```
class human:
    def __init__(self,n,o):
        self.name=n;
        self.occupation=o;
    def do_work(self):
        if self.occupation=="tennis palyer":
            print(self.name,"plays tennis")
        elif self.occupation=="shoots movie":
            print(self.name,"shoots films")
        elif self.name=="Ayush Shaurya Jha" :
            print(self.name,"(Legend) make other work for them")
    def speaks(self):
        print(self.name,"speaks Hindi")
sania_mirza=human("Sania Mirza","tennis palyer")
ajay_devgan=human("Ajay devgan","shoots movie")
ayush=human("Ayush Shaurya Jha","Legend")

ayush.do_work()
sania_mirza.do_work()
ajay_devgan.speaks()
```

```
Ayush Shaurya Jha (Legend) make other work for them
Sania Mirza plays tennis
Ajay devgan speaks Hindi
```

```
%%HTML
<iframe width="1500" height="350" src="https://youtu.be/-YQ3ESLbx3U"
frameborder="85" allowfullscreen ></iframe>
```

```
<IPython.core.display.HTML object>
```

```
import numpy as np

a=np.array([5,6,9])
a[0]
a.ndim
```

```
1
```

```
a[1]
```

```
6
```

```
a=np.array([[1,2,3],[4,5,6],[7,8,9]],dtype=np.float64)
print(a.ndim)
print(a.itemsize)
```

```
2
8
```

```
a
```

```
array([[1., 2., 3.],
       [4., 5., 6.],
       [7., 8., 9.]])
```

```
  a.size
```

```
9
```

```
a.shape
```

```
(3, 3)
```

```
a=np.array([[1,2,3],[4,5,6],[7,8,9]],dtype=complex)
a
```

```
array([[1.+0.j, 2.+0.j, 3.+0.j],
       [4.+0.j, 5.+0.j, 6.+0.j],
       [7.+0.j, 8.+0.j, 9.+0.j]])
```

```
np.ones((3,6))
```

```
array([[1., 1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1.]])
```

```
np.zeros( (3,6) )
```

```
array([[0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0.]])
```

```
l=range(5)
l
```

```
range(0, 5)
```

```
l[0],l[1]
```

```
(0, 1)
```

```
l[1]
```

```
1
```

```
np.arange(1,5)
```

```
array([1, 2, 3, 4])
```

generates number b/w 1 and 5 with gap of 2

```
np.arange(1,5,2)
```

```
array([1, 3])
```

generates 10 linearly spaced numbers b/w 1 and 5

```
np.linspace(1,5,10)
```

```
array([1.        , 1.44444444, 1.88888889, 2.33333333, 2.77777778,
       3.22222222, 3.66666667, 4.11111111, 4.55555556, 5.        ])
```

```
 a
```

```
array([[1.+0.j, 2.+0.j, 3.+0.j],
       [4.+0.j, 5.+0.j, 6.+0.j],
       [7.+0.j, 8.+0.j, 9.+0.j]])
```

```
a.reshape(9,1)
```

```
array([[1.+0.j],
       [2.+0.j],
       [3.+0.j],
       [4.+0.j],
       [5.+0.j],
       [6.+0.j],
       [7.+0.j],
       [8.+0.j],
       [9.+0.j]])
```

```
a.reshape(1,9)
```

```
array([[1.+0.j, 2.+0.j, 3.+0.j, 4.+0.j, 5.+0.j, 6.+0.j, 7.+0.j,
8.+0.j,
        9.+0.j]])
```

functions don't alter shape of original array

```
a

array([[1.+0.j, 2.+0.j, 3.+0.j],
       [4.+0.j, 5.+0.j, 6.+0.j],
       [7.+0.j, 8.+0.j, 9.+0.j]])
```

flatten array using "ravel()"function and make it 1-D

```
a.ravel()

array([1.+0.j, 2.+0.j, 3.+0.j, 4.+0.j, 5.+0.j, 6.+0.j, 7.+0.j, 8.+0.j,
       9.+0.j])

a.max()

(9+0j)

a.min()

(1+0j)

a.sum()

(45+0j)
```

