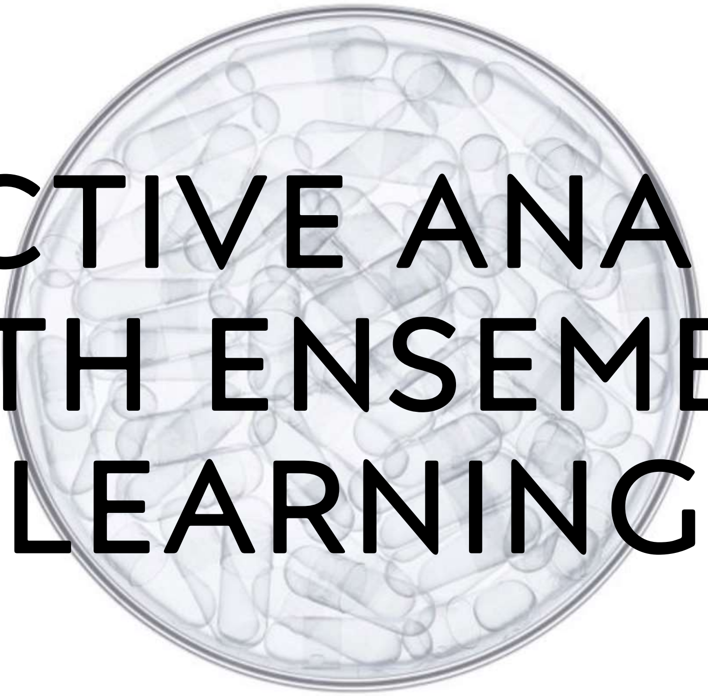


PREDICTIVE ANALYTICS WITH ENSEMBLE LEARNING

TIM AJAR KECERDASAN ARTIFICIAL

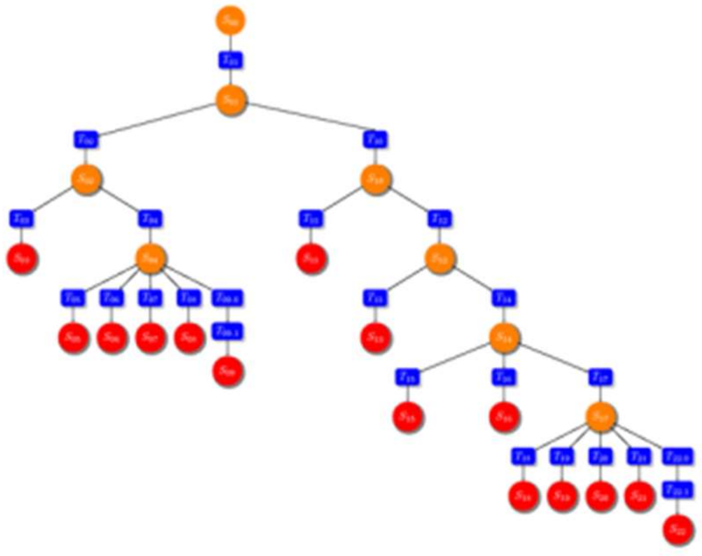
AGENDA

INTRODUCTION

DECISION TREES CLASSIFIER

RANDOM FOREST

WHAT ARE DECISION TREES?



- Decision trees are machine learning models that are in the form of tree structures.
- A decision tree is a way to partition a dataset into distinct branches. The branches or partitions are then traversed to make simple decisions.
- The decision process starts at the root node at the top of the tree. Each node in the tree is a decision rule. Algorithms construct these rules based on the relationship between the input data and the target labels in the training data. The values in the input data are utilized to estimate the value of the output.

ENTROPY

$$H(X) = -\sum_{i=0}^{n} P(x_i)\ log_2 P(x_i)$$

where, H-Entropy, n-number of classes, P(x^i)-probability of getting x of class i.

- Entropy is a concept that stems from information theory, which measures the impurity of the sample values.
- Entropy values can fall between 0 and 1. If all samples in data set, S, belong to one class, then entropy will equal zero. If half of the samples are classified as one class and the other half are in another class, entropy will be at its highest at 1
- In order to select the best feature to split on and find the optimal decision tree, the attribute with the smallest amount of entropy should be used.

