20/10/22
Thursday

# Decision Tree in ML :-

13. Decision Tree :- It is a Machine learning algorithm based Supervised learning, that can be used for both regression & classification problem.

- The goal is to build up a classifier model that can predict the class or value of the target variable.

  Ex - In Medical field,
  In Educational Sector,
  In Banking Sector.

## Decision Tree classifier —

a) ~~(Entropy)~~ Gini ~~Index~~ Impurities → Purity Split, Impurities
b) Information Gain → feature split.

Entropy formula

$$H(S) = -P_+ \log_2 P_+ - P_- \log_2 P_-$$
$$= -(P) + \log_2 P + (-P) - \log_2 P -$$

↑ Purity

Gini - Impurities

$$G.I = 1 - \sum_{i=1}^{n} (P)^2$$

↑ Information Gain

try to correct use in formula


Gyes | 3N₀

Impurity split 3Y | 3N



3Y | 0N₀ , Purity split

leaf Node

Using Entropy formula -

$\Rightarrow H(C_2) = -P_+ \log_2 P_+ - P_- \log_2 P_-$ $\quad \{ P_+ = \frac{3}{3} = 1$

H(S) $\Rightarrow$ H(S)

$\Rightarrow -\frac{3}{3} \log_2\left(\frac{3}{3}\right) - \frac{0}{3} \log_2\left(\frac{0}{3}\right)$

$\Rightarrow -1 \log_2 1$

$\Rightarrow 0$ = pure split

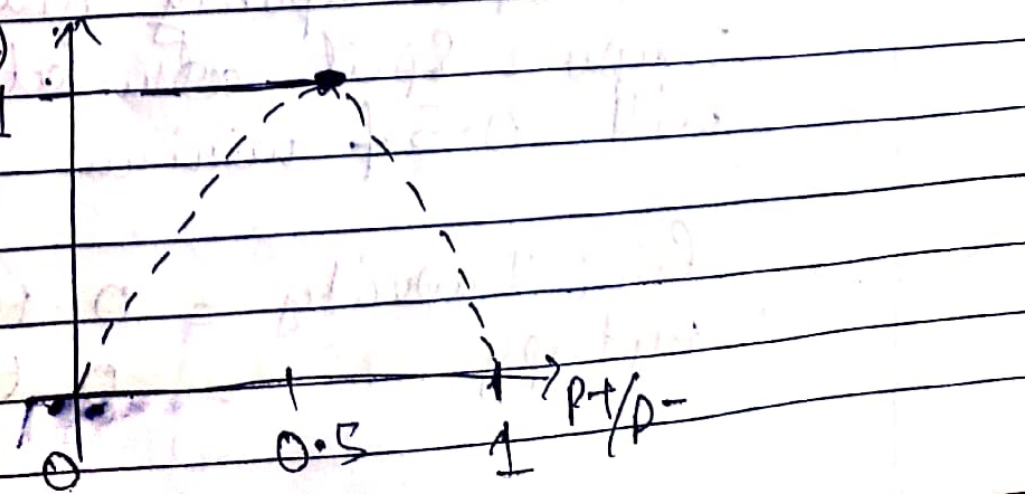
$\Rightarrow H(C_1) \Rightarrow -\frac{3}{6} \log_2(3/6) - \frac{3}{6} \log_2(3/6)$ $\quad \{ P_+ = 0.$ $P_- = 0/-$ $= 0.$

$\Rightarrow 1$ = impure split

H(S) H(C2)

1

Entropy graph $\Rightarrow$

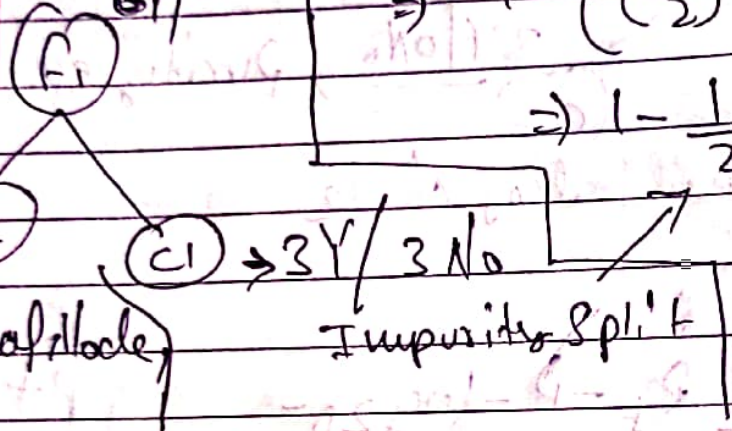


Gini Impurity $\Rightarrow 1 - \sum_{i=1}^{n} (p)^2$ or

$\Rightarrow 1 - ((p+)^2 + (p-)^2)$

$\Rightarrow 1 - \left(\left(\frac{1}{2}\right)^2 + \left(\frac{1}{2}\right)^2\right)$

