HEALTH CARE ANALYTICS BIT

AGENDA:

1. Data exploration and pre-processing
2. Perform mean, median and mode for dataset
3. Built the model to perform clustering using K-means
4. Implement linear and logistic regression
5. Implement K-NN algorithm to classify a dataset

1. DATA EXPLORATION AND PRE-PROCESSING

AIM:

Write the command for upload, read and display the dataset in colab using python language.

DATASET TITLE:

Heart Failure Prediction

DATASET DESCRIPTION:

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. We chose this dataset to predict the heart functioning by their clinical records.

WEBSITE DESCRIPTION:

We have chosen this dataset from “Kaggle.com”. Kaggle is an online community platform for data scientists and machine learning enthusiasts. Kaggle allows users to collaborate with other users, find and publish datasets, use GPU integrated notebooks, and compete with other data scientists to solve data science challenges.

PROCEDURE:

Step 1: Search the dataset from the kaggle.com

Step 2: Download the dataset in zip format and extract into .cvs format

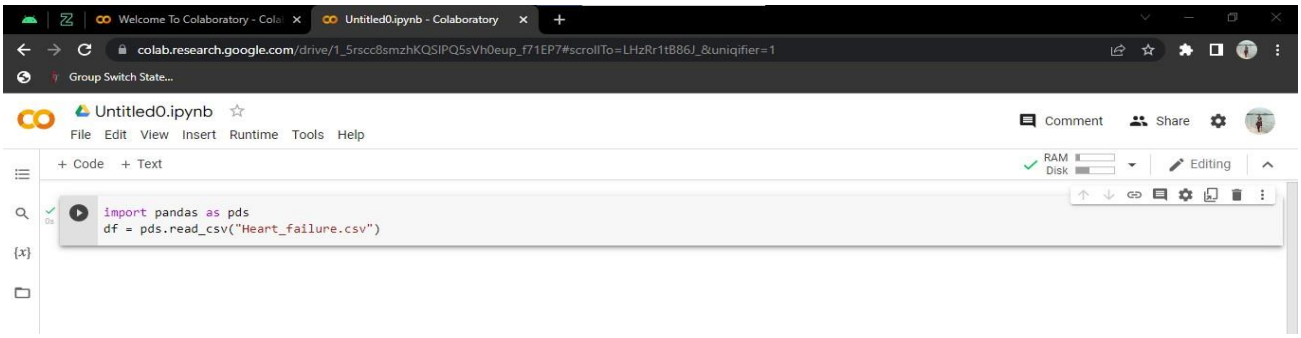
Step3: Create user account in Colab

Step 4: Import and upload the files in the colab

Step 5: Write the command to read, display the data frame

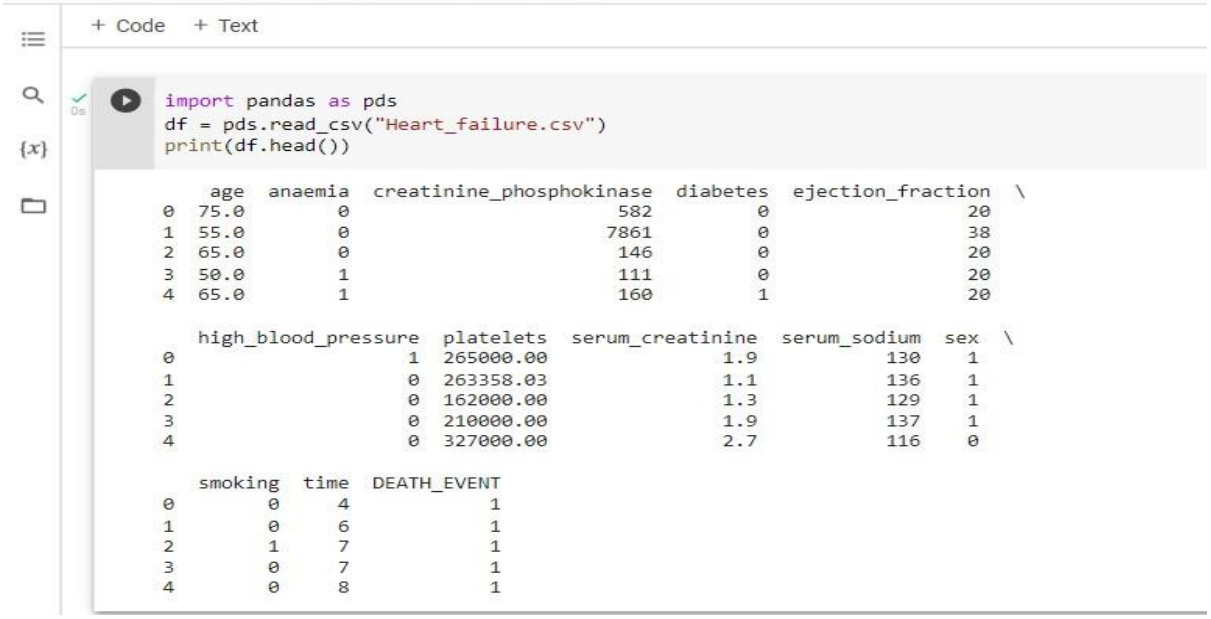
OUTCOME:

