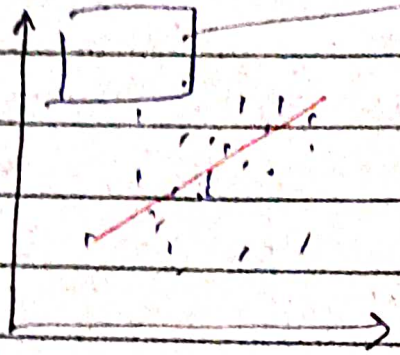
03.10.2023
Tuesday

.ravel() → contigenous array

Date: Page:

import matplotlib.pyplot as plt

plt.scatter(x, y, color='r') → Dots print होंगे
plt.plot(x, pred-y, color='g') → line आएगी regression
plt.show()



→ legend off कर सकते हैं

headbrain.csv

Age	Gender	headsize	brain weight

M.2: Scikit Learn Method :-

install sklearn.

from sklearn.linear_model import LinearRegression
import numpy as np
import pandas as pd

data = pd.read_csv('headbrain.csv')
x = data['headsize']
y = data['brainweight']

reg = LinearRegression()
reg.fit(x, y)
y_pred = reg.predict(x)

09/10/2023 Monday

Logistic Regression
↳ supervised M.L. Algo.

Date: Page:

import matplotlib.pyplot as plt
plt.scatter(x, y) → points = original
plt.plot(x, y_pred) → Regression = Predicted values

LOGISTIC REGRESSION :-

from sklearn.linear_model import LogisticRegression
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix

data = pd.read_csv('iris.csv')

x = data[['sepal length', 'sepal width', 'petal length', 'petal width']]
y = data['varieties']

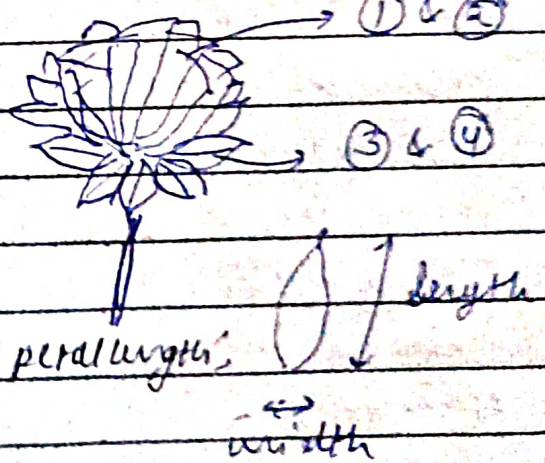
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3, random_state = 1)

70% Training data set (maximum samples को Train करना पड़ता है) 30% Testing data set

150 rows
iris.csv
↓
iris flower has 3 categories.
↙
setosa, versicolor, virgin color

each has 50 corresponding samples.

① petal length, ② petal width
③ sepal length, ④ sepal width
⑤ class label



we can also use Cross validation.

random state = 1 ⇒ dividing randomly
↳ if we again execute the program then we will get same samples in training & testing set.
for multiple iteration ⇒ we will get same performance. (until we have same random state)

# Constructing the model :-
LogReg = LogisticRegression()

# Training :-
LogReg.fit(x_train, y_train)

# Prediction :-
y_pred = LogReg.predict(x_test)

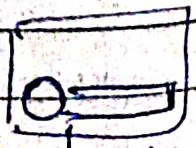
print(confusion_matrix(y_test, y_pred))

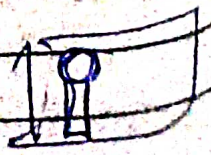
pre. & recall will be calculated individually & then take avg.

