DAY NO: 2

ASSIGNMENT ON MACHINE LEARNING AND DEEP LEARNING MODEL WITH CODE

PRESENTED BY ST AIML :

AADIL RAYEEN [42]

PRIYANSHU SAH [43]

HARSH SHARMA [46]

PRASHANT SHARMA [48]

UNDER THE GUIDANCE OF

MS. PRANITI PATIL,

ASSISTANT PROFESSOR

AIML-DEPT

Deep Learning Model and Algorithm

Multilayer Perceptron (MLP) Algorithm

- In Deep Learning, the main aim is to mimic the human brain.
- How to mimic?
- To mimic with the help of neurones forming neural network as likehuman brain.
- Deep learning requires huge amount of data (BIG DATA)

Study Time	Playing Time	SleepingTime	Result
4 hr	6 hr	7 hr	0
2 hr	5 hr	8 hr	0
7 hr	3 hr	6 hr	1
5 hr	2 hr	7 hr	1

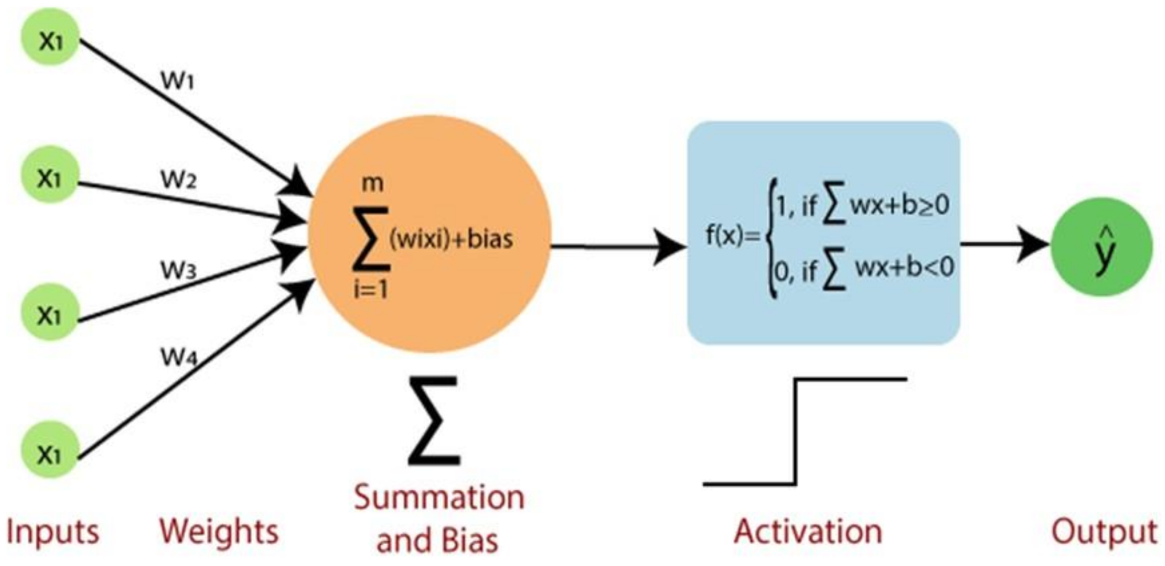
- The above table is a binary classification problem.
- Our aim is to predict whether the student will pass or fail.
- How to predict?

Perceptron :

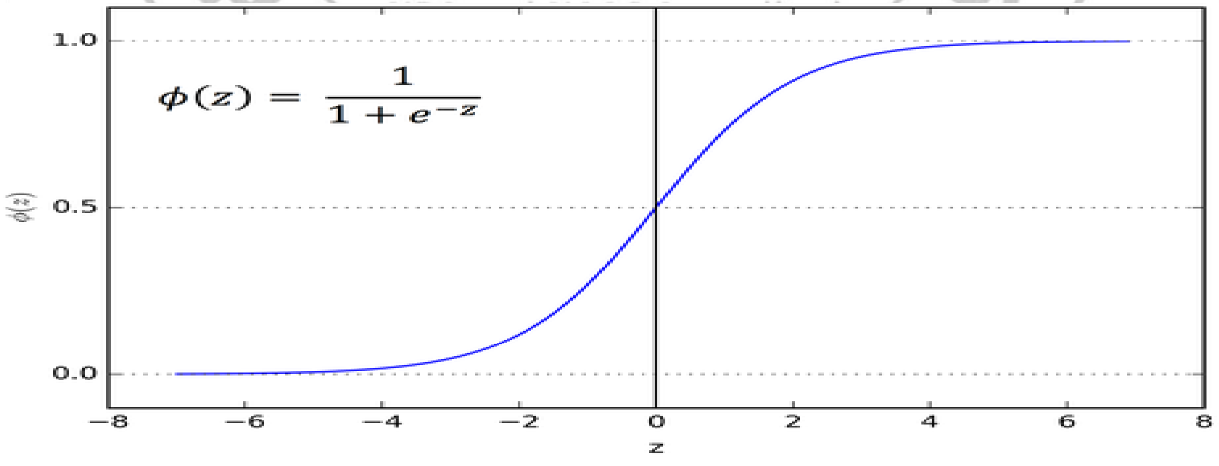
- Here inputs are the independent features.
- Weights are the computer assumed based numbers taken asrandomly.
- Now summation added with bias -> x1w1+x2w2+x3w3+….xnwn(depending on the number of features).
- What is bias?
- Bias is a constant number in the equation assumed randomly likeintercept (c) in the linear regression.
- Summation and activation happens in the hidden layer.
- Number of hidden layers dependent on the number of inputsgiven.

