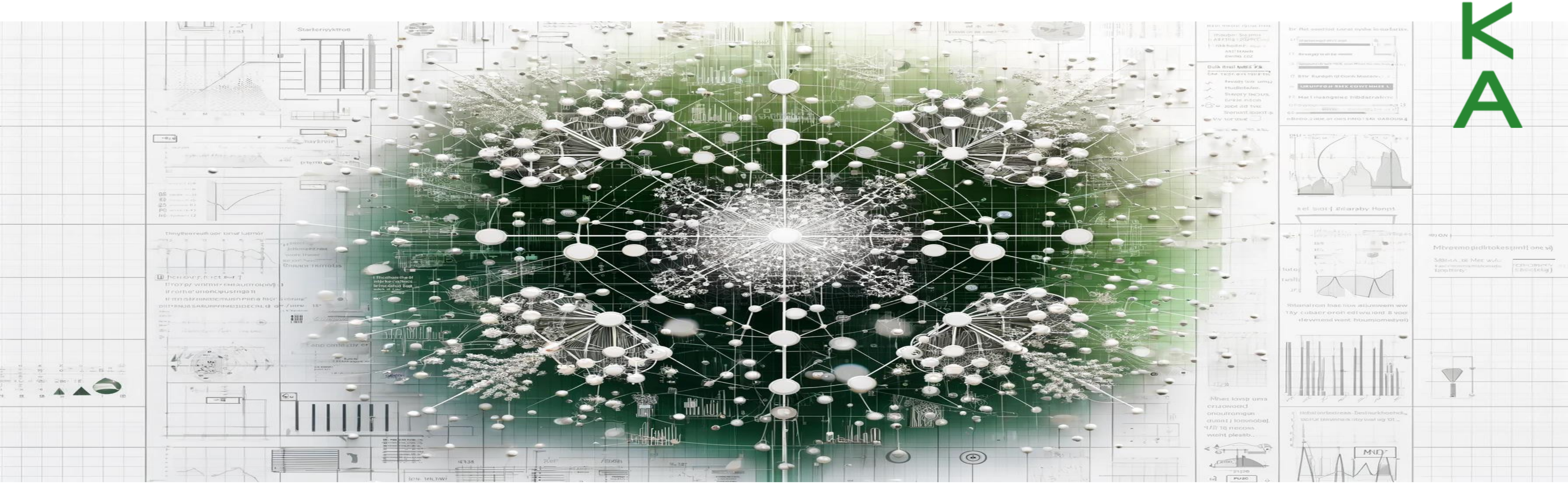
Hochschule Karlsruhe
University of
Applied Sciences

Fakultät für
Elektro- und
Informationstechnik

K-Nearest Neighbors And Decision Trees



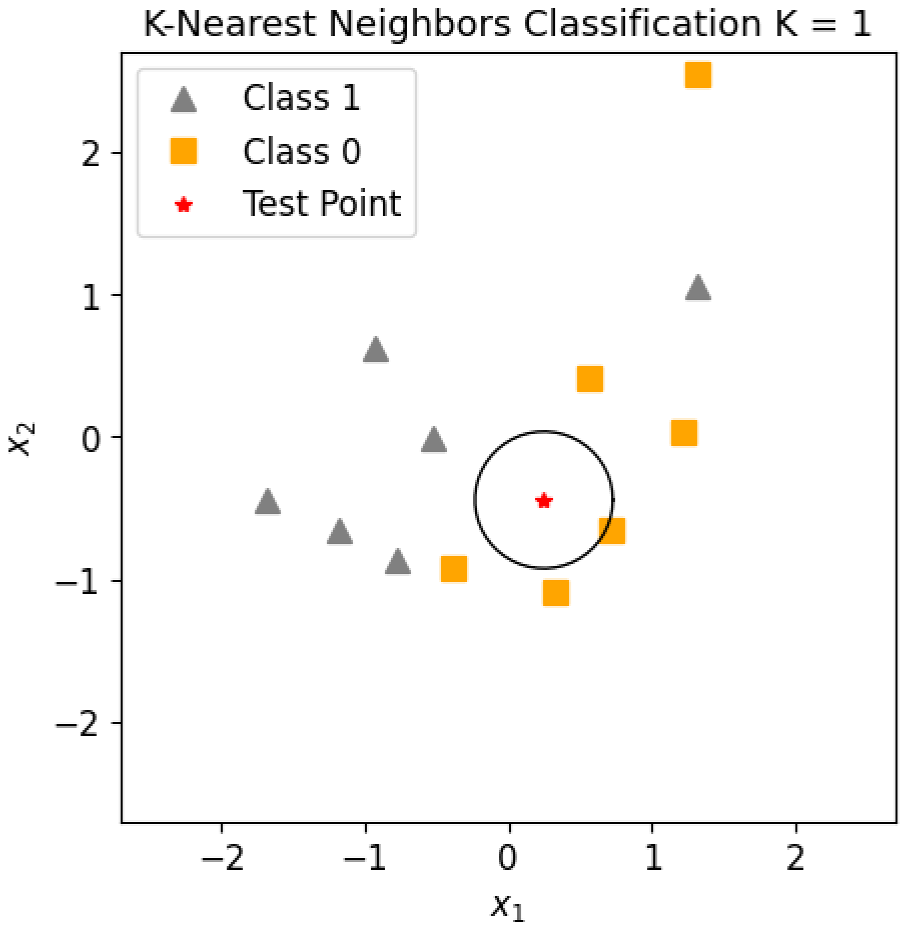
Source: DALL.E

Kawther Aboalam

K-Nearest Neighbors (KNN)

Introduction

- K-Nearest Neighbors (KNN) is a simple supervised machine learning algorithm.
- Instance-Based Learning (Lazy Learning):
  - Absence of a Model: Does not construct a model during the training phase
  - Storage of Instances: Stores instances of the training data to make Predictions
  - On-Demand Computation: Performs computation at the time of prediction rather than during training
- Non-Parametric: Assume that the data distribution cannot be defined in terms of a finite set of parameters. The amount of information that can be captured about the data can grow as the amount of data grows. This makes it more flexible
- Versatile Application: Applicable to both classification and regression tasks



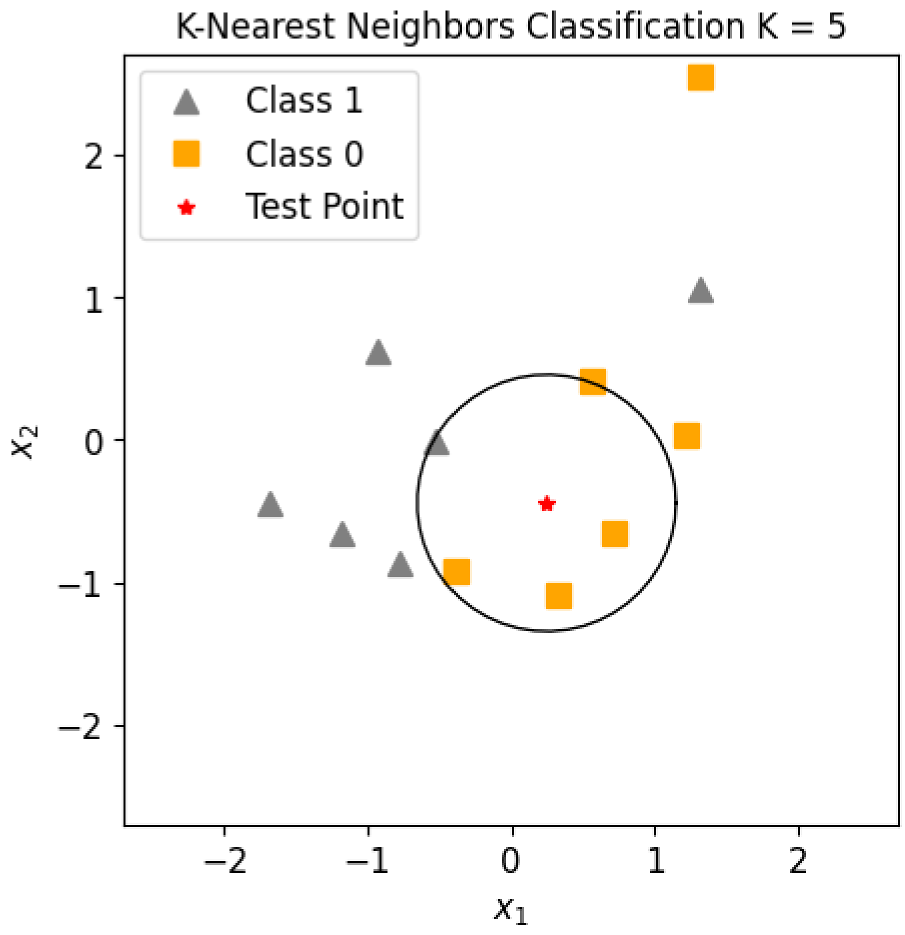
K-Nearest Neighbors (KNN)

How does the KNN algorithm work?

- KNN is **a local approach**, because it makes predictions based on the k-nearest training data points in the feature space
- The steps can be summarized as the following:
  1. Choose the number of neighbors (k)
  2. Calculate the distance between the test point and all training points
  3. Identify the K points in the training data that are closest to the test point
  4. For classification tasks, predict the most frequent class among its k nearest neighbors

$$\hat{y} = \text{mode}(y_1, y_2, \ldots, y_k)$$

Where, y_i is the class label of the ith nearest neighbor



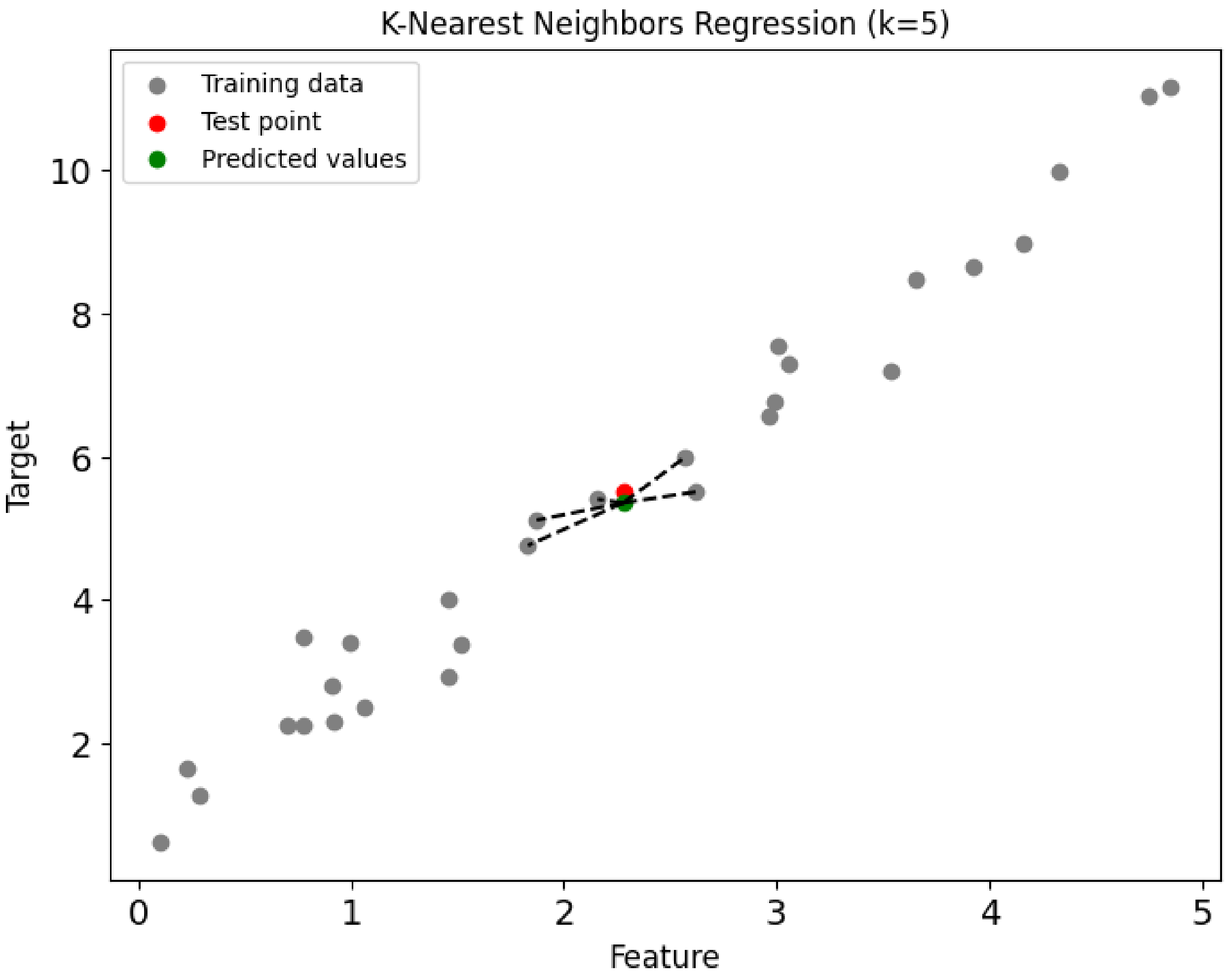
K-Nearest Neighbors (KNN)

How does the KNN algorithm work?

- KNN makes predictions based on the k-nearest training data points in the feature space
- The steps can be summarized as the following:
  1. Choose the number of neighbors (k)
  2. Calculate the distance between the test point and all training points
  3. Identify the K points in the training data that are closest to the test point
  4. For regression, predict the average of the target values of the k nearest neighbors

$$\hat{y} = \frac{1}{k}\sum_{i=1}^{k} y_i$$

where, y_i is the target value of the i^{th} nearest neighbor



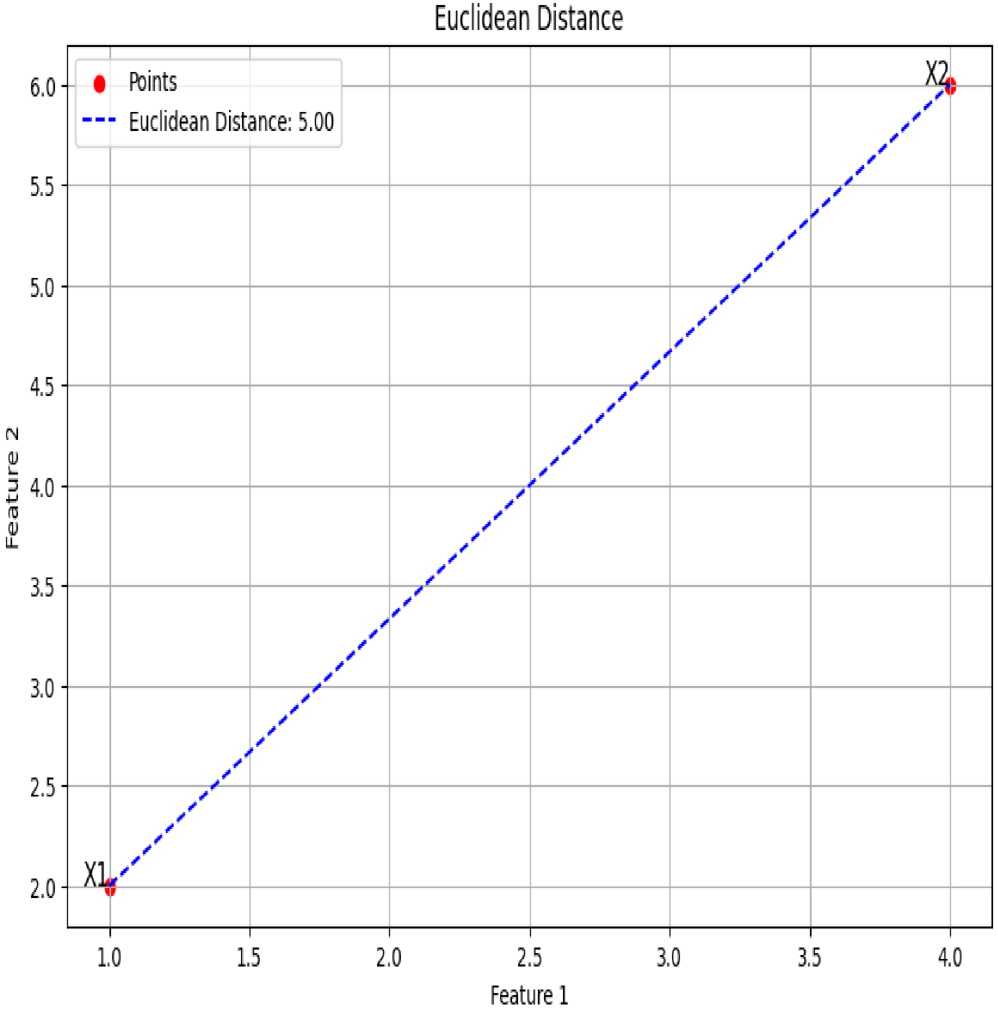
K-Nearest Neighbors (KNN)

How to calculate the distances?

