Machine Learning Practical 2 - Report

April 2019

Introduction

The purpose of the practical was to create and compare multiple classification algorithms on the provided dataset to classify based on another dataset and evaluate its performance. Several classification algorithms were run and compared.

To run the code, run *python3 P2.py*, the submission uses the following libraries: Pandas, Scikit Learn, Numpy, Statistics, Ast, Argparse, Scipy, Math, and Matplotlib. In the case of versioning issues, a requirements.txt file is attached

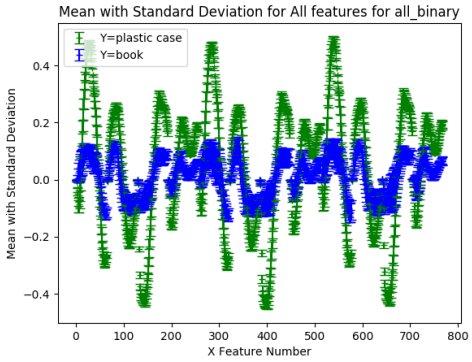
There were several classification algorithms examined. The first was a simple logistic regression, the second was a decision tree, the third was random forests, and the final algorithm explored was an SVM.

To generate the results of 100 iterations code, run *python3 P1.py -i*, this will generate the plots of the 100 iterations, each with a different random state for the train test split each time, and print the standard deviation and the mean of the accuracy. Though it should be noted that iterative mode may take a long time to finish as it runs all of the ML from the train test split onwards 100 times. The plots generated from this are included as well as all of the other plots in the plots directory, and there are 4 python files in the submission, the main P2.py file, models.py, plots.py, and utils.py.

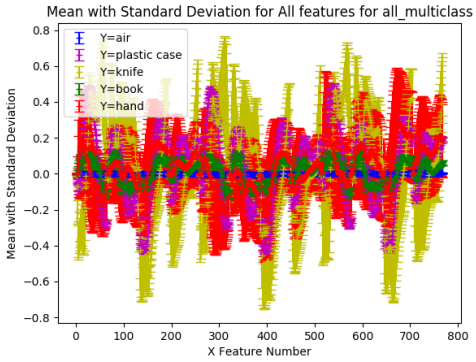
The classification results on the unknown dataset is located in the outputs directory, grouped by dataset type and features used, i.e. all features or a subset of features. Each of the CSV files contains the output from the 4 algorithms for that dataset and feature space. All of the plots used in the report at located in the plots folder. Both datasets are attached in their respective folders. The data is also included in the binaryTask and the multiClassTask folders.

Loading and Cleaning the Data

Pandas was used to load the data for the model, this was done to keep it in a format which scikit learn would be able to use, i.e. dataframes. With regards to cleaning the data, there was nothing required. There were no null values or empty values in the dataset thus the dataset did not have to be cleaned. No new features were created.



(a) Binary Classes Standard Deviation plotted against the mean



(b) Multi-Class Standard Deviation plotted against the mean

Figure 1: Initial attempt at visualising the features for both the binary and multi-class datasets using the mean and the standard deviation

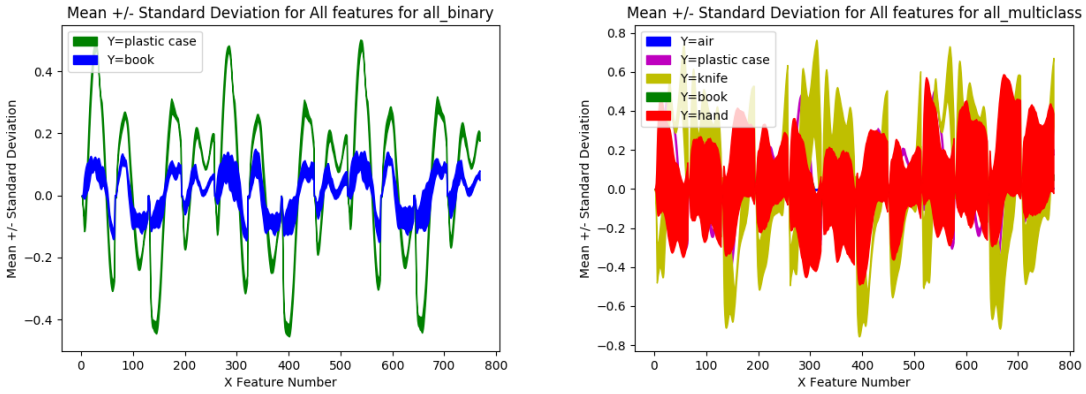
Analysing and Visualising the Data

Before any visualisation or analysis was performed on the dataset, the data was split into a training set and a test set, to avoid data leakage and not influence any decisions made in the training set. This was done using the train test split function in Scikit Learn, using the stratify option. This was performed as it was a simple approach which guaranteed random splits in the data when picking data. To get the results as close to reproducible as possible the random_state of 42 was used. It should be noted that despite this, the results may not exactly match, thus all of the results submitted are provided. The split used was 70% train, 30% test. Stratify was used as a parameter to ensure that there is a consistent split of classes in both the train and test, the exact split was the same as the input data, i.e. the Y output class.

