DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)

T.E/SEM VI/REV-2019 “C” SCHEME/CSE-(AI&ML)
Academic Year: 2022-23

NAME	**SINGH SUDHAM DHARMENDRA**
BRANCH	**CSE-(AI&ML)**
ROLL NO.	**AIML57**
SUBJECT	**DATA ANALYTICS AND VISUALIZATION LAB**
COURSE CODE	**CSL601**
PRACTICAL NO.	
DOP	
DOS	

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)

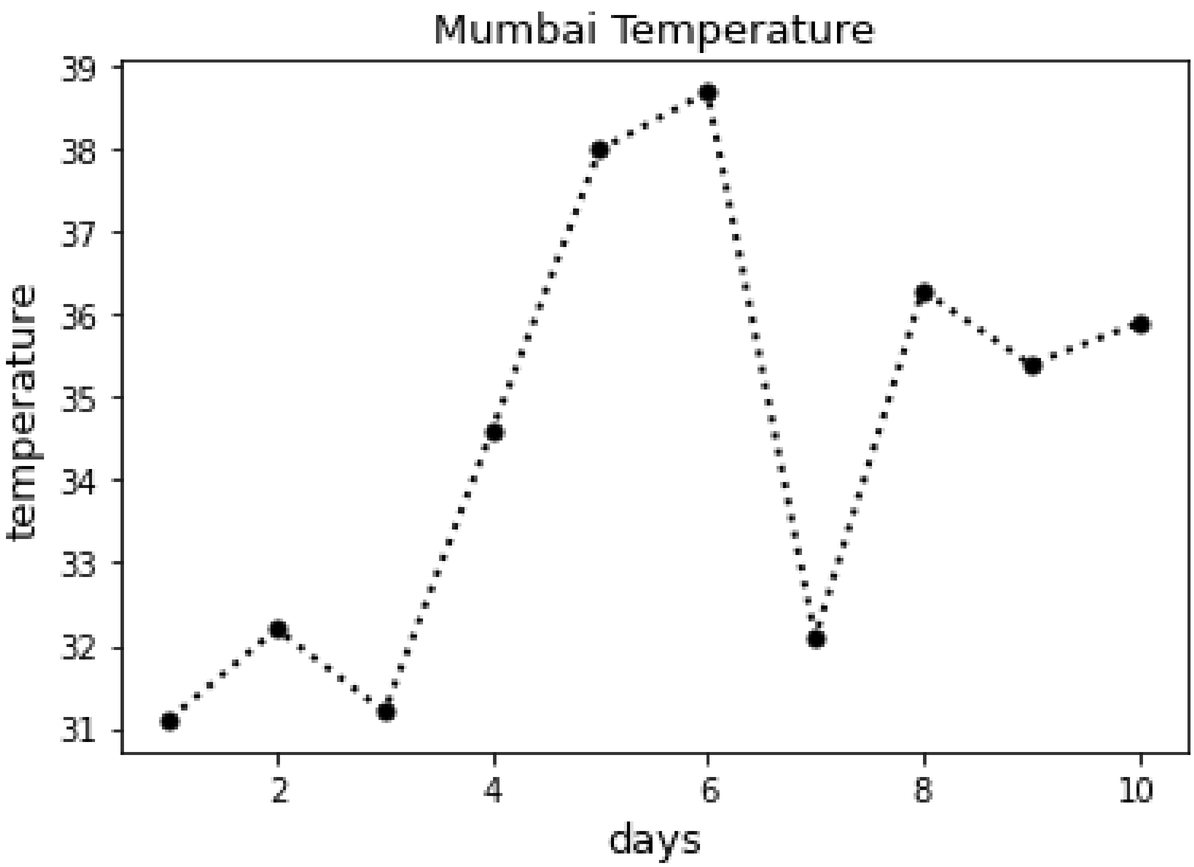
T.E/SEM VI/CBCGS/AIML
Academic Year: 2022-23

NAME	**SINGH SUDHAM DHARMENDRA**
BRANCH	**CSE-(AI&ML)**
ROLL NO.	**57**
SUBJECT	**DATA ANALYTICS AND VISUALIZATION LAB**
COURSE CODE	**CSL601**
PRACTICAL NO.	**01**
DOP	**19/01/2023**
DOS	

Program(input)/Output :

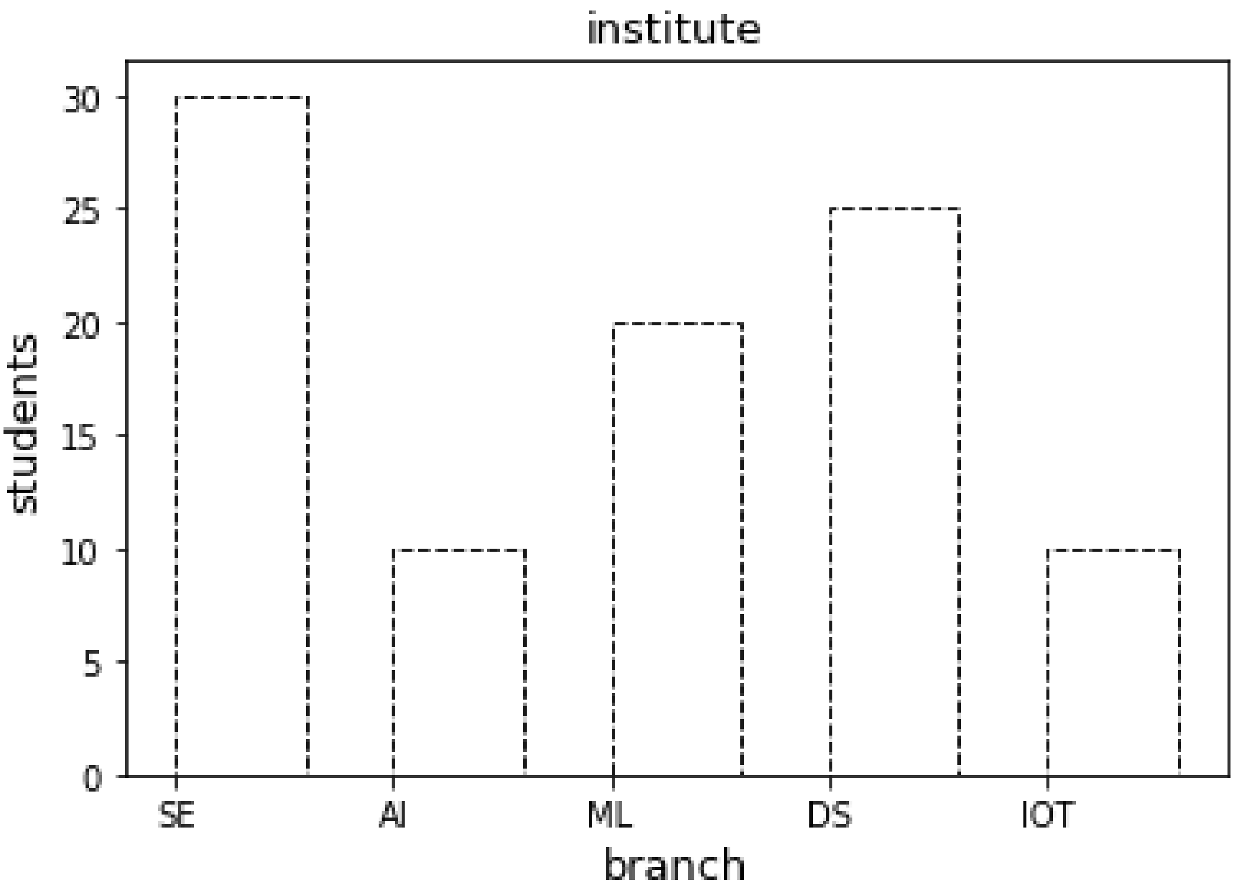
Matplotlib line -

```
import matplotlib.pyplot as plt
days=[1,2,3,4,5,6,7,8,9,10]
temperature=[31.1,32.2,31.2,34.6,38.0,38.7,32.1,36.3,35.4,35.9]
plt.plot(days,temperature,color="k",marker=".",linestyle=":",linewidth=2,markersize=10)
plt.title("Mumbai Temperature")
plt.xlabel("days")
plt.ylabel("temperature")
plt.show()
```



Matplotlib bars -

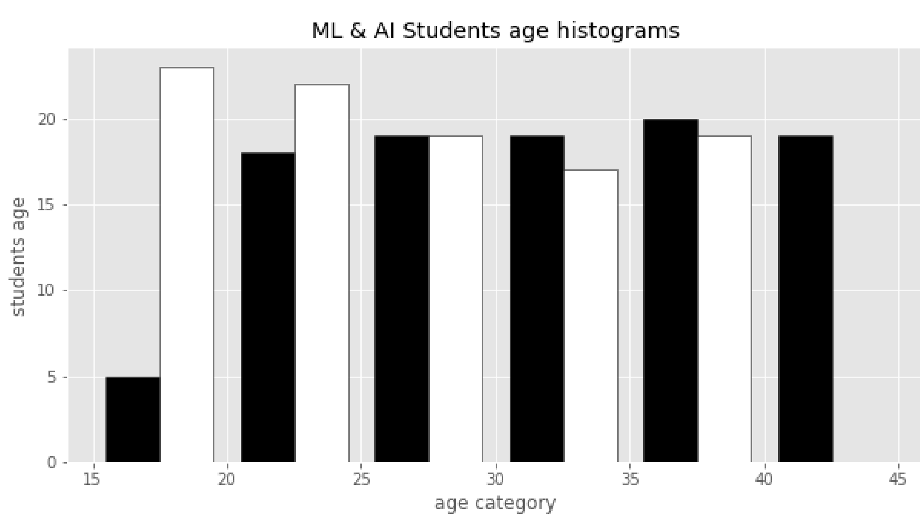
```
import matplotlib.pyplot as plt
import numpy as np
from matplotlib import style
classes=["SE","AI","ML","DS","IOT"]
class_1_students=[30,10,20,25,10]
plt.bar(classes,class_1_students,width=0.6,align="edge",color="w",edgecolor="k",linewi
dth=1,alpha=0.9,linestyle="--",label="Class 1 Students",visible=True)
plt.title("institute",fontsize=13)
plt.xlabel("branch",fontsize=13)
plt.ylabel("students",fontsize=13)
plt.show()
```



Matplotlib histograms -

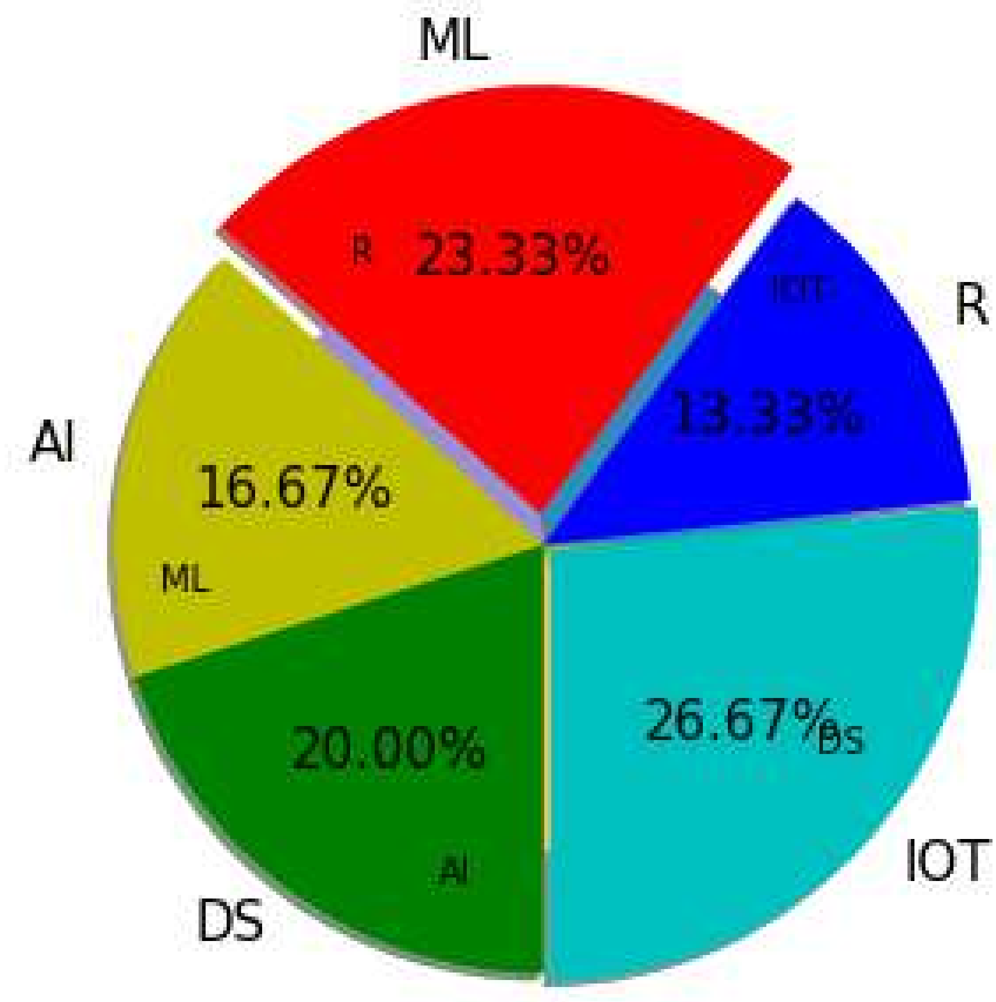
```
import matplotlib.pyplot as plt
import numpy as np
import random
style.use("ggplot")
ml_students_age=np.random.randint(18,45,(100))
ai_students_age=np.random.randint(15,40,(100))
print(ml_students_age)
print(ai_students_age)
bins = [15,20,25,30,35,40,45]
plt.figure(figsize = (10,5))
plt.hist([ml_students_age,ai_students_age],bins,rwidth=0.8,histtype="bar",orientation='v
ertical',color=["k","w"],edgecolor="k",label=["ML Student","AI Student"])
plt.title("ML & AI Students age histograms")
plt.xlabel("age category")
plt.ylabel("students age")
plt.show()
```

```
[31 22 34 22 29 26 40 38 20 41 33 44 32 22 29 29 42 40 20 44 20 28 29 19
 37 37 26 21 31 34 19 35 23 38 20 21 43 31 23 31 25 43 39 19 44 24 33 39
 37 26 44 33 26 26 34 19 36 41 28 30 37 40 19 37 32 23 30 40 21 27 37 37
 29 31 39 35 37 29 44 22 23 32 43 28 41 31 44 40 26 37 33 28 38 31 20 22
 35 40 27 36]
[31 18 39 15 32 23 28 22 37 32 16 35 18 24 15 20 33 36 26 27 31 29 29 32
 20 35 28 39 24 17 26 29 24 38 37 36 24 21 33 34 36 25 25 18 38 17 16 28
 18 20 18 35 26 34 15 24 21 17 36 39 32 27 28 18 33 33 33 25 24 37 27 32
 20 34 23 34 37 37 16 24 16 35 22 17 16 16 26 27 22 16 29 24 24 17 18 32
 24 18 23 39]
```



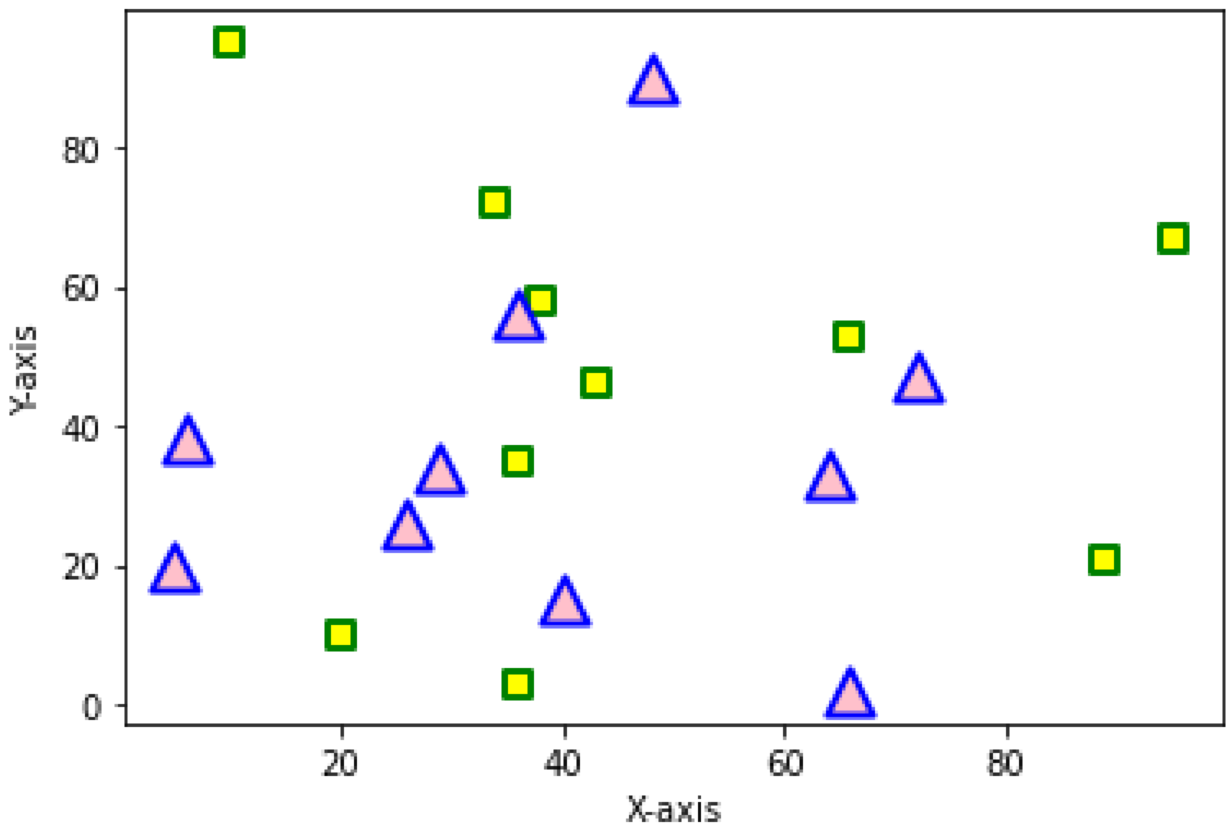
Matplotlib pie-chart -

```
import matplotlib.pyplot as plt
plt.pie([1])
classes = ["IOT",'R','ML', 'AI', 'DS']
class1_students = [40, 20, 35, 25, 30]
plt.pie(class1_students, labels = classes)
explode = [0.03,0,0.1,0,0]
colors = ["c", 'b','r','y','g']
textprops = {"fontsize":15}
plt.pie(class1_students,labels = classes, explode = explode, colors =colors, autopct =
"%0.2f%%", shadow = True, radius = 1.4,startangle = 270,textprops =textprops)
plt.show()
```



