MACHINE LEARNING DA

BY TECHOLAS

Intro..

Machine learning are set of techniques to make machines better at doing things that humans(traditionally) can do better than machines.

Eg: calculating huge math operation machine is better than human,but when driving a car human is better than machines.

So machine learning is something we make machines to learn things like humans do

continues..

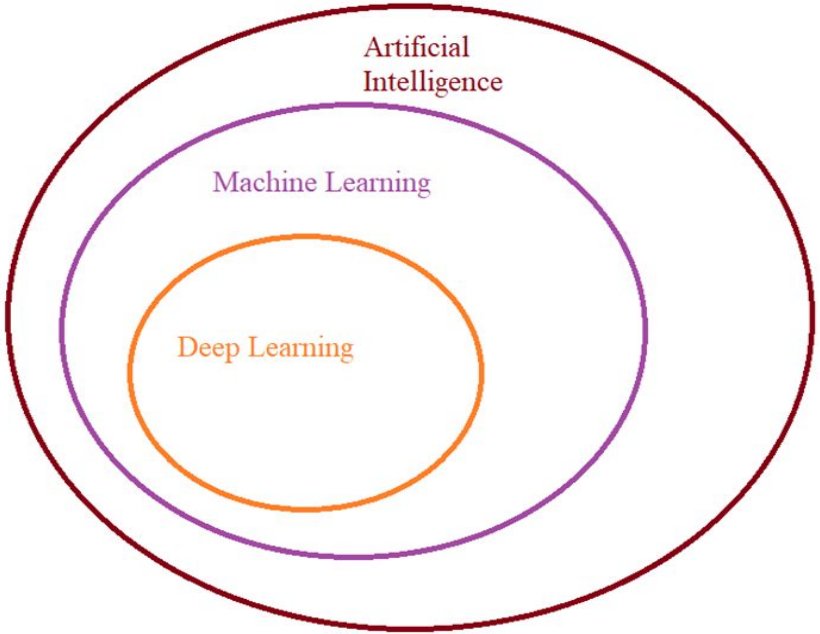
Why humans are better in driving cars??

Because we are continuously learning or training yourself to to be better.

That's what we need to do in machine learning. It is the ability of the machine to learn from the past data.

Machine learning is a subset of Artificial Intelligence (AI).

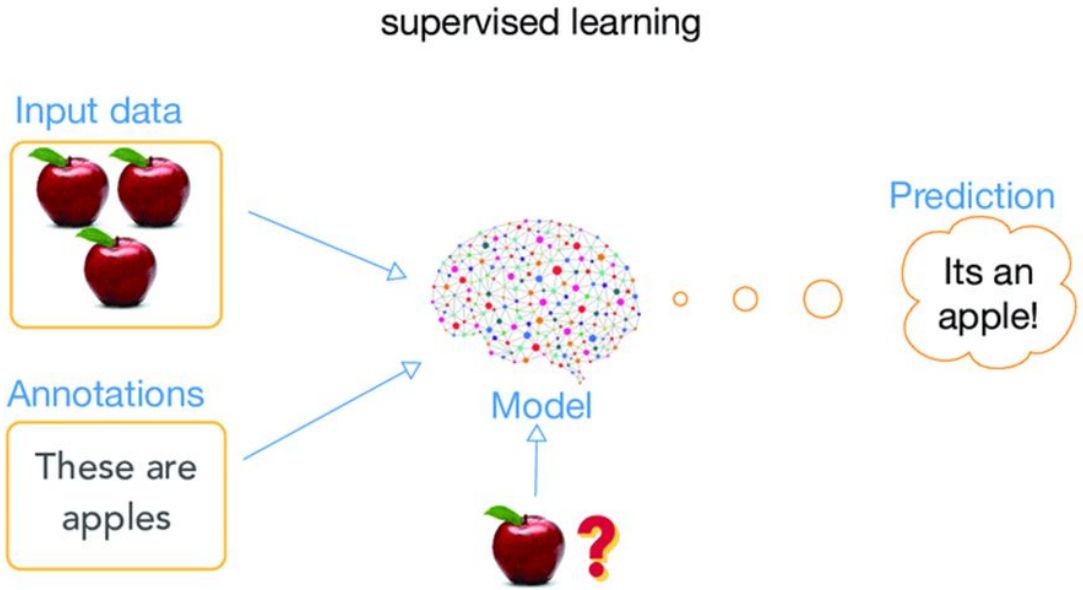
ML models are built on top of statistics, calculus and linear algebra.



TECHOLAS
TECHNOLOGY DEMYSTIFIED

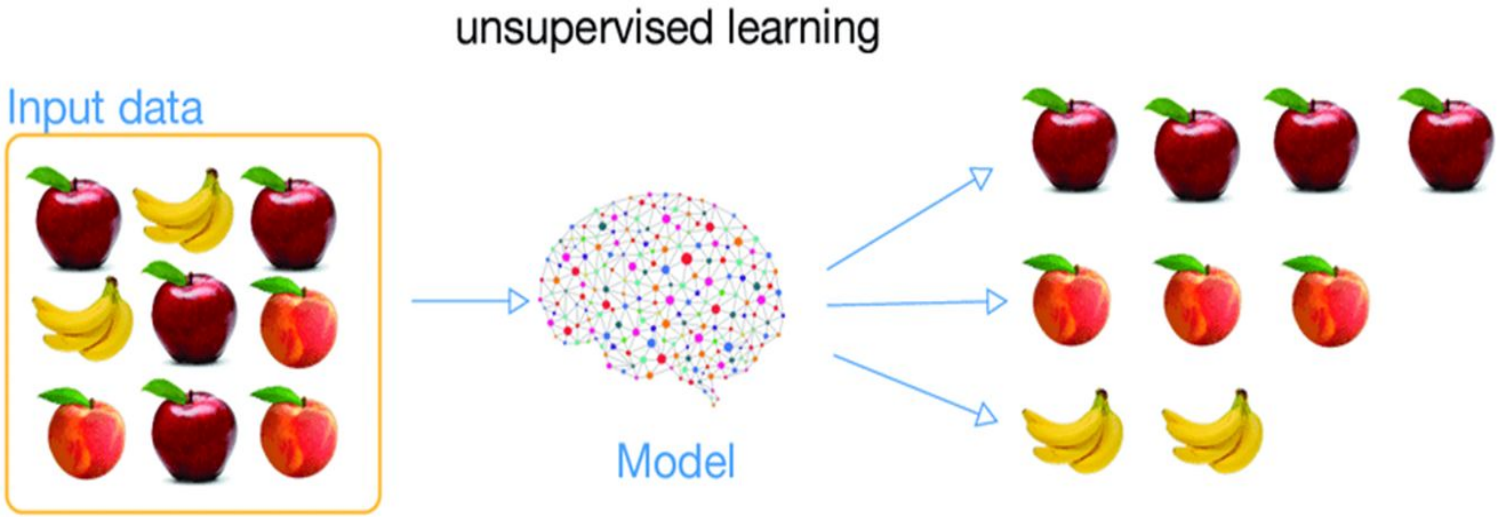
Types of Machine Learning

Supervised Learning: Trains algorithm based on example input and output data labeled by humans.



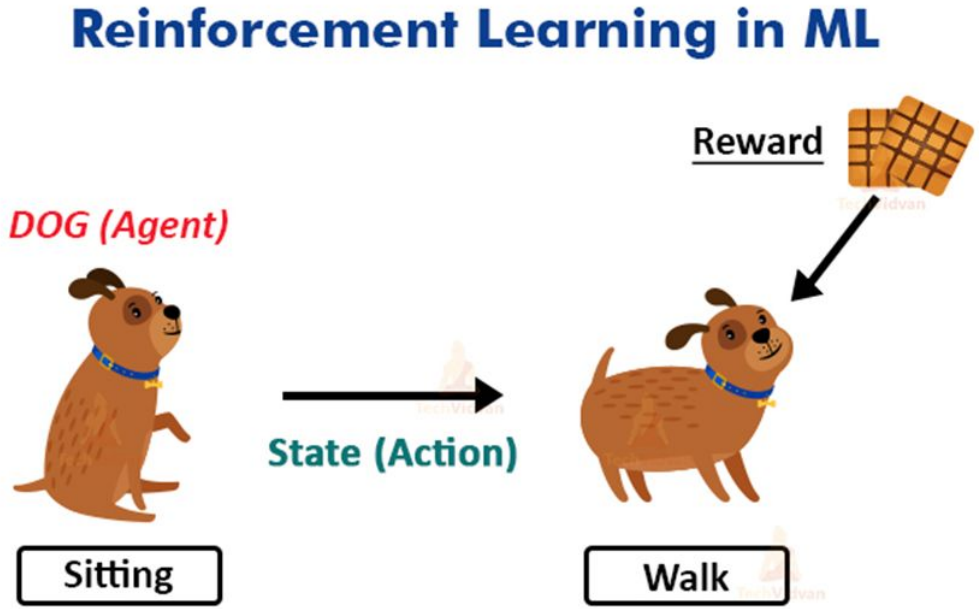
TECHOLAS
TECHNOLOGY DEMYSTIFIED

Unsupervised Learning: Provides the algorithm unlabeled data in order to allow it to find structure within the input data.

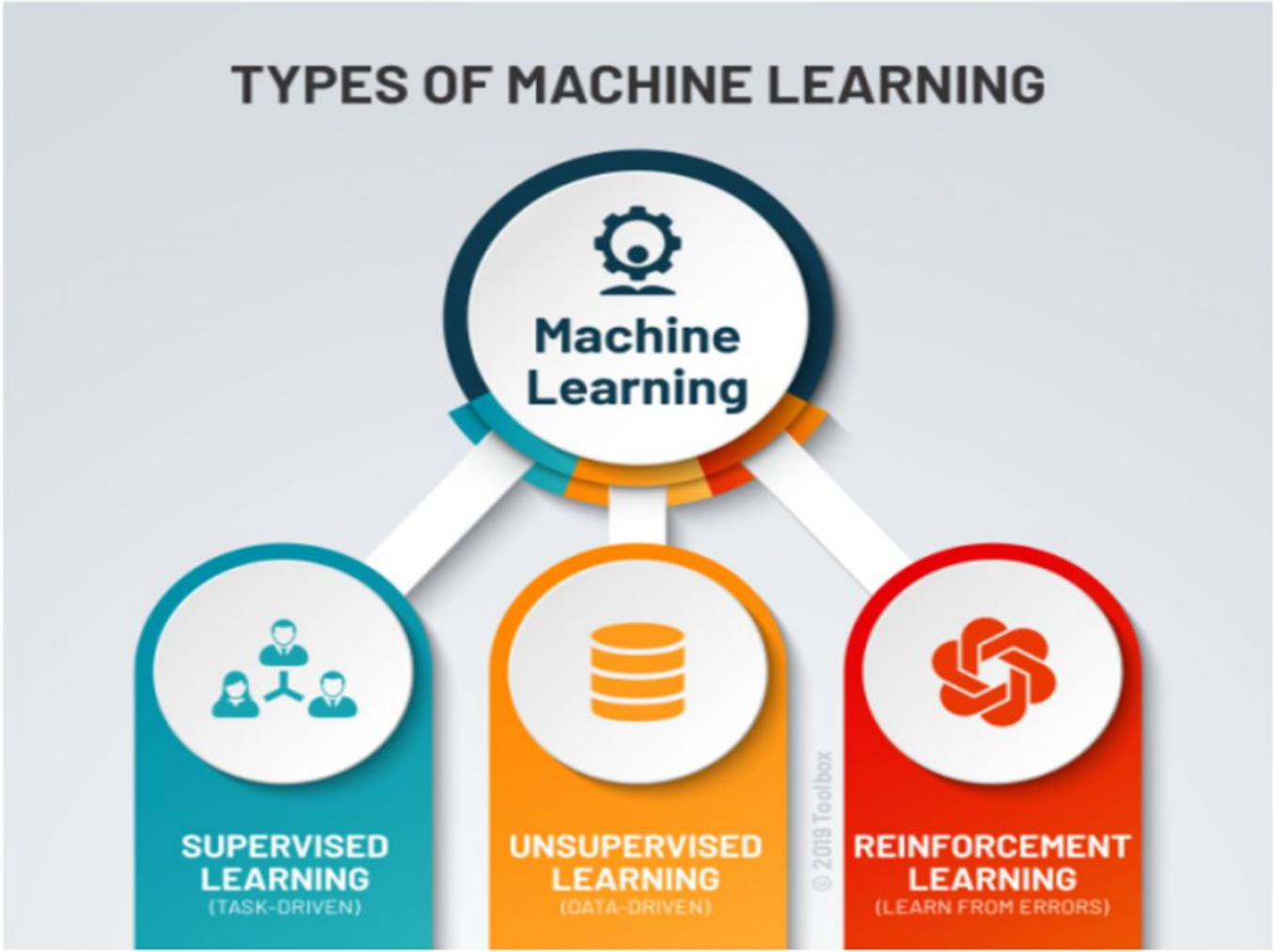


TECHOLAS
TECHNOLOGY DEMYSTIFIED

Reinforcement Learning: Concerned with how intelligent agents ought to take actions in an environment to maximize the notion of cumulative reward.



TECHOLAS
TECHNOLOGY DEMYSTIFIED



TECHOLAS
TECHNOLOGY DEMYSTIFIED

Types of supervised learning problems

Regression: When the model is supposed to predict numerical values, it is called a regression problem. For example, house price prediction.

Classification: When the data is supposed to be classified into different classes, it is a classification problem. If there are only 2 classes, it is a binary classification problem. Example, to predict whether a person has a disease or not. If there are more than 2 classes, it is a multi-class classification problem. Example, predict whether a student gets grade A, B, C or D in an examination.

Installing SKLEARN

conda install scikit-learn #anaconda users

Pip3 install scikit-learn

SUPERVISED LEARNING

SIMPLE LINEAR REGRESSION..

IT is the very basic machine learning algorithm to create model and train

It's based on the equation **y=mx+c**

y=dependent variable

x=independent variable

m=slope

c=y intercept

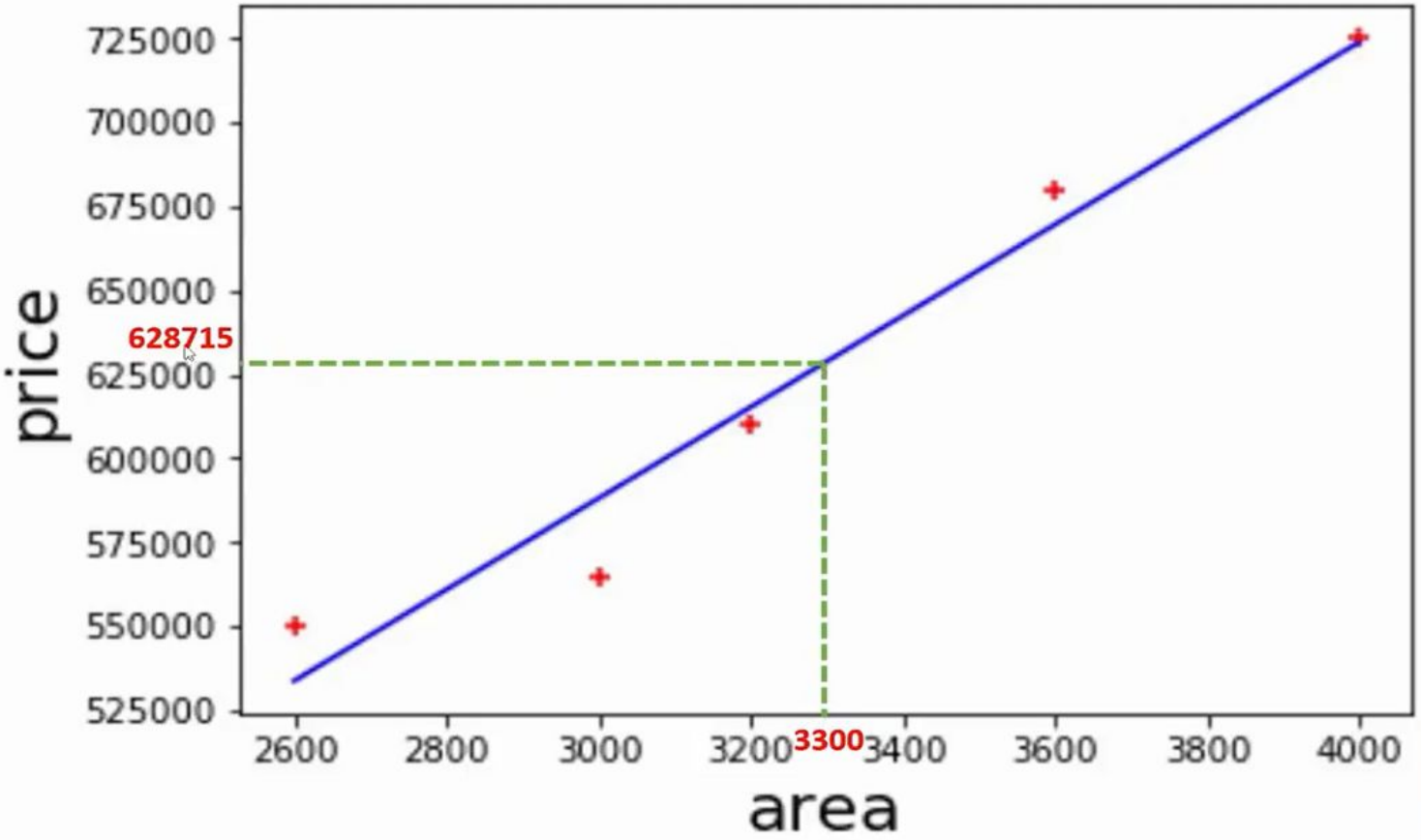
TECHOLAS
TECHNOLOGY DEMYSTIFIED

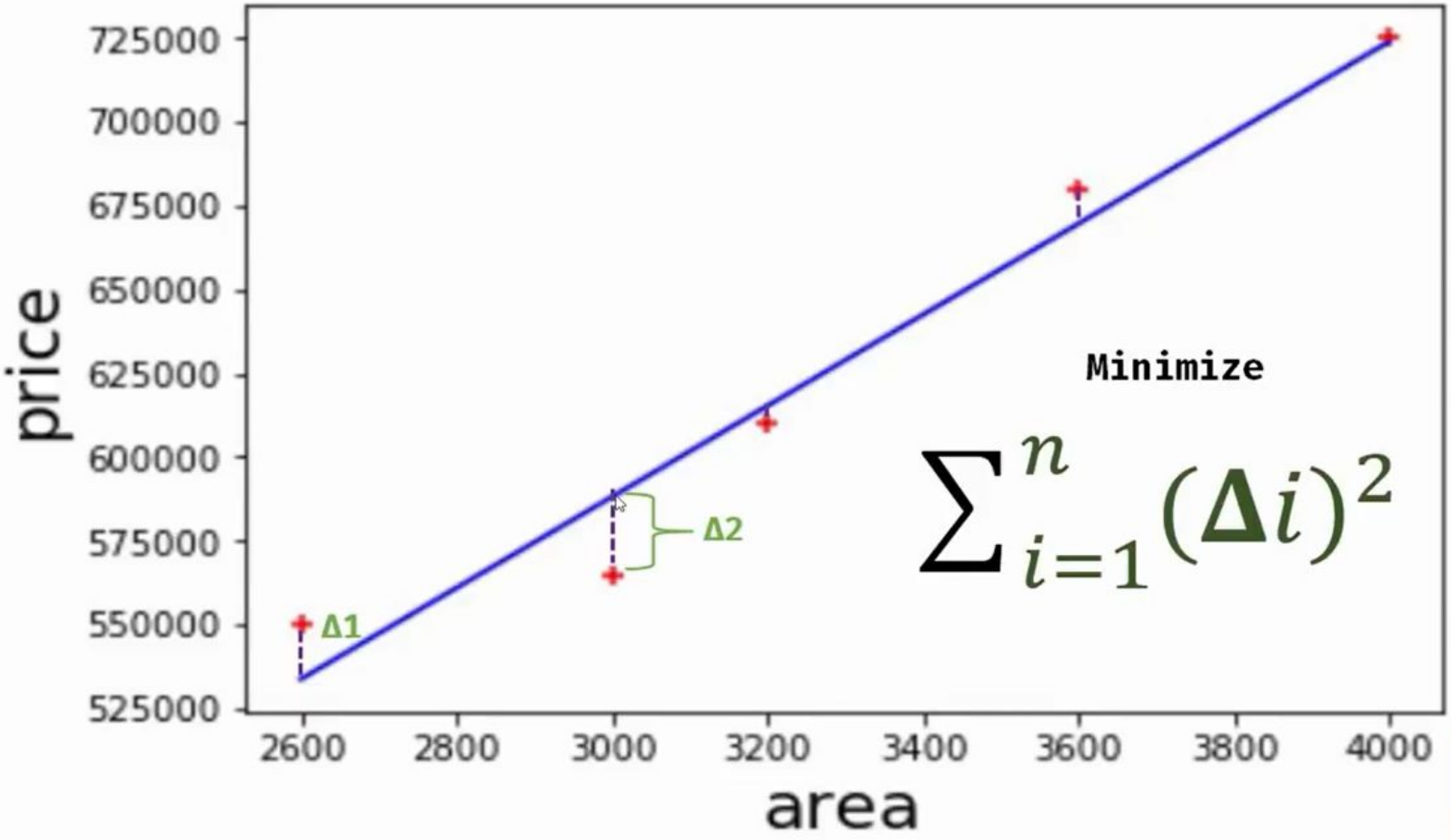

price = m * area + b
Dependent variable
Independent variable

Simple Linear Regression..

Home prices in Monroe Twp, NJ (USA)

area	price
2600	550000
3000	565000
3200	610000
3600	680000
4000	725000


725000
700000
675000
650000
628715
625000
600000
575000
550000
525000
price
2600
2800
3000
3200
3300
3400
3600
3800
4000
area


725000
700000
675000
650000
625000
600000
575000
550000
525000
price
2600
2800
3000
3200
3400
3600
3800
4000
area
Δ1
Δ2
Minimize
$\sum_{i=1}^{n} (\Delta i)^2$

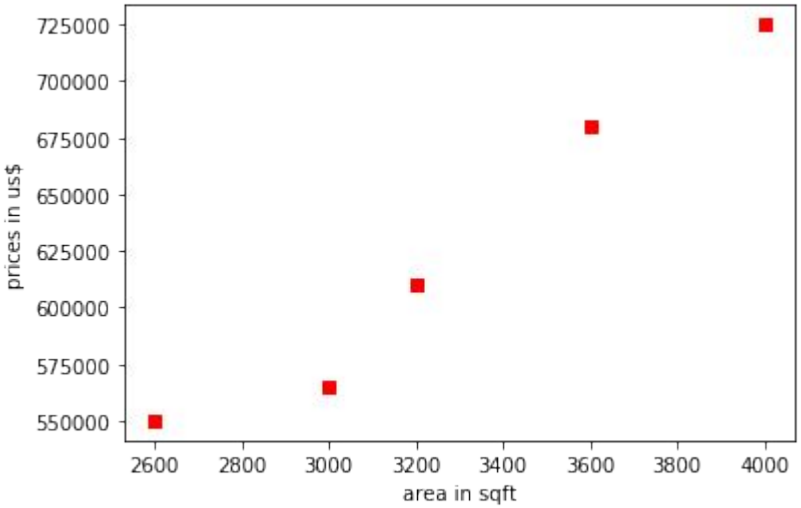
In notebook..

Import modules and load data

1. import pandas as pd
2. import numpy as np
3. from matplotlib import pyplot as plt
4. from sklearn import linear_model
5. data=pd.read_csv('homeprices.csv')
6. data

continues..

Plot the data...



1. plt.xlabel('area in sqft')
2. plt.ylabel('prices in us$')
3. plt.scatter(data.area,data.price,marker='s',color='red')

TECHOLAS
TECHNOLOGY DEMYSTIFIED

Linear regression continues..

Create the model and train the data..

fit→ for train the data

```
reg_model=linear_model.LinearRegression()
reg_model.fit(data[['area']],data.price)
```

```
print(reg_model.coef_)
print(reg_model.intercept_)
```

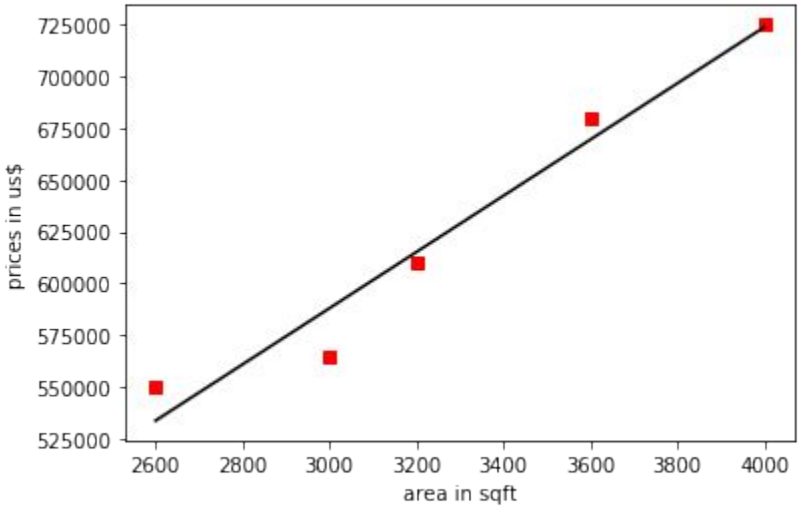
predict→ predict future data

```
reg_model.predict([[2800],[3000]])
```

TECHOLAS
TECHNOLOGY DEMYSTIFIED

plotting..

```
pre=reg_model.predict(data[['area']])

plt.xlabel('area in sqft')

plt.ylabel('prices in us$')

plt.scatter(data.area,data.price,marker='s',color='red')

plt.plot(data.area,pre,color='black')
```



TECHOLAS
TECHNOLOGY DEMYSTIFIED

exercise

Predict canada's per capita income in year 2020. build a regression model and predict the per capita income for canadian citizens in year 2020

https://drive.google.com/open?id=1nzT2GxwE7MK25XT_iv1Fnn_WrLaSRTqB

https://drive.google.com/o

TECHOLAS
TECHNOLOGY DEMYSTIFIED

Linear regression with multiple variable..

Want to create model with more than one independent model.ie the dependent variable value can predict based on more than one independent variable

area	bedrooms	age	price
2600	3	20	550000
3000	4	15	565000
3200		18	610000
3600	3	30	595000
4000	5	8	760000
4100	6	8	810000

Equation..

$$y = m_1 x_1 + m_2 x_2 + m_3 x_3 + b$$

$$price = m_1 * area + m_2 * bedrooms + m_3 * age + b$$

TECHOLAS
TECHNOLOGY DEMYSTIFIED

In Notebook..

Import and load the data

1. import pandas as pd
2. import numpy as np
3. from matplotlib import pyplot as plt
4. from sklearn import linear_model
5. data=pd.read_csv('homeprices.csv')
6. data

Data preprocessing..

```
median_=data.bedrooms.median()

#data.bedrooms=data.bedrooms.fillna(median_)

data['bedrooms'].loc[2]=median_

data
```

TECHOLAS
TECHNOLOGY DEMYSTIFIED

Create,train and predict

```
model=linear_model.LinearRegression()

model.fit(data[['area','bedrooms','age']],data.price)

model.predict([[3000,4,15]])
```

exercise..