Axis-0 are Columns and Axis-1 are Rows

```
a.sum(axis=0)

array([12.+0.j, 15.+0.j, 18.+0.j])

a.sum(axis=1)

array([ 6.+0.j, 15.+0.j, 24.+0.j])

np.sqrt(a)

array([[1.        , 1.41421356, 1.73205081],
       [2.        , 2.23606798, 2.44948974],
       [2.64575131, 2.82842712, 3.        ]])
```

Standard Deviation of all numbers in array

```
y=np.std(a)
print(y)
```

```
2.581988897471611
```

```
y=np.std(a)
formatted_y = "%.2f"%y
print(formatted_y)
```

```
2.58
```

```
c=np.array([[1,2],[3,4]])
d=np.array([[5,6],[7,8]])
```

```
c+d
```

```
array([[ 6,  8],
       [10, 12]])
```

```
c*d
```

```
array([[ 5, 12],
       [21, 32]])
```

```
c/d
```

```
array([[0.2       , 0.33333333],
       [0.42857143, 0.5       ]])
```

DOT() do matrix product of two arrays

```
c.dot(d)
```

```
array([[19, 22],
       [43, 50]])
```

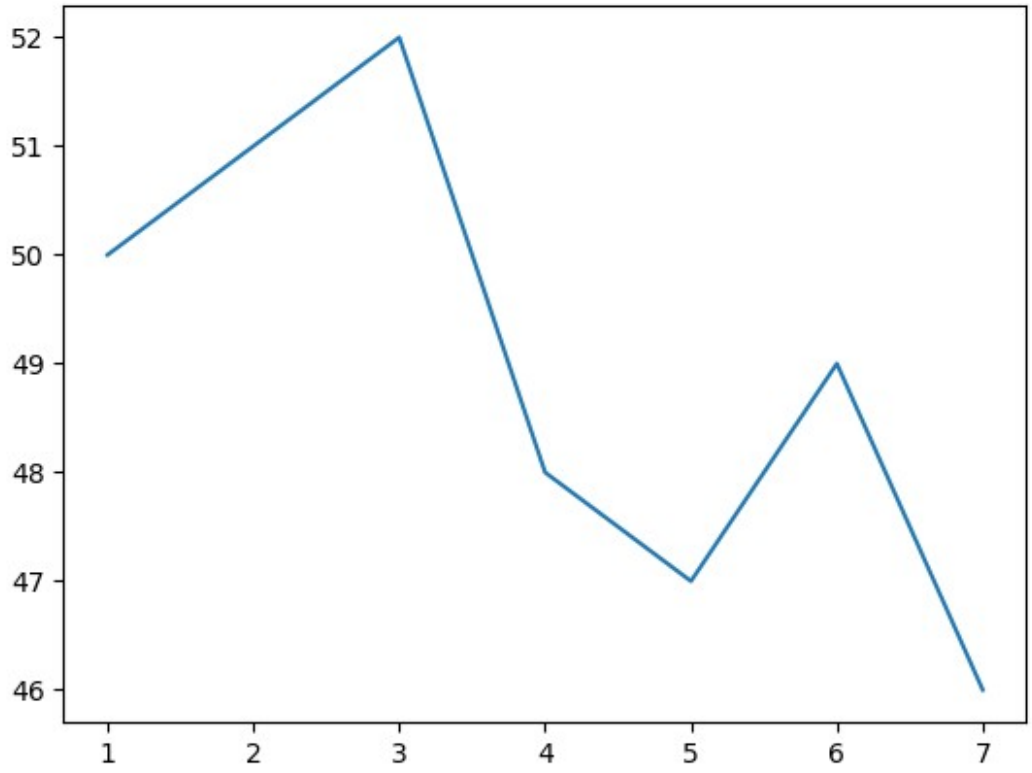
Thank You

```
pip install matplotlib
```

```
Defaulting to user installation because normal site-packages is not
writeable
Requirement already satisfied: matplotlib in c:\programdata\anaconda3\
lib\site-packages (3.7.0)
Requirement already satisfied: numpy>=1.20 in c:\programdata\
anaconda3\lib\site-packages (from matplotlib) (1.23.5)