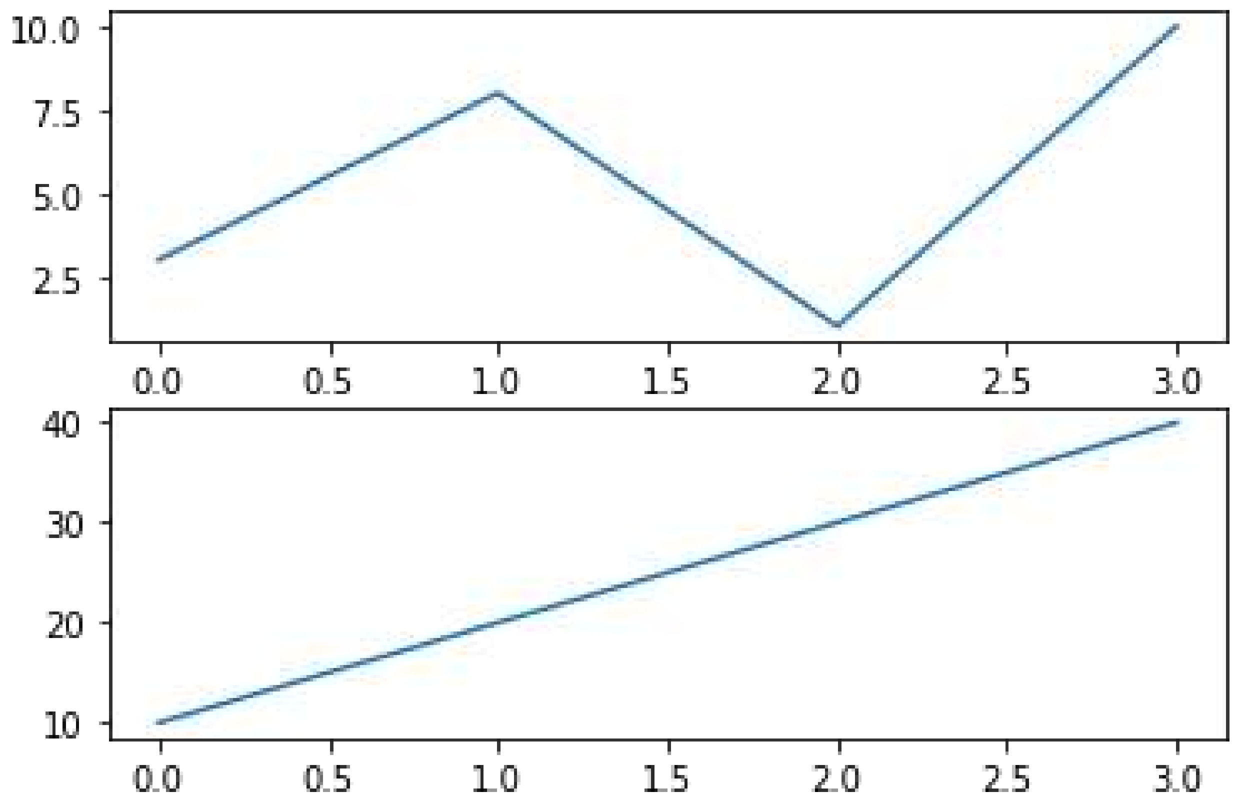
Matplotlib scatter -

```
import matplotlib.pyplot as plt
x1 = [89, 43, 36, 36, 95, 10,66, 34, 38, 20]
y1 = [21, 46, 3, 35, 67, 95,53, 72, 58, 10]
x2 = [26, 29, 48, 64, 6, 5,36, 66, 72, 40]
y2 = [26, 34, 90, 33, 38,20, 56, 2, 47, 15]
plt.scatter(x1, y1, c ="yellow",linewidths = 2,marker ="s",edgecolor ="green",s = 50)
plt.scatter(x2, y2, c ="pink",linewidths = 2,marker ="^",edgecolor ="blue",s = 200)
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.show()
```

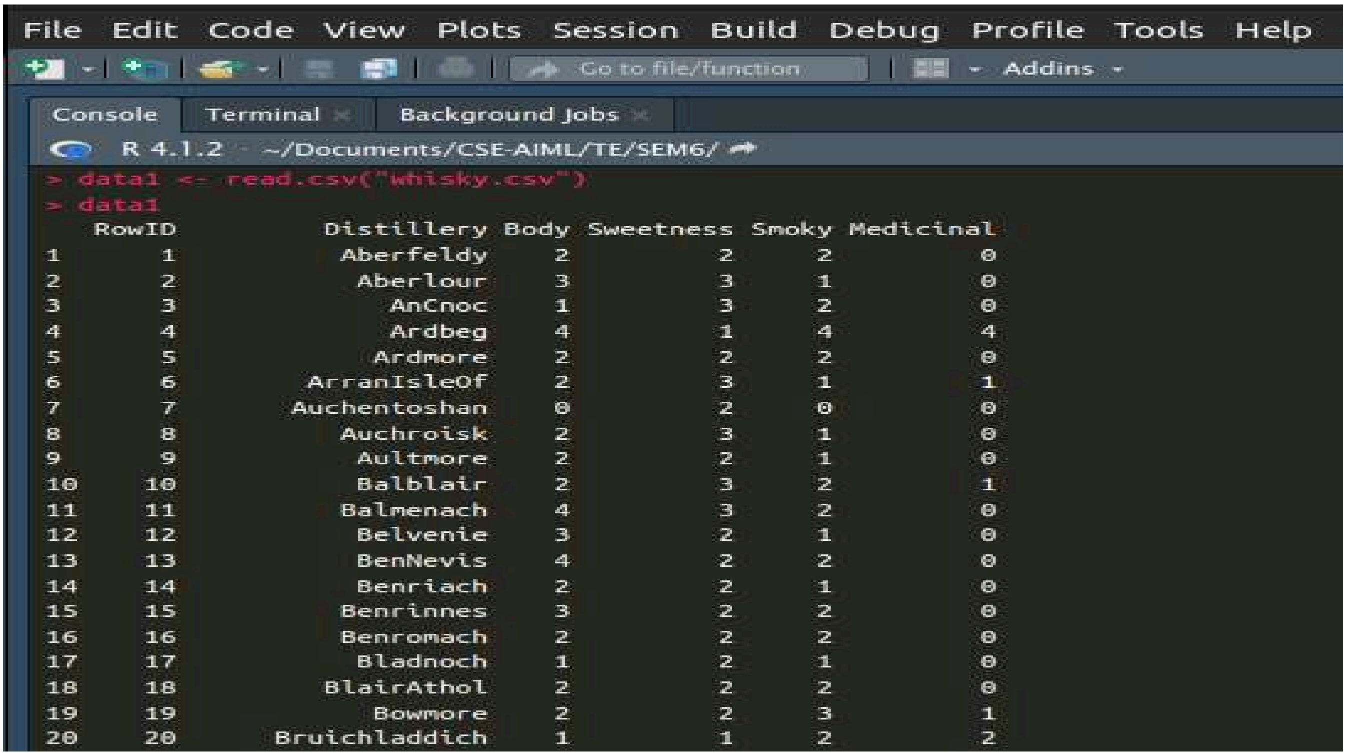


Matplotlib subplot -

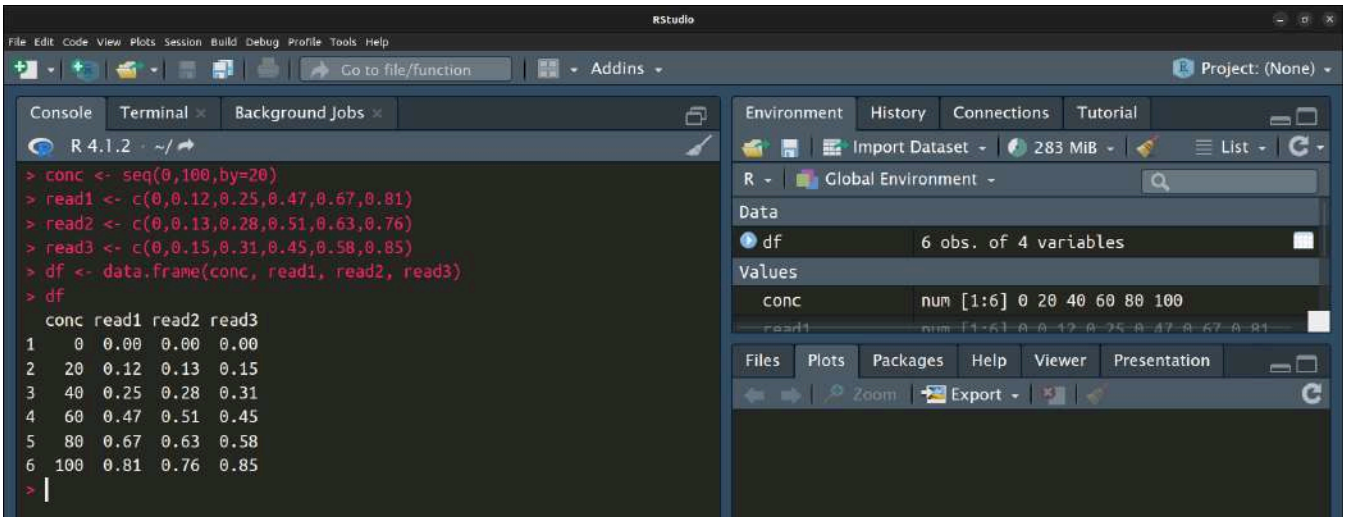
```
import matplotlib.pyplot as plt
import numpy as np
x = np.array([0, 1, 2, 3])
y = np.array([3, 8, 1, 10])
plt.subplot(2, 1, 1)
plt.plot(x,y)
x = np.array([0, 1, 2, 3])
y = np.array([10, 20, 30, 40])
plt.subplot(2, 1, 2)
plt.plot(x,y)
plt.show()
```



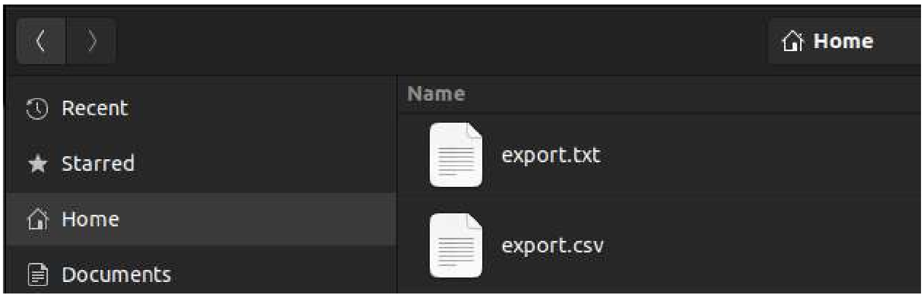
- **Importing dataset in R**



- **Exporting dataset in R**



Output : exported dataset in txt and csv file



- **Quick overview**

-> head(df)

-> tail(df)

-> summary(df)

-> str(df)

```
> head(df)
  RowID  Distillery Body Sweetness Smoky Medicinal Tobacco Honey Spicy Winey Nutty Malty Fruity Floral   Postcode Latitude
1     1   Aberfeldy    2         2     2         0       0     2     1     2     2     2      2      2 \tPH15 2EB   286580
2     2   Aberlour     3         3     1         0       0     4     3     2     2     3      3      2 \tAB38 9PJ   326340
3     3      AnCnoc    1         3     2         0       0     2     0     0     2     2      3      2  \tAB5 5LI   352960
4     4      Ardbeg    4         1     4         4       0     0     2     0     1     2      1      0 \tPA42 7EB   141560
5     5     Ardmore    2         2     2         0       0     1     1     1     2     3      1      1 \tAB54 4NH   355350
6     6 ArranIsleOf    2         3     1         1       0     1     1     1     0     1      1      2   KA27 8HJ   194050
  Longitude
1    749680
2    842570
3    839320
4    646220
5    829140
6    649950
> tail(df)
   RowID   Distillery Body Sweetness Smoky Medicinal Tobacco Honey Spicy Winey Nutty Malty Fruity Floral Postcode Latitude
81    81    Teaninich    2         2     2         1       0     0     2     0     0     0      2      2 IV17 0XB   265360
82    82    Tobermory    1         1     1         0       0     1     0     0     1     2      2      2 PA75 6NR   150450
83    83      Tomatin    2         3     2         0       0     2     2     1     1     2      0      1 IV13 7YT   279120
84    84    Tomintoul    0         3     1         0       0     2     2     1     1     2      1      2 AB37 9AQ   315100
85    85      Tormore    2         2     1         0       0     1     0     1     2     1      0      0 PH26 3LR   315180
86    86 Tullibardine    2         3     0         0       1     0     2     1     1     2      2      1  PH4 1QG   289690
   Longitude
81    869120
82    755070
83    829630
84    825560
85    834960
86    708850
```

```
> summary(df)
     RowID         Distillery             Body        Sweetness         Smoky          Medicinal          Tobacco
 Min.   : 1.00   Length:86          Min.   :0.00   Min.   :1.000   Min.   :0.000   Min.   :0.0000   Min.   :0.0000
 1st Qu.:22.25   Class :character   1st Qu.:2.00   1st Qu.:2.000   1st Qu.:1.000   1st Qu.:0.0000   1st Qu.:0.0000
 Median :43.50   Mode  :character   Median :2.00   Median :2.000   Median :1.000   Median :0.0000   Median :0.0000
 Mean   :43.50                      Mean   :2.07   Mean   :2.291   Mean   :1.535   Mean   :0.5465   Mean   :0.1163
 3rd Qu.:64.75                      3rd Qu.:2.00   3rd Qu.:3.000   3rd Qu.:2.000   3rd Qu.:1.0000   3rd Qu.:0.0000
 Max.   :86.00                      Max.   :4.00   Max.   :4.000   Max.   :4.000   Max.   :4.0000   Max.   :1.0000
     Honey           Spicy           Winey            Nutty           Malty           Fruity          Floral
 Min.   :0.000   Min.   :0.000   Min.   :0.0000   Min.   :0.000   Min.   :0.000   Min.   :0.000   Min.   :0.000
 1st Qu.:1.000   1st Qu.:1.000   1st Qu.:0.0000   1st Qu.:1.000   1st Qu.:1.000   1st Qu.:1.000   1st Qu.:1.000
 Median :1.000   Median :1.000   Median :1.0000   Median :2.000   Median :2.000   Median :2.000   Median :2.000
 Mean   :1.244   Mean   :1.384   Mean   :0.9767   Mean   :1.465   Mean   :1.802   Mean   :1.802   Mean   :1.698
 3rd Qu.:2.000   3rd Qu.:2.000   3rd Qu.:1.0000   3rd Qu.:2.000   3rd Qu.:2.000   3rd Qu.:2.000   3rd Qu.:2.000
 Max.   :4.000   Max.   :3.000   Max.   :4.0000   Max.   :4.000   Max.   :3.000   Max.   :3.000   Max.   :4.000
   Postcode            Latitude        Longitude
 Length:86          Min.   :126680   Min.   : 554260
 Class :character   1st Qu.:265672   1st Qu.: 755698
 Mode  :character   Median :319515   Median : 839885
                    Mean   :287247   Mean   : 802660
                    3rd Qu.:328630   3rd Qu.: 850770
                    Max.   :381020   Max.   :1009260
> str(df)
'data.frame':	86 obs. of  17 variables:
 $ RowID     : int  1 2 3 4 5 6 7 8 9 10 ...
 $ Distillery: chr  "Aberfeldy" "Aberlour" "AnCnoc" "Ardbeg" ...
 $ Body      : int  2 3 1 4 2 2 0 2 2 2 ...
 $ Sweetness : int  2 3 3 1 2 3 2 3 2 3 ...
 $ Smoky     : int  2 1 2 4 2 1 0 1 1 2 ...
 $ Medicinal : int  0 0 0 4 0 1 0 0 0 1 ...
 $ Tobacco   : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Honey     : int  2 4 2 0 1 1 1 2 1 0 ...
 $ Spicy     : int  1 3 0 2 1 1 1 1 0 2 ...
 $ Winey     : int  2 2 0 0 1 1 0 2 0 0 ...
 $ Nutty     : int  2 2 2 1 2 0 2 2 2 2 ...
 $ Malty     : int  2 3 2 2 3 1 2 2 2 1 ...
 $ Fruity    : int  2 3 3 1 1 1 3 2 2 2 ...
 $ Floral    : int  2 2 2 0 1 2 3 1 2 1 ...
 $ Postcode  : chr  "\tPH15 2EB" "\tAB38 9PJ" "\tAB5 5LI" "\tPA42 7EB" ...
 $ Latitude  : int  286580 326340 352960 141560 355350 194050 247670 340754 340754 270820 ...
 $ Longitude : int  749680 842570 839320 646220 829140 649950 672610 848623 848623 885770 ...
```

- **Cleaning dataset**

-> duplicated(df)

-> na.omit(df)

