411471 0.08540194
 0.00305682 0.10208576 0.01971497 0.00145467 0.0289823  0.00259725
 0.00610823 0.06439224 0.00209176 0.07132157 0.00145467 0.09075559
 0.00145467 0.08427133 0.03535847 0.00145467 0.01869793 0.00145467
 0.01284831 0.10649951 0.00464333 0.11866318 0.00533541 0.12133524
 0.00623958 0.13092984 0.00231908 0.03273838 0.0068179  0.02748162
 0.00145467 0.10020839 0.00259725 0.10997578 0.00411471 0.12669992
 0.00953965 0.12068327 0.00625724 0.02364802 0.01161243 0.03037116]
```

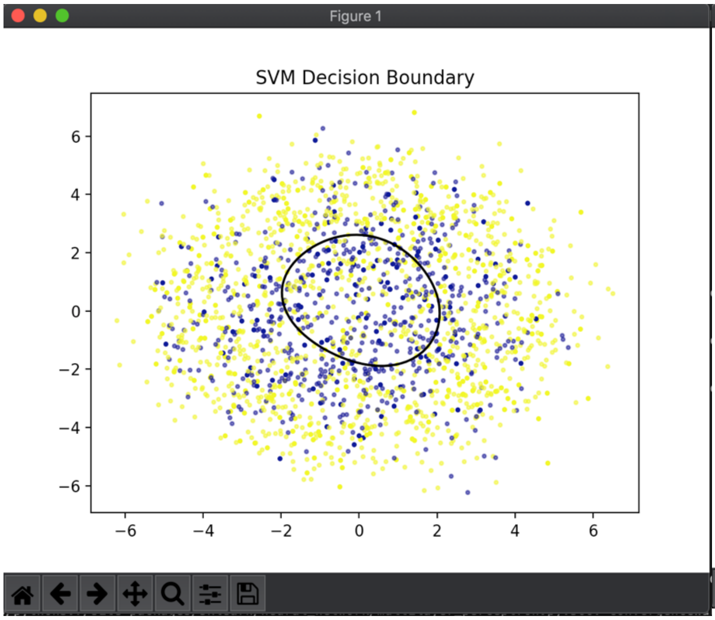
MLP accuracy score: [0.7035175879396985]

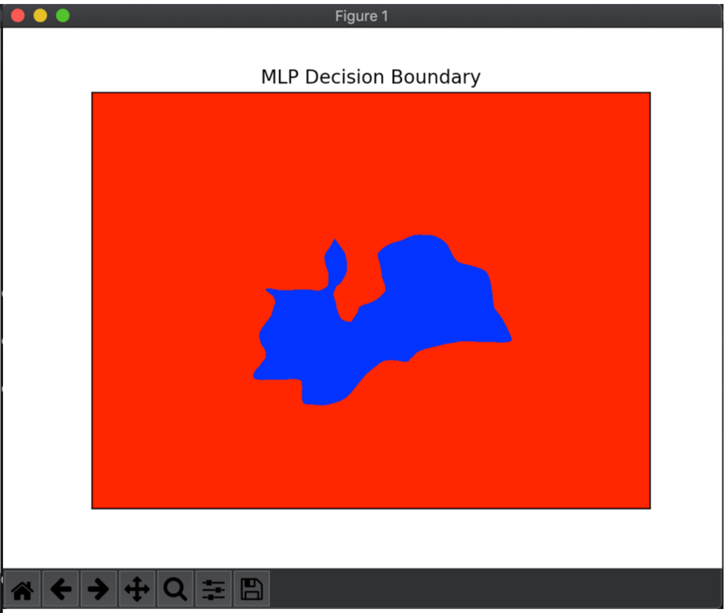
- In addition to the cross validation outputs for means and stds, we can see the "accuracy_score" function return over 70% when used on the predicted output of the MLP classifier. This is consistently higher than SVM

```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         0
           1       1.00      0.65      0.79       199