Actual \ predicted	Setosa	Versicolor	Virginica
Setosa	8	1	1
Versicolor	0	9	1
Virginica	2	1	7

Accuracy = $\frac{8+9+7}{30} = 0.8$

	Setosa	Versicolor	Virginica	Avg.
~~Accuracy~~ Precision	$\frac{8}{10}$	$\frac{9}{10}$	$\frac{7}{10}$	0.8
~~Accuracy~~ Recursion	$\frac{8}{10}$	$\frac{9}{11}$	$\frac{7}{9}$	2.396





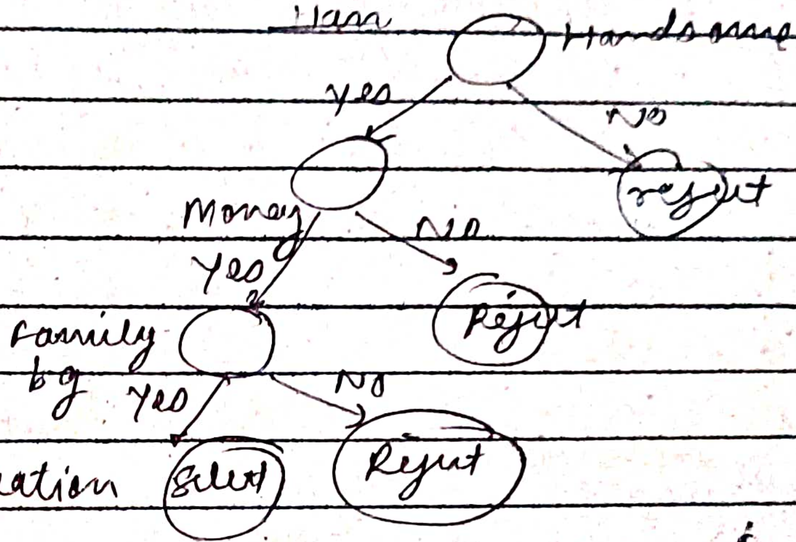
Recall out of total predicted — Precision = $\frac{TP}{TP+FP}$ saare mein many are correct.

Date: Page:

$$f_1\text{-score} = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

DECISION TREE

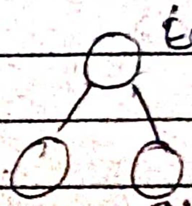
1. root node
2. leaf node
3. parent node
4. child node
5. subtree



→ supervised ML Algo. used for classification & regression

ASM Attribute Selection Metric :-

(Evaluation on basis of)



1. Entropy & Information Gain (IG)
   - ↳ randomness due to feature (impurity)
   - ↳ purity of a sample (data set)
2. Ginny Index

Decision Tree :-

I. Start with the root node, having complete dataset

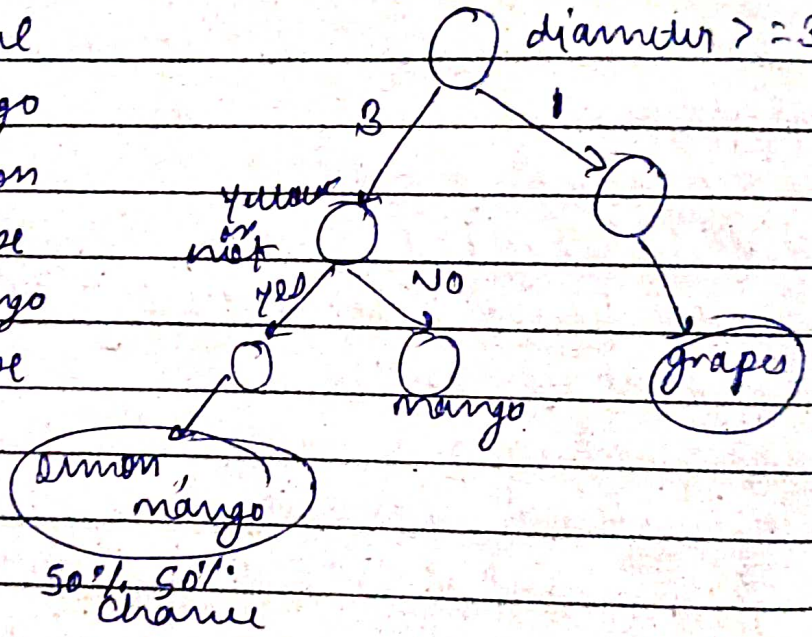
as an input.

