

# **AI\_PHASE3**

## **TEAM MEMBERS**

**M.ABISHEK**

**K.BALU**

**M.KARTHICK**

**K.NITHIN**

# Market Basket Insights

## Important note:

This notebook has become very large, and I could not add more information to it, so I will divide it into a number of notebooks. So you can understand the content And be an excellent reference for you

## Table of Content:

Machine Learning and Types

Application of Machine Learning

Steps of Machine Learning

Factors help to choose algorithm

Algorithm

Evaluate Algorithms

Linear Regression

Logistic Regression

Support Vector Machine

Naive Bayes Algorithm

KNN

Perceptron

Random Forest

Decision Tree

Extra Tree

Gradient Boosting

Light GBM

XGBoost

Catboost

Stochastic Gradient Descent

Lasso

Kernel Ridge Regression

Bayesian Ridge

Elastic Net Regression

LDA

K-Means Algorithm

CNN

LSTM

PCA

Apriori

Prophet

ARIMA

**Machine Learning:**

## Linear Regression:

It is a basic and commonly used type of predictive analysis. These regression estimates are used to explain the relationship between one dependent variable and one or more independent variables.  $Y = a + bX$  where

$Y$  - Dependent Variable

$a$  - intercept - Bias

$X$  - Independent variable

$b$  - Slope - Weights

Hypothesis:  $h_{\theta}(x) = \theta_0 + \theta_1 x$

Parameters:  $\theta_0, \theta_1$

Cost Function:  $J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$

Goal:  $\underset{\theta_0, \theta_1}{\text{minimize}} J(\theta_0, \theta_1)$

Example:  $\text{University GPA}' = (0.675)(\text{High School GPA}) + 1.097$

```
import numpy as np
```

```
import pandas as pd
```

```
from matplotlib import pyplot as plt
```

```

from sklearn.linear_model import LinearRegression

from sklearn.metrics import mean_squared_error, r2_score

train = pd.read_csv("../input/random-linear-regression/train.csv")
test = pd.read_csv("../input/random-linear-regression/test.csv")

train = train.dropna()

test = test.dropna()

train.head()

```

OUTPUT :

	x	y
0	24.0	21.549452
1	50.0	47.464463
2	15.0	17.218656
3	38.0	36.586398
4	87.0	87.288984

### Model with plots and accuracy:

```

X_train = np.array(train.iloc[:, :-1].values)

y_train = np.array(train.iloc[:, 1].values)

X_test = np.array(test.iloc[:, :-1].values)

y_test = np.array(test.iloc[:, 1].values)

model = LinearRegression(fit_intercept=True,
normalize=True,copy_X=True,n_jobs=-1)

model.fit(X_train, y_train)

y_pred = model.predict(X_test)

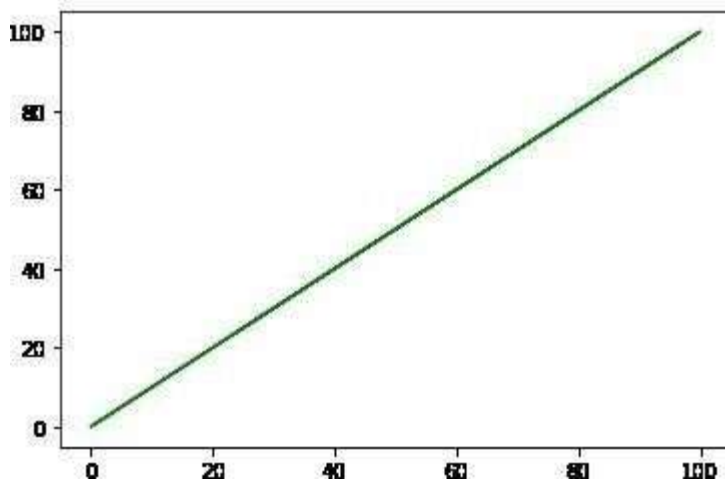
```

```
accuracy = model.score(X_test, y_test)
```

```
plt.plot(X_train, model.predict(X_train), color='green')
```

```
plt.show()
```

```
print(accuracy)
```



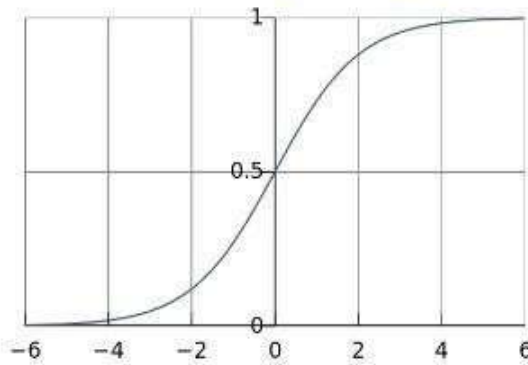
```
0.9888014444327563
```

### *Logistic Regression:*

It's a classification algorithm, that is used where the response variable is categorical. The idea of Logistic Regression is to find a relationship between features and probability of particular outcome.