After splitting the data into the train and the test data, the train data was visualised. This was challenging as the consisted of 768 input features, so multiple different visualisations were tried. The first was to look at the the mean across all of the samples for each input feature. This is shown in Figure 1, which was the original attempt at visualising the data. This shows the mean of each class, plus or minus the standard deviation. This is split by output class so there is different lines across features for each feature. This visualisation was difficult to gauge anything from, so instead of error bars fill was used between the mean plus the standard deviation, and the mean minus the standard deviation. This is visualised in Figure 2. For the binary data it appears the data is very nicely linearly separable, whereas for the multi-class data, some of the classes are being overshadowed by other classes, thus if Figure 1b is examined, it is apparent that the data is not easily linearly separable.

Preparing Inputs

To find any input featured which weren't useful, a plot was constructed for each X value for a Y output value of the Spearman Rank Coefficient and corresponding P values to examine correlation



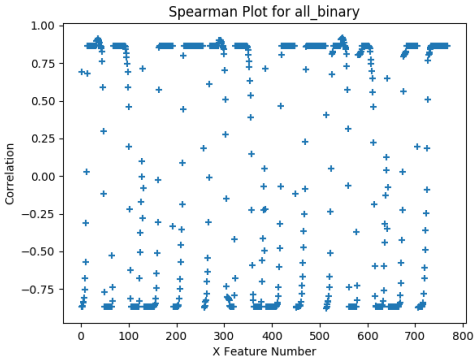
(a) Binary Classes Standard Deviation plotted against the mean

(b) Multi-Class Standard Deviation plotted against the mean

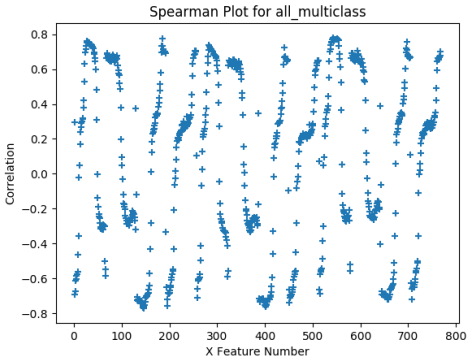
Figure 2: Second, more successful attempt at visualising the features for both the binary and multi-class datasets using the mean and the standard deviation

for each input feature to the Y's. This was done as Spearman can be used to assess how well the relationship between 2 variables can be described as a monotonic function, which perfectly can describe the relationship between the inputs and the outputs. P values were also plotted generated from the Spearman coefficient, where the null hypothesis was that there is no monotonic relationship between the 2 variables.

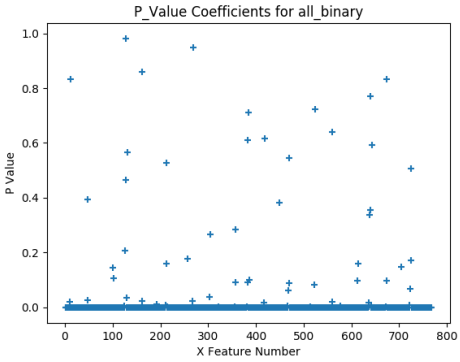
As shown in 3, for both binary and multi-class data, the Spearman coefficient is close to 0 for quite a number of features, and the p values are also high for a number of features as well, and following a similar methodology, some input features failed the p value test by being well above 0.1. This would suggest that some features are not related to the outputs at all, as the correlation is almost non existent. This suggests that some feature selection may be necessary. However, it should be noted that there are a very small number of samples, thus having all of the features may be beneficial to getting better results. Thus both approaches were tested, feature selecting and using all of the inputs.



