Hands-on Examples
Artificial Intelligence with Real-time Application Projects

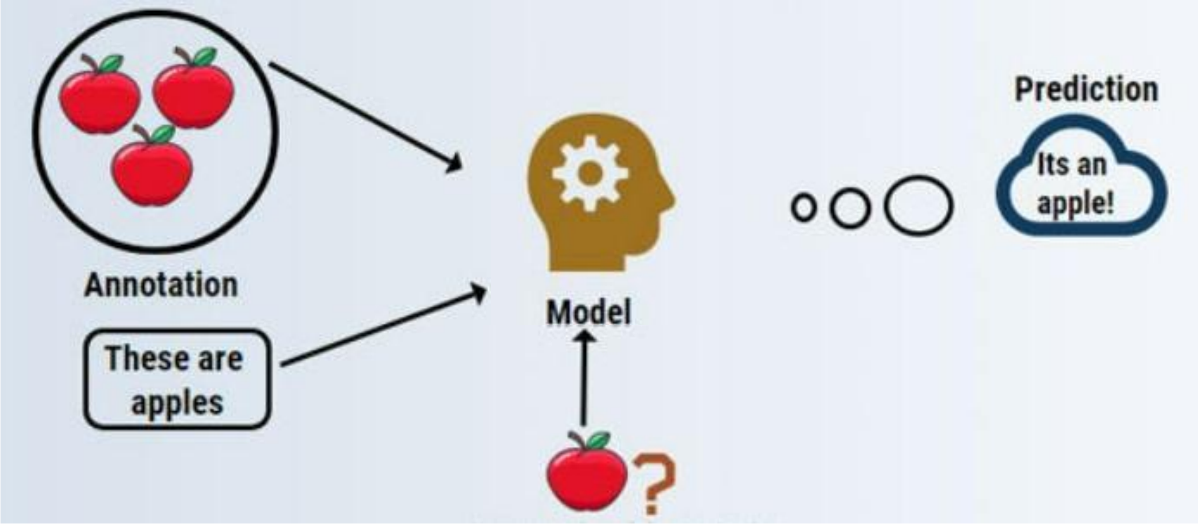
Types of Machine Learning

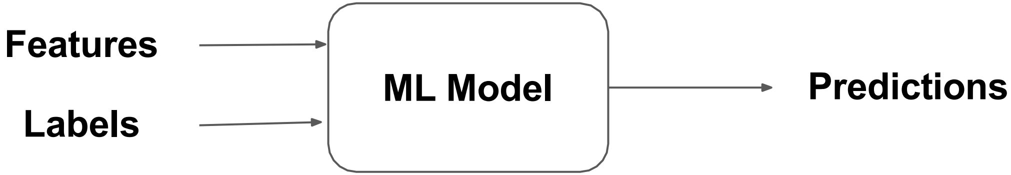
1. Supervised Machine Learning

2. Unsupervised Machine Learning

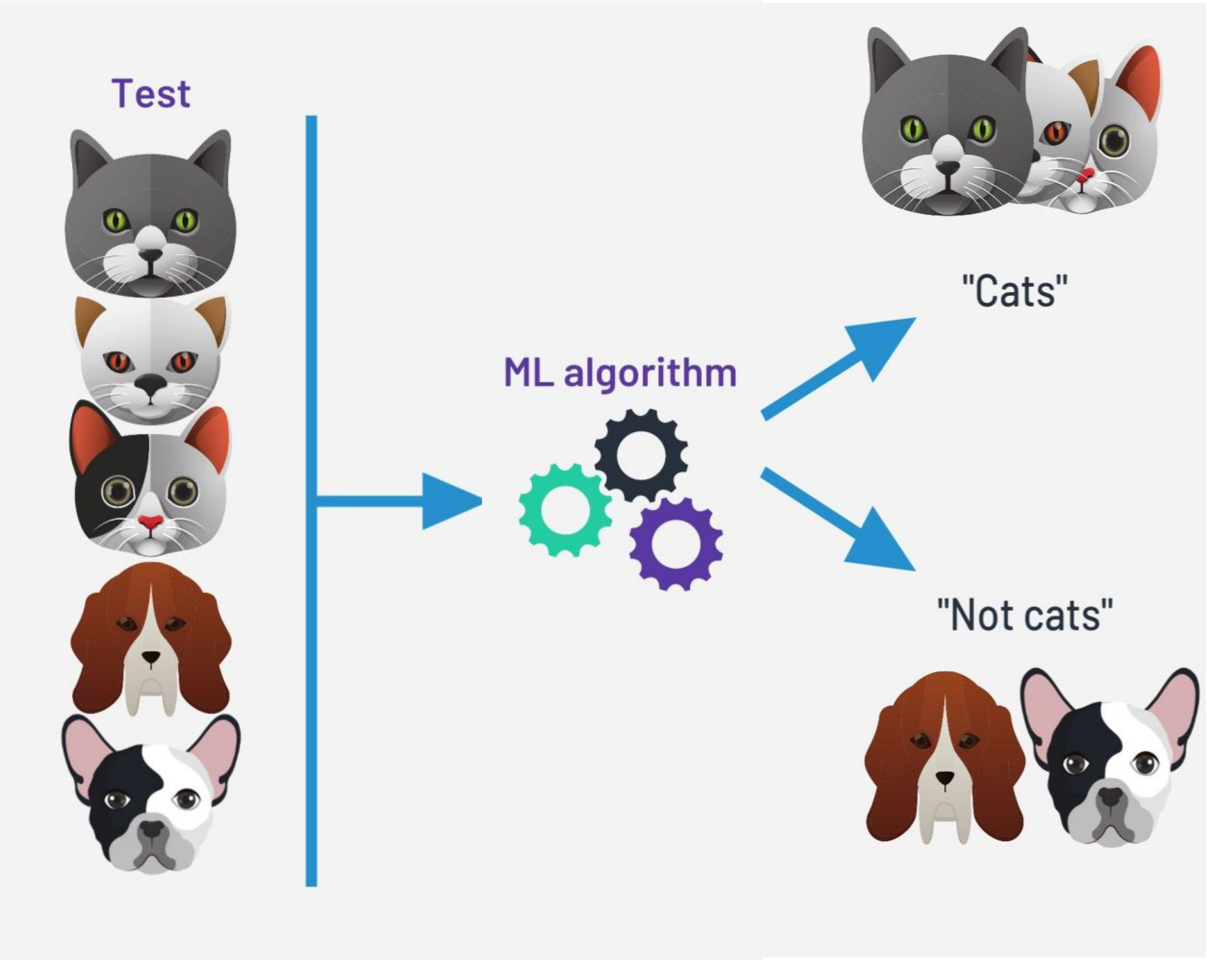
3. Reinforcement Machine Learning

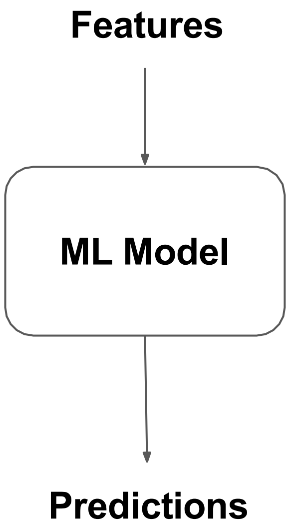
1. Supervised Machine Learning

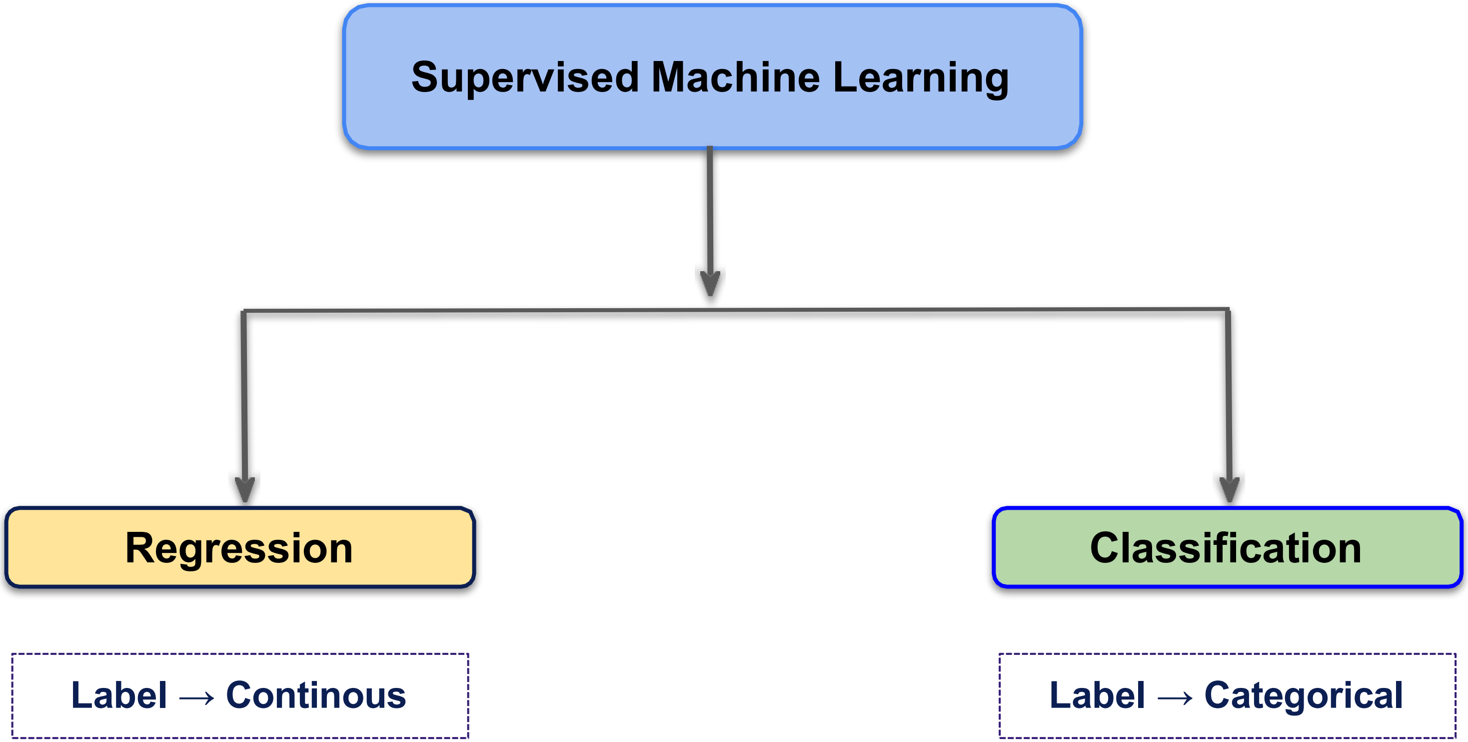




2. Unsupervised Machine Learning






Supervised Machine Learning
Regression
Classification
Label → Continous
Label → Categorical