$$\text{odds} = \frac{p(x)}{1-p(x)} = \frac{\text{probability of event occurrence}}{\text{probability of not event occurrence}}$$

# Binary Logistic Regression

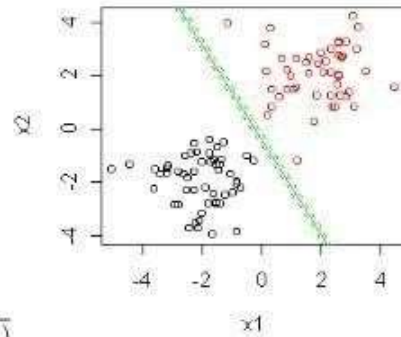


$$P(y=1|\vec{x}, \vec{w}) = \frac{\exp(d)}{1+\exp(d)} = \frac{\exp(\vec{x} \vec{w})}{1+\exp(\vec{x} \vec{w})}$$

$$P(y=0|\vec{x}, \vec{w}) = \frac{1}{1+\exp(\vec{x} \vec{w})}$$

$$\log \frac{P(y=1|\vec{x}, \vec{w})}{P(y=0|\vec{x}, \vec{w})} = \vec{x} \vec{w}$$

$$d = \frac{ax_1 + bx_2 + cx_0}{\sqrt{a^2 + b^2}} \text{ where } x_0 = 1$$



$$w_0 = \frac{c}{\sqrt{a^2 + b^2}} \text{ where } w_0 \text{ is called as intercept}$$

$$w_1 = \frac{a}{\sqrt{a^2 + b^2}}$$

$$w_2 = \frac{b}{\sqrt{a^2 + b^2}}$$

## Libraries and data:

```
import sklearn
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import r2_score
from statistics import mode
```

```
train = pd.read_csv("../input/titanic/train.csv")
```

```
test = pd.read_csv("../input/titanic/test.csv")
```

```
train.head()
```

OUTPUT:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

## Model and Accuracy:

```

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,
random_state=5)

from sklearn.linear_model import LogisticRegression

#linear_model.LogisticRegression(penalty='l2' , dual=False, tol=0.0001, C=1.0, fit
_intercept=True, intercept_scaling=1,
#  class_weight=None, random_state=None, solver='warn' , max_iter=100,
#  multi_class='warn' , verbose=0, warm_start=False, n_jobs=None)

```

```

model = LogisticRegression(max_iter = 500000)
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

accuracy = model.score(X_test, y_test)

print(accuracy)

```

OUTPUT:

0.8251121076233184

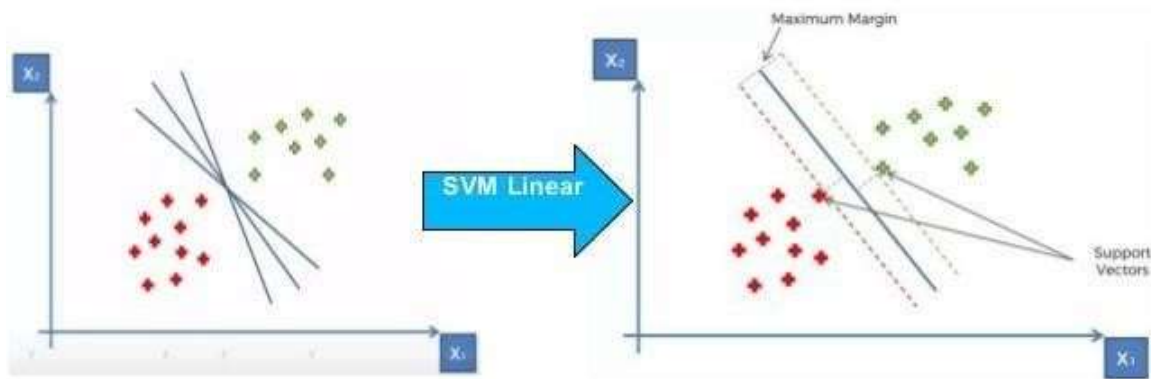
## Support Vector Machine:



## Classification Model : SVM - Linear

### Linearly separate the data points

Support Vector Machine" (SVM) is a **supervised machine** learning algorithm which can be used for **both classification or regression challenges**. However, it is mostly used in classification problems. In this algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the **hyper-plane that differentiate the two classes** very well (look at the below snapshot).