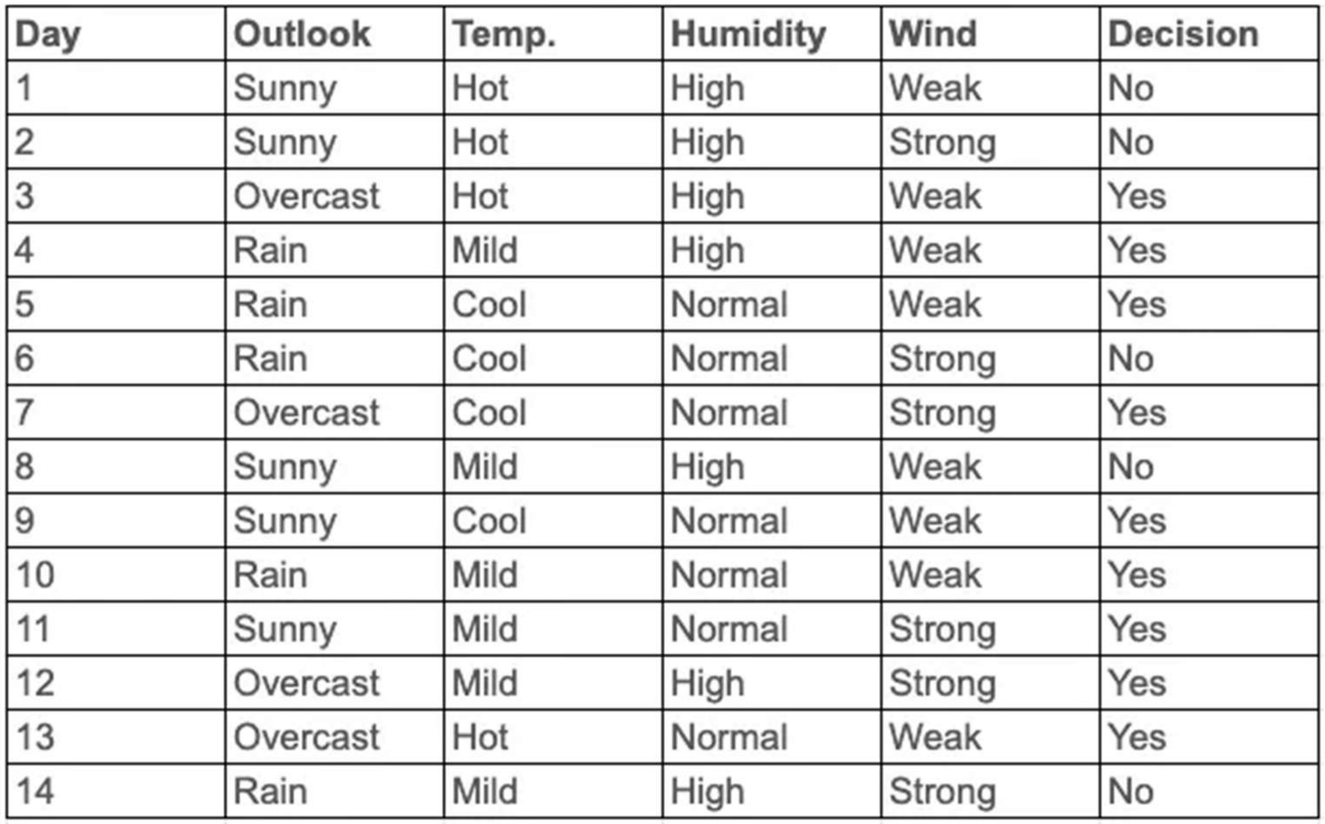
INFORMATION GAIN

$$Gain(S,A) = H(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} H(S_v)$$

where, Values(A) is the set of all possible values for attribute A, and S_v is the subset of S, for which attribute A has value v.

- Information gain represents the difference in entropy before and after a split on a given attribute. The attribute with the highest information gain will produce the best split as it's doing the best job at classifying the training data according to its target classification.

LET'S WALK THROUGH AN EXAMPLE TO SOLIDIFY THESE CONCEPTS. IMAGINE THAT WE HAVE THE FOLLOWING ARBITRARY DATASET:



Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Questions:
how to decide which attribute forms the root node?
Ans: The attribute which has the highest Information Gain is to be chosen as the root node, Because the higher the Information gain value the better it classifies the training data.

Step 1: Let's calculate Entropy for ***PlayTennis***

From PlayTennis data,

# of days = 14
Classifications, n = 2 (yes/no)
# of yes – 9
# of no - 5
$P(\text{yes}) = \frac{9}{14}$, $P(\text{no}) = \frac{5}{14}$

$$H(\text{PlayTennis}) = -\left[P(yes) log_2 P(yes) + P(no)\ log_2 P(no) \right]$$

$$H(\text{PlayTennis}) = -\left[\frac{9}{14} log_2\left(\frac{9}{14}\right) + \frac{5}{14} log_2\left(\frac{5}{14}\right)\right]$$

$$H(\text{PlayTennis}) = 0.940$$

Step 2: let's calculate the IG of attributes Outlook, Temp, Humidity, and Wind.
Step 2.1: calculating Information Gain of attribute outlook.

$$\begin{aligned}\text{Gain}(\text{PlayTennis, Outlook}) &\\ = \text{H}(\text{PlayTennis}) &\\ - \Bigg[\frac{|\text{PlayTennis}_{(\text{sunny})}|}{|\text{PlayTennis}|} \text{H}(\text{PlayTennis}_{(\text{sunny})}) &\\ + \frac{|\text{PlayTennis}_{(\text{overcast})}|}{|\text{PlayTennis}|} \text{H}(\text{PlayTennis}_{(\text{overcast})}) &\\ + \frac{|\text{PlayTennis}_{(\text{rain})}|}{|\text{PlayTennis}|} \text{H}(\text{PlayTennis}_{(\text{rain})})\Bigg] &\end{aligned}$$