- Assume a dataset with m features. The feature vector X consists of m components: $x_1, x_2, \ldots, x_m$
- The distances between two points $X_1 = (x_{11}, x_{12}, \ldots, x_{1m})$ and $X_2 = (x_{21}, x_{22}, \ldots, x_{2m})$ in this dataset can be computed using various distance metrics
- The most common distance metrics for KNN are:

1. Euclidean Distance

$$d_{euclidean}(X_1, X_2) = \sqrt{\sum_{i=1}^{m} (x_{1i} - x_{2i})^2}$$



K-Nearest Neighbors (KNN)

How to calculate the distances?

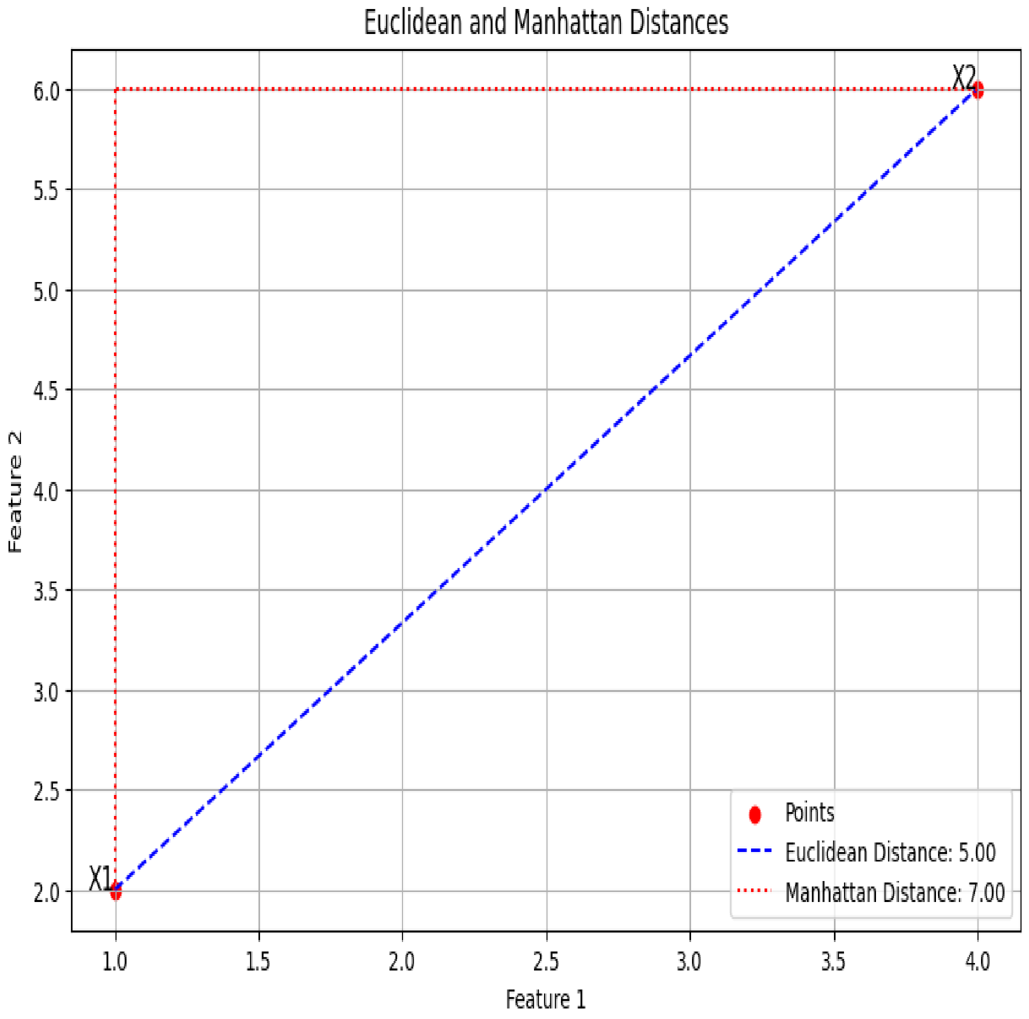
+ Assume a dataset with m features. The feature vector X consists of m components: x_1, x_2,…, x_m
+ The distances between two points $X_1 = (x_{11}, x_{12}, \ldots, x_{1m})$ and $X_2 = (x_{21}, x_{22}, \ldots, x_{2m})$ in this dataset can be computed using various distance metrics
+ The most common distance metrics for KNN are:

1. Euclidean Distance

$$d_{euclidean}(X_1, X_2) = \sqrt{\sum_{i=1}^{m}(x_{1i} - x_{2i})^2}$$

2. Manhattan Distance

$$d_{manhattan}(X_1, X_2) = \sum_{i=1}^{m}|x_{1i} - x_{2i}|$$



K-Nearest Neighbors (KNN)

How to calculate the distances?

3. Minkowski Distance

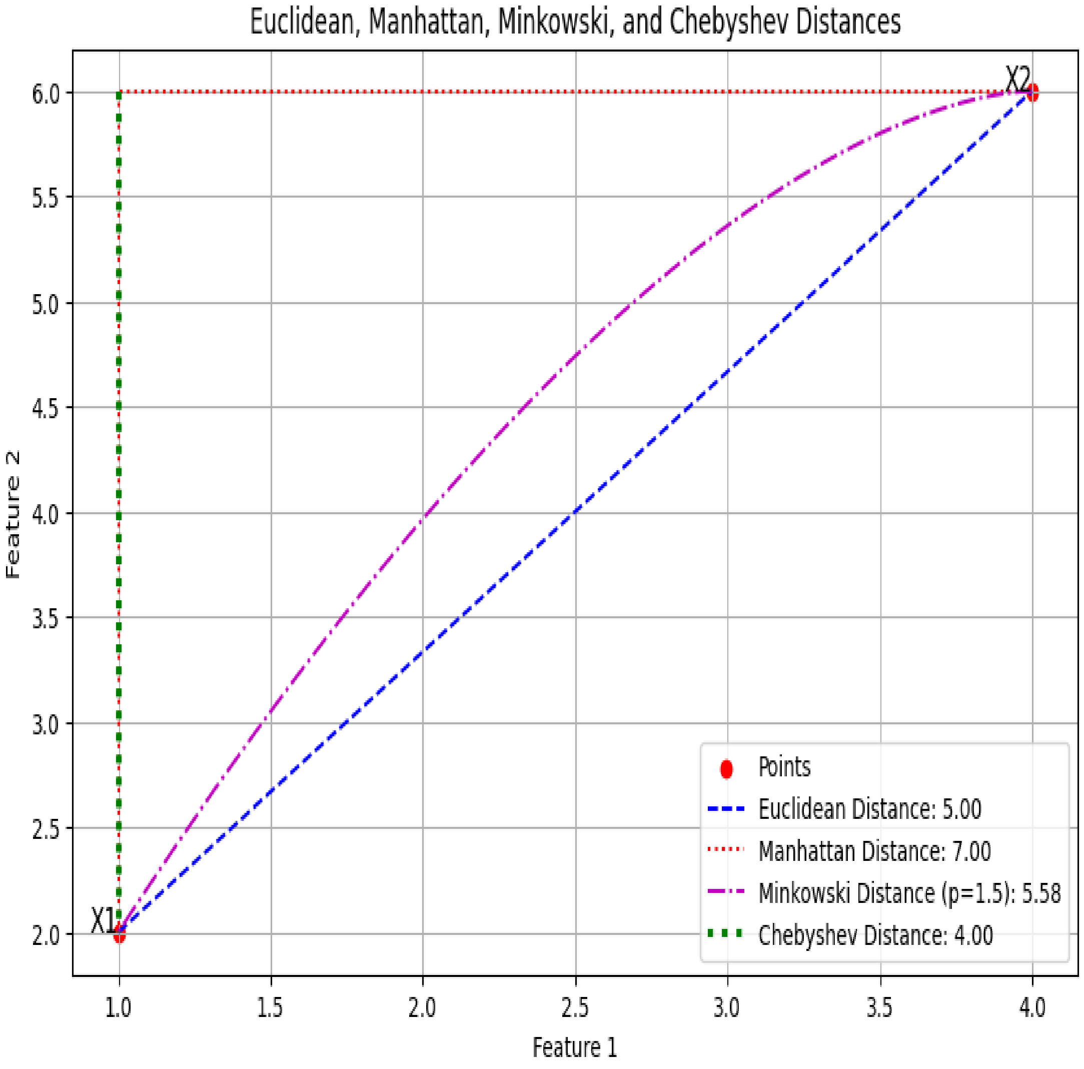
- It is a parameterized distance metric

$$d_{minkowski}(X_1, X_2, p) = \left(\sum_{i=1}^{m} |x_{1i} - x_{2i}|^{\mathrm{p}}\right)^{1/p}$$

- For p=1, it represents the Manhattan distance.
- For p=2, it represents the Euclidean distance.
- For other values of p, it provides a flexible metric that can be tuned to the specific problem.
- In the limiting case of p reaching infinity, the Chebyshev distance is obtained:

$$\lim_{p\to\infty} \left(\sum_{i=1}^{m} |x_{1i} - x_{2i}|^{\mathrm{p}}\right)^{1/p} = max_i\,(|x_{1i} - x_{2i}|)$$

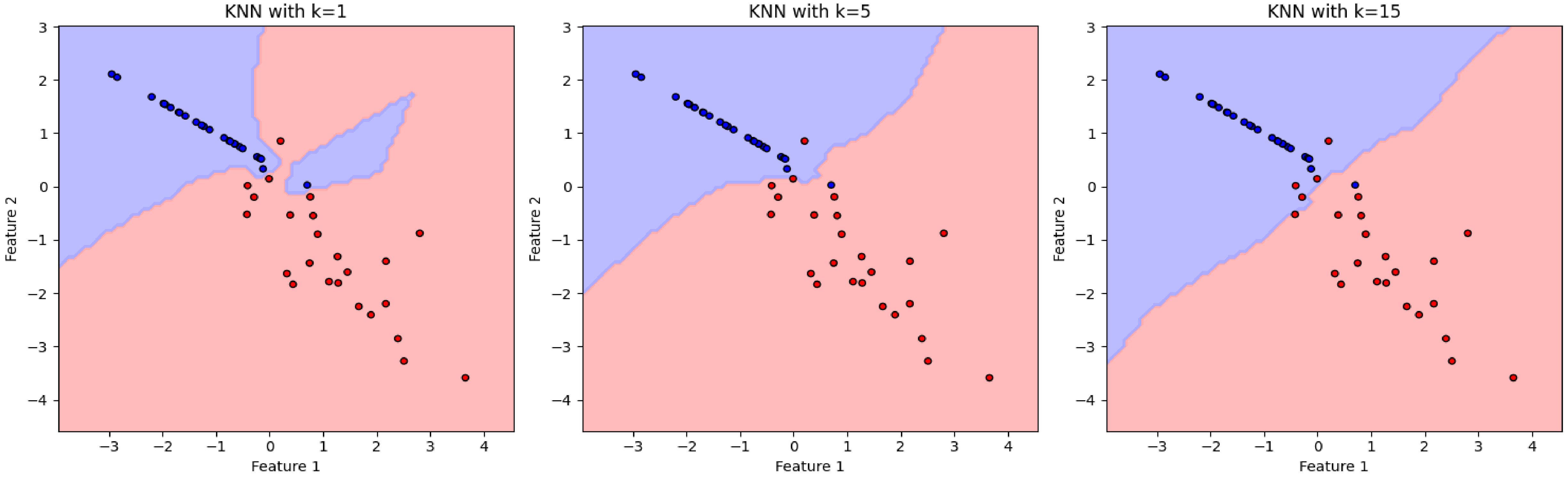
- Selecting a suitable distance metric can improve the results of KNN



K-Nearest Neighbors (KNN)

What is the impact of the value “k” on decision boundaries?

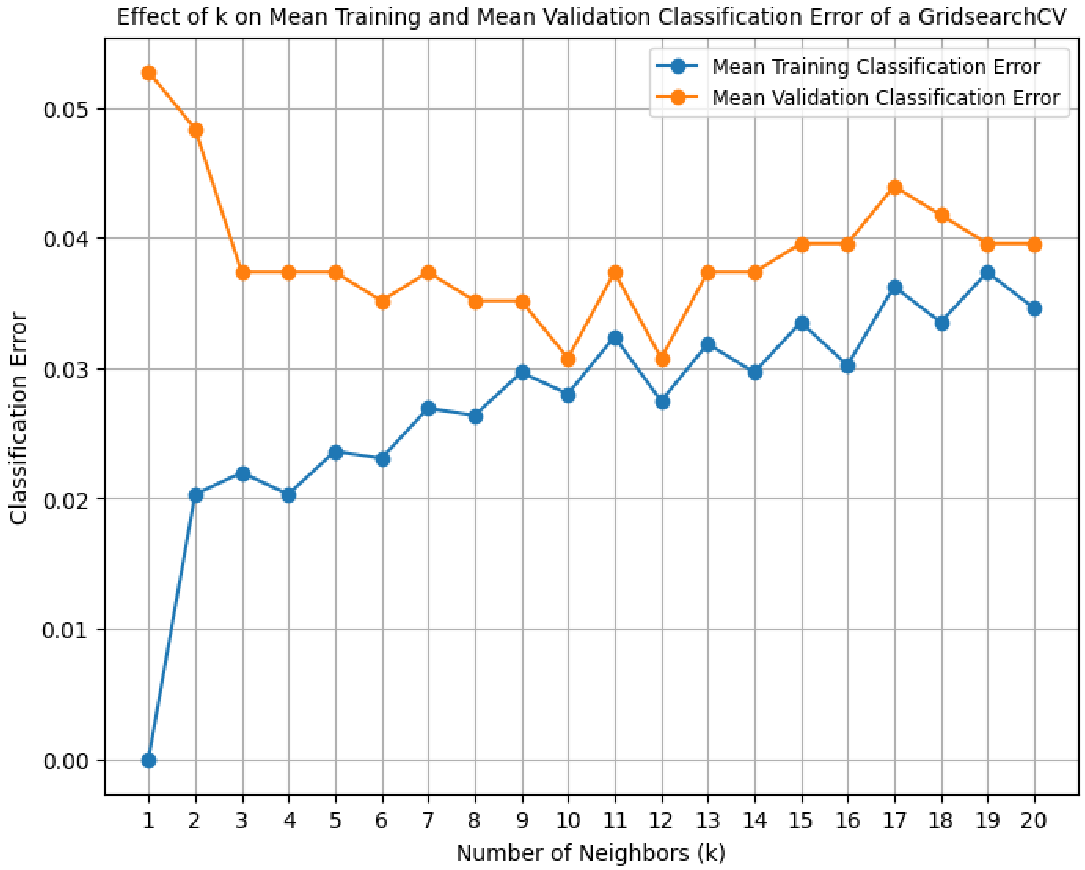
- K is a key hyperparameter that significantly affects the model's performance.
- Selecting an appropriate k value involves balancing the trade-off between bias and variance.
- For small k values, the decision boundaries are too complex, leading to poor performance on the validation set.
- For large k values, the decision boundaries are too simple, and the model lacks the flexibility to capture the underlying data structure.