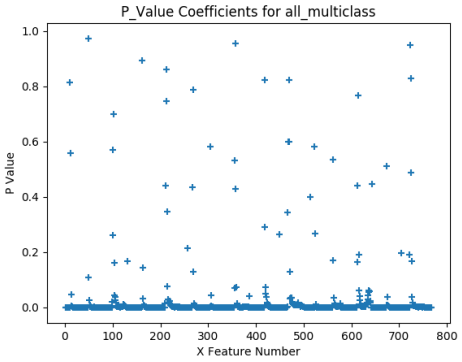
(a) Spearman Binary Plot



(b) Spearman Multiclass Plot



(c) P Value Binary Plot

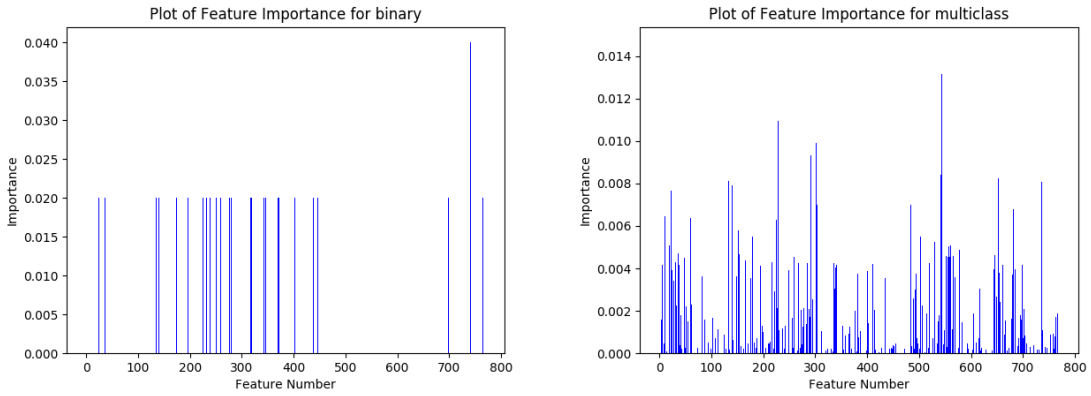


(d) P Value Multiclass Plot

Figure 3: Spearman coefficient and P value plots for features in both the binary and the multi-class datasets

Since there were too many features to select manually, an algorithm was used for feature selection. Due to the small number of samples, the features were picked based on the entire dataset, rather than just the training set after splitting. This would normally be picked only on the train set, but was instead trained on the entire dataset due to the small number of samples provided. There were still training and test sets used, but to calculate the features required, the features were picked across the dataset. The algorithm used was an extra trees classifier[1]. This was used as it would rank the feature by importance, and would take at maximum the square root of the number of features, it was set up with 50 estimators as a starting point following the example of feature importance selection, this is something which could be investigated in the future, and was not tries due to time constraints. The feature importance of any particular feature is the gini importance or mean decrease impurity. This is the total decrease in node impurity for a feature weighted by the probability of reaching specific nodes (which is approximated by the percentage of samples reaching that node) averaged over all trees used in the classifier. [1]. The feature importance graphs are shown Figure 4, and show a variety of features selected. In the binary, it is quite a big gap between features which are relevant,

[1]https://scikit-learn.org/stable/modules/feature$_s$*election.html*



(a) Binary Feature Importance

(b) Multi-class Feature Importance

Figure 4: Feature Importance graphs for binary and multi-class dataset features

and features which aren't, whereas there seems to be more differences in the multi-class data.

Feature scaling was not performed as all of the features appeared on the same scale, a basis expansion was not thought relevant for the problem due to the large feature set combined with the small sample size.

Selection and Training of a ML Model

There were several different classification models selected. All of the implementations used were from the scikit library. The first was a simple logistic regression model. This was chosen as it was the simplest possible model which can perform classification, and the approach with this was just to classify out of the box. By default because of the optimisation algorithm picked, L2 regularisation was performed by the algorithm. The default solver optimisation algorithm was lbfgs, which is the Limited memory implementation of the Broyden–Fletcher–Goldfarb–Shanno algorithm which itself is an iterative approach to try to converge onto a stationary point, i.e. the minimum of the cost function. The maximum number of iterations for this was set to 200, to ensure that the solver could converge. It was set to automatically classify multi-label classes.

The other models chosen were decision trees, random forests, and support vector machines. Decision trees and random forests were both chosen as they are a different type of classification algorithm, and support vector machines was a comparison on another type of algorithm. Decision trees was chosen as it was the simplest tree based algorithm. This works by creating a tree like structure based on decisions made on the input features, this allows prediction by moving the input through the tree, based on the values of the input features and the conditions at each node. These partition the feature space to build boundaries which can be represented as nodes. Random forests work by having multiple trees created from samples randomly taken from the data. This is repeated for multiple samples and the average is taken as the result of a probability class.

The parameters used for both the random forests and the decision trees were the default parameters,

the number of estimators in the random forests was set to the default 10. The quality of a split was determined by the "Gini" split, which is the same parameter used by the feature importance mechanism.

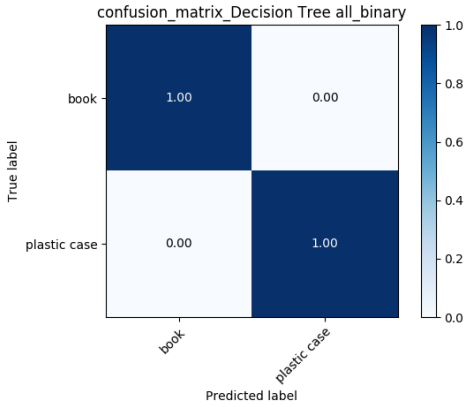
Support vector machines were tried as an extension of the logistic regression, specifically a maximal margin classifier which fits a hyperplane across the input feature space to create decision boundaries between output classes. SVMs were explored as they made a classifier which differed from both of the tree based classifiers and was different from logistic regression. The kernel used was the radial basis function kernel, which was the default used, this puts the features into a higher dimensional feature space in order to build better decision boundaries. The gamma, or the coefficient of the kernel, used was also the default or "scale". This was used as it was the suggested default, and was defined as the parameter of the kernel. Given more time, the best "gamma" and "c" values could be found for this dataset through experimentation.

