

CS5014 Practical 2 Report

1 Introduction

The objective of this assignment is to come up with models using a part of the GQA dataset to tackle two classification tasks. Training the best model for classification involves the following steps, which will be introduced in the later sections: Data description, Preprocessing, Model selection, Training and validation, as well as Test Summary. In addition, the instruction of the program is attached in the appendix.

2 Data description

At the very first step, the precise insight of the dataset is essential since it can guide the strategy of other sections. The dataset contains the information of a set of objects that are targeted by a bounding box in the images, including object ID, position and size of the box, target attributes(colour and texture), and features for predicting. And the features represent colour histogram, oriented gradients histogram, and complex cells responses. Note that a certain value of a column can be considered as a value on a matrix.

For the targets of classification, the colour and texture of the objects have 12 and 9 classes respectively and they are not in the same column, i.e. there are 2 multiclass classification problems.

```
[ ] labels_color = data['color']
counter_color = Counter(labels_color)
for c,count in counter_color.items():
    per = count / len(labels_color) * 100
    print('Class=%s, Count=%d, Percentage=%.3f%%' % (c, count, per))
```

```
Class=white, Count=476, Percentage=35.655%
Class=brown, Count=218, Percentage=16.330%
Class=green, Count=297, Percentage=22.247%
Class=gray, Count=101, Percentage=7.566%
Class=blue, Count=87, Percentage=6.517%
Class=black, Count=73, Percentage=5.468%
Class=yellow, Count=20, Percentage=1.498%
Class=orange, Count=18, Percentage=1.348%
Class=red, Count=16, Percentage=1.199%
Class=pink, Count=17, Percentage=1.273%
Class=purple, Count=6, Percentage=0.449%
Class=beige, Count=6, Percentage=0.449%
```

```
▶ labels_texture = data['texture']
counter_texture = Counter(labels_texture)
for c,count in counter_texture.items():
    per = count / len(labels_texture) * 100
    print('Class=%s, Count=%d, Percentage=%.3f%%' % (c, count, per))
```

```
📄 Class=fluffy, Count=510, Percentage=38.202%
Class=grassy, Count=272, Percentage=20.375%
Class=blurry, Count=171, Percentage=12.809%
Class=smooth, Count=161, Percentage=12.060%
Class=crusty, Count=10, Percentage=0.749%
Class=fuzzy, Count=106, Percentage=7.940%
Class=gravel, Count=31, Percentage=2.322%
Class=rippled, Count=62, Percentage=4.644%
Class=coarse, Count=5, Percentage=0.375%
Class=barbed, Count=7, Percentage=0.524%
```

Figure 1. Statistical description of the targets

Statistics show an imbalance in the classes of the 2 objectives, with the proportion of minority and majority classes differing by even tens of times. Moreover, this also suggests the use of balanced accuracy when evaluating models, which reflects the results of predictions for minority classes

Overall, the dataset is unbalanced in terms of classes and contains a number of columns that are not useful for classification, while the number of features is relatively large, which is 440, but that of the rows is only 1344. Therefore, it is possible to reduce its dimensionality or perform feature selection, and balance the data in the preprocessing.

3 Preprocessing

3.1 cleaning data

The first step in pre-processing is to remove the irrelevant columns and deal with missing or outlier values. This assignment simply removes the columns for ID, and the position and size of the bounding box, and simply fills in the missing values with the median. In fact, almost all the data is lost in the rows that have missing values, but the missing values cannot be deleted directly since the practical's test requires all predicted values.