-> is.na(df)

```
> duplicated(df)
 [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[22] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[43] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[64] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[85] FALSE FALSE
```

```
> na.omit(df)
   RowID      Distillery Body Sweetness Smoky Medicinal Tobacco Honey Spicy Winey Nutty Malty Fruity Floral   Postcode Latitude
1      1       Aberfeldy    2         2     2         0       0     2     1     2     2     2      2      2 \tPH15 2EB   286580
2      2        Aberlour    3         3     1         0       0     4     3     2     2     3      3      2 \tAB38 9PJ   326340
3      3          AnCnoc    1         3     2         0       0     2     0     0     2     2      3      2  \tAB5 5LI   352960
4      4          Ardbeg    4         1     4         4       0     0     2     0     1     2      1      0 \tPA42 7EB   141560
5      5         Ardmore    2         2     2         0       0     1     1     1     2     3      1      1 \tAB54 4NH   355350
6      6     ArranIsleOf    2         3     1         1       0     1     1     1     0     1      1      2   KA27 8HJ   194050
7      7    Auchentoshan    0         2     0         0       0     1     1     0     2     2      3      3    G81 4SJ   247670
8      8       Auchroisk    2         3     1         0       0     2     1     2     2     2      2      1 \tAB55 3XS   340754
9      9        Aultmore    2         2     1         0       0     1     0     0     2     2      2      2 \tAB55 3QY   340754
10    10        Balblair    2         3     2         1       0     0     2     0     2     1      2      1 \tIV19 1LB   270820
11    11       Balmenach    4         3     2         0       0     2     1     3     3     0      1      2 \tPH26 3PF   307750
12    12        Belvenie    3         2     1         0       0     3     2     1     0     2      2      2 \tAB55 4DH   332680
13    13        BenNevis    4         2     2         0       0     2     2     0     2     2      2      2 \tPH33 6TJ   212600
```

```
   Longitude
1     749680
2     842570
3     839320
4     646220
5     829140
6     649950
7     672610
8     848623
9     848623
10    885770
11    827170
12    840840
```

```
41    849140
42    841240
43    844930
44    840300
45    838160
46    840840
47    682750
48    666690
49    828780
50    861040
51    883450
52    849170
53    723580
54   1009260
55    863970
56    667040
57    841570
58    645730
 [ reached 'max' / getOption("max.print") -- omitted 28 rows ]
```

```
> is.na(df)
     RowID Distillery  Body Sweetness Smoky Medicinal Tobacco Honey Spicy Winey Nutty Malty Fruity Floral Postcode Latitude
[1,] FALSE      FALSE FALSE     FALSE FALSE     FALSE   FALSE FALSE FALSE FALSE FALSE FALSE  FALSE  FALSE    FALSE    FALSE
[2,] FALSE      FALSE FALSE     FALSE FALSE     FALSE   FALSE FALSE FALSE FALSE FALSE FALSE  FALSE  FALSE    FALSE    FALSE
[3,] FALSE      FALSE FALSE     FALSE FALSE     FALSE   FALSE FALSE FALSE FALSE FALSE FALSE  FALSE  FALSE    FALSE    FALSE
[4,] FALSE      FALSE FALSE     FALSE FALSE     FALSE   FALSE FALSE FALSE FALSE FALSE FALSE  FALSE  FALSE    FALSE    FALSE
[5,] FALSE      FALSE FALSE     FALSE FALSE     FALSE   FALSE FALSE FALSE FALSE FALSE FALSE  FALSE  FALSE    FALSE    FALSE
[6,] FALSE      FALSE FALSE     FALSE FALSE     FALSE   FALSE FALSE FALSE FALSE FALSE FALSE  FALSE  FALSE    FALSE    FALSE
[7,] FALSE      FALSE FALSE     FALSE FALSE     FALSE   FALSE FALSE FALSE FALSE FALSE FALSE  FALSE  FALSE    FALSE    FALSE
```

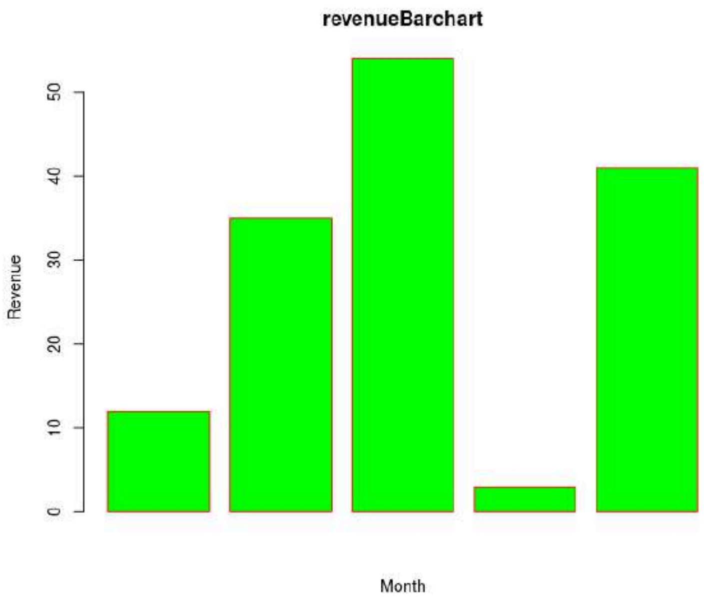
- **Exploring pattern**

```
> install.packages("ggplot2")
Installing package into '/home/computer/R/x86_64-pc-linux-gnu-library/4.1'
(as 'lib' is unspecified)
also installing the dependencies 'colorspace', 'utf8', 'farver', 'labeling', 'munsell',
le', 'isoband', 'lifecycle', 'rlang', 'scales', 'tibble', 'vctrs', 'withr'
```

```
* DONE (ggplot2)

The downloaded source packages are in
        '/tmp/Rtmpz27f3a/downloaded_packages'
```

```
> library("ggplot2")
> H <- c(12,35,54,3,41)
>
> m <- c("feb","mar","april","may","june")
```

```
> barplot(H,names.arr=m,xlab = "Month",ylab = "Revenue",col = "green",main = "revenueBarchart",border = "red")
Warning messages:
1: In plot.window(xlim, ylim, log = log, ...) :
  "names.arr" is not a graphical parameter
2: In title(main = main, sub = sub, xlab = xlab, ylab = ylab, ...) :
  "names.arr" is not a graphical parameter
3: In axis(if (horiz) 1 else 2, cex.axis = cex.axis, ...) :
  "names.arr" is not a graphical parameter
>
```



- **Multiple Linear Regression in Python**

```
In [2]: import numpy as np

        # Creating a two-dimensional array
        arr2d = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])

        # Computing the mean of the entire array
        mean = np.mean(arr2d)
        print("Mean of the entire array:", mean)

        # Computing the mean of each column
        mean_col = np.mean(arr2d, axis=0)
        print("\nMean of each column:")
        print(mean_col)

        # Computing the standard deviation of each row
        std_row = np.std(arr2d, axis=1)
        print("\nStandard deviation of each row:")
        print(std_row)

        # Computing the sum of each row
        sum_row = np.sum(arr2d, axis=1)
        print("\nSum of each row:")
        print(sum_row)

        # Computing the maximum value of each column
        max_col = np.max(arr2d, axis=0)
        print("\nMaximum value of each column:")
        print(max_col)
```

```
Mean of the entire array: 5.0

Mean of each column:
[4. 5. 6.]

Standard deviation of each row:
[0.81649658 0.81649658 0.81649658]

Sum of each row:
[ 6 15 24]

Maximum value of each column:
[7 8 9]
```

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)

T.E/SEM VI/CBCGS/AIML
Academic Year: 2022-23

NAME	**SINGH SUDHAM DHARMENDRA**
BRANCH	**CSE-(AI&ML)**
ROLL NO.	**57**
SUBJECT	**DATA ANALYTICS AND VISUALIZATION LAB**
COURSE CODE	**CSL601**
PRACTICAL NO.	**04**
DOP	
DOS	

Experiment No. : 04

Aim : Time Series Analysis in Python/R

Theory :

- A time series is a set of observations that are recorded over time, typically at regular intervals. Time series analysis is used to analyze such data to extract useful information and make predictions about future values.
- For example, consider monthly sales data for a company for the past 3 years, where each observation represents the total sales in a particular month. This data can be represented as a time series, where the time variable is the month and the sales variable is the value recorded for that month.
- Using time series analysis, we can extract information about the trend, seasonality, and other patterns in the data. For instance, we can plot the time series to visualize the trend and seasonality, and we can use autocorrelation analysis to identify any correlation between the sales data at different lags.
- Using this information, we can develop a model to forecast future sales. For example, we can use a time series model such as ARIMA (Autoregressive Integrated Moving Average) or exponential smoothing to make predictions about future sales based on past trends and patterns in the data.
- Overall, time series analysis provides a powerful tool for analyzing and forecasting data that is collected over time, and it has applications in various fields such as finance, economics, and engineering.

Implementation :

Write a program to perform Time Series Analysis in Python in :

```
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.tsa.stattools import adfuller
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.tsa.arima_model import ARIMA
data = pd.read_csv("sales_data.csv", parse_dates=["Date"], index_col="Date") # Load the data
# Plot the data
plt.figure(figsize=(10, 4))
plt.plot(data)
plt.title("Sales Data")
plt.xlabel("Date")
plt.ylabel("Sales")
plt.show()
# Check for stationarity using the Augmented Dickey-Fuller Test
result = adfuller(data["Sales"])
print("ADF Statistic:", result[0])
print("p-value:", result[1])
print("Critical Values:")
for key, value in result[4].items():
print(f"\t{key}: {value}")
```

```
# Plot the autocorrelation and partial autocorrelation functions
fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(10, 6))
plot_acf(data, ax=ax1, lags=20)
plot_pacf(data, ax=ax2, lags=20)
plt.show()
# Fit an ARIMA model
model = ARIMA(data, order=(1, 0, 0))
results = model.fit()
print(results.summary()) # Print the model summary
# Plot the residuals
plt.figure(figsize=(10, 4))
plt.plot(results.resid)
plt.title("Residuals")
plt.xlabel("Date")
plt.ylabel("Residual")
plt.show()
forecast = results.forecast(steps=3) # Make predictions for the next 3 months
print("Forecasted Sales:", forecast[0])
```

Output -

```
ADF Statistic: -2.5590145552827387
p-value: 0.10236480415917944
Critical Values: 1%: -3.5246240467919034, 5%: -2.902607073170798,10%: -2.5886780263023037
 ARMA Model Results
==============================================================================
Dep. Variable:                Sales No.          Observations: 100
Model:                        ARMA(1, 0)         Log Likelihood -446.047
Method:                       css-mle            S.D. of innovations 39.365
Date:                         Fri, 16 Apr 2023   AIC 898.095
Time:                         15:32:45           BIC 906.743
Sample:                       01-01-2019         HQIC 901.654
                       - 04-10-2021
==============================================================================
                coef        std err       z  P>|z|          [0.025 0.975]
------------------------------------------------------------------------------
const          97.1495     50.447        1.924              0.054 -1.696   195.995
ar.L1.Sales    0.3677      0.098         3.751              0.000 0.175 0.560
 Roots
==============================================================================
 Real           Imaginary          Modulus              Frequency
------------------------------------------------------------------------------
AR.1           2.7185             +0.0000j             2.7185 0.0000
------------------------------------------------------------------------------
Forecasted Sales: [1117.02566666 1160.17786267 1203.
```

Conclusion : We had successfully studied and understood time series analysis in Python/R

- **Importing dataset**
- **Loading the data**
- **Splitting the data into training and testing sets**
- **Fitting an ARIMA model**
- **Print the model summary**

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.tsa.arima_model import ARIMA
df = pd.read_csv('https://raw.githubusercontent.com/liannewriting/YouTube-videos-public
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 393 entries, 0 to 392
Data columns (total 1 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   traffic  393 non-null    int64
dtypes: int64(1)
memory usage: 3.2 KB
```

In [2]:
```
train_data = df[:80]
test_data = df[80:]
```

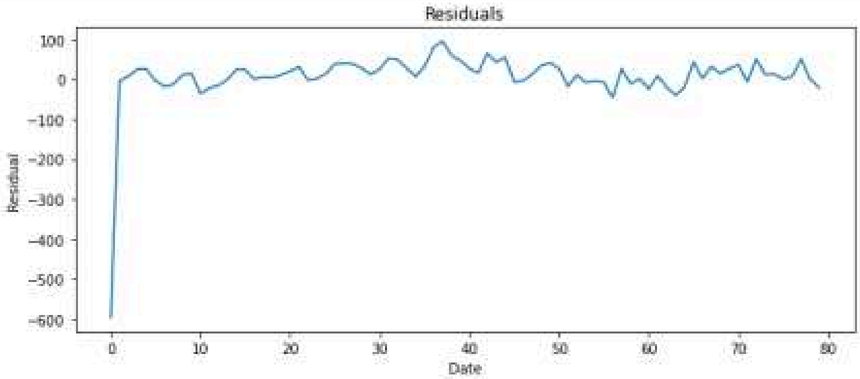
In [3]:
```
from statsmodels.tsa.arima.model import ARIMA
model = ARIMA(train_data, order=(1, 0, 0))
results = model.fit()
```

In [4]:
```
print(results.summary())
```

```
                               SARIMAX Results
==============================================================================
Dep. Variable:                traffic   No. Observations:                   80
Model:                 ARIMA(1, 0, 0)   Log Likelihood                -389.610
Date:                Mon, 17 Apr 2023   AIC                            785.220
Time:                        14:28:00   BIC                            792.366
Sample:                             0   HQIC                           788.085
                                 - 80
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const       1528.5952    562.454      2.718      0.007     426.205    2630.985
ar.L1          0.9985      0.010     99.300      0.000       0.979       1.018
sigma2       924.5967    135.454      6.826      0.000     659.112    1190.081
===================================================================================
Ljung-Box (L1) (Q):                  18.54   Jarque-Bera (JB):                 1.37
Prob(Q):                              0.00   Prob(JB):                         0.50
Heteroskedasticity (H):               1.53   Skew:                             0.31
Prob(H) (two-sided):                  0.28   Kurtosis:                         3.15
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
```

- **Plotting the residuals**
- **Making predictions for the test set**
- **Plotting the actual and predicted values**

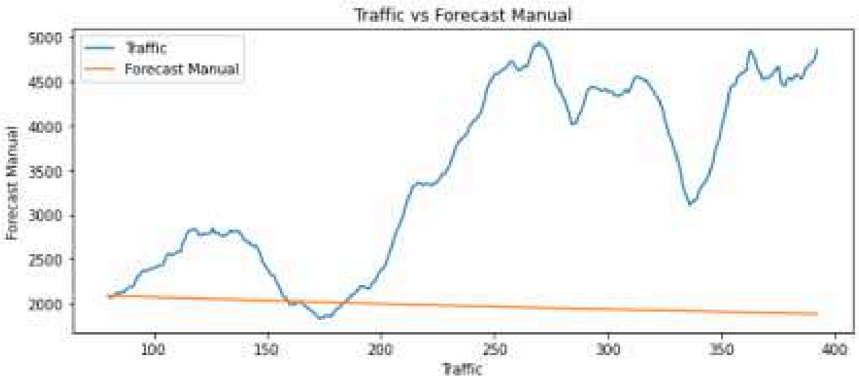
In [5]:
```
plt.figure(figsize=(10, 4))
plt.plot(results.resid)
plt.title("Residuals")
plt.xlabel("Date")
plt.ylabel("Residual")
plt.show()
```



In [6]:
```
predictions = results.forecast(steps=len(test_data))
predictions
```

Out[6]:
```
80     2086.188475
81     2085.378130
82     2084.568962
83     2083.760970
84     2082.954152
          ...
388    1884.865769
389    1884.348004
390    1883.830992
391    1883.314731
392    1882.799220
Name: predicted_mean, Length: 313, dtype: float64
```

In [8]:
```
plt.figure(figsize=(10, 4))
plt.plot(test_data, label="Traffic")
plt.plot(predictions, label="Forecast Manual")
plt.title("Traffic vs Forecast Manual")
plt.xlabel("Traffic")
plt.ylabel("Forecast Manual")
plt.legend()
plt.show()
```



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)

T.E/SEM VI/CBCGS/AIML
Academic Year: 2022-23

NAME	**SINGH SUDHAM DHARMENDRA**
BRANCH	**CSE-(AI&ML)**
ROLL NO.	**57**
SUBJECT	**DATA ANALYTICS AND VISUALIZATION LAB**
COURSE CODE	**CSL601**
PRACTICAL NO.	
DOP	**09/02/2023**
DOS	

HYPOTHESIS TESTING

Program(input)/Output :

- One Sample t test ->

In [2]:

```
ages=[10,20,35,50,28,40,55,18,76,55,30,25,43,18,30,28,
      14,24,16,17,32,35,26,27,65,18,43,23,21,20,19,70]
len(ages)
```

Out[2]:

```
32
```

In [3]:

```
import numpy as np
ages_mean=np.mean(ages)
print(ages_mean)
```

```
32.21875
```

#Let's take sample :

In [22]:

```
sample_size=10
age_sample=np.random.choice(ages,sample_size)
age_sample
```

Out[22]:

```
array([35, 55, 21, 18, 32, 76, 19, 25, 23, 25])
```

In [23]:

```
from scipy.stats import ttest_1samp as ttest_1samp
ttest,p_value=ttest_1samp(age_sample,30)
print(p_value)
```

```
0.6347125419461657
```

In [24]:

```
if p_value < 0.05:
    print("we are rejecting null hypothesis")
else:
    print("we are acceptin null hypothesis")
```

```
we are acceptin null hypothesis
```

#Some more examples :

In [29]:

```
import numpy as np
import pandas as pd
import scipy.stats as stats
import math
np.random.seed(6)
school_ages=stats.poisson.rvs(loc=18,mu=35,size=1500)
classA_ages=stats.poisson.rvs(loc=18,mu=30,size=60)
classA_ages.mean()
```

Out[29]:

```
46.9
```

In [32]:

```
p_value=stats.ttest_1samp(a=classA_ages,popmean=school_ages.mean())
school_ages.mean()
print(p_value)
```

Out[32]:

```
53.303333333333335
```

In [39]:

```
if p_value < 0.05:
    print("we are rejecting null hypothesis")
else:
    print("we are accepting null hypothesis")
```

```
we are accepting null hypothesis
```

- Independent t-test ->

In [40]:

```
np.random.seed(12)
classB_ages=stats.poisson.rvs(loc=18,mu=33,size=60)
classB_ages.mean()
```

Out[40]:

```
50.63333333333333
```

In [58]:

```
_,p_value=stats.ttest_ind(a=classA_ages,b=classB_ages)
if p_value < 0.05:
    print("we are rejecting null hypothesis")
else:
    print("we are accepting null hypothesis")
```

```
we are rejecting null hypothesis
```

- Paired t-test ->

In [45]:

```
weight_1=[25,30,28,35,28,34,26,29,30,26,28,32,31,30,45]
weight_2=weight_1+stats.norm.rvs(scale=5,loc=-1.25,size=15)
print(weight_1)
print(weight_2)
```

```
[25, 30, 28, 35, 28, 34, 26, 29, 30, 26, 28, 32, 31, 30, 45]
[30.57926457 34.91022437 29.00444617 30.54295091 19.86201983 37.578731
74
 18.3299827  21.3771395  36.36420881 32.05941216 26.93827982 29.519014
 26.42851213 30.50667769 41.32984284]
```

In [48]:

```
weight_df=pd.DataFrame({"weight_10":np.array(weight_1),
                        "weight_20":np.array(weight_2),
                        "weight_change":np.array(weight_2)-np.array(weight_1)})
```

In [50]:

```
weight_df,p_value=stats.ttest_rel(a=weight_1,b=weight_2)
print(p_value)
```

```
0.5732936534411279
```

In [51]:

```
if p_value<0.05:
    print("we are rejecting null hypothesis")
else:
    print("we are accepting null hypothesis")
```

