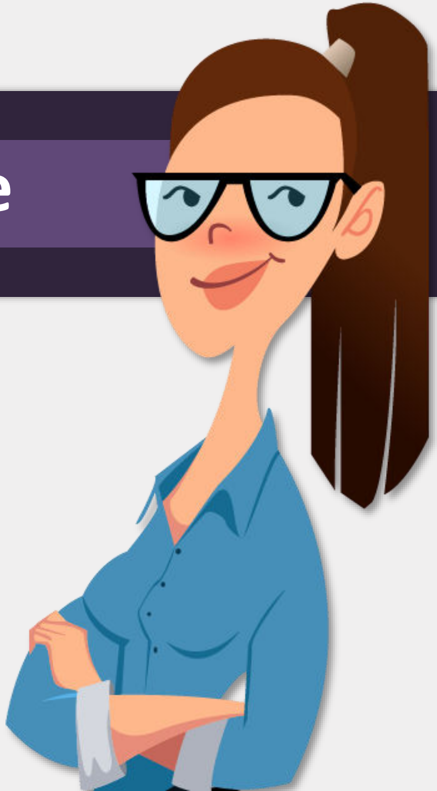
IntelliPaat

Python Certification Course



Decision Tree & Random Forest

Things you will learn after this Session

Supervised Learning: Classification

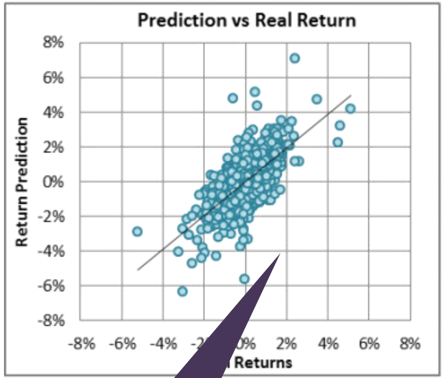
- Understanding Classification
- Decision Tree
- Demo on Decision Tree
- Bagging
- Random Forest
- Demo on Random Forest

What is Classification ?

"Classification is the process of **grouping things according to similar features** they share"

Classification vs Regression

Regression vs Classification



Continuous Values

Categorical Values

Types of Classification

Types of Classification

- Logistic Regression
- Decision Tree
- Random Forest
- K- Nearest Neighbour
- Naïve Byes

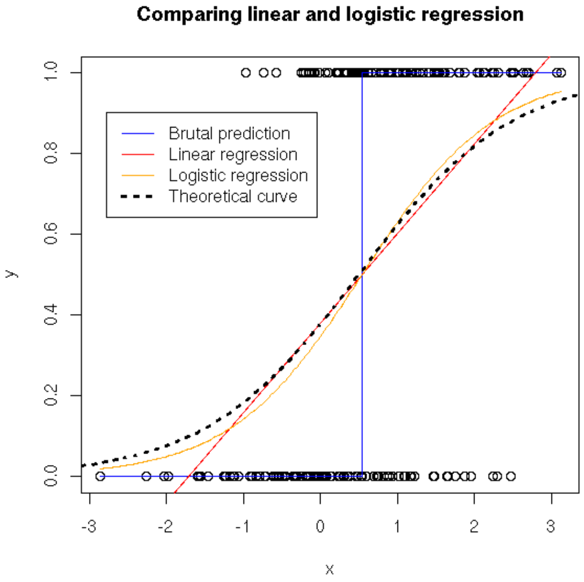
Types of Classification

- **Logistic Regression**
- Decision Tree
- Random Forest
- K- Nearest Neighbour
- Naïve Byes

"Logistic Regression is used when the dependent variable(target) is categorical."

For example,

Predict whether an email is spam (1) or (0)

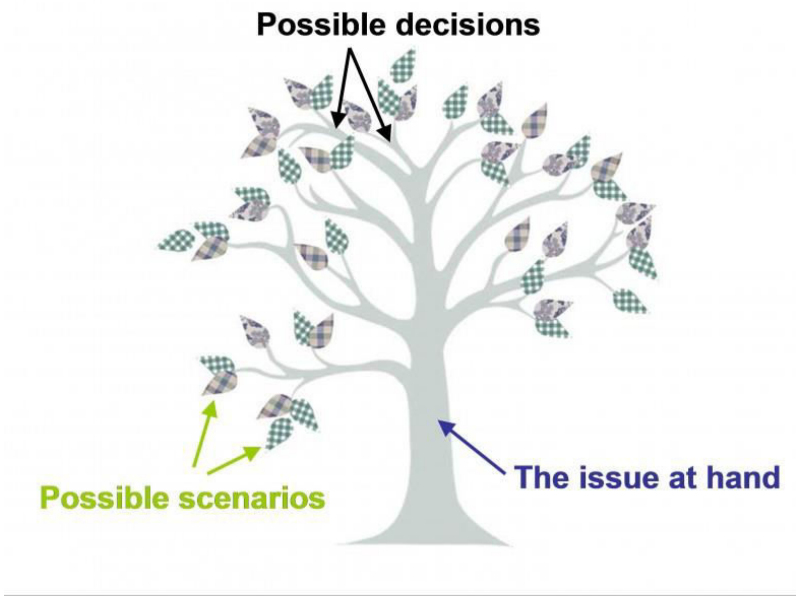


Types of Classification

- Logistic Regression
- **Decision Tree**
- Random Forest
- K- Nearest Neighbour
- Naïve Byes

"Graphical representation of all the possible solutions to a decision"

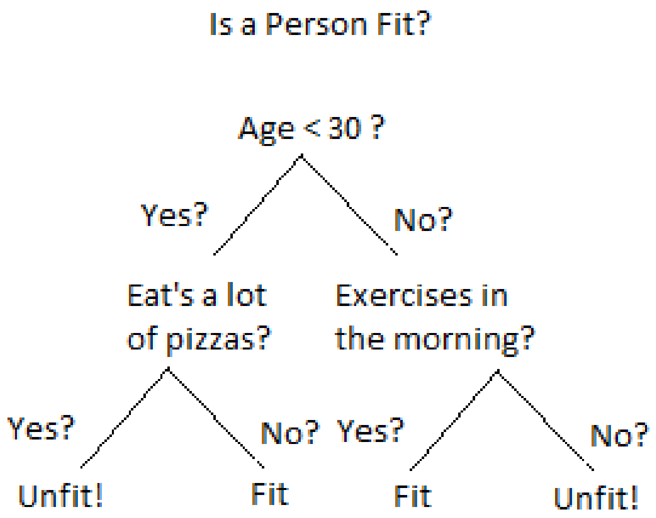
- Decisions are mainly based on some conditions
- Decision made can be easily explained



Types of Classification

- Logistic Regression
- **Decision Tree**
- Random Forest
- K- Nearest Neighbour
- Naïve Byes

For example,



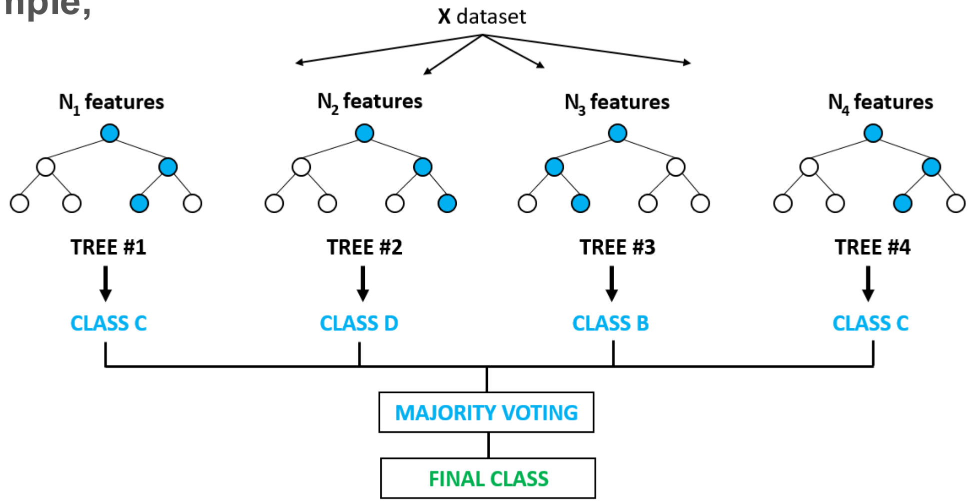
Types of Classification

- Logistic Regression
- Decision Tree
- Random Forest
- K- Nearest Neighbour
- Naïve Byes

"Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction"

- Correct decision trees' habit of overfitting their training set
- Trained with the "bagging" method

For example,



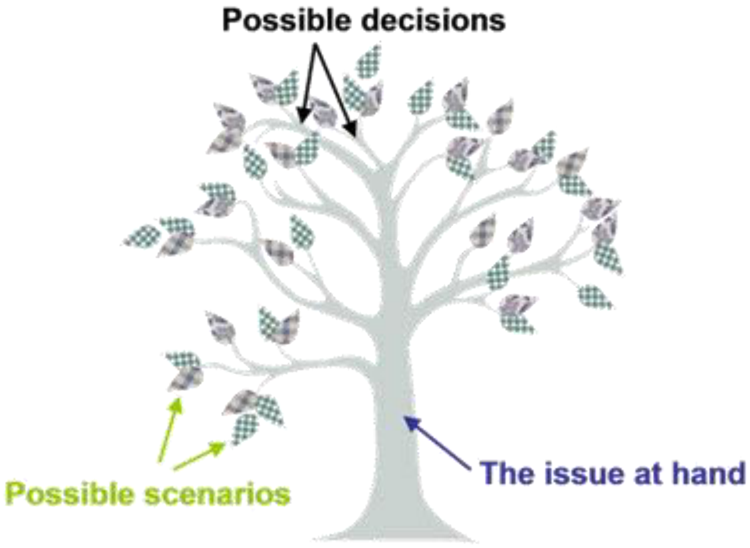
Understanding Decision Tree

What is Decision Tree?