Read the dataset in the code lab



Display the dataset in the code lab

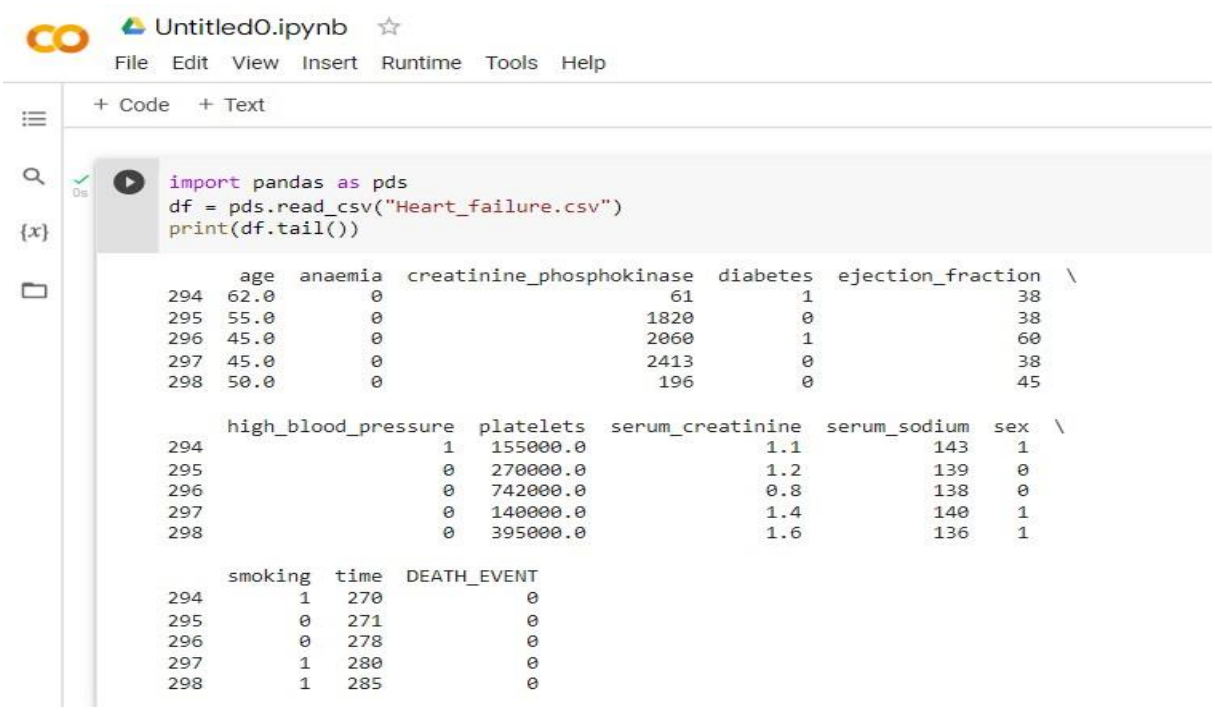
Head command – used to display the first 5 rows in the data frames



```
+ Code  + Text

import pandas as pds
df = pds.read_csv("Heart_failure.csv")
print(df.head())
```

```
    age  anaemia  creatinine_phosphokinase  diabetes  ejection_fraction  \
0  75.0        0                       582         0                 20
1  55.0        0                      7861         0                 38
2  65.0        0                       146         0                 20
3  50.0        1                       111         0                 20
4  65.0        1                       160         1                 20

   high_blood_pressure  platelets  serum_creatinine  serum_sodium  sex  \
0                    1  265000.00               1.9           130    1
1                    0  263358.03               1.1           136    1
2                    0  162000.00               1.3           129    1
3                    0  210000.00               1.9           137    1
4                    0  327000.00               2.7           116    0

   smoking  time  DEATH_EVENT
0        0     4            1
1        0     6            1
2        1     7            1
3        0     7            1
4        0     8            1
```

Tail command – used to display the last 5 rows in the data frames



```
Untitled0.ipynb
File  Edit  View  Insert  Runtime  Tools  Help

+ Code  + Text

import pandas as pds
df = pds.read_csv("Heart_failure.csv")
print(df.tail())
```

```
     age  anaemia  creatinine_phosphokinase  diabetes  ejection_fraction  \
294  62.0        0                        61         1                 38
295  55.0        0                      1820         0                 38
296  45.0        0                      2060         1                 60
297  45.0        0                      2413         0                 38
298  50.0        0                       196         0                 45

     high_blood_pressure  platelets  serum_creatinine  serum_sodium  sex  \
294                    1   155000.0               1.1           143    1
295                    0   270000.0               1.2           139    0
296                    0   742000.0               0.8           138    0
297                    0   140000.0               1.4           140    1
298                    0   395000.0               1.6           136    1

     smoking  time  DEATH_EVENT
294        1   270            0
295        0   271            0
296        0   278            0
297        1   280            0
298        1   285            0
```

