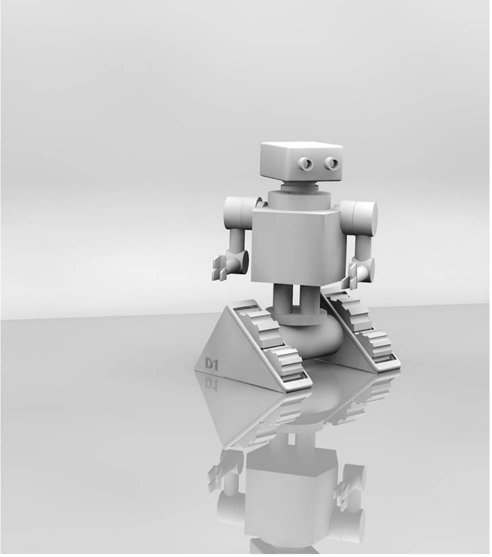


CPSC 481
Artificial Intelligence

Dr. Mira Kim
Mira.kim@fullerton.edu

What we will cover this week

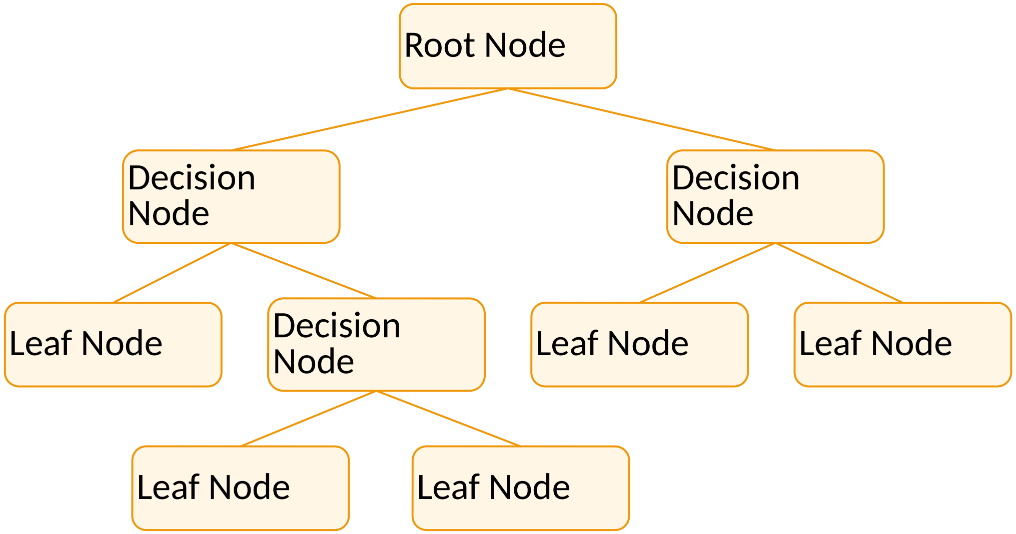
- Decision Trees
- Case Study on Decision Trees

Overview of Decision Tree

- Goal of Decision Tree
  - To make a decision/prediction based on input by splitting the data into subsets based on the value of input features
- Input
  - New observation/data point
- Output
  - Class labels or values
- Usage of Decision Tree
  - Supervised learning such as Classification and regression
  - Capturing non-linear relationships

Elements of Decision Tree

- Root Node
  - Node dividing the whole dataset into two or more sets
- Decision Node
  - Node splitting an input dataset based on some criteria
- Leaf Node (Terminal Node)
  - Class, Result of Classification

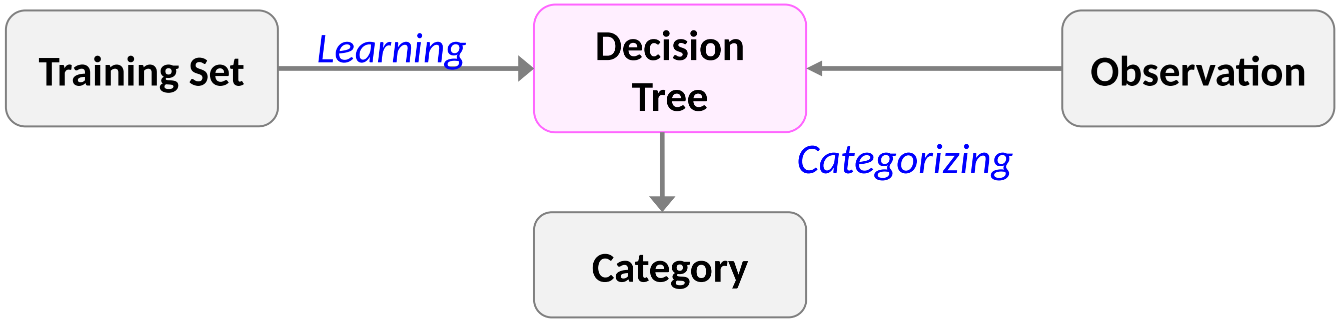


Process of Decision Tree

1. **Model Training**: Construct Decision Tree with a training dataset, where it learns to make decisions based on the features and the target outcome
2. **New Data Point**: After the model has been trained, you can input a new observation
3. **Prediction**: Decision tree uses the values of the features in the new data point to navigate through the tree until it reaches a leaf node
4. **Output**: The prediction for the new data point, which could be a class label in classification or a numerical value in regression

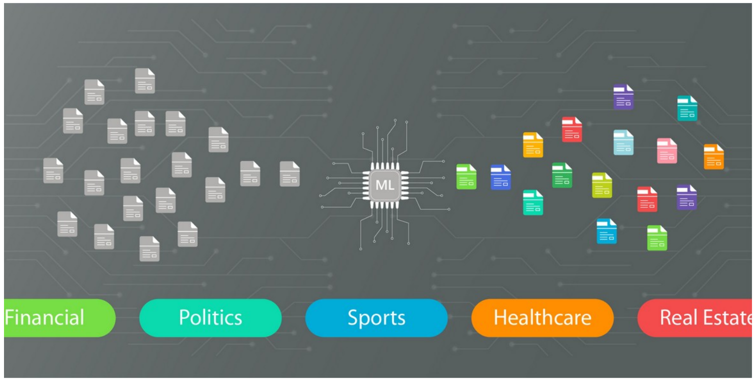
Building and Using Decision Tree

- Learning Stage
  – Building a Decision Tree with Training Set
- Categorizing Stage
  – Classifying observation with Decision Tree