We know, H(PlayTennis) = 0.940
Sunny outlook on PlayTennis,

Sunny outlook on PlayTennis,

Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes

$$\mathrm{H}\ (PlayTennis|outlook = sunny) = -[P(no)\lg \mathrm{P}(no) + P(yes)\lg P(yes)]$$

$$= -\left[\frac{3}{5}\lg\frac{3}{5} + \frac{2}{5}\lg\frac{2}{5}\right]$$

$$= -[-0.442\ -0.528]\ => 0.970$$

$** \ (log_2\ can\ be\ written\ as \lg)$

$H(PlayTennis|outlook = overcast) = 0$

$H(PlayTennis|outlook = rain) = 0.970$

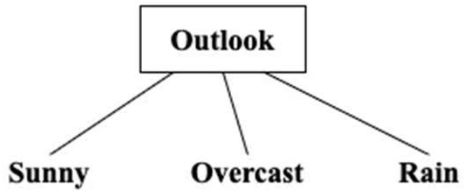
Now, we can calculate Gain(PlayTennis, Outlook)

$$\text{Gain(PlayTennis, Outlook)} = 0.940 - \left[\left(\frac{5}{14} * 0.970\right) + \left(\frac{4}{14} * 0\right) + \left(\frac{5}{14} * 0.970\right)\right]$$

$$= 0.940 - 0.692 \Rightarrow 0.248$$

Gain(PlayTennis,outlook) = 0.248
Gain(PlayTennis,temp.) = 0.029
Gain(PlayTennis,outlook) = 0.151
Gain(PlayTennis,outlook) = 0.048

Since outlook has the highest Information Gain, we choose outlook as the root node.



The next stage of the tree has three attributes namely Sunny, Overcast and Rain. we need to extend the tree for each of these attributes such that they lead to a decision.

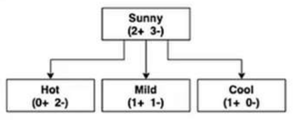
Step 4: Sunny Outlook on PlayTennis

Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes

Step 4.1: We need to find IG of Temp., Humidity, and Wind with respect to Outlook=Sunny

$$Gain(outlook = sunny|Temp.) = H(PlayTennis|outlook = sunny) - \sum_{v \in values(A)} \frac{|S_v|}{|S|} H(S_v)$$

Where, v ∈ {hot, mild, cold} and S -> (PlayTennis|outlook=sunny)



$$H(hot) = -\left[\frac{0}{2}\lg\frac{0}{2} + \frac{2}{2}\lg\frac{2}{2}\right] = 0$$

$$H(mild) = -\left[\frac{1}{2}\lg\frac{1}{2} + \frac{1}{2}\lg\frac{1}{2}\right] = 0$$

$$H(cool) = -\left[\frac{1}{1}\lg\frac{1}{1} + \frac{0}{1}\lg\frac{0}{1}\right] = 0$$

$$Gain(outlook = sunny|Temp.) = 0.970 - [\tfrac{2}{5} * 0 + \tfrac{2}{5} * 1 + \tfrac{1}{5} * 0]$$