. This file contains hiring statics for a firm such as experience of candidate, his written test score and personal interview score. Based on these 3 factors, HR will decide the salary. Given this data, you need to build a machine learning model for HR department that can help them decide salaries for future candidates. Using this predict salaries for following candidates,

2 yr experience, 9 test score, 6 interview score

12 yr experience, 10 test score, 10 interview score

https://drive.google.com/open?id=1S-bu1Xni34HZNwcPLJkCUWSBgBnTlSxf

TECHOLAS
TECHNOLOGY DEMYSTIFIED

ml

Machine learning consists of 2 steps typically

1. train the model
2. Ask questions to the model

The more you train the model ,The model will be more accurate

But if the training data is so huge then the training time will be high.

So every time training a model is time consuming process.

Saving The Trained Model..

Save a model using pickle model.By using dump function and load

```
import pickle
with open('pickle_model','wb') as f:
    pickle.dump(reg_model,f)
with open('pickle_model','rb') as d:
    model=pickle.load(d)
model.predict([[2000]])
```

TECHOLAS
TECHNOLOGY DEMYSTIFIED

Saving continues..

Saving using joblib module

```
from sklearn.externals import  joblib

joblib.dump(reg_model,'joblib_model')

new_model=joblib.load('joblib_model')

new_model.predict([[30000]])
```

TECHOLAS
TECHNOLOGY DEMYSTIFIED

Finding how accurate your model??

model.score(X,Y)

Evaluation metrics for regression problems

A regression model can never make the predicted values equal to the true target values. It can only be checked for how well the results are reproduced by the model minimizing the error between the true and predicted values.

The score() method returns the **R2 score**, also called the coefficient of determination, which is the evaluation metric for a regression model.

R2= 1- SSres / SStot

SSres is the sum of squares of the residual errors.
SStot is the total sum of the errors.

Dummy variables and 1 hot encoding..

Consider situation like in our data that we are using to train the model,is not numbers and the reality is it always will not be.

How should we handle text data in numeric model?

town	area	price
monroe township	2600	550000
monroe township	3000	565000
monroe township	3200	610000
monroe township	3600	680000
monroe township	4000	725000
west windsor	2600	585000
west windsor	2800	615000
west windsor	3300	650000
west windsor	3600	710000
robbinsville	2600	575000
robbinsville	2900	600000
robbinsville	3100	620000
robbinsville	3600	695000

continues,,

So while dealing with categorical data label for text data is not practical .

So the technique is **1 hot encoding**

So we are going to create extra variable in our data and these variables are known as dummy variables.

Dummy variables

town	area	price	monroe township	west windsor	robbinsville
monroe township	2600	550000	1	0	0
monroe township	3000	565000	1	0	0
monroe township	3200	610000	1	0	0
monroe township	3600	680000	1	0	0
monroe township	4000	725000	1	0	0
west windsor	2600	585000	0	1	0
west windsor	2800	615000	0	1	0
west windsor	3300	650000	0	1	0
west windsor	3600	710000	0	1	0
robbinsville	2600	575000	0	0	1

In notebook

Create the Data

```
data=pd.read_csv('homeprices.csv')
dummies=pd.get_dummies(data.town)
merge=pd.concat([data,dummies],axis=1)
merge=merge.drop(['town'],axis=1)
merge
```

Train the model

```
X=merge.drop(['price'],axis=1)
Y=merge.price
model=linear_model.LinearRegression()
model.fit(X,Y)
model.predict([[2000,0,1,0]])
```

TECHOLAS
TECHNOLOGY DEMYSTIFIED

Using 1 hot encoder..

Creating X and Y

```
df=data
Df
from sklearn.preprocessing import OneHotEncoder


Y=df.price
```

```
from sklearn.compose import ColumnTransformer
ts = ColumnTransformer(
    transformers=[
        ("abc",  OneHotEncoder(), [1]   )
    ],
    remainder='passthrough' )
X = ts.fit_transform(X)
```

continues..

Train And Predict

```
model=linear_model.LinearRegression()
model.fit(X,Y)
model.predict([[1,0,0,2000]])
```

Exercise

https://drive.google.com/open?id=1Qov_4VWCzohLDoogJpQfAbA0Hw9_w8wT

1) Predict price of a mercedes benz that is 4 yr old with mileage 45000

2) Predict price of a BMW X5 that is 7 yr old with mileage 86000

3) Tell me the score (accuracy) of your model. (Hint: use LinearRegression().score())

Split the data to train and test..

Training the model with the entire dataset is not a good practice. When the same dataset is used for testing the model, it may not reflect the actual performance of the model.

So in machine learning, the dataset is split into 2 parts .

The first part is for training the model

The second part is for testing the model.

This ensures that the model has not seen the testing data during the training phase.

In notebook..

```
import pandas as pd

import numpy as np

from sklearn import linear_model

from sklearn.model_selection import train_test_split
```

continues..

```
data=pd.read_csv('carprices.csv')
x=data[['Mileage','Age(yrs)']]
y=data[['Sell Price($)']]
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=.3)
```

Model and predict..

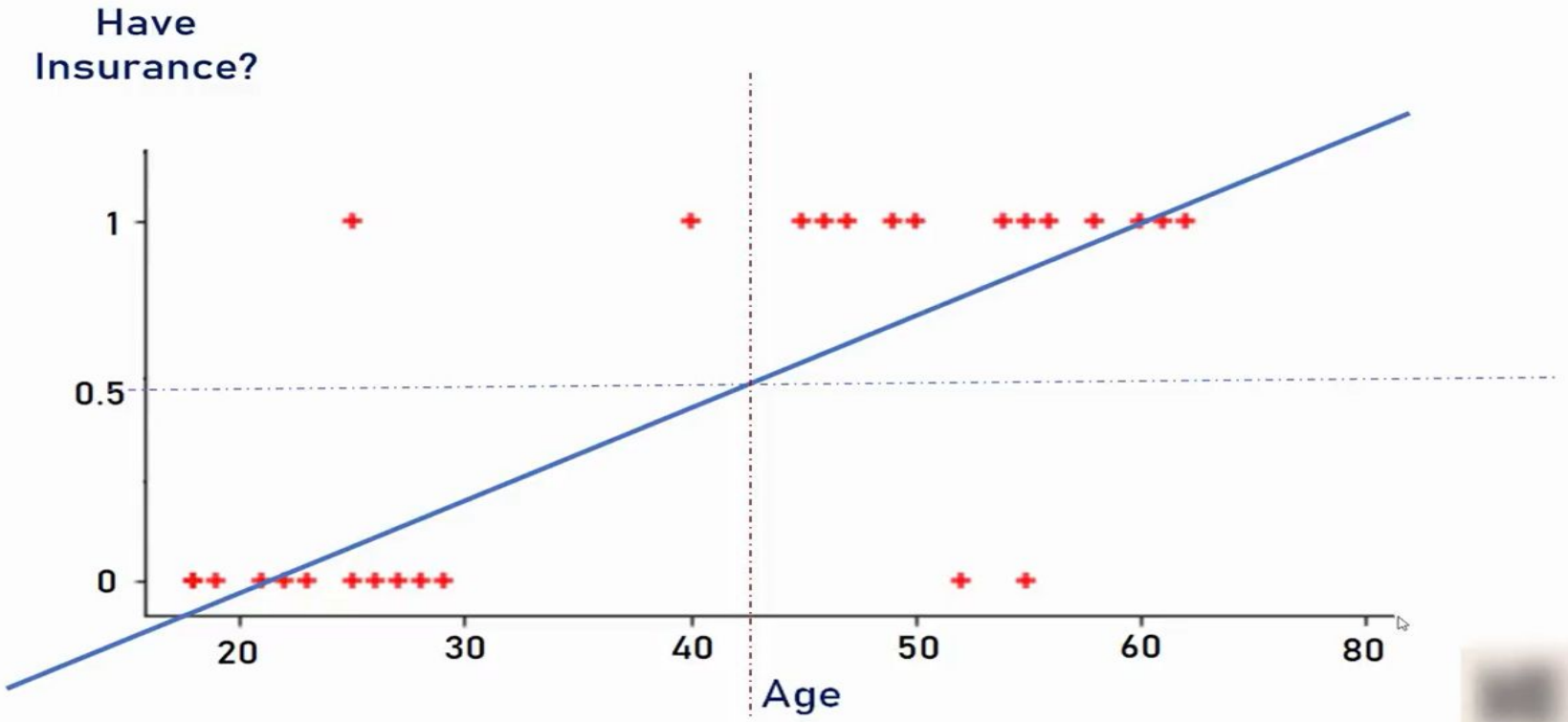
```
model=linear_model.LinearRegression()

model.fit(xtrain,ytrain)

model.predict(xtest)

ytest
```

Logistic regression

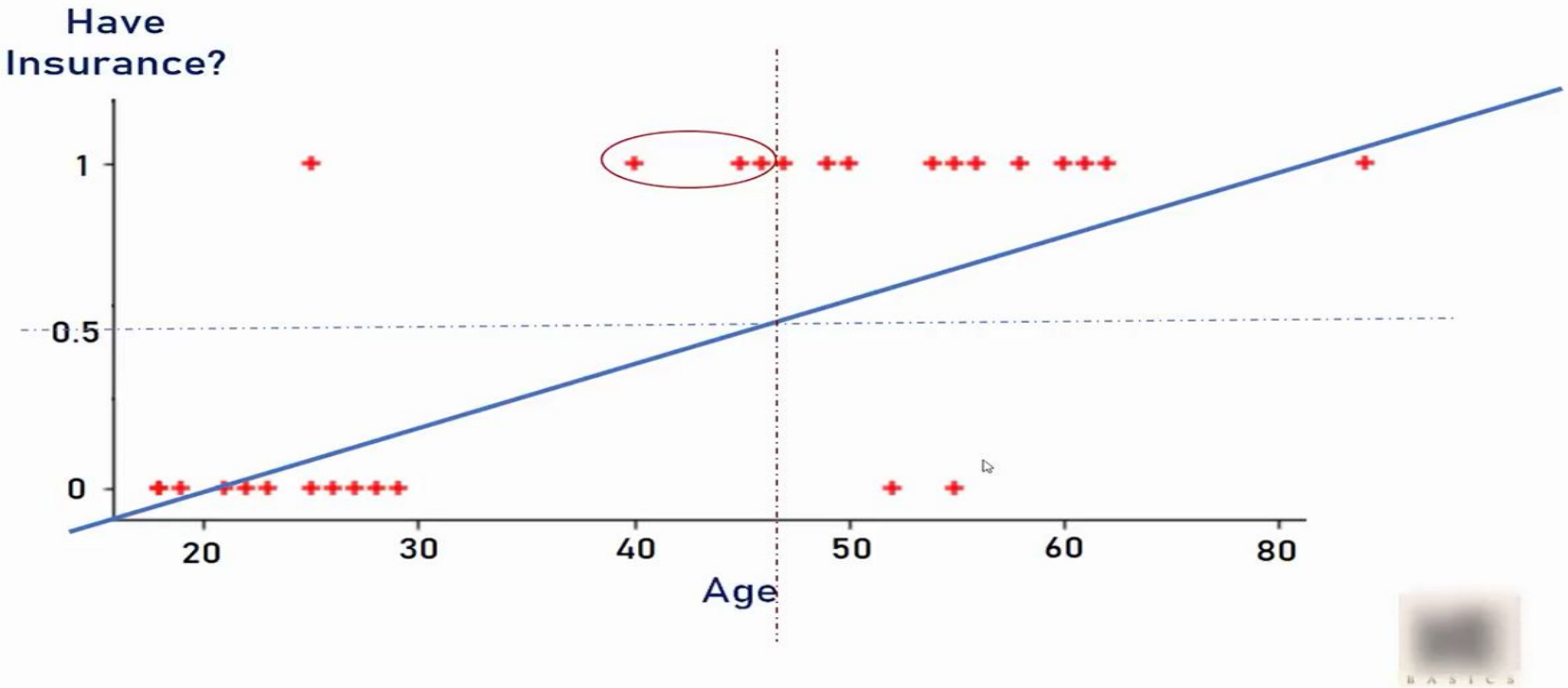
Logistic regression is used to predict classification problems.there are 2 categories

1. Binary logistic regression(binary classification)
2. Multivariable logistic regression(multi class classification)

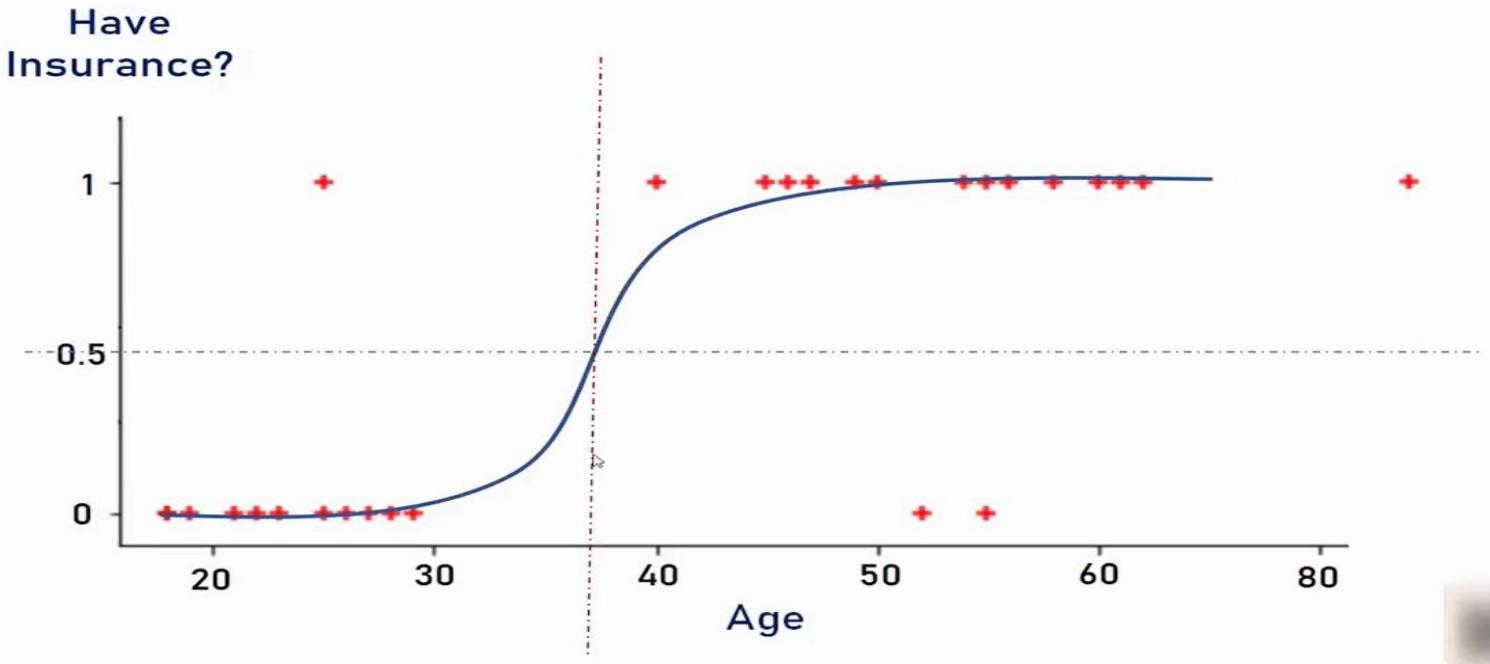
Classification problem using linear regression



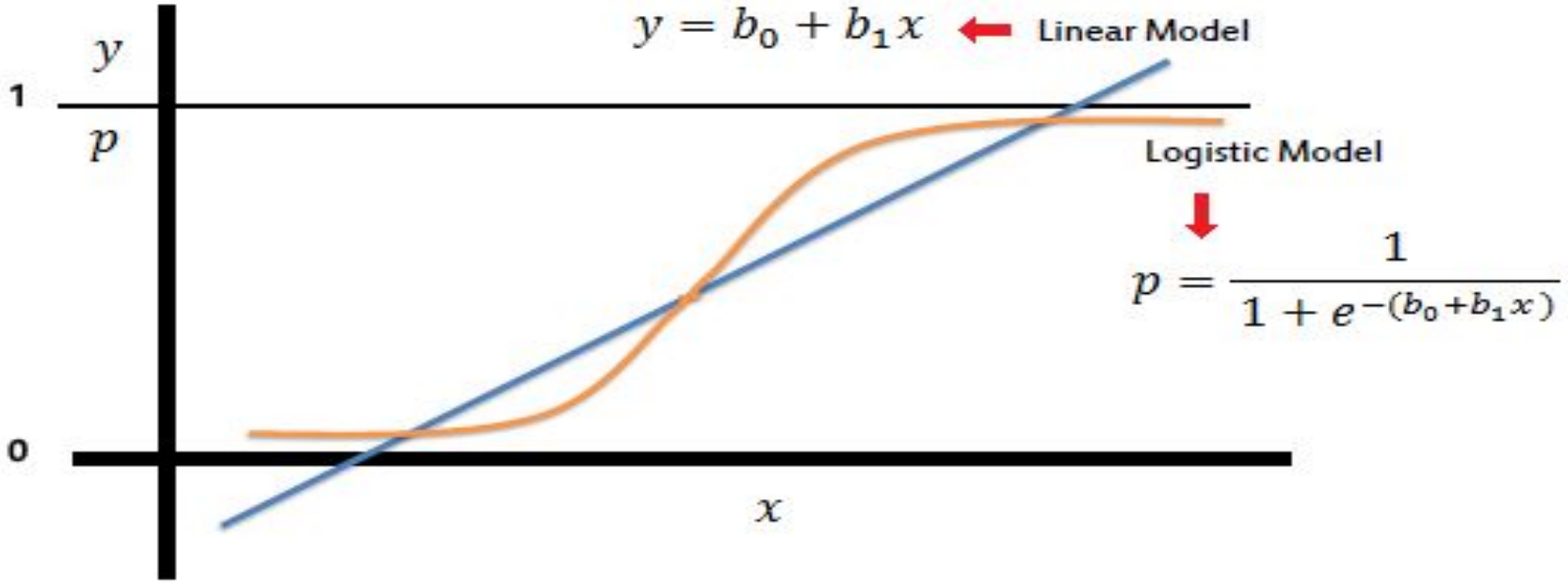
Problem with linear regression



Solution.. Logit function (Sigmoid function)



Logistic regression..



In notebook..

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LogisticRegression
```

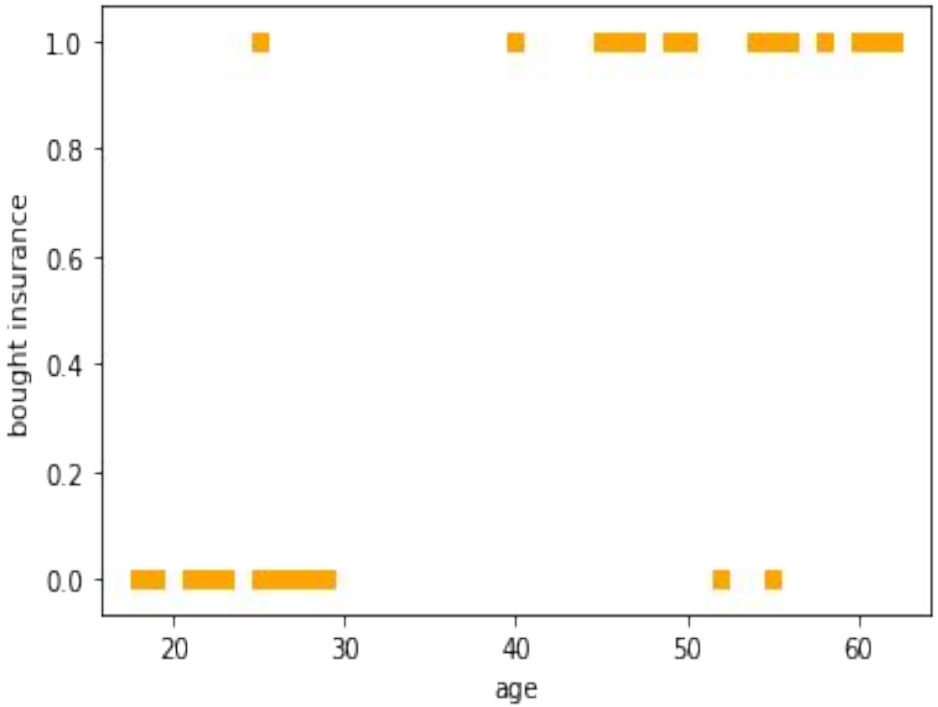
Data fetch and train_test_split..

```
data=pd.read_csv('insurance_data.csv')

x=data[['age']]

y=data['bought_insurance']

X_train, X_test, y_train, y_test=train_test_split(x,y,test_size=.2)
```

TECHOLAS
TECHNOLOGY DEMYSTIFIED

Plot it..

```
plt.scatter(x,y,color='orange',marker='s')

plt.xlabel('age')

plt.ylabel('bought insurance')
```



TECHOLAS
TECHNOLOGY DEMYSTIFIED

Model it and predict..

```
model=LogisticRegression()

model.fit(X_train,y_train)

model.predict(X_test)

y_test

model.score(X_test,y_test)
```

assignment

Download employee retention dataset from here:
https://www.kaggle.com/giripujar/hr-analytics.

1. Now do some exploratory data analysis to figure out which variables have direct and clear impact on employee retention (i.e. whether they leave the company or continue to work)
2. Plot bar charts showing impact of employee salaries on retention
3. Plot bar charts showing correlation between department and employee retention
4. Now build logistic regression model using variables that were narrowed down in step 1
5. Measure the accuracy of the model

Logistic regression with multi class classifications

This kind of problem will have multiple classifications.

Like which team will win the t20 world cup?--> its having multiple options

We are going to identify handwritten digit recognition

In notebook..

```
import matplotlib.pyplot as plt

from sklearn.datasets import load_digits

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LogisticRegression

from sklearn.metrics import confusion_matrix
```

Load dataset by sklearn..

1. digits=load_digits()`
2. len(digits)
3. dir(digits)
4. digits.data[100]
5. digits.target[0:5]
6. digits.images[0]
7. plt.matshow(digits.images[0])

Model creation..

```
xtrain,xtest,ytrain,ytest=train_test_split(digits['data'],digits['target'],test_size=.2)

model=LogisticRegression()

model.fit(xtrain,ytrain)

model.score(xtest,ytest)
```

TECHOLAS
TECHNOLOGY DEMYSTIFIED

Random prediction..

plt.matshow(digits.images[888])

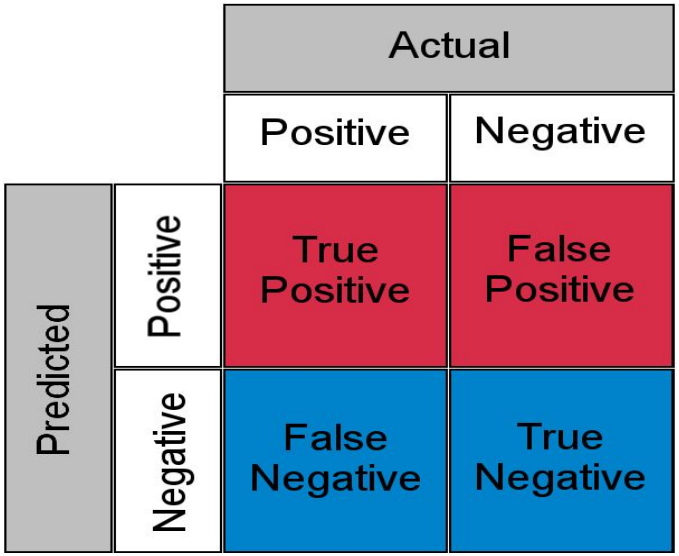
digits.target[888]

model.predict([digits.data[888]])

Why score not 1?? Where failed??

CONFUSION MATRIX?? WHAT IS THAT??

It is a performance measurement for machine learning classification problem where output can be two or more classes.



TECHOLAS
TECHNOLOGY DEMYSTIFIED

Confusion matrix continues..

Consider a binary classification problem detecting the presence of a disease in a person. There are 2 possibilities; yes or no.

True positives (TP): Cases where the model predicted yes and they do have a disease
True negatives (TN): Predicted no and they don't have the disease.
False positives (FP): Predicted yes, but they don't actually have the disease (Type I error)
False negatives (FN): Predicted no, but they do have the disease (Type II error)

Creating confusion matrix..

