CS5014 P1: Dry Bean Classification

220029176

University of St Andrews

February 2022

CS5014 P1: Dry Bean Classification ___ *1*

Table of Figures ___ **3**

Table of Equations ___ **3**

List of Tables ___ **3**

Part 1: Data Processing ___ **4**

A How did you load and clean the data, and why? ___ 4

B How did you split the data into test/train set and why? ___ 4

C How did you process the data including encoding, conversion, and scaling, and why? ___ 4

D How did you ensure that there is no data leakage? ___ 5

Part 2: Training ___ **6**

A Train using penalty='none' and class weight=None. What is the best and worst classification accuracy you can expect and why? ___ 6

B Explain each of the following parameters used by LogisticRegression in your own words: penalty, tol, max iter. ___ 6

C Train using balanced class weights (setting class weight='balanced'). What does this do and why is it useful? ___ 6

D LogisticRegression provides three functions to obtain classification results. Explain the relationship between the vectors returned by predict(), decision function(), and predict proba(). ___ 7

Part 3: Evaluation ___ **8**

A What is the classification accuracy of your model and how is it calculated? Give the formula. ___ 8

B What is the balanced accuracy of your model and how is it calculated? Give the formula. ___ 9

C Show the confusion matrix for your classifier for both unbalanced (2a) and balanced (2c) cases. Discuss any differences. ___ 9

D Show the precision and recall of your algorithm for each class, as well as the micro and macro averages. Explain the difference between the micro and macro averages. ___ 10

Part 4: Advanced Tasks ___ **11**

A Set the penalty parameter in LogisticRegression to 'l2'. Give the equation of the cost function used by LogisticRegression and the gradient of this l2-regularised cost and explain what this penalty parameter does. ___ 11

B Implement a 2nd degree polynomial expansion on the dataset. Explain how many dimensions this produces and why. ___ 12

C Compare the results of regularised and unregularised classifiers on the expanded data and explain any differences. ___ 13

D Extend your solution to provide Precision-Recall plots for each class of dry beans. You may need to independently explore advanced scikit-learn functionality for this. ___ 14

References ___ **18**

Appendices ___ **19**

Table of Figures

Figure 1: Confusion Matrix for Unbalanced 10
Figure 2: Confusion Matrix for Balanced 10
Figure 3: Barbunya Precision-Recall Plot 14
Figure 4: Bombay Precision-Recall Plot 15
Figure 5: Cali Precision-Recall Plot 15
Figure 6: Dermason Precision-Recall Plot 16
Figure 7: Horoz Precision-Recall Plot 16
Figure 8: Seker Precision-Recall Plot 17
Figure 9: Sira Precision-Recall Plot 17

Table of Equations

Equation 1: The Chi Square Algorithm 5
Equation 2: Z-Score Equation 5
Equation 3: Balanced Class Weights Equation 7
Equation 4: Classification Accuracy Equation 8
Equation 5: Classification Accuracy Equation for Multi-Class models 8
Equation 6: Balanced Accuracy Equation 9
Equation 7: Cost Function for L2 Logistic Regression 12
Equation 8: L2-Regularised Gradient Equation 12
Equation 9: 2nd Degree Polynomial Expansion Equation 13

List of Tables

Table 1: The percentage of the population of each bean class within the bean dataset 7
Table 2: The Classification Report for the Given Model 11
Table 3: A summarisation of the Evaluation 13

Part 1: Data Processing

A How did you load and clean the data, and why?

The data set was loaded into the Python script using the Pandas module functionality. The data set was configured in comma-separated value (CSV) format. The dataset was read into a DataFrame to be used for further pre-processing.

To clean the data, all rows were checked for null, or invalid (such as irregular negatives) data and any rows found were removed, there were no null or invalid data present within this data set. Duplicate rows in the data set were also removed, of which there were 68 duplicate rows. These rows were removed to reduce the risk of equivalent data being split into both training and testing data sets, leading to data leakage.

B How did you split the data into test/train set and why?

The dataset was split into three subsets, namely training, validation, and testing. This was achieved by employing the "train_test_split" method from the widely-used "sklearn" library. Initially, the data was partitioned into an 80:10:10 ratio for the training, validation, and testing subsets, respectively, which is a common practice in real-world scenarios, providing ample data for both training and evaluation purposes (Baheti, 2023). The training set was used to train the model, the validation set was used to fine tune the model and the testing set was used for final evaluation.

To explore the impact of this ratio on the model's performance, four variations were tested, with split ratios of 90%, 85%, 75%, and 70%, respectively, and the training and validation accuracies were compared, illustrated in Appendix 1. After careful evaluation, the initial 80:10:10 split ratio was determined to be optimal, exhibiting high accuracy while mitigating overfitting risks. To ensure reproducibility when testing and evaluating the model, a random state of "1" was specified. Furthermore, the data was stratified based on the "Class" column when partitioning into the three subsets, ensuring that each subset contained a similar proportion of samples for each target class, given that the dataset was imbalanced (as illustrated in Appendix 2).

C How did you process the data including encoding, conversion, and scaling, and why?

The logistic regression algorithm requires numerical or float inputs. Given that all inputs were already integers or floats, no further encoding was necessary. To increase the efficiency and tractability of the learning process by reducing redundancy, feature selection was conducted on the input set. Three distinct feature selection algorithms were implemented and evaluated based on the time taken, as well as the training and validation accuracy (refer to Appendix 3). After meticulous analysis, Peterson's Chi Square algorithm was chosen as the optimal algorithm based on its performance.

Peterson's Chi Square algorithm (*see Equation 1*), a well-known feature selection algorithm, is commonly used to identify and remove redundant features from a given dataset. In this algorithm, a statistical test called the Chi-Square test is employed to measure the independence between a feature and a target variable. The Chi-Square test is conducted by comparing the observed frequency of each feature with the expected frequency based on the target variable's distribution. Features that exhibit a high degree of independence from the target variable are likely to be irrelevant and can be safely removed without any significant impact on model performance. On the

other hand, features that display a low degree of independence from the target variable are deemed crucial and are retained in the dataset. By removing redundant features and retaining only the most relevant ones, the learning process can be made more tractable and efficient. The number of features were determined by analysing the relationship between the accuracy of the training and validation sets against time taken for the model to converge, see Appendix 4.

$$X_C^2 = \sum \frac{(O_i - E_i)^2}{E_i}$$

Equation 1: The Chi Square Algorithm

Where:

C = Degree of Freedom

O = Observed Value(s)

E = Expected Value(s)

The suitability of normalisation versus standardisation for scaling was investigated by scrutinising outliers. To accomplish this, the Z-score of all input features was computed to identify outliers (*see Equation 2*). The Z-score is defined as the number of standard deviations that a given value deviates from the mean. Outliers were identified as values that exceeded 3 standard deviations from the mean, as stated by Mahmood (2022). Using this Z-score criterion, 1304 outliers were detected and eliminated, which accounted for 9.8% of the dataset. However, this also resulted in the exclusion of all Bombay bean samples. As a result, normalisation was deemed unsuitable, and instead, standardisation was used by employing the StandardScaler() method. This method uses the Z-Score equation (see Equation 2) to map each feature's value to the number of standard deviations away from its mean. This ensures there are not any dominating features in the logistical regression which can lead to bias, and helps to improve performance and stability of the model.

$$Z = \frac{x - \mu}{\sigma}$$

Equation 2: Z-Score Equation

Where:

μ = Mean

σ = Standard Deviation

D How did you ensure that there is no data leakage?