K-Nearest Neighbors (KNN)

How to choose k?

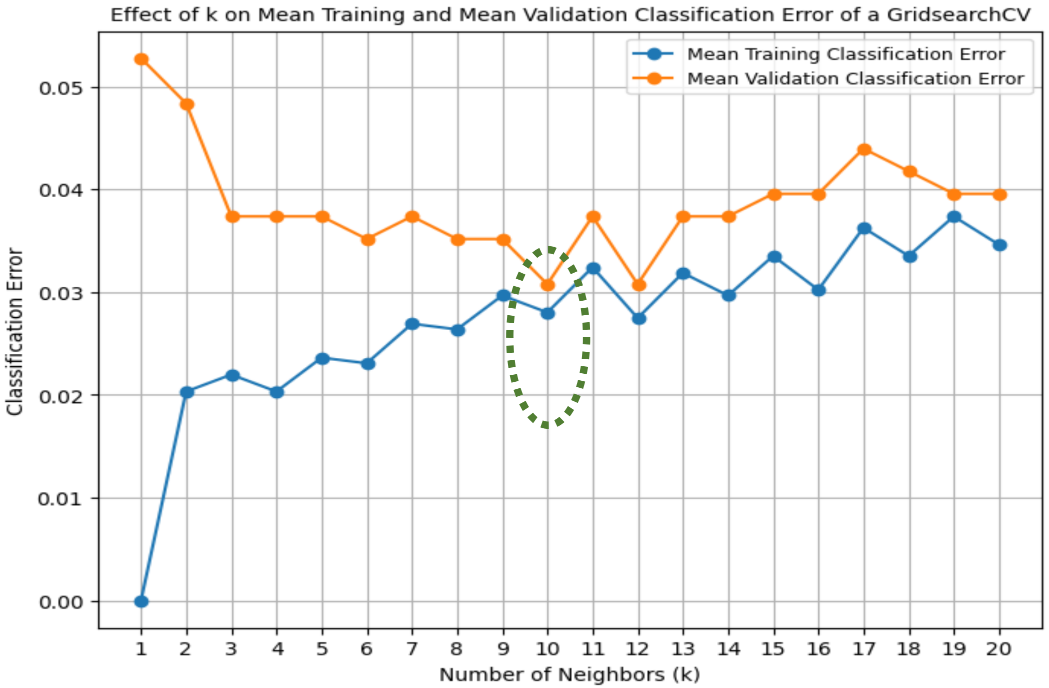
- The value of k in the K-Nearest Neighbors (KNN) algorithm significantly impacts the training and validation error
- Using grid search optimization method and K-fold cross validation strategy, k is tuned as a hyperparameter to select the best k for a specific problem
- Small k Values:
  - For k=1, the training classification error is zero. Where each training point is essentially its own nearest neighbor, leading to perfect classification on the training set.
  - The validation classification error tends to be high due to overfitting because of capturing noise and minor fluctuations in the training data

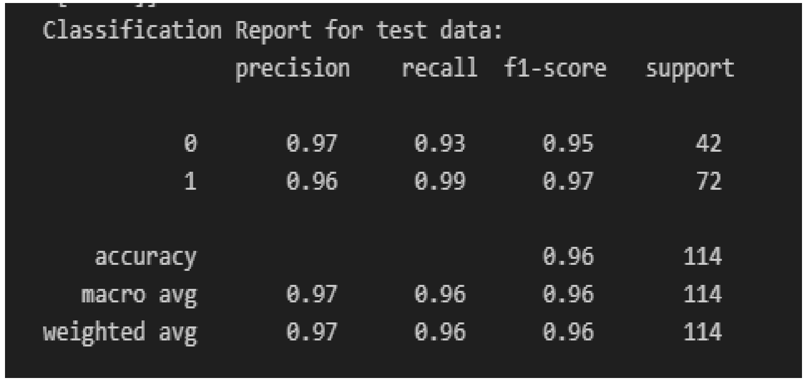


K-Nearest Neighbors (KNN)

How to choose k?

- Large k Values:
  - The algorithm averages over a larger number of neighbors and it is less sensitive to individual data points resulting in higher training error. This smoothing effect reduces variance but increases bias.
  - The validation error typically decreases initially as k increases from very small values, reaching an optimal point. However, as k becomes too large, the validation error starts to increase again due to underfitting.
- Looking for an "elbow" point where the rate of improvement slows. This point is often a good choice for k
- For the breast cancer dataset, k =10 is the selected as the best hyperparameter by the GridsearchCV approach.



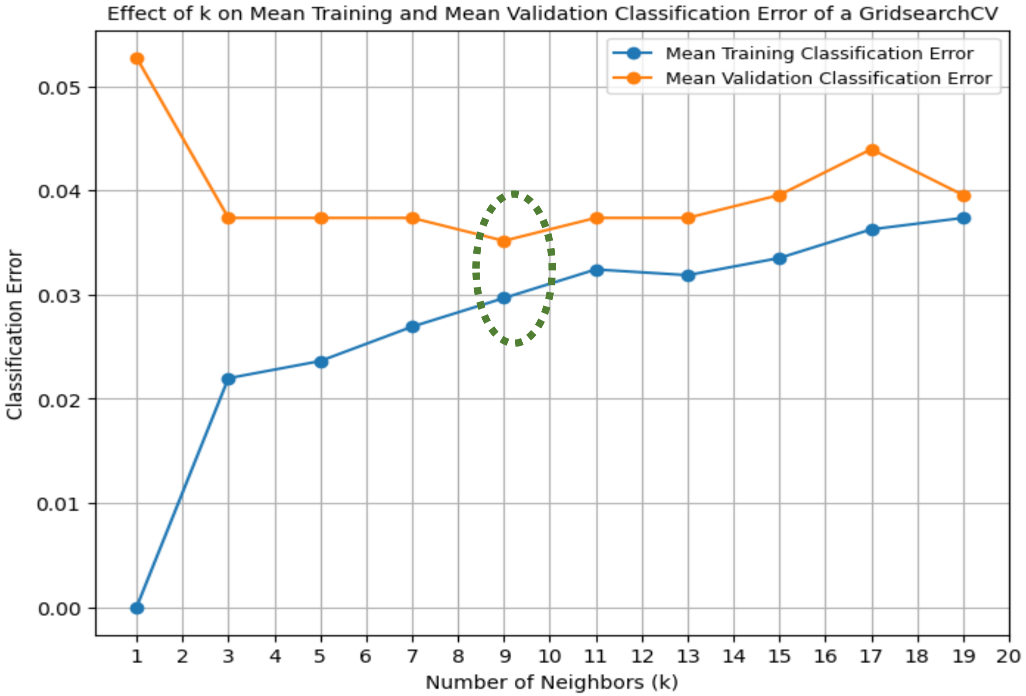


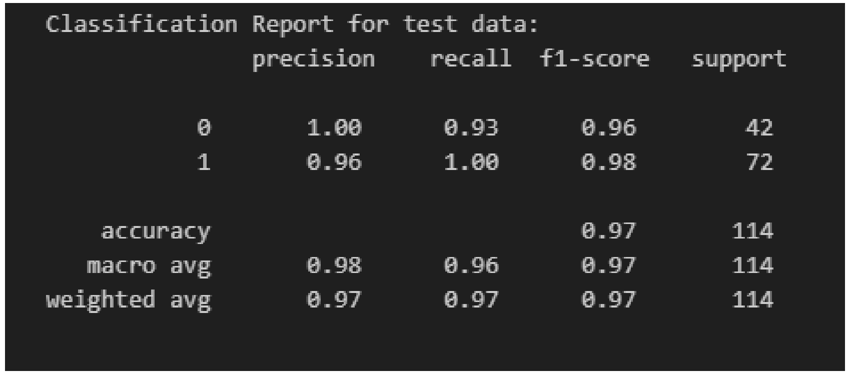
```
Classification Report for test data:
              precision    recall  f1-score   support

           0       0.97      0.93      0.95        42
           1       0.96      0.99      0.97        72

    accuracy                           0.96       114
   macro avg       0.97      0.96      0.96       114
weighted avg       0.97      0.96      0.96       114
```

K-Nearest Neighbors (KNN)

How to choose k?

- But an even k value (like 10) for a binary classification can lead to equal votes between the 2 classes.
- To reduce this risk in binary classification problems:
  - Use an odd value for k to avoid equal voting in binary classification. As passing only odd values to the GridsearchCV approach, the best selected k=9.
  - Alternatively, use distance-weighted voting, which gives more influence to closer neighbors and helps break equal voting risk even when k is an even number.





```
Classification Report for test data:
              precision    recall  f1-score   support

           0       1.00      0.93      0.96        42
           1       0.96      1.00      0.98        72

    accuracy                           0.97       114
   macro avg       0.98      0.96      0.97       114
weighted avg       0.97      0.97      0.97       114
```

K-Nearest Neighbors (KNN)

Distance Weighting in KNN

- Uniform Weighting
  - All neighbors contribute equally to the prediction. This is the default setting for KNN implementation in sci-kit learn
  - For classification tasks: $\hat{y} = \text{mode}(y_1, y_2, \ldots, y_k)$ + For regression tasks: $\hat{y} = \frac{1}{k}\sum_{i=1}^{k} y_i$
- Distance Weighting
  - Neighbors closer to the test point have a greater influence on the prediction than those further away
  - For classification tasks:

$$\hat{y} = argmax_c \sum_{i=1}^{k} w_i.1(y_i = c)$$

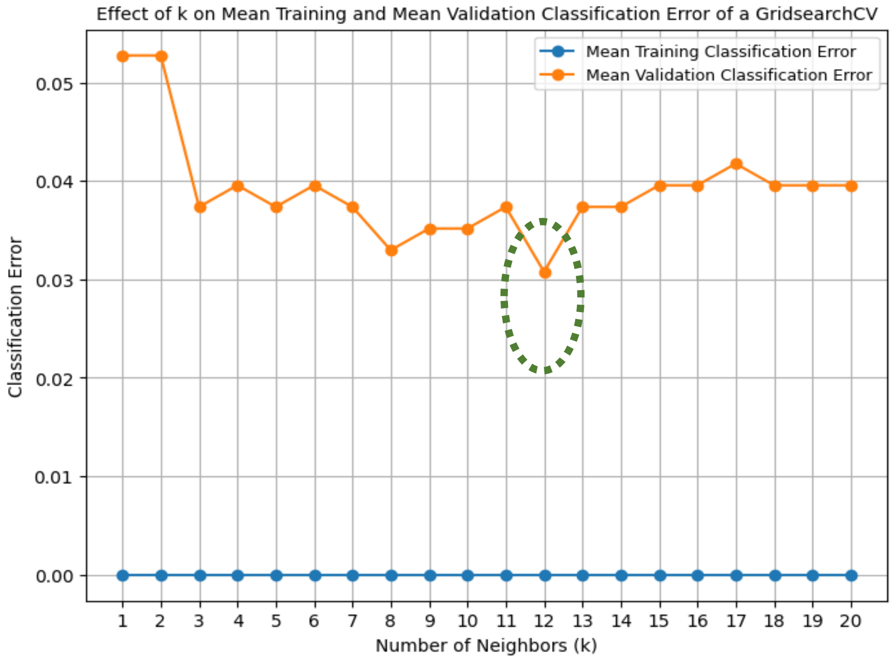
  - $w_i = \frac{1}{d_i}$ is the weight assigned to the ith nearest neighbor, with d_i being the distance of the ith nearest neighbor to the test point
  - $1(y_i = c)$ is an indicator function that is 1 if the class of the ith nearest neighbor is c, and 0 otherwise.

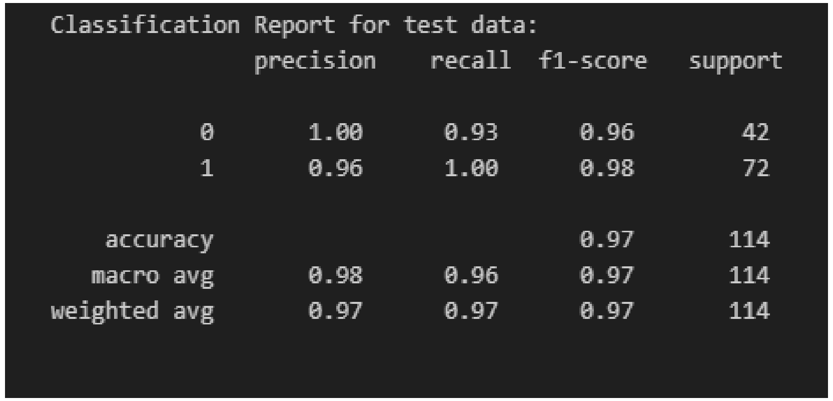
  - For regression tasks: $\hat{y} = \frac{\sum_{i=1}^{k}\frac{y_i}{d_i}}{\sum_{i=1}^{k}\frac{1}{d_i}}$
    - with d_i being the distance of the ith nearest neighbor to the test point

K-Nearest Neighbors (KNN)

+ Distance-Weighted Voting Results

- Optimal K = 12:
  Applying distance-weighted voting on the Breast Cancer dataset yields an optimal neighborhood size of 12.
- It produces the same classification report as the one obtained when restricting GridSearchCV to odd K values.