```
we are accepting null hypothesis
```

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)

T.E/SEM VI/CBCGS/AIML
Academic Year: 2022-23

NAME	**SINGH SUDHAM DHARMENDRA**
BRANCH	**CSE-(AI&ML)**
ROLL NO.	**57**
SUBJECT	**DATA ANALYTICS AND VISUALIZATION LAB**
COURSE CODE	**CSL601**
PRACTICAL NO.	
DOP	**09/02/2023**
DOS	

DATA-CLEANING

Program(input)/Output :

In [1]:

```
import pandas as pd
data = pd.read_csv('/home/computer/Documents/sudham/student.csv')
```

In [2]:

```
data
```

Out[2]:

	id	name	class	mark	gender
0	1	John Deo	Four	75.0	female
1	2	Max Ruin	Three	85.0	male
2	3	Arnold	Three	55.0	male
3	4	Krish Star	Four	60.0	female
4	5	John Mike	Four	60.0	female
5	6	Alex John	Four	55.0	male
6	7	My John Rob	Fifth	78.0	male
7	8	Asruid	Five	85.0	male
8	9	Tes Qry	Six	78.0	male
9	10	Big John	Four	55.0	female
10	11	Ronald	Six	89.0	female
11	12	Recky	Six	94.0	female
12	13	Kty	Seven	88.0	female
13	14	Bigy	Seven	88.0	female
14	15	Tade Row	Four	88.0	male
15	16	Gimmy	Four	88.0	male
16	17	Tumyu	Six	54.0	male
17	18	Honny	xyz	75.0	male
18	19	Tinny	NaN	NaN	male
19	20	Jackly	Nine	65.0	female
20	21	Babby John	Four	69.0	female
21	22	Reggid	Seven	55.0	female
22	23	Herod	Eight	79.0	male
23	24	Tiddy Now	Seven	78.0	male
24	25	Giff Tow	Seven	88.0	male
25	26	Crelea	Seven	79.0	NaN
26	27	Big Nose	Three	81.0	female
27	28	Rojj Base	Seven	86.0	female
28	29	Tess Played	Seven	55.0	male
29	30	Reppy Red	Six	79.0	female
30	31	Marry Toeey	Four	88.0	male
31	32	Binn Rott	Seven	90.0	female
32	33	Kenn Rein	Six	96.0	female
33	34	Gain Toe	Seven	69.0	male
34	35	Rows Noump	Six	88.0	female
35	36	Gimmy	Four	88.0	male
36	36	Gimmy	Four	88.0	male

data_frame.head()

In [3]:

```
data.head()
```

Out[3]:

	id	name	class	mark	gender
0	1	John Deo	Four	75.0	female
1	2	Max Ruin	Three	85.0	male
2	3	Arnold	Three	55.0	male
3	4	Krish Star	Four	60.0	female
4	5	John Mike	Four	60.0	female

data_frame.dropna()

In [4]:

```
data = pd.read_csv('/home/computer/Documents/sudham/student.csv')
new_df = data.dropna() #column 19 & 26 dropped
print(new_df.to_string())
```

```
    id          name  class  mark  gender
0    1      John Deo   Four  75.0  female
1    2      Max Ruin  Three  85.0    male
2    3        Arnold  Three  55.0    male
3    4    Krish Star   Four  60.0  female
4    5     John Mike   Four  60.0  female
5    6     Alex John   Four  55.0    male
6    7   My John Rob  Fifth  78.0    male
7    8        Asruid   Five  85.0    male
8    9       Tes Qry    Six  78.0    male
9   10      Big John   Four  55.0  female
10  11        Ronald    Six  89.0  female
11  12         Recky    Six  94.0  female
12  13           Kty  Seven  88.0  female
13  14          Bigy  Seven  88.0  female
14  15      Tade Row   Four  88.0    male
15  16         Gimmy   Four  88.0    male
16  17         Tumyu    Six  54.0    male
17  18         Honny    xyz  75.0    male
19  20        Jackly   Nine  65.0  female
20  21    Babby John   Four  69.0  female
21  22        Reggid  Seven  55.0  female
22  23         Herod  Eight  79.0    male
23  24     Tiddy Now  Seven  78.0    male
24  25      Giff Tow  Seven  88.0    male
26  27      Big Nose  Three  81.0  female
27  28     Rojj Base  Seven  86.0  female
28  29   Tess Played  Seven  55.0    male
29  30     Reppy Red    Six  79.0  female
30  31   Marry Toeey   Four  88.0    male
31  32     Binn Rott  Seven  90.0  female
32  33     Kenn Rein    Six  96.0  female
33  34      Gain Toe  Seven  69.0    male
34  35    Rows Noump    Six  88.0  female
35  36         Gimmy   Four  88.0    male
36  36         Gimmy   Four  88.0    male
```

data_frame.dropna()

In [5]:

```
data = pd.read_csv('/home/computer/Documents/sudham/student.csv')
data.dropna(inplace = True) #column 19 & 26 dropped
print(new_df.to_string())
```

```
    id          name   class  mark  gender
0    1      John Deo    Four  75.0  female
1    2      Max Ruin   Three  85.0    male
2    3        Arnold   Three  55.0    male
3    4    Krish Star    Four  60.0  female
4    5     John Mike    Four  60.0  female
5    6     Alex John    Four  55.0    male
6    7   My John Rob   Fifth  78.0    male
7    8        Asruid    Five  85.0    male
8    9       Tes Qry     Six  78.0    male
9   10      Big John    Four  55.0  female
10  11        Ronald     Six  89.0  female
11  12         Recky     Six  94.0  female
12  13           Kty   Seven  88.0  female
13  14          Bigy   Seven  88.0  female
14  15     Tade Row    Four  88.0    male
15  16         Gimmy    Four  88.0    male
16  17         Tumyu     Six  54.0    male
17  18         Honny     xyz  75.0    male
19  20        Jackly    Nine  65.0  female
20  21    Babby John    Four  69.0  female
21  22        Reggid   Seven  55.0  female
22  23         Herod   Eight  79.0    male
23  24     Tiddy Now   Seven  78.0    male
24  25      Giff Tow   Seven  88.0    male
26  27      Big Nose   Three  81.0  female
27  28     Rojj Base   Seven  86.0  female
28  29   Tess Played   Seven  55.0    male
29  30     Reppy Red     Six  79.0  female
30  31   Marry Toeey    Four  88.0    male
31  32     Binn Rott   Seven  90.0  female
32  33     Kenn Rein     Six  96.0  female
33  34      Gain Toe   Seven  69.0    male
34  35    Rows Noump     Six  88.0  female
35  36         Gimmy    Four  88.0    male
36  36         Gimmy    Four  88.0    male
```

data_frame.fillna()

In [6]:

```
data = pd.read_csv('/home/computer/Documents/sudham/student.csv')
data.fillna('UNKNOWN',inplace = True) #column 19 & 26 replace with 13
print(new_df.to_string())
```

```
    id          name   class  mark  gender
0    1      John Deo    Four  75.0  female
1    2      Max Ruin   Three  85.0    male
2    3        Arnold   Three  55.0    male
3    4    Krish Star    Four  60.0  female
4    5     John Mike    Four  60.0  female
5    6     Alex John    Four  55.0    male
6    7   My John Rob   Fifth  78.0    male
7    8        Asruid    Five  85.0    male
8    9       Tes Qry     Six  78.0    male
9   10      Big John    Four  55.0  female
10  11        Ronald     Six  89.0  female
11  12         Recky     Six  94.0  female
12  13           Kty   Seven  88.0  female
13  14          Bigy   Seven  88.0  female
14  15     Tade Row    Four  88.0    male
15  16         Gimmy    Four  88.0    male
16  17         Tumyu     Six  54.0    male
17  18         Honny     xyz  75.0    male
19  20        Jackly    Nine  65.0  female
20  21    Babby John    Four  69.0  female
21  22        Reggid   Seven  55.0  female
22  23         Herod   Eight  79.0    male
23  24     Tiddy Now   Seven  78.0    male
24  25      Giff Tow   Seven  88.0    male
26  27      Big Nose   Three  81.0  female
27  28     Rojj Base   Seven  86.0  female
28  29   Tess Played   Seven  55.0    male
29  30     Reppy Red     Six  79.0  female
30  31   Marry Toeey    Four  88.0    male
31  32     Binn Rott   Seven  90.0  female
32  33     Kenn Rein     Six  96.0  female
33  34      Gain Toe   Seven  69.0    male
34  35    Rows Noump     Six  88.0  female
35  36         Gimmy    Four  88.0    male
36  36         Gimmy    Four  88.0    male
```

data_frame[ ].fillna()

In [7]:

```
data = pd.read_csv('/home/computer/Documents/sudham/student.csv')
data["gender"].fillna('MALE',inplace = True) #filled gender na with.....
print(new_df.to_string())
```

```
    id          name  class  mark  gender
0    1      John Deo   Four  75.0  female
1    2      Max Ruin  Three  85.0    male
2    3        Arnold  Three  55.0    male
3    4    Krish Star   Four  60.0  female
4    5     John Mike   Four  60.0  female
5    6     Alex John   Four  55.0    male
6    7   My John Rob  Fifth  78.0    male
7    8        Asruid   Five  85.0    male
8    9       Tes Qry    Six  78.0    male
9   10      Big John   Four  55.0  female
10  11        Ronald    Six  89.0  female
11  12         Recky    Six  94.0  female
12  13           Kty  Seven  88.0  female
13  14          Bigy  Seven  88.0  female
14  15      Tade Row   Four  88.0    male
15  16         Gimmy   Four  88.0    male
16  17         Tumyu    Six  54.0    male
17  18         Honny    xyz  75.0    male
19  20        Jackly   Nine  65.0  female
20  21    Babby John   Four  69.0  female
21  22        Reggid  Seven  55.0  female
22  23         Herod  Eight  79.0    male
23  24     Tiddy Now  Seven  78.0    male
24  25      Giff Tow  Seven  88.0    male
26  27      Big Nose  Three  81.0  female
27  28     Rojj Base  Seven  86.0  female
28  29   Tess Played  Seven  55.0    male
29  30     Reppy Red    Six  79.0  female
30  31   Marry Toeey   Four  88.0    male
31  32     Binn Rott  Seven  90.0  female
32  33     Kenn Rein    Six  96.0  female
33  34      Gain Toe  Seven  69.0    male
34  35    Rows Noump    Six  88.0  female
35  36         Gimmy   Four  88.0    male
36  36         Gimmy   Four  88.0    male
```

data[ ].mean()
data_frame[ ].fillmax()

In [8]:

```
data = pd.read_csv('/home/computer/Documents/sudham/student.csv')
x = data["mark"].mean() #column 19 & 26 replaced with mean
data["mark"].fillna(x, inplace = True)
print(new_df.to_string())
```

```
    id          name  class  mark  gender
0    1      John Deo   Four  75.0  female
1    2      Max Ruin  Three  85.0    male
2    3        Arnold  Three  55.0    male
3    4    Krish Star   Four  60.0  female
4    5     John Mike   Four  60.0  female
5    6     Alex John   Four  55.0    male
6    7   My John Rob  Fifth  78.0    male
7    8        Asruid   Five  85.0    male
8    9       Tes Qry    Six  78.0    male
9   10      Big John   Four  55.0  female
10  11        Ronald    Six  89.0  female
11  12         Recky    Six  94.0  female
12  13           Kty  Seven  88.0  female
13  14          Bigy  Seven  88.0  female
14  15      Tade Row   Four  88.0    male
15  16         Gimmy   Four  88.0    male
16  17         Tumyu    Six  54.0    male
17  18         Honny    xyz  75.0    male
19  20        Jackly   Nine  65.0  female
20  21    Babby John   Four  69.0  female
21  22        Reggid  Seven  55.0  female
22  23         Herod  Eight  79.0    male
23  24     Tiddy Now  Seven  78.0    male
24  25      Giff Tow  Seven  88.0    male
26  27      Big Nose  Three  81.0  female
27  28     Rojj Base  Seven  86.0  female
28  29   Tess Played  Seven  55.0    male
29  30     Reppy Red    Six  79.0  female
30  31   Marry Toeey   Four  88.0    male
31  32     Binn Rott  Seven  90.0  female
32  33     Kenn Rein    Six  96.0  female
33  34      Gain Toe  Seven  69.0    male
34  35    Rows Noump    Six  88.0  female
35  36         Gimmy   Four  88.0    male
36  36         Gimmy   Four  88.0    male
```

data[ ].median()
data_frame[ ].fillmax()

In [9]:

```
data = pd.read_csv('/home/computer/Documents/sudham/student.csv')
x = data["mark"].median() #column 19 & 26 replaced with median
data["mark"].fillna(x, inplace = True)
print(new_df.to_string())
```

```
    id         name  class  mark  gender
0    1     John Deo   Four  75.0  female
1    2     Max Ruin  Three  85.0    male
2    3       Arnold  Three  55.0    male
3    4   Krish Star   Four  60.0  female
4    5    John Mike   Four  60.0  female
5    6    Alex John   Four  55.0    male
6    7  My John Rob  Fifth  78.0    male
7    8       Asruid   Five  85.0    male
8    9      Tes Qry    Six  78.0    male
9   10     Big John   Four  55.0  female
10  11       Ronald    Six  89.0  female
11  12        Recky    Six  94.0  female
12  13          Kty  Seven  88.0  female
13  14         Bigy  Seven  88.0  female
14  15     Tade Row   Four  88.0    male
15  16        Gimmy   Four  88.0    male
16  17        Tumyu    Six  54.0    male
17  18        Honny    xyz  75.0    male
19  20       Jackly   Nine  65.0  female
20  21   Babby John   Four  69.0  female
21  22       Reggid  Seven  55.0  female
22  23        Herod  Eight  79.0    male
23  24    Tiddy Now  Seven  78.0    male
24  25     Giff Tow  Seven  88.0    male
26  27    Big Nose  Three  81.0  female
27  28    Rojj Base  Seven  86.0  female
28  29  Tess Played  Seven  55.0    male
29  30    Reppy Red    Six  79.0  female
30  31  Marry Toeey   Four  88.0    male
31  32    Binn Rott  Seven  90.0  female
32  33    Kenn Rein    Six  96.0  female
33  34     Gain Toe  Seven  69.0    male
34  35   Rows Noump    Six  88.0  female
35  36        Gimmy   Four  88.0    male
36  36        Gimmy   Four  88.0    male
```

data[ ].mode()
data_frame[ ].fillmax()

In [10]:

```
data = pd.read_csv('/home/computer/Documents/sudham/student.csv')
x = data["mark"].mode() #column 19 & 26 replaced with mode
data["mark"].fillna(x, inplace = True)
print(new_df.to_string())
```

```
    id         name  class  mark  gender
0    1     John Deo   Four  75.0  female
1    2     Max Ruin  Three  85.0    male
2    3       Arnold  Three  55.0    male
3    4   Krish Star   Four  60.0  female
4    5    John Mike   Four  60.0  female
5    6    Alex John   Four  55.0    male
6    7  My John Rob  Fifth  78.0    male
7    8       Asruid   Five  85.0    male
8    9      Tes Qry    Six  78.0    male
9   10     Big John   Four  55.0  female
10  11       Ronald    Six  89.0  female
11  12        Recky    Six  94.0  female
12  13          Kty  Seven  88.0  female
13  14         Bigy  Seven  88.0  female
14  15     Tade Row   Four  88.0    male
15  16        Gimmy   Four  88.0    male
16  17        Tumyu    Six  54.0    male
17  18        Honny    xyz  75.0    male
19  20       Jackly   Nine  65.0  female
20  21   Babby John   Four  69.0  female
21  22       Reggid  Seven  55.0  female
22  23        Herod  Eight  79.0    male
23  24    Tiddy Now  Seven  78.0    male
24  25     Giff Tow  Seven  88.0    male
26  27    Big Nose  Three  81.0  female
27  28    Rojj Base  Seven  86.0  female
28  29  Tess Played  Seven  55.0    male
29  30    Reppy Red    Six  79.0  female
30  31  Marry Toeey   Four  88.0    male
31  32    Binn Rott  Seven  90.0  female
32  33    Kenn Rein    Six  96.0  female
33  34     Gain Toe  Seven  69.0    male
34  35   Rows Noump    Six  88.0  female
35  36        Gimmy   Four  88.0    male
36  36        Gimmy   Four  88.0    male
```

data_frame.dropna()

In [11]:

```
data = pd.read_csv('/home/computer/Documents/sudham/student.csv')
data.dropna(subset=['class'],inplace = True) #column 19 dropped
print(new_df.to_string())
```

```
    id          name   class  mark  gender
0    1      John Deo    Four  75.0  female
1    2      Max Ruin   Three  85.0    male
2    3        Arnold   Three  55.0    male
3    4    Krish Star    Four  60.0  female
4    5     John Mike    Four  60.0  female
5    6     Alex John    Four  55.0    male
6    7   My John Rob   Fifth  78.0    male
7    8        Asruid    Five  85.0    male
8    9       Tes Qry     Six  78.0    male
9   10      Big John    Four  55.0  female
10  11        Ronald     Six  89.0  female
11  12         Recky     Six  94.0  female
12  13           Kty   Seven  88.0  female
13  14          Bigy   Seven  88.0  female
14  15      Tade Row    Four  88.0    male
15  16         Gimmy    Four  88.0    male
16  17         Tumyu     Six  54.0    male
17  18         Honny     xyz  75.0    male
19  20        Jackly    Nine  65.0  female
20  21    Babby John    Four  69.0  female
21  22        Reggid   Seven  55.0  female
22  23         Herod   Eight  79.0    male
23  24     Tiddy Now   Seven  78.0    male
24  25      Giff Tow   Seven  88.0    male
26  27      Big Nose   Three  81.0  female
27  28     Rojj Base   Seven  86.0  female
28  29   Tess Played   Seven  55.0    male
29  30     Reppy Red     Six  79.0  female
30  31   Marry Toeey    Four  88.0    male
31  32     Binn Rott   Seven  90.0  female
32  33     Kenn Rein     Six  96.0  female
33  34      Gain Toe   Seven  69.0    male
34  35    Rows Noump     Six  88.0  female
35  36         Gimmy    Four  88.0    male
36  36         Gimmy    Four  88.0    male
```

data_frame.loc[ ]= “”

In [12]:

```
data = pd.read_csv('/home/computer/Documents/sudham/student.csv')
data.loc[17,'class'] = "ten" #column 17 replaced
print(new_df.to_string())
```