```
y_predicted=model.predict(xtest)

cm=confusion_matrix(y_predicted,ytest)

import seaborn as sn

sn.heatmap(cm,annot=True)

plt.xlabel('predicted')

plt.ylabel('truth')

plt.figure(figsize=(20,7))
```

TECHOLAS
TECHNOLOGY DEMYSTIFIED

Classification report

It is used to measure the quality of predictions of a classification algorithm.

Support: It is the number of actual occurrences of each class in the test set. It reflects whether the dataset is balanced or not.

Precision: It is the no: of true positives out of total positive predictions. It is the ability of a classifier not to label an instance positive that is actually negative.
Precision = TP/(TP+FP)

continues...

Recall: It is the fraction of positives that were correctly identified. It defines the ability of a classifier to find all positive instances.
Recall = TP/(TP+FN)

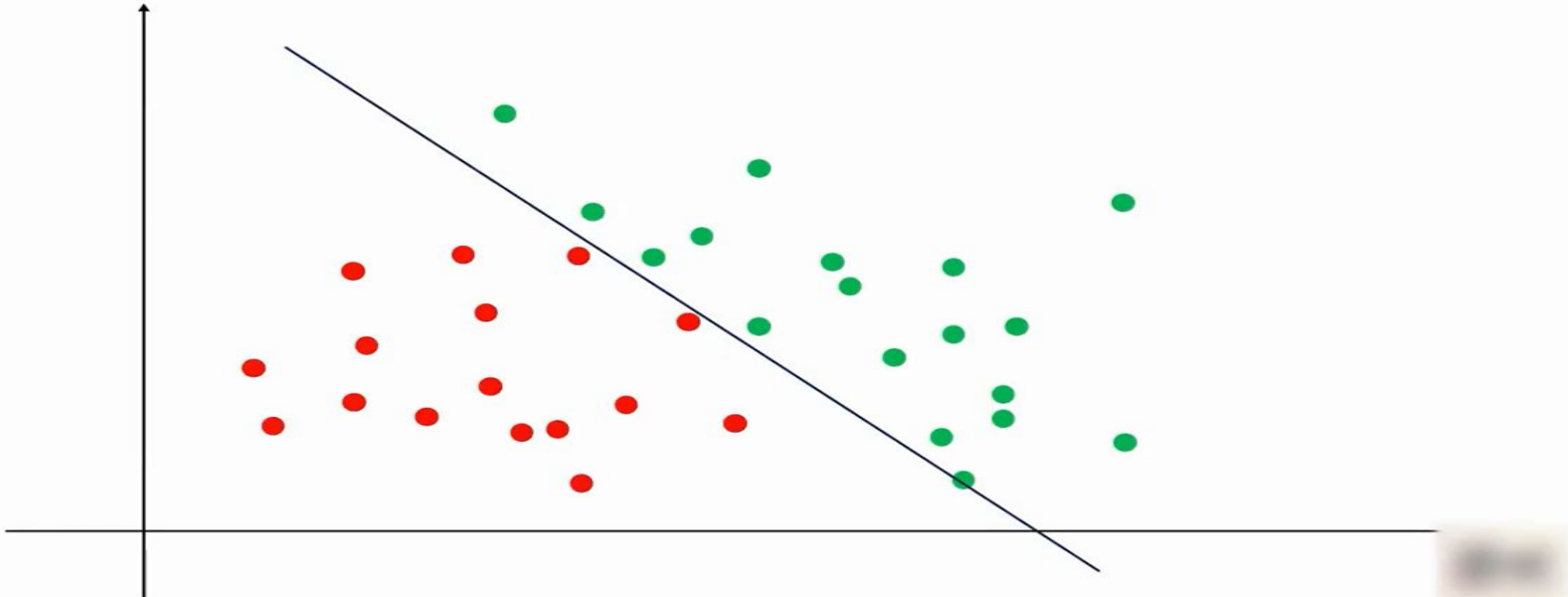
F1 score: It is the weighted mean of precision and recall such that the best score is 1 and the worst is 0.
F1 score = (2*recall*precision)/(recall+precision)

Macro average: It is the normal average of these metrics

Weighted average: Weighted average of the metrics which differs from the macro average for unbalanced datasets.

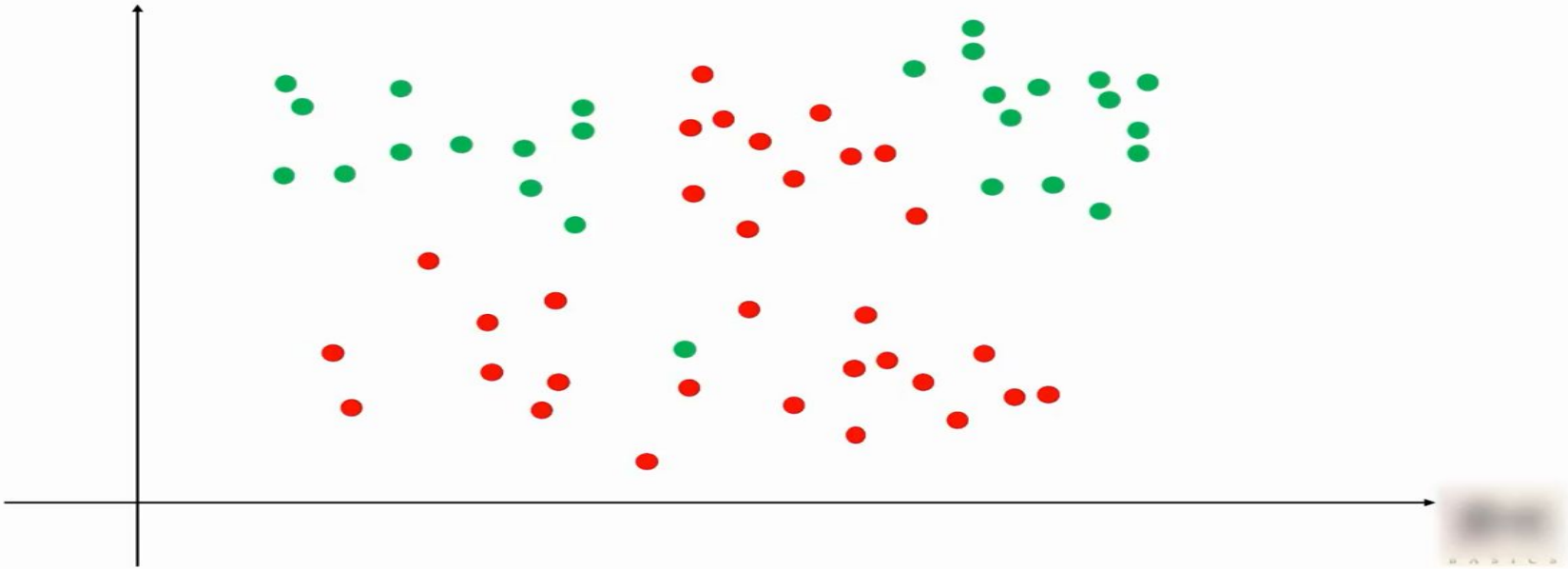
Decision Tree Algorithm..

When data is like this you can easily draw a decision boundary as a straight line and you can use linear regression to do the prediction



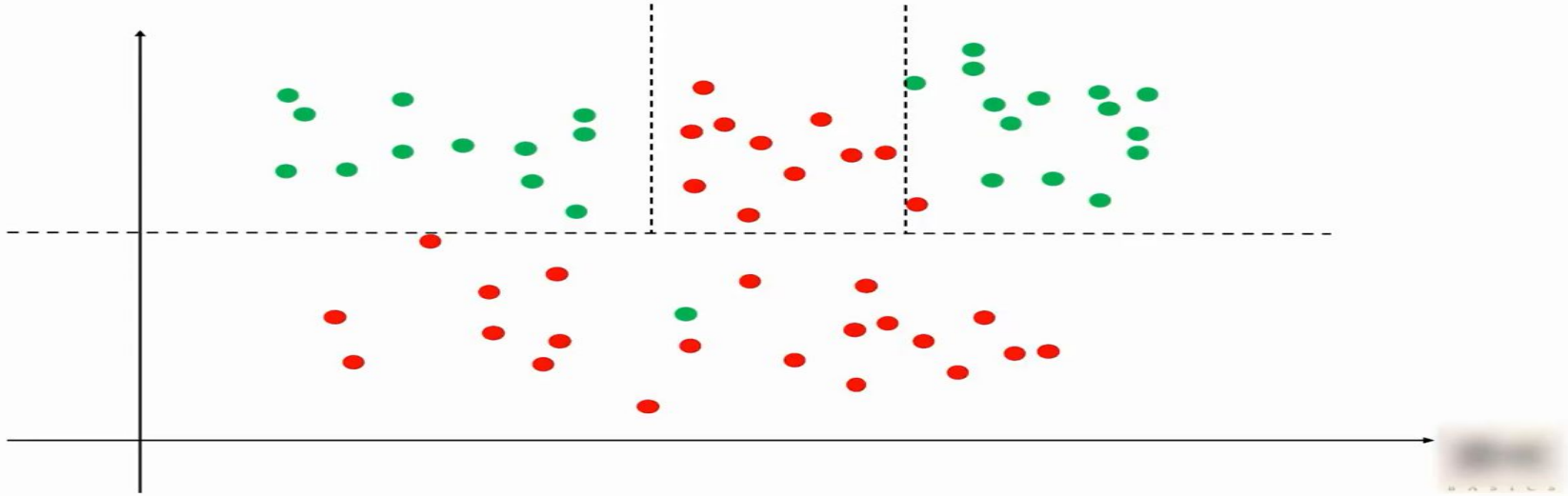
When data is complex..

It is difficult to find the decision boundary.. Ie its not possible to draw a single line to create the boundaries.so linear regression is not possible here.

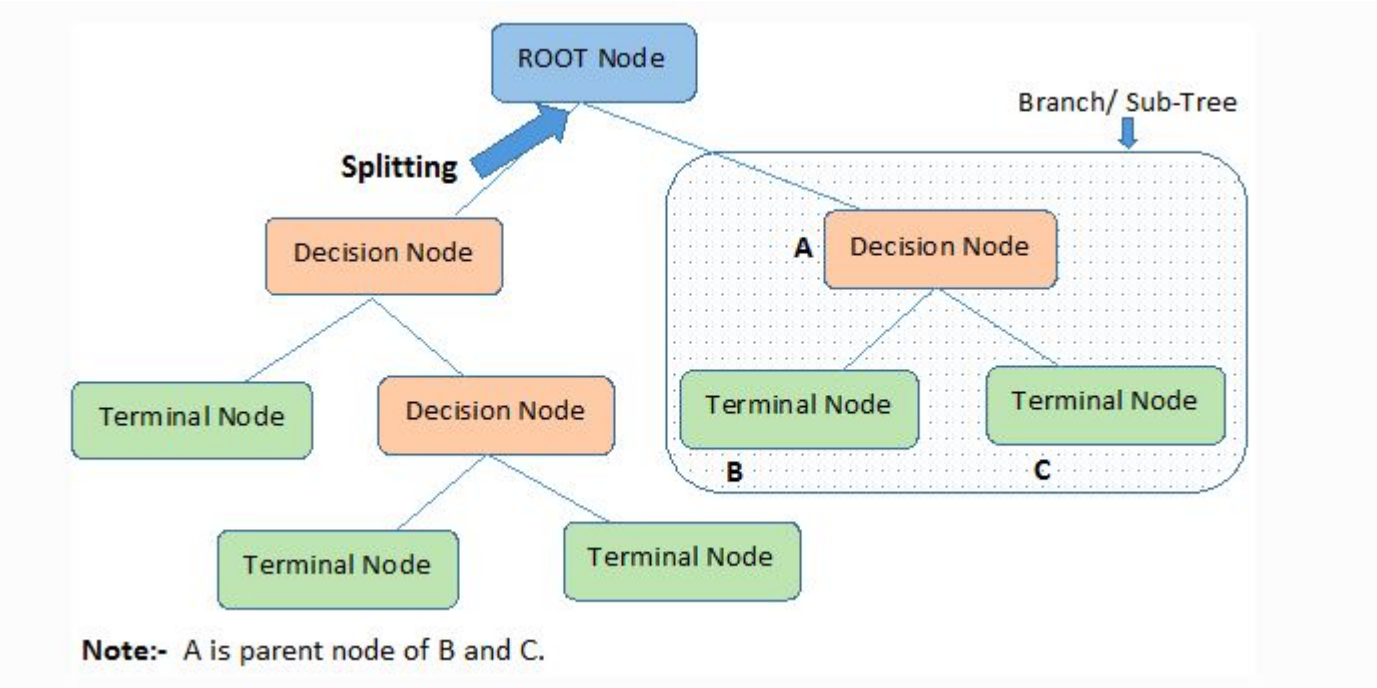


then??

You need to have different different boundaries to predict.. We use decision tree to solve this kind of problem. It is used for both classification and regression problems.



Structure of a decision tree



TECHOLAS
TECHNOLOGY DEMYSTIFIED

Terminologies related to decision tree

Root Node: It represents the entire population and can be further divided into two or more sets.

Splitting: Dividing a node into two or more sub-nodes.

Decision node: When a sub-node splits into further sub-nodes, it is called the decision node.

Leaf/Terminal node: Nodes that do not split further.

Pruning: Removing sub-nodes of a decision node is called pruning; opposite of splitting.

continues..

Branch/sub-tree: a sub-section of an entire tree

Parent and Child node: A node which is divided into sub-nodes is called a parent node of sub-nodes whereas sub-nodes are the child of a parent node.

How the algorithm works

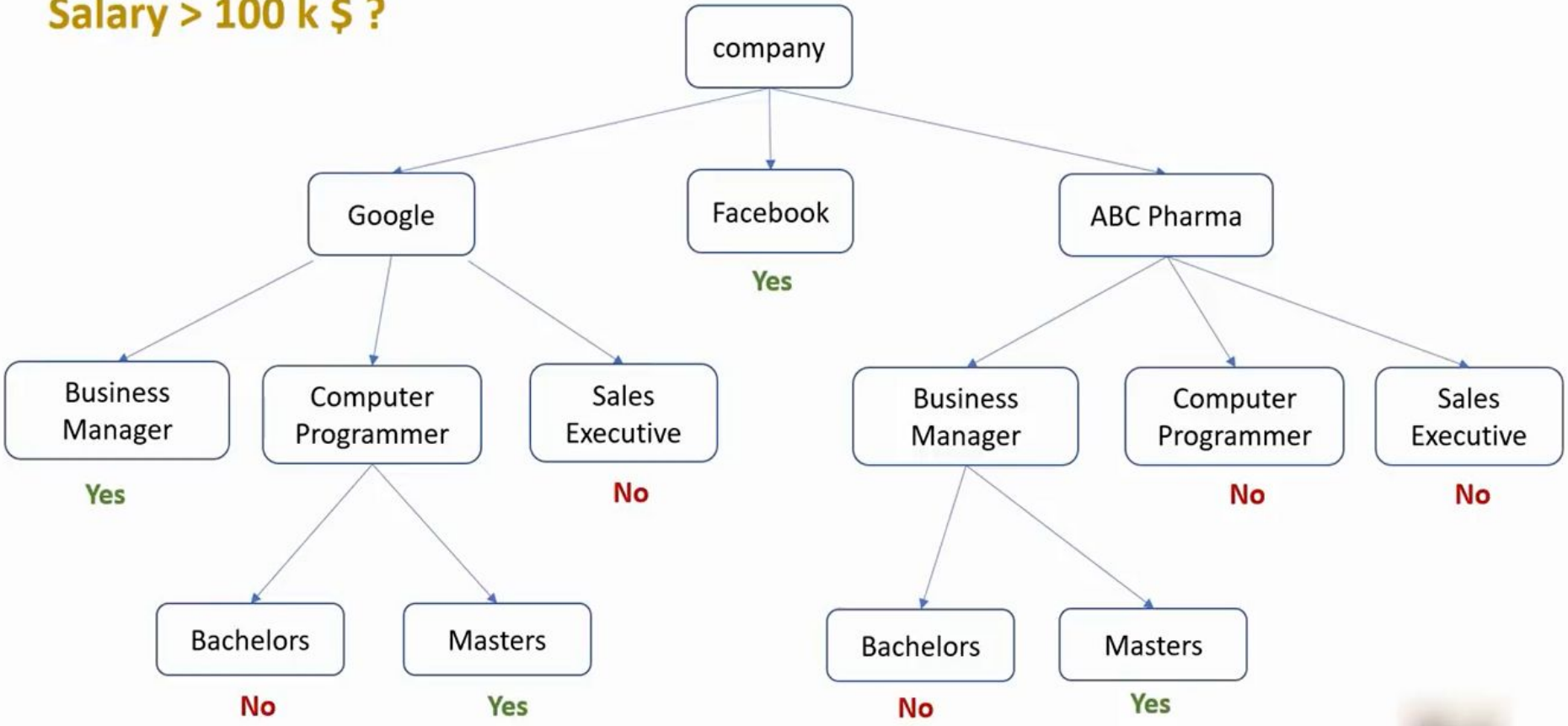
Decision trees use multiple algorithms to decide to split a node into two or more sub-nodes. If the dataset contains N attributes, then deciding which attribute to be place at the root or at different levels of the tree is done based on criteria like entropy, information gain etc.

The higher the entropy, the more randomness (impurity) a node has.

The attribute with the higher information gain is places at the root.

Company	Job	Degree	Salary_more_then_100k
google	sales executive	bachelors	0
google	sales executive	masters	0
google	business manager	bachelors	1
google	business manager	masters	1
google	computer programmer	bachelors	0
google	computer programmer	masters	1
abc pharma	sales executive	masters	0
abc pharma	computer programmer	bachelors	0
abc pharma	business manager	bachelors	0
abc pharma	business manager	masters	1
facebook	sales executive	bachelors	1
facebook	sales executive	masters	1
facebook	business manager	bachelors	1
facebook	business manager	masters	1
facebook	computer programmer	bachelors	1
facebook	computer programmer	masters	1

Salary > 100 k $?


company
Google
Facebook
ABC Pharma
Yes
Business Manager
Computer Programmer
Sales Executive
Yes
No
Business Manager
Computer Programmer
Sales Executive
No
No
Bachelors
Masters
No
Yes
Bachelors
Masters
No
Yes

In notebook

```
import pandas as pd

from sklearn import tree

from sklearn.preprocessing import LabelEncoder
```

Load the data..

```
data=pd.read_csv('salaries.csv')

inputs=data.drop(['salary_more_then_100k'],axis='columns')

target=data['salary_more_then_100k']
```

Data preprocessing..

```
le_data=LabelEncoder()

company_l=le_data.fit_transform(inputs['company'])

job_l=le_data.fit_transform(inputs['job'])

degree_l=le_data.fit_transform(inputs['degree'])

inputs['company']=company_l

inputs['job']=job_l

inputs['degree']=degree_l
```

TECHOLAS
TECHNOLOGY DEMYSTIFIED

Create model predict..

model=tree.DecisionTreeClassifier()

model.fit(inputs,target)

model.score(inputs,target)

model.predict([[2,2,0]])

data

assignment..

In this file using following columns build a model to predict if person would survive or not,

1. Pclass
2. Sex
3. Age
4. Fare

Calculate score of your model

RANDOM FOREST ...

Random forest algorithm is another machine learning technique. It is an ensemble model. Ensemble models are those which combine predictions from two or more models.

From the name ,it’s forest. Forest means too many trees. And tree in machine learning is decision tree algorithm.

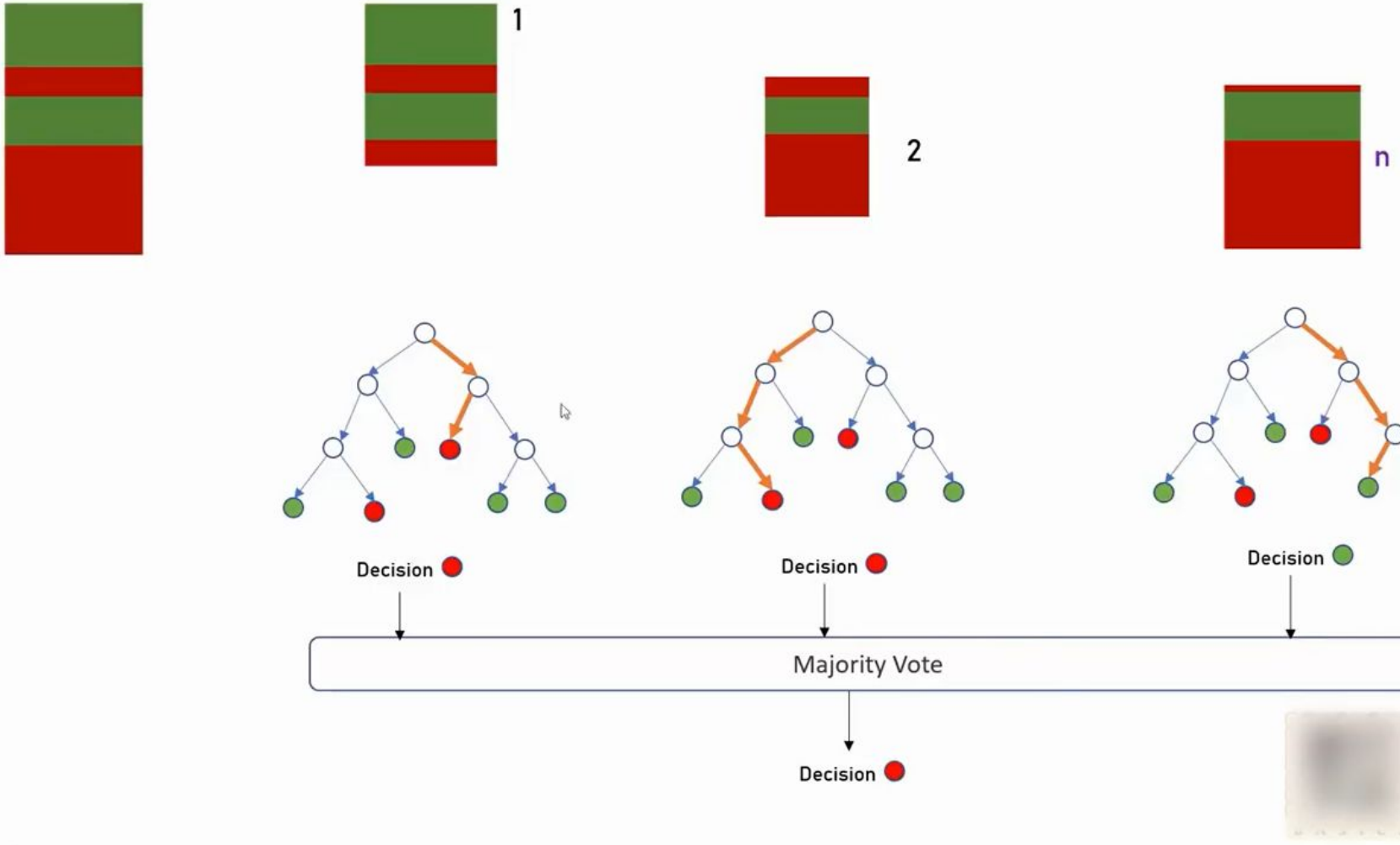
So its randomly creating number of decision trees from a dataset by choosing number of random datasets.

continues..

Ensemble techniques are of 2 types: bagging and boosting. Random Forest algorithm is a bagging technique.

It is used both for classification as well as regression problems.

Each decision tree will make a decision and the majority decision is chosen in the case of classification and the mean of the decisions is chosen in case of regression.


1
2
n
Decision
Decision
Decision
Majority Vote
Decision

In notebook..

```
import pandas as pd

from matplotlib import pyplot as plt

import seaborn

from sklearn.datasets import load_digits

from sklearn.model_selection import train_test_split

from sklearn.ensemble import RandomForestClassifier

digits=load_digits()
```

TECHOLAS
TECHNOLOGY DEMYSTIFIED

Cross check the data..

- for i in range(3):
  plt.matshow(digits.images[i])
- digits.target
- digits.target_names

Create the data frame and split..

```
data=pd.DataFrame(digits.data)

data['target']=digits.target

X=data.drop(['target'],axis=1)

Y=data.target

xtrain,xtest,ytrain,ytest=train_test_split(X,Y,test_size=.2)
```

TECHOLAS
TECHNOLOGY DEMYSTIFIED

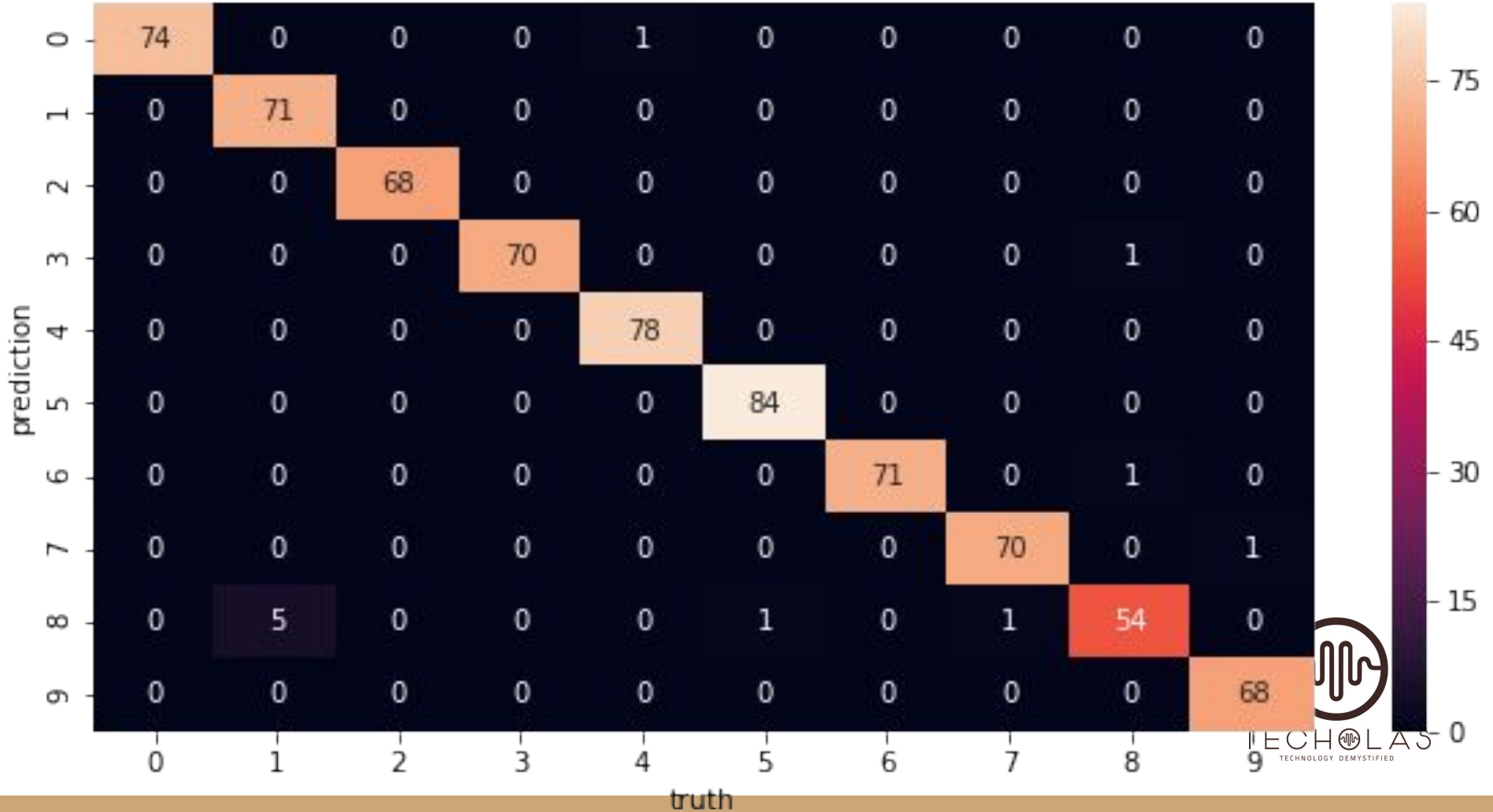
Create the model..

```
model=RandomForestClassifier()

model.fit(xtrain,ytrain)

model.score(xtest,ytest)
```

Confusion matrix..