To prevent data leakage, all data-dependent operations were carried out after splitting the dataset into distinct training, validation, and testing sets. Additionally, the elimination of duplicate rows guarantees that corresponding data will not inadvertently find its way into both the training and testing sets. This approach ensures that the data contained in the training set has no foreknowledge of the information in the test set. Furthermore, the standardisation and feature selection processes were implemented independently on each set to avoid any potential influence from the validation or test sets. By adopting these measures, we have maintained the integrity and independence of the data sets, which is essential for producing accurate and reliable results.

Part 2: Training

A Train using penalty='none' and class weight=None. What is the best and worst classification accuracy you can expect and why?

The pre-processed data set was used for the fitting of the classification model. The model was trained using logistical regression with penalty='none' and class weight=None as the parameters. The classification accuracy against the training set is 92.4%.

To establish a lower baseline for evaluating the performance of a classification model, a dummy classifier with the uniformly random strategy was employed. This strategy, which assigns equal probabilities to all classes irrespective of their frequency or distribution in the dataset, results in the model predicting each class with the same frequency, namely 26.2%. This model is considered to be the worst possible classification model when dealing with imbalanced or disproportionate data as seen within this dataset, see Appendix 2.

The classifier was trained without any regularisation penalty, which may have led to over-fitting of the model to the training data. In such cases, the model becomes too complex, and may fail to capture general trends and patterns in the data, leading to poor performance on unseen data. Over-fitting can be so severe that the model achieves a training accuracy of 100%, which is not ideal as it provides a false sense of performance. Therefore, when evaluating the model on the training data, the expected classification accuracies can range from the worst-case scenario of 26.2% (corresponding to the dummy classifier with a uniform strategy), to the best possible accuracy of 100% (obtained when the model is severely overfit to the training data).

B Explain each of the following parameters used by LogisticRegression in your own words: penalty, tol, max iter.

The '**penalty**' parameter of a regularised classifier determines the type of norm used in penalising the loss function. By adding a penalty term to the loss function, the coefficients of the regression are effectively shrunk towards zero, which limits the flexibility of the model and helps prevent over-fitting. This regularisation can improve the model's ability to generalise to new, unseen datasets.

The '**tol**' parameter specifies the tolerance level for the stopping criteria of the model. If the algorithm has converged within the tolerance range for an iteration, it is considered to have reached the stopping criteria.

The '**max_iter**' parameter specifies the maximum number of iterations that the logistic regression algorithm can perform before stopping. This parameter can help prevent the algorithm from running indefinitely and ensures that the algorithm stops after a fixed number of iterations, even if the stopping criteria has not been met.

C Train using balanced class weights (setting class weight='balanced'). What does this do and why is it useful?

The pre-processed data set was used for the fitting of the classification model. The model was trained using logistical regression with penalty='none' and class weight='balanced' as the parameters. The classification accuracy against the training set is 92.2%.

Table 1 highlights a class distribution imbalance in the bean dataset, which can introduce biases into the learning algorithm and negatively impact its performance.

Bean Class	Percentage of Population (%)
Dermason	26
Sira	19.3
Seker	14.9
Horoz	14.2
Cali	12
Barbunya	9.75
Bombay	3.85

Table 1: The percentage of the population of each bean class within the bean dataset

To tackle this issue, a class weight parameter can be used in logistic regression to adjust for the imbalanced classes. This involves assigning weights to both the minority and majority output classes during training, thus reducing the bias. By assigning a higher weight to the minority class, misclassifications are penalised more severely, while the weight for the majority class is reduced to decrease its impact. To balance the class weights, an Equation 3 is used to assign the weights inversely proportional to their frequencies.

$$weight_{class} = \frac{no.samples}{no.classes \times no.occurrances(class)}$$

Equation 3: Balanced Class Weights Equation

Where:

No.samples = Number of samples

No.classes = Number of classes

No.occurrances(class) = Number of occurrences of class

D LogisticRegression provides three functions to obtain classification results. Explain the relationship between the vectors returned by predict(), decision function(), and predict proba().

LogisticRegression is a classification algorithm that provides three functions to obtain classification results.

Decision function(), predicts the confidence score for each sample, which is proportional to the signed distance from the sample to the decision boundary. This decision boundary is a hyperplane that separates the feature space into different regions, one for each class.

Predict proba(), provides the probability estimates for each sample belonging to each output class. In this case, since there are 7 output classes, the probabilities of all 7 classes are estimated and normalised to one for each sample.

Predict(), this method returns the predicted class labels for the input samples. It gives a vector of class labels, where each element relates to the predicted class for the corresponding input sample.

In conclusion of the three functions, the decision function() returns the confidence score that is then utilised by predict proba() to generate probability estimates, which are in turn used by predict() to derive the predicted output class for each sample.

Part 3: Evaluation

A What is the classification accuracy of your model and how is it calculated? Give the formula.

The testing Set Classification Accuracy with (penalty='none', class weight='none') = 93.3%

The testing Set Classification Accuracy with (penalty='none', class weight='balanced') = 92.6%

The accuracy of classification was computed by aggregating the correctly predicted classifications and normalising them by the total sum of all the true positives and negatives. This includes both the correctly and incorrectly predicted classifications, as shown in Equation 4.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Equation 44: Classification Accuracy Equation

Where:

TP = True Positive Count

TN = True Negative Count

FP = False Positive Count

FN = False Negative Count

Another way to calculate the classification accuracy is by dividing the sum of the indicator function evaluations over all samples by the total number of samples. The indicator function evaluates to 1 if the predicted output, $\hat{y}_i$, matches the true output, y_i. If not, 0 is outputted. Please see Equation 5 below for a formal expression of this approach.

$$accuracy(y, \hat{y}) = \frac{1}{n}\sum_{i=0}^{n-1} 1(\hat{y}_i = y_i)$$

Equation 55: Classification Accuracy Equation for Multi-Class models

Where:

y = The true output set

$\hat{y}$ = The predicted output set

n = The total number of samples

$\hat{y}_i$ = The predicted output of sample i

y_i = The true output of sample i

B What is the balanced accuracy of your model and how is it calculated? Give the formula.