```
    id          name   class  mark  gender
0    1      John Deo    Four  75.0  female
1    2      Max Ruin   Three  85.0    male
2    3        Arnold   Three  55.0    male
3    4    Krish Star    Four  60.0  female
4    5     John Mike    Four  60.0  female
5    6     Alex John    Four  55.0    male
6    7   My John Rob   Fifth  78.0    male
7    8        Asruid    Five  85.0    male
8    9       Tes Qry     Six  78.0    male
9   10      Big John    Four  55.0  female
10  11        Ronald     Six  89.0  female
11  12         Recky     Six  94.0  female
12  13           Kty   Seven  88.0  female
13  14          Bigy   Seven  88.0  female
14  15      Tade Row    Four  88.0    male
15  16         Gimmy    Four  88.0    male
16  17         Tumyu     Six  54.0    male
17  18         Honny     xyz  75.0    male
19  20        Jackly    Nine  65.0  female
20  21    Babby John    Four  69.0  female
21  22        Reggid   Seven  55.0  female
22  23         Herod   Eight  79.0    male
23  24     Tiddy Now   Seven  78.0    male
24  25      Giff Tow   Seven  88.0    male
26  27      Big Nose   Three  81.0  female
27  28     Rojj Base   Seven  86.0  female
28  29   Tess Played   Seven  55.0    male
29  30     Reppy Red     Six  79.0  female
30  31   Marry Toeey    Four  88.0    male
31  32     Binn Rott   Seven  90.0  female
32  33     Kenn Rein     Six  96.0  female
33  34      Gain Toe   Seven  69.0    male
34  35    Rows Noump     Six  88.0  female
35  36         Gimmy    Four  88.0    male
36  36         Gimmy    Four  88.0    male
```

data_frame.loc[ ]

In [17]:

```
data = pd.read_csv('/home/computer/Documents/sudham/student.csv')
for x in data.index:
    if data.loc[x,"mark"]>60:
        data.loc[x,"mark"] = 100
print(data.to_string())
```

```
    id         name  class   mark  gender
0    1     John Deo   Four  100.0  female
1    2     Max Ruin  Three  100.0    male
2    3       Arnold  Three   55.0    male
3    4   Krish Star   Four   60.0  female
4    5    John Mike   Four   60.0  female
5    6    Alex John   Four   55.0    male
6    7  My John Rob  Fifth  100.0    male
7    8       Asruid   Five  100.0    male
8    9      Tes Qry    Six  100.0    male
9   10     Big John   Four   55.0  female
10  11       Ronald    Six  100.0  female
11  12        Recky    Six  100.0  female
12  13          Kty  Seven  100.0  female
13  14         Bigy  Seven  100.0  female
14  15     Tade Row   Four  100.0    male
15  16        Gimmy   Four  100.0    male
16  17        Tumyu    Six   54.0    male
17  18        Honny    xyz  100.0    male
18  19        Tinny    NaN    NaN    male
19  20       Jackly   Nine  100.0  female
20  21   Babby John   Four  100.0  female
21  22       Reggid  Seven   55.0  female
22  23        Herod  Eight  100.0    male
23  24    Tiddy Now  Seven  100.0    male
24  25     Giff Tow  Seven  100.0    male
25  26       Crelea  Seven  100.0     NaN
26  27     Big Nose  Three  100.0  female
27  28    Rojj Base  Seven  100.0  female
28  29  Tess Played  Seven   55.0    male
29  30    Reppy Red    Six  100.0  female
30  31  Marry Toeey   Four  100.0    male
31  32    Binn Rott  Seven  100.0  female
32  33    Kenn Rein    Six  100.0  female
33  34     Gain Toe  Seven  100.0    male
34  35   Rows Noump    Six  100.0  female
35  36        Gimmy   Four  100.0    male
36  36        Gimmy   Four  100.0    male
```

data_frame.drop()

In [18]:

```
data = pd.read_csv('/home/computer/Documents/sudham/student.csv')
for x in data.index:
    if data.loc[x,"mark"]<60:
        data.drop(x, inplace = True)
print(data.to_string())
```

```
    id         name  class  mark  gender
0    1     John Deo   Four  75.0  female
1    2     Max Ruin  Three  85.0    male
3    4   Krish Star   Four  60.0  female
4    5    John Mike   Four  60.0  female
6    7  My John Rob  Fifth  78.0    male
7    8       Asruid   Five  85.0    male
8    9      Tes Qry    Six  78.0    male
10  11       Ronald    Six  89.0  female
11  12        Recky    Six  94.0  female
12  13          Kty  Seven  88.0  female
13  14         Bigy  Seven  88.0  female
14  15     Tade Row   Four  88.0    male
15  16        Gimmy   Four  88.0    male
17  18        Honny    xyz  75.0    male
18  19        Tinny    NaN   NaN    male
19  20       Jackly   Nine  65.0  female
20  21   Babby John   Four  69.0  female
22  23        Herod  Eight  79.0    male
23  24    Tiddy Now  Seven  78.0    male
24  25     Giff Tow  Seven  88.0    male
25  26       Crelea  Seven  79.0     NaN
26  27     Big Nose  Three  81.0  female
27  28    Rojj Base  Seven  86.0  female
29  30    Reppy Red    Six  79.0  female
30  31  Marry Toeey   Four  88.0    male
31  32    Binn Rott  Seven  90.0  female
32  33    Kenn Rein    Six  96.0  female
33  34     Gain Toe  Seven  69.0    male
34  35   Rows Noump    Six  88.0  female
35  36        Gimmy   Four  88.0    male
36  36        Gimmy   Four  88.0    male
```

data_frame.duplicated()

In [19]:

```
data = pd.read_csv('/home/computer/Documents/sudham/student.csv')
print(data.duplicated()) #column 3,9,18 & 21 removed
```

```
0     False
1     False
2     False
3     False
4     False
5     False
6     False
7     False
8     False
9     False
10    False
11    False
12    False
13    False
14    False
15    False
16    False
17    False
18    False
19    False
20    False
21    False
22    False
23    False
24    False
25    False
26    False
27    False
28    False
29    False
30    False
31    False
32    False
33    False
34    False
35    False
36     True
dtype: bool
```

data_frame.drop_duplicates()

In [20]:

```
data = pd.read_csv('/home/computer/Documents/sudham/student.csv')
data.drop_duplicates(inplace = True) #column36 is removed
print(data.to_string())
```

```
    id          name  class  mark  gender
0    1      John Deo   Four  75.0  female
1    2      Max Ruin  Three  85.0    male
2    3        Arnold  Three  55.0    male
3    4    Krish Star   Four  60.0  female
4    5     John Mike   Four  60.0  female
5    6     Alex John   Four  55.0    male
6    7   My John Rob  Fifth  78.0    male
7    8        Asruid   Five  85.0    male
8    9       Tes Qry    Six  78.0    male
9   10      Big John   Four  55.0  female
10  11        Ronald    Six  89.0  female
11  12         Recky    Six  94.0  female
12  13           Kty  Seven  88.0  female
13  14          Bigy  Seven  88.0  female
14  15      Tade Row   Four  88.0    male
15  16         Gimmy   Four  88.0    male
16  17         Tumyu    Six  54.0    male
17  18         Honny    xyz  75.0    male
18  19         Tinny    NaN   NaN    male
19  20        Jackly   Nine  65.0  female
20  21    Babby John   Four  69.0  female
21  22        Reggid  Seven  55.0  female
22  23         Herod  Eight  79.0    male
23  24     Tiddy Now  Seven  78.0    male
24  25     Giff Tow   Seven  88.0    male
25  26        Crelea  Seven  79.0     NaN
26  27      Big Nose  Three  81.0  female
27  28     Rojj Base  Seven  86.0  female
28  29   Tess Played  Seven  55.0    male
29  30     Reppy Red    Six  79.0  female
30  31   Marry Toeey   Four  88.0    male
31  32     Binn Rott  Seven  90.0  female
32  33     Kenn Rein    Six  96.0  female
33  34      Gain Toe  Seven  69.0    male
34  35    Rows Noump    Six  88.0  female
35  36         Gimmy   Four  88.0    male
```

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)

T.E/SEM VI/CBCGS/AIML
Academic Year: 2022-23

NAME	**SINGH SUDHAM DHARMENDRA**
BRANCH	**CSE-(AI&ML)**
ROLL NO.	**57**
SUBJECT	**DATA ANALYTICS AND VISUALIZATION LAB**
COURSE CODE	**CSL601**
PRACTICAL NO.	
DOP	
DOS	

DATA ANALYTICS LIBRARIES

NUMPY:

NumPy is a very popular python library for large multi-dimensional array and matrix processing, with the help of a large collection of high-level mathematical functions. It is very useful for fundamental scientific computations in Machine Learning. It is particularly useful for linear algebra, Fourier transform, and random number capabilities. High-end libraries like TensorFlow use NumPy internally for manipulation of Tensors.

PROGRAM:

```
# operations
import numpy as np
# Creating two arrays of rank 2
x = np.array([[1, 2], [3, 4]])
y = np.array([[5, 6], [7, 8]])
# Creating two arrays of rank 1
v = np.array([9, 10])
w = np.array([11, 12])
# Inner product of vectors
print(np.dot(v, w), "\n")
# Matrix and Vector product
print(np.dot(x, v), "\n")
# Matrix and matrix product
print(np.dot(x, y))
```

OUTPUT:

```
In [1]: # Python program using NumPy

        import numpy as np

        # Creating two arrays of rank 2
        x = np.array([[1, 2], [3, 4]])
        y = np.array([[5, 6], [7, 8]])

        # Creating two arrays of rank 1
        v = np.array([9, 10])
        w = np.array([11, 12])

        # Inner product of vectors
        print(np.dot(v, w), "\n")

        # Matrix and Vector product
        print(np.dot(x, v), "\n")

        # Matrix and matrix product
        print(np.dot(x, y))

        219

        [29 67]

        [[19 22]
         [43 50]]
```

SCIPY:

SciPy is a very popular library among Machine Learning enthusiasts as it contains different modules for optimization, linear algebra, integration and statistics. There is a difference between the SciPy library and the SciPy stack. The SciPy is one of the core packages that make up the SciPy stack. SciPy is also very useful for image manipulation.

PROGRAM:

from scipy import io
import numpy as np
arr = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9,])
#Export:
io.savemat('arr.mat', {"vec": arr})
#Import:
mydata = io.loadmat('arr.mat')
print(mydata)

OUTPUT:

```
from scipy import io
import numpy as np

arr = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9,])

#Export:
io.savemat('arr.mat', {"vec": arr})

#Import:
mydata = io.loadmat('arr.mat')

print(mydata)
```

```
{
  '__header__': b'MATLAB 5.0 MAT-file Platform: nt, Created on: Tue Sep 22 13:12:32 2020',
  '__version__': '1.0',
  '__globals__': [],
  'vec': array([[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]])
}
```

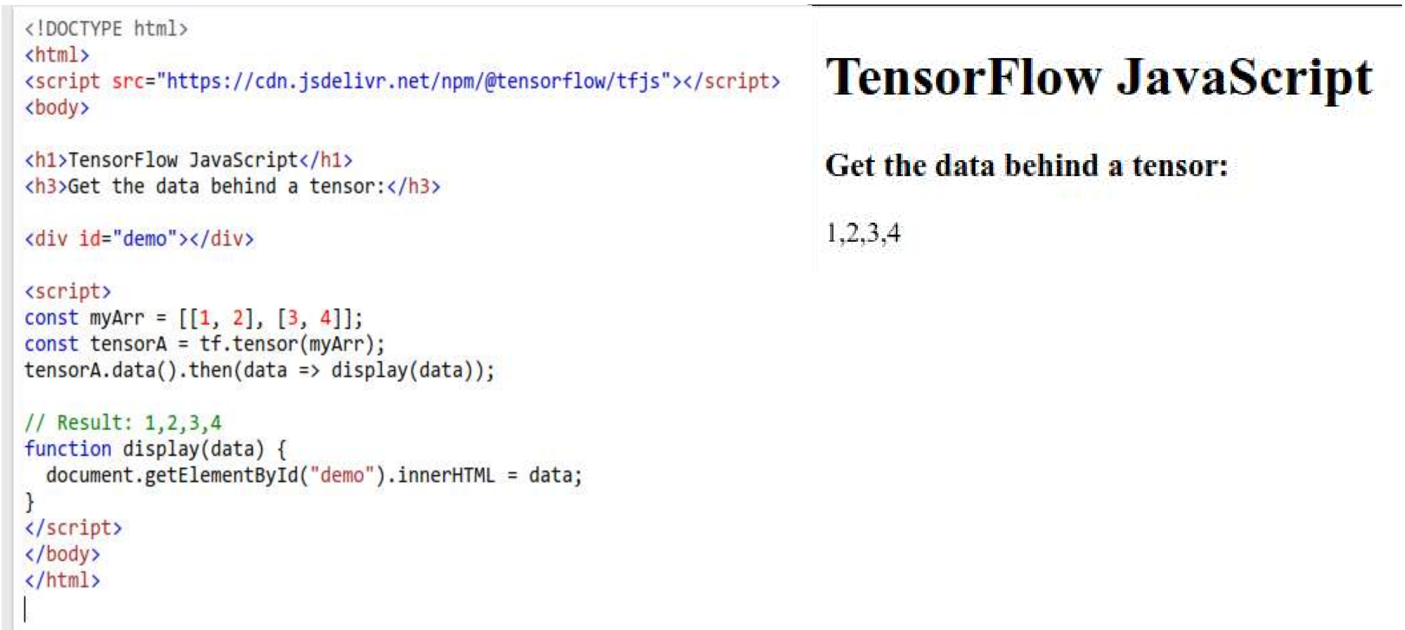
TENSORFLOW:

TensorFlow is a very popular open-source library for high performance numerical computation developed by the Google Brain team in Google. As the name suggests, Tensorflow is a framework that involves defining and running computations involving tensors. It can train and run deep neural networks that can be used to develop several AI applications. TensorFlow is widely used in the field of deep learning research and application.

PROGRAM:

```
<!DOCTYPE html>
<html>
<script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs"></script>
<body>
<h1>TensorFlow JavaScript</h1>
<h3>Get the data behind a tensor:</h3>
<div id="demo"></div>
<script>
const myArr = [[1, 2], [3, 4]]; const tensorA = tf.tensor(myArr);
tensorA.data().then(data => display(data));
// Result: 1,2,3,4 function display(data) {
        document.getElementById("demo").innerHTML = data;
}
</script>
</body>
</html>
```

OUTPUT:



PANDAS:

Pandas is a popular Python library for data analysis. It is not directly related to Machine Learning. As we know that the dataset must be prepared before training. In this case, Pandas comes handy as it was developed specifically for data extraction and preparation. It provides high-level data structures and a wide variety of tools for data analysis. It provides many inbuilt methods for grouping, combining and filtering data.

PROGRAM:

```
# arranging a given set of data
# into a table
# importing pandas as pd
import pandas as pd
data = {"country": ["Brazil", "Russia", "India", "China", "South Africa"],
        "capital": ["Brasilia", "Moscow", "New Delhi", "Beijing", "Pretoria"],
        "area": [8.516, 17.10, 3.286, 9.597, 1.221],
        "population": [200.4, 143.5, 1252, 1357, 52.98] }
data_table = pd.DataFrame(data)
print(data_table)
```

OUTPUT:

```
In [4]: # Python program using Pandas for
        # arranging a given set of data
        # into a table

        # importing pandas as pd
        import pandas as pd

        data = {"country": ["Brazil", "Russia", "India", "China", "South Africa"],
            "capital": ["Brasilia", "Moscow", "New Delhi", "Beijing", "Pretoria"],
            "area": [8.516, 17.10, 3.286, 9.597, 1.221],
            "population": [200.4, 143.5, 1252, 1357, 52.98] }

        data_table = pd.DataFrame(data)
        print(data_table)

                country    capital    area  population
        0        Brazil   Brasilia   8.516      200.40
        1        Russia     Moscow  17.100      143.50
        2         India  New Delhi   3.286     1252.00
        3         China    Beijing   9.597     1357.00
        4  South Africa   Pretoria   1.221       52.98
```

MATPLOTLIB:

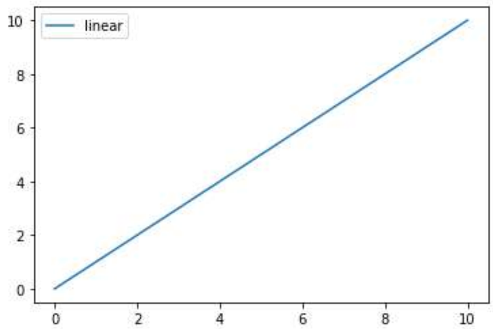
Matplotlib is a very popular Python library for data visualization. Like Pandas, it is not directly related to Machine Learning. It particularly comes in handy when a programmer wants to visualize the patterns in the data. It is a 2D plotting library used for creating 2D graphs and plots. A module named pyplot makes it easy for programmers for plotting as it provides features to control line styles, font properties, formatting axes, etc. It provides various kinds of graphs and plots for data visualization, viz., histogram, error charts, bar charts, etc.

PROGRAM:

```
# for forming a linear plot
import matplotlib.pyplot as plt
import numpy as np
# Prepare the data
x = np.linspace(0, 10, 100)
# Plot the data
plt.plot(x, x, label ='linear')
# Add a legend
plt.legend()
# Show the plot
plt.show()
```

OUTPUT:

In [5]:
```
# Python program using Matplotlib
# for forming a linear plot

import matplotlib.pyplot as plt
import numpy as np

# Prepare the data
x = np.linspace(0, 10, 100)

# Plot the data
plt.plot(x, x, label ='linear')

# Add a legend
plt.legend()

# Show the plot
plt.show()
```



In [ ]:

SCIKIT-LEARN :

Scikit-learn is one of the most popular ML libraries for classical ML algorithms. It is built on top of two basic Python libraries, viz., NumPy and SciPy.

PROGRAM/OUTPUT:

In [11]:
```
# Python script using Scikit-learn
# for Decision Tree Classifier

# Sample Decision Tree Classifier
from sklearn import datasets
from sklearn import metrics
from sklearn.tree import DecisionTreeClassifier

# load the iris datasets
dataset = datasets.load_iris()

# fit a CART model to the data
model = DecisionTreeClassifier()
model.fit(dataset.data, dataset.target)
print(model)

# make predictions
expected = dataset.target
predicted = model.predict(dataset.data)

# summarize the fit of the model
print(metrics.classification_report(expected, predicted))
print(metrics.confusion_matrix(expected, predicted))
```

```
DecisionTreeClassifier()
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        50
           1       1.00      1.00      1.00        50
           2       1.00      1.00      1.00        50

    accuracy                           1.00       150
   macro avg       1.00      1.00      1.00       150
weighted avg       1.00      1.00      1.00       150

[[50  0  0]
 [ 0 50  0]
 [ 0  0 50]]
```

PYTORCH

PyTorch is an open source machine learning library for Python and is completely based on Torch. It is primarily used for applications such as natural language processing.

PROGRAM/OUTPUT:

In [2]:
```
# importing torch
import torch