```
y_truth=ytest

y_prdicted=model.predict(xtest)

from sklearn.metrics import confusion_matrix

cm=confusion_matrix(y_truth,y_prdicted)

plt.figure(figsize=(10,7))

seaborn.heatmap(cm,annot=True)
```

TECHOLAS
TECHNOLOGY DEMYSTIFIED


prediction
truth
0 1 2 3 4 5 6 7 8 9
74 0 0 0 1 0 0 0 0 0
0 71 0 0 0 0 0 0 0 0
0 0 68 0 0 0 0 0 0 0
0 0 0 70 0 0 0 0 1 0
0 0 0 0 78 0 0 0 0 0
0 0 0 0 0 84 0 0 0 0
0 0 0 0 0 0 71 0 1 0
0 0 0 0 0 0 0 70 0 1
0 5 0 0 0 1 0 1 54 0
0 0 0 0 0 0 0 0 0 68
75
60
45
30
15
0
TECHOLAS
TECHNOLOGY DEMYSTIFIED

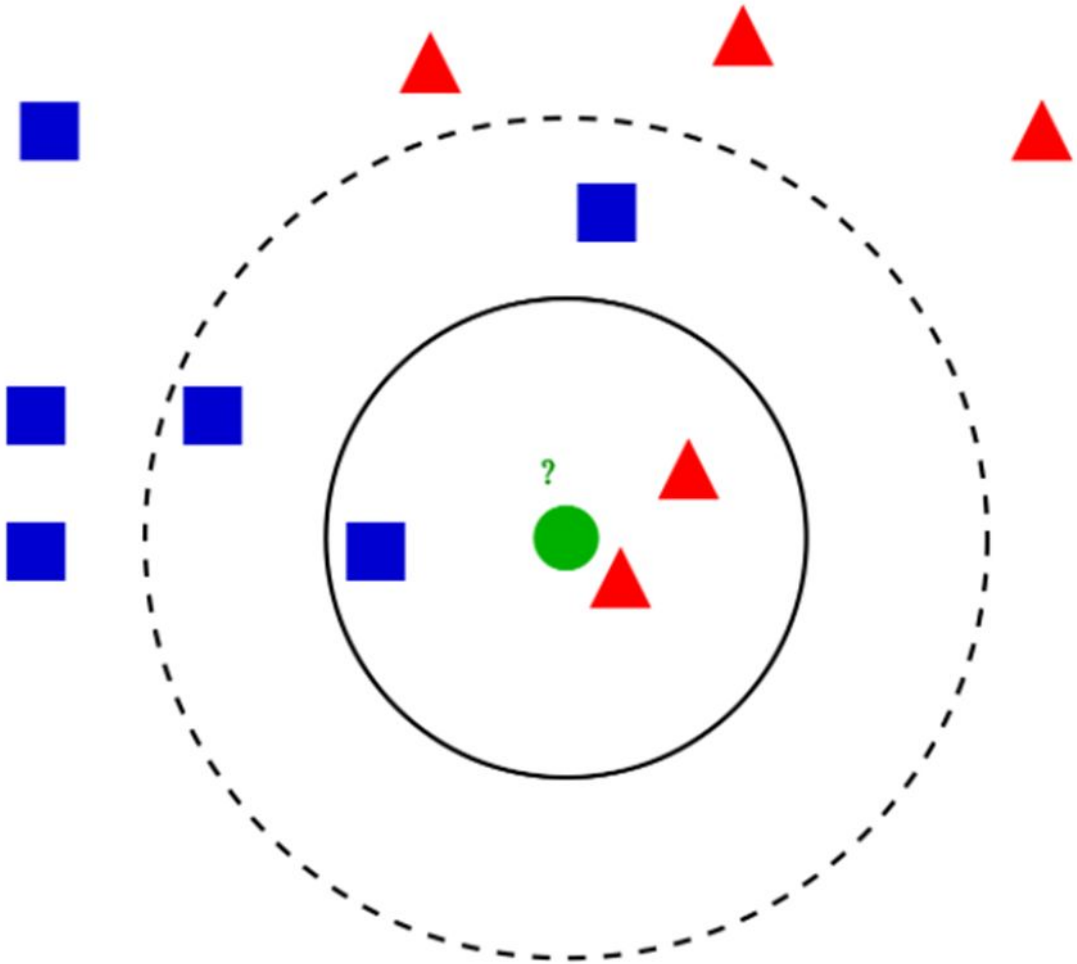
Assignment..

Use famous iris flower dataset from sklearn.datasets to predict flower species using random forest classifier.

1. Measure prediction score using default n_estimators (10)
2. Now fine tune your model by changing number of trees in your classifier and tell me what best score you can get using how many trees

K Nearest Neighbor (KNN)

- Used for both classification and regression problems
- Uses feature similarity to predict values
- Chooses the k nearest neighbors of the new data point and predicts the target value (in case of regression) or target class (in case of classification) accordingly.



TECHOLAS
TECHNOLOGY DEMYSTIFIED

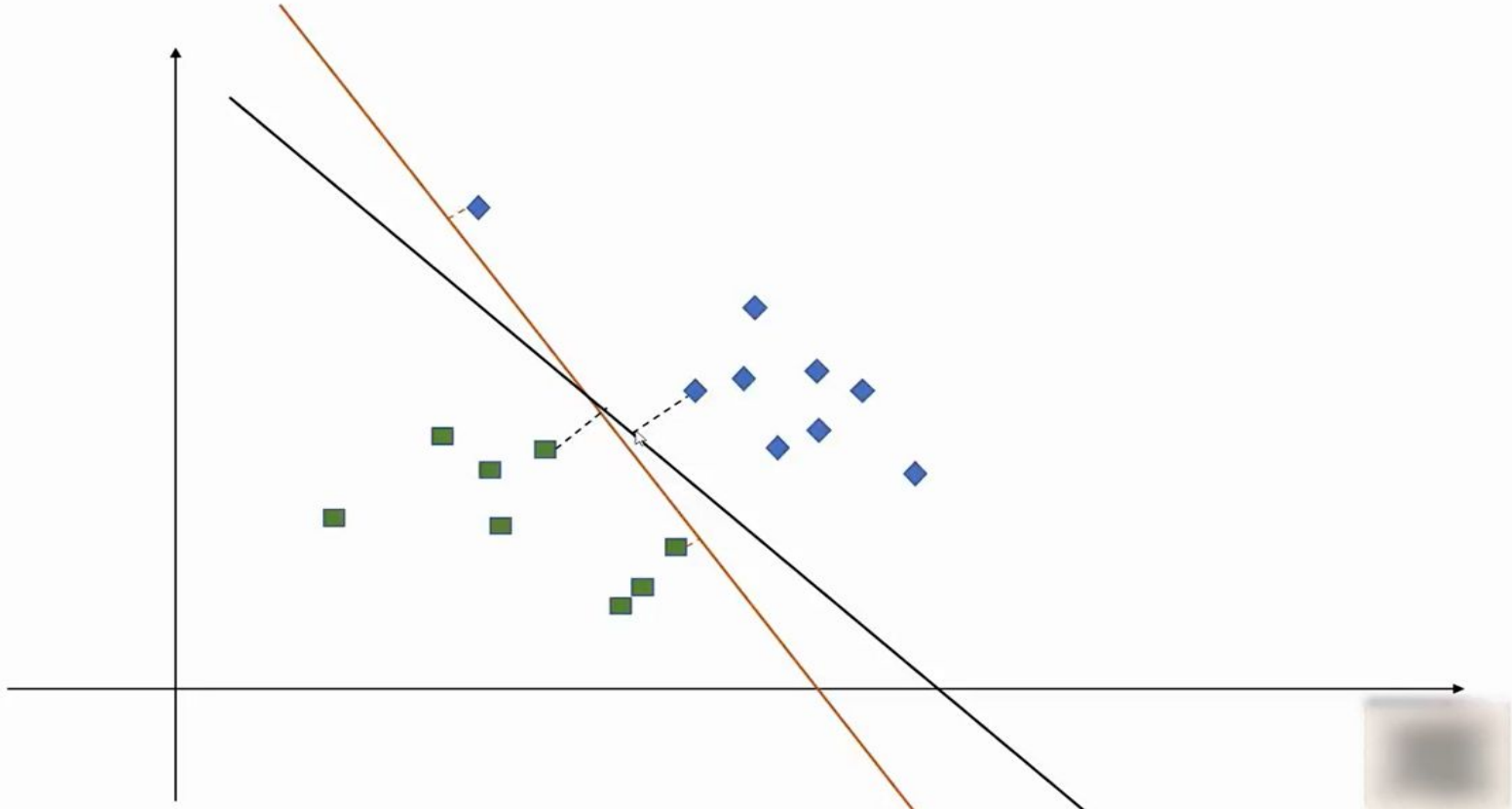
Steps (Classification)

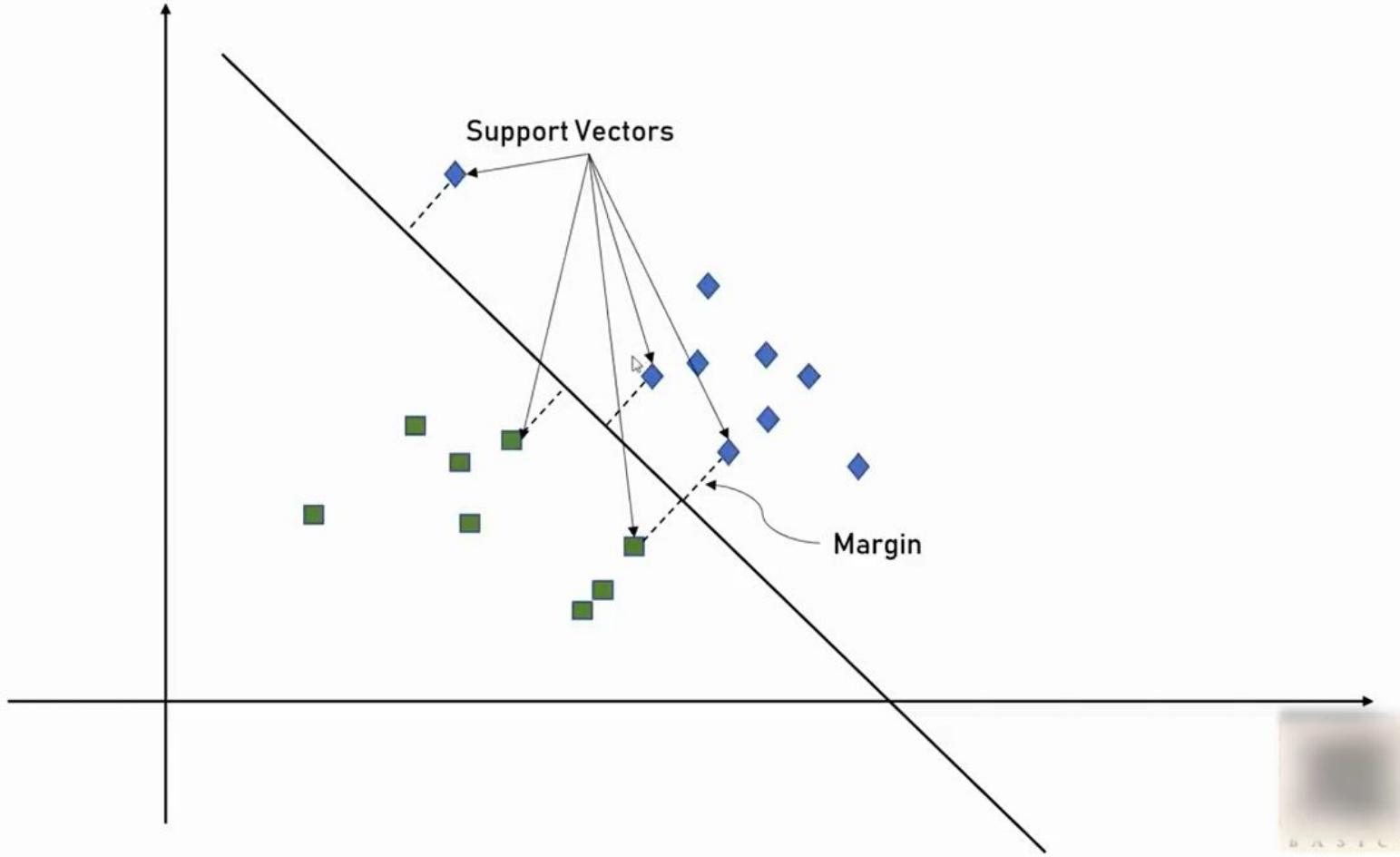
- Select K (no of neighbors)
- Find K nearest neighbors of the new data point based on Euclidean distance
- Among the K neighbors find the number of neighbors in each class.
- Assign the data point to that class having most number of neighbors

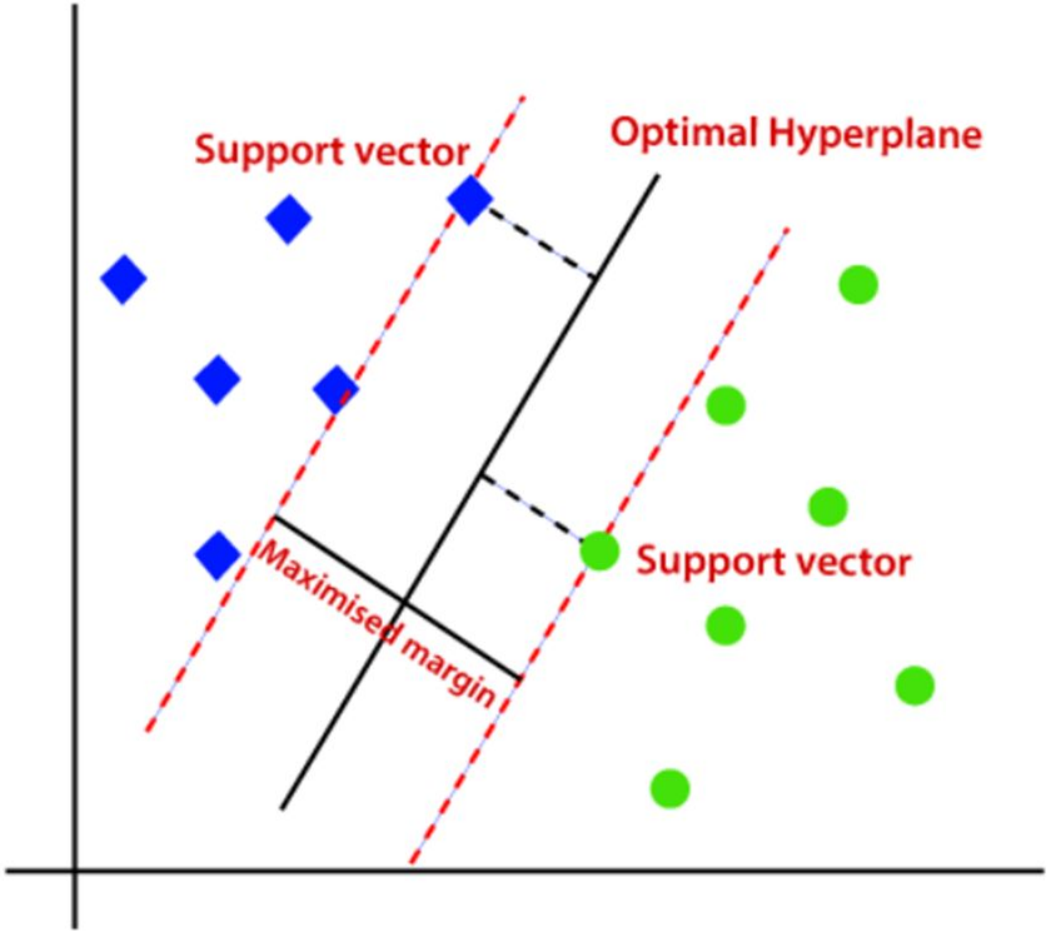
Steps (Regression)

- Select K
- Choose K nearest neighbors of the new point
- Make the target prediction by taking the average of all these neighboring data points.

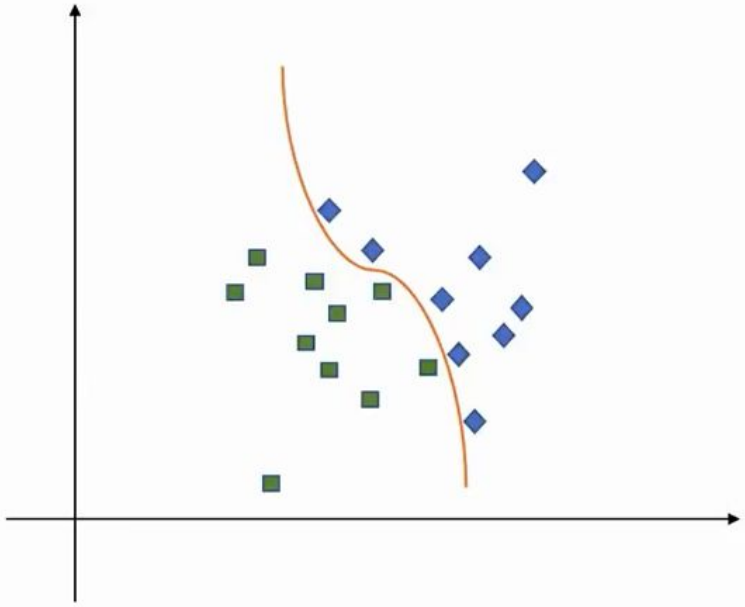
Support vector machine..



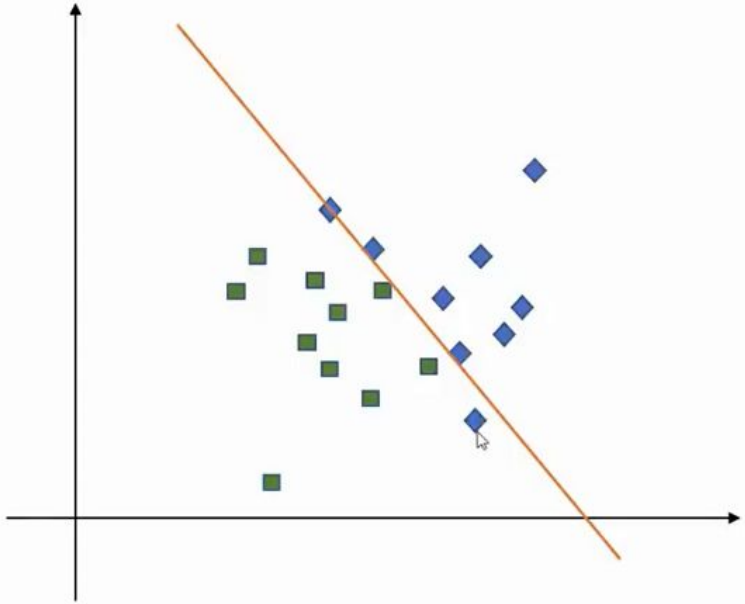

Support Vectors
Margin



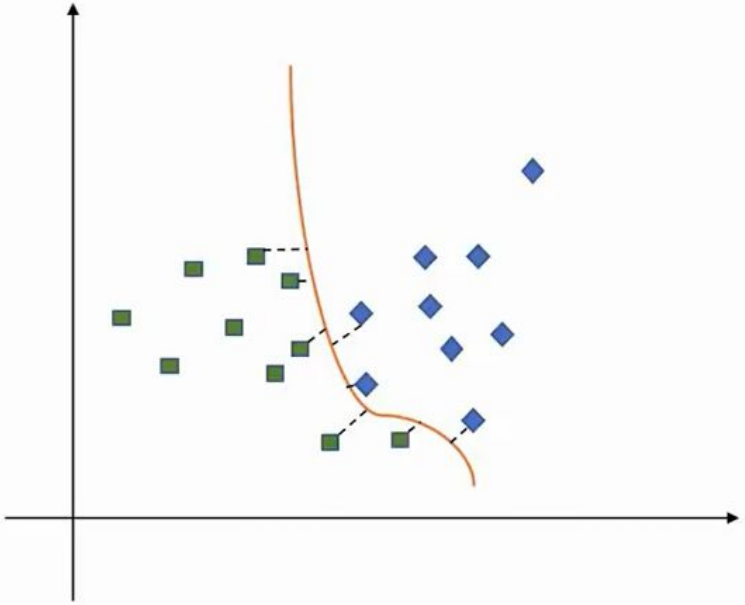
TECHOLAS
TECHNOLOGY DEMYSTIFIED



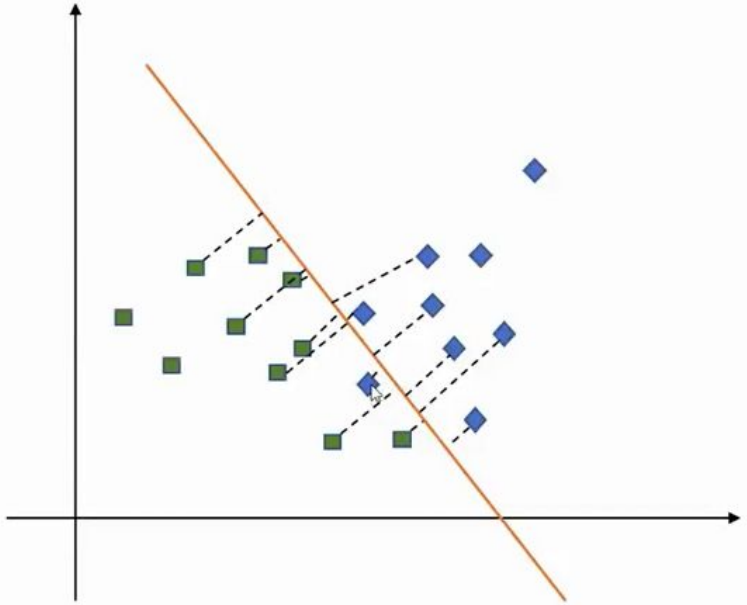
High Regularization (C)



Low Regularization (C)

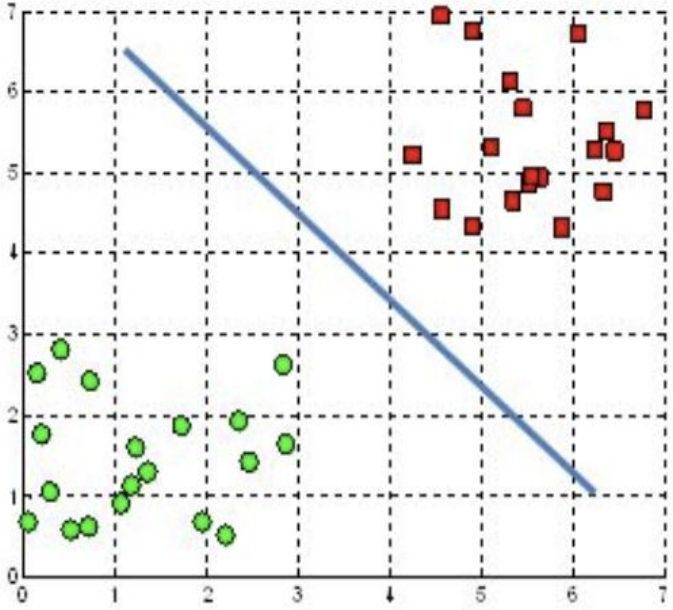


High Gamma

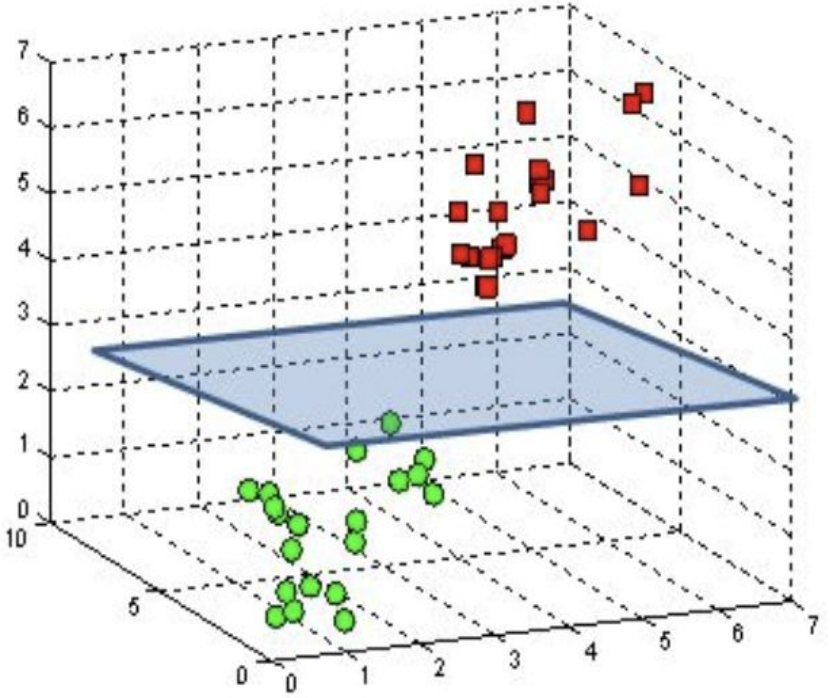


Low Gamma

A hyperplane in $\mathbb{R}^2$ is a line



A hyperplane in $\mathbb{R}^3$ is a plane



TECHOLAS
TECHNOLOGY DEMYSTIFIED

hyperplane

Hyperplanes are decision boundaries that help classify the data points.

Data points falling on either side of the hyperplane can be attributed to different classes. Also, the dimension of the hyperplane depends upon the number of features. If the number of input features is 2, then the hyperplane is just a line.

If the number of input features is 3, then the hyperplane becomes a two-dimensional plane. It becomes difficult to imagine when the number of features exceeds 3.

Support vectors..

Support vectors are data points that are closer to the hyperplane and influence the position and orientation of the hyperplane.

Using these support vectors, we maximize the margin of the classifier.

Svm in notebook..

```
import pandas as pd

from sklearn.datasets import load_iris

iris=load_iris()

from matplotlib import pyplot as plt

from sklearn.model_selection import train_test_split

from sklearn.svm import SVC
```

Load data..

```
dir(iris)

iris.feature_names

data=pd.DataFrame(iris.data,columns=iris.feature_names)

data['targets']=iris.target

iris.target_names

data['flower_name']=data.targets.apply(lambda x:iris.target_names[x])
```

Plot it..

```
data0=data[data.targets==0]
data1=data[data.targets==1]
data2=data[data.targets==2]

plt.scatter(data0['sepal length (cm)'],data0['sepal width (cm)'],marker='+',color='orange')

plt.scatter(data1['sepal length (cm)'],data1['sepal width (cm)'],marker='*',color='green')

plt.scatter(data2['petal length (cm)'],data2['petal width (cm)'],marker='+',color='orange')plt.scatter(data1['petal length (cm)'],data1['petal width (cm)'],marker='*',color='green')
```

split..

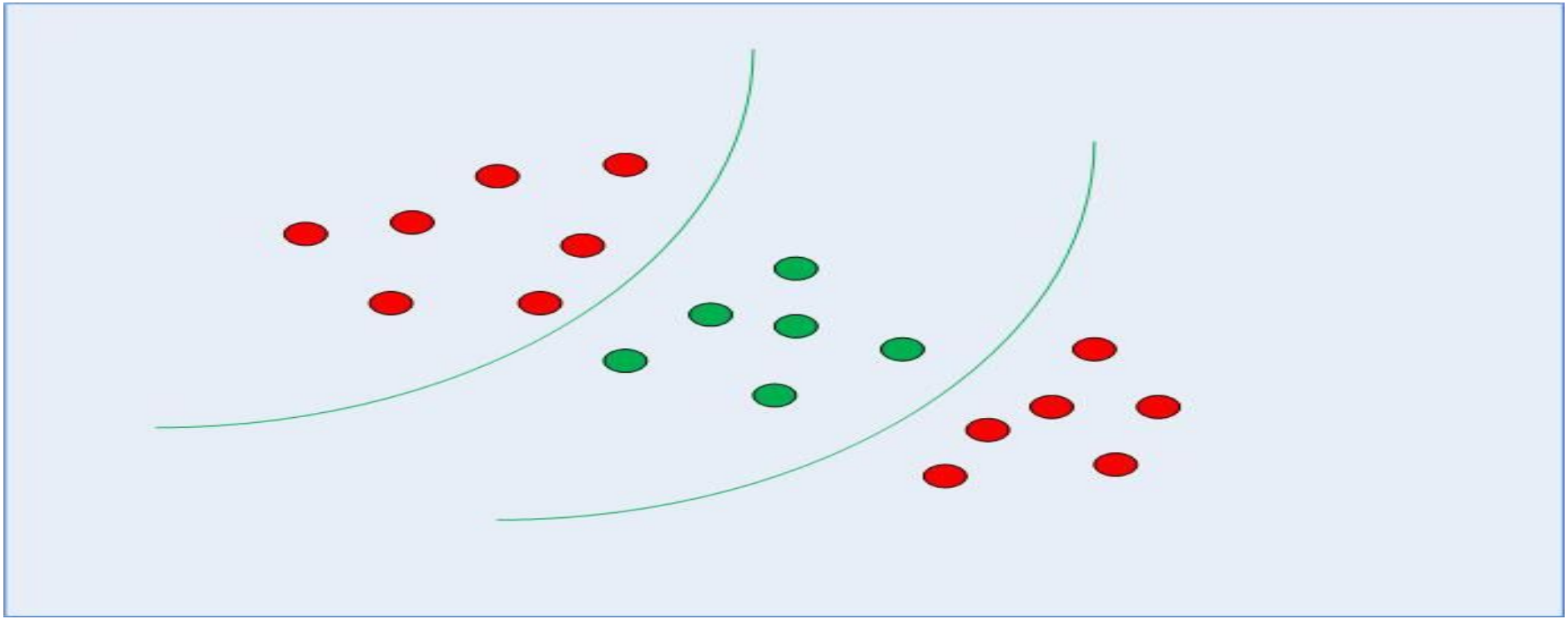
```
x=data.drop(['targets','flower_name'],axis='columns')

y=data.targets

xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=.2)
```

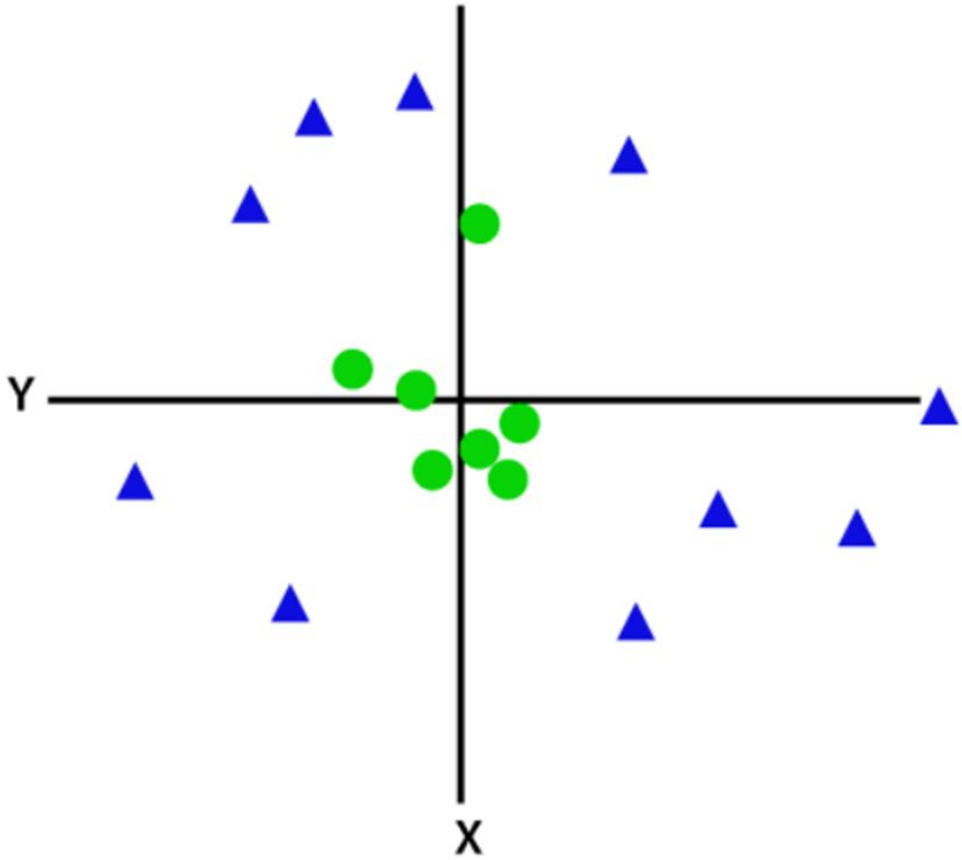
Create the model and test it..