The testing Set Classification Balanced Accuracy with (penalty='none', class weight='none') = 94.2%

The testing Set Classification Balanced Accuracy with (penalty='none', class weight='balanced') = 94%

The balanced classification accuracy is calculated by adding the precision and recall values and then dividing the sum by 2 (*see Equation 6*).

$$Balanced\ Accuracy = \frac{1}{2}\left(\frac{TP}{P} + \frac{TN}{N}\right)$$

Equation 66: Balanced Accuracy Equation

Where:

$\frac{TP}{P}$ *= The Precision of the model*

$\frac{TN}{N}$ *= The Recall of the model*

TP = True Positive Count

TN = True Negative Count

P = Total Positive Count

N = Total Negative Count

C Show the confusion matrix for your classifier for both unbalanced (2a) and balanced (2c) cases. Discuss any differences.

The confusion matrices for the unbalanced and balanced cases are shown in Figure 1 and Figure 2 respectively. Looking at the two confusion matrices, the precision over the smaller populated bean class such Barbunya and Horoz is higher in the balanced matrix than the unbalanced matrix. However, the precision in the higher populated bean classes such as Dermason and Sira went down. This is because the accuracy is more evenly distributed across all classes within a balanced model, and the precision and recall values are also more balanced. This is due to the balanced case considering the class imbalance and adjusting the weights accordingly to ensure that all classes are given equal consideration during the classification process.

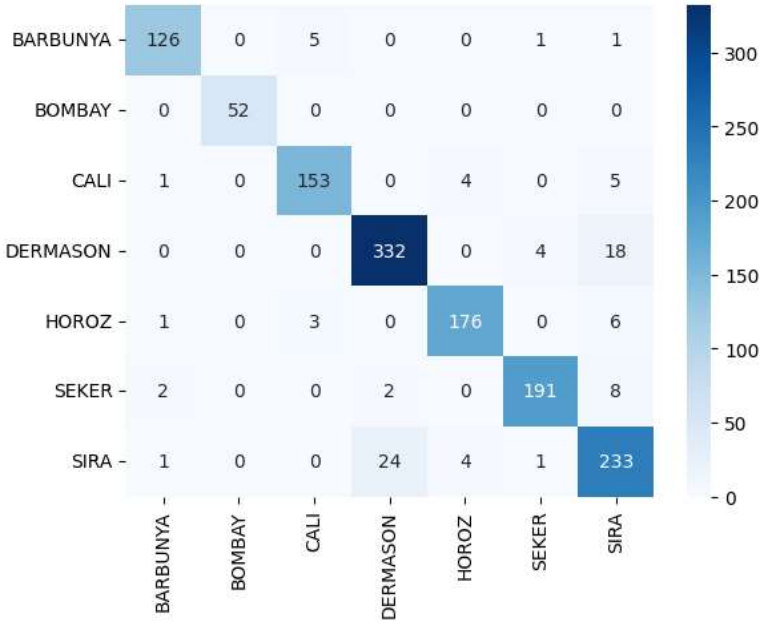


Figure 1: Confusion Matrix for Unbalanced

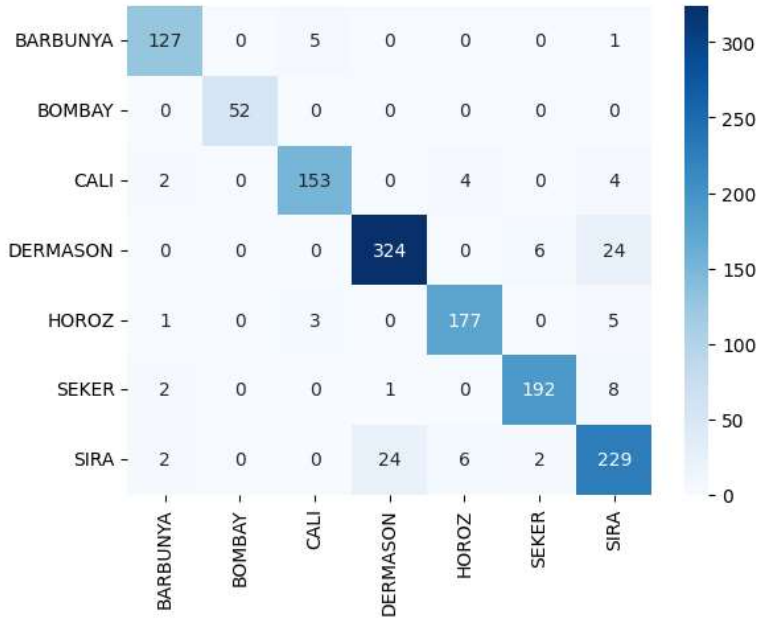


Figure 2: Confusion Matrix for Balanced

D Show the precision and recall of your algorithm for each class, as well as the micro and macro averages. Explain the difference between the micro and macro averages.

The testing Set Classification Balanced Accuracy with (penalty='none', class weight='balanced') was used for this evaluation of the precision recall, micro and macro averages, see Table 2.

Bean Class	Precision	Recall
BARBUNYA	95%	95%
BOMBAY	100%	100%
CALI	95%	94%
DERMASON	93%	92%
HOROZ	95%	95%
SEKER	96%	95%
SIRA	85%	87%

Table 2: The Classification Report for the Given Model

The macro average is the average of the precision and recall values for each class. It treats all classes equally and gives equal weight to each class, regardless of the number of samples in each.

Calculation based on Table 2:

Macro Precision Average: (0.95 + 1.00 + 0.95 + 0.93 + 0.95 + 0.96 + 0.85) / 7 = 0.94

Macro Recall Average: (0.95 + 1.00 + 0.94 + 0.92 + 0.95 + 0.95 + 0.87) / 7 = 0.94

The micro-average precision and recall score is obtained by computing the true positives, true negatives, false positives, and false negatives of the model across all classes, and using these values to calculate the precision and recall metrics.

Calculation based of Figure 2:

Micro Precision Average: (127 + 52 + 153 + 324 + 177 + 192 + 229) / (127 + 52 + 5 + 2 + 4 + 1 + 52 + 153 + 324 + 6 + 24 + 3 + 177 + 192 + 8 + 2 + 24 + 6 + 2 + 229) = 0.93

Micro Recall Average: (127 + 52 + 153 + 324 + 177 + 192 + 229) / (127 + 52 + 5 + 2 + 4 + 1 + 127 + 52 + 153 + 324 + 6 + 24 + 1 + 177 + 190 + 10 + 1 + 23 + 6 + 2 + 231) = 0.93

From these results, we can see that the macro average and micro average are very similar, which indicates that there is no significant class imbalance in the dataset.

Part 4: Advanced Tasks

A Set the penalty parameter in LogisticRegression to 'l2'. Give the equation of the cost function used by LogisticRegression and the gradient of this l2-regularised cost and explain what this penalty parameter does.