The training data was then used to train the models, this used the "fit" function in Scikit Learn. After training, the model was used to predict on the test data using the "predict" function provided by the Scikit interface, and the result was compared to the actual output test set. The metrics used to compare the accuracy of the models was the accuracy score[2]. The confusion matrix was also outputted for each examples, this is a measure of how much of the predicted output is classified correctly, v.s. overall how much of the prediction is incorrectly classified as another class.

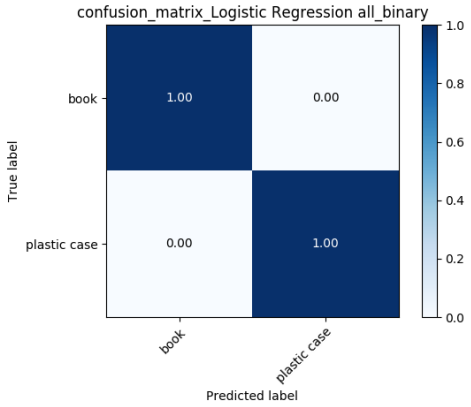
Evaluation of Performance

To evaluate each of the models, each regression was run 100 times with different sections of the overall data used for the train test split, and the results shown in 9 and the mean and standard deviation of the accuracy (3 d.p.) are shown in Tables 1 and 2. This was done to ensure that the results would be the most accurate they could be and not biased either way by an (un)favourable split in the data. The (normalised) confusion matrices from running it with the random state of 42 are also shown in Figures 5, 6, 7, and 8.

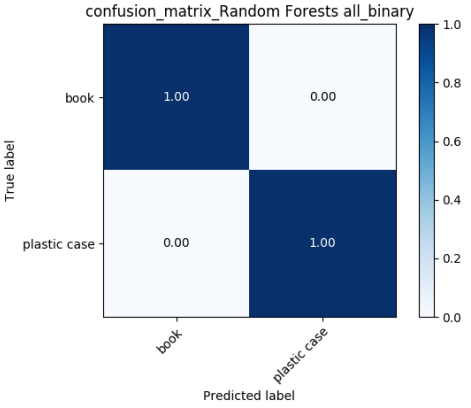
[2]https://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy$_s core.html$



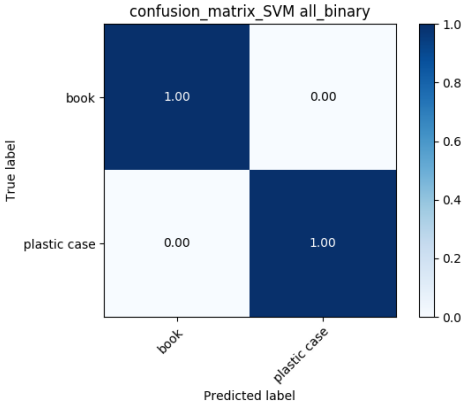
(a) Decision Tree



(b) Logistic Regression

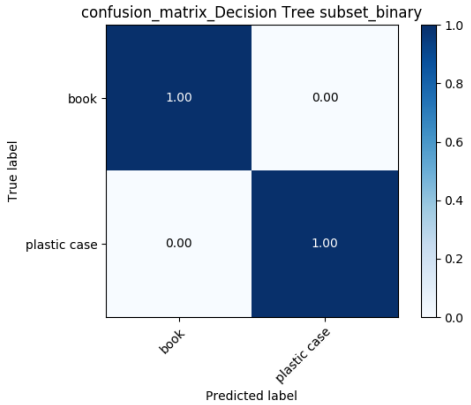


(c) Random Forests

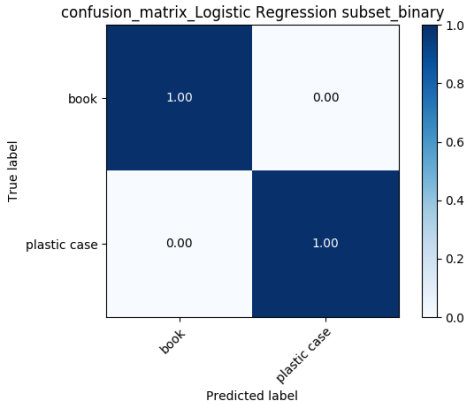


(d) SVM

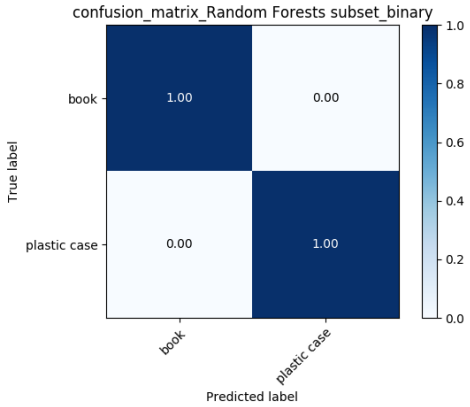
Figure 5: Example of confusion matrices for all classifiers using all features on the Binary dataset



