IIT ROORKEE

A Project Report on

" HEART FAILURE PREDICTION"

by

Mr.Ashwin Dubey

A Project Report is made and submitted to EICT-IIT ROORKEE on the basis of knowledge gained about DATA SCIENCE by the course conducted by EICT-IIT ROORKEE.

Content

1. TITLE OF PROJECT

“HEART FAILURE PREDICTION” by

- K-Nearest Neighbors
- Logistic Regression

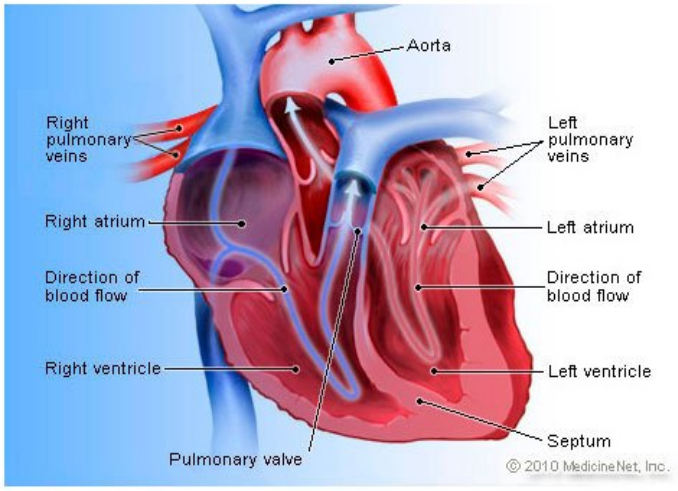
2.Brief on the project

Cardiovascular diseases (CVDs) are the **number 1 cause of death globally**, taking an estimated **17.9 million lives each year**, which accounts for **31% of all deaths worldwide**.

Heart failure is a common event caused by CVDs and this dataset contains 12 features that can be used to predict mortality by heart failure.

Most cardiovascular diseases can be prevented by addressing behavioural risk factors such as tobacco use, unhealthy diet and obesity, physical inactivity and harmful use of alcohol using population-wide strategies.

People with cardiovascular disease or who are at high cardiovascular risk (due to the presence of one or more risk factors such as hypertension, diabetes, hyperlipidaemia or already established disease) need **early detection** and management wherein a machine learning model can be of great help.



Objective: To create a classification filter (Using Logistics Regression & KNN Classification Algorithm) to predict Heart Failure. Also Comparing the performance of the filters

3.Deliverables of the project

The Heart failure.csv dataset can get from a URL source or directly from a kaggle website. After getting a dataset first all process like data cleaning, Data Reduction, Data Analysis, Feature Engineering etc should be done. By observing we can assume that a classification modeling can be done on this dataset. And after successful modeling we can get accuracy score of the trained and test model. From this modeling score we can easily assume how a heart can get fail. For example: A man of age 75 not having anemia, 582 creatinine, with 20 ejection fraction, having high blood pressure, platelets of 265000, serum creatinine of 1.9 , serum sodium of 130, and and no smoking this patient have a chance of heart failure and can die. If a man of age 49, having anemia, having a 80 creatinine, no diebities, having 30 ejection fraction, having high blood pressure with 427000 platelets, having serus cretinine 1, serum sodium 138, and also she don't smokes and she have no chance of death from heart failure. This type of data when trained we can get a model from which we can further assume that a man or women have a tendency of heart failure or not and he or she have risk of death or not.

The more accuracy we get the better the model we have created. This model can easily be used for example a Insurance Company, where they can find if there is any chance of pre mature death from heart failure, before giving a policy to someone. This can be done by asking a question to someone like:

" Do you have anemia? What is your Blood Platelets? Do you have Diabetes? What is your Cretinine level? What is your Ejection Fraction? Do you have blood pressure or not? What is your Serum Sodium and Cretinine? What is your age and do you smoke or not?"

This type of Question can be asked to a person and the answer get from them can help him predicting the chance of heart failure in future. For this puropose the Algorithm we made can be very useful.

