Katarzyna Michalska, 244577IV

1 Distance functions

Distance functions such as Euclidean, Manhattan, Chebyshev and Minkowski, with various values of the parameter P were implemented in R. The following plot illustrates the geometric interpretation of these distance functions, highlighting the effects of different P values in the Minkowski distance, based on the referenced articles. [1].

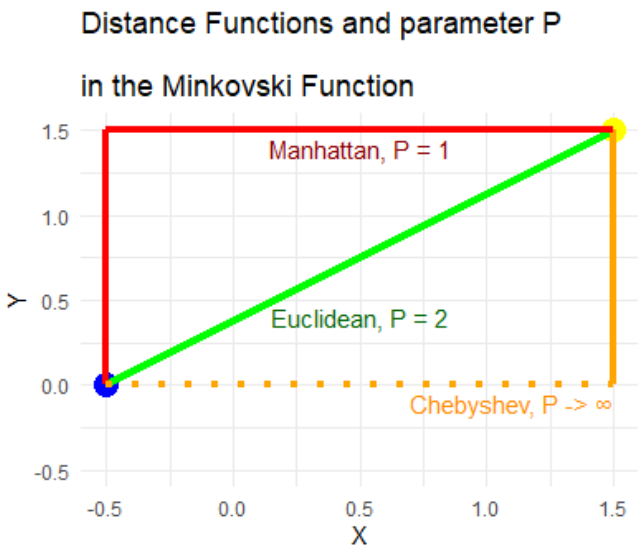


Figure 1: Distance functions and parameter P in the Minkowski function.

2 Feature selection

2.1 The Entropy Function

The entropy function was implemented based on lecture slides and tested on a dataset split by two different thresholds. The overall entropy was 1, as the dataset had an equal number of samples in two classes. For threshold = 1, the left subset had an entropy of 0.177 (dominated by label B), while the right had an entropy of 0.403 (containing more label A). For threshold = 3, the left subset had an entropy of 0.999 (almost equal samples of labels A and B), and the right had an entropy of 0 (only label A).

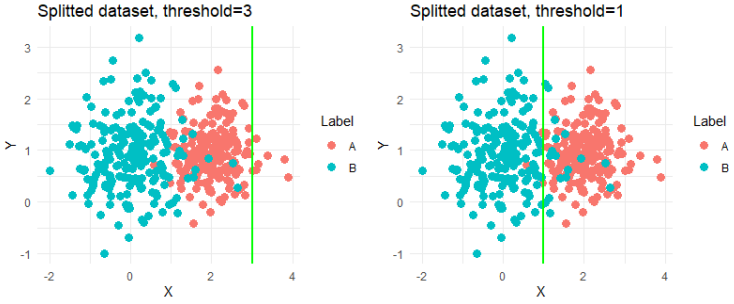


Figure 2: Datasets.

2.2 The Fisher Score

The Fisher Score was tested in two scenarios: one separable and the other non-separable. In the separable case, the Fisher Score is expected to be higher than in the non-separable case, as it indicates how well labels are distinguished from each other. The plot below illustrates the datasets for both scenarios.

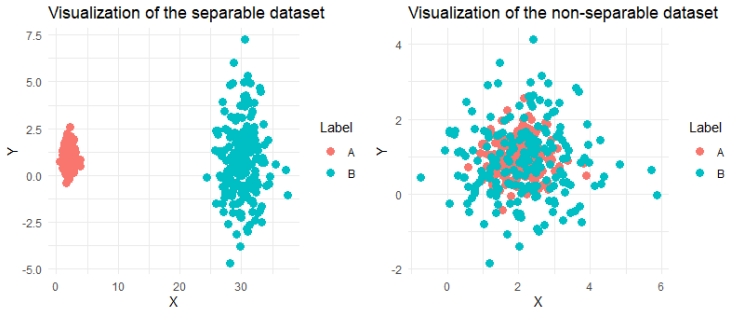


Figure 3: Separable set.

As expected, in the first case, the Fisher Score achieved a value of 0.004 for feature X and 0.003 for feature Y, while in the second case, it achieved 80.03 for feature X and 43.953 for feature Y.

3 Classification

A decision tree classifier was implemented in R, followed by a graphical illustration of the results, with guidance from the article [2]. First, the Gini function was implemented to measure impurity in the dataset. Next, a function responsible for splitting the dataset based on a given threshold was created. Then, the algorithm for selecting the optimal split was implemented. Finally, the decision tree was built using the training data, and a function to make predictions on the test data was implemented.

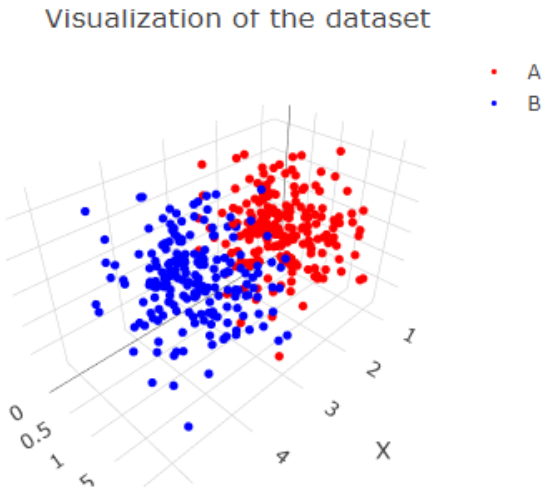


Figure 4: Graphical illustration of the dataset.

Custom performance metrics, including accuracy, precision, recall, and F1 score, were also developed. Their values were evaluated during the cross-validation process. In this process

data was split into 5 folds to assess the training and testing performance. The following table 1 presents the values of each metric for each iteration.