(II) Selection of most significant attribute based on attribute selection Metric (IG)

(III) Evaluate the attribute & divide the tree into subtree based on possible values of attribute.

(IV) Repeat step (II) & step (III) for each subtree until all the attributes are evaluated.

Color	Diameter	Label
Green	3	Mango
Yellow	3	Lemon
Red	1	Grape
Yellow	3	Mango
Red	1	Grape



Entropy of dataset, S $E(S) = -p_i \log_2 p_i$

$= -\frac{6}{10}\log_2\frac{6}{10} - \frac{4}{10}\log_2\frac{4}{10}$

$= 0.97$

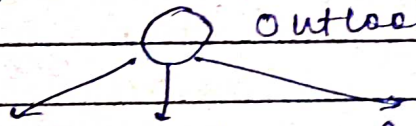
IG (Outlook) = 0.37

IG (Humidity) = 0.045

IG (Wind) = 0.02

10/10/2023
Tuesday

Day	Outlook	Humidity	Wind	Play
1	Sunny	High	Weak	No
2	Sunny	High	Strong	No
3	Cloudy	High	Weak	Yes
4	Rainy	High	Weak	Yes
5	Rainy	Normal	Strong	No
6	Cloudy	Normal	Strong	Yes
7	Sunny	High	Weak	No
8	Rainy	Normal	Weak	Yes
9	Sunny	Normal	Strong	Yes
10	Cloudy	High	Weak	Yes



Sunny | Cloudy | Raining

D1 D2 D7 D9 | D3 D6 D10 | D4 D5 D8

N N N Y | ← Yes → | Y N Y

Yes

Humid Wind

	Humid	Wind	
D1	High	Weak	No
D2	High	Strong	No
D7	High	Weak	No
D9	Normal	Strong	Yes