4. Resources

Dataset Source:- https://www.kaggle.com/andrewmvd/heart-failure-clinical-data

Software used: To make a modeling of the dataset we have used software like

- Anaconda
- Python Language
- Jupyter Notebook or Pycharm for IDE

Further we have installed many other tools like 'Matplotlib', 'Seaborn', 'Scikit Learn' etc.

Here the Algorithm we are making are of KNN and Logistic Regression. So for this Algorithm it is necessary to install 'sklearn'. After that we can import 'KNeighborClassifiers' from 'sklearn' . We can also import 'LogisticRegression' from 'model_selection' of sklearn.

Reference:

1. ' Krish Naik' YouTube channel
2. 'Khan Academy' Youtube Channel
3. Data Science handbook by 'Jake Vender Plas'
4. 5 Minute Engineering Channel

5. Details of Individual

Name: Ashwin Gorakhnath Dubey

Email: ashwindubey1051994@gmail.com

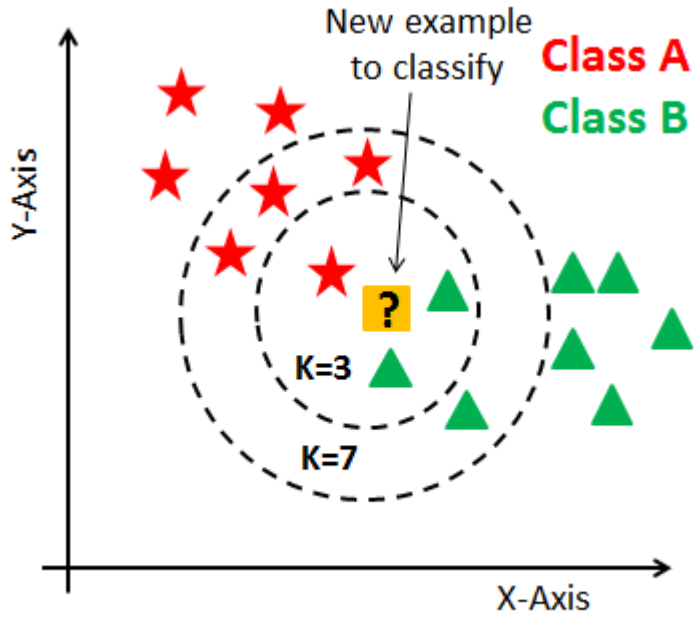
Phone number:- 8275533544

6. MILESTONES

(a) Define:

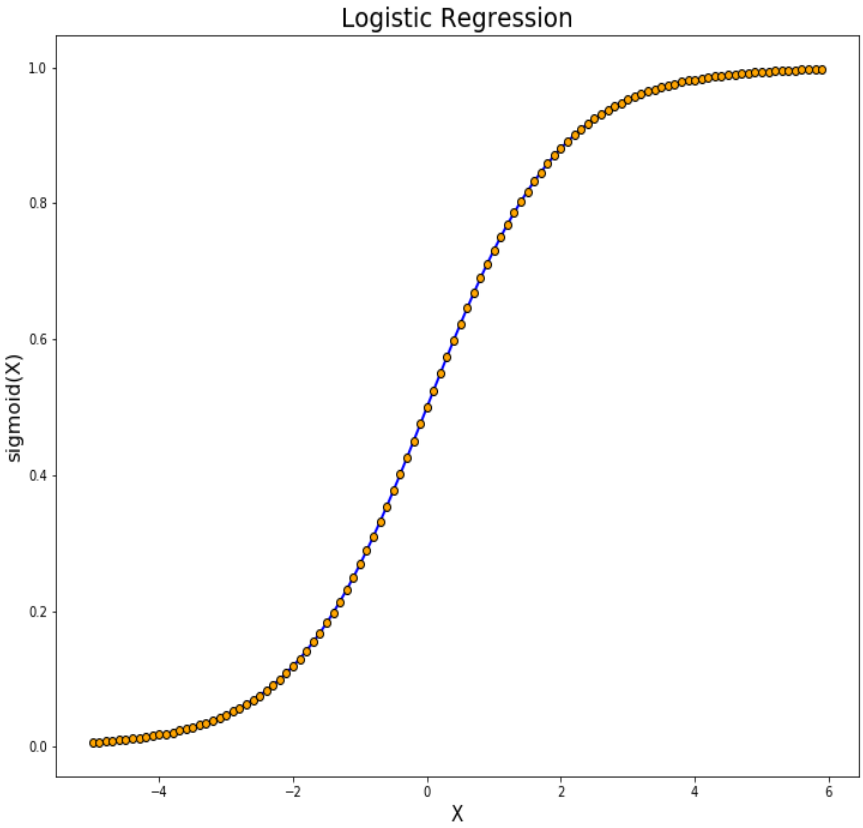
(i) **KNN:** The abbreviation KNN stands for "K-Nearest Neighbour". It is a supervised machine learning algorithm. The algorithm can be used to solve both classification and regression problem statements. The number of nearest neighbours to a new unknown variable that has to be predicted or classified is denoted by the symbol 'K'.