Requirement already satisfied: packaging>=20.0 in c:\programdata\
anaconda3\lib\site-packages (from matplotlib) (22.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\programdata\
anaconda3\lib\site-packages (from matplotlib) (3.0.9)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\programdata\
anaconda3\lib\site-packages (from matplotlib) (1.4.4)
Requirement already satisfied: python-dateutil>=2.7 in c:\programdata\
anaconda3\lib\site-packages (from matplotlib) (2.8.2)
Requirement already satisfied: pillow>=6.2.0 in c:\programdata\
anaconda3\lib\site-packages (from matplotlib) (9.4.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\programdata\
anaconda3\lib\site-packages (from matplotlib) (4.25.0)
Requirement already satisfied: cycler>=0.10 in c:\programdata\
anaconda3\lib\site-packages (from matplotlib) (0.11.0)
Requirement already satisfied: contourpy>=1.0.1 in c:\programdata\
anaconda3\lib\site-packages (from matplotlib) (1.0.5)
Requirement already satisfied: six>=1.5 in c:\programdata\anaconda3\
lib\site-packages (from python-dateutil>=2.7->matplotlib) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

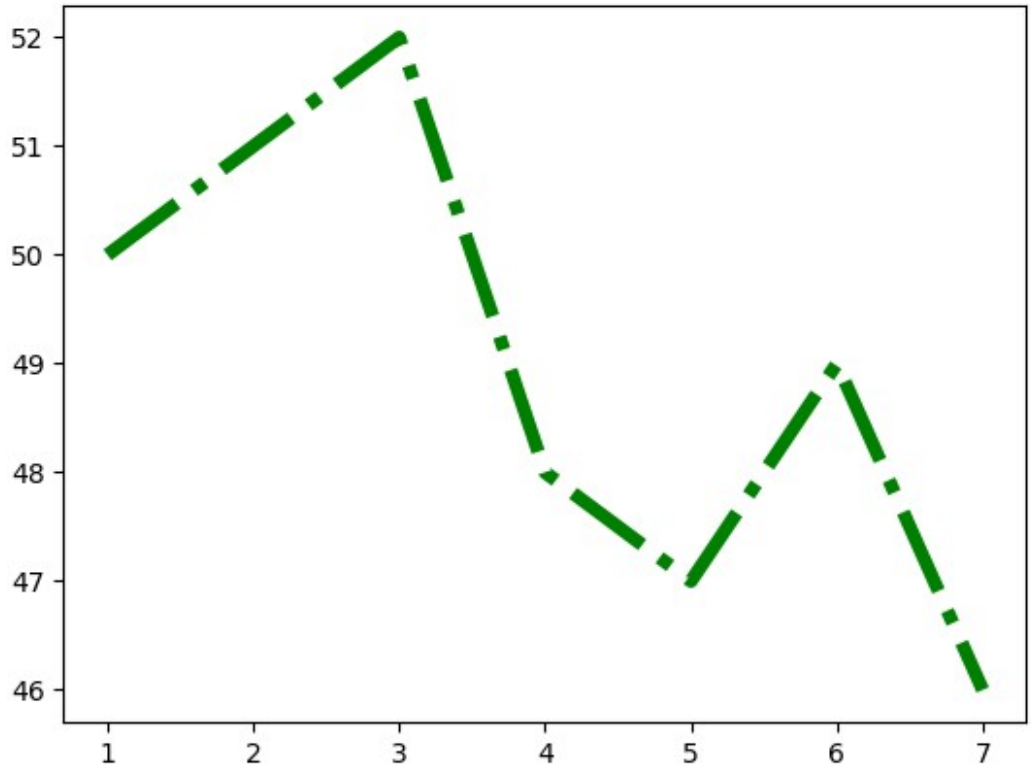
```
import matplotlib.pyplot as plt
%matplotlib inline

x=[1,2,3,4,5,6,7]
y=[50,51,52,48,47,49,46]

