University of St Andrews

Practical 2

Classification of object colour using optical spectroscopy

Student ID: 120022067

University of St Andrews

CS5014 Machine Learning

Contents

List of Figures

List of Tables

Listings

1 Introduction

The aim of this assignment is to gain experience in working with real experimental, imperfect, and limited data which has not been analysed before. In order to achieve this, a classification model needs to be designed, implemented and trained on the provided dataset.

1.1 Software and Libraries

This section covers the third party software and libraries used in this assignment.

1.1.1 Graphviz

This open source graph visualization software is used to visualise the decision tree classifier.

1.1.2 Pandas

Pandas is used to load the data and perform analysis.

1.1.3 Matplotlib

The nature of this practical requires a tool to visualise the data and results. This python package serves that purpose.

1.1.4 Jupyer

This is a useful package for doing analysis on the dataset.

1.1.5 Scikit-learn

This package provides several classification algorithms. This allows training of classification models to be performed with ease.

2 Binary Dataset

The preprocessing steps performed on the binary dataset are explained in the subsections below.

2.1 Loading Data

The code in Listing 1 is used to load the dataset as pandas DataFrame. In this case, wavelength values are used as headers of the features. The headers are in the form wl_[wavelength]. This makes it easier to identify each feature in the dataset.

```
# load the data and assign the headers
wavelength = pd.read_csv(data_path+"Wavelength.csv", header=None)
wavelength_list = []
for wl in wavelength[0]:
    name = "wl_" + str(wl)
    wavelength_list.append(name)
df = pd.read_csv(data_path+'X.csv', header=None, names=wavelength_list)
y = pd.read_csv(data_path+'y.csv', header=None)
XToClassify = pd.read_csv(data_path+'XToClassify.csv', header=None, names=
    wavelength_list)
df['y'] = y
```

Listing 1: The code used to load the dataset.

2.2 Data Structure Analysis

All analysis processes were done in jupyter notebook. The codes can be found in ***analysis.ipynb***.

	wl_420.852	wl_421.228	wl_421.605	wl_421.98199999999997	wl_422.358	wl_422.735	wl_423.111	wl_423.48800000000006	wl_423.86400000000003	wl_424
count	180.000000	180.000000	180.000000	180.000000	180.000000	180.000000	180.000000	180.000000	180.000000	180.000
mean	-7.285778	26.623167	-39.141444	31.432056	19.583389	-27.309889	-1.363278	16.758222	-1.462556	20.104
std	28.066784	38.301483	38.125418	16.871401	19.078221	25.123531	28.830671	24.518862	24.424941	22.169
min	-74.140000	-92.590000	-143.410000	-13.050000	-25.910000	-87.240000	-72.790000	-64.190000	-79.430000	-44.400
25%	-24.740000	2.035000	-63.590000	19.257500	6.325000	-41.460000	-20.357500	2.350000	-15.630000	5.200
50%	-8.725000	28.960000	-35.035000	31.315000	18.445000	-27.265000	-1.695000	16.980000	0.300000	19.925
75%	11.212500	52.927500	-12.890000	44.292500	34.950000	-8.497500	19.560000	35.207500	14.840000	33.822
max	71.960000	145.480000	58.810000	72.410000	66.890000	47.220000	82.840000	79.350000	71.910000	75.150

Figure 1: Summary of each numerical attribute in the binary dataset.

The summary in Figure 1 shows that there is no missing value in the dataset (value of count matches the number of rows).

2.3 Cleaning the Data

There is not a lot of data cleaning task involve because the provided dataset is already cleaned. There is no missing value or mismatched value found within the dataset.

2.4 Splitting Train/Test Set

```
# split training set and testing set (80/20 ratio)
print("Splitting train/test set")
split = StratifiedShuffleSplit(n_splits=1, test_size=0.2, random_state=42)
for train_index, test_index in split.split(df, df["y"]):
    train_set = df.loc[train_index]
    test_set = df.loc[test_index]
```

Listing 2: The code used to split the dataset into train/test sets.

For this practical, the size of the test set is 20% of the dataset. The rest of the data will be used to train the model. The code shown in Listing 2 is used to split the dataset into train/test sets. **StratifiedShuffleSplit** is used to ensure that training set and testing set are representative of the classes of colours in the whole dataset i.e. the test set should have colour classes proportions identical to those in the full dataset.

2.5 Feature Selection

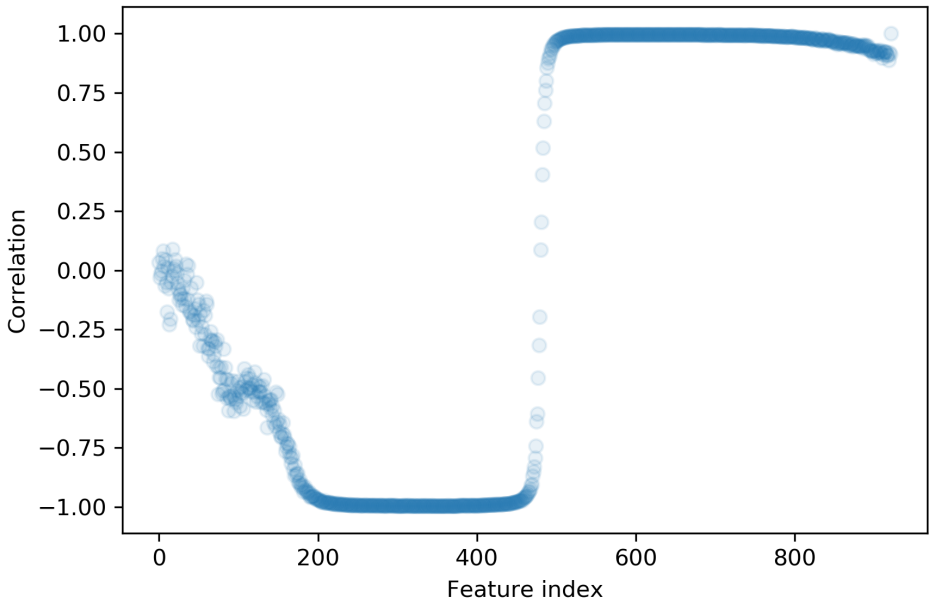

Figure 2: Correlation between each attribute and colour class.