$\Rightarrow 1 - \frac{1}{2} = 0.5$

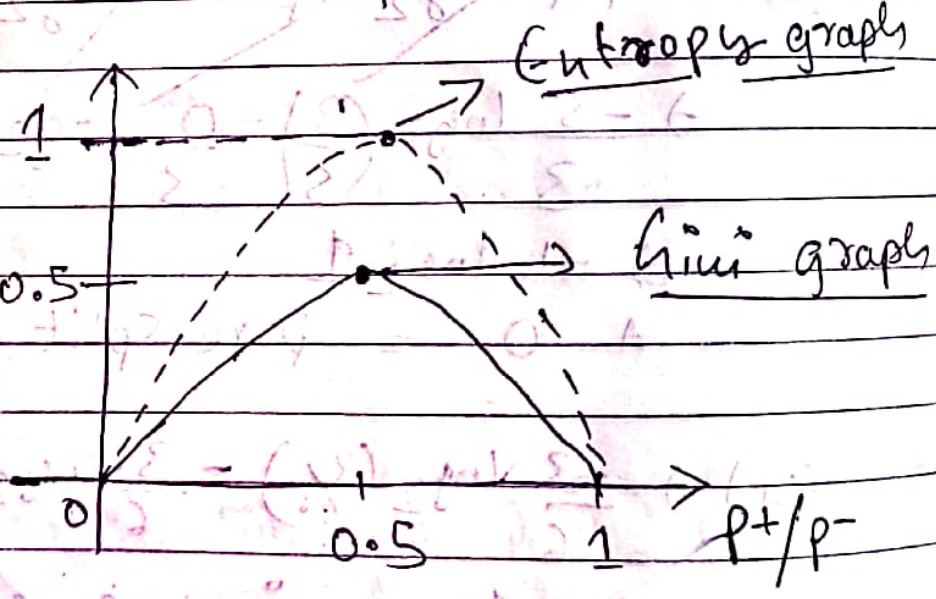



$1 - (p+)^2 + (p-)^2$

$1 - (1)^2 + (0)^2$

$1 - 1$

$= 0$ Impurity




∴ when Gini Impurity gives output for Impure Split then always or maximum will 0.5 & minimum 0

Gini Impurity = 0 to 0.5

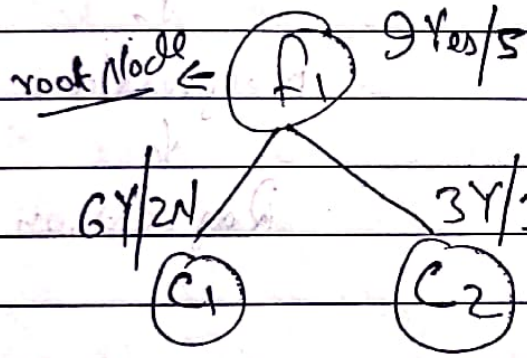
Entropy = 0 to 1

Note :- for long dataset we should use Gini Impurity as it has simple math instead of Entropy and Gini Impurity will consume less time than Entropy.

## Information Gain :-

$$Gain(S, F_1) \Rightarrow H(S) - \sum_{v \in Eval} \frac{|S_v|}{|S|} H(S_v)$$

$H(S_v) = C_1$ & $C_2$




$$[F_1] \; H(S) = -P_+ \log_2 P_+ - P_- \log_2 P_-$$

$$\Rightarrow \frac{-9}{14} \log_2 (9/14) - \frac{5}{14} \log_2 (5/14)$$

$$= 0.94$$

$$H(C_1) = \frac{-6}{8} \log_2 \left(\frac{6}{8}\right) - \frac{2}{8} \log_2 (2/8) = 0.81$$

$$H(C_2) \Rightarrow 1$$

$H(S) = 0.94$, $|S| = 14$

$$Gain(S, F_1) = 0.94 - \left[\frac{8}{14} \times 0.81 + \frac{6}{14} \times 1\right]$$

$$= 0.049$$

## Decision Tree

- Classification
  - i) Post-pruning
  - ii) Pre-pruning
- Regression
  - & Regression

# Post pruning & Pre-pruning in Decision Tree

Post pruning & Pre-pruning used to control the overfitting in Decision Tree.