$$= 0.970 - \frac{2}{5}$$

$$= 0.570$$

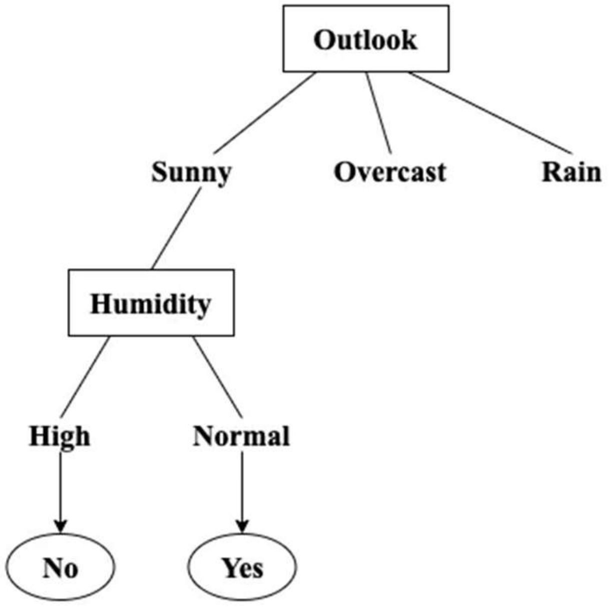
$$Gain(outlook = sunny|Temp.) = 0.570$$

$$Gain(outlook = sunny|Humidity) = 0.970$$

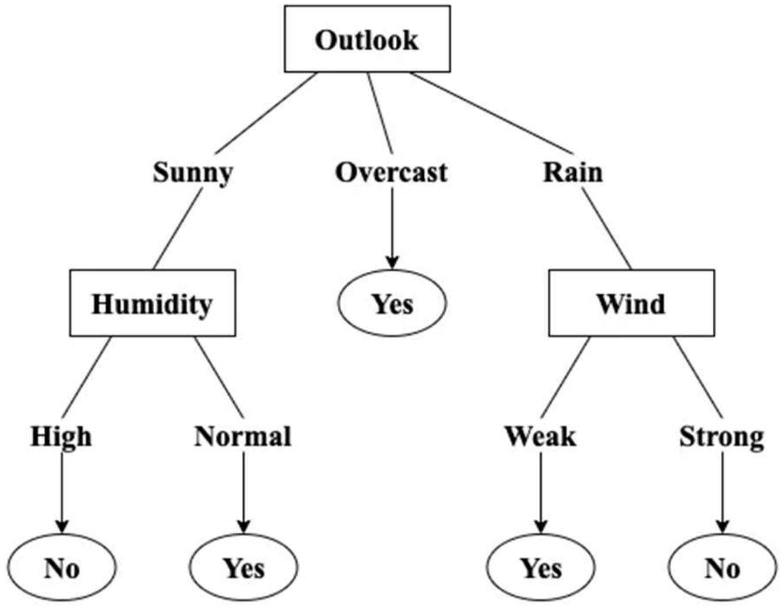
$$Gain(outlook = sunny|Wind) = 0.019$$

Step 4.2: Humidity has the highest gain, hence Sunny can be further extended with humidity.

Humidity has two values. High leads to decision No and Normal leads to decision Yes.