Describe command – used to display the basic details

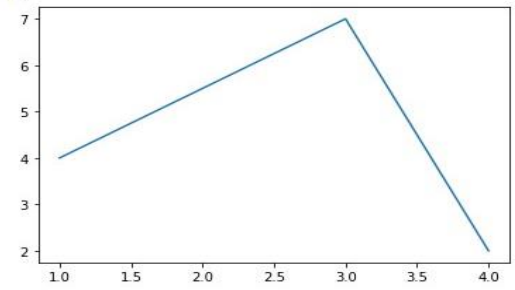
```
import pandas as pds
df = pds.read_csv("Heart_failure.csv")
print(df.describe())
```

```
              age     anaemia  creatinine_phosphokinase    diabetes  \
count  299.000000  299.000000                299.000000  299.000000
mean    60.833893    0.431438                581.839465    0.418060
std     11.894809    0.496107                970.287881    0.494057
min     40.000000    0.000000                 23.000000    0.000000
25%     51.000000    0.000000                116.500000    0.000000
50%     60.000000    0.000000                250.000000    0.000000
75%     70.000000    1.000000                582.000000    1.000000
max     95.000000    1.000000               7861.000000    1.000000

       ejection_fraction  high_blood_pressure      platelets  \
count         299.000000           299.000000     299.000000
mean           38.083612             0.351171  263358.029264
std            11.834841             0.478136   97804.236869
min            14.000000             0.000000   25100.000000
25%            30.000000             0.000000  212500.000000
50%            38.000000             0.000000  262000.000000
75%            45.000000             1.000000  303500.000000
max            80.000000             1.000000  850000.000000

       serum_creatinine  serum_sodium         sex     smoking        time  \
count        299.00000    299.000000  299.000000  299.00000  299.000000
mean           1.39388    136.625418    0.648829    0.32107  130.260870
std            1.03451      4.412477    0.478136    0.46767   77.614208
min            0.50000    113.000000    0.000000    0.00000    4.000000
25%            0.90000    134.000000    0.000000    0.00000   73.000000
50%            1.10000    137.000000    1.000000    0.00000  115.000000
75%            1.40000    140.000000    1.000000    1.00000  203.000000
max            9.40000    148.000000    1.000000    1.00000  285.000000

       DEATH_EVENT
count   299.00000
mean      0.32107
std       0.46767
min       0.00000
25%       0.00000
50%       0.00000
75%       1.00000
max       1.00000
```

Plot Command – used to display the line graph

```
import matplotlib.pyplot as plt
x1 = [1, 3, 4]
y1 = [4, 7, 2]
plt.plot(x1, y1)
plt.plot()
```

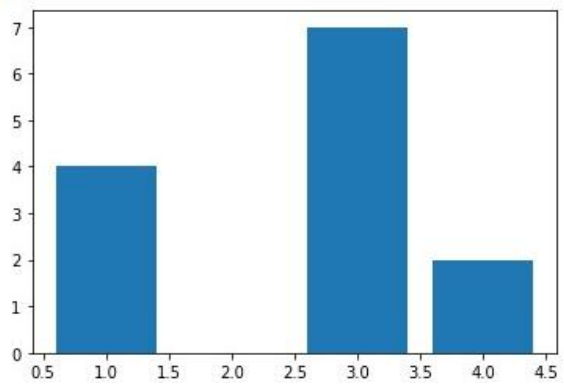
```
[]
```



Bar command – used to display the bar graph

```
import matplotlib.pyplot as plt
x1 = [1, 3, 4]
y1 = [4, 7, 2]
plt.bar(x1, y1)
plt.plot()
```

```
[]
```



SUMMARY REPORT:

By completing the Task - 1 we learn how to use a colab platform, By using colab platform, we add some basic commands to upload, display and delete in the dataset on colab platform. In colab platform there are many basic commands are available to use and modify the dataset on colab. This colab platform is used to write a commands in python language. In colab platform we importing the commands in python language to get the exact result of the data frame and we print the data frame in colab platform.

2. PERFORM MEAN, MEDIAN AND MODE FOR DATASET

AIM:

To Calculate the Mean, Median and Mode for the given dataset in the Colab platform using the python language.

PROCEDURE:

Step 1: Collect the data of values from the extracted dataset to calculate the Mean, Median and Mode.

Step 2: Open the Colab platform and write the code in it.

Step 3: Run the code and get the output of Mean, Median and Mode.

IMPLEMENTATION:

MEAN:

```
import numpy as np

import pandas as pd

import os

df=pd.read_csv(r'drive/MyDrive/Colab Notebooks/heart_failure_clinical_records_dataset.csv')

df['age'].mean()
```

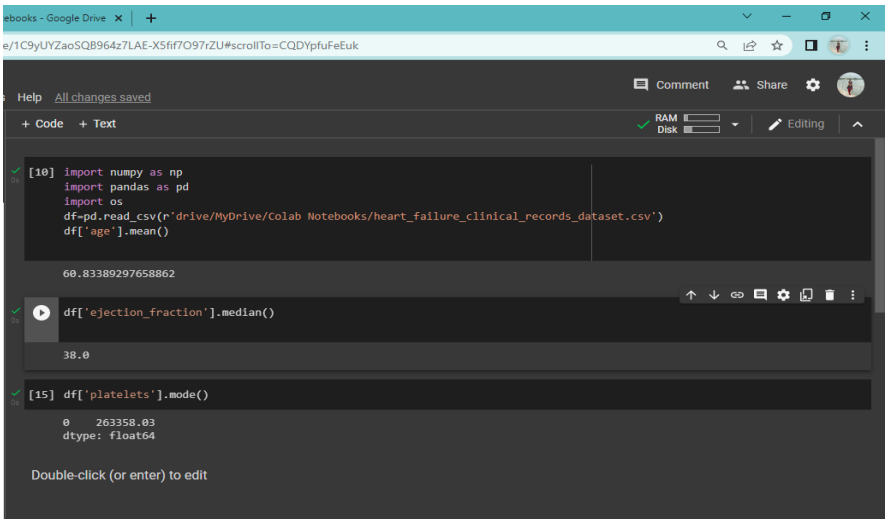
MEDIAN:

```
import numpy as np

import pandas as pd

import os

df=pd.read_csv(r'drive/MyDrive/Colab Notebooks/heart_failure_clinical_records_dataset.csv')

df['ejection_fraction'].median()
```