Humidity: 3/4 High, 1/4 Normal

Wind: Weak

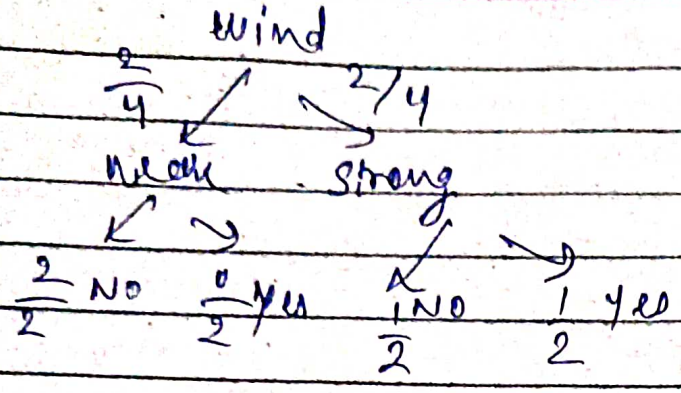
High: 0 No, 3 Yes

Normal: 1 Yes, 0 No

Entropy = $-\frac{1}{4}\log_2\frac{1}{4} - \frac{3}{4}\log_2\frac{3}{4} = 0.8$

	Humidy	Wind	Play
D4	High	Weak	Y
D5	Normal	Strong	N
D8	Norm	Weak	Y

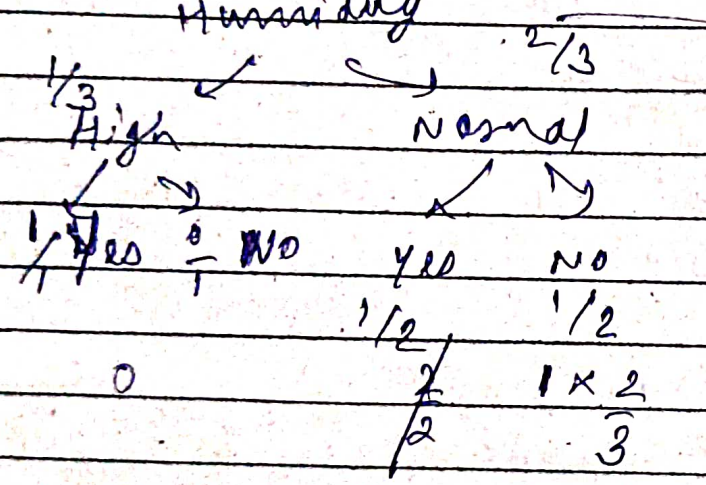
$IG(\text{Humidity}) = \frac{3}{4}\left(-\frac{3}{3}\log_2\frac{3}{3}\right) = 0.81$ (> 0)



$$\frac{2}{4}\left(-\frac{2}{2}\log_2\frac{2}{2}\right) + \frac{2}{4}\left(-\frac{2}{2}\log_2\frac{1}{2}\right) + 1$$

$$0.81 - \frac{1}{2}$$

$$= 0.91$$

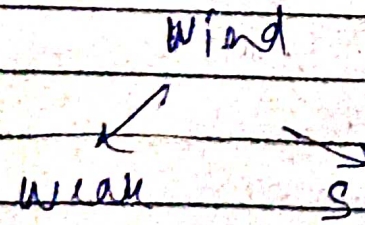


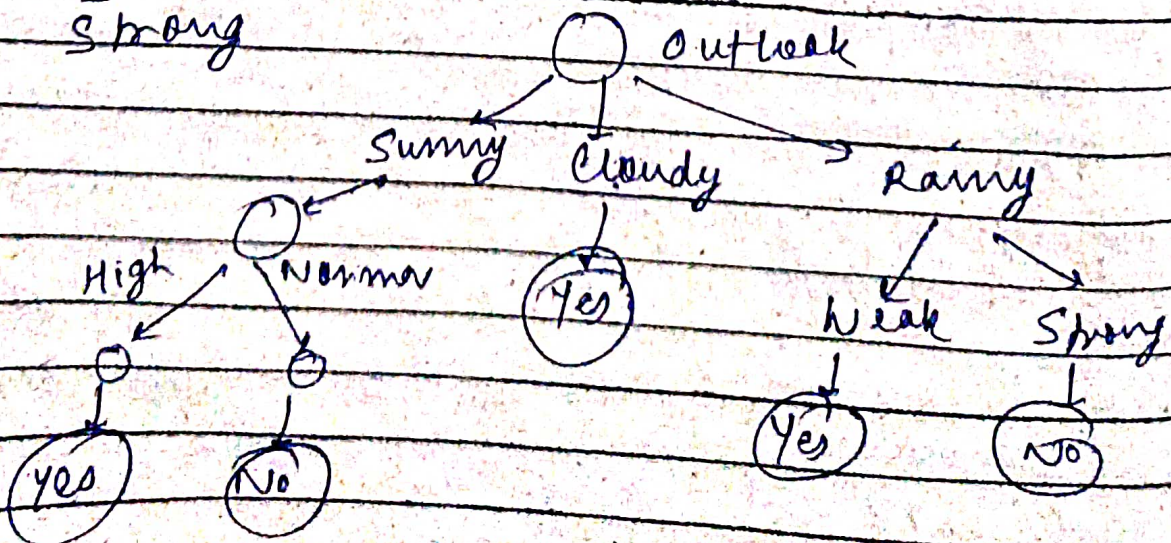
$$H(S) = -\frac{2}{3}\log_2\frac{2}{3} - \frac{1}{3}\log_2\frac{1}{3}$$

$$= 0.918$$

$$IG(\text{Humidity}) = 0.25$$

$$IG(\text{Wind}) = 0.91$$





16.10.2023 Monday

Random forest or k-fold cross validation → To overcome disadvantages

Date: Page:

Decision Tree

'iris' Data set

```
import pandas as pd
from sklearn.tree from import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix

Data = pd.read_csv('iris.csv')
x = Data [['sepal length', 'sepal width', 'petal length',
           'petal width']]
y = Data ['varieties']

x_train, x_test, y_train, y_test = train_test_split (x, y, test_size=0.3,
                                                     random_state=1)
DTmodel = DecisionTreeClassifier ()
DTmodel.fit (x_train, y_train)
y_pred = DTmodel.predict (x_test)
```

Advantages :—

→ Human type method to make decision.