Feature selection step is necessary because it can reduce overfitting, improve accuracy, and reduce training time (Brownlee 2016). Another reason to use only a subset of features is because large number of features are difficult to visualise. Feature selection method used is Recursive Feature Elimination (RFE). According to Brownlee (2016), RFE recursively removes attributes and builds a model on those attributes that remain. The correlation plot in Figure 2 suggests that most features are highly correlate with the colour class (most have value greater than 0.9). This indicates that most of these features are good predictors. Therefore, we will only use top 5 features output by RFE algorithm. The features selected by RFE are:

- wl_738.383
- wl_742.1239999999999
- wl_746.877
- wl_748.9110000000001
- wl_749.5889999999999

```
# use logistic regression as an estimator
model = LogisticRegression()
rfe = RFE(model, n_features)
fit = rfe.fit(train_set.drop("y", axis=1), train_set["y"])
selected_features = []
for feature, selected in zip(feature_names, fit.support_):
  if selected:
    selected_features.append(feature)
return selected_features
```

Listing 3: The code used to select features with RFE algorithm.

Figure 3 shows the scatter matrix of the training set after feature selection process. It is clear that the colour classes are linearly separable. It may even be possible to build a classifier with only one feature. However, all 5 features will be kept because classification model such as neural network may not converge if insufficient features are used.

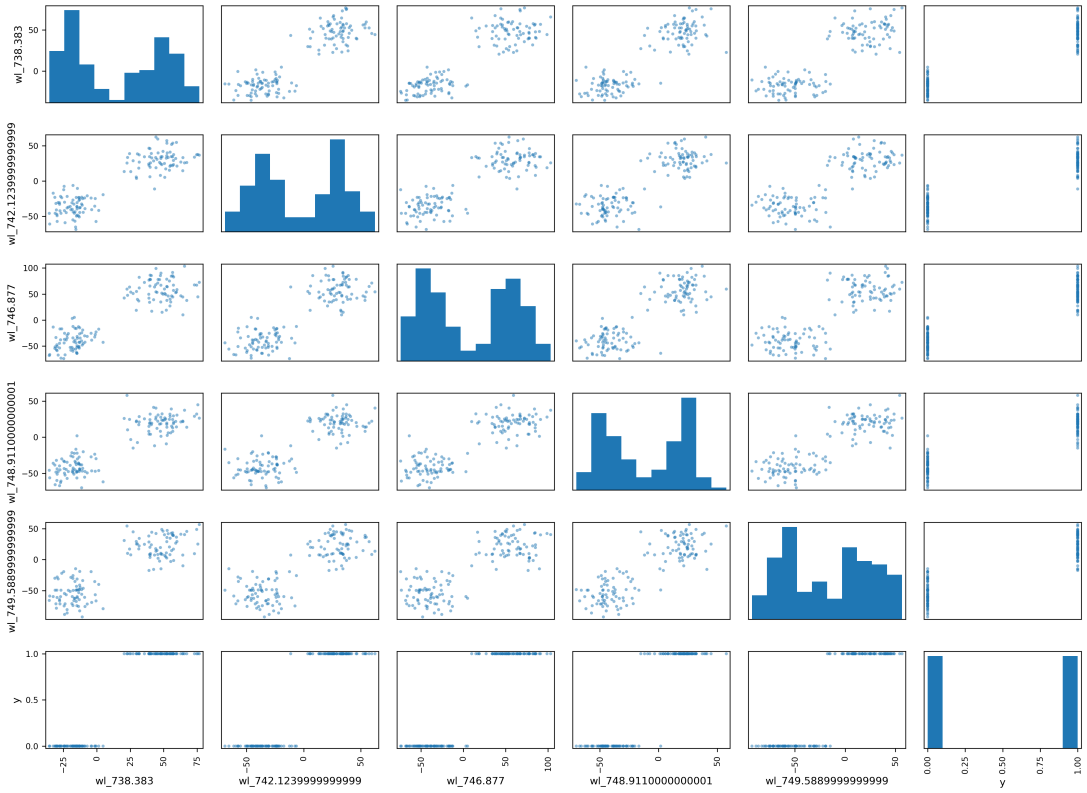

Figure 3: Scatter matrix of the training set.

2.6 Feature Scaling

This transformation needs to be applied to the numerical features because classification algorithms such as Support Vector Machine are sensitive to the feature scales (Géron 2017). Therefore, standardization was applied to the numerical attributes. The test set is standardised using the mean/variance obtained from the training set. Standardization was chosen over min-max scaling because it is much less affected by outliers (Géron 2017).

2.7 Saving Data

All preprocessed data are saved in **prepared_data/binary** folder as text files so that they can be loaded into classification models with ease.

3 Multiclass Dataset

The preprocessing steps performed on the multiclass dataset are explained in the subsections below.

3.1 Loading Data

The method for loading the data was discussed in Section 2.1

3.2 Data Structure Analysis

	wl_420.852	wl_421.228	wl_421.605	wl_421.98199999999997	wl_422.358	wl_422.735	wl_423.111	wl_423.48800000000006	wl_423.86400000000003	wl_424
count	450.000000	450.000000	450.000000	450.000000	450.000000	450.000000	450.000000	450.000000	450.000000	450.000
mean	0.831778	36.078489	-29.759578	36.237622	24.554111	-19.815911	5.518956	24.440200	6.366400	27.666
std	28.661154	38.447405	40.636535	17.290104	20.467866	25.229021	29.500466	26.104627	26.060944	24.427
min	-74.140000	-92.590000	-143.410000	-13.050000	-32.810000	-87.240000	-72.790000	-64.190000	-79.430000	-44.400
25%	-18.747500	9.965000	-56.740000	25.737500	10.487500	-36.810000	-14.835000	7.092500	-11.870000	10.972
50%	0.620000	37.705000	-29.555000	35.720000	24.200000	-20.545000	5.030000	25.905000	5.090000	27.085
75%	19.420000	60.480000	-2.220000	47.747500	38.565000	-4.895000	27.130000	43.212500	23.150000	43.130
max	75.490000	157.780000	99.110000	97.300000	81.810000	49.170000	85.030000	111.630000	109.890000	119.380

8 rows × 922 columns

Figure 4: Summary of each numerical attribute in the multiclass dataset.