K Nearest Neighbour is a simple algorithm that stores all the available cases and classifies the new data or case based on a similarity measure. It is mostly used to classifies a data point based on how its neighbours are classified.



(5)

(ii) **Logistic Regression:** prior observations of a data set. Logistic regression has become an important tool in the discipline of machine learning. The approach allows an algorithm being used in a machine learning application to classify incoming data based on historical data. Logistic regression can also play a role in data preparation activities by allowing data sets to be put into specifically predefined buckets during the extract, transform, load (ETL) process in order to stage the information for analysis.



(5)

(b) Choose the python platform:

Here we are using IDE Jupyter Notebook powered by Anaconda and the language we are using is Python.

(c) Understanding a business problem:

In the real world business problem, main problem arries when predicting the future profit. It is a old line but its ture that 'No one has saw the future'. It is true, but one can predict future by considering the past. Which action taken at past, Which action taken at which time, When the J-curve, When was the slope, by considering this points the future can be predict. This can be done by training model in the machine on past data and future model can be predict. Here we are dealing with Heart failure data. And on this data we are applying a Supervised Machine Algorithm like **K-Nearest Neighbors** and **Logistic Regression**.

This model can be very useful for the business of Medical and Insurance Company. Here we will split the data say 20% testing and 80% training data. Then we will make a model of KNN and Logistic Regression and then we will try to fit the Training data. After successful fitting we will try to gain a score of test data on basis of trained data. This should be keep in mind that the accuracy of test score should always be smaller then trained data. Because we are splitting training data and testing data in the ratio of 20:80.

By this model business like Medical field and Insurance Company will easily predict the future of that individual whether he or she is eligible for Insurance or he/she is fit or specific job or if he /she need to be very cautious about their heart.

So, in the Business field this machine learning will be very supportive.

(d) Get the data:

Here we will deal with Heart failure dataset and this dataset can found out from URL- https://www.kaggle.com/andrewmvd/heart-failure-clinical-data.

After getting the dataset we have to first load it. For loading we are using Jupyter Notebook and in there we are writing several codes to import required tools such as :

- `import pandas as pd`
- `import numpy as np`
- `import matplotlib.pyplot as plt`

After importing tools we will load dataset for that we will write:

`df = pd.read_csv('heart_failure.csv'),` where 'df' is given any name, 'pd' (pandas) is use to load a dataset, 'read' is written to read the file and 'csv' is the format of file. The loaded file is thus looks like:

In [13]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
```

In [14]:
```
from sklearn.neighbors import KNeighborsClassifier
```

In [15]:
```
df=pd.read_csv('heart_failure.csv')
df.head()
```

Out[15]:

	age	anaemia	creatinine_phosphokinase	diabetes	ejection_fraction	high_blood_pressure	platelets	serum_creatinine	serum_sodium	sex	smoking	time
0	75.0	0	582	0	20	1	265000.00	1.9	130	1	0	4
1	55.0	0	7861	0	38	0	263358.03	1.1	136	1	0	6
2	65.0	0	146	0	20	0	162000.00	1.3	129	1	1	7
3	50.0	1	111	0	20	0	210000.00	1.9	137	1	0	7
4	65.0	1	160	1	20	0	327000.00	2.7	116	0	0	8

Then we will write,

`Df.info()`, by this we can get information of dataset that how many int, float, category, numerical are present in the dataset. This will look like

```
In [26]: 1 df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 299 entries, 0 to 298
Data columns (total 13 columns):
 #   Column                    Non Null Count  Dtype
---  ------                    --------------  -----
 0   age                       299 non-null    float64
 1   anaemia                   299 non-null    int64
 2   creatinine_phosphokinase  299 non-null    int64
 3   diabetes                  299 non-null    int64
 4   ejection_fraction         299 non-null    int64
 5   high_blood_pressure       299 non-null    int64
 6   platelets                 299 non-null    float64
 7   serum_creatinine          299 non-null    float64
 8   serum_sodium              299 non-null    int64
 9   sex                       299 non-null    int64
 10  smoking                   299 non-null    int64
 11  time                      299 non-null    int64
 12  DEATH_EVENT               299 non-null    int64
dtypes: float64(3), int64(10)
memory usage: 30.5 KB

In [27]: 1 df.isnull().sum()
```

After this we will write *df.isnull().sum()* to find the null values in dataset. Here we got no null values. This looks as,

```
In [27]: 1 df.isnull().sum()

Out[27]: age                       0
         anaemia                   0
         creatinine_phosphokinase  0
         diabetes                  0
         ejection_fraction         0
         high_blood_pressure       0
         platelets                 0
         serum_creatinine          0
         serum_sodium              0
         sex                       0
         smoking                   0
         time                      0
         DEATH_EVENT               0
         dtype: int64
```

After this we will write df.describe for the statistical purpose. This will looks as,

```
In [28]: 1 df.describe()
```

Out[28]:

	age	anaemia	creatinine_phosphokinase	diabetes	ejection_fraction	high_blood_pressure	platelets	serum_creatinine	serum_sodium
count	299.000000	299.000000	299.000000	299.000000	299.000000	299.000000	299.000000	299.00000	299.000000
mean	60.833893	0.431438	581.839465	0.418060	38.083612	0.351171	263358.029264	1.39388	136.625418
std	11.894809	0.496107	970.287881	0.494067	11.834841	0.478136	97804.236869	1.03451	4.412477
min	40.000000	0.000000	23.000000	0.000000	14.000000	0.000000	25100.000000	0.50000	113.000000
25%	51.000000	0.000000	116.500000	0.000000	30.000000	0.000000	212500.000000	0.90000	134.000000
50%	60.000000	0.000000	250.000000	0.000000	38.000000	0.000000	262000.000000	1.10000	137.000000
75%	70.000000	1.000000	582.000000	1.000000	45.000000	1.000000	303500.000000	1.40000	140.000000
max	95.000000	1.000000	7861.000000	1.000000	80.000000	1.000000	850000.000000	9.40000	148.000000

In this stage we will rename our DEATH_EVENT column into Target(not compulsory) by writing *heart=df.rename(columns = {'DEATH_EVENT':'Target'})*

Whatever the output will be it will be stored in 'heart'. Our dataframe will look as,

```
In [19]: 1 heart=df.rename(columns = {'DEATH_EVENT':'Target'})

In [20]: 1 heart.head()
```

Out[20]:

	age	anaemia	creatinine_phosphokinase	diabetes	ejection_fraction	high_blood_pressure	platelets	serum_creatinine	serum_sodium	sex	smoking	time
0	75.0	0	582	0	20	1	265000.00	1.9	130	1	0	4
1	55.0	0	7861	0	38	0	263358.03	1.1	136	1	0	6
2	65.0	0	146	0	20	0	162000.00	1.3	129	1	1	7
3	50.0	1	111	0	20	0	210000.00	1.9	137	1	0	7
4	65.0	1	160	1	20	0	327000.00	2.7	116	0	0	8