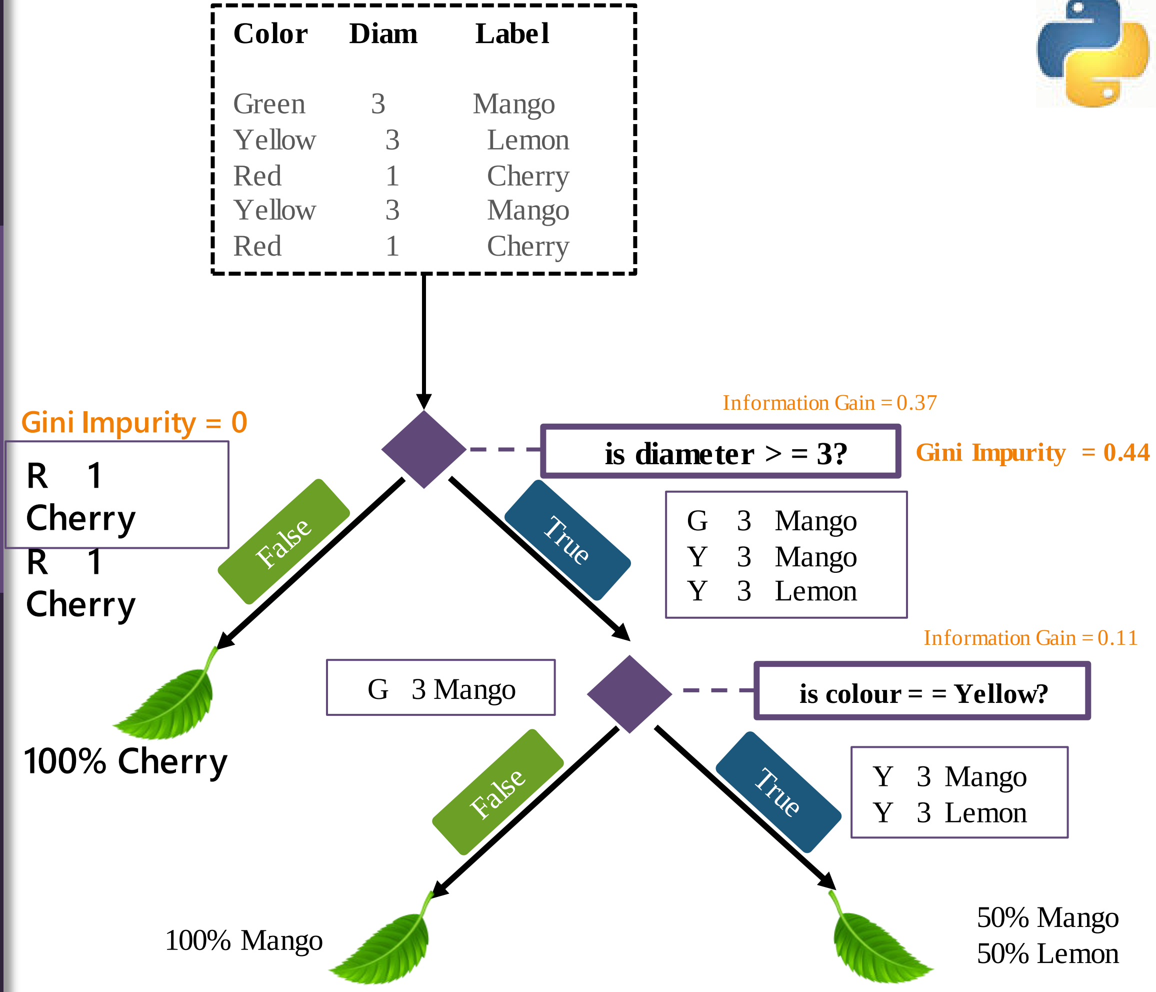
RECAP

> "Graphical representation of all the possible solutions to a decision"

- Decisions are mainly based on some conditions
- Decision made can be easily explained



Visualizing a Decision Tree



Decision Tree: Terminology

Pruning

Opposite of Splitting, basically removing unwanted branches from the tree

Branch/SubTree

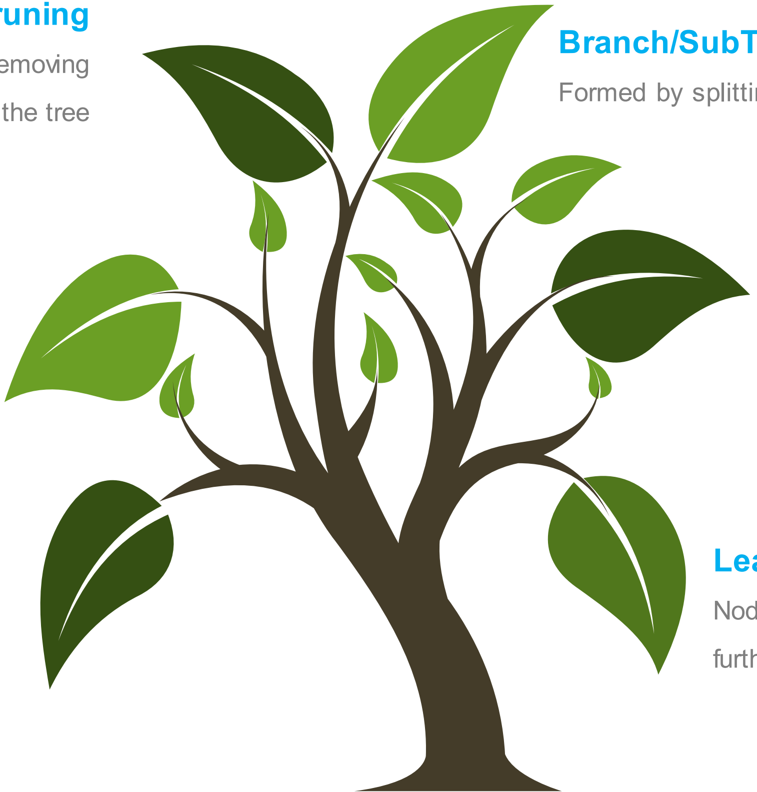
Formed by splitting the tree/node

Parent/Child Node

Root node is the parent node and all the other nodes branched from it is known as child node

Splitting

Dividing the root node/sub node into different parts on the basis of some condition.



Root Node

It represents the entire population or sample and this further gets divided into two or more homogenous sets.

Leaf Node

Node cannot be further segregated into further nodes

Creating a Decision Tree

This is our Dataset

outlook	temp.	humidity	windy	play
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
rainy	mild	high	false	yes
rainy	cool	normal	false	yes
rainy	cool	normal	true	no
overcast	cool	normal	true	yes
sunny	mild	high	false	no
sunny	cool	normal	false	yes
rainy	mild	normal	false	yes
sunny	mild	normal	true	yes
overcast	mild	high	true	yes
overcast	hot	normal	false	yes
rainy	mild	high	true	no

Creating a Decision Tree

How to decide, whether we will play or not?

outlook	temp.	humidity	windy	play
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
rainy	mild	high	false	yes
rainy	cool	normal	false	yes
rainy	cool	normal	true	no
overcast	cool	normal	true	yes
sunny	mild	high	false	no
sunny	cool	normal	false	yes
rainy	mild	normal	false	yes
sunny	mild	normal	true	yes
overcast	mild	high	true	yes
overcast	hot	normal	false	yes
rainy	mild	high	true	no

Creating a Decision Tree

Which one among them should you pick first?

outlook	temp.	humidity	windy	play
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
rainy	mild	high	false	yes
rainy	cool	normal	false	yes
rainy	cool	normal	true	no
overcast	cool	normal	true	yes
sunny	mild	high	false	no
sunny	cool	normal	false	yes
rainy	mild	normal	false	yes
sunny	mild	normal	true	yes
overcast	mild	high	true	yes
overcast	hot	normal	false	yes
rainy	mild	high	true	no

Creating a Decision Tree

Answer: Determine the attribute that best classifies the training data

outlook	temp.	humidity	windy	play
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
rainy	mild	high	false	yes
rainy	cool	normal	false	yes
rainy	cool	normal	true	no
overcast	cool	normal	true	yes
sunny	mild	high	false	no
sunny	cool	normal	false	yes
rainy	mild	normal	false	yes
sunny	mild	normal	true	yes
overcast	mild	high	true	yes
overcast	hot	normal	false	yes
rainy	mild	high	true	no

Creating a Decision Tree

But How do we choose the best attribute?

Or

How does a tree decide where to split?

outlook	temp.	humidity	windy	play
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
rainy	mild	high	false	yes
rainy	cool	normal	false	yes
rainy	cool	normal	true	no
overcast	cool	normal	true	yes
sunny	mild	high	false	no
sunny	cool	normal	false	yes
rainy	mild	normal	false	yes
sunny	mild	normal	true	yes
overcast	mild	high	true	yes
overcast	hot	normal	false	yes
rainy	mild	high	true	no

How do we split a Tree?