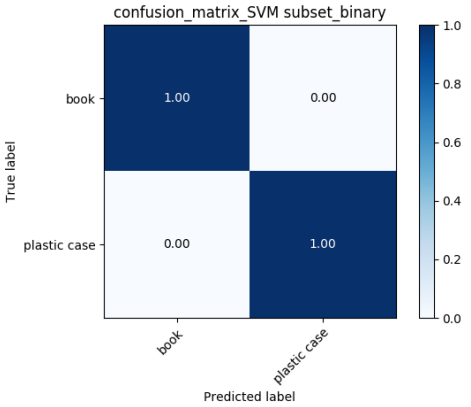
(a) Decision Tree



(b) Logistic Regression

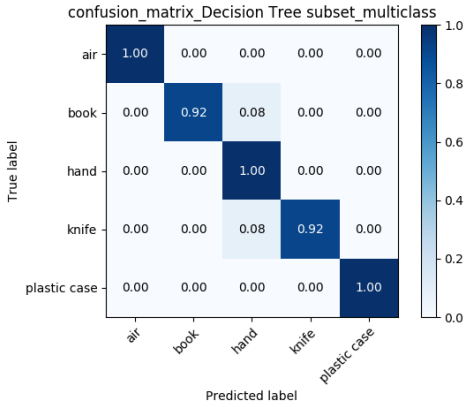


(c) Random Forests

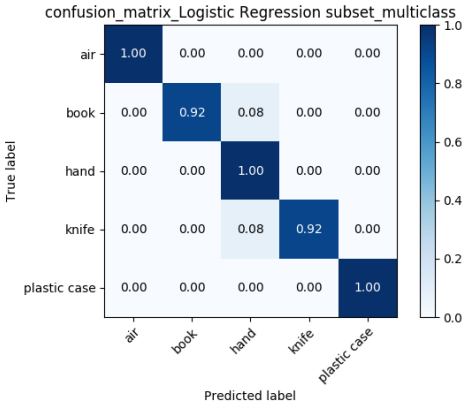


(d) SVM

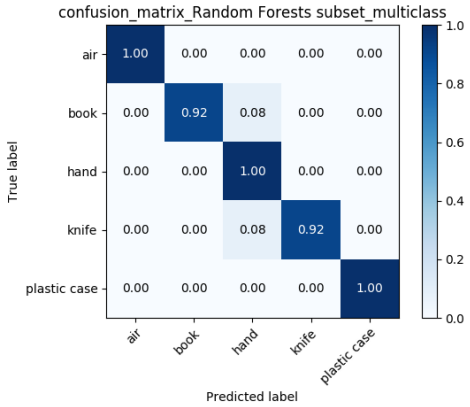
Figure 6: Example of confusion matrices for all classifiers using a subset of features on the Binary dataset