3.2 Standardisation



	lightness_0.0	lightness_0.1	lightness_0.2	lightness_1.0	lightness_1.1	lightness_1.2	lightness_2.0	lightness_2.1	lightness_2.2	redgreen_0.0	redgreen_0.1	redgreen_0.2	redgreen_1.0	redgreen_1.1	redgreen_1.2
count	1334.000	1334.000	1334.000	1334.000	1334.000	1334.000	1334.000	1334.000	1334.000	1334.000	1334.000	1334.000	1334.000	1334.000	1334.000
mean	137.553	137.254	132.095	140.414	139.123	131.778	137.729	137.642	131.040	137.553	137.254	132.095	140.414	139.123	131.778
std	50.168	51.806	51.421	50.425	55.590	52.533	50.630	52.425	52.397	50.168	51.806	51.421	50.425	55.590	52.533
min	0.297	0.010	2.643	2.474	1.661	2.751	6.379	1.516	1.148	0.297	0.010	2.643	2.474	1.661	2.751
25%	102.832	101.450	95.431	105.379	99.541	93.205	103.152	100.355	93.183	102.832	101.450	95.431	105.379	99.541	93.205
50%	137.411	136.664	132.443	141.348	138.052	132.405	137.983	138.494	131.876	137.411	136.664	132.443	141.348	138.052	132.405
75%	172.478	173.976	167.242	177.220	179.167	168.563	173.803	174.387	167.378	172.478	173.976	167.242	177.220	179.167	168.563
max	253.814	253.686	252.647	254.884	254.910	252.861	252.592	253.298	254.004	253.814	253.686	252.647	254.884	254.910	252.861

8 rows x 441 columns

Figure 2. Statistical description of the dataset

Standardisation can effectively improve the training speed and performance of a model. For example, standardised data may speed up the convergence of the neural network gradient and make it easier to converge to a global minimum. The 3 types of features in the dataset have inconsistent magnitudes and do not conform to a normal distribution, so they are standardised to be used as inputs to the model

3.3 Feature selection

If the model is powerful enough and the data have too many features, the model can

identify a single point by a particular set of features and build a special case, that is, overfitting.

For feature selection, dimensionality reduction by PCA is not necessary because the features are already derived from the matrix while the new features are not interpretable if PCA is implemented. Thus the project implements a method for selecting features with random forests (rf_selection) in order to make the process easily interpretable.

For each decision tree, the significant difference between the Gini coefficients before and after the nodes are divided by the feature X, the more significant the effect of X on the result, and the importance of each feature can be obtained by weighting each tree.

From observation, the significance of each feature is small, between 0.1% and 0.5%, and the project filters out features with a significance less than 0.003.

Despite this, feature selection still provides limited performance improvements to the model

3.4 Resampling

Resampling is a general method for dealing with imbalanced datasets. It is more common to process data with oversampling instead of undersampling, which removes the data as it would lose information that helps to fit. Nevertheless, there is a combination of oversampling and undersampling, which mostly performs better than the separate methods. the Imblearn library implements a method that combines oversampling method SMOTE and undersampling method Tomek, and is used by the program.(Lemaître et al., 2017)

However, it is evident from the later experiments that resampling does not necessarily improve the balanced accuracy of the model. The reason will be discussed later.

3.5 Dataset split

Sklearn's `train_test_split` easily splits the dataset into a training set and a validation set, and since the assignment already provides a test set. Considering the small size of the data set, the split ratio is set at 0.9 and a fixed random number seed is used for sampling.

It is essential to note that data splitting should be the first step in pre-processing, otherwise it can lead to data leakage as the test set would be used for resampling or standardisation.

4 Model selection

The neural network and the support vector machine each have pros and cons in such a task. The neural network can be adapted to multiple classification problems by increasing the number of output layer units, while the SVM usually uses the OVR (one vs. Rest) strategy to predict each class separately, so it is slower and take up more memory(Pedregosa et al., 2011). On the other hand, the SVM is better at fitting imbalanced data(by setting balanced class weight), but the neural network is not capable to weigh different classes.

A series of experiments have been done to investigate the performance of preprocessing and different models.