Whenever we make Decision Tree to leaf node for training the dataset, the accuracy of Test will less which is known as overfitti And prevent overfitting we use the proces kno as Post pruning & Pre-pruning.

Post pruning → for Small Dataset
Pre pruning → for Bigger Dataset

Max depth
↓
Hyper parameter

max dept is a hyper paramet used in Decision Tree for Post pruning to control overfitting.

# Decision - Tree Classification -

## Code
## Post-Pruning Decision Tree Classification -

Code no. ① >>> import pandas as pd
>>> import matplotlib.pyplot as plt
>>> %matplotlib inline

Code no. ② >>> from SKlearn.datasets import load_iris
>>> iris = load_iris()
>>> iris

(# directly importing iris dataset from SK.learn in array form that is binary form)

Code no. ③ >>> iris.data (# these are the data of the dataset)

Code no. ④ >>> iris.target (# these are the output of data)

∴ Now, the data set we had imported from SK.learn is in array form so we need to import the dataset in <Pandas.core> for understanding the data or ...

⇒ we can import from Seaborn or directly from Pandas libraries.

Code no. ⑤ >>> import Seaborn as sns
>>> df = sns.load_dataset('iris')
>>> df

df =

| Sepal length | Sepal wid | petal leng | Petal width | Species |
|---|---|---|---|---|
| 5.1 | 3.5 | 1.4 | 0.2 | setosa |
| 4.9 | 3.0 | 1.4 | 0.2 | Setosa |
| 6.7 | 3.0 | 5.2 | 2.3 | virginica |
| 6.3 | 2.5 | 5.0 | 1.9 | Virginica |

Code no ⑥ :-
To distguish between independence features & dependence

```
>>> x = df.iloc[:,:4]      (# independence feature)
>>> y = iris.target        (# dependence feature)
```

To check print (x, y)

Code no ⑦ :-
```
# train_test_Split
>>> from sklearn.model_Selection import train_test_split
>>> x_train, x_test, y_train, y_test = train_test_split(
        (x, y, test_size = 0.33, random_state = 42))
```

Code no ⑧
```
>>> from sklearn.tree import DecisionTreeClassifier
```
(Importing decision Tree classifier)

Code no (9) :- Now we are doing Post pruning in this data set in which

# Post pruning

```
>>> tree_model = DecisionTreeClassifier()
```

(# here it will go till the leaf node because we didn't give the parameter (max_depth=2)

or

```
>>> tree_model = DecisionTreeClassifier(max_depth=2)
```

:- Remember (max_depth=3) plays an very important role in Decision Tree

Code no (10) :- fit the model or place in training.

```
>>> tree_model.fit(x_train, y_train)
```

Code no (11)

```
>>> from sklearn import tree
>>> plt.figure(figsize=(15,10))
>>> tree.plot_tree(tree_model, filled=True)
```

:- # To make or to draw Decision Tree diagram

>>> y_pred = tree_model.predict(x_test) (# to predict the x_test)

③ # to check accuracy score & classification report import this #

>>> from sklearn.metrics import accuracy_score, classification_report

>>> score = accuracy_score(y_pred, y_test)
>>> score (# to predict the accuracy score)

print(classification_report(y_pred, y_test)
(# to predict the classification report)

# our decision_Trees_classifier model has been prepare with us we can predict our own data

>>> tree_model.predict([[5.2, 3.6, 1.4, 0.377])

# Decision - Tree Classification —

## Pre - Pruning Decision - Tree Classifier —

∴ We will use same data set from the previous page / Post-Pruning and most of the code will be same till Post-pru

∴ Code no 1 to 7 will be same or if something will wrong then it will be minor error

Code ⑧ We will use some special parameters in the variable name as parameter —

```
>>> parameter = {
    'criterion': ['gini', 'entropy', 'log_loss'],
    'splitter': ['best', 'random'],
    'max_depth': [1, 2, 3, 4, 5],
    'max_features': ['auto', 'sqrt', 'log2']
    }
```

