Logistic Regression

* Machine learning algorithm used to **solve classification problems**.
* It is a Predictive analysis algorithm and **based on the concept of probability**.
* This is also a call this a Linear Regression model but the uses a complex function instead of linear function. This complex function we call it as **“Sigmoid Function”.**
* The hypothesis of the logistic regression tends it to limit the complex function between 0 and 1.
* Sigmoid function is used to **map predicted values to the probabilities.** This function maps any real value into another values b/w 0 and 1.
* Classification algorithm used to assign observations to a discrete set of classes.
* Examples: Email spam or not spam, Online Transaction Fraud or not Fraud.

We basically have 2 classes and we decide with a threshold value above which we classify values into class 1 and the value goes below threshold value to class 2.

Classification: is technique of learning where an instance is mapped to one of the many labels. The machine learns the patterns from the data in such a way that the learned representation successfully maps the original dimension to the suggested label/class without any more intervention from a human expert.

## Implementing logistic Regression

1. Collecting Data
2. Analyzing Data
3. Data Wrangling
4. Train & Test
5. Accuracy Check

# Coding Part

# #importing libraries

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

from sklearn.cross\_validation import train\_test\_split

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import confusion\_matrix

from sklearn.metrics import classification\_report

# #collectiong data

Titanic = pd.read\_csv(r"C:\Users\kumars262\Downloads\train.csv")

#Titanic.info() - provides all the information about data

#Titanic.head() - provides data for first 5 records

#Titanic.tail() - provides data for last 5 records

#Titanic.count()- provides data about all the data inclues number of records, datatypes

#Titanic.describe()

# #analyzing the data

#sns.countplot(x="Sex", data=Titanic) -> gives count of men vs women

#sns.countplot(x="Survived", data=Titanic) -> gives count of peopel survived vs dead

#sns.countplot(x="Survived", hue="Sex", data=Titanic) -> gives count of peopel survived vs dead with respect to gender

#sns.countplot(x="Survived", hue="Pclass", data=Titanic) -> gives count of peopel survived vs dead with respect to PassClass

#Titanic["Age"].plot.hist() -> gives age details in a histogram

#Titanic["Fare"].plot.hist() -> gives fare details in a histogram

#Titanic[Titanic["Sex"].str.match("female")].count() -> give you number of females

#Titanic[Titanic["Sex"].str.match("male")].count() -> gives you number of male

#Titanic[Titanic["Name"].str.match("Brown")] -> gives you number of people with the name Brown

#sns.boxplot(x="Pclass", y="Age", data=Titanic) -> gives you age with respect to Pclass

#Cleaning the data

#Titanic.isnull() -> it will show false if not null and true if null in all the columns

#Titanic.isnull().sum() --> it will give you sum of null values in each column

#plt.figure(figsize=(20.0,10.0))

#sns.heatmap(Titanic.isnull())

## #Filling missing values (with mean)

#mean = Titanic["Age"].mean() -> we have got the mean of the values in the Age column

#Titanic["Age"] = Titanic["Age"].fillna(mean) -> we have added mean value to the null values in the column.

#Titanic.isnull().sum()

#Filligng missing values with Imputer Method

#from sklearn.preprocessing import Imputer

#imp = Imputer(missing\_values='NaN', strategy='most\_frequent', axis=0)

#imp.fit(Titanic)

#Titanic = imp.transform(Titanic)

mean = Titanic["Age"].mean()

Titanic["Age"] = Titanic["Age"].fillna(mean)

Titanic.isnull().sum()

#removing Cabin column has it has 687 null vaules

Titanic.drop("Cabin", inplace=True, axis=1)

Titanic.dropna(inplace=True) -> this removes all the rows which has null values

#Inplace=True or False

# if we are using Flase, we need to assign to a new df, since it returns a new copy of the df

# if we mention True, it performs the operation and returns nothing. so need to assign this to a new df.

#Titanic.info()

#sns.heatmap(Titanic.isnull(),cbar=False, yticklabels=False)

Titanic.isnull().sum()

#we are done with cleaning the data

# Data modelling

#we are converting categorical values to numerical values - get\_dummies is the function in pandas

pd.get\_dummies(Titanic["Sex"])

pd.get\_dummies(Titanic["Pclass"])

pd.get\_dummies(Titanic["Embarked"])

Titanic.head()

#we can do is to remove the first column because one column indicates the value of the other column.

#For example, if the male is 1, then the female will be 0 and vice versa.

Sex=pd.get\_dummies(Titanic["Sex"], drop\_first=True)

Pclass=pd.get\_dummies(Titanic["Pclass"], drop\_first=True)

Embarked=pd.get\_dummies(Titanic["Embarked"], drop\_first = True)

#adding these columns to the original dataset

Titanic = pd.concat([Titanic,Sex,Pclass,Embarked], axis=1)

Titanic.head()

Titanic.drop(["PassengerId","Pclass","Name","Sex","Ticket","Embarked" ], axis=1, inplace=True)

X=Titanic.drop("Survived", axis=1)

#X is having all the data except survived column (features)

Y=Titanic["Survived"]

#Y is having survived data(Target dat)

X\_train,X\_test, Y\_train, Y\_test = train\_test\_split(X,Y,random\_state=1, test\_size=0.3)

logmodel=LogisticRegression()

logmodel.fit(X\_train,Y\_train)

Y\_pred = logmodel.predict(X\_test)

#print(classification\_report(Y\_test,Y\_pred))

#print(confusion\_matrix(Y\_test, Y\_pred))

plt.scatter(X\_test["Age"], Y\_pred)