The summary in Figure 4 shows that there is no missing value in the dataset (value of count matches the number of rows).

3.3 Cleaning the Data

Similar to binary dataset, there is not a lot of data cleaning task involve because the provided dataset is already cleaned. There is no missing value or mismatched value found within the dataset.

3.4 Splitting Train/Test Set

The data splitting method is the same as the method used to split binary dataset (see Section 2.4).

3.5 Feature Selection

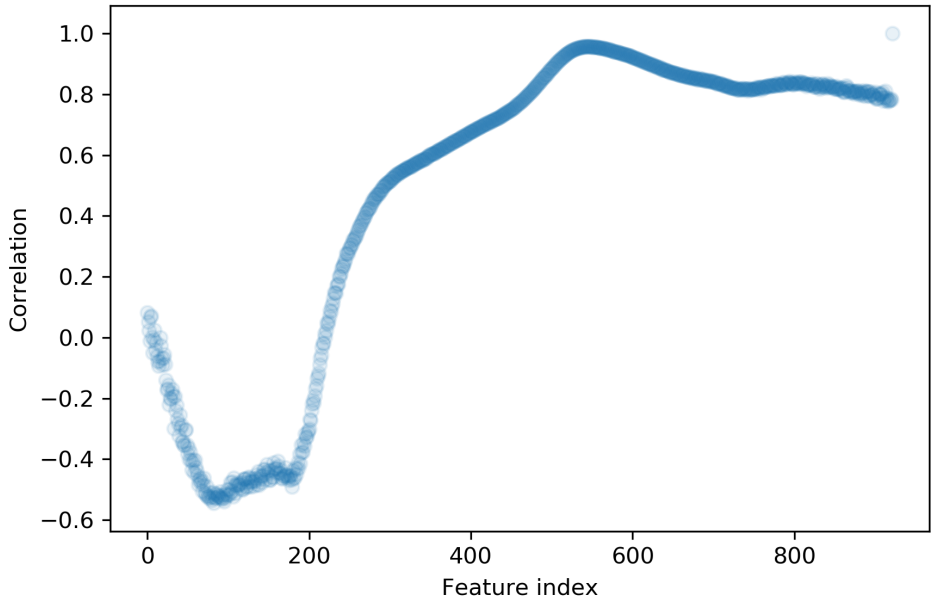

Figure 5: Correlation between each attribute and colour class.

RFE is used to perform feature selection again. The correlation plot in Figure 5 also suggests that most features are still highly correlate with the colour class. Although the correlations are not as high as the binary dataset, it is still a good indication that many of these features are good predictors. Therefore, we will only use top 5 features output by RFE algorithm. The features selected by RFE are:

- wl_453.871
- wl_454.99199999999996
- wl_518.706
- wl_574.432
- wl_634.521

Figure 6 shows the scatter matrix of the training set after feature selection process. It seems that these features are not perfect separators of classes like the binary dataset. However, these features should be sufficient to build a good classification model.

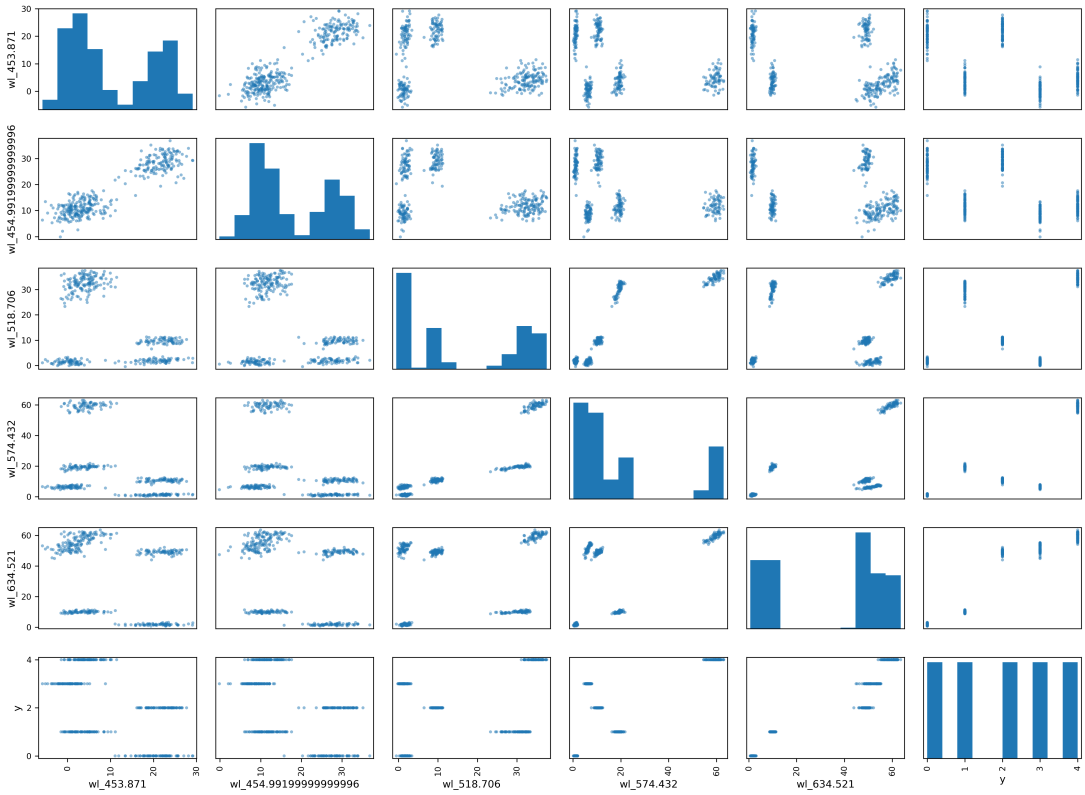

Figure 6: Scatter matrix of the training set.

3.6 Feature Scaling

Like binary dataset, standardization was applied to the numerical attributes. The test set is standardised using the mean/variance obtained from the training set.

3.7 Saving Data

All preprocessed data are saved in **prepared_data/multiclass** folder as text files so that they can be loaded into classification models with ease.