Code ⑨ ∴ for pre-pruning we have to import model library name as GridSearchCV

```
>>> from sklearn.model_selection import GridSearchCV
```

Code no (10) :- Now import the Decision Tree classifier() & give the parameter in a Variable

```
>>> treemodel = DecisionTreeClassifier()
>>> cv = GridSearchCV(estimator = treemodel,
                      param_grid = parameter, cv = 5,
                      scoring = 'accuracy')
```

(# cv = Corros Validation
We can use max_dept = 3 in the treemodel)

Code no (11) `>>> cv.fit(x_train, y_train)`

Code no (12) :- To check which parameter are the best or useable in the data as we are doing Pre-Pruning we will use

```
>>> cv.best_params_
```

Code no (13) `>>> y_pred = cv.predict(x_test)`

Code no (14) :- Now if we want to check the accuracy score or the classification report of the data set which we train & tested we will use -

```
>>> from sklearn.metrics import accuracy_score,
    classification_report
>>> Score = accuracy_score(y_pred, y_test)
```

# Check classification report

Code (15)

```
>>> print(classification_report(y_pred, y_test))
```

```
>>> clf.predict([[5, 4, 2, 3]])
```

---

Note

We use "MSE" "mean square error" to split in Decision Tree Regression.
whereas
we use "gini impurity" or "entropy" in Decision Tree Classification

also Note

also Remember that
In Regression we use 'mse' most of the time
whereas
we generally use "gini Impurity" or "entropy" in classification.

## — Decision Tree —

⇒ Consider for both Regression and classification Problems.

$$\text{Gini} = 1 - \sum_{i=1}^{C} (P_i)^2$$

| Class | Gender | Stay in Hotel |
|---|---|---|
| 9 | M | Yes |
| 10 | F | No |
| 8 | F | Yes |
| 8 | F | No |
| 9 | M | Yes |
| 10 | M | No |
| 11 | f | Yes |
| 11 | M | Yes |
| 8 | f | Yes |
| 9 | M | No |
| 11 | M | No |
| 11 | M | Yes |
| 10 | F | No |
| 10 | M | Yes |

data = 1

∴ Gini impurities work well with the categorical data.

Table-1

| class | Stay in Hotel | Total | P(Y) | P(N) |
|---|---|---|---|---|
| 8 | Y=2, N=1 | 3 | 2/3 | 1/3 |
| 9 | Y=2, N=1 | 3 | 2/3 | 1/3 |
| 10 | Y=1, N=3 | 4 | 1/4 | 3/4 |
| 11 | Y=3, N=1 | 4 | 3/4 | 1/4 |
| | | 14 | | |

(Table 1 ==)

Gini = $1 - \sum_{i=1}^{c} (P_i)^2$ (Here $P_i$ is probability)

$\Rightarrow G(8) = 1 - [P(Y)^2] - P(N)^2$

$\Rightarrow 1 - (2/3)^2 - (1/3)^2 = 4/9$

$\Rightarrow G(9) = 1 - (2/3)^2 - (1/3)^2 = \frac{4}{9}$

$\Rightarrow G(10) = 1 - (1/4)^2 - (3/4)^2 = \frac{6}{16}$

$\Rightarrow G(11) = 1 - (3/4)^2 - (1/4)^2 = \frac{6}{16}$

$\Rightarrow [G(c) = \sum \frac{\text{No. of instance for class}}{\text{total No. of instance}} \times G(c)$

This is also for finding total entropy

$\Rightarrow \frac{n_8}{T} \times G(8) + \frac{n_9}{T} \times G(9) + \frac{n_{10}}{T} \times G(10) + \frac{n_{11}}{T} \times G(11)$

$\Rightarrow \frac{3}{14} \times \frac{4}{9} + \frac{3}{14} \times \frac{4}{9} + \frac{4}{14} \times \frac{6}{16} + \frac{4}{14} \times \frac{6}{16} = 0.404$

So, Gini of entire column [Class] $G(c) = 0.404$

Same thing we will try to do with the Gender Column to find Gini impurity and we get 0.482

Gini impurity of class = 0.404
" " Gender = 0.482

∴ Note - we will always try to select less Gini-impurity column

So we will try to use Class as it has less gini compare than Gender.

∴ In this case will try to take Class as a root node

Note - ∴ we have to always choose a less Gini-impurity Columns for root node.