1. Simple & Easy to understand.
2. It is useful for Decision making problem.
3. Less requirement of data cleaning

Disadvantages :—

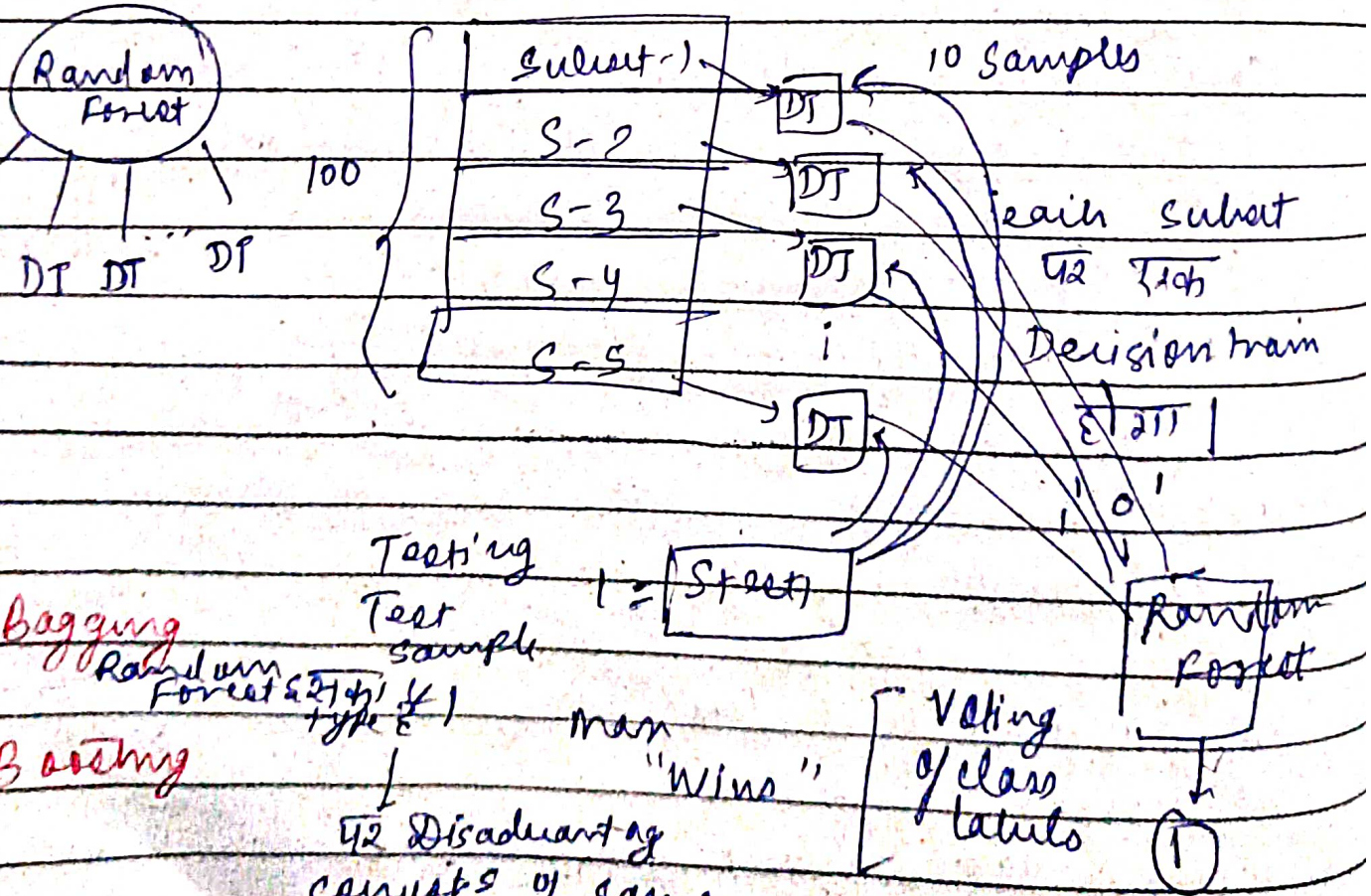
1. Increase in layers & tree results in ↑ in complexity.
2. Overfitting problem may occur
3. More the number of ~~classes~~ class labels, complexity of DT may increase