Supervised Machine Learning

Regression Models

- 1. Linear Regression
- 2. Decision Tree Regressor
- 2. Support Vector Regressor

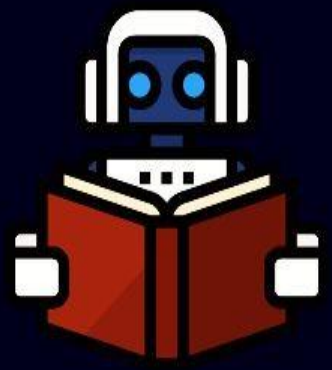
Classification Models

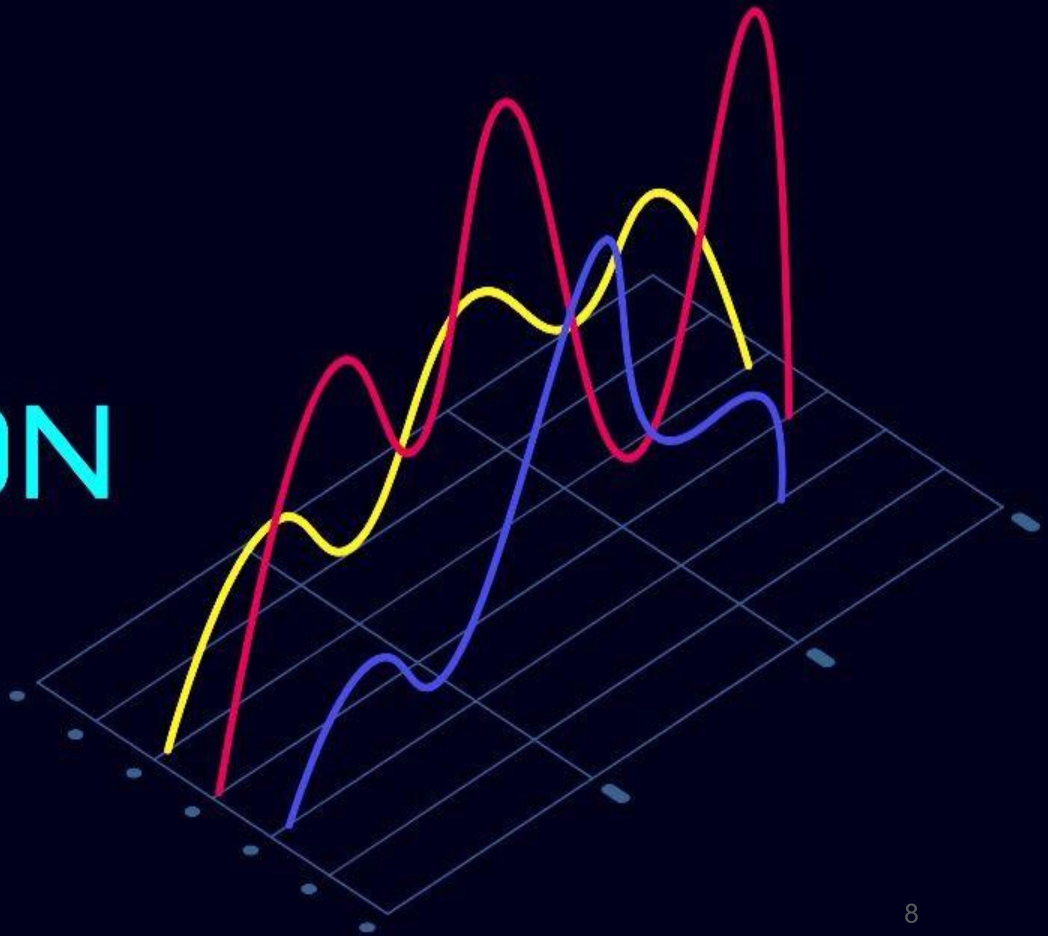
- 1. Logistic Regression
- 2. Support Vector Classifier
- 3. Naive-Bayes
- 4. Random Forest / Decision Tree
- 5. K Nearest Neighbour (KNN)

Unsupervised Machine Learning

- 1. K-Means Clustering
- 2. DBSCAN Clustering

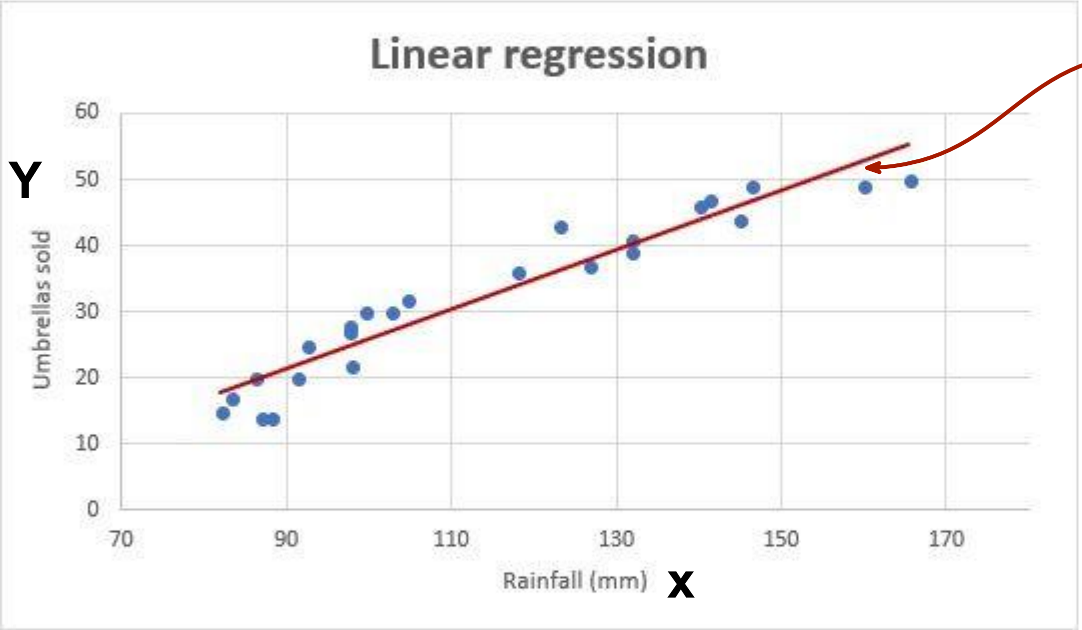
LINEAR REGRESSION





Linear Regression

- Linear regression is a regression model which tries to predict the relationship between the dependent variable Y and independent variable X in a linear fashion.



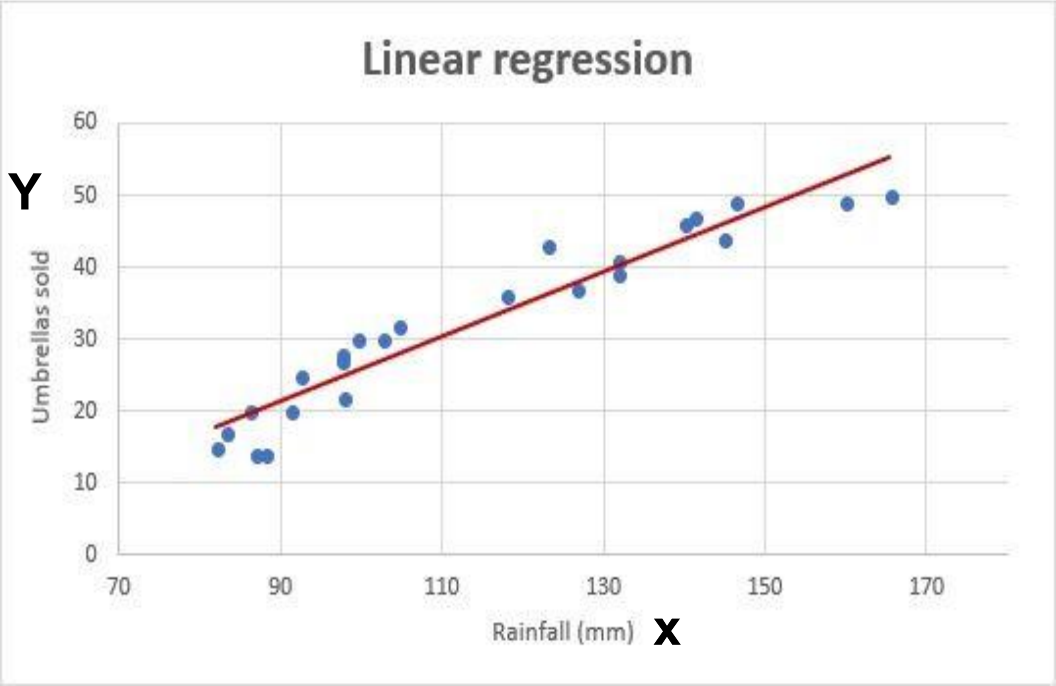
$$Y = mx + c$$

m - slope

c - intercept

Cost Function:

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



Linear Regression Types

1. Simple Linear Regression

- Single Feature (X) ,
- Single Label (Y)

Ad Expenses in $ (x)

Sales in $ (Y)

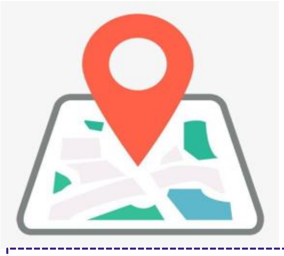
$$Y = mx + c$$

Ad_exp	Sales
24	724
28	756
32	782
39	831
44	853
45	860
54	896
58	914
62	924
65	938
68	947
76	971

2. Multi Linear Regression

- Multiple Features (X1, X2, X3….Xn) , Single Label (Y)

No. of Rooms(X1)



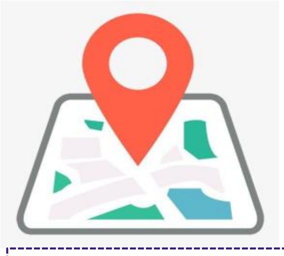
Location (X2)

House Age(X3)

Distance from Graveyard (X4)

House Price (Y)

No. of Rooms(X1)



Location (X2)

House Age(X3)

Distance from Graveyard (X4)

House Price (Y)

$$Y = m1 * X1 + m2 * X2 + m3 * X3 + m4 * X4 + c$$

m1, m2, m3 & m4 → Weights

X1, X2, X3 & X4 → Features

Practical Implementation of Linear Regression:

Project 1: Experience based Salary Prediction

Employee Exp.