plt.plot(x,y)
```

```
[<matplotlib.lines.Line2D at 0x26d87ddca90>]
```



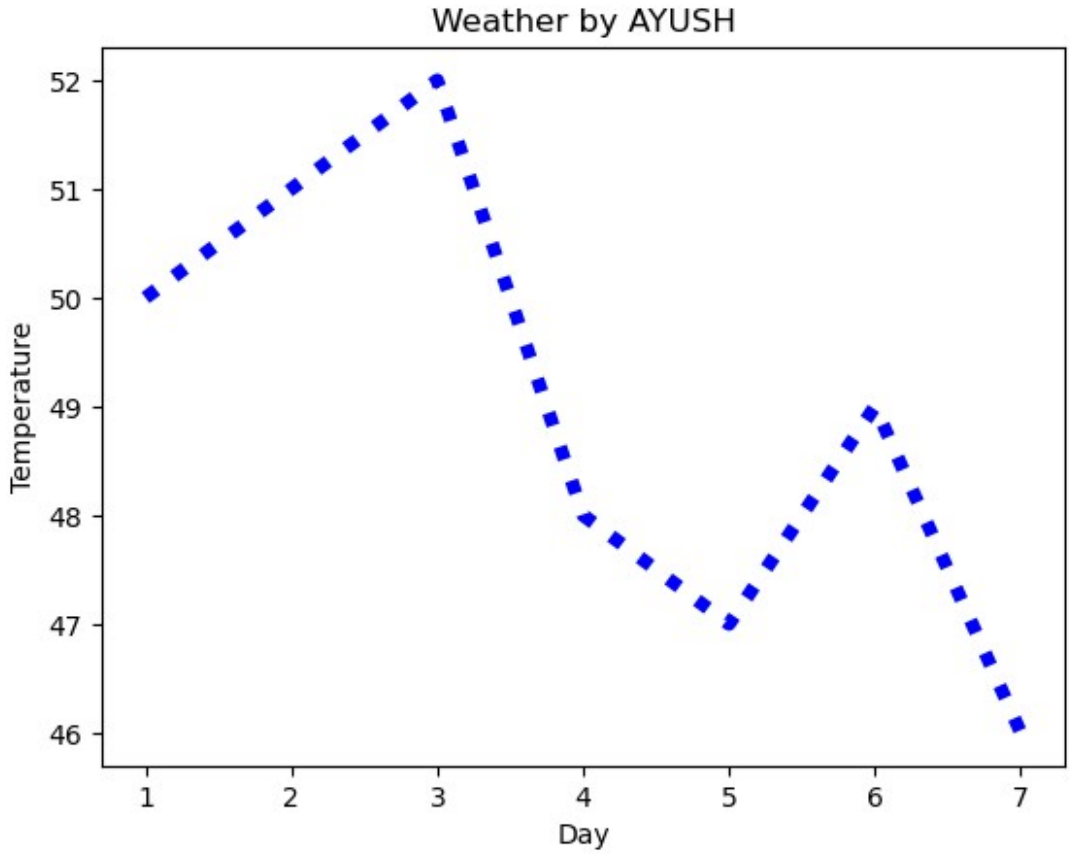

```
plt.plot(x,y,color='green',linewidth=5,linestyle="dashdot")
```

```
[<matplotlib.lines.Line2D at 0x26d8c489990>]
```




```
plt.xlabel("Day")
plt.ylabel("Temperature")
plt.title("Weather by AYUSH")
plt.plot(x,y,color='blue',linewidth=5,linestyle="dotted")
```

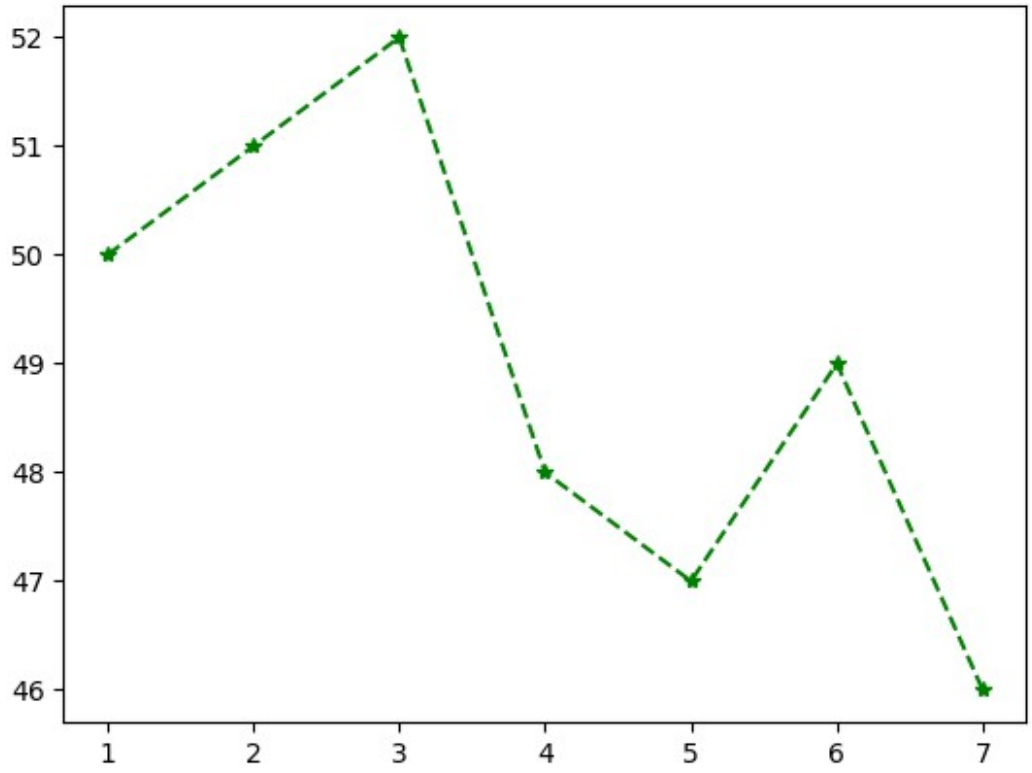
```
[<matplotlib.lines.Line2D at 0x1d106052e90>]
```



Lets see use Of format string