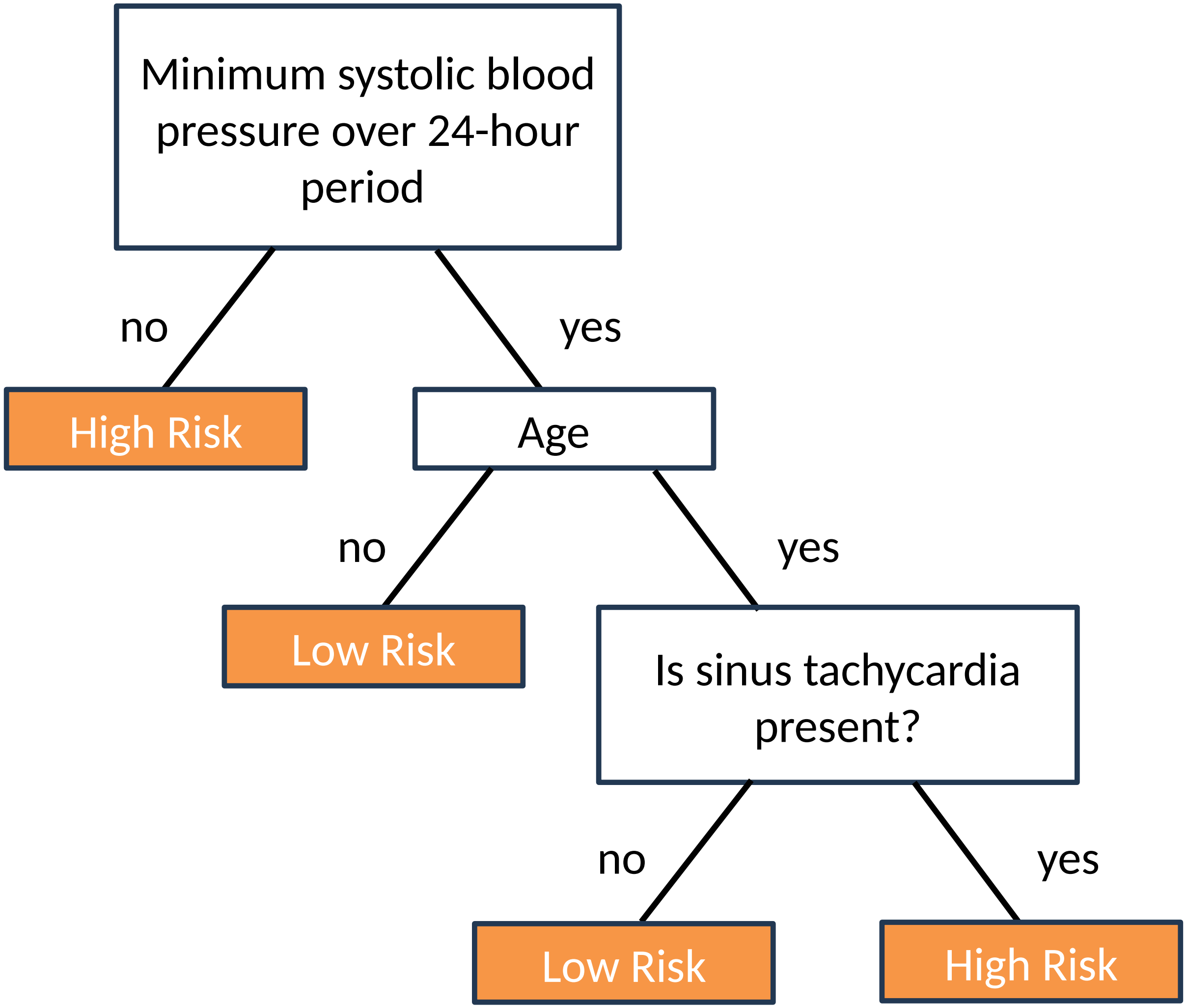


Examples of Decision Tree

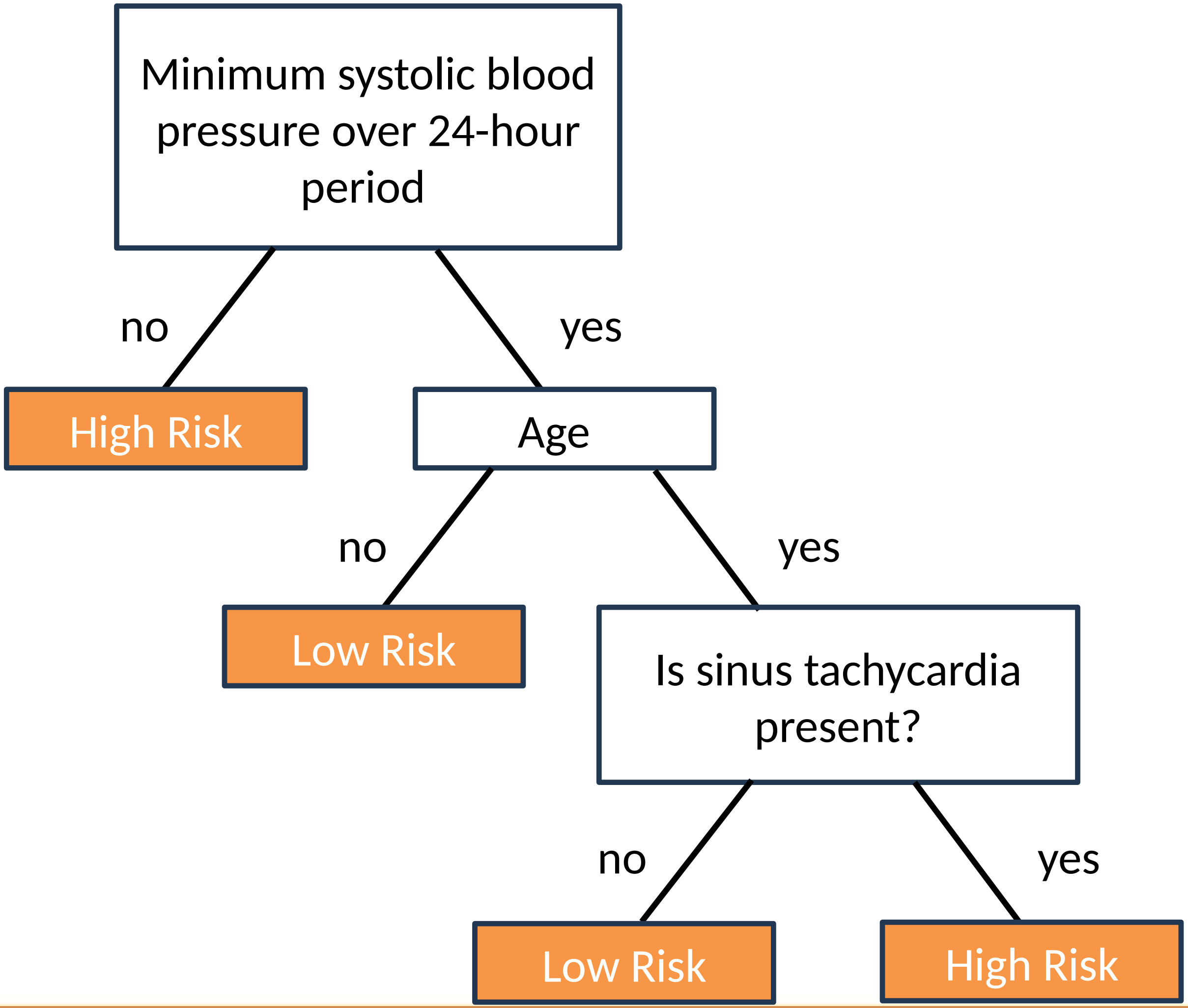
- Medical Domain
  - Predicting tumor cells as Benign or Malignant
- Finance Domain
  - Classifying credit card transactions as *Legitimate* or *Fraudulent*
- Media Domain
  - Categorizing news stories as finance, weather, entertainment, sports, etc.
- Chemical Domain
  - Classifying chemical compounds as hazardous or non-hazardous