4 Training Models

For this practical, four classification models are compared against each other. The models are logistic regression, decision tree classifier, Multi-layer Perceptron, and Linear SVM. All model objects are saved in ***models*** directory after training. The code used to train the models can be found in **/src/train_model.py**.

4.1 Performance Measures

In order to compare the performance of each classification model, different performance measures are used here: 5-fold cross validation score (train set), accuracy score, and confu-

sion matrix. The performance measures are obtained by using sklearn's **cross_val_score**, **accuracy_score**, and **confusion_matrix**.

4.2 Logistic Regression

Logistic regression can be used to estimate the probability that an instance belongs to a specific class (Géron 2017). If the probability is greater than 50% then the algorithm classify that the instance belongs to that class (Géron 2017). This model can also be used in multiclass classification because logistic regression in sklearn uses one-vs-rest (OvR) scheme to deal with multiclass dataset.

Dataset	Train CV score (avg)	Test accuracy score
Binary	1.0	1.0
Multiclass	1.0	1.0

Table 1: The performance of logistic regression model

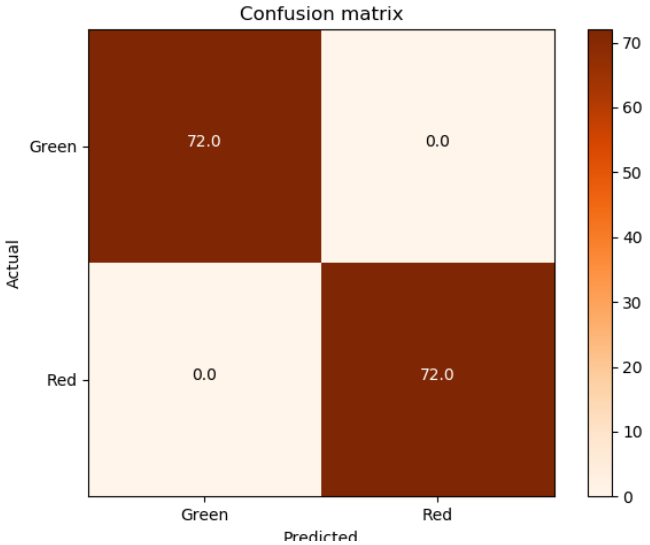

Figure 7: Confusion matrix of binary train set (logistic regression).

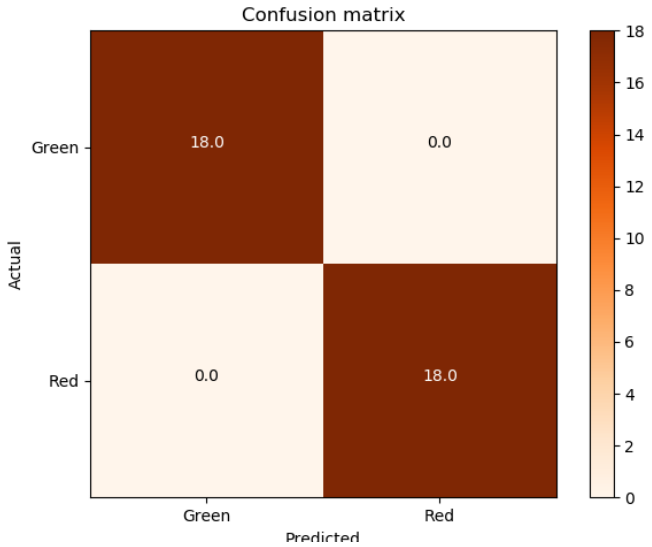

Figure 8: Confusion matrix of binary test set (logistic regression).

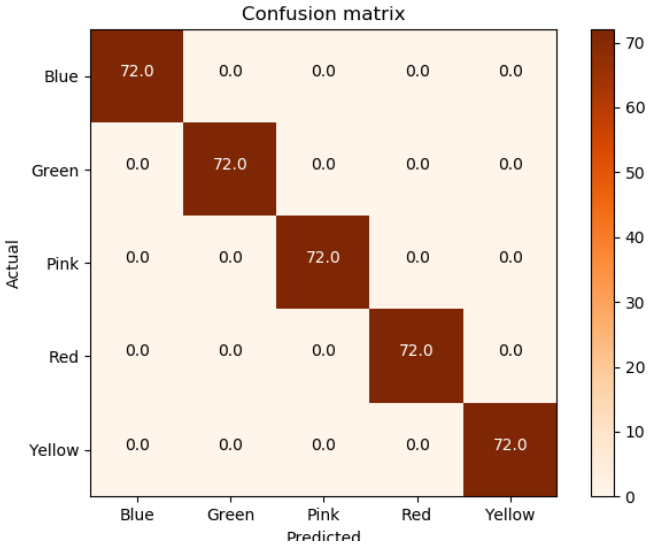

Figure 9: Confusion matrix of multiclass train set (logistic regression).

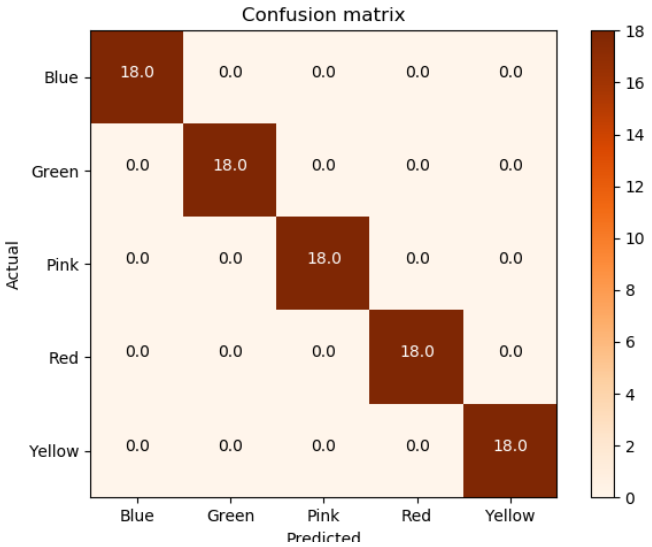

Figure 10: Confusion matrix of multiclass test set (logistic regression).