- Training of the data happens in the hidden layer.
- What actually happens in the hidden layer?
- As soon as the inputs comes to the hidden layer, the summationof the product of inputs and weights get assigned.

Value of the weights decide what level it should activated to passthe signal to the neurone.


X1
X1
X1
X1
W1
W2
W3
W4
$\sum_{i=1}^{m}$ (wixi)+bias
f(x)= {1, if ∑ wx+b≥0 ; 0, if ∑ wx+b<0
ŷ
Inputs
Weights
Summation and Bias
Activation
Output

Activation Function :



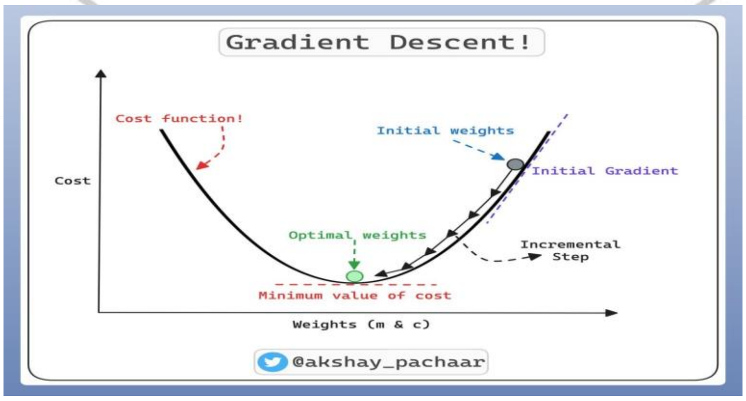
Sigmoid Function

- Sigmoid Activation Function is used for binary classification.
- The output of the sigmoid function is either 0 or 1.
- Sigmoid function can give the value between 0 to 1, if it >= 0.5 itis taken as 1 and if it is < 0.5 it is taken as 0.
- If it is not equal to the desired output, backward propagation happens.
- What is backward propagation?
- Backward propagation is the process of going back in the algorithm and changing the value of the weights.
- The aim of the backward propagation is to update the value ofthe weights.
- And then forward propagation happens to get the desired output.
- Backward propagation and forward propagation happens untilwe get the accurate output.
- Error of the output = Actual output - Predicted output
- Error of the output should be closer to 0.
- How to handle the errors?

$$MSE = \frac{1}{N}\sum_{i}^{N}(Y_i - \hat{Y}_i)^2$$

Loss function

- While happening backward propagation, optimisers are used to update the weights.
- Gradient Descent is a kind of Optimiser in deep learning.