```
In [21]: 1 heart.shape

Out[21]: (299, 13)
```

Here we have also saw shape by writing *heart.shape*.

(e) Explore and preprocess data:

In this part we will first see the structure of data. For this various measures we have to take like data reduction, data cleaning, look for the null values, clean unnecessary row or column.

Here we will devide dataframes column into x and y by writing,

x=heart[['age','ejection_fraction','serum_creatinine','serum_sodium','time']]

y=heart['Target']

(f) EDA:

EDA is short form of Exploratory Data Analysis.

Here we write

import seaborn as sns (to make a heat map and correlation map)

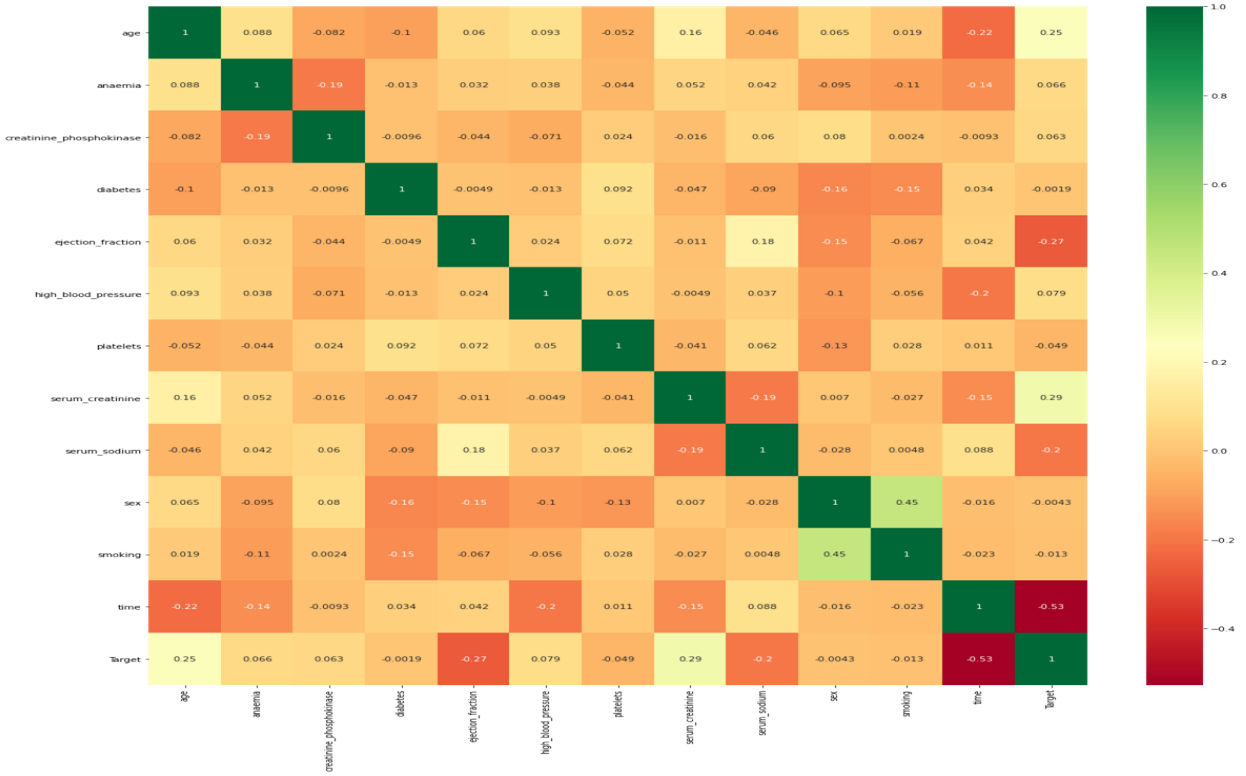
corrmat=heart.corr()

top_corr_features=corrmat.index

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

g=sns.heatmap(heart[top_corr_features].corr(),annot=True,cmap='RdYlGn')

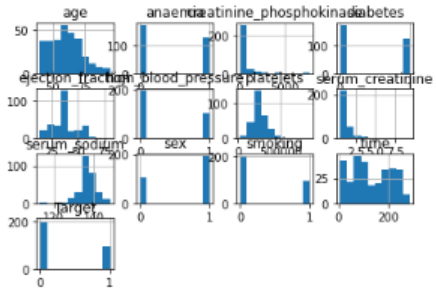
The heatmap will looks as,



Further we can see histogram plot by writing *heart.hist()*. We can get the histogram of every column. It will look as,