# creating a tensors
t1=torch.tensor([1, 2, 3, 4])
t2=torch.tensor([[1, 2, 3, 4],
                 [5, 6, 7, 8],
                 [9, 10, 11, 12]])

# printing the tensors:
print("Tensor t1: \n", t1)
print("\nTensor t2: \n", t2)

# rank of tensors
print("\nRank of t1: ", len(t1.shape))
print("Rank of t2: ", len(t2.shape))

# shape of tensors
print("\nRank of t1: ", t1.shape)
print("Rank of t2: ", t2.shape)
```

```
Tensor t1:
 tensor([1, 2, 3, 4])

Tensor t2:
 tensor([[ 1,  2,  3,  4],
        [ 5,  6,  7,  8],
        [ 9, 10, 11, 12]])

Rank of t1:  1
Rank of t2:  2

Rank of t1:  torch.Size([4])
Rank of t2:  torch.Size([3, 4])
```

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)

T.E/SEM VI/CBCGS/AIML
Academic Year: 2022-23

NAME	**SINGH SUDHAM DHARMENDRA**
BRANCH	**CSE-(AI&ML)**
ROLL NO.	**57**
SUBJECT	**DATA ANALYTICS AND VISUALIZATION LAB**
COURSE CODE	**CSL601**
PRACTICAL NO.	
DOP	**03/02/2023**
DOS	

Program(input)/Output :
data_frame

In [1]:
```
import pandas as pd
wine=pd.read_csv("https://raw.githubusercontent.com/YBI-Foundation/Dataset/main/Wine.csv")
```

In [2]:
```
wine
```

Out[2]:

	class_label	class_name	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	prc
0	1	Barolo	14.23	1.71	2.43	15.6	127	2.80	3.06	0.28	
1	1	Barolo	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26	
2	1	Barolo	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30	
3	1	Barolo	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24	
4	1	Barolo	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39	
...	...	...	...	...	...	...	...	...	...	...	
173	3	Barbera	13.71	5.65	2.45	20.5	95	1.68	0.61	0.52	
174	3	Barbera	13.40	3.91	2.48	23.0	102	1.80	0.75	0.43	
175	3	Barbera	13.27	4.28	2.26	20.0	120	1.59	0.69	0.43	
176	3	Barbera	13.17	2.59	2.37	20.0	120	1.65	0.68	0.53	
177	3	Barbera	14.13	4.10	2.74	24.5	96	2.05	0.76	0.56	

178 rows × 15 columns

data_frame.head()

In [3]:
```
wine.head()
```

Out[3]:

	class_label	class_name	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proar
0	1	Barolo	14.23	1.71	2.43	15.6	127	2.80	3.06	0.28	
1	1	Barolo	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26	
2	1	Barolo	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30	
3	1	Barolo	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24	
4	1	Barolo	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39	

data_frame.tail()

In [4]:
```
wine.tail()
```

Out[4]:

	class_label	class_name	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	prc
173	3	Barbera	13.71	5.65	2.45	20.5	95	1.68	0.61	0.52	
174	3	Barbera	13.40	3.91	2.48	23.0	102	1.80	0.75	0.43	
175	3	Barbera	13.27	4.28	2.26	20.0	120	1.59	0.69	0.43	
176	3	Barbera	13.17	2.59	2.37	20.0	120	1.65	0.68	0.53	
177	3	Barbera	14.13	4.10	2.74	24.5	96	2.05	0.76	0.56	

data_frame.info()

In [5]:
```
wine.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 178 entries, 0 to 177
Data columns (total 15 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   class_label           178 non-null    int64
 1   class_name            178 non-null    object
 2   alcohol               178 non-null    float64
 3   malic_acid            178 non-null    float64
 4   ash                   178 non-null    float64
 5   alcalinity_of_ash     178 non-null    float64
 6   magnesium             178 non-null    int64
 7   total_phenols         178 non-null    float64
 8   flavanoids            178 non-null    float64
 9   nonflavanoid_phenols  178 non-null    float64
 10  proanthocyanins       178 non-null    float64
 11  color_intensity       178 non-null    float64
 12  hue                   178 non-null    float64
 13  od280                 178 non-null    float64
 14  proline               178 non-null    int64
dtypes: float64(11), int64(3), object(1)
memory usage: 21.0+ KB
```

data_frame.describe()

In [6]:
```
wine.describe()
```

Out[6]:

	class_label	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	pro
count	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	
mean	1.938202	13.000618	2.336348	2.366517	19.494944	99.741573	2.295112	2.029270	0.361854	
std	0.775035	0.811827	1.117146	0.274344	3.339564	14.282484	0.625851	0.998859	0.124453	
min	1.000000	11.030000	0.740000	1.360000	10.600000	70.000000	0.980000	0.340000	0.130000	
25%	1.000000	12.362500	1.602500	2.210000	17.200000	88.000000	1.742500	1.205000	0.270000	
50%	2.000000	13.050000	1.865000	2.360000	19.500000	98.000000	2.355000	2.135000	0.340000	
75%	3.000000	13.677500	3.082500	2.557500	21.500000	107.000000	2.800000	2.875000	0.437500	
max	3.000000	14.830000	5.800000	3.230000	30.000000	162.000000	3.880000	5.080000	0.660000	

data_frame.columns

In [8]:
```
wine.columns
```

Out[8]:
```
Index(['class_label', 'class_name', 'alcohol', 'malic_acid', 'ash',
       'alcalinity_of_ash', 'magnesium', 'total_phenols', 'flavanoids',
       'nonflavanoid_phenols', 'proanthocyanins', 'color_intensity', 'hue',
       'od280', 'proline'],
      dtype='object')
```

data_frame.nlargest()

In [11]:
```
wine.nlargest(4,'alcohol')
```

Out[11]:

	class_label	class_name	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proa
8	1	Barolo	14.83	1.64	2.17	14.0	97	2.8	2.98	0.29	
13	1	Barolo	14.75	1.73	2.39	11.4	91	3.1	3.69	0.43	
6	1	Barolo	14.39	1.87	2.45	14.6	96	2.5	2.52	0.30	
14	1	Barolo	14.38	1.87	2.38	12.0	102	3.3	3.64	0.29	

data_frame.sort values()

In [12]:
```
wine.sort values('ash',ascending = False)
```

Out[12]:

	class_label	class_name	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	pro
121	2	Grignolino	11.56	2.05	3.23	28.5	119	3.18	5.08	0.47	
25	1	Barolo	13.05	2.05	3.22	25.0	124	2.63	2.68	0.47	
112	2	Grignolino	11.76	2.68	2.92	20.0	103	1.75	2.03	0.60	
4	1	Barolo	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39	
169	3	Barbera	13.40	4.60	2.86	25.0	112	1.98	0.96	0.27	
...	...	...	...	...	...	...	...	...	...	...	
69	2	Grignolino	12.21	1.19	1.75	16.8	151	1.85	1.28	0.14	
76	2	Grignolino	13.03	0.90	1.71	16.0	86	1.95	2.03	0.24	
66	2	Grignolino	13.11	1.01	1.70	15.0	78	2.98	3.18	0.26	
100	2	Grignolino	12.08	2.08	1.70	17.5	97	2.23	2.17	0.26	
59	2	Grignolino	12.37	0.94	1.36	10.6	88	1.98	0.57	0.28	

178 rows × 15 columns

data_frame.loc[ ]

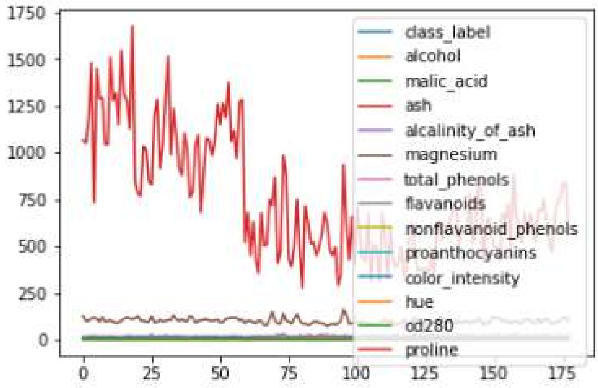
In [13]:
```
wine.loc[100:300,'malic acid']
```

Out[13]:
```
100    2.08
101    1.34
102    2.45
103    1.72
104    1.73
       ...
173    5.65
174    3.91
175    4.28
176    2.59
177    4.10
Name: malic_acid, Length: 78, dtype: float64
```

data_frame.plot()

```
In [15]: wine.plot()
```

```
Out[15]: <AxesSubplot:>
```



data_frame.shape

```
In [18]: wine.shape
```

```
Out[18]: (178, 15)
```

data_frame.corr()

```
In [19]: wine.corr()
```

Out[19]:

	class_label	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoic
class_label	1.000000	-0.328222	0.437776	-0.049643	0.517859	-0.209179	-0.719163	-0.847498	
alcohol	-0.328222	1.000000	0.094397	0.211545	-0.310235	0.270798	0.289101	0.236815	
malic_acid	0.437776	0.094397	1.000000	0.164045	0.288500	-0.054575	-0.335167	-0.411007	
ash	-0.049643	0.211545	0.164045	1.000000	0.443367	0.286587	0.128980	0.115077	
alcalinity_of_ash	0.517859	-0.310235	0.288500	0.443367	1.000000	-0.083333	-0.321113	-0.351370	
magnesium	-0.209179	0.270798	-0.054575	0.286587	-0.083333	1.000000	0.214401	0.195784	
total_phenols	-0.719163	0.289101	-0.335167	0.128980	-0.321113	0.214401	1.000000	0.864564	
flavanoids	-0.847498	0.236815	-0.411007	0.115077	-0.351370	0.195784	0.864564	1.000000	
nonflavanoid_phenols	0.489109	-0.155929	0.292977	0.186230	0.361922	-0.256294	-0.449935	-0.537900	
proanthocyanins	-0.499130	0.136698	-0.220746	0.009652	-0.197327	0.236441	0.612413	0.652692	
color_intensity	0.265668	0.546364	0.248985	0.258887	0.018732	0.199950	-0.055136	-0.172379	

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)

T.E/SEM VI/CBCGS/AIML
Academic Year: 2022-23

NAME	**SINGH SUDHAM DHARMENDRA**
BRANCH	**CSE-(AI&ML)**
ROLL NO.	**57**
SUBJECT	**DATA ANALYTICS AND VISUALIZATION LAB**
COURSE CODE	**CSL601**
PRACTICAL NO.	
DOP	**23/02/2023**
DOS	

Simple linear regression

- **Bitcoin and prediction**

Part 1

Importing important modules and excel-csv dataset file

In [1]:
```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
```

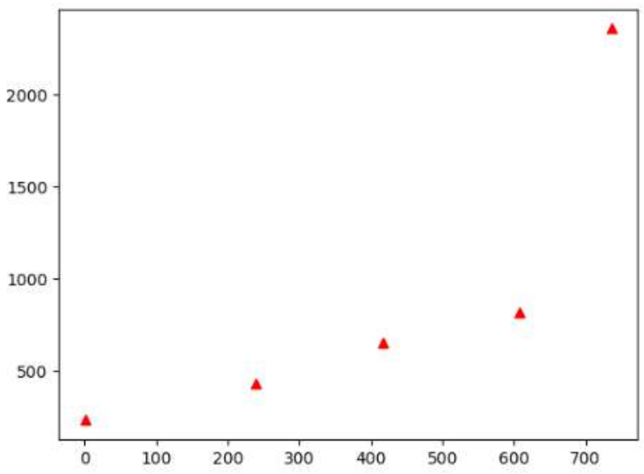
In [2]:
```
df=pd.read_csv("D:\StudyTime\TE\SEM6\DAVL\SUDHAM\dataset.csv")
df
```

Out[2]:

	Sr. No.	Bitcoin Price	No. Of Days
0	0	234.31	1
1	1	431.76	240
2	2	652.14	417
3	3	817.26	607
4	4	2358.96	736

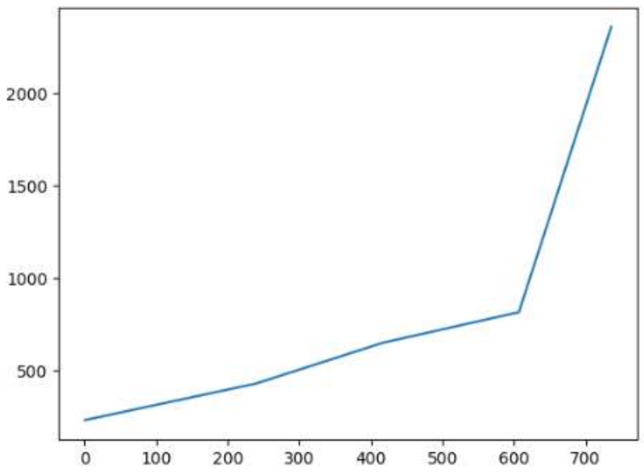
In [3]:
```
x=df['No. Of Days']
y=df['Bitcoin Price']
plt.scatter(x,y,color='red',marker='^')
```

Out[3]: <matplotlib.collections.PathCollection at 0x14740111d60>



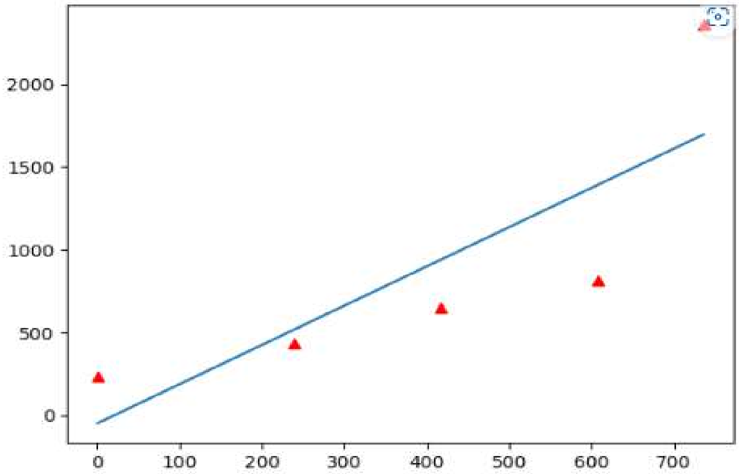
In [4]:
```
b,a=np.polyfit(x,y,1)
plt.plot(x,y)
```

Out[4]: [<matplotlib.lines.Line2D at 0x14740856160>]



Part 2
Testing/Predicting the dataset

In [5]:
```
plt.scatter(x,y,color='red',marker='^')
b,a=np.polyfit(x,y,1)
y1=a+b*x
plt.plot(x,y1)
plt.show()
```



In [6]:
```
y1=a+b*800
y1
```

Out[6]: 1848.313563392002

In [7]:
```
y1=a+b*1000
y1
```

Out[7]: 2323.264820716665

In [8]:
```
df=pd.read_csv("D:\StudyTime\TE\SEM6\DAVL\SUDHAM\dataset1.csv")
df
```

Out[8]:

	Sr. No.	Bitcoin Price	No. Of Days	Days
0	0	234.31	1	800
1	1	431.76	240	900
2	2	652.14	417	1000
3	3	817.26	607	1500
4	4	2358.96	736	2000

In [9]:
```
y1=a+b*df.Days
y1
```

Out[9]:
```
0    1848.313563
1    2085.789192
2    2323.264821
3    3510.642964
4    4698.021107
Name: Days, dtype: float64
```

In [11]:
```
df['price']=y1
df
```

Out[11]:

	Sr. No.	Bitcoin Price	No. Of Days	Days	price
0	0	234.31	1	800	1848.313563
1	1	431.76	240	900	2085.789192
2	2	652.14	417	1000	2323.264821
3	3	817.26	607	1500	3510.642964
4	4	2358.96	736	2000	4698.021107

In [13]:
```
df.to_clipboard()
```

- **Experience and salary**

Part 1

Importing important modules and excel-csv dataset file

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

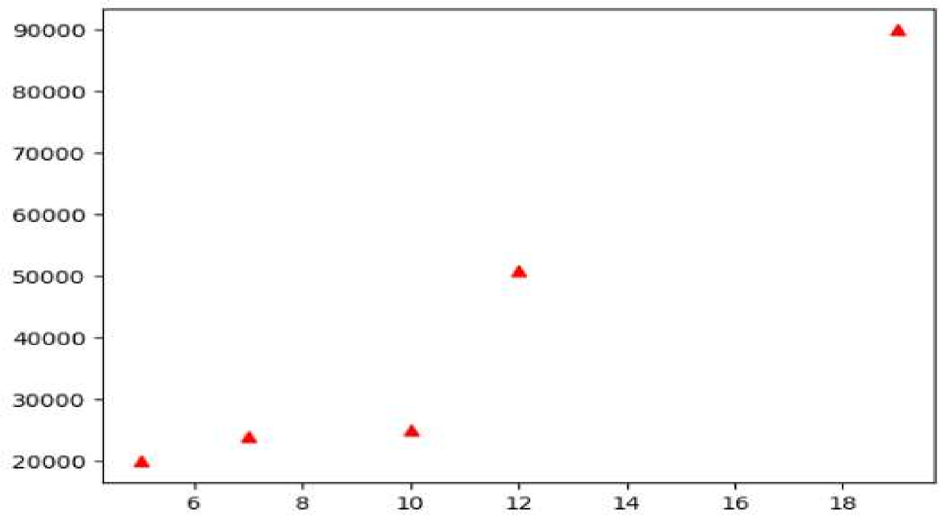
In [2]:
```
df=pd.read_excel('D:/StudyTime/TE/SEM6/DAVL/employee_data.xls.ods')
df
```

Out[2]:

	Sr. No.	Experience	Salary
0	0	5	20000
1	1	7	24000
2	2	10	25000
3	3	12	51000
4	4	19	90000

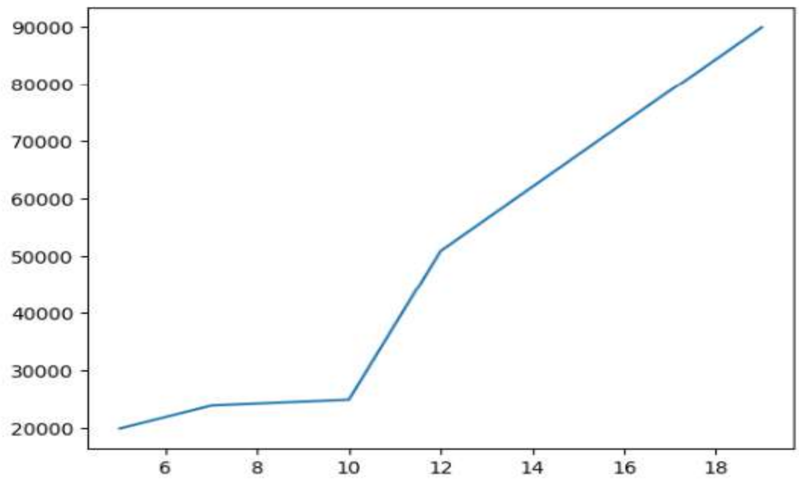
In [3]:
```
x = df['Experience']
y = df['Salary']
plt.scatter(x,y,color='red',marker='^')
```

Out[3]: <matplotlib.collections.PathCollection at 0x16386571a90>



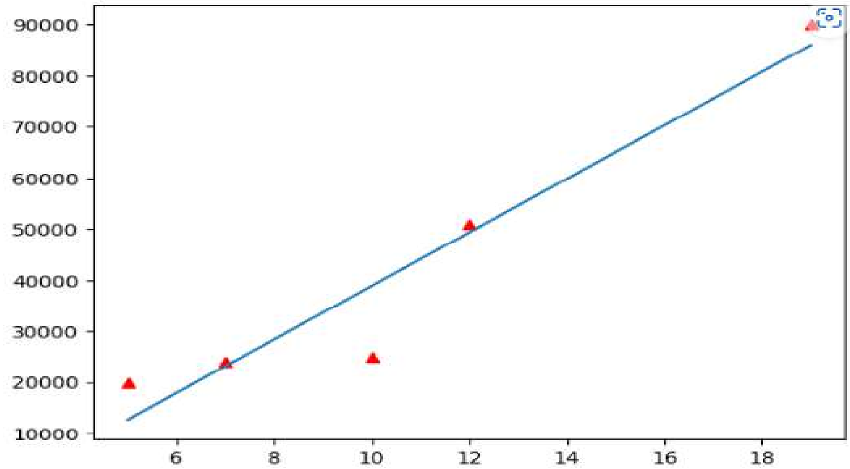
In [5]:
```
b,a=np.polyfit(x,y,1)
plt.plot(x,y)
```

Out[5]: [<matplotlib.lines.Line2D at 0x16386e7d280>]



Part 2
Testing/Predicting the dataset

In [6]:
```
plt.scatter(x,y,color='red',marker='^')
b,a=np.polyfit(x,y,1)
y1=a+b*x
plt.plot(x,y1)
plt.show()
```



In [7]:
```
y1=a+b*24
y1
```
Out[7]: 112201.36518771334

In [9]:
```
df=pd.read_excel('D:/StudyTime/TE/SEM6/DAVL/employee_data1.ods')
df
```
Out[9]:

	Sr. No.	Experience	Salary	Increment
0	0	5	20000	1000
1	1	7	24000	1200
2	2	10	25000	3000
3	3	12	51000	7600
4	4	19	90000	9990

In [10]:
```
y1=a+b*df.Increment
y1
```
Out[10]:
```
0    5.225375e+06
1    6.273157e+06
2    1.570319e+07
3    3.980217e+07
4    5.232316e+07
Name: Increment, dtype: float64
```

In [11]:
```
df['Added']=y1
df
```
Out[11]:

	Sr. No.	Experience	Salary	Increment	Added
0	0	5	20000	1000	5.225375e+06
1	1	7	24000	1200	6.273157e+06
2	2	10	25000	3000	1.570319e+07
3	3	12	51000	7600	3.980217e+07
4	4	19	90000	9990	5.232316e+07

In [16]:
```
df['Bonus']=a+b*df.Increment
df
```
Out[16]:

	Sr. No.	Experience	Salary	Increment	Added	Bonus
0	0	5	20000	1000	5.225375e+06	5.225375e+06
1	1	7	24000	1200	6.273157e+06	6.273157e+06
2	2	10	25000	3000	1.570319e+07	1.570319e+07
3	3	12	51000	7600	3.980217e+07	3.980217e+07
4	4	19	90000	9990	5.232316e+07	5.232316e+07

AIML57_SUDHAM

In [ ]:
```
df.to_clipboard()
```

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)

T.E/SEM VI/CBCGS/AIML
Academic Year: 2022-23

NAME	**SINGH SUDHAM DHARMENDRA**
BRANCH	**CSE-(AI&ML)**
ROLL NO.	**57**
SUBJECT	**DATA ANALYTICS AND VISUALIZATION LAB**
COURSE CODE	**CSL601**
PRACTICAL NO.	**0**
DOP	
DOS	

Experiment No. : 06

Aim : Implementation of Spam filter/Sentiment analysis in python/R.

Theory :

- **Text analytics** is a branch of natural language processing (NLP) that deals with the automated processing and analysis of large amounts of unstructured text data. Two common applications of text analytics are spam filtering and sentiment analysis.
- **Spam filters** are used to automatically identify and remove unwanted or unsolicited emails, messages or comments. Spam filters typically use machine learning algorithms to learn from a large set of examples and identify patterns that distinguish spam from legitimate messages. They may also use various text processing techniques such as content analysis, text classification, and clustering to identify spam messages.
- **Sentiment analysis**, on the other hand, is the process of automatically detecting the sentiment or emotion expressed in a piece of text, such as a tweet or a review. Sentiment analysis is used to analyze customer feedback, social media posts, and other types of user-generated content to understand the overall sentiment towards a product or service. Sentiment analysis algorithms typically use natural language processing techniques such as part-of-speech tagging, parsing, and machine learning to identify and classify the sentiment expressed in text as positive, negative or neutral.

  Overall, text analytics is a powerful tool for businesses and organizations to gain insights from large volumes of text data, including social media posts, customer reviews, and feedback, and make data driven decisions

Implementation :

Write a program to perform Spam filter/Sentiment analysis in python :

- **Program for Spam filter:**

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
# Load the data
data = pd.read_csv("spam.csv", encoding="latin-1")
# Split the data into training and testing sets
train_data, test_data, train_target, test_target = train_test_split(data["text"], data["class"],
test_size=0.2)
# Vectorize the text data
vectorizer = CountVectorizer()
train_features = vectorizer.fit_transform(train_data)
test_features = vectorizer.transform(test_data)
# Fit a Naive Bayes model
model = MultinomialNB()
model.fit(train_features, train_target)
# Evaluate the model
```

```
accuracy = model.score(test_features, test_target)
print("Accuracy:", accuracy)
# Test the model with new data
new_data = ["Congratulations! You've won a free vacation to Hawaii. Reply now to claim your prize."]
new_features = vectorizer.transform(new_data)
prediction = model.predict(new_features)
print("Prediction:", prediction[0])
```