Salary Prediction

Feature → Numeric

Labels → Continuous

Machine Learning Model Building Steps:

1. Load Data Set
2. Check for null value & perform Data Cleaning
3. Check Features Correlation
4. Convert Categorical Data into Numerical
5. Extract Features (X) & Labels (Y) from Data Set
6. Split the data into train & test. Then do Feature Scaling
7. Train the Model using suitable ML Models
8. Check accuracy & perform Validation

Practical Implementation of Linear Regression:

1. Load Data Set

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

data=pd.read_csv("Salary.csv")
print(data.columns)
print(data.shape)
data.head()
```

3. Check for null value & perform Data Cleaning

```
data.isnull().sum()
```

Visualize the Feature & Label

```
sns.set_theme()
sns.scatterplot(data = data, x = data['Salary'], y = data['YearsExperience'])
```

5. Extract Features (X) & Labels (Y) from Data Set

```
x = data.iloc[:,:-1]
y = data.iloc[:,-1]
```

6. Split the data into train & test. Then do Feature Scaling

```
from sklearn.model_selection import train_test_split
```

```
xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size = 0.2, random_state = 42)
```

7. Train the Model using suitable ML Models

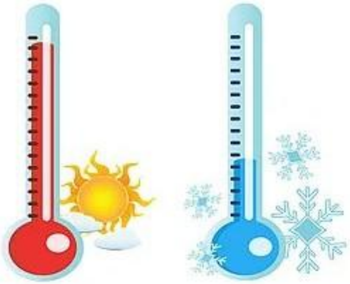
```
from sklearn.linear_model import LinearRegression

L=LinearRegression()
L.fit(xtrain,ytrain)
```

```
y_pred=L.predict(xtest)
print(y_pred)
print(L.score(xtest, ytest))
```

```
print(y_pred)
print(ytest)
```

Assignment Project 1: Ice Cream Sales Prediction

Temperature



Feature → Numeric

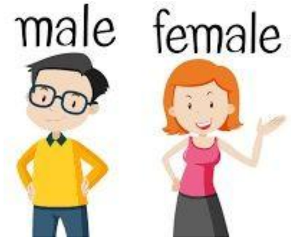
Revenue

Labels → Numeric

Practical Implementation of Linear Regression:

Project 2: Insurance Expense Prediction using Linear Regression

Age (X1)



sex (X2)

BMI (X3)

Children (X4)

Smoker (X5)

Region (X6)

Expanses(Y)

Practical Implementation of Linear Regression:

1. Load Data Set

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

data = pd.read_excel("insurance.xlsx")
data.head()
```

2. Check Features Correlation

```
print(data.shape)
data.info()
fig, ax = plt.subplots(figsize=(8,8))

corr = data.corr()
sns.heatmap(corr , annot = True , ax=ax)
```

3. Check for null value & perform Data Cleaning

```
data.isnull().sum()
```

4. Convert Categorical Data into Numerical

```
data.info()
```

```
print(data["sex"].value_counts())
print(data["smoker"].value_counts())
print(data["region"].value_counts())
```

```
from sklearn.preprocessing import LabelEncoder
Le = LabelEncoder()
```

```
data["sex"] = Le.fit_transform(data["sex"])
data["smoker"] = Le.fit_transform(data["smoker"])
data["region"] = Le.fit_transform(data["region"])
```

```
data.info()
```

5. Extract Features (X) & Labels (Y) from Data Set

```
X = data.iloc[:,:-1].values
Y = data.iloc[:,-1].values
print(X.shape , Y.shape)
```

6. Split the data into train & test. Then do Feature Scaling

```
from sklearn.model_selection import train_test_split

Xtrain , Xtest , Ytrain , Ytest = train_test_split(X , Y , test_size = 0.2 , random_state = 4)
print(Xtrain.shape , Xtest.shape , Ytrain.shape , Ytest.shape)

from sklearn.preprocessing import StandardScaler

Scaler = StandardScaler()
Xtrain = Scaler.fit_transform(Xtrain)
Xtest = Scaler.transform(Xtest)
```

7. Train the Model using suitable ML Models

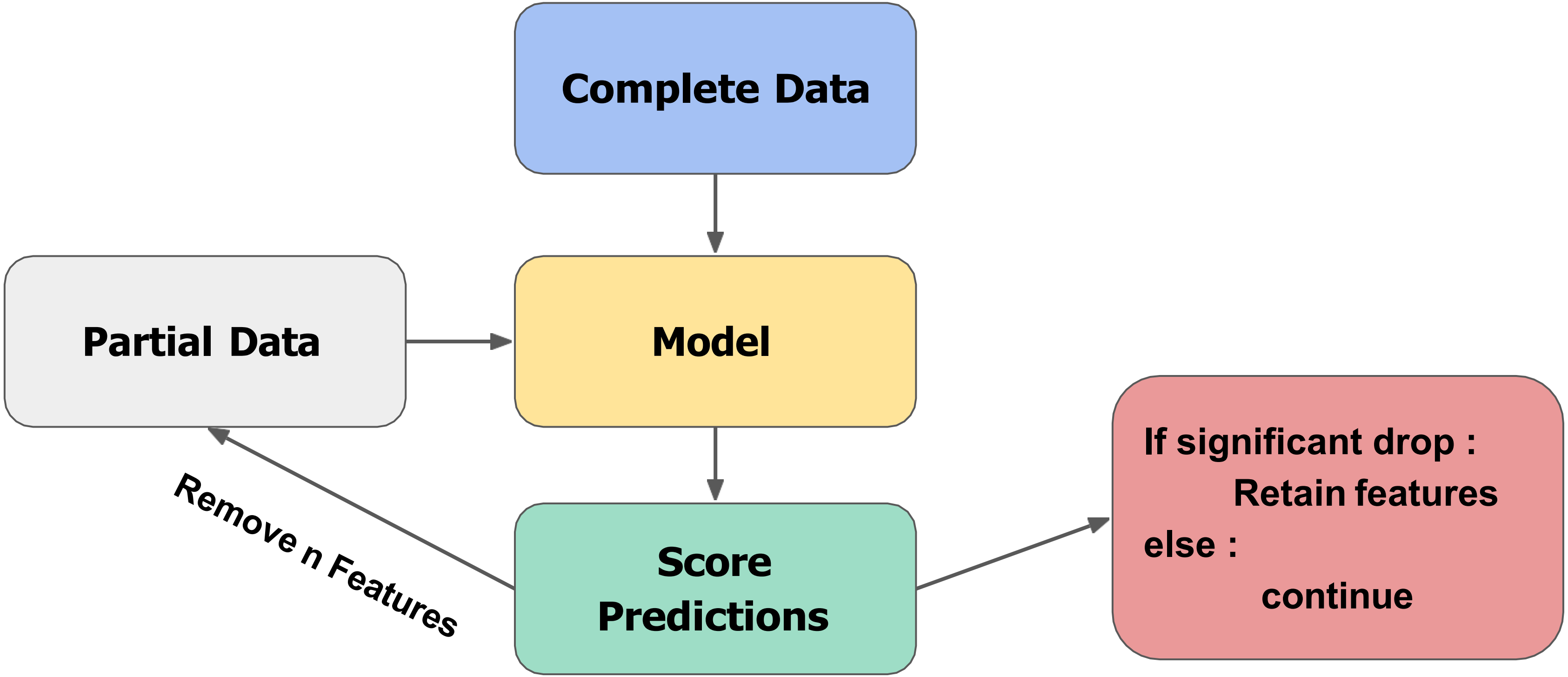
```python
from sklearn.linear_model import LinearRegression

model = LinearRegression()
model.fit(Xtrain , Ytrain)
print(model.coef_)
print(model.intercept_)
# Y = W.X + c
model.coef_.dot(Xtest[10,:]) + model.intercept_
```

8. Check Accuracy Score of ML Model

```python
model.score(Xtest,Ytest)
```

Feature Selection using RFECV:

- RFECV stands for “Recursive Feature Elimination & Cross Validation”
- Eliminating unimportant feature recursively



Important Functions:

RFECV(model, step, min_features_to_select, n_jobs)

.support()

.ranking()

7. Train the Model using suitable ML Models & RFECV

```
from sklearn.linear_model import LinearRegression
from sklearn.feature_selection import RFECV

model = LinearRegression()
rfecv = RFECV(model , step = 1, min_features_to_select = 4 , n_jobs = -1)
rfecv.fit(Xtrain , Ytrain)

print(rfecv.support_)
print(rfecv.ranking_)

selected_features = np.where(rfecv.support_)[0]
Xtrain = Xtrain[:,selected_features]
Xtest = Xtest[:,selected_features]
model.fit(Xtrain , Ytrain)
```

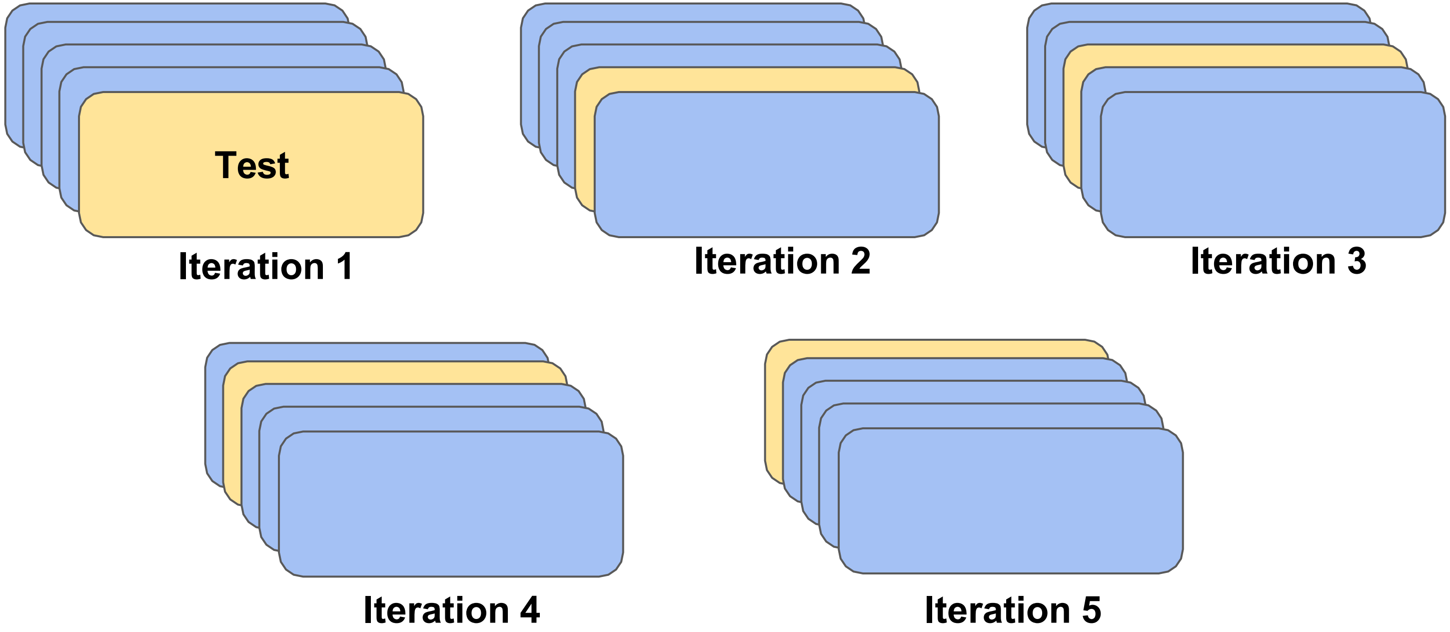
8. Check Accuracy Score of ML Model

```
model.score(Xtest,Ytest)
```