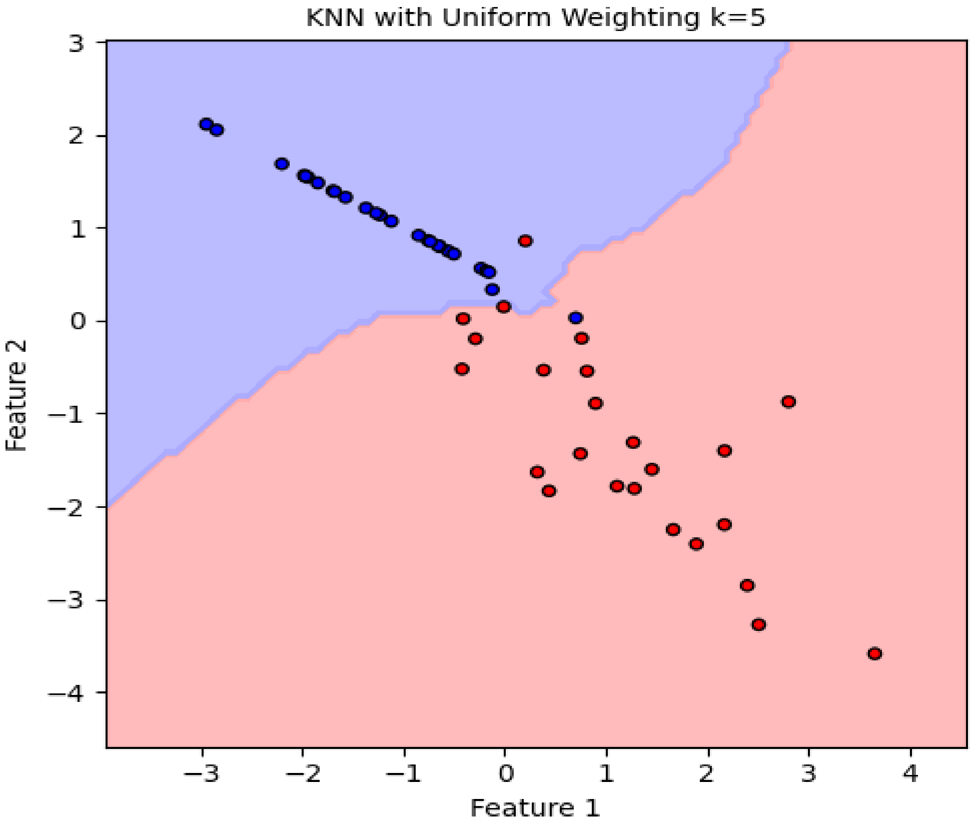


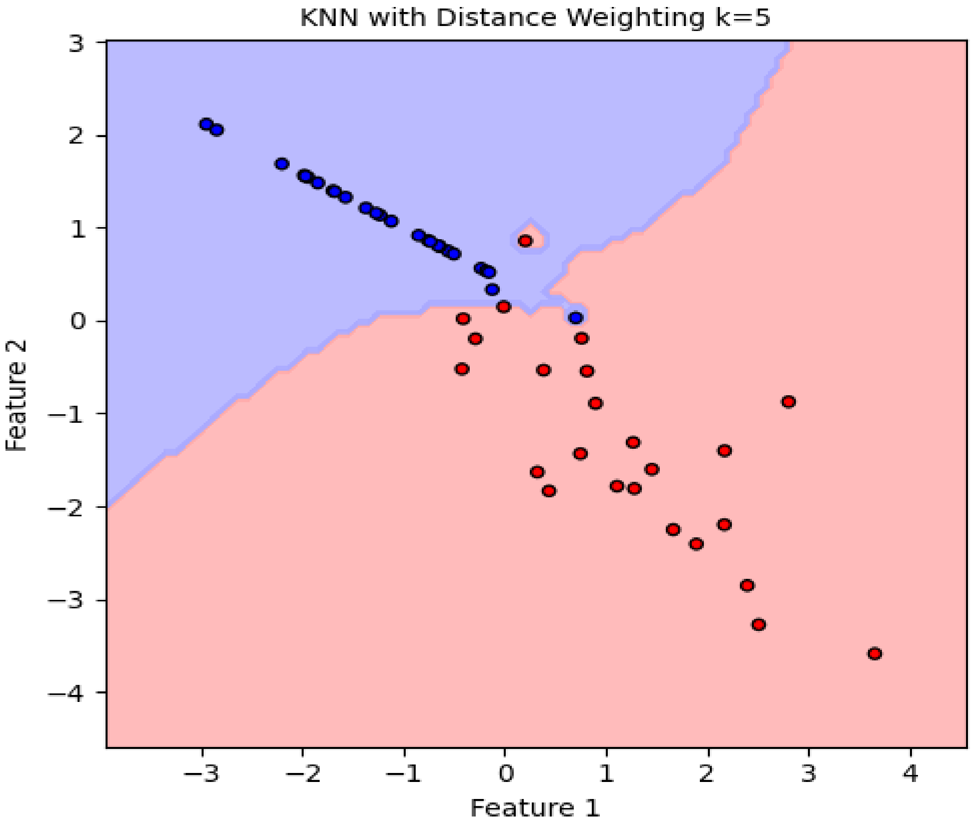
```
Classification Report for test data:
              precision    recall  f1-score   support

           0       1.00      0.93      0.96        42
           1       0.96      1.00      0.98        72

    accuracy                           0.97       114
   macro avg       0.98      0.96      0.97       114
weighted avg       0.97      0.97      0.97       114
```

K-Nearest Neighbors (KNN)

Distance Weighting in KNN

- Uniform weighting and distance weighting for a classification task
  - Distance weighting can lead to smoother and more accurate decision boundaries, especially in cases where the data points are not uniformly distributed.

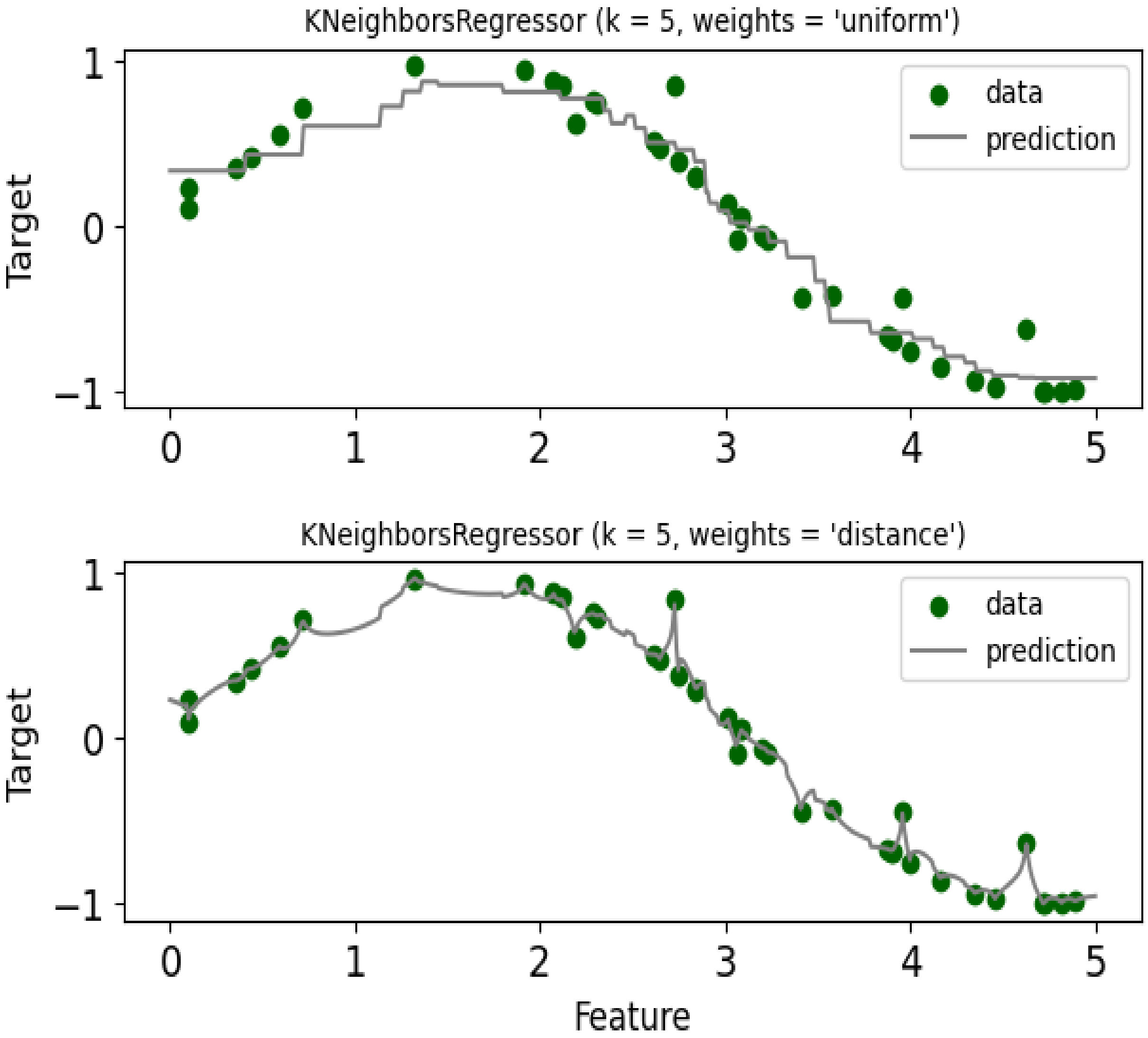




K-Nearest Neighbors (KNN)

Distance Weighting in KNN

- Uniform weighting and distance weighting for a regression task
- Distance weighting can provides better performance when closer neighbors are more relevant
- Helps in reducing the impact of distant neighbors, improving robustness to variations in data density



Decision Trees

Introduction

- A type of the supervised machine learning algorithms
- Model-Based Learning:
  - Constructs a model during the training phase by recursively splitting the dataset based on feature values to create a tree structure.
- Non-Parametric:
  - Does not make any assumptions about the distribution of the data. The complexity of the model can grow as the amount of data increases, making it flexible in capturing various data patterns
- Interpretability:
  - Provides clear and interpretable decision rules by visualizing the tree structure, making it easy to understand the decision-making process
- Versatile Application: Applicable to both classification and regression tasks

Decision Trees

Classification Trees: Impurity Measures

- One of the common impurity measures, which is used for classification decision trees is **Gini Impurity**
- Gini impurity measures how often a randomly chosen element of a set would be incorrectly labeled if it were labeled randomly and independently according to the distribution of labels in the set. It reaches its minimum (zero) when all cases in the node fall into a single target category
- For a dataset with K classes and relative frequencies p_i, $i \in$ {1,2,..,K}, the probability of choosing an item with label i is p_i, and the probability of not i is 1− p_i
- The Gini impurity GI is computed by summing pairwise products of these probabilities for each class label:

$$GI = \sum_{i=1}^{K} P_i(1 - P_i) = \sum_{i=1}^{K} P_i - \sum_{i=1}^{K} p_i^2 = 1 - \sum_{i=1}^{K} p_i^2$$

p_i: relative frequency of class i (the proportion of instances in class i)

- Example: Consider a dataset with two classes, A and B as shown:

$$GI = 1-(p_A^2 + p_B^2) = 1 - \left((\tfrac{2}{5})^2 + (\tfrac{3}{5})^2\right) = 1 - 0.52 = 0.48$$

Class
A ▲
B ☺
B ☺
B ☺
A ▲

Decision Trees

Classification Trees: Impurity Measures

- One of the common impurity measures in classification decision trees is **Gini Impurity**
- Gini impurity measures how often a randomly chosen element of a set would be incorrectly labeled if it were labeled randomly and independently according to the distribution of labels in the set. It reaches its minimum (zero) when all cases in the node fall into a single target category
- For a dataset with K classes and relative frequencies p_i, $i \in$ {1,2,.,K}, the probability of choosing an item with label i is p_i, and the probability of not i is 1− p_i
- The Gini impurity GI is computed by summing pairwise products of these probabilities for each class label:

$$GI = \sum_{i=1}^{K} P_i(1 - P_i) = \sum_{i=1}^{K} P_i - \sum_{i=1}^{K} p_i^2 = 1 - \sum_{i=1}^{K} p_i^2$$

p_i: relative frequency of class i (the proportion of instances in class i)

- Example: Consider a dataset with two classes, A and B as shown:

$$GI = 1 - (p_A^2 + p_B^2) = 1 - \left((\tfrac{3}{6})^2 + (\tfrac{3}{6})^2\right) = 1 - 0.5 = 0.5$$

- For K classes, maximum GI occurs when all classes are equally likely: $p_i = \frac{1}{K} \Rightarrow \mathrm{GI}_{\max} = 1 - \frac{1}{K}$
- If $K = 2$, then $\mathrm{GI}_{\max} = 0.5$

Class
A ▲
B ☺
B ☺
B ☺
A ▲
A ▲

Decision Trees

Classification Trees: Impurity Measures

- One of the common impurity measures in classification decision trees is **Gini Impurity**
- Gini impurity measures how often a randomly chosen element of a set would be incorrectly labeled if it were labeled randomly and independently according to the distribution of labels in the set. It reaches its minimum (zero) when all cases in the node fall into a single target category
- For a dataset with K classes and relative frequencies p_i, $i \in \{1,2,.,K\}$, the probability of choosing an item with label i is p_i, and the probability of not i is 1− p_i
- The Gini impurity GI is computed by summing pairwise products of these probabilities for each class label:

$$GI = \sum_{i=1}^{K} P_i(1 - P_i) = \sum_{i=1}^{K} P_i - \sum_{i=1}^{K} p_i^2 = 1 - \sum_{i=1}^{K} p_i^2$$

p_i: relative frequency of class i (the proportion of instances in class i)

- Example: Consider a dataset with two classes, A and B as shown:

$$GI = 1-(p_A^2 + p_B^2) = 1 - \left((\tfrac{0}{6})^2 + (\tfrac{6}{6})^2\right) = 1 - 1 = 0.0$$

- All of the dataset points has the same class B. It is pure B. Therefore, the impurity is zero.
- Gini impurity measure ranges from 0 (pure node) to 0.5 (maximum impurity for two classes).

Class
B ☺
B ☺
B ☺
B ☺
B ☺
B ☺

Decision Trees

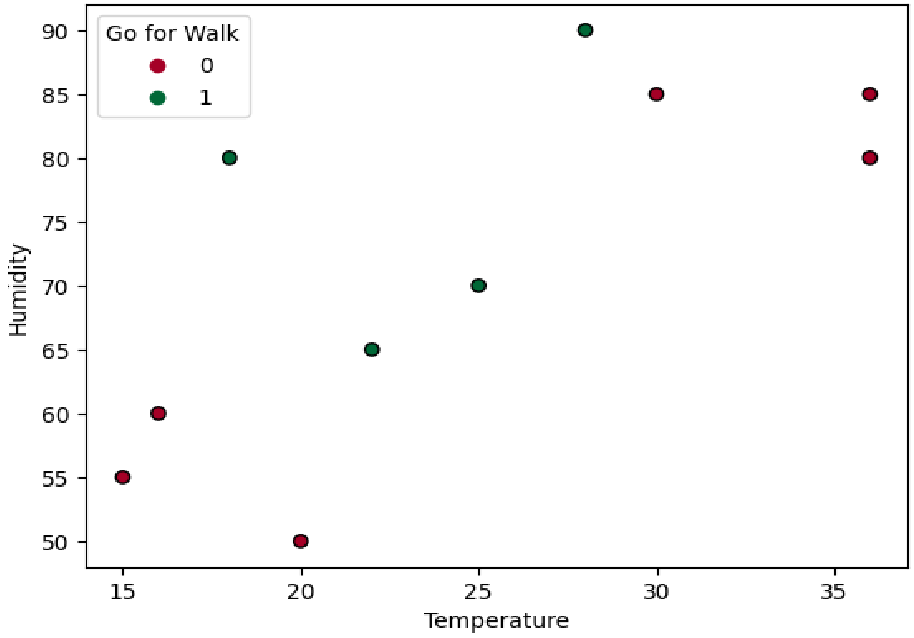
Classification Tree

- Assume having a dataset with two features: Temperature and Humidity.
- The target variable is Go_for_walk, which indicates whether the conditions are favorable for a walk or not (1 or 0)
- Samples: 10 (total number of instances in the dataset)
- Distribution of classes: 6 instances of class 0 and 4 instances of class 1)
- Class 0 is the majority class
- $Gini$ impurity:

$$GI = 1 - (p_1^2 + p_0^2) = 1 - \left((\frac{4}{10})^2 + (\frac{6}{10})^2 \right)$$