Example 1) Determining risk for a patient

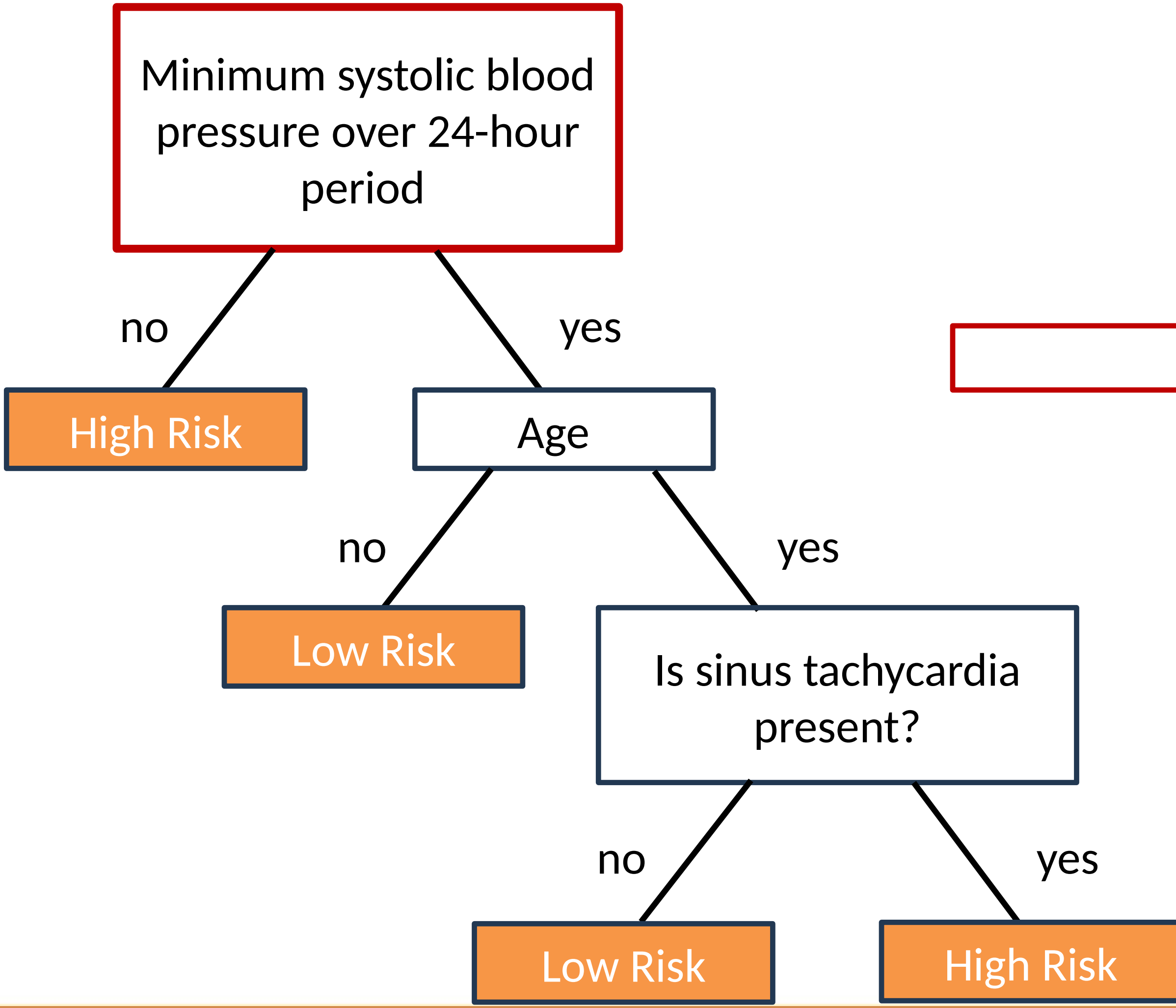


Example 1) Determining risk for a patient



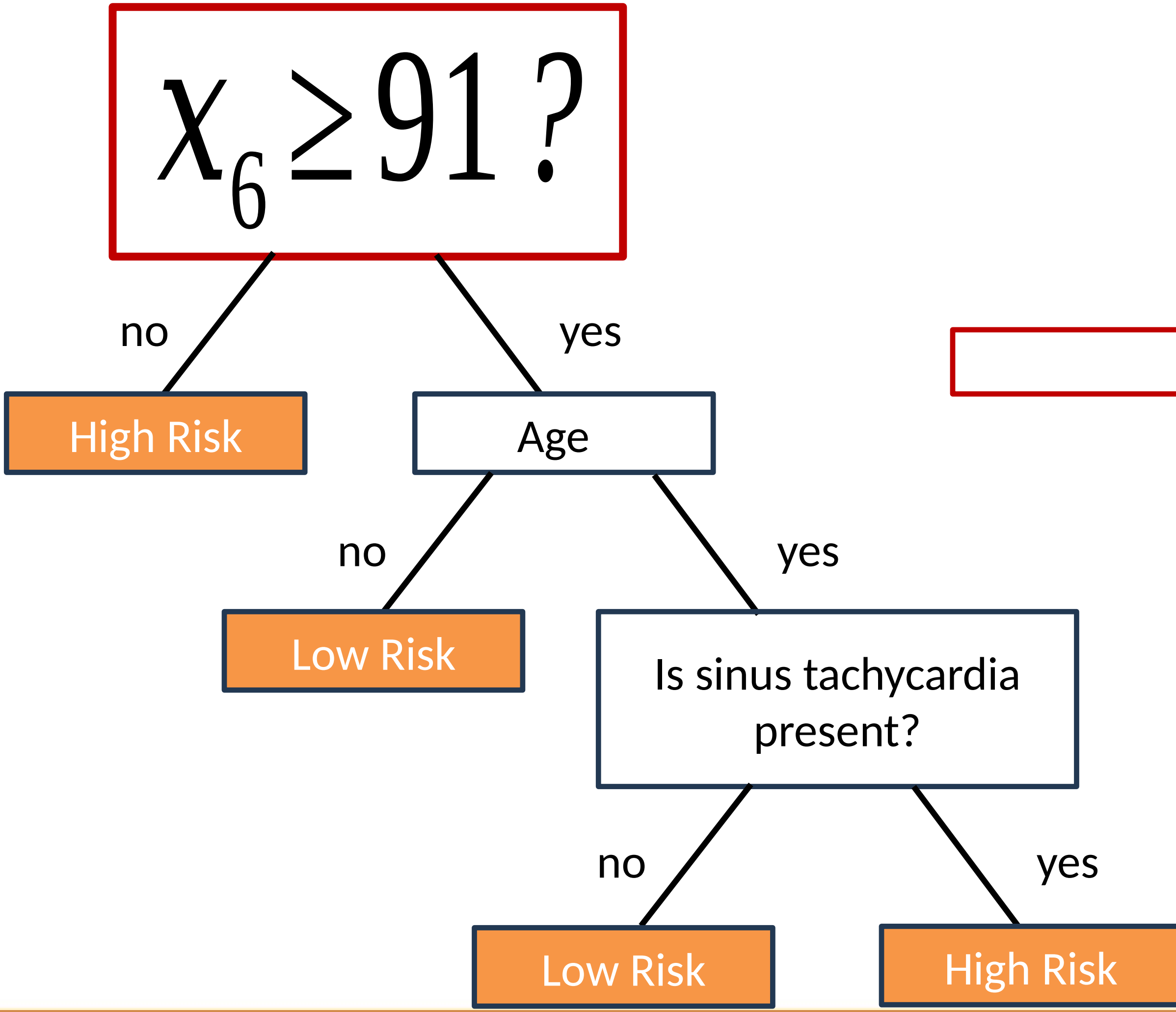
Features:

Example 1) Determining risk for a patient



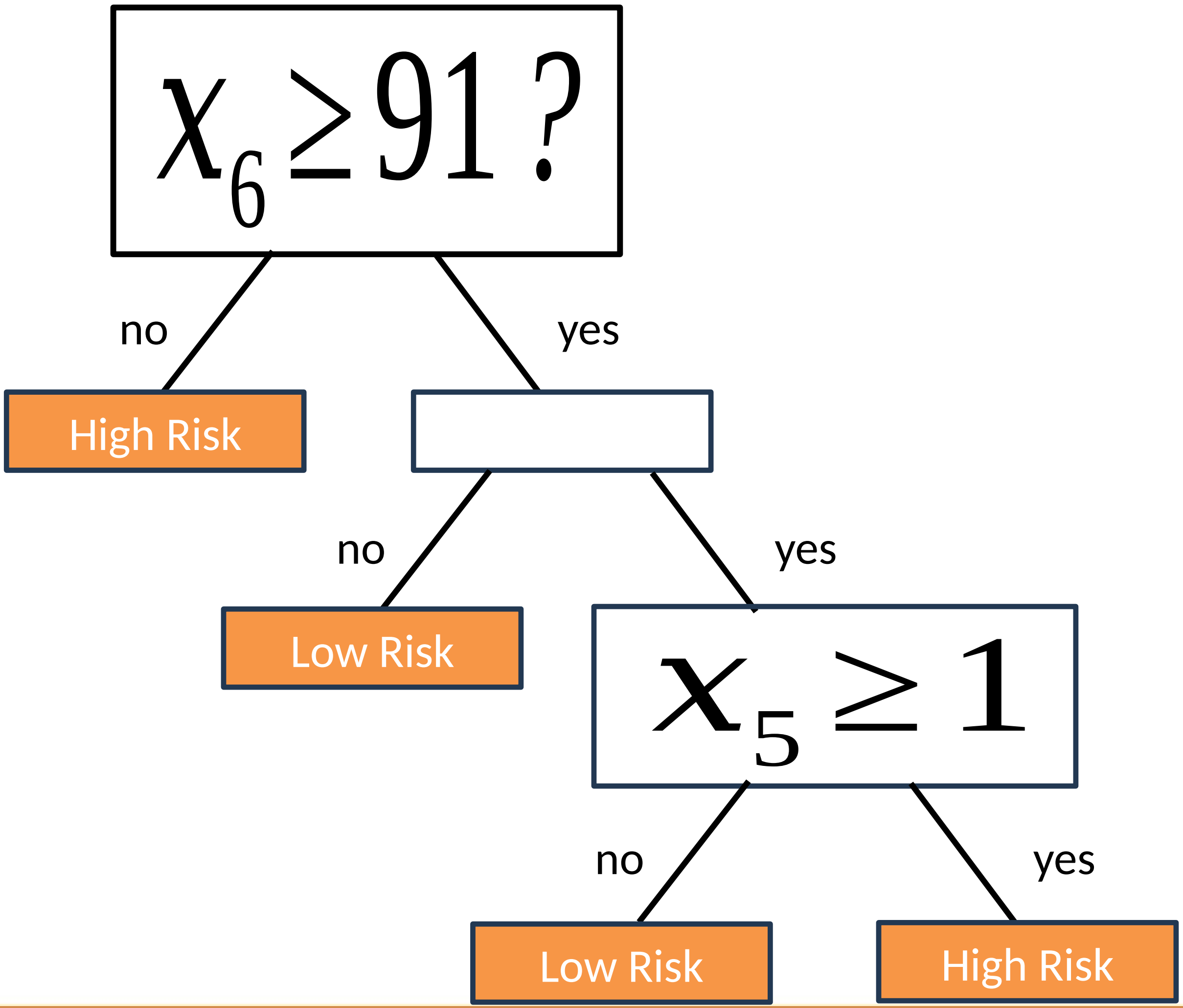
Features:

Example 1) Determining risk for a patient



Features:

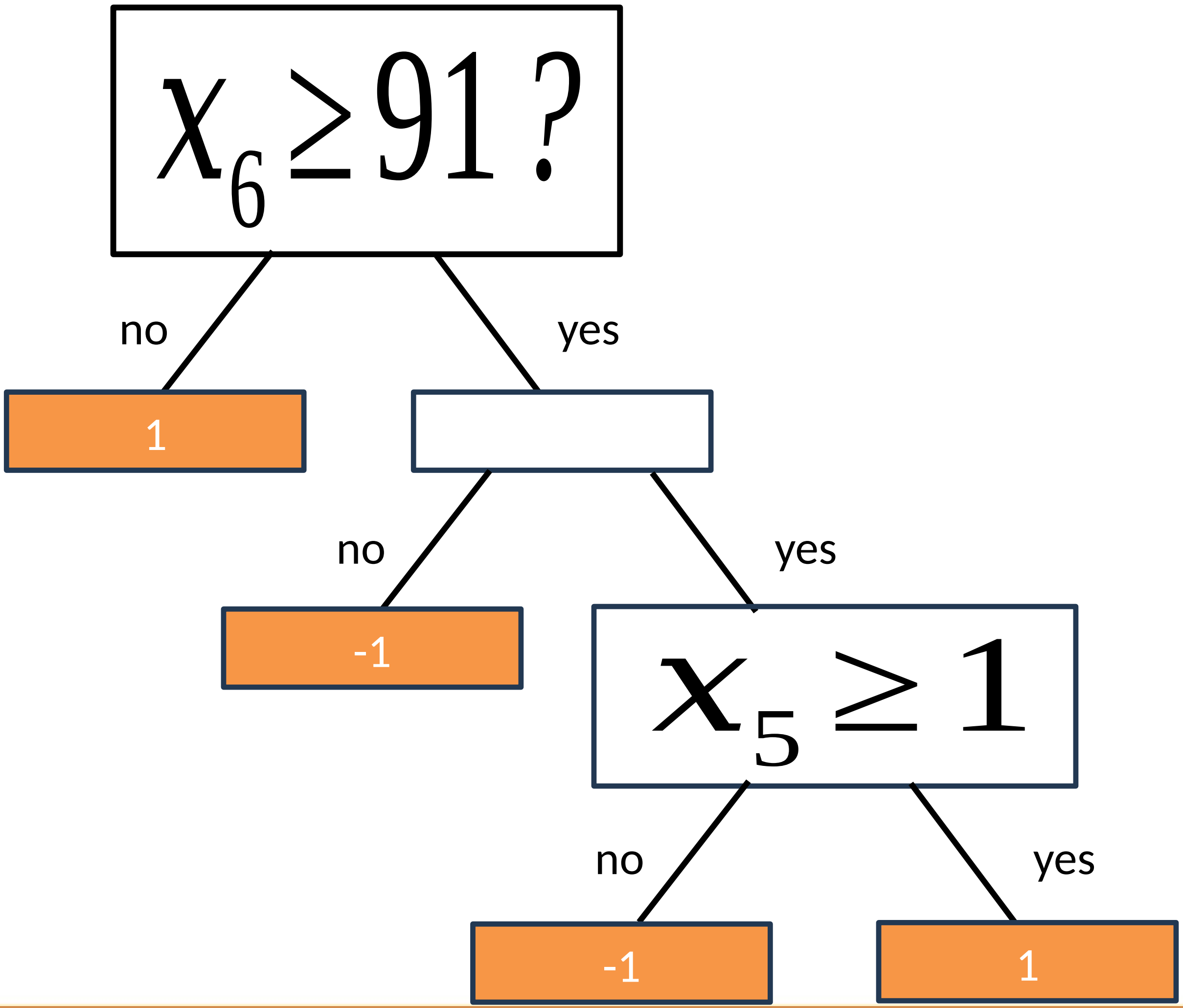
Example 1) Determining risk for a patient



Features:

Labels y:

Example 1) Determining risk for a patient

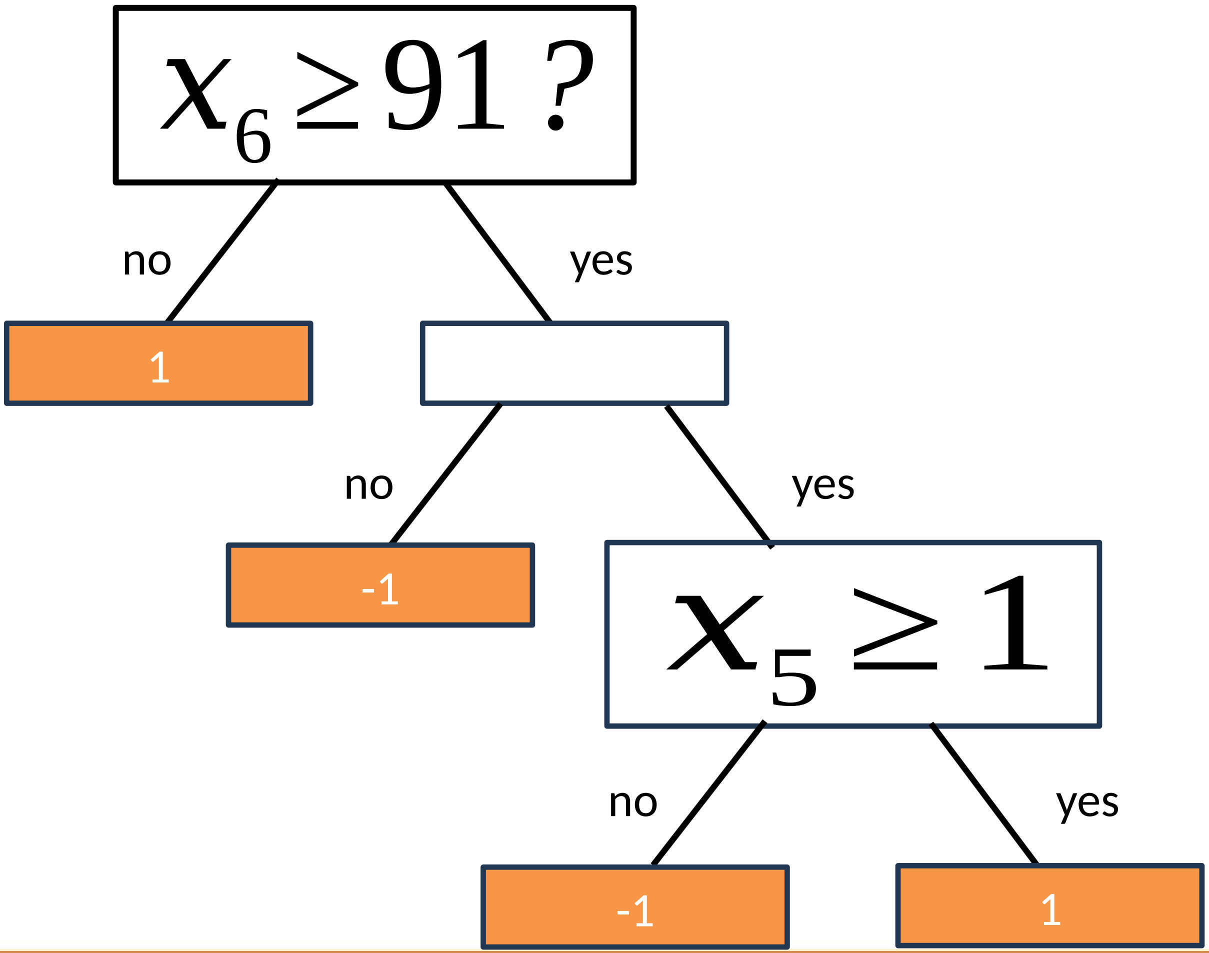


Features:

Labels y:

Example 1) Determining risk for a patient

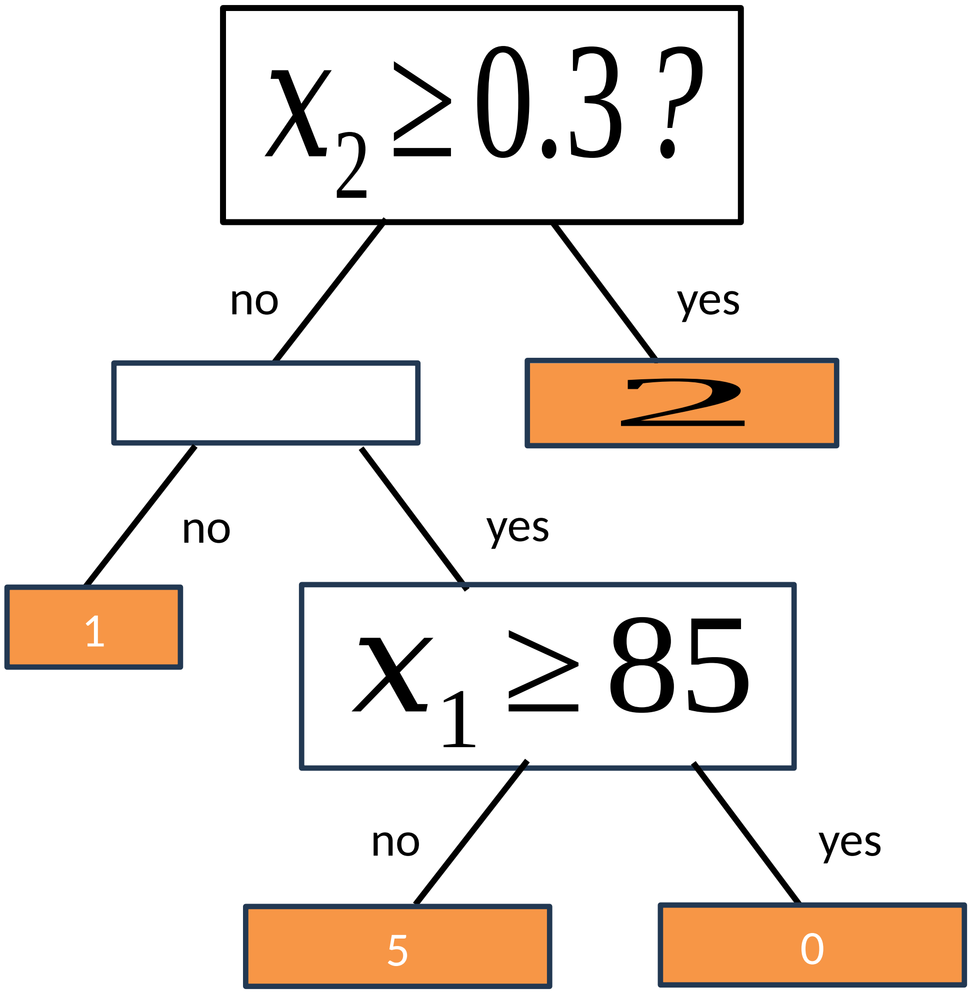
10/01/2023, 51, 5’8”, 170lbs, 0, 118, 80



Features:

Labels y:

Example 2) How many miles will I run?