```
model=SVC()

model.fit(xtrain,ytrain)

model.score(xtest,ytest)

model.predict(xtest)
```

Kernel SVM

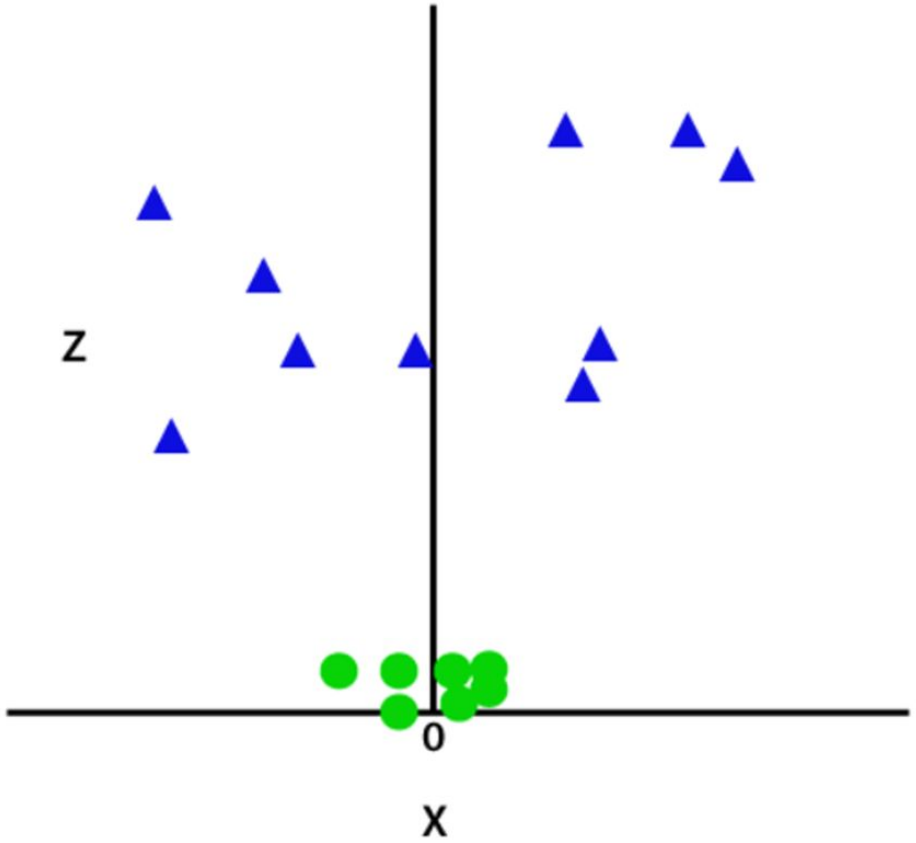
In the case of non-linearly separable data, such as the one shown below, a straight line cannot be used as a decision boundary.



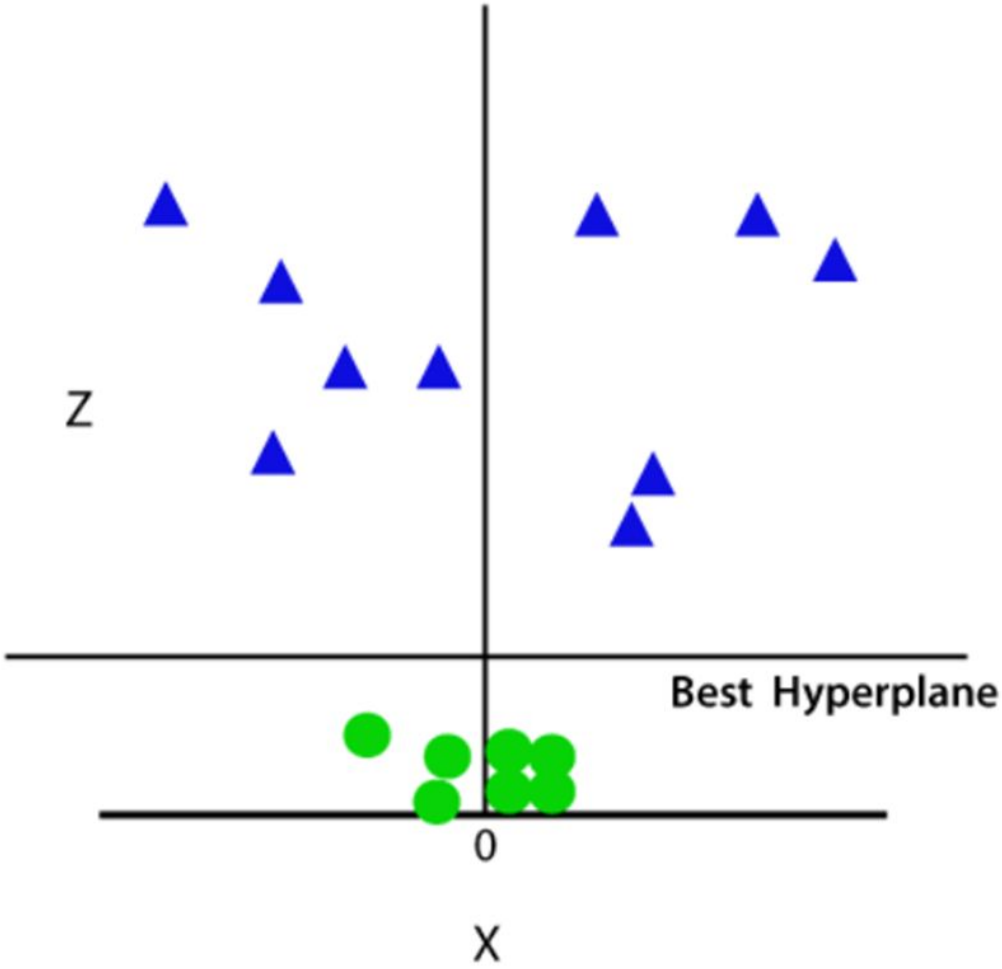


TECHOLAS
TECHNOLOGY DEMYSTIFIED

Adding a 3rd dimension $z = x^2 + y^2$

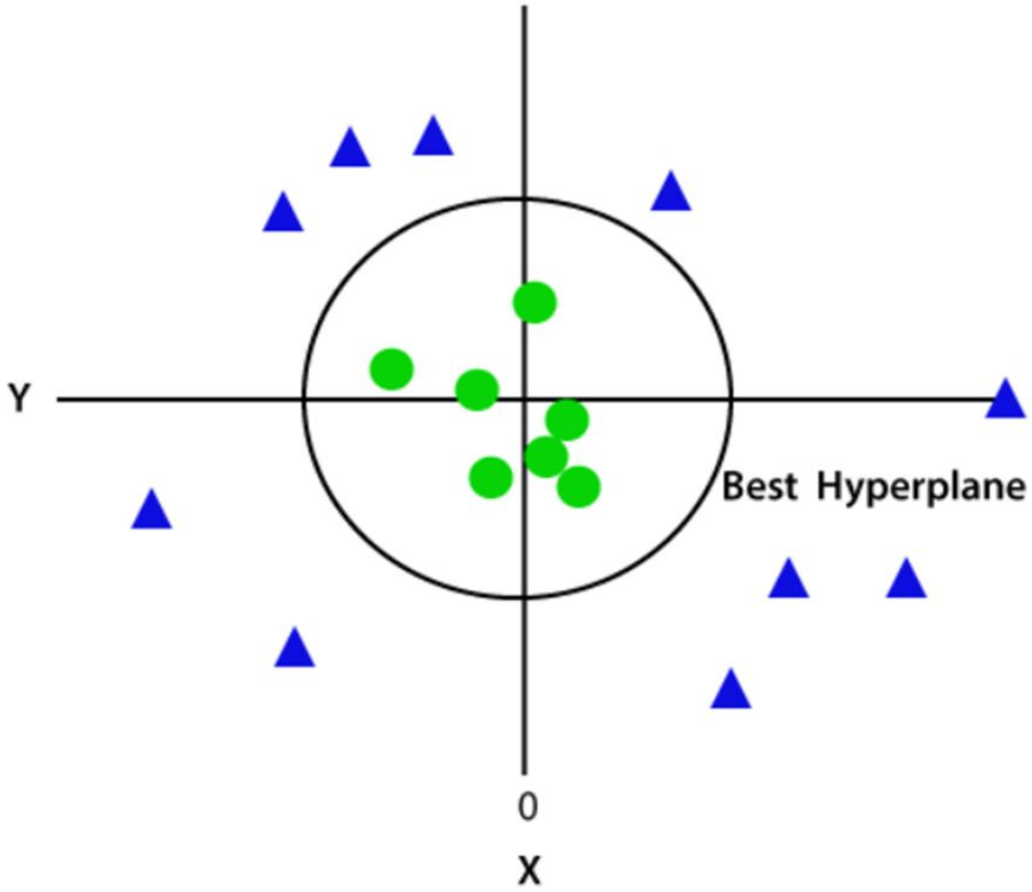


TECHOLAS
TECHNOLOGY DEMYSTIFIED



TECHOLAS
TECHNOLOGY DEMYSTIFIED

Transforming back to 2D plane with z=1



TECHOLAS
TECHNOLOGY DEMYSTIFIED

Kernel svm continues..

In case of non-linearly separable data, the simple SVM algorithm cannot be used. Rather, a modified version of SVM, called Kernel SVM, is used.

In case of linear the kernal is linear

Linear kernel

svclassifier = SVC(kernel=linear)

Different kernels..

1. Polynomial Kernel

In the case of polynomial kernel, you also have to pass a value for the `degree` parameter of the `SVC` class. This basically is the degree of the polynomial. Take a look at how we can use a polynomial kernel to implement kernel SVM:

```
svclassifier = SVC(kernel='poly', degree=8)
```

continues..

2. Gaussian Kernel

Take a look at how we can use polynomial kernel to implement kernel SVM:

svclassifier = SVC(kernel='rbf')

The default kernel is rbf

exercise..

Train SVM classifier using sklearn digits dataset (i.e. from sklearn.datasets import load_digits) and then,

1. Measure accuracy of your model
2. Tune your model further using regularization and gamma parameters and try to come up with highest accuracy score
3. Use 80% of samples as training data size

Naive bayes Classifier..

Naive Bayes classifiers are a collection of classification algorithms based on **Bayes' Theorem**

Bayes' Theorem:

Bayes' Theorem finds the probability of an event occurring given the probability of another event that has already occurred. Bayes' theorem is stated mathematically as the following equation:

Conditional Probability: Bayes' Theorem

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

Example

Consider the following data

Event	Will play/not
Rainy	No
Sunny	Yes
Sunny	Yes
Rainy	No
Rainy	Yes
Sunny	No

	Yes	No	Total
Rainy	1	2	3
Sunny	2	1	3
Total	3	3	6

CONT...

P(Yes|Sunny) = (P(Sunny|yes) P(yes))/P(sunny)

= (2/3 .3/6)/ (3/6) = 2/3

P(No|Sunny)= (P(Sunny|no) P(no))/P(sunny)

= (1/3 .3/6)/ (3/6) = 1/3

Here, P(Yes|Sunny) > P(No|Sunny)

Thus, a child has more probability to go out for playing if it is sunny

Naive bayes

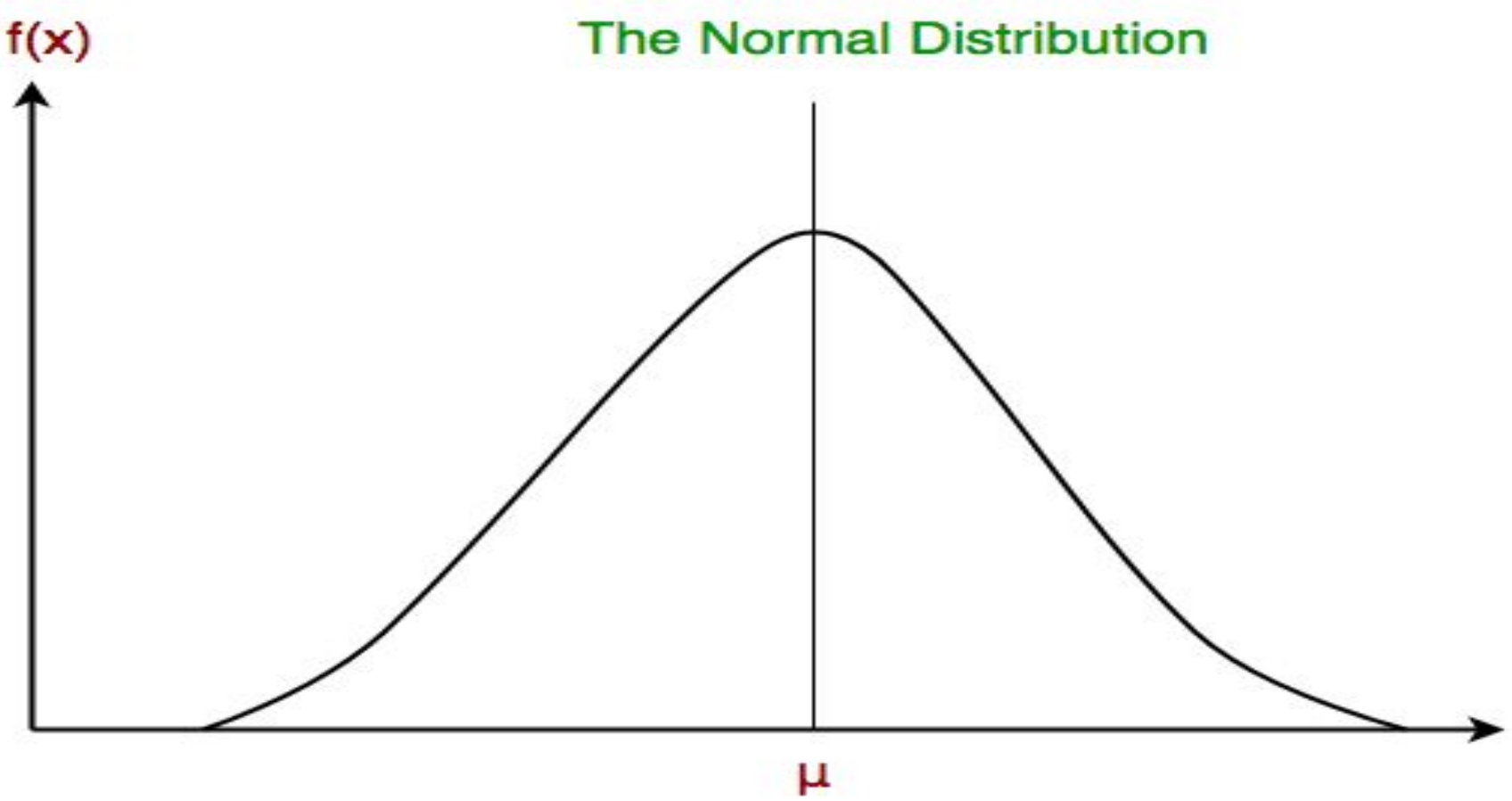
The fundamental Naive Bayes assumption is that each feature makes an:

- independent
- Equal

  contribution to the outcome.

Gaussian Naive Bayes classifier

In Gaussian Naive Bayes, continuous values associated with each feature are assumed to be distributed according to a **Gaussian distribution**. A Gaussian distribution is also called Normal distribution. When plotted, it gives a bell shaped curve which is symmetric about the mean of the feature values as shown below:


f(x)
The Normal Distribution
μ

In notebook..

```
import pandas as pd

import numpy as np

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

data.drop(['PassengerId','Name','SibSp','Parch','Ticket','Cabin','Embarked'],axis=1,
inplace=True)

input=data.drop(['Survived'],axis=1)

target=data.Survived
```

Create the proper data..

- dummies=pd.get_dummies(data.Sex)
- inputs=pd.concat([inputs,dummies],axis=1)
- inputs.columns[inputs.isna().any()]
- inputs.Age=inputs.Age.fillna(inputs.Age.mean())
- inputs.columns[inputs.isna().any()]
- from sklearn.model_selection import train_test_split
- xtrain,xtest,ytrain,ytest=train_test_split(inputs,target,test_size=.3)

Create the model and find the score..

```
from sklearn.naive_bayese  import GaussianNB

model=GaussianNB()

model.fit(xtrain,ytrain)

model.score(xtest,ytest)
```

Predict the probability

```
model.predict_proba(xtest)
```

Spam detector..

```
import pandas as pd

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

data.head()
```

TECHOLAS
TECHNOLOGY DEMYSTIFIED

Make data numbers..

```
data['spam']=data.Category.apply(lambda x : 1 if x=='ham'  else 0)

data=data.drop(['Category'],axis=1)

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(data.Message,data.spam)
```

But still message column is there....What to do???

This is the first document

This document is the second document

And this is the third one.

Is this the first document?

and, document, first, is, one, second, the, third, this

and	document	first	is	one	second	the	third	this
0	1	1	1	0	0	1	0	1
0	2	0	1	0	1	1	0	1
1	0	0	1	1	0	1	1	1
0	1	1	1	0	0	1	0	1

Use vectorizer..

```
from sklearn.feature_extraction.text import CountVectorizer

cv=CountVectorizer()

xtrain_counter=cv.fit_transform(X_train.values)

xtrain_counter.shape
```

Model and predict..

```
from sklearn.naive_bayes import MultinomialNB

model=MultinomialNB()

model.fit(xtrain_counter,y_train)
```

predict

```
emails = [ 'Hey mohan, can we get together to watch footbal game tomorrow?',

    'Upto 20% discount on parking, exclusive offer just for you. Dont miss this reward!'

]

emails_counter=cv.transform(emails)

model.predict(emails_counter)

model.score(cv.transform(X_test),y_test)
```

Using pipeline..

```
from sklearn.pipeline import Pipeline

p=Pipeline([
    ('v',CountVectorizer()),
    ('mnb',MultinomialNB())
])

p.fit(X_train,y_train)

p.predict(X_test)
```

Exercise..

Use sklearn wine dataset to classify wine into 3 categories..

Try both Gaussian and multinomial..

CROSS VALIDATION..

Evaluating model performance..

We have different machine learning classifier like logistic regression,decision tree,random forest,SVM

But when we are dealing with a problem or a dataset ,how we will choose a classifier??

For that we should know which one is scoring better.So for that purpose we can choose cross validation..

Different ways to do validate..

Option 1: Use the whole dataset for training and use the same dataset for testing

But this is not the best method because while we are training the data the model already seen the data.and we are testing on the same data.

Option 2

Split the available dataset into train and test

```
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest=train_test_split(X,Y,test_size=.3)
```

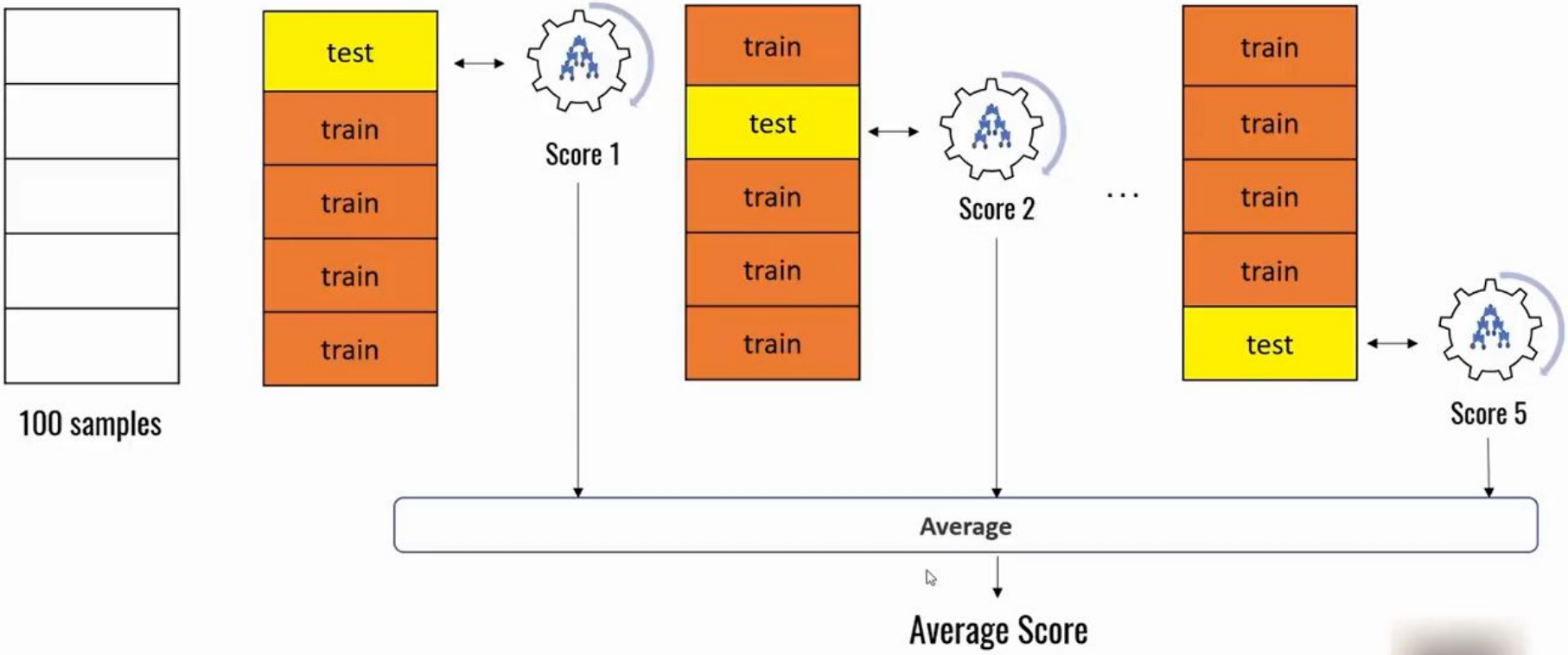
The 1st problem with this approach is the splitted data may not be uniform,for example if we are training some maths questions to our model and when we splits consider we got our train model full of trigonometry questions and the test contains full of exponential questions.then the model will fail

The 2nd problem with this approach is ,when ever you run the above code it will give different different datasets.So the score will vary.So we need to take number of times to validate which classifier performs better.

Option 3

K FOLD cross validation..

In this, we first divide the whole dataset into number of folds and will have that many iterations also .i.e, in each iteration, one fold will be set as the test data and the remaining folds as training data. For the next iteration it will take another fold for testing and all others for training. This is done for all the folds.


100 samples
test
train
train
train
train
Score 1
train
test
train
train
train
Score 2
...
train
train
train
train
test
Score 5
Average
Average Score

In notebook..kfold sample

```
from sklearn.model_selection import KFold
fold=KFold(n_splits=4)
fold

for train_index,test_index in fold.split([1,2,3,4,5,6,7,8,9,10,11,12]):
    print(train_index,test_index)
```

TECHOLAS
TECHNOLOGY DEMYSTIFIED

Lets do it with digits data set..

```
from sklearn.ensemble import RandomForestClassifier

from sklearn.linear_model import LogisticRegression

from sklearn.svm import SVC

from sklearn.model_selection import StratifiedKFold

fold=StratifiedKFold(n_splits=4)

fold
```

Get the scores method..

```
def get_score_from_model(model,xtrain,xtest,ytrain,ytest):
    model.fit(xtrain,ytrain)
    return model.score(xtest,ytest)
```

Find the scores..

```
logistic_scores=[]
rf_scores=[]
SVC_scores=[]
for train_index,test_index in fold.split(digits.data,digits.target):
    xtrain,xtest,ytrain,ytest=digits.data[train_index],digits.data[test_index],\
    digits.target[train_index],digits.target[test_index]
    logistic_scores.append(get_score_from_model(LogisticRegression(),xtrain
    ,xtest,ytrain,ytest))
    rf_scores.append(get_score_from_model(RandomForestClassifier(),
    xtrain,xtest,ytrain,ytest))
    SVC_scores.append(get_score_from_model(SVC(),xtrain,xtest,ytrain,ytest))
```

continued..

```
print(logistic_scores)

print(rf_scores)

print(SVC_scores)
```

You can take the average from the list and find which classifier performs better..

In short way..