Table 2

| Gender | Class in Math | | T | P(Y) | P(N) |
|---|---|---|---|---|---|
| M | Y=5 | N=3 | 8 | 5/8 | 3/8 |
| F | Y=3 | N=3 | 6 | 1/2 | 1/2 |
| | | | 14 | | |

Gender

$$G(M) = 1 - \left(\frac{5}{8}\right)^2 - \left(\frac{3}{8}\right)^2 = 0.468$$

$$G(F) = 1 - \left(\frac{1}{2}\right)^2 - \left(\frac{1}{2}\right)^2 = 0.5 = \frac{1}{2}$$

Q.

$$G(Age) = \frac{8}{14} \times 0.468 + \frac{6}{14} \cdot \frac{1}{2} = 0.482$$

— Entropy & Information Gain —

Entropy: randomness / scatterness / degree of freedom
(H C is no. of classes)

Entropy = $E = -\sum_{i=1}^{C} P \log_2(P)$ (H Here P is probability)

Information Gain — It is the difference between a Entropy.

$\Rightarrow$ IF $G = E_{\text{Before/Label}} - E_{\text{after}}$ [ Use before means label column ]

$$E = -\sum_{i=1}^{C} P \log_2(P)$$

∴ from the above Data 1 we will now find Entropy for Label Column -

In "Stay in Hotel" $\Rightarrow$ 14 records $n(Y) = 8$
$n(N) = 6$

$\Rightarrow E(L) = -P(Y) \log_2 P(Y) - P(N) \log_2 P(N)$

$\Rightarrow -\frac{8}{14} \log\left(\frac{8}{14}\right) - \left(\frac{6}{14}\right) \log\left(\frac{6}{14}\right)$

$\Rightarrow 0.98522$

∴ Now find the entropy for every class column check from Table 1 extract from Data 1.

(Entropy)

$\Rightarrow E(8) \Rightarrow \frac{-2}{3}\log(2/3) - 1/3\log(1/3) = 0.918$

$\Rightarrow E(9) \Rightarrow -2/3\log_2(2/3) - 1/3\log_2(1/3) = 0.918$

$\Rightarrow E(10) \Rightarrow -1/4\log_2(1/4) - 3/4\log_2(3/4) = 0.811$

$\Rightarrow E(11) \Rightarrow -3/4\log_2(3/4) - 1/4\log_2(1/4) = 0.811$

Total No. of instances (data) = 14

Total Entropy or Information of class

Entropy Information (class) $= \frac{3}{14} \times 0.918 + \frac{3}{14} \times 0.918 + \frac{4}{14} \times 0.9$
$+ \frac{4}{14} \times 0.811$

Total entropy/Information $\Rightarrow 0.8574 = E(\text{class})$
(Here entropy or information are same)

$I.G(\text{class}) = E_{\text{total}} - E_{\text{after}}$
$\Rightarrow 0.98522 - 0.8574$
$\Rightarrow 0.127$

∴ Information Gain for Class column is 0.127.

Now we will find Information Gain for the Gender column. So we have to find Entropy of Gender (check Table no-2 for data)

$E(M) = -5/8 \log_2 5/8 - 3/8 \log_2 3/8 \Rightarrow 0.954$

$E(F) = -3/6 \log 3/6 - 3/6 \log 3/6 = 1$

Entropy | Information of Gender $\Rightarrow \frac{8}{14} \times 0.954 + \frac{6}{14} \times 1$

$\Rightarrow E(G) / I(G) \Rightarrow 0.974$

$\therefore$ I.G(Gender) $\Rightarrow E_{(Total)} - E_{(Gender)}$

$\Rightarrow 0.98522 - 0.974$

$\Rightarrow 0.01$

So,

I.G(class) $\Rightarrow$ 0.1278 | E(class) = 0.8574

I.G(Gender) $\Rightarrow$ 0.01 | E(Gender) = 0.974

E(total) $\Rightarrow$ 0.98522 (Stay in Hotel)

Entropy = randomness | degree of freedom

Note - We will choose that column which have high Information Gain i.e class has less entropy

- High Information Gain is equal to low Entropy in a particular column (ie in class)

Note → If our label and feature are Categorical in this case we can use Gini Impurities & Entropy or Information Gain.
This is a Classification Problem.