The penalty parameter in logistic regression with L2 regularisation allows us to control the balance between model flexibility and generalisation performance. By tuning this hyperparameter, we can

find the best trade-off between bias and variance in the model, leading to better performance on unseen data. The equation for the cost function used by logistical regression, as seen in Equation 7, was derived from CS5014 Lecture 5 (Harris-Birtill & Terzić, 2023) and CS5014 Lecture 9 (Terzić & Harris-Birtill, 2023).

$$Cost(\theta; XY) = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i log g\left(x_i^T\theta\right) + (1 - y_i)\log\left(1 - g\left(x_i^T\theta\right)\right)\right] + \lambda\theta^T\theta$$

Equation 77: Cost Function for L2 Logistic Regression

Where:

$\lambda\theta^T\theta$ = The L2 penalty term, adds a penalty to the cost function for large values of the coefficients θ

$\sum_{i=1}^{N}\left[y_i log g\left(x_i^T\theta\right) + (1 - y_i)\log\left(1 - g\left(x_i^T\theta\right)\right)\right]$, *Logistic loss*

$g\left(x_i^T\theta\right) = \frac{1}{1+e^{-x^T\theta}}$, *Logistic Function*

N = Number of samples

The gradient equation of this L2-regularised cost is seen in Equation 8.

$$\frac{d}{d\theta_J} = \sum_{i=1}^{N}\left(g\left(x_i^T\theta\right) - y_i\right) + \lambda\theta^T\theta$$

Equation 88: L2-Regularised Gradient Equation

Where:

$\lambda\theta^T\theta$ = The L2 penalty term, adds a penalty to the cost function for large values of the coefficients θ

$g\left(x_i^T\theta\right) = \frac{1}{1+e^{-x^T\theta}}$, *Logistic Function*

B Implement a 2nd degree polynomial expansion on the dataset. Explain how many dimensions this produces and why.

Number of Features Before 2nd Degree Polynomial expansion: 11

Number of Features Resulting from a 2nd Degree Polynomial Expansion: 67

To implement a 2nd degree polynomial expansion on a dataset with 11 features, we would create new features by taking every possible combination of the original features up to degree 2, which includes squared terms of the original features and interaction terms between every pair. This process results in a total of 67 features, which is significantly higher than the original 11 features. The formula to calculate the total number of features resulting from a 2nd degree polynomial expansion can be seen in Equation 9 below.

$$No.Features = 1 + \frac{n + n^2}{2}$$

Equation 99: 2nd Degree Polynomial Expansion Equation

Where:

n = Original number of features

C Compare the results of regularised and unregularised classifiers on the expanded data and explain any differences.

The testing Set Classification Balanced Accuracy with (penalty='none', class weight='balanced') was used to represent the no regularised model. The testing Set Classification Balanced Accuracy with (penalty="l2", class weight='balanced') was used to represent the regularised model in Table 3.

Accuracy Measurements	No Regularised	Regularised
Classification Accuracy	92.6%	92.5%
Balanced Accuracy	94%	93.7%
Confusion Matrix	[[127 0 5 0 0 0 1] [0 52 0 0 0 0 0] [2 0 153 0 4 0 4] [0 0 0 324 0 6 24] [1 0 3 0 177 0 5] [2 0 0 1 0 192 8] [2 0 0 24 6 2 229]]	[[125 0 5 0 1 1 1] [0 52 0 0 0 0 0] [2 0 154 0 4 0 3] [0 0 0 326 0 5 23] [1 0 4 0 175 0 6] [2 0 0 1 0 190 10] [1 0 0 23 6 2 231]]

Table 3: A summarisation of the Evaluation

The regularised and unregularised classifiers have similar classification accuracy, with the unregularised classifier having a slightly higher accuracy of 92.6% compared to the regularised classifier's accuracy of 92.5%. Additionally, when looking at the balanced accuracy, which considers the class imbalances in the dataset, the unregularised classifier has a higher score of 94% compared to the regularised classifier's score of 93.7%. This suggests that the unregularised classifier is better at predicting the minority classes in the dataset.

Looking at the confusion matrix, we can see that there are some differences in the number of misclassifications between the regularised and unregularised classifiers. For example, the regularised classifier has one additional misclassification in the first class compared to the unregularised classifier. However, these differences are relatively small and may not be statistically significant.

Overall, it appears that the regularisation had a slight detrimental impact on the classification performance of the model, and therefore unregularised is preferred in this instance.

D Extend your solution to provide Precision-Recall plots for each class of dry beans. You may need to independently explore advanced scikit-learn functionality for this.

The testing Set Classification Balanced Accuracy with (penalty='none', class weight='balanced') was used for this question.

To create Precision-Recall plots for each class of dry bean, I converted the classes into integers using the label_binarize function from the sklearn pre-processing module. This function performed hot key encoding by splitting the column into seven unique columns, one for each bean type. Subsequently, the precision-recall curves were plotted using the precision_recall_curve function from the sklearn metrics module. Please see Figures 3 to 9 below.