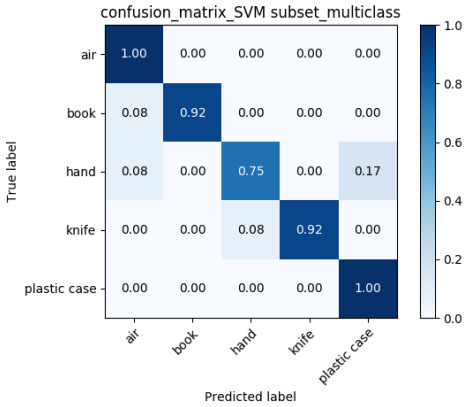
(a) Decision Tree



(b) Logistic Regression

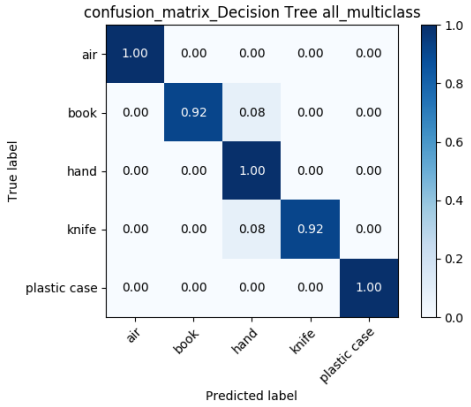


(c) Random Forests

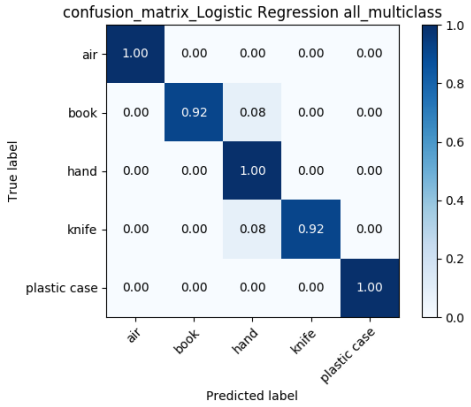


(d) SVM

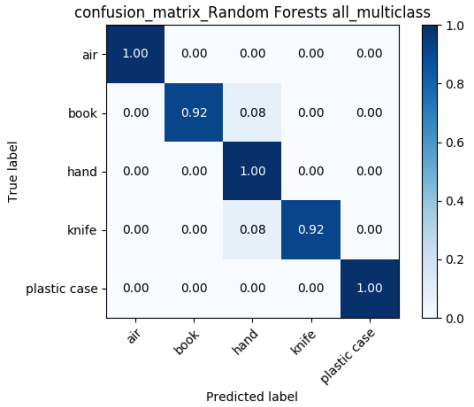
Figure 7: Example of confusion matrices for all classifiers using a subset of features on the multi-class dataset



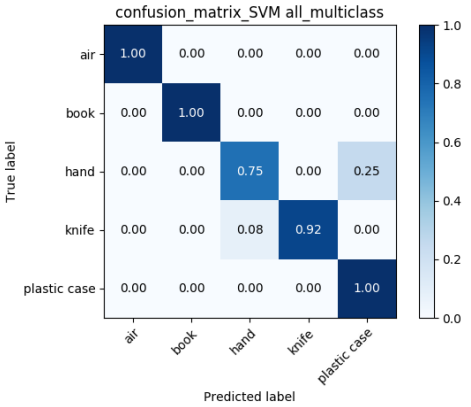
(a) Decision Tree



(b) Logistic Regression

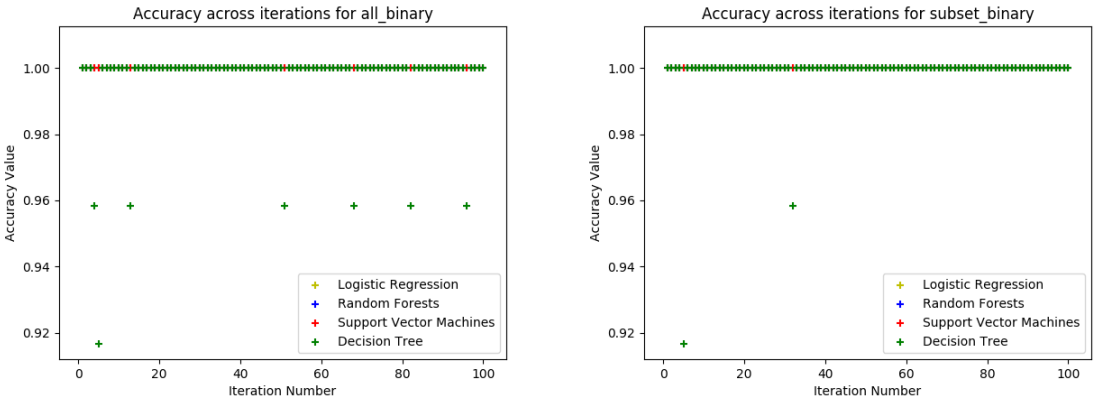


(c) Random Forests

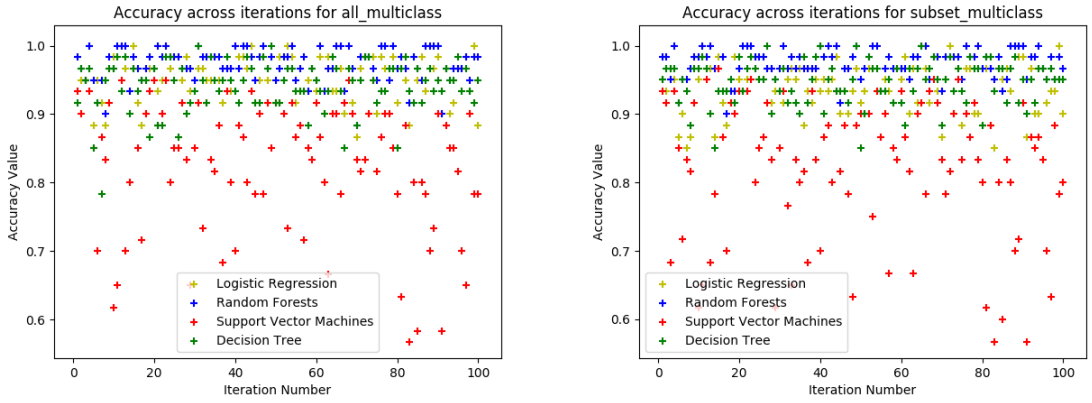


(d) SVM

Figure 8: Example of confusion matrices for all classifiers using all of the features on the multi-class dataset