Entropy

Defines randomness in the data

It is a metric which measures the impurity

The first step to solve the problem of a decision tree

Information Gain

The information gain is the decrease in entropy after a dataset is split on the basis of an attribute. Constructing a decision tree is all about finding attribute that returns the highest information gain

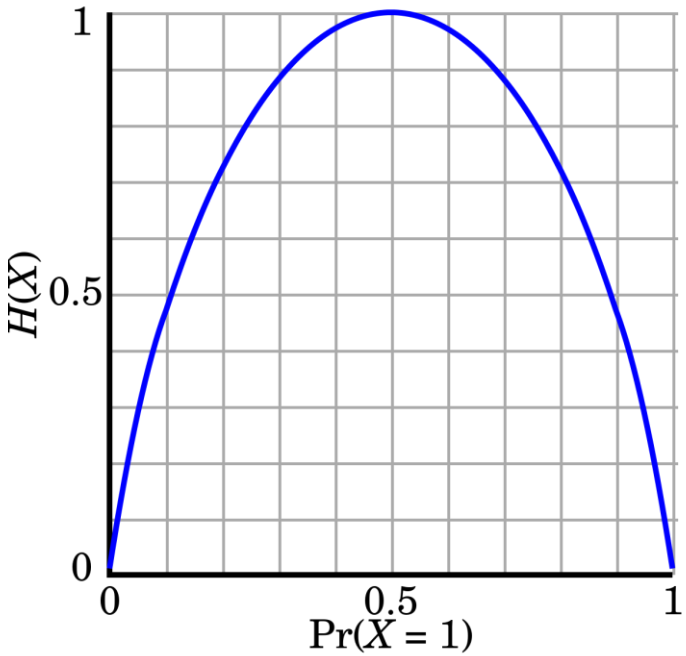
Gini Index

The measure of impurity (or purity) used in building decision tree in CART is Gini Index

Reduction in Variance

Reduction in variance is an algorithm used for continuous target variables (regression problems). The split with lower variance is selected as the criteria to split the population

Calculating Entropy



Entropy(s) = $-\,P(yes)\log_2 P(yes) - P(no)\log_2 P(no)$

Where,

- **S** is the **total sample space**,
- **P(yes)** is **probability of yes**

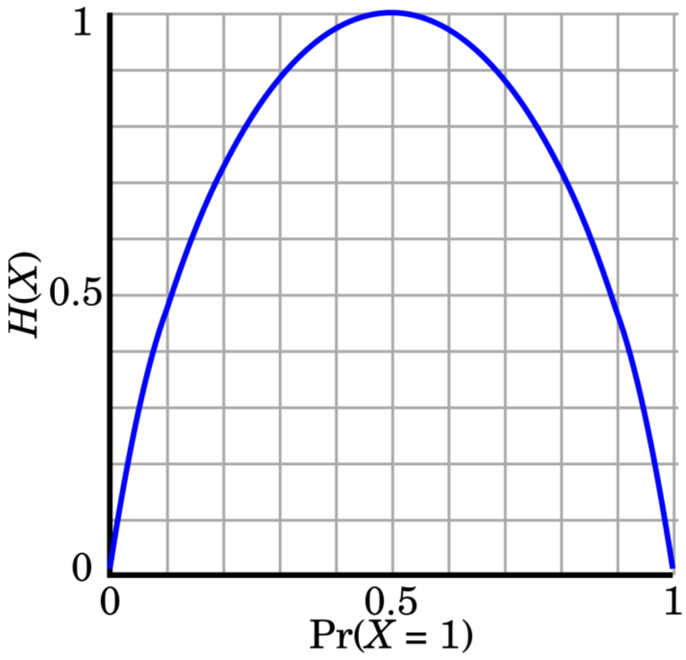
If number of *yes* = number of *no ie P(S) = 0.5*

$\Rightarrow$ *Entropy(s) = 1*

If it contains all yes or all no ie P(S) = 1 or 0

$\Rightarrow$ *Entropy(s) = 0*

Calculating Entropy



$E(S) = -P(Yes)\log_2 P(Yes) - P(no)\log_2 P(no)$

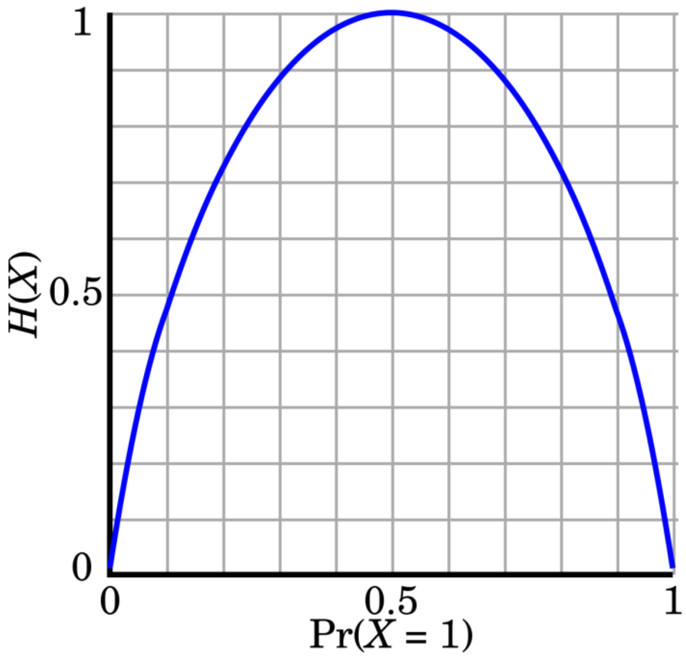
When P(Yes) =P(No) = 0.5 ie YES + NO = Total Sample(S)

$E(S) = -0.5 \log_2 0.5 - \boldsymbol{0.5} \log_2 0.5$

$E(S) = -0.5(\log_2 0.5 - \log_2 0.5)$

$E(S) = 1$

Calculating Entropy



$E(S) = -P(Yes) \log_2 P(Yes)$

When P(Yes) = 1 ie YES = Total Sample(S)

$E(S) = 1 \ \log_2 1$

$E(S) = 0$

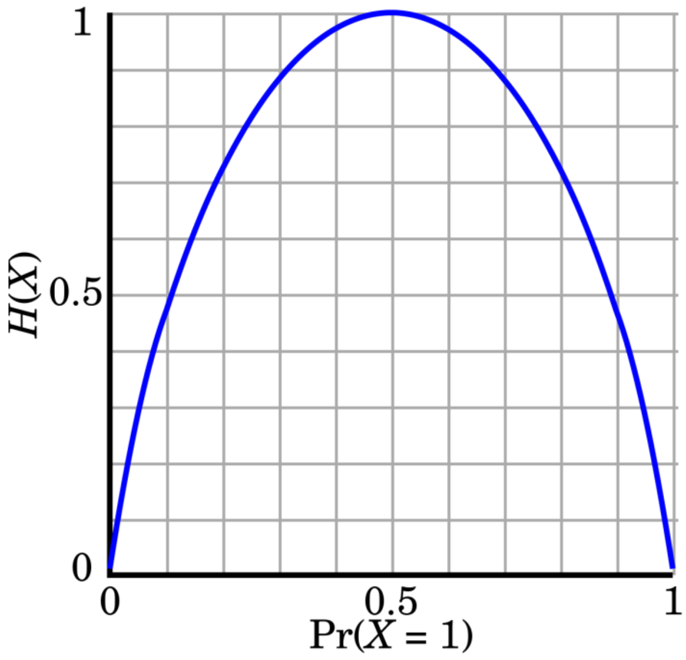
$E(S) = -P(No) \log_2 P(No)$

When P(No) = 1 ie No = Total Sample(S)

$E(S) = 1 \ \log_2 1$

$E(S) = 0$

Calculating Entropy



$E(S) = -P(Yes)\log_2 P(Yes)$

Total Sample(S) = all Yes = P(Yes) = 1

$E(S) = 1\ \log_2 1$

$E(S) = 0$

$E(S) = -P(No)\log_2 P(No)$

Total Sample(S) = all No = P(No) = 1

$E(S) = 1\ \log_2 1$

$E(S) = 0$

Calculating Entropy

	outlook	temp.	humidity	windy	play
D1	sunny	hot	high	false	no
D2	sunny	hot	high	true	no
D3	overcast	hot	high	false	yes
D4	rainy	mild	high	false	yes
D5	rainy	cool	normal	false	yes
D6	rainy	cool	normal	true	no
D7	overcast	cool	normal	true	yes
D8	sunny	mild	high	false	no
D9	sunny	cool	normal	false	yes
D10	rainy	mild	normal	false	yes
D11	sunny	mild	normal	true	yes
D12	overcast	mild	high	true	yes
D13	overcast	hot	normal	false	yes
D14	rainy	mild	high	true	no

