


# SUPER MARKET SALES PREDICTION

**TEAM SVM**

**GUIDE: DR.TANVEER SARDAR**

# CONTENT: -

- Problem definition
- Objectives
- Classifiers used
- Future scope
- Advantages and
- Disadvantages
- Challenges

# PROBLEM DEFINITION:

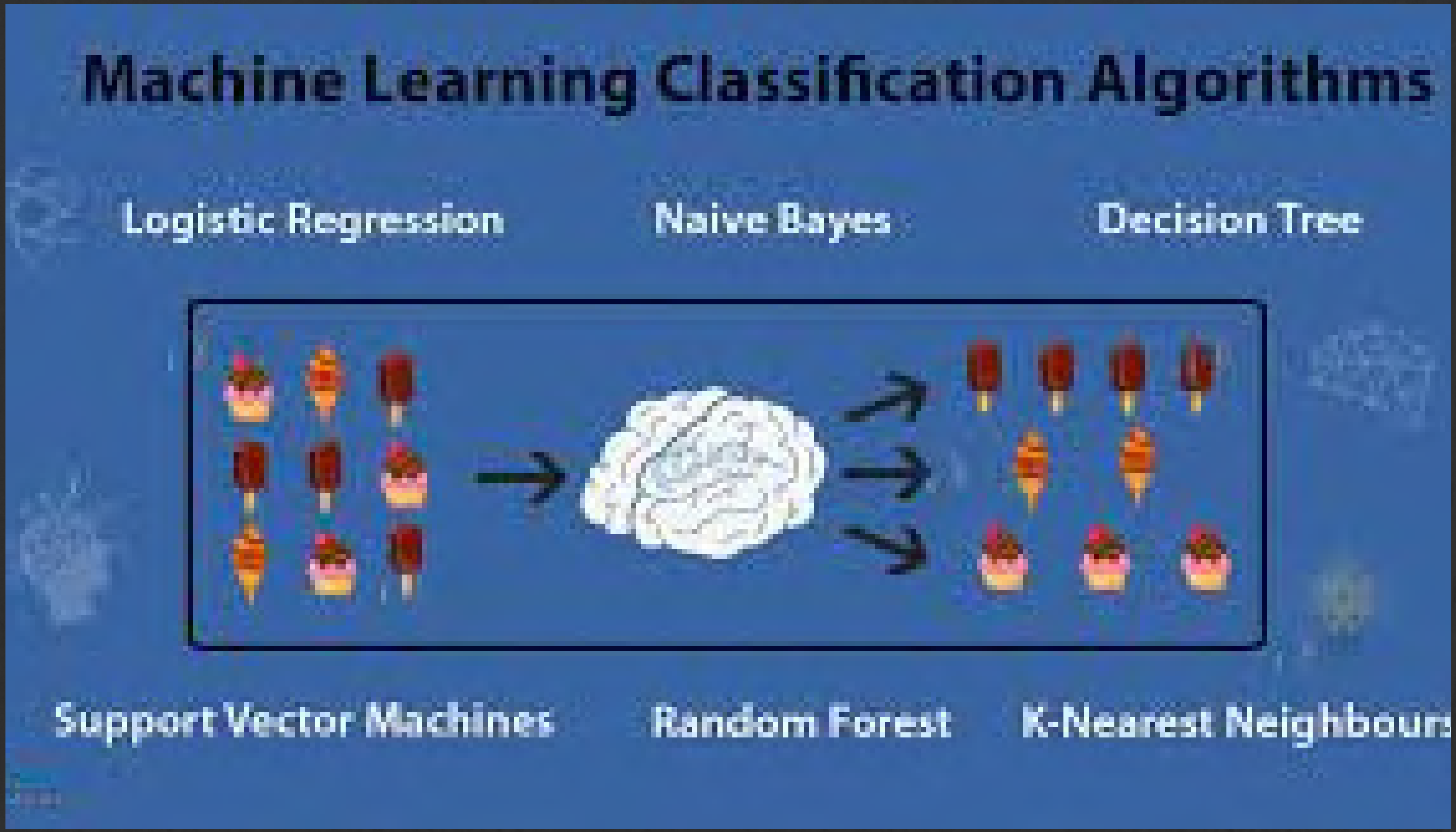
To find out what role certain properties of an item play and how they affect their sales by understanding Super Market sales. A predictive model can be built to find out for every store, the key factors that can increase their sales and what changes could be made to the product or store’s characteristics.

# STEP BY STEP PROCESS:

- Data Set gathering
- Analysing the data set
- Cleaning the data set
- Exploratory Data Analysis
- Visualization
- Testing and training the dataset
- Building the model
- Developing predictive system

# CLASSIFIERS USED:

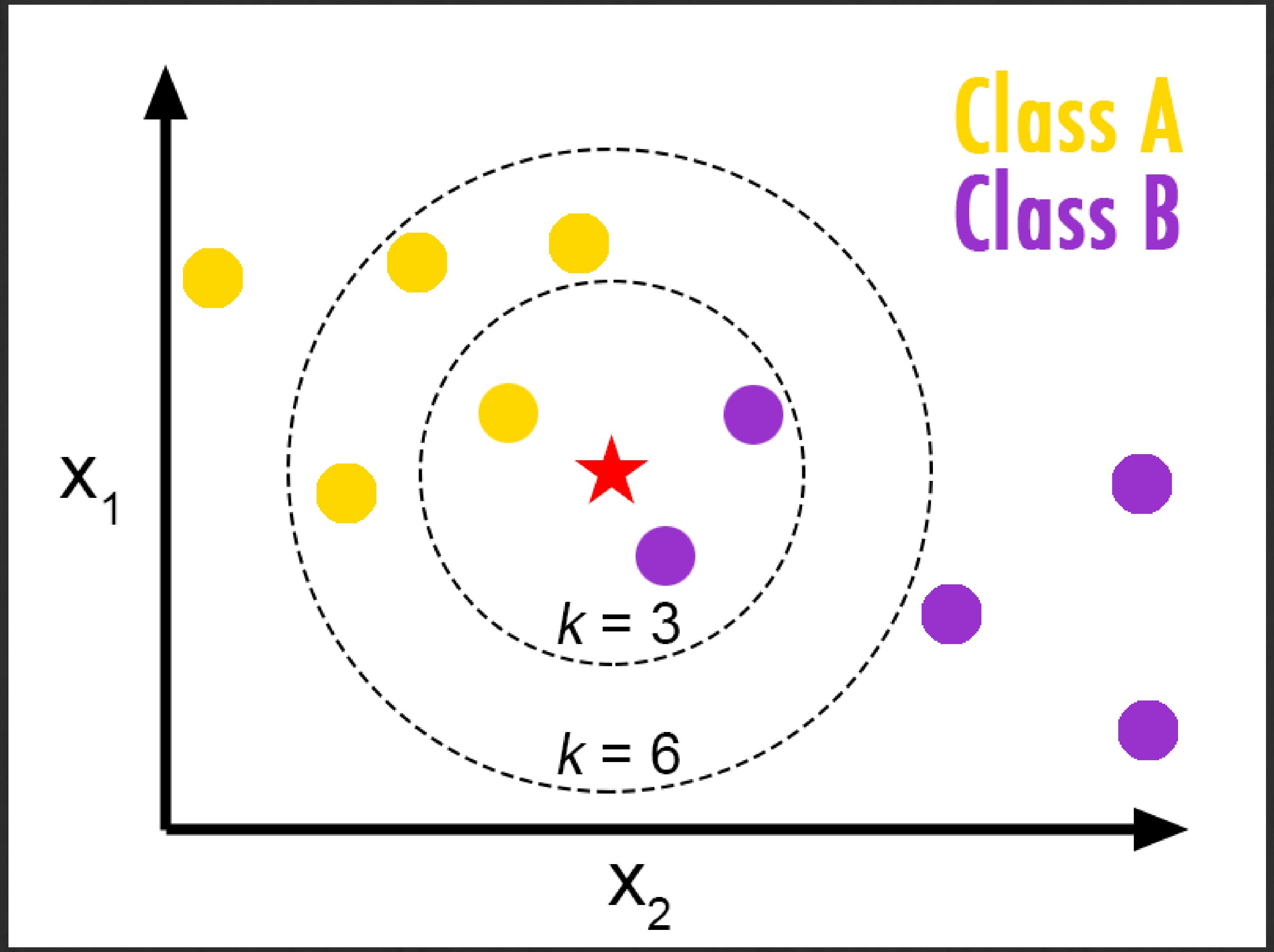
1. KNeighborsClassifier
2. Support Vector Systems
3. Naive Bayes
4. Decision tree classifier
5. Random Forest Classifier
6. AdaBoostClassifier
7. Gradient Boosting Classifier
8. XGBClassifier
9. ExtraTreesClassifier
10. Bagging Classifier



# KNeighborsClassifier:

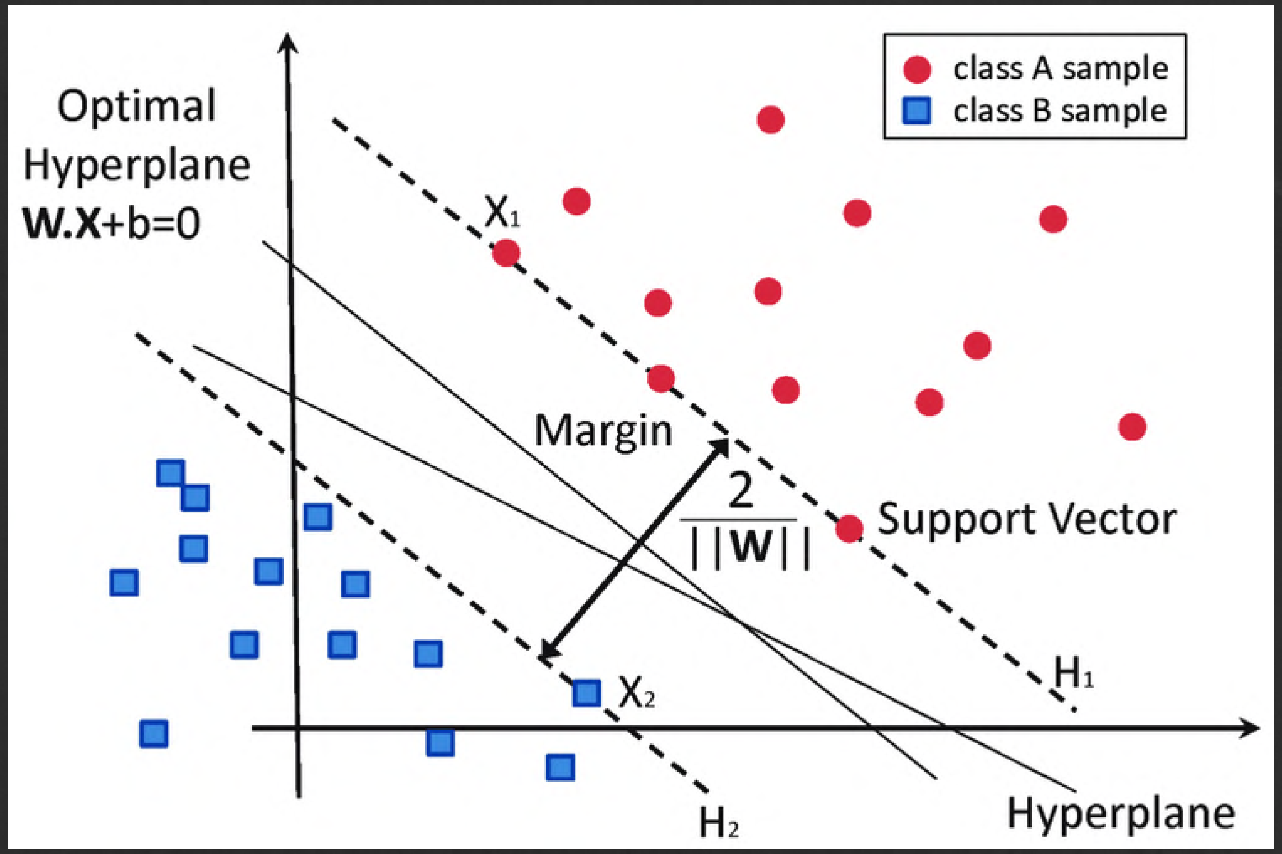
- K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.
- K NN algorithm stores all the available data and classifies a new data point based on the similarity.This means when new data appears then it can be easily classified into a well suite category
- This KNN algorithm is used for regression as well as for the classification but mostly it is used for the classification problems

sklearn.neighbors.KNeighborsClassifier(n_neighborsint, default=5)



# Support Vector Systems:

- Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.
- The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future.