Features:

Labels y:

In-class exercise)
Predicting home energy usage

- Design a decision tree using the following information
  - For the initial split, **Square Footage** may be strongly correlated with energy usage since larger homes tend to consume more energy
  - Assume more residents likely lead to higher energy usage
  - Assume older homes might be less energy-efficient

- Use the following features and labels

Features
- **Square Footage**: the size of the home
- **Age of Home**: number of years since the home was built
- **Number of Residents**: number of people living in the home
- **Average Outdoor Temperature**: the average temperature outside the home

Labels y:
- **Energy Usage Category**: Low, High

Decision Tree Algorithms

- CART (Classification And Regression Tree)
  - Algorithm for Generating decision trees for classification and regression
  - Typically creates binary trees
- ID3 (Iterative Dichotomiser 3)
  - Algorithm for Generating n-ary trees for classification
  - Uses entropy (measure of unpredictability) and information gain to make the best decision at each node
- C5.0
  - Improved Algorithm from ID3
  - Fast and uses less memory
  - Can handle both continuous and discrete features

Issues in Building Decision Tree

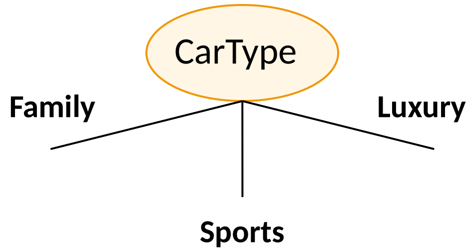
- Greedy Strategy
  - Split the records based on an attribute test that optimizes certain criterion.
- Determine how to split the instances
  - How to specify the attribute test condition?
  - How to determine the best split?
- Determine when to stop splitting

How to Specify Test Condition?

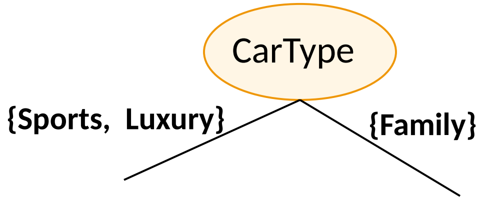
- Depends on attribute types
  - Nominal
    - To consist of discrete values and independence between values
    - i.e., Colors, Countries, Zip codes
    - Splits can be multi-way (one for each category) or binary
  - Ordinal
    - To consist of discrete values and have ordered relationship between values
    - i.e., Lecture Grade, Cancer Stage
    - Splits are often binary and should preserve the order of the categories
  - Continuous
    - To consist of continuous values
    - i.e., Height, weight, age
    - Binary splits are based on a threshold that divides the instances into
      - those with attribute values less or equal
      - those with greater values
- Depends on number of ways to Split
  - 2-way Split ☾ Binary Tree
  - Multi-way Split ☾ N-ary Tree

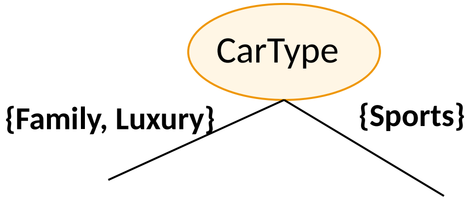
Splitting Based on Nominal Attributes

- Multi-way split
  – Use as many partitions as distinct values.



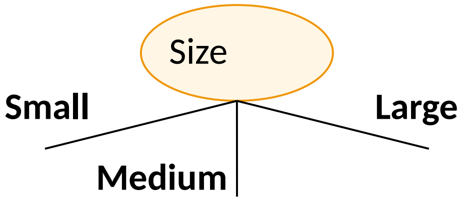
- Binary split
  – Divides values into two subsets.
  – Need to find optimal partitioning.



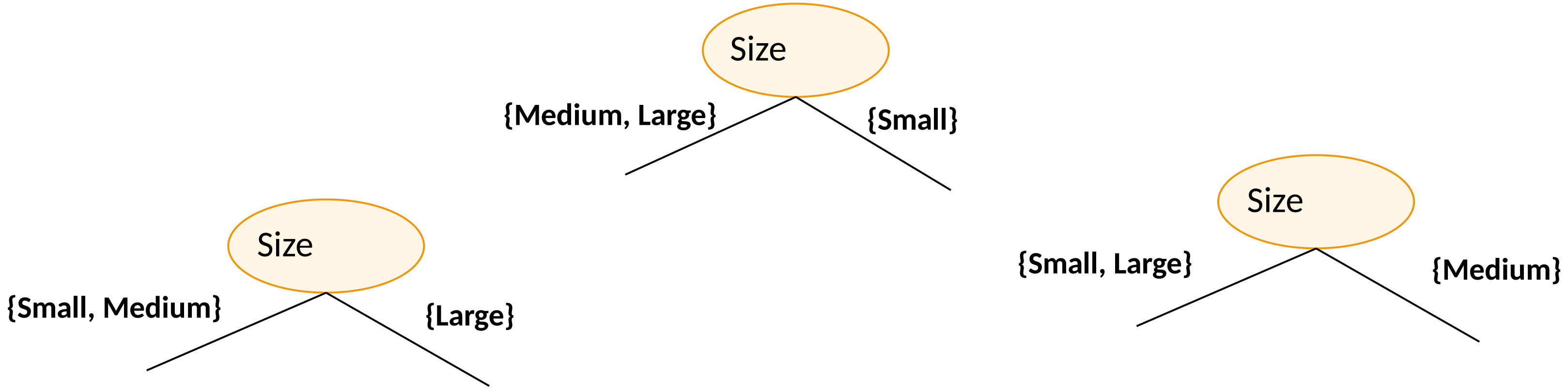


Splitting Based on Ordinal Attributes

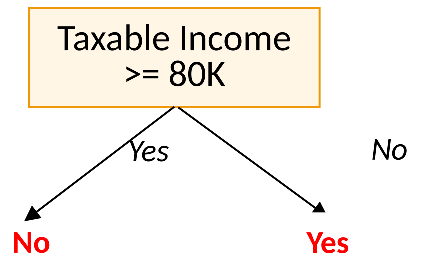
- Multi-way Split
  - Use as many partitions as distinct values.



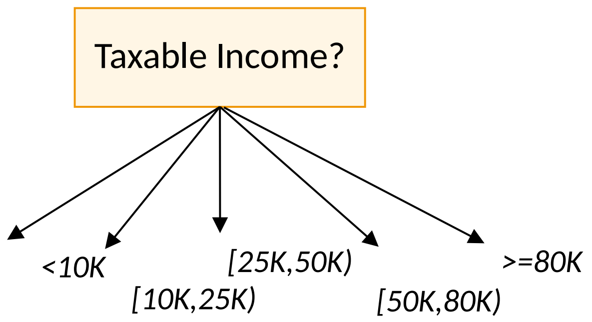
- Binary Split
  - Divides values into two subsets.
  - Need to find optimal partitioning.