The performance of logistic regression model can be found in Table 1. The confusion matrices are shown in Figure 7, Figure 8, Figure 9, and Figure 10. Logistic regression model was able to classify all colour classes correctly for both binary and multiclass datasets. These results are expected because the visualisations in Section 2.5 and Section 3.5 show that selected features are good separator for the colour classes.

4.3 Decision Tree

Decision tree is a machine learning algorithm capable of performing classification task (Géron 2017). It was selected because it can fit complex datasets.

Dataset	Train CV score (avg)	Test accuracy score
Binary	0.99	1.0
Multiclass	0.99	1.0

Table 2: The performance of decision tree model

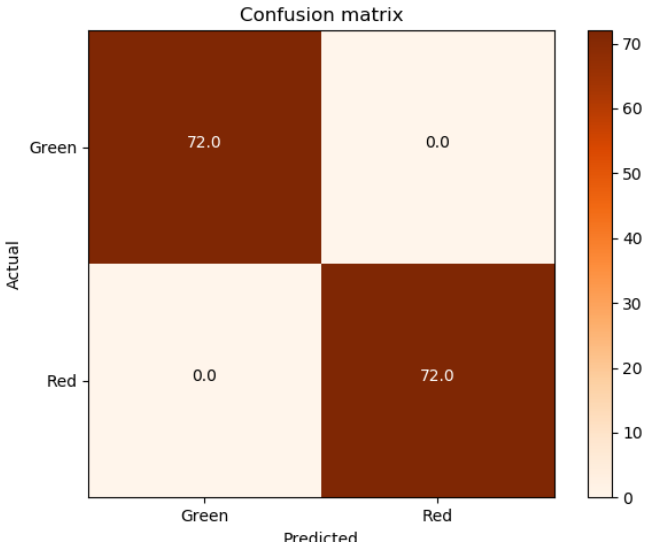

Figure 11: Confusion matrix of binary train set (decision tree).

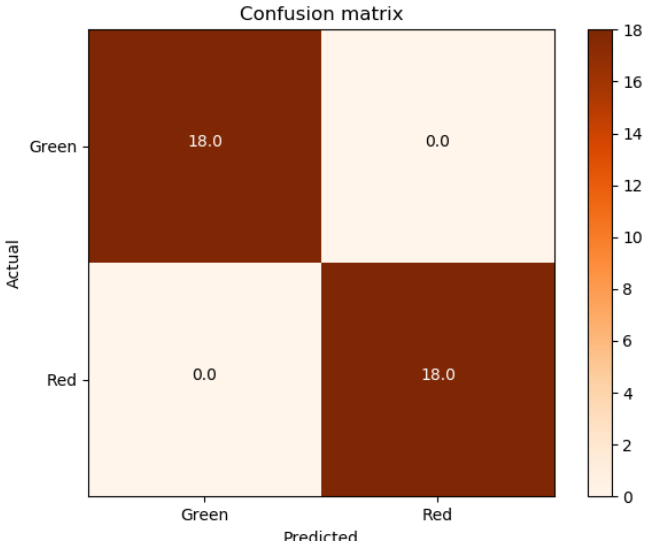

Figure 12: Confusion matrix of binary test set (decision tree).

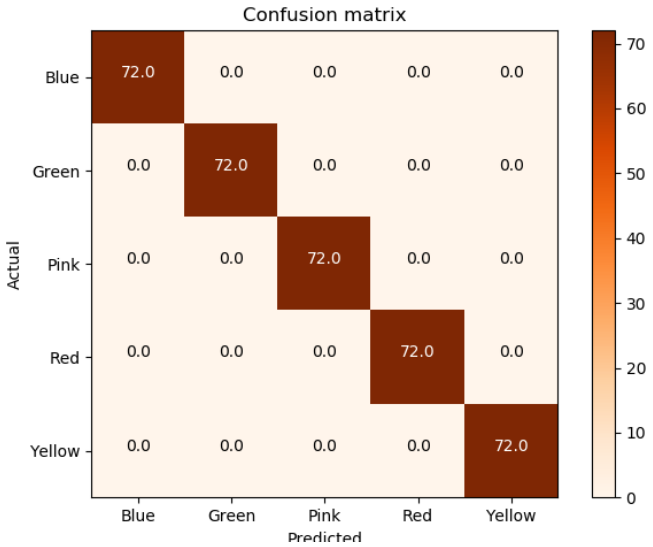

Figure 13: Confusion matrix of multiclass train set (decision tree).

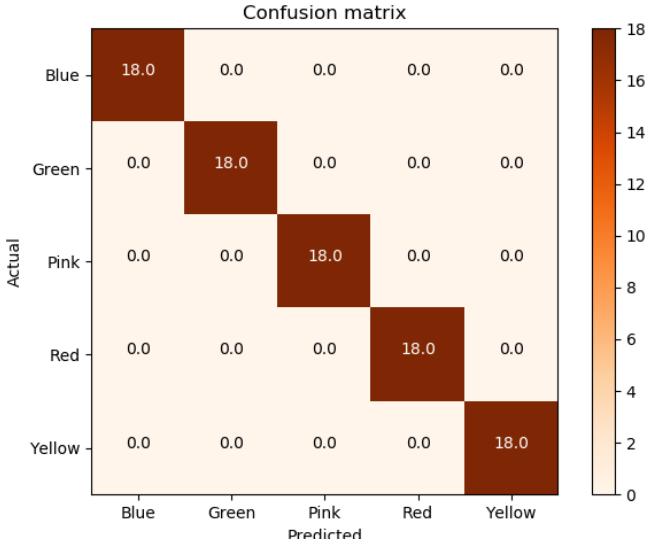

Figure 14: Confusion matrix of multiclass test set (decision tree).

The performance of decision tree model can be found in Table 2. The confusion matrices are shown in Figure 11, Figure 12, Figure 13, and Figure 14. As expected, the decision tree was able to classify all colour classes correctly. It is worth mentioning that the average cross validation score is not 1.0 like the other classifiers. The cause of this could be the size of the datasets. The performance may be inconsistent across different folds because the datasets are too small.

As shown in Figure 15, Decision tree is able to use only one feature to classify the colour classes. This confirms the theory from the preprocessing step which suggests that one feature might be sufficient for the binary dataset. However, Figure 16 shows that two features are needed to classify the colour classes. This indicates that the more classes we have, the more features are needed in the classification task.