```
from sklearn.model_selection import cross_val_score

l_scores=cross_val_score(LogisticRegression(),X=digits.data,y=digits.target)

svc_scores=cross_val_score(SVC(),X=digits.data,y=digits.target)

rf_scores=cross_val_score(RandomForestClassifier(),X=digits.data,y=digits.target)

print(l_scores)

print(svc_scores)

print(rf_scores)
```

TECHOLAS
TECHNOLOGY DEMYSTIFIED

Exercise..

Using cross validation find which classifier is better to predict iris dataset.

Hyperparameter tuning by grid search cv..

For iris flower dataset in sklearn library, we are going to find out best model and best hyper parameters using GridSearchCV

```
from sklearn.model_selection import GridSearchCV

from sklearn.svm import SVC

from sklearn.datasets import load_iris

import pandas as pd

iris=load_iris()
```

Create grid Search cv for SVC

```
svc_model=SVC()

gds=GridSearchCV(svc_model,{'gamma':[0,1,10,20],\

                 'C':[1,5,10,20],\

                 'kernel':['linear','rbf','poly']},cv=5,return_train_score=True)
```

Get the result and store in a dataframe

gds.fit(iris.data,iris.target)

gds.cv_results_

data=pd.DataFrame(gds.cv_results_)

Find The Best

gds.best_params_

Find Best **models with different hyperparameters**

```
from sklearn import svm

from sklearn.ensemble import RandomForestClassifier

from sklearn.linear_model import LogisticRegression
```

continues..

```
model_params = {
    'svm': {
        'model': svm.SVC(gamma='auto'),
        'params' : {
            'C': [1,10,20],
            'kernel': ['rbf','linear']
        } },
    'random_forest': {
        'model': RandomForestClassifier(),
        'params' : {
            'n_estimators': [1,5,10]
        } },
    'logistic_regression' : {
        'model': LogisticRegression(),
        'params': {
            'C': [1,5,10]
        }
    }
```

TECHOLAS
TECHNOLOGY DEMYSTIFIED

continues..

```
scores=[]
for model_name in models:
    mp=models[model_name]
    gds=GridSearchCV(mp['model'],mp['params'],cv=5,return_train_score=True)
    gds.fit(iris.data, iris.target)
    scores.append({'model':model_name,'best_score':gds.best_score_,
        'best_params':gds.best_params_})
```

Get the result in the dataframe..

```
df = pd.DataFrame(scores,columns=['model','best_score','best_params'])

