Student Number: 215709

1. Approach

For this approach, it was decided that the most optimal option would be to make use of Scikit learn. Scikit learn has a machine learning method known as a Support Vector Machine. A Support Vector Machine is a supervised machine learning method that is commonly used for classification, regression, and outliers' detection. The main reason for choosing this machine learning method was due to the efficiency in a high dimensional space which is valuable due to the nature of the data provided to process. It was also chosen due to the simplicity of the code needed to get results with a high accuracy due to the creation of hyperplanes

2. Methods

2.1. Data Import

At the beginning of the program, libraries that are needed are imported and the data is imported from a Google drive. Each set of data is stored into a Pandas DataFrame.

Once the data has been retrieved, all the NaN values that were contained within the data are instead replaced with another value (-1 in this case) and converted into a NumPy array.

2.2. Data Retrieval

Once the data has been imported it must have the label and the confidence columns removed so that only the features remain. To do this I create a new variable and store all rows and columns excluding the last two columns into it from the training1 file and the training2 file. Once the features are retrieved, they are placed into another array using the concatenate function from NumPy.

This same data retrieval is done to the label column from both the training1 and training2 files. The label data from each file is stored into a separate variable and cast into an int type as they are stored as floats in the csv files which will not be compatible with the model. The two arrays are then concatenated together to form a new variable.

2.3. Data Split

So that testing can be done using the training1 and training2 data, a split is needed. To do this the train test split function is used from scikit learn. The features and the labels are split with an 80/20 split. 80% of the data is assigned to the training sample

2.4. Data Pre-Processing

Before creating the model, the data needs to be processed. First, a standard scalar is used, which will set the mean to zero and the variance to one. I used the fit function on the features from the training set.

The next step is to normalize the data sets by calling the transform function from my scalar. All the normalized data is stored into new variables. This is done so that any data can be used at a later point should it need to be.

The last piece of data pre-processing was a feature selector. The feature selector was used to select only the important features from both data sets to reduce the overall dimensionality. This should improve the accuracy by focusing in on the data that matters the most. To make this a random forest regressor was used with 50 estimators and a maximum allowed depth of 10. There was also a limit of 100 features as the code took a very long time to compile.

2.5. Model Creation

In creating a model, the best approach is to experiment with the hyperparameters. As such, four models were created with different kernels to see the impact of the accuracy of each different type. For all models besides one, the gamma and other hyperparameters were default.

Two models shared a kernel, but the gamma was adjusted for a single model to see how this compared. The three different kernel types where RBF, linear and polynomial. The scaling was different on the second RBF model which instead featured a gamma with a value of 0.0005 while the first RBF model used auto gamma scaling.

3. Results and Discussion

The Accuracy of all the models were recorded and stored into a new array called accuracy. The accuracy can then be interpreted into a graph so that it can be visualized. The accuracy of the model with the RBF kernel and a 0.0005 gamma scale had an accuracy of 72.5%, the highest of all the different models. The second highest model was the model that had the auto gamma RBF kernel at 72.2%. The next two models, with the linear and the polynomial kernels, had 70.4% and 70.9% accuracy ratings, putting them at the lower end of the four models.

The feature selection in the data pre-processing had little to no real positive impact on the overall accuracy but did have a significant impact on the runtime of the program.

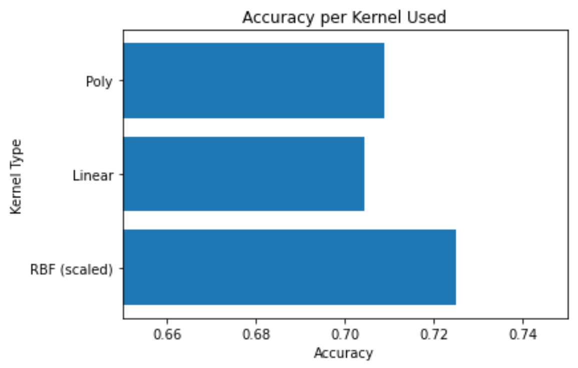

Figure 1: Bar chart showing the accuracy of different models using different kernels

It is possible that better results could be achieved by using different machine learning approaches such as a random forest which would use many decision trees to vote on the classification. This would be a more expensive approach computationally but perhaps could have yielded better results. It is also possible that giving the feature selector more time to run could have increased accuracy at the expense of time taken.

Better results could have potentially been achieved by creating the training data with more regard to the confidence, for example if a model had a confidence value of 1, then three people agreed and so it bears a stronger weight than a feature with only two out of the three people stating it would be memorable. A better approach to this could have been to duplicate this data so that it appears more frequently inside of the training set and has a higher weight in training. Ultimately it came down to computation time versus the accuracy gained and the choice was made that computation time was something to make as small as possible, especially when dealing with data that in high dimensionality.

By creating this Support Vector Machine, it is clear to see that they are very easy and quick to create in exchange for the high accuracy awarded by them and as such can be seen as a worthwhile approach for a classifier. They can also be improved on even more by performing more data pre-processing to make better use of the data supplied. There are also many other good approaches that may or may not have been as good as a Support Vector Classifier, but it has been proven that they are effective at classifying data in a high dimensionality with a not perfect approach to data pre-processing.