Splitting Based on Continuous Attributes



Binary Split



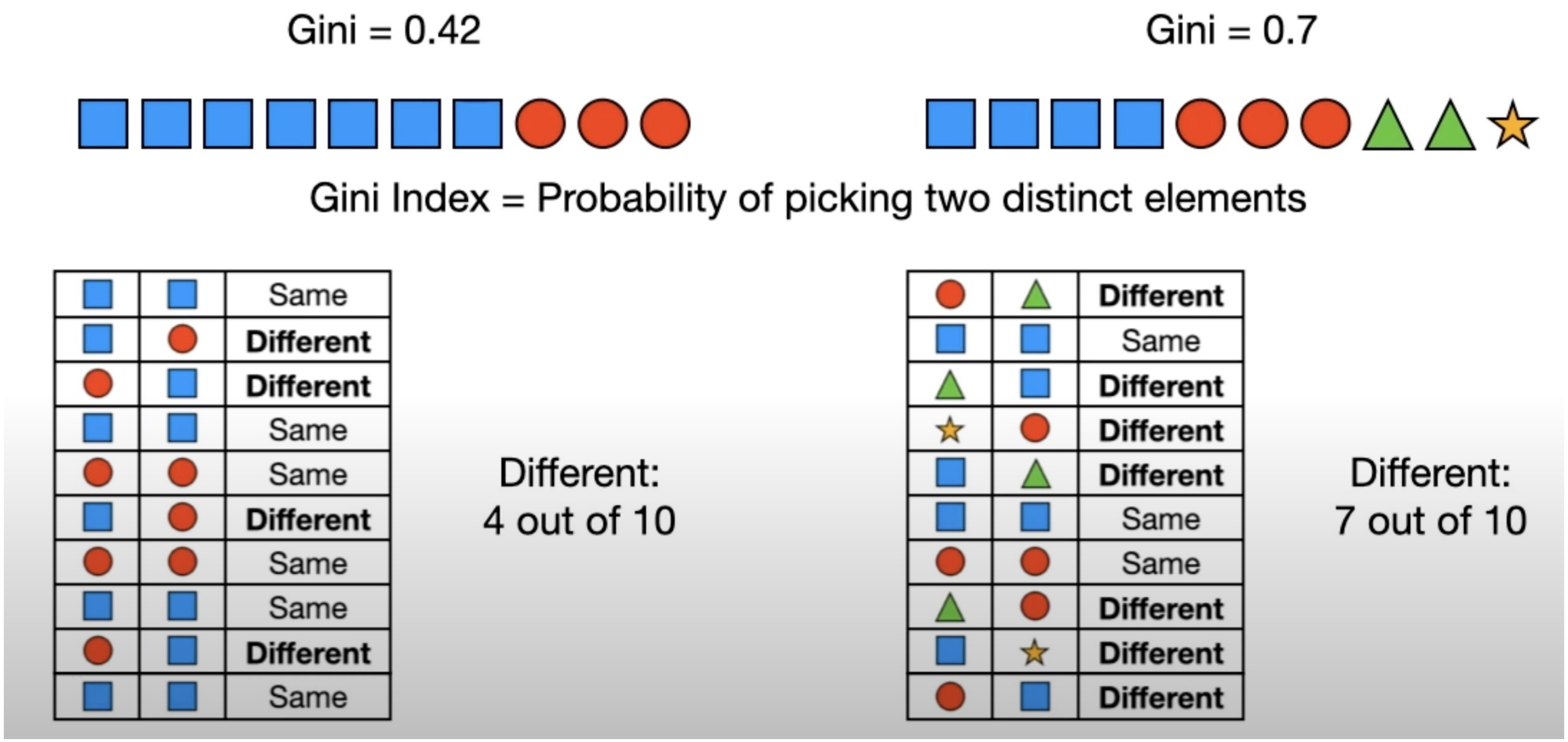
Multi-way Split

Measures of Node Impurity

- Gini impurity – probability of misclassifying a randomly chosen element in a set
- Entropy - measures the amount of uncertainty or randomness in a set

Gini impurity

- Which set is more diverse/impure?



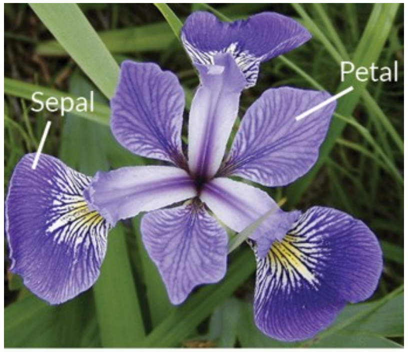
Source - https://www.youtube.com/watch?v=u4IxOk2ijSs&t=94s

Decision Tree – Case Study

Problem Statement: Iris Classifier

- Purpose
  - This program is to classify ‘Iris’, blue-eyed grass
- Species of Iris
  - Setosa
  - Versicolor
  - Virginica

Iris Setosa

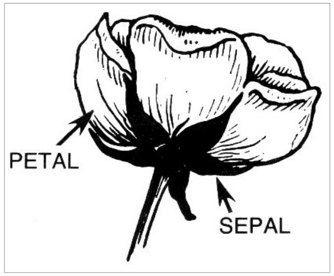


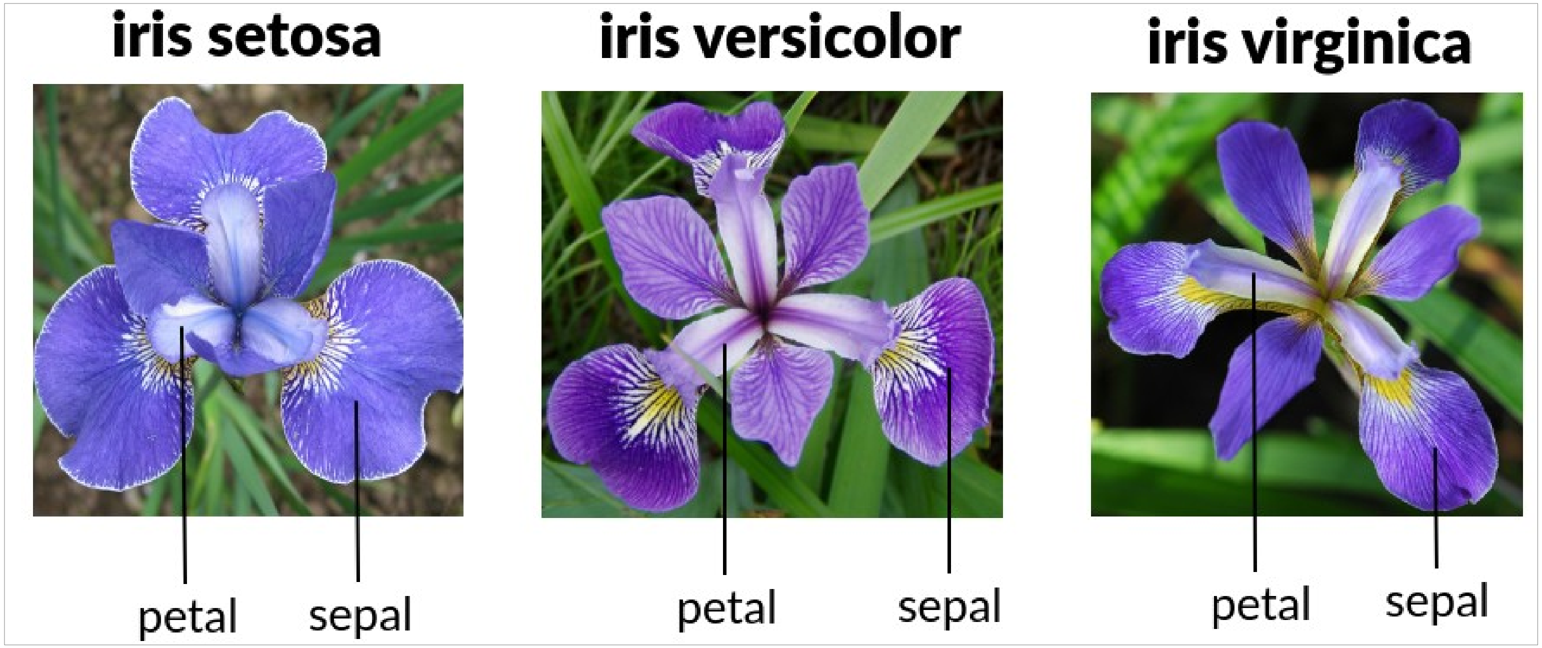
Iris Versicolor

Iris Virginica

Problem Statement: Iris Classifier

- How to Distinguish Iris Species?
  - Length and Width of Sepal
  - Length and Width of Petal





Dataset of Iris

- Iris Dataset is available for research.
  - A total of 150 Irises
    - 50 for Setosa
    - 50 for Versicolor
    - 50 for Virginica
- Attributes for Iris Instances
  - SepalLengthCm
  - SepalWidthCm
  - PetalLengthCm
  - PetalWidthCm
  - Species (as Label)