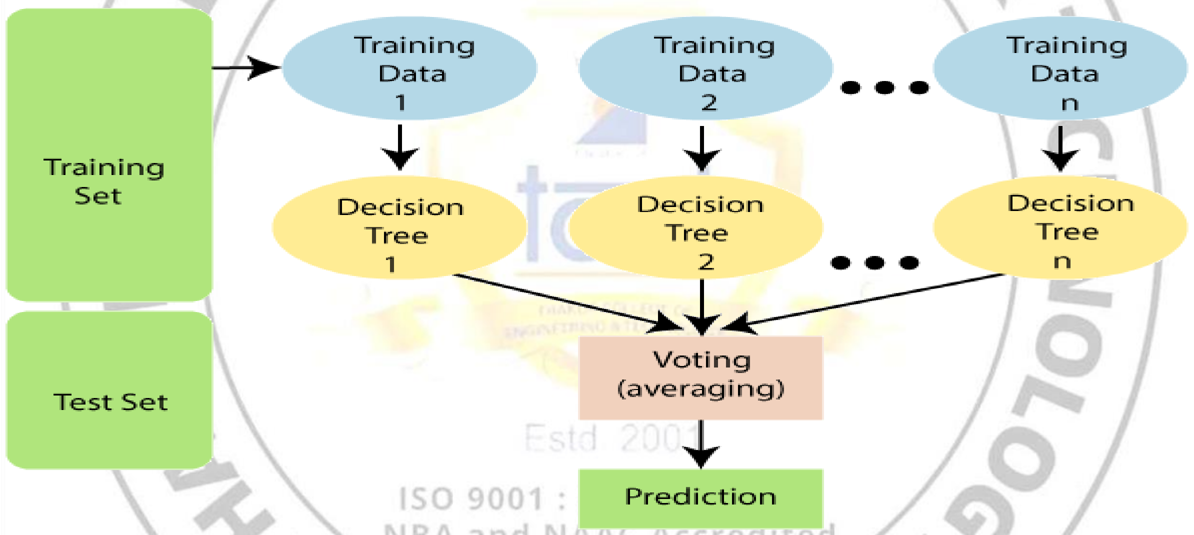
MACHINE LEARNING

RANDOM FOREST ALGORITHM

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of **ensemble learning,** *which is a process of* combining multiple classifiers to solve a complex problem and improve the model's performance.

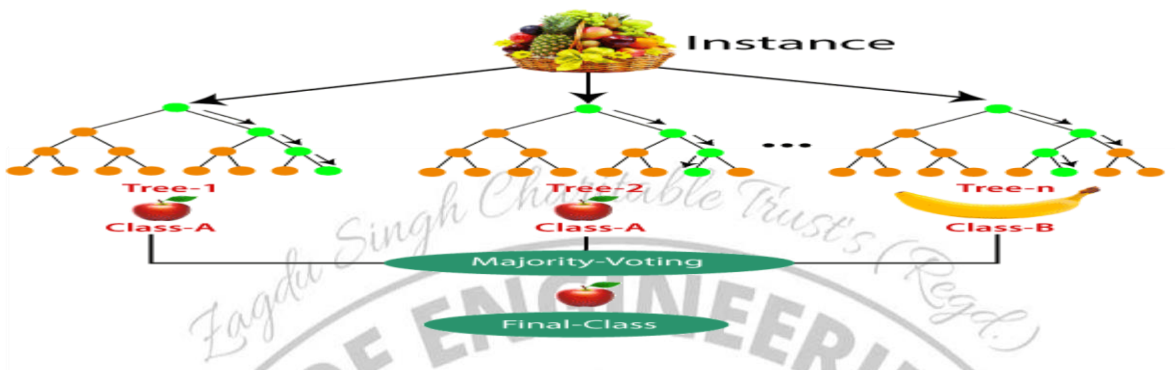
The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

The below diagram explains the working of the Random Forest algorithm:



The working of the algorithm can be better understood by the below example:

Example: Suppose there is a dataset that contains multiple fruit images. So, this dataset is given to the Random forest classifier. The dataset is divided into subsets and given to each decision tree. During the training phase, each decision tree produces a prediction result, and when a new data point occurs, then based on the majority of results, the Random Forest classifier predicts the final decision. Consider the below image:



Application:

There are mainly four sectors where Random forest mostly used:

1. **Banking:** Banking sector mostly uses this algorithm for the identification of loan risk.
2. **Medicine:** With the help of this algorithm, disease trends and risks of the disease can be identified.
3. **Land Use:** We can identify the areas of similar land use by this algorithm.
4. **Marketing:** Marketing trends can be identified using this algorithm.