Experiment	Algorithm	Preprocessing	Mean	Balanced	Mean	Balanced
------------	-----------	---------------	------	----------	------	----------

			Acc(color)	Acc(texture)
1	SVM	Imputer, Standardisation	0.204	0.278
1	NN	Imputer, Standardisation	0.162	0.208
1	Logistic Regression	Imputer, Standardisation	0.149	0.198
2	SVM	Imputer, Standardisation, Feature selection	0.197	0.252
2	NN	Imputer, Standardisation, Feature selection	0.165	0.21
2	Logistic Regression	Imputer, Standardisation, Feature selection	0.17	0.292
3	SVM	Imputer, Standardisation,Resampling	0.163	0.204
3	NN	Imputer, Standardisation,Resampling	0.176	0.222
3	Logistic Regression	Imputer, Standardisation,Resampling	0.146	0.194
4	SVM	Imputer, Standardisation,Resampling, Feature selection	0.174	0.208
4	NN	Imputer, Standardisation,Resampling, Feature selection	0.172	0.245
4	Logistic Regression	Imputer, Standardisation,Resampling, Feature selection	0.171	0.266

Table 1. Model Evaluation experiments

The neural network uses 2 hidden layers with 100 units, and SVM uses a balanced class weight in the experiment. The score is generated by stratified KFold cross-validation.

The table showing algorithms that are optimal under different pre-processing are inconsistent. SVM can derive good predictions under the most basic conditions, feature selection optimizes logistic regression more significantly but also cut down the performance of SVM, while resampling only improves the performance of NN.

The reason that resampling does not optimise the model may be that oversampling makes the data too noisy. In detail, after resampling, the sample size of the training set may reach over 5,000, which is three times larger than the original, thus reducing the performance of SVM and logistic regression instead, which can balance the class weights.

Furthermore, feature selection filters out the less important features, improving the speed of training, even though it may sacrifice performance. Besides, the variance and upper limit of the balanced accuracy of NN is slightly higher than SVM in all experiments, i.e. neural networks may sometimes perform better.

In conclusion, the neural network might be chosen as the main algorithm and the SVM of Experiment 1 can be used as a benchmark, and if the neural network can exceed it after adjusting the parameters, then the neural network is finally used.

5 Training

5.1 Hyperparameter tuning

For neural networks, the relatively important hyperparameters are the learning rate, the regularisation, the hidden layer structure, the activation function, and the optimizer. The activation functions and optimisers use the default values, i.e. 'relu' and 'adam', as the relu function avoids the problem of vanishing gradients and the

stochastic gradient-based optimizer(adam) has good validation scores and convergence speed on large datasets compared to gradient descent. Other parameters can be indicated by grid search. The program implements Sklearn's GridSearchCV, which ranks different combinations of parameters by cross-validation.

Although both use cross-validation, data leakage happens in grid search if resampling is used since the validation set is also used to calculate virtual data. but model selection does not occur, which is a strange phenomenon. But even without resampling, the ranking is still valid.

	params	mean_test_score	std_test_score	rank_test_score
0	{'alpha': 0.001, 'hidden_layer_sizes': [100, 100], 'learning_rate_init': 0.001}	0.163946	0.021079	9
1	{'alpha': 0.001, 'hidden_layer_sizes': [100, 100], 'learning_rate_init': 0.01}	0.161214	0.019169	12
2	{'alpha': 0.001, 'hidden_layer_sizes': [100, 100], 'learning_rate_init': 0.005}	0.16193	0.01404	11
3	{'alpha': 0.001, 'hidden_layer_sizes': [100, 100, 100], 'learning_rate_init': 0.001}	0.180662	0.035115	2
4	{'alpha': 0.001, 'hidden_layer_sizes': [100, 100, 100], 'learning_rate_init': 0.01}	0.174424	0.028474	4
5	{'alpha': 0.001, 'hidden_layer_sizes': [100, 100, 100], 'learning_rate_init': 0.005}	0.168105	0.025148	7
6	{'alpha': 0.01, 'hidden_layer_sizes': [100, 100], 'learning_rate_init': 0.001}	0.174681	0.035658	3
7	{'alpha': 0.01, 'hidden_layer_sizes': [100, 100], 'learning_rate_init': 0.01}	0.166103	0.02243	8
8	{'alpha': 0.01, 'hidden_layer_sizes': [100, 100], 'learning_rate_init': 0.005}	0.162251	0.018659	10

9	{'alpha': 0.01, 'hidden_layer_sizes': [100, 100, 100], 'learning_rate_init': 0.001}	0.18321	0.033108	1
10	{'alpha': 0.01, 'hidden_layer_sizes': [100, 100, 100], 'learning_rate_init': 0.01}	0.170889	0.033047	5
11	{'alpha': 0.01, 'hidden_layer_sizes': [100, 100, 100], 'learning_rate_init': 0.005}	0.170264	0.028633	6

Table 2. Ranking of model with various params

The results show that a hidden layer structure with 3 layers of 100 cells, a learning rate of 0.001 and a regularization rate of 0.01 are relatively good parameters.