```
In [23]: 1 heart.hist()

Out[23]: array([[<AxesSubplot:title={'center':'age'}>,
                 <AxesSubplot:title={'center':'anaemia'}>,
                 <AxesSubplot:title={'center':'creatinine_phosphokinase'}>,
                 <AxesSubplot:title={'center':'diabetes'}>],
                [<AxesSubplot:title={'center':'ejection_fraction'}>,
                 <AxesSubplot:title={'center':'high_blood_pressure'}>,
                 <AxesSubplot:title={'center':'platelets'}>,
                 <AxesSubplot:title={'center':'serum_creatinine'}>],
                [<AxesSubplot:title={'center':'serum_sodium'}>,
                 <AxesSubplot:title={'center':'sex'}>,
                 <AxesSubplot:title={'center':'smoking'}>,
                 <AxesSubplot:title={'center':'time'}>],
                [<AxesSubplot:title={'center':'Target'}>, <AxesSubplot:>,
                 <AxesSubplot:>, <AxesSubplot:>]], dtype=object)
```



(g) Data Preprocessing:

In this section we will split the dataframe and for that we have to first import train_test_split by writing,

from sklearn.model_selection import train_test_split

(h) Create Model:

By looking the dataset we can identify that the dataset is of classification. We can try many Algorithm like

(1) K-Nearest Neighbor
(2) Logistic Regression
(3) Support Vector Machine (12)

(4) Decision tree
(5) Random forest
(6) Naive Bayes
(7) Gradient Boosting

Here we will apply only KNN and Logistic regression.

- In **KNN** we will split x and y into two set say x1_train, y1_train , x1_test, y1_test. Our model will be trained on training data and we will check accuracy on testing data. We have to write a code for this.

```
x1_train, x1_test, y1_train, y1_test =
train_test_split(x1,y1,test_size=0.3,random_state=16)
```

Here we have split training data and testing data in 70:30 ratio.

After that we can see the shape of this data by writing

```
x1_train.shape, x1_test.shape, y1_train.shape,
y1_test.shape
```

For solving it with KNN we have to first import KNN by writing

```
from sklearn.neighbors import KNeighborsClassifier
```

then we have to write

```
from sklearn.model_selection import cross_val_score

knn_scores=[]

for k in range(1,35):

    knn_classifier=KNeighborsClassifier(n_neighbors = k)


score=cross_val_score(knn_classifier,x1_train,y1_train,cv=10)

 knn_scores.append(score.mean())
```

Here k is n_neighbors . We have iterate k from 1to 35.And the data is trained on 10 cross validation. What ever the mean of k we got is use in the following codeings.

plt.plot([k for k in range(1,35)],knn_scores,color='blue')

for i in range (1,35):

plt.text(i,knn_scores[i-1],(i,knn_scores[i-1]))

plt.xticks([i for i in range(1,35)])

plt.xlabel('Number of Neighbors(k)')

plt.ylabel('Score')

plt.title('K Neighbors Classifier score for different k values'.

By this we can get a graph of K. And from that we can find the highest value of K for 35 itterators. The graph can be seen as,

After this we can fit our model and test the accuracy. Here we have got the accuracy of trained model of `88.51674641148325%` and the accuracy for test data is `84.44444444444444 %`.

- In **Logistic regression** we have split the x and y into two set x2_train,y2_train, x_2_test, y2_test. Our model will be trained on training data and we will check accuracy on testing data. We have to write a code for this.