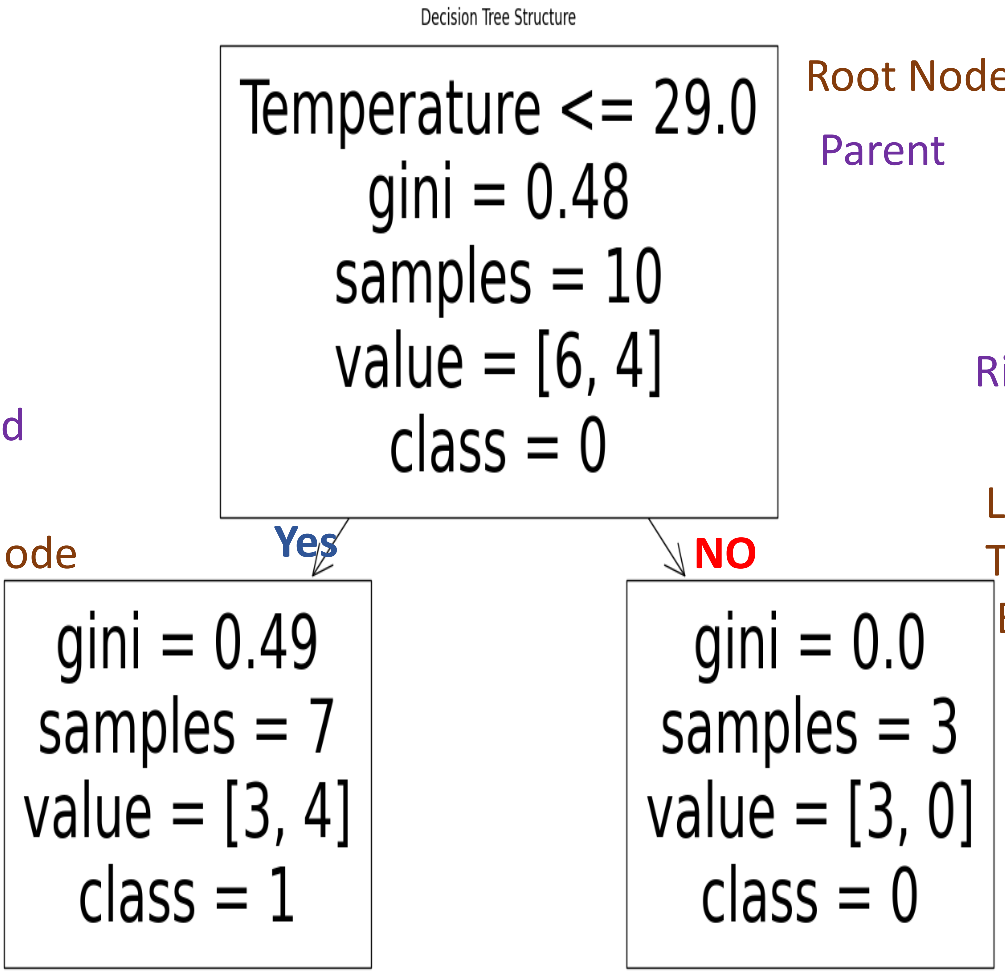
$$= 1 - 0.52 = 0.48$$

Temperature	Humidity	Go_for_walk
22	65	1
25	70	1
15	55	0
18	80	1
30	85	0
16	60	0
28	90	1
20	50	0
36	80	0
36	85	0



Decision Trees

Classification Tree



Decision Tree Structure

Temperature <= 29.0
gini = 0.48
samples = 10
value = [6, 4]
class = 0

Yes

NO

gini = 0.49
samples = 7
value = [3, 4]
class = 1

gini = 0.0
samples = 3
value = [3, 0]
class = 0

Root Node

Parent

Right Child

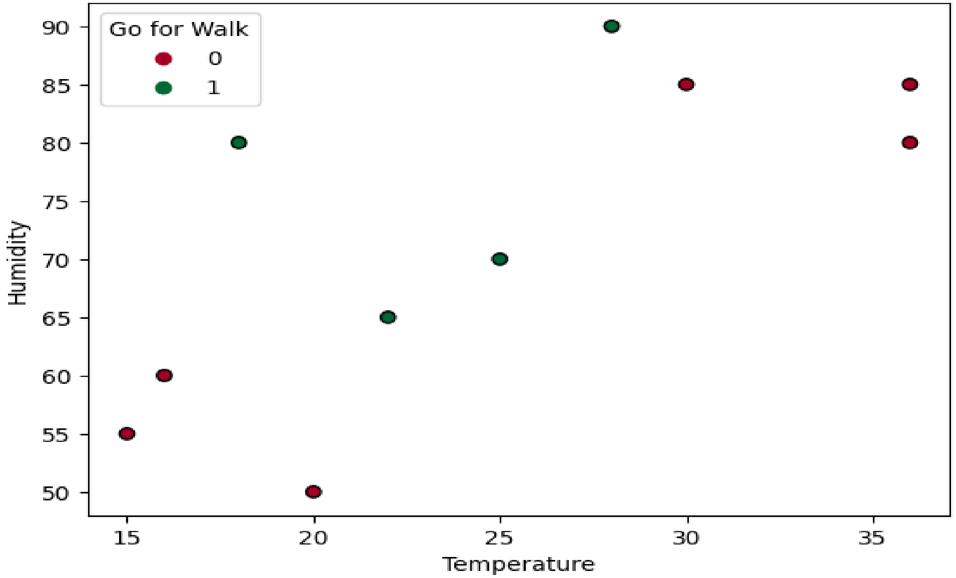
Left Child

Internal Node

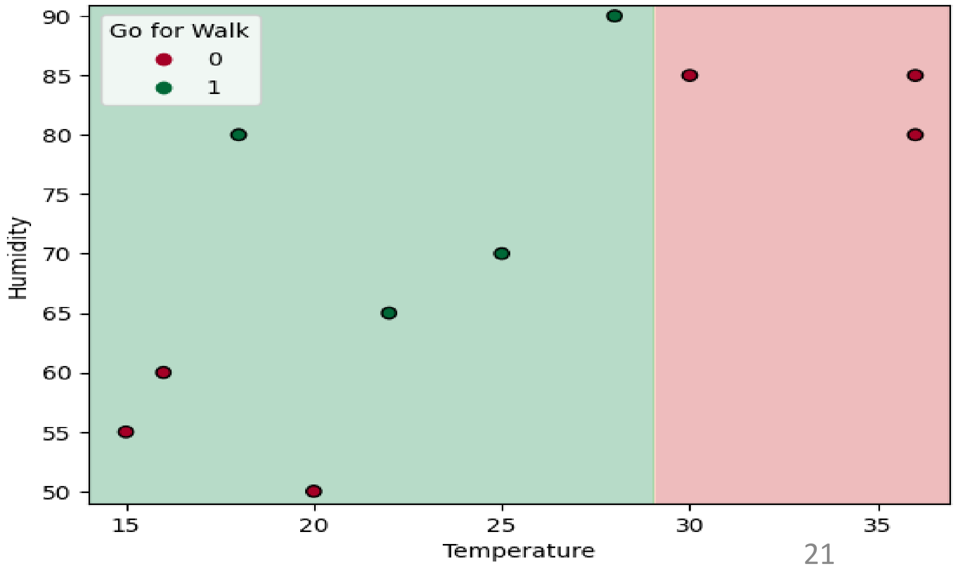
Leaf Node

Terminal Node

End Node



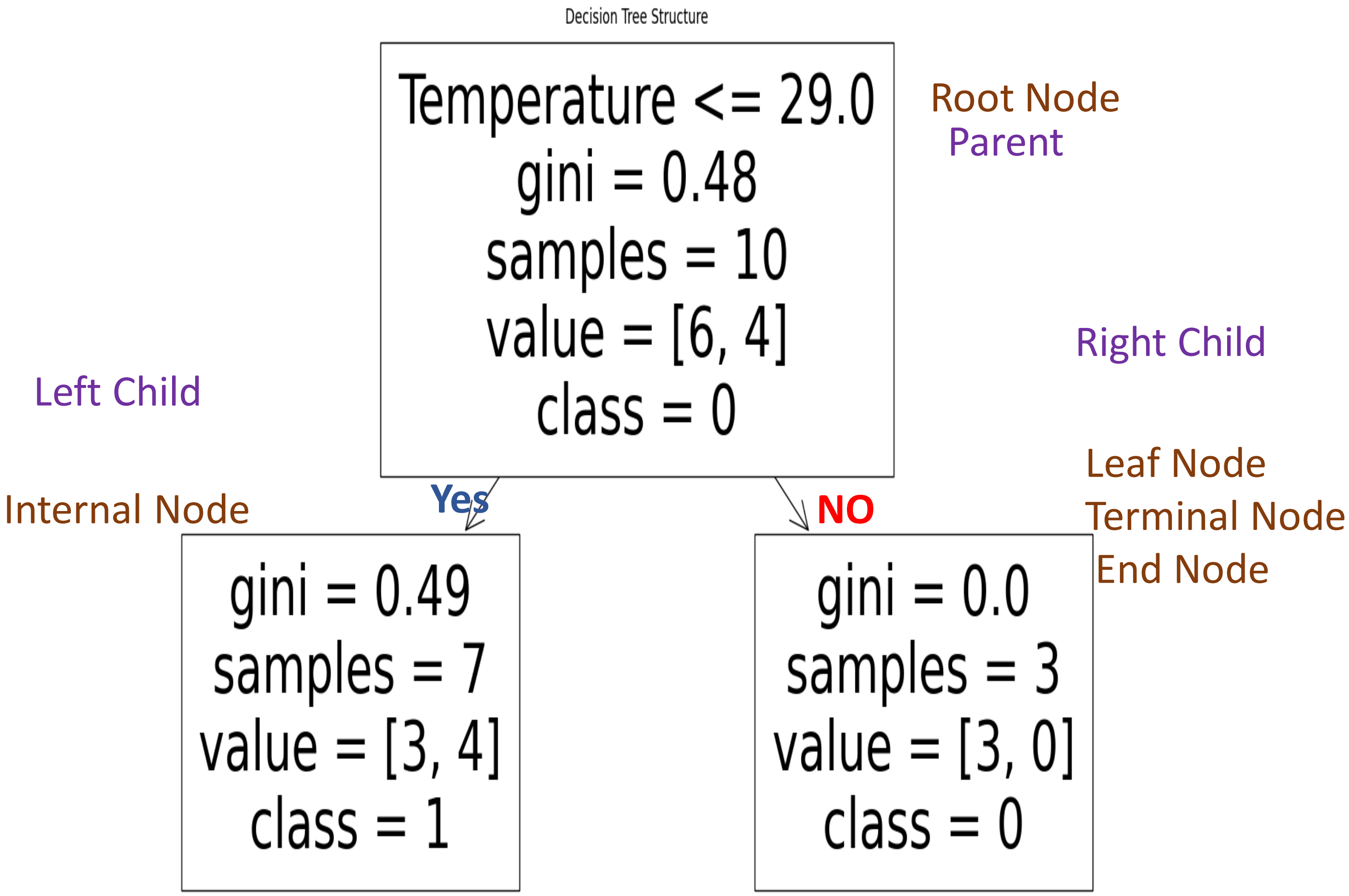
Decision Surface of the Decision Tree



Decision Trees

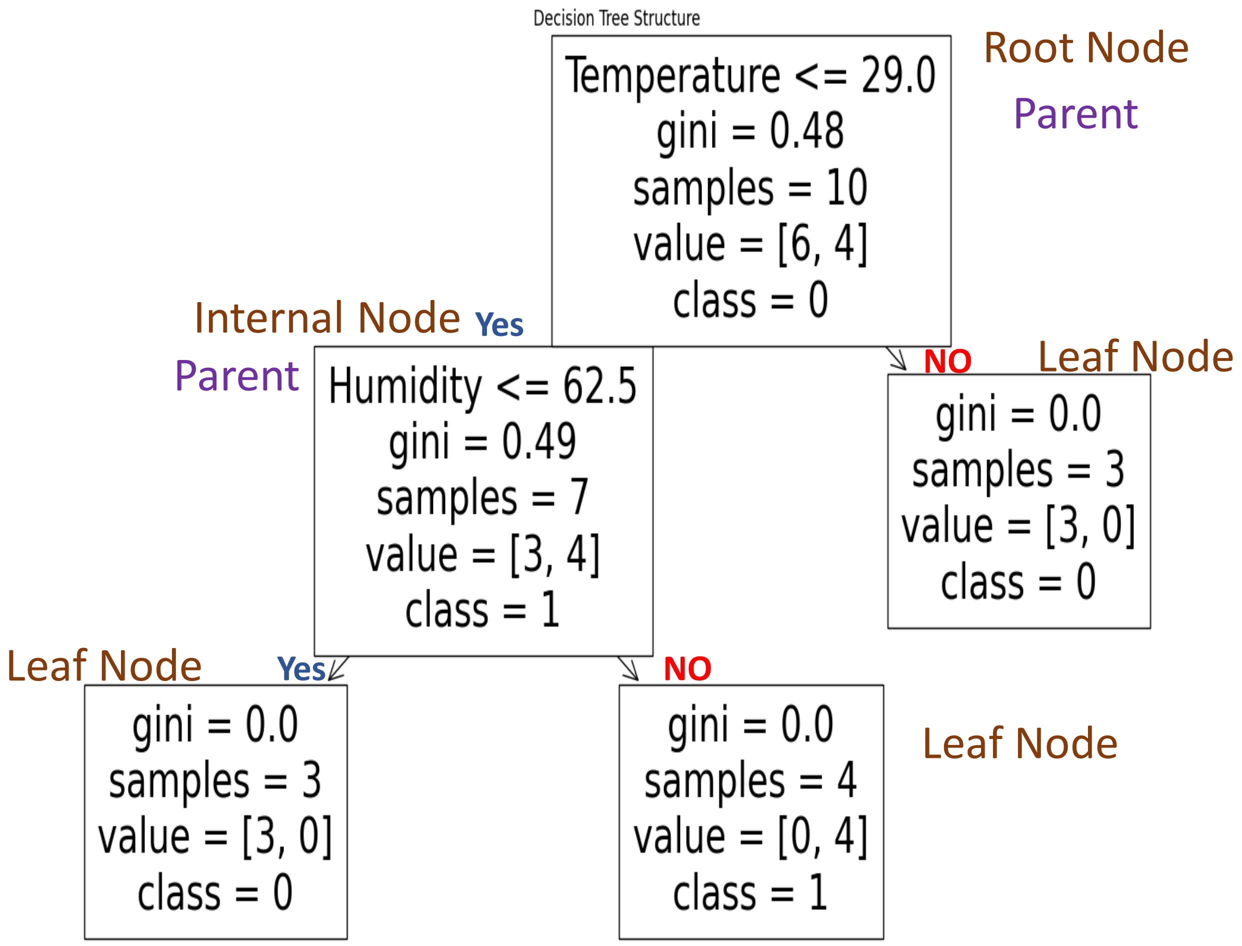
Classification Tree

- Root Node: The top node that represents the entire dataset and provides the first decision rule.
- Internal Nodes: Intermediate nodes that apply decision rules to further split the data into subsets.
- Leaf Nodes: Terminal nodes that provide the final classification or prediction for the subsets of data.



