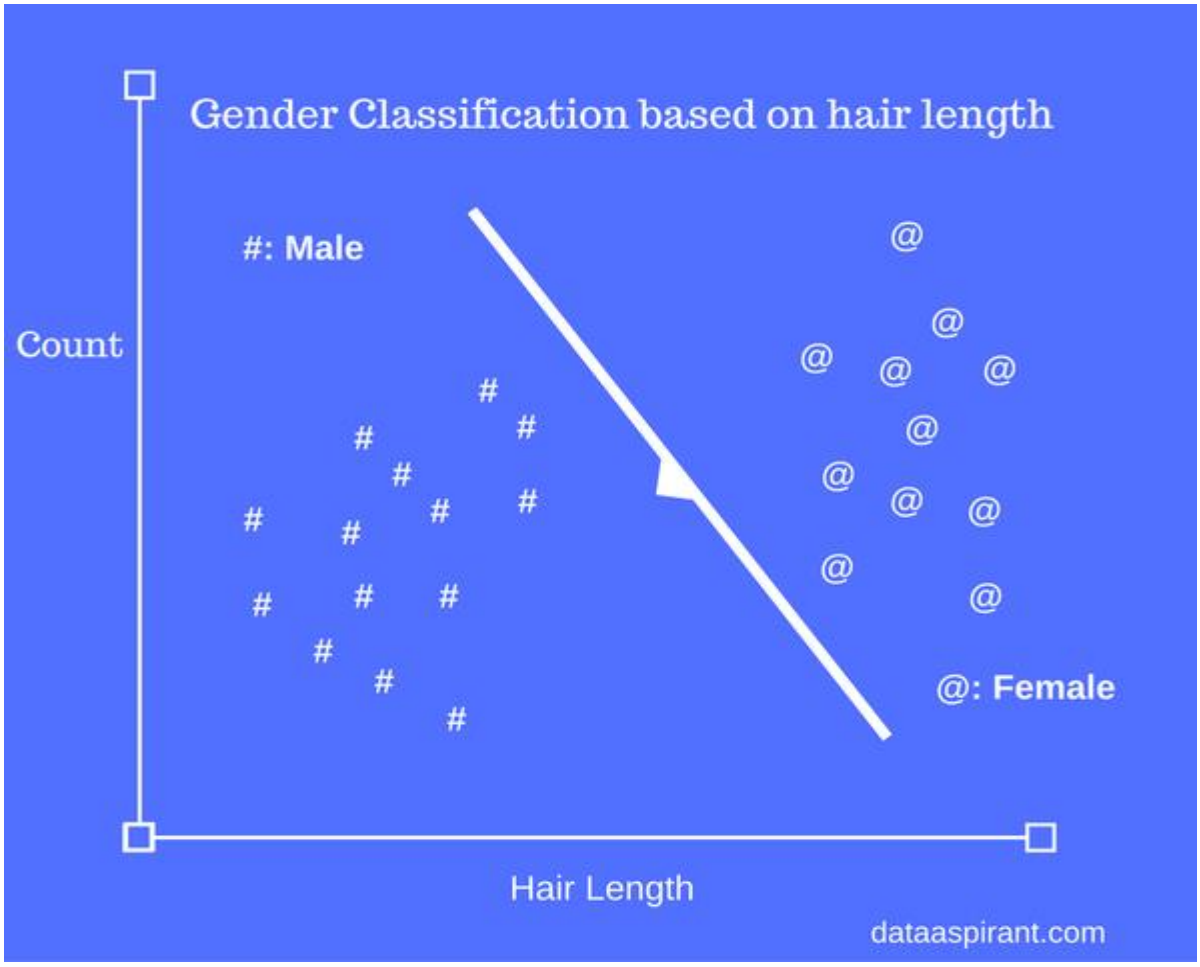
Experiment – 3 Implementation of classification algorithm

Name:	**Shaikh Shadab**	**Rollno**	**:**	**17DCO74**
Class :	**TE.CO**	**Batch**	**:**	**B3**