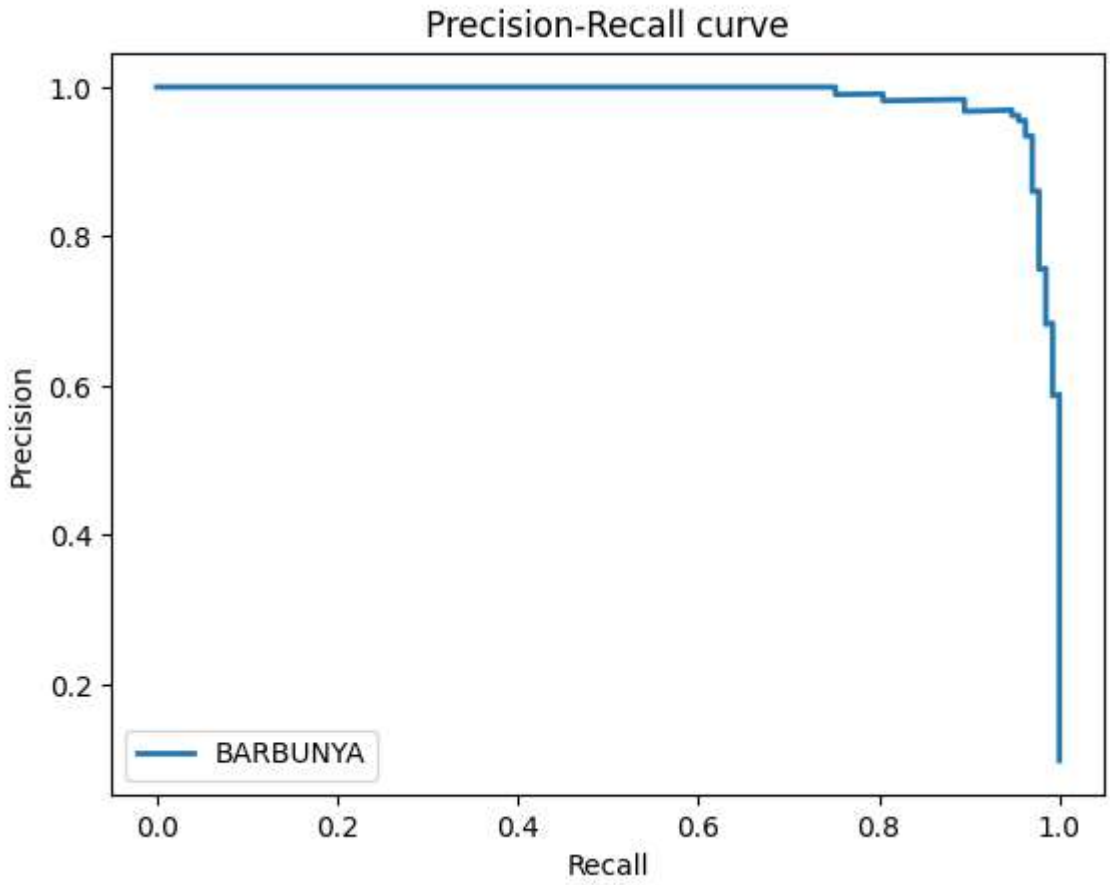


Figure 3: Barbunya Precision-Recall Plot

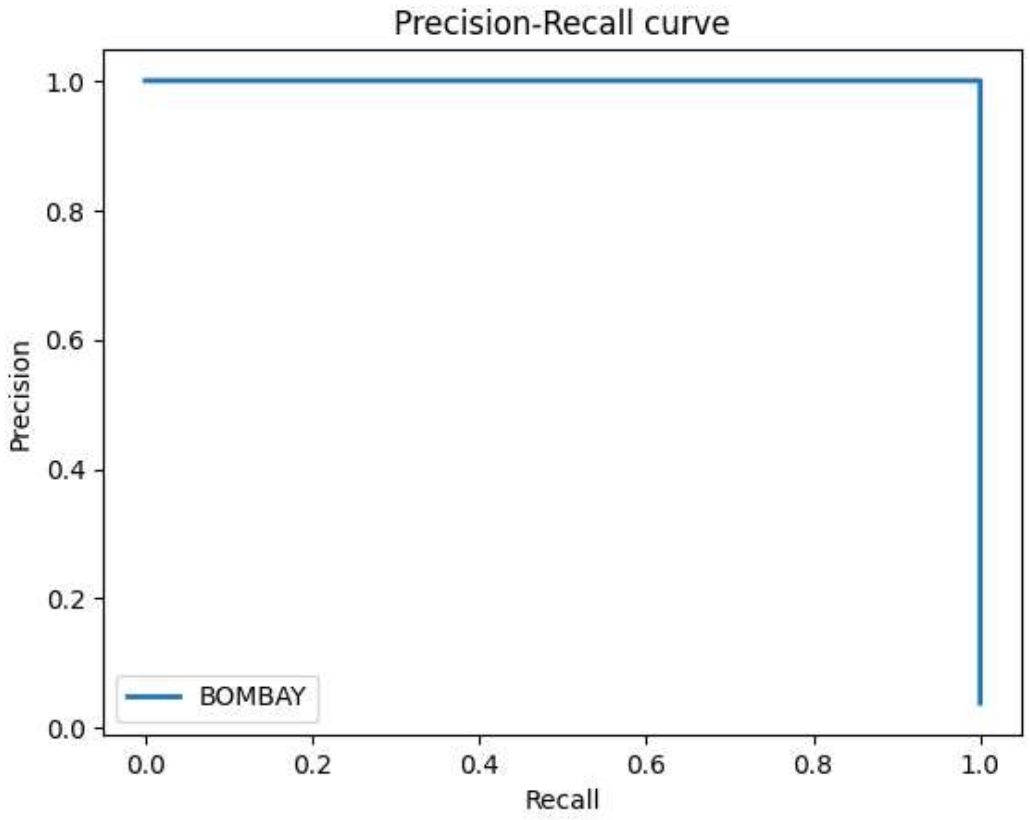


Figure 4: Bombay Precision-Recall Plot

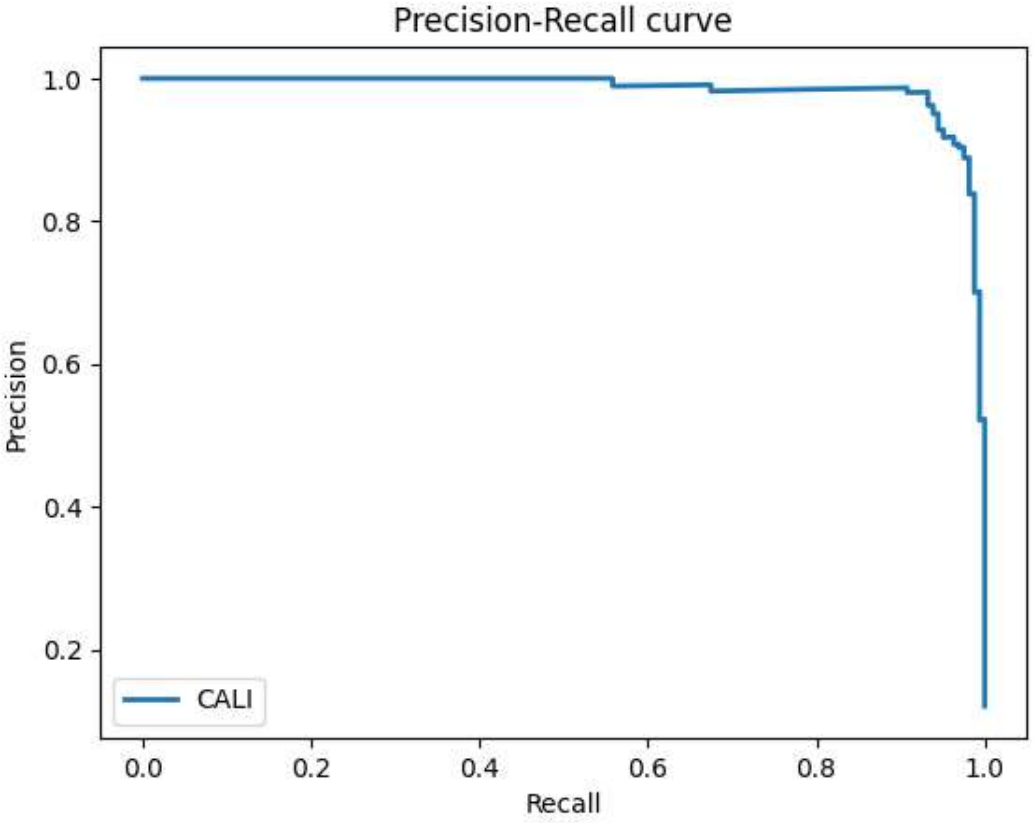


Figure 5: Cali Precision-Recall Plot

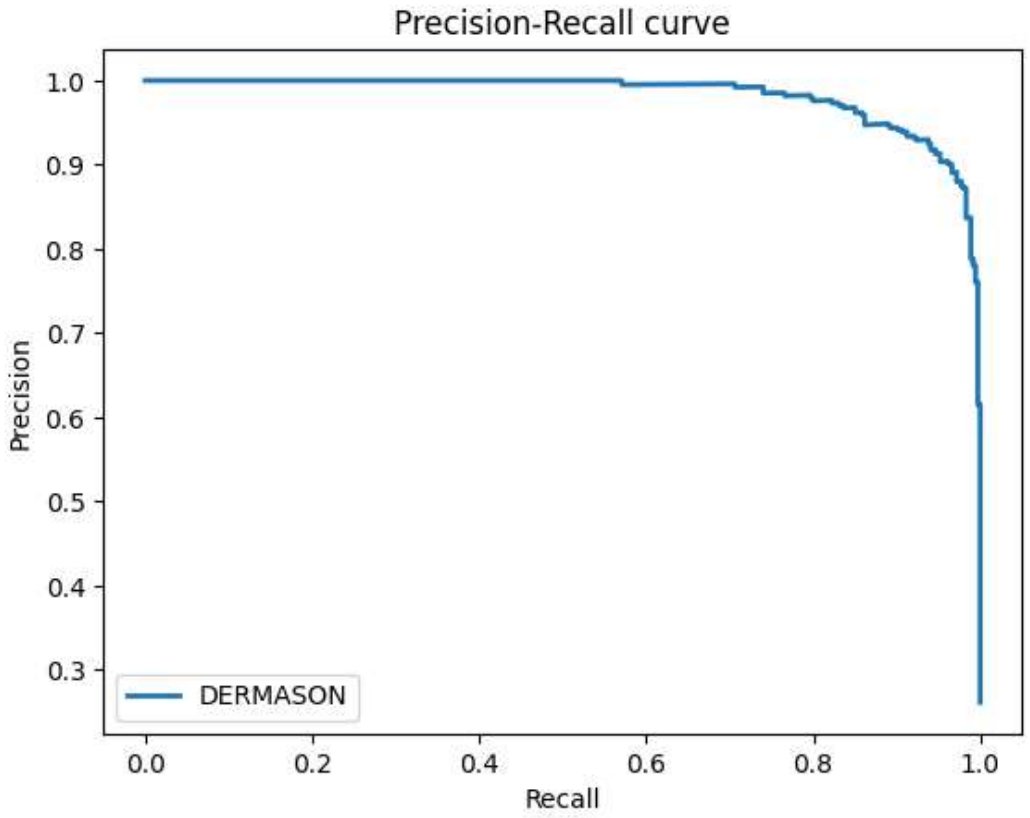