    accuracy                           0.65       199
   macro avg       0.50      0.33      0.40       199
weighted avg       1.00      0.65      0.79       199
```

"The F-beta score can be interpreted as a weighted harmonic mean of the precision and recall, where an F-beta score reaches its best value at 1 and worst score at 0."

This is a classification_report example from sklearn where we can see that the F1 score for the MLP classifier with varying F1-scores. The higher the score, the less likely the classifier is to not label false positives, as well as correctly label true positives.





- After running the program a few times, we can see consistent decision boundaries on the output plots, similar to the ones above, for both SVM and MLP. The MLP seems to be more non-linear, which can also be a result of its slightly higher accuracy than SVM, however, it is a very small accuracy advantage (<5%).

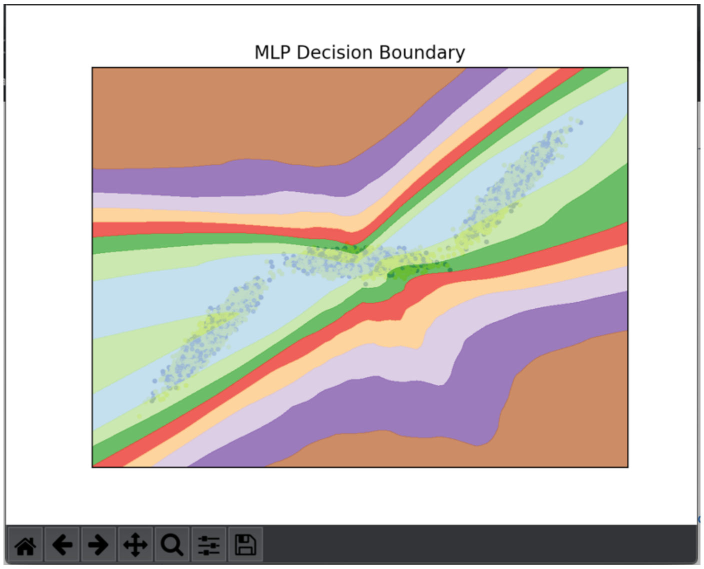
After running the algorithm on the given test data with the chosen hyperparameters from the cross validation exercise, we can see the following:

SVM Best accuracy score: 0.671875
MLP Best accuracy score: 0.65375

We see the final result is that SVM scores slightly better and MLP has scored slightly lower than expected. This could be due to the small training set compared to the test set of data. If these were given more samples to train, they would most likely fit more accurately to the final classification of data.

Question 2

Using similar methods to the previous question, we can run 10-fold cross validation on two different classifiers(GridSearchCV), this time both MLP and find the most accurate one. To do this, two different activation function have to be tested: *logistic* and *softplus*
Softplus is not supported in the MLPclassifier toolkit but 'tanh' is which has a larger curvature much like softplus does over logistic. When we run these two against each other we can see these results:



Best MLP parameters found for K= 10 :
 {'activation': 'logistic', 'alpha': 0.0001, 'hidden_layer_sizes': (50, 50, 50), 'learning_rate': 'constant', 'solver': 'sgd'}
Results on the MLP accuracy test set:
MLP mean_test_score: [0.69717172]
MLP std_test_score: [0.00027773]
MLP accuracy score: [1.0]

```
          precision    recall  f1-score   support

       1         1.00      1.00      1.00      1099

    accuracy                           1.00      1099
   macro avg     1.00      1.00      1.00      1099
weighted avg     1.00      1.00      1.00      1099
```

We can see that while running the models with K=10 we can get an accuracy of 100%. However, we can expect this to go down with the full test data set. It is also interesting that logistic has won for this group of data. Now we can run the classifier again, on the whole test data set, and see how well it does.

To determine this, we can see:

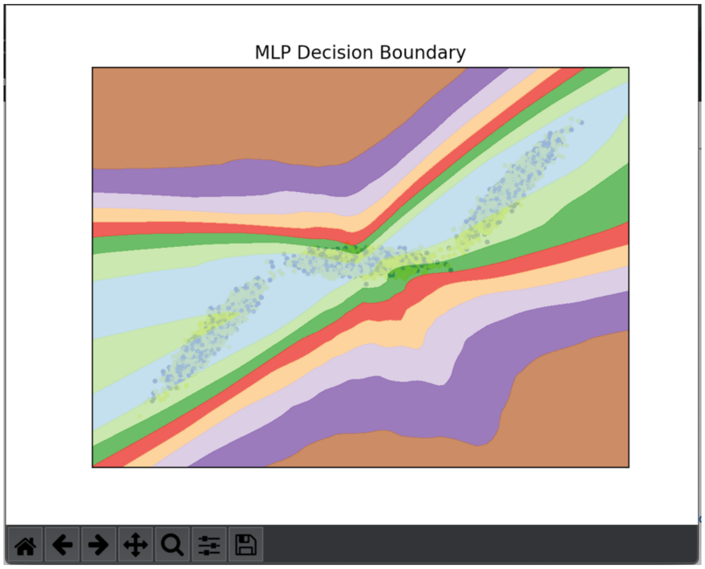
For the mean squared error, we can easily calculate that from the predicted values:

MLP Mean Squared Error: 0.27204545454545453

And for the best accuracy score based on model selection:

MLP Best accuracy score: 0.7279545454545454

The decision boundary is representative of this accuracy:


MLP Decision Boundary