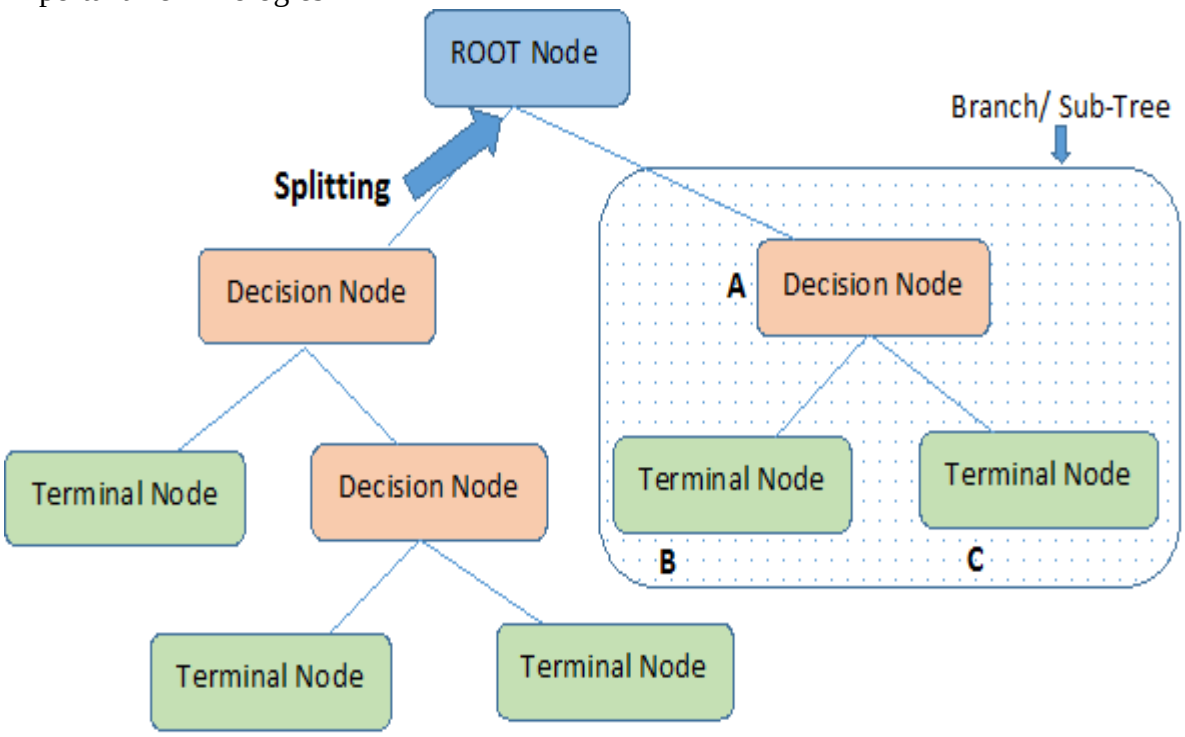
#Theory

In classification, the idea is to predict the target class by analysis the training dataset. This could be done by finding proper boundaries for each target class. In a general way of saying, Use the training dataset to get better boundary conditions which could be used to determine each target class. Once the boundary conditions determined, the next task is to predict the target class as we have said earlier. The whole process is known as classification.



- Decision Trees:
  Decision tree builds classification or regression models in the form of a tree structure. It breaks down a data set into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes. A decision node has two or more branches and a leaf node represents a classification or decision. The topmost decision node in a tree which corresponds to the best predictor called root node. Decision trees can handle both categorical and numerical data.

- Important Terminologies



Note:- A is parent node of B and C.

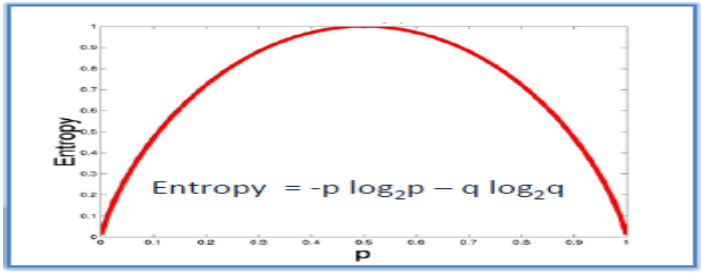
1. Root Node: It represents entire population or sample and this further gets divided into two or more homogeneous sets.
2. Splitting: It is a process of dividing a node into two or more sub-nodes.
3. Decision Node: When a sub-node splits into further sub-nodes, then it is called decision node.
4. Leaf/ Terminal Node: Nodes with no children (no further split) is called Leaf or Terminal node.
5. Pruning: When we reduce the size of decision trees by removing nodes (opposite of Splitting), the process is called pruning.
6. Branch / Sub-Tree: A sub section of decision tree is called branch or sub-tree.
7. Parent and Child Node: A node, which is divided into sub-nodes is called parent node of sub-nodes whereas sub-nodes are the child of parent node.