i) feature = categorial, outcome = Categ. ⇒ Classification

ii) feature ⇒ continus, outcome ⇒ Categ. ⇒ Classification

iii) feature is continus, Outcome is Continuous ⇒ Regression

* *

## More Algorithms of Decision Tree

i) ID3 algorithm ⇒ Iterative Dichotomiser ⇒ Classification prob

ii) C4.5 algorithm ⇒ revision of ID3 ⇒ classification problem

iii) CART algorithm ⇒ Classification Regression Tree ⇒ for both

Note - In multi-classification problem we try to see accuracy

Note ⇒ # to create meta file for decision tree -

```
>>> out_file = dot_for_dt = open("dt_model_meta.dot", 'w')
>>> tree.export_graphviz(dt_model, out_file = out_file,
                         feature = n.columns)
```

# Random-forest

# Bootstrap tech is also known as Bagging.

Ensemble Technique

| Bagging | Boosting | Stacking |
|---|---|---|
| S.V.M | Ada-Boost | |
| KNN | Xg Boost | |
| Naive Bais | Gradient Boost | |
| Decision Tree | | |
| Random forest | | |

## Random-forest

Random-forest – It is one of the algorithm and a part of ensemble technique work with bagging process.

Note → we want to take ML model we always try to take a model which have low bias & low variance but to get these two simultaneously is very hard and difficult because when bias is low than variance is high & when variance is low than bias is high. So, To solve this problem we use Regularization, Bagging & Boosting.

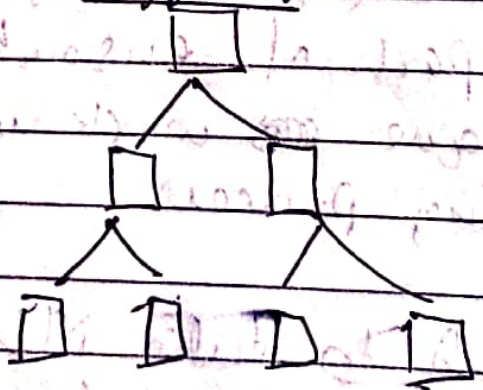
Note ⇒ In Bagging we should always try to take a model who has low Bias & High Variance (LB & HV) like (Decision Tree (Fully) (Knn)

but

In Boosting we should always try to take a model who has High Bias & low Variance for ex (shallow decision tree (depth should be very less), (decision stump)

Note - In Bagging model learn parallel & In Boosting model learn sequence

Bagging



Boosting

Note — Ensemble Technique is used to control the Overfitting in Decision Tree.

31/10/22
Monday

# Random - forest

Topic 14 Random-forest :- It is constructed using multiple decision-tree & the final decision is obtained by majority votes of the decision trees.

Random forest is Combination of decision-trees

## Importance Terms in Random Forest :-

i) Root Node - Training data set is fed at the root node.

ii) Splitting - Gini & entropy method to decide the optimal split.

iii) Decision nodes - Provides the link to the leaf nodes.

iv) Lead-nodes :- end-point and no further division takes place.

# How to work on Random-forest —

- Random Sampling with replacement.
  ⇓
- Ensemble technique used bootstrap aggregation / bagging.
  ⇓
- How features selection is done in classification & regression problem.
  ⇓
- The best split is chosen based on Gini Impurity or Information Gain methods.

# How features selection is Done?

i) classification — by default the features selecti is taken as the square root o total number of all the features.

ii) Regression — by default, features are selecte by the total number of all features divided by 3.

# Advantages of Random Forest :-

i) Low Variance — Multiple decision tree.

ii) Reduced overfitting — Uses bootstrapped aggregation.

iii) Normalization not needed.

iv) Good Accuracy

v) Suitable for both classification and Regression problem.

vi) Work well with both Categorical & Continuous data.

vii) Performs well on large data-sets.

# Disadvantage of Random Forest —

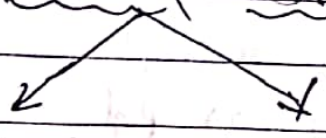
i) More training time is required.

ii) Interpretation is complex

iii) More memory utilization.

iv) Computationally expensive.

# Random forest



Classification  Regression

Note - i) Random forest is a ensemble learning method.

ii) Random forest classification is work on Majority-Voting. /or Gives the output on Majority ~~o~~tcomes.

But In

Random forest Regression is work on ~~outcomes~~ Average.