**sklearn.svm import SVC**

# Naive Bayes:

- Naïve Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems.
- It is mainly used in text classification that includes a high-dimensional training dataset.
- It is a probabilistic classifier, which means it predicts on the basis of the probability of an object.

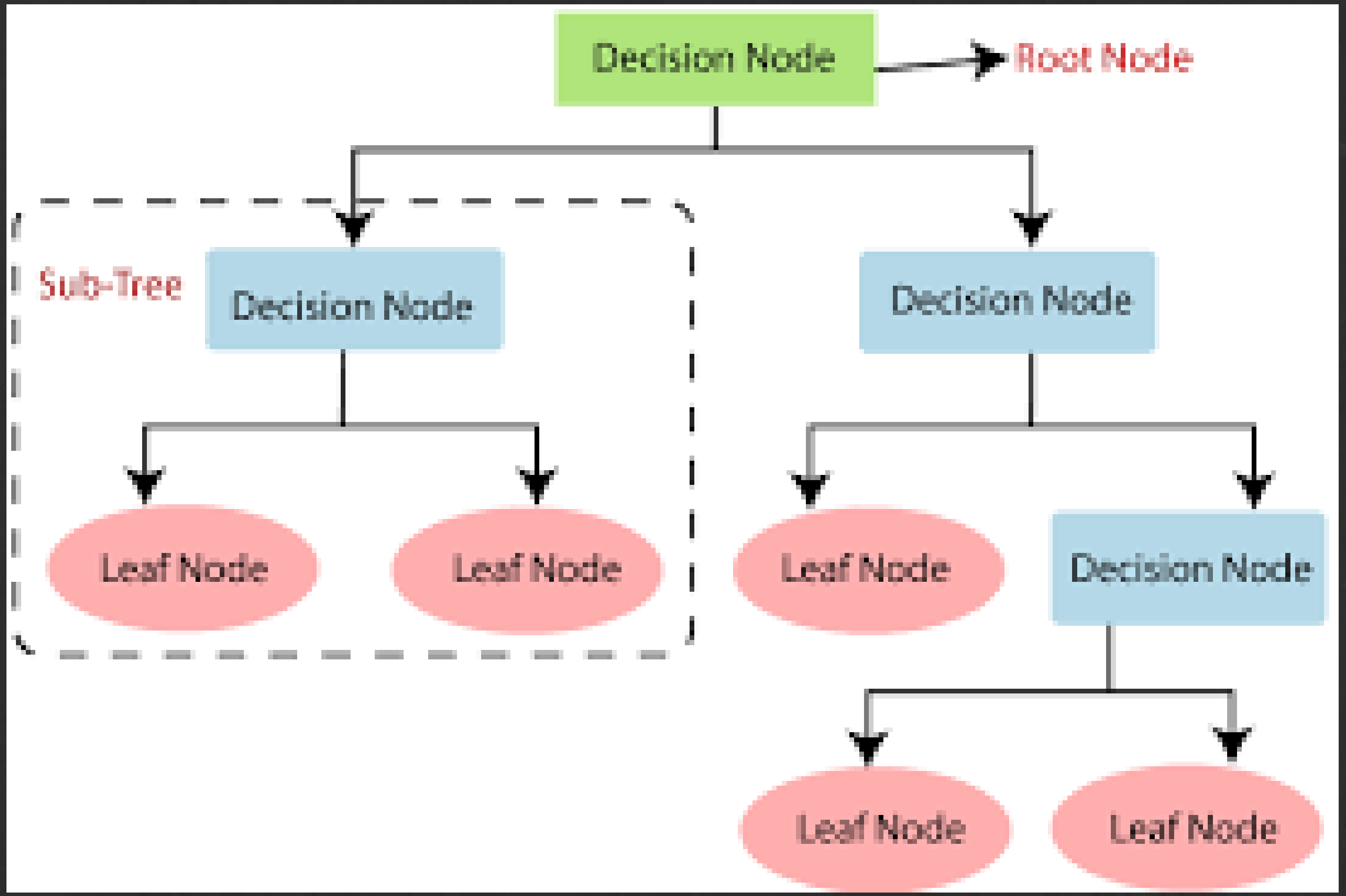
P(A|B) is Posterior probability: Probability of hypothesis A on the observed event B.

P(B|A) is Likelihood probability: Probability of the evidence given that the probability of a hypothesis is true

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

# Decision tree classifier

- Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome
- fromsklearn.treeimport DecisionTreeClassifier
  dtree=DecisionTreeClassifier(max_depth=6, random_state=123,criterion='entropy')



# Random Forest Classifier:

- 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.
- Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset.

```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
```

# AdaBoostClassifier:

**An AdaBoost classifier is a meta-estimator that begins by fitting a classifier on the original dataset and then fits additional copies of the classifier on the same dataset but where the weights of incorrectly classified instances are adjusted such that subsequent classifiers focus more on difficult cases.**

```
from sklearn.ensemble import AdaBoostClassifier
adb = AdaBoostClassifier(base_estimator = None)
```

# Gradient Boosting Classifier:

Gradient boosting classifiers are a group of machine learning algorithms that combine many weak learning models together to create a strong predictive model. Decision trees are usually used when doing gradient boosting.

```
from sklearn.ensemble import GradientBoostingClassifier
gbc=GradientBoostingClassifier()
```

# XGBClassifier :

XGBoost, which stands for Extreme Gradient Boosting, is a scalable, distributed gradient-boosted decision tree (GBDT) machine learning library. It provides parallel tree boosting and is the leading machine learning library for regression, classification, and ranking problems.

# ExtraTreesClassifier:

It is a type of ensemble learning technique which aggregates the results of multiple de-correlated decision trees collected in a “forest” to output it's classification result.

```
from sklearn.ensemble import ExtraTreesClassifier
etc = ExtraTreesClassifier(n_estimators=100, random_state=0)
```

# Bagging Classifier:

A Bagging classifier is an ensemble meta-estimator that fits base classifiers each on random subsets of the original dataset and then aggregate their individual predictions (either by voting or by averaging) to form a final prediction.

# KNN classifier:

| Product Line | Unit Price | Quantity | Rating |
|---|---|---|---|
| Health and Beauty | 74.69 | 7 | 9.1 |
| Electronic and accessories | 15.28 | 5 | 9.6 |
| Sport and Travel | 86.31 | 7 | 5.3 |
| Food and Beverages | 54.84 | 3 | 5.9 |
| Fashion | 17.94 | 5 | ? |

Distance Equation : $\sqrt{(x2-x1)^2+(y2-y1)^2}$

1st:

| | | |
|---|---|---|
| Health and Beauty | 74.69 | 7 |

$$\sqrt{(17.94-74.69)^2+(5-7)^2}$$

$$\sqrt{(-56.75)^2+(-2)^2}$$

$$=\sqrt{3224.56}$$

$$=56.78 \quad \text{distance}$$

2nd:

| | | |
|---|---|---|
| Electronic accessories | 15.28 | 5 |

$$\sqrt{(17.94-15.28)^2+(5-5)^2}$$

$$\sqrt{(2.66)^2}$$

=2.66

3rd:

| | | |
|---|---|---|
| Sport and Travel | 86.31 | 7 |

$$\sqrt{(17.94-86.31)^2+(5-7)^2}$$

$$\sqrt{4674.4+4}$$

$$\sqrt{4678.4}$$

= 68.39

4th:

| | | |
|---|---|---|
| Food and Beverages | 54.84 | 3 |

$$\sqrt{(17.94-54.84)^2+(5-3)^2}$$

$$\sqrt{(-36.9)^2+4}$$

=36.9

5th:

| | | |
|---|---|---|
| Home and Life style | 73.56 | 10 |

$$\sqrt{(17.94-73.56)^2+(5-10)^2}$$

$$\sqrt{(-55.92)^2+(-5)^2}$$

$$\sqrt{3152.04}$$

=56.14

| Product Line | Unit Price | Quantity | Distance | Ratio |
|---|---|---|---|---|
| Health | 74.69 | 7 | 56.78 | 9:1 |
| Electronic | 15.28 | 5 | **2.66** | 9:6 |
| Sport | 86.31 | 7 | 68.39 | 5:3 |
| Food | 54.84 | 3 | **36.9** | 5:9 |
| Home | 73.56 | 10 | **56.14** | 8 |

Now K=3

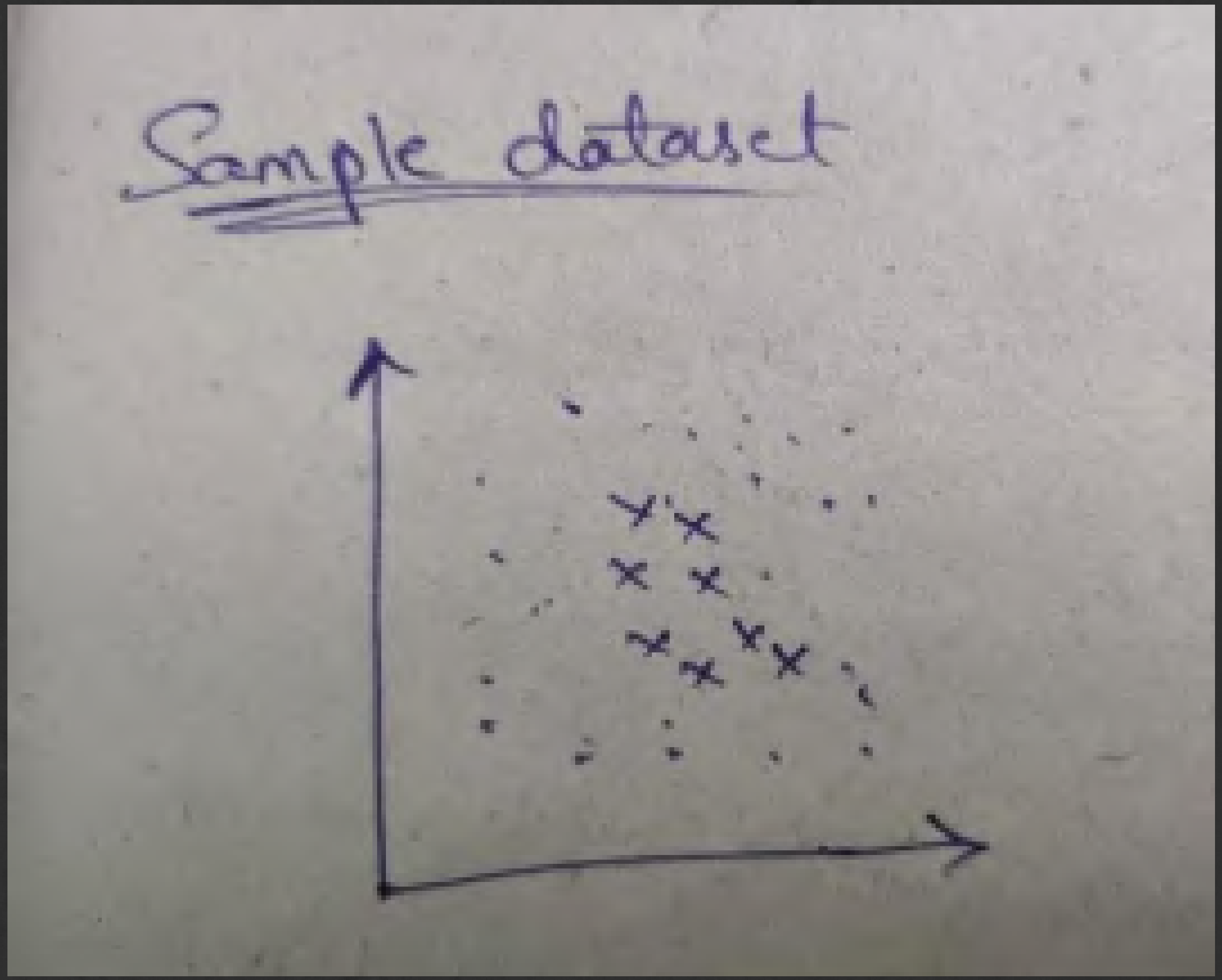
We should take 3 closest distance of fashion