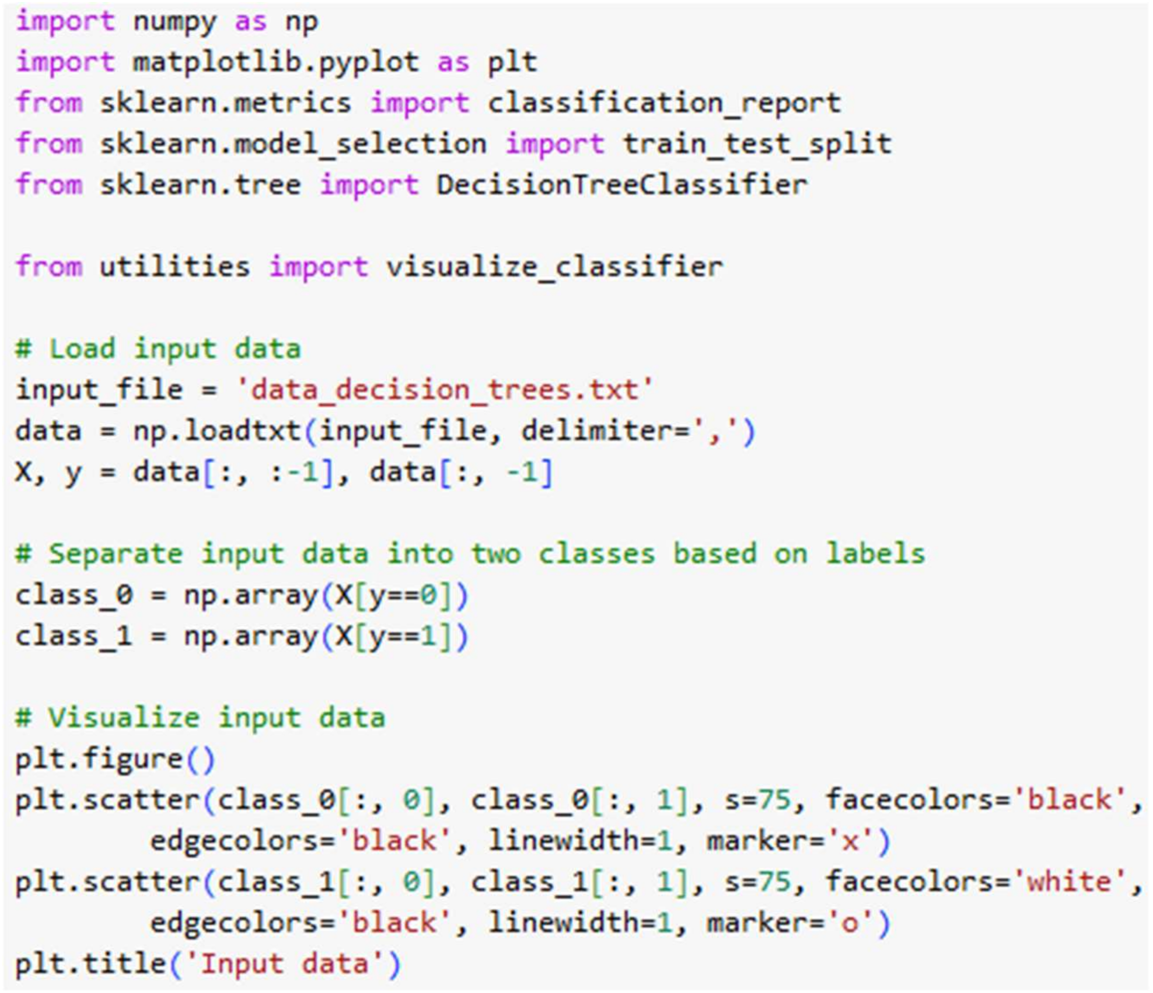
EXAMPLE: DECISION TREE



https://medium.com/@sudhakar.rulz/decision-tree-for-beginners-7cdc319c1470

BUILDING A DECISION TREE CLASSIFIER

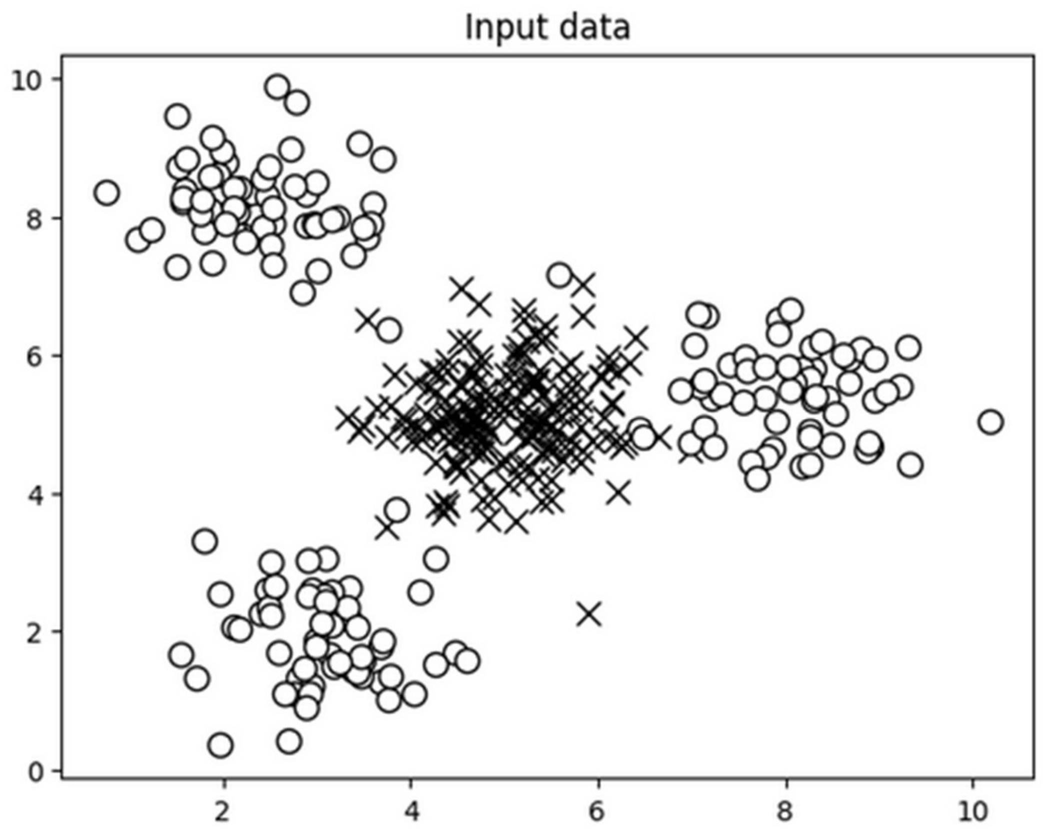
- Download resource (utility.py dan data_decision_trees.txt)
- Open google colab



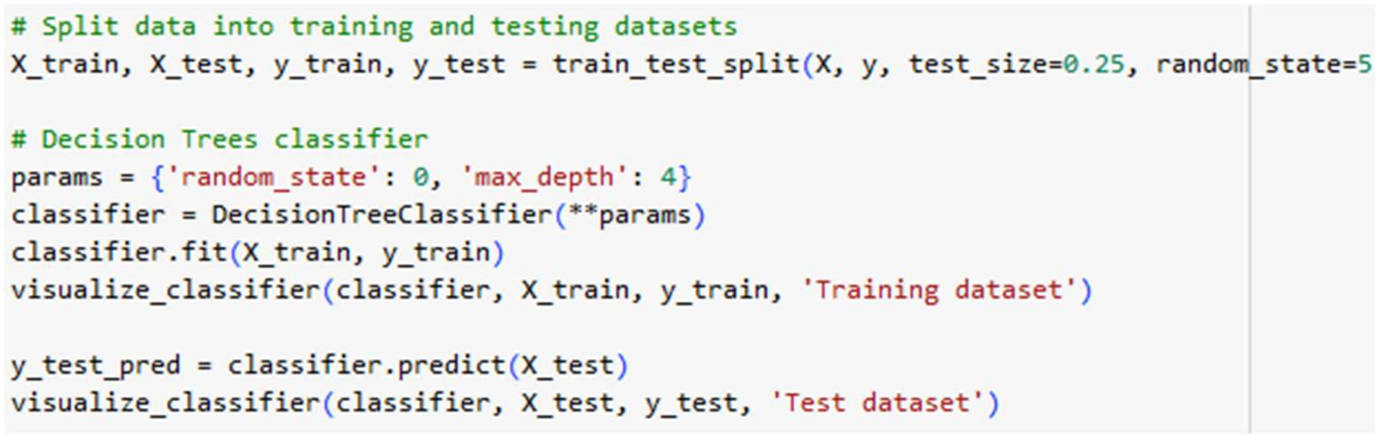
```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier

from utilities import visualize_classifier

# Load input data
input_file = 'data_decision_trees.txt'
data = np.loadtxt(input_file, delimiter=',')
X, y = data[:, :-1], data[:, -1]

# Separate input data into two classes based on labels
class_0 = np.array(X[y==0])
class_1 = np.array(X[y==1])

# Visualize input data
plt.figure()
plt.scatter(class_0[:, 0], class_0[:, 1], s=75, facecolors='black',
        edgecolors='black', linewidth=1, marker='x')
plt.scatter(class_1[:, 0], class_1[:, 1], s=75, facecolors='white',
        edgecolors='black', linewidth=1, marker='o')
plt.title('Input data')
```