Note that the regularization formula is squared for the neural network, i.e. only the L2 penalty.

5.2 Validation

This step divides the dataset by 9:1 and validates it by learning curve and score. The learning curve reveals the generalisation ability of the model and also guides further tuning of the hyperparameters.

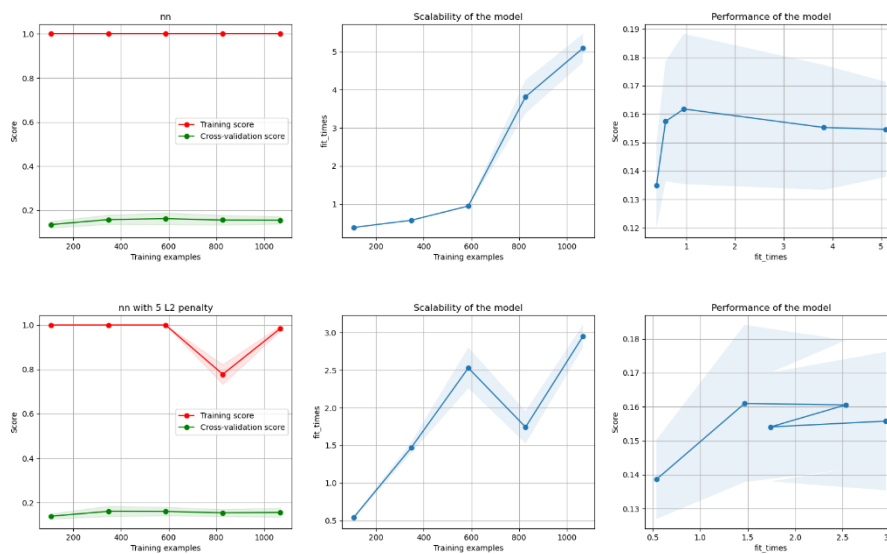


Figure 3. Learning curves of NN with $\alpha = 0.01$ and $\alpha = 5$

However, the curves revealed that the neural network has been overfitted. the issue does not be solved even stopping resampling and increasing the regularisation, so SVM was considered as the main algorithm.

The following graphs show the results of the same steps with SVM:

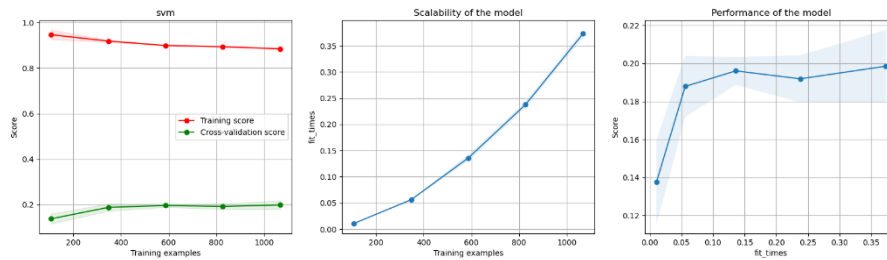


Figure 4. Learning curves of SVM with $C=1$ and $\gamma = \text{auto}$

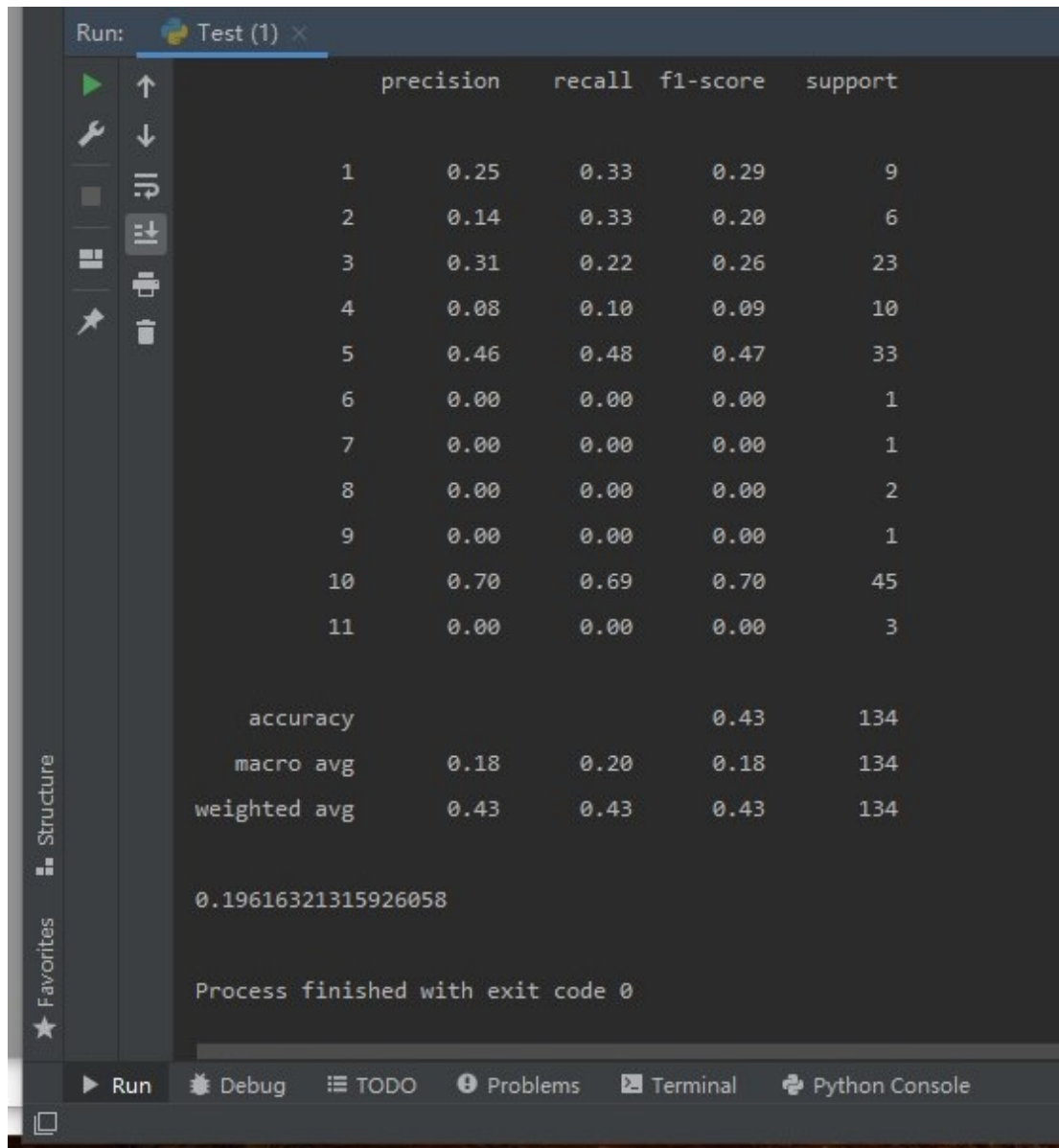


Figure 5. Classification report and the balanced accuracy of SVM

The best hyperparameters are $C = 1$, $\gamma = \text{auto}$, that is $1/\text{number of features}$, and feature selection as well as resampling are not applied.

