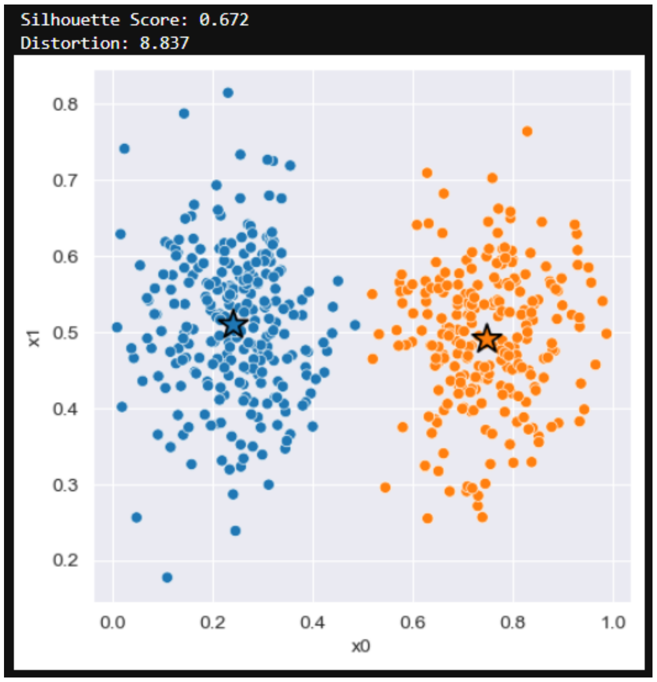
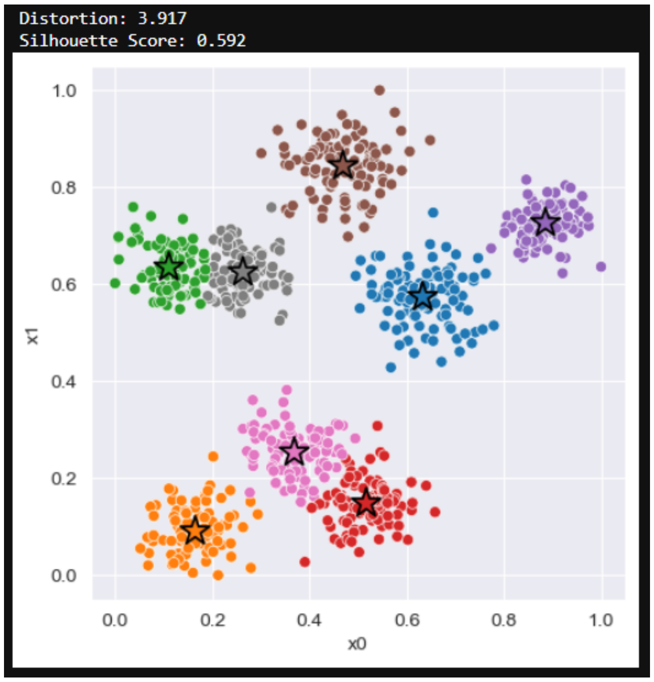
K-means

Results and Plots



```
Silhouette Score: 0.672
Distortion: 8.837
```



```
Distortion: 3.917
Silhouette Score: 0.592
```

Algorithm Explanation

The KMeans class is initialised with two parameters: clusters and iterations. It uses a maximin-similar algorithm to initialise centroids, starting with a random point and then maximising the minimum distance to existing centroids. The fit() method assigns data points to the nearest centroids using Euclidean distance and updates centroids by computing the mean of assigned data points. This process repeats for the specified number of iterations or until centroids stabilise. The predict() method is used to assign new data points to existing clusters after the model has been trained. It calculates the euclidean distances between the data points and the centroids and assigns each data point to the cluster associated with the nearest centroid. The find_best_model() function runs the K-means algorithm multiple times and selects the model with the lowest distortion. This is done to mitigate the issue of finding suboptimal solutions due to the randomness in centroid initialisation.