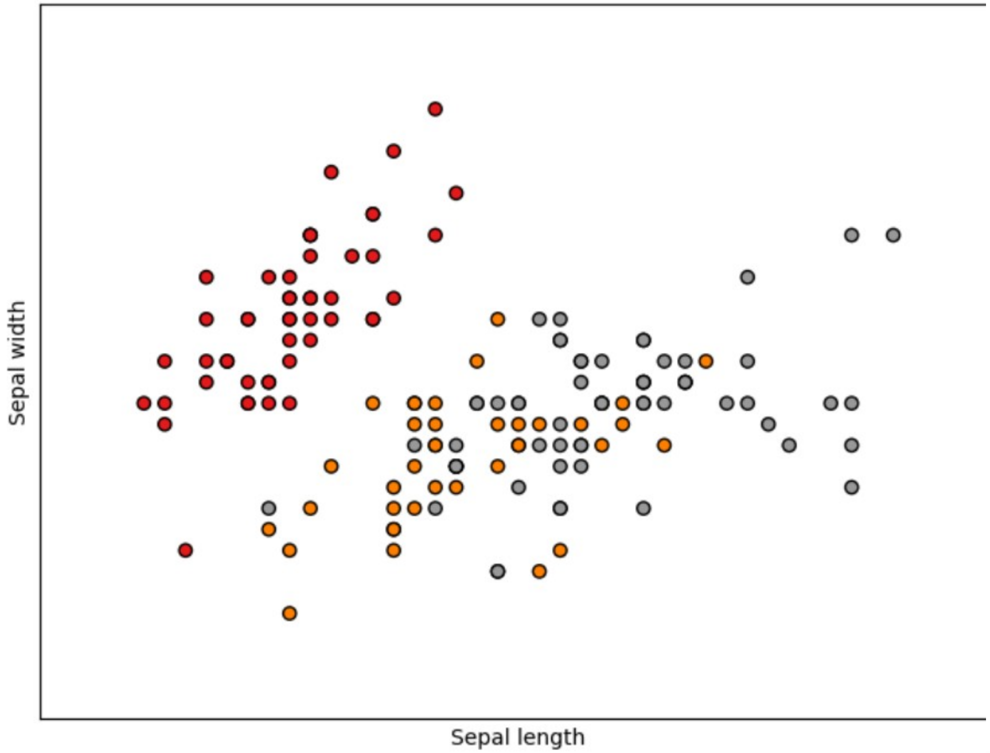
Classes	3
Samples per class	50
Samples total	150
Dimensionality	4
Datatype	real, positive

Dataset of Iris

Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
1	5.1	3.5	1.4	0.2	Iris-setosa
2	4.9	3	1.4	0.2	Iris-setosa
3	4.7	3.2	1.3	0.2	Iris-setosa
4	4.6	3.1	1.5	0.2	Iris-setosa
5	5	3.6	1.4	0.2	Iris-setosa
⋮					
51	7	3.2	4.7	1.4	Iris-versicolor
52	6.4	3.2	4.5	1.5	Iris-versicolor
53	6.9	3.1	4.9	1.5	Iris-versicolor
54	5.5	2.3	4	1.3	Iris-versicolor
55	6.5	2.8	4.6	1.5	Iris-versicolor
56	5.7	2.8	4.5	1.3	Iris-versicolor
⋮					
101	6.3	3.3	6	2.5	Iris-virginica
102	5.8	2.7	5.1	1.9	Iris-virginica
103	7.1	3	5.9	2.1	Iris-virginica
104	6.3	2.9	5.6	1.8	Iris-virginica
105	6.5	3	5.8	2.2	Iris-virginica

Distribution of Iris Instances

- Red for Setosa
- Orange for Vesicolor
- Gray for Virginica



Preparing Training and Test Sets

- 80% for Training Set

```
[[1.4 0.2]
 [4.5 1.7]
 [4.8 1.4]
 [1.5 0.1]
 [4.  1.3]
 [5.  1.9]
 [4.2 1.3]
 [5.1 1.5]
 [6.1 2.3]
 [6.7 2.2]
 [6.6 2.1]
 [4.5 1.5]
 [3.5 1. ]
 [4.1 1. ]
 [4.  1.2]
 [1.6 0.2]
 [1.5 0.3]
 [5.8 1.8]
 [5.5 1.8]
 [1.5 0.4]
 [4.9 1.5]
 [1.6 0.2]
 [6.4 2. ]
 [1.6 0.2]
 [3.7 1. ]
 [5.6 1.8]
 [5.3 1.9]
 [1.6 0.4]
 [4.9 2. ]
 [1.3 0.3]
 [1.5 0.4]
 [4.  1.3]
 [4.6 1.4]
 [1.5 0.4]
 [5.4 2.1]
 [5.4 2.3]
 [3.5 1. ]
 [3.9 1.4]
 [5.7 2.5]
 [4.8 1.8]
 [1.4 0.3]
 [4.7 1.4]
 [1.3 0.4]
 [6.1 1.9]
 [1.4 0.2]
 [1.4 0.3]
 [5.1 1.9]
 [1.2 0.2]
 [1.3 0.3]
 [1.4 0.1]
 [1.4 0.3]
 [3.6 1.3]
 [6.  2.5]
 [4.5 1.3]
 [1.4 0.2]
 [5.1 1.8]
 [4.1 1.3]
 [5.6 2.4]
 [1.4 0.2]
 [5.1 1.9]
 [1.3 0.2]
 [4.  1.3]
 [5.  2. ]
 [1.9 0.2]
 [5.1 1.6]
 [4.4 1.2]
 [6.7 2. ]
 [4.5 1.5]
 [4.6 1.5]
 [5.3 2.3]
 [1.5 0.2]
 [1.7 0.5]
 [1.4 0.2]
 [5.  1.5]
 [3.9 1.2]
 [5.8 1.6]
 [3.8 1.1]
 [5.2 2.3]
 [4.9 1.8]
 [4.5 1.6]
 [1.5 0.2]
 [5.1 2.4]
 [4.5 1.5]
 [1.6 0.6]
 [5.7 2.3]
 [1.5 0.1]
 [6.1 2.5]
 [3.  1.1]
 [4.7 1.5]
 [1.6 0.2]
 [4.3 1.3]
 [5.1 2. ]
 [1.9 0.4]
 [1.1 0.1]
 [5.7 2.1]
 [6.9 2.3]
 [5.5 2.1]
 [4.4 1.3]
 [4.8 1.8]
 [1.7 0.4]
 [5.9 2.3]
 [4.2 1.3]
 [6.  1.8]
 [5.9 2.1]
 [1.5 0.2]
 [4.3 1.3]
 [4.7 1.2]
 [4.2 1.2]
 [4.1 1.3]
 [4.4 1.4]
 [1.5 0.2]
 [5.6 2.1]
 [4.  1. ]
 [4.2 1.5]
 [1.6 0.2]
 [1.4 0.1]
 [1.3 0.2]
 [1.4 0.2]
 [4.9 1.5]
 [1.5 0.2]]
```

- 20% for Test Set

```
[[1.3 0.2]
 [4.9 1.8]
 [3.3 1. ]
 [3.3 1. ]
 [1.3 0.2]
 [4.7 1.6]
 [1.  0.2]
 [1.7 0.2]
 [5.2 2. ]
 [4.5 1.5]
 [6.3 1.8]
 [5.1 2.3]
 [5.8 2.2]
 [4.5 1.5]
 [1.5 0.2]
 [1.4 0.2]
 [1.2 0.2]
 [3.9 1.1]
 [4.7 1.4]
 [4.8 1.8]
 [1.5 0.2]
 [5.6 2.4]
 [4.6 1.3]
 [5.6 1.4]
 [5.6 2.2]
 [5.  1.7]
 [4.4 1.4]
 [1.7 0.3]
 [5.5 1.8]
 [1.4 0.2]]
```

Library

- Scikit-Learn
  - ML Library for Python
  - *DecisionTreeClassifier* class
- Hyperparameters
  - max_depth: ‘2’
  - criterion: ‘gini’
  - splitter: ‘best’
  - min_samples_split: 2
  - min_samples_leaf: 1
  - min_weight_fraction_leaf: 0.0
  - max_features: None
  - max_leaf_nodes: None
  - min_impurity_decrease: 0.0

Step 1. Loading Dataset

- To load iris dataset from Scikit-Learn
  – To use load_iris() method
    - The iris dataset is provided from scikit-learn library as default

```
from sklearn.metrics import classification_report, accuracy_score
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from joblib import dump, load

# To load Iris Dataset
data_iris = load_iris()
```

Step 2. Splitting Dataset

- To split the dataset as Training set and Test set
  - To apply petal length and petal width as features
    - Values for Training Set
      - X_train, y_train
    - Values for Test Set
      - X_test, y_test

```
X = data_iris.data[:, 2:]    # To set petal length and petal width to features
y = data_iris.target         # To set labels

# To split dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y, random_state=42)
```

Step 3. Training Model

- To train Decision Tree Model
  - Code
    - To use *fit()* method

```
iris_classifier = DecisionTreeClassifier(criterion="gini", max_depth=2) # To define classifier object
iris_classifier.fit(X_train, y_train)   # To train model using training set
```