Overfitting :- When we train the model too much, such that dataset is trained also on noise. which will create conflict.

Underfitting :- Not completely trained on data. so classification ~~may~~ not be correct.

ENSEMBLE LEARNING

↳ Random Forest consists of multiple trees (Decision trees)
(Supervised Classification Algorithm)



① Bagging

② Boosting

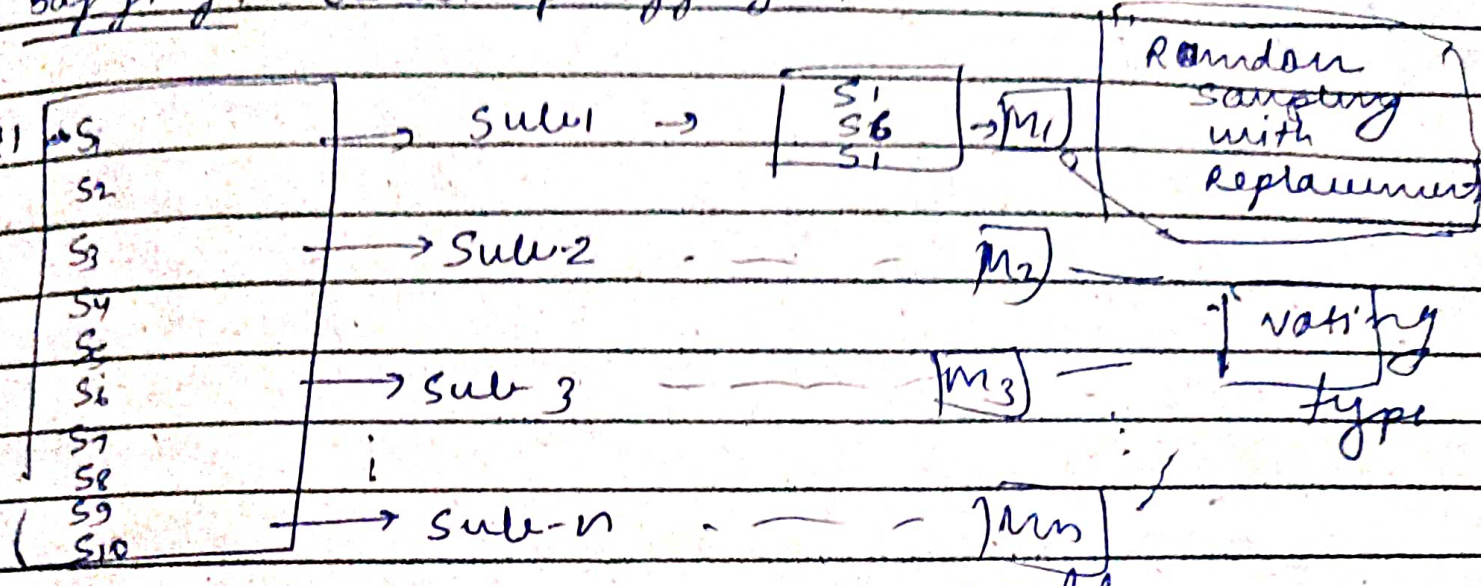
↳ Disadvantage consists of same DT. But in bagging multiple different classification algos can be used