Prepared by Manish

©

IT.mnchoudhary@gmail.com

Support Vector Machines are perhaps one of the most popular and talked about machine learning algorithms. It is primarily a classifier method that performs classification tasks by constructing hyperplanes in a multidimensional space that separates cases of different class labels. SVM supports both regression and classification tasks and can handle multiple continuous and categorical variables.

Example: One class is linearly separable from the others like if we only had two features like Height and Hair length of an individual, we'd first plot these two variables in two dimensional space where each point has two co-ordinates

### Libraries and Data:

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

```
from sklearn.model_selection import cross_val_score
```

```
from sklearn.svm import SVC
```

```
data_svm = pd.read_csv("../input/svm-classification/UniversalBank.csv")
```

```
data_svm.head()
```

	ID	Age	Experience	Income	ZIP Code	Family	CCAvg	Education	Mortgage	Personal Loan	Securities Account	CD Account	Online	CreditCard
0	1	25	1	49	91107	4	1.6	1	0	0	1	0	0	0
1	2	45	19	34	90089	3	1.5	1	0	0	1	0	0	0
2	3	39	15	11	94720	1	1.0	1	0	0	0	0	0	0
3	4	35	9	100	94112	1	2.7	2	0	0	0	0	0	0
4	5	35	8	45	91330	4	1.0	2	0	0	0	0	0	1

## **Model and Accuracy:**

```
X = data_svm.iloc[:,1:13].values
```

```
y = data_svm.iloc[:, -1].values
```

```
X_train,X_test,y_train,y_test = train_test_split(X,y, test_size = 0.25, random_state = 0)
```

```
'''
```

```
sklearn.svm.SVC(C=1.0, kernel='rbf', degree=3, gamma='auto_deprecated', coef0=0.0,  
shrinking=True,
```

```
probability=False,tol=0.001,cache_size=200,  
class_weight=None,verbose=False,
```

```
max_iter=-1, decision_function_shape='ovr', random_state=None)
```

```
'''
```

```
classifier = SVC(kernel = 'rbf', random_state = 0)
```

```
classifier.fit(X_train, y_train)
```

```
y_pred = classifier.predict(X_test)
```

```
accuracies = cross_val_score(estimator = classifier,X = X_train,y = y_train, cv = 10)
```

```
accuracies.mean()
```

**OUTPUT:**

0.7077333333333333

## Naive Bayes Algorithm:

# Naive Bayes

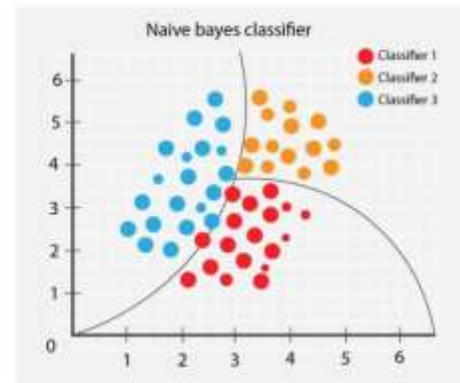


In machine learning, naive Bayes classifiers are a family of simple "probabilistic classifiers" based on applying Bayes' theorem with strong (naive) independence assumptions between the features.

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

using Bayesian probability terminology, the above equation can be written as

$$\text{Posterior} = \frac{\text{prior} \times \text{likelihood}}{\text{evidence}}$$



A naive Bayes classifier is not a single algorithm, but a family of machine learning algorithms which use probability theory to classify data with an assumption of independence between predictors. It is easy to build and particularly useful for very large data sets. Along with simplicity, Naive Bayes is known to outperform even highly sophisticated classification methods.

Example: Emails are given and we have to find the spam emails from that. A spam filter looks at email messages for certain key words and puts them in a spam folder if they match.

## Libraries and Data:

```
from sklearn.naive_bayes import GaussianNB
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score
```

```

data =
pd.read_csv('../input/classification-suv-dataset/Social_Network_Ads.
csv')
data_nb =
data
data_nb.head(
)

```

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0

## Model and Accuracy:

```

X = data_nb.iloc[:,
[2,3]].values
data_nb.iloc[:, 4].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state =
0)
sc_X = StandardScaler()
X_train =
sc_X.fit_transform(X_train)
X_test =
sc_X.transform(X_test)

'''

#sklearn.naive_bayes.GaussianNB(priors=None, var_smoothing=1e-09)

'''

classifier=GaussianNB()
classifier.fit(X_train,y_train)
y_pred=classifier.predict(X_test)

```

```
acc=accuracy_score(y_test,  
y_pred)  
print(acc)
```

## KNN :

KNN does not learn any model. and stores the entire training data set which it uses as its representation. The output can be calculated as the class with the highest frequency from the K-most similar instances. Each instance in essence votes for their class and the class with the most votes is taken as the prediction