14 instances: 9 YES and 5 NO

So we have the formula,

$$E(S) = -P(Yes)\log_2 P(Yes) - P(No)\log_2 P(No)$$

$$E(S) = -(9/14)^* \log_2 9/14 - (5/14)^* \log_2 5/14$$

$$E(S) = 0.41+0.53 = 0.94$$

Calculating Entropy

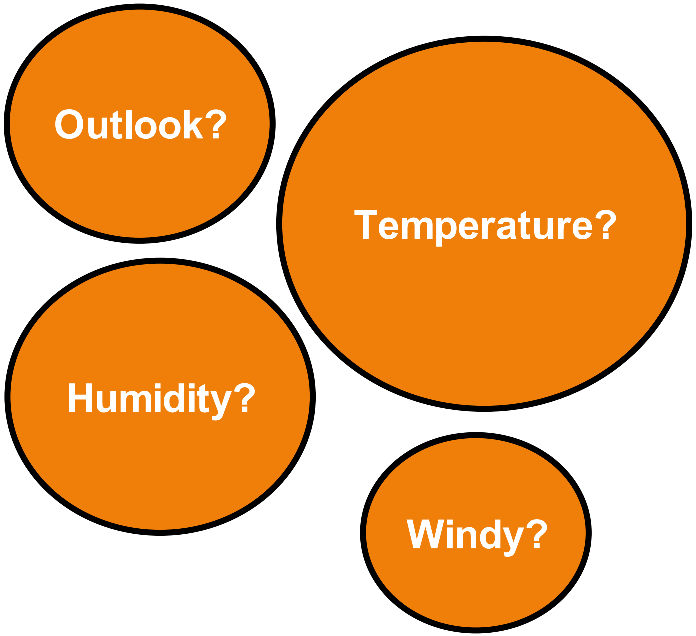
	outlook	temp.	humidity	windy	play
D1	sunny	hot	high	false	no
D2	sunny	hot	high	true	no
D3	overcast	hot	high	false	yes
D4	rainy	mild	high	false	yes
D5	rainy	cool	normal	false	yes
D6	rainy	cool	normal	true	no
D7	overcast	cool	normal	true	yes
D8	sunny	mild	high	false	no
D9	sunny	cool	normal	false	yes
D10	rainy	mild	normal	false	yes
D11	sunny	mild	normal	true	yes
D12	overcast	mild	high	true	yes
D13	overcast	hot	normal	false	yes
D14	rainy	mild	high	true	no

If S: total collection,

Information Gain = Entropy(S) – [(Weighted Avg) x Entropy(each feature)]

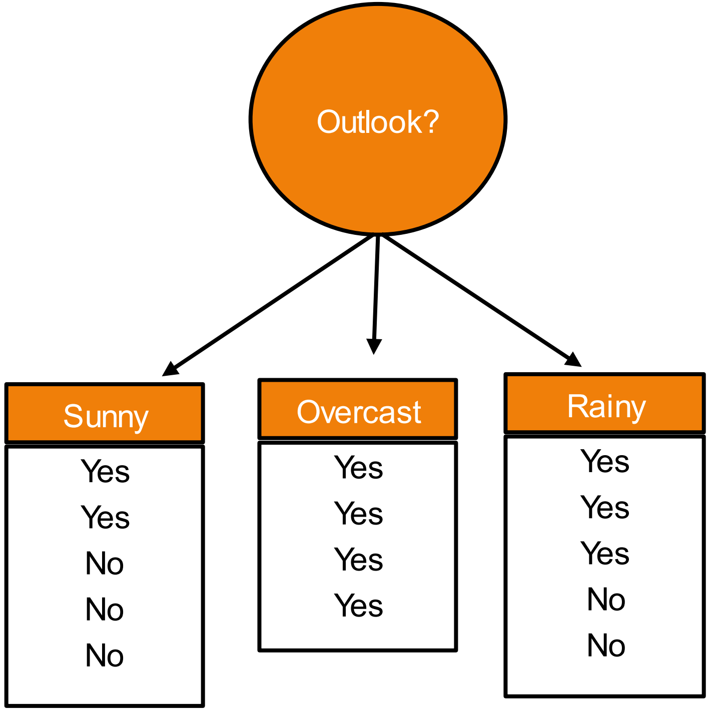
Calculating Entropy

	outlook	temp.	humidity	windy	play
D1	sunny	hot	high	false	no
D2	sunny	hot	high	true	no
D3	overcast	hot	high	false	yes
D4	rainy	mild	high	false	yes
D5	rainy	cool	normal	false	yes
D6	rainy	cool	normal	true	no
D7	overcast	cool	normal	true	yes
D8	sunny	mild	high	false	no
D9	sunny	cool	normal	false	yes
D10	rainy	mild	normal	false	yes
D11	sunny	mild	normal	true	yes
D12	overcast	mild	high	true	yes
D13	overcast	hot	normal	false	yes
D14	rainy	mild	high	true	no



Calculating Entropy

	outlook	temp.	humidity	windy	play
D1	sunny	hot	high	false	no
D2	sunny	hot	high	true	no
D3	overcast	hot	high	false	yes
D4	rainy	mild	high	false	yes
D5	rainy	cool	normal	false	yes
D6	rainy	cool	normal	true	no
D7	overcast	cool	normal	true	yes
D8	sunny	mild	high	false	no
D9	sunny	cool	normal	false	yes
D10	rainy	mild	normal	false	yes
D11	sunny	mild	normal	true	yes
D12	overcast	mild	high	true	yes
D13	overcast	hot	normal	false	yes
D14	rainy	mild	high	true	no



Calculating Entropy

	outlook	temp.	humidity	windy	play
D1	sunny	hot	high	false	no
D2	sunny	hot	high	true	no
D3	overcast	hot	high	false	yes
D4	rainy	mild	high	false	yes
D5	rainy	cool	normal	false	yes
D6	rainy	cool	normal	true	no
D7	overcast	cool	normal	true	yes
D8	sunny	mild	high	false	no
D9	sunny	cool	normal	false	yes
D10	rainy	mild	normal	false	yes
D11	sunny	mild	normal	true	yes
D12	overcast	mild	high	true	yes
D13	overcast	hot	normal	false	yes
D14	rainy	mild	high	true	no

$E(Outlook = Sunny) = -2/5 \log_2 2/5 - 3/5 \log_2 3/5 = 0.971$

$E(Outlook = Overcast) = -1 \log_2 1 - 0 \log_2 0 = 0$

$E(Outlook = Rainy) = -3/5 \log_2 3/5 - 2/5 \log_2 2/5 = 0.971$

Information from outlook,

$I(Outlook) = 5/14 \times 0.971 + 4/14 \times 0 + 5/14 \times 0.971 = 0.693$

Information gained from outlook,

$Gain(Outlook) = E(S) - I(Outlook)$

$0.94 - 0.693 = 0.247$

Calculating Entropy

	outlook	temp.	humidity	windy	play
D1	sunny	hot	high	false	no
D2	sunny	hot	high	true	no
D3	overcast	hot	high	false	yes
D4	rainy	mild	high	false	yes
D5	rainy	cool	normal	false	yes
D6	rainy	cool	normal	true	no
D7	overcast	cool	normal	true	yes
D8	sunny	mild	high	false	no
D9	sunny	cool	normal	false	yes
D10	rainy	mild	normal	false	yes
D11	sunny	mild	normal	true	yes
D12	overcast	mild	high	true	yes
D13	overcast	hot	normal	false	yes
D14	rainy	mild	high	true	no

Outlook:
Info 0.693
Gain: 0.940-0.693 $\boxed{0.247}$

Temperature:
Info 0.911
Gain: 0.940-0.911 0.029

Humidity:
Info 0.788
Gain: 0.940-0.788 0.152

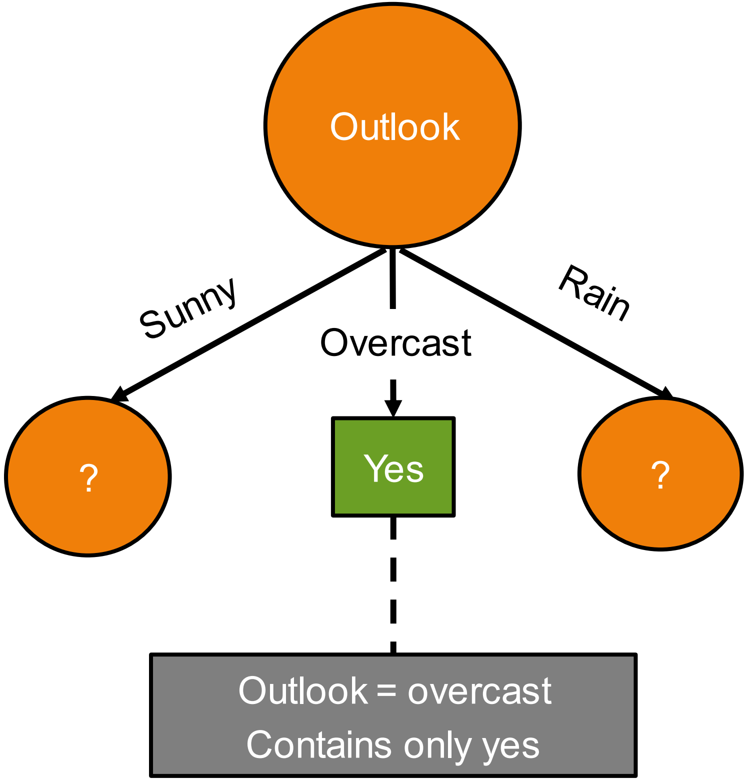
Since Max gain = 0.247,

ROOT Node:

OUTLOOK

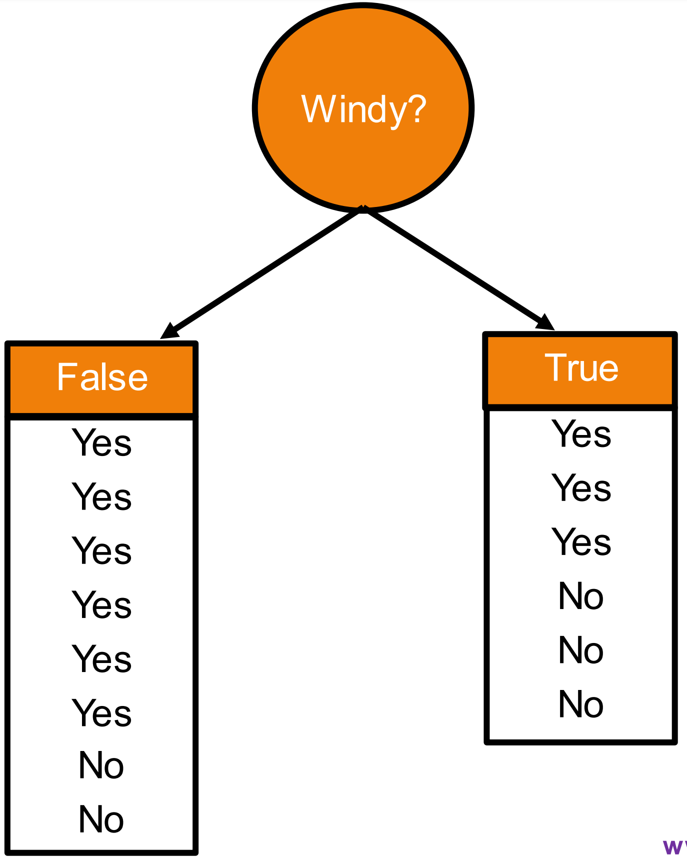
Calculating Entropy

	outlook	temp.	humidity	windy	play
D1	sunny	hot	high	false	no
D2	sunny	hot	high	true	no
D3	overcast	hot	high	false	yes
D4	rainy	mild	high	false	yes
D5	rainy	cool	normal	false	yes
D6	rainy	cool	normal	true	no
D7	overcast	cool	normal	true	yes
D8	sunny	mild	high	false	no
D9	sunny	cool	normal	false	yes
D10	rainy	mild	normal	false	yes
D11	sunny	mild	normal	true	yes
D12	overcast	mild	high	true	yes
D13	overcast	hot	normal	false	yes
D14	rainy	mild	high	true	no



Calculating Entropy

	outlook	temp.	humidity	windy	play
D1	sunny	hot	high	false	no
D2	sunny	hot	high	true	no
D3	overcast	hot	high	false	yes
D4	rainy	mild	high	false	yes
D5	rainy	cool	normal	false	yes
D6	rainy	cool	normal	true	no
D7	overcast	cool	normal	true	yes
D8	sunny	mild	high	false	no
D9	sunny	cool	normal	false	yes
D10	rainy	mild	normal	false	yes
D11	sunny	mild	normal	true	yes
D12	overcast	mild	high	true	yes
D13	overcast	hot	normal	false	yes
D14	rainy	mild	high	true	no



Calculating Entropy

	outlook	temp.	humidity	windy	play
D1	sunny	hot	high	false	no
D2	sunny	hot	high	true	no
D3	overcast	hot	high	false	yes
D4	rainy	mild	high	false	yes
D5	rainy	cool	normal	false	yes
D6	rainy	cool	normal	true	no
D7	overcast	cool	normal	true	yes
D8	sunny	mild	high	false	no
D9	sunny	cool	normal	false	yes
D10	rainy	mild	normal	false	yes
D11	sunny	mild	normal	true	yes
D12	overcast	mild	high	true	yes
D13	overcast	hot	normal	false	yes
D14	rainy	mild	high	true	no

$E(Windy = True) = 1$

$E(Windy = False) = 0.811$

Information from windy,

$I(Windy) = 8/14 \times 0.811 + 6/14 \times 1 = 0.892$

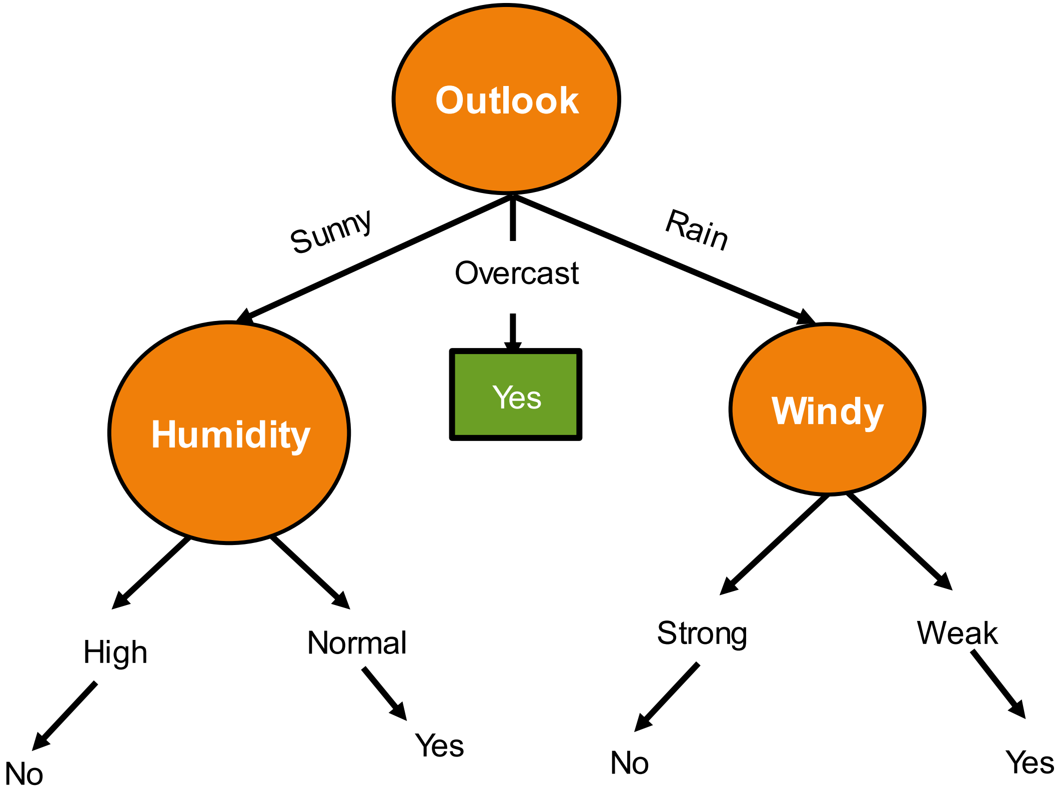
Information gained from windy,

$Gain(Windy) = E(S) - I(Windy)$

$0.94 - 0.892 = 0.048$

Calculating Entropy

	outlook	temp.	humidity	windy	play
D1	sunny	hot	high	false	no
D2	sunny	hot	high	true	no
D3	overcast	hot	high	false	yes
D4	rainy	mild	high	false	yes
D5	rainy	cool	normal	false	yes
D6	rainy	cool	normal	true	no
D7	overcast	cool	normal	true	yes
D8	sunny	mild	high	false	no
D9	sunny	cool	normal	false	yes
D10	rainy	mild	normal	false	yes
D11	sunny	mild	normal	true	yes
D12	overcast	mild	high	true	yes
D13	overcast	hot	normal	false	yes
D14	rainy	mild	high	true	no



Understanding Confusion Matrix

Things you will learn after this Session

Understanding Confusion Matrix

- What is Confusion Matrix?
- How to calculate a Confusion Matrix?
- Interpreting a Confusion Matrix
- Creating a Confusion Matrix in Python

Confusion Matrix

What is Confusion Matrix?

"The confusion matrix shows the ways in which your classification model is confused when it makes predictions."

- Is a summary of prediction results on a classification problem.
- Key to Confusion Matrix: Summarize the count value of correct and ir class)

Confusion Matrix

N=150	Predicted Fire	Predicted Fire (Negative)	Actual
Actual Alarm (True)	TP: 40	FP: 10	50
Actual Alarm (False)	FN: 5	TN: 95	100
Predicted Fire	45	105	150

Confusion Matrix

HOW TO calculate a Confusion Matrix?

- You need a test dataset or a validation dataset with expected outcome values.
- Make a prediction for each row in your test dataset.
- From the expected outcomes and predictions, count:
  - The number of correct predictions for each class.
  - The number of incorrect predictions for each class, organized by the class that was predicted.

Confusion Matrix

HOW TO calculate a Confusion Matrix?

Correct Prediction: 7/10

Accuracy: 70%

Expected	Predicted
man	woman
man	man
woman	woman
man	man
woman	man
woman	woman
woman	woman
man	man
man	woman
woman	woman

IntelliPaat

Confusion Matrix

HOW TO calculate a Confusion Matrix?

men classified as men: 3

women classified as women: 4

Expected	Predicted
man	woman
man	man
woman	woman
man	man
woman	man
woman	woman
woman	woman
man	man
man	woman
woman	woman

Confusion Matrix

HOW TO calculate a Confusion Matrix?

Expected	Predicted
man	woman
man	man
woman	woman
man	man
woman	man
woman	woman
woman	woman
man	man
man	woman
woman	woman

men classified as women: 2

woman classified as men: 1

Confusion Matrix

HOW TO calculate a Confusion Matrix?

	men	women
men	3	1
women	2	4

Expected	Predicted
man	woman
man	man
woman	woman
man	man
woman	man
woman	woman
woman	woman
man	man
man	woman
woman	woman

Confusion Matrix

HOW TO calculate a Confusion Matrix?

	men	women
men	3	1
women	2	4

- **Total actual men:** (3 + 2)
- **Total actual women:** (1 +4).
- **Total Correct values:** (3 + 4).