Trained Parallily models

Also known as

Concept used

① Bagging :- Bootstrap Aggregation

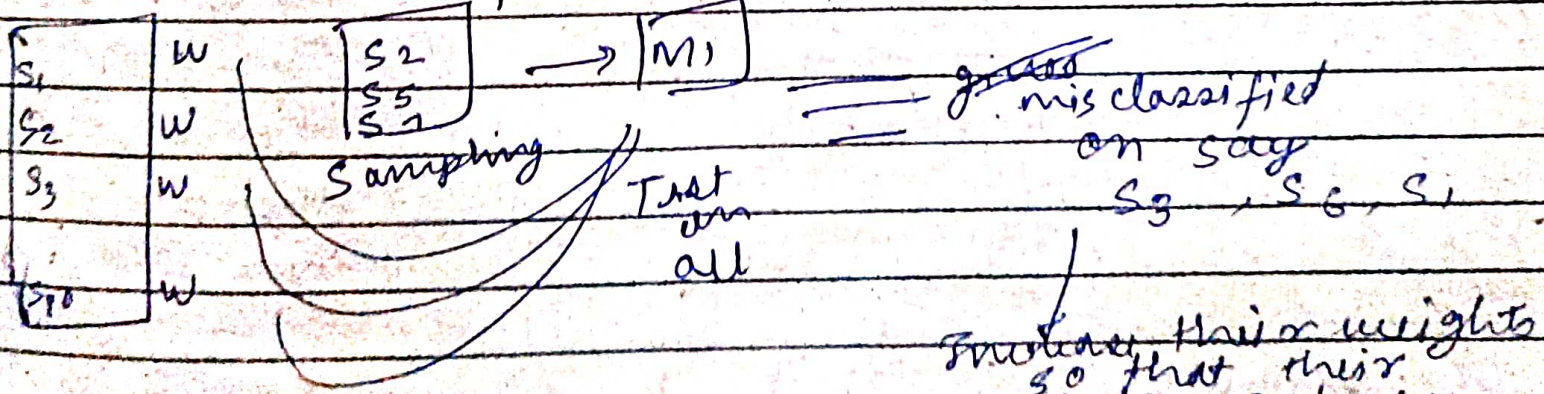


Bagging ~~no~~ trained

↳ model Creation : ~~Sequentially~~

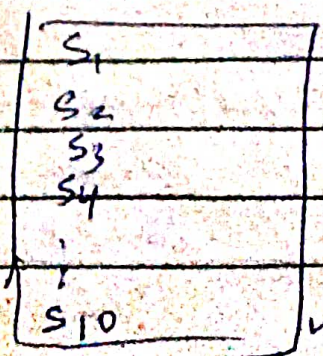
② Boosting : Selection on the basis of wt.

wt. ↑ ⇒ prob. ↑ ;



Decide your iterations (models) till you want to do!