Figure 6: Dermason Precision-Recall Plot

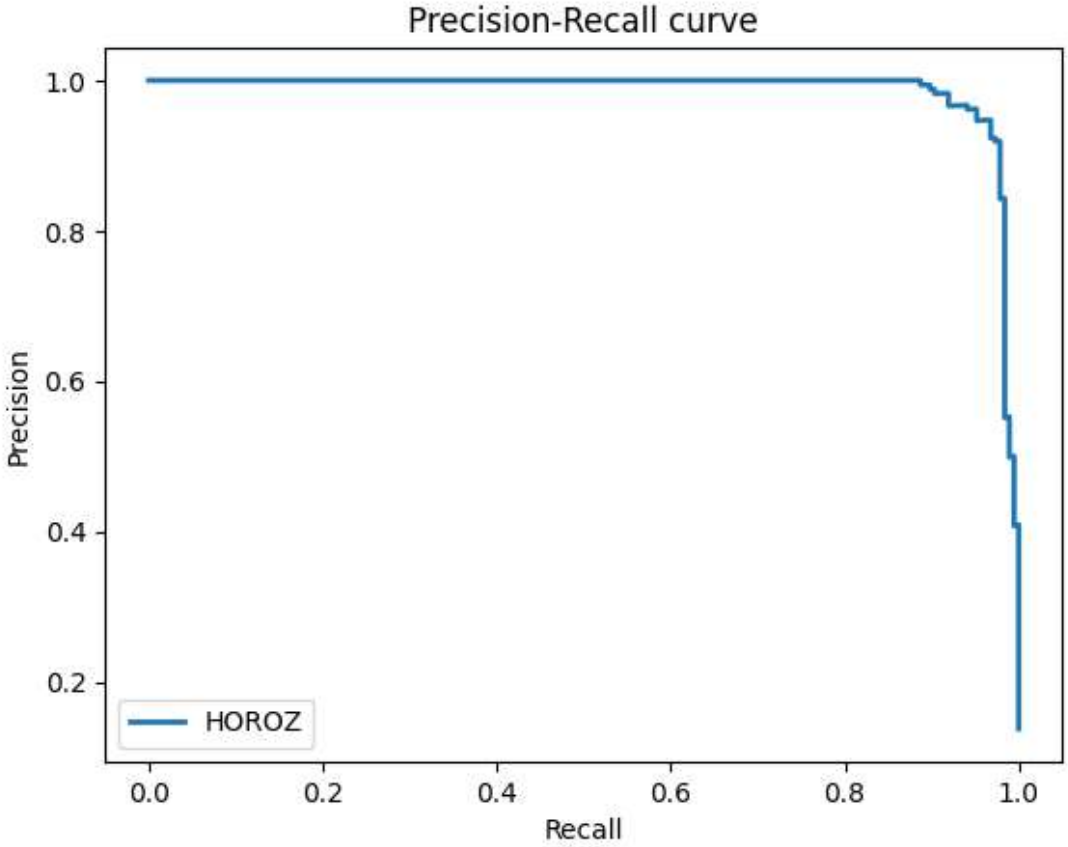


Figure 7: Horoz Precision-Recall Plot

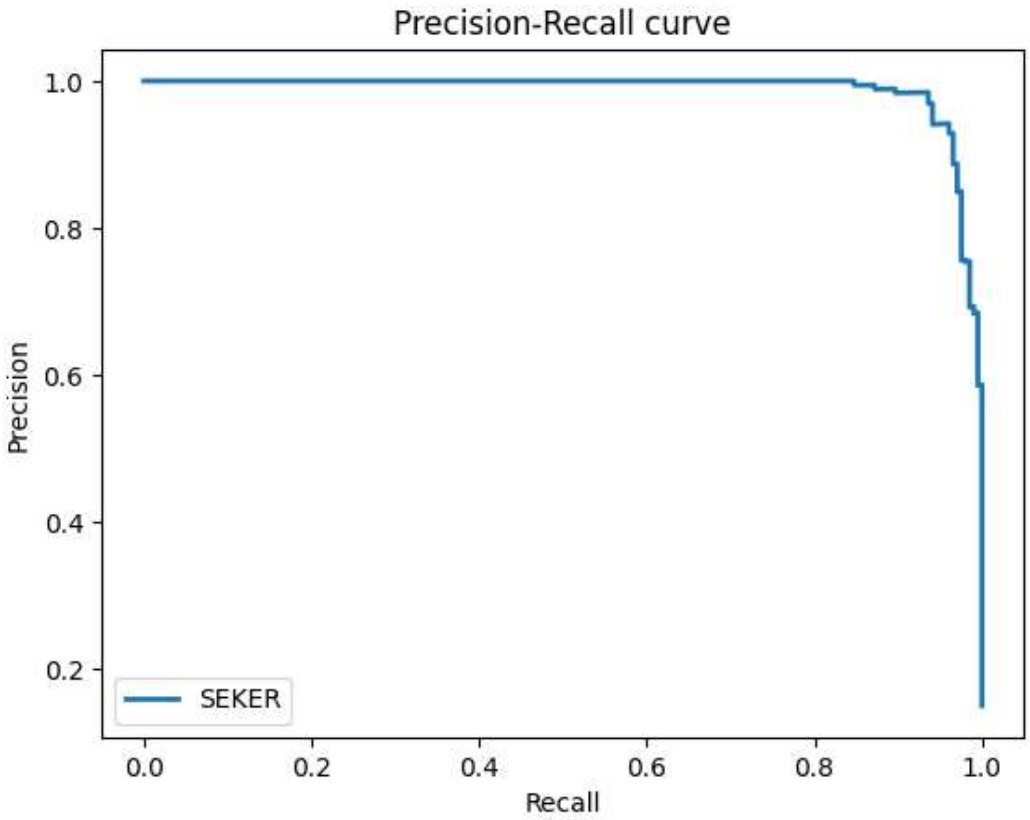


Figure 8: Seker Precision-Recall Plot

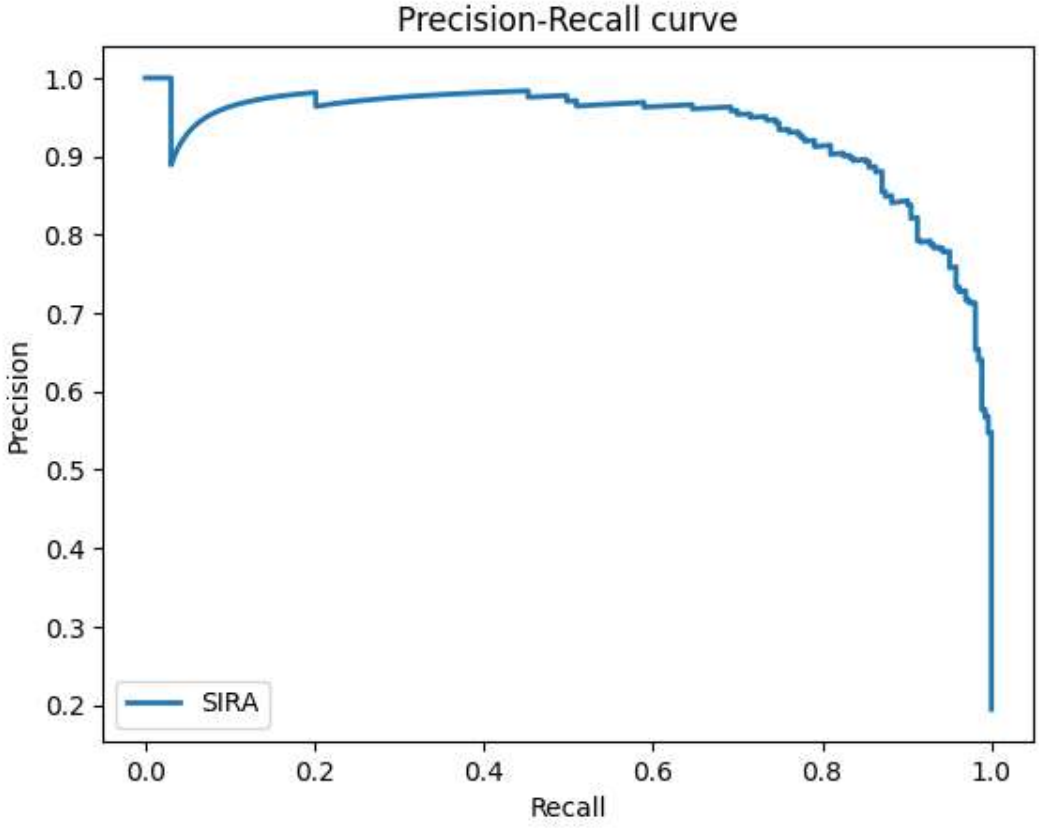


Figure 9: Sira Precision-Recall Plot

References

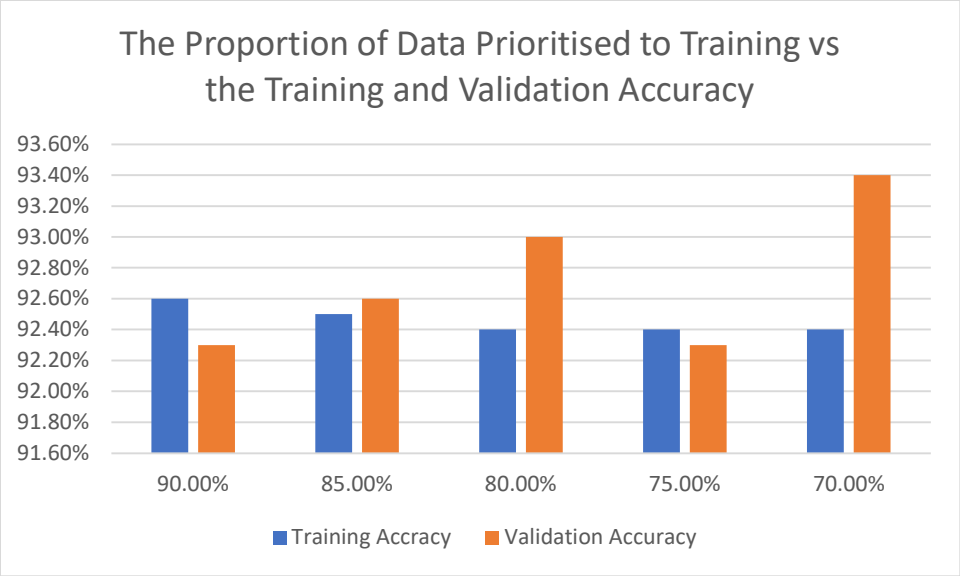
Baheti, P. (2023) *Train test validation split: How to & best practices [2023]*, *V7*. Available at: https://www.v7labs.com/blog/train-validation-test-set (Accessed: February 20, 2023).

Harris-Birtill, D. and Terzić, K. (2023) "Lecture 5: Logistic Regression," *CS5014 Machine Learning*, 20 February.