K folds Cross Validation:

- 20% Test data is too small to be confident about predictions
- We are not validating our predictions on the remaining 80% of the data as we have trained our model on it

K folds cross validation solves these issues



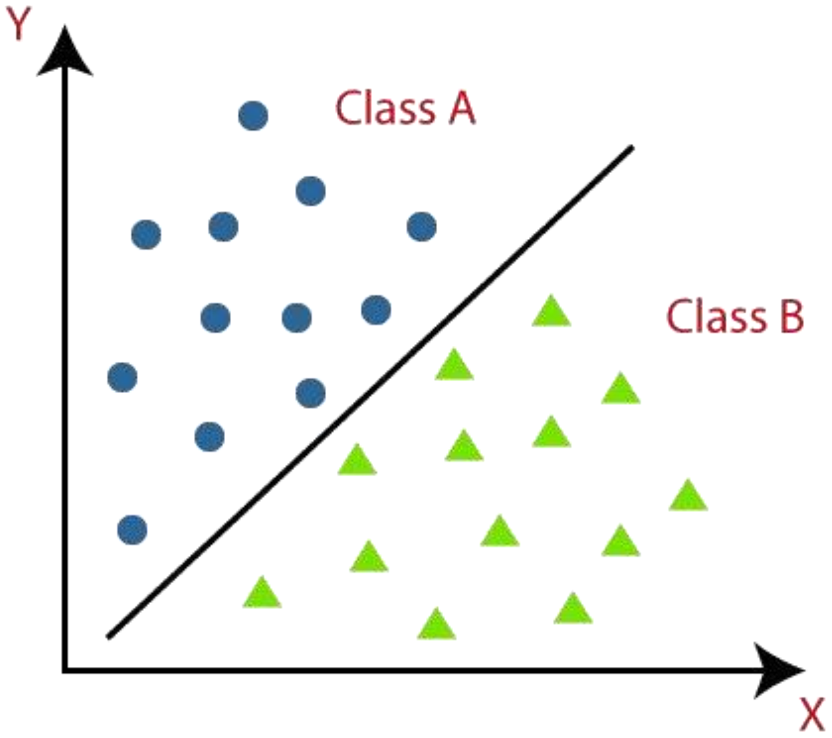
Disadvantages of K-folds Cross Validation:

We re-create Train and Test split in every Iteration and train our model from scratch

- **If k = 5 , we need to train the model 5 times**
- **Computationally expensive when data is too large**

K-folds Cross Validation:

```
from sklearn.model_selection import KFold

k_fold = KFold(n_splits=5)

test_scores = []
for train_idx , test_idx in k_fold.split(X):
    Xtrain = X[train_idx]
    Ytrain = Y[train_idx]

    Xtest = X[test_idx]
    Ytest = Y[test_idx]

    model = LinearRegression()
    model.fit(Xtrain , Ytrain)

    test_scores.append(model.score(Xtest , Ytest))
```


Y
Class A
Class B
X

Classification Models

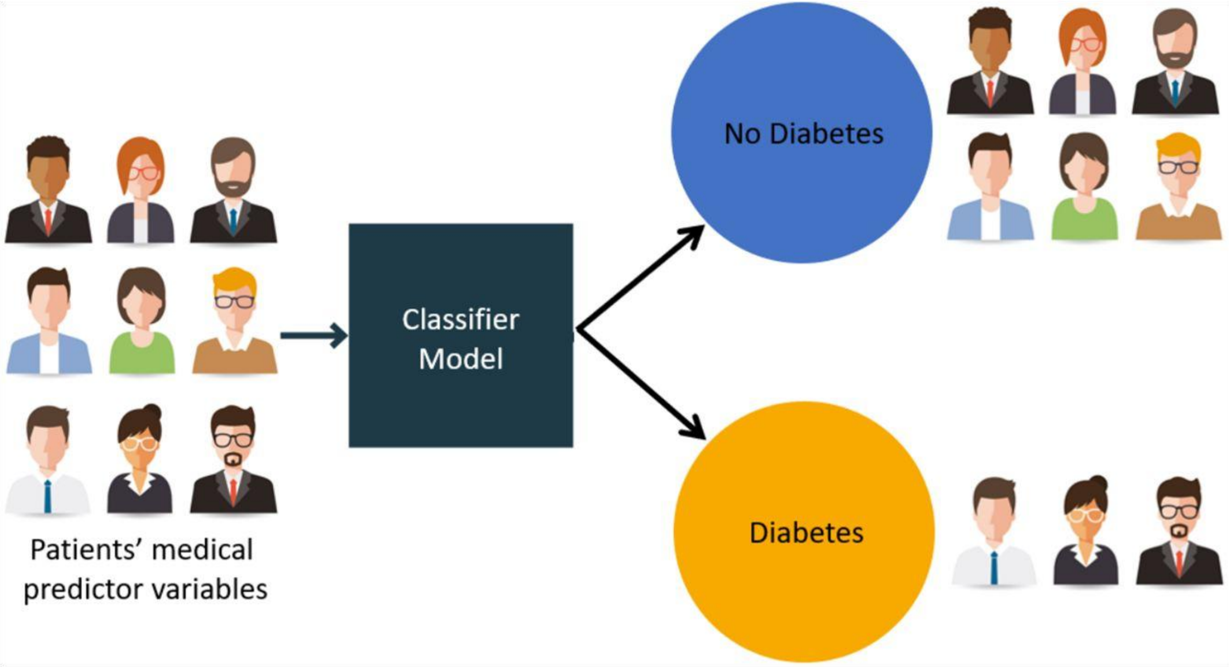
1. Logistic Regression
2. SVC
3. Decision Tree
4. K-Nearest Neighbour
5. Random Forest

LOGISTIC REGRESSION

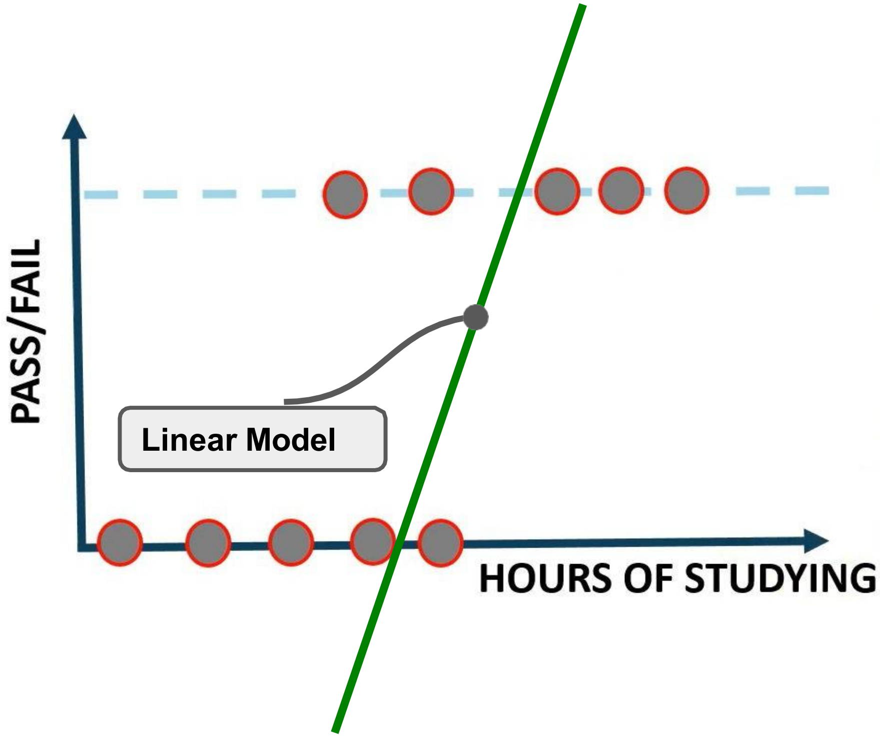
LOGISTIC REGRESSION IS EASY TO INTERPRETABLE OF ALL CLASSIFICATION MODELS.