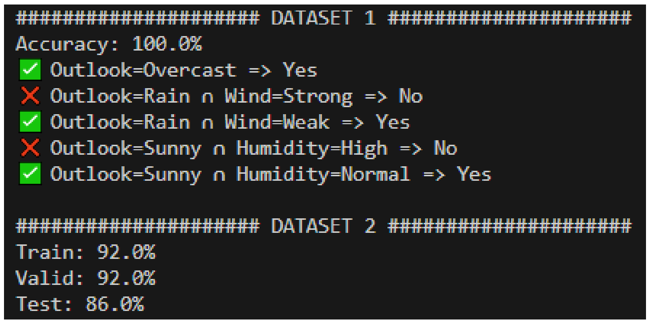
KMeans is a clustering algorithm that can be used to discover categories or groups in unlabelled datasets, unsupervised. As such, it has many interesting use cases, among many: customer segmentation, document classification, anomaly and bot detection or maybe even in medical image analysis.

Inductive Bias and Solving the Second Dataset

K-means aims to minimise within-cluster variance to get nice and “compact” clusters, assuming clusters are dense and often, similarly sized. Also it is sensitive to outliers and the initial placement of cluster centroids, making e.g. overlapping clusters as in dataset 2 hard. To work around this, I normalised the data for dataset 2, balancing the influence of different features by putting them on a common scale. I also tried multiple initialisation methods, landing on maximin. Additionally, I initialise the K-Means algorithm a number of times to keep the model with the lowest distortion.

Decision Tree

Results



```
#################### DATASET 1 ####################
Accuracy: 100.0%
✅ Outlook=Overcast => Yes
❌ Outlook=Rain ∩ Wind=Strong => No
✅ Outlook=Rain ∩ Wind=Weak => Yes
❌ Outlook=Sunny ∩ Humidity=High => No
✅ Outlook=Sunny ∩ Humidity=Normal => Yes

#################### DATASET 2 ####################
Train: 92.0%
Valid: 92.0%
Test: 86.0%
```

Algorithm Explanation

The DecisionTree class initialises an empty tree without any hyperparameters. In the fit() method, the decision tree is constructed through recursive feature selection. select_best_split() chooses the feature with the highest information gain (entropy reduction) using get_information_gain(). Recursion continues by calling fit() to the data subsets resulting from the split. This repeats until all labels are the same (leaf node) or no features remain to split (majority label is chosen).
The predict() method uses the trained tree to predict labels for input samples. It iterates through samples, starting at the root, following branches based on feature values until reaching a leaf node. Predicted labels are collected and returned as a list of predictions.

Decision trees are well-suited to classification problems where instances are described by a fixed set of attributes with discrete values (though real-valued attributes can be handled by more advanced versions of the algorithm). It works best when the target function has discrete output values, allows for disjunctive descriptions, and can handle training data errors and missing attribute values. Decision trees have been successfully applied to solve practical problems like medical diagnosis, equipment malfunction prediction, and loan applicant classification (page 54, “Machine Learning”, Tom M. Mitchell).

Inductive Bias and Solving the Second Dataset

ID3’s inductive bias is that shorter trees are preferred over longer trees, and that high information gain attributes should be placed closer to the root. In other words, ID3 tries to create a tree that is as simple as possible while still accurately classifying the training data. It assumes that a shorter tree is more likely to generalise well to unseen data.

The second dataset contained an irrelevant feature, "Birth Month," which doesn't contribute to predicting the target variable, "Outcome." This irrelevant feature can mislead the algorithm by causing it to learn irrelevant data, leading to poor generalisation beyond the training set. This issue is linked to the algorithm's inductive bias, where irrelevant features will be greedlily included in the tree if they happen to give high information gain during training, potentially resulting in overfitting. To get around this problem, I removed the "Birth Month" column. Before removal, the algorithm achieved a high training data accuracy of 100%, but struggled to see the“true” patterns, resulting in low accuracy (~66%) for other datasets. After removing "Birth Month," accuracy increased to 92% for the validation dataset and 86% for the testing dataset.