```
x2_train, x2_test, y2_train, y2_test =
train_test_split(x2,y2,test_size=0.3,random_
state=42)
```

Here we have split training data and testing data in 70:30 ratio.

After that we can see the shape of this data by writing

```
 x2_train.shape, x2_test.shape,
y2_train.shape, y2_test.shape
```

For solving it with Logistic Regression we have to first import LogisticRegression by writing

```
from sklearn.linear_model import LogisticRegression
```

Then we have to write

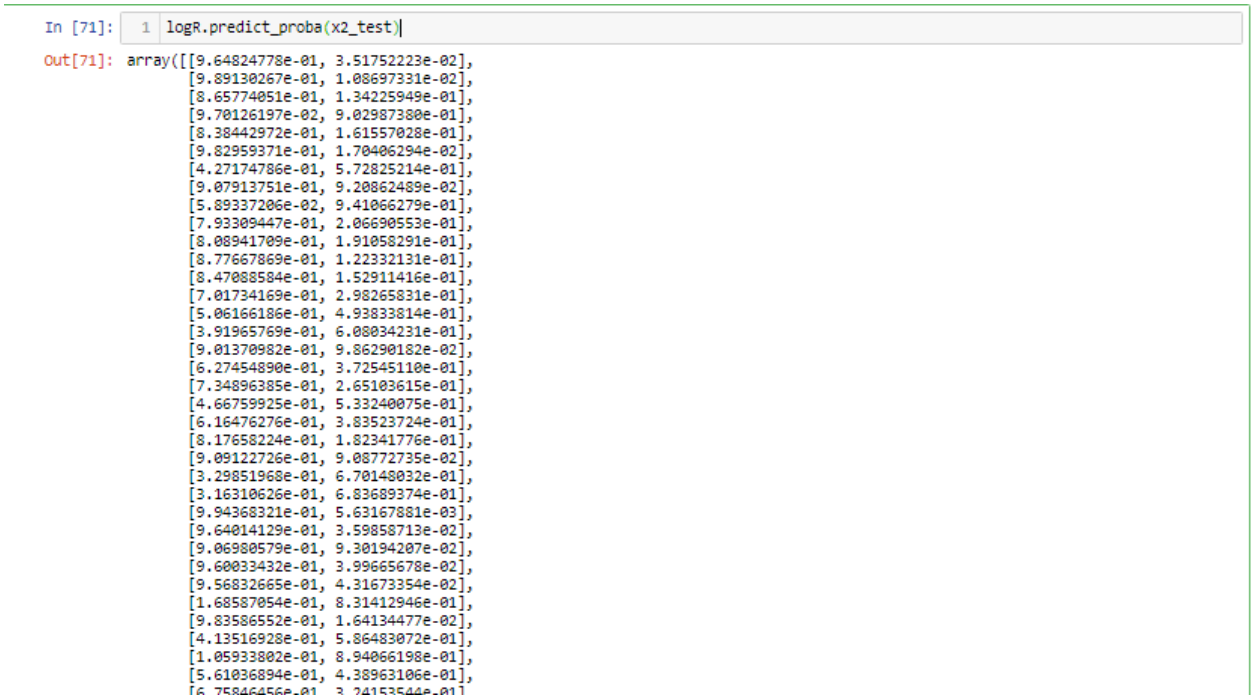
```
logR=LogisticRegression(max_iter=500).fit(x2_train,y2_train)
```

By writing this line we will iterate Logisic Regression upto max 500, and then fit X_train and y2_train and assign this value into logR.

Then we can predict x2_test by writing

logR.predict_proba(x2_test)

by this we can get prediction like



```
In [71]: 1 logR.predict_proba(x2_test)