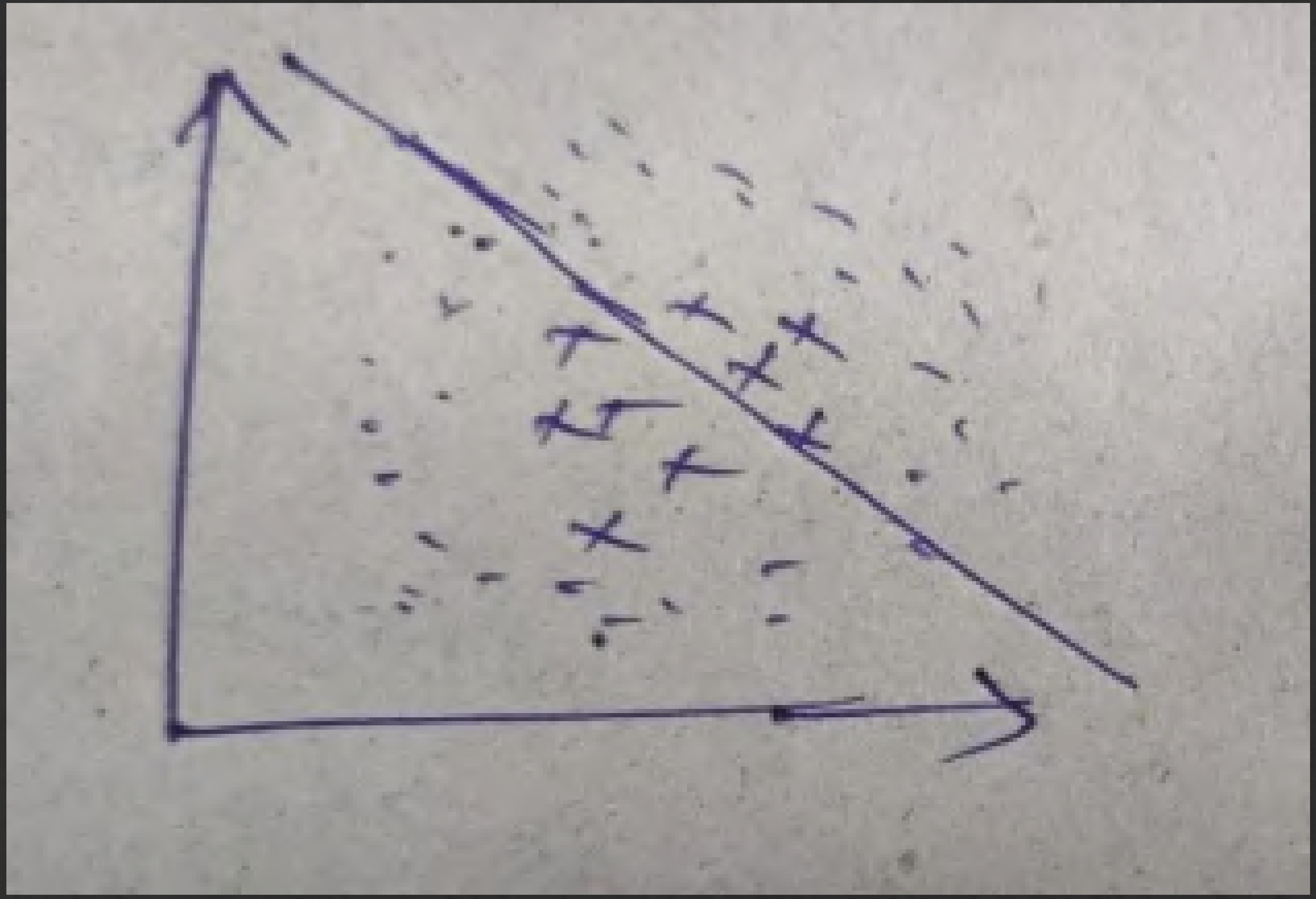
| | | |
|---|---|---|
| Electronic | 2.66 | 9:6 |
| Food | 36.9 | 5:9 |
| Home | 56.14 | 8 |

So, Fashion can get the rating in between 9:6, 5:9 , 8

# SVM classifier:

Explanation:
Here in SVM we do problems with graphs Here we take points as assumption only
Sample datset:
Step 1: Segregate the dataset into 2 classes
The Line which divides the data set is called as Hyper Lane.



we have to find nearest point, after finding the nearest point, we have to draw plane for 2 nearest points we have to find the distance between the Marginal plane.

In the above equation b is value through which the intercept passes

So to compute the distance between two marginal planes we will consider two nearer points as x1 and x2

equations are :

$$w^t x_i + b = -1$$

$$w^t x_2 + b = +1$$

so difference between these equations are:

$$w^t x_1 + b = -1$$

$$w^t x_2 + b = +1$$

---

$$W^t(x_2 - x_1) = 2$$

We have to find x2-x1 so to remove wt we have to divide the whole equation by ||w|| i.e norm of w

So $\frac{2}{||w||}$ is our optimizing function

So we need to maximize it

Such that $y_i = \begin{cases} 1, & wtx + b \geq 1 \\ 1 & wtx + b \leq -1 \end{cases}$

So finally $y_i = w^t x_i + b_i \geq 1$

When we compute this equation it should be greater than or equal to 1 otherwise it is treated as misclassification
If we get misclassification we will minimize the function.

$$\text{Min}\ \frac{||w||}{2} + c \sum_{i=1} \alpha i$$

As it is difficult to classify all the data points which are present in dataset we used other classifier to get the better accuracy .

# NAIVE BAYES CLASSIFIER:

$$p(a/b) = \frac{p(b/a) * p(a)}{p(b)}$$

$p(a/b)$=it is known as posterior probability

$P(a)$=it is known as marginal probability

Data set : $x=\{x_1,x_2,x_3,\ldots\ldots,x_n\}$

$$p(y/x_1,x_2,x_3,\ldots.x_n) = \frac{p(x_1/y)p(x_2/y)p(x_3/y)\ldots\ldots p(x_n/y) * p(y)}{p(x_1)p(x_2)p(x_3)\ldots\ldots p(x_n)}$$

$$= \frac{p(y)\pi_{i=1}^{n}\ p(x_i/y)}{p(x_1)p(x_2)p(x_3)\ldots\ldots p(x_n)}$$

$$P(y/x_1, x_2, \ldots\ldots x_n) \alpha p(y)\pi_{i=1}^{n}\ p(x_i/y)$$

$$Y = \text{ arg max } p(y)\pi_{i=1}^{n} p(x_i/y)$$

**Example:**

**Outlook:**

| | yes | no | P(y) | P(n) |
|---|---|---|---|---|
| Sunny | 2 | 3 | 2/9 | 3/5 |
| Overcast | 4 | 0 | 4/9 | 0/5 |
| Rainy | 3 | 2 | 3/9 | 2/5 |
| total | 9 | 5 | 100% | 100% |

**Temperature:**

| | yes | no | P(y) | P(n) |
|---|---|---|---|---|
| Hot | 2 | 2 | 2/9 | 2/5 |
| Mild | 4 | 2 | 4/9 | 2/5 |
| Cool | 3 | 1 | 3/9 | 1/5 |
| total | 9 | 5 | 100% | 100% |

We are finding for the probability for the given condition

$$p\left(\frac{yes}{today}\right) = \frac{p\left(\frac{sunny}{yes}\right) * p\left(\frac{hot}{yes}\right) * p(yes)}{p(today)}$$

=2/9*2/9*9/14

=0.031

$$p\left(\frac{no}{today}\right) = \frac{p\left(\frac{sunny}{no}\right) * p\left(\frac{hot}{no}\right) * p(no)}{p(today)}$$

=3/5*2/5*5/14

=0.08571

Play:

| | | P(y)&p(n) |
|---|---|---|
| Yes | 9 | 9/14 |
| No | 5 | 5/14 |
| total | 14 | 100% |

$$P(yes) = \frac{0.031}{0.031+0.08571}$$

=0.27

P(n)=1-0.27

=0.73.

# DECISION TREE CLASSIFICATION:

Formulas:

Information gain,

$$I(p,n) = -\frac{p}{s}\log_2\frac{p}{s} - \frac{n}{s}\log_2\frac{n}{s}$$

S=total sample space

S=(p+n)

$$log_2 = \frac{log_{10}x}{log_{10}2}$$

Entropy:

$$E(a) = \sum_{i=1}^{v}\frac{p_{i+n_i}}{p+n}i(p_i,n_i)$$

Gain:

$i$=1

$$gain(a) = I(p, n) - E(a)$$

DATA SET:

| Day | Outlook | Temperature | Humidity | Wind | Play |
|---|---|---|---|---|---|
| 1 | Sunny | Hot | High | Weak | No |
| 2 | Sunny | Hot | High | Strong | No |
| 3 | Sunny | Hot | High | Weak | Yes |
| 4 | Overcast | Mild | High | Weak | Yes |
| 5 | Rain | Cool | Normal | Weak | Yes |

| | | | | | |
|---|---|---|---|---|---|
| 6 | Rain | Cool | Normal | Strong | No |
| 7 | Overcast | Cool | Normal | Strong | Yes |
| 8 | Sunny | Mild | High | Weak | No |
| 9 | Sunny | Cool | Normal | Weak | Yes |
| 10 | Rain | Mild | Normal | Weak | Yes |
| 11 | Sunny | Mild | Normal | Strong | Yes |
| 12 | Overcast | Mild | High | Strong | Yes |
| 13 | Overcast | Hot | Normal | Weak | Yes |

| | | | | | |
|---|---|---|---|---|---|
| 14 | rain | Mild | High | Strong | No |

Information gain=$-\left[\frac{9}{14}\log_2\frac{9}{14}+\frac{5}{14}\log_2\frac{5}{14}\right]$

=0.409+0.530

=0.940

Calculate entropy for outlook 1.sunny(5)-yes(2) No(3)

I gain(outook,sunny)=-$\left[\frac{2}{5}\ \log_2\frac{2}{5}+\frac{3}{5}\log_2\frac{3}{5}\right]$