MODE:

```
import numpy as np

import pandas as pd

import os

df=pd.read_csv(r'drive/MyDrive/Colab Notebooks/heart_failure_clinical_records_dataset.csv')

df['platelets'].mode()
```

OUTCOME:



DESCRIPTION:

In Machine Learning (and in mathematics) there are often three values that interests us:

Mean: The mean is the average of all numbers and is sometimes called the arithmetic mean.

Median: The median is the middle number in a group of numbers.

Mode: The mode is the number that occurs most often within a set of numbers.

RESULT:

Thus we found a values of mean, median and mode for columns in the datasets

3. BUILT THE MODEL TO PERFORM CLUSTERING USING K-MEANS

AIM:

To demonstrate k-means clustering for the given dataset in the colab platform. The dataset we have chosen is heart failure prediction

PROCEDURE:

Step 1: Open the Kaggle platform and download the dataset.

Step 2: Demonstrate k-means clustering using matplotlib library.

Step 3: In the colab platform type the program in the code area.

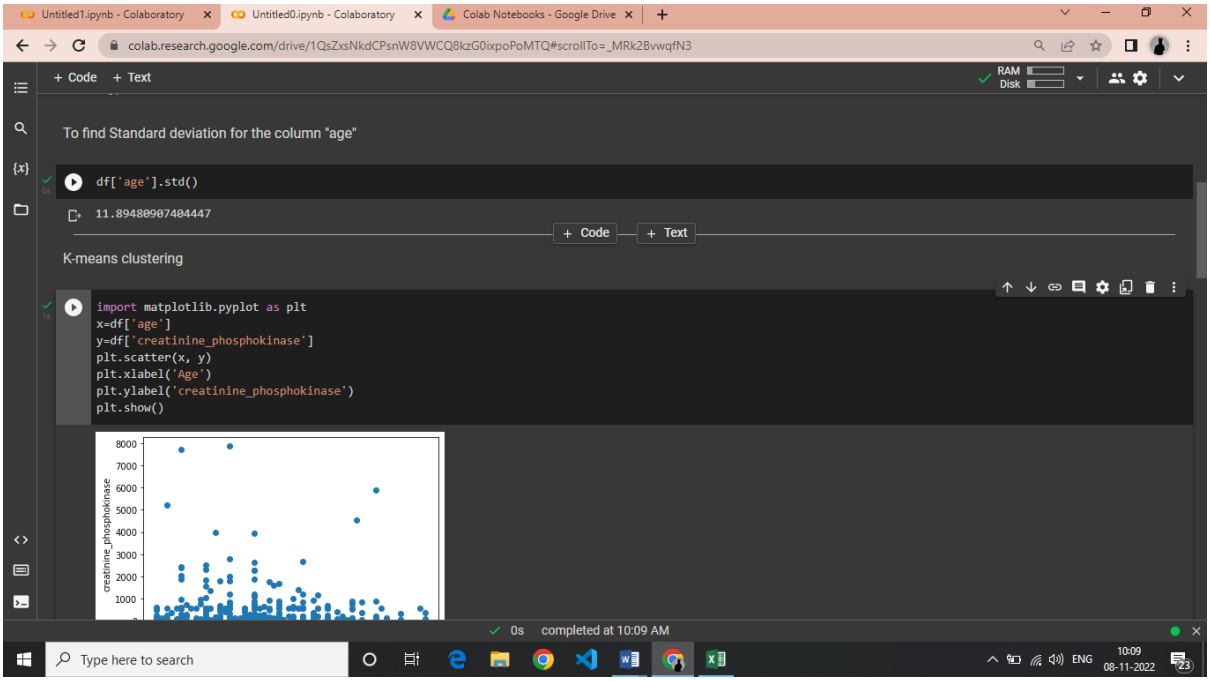
Step 4: Run the code.

Step 5: Take a screenshot of the output

IMPLEMENTATION:

```
import numpy as np
import pandas as pd
import os
df=pd.read_csv(r'drive/MyDrive/Colab Notebooks/ heart_failure_clinical_records_dataset.csv')

import matplotlib.pyplot as plt
x=df['age']
y=df['ceratinine_phosphokinase']
plt.scatter(x, y)
plt.xlabel('Age')
plt.ylabel('ceratinine_phosphokinase')
plt.show()
```

OUTCOME:



DESCRIPTION:

- K-means is an unsupervised learning method for clustering data points. The algorithm iteratively divides data points into K clusters by minimizing the variance in each cluster.
- K-means clustering requires us to select K, the number of clusters we want to group the data into.
- The elbow method lets us graph the inertia (a distance-based metric) and visualize the point at which it starts decreasing linearly. This point is referred to as the “elbow” and is a good estimate for the best value for K based on our data.