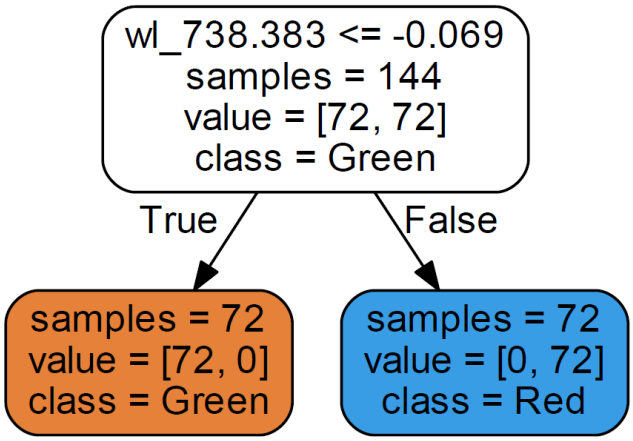

Figure 15: Visualisation of decision tree (binary).

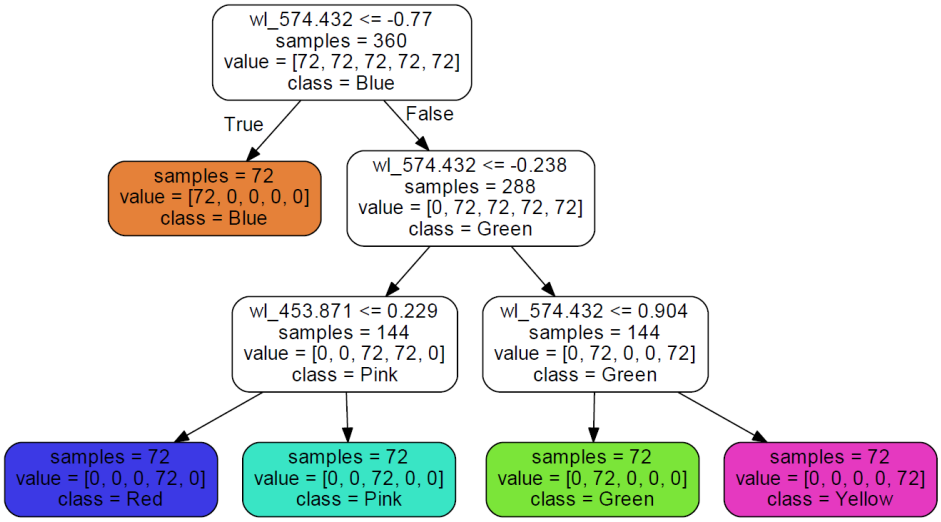

Figure 16: Visualisation of decision tree (multiclass).

4.4 Multi-layer Perceptron (MLP)

MLP is is a supervised learning algorithm that can learn non-linear models.

4.4.1 Neural Network Configuration

The network used is a multilayer feedforward neural network, with one hidden layer, and sigmoid activation function. The networks are trained using backpropagation. The structure of the networks used to train binary and multiclass dataset are shown in Figure 17 and Figure 18. Both networks have 4 hidden units, but the network used with binary dataset has 1 output neurons, while the one used with multiclass dataset has 5 output neurons.

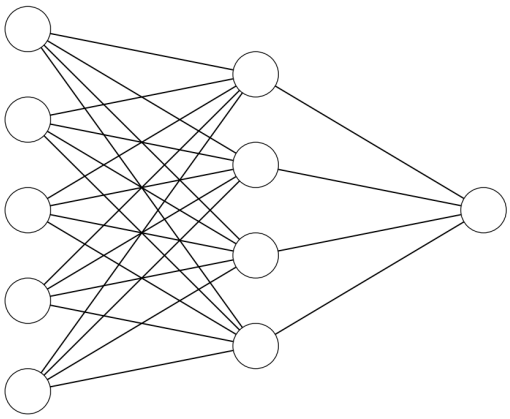

Figure 17: Neural network structure for binary dataset.

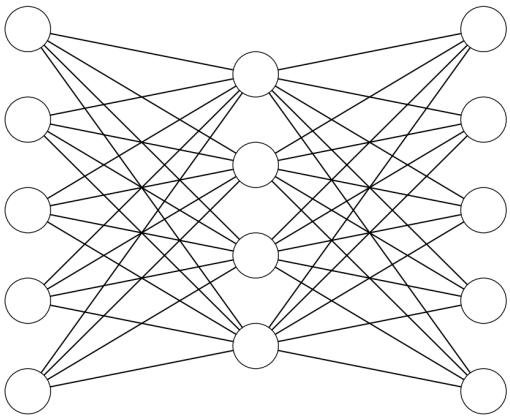

Figure 18: Neural network structure for multiclass dataset.

4.4.2 Performance

Dataset	Train CV score (avg)	Test accuracy score
Binary	1.0	1.0
Multiclass	1.0	1.0

Table 3: The performance of MLP model

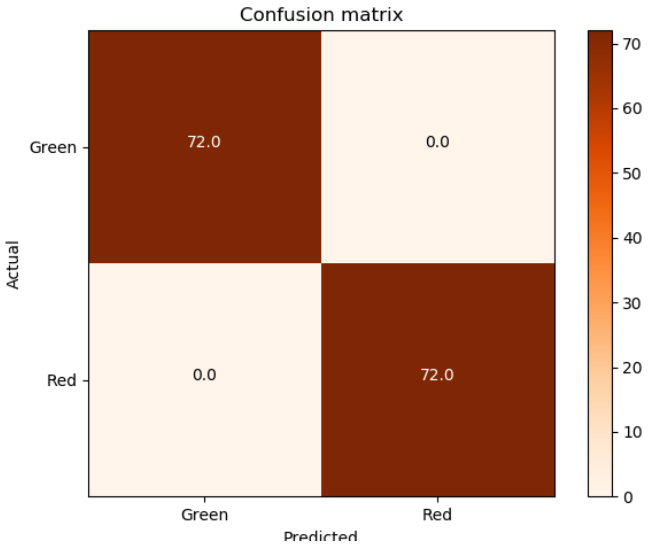

Figure 19: Confusion matrix of binary train set (mlp).