Hyperparameter - CCP -

Pruning of Decision Tree

Some important Code and Note regarding Decision Tree

# To plot decision Tree

```
>>> import sklearn
>>> from sklearn import tree
>>> tree.plot_tree(dt_model, filled=True)
```

=> (model name)

Or to know the class name

code 4

```
>>> plt.figure(figsize=(20,20))
>>>> tree.plot_tree(model_name, filled=True, class_names=[str(i) for i in set(yt)]
>>> plt.savefig("dt_model") # to save the dt.
```

# (ccp_alpha= ) it is a pruning operation in D.T to control the branches or the Minimal cost-Complexity pruning.

— Pruning of Decision Tree —

=> model name

code 2

```
>>> path = dt_model_1.cost_complexity_pruning_path(X_train, y_train)
>>> ccp_alpa = path.ccp_alphas
>>> ccp_alpa
```

-ccp-

ccp_alpha = It is cost complexity pruning technique in Decision Tree that is used to overcome problem of Overfitting. It provide option to control the size of a tree. (code No -2)

- After code No 2 we will find the no. of ccp_alpha for the model -

```
>>> dt_model_2 = [ ]
>>> for ccp in ccp_alphas:
>>>     dt_m = DecisionTreeClassifier(ccp_alpha=ccp)
>>>     dt_m.fit(x_train, y_train)
>>>     dt_model_2.append(dt_m)

>>> dt_model_2   (# here we will get a list of ccp alpha which is needed for the model)
```

Note → We are supposed to select that ccp alphas Score Value where train & test score is almost closer to each other.

∴ from the code no 3 we will many model of Decision Tree of diff ccp_alpha so we could check train & test accuracy and select best Decision Tree model with the help of ccp technique

Date
Page

Code no 4 >>> train_score = [i.score(x_train, y_train) for i in dt_model_2]

>>> train_score (# for training score)

>>> test_score = [i.score(x_test, y_test) for i in dt_model_2]

>>> test_score (# for test score)

# from the above code we will get all train & test score of the model and we

Code no 5 — :- Now we will compare the accuracy score with ccp_alphas for better understanding in graph

```
>>> import matplotlib.pyplot as plt
>>> fig, ax = plt.subplots()
>>> ax.set_xlabel("ccp_alphas")
>>> ax.set_ylabel("accuracy_score")
>>> ax.set_title("Accuracy vs alpha")
>>> ax.plot(ccp_alphas, train_score, drawstyle="steps-post",
            label="train", marker="o")
>>> ax.plot(ccp_alphas, test_score, drawstyle="steps-post",
            label="test", marker="o")
>>> ax.legend()
>>> plt.show()
```

## — Grid Search CV —

Note — from the code no 5 we will get a comparision graph of Accuracy Score & ccp_alpha and the nearest for both we will select that point of the alpha value for the further process

### —— Grid Search CV ——

```
>>> from sklearn.model_selection import GridSearchCV
>>> grid_param = {
        "criterion": ['gini', 'entropy'],
        "splitter": ["best", "random"],
        "max_depth": range(2, 10, 1),
        "max_samples_leaf": range(2, 10, 1),
        "min_samples_leaf": range(1, 10, 1),
        "ccp_alpha": np.random.rand(20)
    }
# it will take very, very, very much time
>>> grid_ccp_model = GridSearchCV(estimator=dt_model,
        param_grid=grid_param, cv=10,
        n_jobs=-1)
# Here estimator means the the previous best model
  without the using hyper parameter
# cv - cross validation divided 9 train data
  & 1 testing data if we take cv=10
# n_jobs = -1 means that the processor of our
  CPU & GPU will work -1 mean all
```

Note :- It will take very very time for using all parameter

Date
Page

Note → GridSearchCV & RandomizedCV can be used ; u can choose

code no 8 → >>> grid_ccp.fit(x_train, y_train)
# it will take much time
~~>>> grid_ccp.best_params_~~
>>> grid_ccp.best_params_
↓↓
# Here we will get best parameter to use and we will use it by creating new model variable as a best model

code no 9 >>> dt_grid_ccp = DecisionTreeClassifier(criterion=" ", max_depth=" ", min_sample_split=" ", max_sample=" ", splitter=" ")

# In the blank we will give the output of code no 8 that is best_params_

code no 10 >>> dt_grid_ccp.fit(x_train, y_train)
# fit our new pruning model

code no 11 >>> dt_grid_ccp.score(x_train, y_train)
>>> dt_grid_ccp.score(x_test, y_test)

Note → No algorithm is the best or worst one it depend on the nature of the dataset and it depend on the parameter which we are going to select