Out[71]: array([[9.64824778e-01, 3.51752223e-02],
                [9.89130267e-01, 1.08697331e-02],
                [8.65774051e-01, 1.34225949e-01],
                [9.70126197e-02, 9.02987380e-01],
                [8.38442972e-01, 1.61557028e-01],
                [9.82959371e-01, 1.70406294e-02],
                [4.27174786e-01, 5.72825214e-01],
                [9.07913751e-01, 9.20862489e-02],
                [5.89337206e-02, 9.41066279e-01],
                [7.93309447e-01, 2.06690553e-01],
                [8.08941709e-01, 1.91058291e-01],
                [8.77667869e-01, 1.22332131e-01],
                [8.47088584e-01, 1.52911416e-01],
                [7.01734169e-01, 2.98265831e-01],
                [5.06166186e-01, 4.93833814e-01],
                [3.91965769e-01, 6.08034231e-01],
                [9.01370982e-01, 9.86290182e-02],
                [6.27454890e-01, 3.72545110e-01],
                [7.34896385e-01, 2.65103615e-01],
                [4.66759925e-01, 5.33240075e-01],
                [6.16476276e-01, 3.83523724e-01],
                [8.17658224e-01, 1.82341776e-01],
                [9.09122726e-01, 9.08772735e-02],
                [3.29851968e-01, 6.70148032e-01],
                [3.16310626e-01, 6.83689374e-01],
                [9.94368321e-01, 5.63167881e-03],
                [9.64014129e-01, 3.59858713e-02],
                [9.06980579e-01, 9.30194207e-02],
                [9.60033432e-01, 3.99665678e-02],
                [9.56832665e-01, 4.31673354e-02],
                [1.68587054e-01, 8.31412946e-01],
                [9.83586552e-01, 1.64134477e-02],
                [4.13516928e-01, 5.86483072e-01],
                [1.05933802e-01, 8.94066198e-01],
                [5.61036894e-01, 4.38963106e-01],
                [6.75846456e-01, 3.24153544e-01]
```

After this we can see the shape by writing

logR.predict_proba(x2_test).shape

we got (90,2)

We can get the intercepted values of y when x=0 by writting

print(logR.intercept_)

we got `[0.00014791]`

We can get the coefficient by writing

```
print(logR.coef_)
```

We got the output as

```
[[ 6.20992583e-02 -4.39899965e-04  1.15028683e-04  1.04163633e-04
-6.33191396e-02 -7.13466661e-04 -1.42386203e-06  8.10204748e-03
 1.16685454e-03 -9.36201134e-04 -7.00102398e-06 -2.11995571e-02]]
```

Now we have to predict our x2_test data, for this we have to write

```
pred=logR.predict(x2_test)
```

The output got is

```
array([0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1,
0, 0,
       0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1,
0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0,
1, 1,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0,
1, 1,
       1, 0], dtype=int64)
```

Now we can make a table of actual vs predict by writing

```
df1 = pd.DataFrame({'actual': y2_test, 'predictions':
pred})
```

The first 10 row we got is as,

```
In [76]:  1 df1 = pd.DataFrame({'actual': y2_test, 'predictions': pred})
          2 df1.head(10)
```

Out[76]:

	actual	predictions
281	0	0
265	0	0
164	1	0
9	1	1
77	0	0
278	0	0
93	1	1
109	0	0
5	1	1
173	0	0

We have to import confussion matrix by writing

```
from sklearn.metrics import confusion_matrix
```

This is usefull to count the correct and uncorrect prediction.

By writing

```
confusion_matrix(y2_test,pred)
```

we will get the output

```
array([[50,  3],
       [15, 22]], dtype=int64)
```

Now the accuracy we can find by writing

```
LogR_accuracy=(sm.accuracy_score(y2_test,pred))*100
```

We get `84.4444%` accuracy.

So our model is completed now.

Reference

1. ‘ Krish Naik’ YouTube channel
2. ‘Khan Academy’ Youtube Channel
3. Data Science handbook by ‘Jake Vender Plas’
4. 5 Minute Engineering Channel