- ID3 Algorithm

  The core algorithm for building decision trees is called ID3. Developed by J. R. Quinlan, this algorithm employs a top-down, greedy search through the space of possible branches with no backtracking. ID3 uses Entropy and Information Gain to construct a decision tree.

- Entropy

  A decision tree is built top-down from a root node and involves partitioning the data into subsets that contain instances with similar values (homogenous). ID3 algorithm uses entropy to calculate the homogeneity of a sample. If the sample is completely homogeneous the entropy is zero and if the sample is an equally divided it has entropy of one.



Entropy = -0.5 $\log_2$0.5 – 0.5 $\log_2$0.5 = 1

Entropy controls how a Decision Tree decides to split the data. It actually effects how a Decision Tree draws its boundaries. To build a decision tree, we need to calculate two types of entropy using frequency tables as follows:

i. Entropy using the frequency table of one attribute:

$$E(S) = \sum_{i=1}^{c} - p_i \log_2 p_i$$

Play Golf	
Yes	No
9	5

Entropy(PlayGolf) = Entropy (5,9)
= Entropy (0.36, 0.64)
= - (0.36 $\log_2$ 0.36) - (0.64 $\log_2$ 0.64)
= 0.94

ii. Entropy using the frequency table of two attributes:

$$E(T, X) = \sum_{c \in X} P(c) E(c)$$

		Play Golf		
		Yes	No	
Outlook	Sunny	3	2	5
	Overcast	4	0	4
	Rainy	2	3	5
				14

E(PlayGolf, Outlook) = **P**(Sunny)***E**(3,2) + **P**(Overcast)***E**(4,0) + **P**(Rainy)***E**(2,3)
= (5/14)*0.971 + (4/14)*0.0 + (5/14)*0.971
= 0.693

- Information Gain

  The information gain is based on the decrease in entropy after a dataset is split on an attribute. Constructing a decision tree is all about finding attribute that returns the highest information gain (i.e., the most homogeneous branches).

  Step 1: Calculate entropy of the target.

$$\text{Entropy(PlayGolf)} = \text{Entropy}(5,9)$$
$$= \text{Entropy}(0.36, 0.64)$$
$$= -(0.36 \log_2 0.36) - (0.64 \log_2 0.64)$$
$$= 0.94$$

  Step 2: The dataset is then split on the different attributes. The entropy for each branch is calculated. Then it is added proportionally, to get total entropy for the split. The resulting entropy is subtracted from the entropy before the split. The result is the Information Gain, or decrease in entropy.

		Play Golf	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3
Gain = 0.247			

		Play Golf	
		Yes	No
Temp.	Hot	2	2
	Mild	4	2
	Cool	3	1
Gain = 0.029			

		Play Golf	
		Yes	No
Humidity	High	3	4
	Normal	6	1
Gain = 0.152			

		Play Golf	
		Yes	No
Windy	False	6	2
	True	3	3
Gain = 0.048			

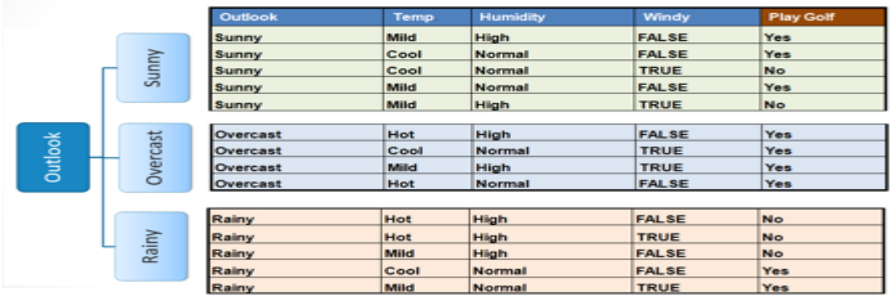
$$Gain(T, X) = Entropy(T) - Entropy(T, X)$$

G(PlayGolf, Outlook) = **E**(PlayGolf) – **E**(PlayGolf, Outlook)

= 0.940 – 0.693 = 0.247

  Step 3: Choose attribute with the largest information gain as the decision node, divide the dataset by its branches and repeat the same process on every branch.