=0.971

I gain(outlook,overcast)=0 I gain (outlook,rain)=0.971

Entropy of outlook=

$$E(a)=\sum_{i=1}^{v}\frac{p_{i+n_i}}{p+n}i(p_i,n_i)$$

=0.694

## Gain:

$$gain(a) = I(p, n) - E(a)$$

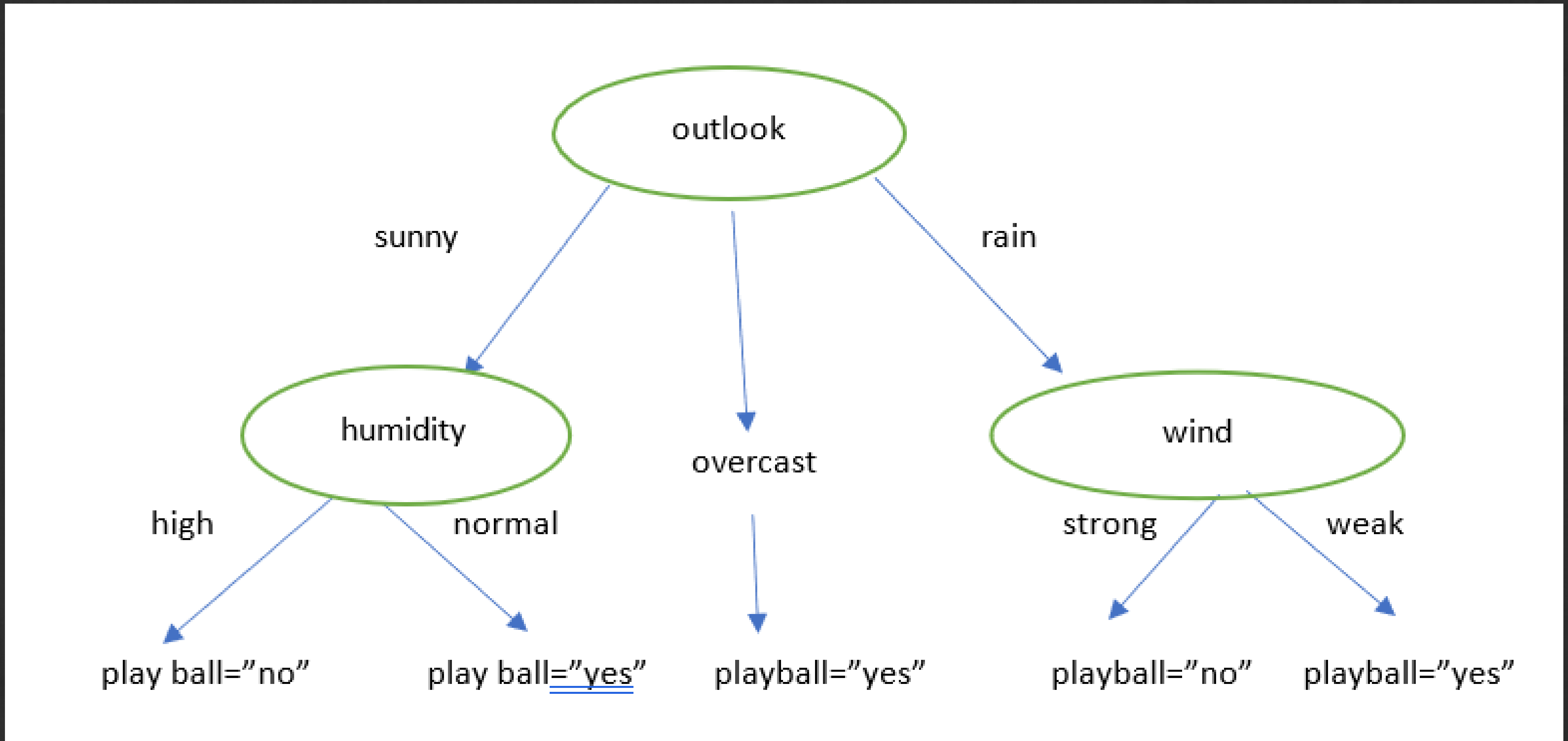
=0.940-0.694
=0.246

Here we have to repeate the same process for temperature,humidity,wind.
After finding all values we have to collect the gain values to draw the decision tree.

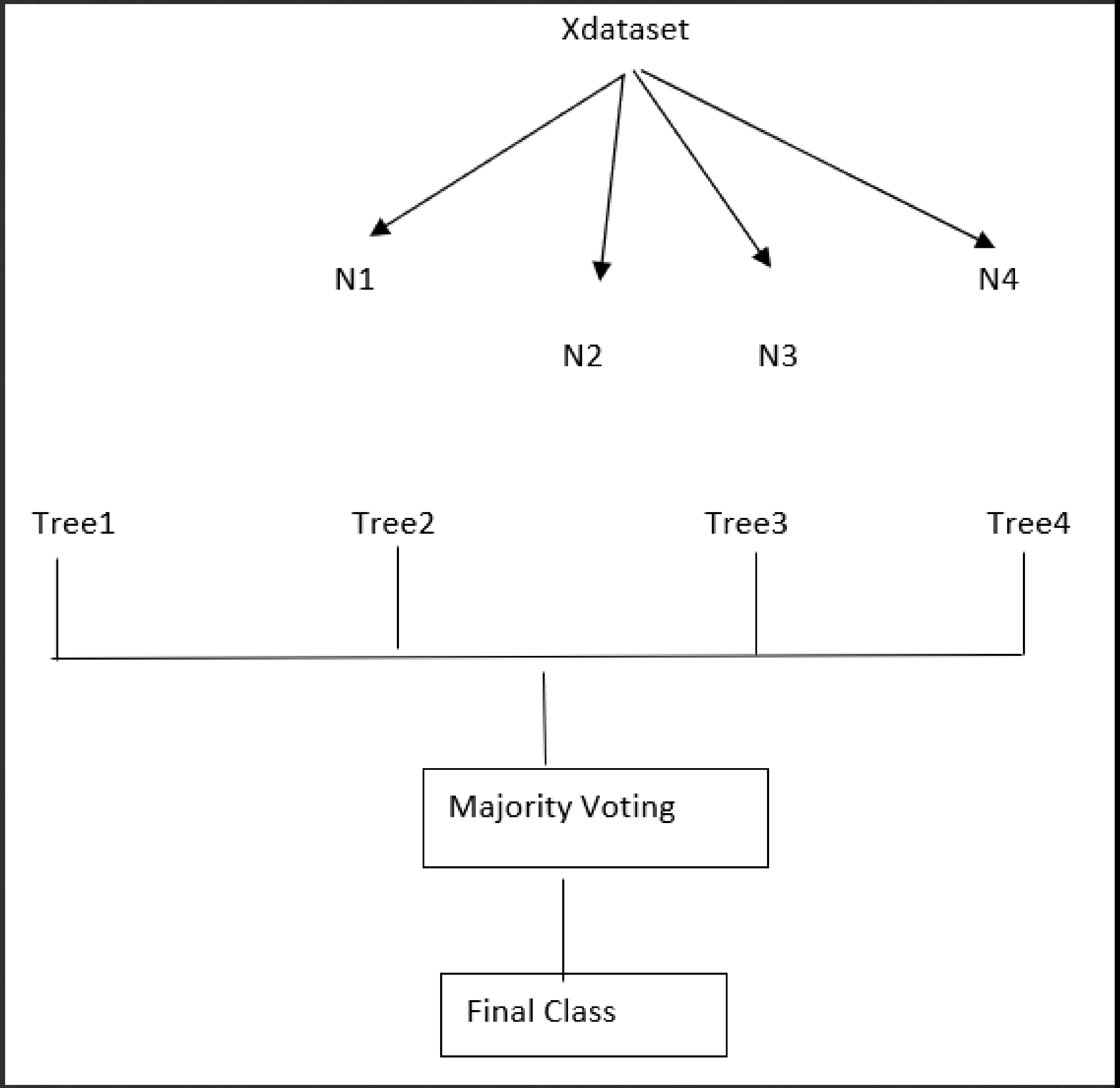
Gain of each attribute:
Outlook-for outlook the gain value is =0.246
Temperature-for temperature the gain value is=0.029
Humidity-for humidity the gain value is
=0.151
Wind-for wind the gain value is =0.048

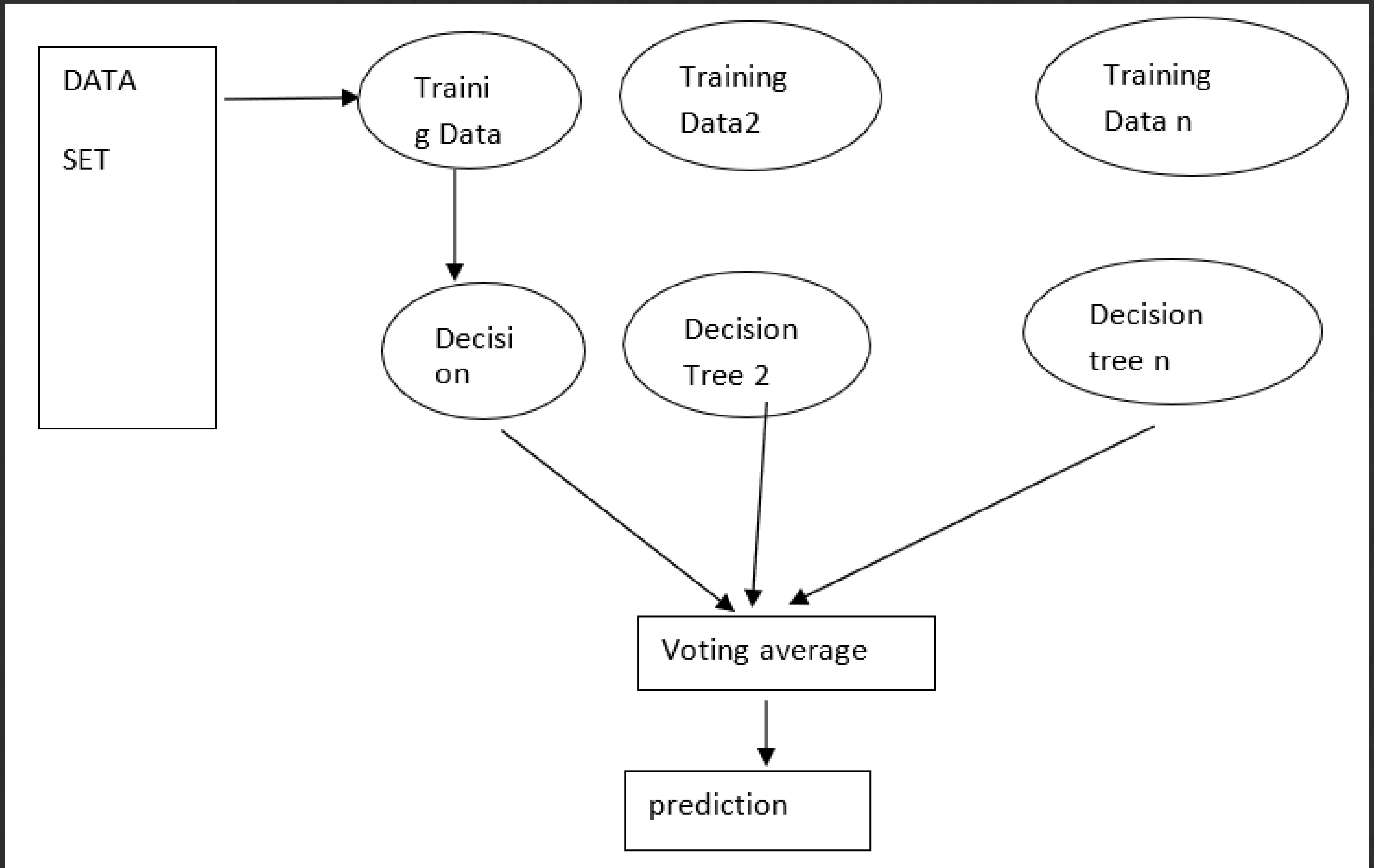
According to the highest gain value the first splitting attribute can be found.
After finding the first splitting attribute we have to do the second splitting attribute.for that again we have to find the entropy values and gain values. like that we have to draw the decision tree.