```
plt.plot(x,y,'g*--')
```

```
[<matplotlib.lines.Line2D at 0x26d8c336260>]
```

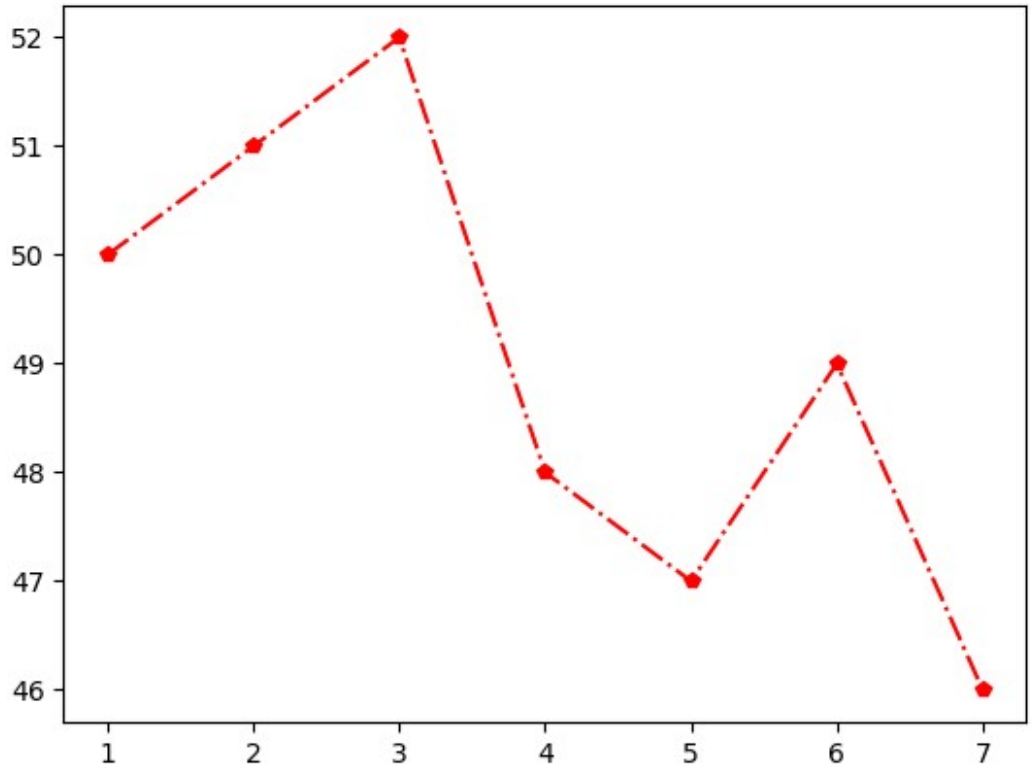



```
"""the above line g*-- indicates green dash dash(--)line with * mark
at points
.we can dash dot(-.)and many other styles,check at matplotlib
website"""
```