Output -

Accuracy: 0.9865470852017937
Prediction: spam

- **Program for Sentiment analysis**

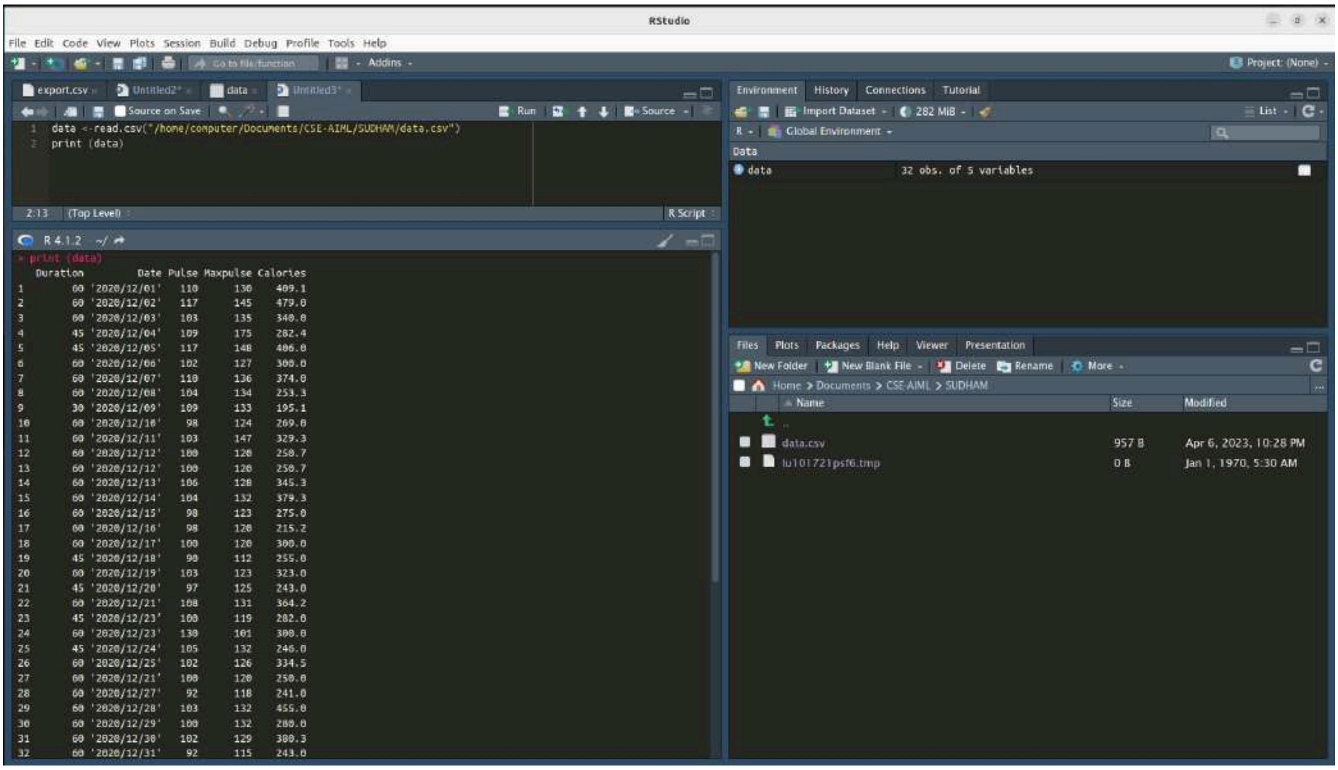
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
# Load the data
data = pd.read_csv("reviews.csv", encoding="latin-1")
# Split the data into training and testing sets
train_data, test_data, train_target, test_target = train_test_split(data["text"],
data["sentiment"], test_size=0.2)
# Vectorize the text data
vectorizer = CountVectorizer()
train_features = vectorizer.fit_transform(train_data)
test_features = vectorizer.transform(test_data)
# Fit a Naive Bayes model
model = MultinomialNB()
model.fit(train_features, train_target)
# Evaluate the model
accuracy = model.score(test_features, test_target)
print("Accuracy:", accuracy)
# Test the model with new data
new_data = ["This movie was great!"]
new_features = vectorizer.transform(new_data)
prediction = model.predict(new_features)
 print("Prediction:", prediction[0])
```

Output :

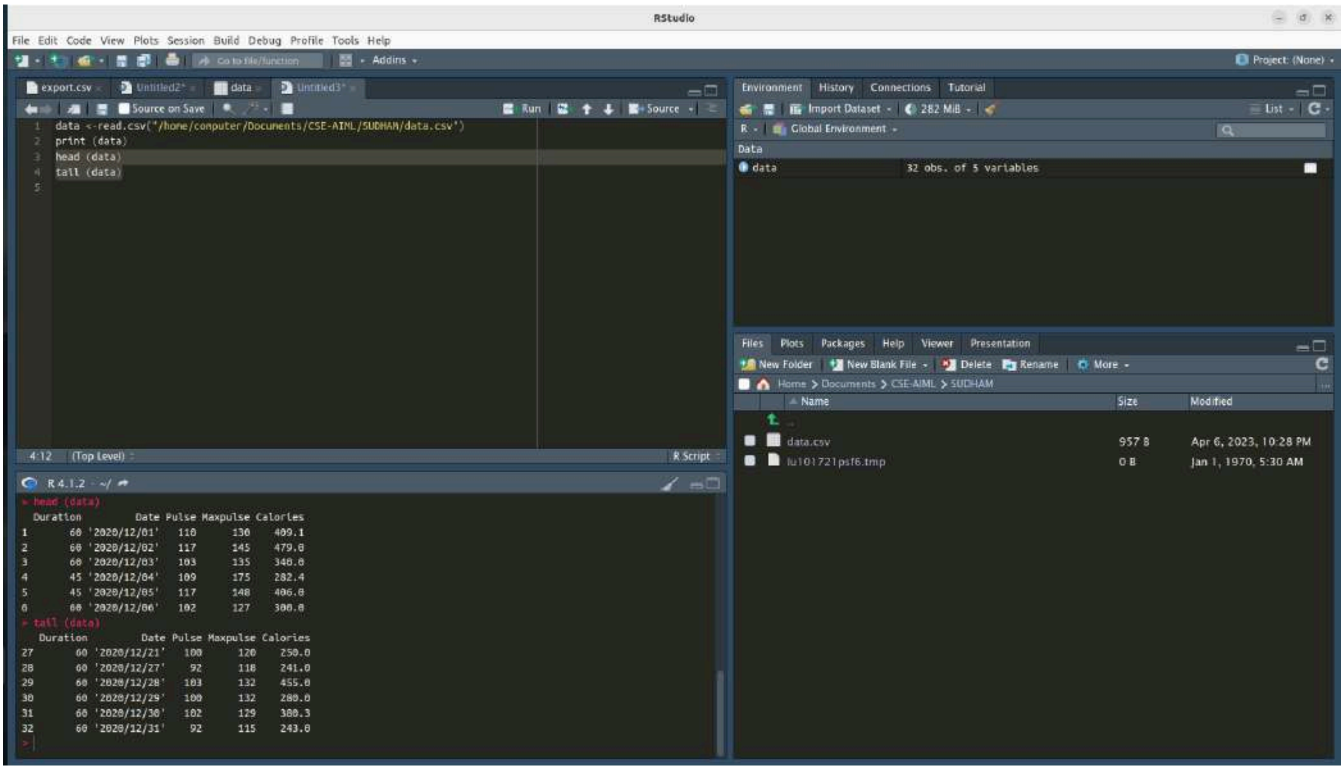
Accuracy: 0.8271
Prediction: positive

Conclusion : We had successfully studied and understood Spam filter/Sentiment analysis in Python/R

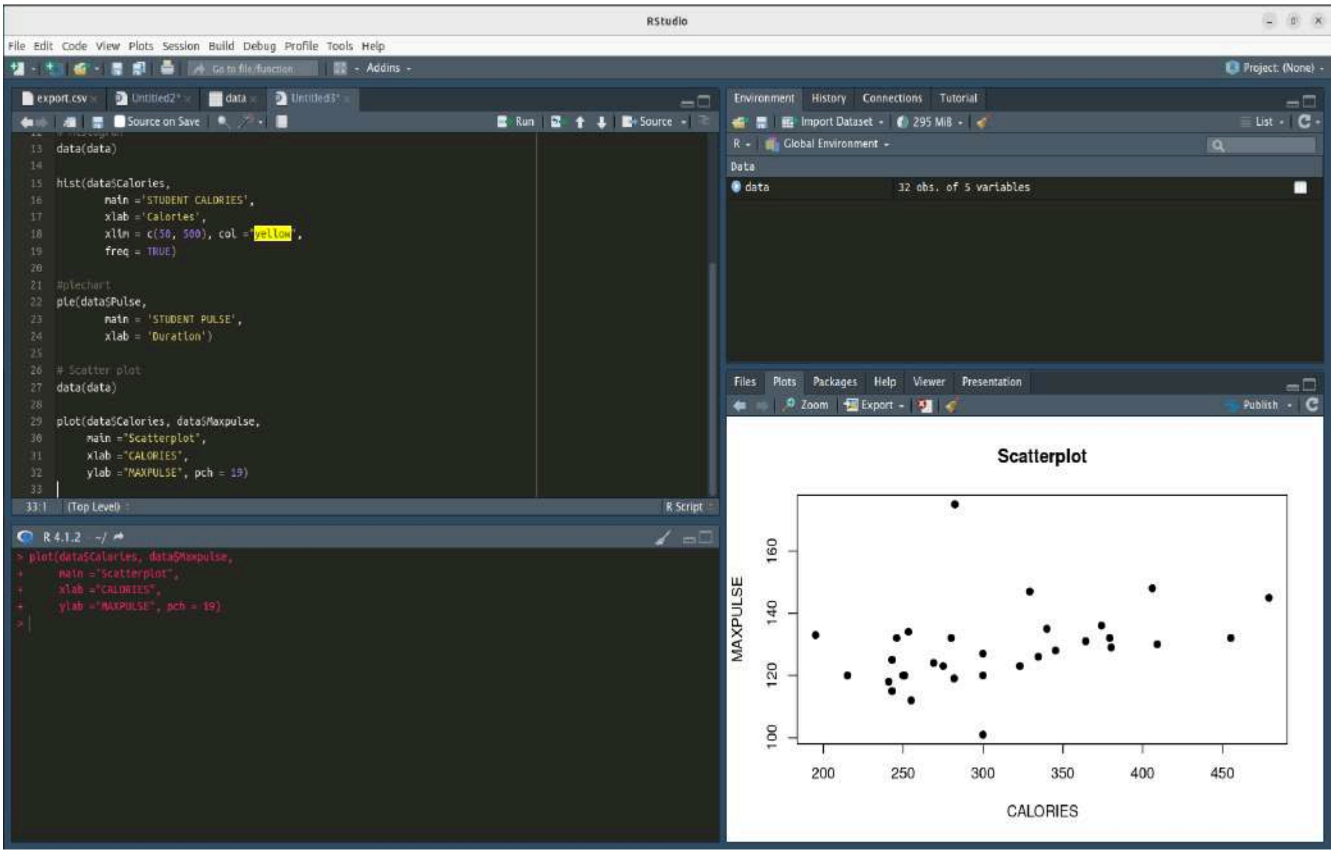
- **Importing dataset**



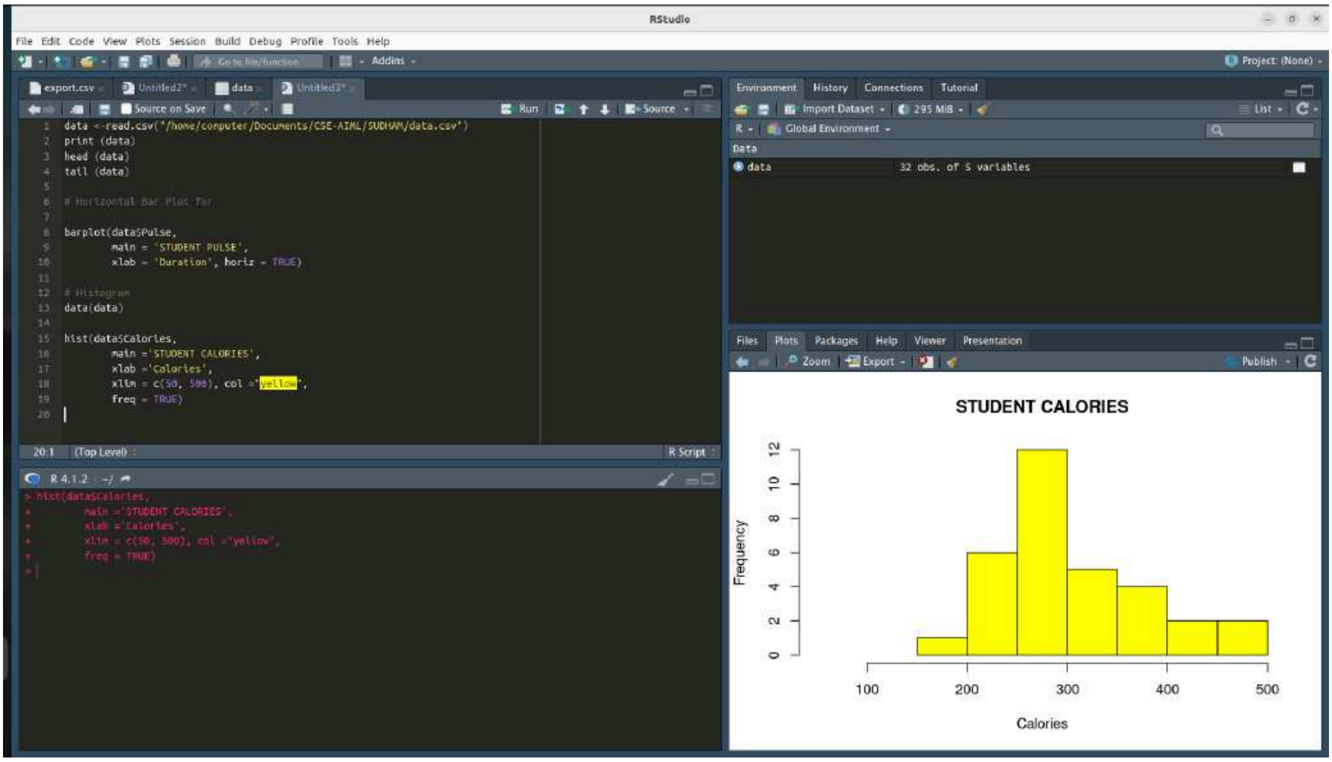
- **Getting quick overview**