# DECISION TREE:



# Random forest algorithm:



In random forest algorithm the given data set is divided into n types of Training data set from that decision are generated. then averaging is done with the help of decision Trees and the final output is the Prediction.

Step1: Select random k data points from the training set.
Step2: build the decision trees associated with the selected dada points
Step3:choose the number N for decision trees that you want to build.
Step:4 Repeat step1 & step2
Step5: For new data points find the Predictions of each decision tree

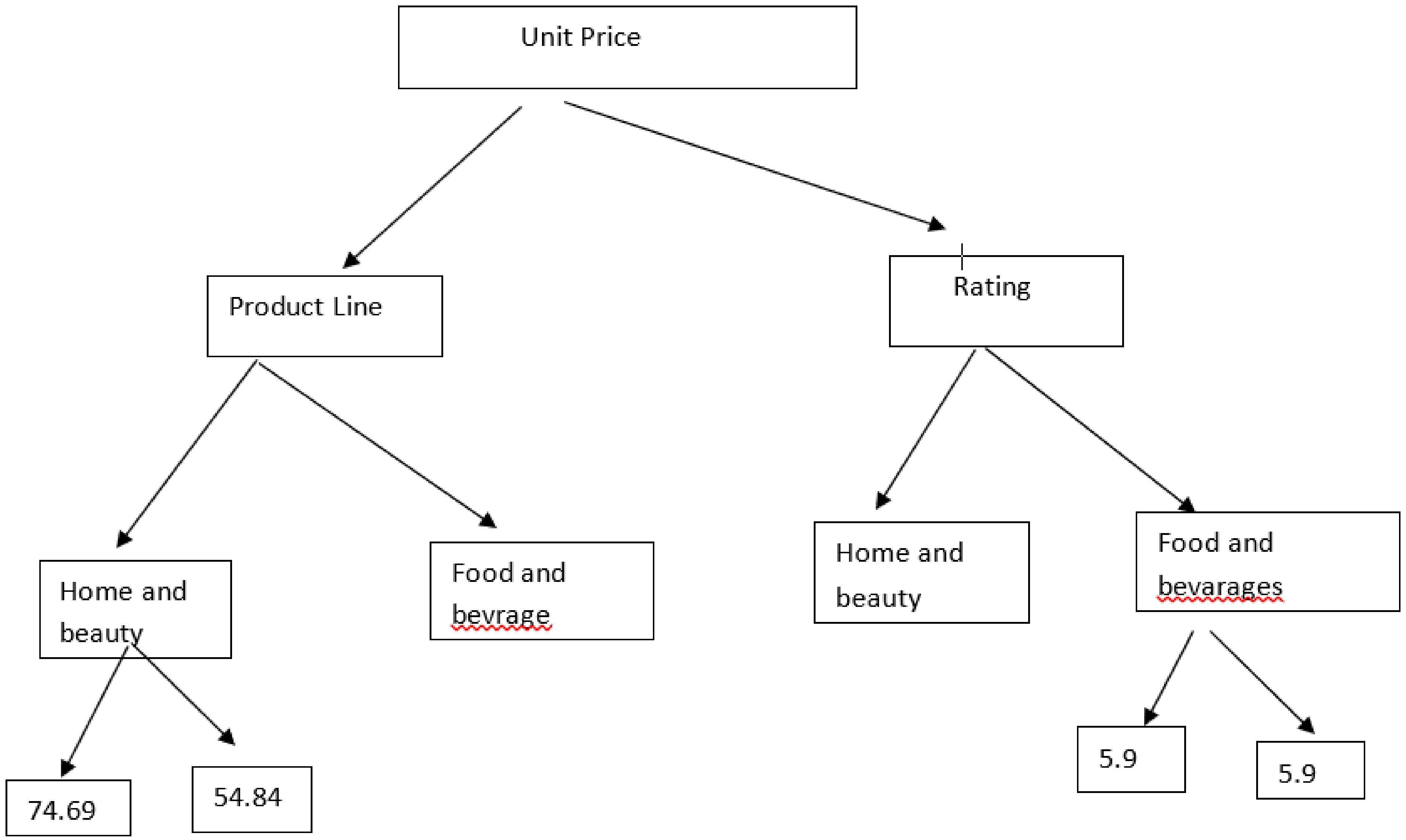
| Product Line | Unit Price | Quantity | Rating |
|---|---|---|---|
| Health and Beauty | 74.69 | 7 | 9:.1 |
| Electronic | 15.28 | 5 | 9.6 |
| Sport and travel | 86.31 | 7 | 5.3 |
| Food and bevarege | 54.84 | 3 | 5.9 |
| Home and life style | 73.56 | 10 | 8 |

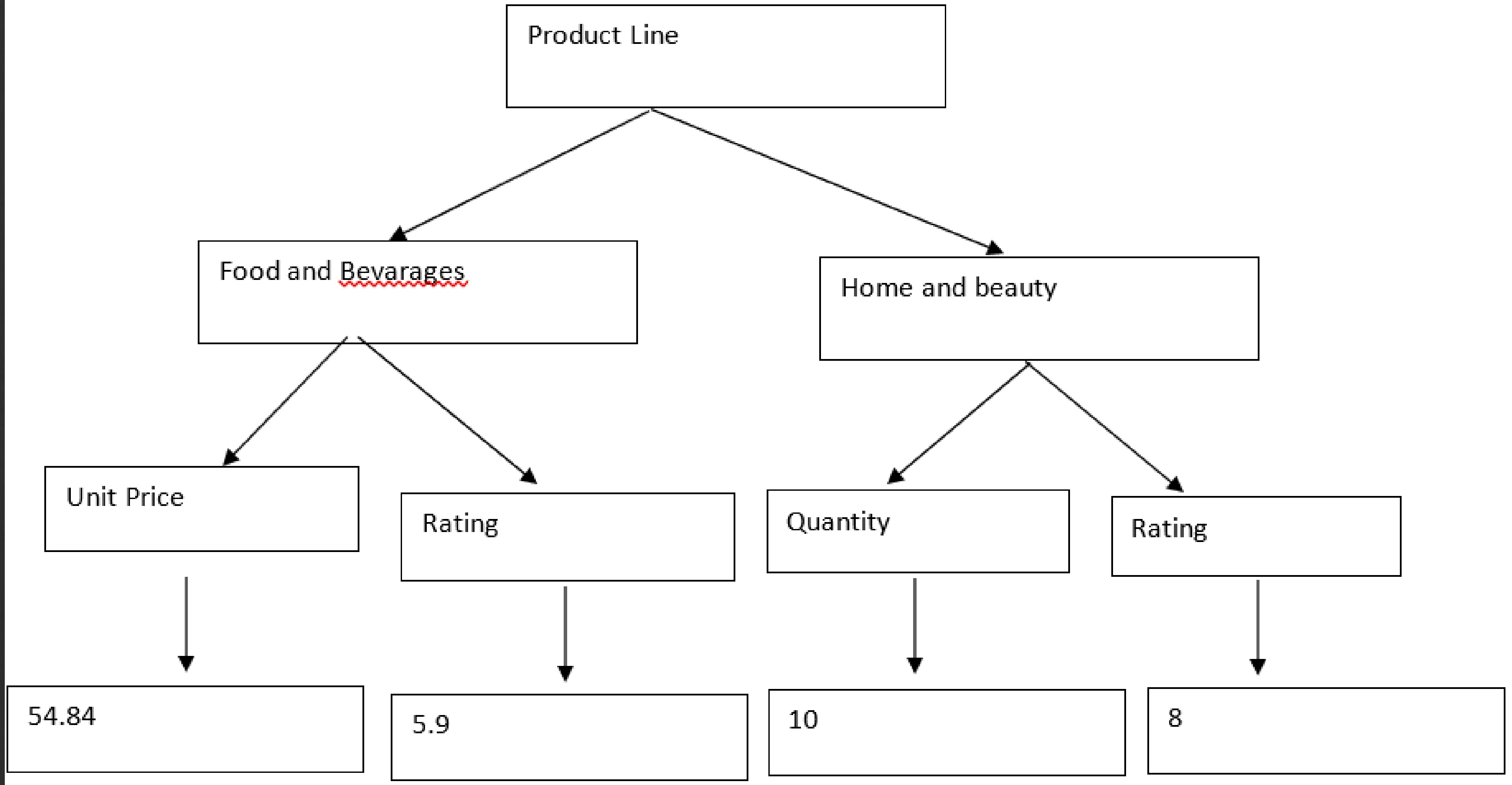
These Sample helps to build the decision trres
We should create a bootstrapped Dataset. Bootstrapped dataset is Nothing but selecting the random Samples from the dataset. We can repeat the samples in the Bootstraped data set to get better accuracy values.

## Bootstrapped Data:

| Product Line | Unit Price | Quantity | Rating |
|---|---|---|---|
| Health and Beauty | 74.69 | 7 | 9:.1 |
| Food and Bevareges | 54.84 | 3 | 5.9 |
| Home and life style | 73.56 | 10 | 8 |
| Sports and Travel | 86.31 | 7 | 5.3 |
| Food and bevareges | 54.84 | 3 | 5.9 |

In the above bootstrapped data Second Sample is repeated to get better accuracy.
From this bootstraped data set we have to take 2 variables so randomly unit price and quantity are taken as root nodes.




**therefore the rating of food and bevereges is 5.9**

# ADABOOST Algorithm:

| Product Line | Unit Price | Quantity | Rating | S weight |
|---|---|---|---|---|
| Health and Beauty | 74.69 | 7 | 9:.1 | 0.2 |
| Electronic accesories | 15.28 | 5 | 9.6 | 0.2 |
| Sport and travel | 86.31 | 7 | 5.3 | 0.2 |
| Food and bevarege | 54.84 | 3 | 5.9 | 0.2 |
| Home and life style | 73.56 | 10 | 8 | 0.2 |

We create stumps to predict if a rating of the product. we will make these predictions based on remaining variables.

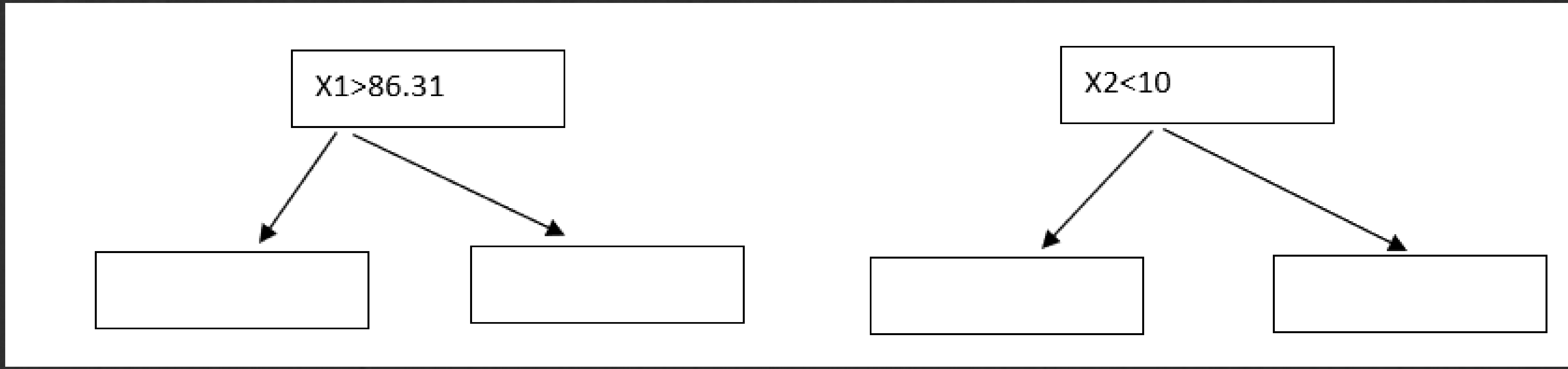
Step1:give each sample a weight which is nothing but sample weight At Start all the samples get the weight

$$\text{Sample weight} = \frac{1}{\text{total number of samples}}$$

$$=1/5$$

Now we have to create Stumps

let us treat unit price as x1 and Quantity as x2




This is model m1

We have to find error rate (α)

$\alpha = \frac{1}{2} ln$(1-error)/error

we will take electronics and sports for this prediction

so error model m1 is

error=0.4

$\alpha = \frac{1}{2} ln(1 - 0.4)/0.4$

$= = \frac{1}{2} ln(0.6/0.4)$

α=0.20

Now we have to find misweight for misclassified and correctly classified new_wt=corr_wt x eα1

new_wt=corr_wt x e-α1

wt=0.2 X e 0.2 =0.24

wt=0.2 X e-0.2 = 0.16

Updated sample weight are 0.24 and 0.16

Average of updated sample weight = 0.96

| X1 | X2 | Prediction | Updated |
|---|---|---|---|
| 74.69 | 7 | 9.1 | 0.16 |
| 15.28 | 5 | 9.6 | 0.24 |
| 86.31 | 7 | 5.3 | 0.24 |
| 54.84 | 3 | 5.9 | 0.16 |
| 73.56 | 10 | 8 | 0.16 |

The rating is between the range.