df
```

Assignment..

By using digits dataset find which model with hyperparameter is the best??

1.Logistic regression classifier

2.SVM classifier

3.Random Forest classifier

4.Naive bayes' classifier(Multinomial)

5.Decision tree classifier

ENSEMBLE LEARNING

Ensemble Techniques

- A composite model combining a series of low performing classifiers for creating an improved classifier.

- It offers more accuracy than individual classifier.

- Ensemble methods can decrease variance using ***bagging*** approach and bias using ***boosting*** approach.

Bias and Variance

Reducible errors in machine learning models have two components: bias and variance.

Bias

- It is a phenomenon that skews the result of an algorithm in favor or against an idea.
- It is the amount that a model's prediction differs from the target value, compared to the training data.
- A high level of bias can lead to underfitting which means the algorithm is unable to capture relevant relations between features and target functions.
- Linear models often have high bias and nonlinear models have low bias.

CONT ...

Variance

- It refers to the changes in the model when using different portions of the training dataset.
- It measures the inconsistency of different predictions using different training sets.
- Variance can lead to overfitting.
- It comes from highly complex models with a large number of features.
- It can capture noise in the datasets.
- A non-linear algorithm will exhibit low bias but high variance since they have a lot of flexibility to fit the model.

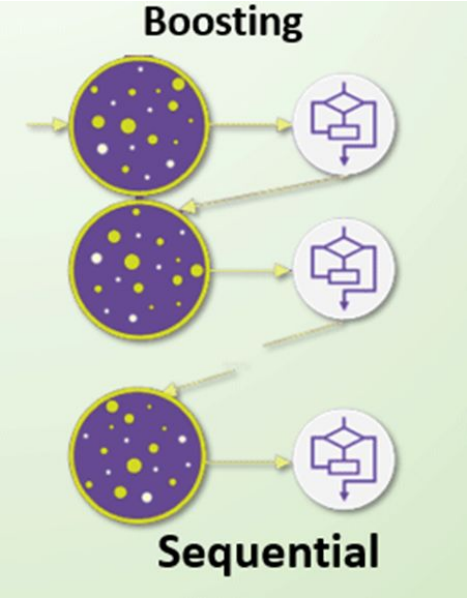
CONT...

Bias-Variance trade-off

- Bias and variance are inversely connected. Increasing bias decreases variance and decreasing bias increases variance.
- Modifying an ML algorithm to better fit a given dataset will lead to low bias but it will increase the variance. This will fit with dataset while increasing the chances of inaccurate predictions.
- Creating a low variance model with a higher bias will reduce the risk of inaccurate predictions but will not properly match the dataset.
- The correct balance between them should be found.

BOOSTING

It is an ensemble modeling technique which combines weak learners to build a strong learner. It is done by building a model by using weak models in series.



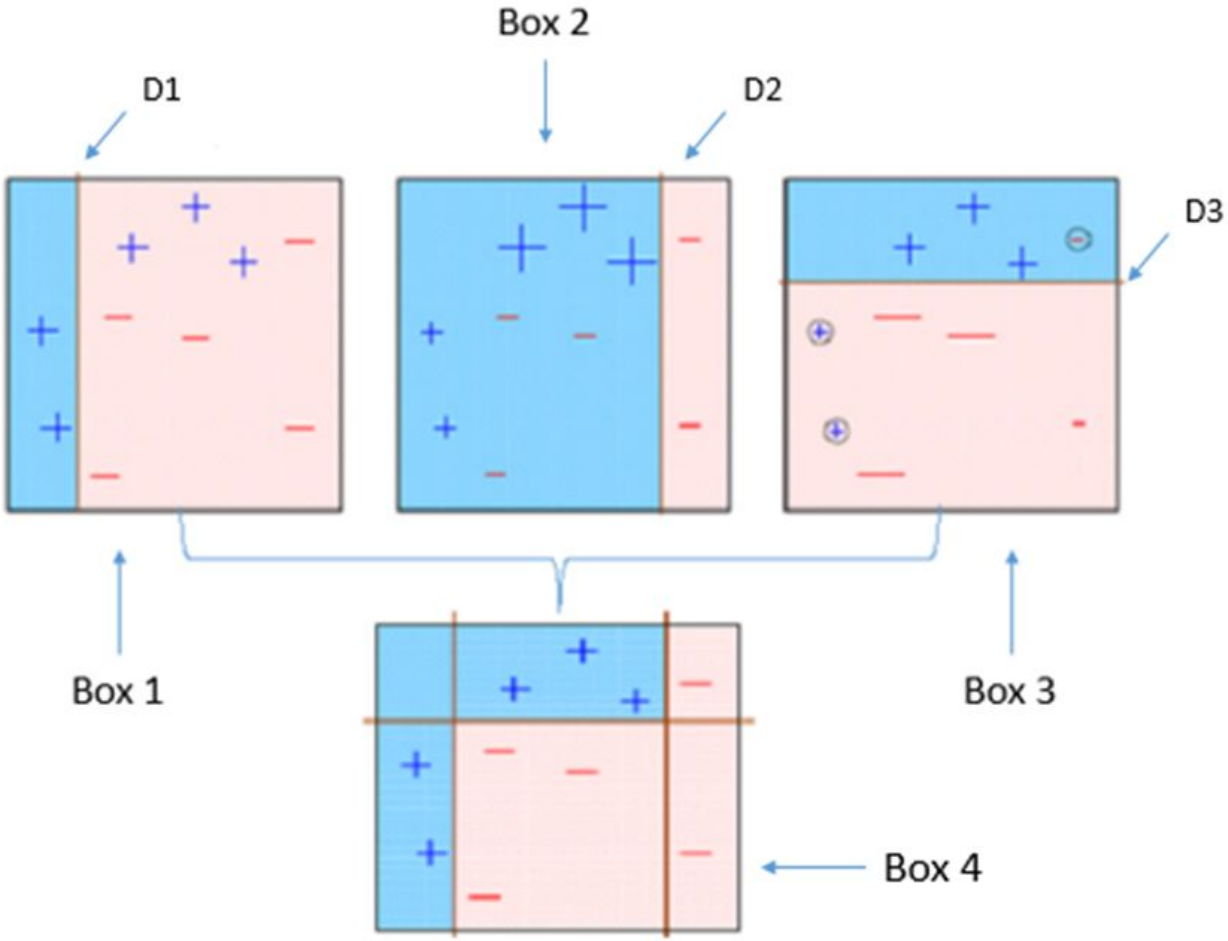
TECHOLAS
TECHNOLOGY DEMYSTIFIED

Boosting Algorithms

- AdaBoost (Adaptive Boosting)
- Gradient Boosting
- XGBoost (Extreme Gradient Boosting)

AdaBoost (Adaptive Boosting)

- It fits a sequence of weak learners on different weighted training data.
- It starts by predicting original data set and gives equal weight to each observation.
- If prediction is incorrect using the first learner, then it gives higher weight to observation which have been predicted incorrectly.
- Being an iterative process, it continues to add learner(s) until a limit is reached in the number of models or accuracy.



TECHOLAS
TECHNOLOGY DEMYSTIFIED

CONT...

Algorithm

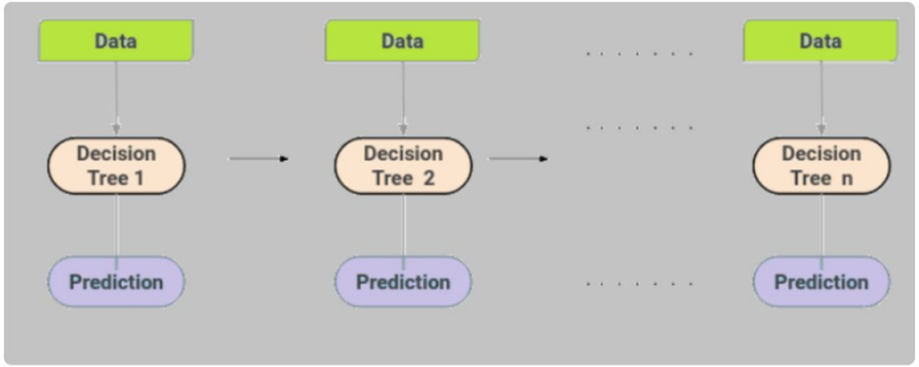
- Initialize the dataset and assign equal weight to each of the data point.
- Provide this input to the model and apply a decision stump to identify the wrongly classified data points.
- Increase the weight of the wrongly classified data points.
- If the required results are achieved, end the process; else repeat the procedure.

Gradient Boosting

- Gradient boosting is used to minimize bias error of the model.
- It has high speed and accuracy for large and complex data.
- The algorithm build models sequentially and try to reduce the errors of the previous model.
- This is done by building a new model on the errors or residuals of the previous model.
- When the target column is continuous, we use **Gradient Boosting Regressor** whereas when it is a classification problem, we use **Gradient Boosting Classifier**.

CONT...

- Here, unlike adaboost algorithm, each predictor is trained using the residual errors of predecessor as labels.
- The nodes in each decision tree take a distinct subset of the features for picking out the best split.
- Thus the decision trees aren't all identical and they are able to capture distinct signals from the data.



TECHOLAS
TECHNOLOGY DEMYSTIFIED

Assignment

XGBoost Algorithm (Classifier and Regressor)

UNSUPERVISED LEARNING

K-means algorithm..

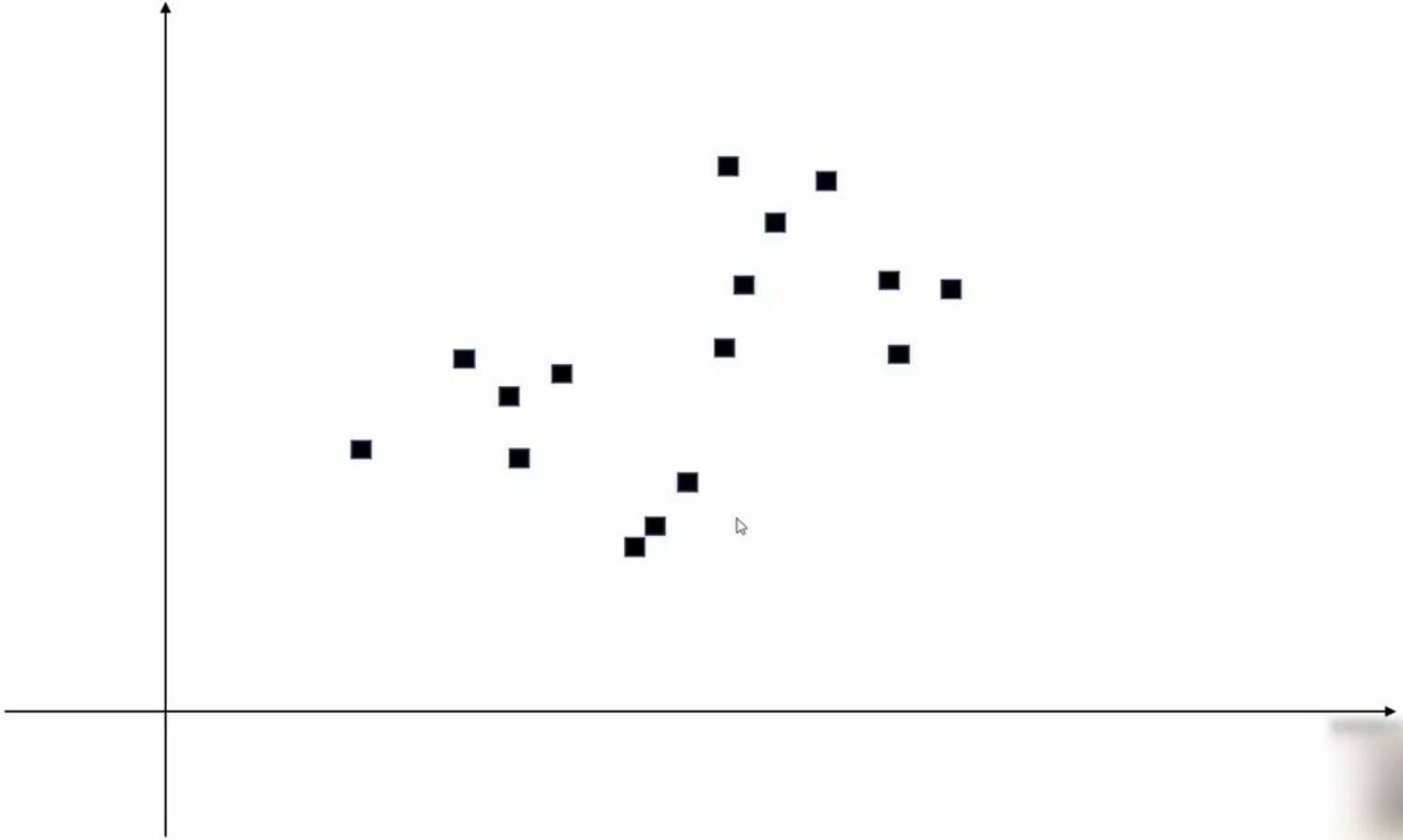
It's an unsupervised learning algorithm.. Till now we discussed all machine learning techniques that comes under supervised learning.

In supervised learning in your dataset the target value will be there.

But in unsupervised learning only some features will be there. We need to find the target variable or clusters from that.

And clusters are the grouped records having a common behaviour.

Here, you don't know actually what you are looking for. And all you are going to do is finding some structures in the dataset.

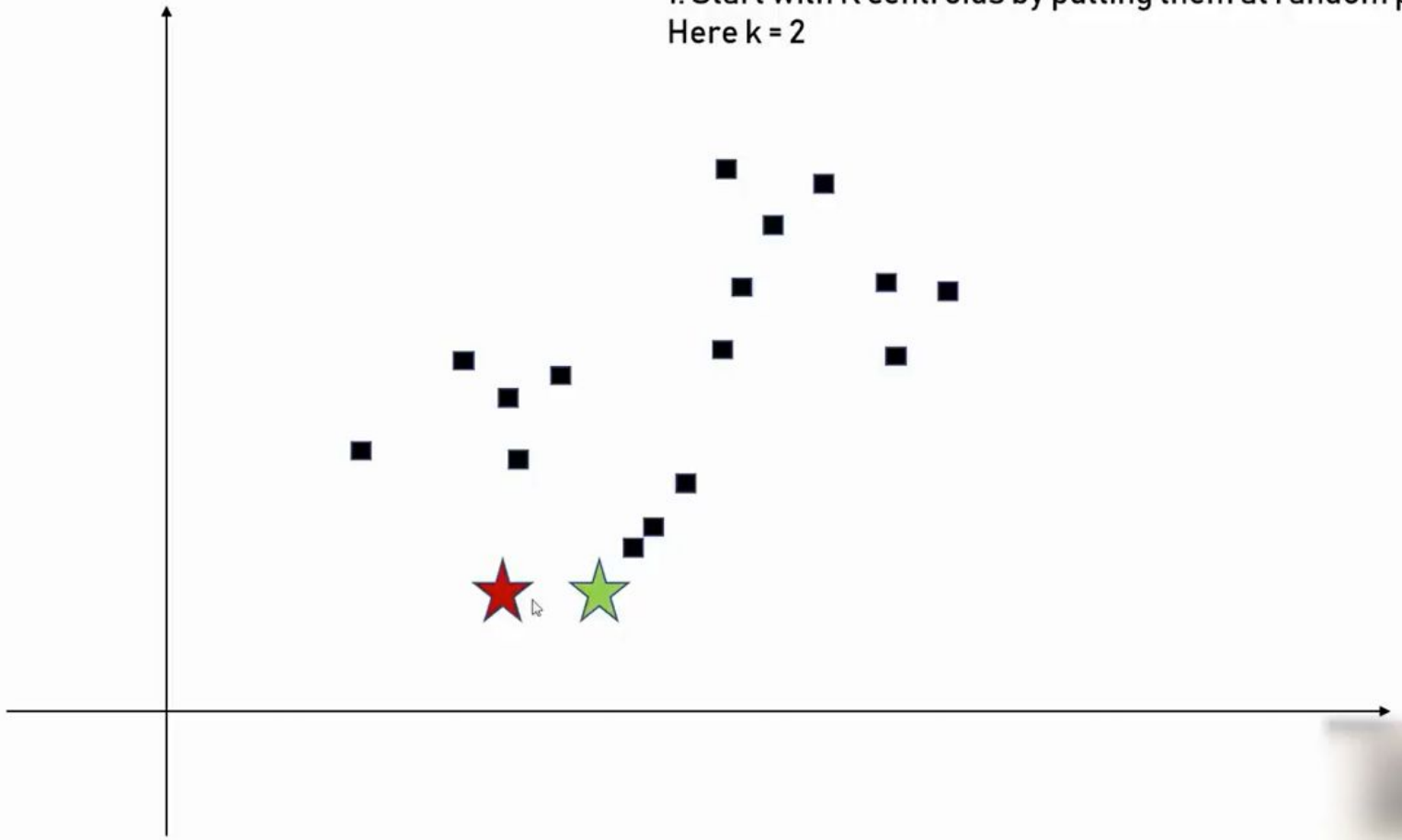


Step 1

Consider this is as our dataset. We don't know what actually we are looking for and how many clusters are there??

But we assume.And the assumption starts from 2. This value is K value.that's why the name k means algorithm

1. Start with K centroids by putting them at random place,
Here k = 2



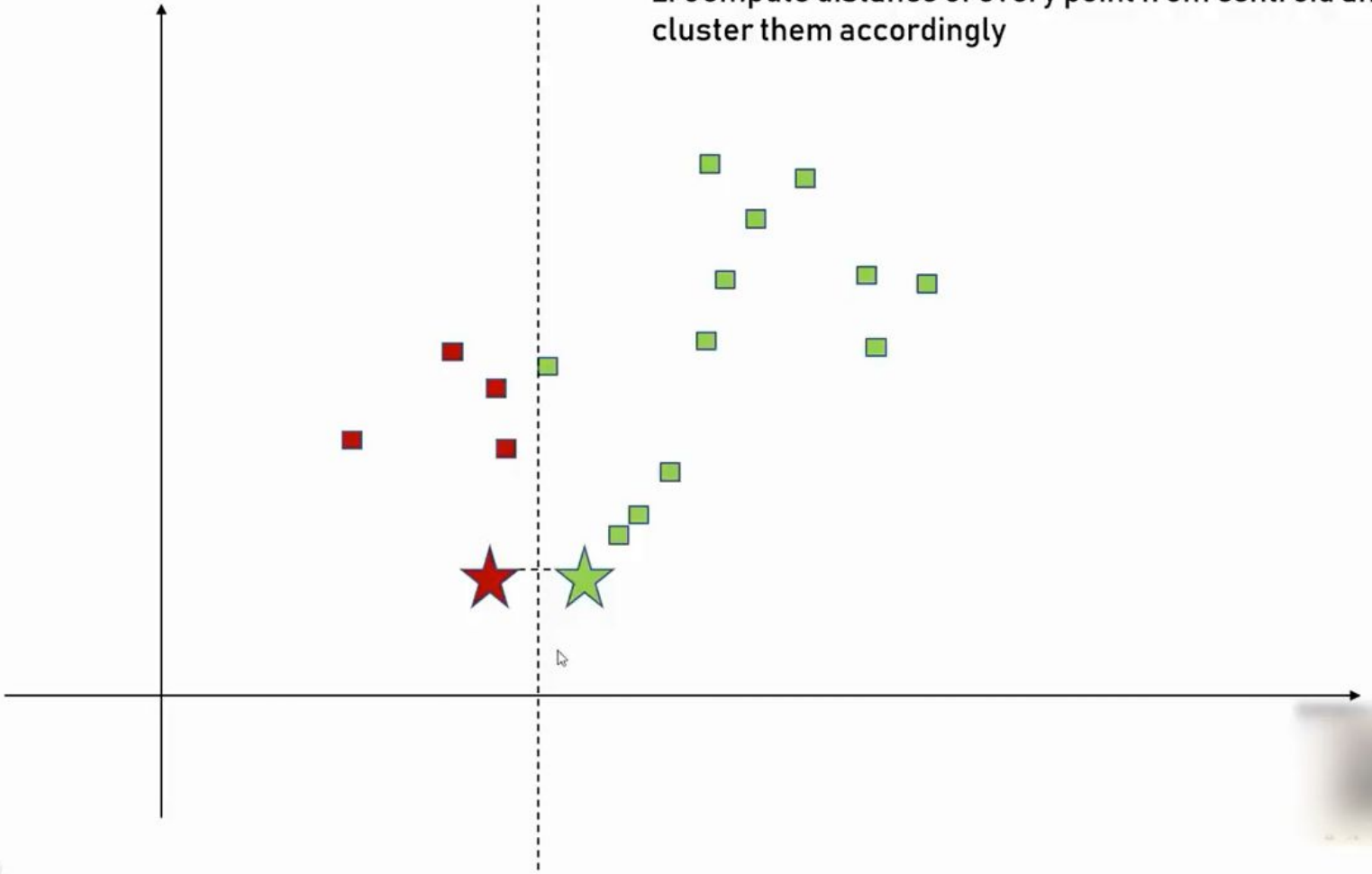
Step 2

So the next step is place 2 random points somewhere.(because we assume k=2).

And those 2 points we consider as the centre of the 2 clusters and these points known as **centroids.**

If we are assuming there are k clusters then k centroids will be there

2. Compute distance of every point from centroid and cluster them accordingly

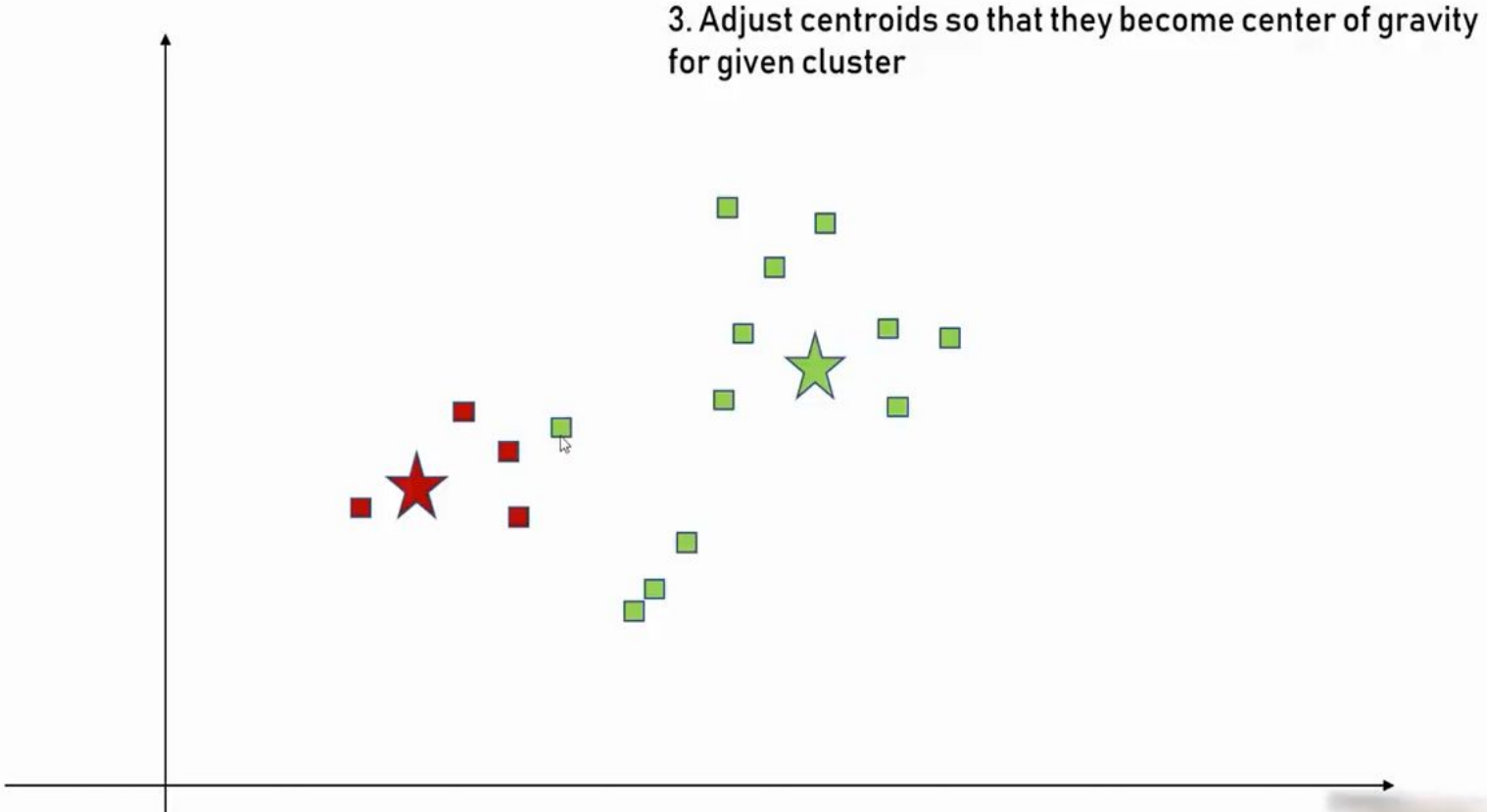


Step 3

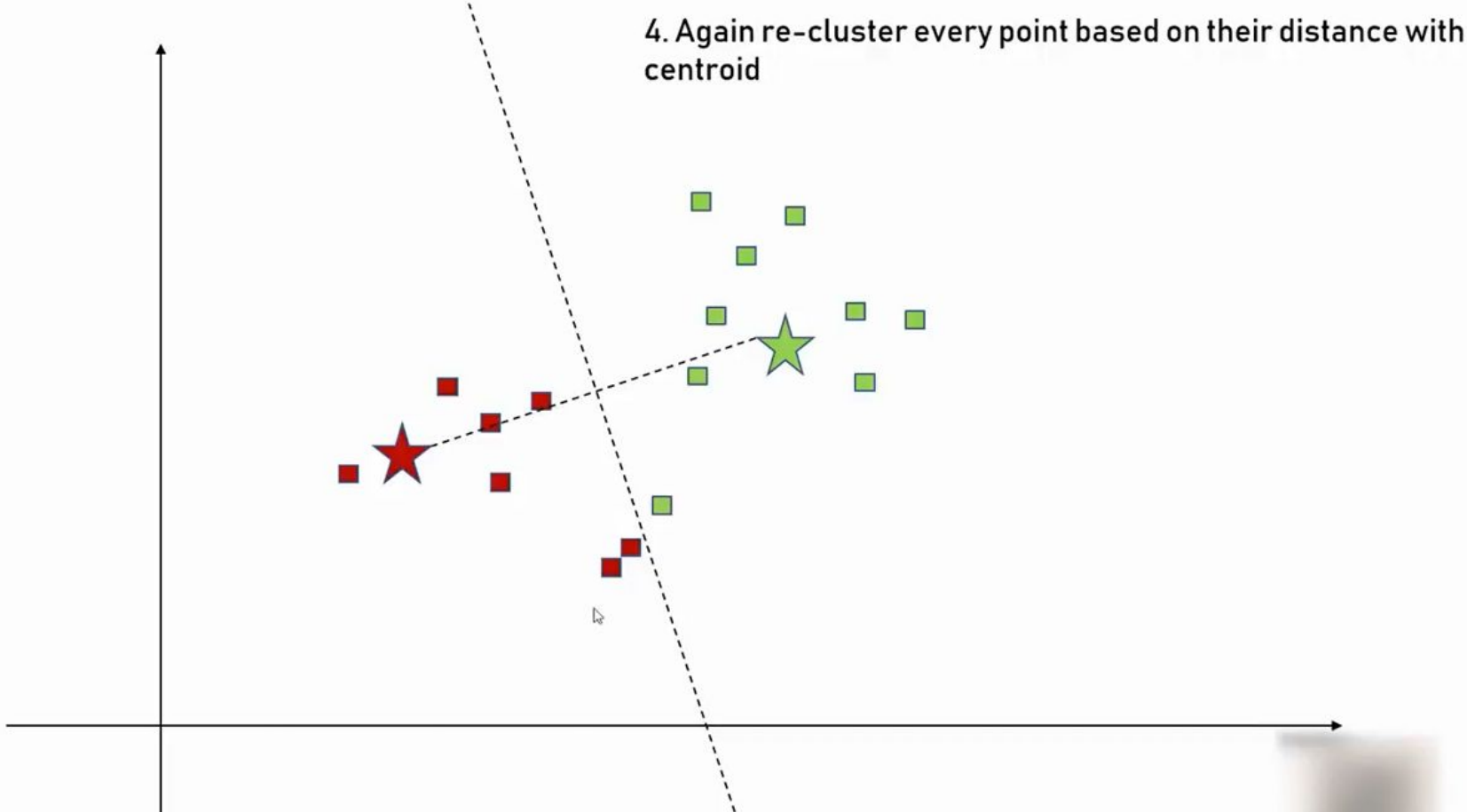
Compute distance to every point from centroid and create an imperfect cluster,

Like those points near to the centroid belongs to one cluster.

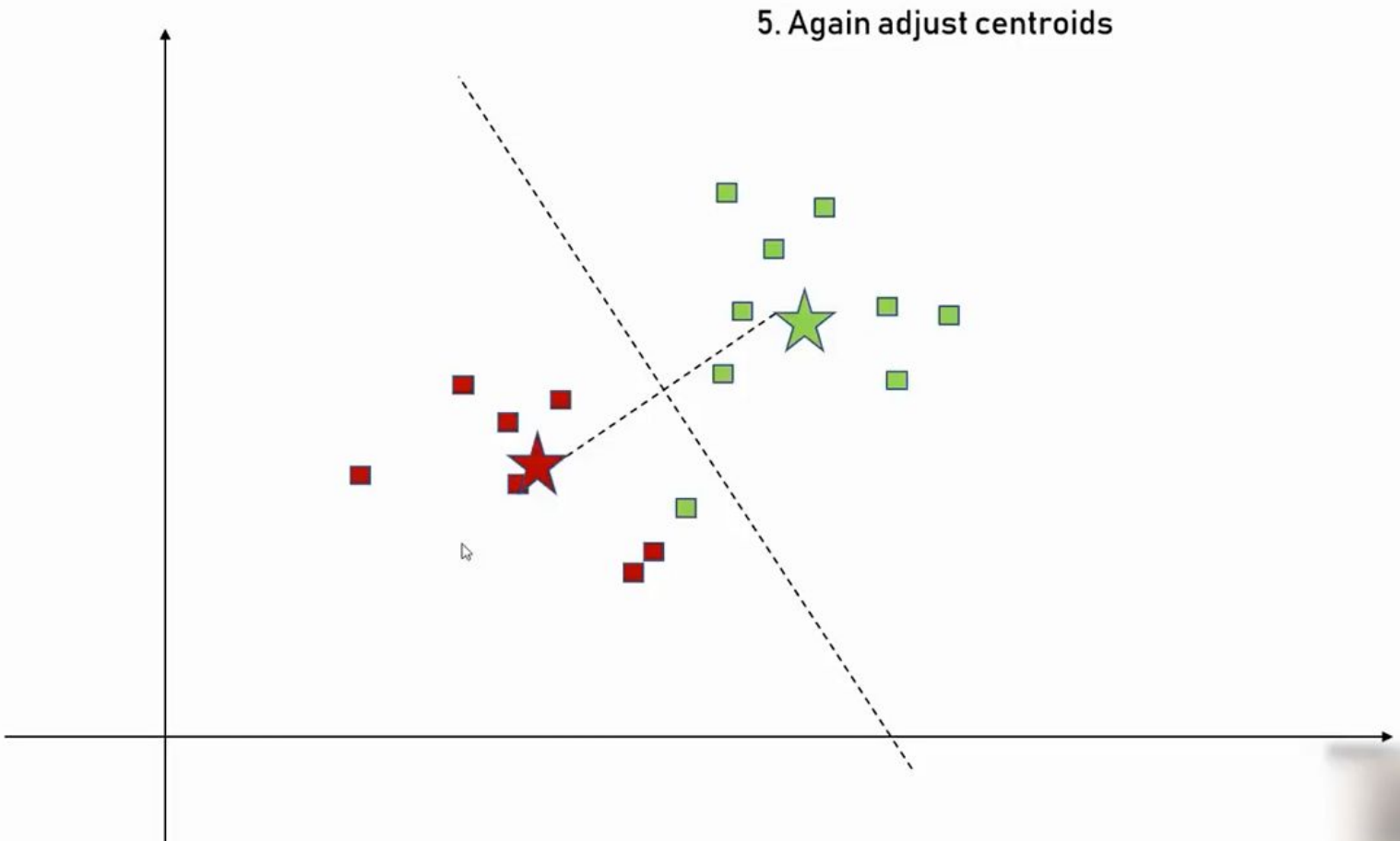
Step 4


3. Adjust centroids so that they become center of gravity for given cluster

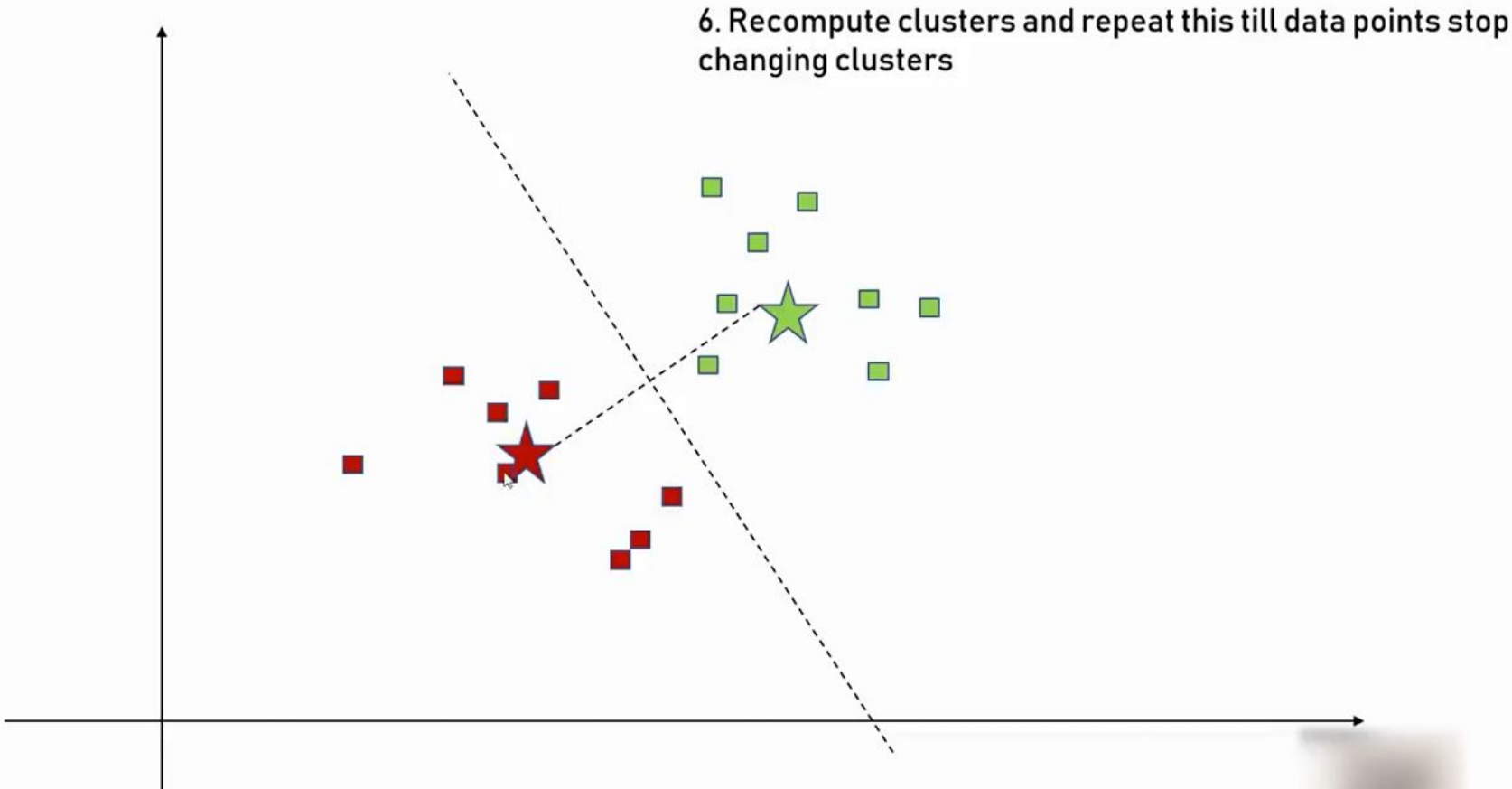
Step 5


4. Again re-cluster every point based on their distance with centroid

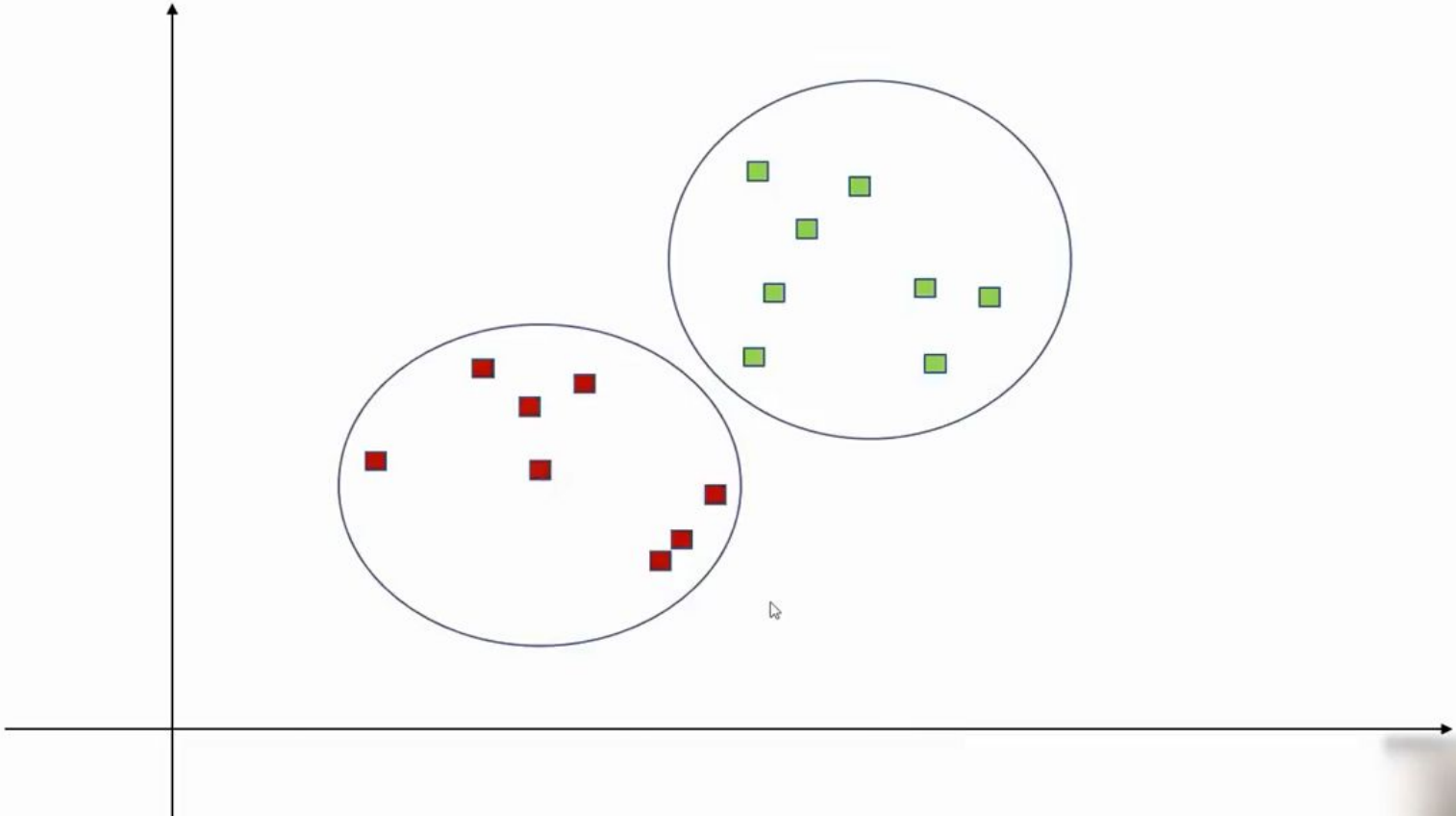
Step 6



Step 7

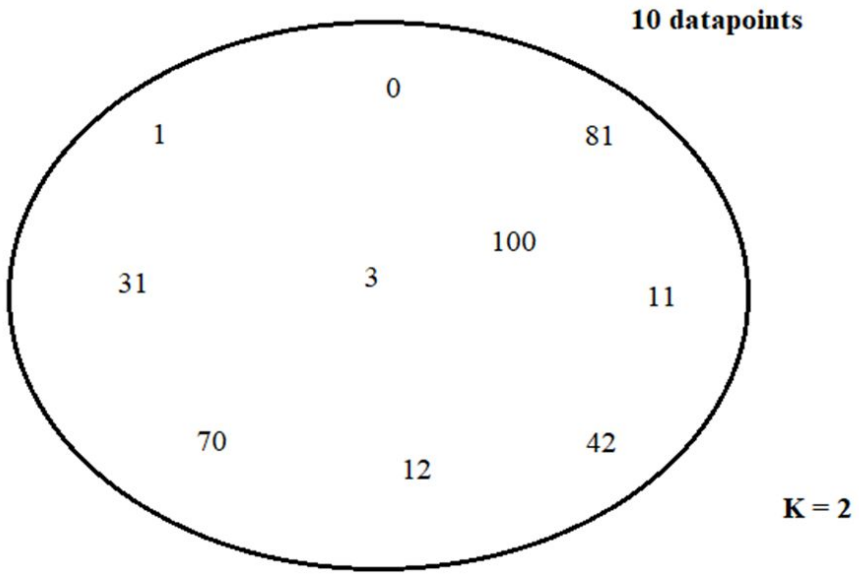

6. Recompute clusters and repeat this till data points stop changing clusters

Finally the clusters..



Example

Clustering example with K=2



TECHOLAS
TECHNOLOGY DEMYSTIFIED

Data points: 0, 1, 3, 11, 12, 31, 42, 70, 81, 100

Iteration 1

Cluster centers : 1, 3

Difference between data points and each centroid:

|0-1| = 1
|1-1| = 0
|3-1| = 2
|11-1| = 10
|12-1| = 11
|31-1| = 30
|42-1| = 41
|70-1| = 69
|81-1| = 80
|100-1| = 99

|0-3| = 3
|1-3| = 2
|3-3| = 0
|11-3| = 8
|12-3| = 9
|31-3| = 28
|42-3| =39
|70-3| = 67
|81-3| = 78
|100-3| = 97

CONT...

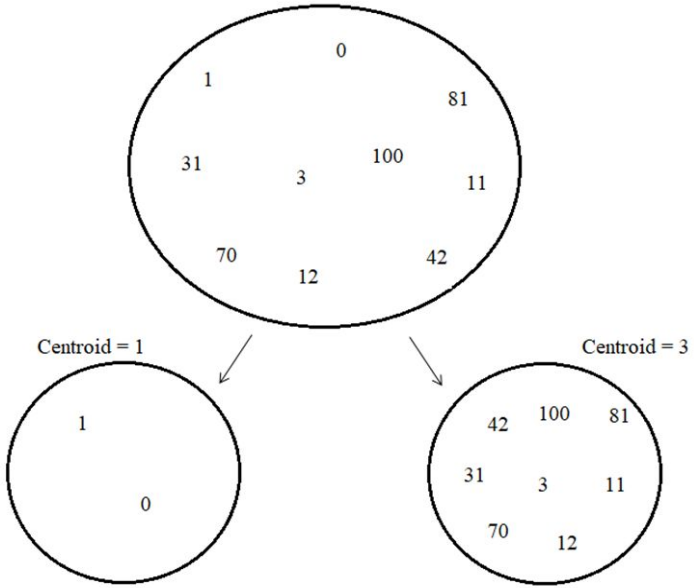
Clustered data:

Cluster 1

1, 0

Cluster 2

12, 3, 81, 100, 31, 11, 70, 42



TECHOLAS
TECHNOLOGY DEMYSTIFIED

CONT...

Centroid updation

New centroid of cluster 1 = (1 + 0)/2 = 0.5

New centroid of cluster 2 = (42+100+31+3+11+70+12+81)/8

= 43.75

TECHOLAS
TECHNOLOGY DEMYSTIFIED

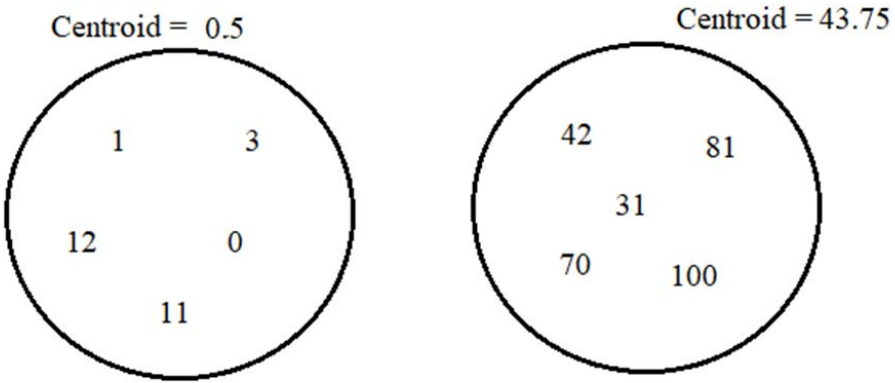
CONT...

Iteration 2

Cluster centers : 0.5, 43.75

Clustered data:

Cluster 1	Cluster 2
0, 1, 3, 11, 12	31, 42, 70, 81, 100



TECHOLAS
TECHNOLOGY DEMYSTIFIED

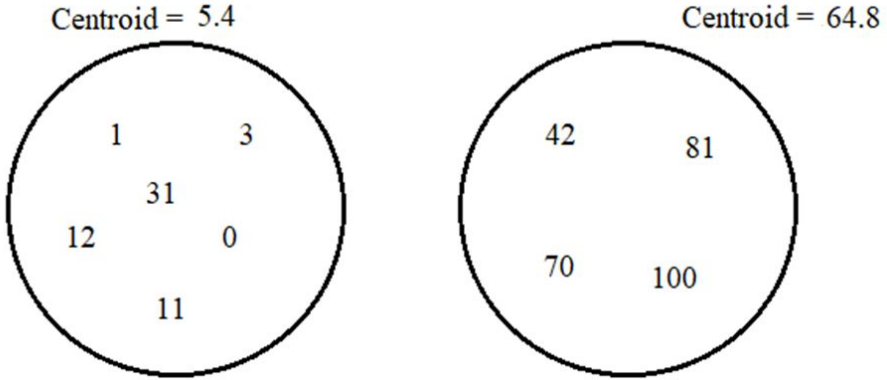
CONT...

Iteration 3

Cluster centers : 5.4, 64.8

Clustered data:

Cluster 1	Cluster 2
0, 1, 3, 11, 12, 31	42, 70, 81, 100



TECHOLAS
TECHNOLOGY DEMYSTIFIED

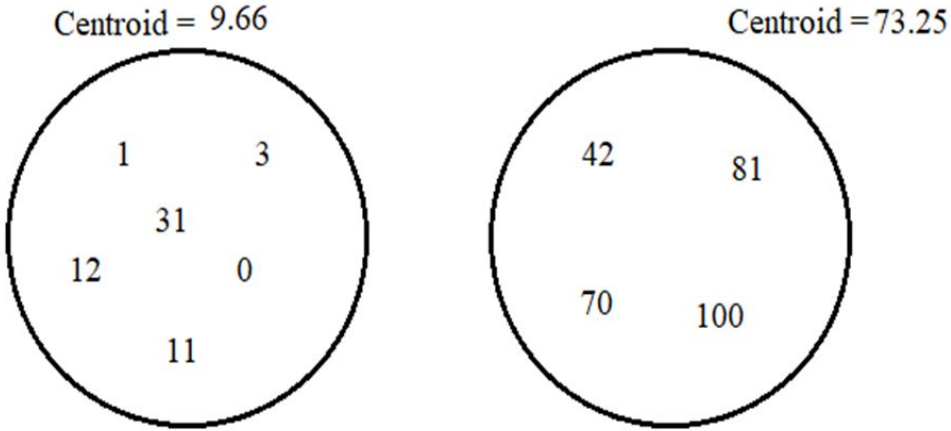
CONT...

Iteration 4

Cluster centers : 9.66, 73.25

Clustered data:

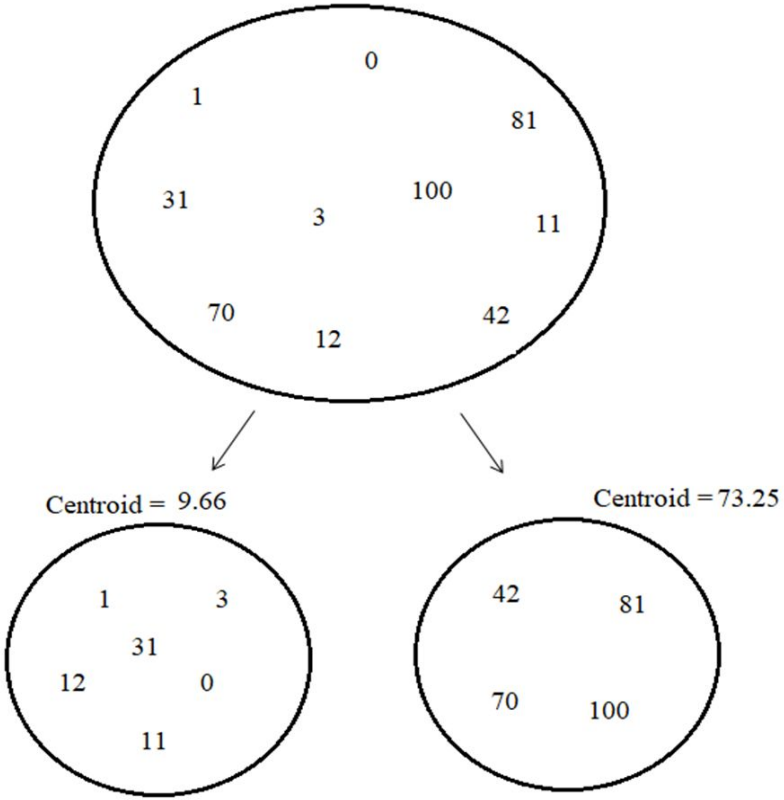
Cluster 1	Cluster 2
0, 1, 3, 11, 12, 31	42, 70, 81, 100



TECHOLAS
TECHNOLOGY DEMYSTIFIED

CONT...

Clustered result

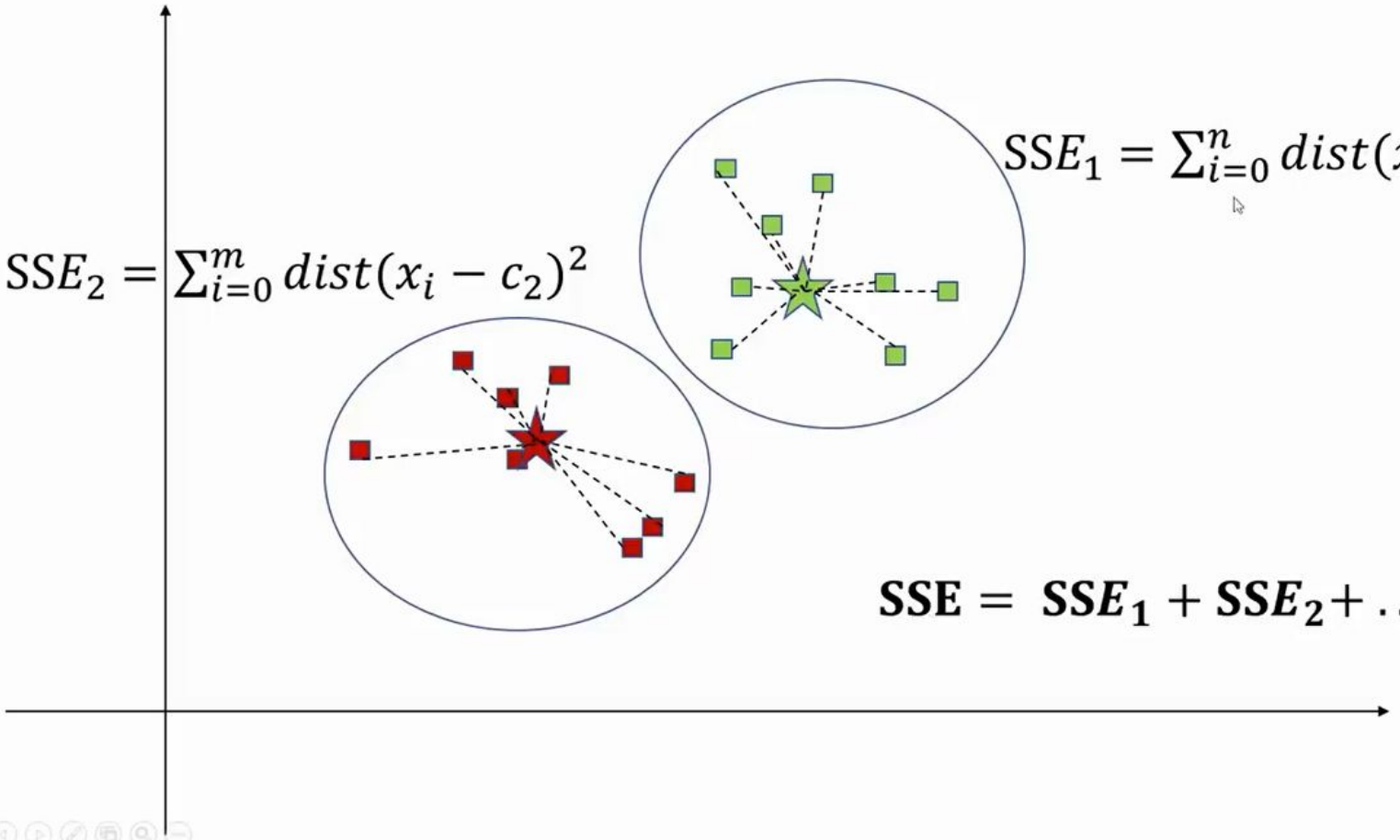


TECHOLAS
TECHNOLOGY DEMYSTIFIED

But how to find the correct number of clusters?

The solution is elbow method..

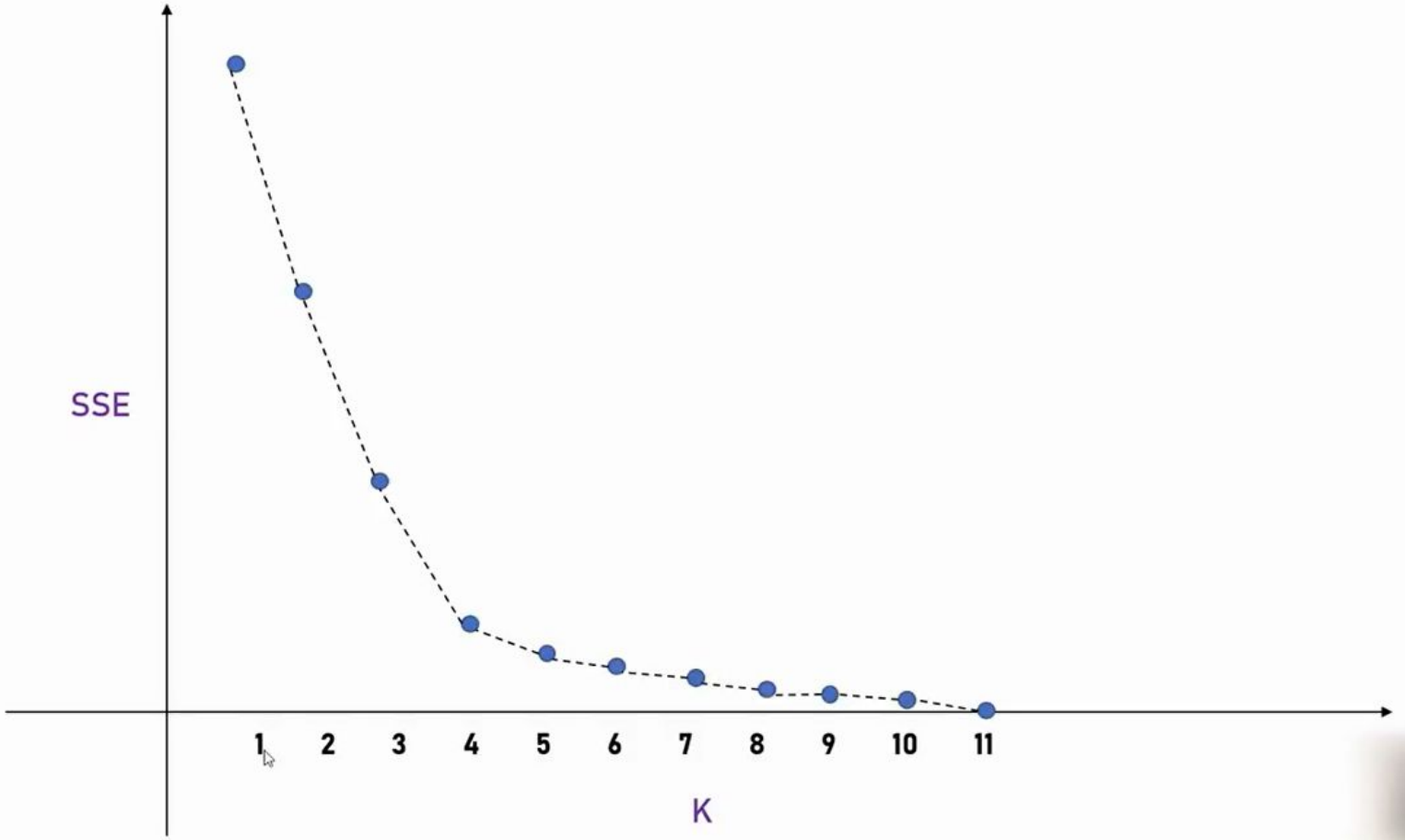
SSE = Sum of Squared Errors



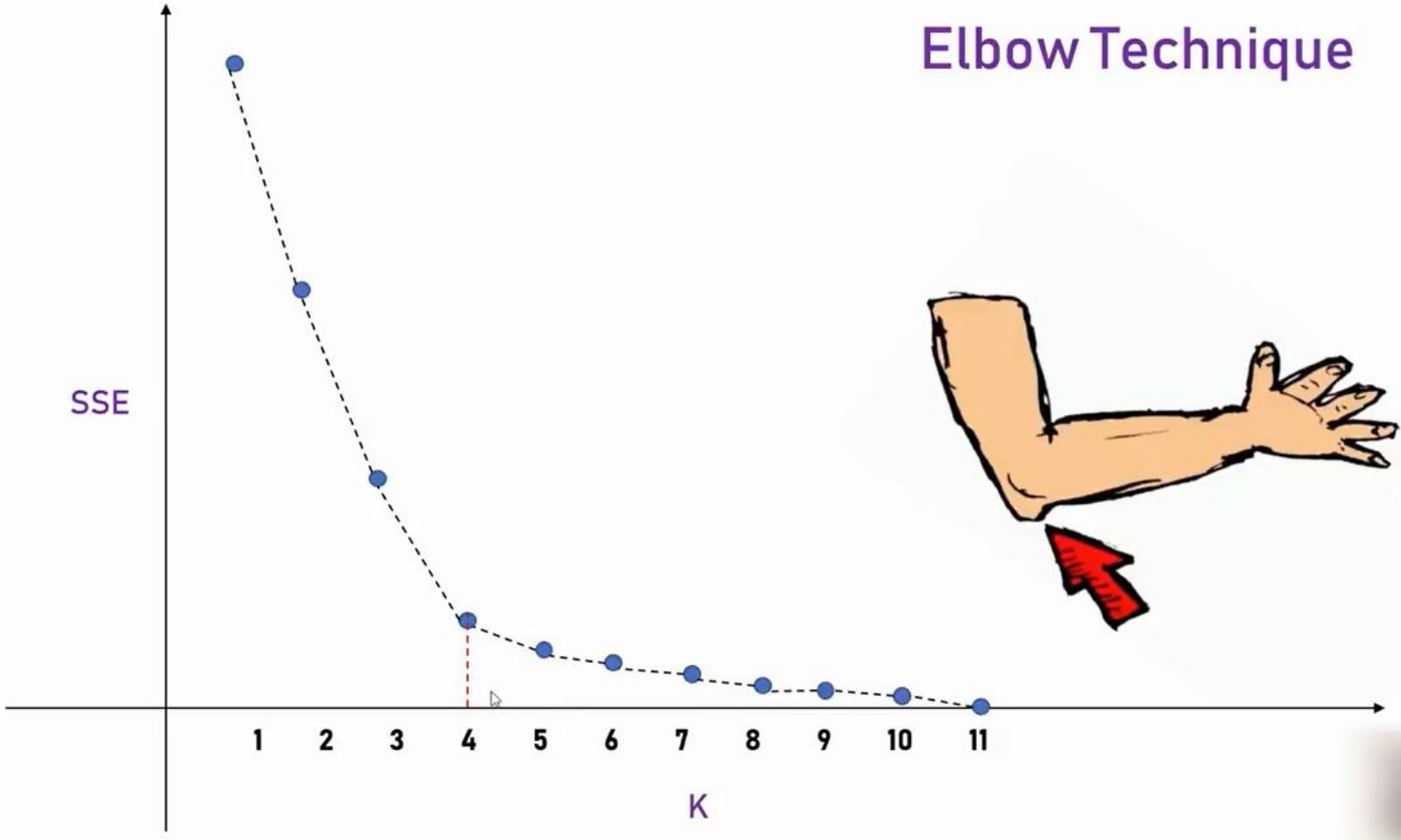
$$SSE_1 = \sum_{i=0}^{n} dist(x_i - c_1)^2$$

$$SSE_2 = \sum_{i=0}^{m} dist(x_i - c_2)^2$$

$$\mathbf{SSE} = \mathbf{SSE_1} + \mathbf{SSE_2} + .. + \mathbf{SSE_k}$$


SSE
1
2
3
4
5
6
7
8
9
10
11
K

Elbow Technique



So in notebook..

We are going to analyze salary of employees.

And we are going to find the clusters belongs to that.

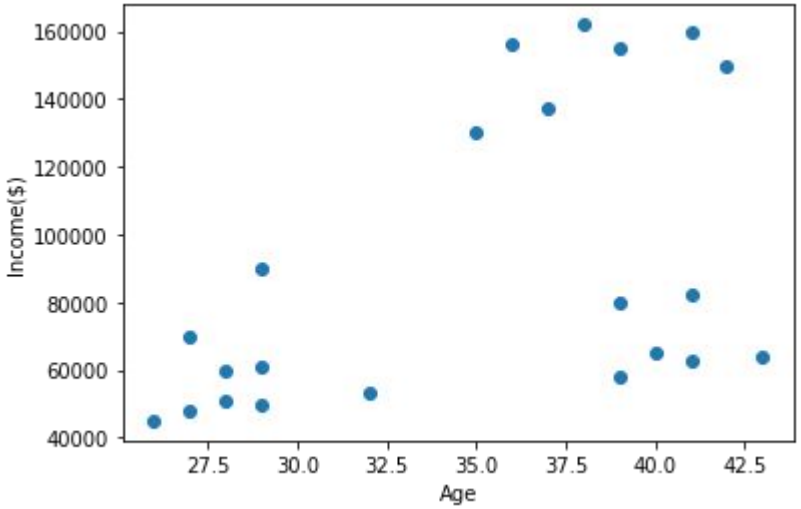
ie the common behaviour in that data,may be the region which the person resides may have higher salaries.. Something like that.

Import and load data..