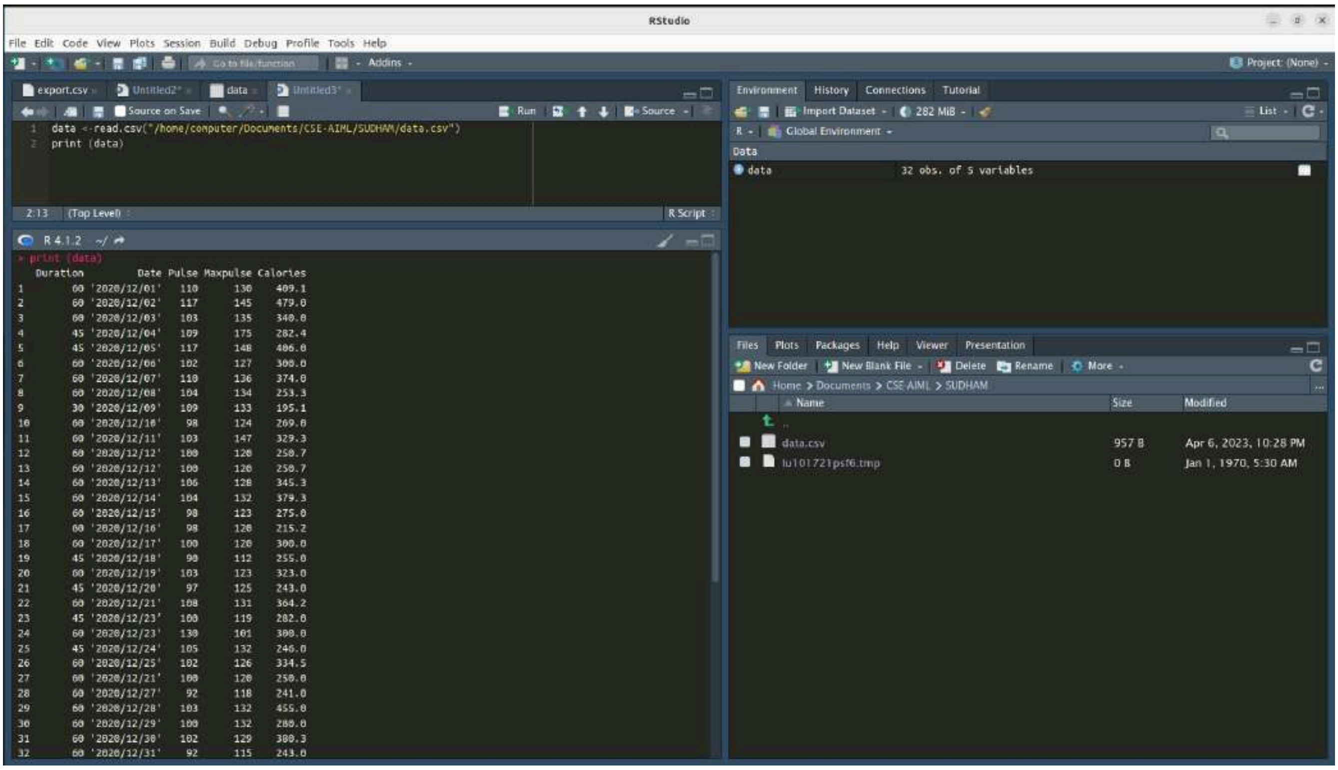
- **Scatter-plot Visualization in R**



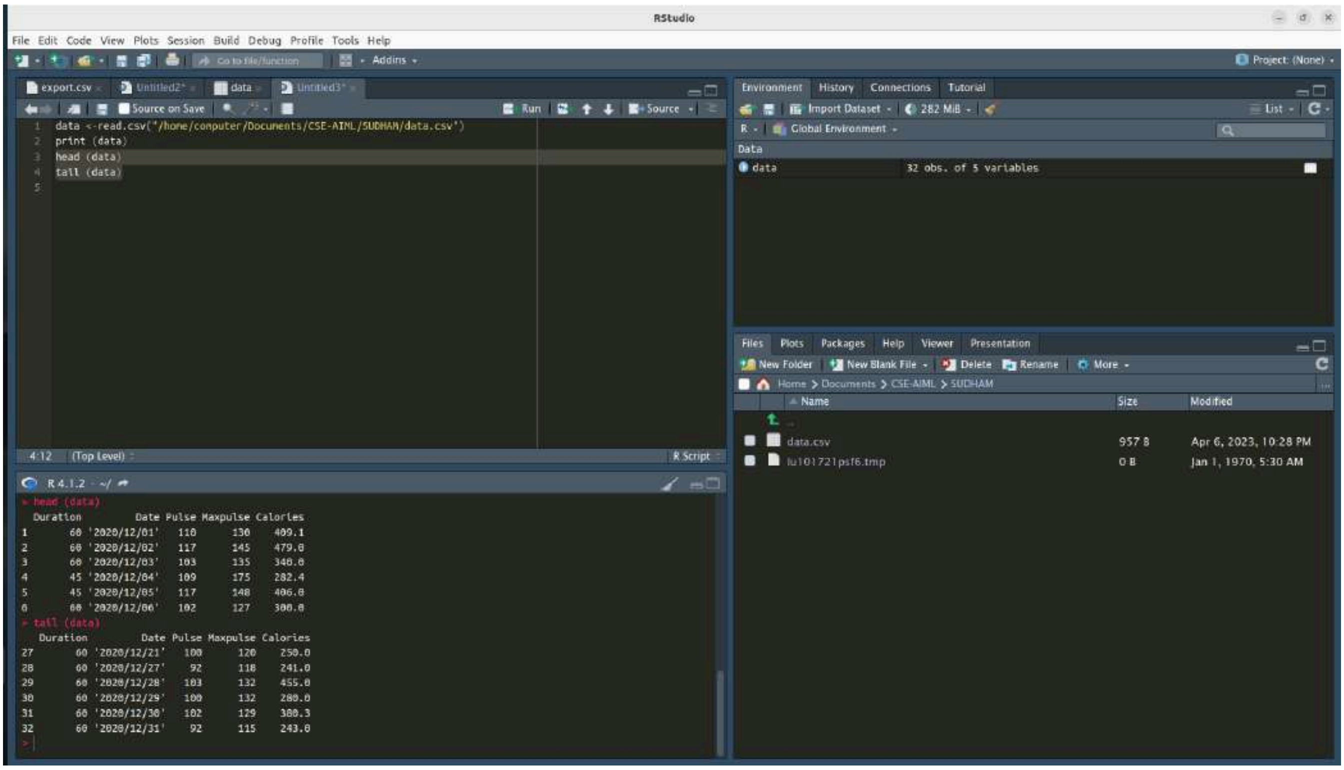
- **Histogram Visualization in R**



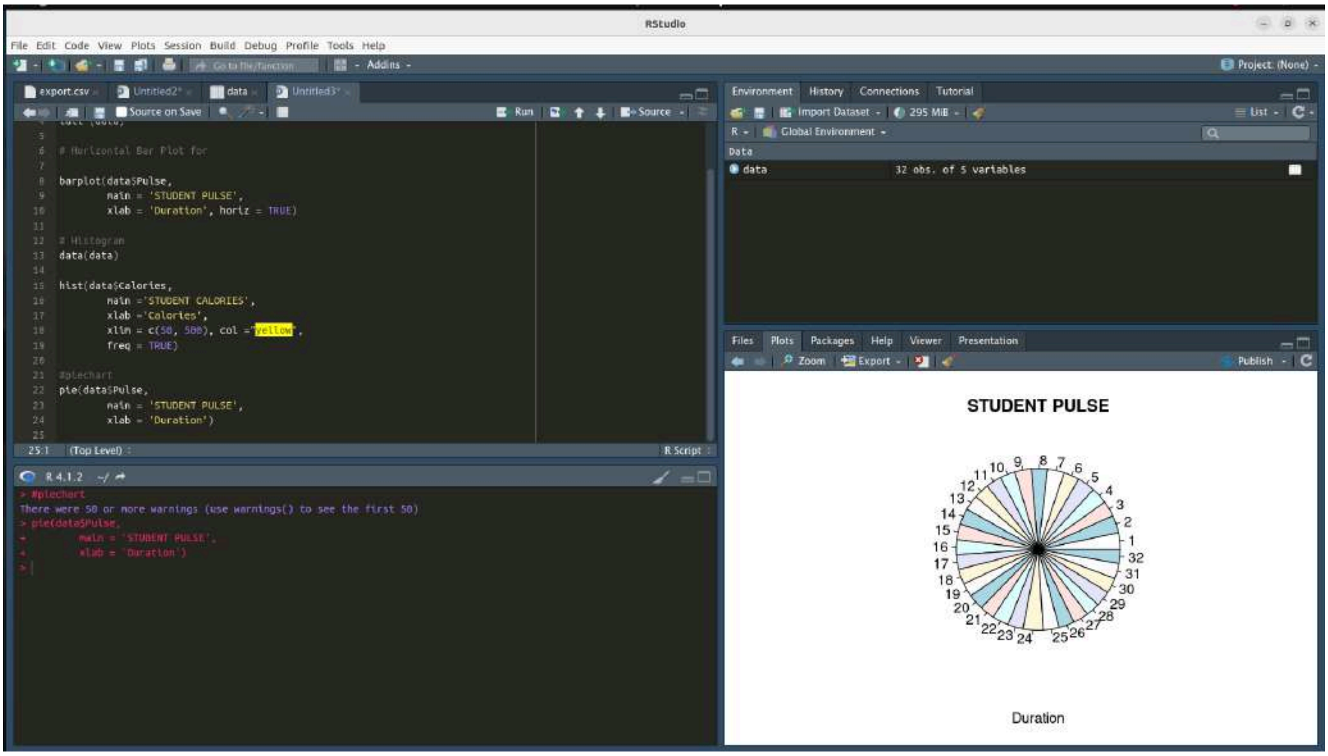
- **Importing dataset**



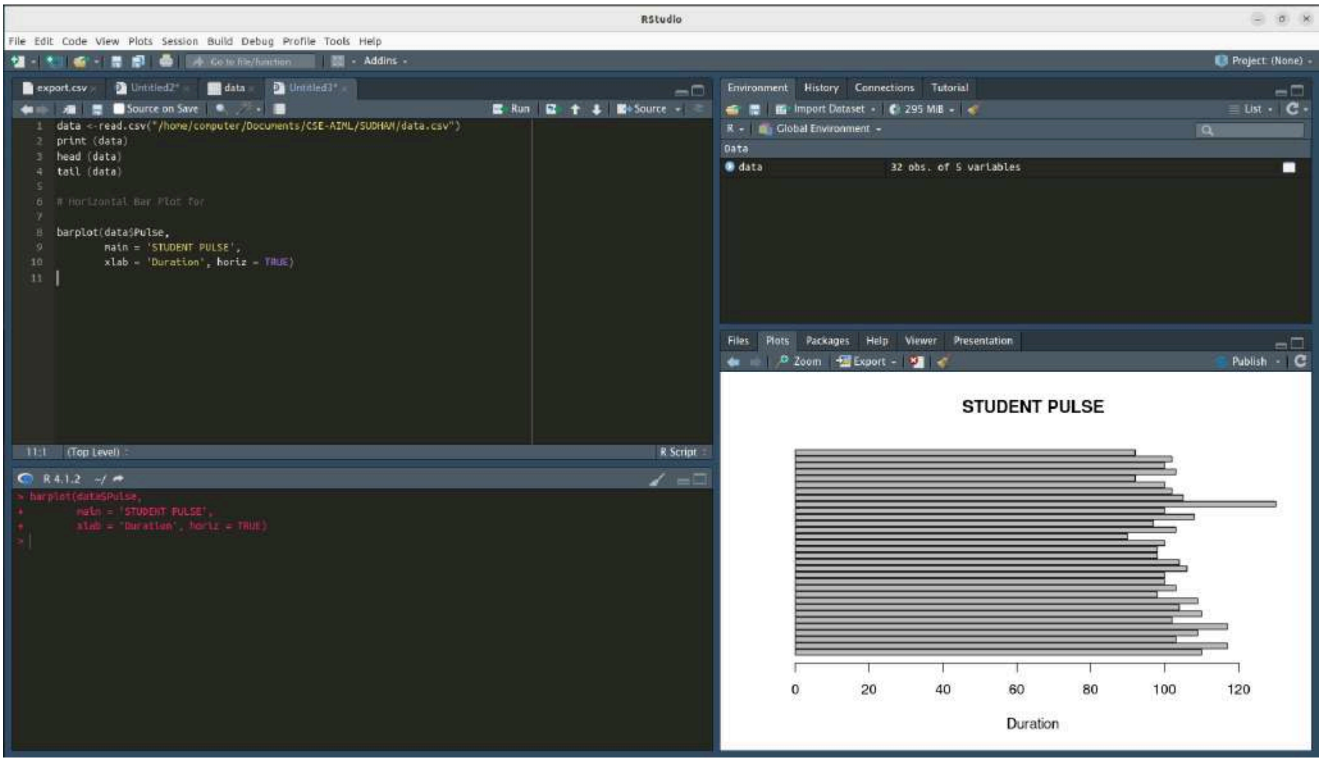
- **Getting quick overview**



- **Pie-Chart Visualization in R**

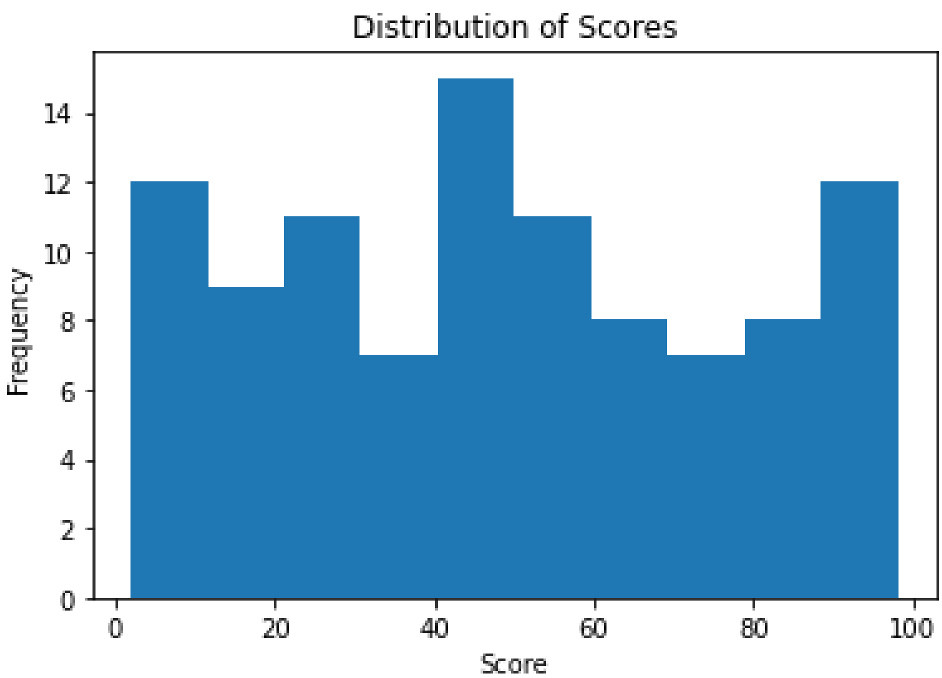


- **Bar-Chart Visualization in R**



Pandas - Histogram :

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

# Generate random data
data = pd.DataFrame({
    'scores': np.random.randint(0, 100, size=100)
})

# Create a histogram of the data
plt.hist(data['scores'], bins=10)

# Set the title and axis labels
plt.title('Distribution of Scores')
plt.xlabel('Score')
plt.ylabel('Frequency')

# Display the plot
plt.show()
```



Program :
Seaborn - barplot :

```
import seaborn as sns
import matplotlib.pyplot as plt

# Define your data
x = ["A", "B", "C", "D"]
y = [10, 20, 30, 40]

# Create a Seaborn bar plot
sns.set_style("whitegrid")  # Set the plot style
sns.barplot(x=x, y=y)  # Create the bar plot
plt.xlabel("Categories")  # Set the x-axis label
plt.ylabel("Count")  # Set the y-axis label
plt.title("Bar Plot Example")  # Set the plot title
plt.show()  # Display the plot
```

Output :