Advantages:

- Random Forest is capable of performing both Classification and Regression tasks.
- It is capable of handling large datasets with high dimensionality.
- It enhances the accuracy of the model and prevents the overfitting issue.

```
# importing libraries
import numpy as nm
import matplotlib.pyplot as mtp
import pandas as pd

#importing datasets
data_set= pd.read_csv('user_data.csv')

#Extracting Independent and dependent Variable
x= data_set.iloc[:, [2,3]].values
y= data_set.iloc[:, 4].values

# Splitting the dataset into training 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
st_x= StandardScaler()
x_train= st_x.fit_transform(x_train)
x_test= st_x.transform(x_test)
```

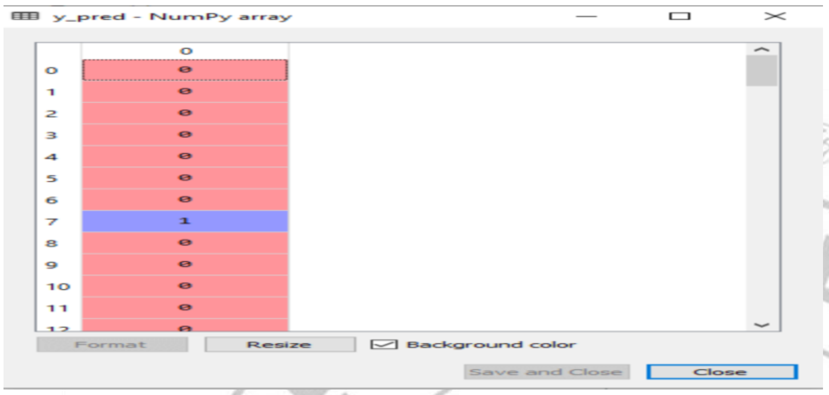
```
#Fitting Decision Tree classifier to the training set
from sklearn.ensemble import RandomForestClassifier
classifier= RandomForestClassifier(n_estimators= 10, criterion="entropy")
classifier.fit(x_train, y_train)
```

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='entropy',
                       max_depth=None, max_features='auto', 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, n_estimators=10,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

```
#Predicting the test set result
y_pred= classifier.predict(x_test)
```

Output:

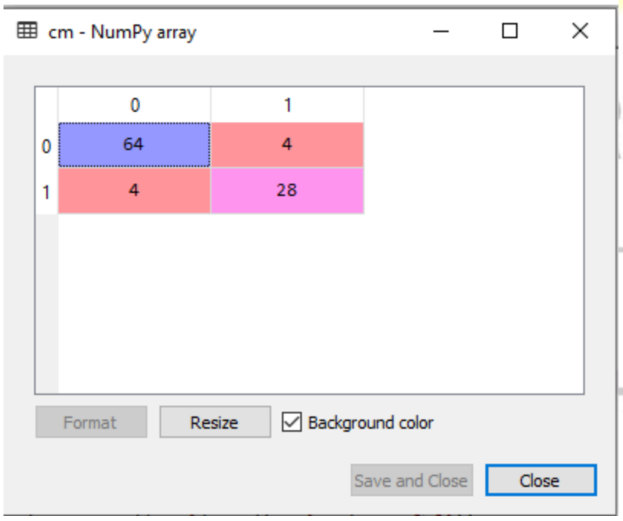
The prediction vector is given as:



```
     0
0    0
1    0
2    0
3    0
4    0
5    0
6    0
7    1
8    0
9    0
10   0
11   0
```

```
#Creating the Confusion matrix
from sklearn.metrics import confusion_matrix
cm= confusion_matrix(y_test, y_pred)
```

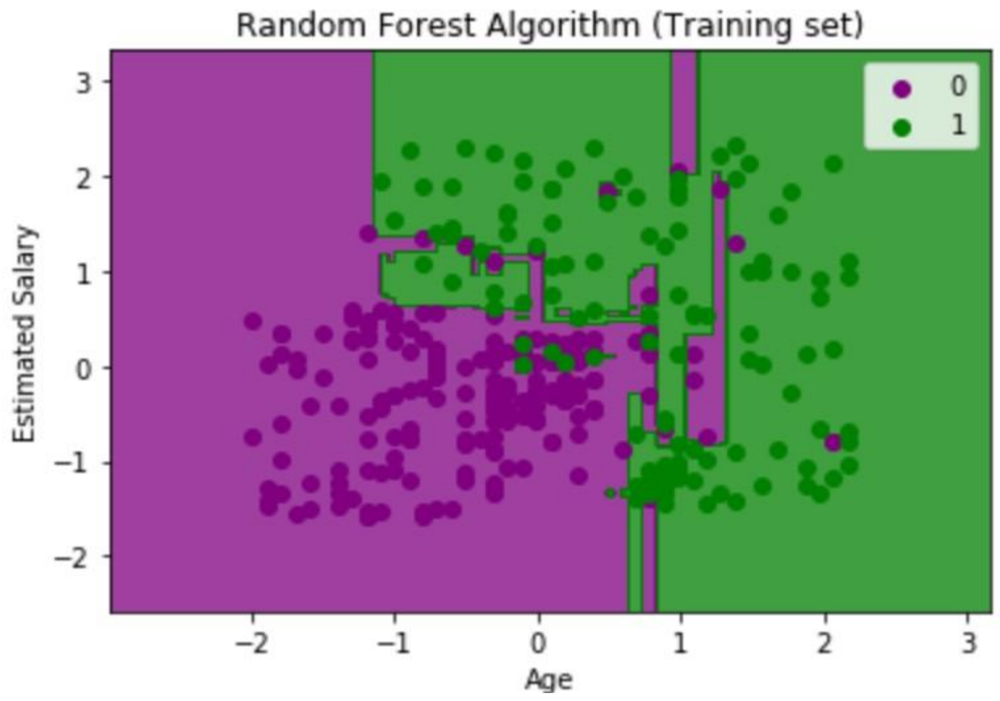
Output:



	0	1
0	64	4
1	4	28

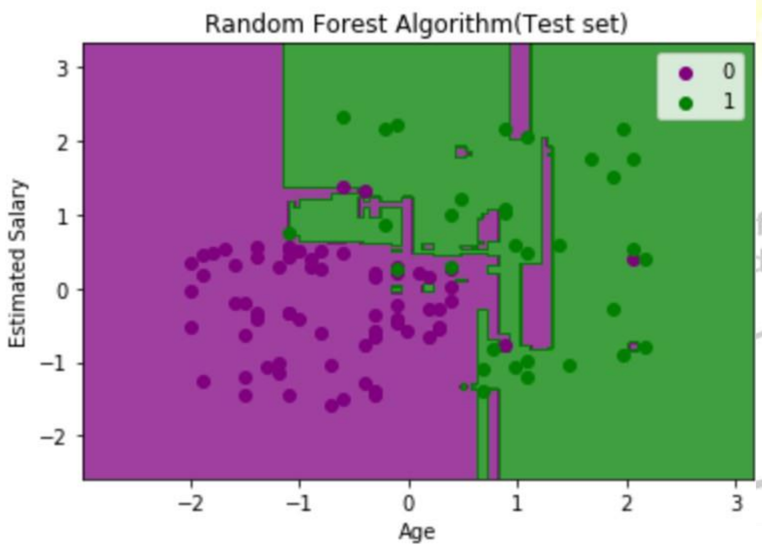
```
from matplotlib.colors import ListedColormap
x_set, y_set = x_train, y_train
x1, x2 = nm.meshgrid(nm.arange(start = x_set[:, 0].min() - 1, stop = x_set[:, 0].max() + 1, step  =0.0
nm.arange(start = x_set[:, 1].min() - 1, stop = x_set[:, 1].max() + 1, step = 0.01))
mtp.contourf(x1, x2, classifier.predict(nm.array([x1.ravel(), x2.ravel()]).T).reshape(x1.shape),
alpha = 0.75, cmap = ListedColormap(('purple','green' )))
mtp.xlim(x1.min(), x1.max())
mtp.ylim(x2.min(), x2.max())
for i, j in enumerate(nm.unique(y_set)):
   mtp.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1],
      c = ListedColormap(('purple', 'green'))(i), label = j)
mtp.title('Random Forest Algorithm (Training set)')
mtp.xlabel('Age')
mtp.ylabel('Estimated Salary')
mtp.legend()
mtp.show()
```

Output:



```
#Visulaizing the test set result
from matplotlib.colors import ListedColormap
x_set, y_set = x_test, y_test
x1, x2 = nm.meshgrid(nm.arange(start = x_set[:, 0].min() - 1, stop = x_set[:, 0].max() + 1, step  =0.0
nm.arange(start = x_set[:, 1].min() - 1, stop = x_set[:, 1].max() + 1, step = 0.01))
mtp.contourf(x1, x2, classifier.predict(nm.array([x1.ravel(), x2.ravel()]).T).reshape(x1.shape),
alpha = 0.75, cmap = ListedColormap(('purple','green' )))
mtp.xlim(x1.min(), x1.max())
mtp.ylim(x2.min(), x2.max())
for i, j in enumerate(nm.unique(y_set)):
   mtp.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1],
       c = ListedColormap(('purple', 'green'))(i), label = j)
mtp.title('Random Forest Algorithm(Test set)')
mtp.xlabel('Age')
mtp.ylabel('Estimated Salary')
mtp.legend()
mtp.show()
```

Output:


Random Forest Algorithm(Test set)
● 0
● 1
Estimated Salary
Age