RESULT:

Here, we implemented how to estimate the best value for K, then use K-means clustering to group the data points into clusters.

4. IMPLEMENT LINEAR AND LOGISTIC REGRESSION

AIM:

To demonstrate linear and logistics regression for the given dataset in the colab platform. The dataset we have chosen is heart failure prediction.

PROCEDURE:

Step 1: Open the Kaggle platform and download the dataset.

Step 2: To demonstrate linear and logistics regression for the value in the dataset.

Step 3: In the colab platform type the program in the code area.

Step 4: Run the code.

Step 5: Take a screenshot of the output.

IMPLEMENTATION:

For Linear Regression:

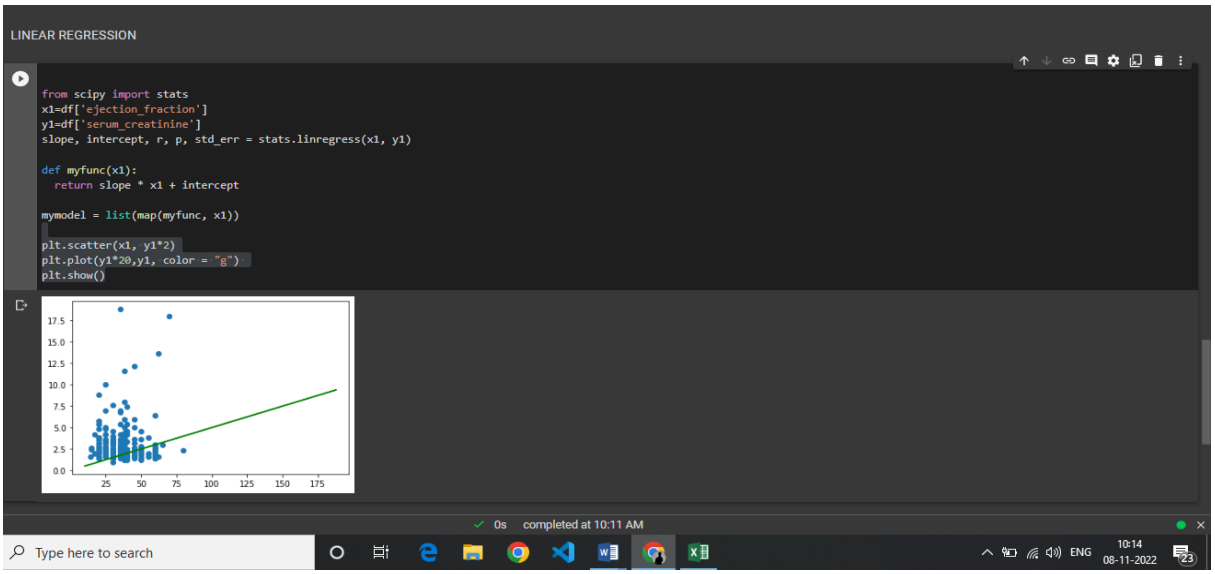
```
from scipy import stats
x1=df[‘ejection_fraction’]
y1=df[‘serum_creatinine’]
slope, intercept, r, p, std_err = stats.linregress(x1, y1)

def myfunc(x1):
  return slope * x1 + intercept

mymodel = list(map(myfunc, x1))

plt.scatter(x1, y1*2)
plt.plot(y1*20,y1, color = “g”)
plt.show()
```

For Logistic Regression:

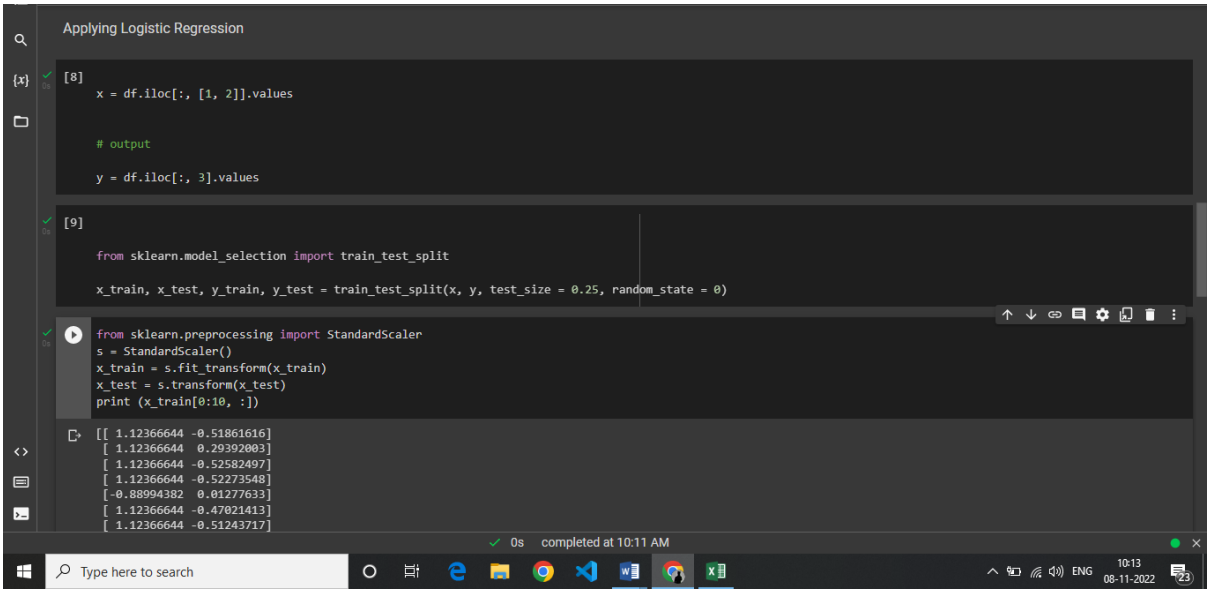
```
import numpy as np
import pandas as pd
import os
df=pd.read_csv(r'drive/MyDrive/Colab Notebooks/heart_failure_clinical_records_dataset.csv')

x = df.iloc[:, [1,2]].values
y = df.iloc[:, 3].values

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 = 0)

from sklearn.preprocessing import StandardScaler
s = StandardScaler()
x_train = s.fit_transform(x_train)
x_test = s.transform(x_test)
print (x_train[0:10, :])

from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0)
classifier.fit(x_train, y_train)

y_pred = classifier.predict(x_train)

from sklearn.metrics import accuracy_score
print("Accuracy : ", accuracy_score(y_train, y_pred))
```