```
from sklearn.cluster import KMeans

import pandas as pd

from matplotlib import pyplot as plt


df = pd.read_csv("income.csv")

df.head()
```

Plot to assume k..?

```
plt.scatter(df.Age,df['Income($)'])

plt.xlabel('Age')

plt.ylabel('Income($)')
```



As you can see there are 3 clusters,

So we assume the value of k=3

TECHOLAS
TECHNOLOGY DEMYSTIFIED

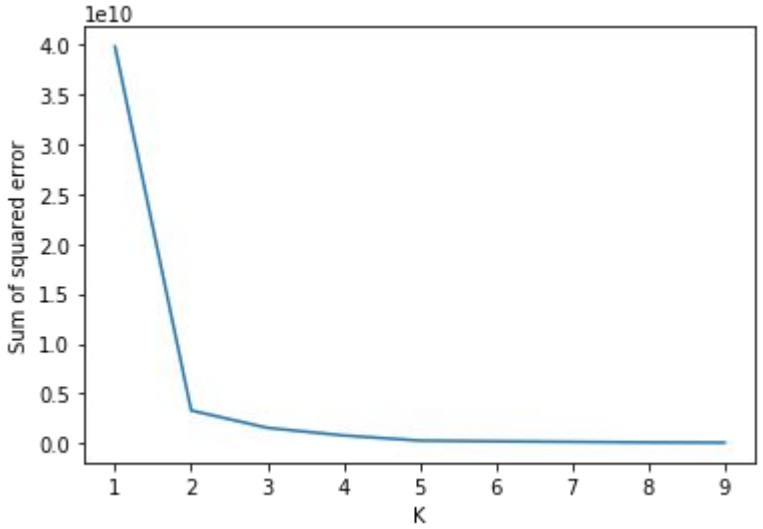
Let's verify with elbow plot..

- sse= []
- k_range=range(1,len(df))
- for k in k_range:
- km=KMeans(n_clusters=k)
- km.fit(df[['Age','Income($)']])
- sse.append(km.inertia_)

TECHOLAS
TECHNOLOGY DEMYSTIFIED

Plot it..

```
plt.xlabel('K')

plt.ylabel('Sum of squared error')

plt.plot(k_range,sse)
```



TECHOLAS
TECHNOLOGY DEMYSTIFIED

Lets create the model..

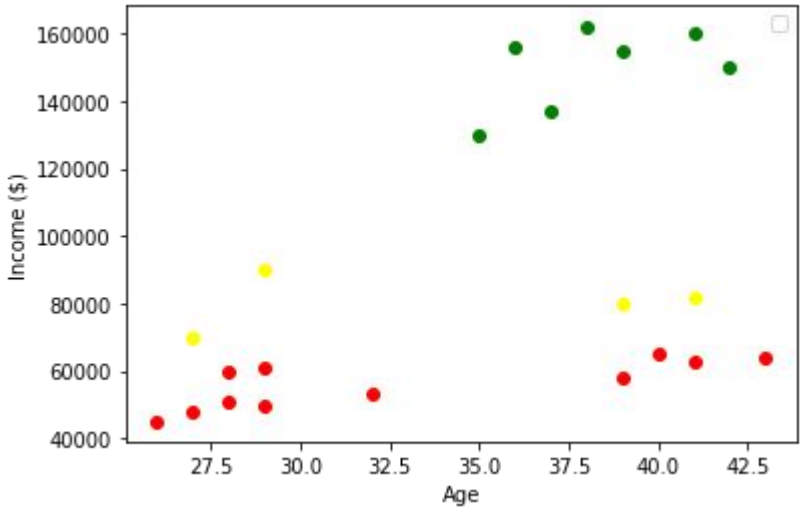
- km=KMeans(n_clusters=3)
- y_predicted=km.fit_predict(df[['Age','Income($)']])
- df['cluster']=y_predicted
- df.head()
- km.cluster_centers_

TECHOLAS
TECHNOLOGY DEMYSTIFIED

Let's plot it ..and check

- df1=df[df['cluster']==0]
- df2=df[df['cluster']==1]
- df3=df[df['cluster']==2]
- plt.scatter(df1['Age'],df1['Income($)'],color='Red')
- plt.scatter(df2['Age'],df2['Income($)'],color='Green')
- plt.scatter(df3['Age'],df3['Income($)'],color='yellow')
- plt.legend()
- plt.xlabel('Age')
- plt.ylabel('Income ($)')

Whats wrong..



The problem is because the range of x axis and y axis

So we should scale the range between 0 and 1.

TECHOLAS
TECHNOLOGY DEMYSTIFIED

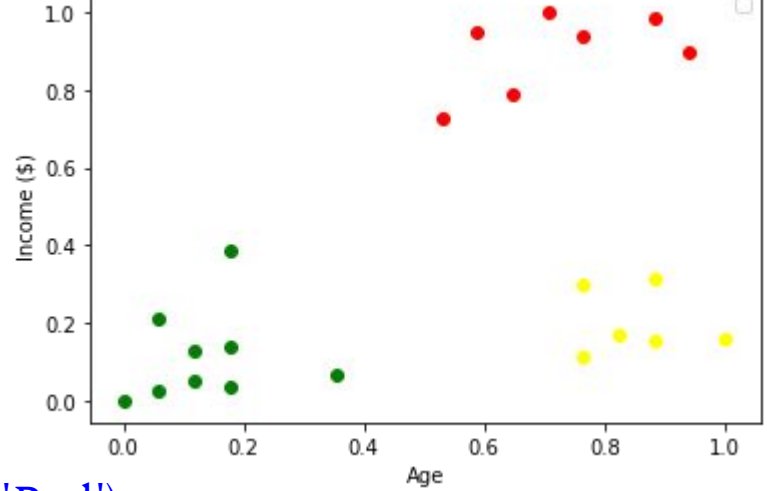
Processing using min max scaler..

```
from sklearn.preprocessing import MinMaxScaler

sca=MinMaxScaler()

sca.fit(df[['Income($)']])

df['Income($)']=sca.transform(df[['Income($)']])

sca.fit(df[['Age']])

df['Age']=sca.transform(df[['Age']])
```

Now do the same..

```
km=KMeans(n_clusters=3)

y_predicted=km.fit_predict(df[['Age','Income($)']])

df['cluster']=y_predicted

df.head()

km.cluster_centers_
```

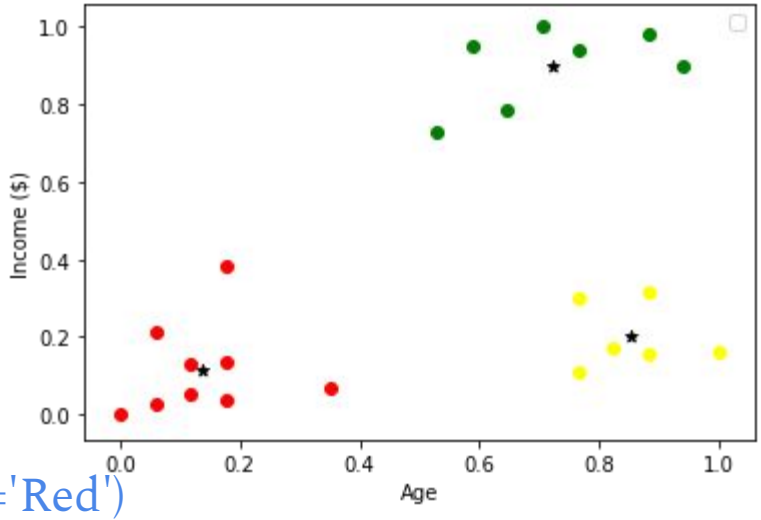
Plot it..



- df1=df[df['cluster']==0]
- df2=df[df['cluster']==1]
- df3=df[df['cluster']==2]
- plt.scatter(df1['Age'],df1['Income($)'],color='Red')
- plt.scatter(df2['Age'],df2['Income($)'],color='Green')
- plt.scatter(df3['Age'],df3['Income($)'],color='yellow')
- plt.legend()
- plt.xlabel('Age')
- plt.ylabel('Income ($)')

TECHOLAS
TECHNOLOGY DEMYSTIFIED

Plot the centroid too...



- df1=df[df['cluster']==0]
- df2=df[df['cluster']==1]
- df3=df[df['cluster']==2]
- plt.scatter(df1['Age'],df1['Income($)'],color='Red')
- plt.scatter(df2['Age'],df2['Income($)'],color='Green')
- plt.scatter(df3['Age'],df3['Income($)'],color='yellow')
- plt.scatter(centroids[:,0],centroids[:,1],color='Black',marker='*')
- plt.legend()
- plt.xlabel('Age')
- plt.ylabel('Income ($)')

TECHOLAS
TECHNOLOGY DEMYSTIFIED

Assignment..

1. Use iris flower dataset from sklearn library and try to form clusters of flowers using petal width and length features. Drop other two features for simplicity.
2. Figure out if any preprocessing such as scaling would help here
3. Draw elbow plot and from that figure out optimal value of k

Hierarchical clustering

- Unsupervised algorithm
- Avoids the need to decide on the number of clusters as in the case of K Means clustering
- It is a distance based algorithm as in the case of K Means clustering.

Types of hierarchical clustering

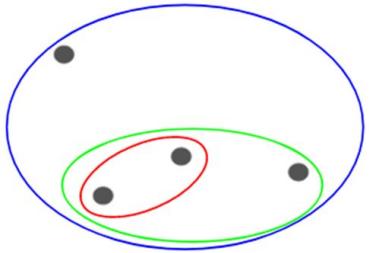
- Agglomerative Hierarchical clustering : good at identifying small clusters
- Divisive Hierarchical clustering : good at identifying large clusters

Agglomerative hierarchical clustering

(Also called additive hierarchical clustering)

Assign each data point to a separate cluster

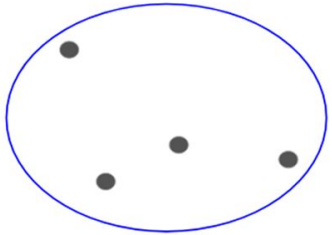
Merge the closest pair of clusters at each iteration and repeat this until a single cluster is left



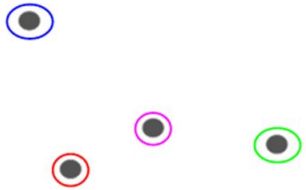
TECHOLAS
TECHNOLOGY DEMYSTIFIED

Divisive hierarchical clustering

It works the opposite way. It starts with a single cluster with all the points in it.



The farthest point in the cluster are split in each iteration until each cluster contains only a single point in it.



TECHOLAS
TECHNOLOGY DEMYSTIFIED

Example

Clustering of students based on their marks.

There is a proximity matrix which stores the distances between the clusters during each iteration

Student_ID	Marks
1	10
2	7
3	28
4	20
5	35

ID	1	2	3	4	5
1	0	3	18	10	25
2	3	0	21	13	28
3	18	21	0	8	7
4	10	13	8	0	15
5	25	28	7	15	0

Proximity matrix

CONT...

Assign points to individual clusters

Look for the smallest distance in the proximity matrix and merge the points with the smallest distance.

ID	1	2	3	4	5
1	0	3	18	10	25
2	3	0	21	13	28
3	18	21	0	8	7
4	10	13	8	0	15
5	25	28	7	15	0

TECHOLAS
TECHNOLOGY DEMYSTIFIED

CONT...

Update the proximity matrix. The marks can be filled using the min, max or average of the values

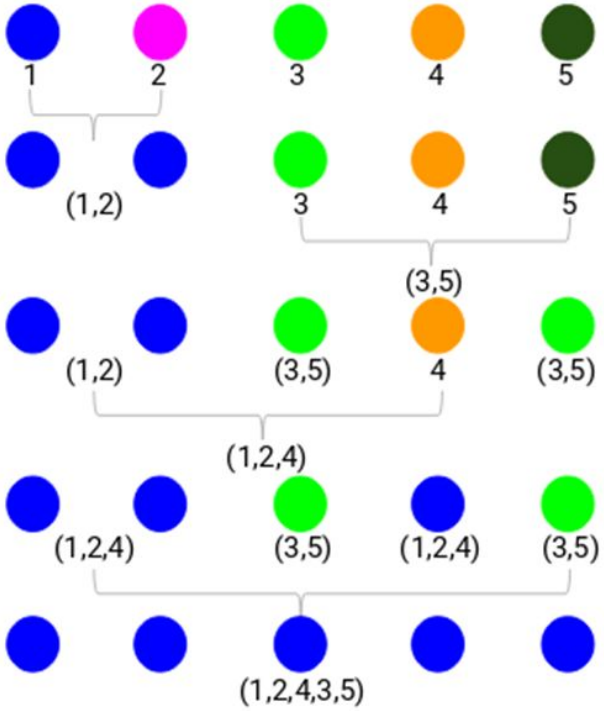
Student_ID	Marks
(1,2)	10
3	28
4	20
5	35

ID	(1,2)	3	4	5
(1,2)	0	18	10	25
3	18	0	8	7
4	10	8	0	15
5	25	7	15	0

Repeat the previous step until only a single cluster is left

CONT...

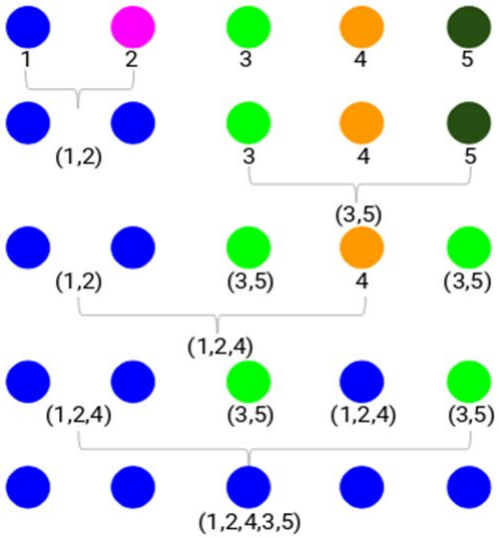
Merged clusters

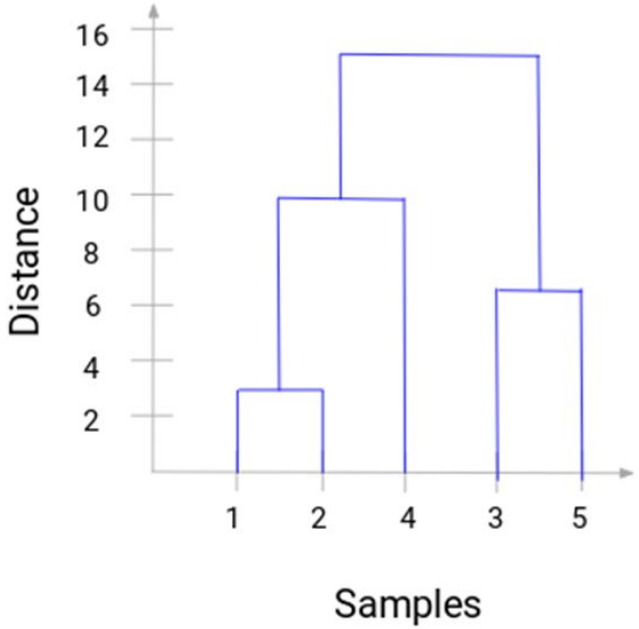


TECHOLAS
TECHNOLOGY DEMYSTIFIED

Choosing number of clusters

We use a dendrogram for this. It is a tree-like diagram that records the sequences of merges or splits.





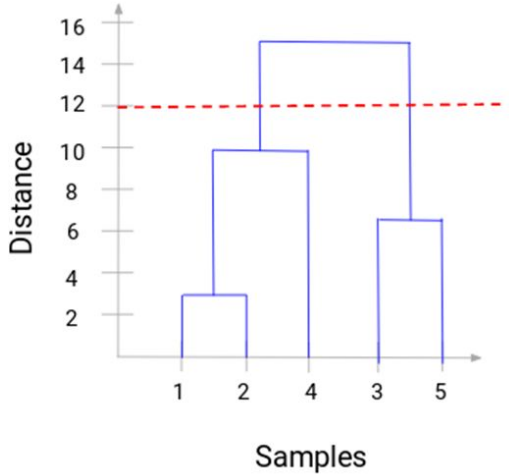
TECHOLAS
TECHNOLOGY DEMYSTIFIED

CONT...

More the distance of the vertical lines in the dendrogram, more the distance between those clusters.

Set a threshold distance and draw a horizontal line, generally cutting the vertical lines with maximum distance of separation.

The number of vertical lines being intersected by this line gives the number of clusters.



TECHOLAS
TECHNOLOGY DEMYSTIFIED