Decision Trees

Classification Tree



Decision Tree Structure

Temperature <= 29.0
gini = 0.48
samples = 10
value = [6, 4]
class = 0

Root Node

Parent

Internal Node

Yes

Parent

Humidity <= 62.5
gini = 0.49
samples = 7
value = [3, 4]
class = 1

NO

Leaf Node

gini = 0.0
samples = 3
value = [3, 0]
class = 0

Leaf Node

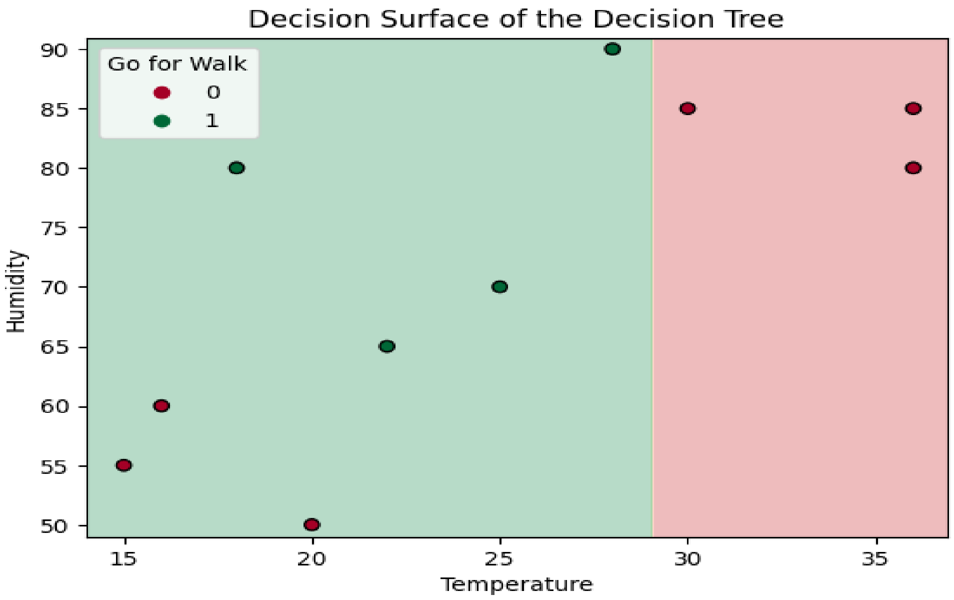
Yes

gini = 0.0
samples = 3
value = [3, 0]
class = 0

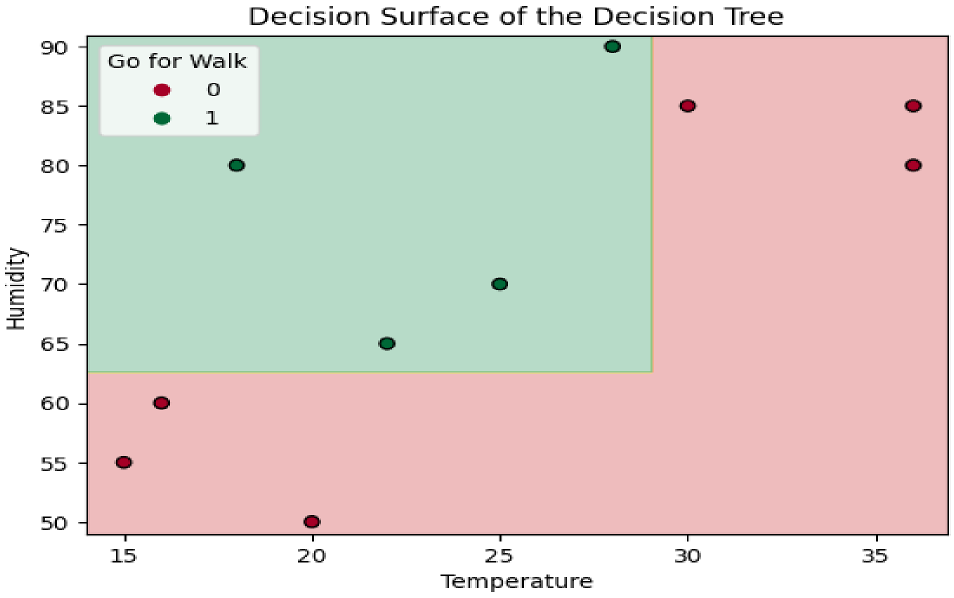
NO

gini = 0.0
samples = 4
value = [0, 4]
class = 1

Leaf Node



Decision Surface of the Decision Tree

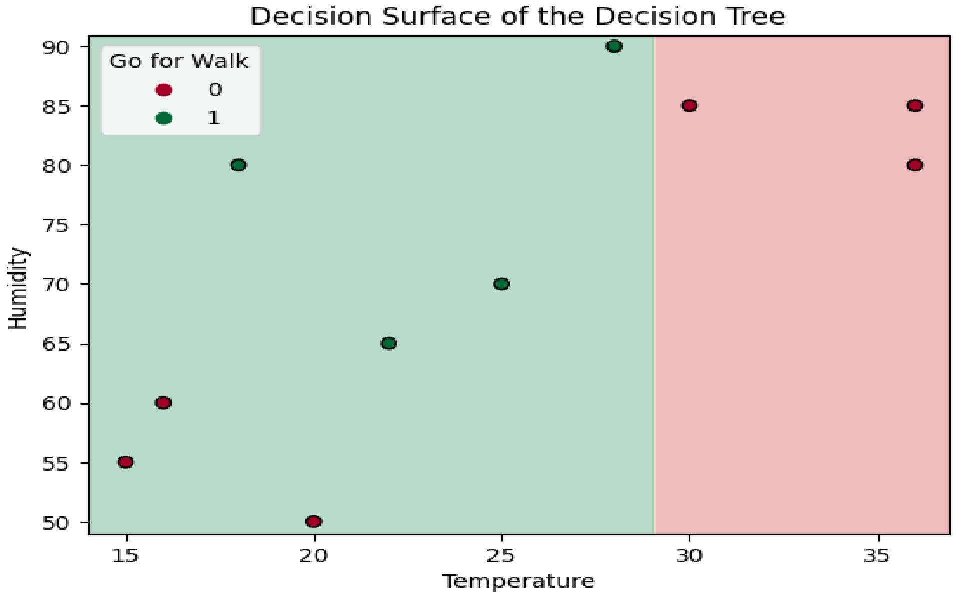


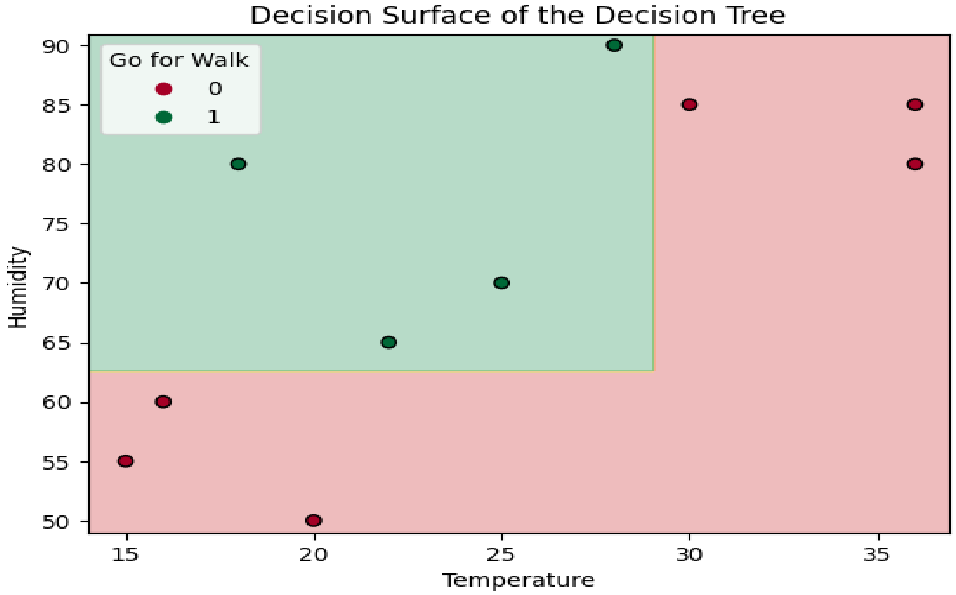
Decision Surface of the Decision Tree

Decision Trees

Classification Tree

- The images illustrate how a decision tree uses features to split the data into different regions, aiming to achieve pure groups
- The decision boundaries shown in the plots correspond to the decision rules defined in the tree structure
- The boundary shows that Temperature <= 29.0 separates the two main regions.
- Within the left region, Humidity <= 62.5 further divides the space into two subregions.





Decision Trees: Classification Tree

Split Impurity Reduction – Node/Split Importance

+ For each split, the reduction in impurity for that split can be computed as follows:

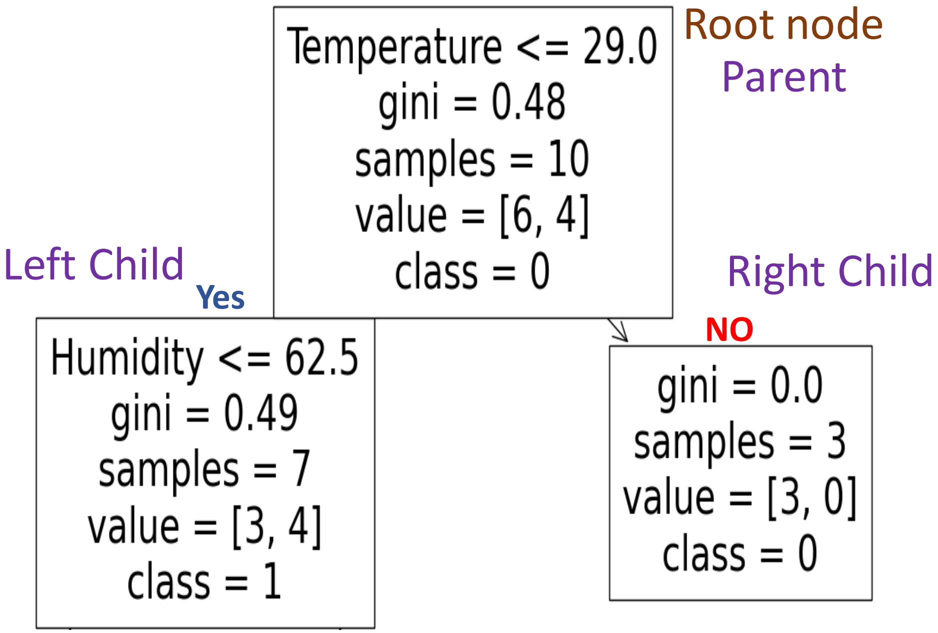
$$\Delta GI(s) = \mathrm{W}_{\mathrm{parent}}.\,\mathrm{GI}_{\mathrm{Parent}} - \left(\mathrm{W}_{\mathrm{Left}}.\,\mathrm{IG}_{\mathrm{Left}} + \mathrm{W}_{\mathrm{Right}}.\,\mathrm{IG}_{\mathrm{Right}}\right)$$

  - $\Delta GI(s)$ is the impurity reduction at split s.
  - $\mathrm{W}_{\mathrm{parent}}$ is the fraction (weight) of samples in the parent node
  - $\mathrm{W}_{\mathrm{Left}}$ is the fraction of samples in the left child node
  - $\mathrm{W}_{\mathrm{Right}}$ is the fraction of samples in the right child node

+ Example: impurity reduction of the shown split starting with the root node

  - $\Delta GI(s) = \frac{10}{N}.0.48 - (\frac{7}{N}.0.49 + \frac{3}{N}.0.0)$
  - N= 10, it is the total number of the samples of the dataset
  - $\Delta GI(s) = 0.137$

+ Split impurity reduction is an indication for the node importance



Decision Trees: Classification Tree

Split Impurity Reduction — Node/ Split Importance

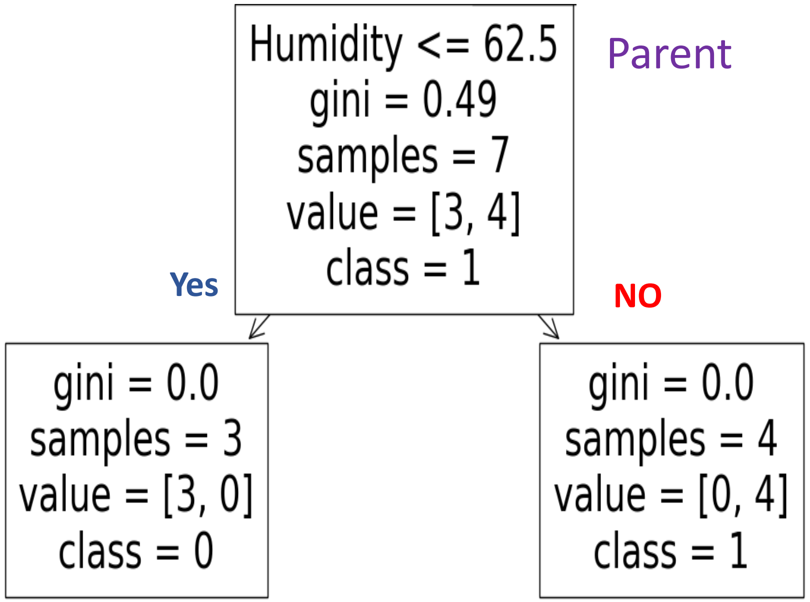
- For each split, the reduction in impurity for that split can be computed as follows:

  $$\Delta GI(s) = \mathrm{W}_{\mathrm{parent}}.\,\mathrm{GI}_{\mathrm{Parent}} - \left(\mathrm{W}_{\mathrm{Left}}.\,\mathrm{IG}_{\mathrm{Left}} + \mathrm{W}_{\mathrm{Right}}.\,\mathrm{IG}_{\mathrm{Right}}\right)$$

  - $\Delta GI(s)$ is the impurity reduction at split s.
  - $\mathrm{W}_{\mathrm{parent}}$ is the fraction (weight) of samples in the parent node
  - $\mathrm{W}_{\mathrm{Left}}$ is the fraction of samples in the left child node
  - $\mathrm{W}_{\mathrm{Right}}$ is the fraction of samples in the right child node

- Example: impurity reduction of the shown split

  - $\Delta GI(s) = \frac{7}{10}.\,0.49 - \left(\frac{3}{10}.\,0.0 + \frac{4}{10}.\,0.0\right) = 0.343$



Decision Trees: Classification Tree

Split Impurity Reduction — Node/ Split Importance

+ For each split, the decrease in impurity for that split can be computed as follows:

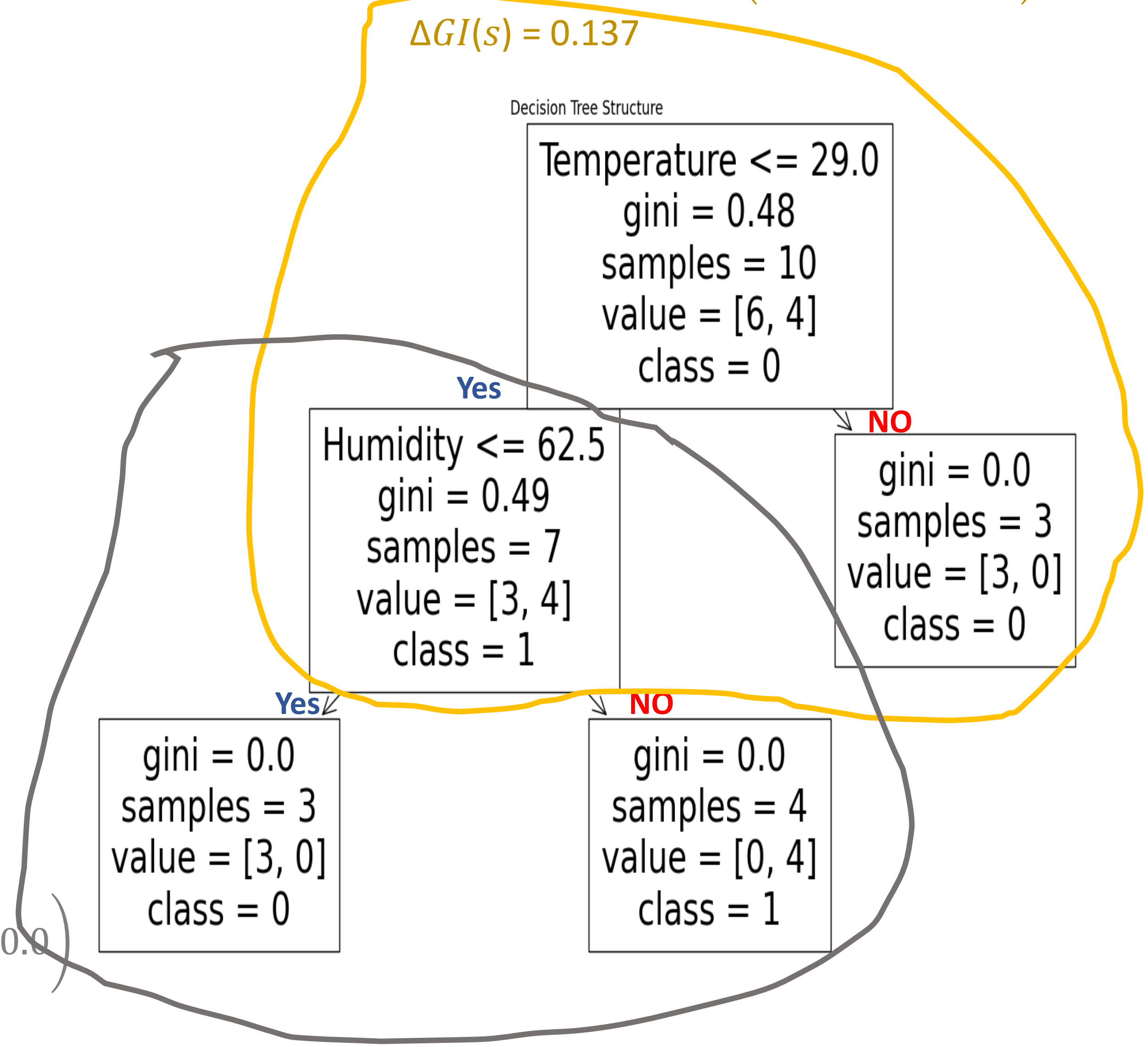
$$\Delta GI(s) = \mathrm{W_{parent}}.\,\mathrm{GI_{Parent}} - \left(\mathrm{W_{Left}}.\,\mathrm{IG_{Left}} + \mathrm{W_{Right}}.\,\mathrm{IG_{Right}}\right)$$

- $\Delta GI(s)$ is the impurity reduction at split s.
- $\mathrm{W_{parent}}$ is the fraction (weight) of samples in the parent node
- $\mathrm{W_{Left}}$ is the fraction of samples in the left child node
- $\mathrm{W_{Right}}$ is the fraction of samples in the right child node

+ Which node has more importance?

$$\Delta GI(s) = \frac{10}{10}.\,0.48 - \left(\frac{7}{10}.\,0.49 + \frac{3}{10}.\,0.0\right)$$

$\Delta GI(s) = 0.137$



Decision Tree Structure

Temperature <= 29.0
gini = 0.48
samples = 10
value = [6, 4]
class = 0

Yes

Humidity <= 62.5
gini = 0.49
samples = 7
value = [3, 4]
class = 1

NO

gini = 0.0
samples = 3
value = [3, 0]
class = 0

Yes

gini = 0.0
samples = 3
value = [3, 0]
class = 0

NO

gini = 0.0
samples = 4
value = [0, 4]
class = 1

$$\Delta GI(s) = \frac{7}{10}.\,0.49 - \left(\frac{3}{10}.\,0.0 + \frac{4}{10}.\,0.0\right)$$

$\Delta GI(s) = 0.343$

Decision Trees: Classification Tree

Feature Importance

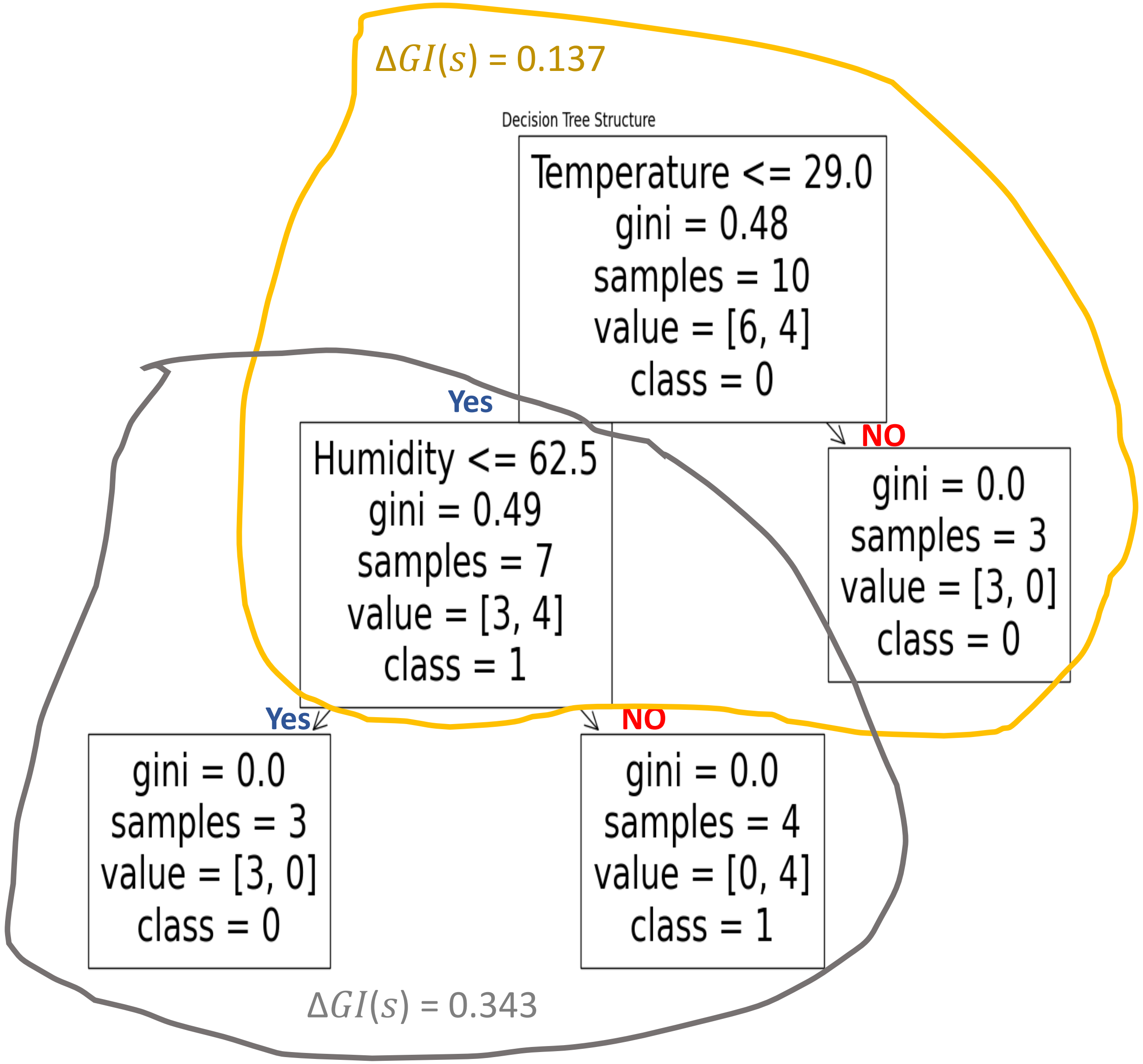
+ The aggregate importance of a feature x_m is the sum of the impurity reductions over all splits where this feature contributes to the decision. It can be mathematically expressed as:

$$AI(x_m) = \sum_{s \in S(x_m)} \Delta GI(s)$$

- Where:
- $AI(x_m)$: The aggregate importance of feature x_m.
- $S(x_m)$: The set of all splits where feature x_m is used.
- $\Delta GI(s)$: The Gini impurity reduction at split s.

+ To ensure that importance values for all features sum to 1, the aggregated importance is normalized to represent the feature importance value as follows:

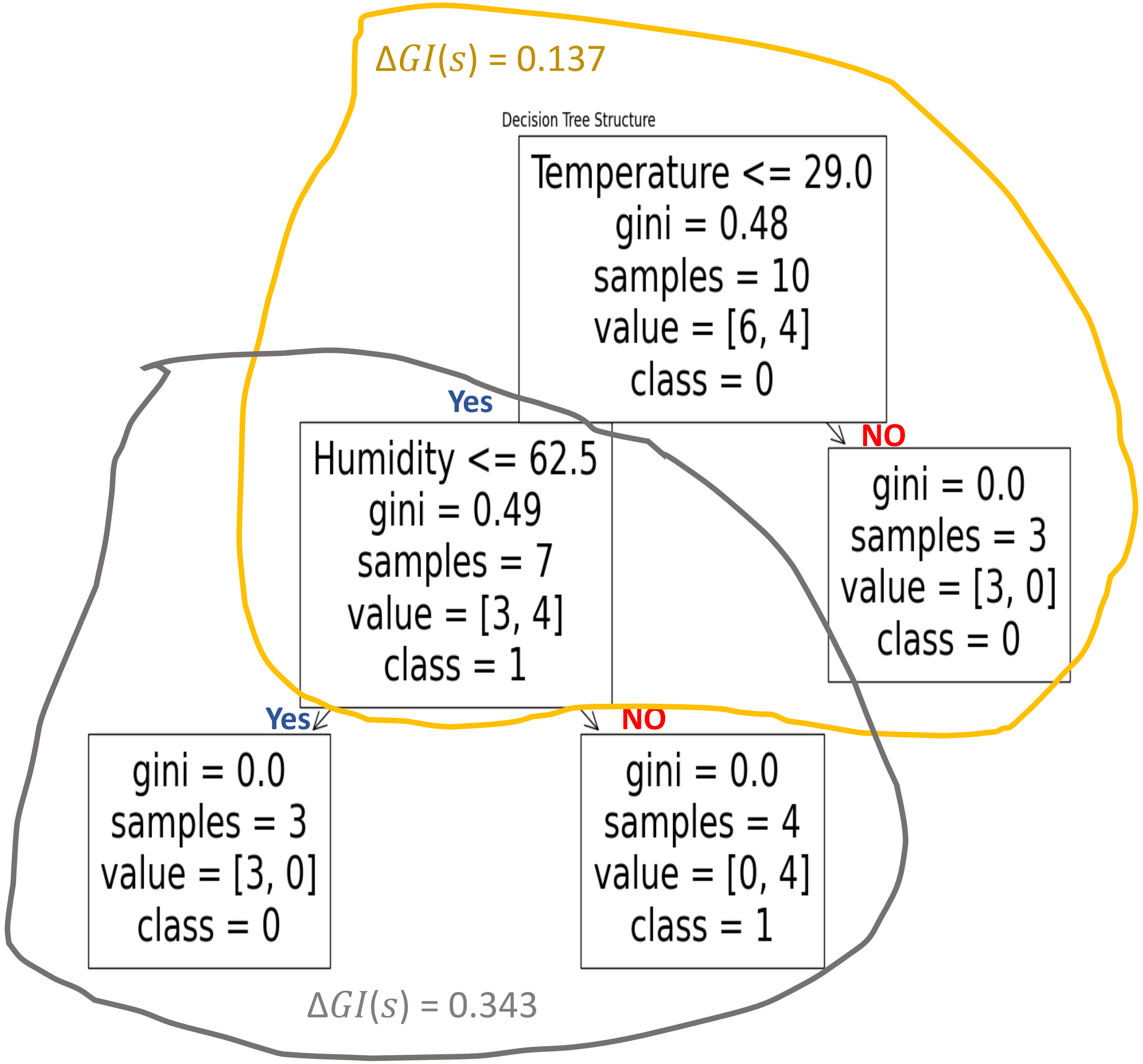
$$Importance(x_m) = \frac{AI(x_m)}{\sum_{i=1}^{M} AI(x_i)}$$



Decision Trees: Classification Tree

Feature Importance

+ The importance of the two features of the Go_to_walk example:
  - $Importance(Temperature) = \frac{0.137}{0.137+0.343} = 0.2857$
  - $Importance(Humidity) = \frac{0.343}{0.137+0.343} = 0.7143$

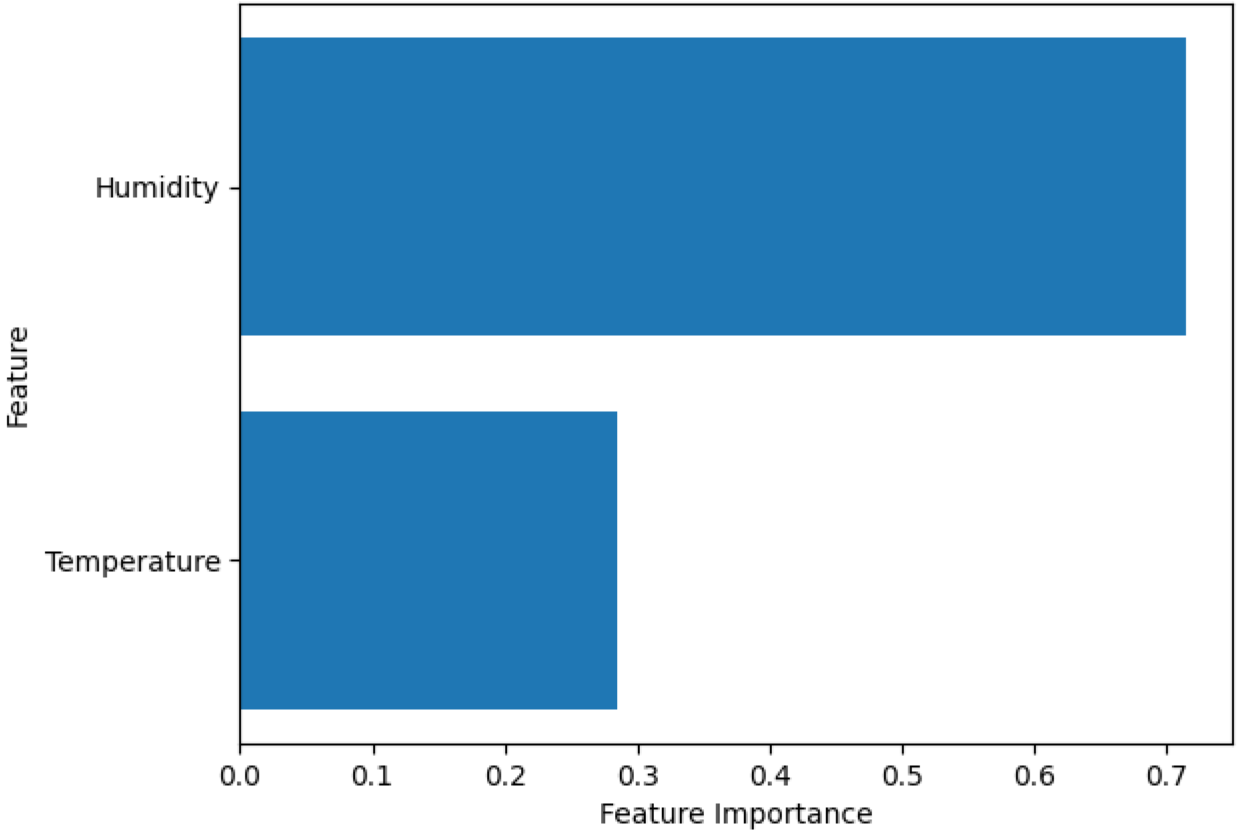


Decision Trees: Classification Tree

Feature Importance

- The importance of the two features of the Go_to_walk example:
  - $Importance(Temperature) = \frac{0.137}{0.137+0.343} = 0.2857$
  - $Importance(Humidity) = \frac{0.343}{0.137+0.343} = 0.7143$
- Humidity has the highest importance, indicating that it plays the most significant role in the decision-making process of the tree.
- **Temperature** has a lower importance, showing it contributes to the model's decisions but is less impactful compared to Humidity.
- The normalized values suggest that both features together account for 100% of the decision-making process.