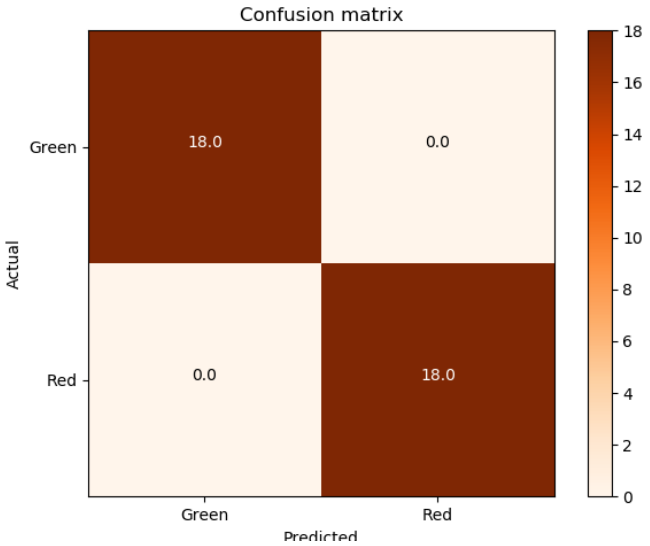

Figure 20: Confusion matrix of binary test set (mlp).

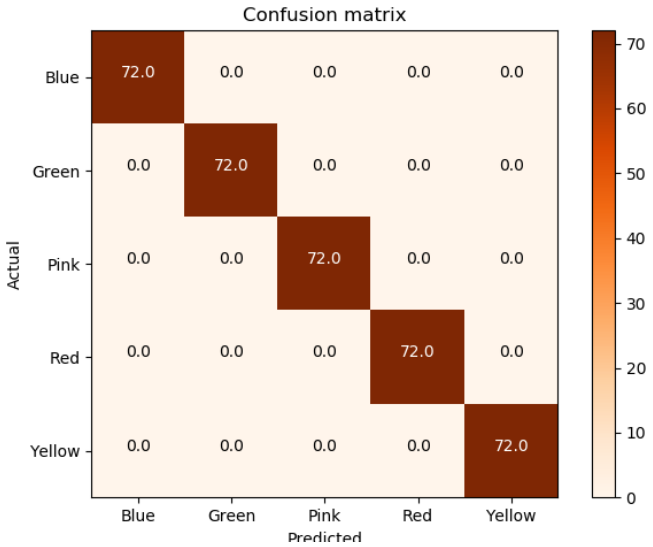

Figure 21: Confusion matrix of multiclass train set (mlp).

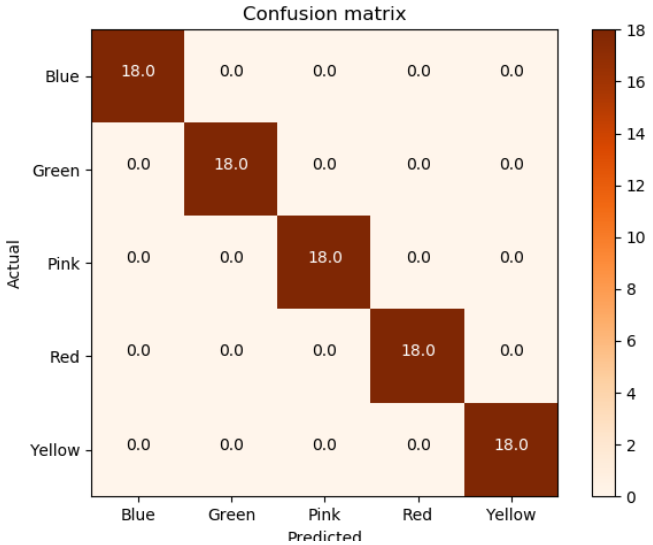

Figure 22: Confusion matrix of multiclass test set (mlp).

The performance of Multi-layer Perceptron model can be found in Table 3. The confusion matrices are shown in Figure 19, Figure 20, Figure 21, and Figure 22. The performance of MLP is not very different from the other algorithms. However, it is not guarantee to produce the same result everytime because MLP initialises with random weights.

4.5 Linear SVM Classification (SVC)

According to (Géron 2017), SVM is a very powerful and versatile machine learning algorithm. It is capable of performing linear or nonlinear classification and regression (Géron 2017).

Dataset	Train CV score (avg)	Test accuracy score
Binary	1.0	1.0
Multiclass	1.0	1.0

Table 4: The performance of SVC model

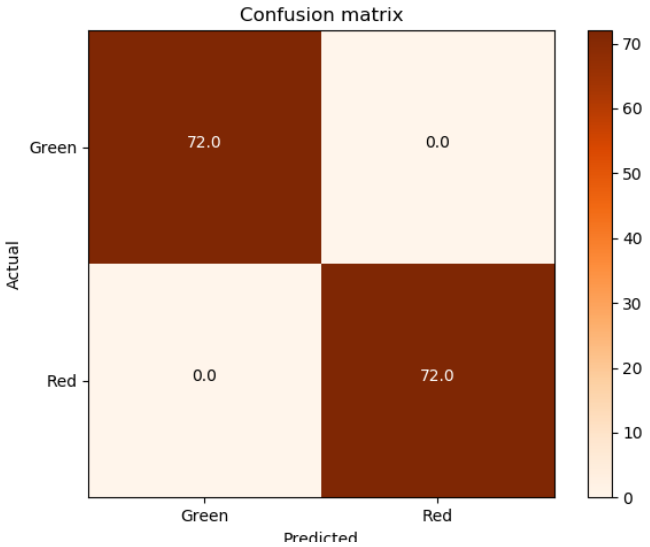

Figure 23: Confusion matrix of binary train set (svc).