BUILDING A DECISION TREE CLASSIFIER

- Download resource (utility.py dan data_decision_trees.txt)
- Open google colab

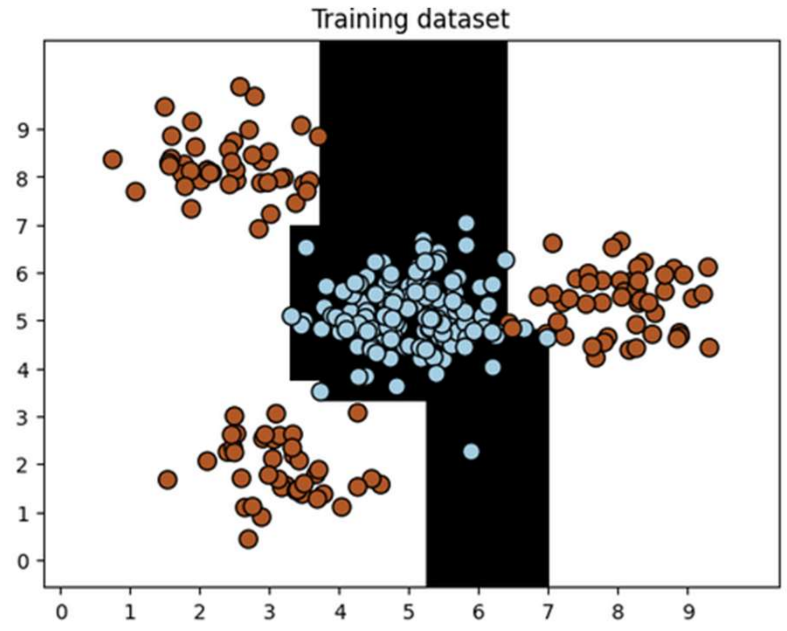


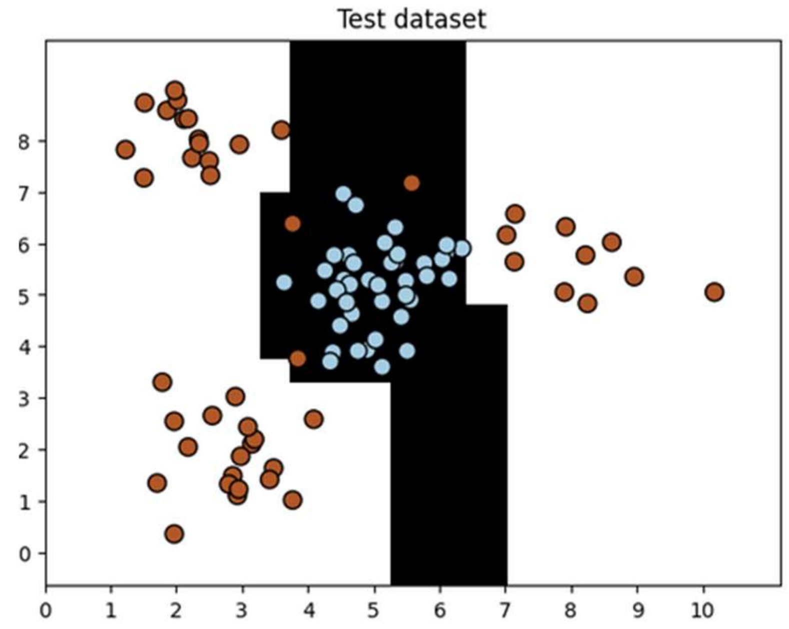
BUILDING A DECISION TREE CLASSIFIER



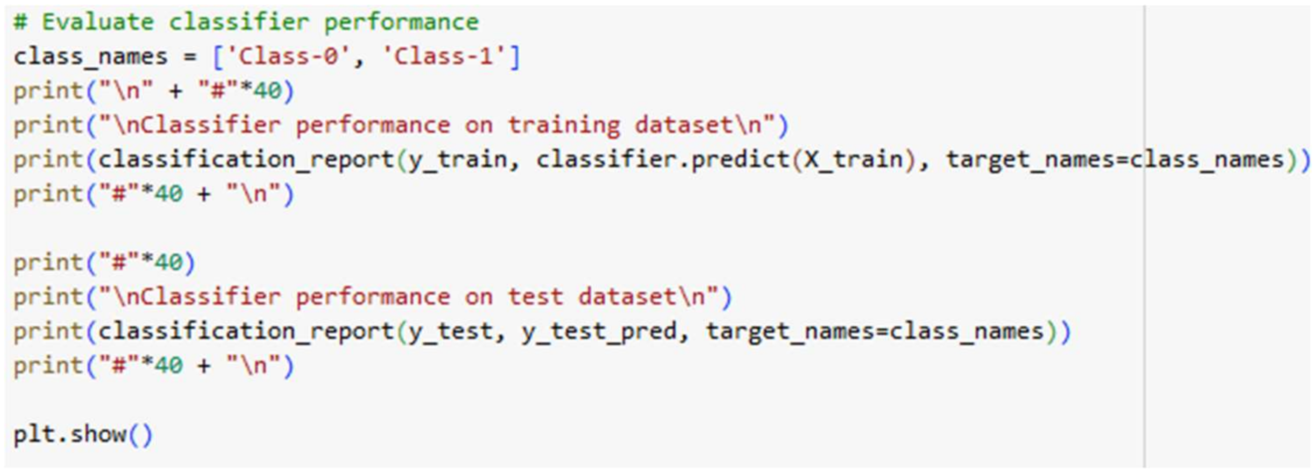
```
# Split data into training and testing datasets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=5

# Decision Trees classifier
params = {'random_state': 0, 'max_depth': 4}
classifier = DecisionTreeClassifier(**params)
classifier.fit(X_train, y_train)
visualize_classifier(classifier, X_train, y_train, 'Training dataset')

y_test_pred = classifier.predict(X_test)
visualize_classifier(classifier, X_test, y_test, 'Test dataset')
```