★		Play Golf	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3
Gain = 0.247			



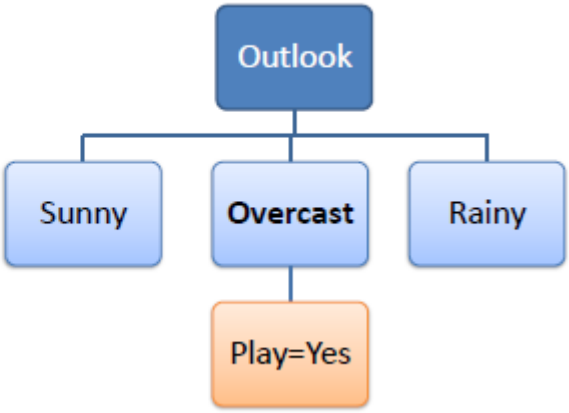
Outlook	Temp	Humidity	Windy	Play Golf
Sunny	Mild	High	FALSE	Yes
Sunny	Cool	Normal	FALSE	Yes
Sunny	Cool	Normal	TRUE	No
Sunny	Mild	Normal	FALSE	Yes
Sunny	Mild	High	TRUE	No

Overcast	Hot	High	FALSE	Yes
Overcast	Cool	Normal	TRUE	Yes
Overcast	Mild	High	TRUE	Yes
Overcast	Hot	Normal	FALSE	Yes

Rainy	Hot	High	FALSE	No
Rainy	Hot	High	TRUE	No
Rainy	Mild	High	FALSE	No
Rainy	Cool	Normal	FALSE	Yes
Rainy	Mild	Normal	TRUE	Yes

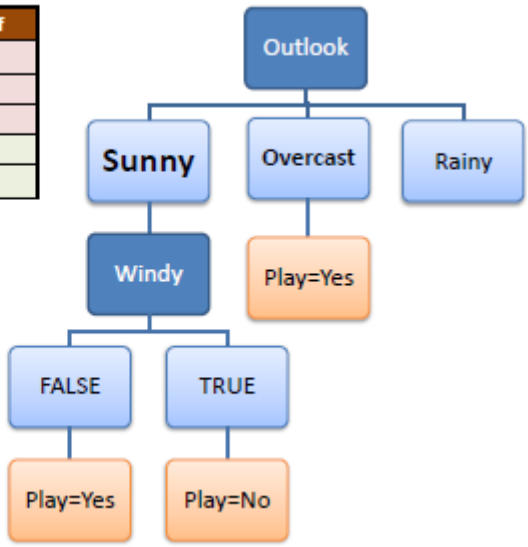
Step 4a: A branch with entropy of 0 is a leaf node.

Temp	Humidity	Windy	Play Golf
Hot	High	FALSE	Yes
Cool	Normal	TRUE	Yes
Mild	High	TRUE	Yes
Hot	Normal	FALSE	Yes



Step 4b: A branch with entropy more than 0 needs further splitting.

Temp	Humidity	Windy	Play Golf
Mild	High	FALSE	Yes
Cool	Normal	FALSE	Yes
Mild	Normal	FALSE	Yes
Cool	Normal	TRUE	No
Mild	High	TRUE	No



Step 5: The ID3 algorithm is run recursively on the non-leaf branches, until all data is classified.

- Decision Tree to Decision Rules

A decision tree can easily be transformed to a set of rules by mapping from the root node to the leaf nodes one by one.

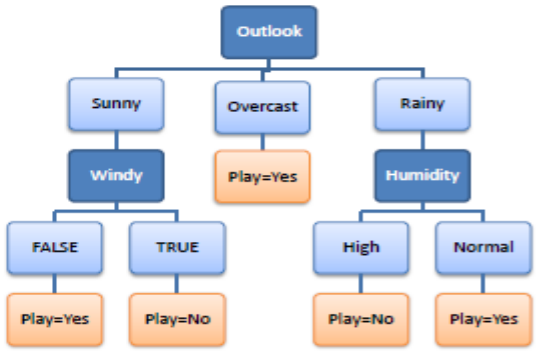
R_1: IF (Outlook=Sunny) AND (Windy=FALSE) THEN Play=Yes