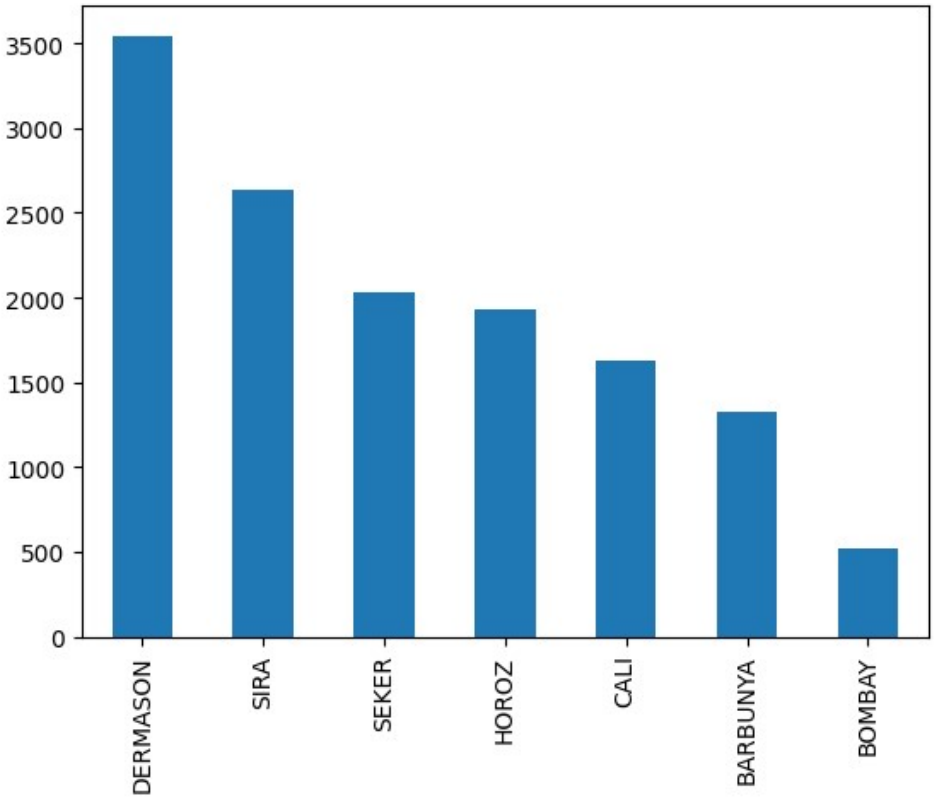
Mahmood, M.S. (2022) *Outlier detection (part 1)*, *Medium*. Towards Data Science. Available at: https://towardsdatascience.com/outlier-detection-part1-821d714524c (Accessed: February 20, 2023).

Terzić, K. and Harris-Birtill, D. (2023) "Lecture 9: Expansions & Regularisation," *CS5014 Machine Learning*, 20 February.

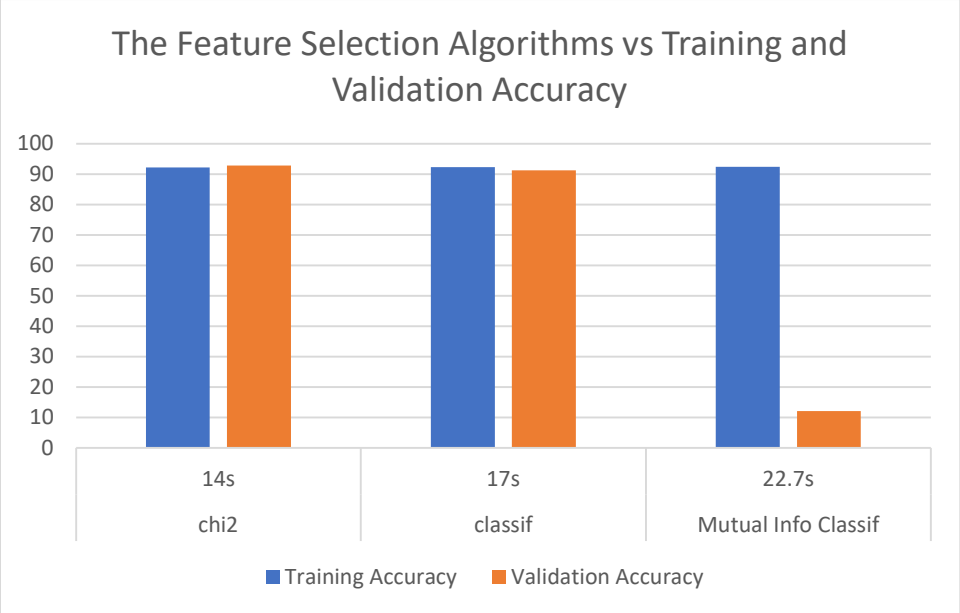
Appendices



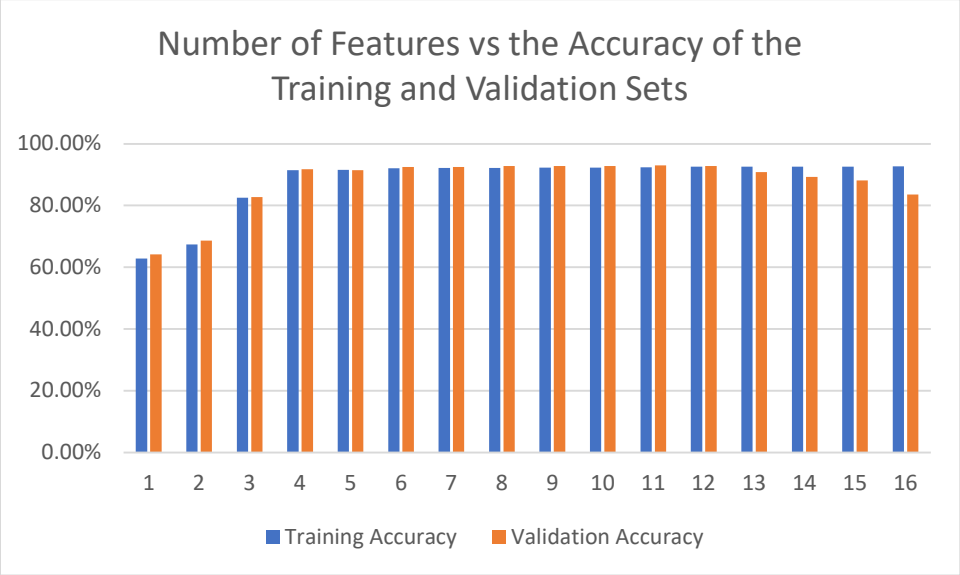
Appendix One: A Bar Chart to Show the Proportion of Data Prioritised to Training Against the Training and Validation Accuracy



Appendix Two: A Bar Chart to Show the Distribution of Each Bean Class within the Bean Dataset



Appendix Three: A Bar Chart to Show the Relationship Between Each Feature Selection Algorithm against Training and Validation Accuracy



Appendix Four: A Bar Chart to Show the Relationship Between the Number of Features Against the Accuracy of the Training and Validation Sets