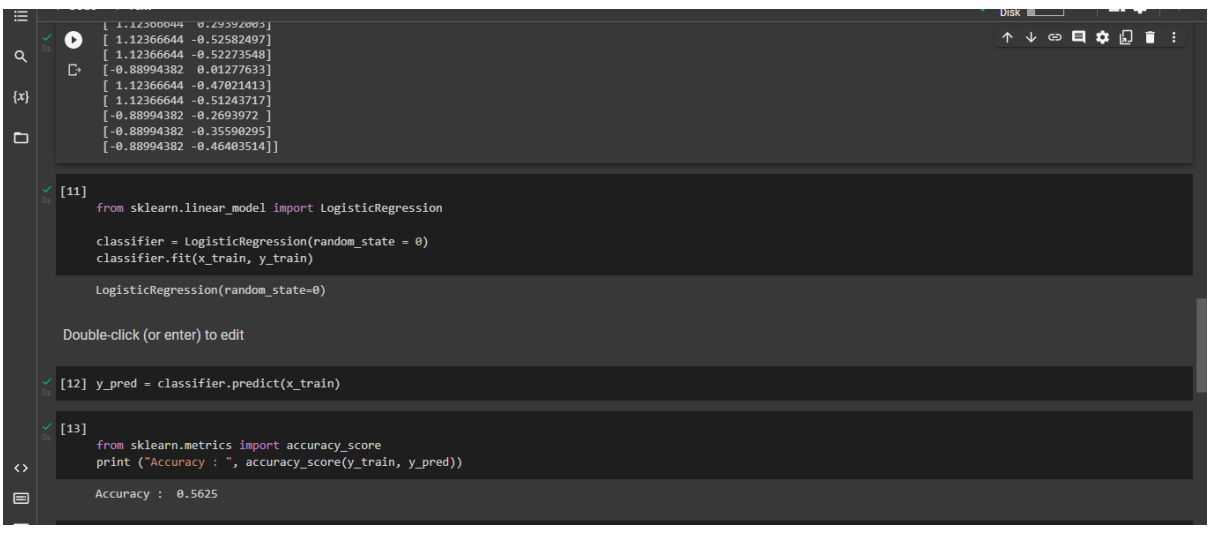
OUTCOME:

Linear Regression



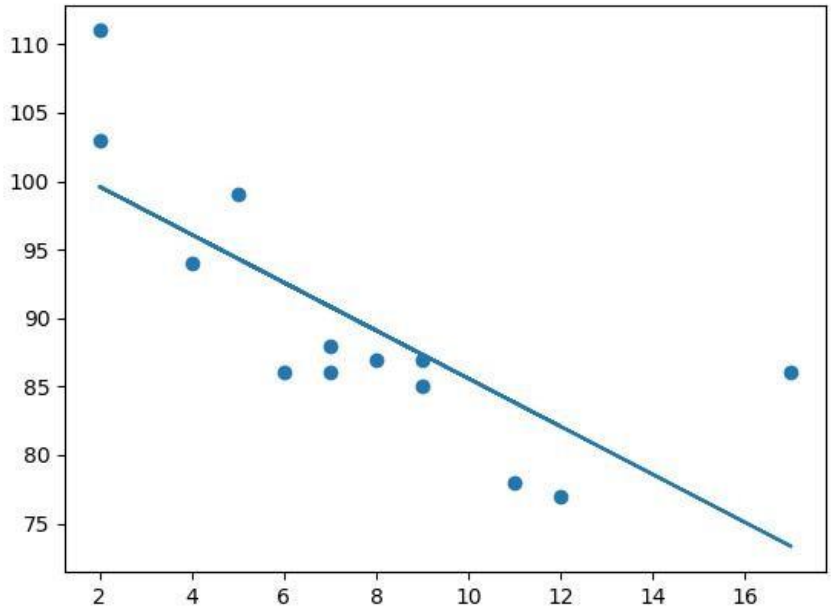
Logistic Regression





DESCRIPTION:

- **Linear regression** is probably one of the most important and widely used regression techniques. It's among the simplest regression methods. One of its main advantages is the ease of interpreting results.