```
'the above line g*-- indicates green dash dash(--)line with * mark at
points\n.we can dash dot(-.)and many other styles,check at matplotlib
website'
```

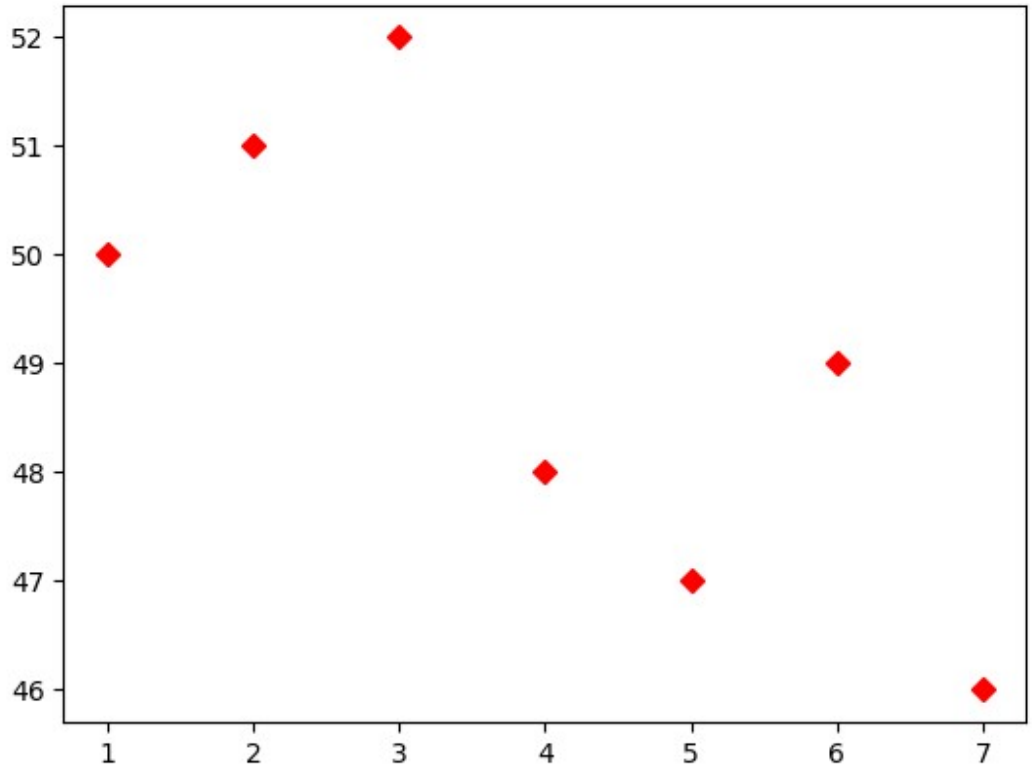
```
plt.plot(x,y,'-.pr')
"""pr is pentagon marker and r is red,D is for diamond marker"""
```

```
'pr is pentagon marker and r is red,D is for diamond marker'
```



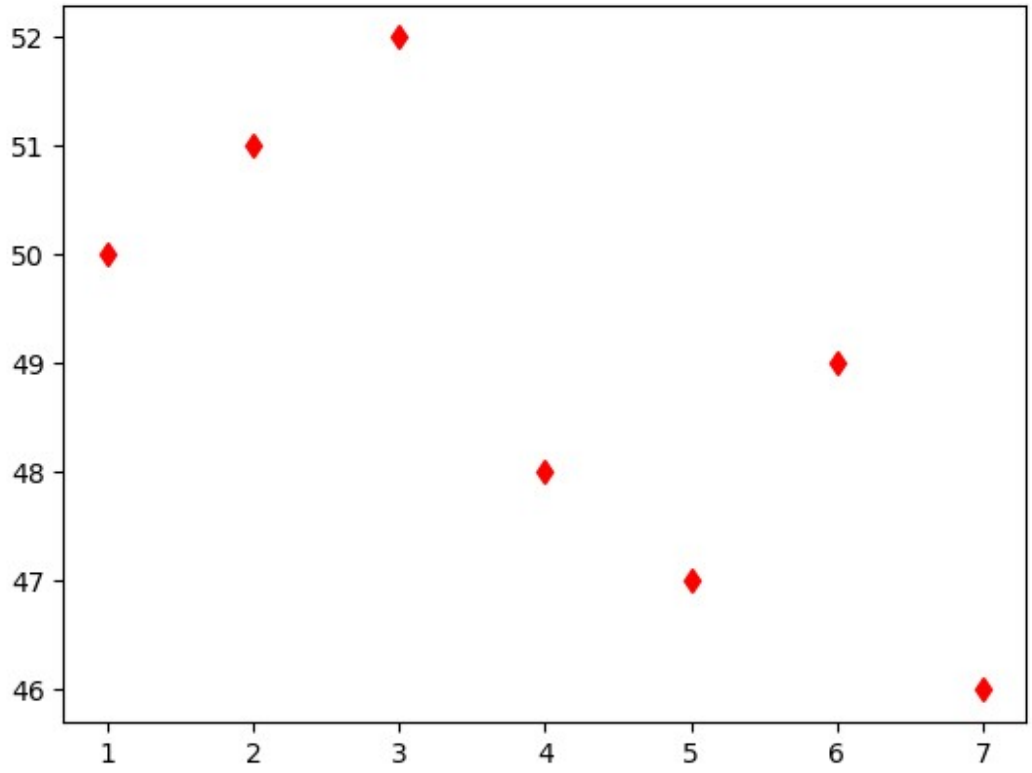