(a) Accuracy Comparison for Binary Dataset



(b) Accuracy Comparison for Multi-class Dataset, left is using all features, the right is using a subset of features

Figure 9: Accuracy Values for binary and Multi-class dataset across 100 iterations of running the program for a subset of features vs all features, left is all features, and the right hand side is a subset of features

Algorithm	Binary Standard Deviation Accuracy	Binary Mean Accuracy	Multi-Class Standard Deviation Accuracy	Multi-Class Mean Accuracy
Logistic Regression	0.0	1.0	0.034	0.953
Random Forests	0.0	1.0	0.026	0.970
SVMs	0.0	1.0	0.095	0.831
Decision Trees	0.015	0.997	0.036	0.943

Table 1: Table of Binary and Multi-class Standard Deviation and Mean values of each model's accuracy running over 100 iterations using all features

Algorithm	Binary Standard Deviation Accuracy	Binary Mean Accuracy	Multi-Class Standard Deviation Accuracy	Multi-Class Mean Accuracy
Logistic Regression	0.0	1.0	0.037	0.941
Random Forests	0.0	1.0	0.025	0.975
SVMs	0.0	1.0	0.095	0.836
Decision Trees	0.0	1.0	0.034	0.948

Table 2: Table of Binary and Multi-class Standard Deviation and Mean values of each model's accuracy running over 100 iterations using a subset of features

The metric used was the accuracy, which is a simple and efficient way of seeing how far apart the prediction is from the actual value, by looking at the number of accurately predicted values. This was also calculated using Scikit Learns' Accuracy Score function for simplicity. The mean was calculated using the python statistics module, and the standard deviation calculated using numpy's standard deviation function. The confusion matrices are matrices which show how accurately the results are classified. This is shown by how often each class is predicted correctly, vs how often it is predicted as another class.

Looking at the binary results, for the random state of 42, all of the confusion matrices shown in Figures 5 and 6 suggest that all of the classes have been correctly identified using both a subset of features and using all of the features by all of the classification algorithms used, as all classes have perfect prediction rates of 1. This is backed up by the 100 iterations model, in Figure 9a, almost everything has a perfect accuracy score, apart from a couple of decision tree values. This results is mirrored in Tables 1 and 2, where the accuracy score is a perfect 1.0 for both a subset of features and all features with the standard deviation being 0 (apart from the small fluctuation for the decision tree which leads to an average of 0.997 and a very small standard deviation). This is perhaps to be expected as shown in Figure 2a, the data appears linearly separable.

Comparing the results on the unseen data, using the random state of 42, both of the results are the same for all classifiers, the first 10 samples are identified as a "1" or a plastic case, and the next 10 samples are identified as "0" or a book. Comparing these results to those achieved by other students, the results appear to be the same. Comparing the results between using all features and a subset of features, the results are the same.

The multi-class results yield several differences however, examining the confusion matrices in Figures 7 and 8, it becomes apparent that none of the classes were perfectly able to distinguish all of the objects. For both using a subset of features and all features, there were several samples which were incorrectly classified. For SVMs there appear to be more both in the subset of features and using all features which are incorrectly classified percentage wise compared to the other three algorithms. Figure 9b also shows differences in classification, where there is significantly more deviation in the results in terms of the algorithms themselves. For the most part, accuracy for three out of the four classification algorithms is higher than 90% on average, apart from one or two points for the decision trees. SVMs however, seemed to be scattered far and wide, suggesting that it is dependant on the split to get a good result. This is manifested in the mean and the standard deviations being better for the other three algorithms compared to SVMs. Random Forests, Decision Trees, and Logistic

Regression all achieve accuracies of nearly 95% for both a smaller subset of features whereas SVMs are under 85% accuracy the standard deviation being almost a third more than the other algorithms.

Comparing the results on the unseen data, using the random state of 42, three the algorithms achieve the same results. SVMs are almost identical to the other three, except it classifies six of the first ten samples differently. For the three matching results the first 10 samples are identified as a "2" or hand, and the next 10 samples are identified as "0" or a air, the next 10 are "3" or knife, the next 10 are "1" or book, and the final 10 are "4" or plastic case. Comparing these results to those achieved by other students, the results appear to be the same as achieved by the three algorithms suggesting SVMs is the algorithm achieving imperfect results. Looking at the subset of features compared to all of the features, a single extra value appears incorrectly classified for logistic regression as well as the SVM values not being accurate for the first set of samples.

Appendices A and B show the results across the classifications of the algorithms. SVMs outputs were inconsistent and thus not part of the final output, but the other three algorithms were used from the all features output.

Critical Discussion of Results and Approach

The discussion of results is split into the binary and the multi-class datasets.

Looking at the binary results, the accuracy rates and confusion matrices suggested that all of the algorithms were able to classify all of the data accurately. This may be due to the data itself being easier to classify, for example as shown in Figure 2a, the data appears easily linearly separable, which would allow algorithms such as SVMs and logistic regression to work very well, and the tree based approach performed by decision trees and random forests would be able to separate the classes well. This accuracy is also manifested on the unseen data where all of the algorithms agreed on the output for both all features and a subset of features, which would suggest that the model was not being overfitted.