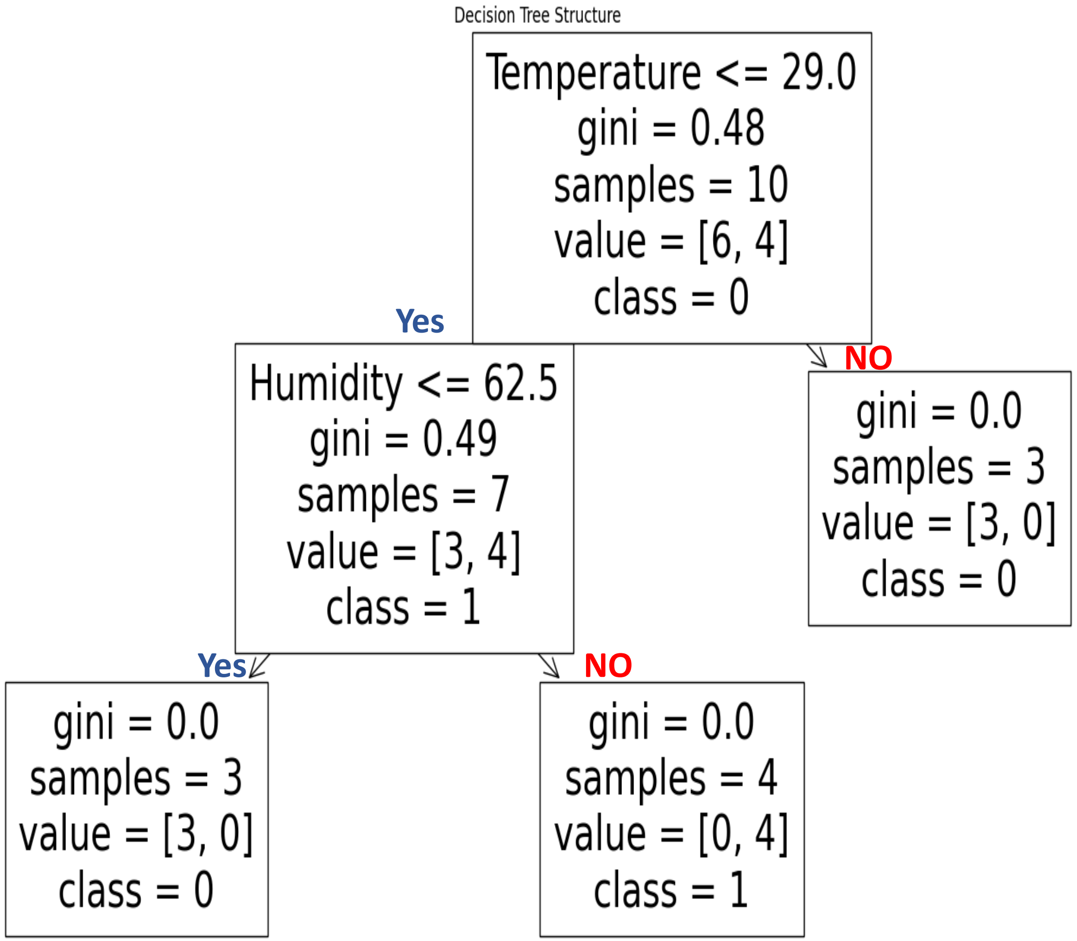
Feature Importance Using scikit_learn



Decision Trees

Classification Tree—Check for Stopping Criteria

- If the selected split does not reduce impurity by more than a predefined threshold (minimum impurity decrease), stop splitting
- If the node contains fewer samples than the minimum required for splitting, stop splitting
- If no further features are available for splitting, stop splitting.
- If each node contains samples from only one class, the nodes are pure and becomes leaf nodes, stop splitting



Decision Trees

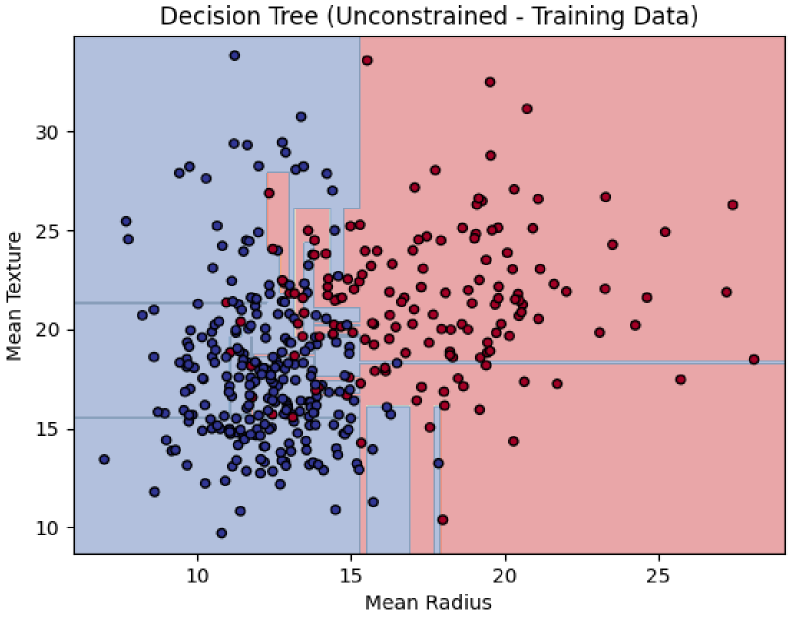
Overfitting And Pruning

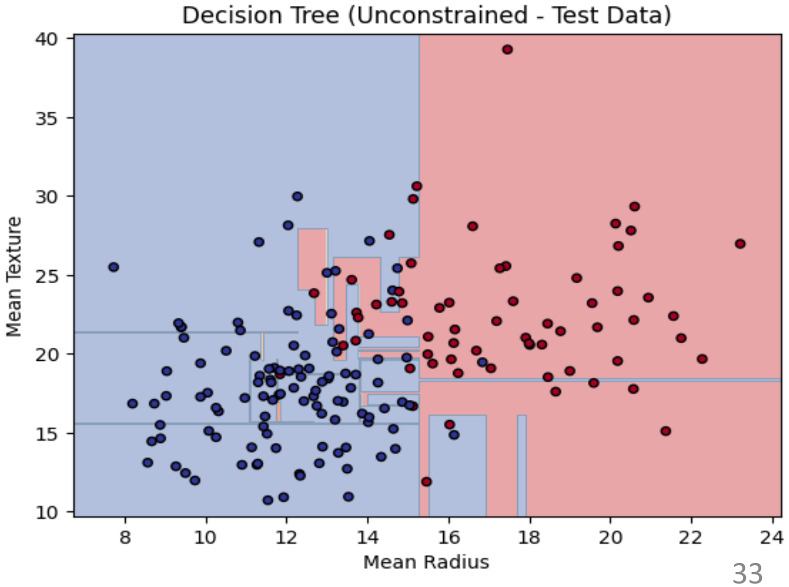
- In decision trees, overfitting occurs when the tree grows too deep, creating many nodes and branches to perfectly fit the training data
- This can lead to a very intricate tree structure that captures noise rather than the true patterns.
- Pruning is a technique used to reduce the complexity of a decision tree by removing parts of the tree
- Types of Pruning:
  - Pre-pruning (Early Stopping): Limits are set on the tree's growth, such as
  - maximum depth,
  - minimum number of samples required to split a node, or
  - minimum impurity decrease required to make a split to prevent the tree from growing too complex by stopping its growth early based on predefined.
  - Post-pruning (Pruning after Tree Construction): The tree is first grown to its maximum size. Then, branches that have little importance are removed. Common techniques include reduced error pruning, cost complexity pruning, or minimum error pruning.

Decision Trees

Overfitting And Pruning: Pre-pruning —*Impact of maximum depth*

- Maximum depth of the decision tree as a hyper parameter can be tuned using grid search optimization method by predefining the range to be examined
- Example: Classification decision tree for the breast cancer dataset and use only the first two features and without constrains
  - Training accuracy: 1.0000
  - Test accuracy: 0.8592
- The unconstrained decision tree typically achieves very high or even perfect accuracy on the training data
- Despite the high training accuracy, the test accuracy is often significantly lower. This indicates overfitting, where the model captures noise and specific patterns in the training data that do not generalize well to new, unseen data.

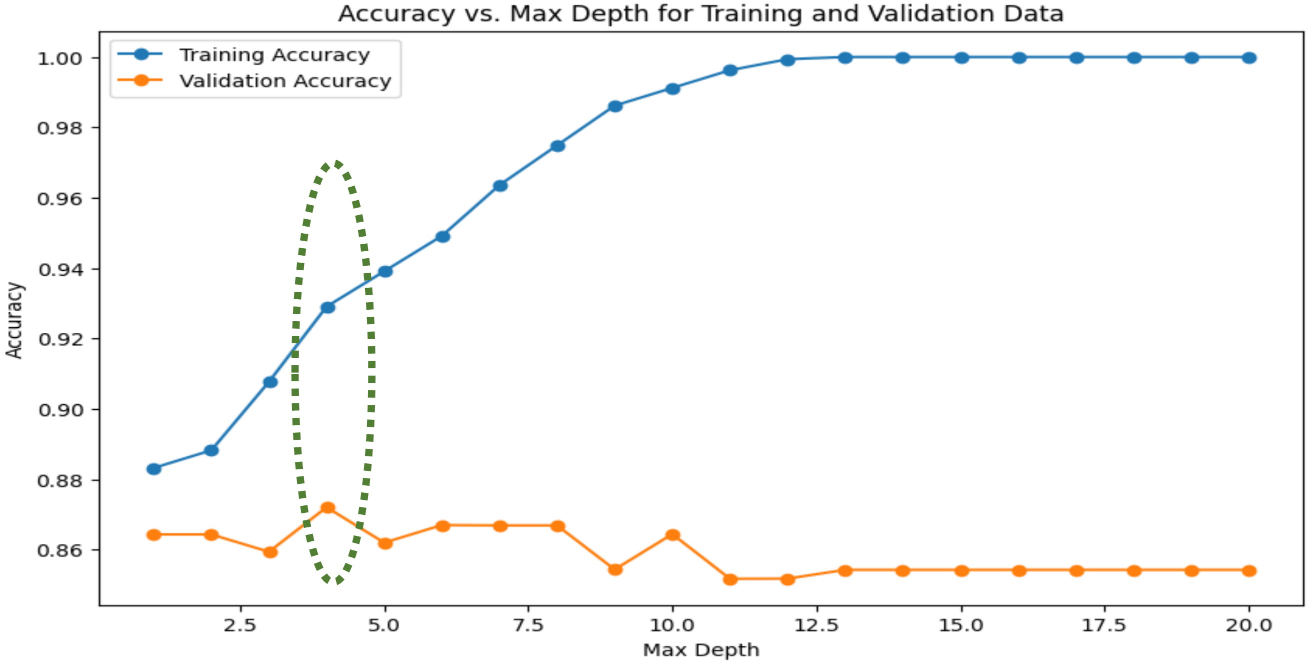




Decision Trees

Overfitting And Pruning: Pre-pruning $-Impact\ of\ maximum\ depth$

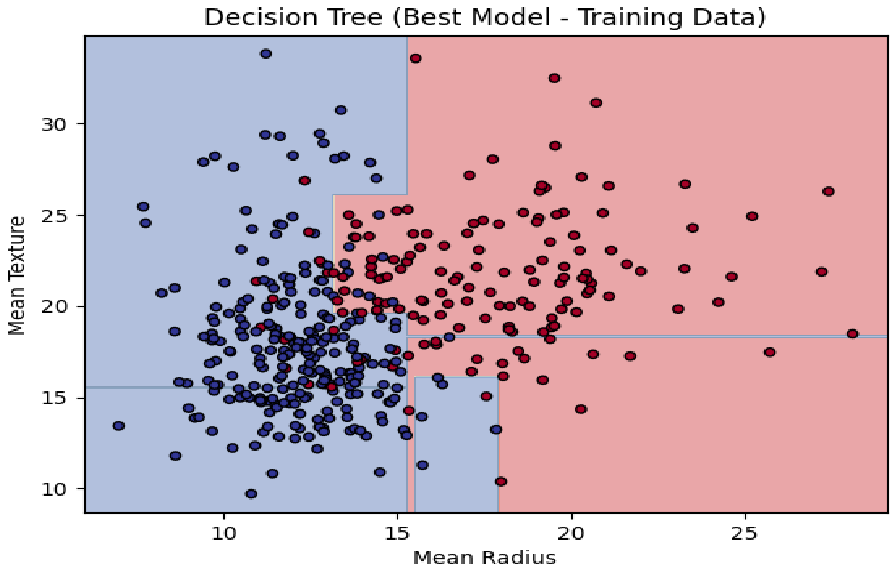
- The plot shows the training and validation accuracies from the grid search against maximum depth
- Best max_depth: 4
- Best validation accuracy: 0.8720

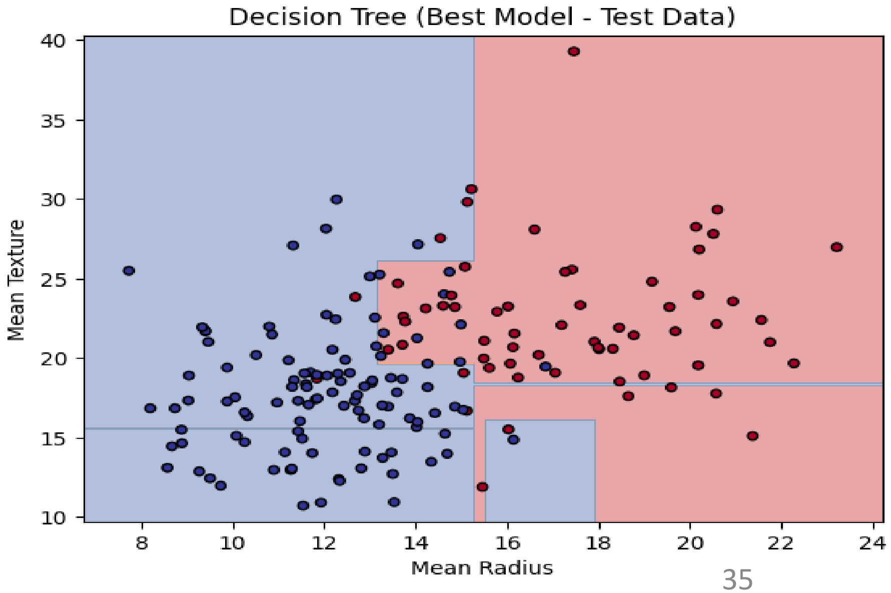


Decision Trees

Overfitting And Pruning: Pre-pruning $-Impact\ of\ maximum\ depth$

- The best estimator has
  - Training accuracy: 0.9296
  - Test accuracy: 0.8947
- Training Accuracy: As maximum depth is tuned and typically constrained to a lower value, the training accuracy might decrease slightly because the tree is not allowed to perfectly fit the training data
- Test Accuracy: The test accuracy is improved compared to the unconstrained tree because the model is better able to generalize.
- The optimal maximum depth balances the complexity of the model, reducing overfitting while still capturing the important patterns in the data.





Hochschule Karlsruhe
University of
Applied Sciences

Fakultät für
Elektro- und
Informationstechnik

www.h-ka.de