Table 1: Good metrics - Decision Tree Classifier

Fold	Accuracy	Precision	Recall	F1 Score
k = 1	0.9250	0.9705	0.8684	0.9167
k = 2	0.9375	0.9375	0.9091	0.9231
k = 3	0.9375	0.9063	0.9355	0.9206
k = 4	0.9625	1.0000	0.9412	0.9697
k = 5	0.9750	1.0000	0.9574	0.9783

Along with the metrics, the ROC curve values were calculated and visualized across the iterations during the cross-validation process. The implementation of ROC curves was supported by the help of packages from the following sources [3].

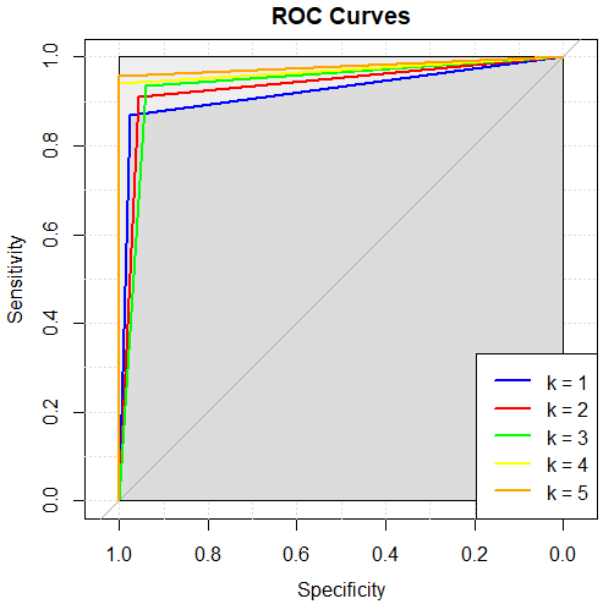
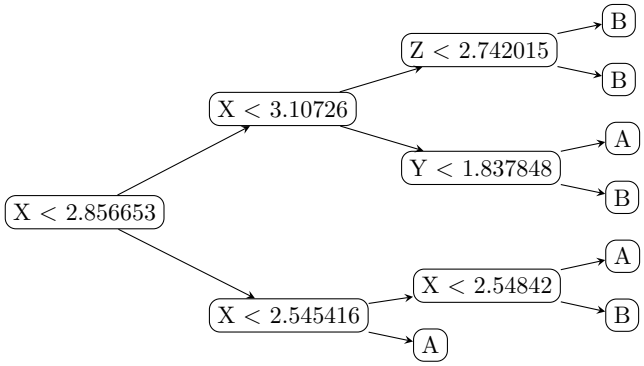


Figure 5: ROC Curves through folds.

By observing the plot above and the metric values in Table 1, we can infer that k=5 was likely the best iteration. This assumption is based on the fact that an ideal ROC curve should approach a 90-degree angle. It is also evident that the metrics are closest to 1 during the 5th iteration. The following graph represents the tree in the 5th iteration.



4 Regression

The feature selection and regression model were built using backward elimination. This algorithm begins by constructing a model that includes all features. It then evaluates the significance of each feature using p-values. Features deemed insignificant are eliminated, and the process continues until no more insignificant features remain. The algorithm was tested on a dataset containing six features, which were reduced to only two through successive iterations. Table 2 illustrates the features that were removed during each iteration.

Table 2: Backward Elimination

Iteration	x1	x2	x3	x4	x5	x6
i = 1	0.250	<2e-16	0.658	<2e-16	0.750	0.741
i = 2	0.250	<2e-16	0.658	<2e-16	deleted	0.741
i = 3	0.250	<2e-16	0.658	<2e-16	deleted	deleted
i = 4	0.250	<2e-16	deleted	<2e-16	deleted	deleted
i = 5	deleted	<2e-16	deleted	<2e-16	deleted	deleted

At the end of the algorithm, only two features—x2 and x4—remained. The covariance matrix was then examined 3, and since everything appeared satisfactory (with high covariance between the features and the target variable, and low covariance among the features), a model was built using these selected features.

Table 3: Covariance Matrix

	x2	x4	y
x2	1	-0.007	0.742
x4	-0.007	1	0.612
y	0.742	0.612	1

The following plot was generated using the function demonstrated in the practice lesson. It displays the model along with the new data.

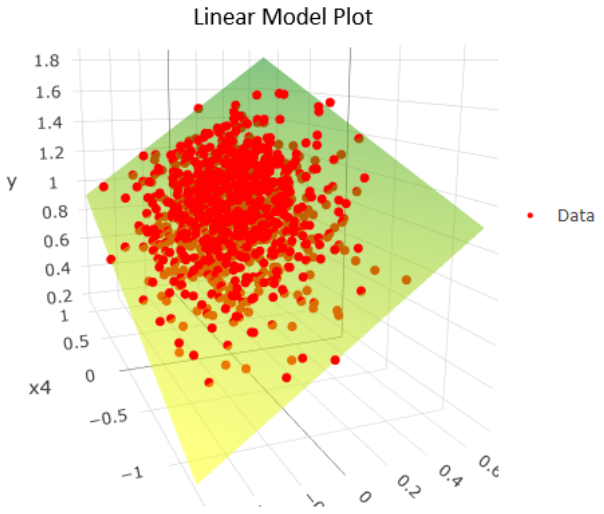


Figure 6: Linear model.

References

[1] Jonte Dancker. A brief introduction to Distance Measures. https://medium.com/@jodancker/a-brief-introduction-to-distance-measures-ac89cbd2298, 2022.

[2] Rcshmin. Decision Trees. https://rpubs.com/Rcshmin/924453, 2022.

[3] Alexandre Hainard Xavier Robin, Natacha Turck. pROC: an open-source package for R and S+ to analyze and compare ROC curves. https://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-12-77, 2011.