In Multi-class however, it is apparent in both the accuracy rates and the confusion matrices, that SVMs perform poorly compared to other algorithms. This could potentially be due to the multiple classes not being linearly separable. This is demonstrated in Figure 2b, where one of the classes completely masks the other classes, demonstrating that the data is not as easily separable. SVMs may struggle to create a working hyperplane in this scenario, and this is manifested in the accuracy being much less than the other values. Looking at the other classifiers, they were all much more accurate, recording accuracy rates of 95% or above, which suggests that they were able to consistently learn to split these classes a lot better. As shown in Figure 9b, the split of the data does not seem to affect the other classifiers as much as the SVMs, which would suggest that classification for the other classifiers is genuine and not a fluke created from a favourable split.

All of the models seemed to agree with each other on the results for the unknown data for all features, apart from SVMs for a small section of multi-class which may be due to the data not being perfectly linearly separable thus making a hyperplane difficult. The results achieved by the other three classifiers seem to match the results found by other students.

Examining the differences between using a subset of features and all of the features, there is almost

no difference for the results both in terms of the confusion matrices or the average accuracy. Firstly this suggests that the approach to feature selection was accurate, as it does seem to be finding the most relevant features. Secondly, if the dataset were to get larger, i.e. have more samples, it would make much more sense to only use a subset of the features for time reasons, as it just may take too long to learn on all of the features if the number of samples increases. The subset of features also produces very similar results on the unseen data as using all of the features, which further suggests that if the problem size were to increase, it would make much more sense to choose a subset of the features.

The approach to solving this problem was to look at several models of varying complexity from simpler Logistic Regression models to more complicated Support Vector Machine and Random Forests. This was explored as it would give multiple strategies in performing classification. The models were tested using both all of the features and a subset of features. This was partially to see if there was a reduction in accuracy using a subset of features due to the small sample set, however it was found that there was almost no difference between using all vs using a subset of features, thus if the dataset were to increase in size, it would almost be obvious to use a subset instead of all of the features for time purposes. The approach to splitting the data was dependant on randomly splitting the data, though maintaining the ratios of classes in the split. which was the best you can often do, however you can potentially run the risk of the train set being different from the test set even with randomness that the results on the training set may not be representative. The results for both the binary and the multi-class dataset suggested the classification algorithms were successful in classifying based on the data provided, a fact backed up by the fact that the unseen data results were uniform for binary, and only slightly differed for SVMs for multi-class, and matched other results found by other students.

One of the factors I would have done if I had more time would be to explore if a higher accuracy could be achieved by tuning the hyperparamaters of the algorithms, e.g. different gamma values for support vector machines, or changing the number of estimators for Random forests. This was not done due to time constraints, and thus in most of the places, the default values had to be used.

Conclusion

In conclusion, I feel I was able to complete the practical fully, I was able to train multiple classification algorithms with successful results.

A Binary Dataset Predictions (collated from both some features and all features)

Sample Number	Predicted Class Number	Label Name
1	1	Plastic Case
2	1	Plastic Case
3	1	Plastic Case
4	1	Plastic Case
5	1	Plastic Case
6	1	Plastic Case
7	1	Plastic Case
8	1	Plastic Case
9	1	Plastic Case
10	1	Plastic Case
11	0	Book
12	0	Book
13	0	Book
14	0	Book
15	0	Book
16	0	Book
17	0	Book
18	0	Book
19	0	Book
20	0	Book

B Multi-Class Dataset All Features

This is taking the most common values across a subset of features and all features, disregarding classifications and SVM results.

Sample Number	Predicted Class Number	Label Name
1	2	Hand
2	2	Hand
3	2	Hand
4	2	Hand
5	2	Hand
6	2	Hand
7	2	Hand
8	2	Hand
9	2	Hand
10	2	Hand
11	0	Air
12	0	Air
13	0	Air
14	0	Air
15	0	Air
16	0	Air
17	0	Air
18	0	Air
19	0	Air
20	0	Air
21	3	Knife
22	3	Knife
23	3	Knife
24	3	Knife
25	3	Knife
26	3	Knife
27	3	Knife
28	3	Knife
29	3	Knife
30	3	Knife
31	1	Book
32	1	Book
33	1	Book
34	1	Book
35	1	Book
36	1	Book
37	1	Book
38	1	Book
39	1	Book
40	1	Book
41	4	Plastic Case
42	4	Plastic Case
43	4	Plastic Case
44	4	Plastic Case
45	4	Plastic Case
46	4	Plastic Case
47	4	Plastic Case
48	4	Plastic Case
49	4	Plastic Case
50	4	Plastic Case

References

[1] Louppe, G. How are feature_importances in randomforestclassifier determined?