Logistic Regression:

- Logistic regression is a classification model
- Classification models predict a discrete value

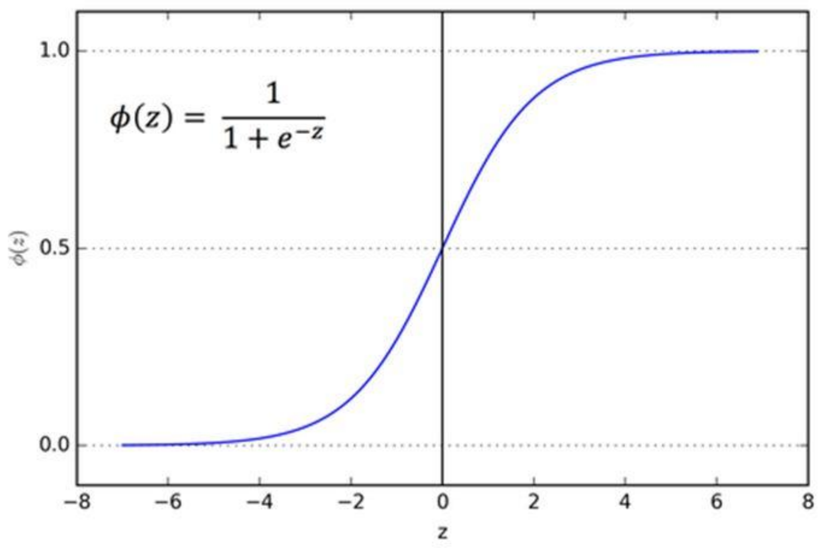


Logistic Regression:



Study Hours	Pass/Fail
1	0
1.5	0
2	0
3	1
3.5	0
4	1
5	1
6	1

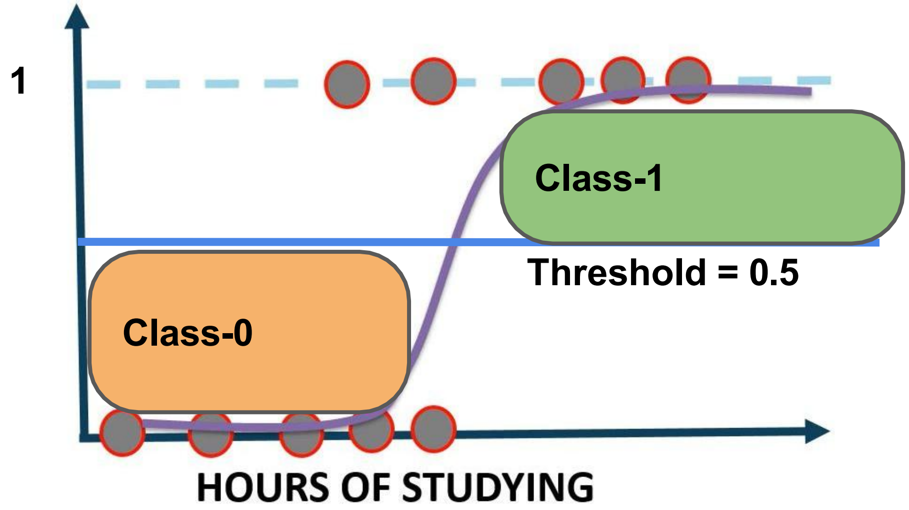
Logistic Regression:



Study Hours	Pass/Fail
1	0
1.5	0
2	0
3	1
3.5	0
4	1
5	1
6	1

Logistic Regression:

Study Hours	Pass/Fail
1	0
1.5	0
2	0
3	1
3.5	0
4	1
5	1
6	1



Linear equation:

- $y = b_0 + b_1 * x$

Apply Sigmoid function:

- $P(x) = sigmoid\ (y)$

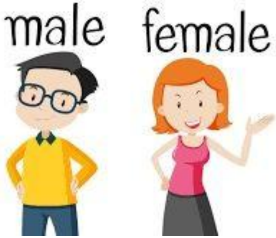
$$P(x) = \frac{1}{1+e^{-y}}$$

$$P(x) = \frac{1}{1+e^{-(b_0+b_1*x)}}$$

Logistic Regression:

Project 3: Customer Churn Prediction using Logistic Regression

Age (X1)



Gender (X2)

Salary (X3)

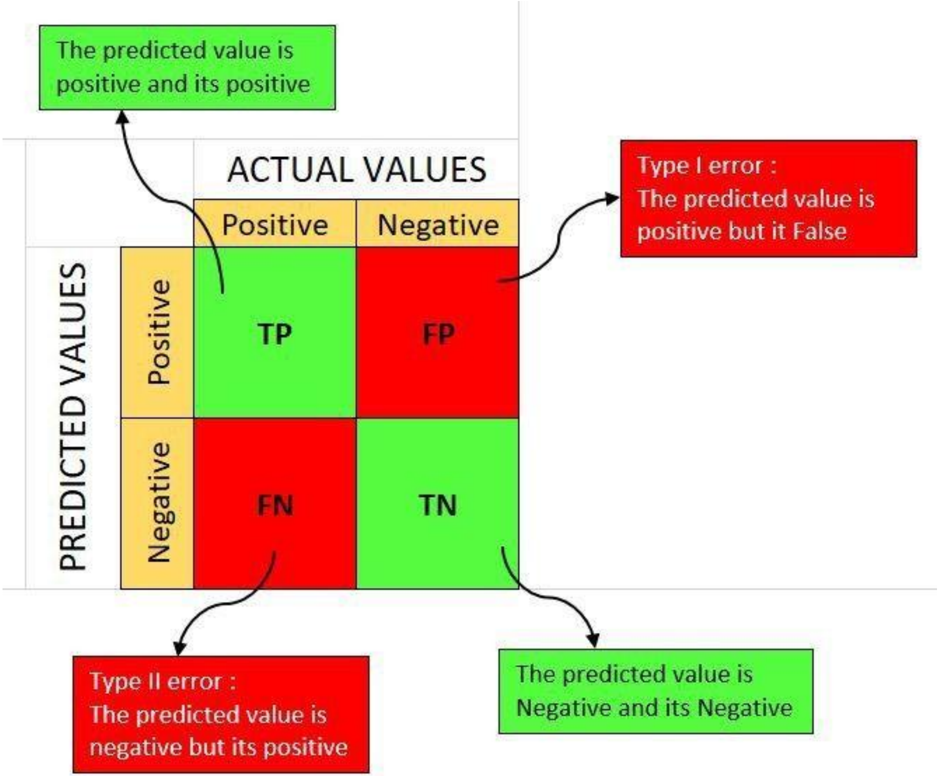
Credit Score (X4)

No. Of Products (X5)

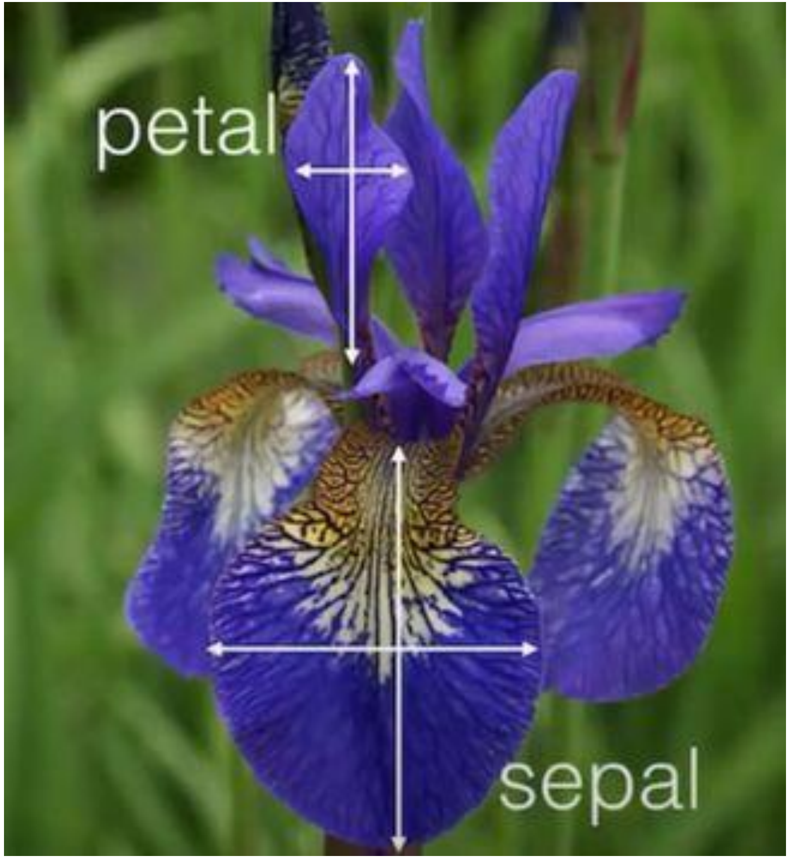
Geography (X6)

Exit or Not (Y)

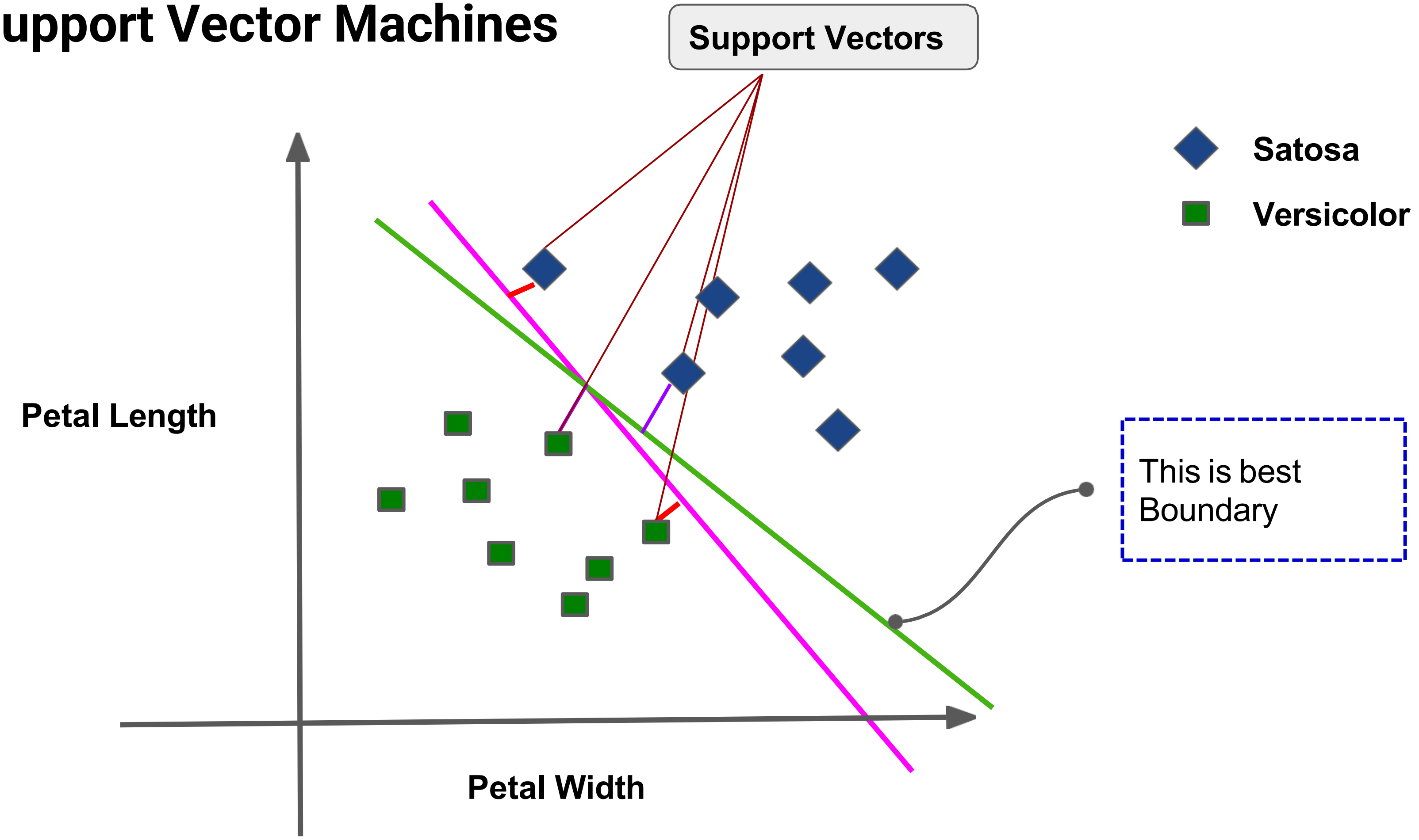
Confusion Matrix:



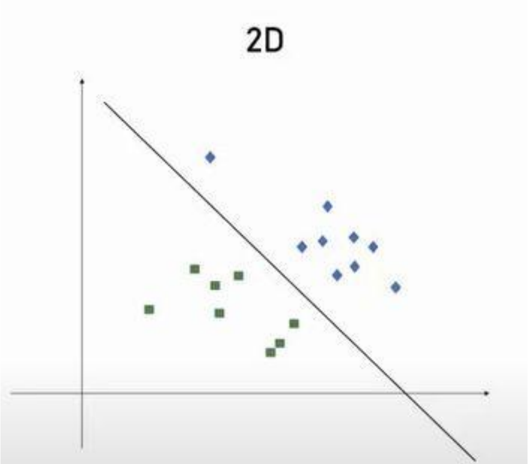
Support Vector Machines

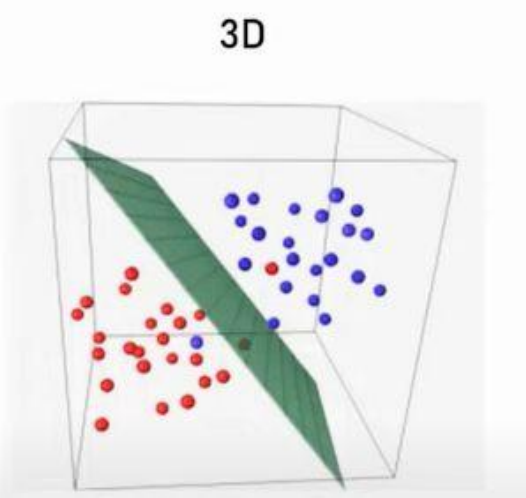


Support Vector Machines



Support Vector Machines

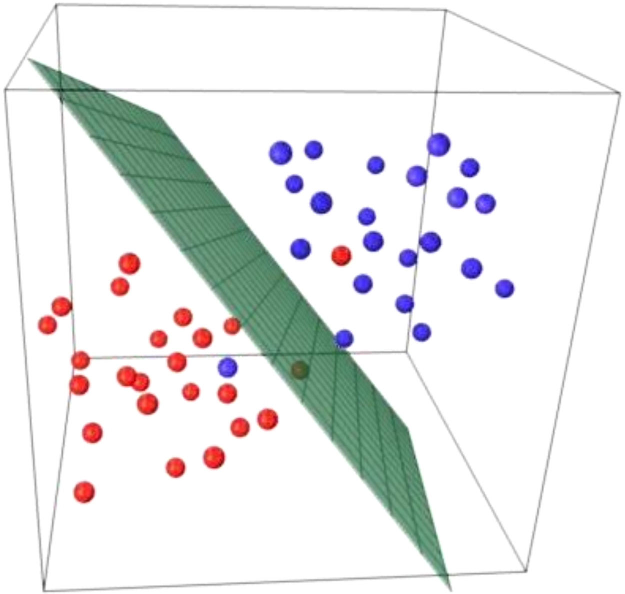


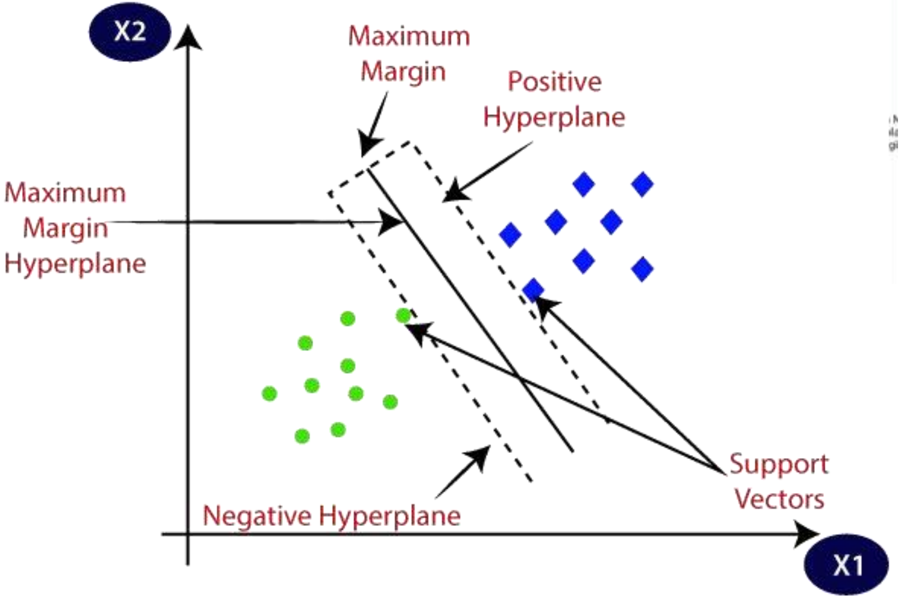


nD

Hyperplanes

Hyperplanes

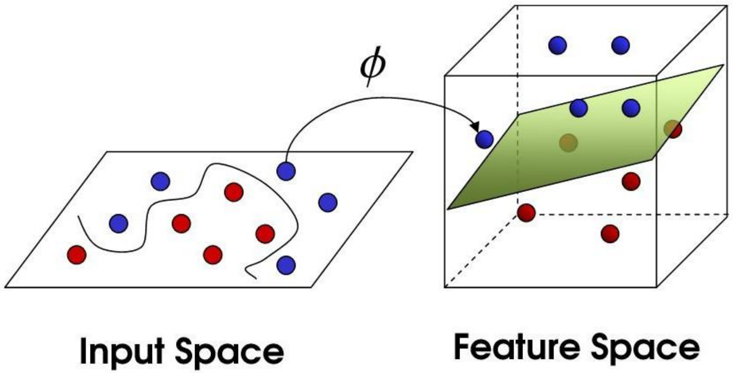




The Kernel Tricks for Support Vector Machines

Understanding Kernel Trick

- Two classes of observations: the blue points and the purple points.
- There are tons of ways to separate these two classes as shown in the graph on the left.

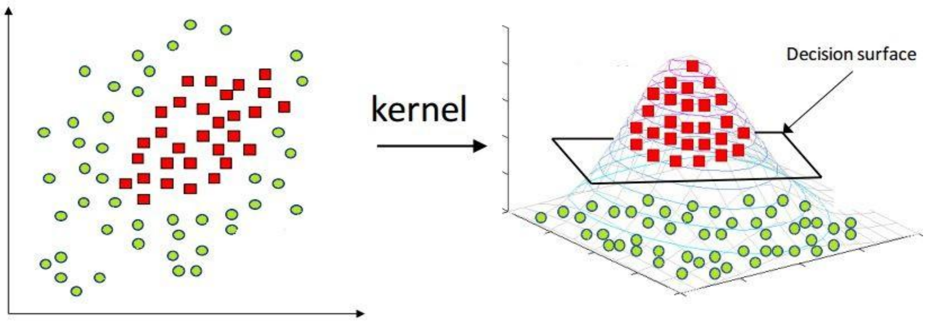


How to ffind best Hyperplane?

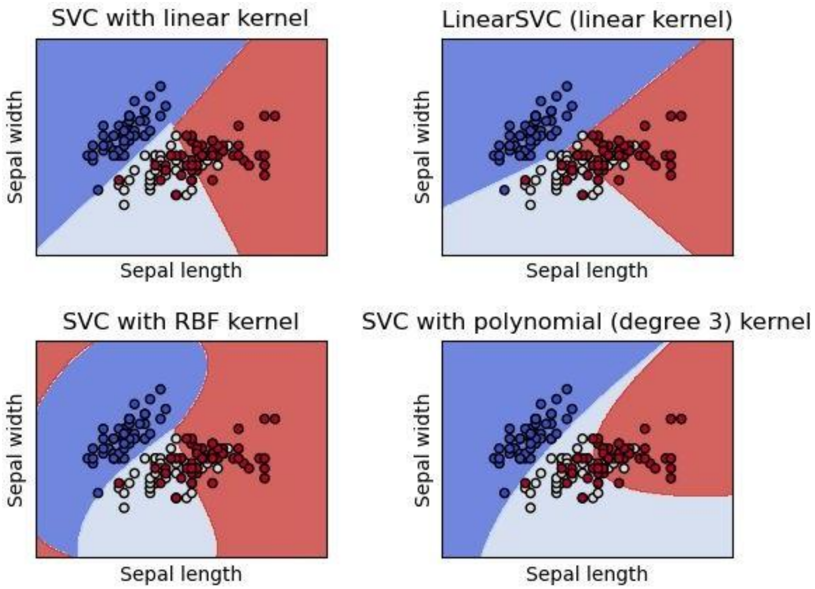
- To find the “best” hyperplane that could maximize the margin between the hyperplane and the nearest data points on each side is the largest.
- Depending on which side of the hyperplane a new data point locates, we could assign a class to the new observation.

Understanding Kernel Trick

- Not all data are linearly separable.
- To map the data from 2-dimensional space to 3-dimensional space, we will be able to find a decision surface that clearly divides between different classes.