- To save and Load the model
  - Code

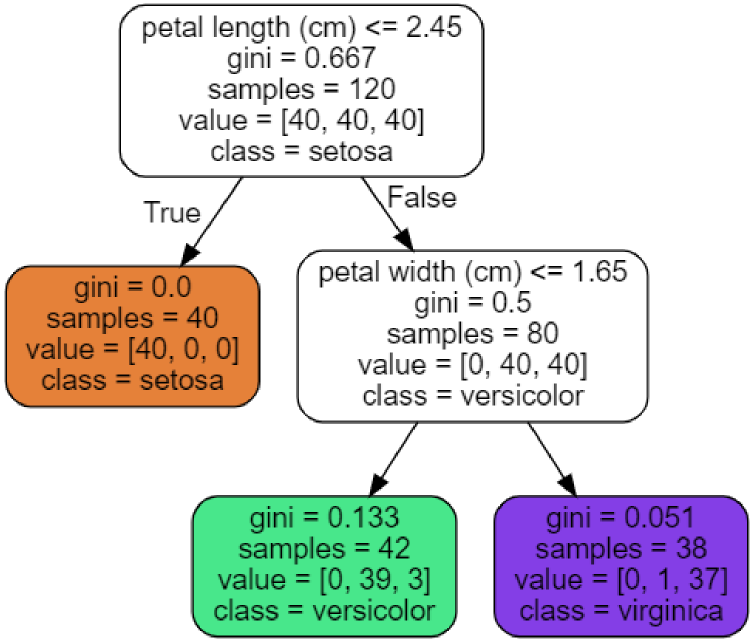
```
dump(iris_classifier, "iris_classifier.joblib")      # To save trained model
iris_classifier = load("iris_classifier.joblib")     # To load saved model
```

  - Result

iris_classifier.joblib	10/22/2020 9:43 PM	JOBLIB File	2 KB

Step 3. Training Model

- To visualize the generated Decision Tree
  - To use graphviz library



Step 4. Predicting Test Set

- Code to predict with test set
  – To use *predict()* method in trained model object
    - To predict labels for training set and test set

```
y_pred_train = iris_classifier.predict(X_train)
y_pred_test = iris_classifier.predict(X_test)
```

Step 5. Evaluating Model

- Use the evaluation methods

```
print("Accuracy for Training Set: ", round(accuracy_score(y_train, y_pred_train),2))
print("Accuracy for Test Set: ", round(accuracy_score(y_test, y_pred_test),2))


print(classification_report(y_train, y_pred_train))
print(classification_report(y_test, y_pred_test))
```

Step 5. Evaluating Model

- To compare the accuracy
  - Accuracy of Training set
    - 0.97 `Accuracy for Training Set:  0.97`
  - Accuracy of Test set
    - 0.93 `Accuracy for Test Set:  0.93`
- To check precision (accuracy of positive predictions), recall (true positive rates), and f1-score (balance of precision and recall)

Performance for Training Set

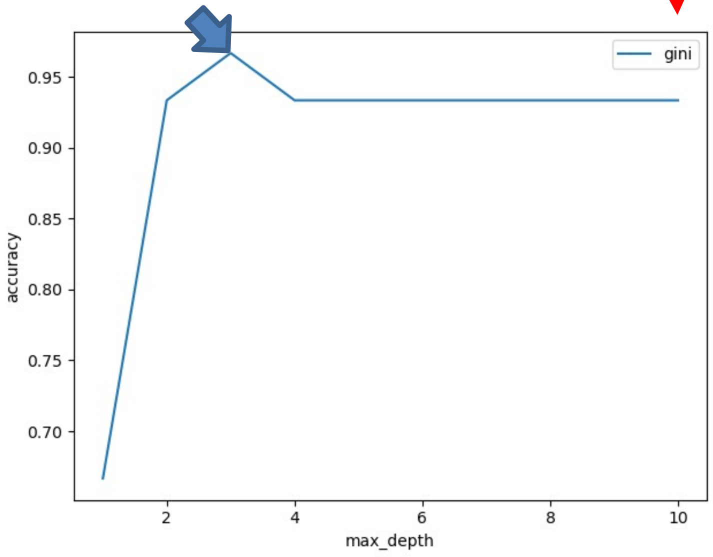
	precision	recall	f1-score	support
0	1.00	1.00	1.00	40
1	0.93	0.97	0.95	40
2	0.97	0.93	0.95	40
accuracy			0.97	120
macro avg	0.97	0.97	0.97	120
weighted avg	0.97	0.97	0.97	120

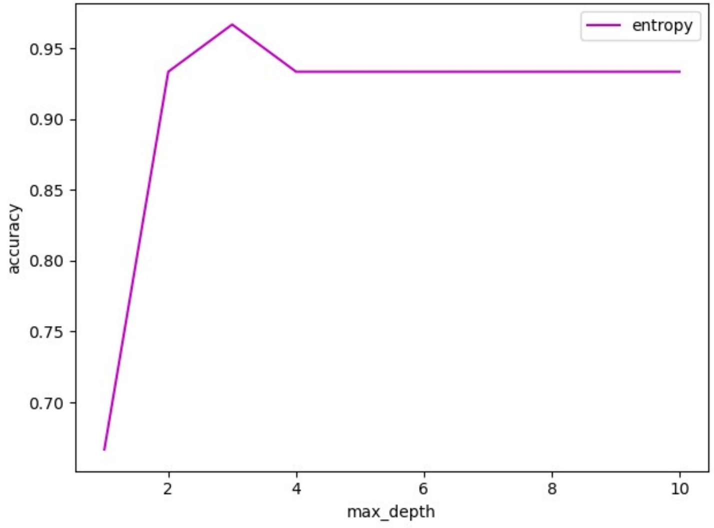
Performance for Test Set

	precision	recall	f1-score	support
0	1.00	1.00	1.00	10
1	0.90	0.90	0.90	10
2	0.90	0.90	0.90	10
accuracy			0.93	30
macro avg	0.93	0.93	0.93	30
weighted avg	0.93	0.93	0.93	30

Step 6. Fine-Tuning Model

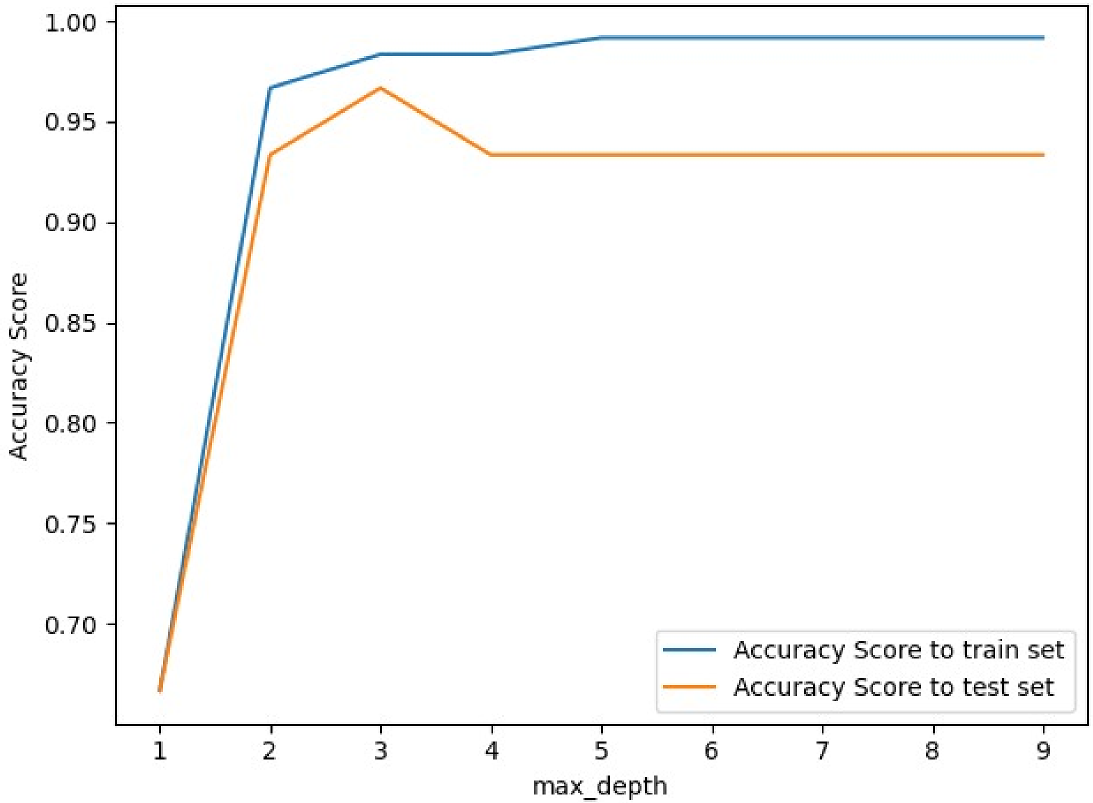
- Comparing criterion=‘gini’ and criterion=‘entropy’
  - The results are same
- Comparison by the max_depth
  - The performance of model is best when max_depth is 3.





Step 6. Fine-Tuning Model

- To check max_depth from 1 to 9
  - max_depth=4
    - overfitted
  - max_depth=3
    - better performance than max_depth of 2



References

- Russel and Norvig, Artificial Intelligence: A Modern Approach, 3rd edition, Prentice Hall, 2010.
- https://tamarabroderick.com/files/ml_6036_2020_lectures/broderick_lecture_12.pdf