Take 5 random numbers between 5 to 10

5.3---9.1

5.9----9.6

8 9

9.1---9.3

9.3---9.6

# Gradient Boosting:

| Unit Price | Quantity | Rating | Yn r1 |
|---|---|---|---|
| 74.69 | 7 | 9.1 | 7.5 1.6 |
| 15.28 | 5 | 9.6 | 7.5 2.1 |
| 86.31 | 7 | 5.3 | 7.5 2.2 |
| 54.84 | 3 | 5.9 | 7.5 1.6 |
| 73.56 | 10 | 8 | 7.5 0.5 |

Step1:

Creating a Base Model Output of the predicted Base Model is average of rating

$$= \frac{9.1 + 9.6 + 5.3 + 5.9 + 8}{5}$$

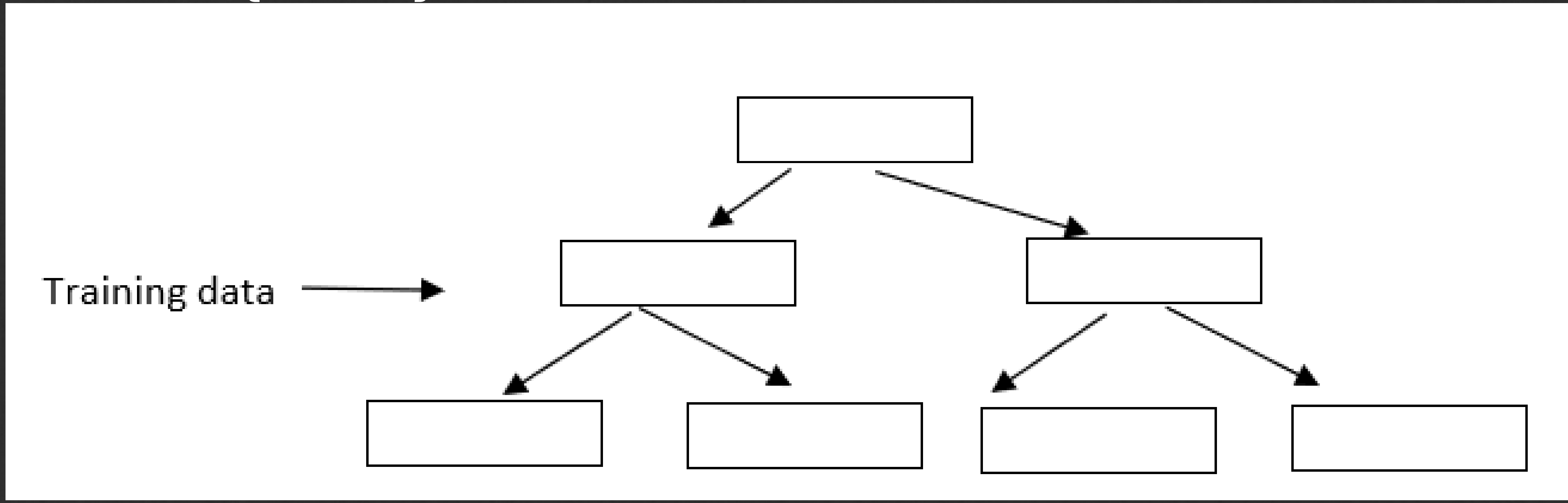
$= 7.58$

$Y=7.5$

Step2

Compute residual or errors R1=(actual-predicted)

Step3:
Create a Decision tree (xi , R1)
Here xi is unit price and quantity R1 is residual




This decision tree is with independent variable and output will be residual If we repeatthissystem we will get another residual R2

Final output = O/P of base model + residuals
=7.58+(1.6)
=9.18

Therefore 9.16 is nearer to 9.1 but to avoid outfitting we will use another algorithms.

# XGB CLASSIFIER:

| customer | Product line | Quantitiy A | res |
|---|---|---|---|
| Member | Health&beauty | 7 | -0.2 |
| Normal | Electric access | 5 | 1.8 |
| Normal | Home&lifestyle | 7 | -0.2 |
| Member | Health&beauty | 8 | -1.2 |
| Normal | Sports&travel | 7 | -0.2 |

PROBABILITY=?

7+5+7+8+7/5

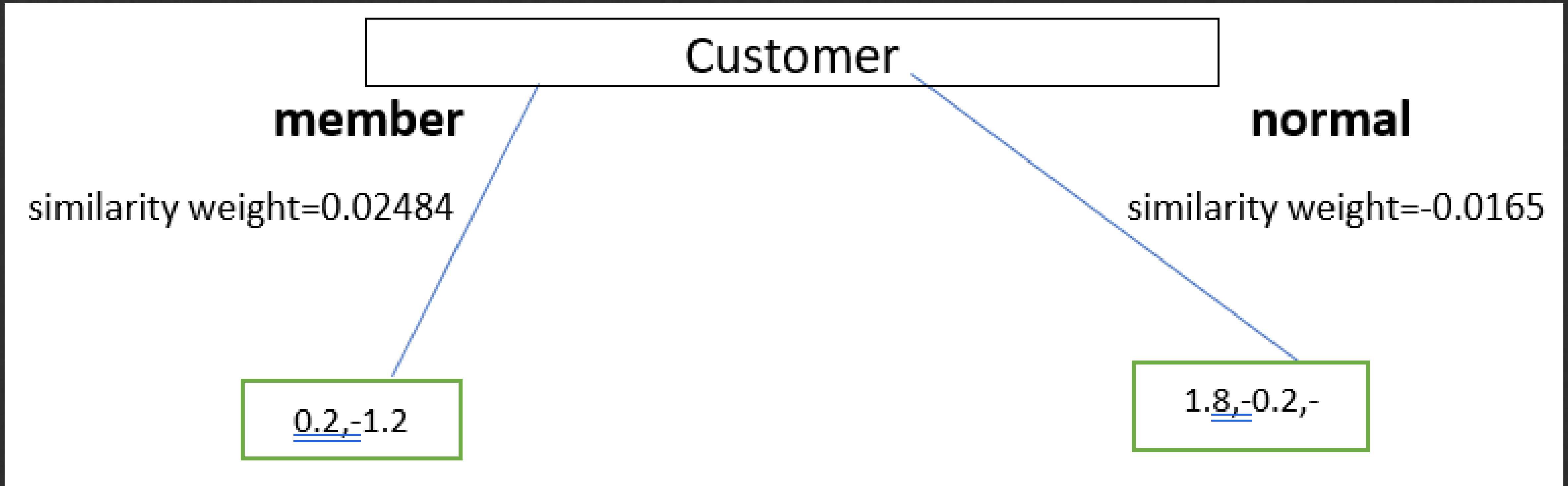
34/5

6.8

Binary tree is constructed based on customer

{-0.2,1.8,-0.2,-1.2,-0.2}

## 1/ constructing tree with root:




## 2/calculating the similarity weight:

$$x = \frac{\sum(res^2)}{\sum(pr(1-pr)+\lambda)}$$

Where,
res=res value
pr=probability
=values to repeat no.of times according to tree constructed

similarity weight for member customer:

=(-0.2+(-1.2))^2/6.8(1-6.8)+ 6.8(1-6.8)

=-1.96/-78.88

=1.96/78.88

=0.02484

similarity weight for normal customer:

=((1.8)+(-0.2)+(-0.2))^2/6.8(1-6.8)+ 6.8(1-6.8)+ 6.8(1-6.8)

=(1.4)^2/-118.32

=1.96/-118.32

=-0.0165

Calculating similarity weight for whole-

{-0.2,1.8,-0.2,-1.2,-0.2}

= [-0.2+1.8+-0.2+-1.2+-0.2]^2/6.8(1-6.8)+ 6.8(1-6.8)+ 6.8(1-6.8)+ 6.8(1-6.8)+ 6.8(1-6.8)

=0/-197.2

=0 -> (similarity weight for total res)

3/ calculating the gain-
= 0.02484+(-0.0165)-0
=0.00834
Finding the accuracy value for the base model:

Log(odd)=log(p/1-p) Where, P=probability
=log(6.8/1-6.8)
Log(-1.1724)
=0.069075

# EXTRA TREE CLASSIFIER:

| customer | Product line | quantity | gender | City |
|---|---|---|---|---|
| Member | Health & beauty | 7 | Female | Yangon |
| Normal | Electronic accessories | 5 | Female | Naypyitaw |
| Normal | Home & lifeline | 7 | Male | Yangon |
| Normal | Sports & travel | 7 | Male | Yangon |
| Member | Health & beauty | 8 | Male | Yangon |
| Normal | Electronic accessories | 7 | Male | Naypyitaw |
| Member | Electronic accessories | 6 | Female | Yangon |
| Normal | Home & lifeline | 10 | Female | Naypyitaw |
| Member | Health & beauty | 2 | Female | Yangon |
| Member | Food & beverages | 3 | Female | Mandalay |
| Member | Fashion accessories | 4 | Female | Mandalay |
| Member | Electronic accessories | 4 | Male | Mandalay |
| Normal | Electronic accessories | 5 | Female | Yangon |
| Normal | Food & beverages | 10 | Male | Yangon |

$$Entropy(S) = \sum_{i=1}^{C} -pi \ log_2(pi)$$

Where,

C=No.of unique class labels

Pi=proportional of rows with output label is i.

$$\text{Entropy(S)}=-9/14\log\left(\frac{9}{14}\right) - 5/14\log\left(\frac{5}{14}\right)$$

Entropy(S)=0.940

## 1st Decision Tree gets data with the features customer and product line

Gain(S,A)=Entropy(S)-$\sum_{VE} Values(A) \, {}^{|S_v|}/_{|S|}$Entropy($S_v$)

Gain(S,Customer)=0.940-(5/14(-2/5$\log_2\left(\frac{2}{5}\right) + -3/5\log_2(\frac{3}{5})) +$
$4/14(-4/4\log_2\left(\frac{4}{4}\right) + (-3/5\log_2\left(\frac{3}{5}\right) + -2/5\log_2\left(\frac{2}{5}\right)))$

Gain(S,Customer)=0.246

Similarly,

Gain(S,Productline)=0.029

## 2nd Decision Tree gets data with the features product line and Gender

Gain(S,Productline)=0.029 Gain(S,Gender)=0.048

3rd Decision Tree gets data with the features Customer and Quantity

Gain(S,Customer)=0.246 Gain(S,Quantity)=0.151

4th Decision Tree gets data with the features product line and Quantity

Gain(S,Productline)=0.029 Gain(S,Quantity)=0.151

5th Decision Tree gets data with the features Gender and Quantity

Gain(S,Gender)=0.048 Gain(S,Quantity)=0.151

Total Information Gain for each features

Total Info Gain for Customer=0.246+0.246

=0.492

Total Info Gain for Productline=0.029+0.029+0.029

=0.087

Total Info Gain for Quantity=0.151+0.151+0.151

=0.453

Total Info Gain for Gender=0.048+0.048

=0.096

Important variable to determine the output label according to the above constructed Extra tree Forest is the Feature "Customer".