Types of Kernels



- **Linear Kernel**
- **Polynomial Kernel**
- **Radial Basis Function (RBF) kernel**

Support Vector Machine Implementation:

```
from sklearn.svm import SVC
from sklearn.metrics import classification_report,confusion_matrix, accuracy_score

#Linear SVM model
svc_clf=SVC(kernel='linear')
svc_clf.fit(X_train,y_train)
y_pred=svc_clf.predict(X_test)
```

```
#confusion matrix
cm=confusion_matrix(y_pred,y_test)
sns.heatmap(cm, annot=True)
```

```
#accuracy and classification report
print(accuracy_score(y_pred,y_test))
print(classification_report(y_pred,y_test))
```

Support Vector Machine Implementation:

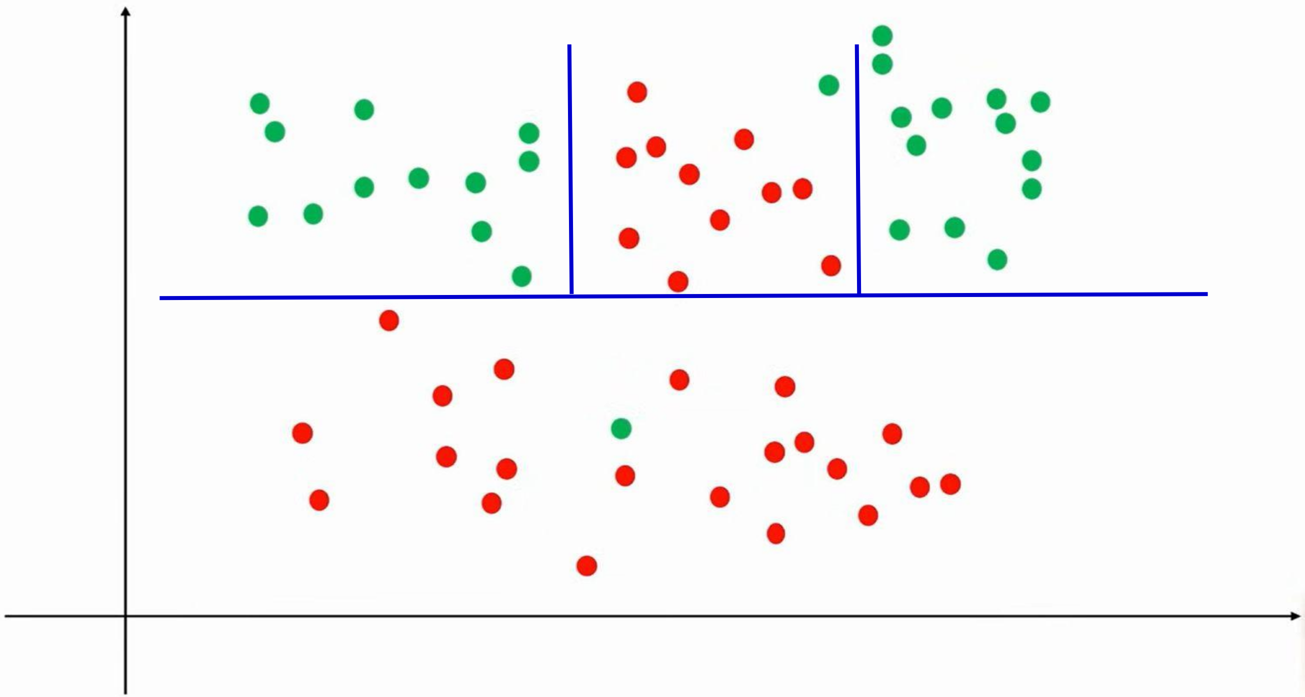
```
from sklearn.svm import SVC
from sklearn.metrics import classification_report,confusion_matrix, accuracy_score
```

```
#rbf svm model
svc_clf_rbf=SVC(kernel='rbf')
svc_clf_rbf.fit(X_train,y_train)
y_pred=svc_clf_rbf.predict(X_test)
```

```
#confusion matrix
cm=confusion_matrix(y_pred,y_test)
sns.heatmap(cm, annot=True)
```

```
#accuracy and classification report
print(accuracy_score(y_pred,y_test))
print(classification_report(y_pred,y_test))
```

Decision Tree



Decision Tree

We can consider Decision Tre ... of ...

1. Spam
    a. Earn fre
    b. Free m
2. Not Spam
    a. Hi good ... ick call I have a
    doubts !

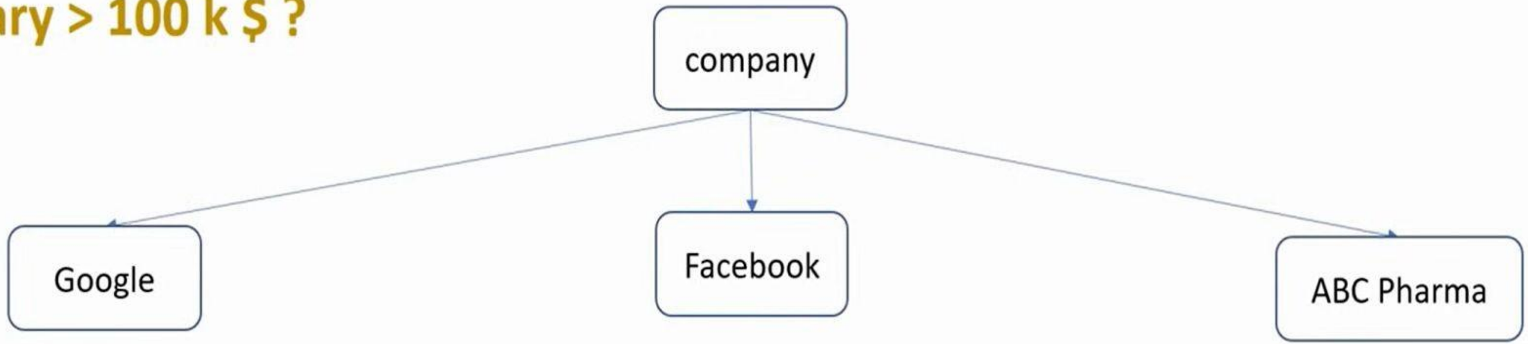
Money

No

Not Spam

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 $?



google	sales executive	bachelors
google	sales executive	masters
google	business manager	bachelors
google	business manager	masters
google	computer programmer	bachelors
google	computer programmer	masters

?

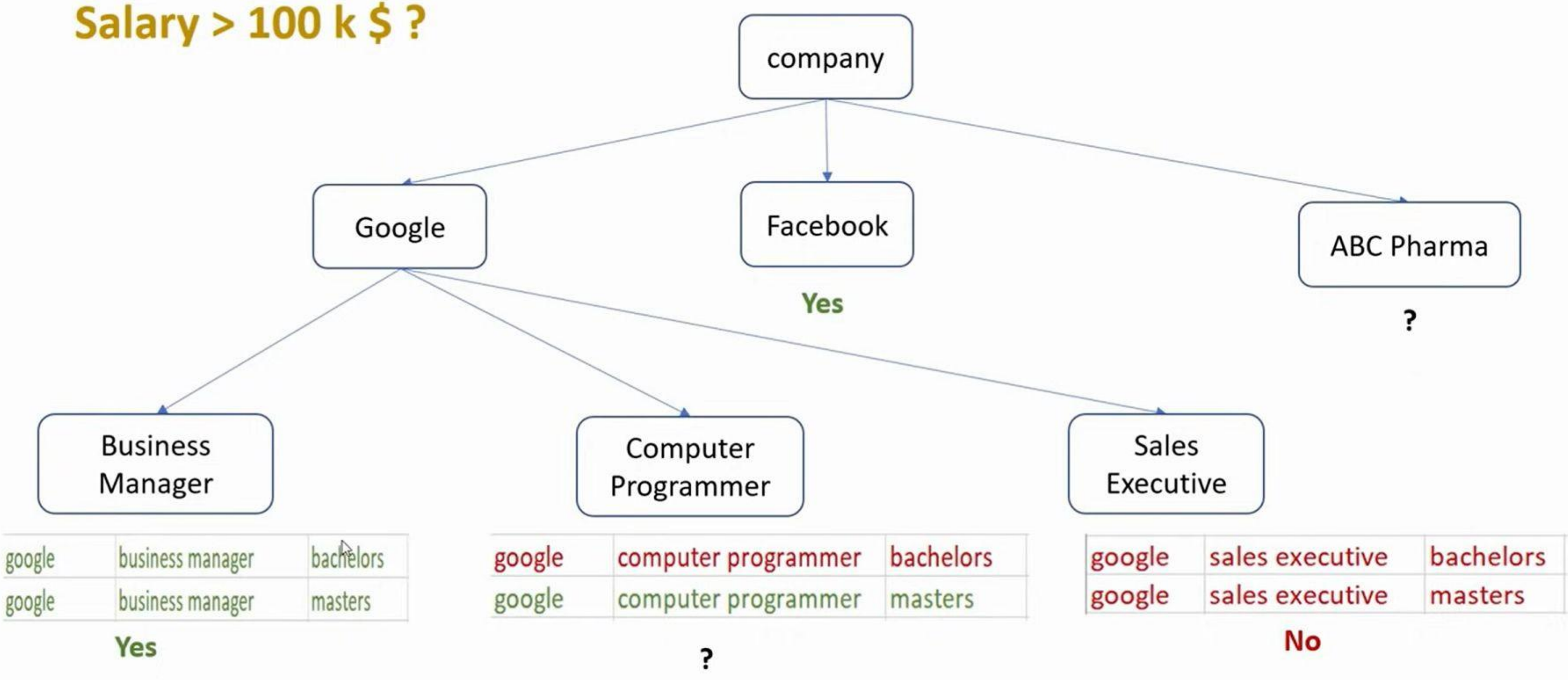
facebook	sales executive	bachelors
facebook	sales executive	masters
facebook	business manager	bachelors
facebook	business manager	masters
facebook	computer programmer	bachelors
facebook	computer programmer	masters