Conclusion: More errors while predicting men as women than predicting women as men.

Expected	Predicted
man	woman
man	man
woman	woman
man	man
woman	man
woman	woman
woman	woman
man	man
man	woman
woman	woman

Confusion Matrix

Interpreting a Confusion Matrix

	Predicted Fire	Predicted Fire
Alarm	True Positive	False Positive
No Alarm	False Negative	True Negative

- **True Positive:** Alarm goes on in case of fire
- **False Positive:** Alarm goes on but no fire
- **False Negative:** No Alarm in case of fire
- **True Negative:** No Alarm no Fire

Confusion Matrix

Interpreting a Confusion Matrix

Example:

N=150	Predicted Fire (Positive)	Predicted Fire (Negative)	Actual Alarm
Actual Alarm YES	TP: 40	FP: 10	50
Actual Alarm NO	FN: 5	TN: 95	100
Predicted Fire	45	105	150

- **True Positive:** Alarm goes on in case of fire
- **False Positive:** Alarm goes on but no fire
- **False Negative:** No Alarm in case of fire
- **True Negative:** No Alarm no Fire

Hands-on: Decision Tree(Regression)

Demo- Decision Tree

- We will be using the Boston dataset to implement a decision tree regression model
- This dataset contains information collected by the U.S Census Service concerning housing in the area of Boston Mass

crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat	medv
0.00632	18.0	2.31	0	0.5380	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.0
0.02731	0.0	7.07	0	0.4690	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.6
0.02729	0.0	7.07	0	0.4690	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
0.03237	0.0	2.18	0	0.4580	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
0.06905	0.0	2.18	0	0.4580	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	36.2
0.02985	0.0	2.18	0	0.4580	6.430	58.7	6.0622	3	222	18.7	394.12	5.21	28.7
0.08829	12.5	7.87	0	0.5240	6.012	66.6	5.5605	5	311	15.2	395.60	12.43	22.9
0.14455	12.5	7.87	0	0.5240	6.172	96.1	5.9505	5	311	15.2	396.90	19.15	27.1
0.21124	12.5	7.87	0	0.5240	5.631	100.0	6.0821	5	311	15.2	386.63	29.93	16.5

Implementing Decision Tree Regressor

Loading the required packages and the Boston dataset:

1

```
In [4]: #Loading required packages
        import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt

        #Boston dataset
        boston = pd.read_csv("boston.csv")
```

Having a glance at the 1st five records:

2

```
boston.head()
```

Out[6]:

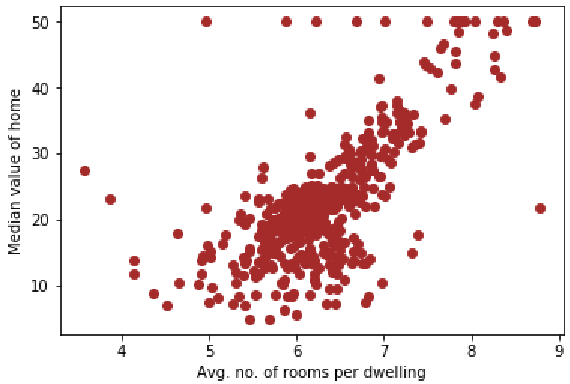
	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat	medv
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	36.2

Implementing Decision Tree Regressor

Building a scatter-plot between 'medv'& 'rm':

In [37]:
```
#scatterplot
plt.scatter(x=boston['rm'],y=boston['medv'],color='brown')
plt.xlabel('Avg. no. of rooms per dwelling')
plt.ylabel('Median value of home')
```

Out[37]: Text(0, 0.5, 'Median value of home')



Implementing Decision Tree Regressor

Getting the features & the target from the original Dataframe:

1

```
In [13]: #Getting the features and the target

         x=pd.DataFrame(boston['rm'])#features
         y=pd.DataFrame(boston['medv'])#target
```

Splitting the data into train & test sets:

2

```
In [14]: #Splitting into train & test

         from sklearn.model_selection import train_test_split
         x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.20)
```

Implementing Decision Tree Regressor

Building the regressor model:

```
In [15]: #Building the model

         from sklearn.tree import DecisionTreeRegressor
         regressor = DecisionTreeRegressor()
         regressor.fit(x_train, y_train)
```

```
Out[15]: DecisionTreeRegressor(criterion='mse', max_depth=None, max_features=None,
                    max_leaf_nodes=None, min_impurity_decrease=0.0,
                    min_impurity_split=None, min_samples_leaf=1,
                    min_samples_split=2, min_weight_fraction_leaf=0.0,
                    presort=False, random_state=None, splitter='best')
```

Implementing Decision Tree Regressor

Predicting the values:

1

```
In [16]:  #predicting the values
          y_pred = regressor.predict(x_test)
```

Finding RMSE value

2

```
In [17]:  #Finding the rmse value
          from sklearn.metrics import mean_squared_error
          mse=mean_squared_error(y_pred, y_test)
          rmse = np.sqrt(mse)
          rmse
```

```
Out[17]:  6.208531272889652
```

Implementing Decision Tree Regressor

Building 2nd model where features are 'lstat', 'rm' & 'age':

1

```
In [31]: #Getting the features and the target

         x=pd.DataFrame(boston[['rm','lstat','age']])#features
         y=pd.DataFrame(boston['medv'])#target
```

Splitting the data into train & test set:

2

```
In [33]: #Splitting into train & test

         from sklearn.model_selection import train_test_split
         x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.20)
```

Building the model on top of the train set:

3

```
#Building the model

from sklearn.tree import DecisionTreeRegressor
regressor = DecisionTreeRegressor()
regressor.fit(x_train, y_train)
```

Implementing Decision Tree Regressor

Predicting the values on test set:

1

```
In [35]: #predicting the values
         y_pred = regressor.predict(x_test)
```

Finding the RMSE value:

1

```
In [36]: #Finding the rmse value
         from sklearn.metrics import mean_squared_error
         mse=mean_squared_error(y_pred, y_test)
         rmse = np.sqrt(mse)
         rmse
```

```
Out[36]: 5.933413196201892
```

Hands-on: Decision Tree(Classifier)

Demo- Decision Tree

- We will be using the iris dataset to implement a decision tree regression model
- This dataset consists of 50 samples from each of three species of Iris (Iris setosa, Iris virginica, Iris versicolor)

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa
4.6	3.4	1.4	0.3	setosa
5.0	3.4	1.5	0.2	setosa
4.4	2.9	1.4	0.2	setosa
4.9	3.1	1.5	0.1	setosa
5.4	3.7	1.5	0.2	setosa

Implementing Decision Tree Classifier

Loading the required packages and the Iris dataset:

1

```
In [1]: #Loading required packages
        import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt

        #Boston dataset
        iris = pd.read_csv("iris.csv")
```

Having a glance at the 1st five records:

2

```
iris.head()
```

Out[2]:

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

Implementing Decision Tree Classifier

Extracting features & target from the original dataframe:

1

```
In [3]: x=pd.DataFrame(iris[['Sepal.Length','Sepal.Width','Petal.Length','Petal.Width']])
        y=iris['Species']
```

Splitting the data into train & test sets:

2

```
In [4]: #Dividing the data into train & test
        from sklearn.model_selection import train_test_split
        x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.30)
```

Building the model on the train set:

3

```
In [5]: #Building decision tree classifier
        from sklearn.tree import DecisionTreeClassifier
        classifier = DecisionTreeClassifier()
        classifier.fit(x_train, y_train)
```

```
Out[5]: DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
                    max_features=None, max_leaf_nodes=None,
                    min_impurity_decrease=0.0, min_impurity_split=None,
                    min_samples_leaf=1, min_samples_split=2,
                    min_weight_fraction_leaf=0.0, presort=False, random_state=None,
                    splitter='best')
```

Implementing Decision Tree Classifier

Predicting the values on the test set:

1

```
In [18]: y_pred = classifier.predict(x_test)
```

Creating confusion matrix for the model:

2

```
In [21]: from sklearn.metrics import confusion_matrix
         print(confusion_matrix(y_test, y_pred))
```

```
[[19  0  0]
 [ 0  9  2]
 [ 0  0 15]]
```

Finding the accuracy for the model:

3

```
In [22]: from sklearn.metrics import accuracy_score
         print(accuracy_score(y_test, y_pred))
```

```
0.9555555555555556
```

Understanding Random Forest

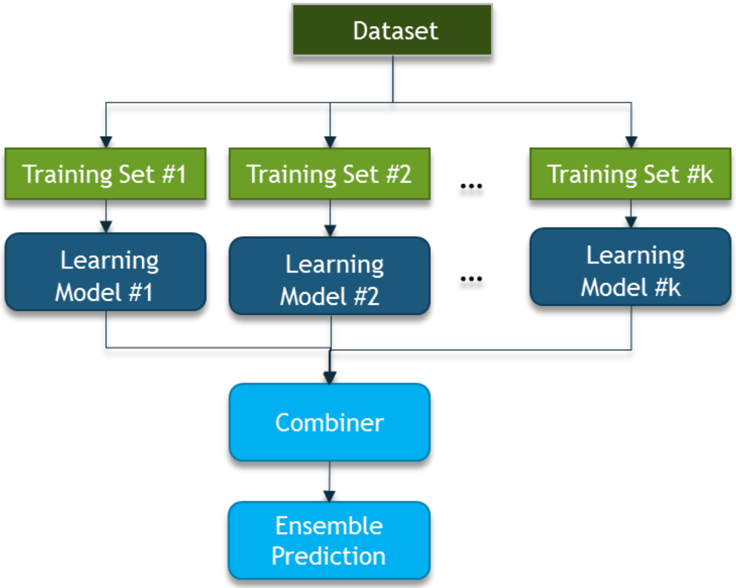
Did You Know That?