R_2: IF (Outlook=Sunny) AND (Windy=TRUE) THEN Play=No

R_3: IF (Outlook=Overcast) THEN Play=Yes

R_4: IF (Outlook=Rainy) AND (Humidity=High) THEN Play=No

R_5: IF (Outlook=Rain) AND (Humidity=Normal) THEN Play=Yes



- Gini Index

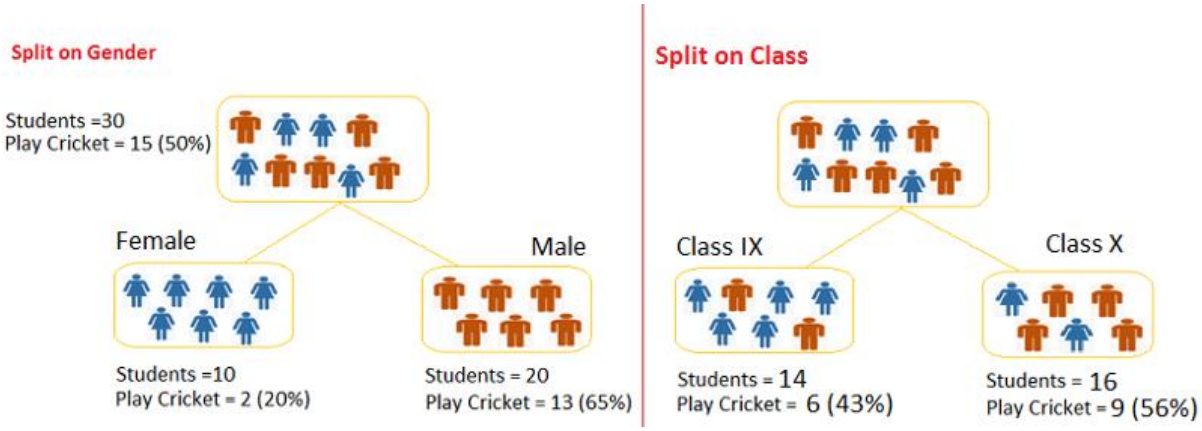
  It says, if we select two items from a population at random then they must be of same class and probability for this is 1 if population is pure.

  1. It works with categorical target variable "Success" or "Failure".
  2. It performs only Binary splits
  3. Higher the value of Gini higher the homogeneity.
  4. CART (Classification and Regression Tree) uses Gini method to create binary splits.

- Steps to Calculate Gini for a split

  1. Calculate Gini for sub-nodes, using formula sum of square of probability for success and failure (p^2+q^2).
  2. Calculate Gini for split using weighted Gini score of each node of that split

  Example:—Referring to example where we want to segregate the students based on target variable (playing cricket or not). In the snapshot below, we split the population using two input variables Gender and Class. Now, we want to identify which split is producing more homogeneous sub-nodes using Gini index.



Split on Gender:

1. Gini for sub-node Female = (0.2)*(0.2)+(0.8)*(0.8)=0.68
2. Gini for sub-node Male = (0.65)*(0.65)+(0.35)*(0.35)=0.55
3. Weighted Gini for Split Gender = (10/30)*0.68+(20/30)*0.55 = 0.59

Similar for Split on Class:

1. Gini for sub-node Class IX = (0.43)*(0.43)+(0.57)*(0.57)=0.51
2. Gini for sub-node Class X = (0.56)*(0.56)+(0.44)*(0.44)=0.51
3. Weighted Gini for Split Class = (14/30)*0.51+(16/30)*0.51 = 0.51

Above, we can see that Gini score for Split on Gender is higher than Split on Class, hence, the node split will take place on Gender.

#Source Code

```
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the dataset
dataset = pd.read_csv('Social_Network_Ads.csv')
X = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:, 4].values

# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)

# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

# Fitting classifier to the Training set
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(criterion='entropy', random_state = 0)
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)

# Visualising the Training set results
from matplotlib.colors import ListedColormap
X_set, y_set = X_train, y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, step
= 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(),
X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
```

```
            c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('Decision Tree Classification (Training set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()

# Visualising the Test set results
from matplotlib.colors import ListedColormap
X_set, y_set = X_test, y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, step
= 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1,                X2,                classifier.predict(np.array([X1.ravel(),
X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('Decision Tree Classification (Test set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show
```

#Output (Training set)

#Output (Test set)

#Conclusion

From this experiment we have gone through the concept of classification algorithm of machine learning. We have seen how classification can be implemented using decision trees. We have also been exposed to important terminologies related to decision trees and have studies in detailed two important algorithms of decision trees that is ID3 which uses information gain and entropy and Gini index which works on binary splits.