Again do the same method ←

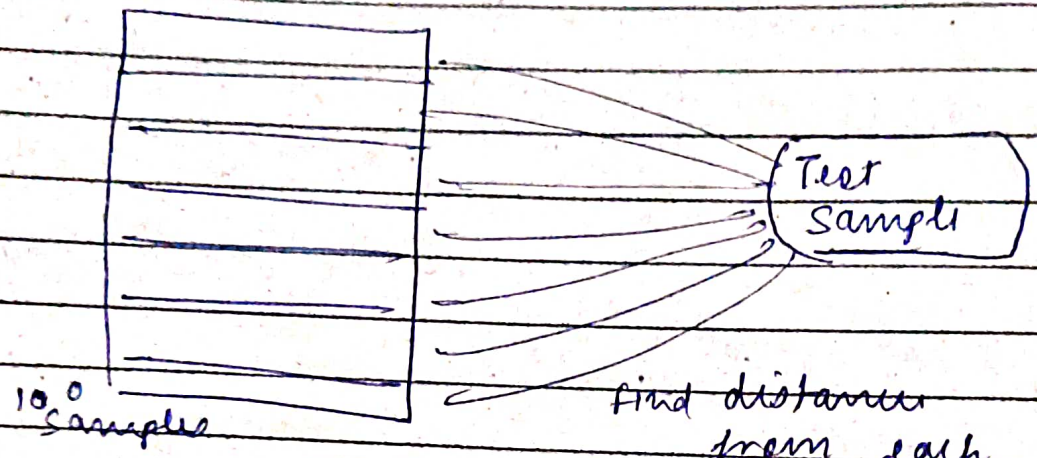


17, 10, 2023
Tuesday

Date: Page:

K-NN (K-Nearest Neighbours)

(Lazy Learner)



find distances from each

⇓

Then take K Nearest Neighbours from among them (Say K=5)

Distance is calculated using Euclidean distance

$$\text{E-distance} = \sqrt{\sum_{i=1}^{N} (x_i - y_i)^2}$$

x_i → Training Sample's feature

y_i → Testing Sample's feature

Two things we need to decide :-

(1) Features to be used in E-distance.

(2) K value decides

(i) For binary classification, take $K = 3$

(ii) For more than that,

counter = class labels = 3
ex:
But not fully correct

Date: Page:

$$K = \frac{\sqrt{N}}{2}$$

Generally k odd

(iii) Trial & Error Method (Algorithmic approach)

Vary $K = 1$ to N

& find best suited value of K for Test sample

& use that further.

Example

	Data Science	OS	Result	E-dist
S(1)	4	3	Fail	5.385
S(2)	6	7	Pass	1
S(3)	7	8	Pass	1
S(4)	5	5	Fail	3.162
S(5)	8	8	Pass	2

→ We calculated New student

$T_1 = (\text{Data Science} = 6, OS = 8)$

(Take $K = 3$)

E-dist. =

$$\sqrt{\left(DS_{\text{Score Train}} - DS_{\text{Score Test}}\right)^2 + \left(OS_{\text{Score Train}} - OS_{\text{Score Test}}\right)^2}$$

50% above (strictly) to pass

↳ Don't use this. Use KNN

$K = 3$

3 Nearest Neighbours

S(2) S(3) S(5)

Pass Pass Pass

Pass 3/3 Fail 0/3

Pass

T_1

Supervised Classification
(i) KNN (ii) Naive Bayes classifier (iii) Decision Tree
(iv) Logistic Regression
Unsupervised classification (Clustering)

Python implementation

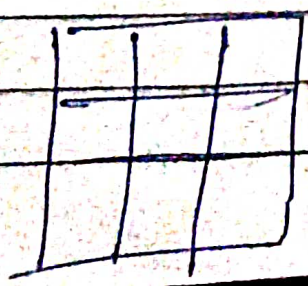
Data set : 'iris.csv'

```
import pandas as pd
from sklearn.neighbours import KNeighboursClassifier
from sklearn.model_selection import test_train_split
from sklearn.metrics import confusion_matrix

df = pd.read_csv('iris.csv')
x = df[['sepal length'], ['sepal width'], ['petal length'],
        ['petal width']]
y = df['varieties']

x_train, x_test, y_train, y_test = train_test_split(x, y,
                                   test_size = 0.3, random_state = 1)

knn model = KNeighbourClassifier(n_neighbours = 7)
                                  // ↳ value of k
                                  // Default, k = 5
knn model.fit(x_train, y_train)
y_pred = knnmodel.predict(x_test)
print(confusion_matrix(y_test, y_pred))
      ↳ // there are other libraries
        // which calculate Accuracy, precision directly
```



Accuracy = (Sum of diagonal values) / (Sum of all)
Precision = (Single ~~column~~ Field value) / (Column sum)
Recall = (Single Field value) / (Row sum)