Yes

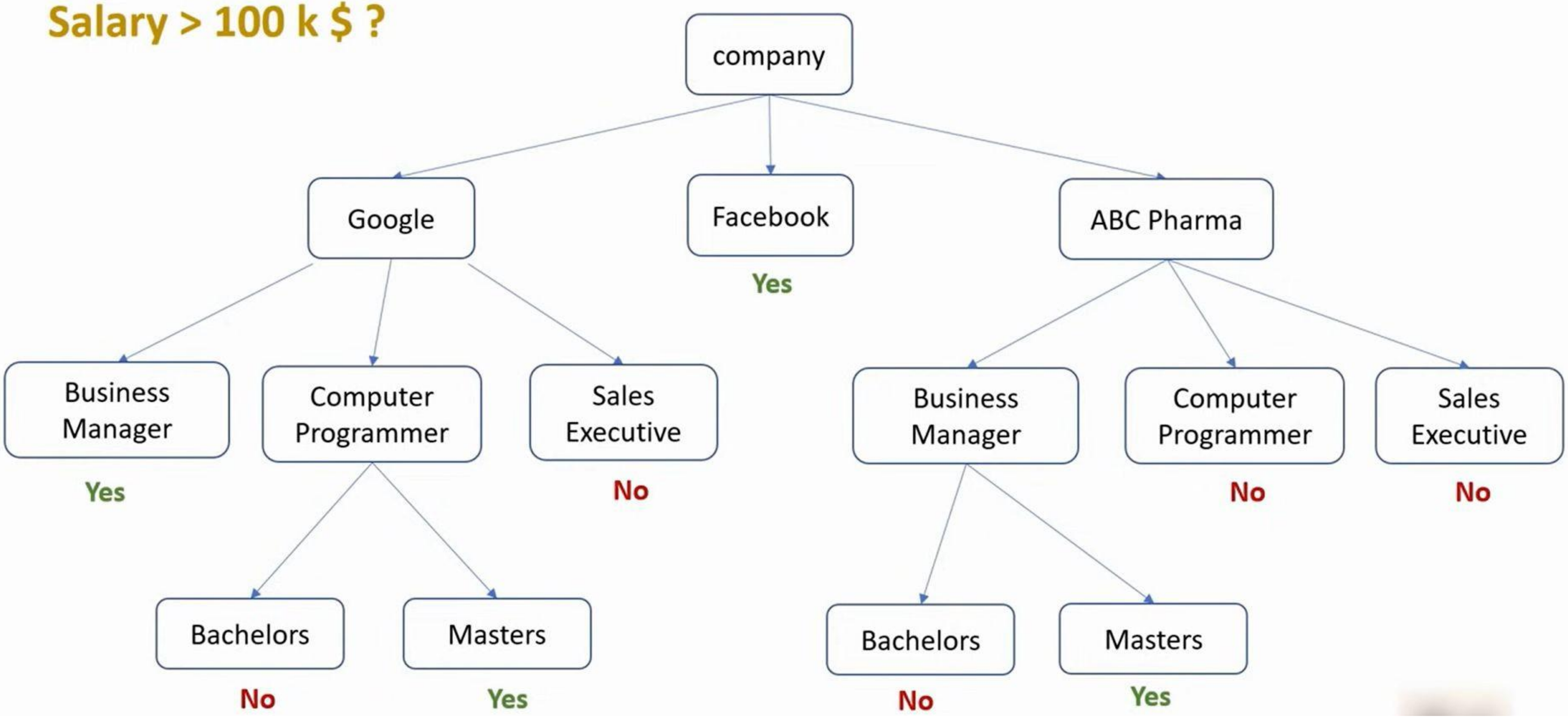
abc pharma	sales executive	masters
abc pharma	computer programmer	bachelors
abc pharma	business manager	bachelors
abc pharma	business manager	masters

?

Salary > 100 k $?


company
Google
Facebook
ABC Pharma
Yes
?
Business Manager
Computer Programmer
Sales Executive
google business manager bachelors
google business manager masters
Yes
google computer programmer bachelors
google computer programmer masters
?
google sales executive bachelors
google sales executive masters
No

Salary > 100 k $?



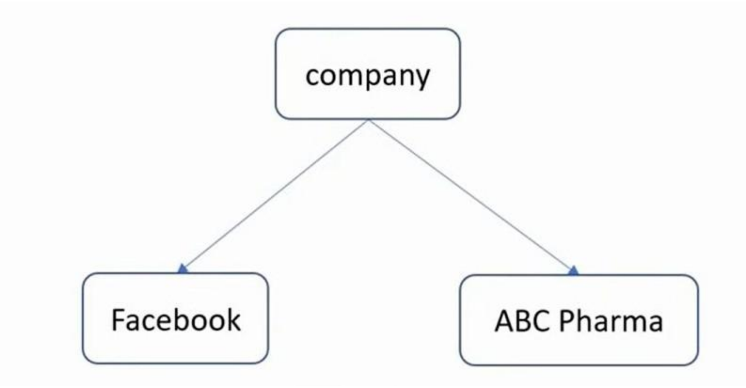
company
Google
Facebook
ABC Pharma
Yes
Business Manager
Computer Programmer
Sales Executive
Business Manager
Computer Programmer
Sales Executive
Yes
No
No
No
Bachelors
Masters
Bachelors
Masters
No
Yes
No
Yes

Decision Tree

- In Decision tree terminology ‘Feature’ or a ‘Column’ is called an ‘Attribute’
- There are mainly two algorithms to control the splitting conditions in a decision tree
  - **Information gain (Entropy)**
  - **Gini index**

$$Entropy = \sum_{i=1}^{C} -p_i * \log_2(p_i)$$

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



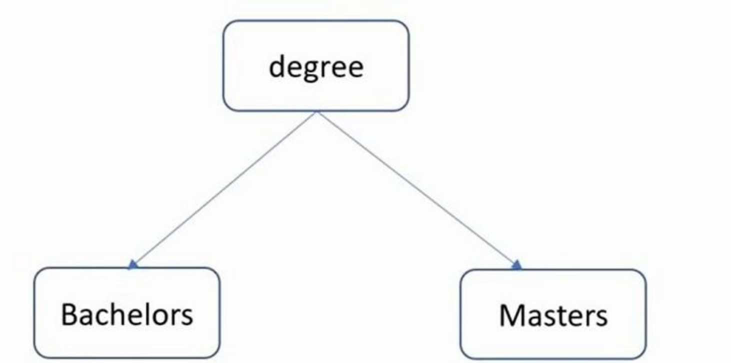
facebook	sales executive	bachelors
facebook	sales executive	masters
facebook	business manager	bachelors
facebook	business manager	masters
facebook	computer programmer	bachelors
facebook	computer programmer	masters

abc pharma	sales executive	masters
abc pharma	computer programmer	bachelors
abc pharma	business manager	bachelors
abc pharma	business manager	masters

6 / 0 (Low Entropy)

1 / 3

High Information Gain



google	sales executive	bachelors
google	business manager	bachelors
google	computer programmer	bachelors
abc pharma	computer programmer	bachelors
abc pharma	business manager	bachelors
facebook	sales executive	bachelors
facebook	business manager	bachelors
facebook	computer programmer	bachelors

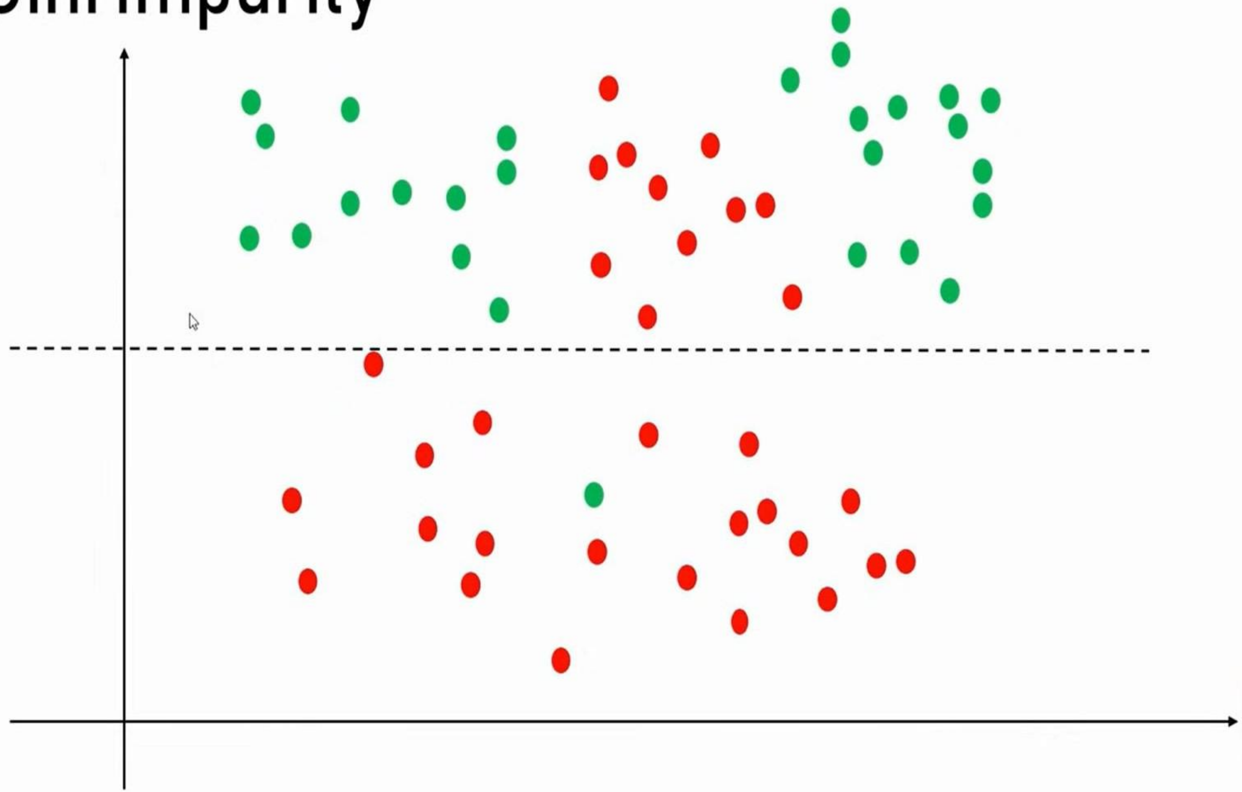
google	sales executive	masters
google	business manager	masters
google	computer programmer	masters
abc pharma	sales executive	masters
abc pharma	business manager	masters
facebook	sales executive	masters
facebook	business manager	masters
facebook	computer programmer	masters

4 / 4 (High Entropy)

6 / 2

Low Information Gain

Gini Impurity



Decision Tree Implementation:

```
from sklearn.tree import DecisionTreeClassifier

model = DecisionTreeClassifier()
model.fit(Xtrain , Ytrain)
```

```
print("Testing Accuracy : " , model.score(Xtest , Ytest))
```

```
y_pred = model.predict(xtest)
```

```
#confusion matrix
cm=confusion_matrix(y_pred,y_test)
sns.heatmap(cm, annot=True)
```

```
#accuracy and classification report
print(accuracy_score(y_pred,y_test))
print(classification_report(y_pred,y_test))
```