## Bagging Classifier:

| customer | Product line | Quantitiy |
|---|---|---|
| Member | Health&beauty | 7 |
| Normal | Electric access | 5 |
| Normal | Home&lifestyle | 7 |
| Member | Health&beauty | 8 |
| Normal | Sports&travel | 7 |

To find the accuracy for bagging we use:

CR=C/A

Where,

CA=correct rate

C=Samples (health & beauty)

A=total no.of samples

CR=2/5

=0.4

Finding the entropy value :

E=Entropy

$$E=1/N*2/L-1*\sum_{j=1}^{N} \min\{(\sum_{i=1}^{L} y_{i,j}), (L-\sum_{i=1}^{L} y_{i,j})\}$$

$$=1/5\ 2/1-1 \sum_{j=1}^{5} \min\{(\sum_{i=5}^{1} 1), (1-\sum_{i=1}^{2} 1)$$

$$=2/5\{\sum_{j=1}^{5} 0\}$$

$$=0.4$$

FINDING PRECISION , F1 SCORE AND RECALL FOR THE CLASSIFIERS WE HAVE USED:

## PRECISION:

Formula:

$$t_P / (t_P + f_P)$$

Where,

tP is the number of true positives

is the number of false positives

The precision is intuitively the ability of the classifier not to label a negative sample as positive.

## RECALL:

FORMULA:

$$t_P / (t_P + f_n)$$

Where,

$t_p$ is the number of true positives

$f_p$ is the number of false negatives

The recall is intuitively the ability of the classifier to find all the positive samples

## F1 SCORE:

## FORMULA:

**2*(precision*recall)/(precision+recall)**

Where,
Precision is noted value above
Recall is the notes value above

## KNN:
## CONFUSION MATRIX:

[[49  51]
[55   45]]

**Precision:**
=49/(49+55)
=0.47
**Recall:**
=49/(49+51)
=0.49
**F1 score:**
=(2*(0.47*0.49))/((0.47+0.49) )
=0.4612/0.96
=0.48

## SVM:
## CONFUSION MATRIX:

[[49  51]
[60   40]]

**Precision:**
=49/(49+60)
=0.449
**Recall:**
=49/(49+51)
=0.49
**F1 score:**
=(2*(0.449*0.49))/((0.449+0.49) )
=0.44/0.939
=0.468

**NAIVE BAYES CLASSIFIER:**
**CONFUSION MATRIX:**

[[35 65]
[34 66]]

**Precision:**
=35/(35+34)
=0.507
**Recall:**
=35/(35+65)
=0.35
**F1 score:**
=(2*(0.507*0.35))/((0.507+0.35) )
=0.354/0.857
=0.413

**DECSION TREE CLASSIFIER:**
**CONFUSION MATRIX:**

[[79 21]
[73 27]]

**Precision:**
=79/(79+73)
=0.519
**Recall:**
=79/(79+21)
=0.79
**F1 score:**
=(2*(0.519*0.79))/((0.519+0.79) )
=0.82/1.309
=0.626

## Random forest classifier:

**Precision:**

Precision= 0.53

**Recall:**

Recall=0.50

**F1 score:**

$$= \frac{2*(0.53*0.53)}{0.53+0.53}$$

F1 score=0.51

## Ada boost algorithm:

**Precision:**

=0.54

**Recall:**

$$= \frac{54}{54+46} = 0.54$$

**F1 score:**

$$\frac{2(0.54*0.54)}{1.08}$$

F1 score: 0.53

## Gradient boosting algorithm:

**Precision:**

=54/100
=0.50

**Recall:**

=54/54+46
=0.48

**F1 score:**

$$\frac{2(0.54*0.54)}{1.08}$$

=0.48

## XGB classifier:

| customer | Product line | Quantitiy | res |
|---|---|---|---|
| Member | Health&beauty | 7 | -0.2 |
| Normal | Electric access | 5 | 1.8 |
| Normal | Home&lifestyle | 7 | -0.2 |
| Member | Health&beauty | 8 | -1.2 |
| Normal | Sports&travel | 7 | -0.2 |

Health and beauty = 2(positive)
Other samples=3(negative)
Total samples=5

**Precision :**

2/(2+2)
=2/4
=0.5

**Recall:**

2/(2+3)
= 2/5
=0.4

**F1 score:**

2*(0.5*0.4)/0.5+0.4
=2*(0.2)/0.9
=0.444

# Extra Tree Classifier:

| customer | Product line | quantity | gender | City |
|---|---|---|---|---|
| Member | Health & beauty | 7 | Female | Yangon |
| Normal | Electronic accessories | 5 | Female | Naypyitaw |
| Normal | Home & lifeline | 7 | Male | Yangon |
| Normal | Sports & travel | 7 | Male | Yangon |
| Member | Health & beauty | 8 | Male | Yangon |
| Normal | Electronic accessories | 7 | Male | Naypyitaw |
| Member | Electronic accessories | 6 | Female | Yangon |
| Normal | Home & lifeline | 10 | Female | Naypyitaw |
| Member | Health & | 2 | Female | Yangon |
| Member | Food & beverages | 3 | Female | Mandalay |
| Member | Fashion accessories | 4 | Female | Mandalay |
| Member | Electronic accessories | 4 | Male | Mandalay |
| Normal | Electronic accessories | 5 | Female | Yangon |
| Normal | Food & beverages | 10 | Male | Yangon |

**Precision:**

7/(7+7)
=7/14
=1/2
=0.5

**Recall:**

7/(7+7)
=0.5

**F1 Score:**

2*(0.5*0.5)/(0.5+0.5)
=2*(0.25)/1
=0.5

## Bagging Classifier:

| customer | Product line | Quantitiy |
| --- | --- | --- |
| Member | Health&beauty | 7 |
| Normal | Electric access | 5 |
| Normal | Home&lifestyle | 7 |
| Member | Health&beauty | 8 |
| Normal | Sports&travel | 7 |

**Precision:**

2/(2+2)
=2/4
=0.5

**Recall:**

2/(2+3)
=2/5
=0.4

**F1 Score:**

2*(0.5*0.4)/(0.5+0.4)
=0.4/0.9
=0.444

# OUTPUTS FOR ALL THE CLASSIFIERS USED:

## KNeighborsClassifier:

In [42]:
```
y_pred=knn.predict(x_test)
from sklearn.metrics import accuracy_score,classification_report,confusion_matrix
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
print("Classification Report is:\n",classification_report(y_test,y_pred))
print("Confusion Matrix:\n",confusion_matrix(y_test,y_pred))
print("Training Score:\n",knn.score(x_train,y_train)*100)
```

```
Classification Report is:
              precision    recall  f1-score   support

           0       0.47      0.49      0.48       100
           1       0.47      0.45      0.46       100

    accuracy                           0.47       200
   macro avg       0.47      0.47      0.47       200
weighted avg       0.47      0.47      0.47       200

Confusion Matrix:
 [[49 51]
 [55 45]]
Training Score:
 64.75
```

## Support Vector Systems:

In [73]:
```
y_pred=svc.predict(x_test)
from sklearn.metrics import accuracy_score,classification_report,confusion_matrix
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
print("Classification Report is:\n",classification_report(y_test,y_pred))
print("Confusion Matrix:\n",confusion_matrix(y_test,y_pred))
print("Training Score:\n",svc.score(x_train,y_train)*100)
```

```
Classification Report is:
              precision    recall  f1-score   support

           0       0.45      0.49      0.47       100
           1       0.44      0.40      0.42       100

    accuracy                           0.45       200
   macro avg       0.44      0.45      0.44       200
weighted avg       0.44      0.45      0.44       200

Confusion Matrix:
 [[49 51]
 [60 40]]
Training Score:
 55.50000000000001
```

## Naive Bayes:

In [48]:
```python
y_pred=gnb.predict(x_test)
from sklearn.metrics import accuracy_score,classification_report,confusion_matrix
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
print("Classification Report is:\n",classification_report(y_test,y_pred))
print("Confusion Matrix:\n",confusion_matrix(y_test,y_pred))
print("Training Score:\n",gnb.score(x_train,y_train)*100)
```

```
Classification Report is:
               precision    recall  f1-score   support

           0       0.51      0.35      0.41       100
           1       0.50      0.66      0.57       100

    accuracy                           0.51       200
   macro avg       0.51      0.51      0.49       200
weighted avg       0.51      0.51      0.49       200

Confusion Matrix:
 [[35 65]
 [34 66]]
Training Score:
 55.125
```

## Decision tree classifier:

In [54]:
```python
y_pred=dtree.predict(x_test)
from sklearn.metrics import accuracy_score,classification_report,confusion_matrix
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
print("Classification Report is:\n",classification_report(y_test,y_pred))
print("Confusion Matrix:\n",confusion_matrix(y_test,y_pred))
print("Training Score:\n",dtree.score(x_train,y_train)*100)
```

```
Classification Report is:
               precision    recall  f1-score   support

           0       0.52      0.79      0.63       100
           1       0.56      0.27      0.36       100

    accuracy                           0.53       200
   macro avg       0.54      0.53      0.50       200
weighted avg       0.54      0.53      0.50       200

Confusion Matrix:
 [[79 21]
 [73 27]]
Training Score:
 63.87500000000001
```

# Random Forest Classifier:

In [56]:

```
y_pred=rfc.predict(x_test)
from sklearn.metrics import accuracy_score,classification_report,confusion_matrix
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
print("Classification Report is:\n",classification_report(y_test,y_pred))
print("Confusion Matrix:\n",confusion_matrix(y_test,y_pred))
print("Training Score:\n",rfc.score(x_train,y_train)*100)
```

```
Classification Report is:
               precision    recall  f1-score   support

           0       0.48      0.49      0.48       100
           1       0.47      0.46      0.47       100

    accuracy                           0.48       200
   macro avg       0.47      0.47      0.47       200
weighted avg       0.47      0.47      0.47       200

Confusion Matrix:
 [[49 51]
 [54 46]]
Training Score:
 100.0
```

# AdaBoostClassifier:

n [58]:

```
y_pred=adb.predict(x_test)
from sklearn.metrics import accuracy_score,classification_report,confusion_matrix
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
print("Classification Report is:\n",classification_report(y_test,y_pred))
print("Confusion Matrix:\n",confusion_matrix(y_test,y_pred))
print("Training Score:\n",adb.score(x_train,y_train)*100)
```

```
Classification Report is:
               precision    recall  f1-score   support

           0       0.54      0.54      0.54       100
           1       0.54      0.54      0.54       100

    accuracy                           0.54       200
   macro avg       0.54      0.54      0.54       200
weighted avg       0.54      0.54      0.54       200

Confusion Matrix:
 [[54 46]
 [46 54]]
Training Score:
```