‘Decision trees have been around for a long time and also known to suffer from bias and variance. You will have a large bias with simple trees and a large variance with complex trees.’

Ensemble methods, which combines several decision trees to produce better predictive performance than utilizing a single decision tree

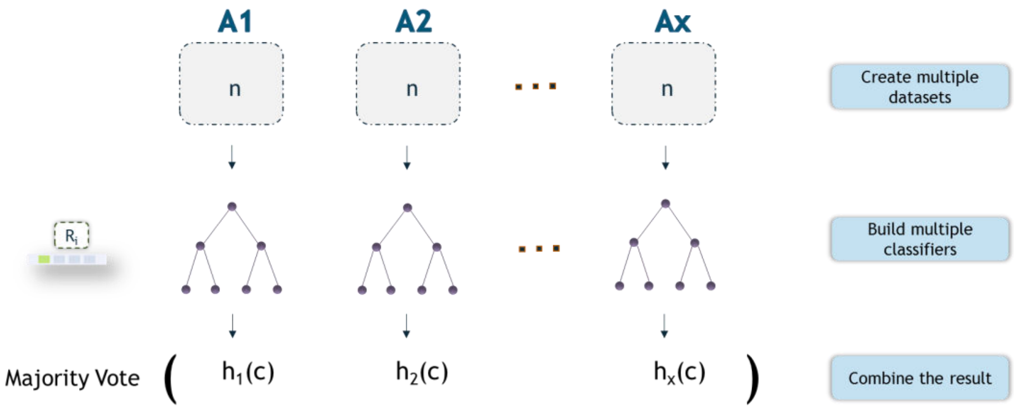
Did You Know That?

Ensemble methods, which combines several decision trees to produce better predictive performance than utilizing a single decision tree



Bagging

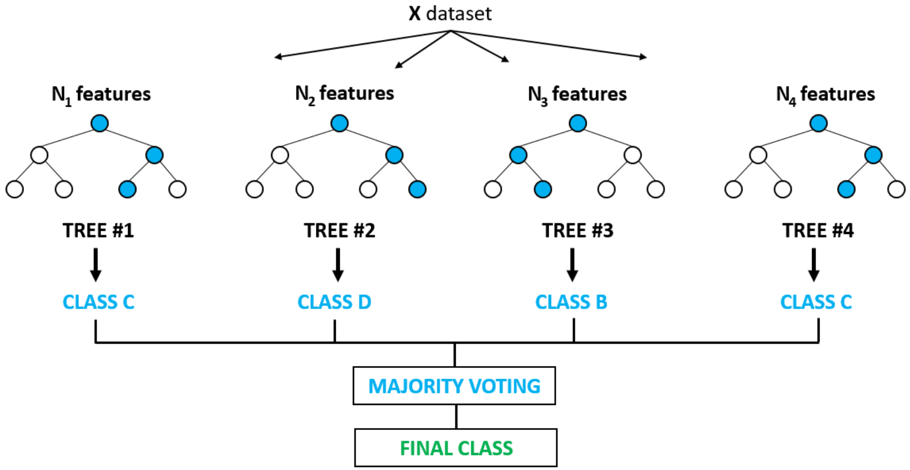
- It is a technique to perform ensemble decision trees
- Used when our goal is to reduce the variance of a decision tree
- Idea is to create several subsets of data from training sample chosen randomly with replacement
- Each collection of subset data is used to train their decision trees ending up with an ensemble of different models



Random Forest

"Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction"

- It is a type of ensemble learning method, where a group of weak models combine to form a powerful model
- Trained with the "bagging" method



Random Forest- How Does it Work?

The algorithm creates random subsets with random values from the complete dataset

From each subset, it creates a decision tree. Each tree is built from a sample

So, it creates multiple decision trees and then merge the results

Random Forest- How Does it Work?

Sampling is done on the training dataset. Every time, a new sample is chosen to build the tree. This introduction of randomness increases the bias and reduces the variances of the model

This prevents the overfitting of the model which is a serious concern in the case of decision trees

This yields much better performing generalized models

Random Forest- How Does it Work?

There are 2 levels of randomness:

At Row Level	Each decision tree gets a random sample of the training data
At Column Level	Each decision tree gets a random sample of columns. Not all trees get the same number of same column

Decision Tress Vs Random Forest

1

- If we input a training dataset with features and labels into a decision tree, it will formulate some set of rules, which will be used to make the predictions

- In comparison, the Random Forest algorithm randomly selects observations and features to build several decision trees and then averages the results

Decision Tress Vs Random Forest

2

- Deep Decision Trees might suffer from overfitting
- Random Forest prevents overfitting most of the time, by creating random subsets of the features and building smaller trees using these subsets

Important Hyperparameters in Random Forest

"The Hyperparameters in random forest are either used to increase the predictive power of the model or to make the model faster"

Increasing the Predictive Power

- *n_estimators* hyperparameter is the number of trees the algorithm builds before taking the maximum voting or taking averages of predictions.
- *max_features* is the maximum number of features Random Forest considers to split a node.
- *min_sample_leaf* determines the minimum number of leafs that are required to split an internal node.

Important Hyperparameters in Random Forest

"The Hyperparameters in random forest are either used to increase the predictive power of the model or to make the model faster"

Increasing the Models Speed

- *n_jobs* hyperparameter tells the engine how many processors it is allowed to use.
- *random_state* makes the model's output replicable.
- *oob_score* also called oob sampling, which is a random forest cross validation method. In this sampling, about one-third of the data is not used to train the model and can be used to evaluate its performance.

Random Forest-Example

This is our Weather Dataset

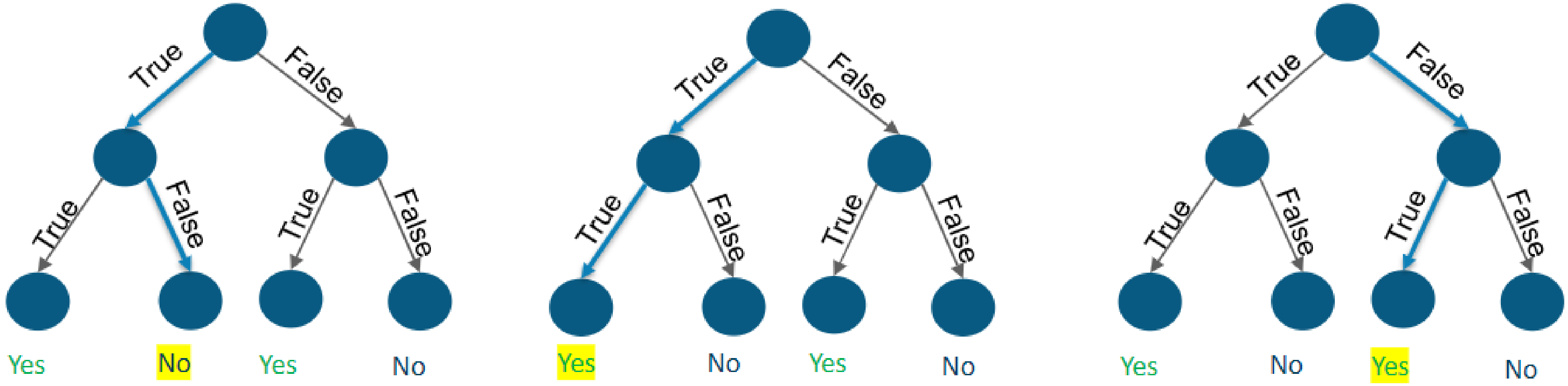
outlook	temp.	humidity	windy	play
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
rainy	mild	high	false	yes
rainy	cool	normal	false	yes
rainy	cool	normal	true	no
overcast	cool	normal	true	yes
sunny	mild	high	false	no
sunny	cool	normal	false	yes
rainy	mild	normal	false	yes
sunny	mild	normal	true	yes
overcast	mild	high	true	yes
overcast	hot	normal	false	yes
rainy	mild	high	true	no

Random Forest-Example

- The first step in Random Forest is that it will divide the data into smaller subsets
- Every subset need not be distinct, some may overlap
- For each subset a decision tree is made

outlook	temp.	humidity	windy	play
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
rainy	mild	high	false	yes
rainy	cool	normal	false	yes
rainy	cool	normal	true	no
overcast	cool	normal	true	yes
sunny	mild	high	false	no
sunny	cool	normal	false	yes
rainy	mild	normal	false	yes
sunny	mild	normal	true	yes
overcast	mild	high	true	yes
overcast	hot	normal	false	yes
rainy	mild	high	true	no

Random Forest-Example



- Output of each tree will be predicted and if any two decision trees predicted that game will happen while one predicted that it won't happen then on the basis of number of votes final output is selected, son in this case '*the game will happen*'

Hands-on: Random Forest

Demo- Random Forest

- We will be using the famous Iris Dataset, collected in the 1930's by Edgar Anderson.
- In this example, we are going to train a random forest classification algorithm to predict the class in the test data