BUILDING A DECISION TREE CLASSIFIER



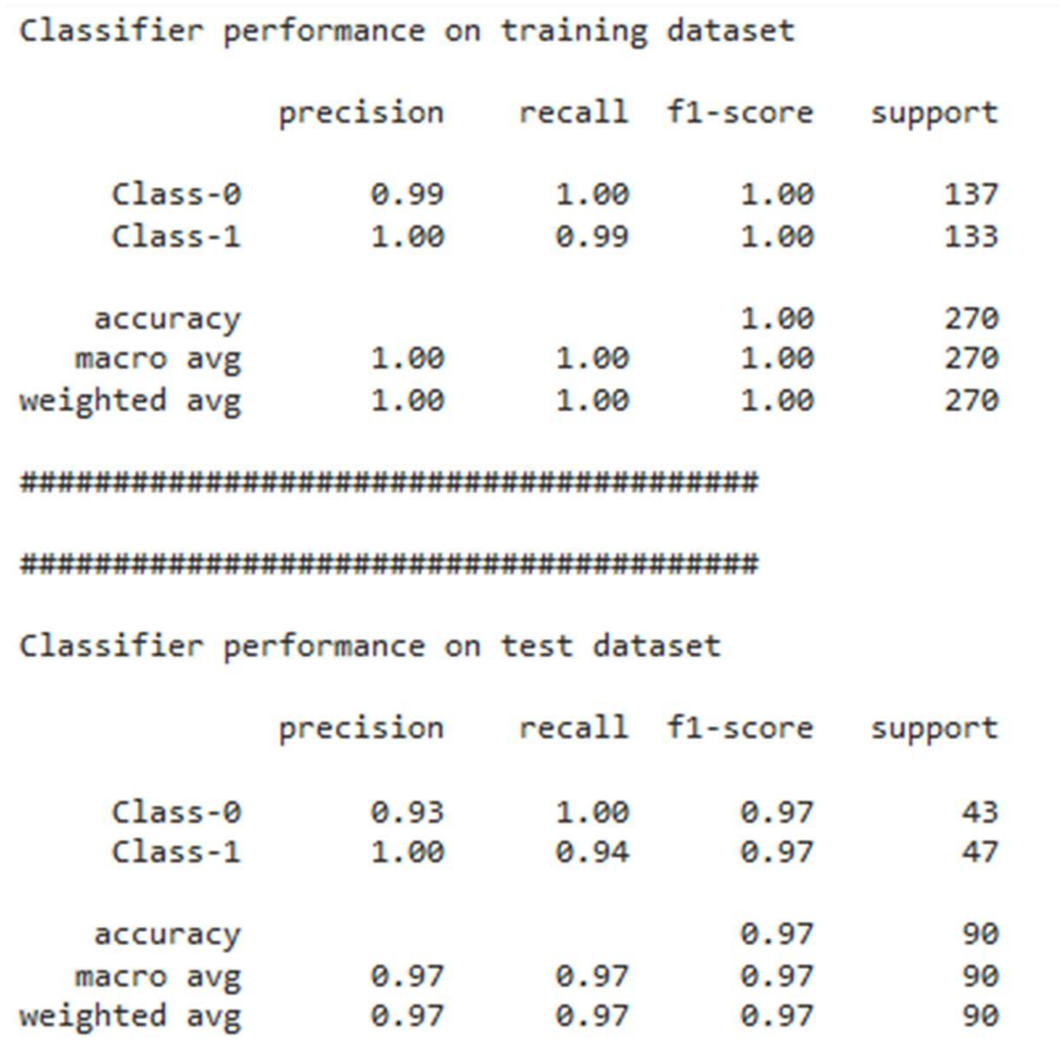


BUILDING A DECISION TREE CLASSIFIER



```
# Evaluate classifier performance
class_names = ['Class-0', 'Class-1']
print("\n" + "#"*40)
print("\nClassifier performance on training dataset\n")
print(classification_report(y_train, classifier.predict(X_train), target_names=class_names))
print("#"*40 + "\n")

print("#"*40)
print("\nClassifier performance on test dataset\n")
print(classification_report(y_test, y_test_pred, target_names=class_names))
print("#"*40 + "\n")

plt.show()
```

BUILDING A DECISION TREE CLASSIFIER



```
Classifier performance on training dataset

              precision    recall  f1-score   support

     Class-0       0.99      1.00      1.00       137
     Class-1       1.00      0.99      1.00       133

    accuracy                           1.00       270
   macro avg       1.00      1.00      1.00       270
weighted avg       1.00      1.00      1.00       270

########################################

########################################

Classifier performance on test dataset

              precision    recall  f1-score   support

     Class-0       0.93      1.00      0.97        43
     Class-1       1.00      0.94      0.97        47

    accuracy                           0.97        90
   macro avg       0.97      0.97      0.97        90
weighted avg       0.97      0.97      0.97        90
```

ENSEMBLE LEARNING

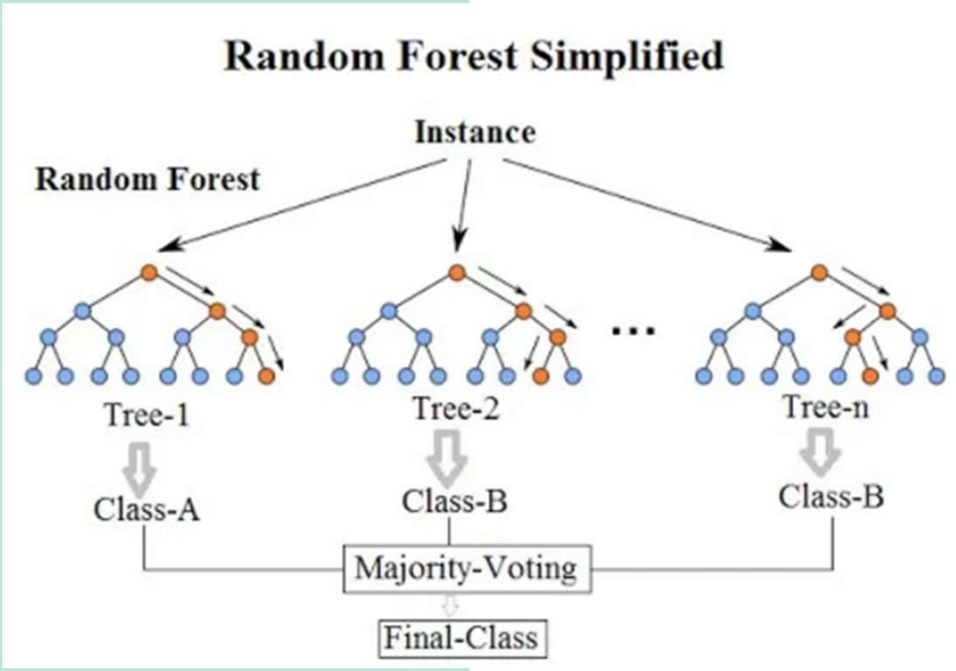
- Ensemble learning, in general, is a model that makes predictions based on a number of different models. By combining individual models, the ensemble model tends to be more flexible (less bias) and less data-sensitive (less variance).
- There are two types of Ensemble Method:

  Bagging: Training a bunch of individual models in a parallel way. Each model is trained by a random subset of the data

  Boosting: Training a bunch of individual models in a sequential way. Each individual model learns from mistakes made by the previous model

RANDOM FOREST?

- A random forest is an instance of ensemble learning where individual models are constructed using decision trees.
- Random forests combines the output of multiple decision trees to reach a single result.



HOW IT WORKS

- Random forest algorithms have three main hyperparameters includes include node size, the number of trees, and the number of features sampled which need to be set before training. The random forest classifier can be used to solve for regression or classification problems.
- The random forest algorithm is made up of a collection of decision trees, and each tree in the ensemble is comprised of a data sample drawn from a training set with replacement, called the bootstrap sample. Of that training sample, one-third of it is set aside as test data
- Another instance of randomness is then injected through feature bagging, adding more diversity to the dataset and reducing the correlation among decision trees. Depending on the type of problem, the determination of the prediction will vary.
- For a regression task, the individual decision trees will be averaged, and for a classification task, a majority vote—i.e. the most frequent categorical variable—will yield the predicted class.

THANK YOU