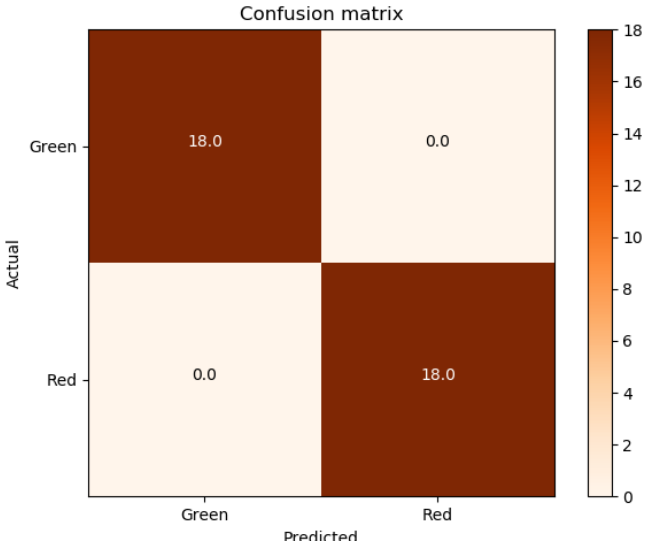

Figure 24: Confusion matrix of binary test set (svc).

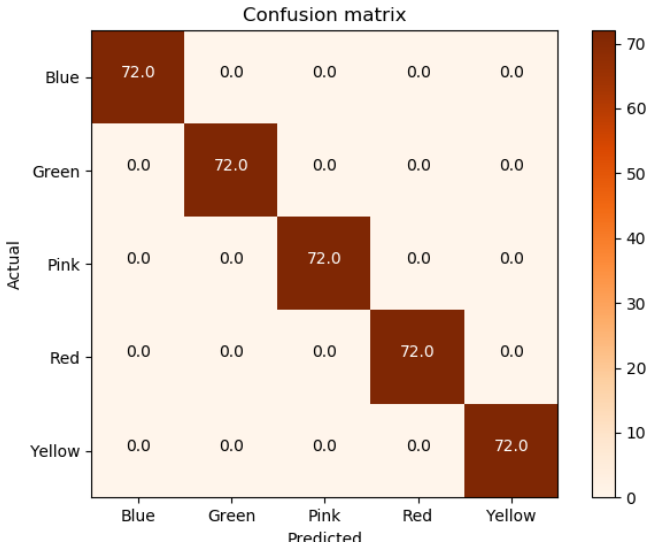

Figure 25: Confusion matrix of multiclass train set (svc).

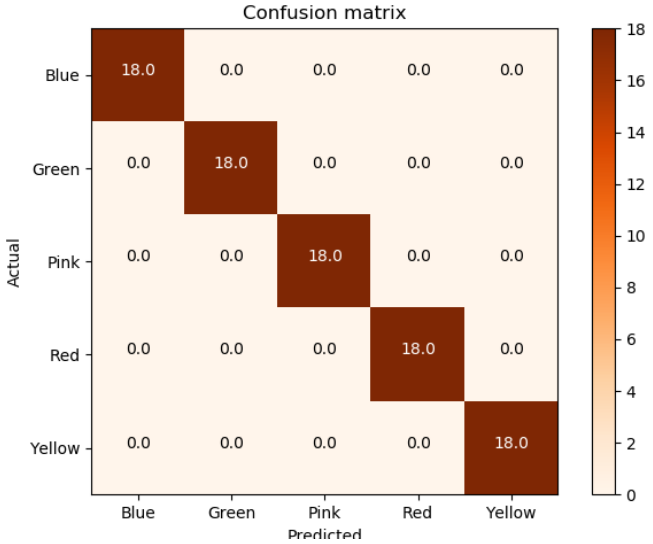

Figure 26: Confusion matrix of multiclass test set (svc).

The performance of SVC model can be found in Table 4. The confusion matrices are shown in Figure 23, Figure 24, Figure 25, and Figure 26. Like other models, SVC managed to classify all colour classes correctly. This is also expected since SVM models are effective in high dimensional spaces (Sklearn 2018*c*).

4.6 Final Model

For this practical, the final model is not chosen based on classification performance because the results are not very different. The final model is Linear SVM Classification. This is because SVMs can handle the situation where the decision boundaries come too close to the instances, which leads to worse performance on new instances (Géron 2017).

Logistic regression is not selected even though its cost function is convex (optimisation algorithm guarantees to find global minimum) (Géron 2017). It cannot deal with the situation where the decision boundaries come too close to the instances like SVM models.

Decision tree is not selected as final model even though it is easy to understand and visualise. It can generate over-complex trees that do not generalise the data well i.e. it is prone to overfitting (Sklearn 2018*a*).

MLP is not selected as final model because the random weights can lead to different validation accuracy (Sklearn 2018*b*). It also requires tuning a number of hyperparameters such as the number of hidden neurons, layers, and iterations (Sklearn 2018*b*).

5 Conclusion

Overall, this practical provides understanding of the procedures of classification task in machine learning. The objective of this practical is to gain experience in working with real experimental, imperfect, and limited data which has not been analysed before.

The majority of the tasks were done in preprocessing stage. The dataset provided is already cleaned, but there are more than 900 features in the datasets. RFE was used to recursively removes features and builds a model on those features that remain. Standardization was applied to the numerical attributes. The test set is also standardised using the mean/variance obtained from the training set.

The training process involves training four classification models on the dataset. The models are logistic regression, decision tree classifier, Multi-layer Perceptron, and SVC. SVC was chosen as final model because it can handle the situation where the decision boundaries come too close to the instances. Due to time constraints, it is not possible to explore other training parameters and training algorithms.

Bibliography

Brownlee, J. (2016), 'Feature selection for machine learning in python', https://machinelearningmastery.com/feature-selection-machine-learning-python. Accessed: 2018-04-19.

Géron, A. (2017), *Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems*, " O'Reilly Media, Inc.".

Sklearn (2018*a*), 'Decision trees', http://scikit-learn.org/stable/modules/tree.html. Accessed: 2018-04-19.

Sklearn (2018*b*), 'Neural network models (supervised)', http://scikit-learn.org/stable/modules/svm.html. Accessed: 2018-04-19.

Sklearn (2018*c*), 'Support vector machines', http://scikit-learn.org/stable/modules/neural_networks_supervised.html. Accessed: 2018-04-19.

Appendices

A Predicted Classes

A.1 Binary

Predictions
0
1
1
0
0
1
0
0
0
1
0
0
1
1
1
0
1
1
0
1
0

Table 5: Predicted classes of binary dataset

A.2 Multiclass

Predictions
0
2
0
2
0
0
0
2
0
4
1
4
3
3
2
0
4
2
4
3
3
4
1
2
1
4
2
3
2