- Python has methods for finding a relationship between data-points and to draw a line of linear regression. We will show you how to use these methods instead of going through the mathematic formula.

- **Logistic regression** is a fundamental classification technique. It belongs to the group of linear classifiers and is somewhat similar to polynomial and linear regression.

- **Logistic regression** is fast and relatively uncomplicated, and it's convenient for you to interpret the results.

- Other cases have more than two outcomes to classify, in this case it is called multinomial. A common example for multinomial logistic regression would be predicting the class of an iris flower between 3 different species.

RESULT:

Here, we implemented how to classify the result with the help of logistics and their regression technique for the particular trained dataset.

5. IMPLEMENT K-NN ALGORITHM TO CLASSIFY A DATASET

AIM:

To implement KNN Algorithm to classify the data's from the dataset with the help of our heart failure prediction dataset.

PROCEDURE:

Step 1: Open the Kaggle platform and download the dataset.

Step 2: To demonstrate K-NN algorithm for the value in the dataset.

Step 3: In the colab platform type the program in the code area.

Step 4: Run the code.

Step 5: Take a screenshot of the output.

IMPLEMENTATION:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn import preprocessing
from sklearn.model_selection import train_test_split


data = pd.read_csv(r'drive/MyDrive/Colab
Notebooks/heart_failure_clinical_records_dataset.csv')
print(data.describe());


gen,gen_count=np.unique(data['anaemia'], return_counts=True)
plt.pie(gen_count , labels = gen,autopct='%1.2f%%')
plt.axis('equal')
plt.title("Anaemia Classification", pad=20)
plt.show()

gen,gen_count=np.unique(data['sex'],
```

return_counts=True)

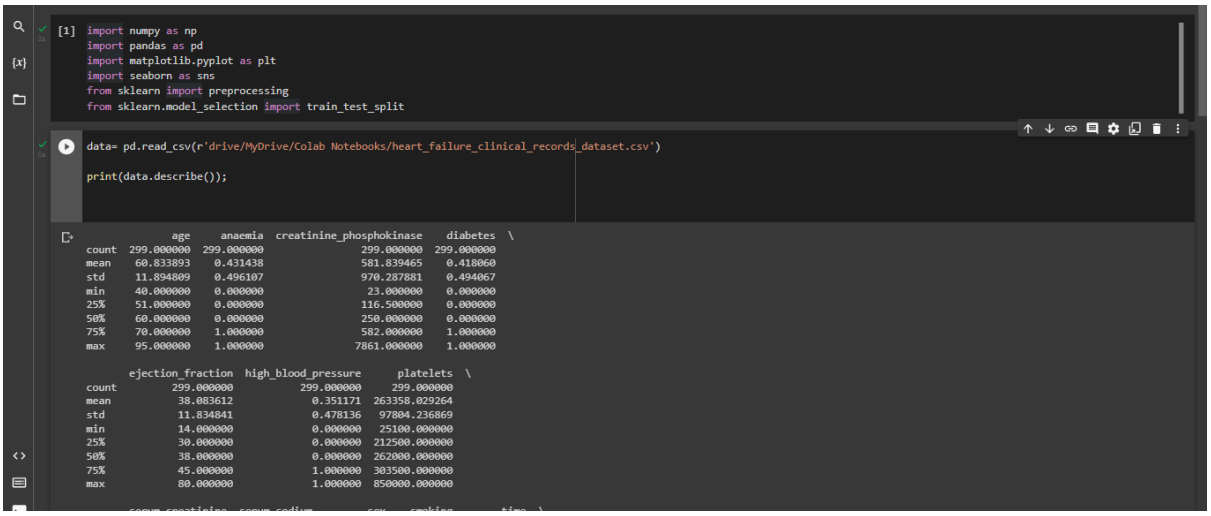
plt.pie(gen_count , labels = gen,autopct='%1.2f%%')

plt.axis('equal')

plt.title("Sex Classification", pad=20)

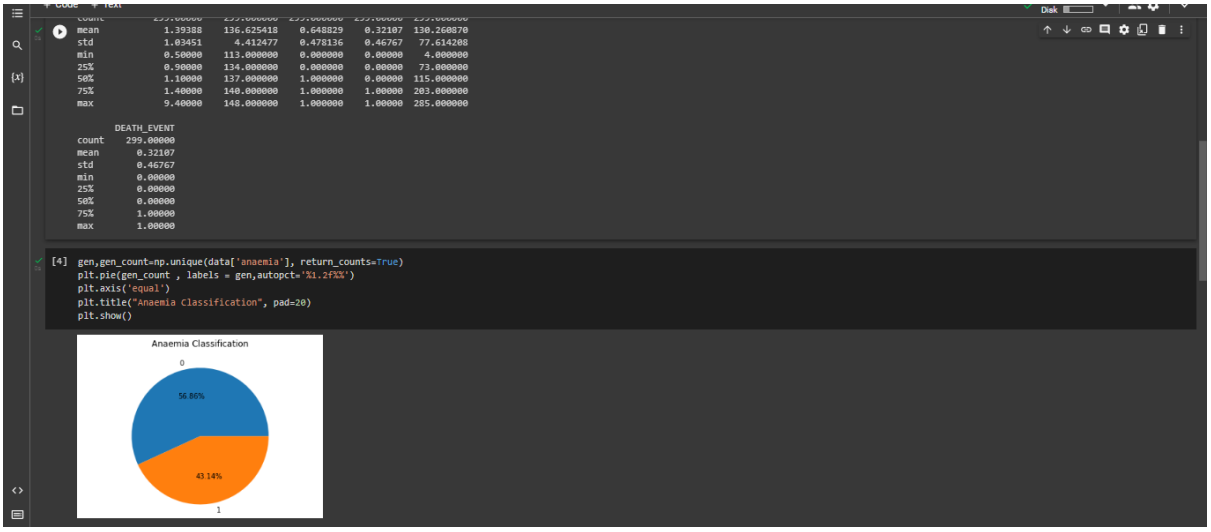
plt.show()

OUTCOME:



```
[1] import numpy as np
    import pandas as pd
    import matplotlib.pyplot as plt
    import seaborn as sns
    from sklearn import preprocessing
    from sklearn.model_selection import train_test_split

    data= pd.read_csv(r'drive/MyDrive/Colab Notebooks/heart_failure_clinical_records_dataset.csv')

    print(data.describe());
```

```
                age     anaemia  creatinine_phosphokinase    diabetes  \
count  299.000000  299.000000                299.000000  299.000000
mean    60.833893    0.431438                581.839465    0.418060
std     11.894809    0.496107                970.287881    0.494067
min     40.000000    0.000000                 23.000000    0.000000
25%     51.000000    0.000000                116.500000    0.000000
50%     60.000000    0.000000                250.000000    0.000000
75%     70.000000    1.000000                582.000000    1.000000
max     95.000000    1.000000               7861.000000    1.000000

       ejection_fraction  high_blood_pressure      platelets  \
count         299.000000           299.000000     299.000000
mean           38.083612             0.351171  263358.029264
std            11.834841             0.478136   97804.236869
min            14.000000             0.000000   25100.000000
25%            30.000000             0.000000  212500.000000
50%            38.000000             0.000000  262000.000000
75%            45.000000             1.000000  303500.000000
max            80.000000             1.000000  850000.000000
```



```
mean          1.39388   136.625418    0.648829    0.32107  130.260870
std           1.03451     4.412477    0.478136    0.46767   77.614208
min           0.50000   113.000000    0.000000    0.00000    4.000000
25%           0.90000   134.000000    0.000000    0.00000   73.000000
50%           1.10000   137.000000    1.000000    0.00000  115.000000
75%           1.40000   140.000000    1.000000    1.00000  203.000000
max           9.40000   148.000000    1.000000    1.00000  285.000000

       DEATH_EVENT
count   299.00000
mean      0.32107
std       0.46767
min       0.00000
25%       0.00000
50%       0.00000
75%       1.00000
max       1.00000
```

```
[4] gen,gen_count=np.unique(data['anaemia'], return_counts=True)
    plt.pie(gen_count , labels = gen,autopct='%1.2f%%')
    plt.axis('equal')
    plt.title("Anaemia Classification", pad=20)
    plt.show()
```


```
gen,gen_count=np.unique(data['sex'], return_counts=True)
plt.pie(gen_count , labels = gen,autopct='%1.2f%%')
plt.axis('equal')
plt.title("Sex Classification", pad=20)
plt.show()
```

DESCRIPTION:

K-nearest neighbors (KNN) algorithm is a type of supervised ML algorithm which used for both classification as well as regression predictive problems.

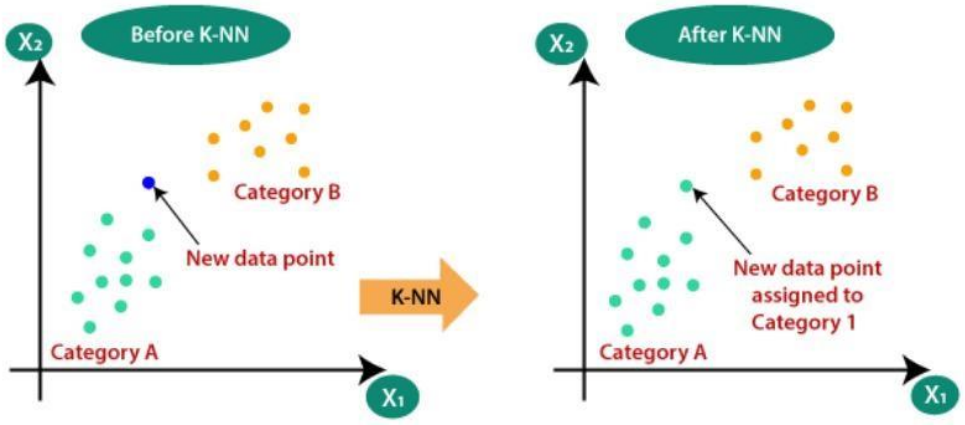
We can understand its working with the help of following steps:

Step 1: For implementing any algorithm, we need dataset. So during the first step of KNN, we must load the training as well as test data. Next, it will choose the top K rows from the sorted array.

Step 2: Next, we need to choose the value of K i.e. the nearest data points.
K can be any integer.

Step 3: Now, it will assign a class to the test point based on most frequent class of these row for each point in the test data do the following

Step 4: End



RESULT:

Here, we implemented how to classify the result with the help of K-NN algorithm technique for the particular trained dataset.