	A	B	C	D	E	F
1	class	petal_len	petal_wid	sepal_len	sepal_width	
2	Iris-virgini	5.5	1.8	6.4	3.1	
3	Iris-virgini	5.9	2.3	6.8	3.2	
4	Iris-virgini	5.4	2.3	6.2	3.4	
5	Iris-virgini	4.8	1.8	6	3	
6	Iris-virgini	5.1	2.3	6.9	3.1	
7	Iris-virgini	5.6	2.4	6.3	3.4	
8	Iris-virgini	5.2	2.3	6.7	3	
9	Iris-virgini	6.7	2	7.7	2.8	
10	Iris-virgini	5.8	2.2	6.5	3	
11	Iris-virgini	5.3	1.9	6.4	2.7	
12	Iris-virgini	5	2	5.7	2.5	
13	Iris-virgini	5.1	1.9	5.8	2.7	

Implementing Random Forest in Python

Loading the iris dataset:

1

```
In [1]: #Loading the iris dataset
        from sklearn import datasets
        iris = datasets.load_iris()
```

Having a glance at the target & feature names:

2

```
In [3]: # print the label species(setosa, versicolor,virginica)
        print(iris.target_names)

        # print the names of the four features
        print(iris.feature_names)
```

```
['setosa' 'versicolor' 'virginica']
['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
```

Implementing Random Forest in Python

Printing 1st five records:

1

```
In [21]: # print the iris data (top 5 records)
         print(iris.data[0:5])
```

```
[[5.1 3.5 1.4 0.2]
 [4.9 3.  1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5.  3.6 1.4 0.2]]
```

Printing iris labels:

2

```
In [22]: # print the iris labels (0:setosa, 1:versicolor, 2:virginica)
         print(iris.target)
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]
```

Implementing Random Forest in Python

Creating Dataframe from the iris dataset:

In [5]:
```
# Creating a DataFrame of given iris dataset.
import pandas as pd
data=pd.DataFrame({
    'sepal length':iris.data[:,0],
    'sepal width':iris.data[:,1],
    'petal length':iris.data[:,2],
    'petal width':iris.data[:,3],
    'species':iris.target
})
data.head()
```

Out[5]:

	sepal length	sepal width	petal length	petal width	species
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

Implementing Random Forest in Python

Separating the columns into features & labels:

1

```
In [23]: # Import train_test_split function
         from sklearn.model_selection import train_test_split

         X=data[['sepal length', 'sepal width', 'petal length', 'petal width']]  # Features
         y=data['species']  # Labels
```

Dividing the data into train & test set:

2

```
In [ ]: # Split dataset into training set and test set
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3) # 70% training and 30% test
```

Implementing Random Forest in Python

Building the Random Forest model & Predicting the values:

1

```
In [7]: #Import Random Forest Model
        from sklearn.ensemble import RandomForestClassifier

        #Create a Gaussian Classifier
        clf=RandomForestClassifier(n_estimators=100)

        #Train the model using the training sets y_pred=clf.predict(X_test)
        clf.fit(X_train,y_train)

        y_pred=clf.predict(X_test)
```

Finding the accuracy of the model built:

1

```
In [8]: #Import scikit-learn metrics module for accuracy calculation
        from sklearn import metrics
        # Model Accuracy, how often is the classifier correct?
        print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.9111111111111111
```

Implementing Random Forest in Python

Making predictions for a single entry:

1

```
In [11]: #Making predictions for a single item
         clf.predict([[3, 5, 4, 2]])
```

```
Out[11]: array([0])
```

2

```
In [13]: #Making predictions for a single item
         clf.predict([[7, 2, 6, 2]])
```

```
Out[13]: array([2])
```

Implementing Random Forest in Python

Finding important features:

```
In [14]: #Finding important features
         import pandas as pd
         feature_imp = pd.Series(clf.feature_importances_,index=iris.feature_names).sort_values(ascending=False)
         feature_imp
```

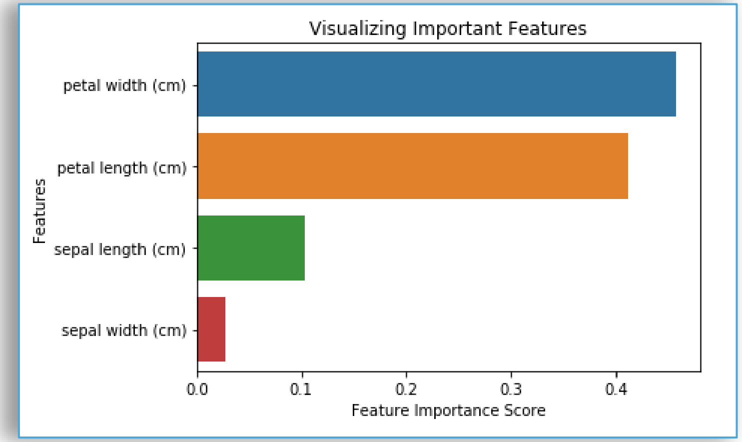
```
Out[14]: petal width (cm)     0.457331
         petal length (cm)    0.411440
         sepal length (cm)    0.103375
         sepal width (cm)     0.027854
         dtype: float64
```

Implementing Random Forest in Python

Visualizing important features:

```
In [17]: #Visualizing feature importance

         import matplotlib.pyplot as plt
         import seaborn as sns
         %matplotlib inline
         # Creating a bar plot
         sns.barplot(x=feature_imp, y=feature_imp.index)
         # Add labels to your graph
         plt.xlabel('Feature Importance Score')
         plt.ylabel('Features')
         plt.title("Visualizing Important Features")
         plt.show()
```



Implementing Random Forest in Python

Building model with only 'Petal length' & 'Petal width' as independent variables:

1

```
In [19]: #Generating model on selected features

         # Import train_test_split function
         from sklearn.model_selection import train_test_split
         # Split dataset into features and labels
         X=data[['petal length', 'petal width']]  # Removed feature "sepal length" & "Sepal Width"
         y=data['species']
         # Split dataset into training set and test set
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.70, random_state=5) # 70% training and 30% test
```

Predicting values and finding accuracy for new model:

2

```
In [20]: from sklearn.ensemble import RandomForestClassifier

         #Create a Gaussian Classifier
         clf=RandomForestClassifier(n_estimators=100)

         #Train the model using the training sets y_pred=clf.predict(X_test)
         clf.fit(X_train,y_train)

         # prediction on test set
         y_pred=clf.predict(X_test)

         #Import scikit-learn metrics module for accuracy calculation
         from sklearn import metrics
         # Model Accuracy, how often is the classifier correct?
         print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.9523809523809523
```

Demo- Random Forest

- We will be using the famous Iris Dataset, collected in the 1930's by Edgar Anderson.
- In this example, we are going to train a random forest classification algorithm to predict the class in the test data

	A	B	C	D	E	F
1	class	petal_len	petal_wid	sepal_len	sepal_width	
2	Iris-virgini	5.5	1.8	6.4	3.1	
3	Iris-virgini	5.9	2.3	6.8	3.2	
4	Iris-virgini	5.4	2.3	6.2	3.4	
5	Iris-virgini	4.8	1.8	6	3	
6	Iris-virgini	5.1	2.3	6.9	3.1	
7	Iris-virgini	5.6	2.4	6.3	3.4	
8	Iris-virgini	5.2	2.3	6.7	3	
9	Iris-virgini	6.7	2	7.7	2.8	
10	Iris-virgini	5.8	2.2	6.5	3	
11	Iris-virgini	5.3	1.9	6.4	2.7	
12	Iris-virgini	5	2	5.7	2.5	
13	Iris-virgini	5.1	1.9	5.8	2.7	

QUIZ

Quiz 1

Which type of machine learning type is used in spam mail classifier?

A	Supervised
B	Unsupervised
C	Reinforcement
D	All of the above

Answer 1

Which type of machine learning type is used in spam mail classifier?

A	Supervised
B	Unsupervised
C	Reinforcement
D	All of the above

Quiz 2

Which type of machine learning is used in self driving car?

A	Supervised
B	Unsupervised
C	Reinforcement
D	All of the above

Answer 2

Which type of machine learning is used in self driving car?

A	Supervised
B	Unsupervised
C	Reinforcement
D	All of the above

Quiz 3

A ___________ is a decision support tool that uses a tree-like graph or model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility.

A	Neural Networks
B	Decision Tree
C	Graph
D	Trees

Answer 3

A __________ is a decision support tool that uses a tree-like graph or model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility.

A	Neural Networks
B	Decision Tree
C	Graph
D	Trees

Quiz 4

Which of the following option is true about k-NN algorithm?

A	It can be used for classification
B	It can be used for regression
C	It can be used in both classification and regression
D	None

Answer 4

Which of the following option is true about k-NN algorithm?

A	It can be used for classification
B	It can be used for regression
C	It can be used in both classification and regression
D	None

Quiz 5

In Random Forest, which of the following is randomly selected?

A	Number of decision trees
B	Features to be considered when building a tree
C	Samples to be given to train individual tree in a forest
D	Both b and c

Answer 5

In Random Forest, which of the following is randomly selected?

A	Number of decision trees
B	Features to be considered when building a tree
C	Samples to be given to train individual tree in a forest
D	Both b and c

Thank You

www.intellipaat.com

CALL US NOW

India : +91-7847955955

US : 1-800-216-8930 (TOLL FREE)

sales@intellipaat.com