Although the variance is still high, it is a better fit than neural networks. The balanced accuracy in the validation set is 0.196 and the result in the test set is 0.219, and this result is reproducible because all pre-processing steps and models are persisted

6 Advanced Task

This section further discusses the impact of feature selection and resampling on the

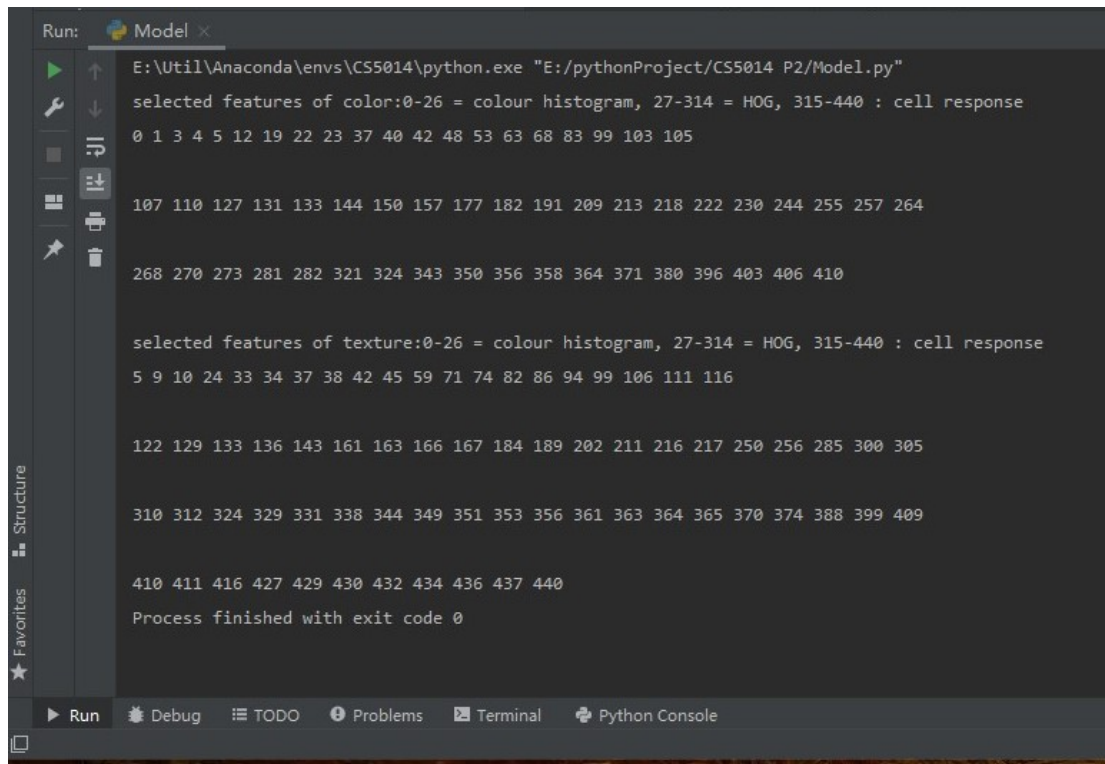
fitting, as well as the simple comparison of additional implementations of random forests versus the SVM.

6.1 Resampling misleads the model

SMOTE computes a number of virtual data points using the original data and $n_{k_neighbours}$. In this example, the resampling generates a very large number of data points because the smallest class in colour prediction has only 6 points, resulting in small $k_neighbours$. Furthermore, the data with that much noise will obviously reduce the classification results.

6.2 Feature importance for each task

As mentioned before, although feature selection filtered a large number of features, it did not solve the problem of overfitting. Another finding, however, was that the features selected for the two tasks had distinct intervals, i.e. the features did not have the same importance for each task.



```
Run: Model x
E:\Util\Anaconda\envs\CS5014\python.exe "E:/pythonProject/CS5014 P2/Model.py"
selected features of color:0-26 = colour histogram, 27-314 = HOG, 315-440 : cell response
0 1 3 4 5 12 19 22 23 37 40 42 48 53 63 68 83 99 103 105

107 110 127 131 133 144 150 157 177 182 191 209 213 218 222 230 244 255 257 264

268 270 273 281 282 321 324 343 350 356 358 364 371 380 396 403 406 410

selected features of texture:0-26 = colour histogram, 27-314 = HOG, 315-440 : cell response
5 9 10 24 33 34 37 38 42 45 59 71 74 82 86 94 99 106 111 116

122 129 133 136 143 161 163 166 167 184 189 202 211 216 217 250 256 285 300 305

310 312 324 329 331 338 344 349 351 353 356 361 363 364 365 370 374 388 399 409

410 411 416 427 429 430 432 434 436 437 440
Process finished with exit code 0
```

Figure 6. the index of features(columns) that selected by Random forest classifier

For colour classification, there are more features from colour histograms and HOGs, while for texture classification it is Cell responses and HOGs

6.3 Random Forest

Since both algorithms suffer from overfitting problems, the program additionally implements a random forest for comparison.