## Gradient Boosting Classifier:

In [61]:
```
y_pred=gbc.predict(x_test)
from sklearn.metrics import accuracy_score,classification_report,confusion_matrix
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
print("Classification Report is:\n",classification_report(y_test,y_pred))
print("Confusion Matrix:\n",confusion_matrix(y_test,y_pred))
print("Training Score:\n",gbc.score(x_train,y_train)*100)
```

```
Classification Report is:
              precision    recall  f1-score   support

           0       0.49      0.50      0.49       100
           1       0.48      0.47      0.48       100

    accuracy                           0.48       200
   macro avg       0.48      0.48      0.48       200
weighted avg       0.48      0.48      0.48       200

Confusion Matrix:
 [[50 50]
 [53 47]]
Training Score:
 88.0
```

## XGB Classifier:

n [64]:
```
y_pred=xgb.predict(x_test)
from sklearn.metrics import accuracy_score,classification_report,confusion_matrix
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
print("Classification Report is:\n",classification_report(y_test,y_pred))
print("Confusion Matrix:\n",confusion_matrix(y_test,y_pred))
print("Training Score:\n",xgb.score(x_train,y_train)*100)
```

```
Classification Report is:
              precision    recall  f1-score   support

           0       0.49      0.56      0.52       100
           1       0.48      0.41      0.44       100

    accuracy                           0.48       200
   macro avg       0.48      0.48      0.48       200
weighted avg       0.48      0.48      0.48       200

Confusion Matrix:
 [[56 44]
 [59 41]]
Training Score:
 62.625
```

# ExtraTrees Classifier:

In [66]:
```
y_pred=etc.predict(x_test)
from sklearn.metrics import accuracy_score,classification_report,confusion_matrix
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
print("Classification Report is:\n",classification_report(y_test,y_pred))
print("Confusion Matrix:\n",confusion_matrix(y_test,y_pred))
print("Training Score:\n",etc.score(x_train,y_train)*100)
```

```
Classification Report is:
              precision    recall  f1-score   support

           0       0.50      0.50      0.50       100
           1       0.50      0.50      0.50       100

    accuracy                           0.50       200
   macro avg       0.50      0.50      0.50       200
weighted avg       0.50      0.50      0.50       200

Confusion Matrix:
 [[50 50]
 [50 50]]
Training Score:
 100.0
```

# Bagging Classifier:

In [67]:
```
from sklearn.ensemble import BaggingClassifier
from sklearn import tree
model = BaggingClassifier(tree.DecisionTreeClassifier(random_state=1))
model.fit(x_train, y_train)
model.score(x_test,y_test)
```

Out[67]: 0.57

In [68]:
```
data = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
data
```

Out[68]:

| | Actual | Predicted |
|---|---|---|
| 993 | 1 | 1 |
| 859 | 0 | 1 |
| 298 | 1 | 1 |
| 553 | 1 | 0 |
| 672 | 0 | 1 |
| ... | ... | ... |
| 679 | 1 | 1 |
| 722 | 1 | 1 |
| 215 | 1 | 0 |
| 653 | 1 | 0 |
| 150 | 0 | 0 |

## Overall Data:

| Classifier | Accuracy | Precision | Recall | F1 Score |
|---|---|---|---|---|
| KNN | 64.75 | 0.47 | 0.49 | 0.48 |
| SVM | 55.50 | 0.449 | 0.49 | 0.46 |
| NAIVE BAYES | 55.125 | 0.50 | 0.35 | 0.41 |
| DECISION TREE | 63.875 | 0.51 | 0.79 | 0.62 |
| RANDOM FOREST | 100.0 | 0.53 | 0.50 | 0.51 |
| ADA BOOST | 67.0 | 0.54 | 0.54 | 0.53 |
| GRADIENT BOOST | 88.0 | 0.50 | 0.48 | 0.48 |
| XGB | 62.625 | 0.5 | 0.4 | 0.44 |
| EXTRA TREE | 100.0 | 0.5 | 0.5 | 0.5 |
| BAGGING | 40.0 | 0.5 | 0.4 | 0.44 |

## Some other data sets and  outputs :

Summary_of_Neighborhood_Sales_for_Brooklyn

Summary_of_Neighborhood_Sales_in_Manhattan

data-agriculture-and-biosciences-sales

data-oklahoma-lottary-commission-retailer-ranking-from-oct

MVA_vehicle_Sales_Counts_by_Month_for_Calender_year_2001_through

## Output:

## MVA_vehicle_Sales_Counts_by_Month_for_Calender_year_2001_through

| **Classifiers** | **Accuracy** |
|---|---|
| KNeighborsClassifier | 12.1212 |
| Support Vector Systems | 1.0101 |
| Naive Bayes | 100.0 |
| Decision tree classifier | 32.3232 |
| Random Forest Classifier | 100.0 |
| AdaBoostClassifier | 17.6767 |
| Gradient Boosting Classifier | 100.0 |
| XGBClassifier | 78.4522 |
| ExtraTreesClassifier | 100.0 |
| Bagging Classifier | 55.2525 |

# data-oklahoma-lottary-commission-retailer-ranking-from-oct

| Classifiers | Accuracy |
|---|---|
| KNeighborsClassifier | 34.9750 |
| Support Vector Systems | 16.4360 |
| Naive Bayes | 70.614277 |
| Decision tree classifier | 55.6170 |
| Random Forest Classifier | 100.0 |
| AdaBoostClassifier | 28.1128 |
| Gradient Boosting Classifier | 99.446 |
| XGBClassifier | 66.1121 |
| ExtraTreesClassifier | 100.0 |
| Bagging Classifier | 66.1504 |

# Summary_of_Neighborhood_Sales_for_Brooklyn

| Classifiers | Accuracy |
|---|---|
| KNeighborsClassifier | 28.2447 |
| Support Vector Systems | 12.2137 |
| Naive Bayes | 57.2519 |
| Decision tree classifier | 71.75572 |
| Random Forest Classifier | 100.0 |
| AdaBoostClassifier | 16.0305 |
| Gradient Boosting Classifier | 100. |
| XGBClassifier | 65.2325 |
| ExtraTreesClassifier | 100.0 |
| Bagging Classifier | 12.1212 |

# Summary_of_Neighborhood_Sales_in_Manhattan_class

| Classifiers | Accuracy |
| --- | --- |
| KNeighborsClassifier | 51.11 |
| Support Vector Systems | 46.66 |
| Naive Bayes | 48.88 |
| Decision tree classifier | 86.66 |
| Random Forest Classifier | 100.0 |
| AdaBoostClassifier | 46.66 |
| Gradient Boosting Classifier | 100.0 |
| XGBClassifier | 55.125 |
| ExtraTreesClassifier | 100.0 |
| Bagging Classifier | 66.666 |

# Summary_of_Neighborhood_Sales_in_Manhattan_home

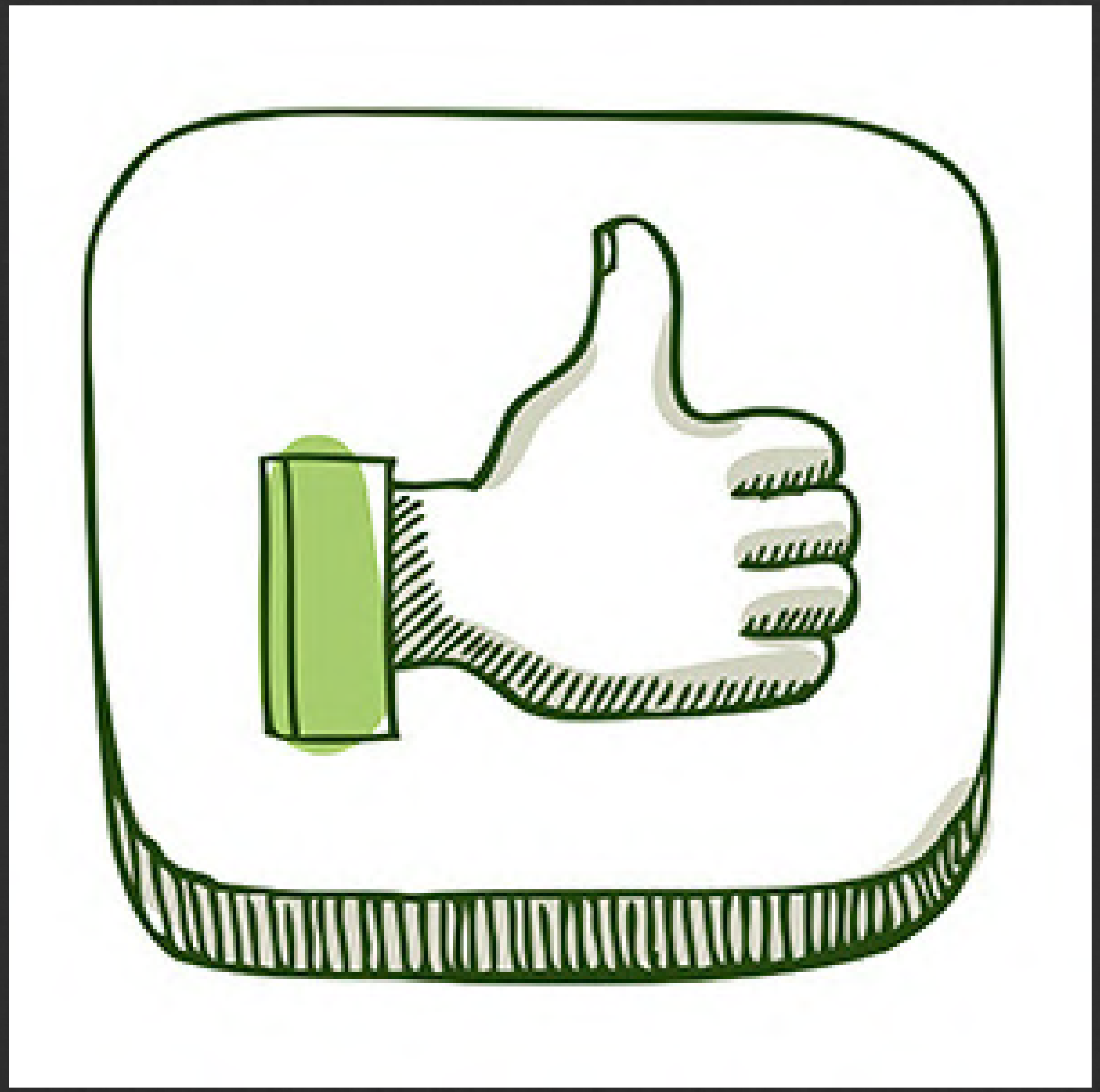
| Classifiers | Accuracy |
|---|---|
| KNeighborsClassifier | 51.11 |
| Support Vector Systems | 46.66 |
| Naive Bayes | 48.88 |
| Decision tree classifier | 86.66 |
| Random Forest Classifier | 100.0 |
| AdaBoostClassifier | 46.66 |
| Gradient Boosting Classifier | 100.0 |
| XGBClassifier | 55.125 |
| ExtraTreesClassifier | 100.0 |
| Bagging Classifier | 66.666 |

# FUTURE SCOPE:

- In this study we used some techniques for sales prediction such are models and XG Boost algorithms which get better efficiency manipulate the trending sales analysis.
- Random Forest gives the best accuracy value. Data mining techniques like Linear Regression, Random Forest Regression and XGBoost had been carried out and the outcomes compared.
- XGBoost which is an expanded gradient boosting algorithm was once found to function the excellent at prediction. Sales prediction performs a necessary function in growing the effectively with which shops can function as it presents important points on the visitors a save can count on to get hold of on a given day.

# ADVANTAGES:

- Sales Planning
- Allocation of Resources
- Key Factor in Business Operations
- Basis of Salesforce Planning
- Major Role in Success



# DISADVANTAGES:

- Lake of Sales History
- Change in Business Environment
- Change in Consumer Behaviour
- Lake of Facts and Data
- Based on Assumptions

# CHALLENGES:


- Seller subjectivity
- A lack of predictive data
- Technology limitations

Thank you!