While SVM projects the features to higher dimensions via kernel functions and finds hyperplanes to complete the classification, RF constructs multiple decision trees for voting the classification by calculating the information entropy of the divided nodes after selecting at most $2/3$ of the feature subset.

RF is less prone to overfitting because it does not use all of the features when fitting the data. Besides, it is also able to use weighted class.(Pedregosa et al., 2011)

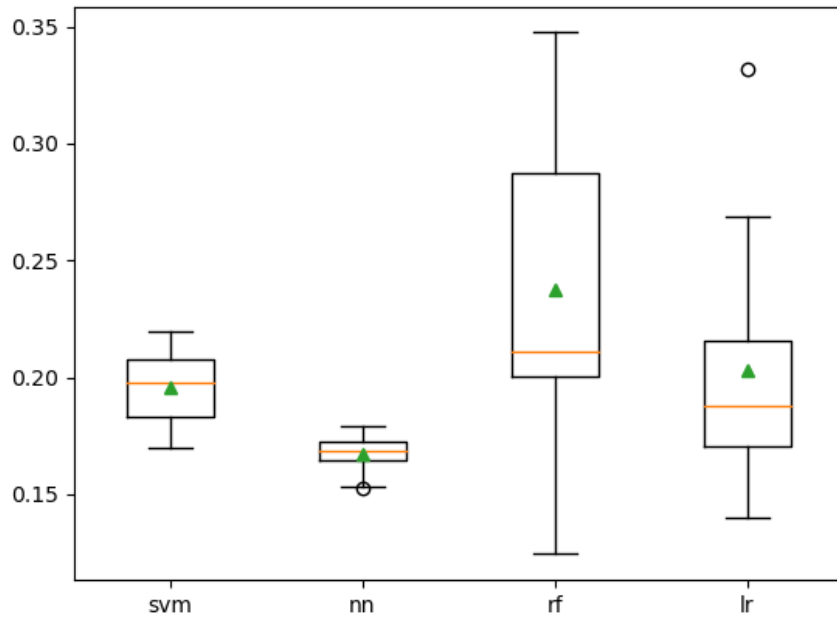


Figure 7. boxplot of mean balanced accuracy in cross-validation

Repeating the previous experiments, RF performed better than SVM with feature selection and was also the best in the test set, achieving a balanced accuracy of 24% in colour classification and 30% in material classification.

7 Conclusion

In summary, one of the insights gained from this assignment is that pre-processing should not only consider the characteristics of the dataset, but also the strengths and weaknesses of the algorithm. The insight about the algorithm is that for small sample datasets it is difficult to avoid overfitting even with a large penalty function, in which case random forests are a potentially good alternative.

8 Reference

- LEMAÎTRE, G., NOGUEIRA, F. & ARIDAS, C. K. 2017. Imbalanced-learn: a python toolbox to tackle the curse of imbalanced datasets in machine learning. *J. Mach. Learn. Res.*, 18, 559–563.
- PEDREGOSA, F., VAROQUAUX, G., GRAMFORT, A., MICHEL, V., THIRION, B., GRISEL, O., BLONDEL, M., PRETTENHOFER, P., WEISS, R., DUBOURG, V., VANDERPLAS, J., PASSOS, A., COUNAPEAU, D., BRUCHER, M., PERROT, M. & DUCHESNAY, É. 2011. Scikit-learn:

Machine Learning in Python. *J. Mach. Learn. Res.*, 12, 2825–2830.

Appendix

Due to the shortage of time and the number of functions in the program, I was not able to write the different functions as command lines.

To test functions in the program, the code under Main.py can be commented out or uncommented according to the instructions in the program.

The program relies on external libraries such as imblearn and joblib, which can be installed via pip:

```
pip install imblearn
```

```
pip install joblib
```