```
plt.plot(x,y,'rD')
```

```
[<matplotlib.lines.Line2D at 0x1d107285420>]
```



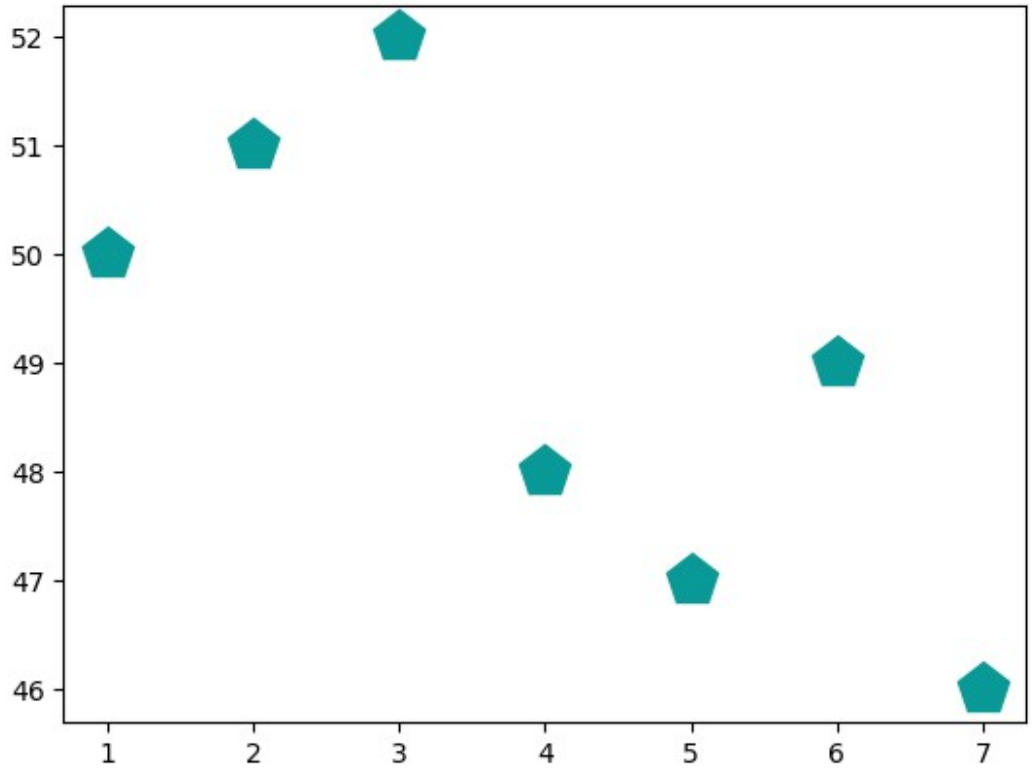

```
plt.plot(x,y,color='red',marker='d',linestyle='')
"""we can use any hexadecimal representation for colors"""
```

```
'we can use any hexadecimal representation for colors'
```



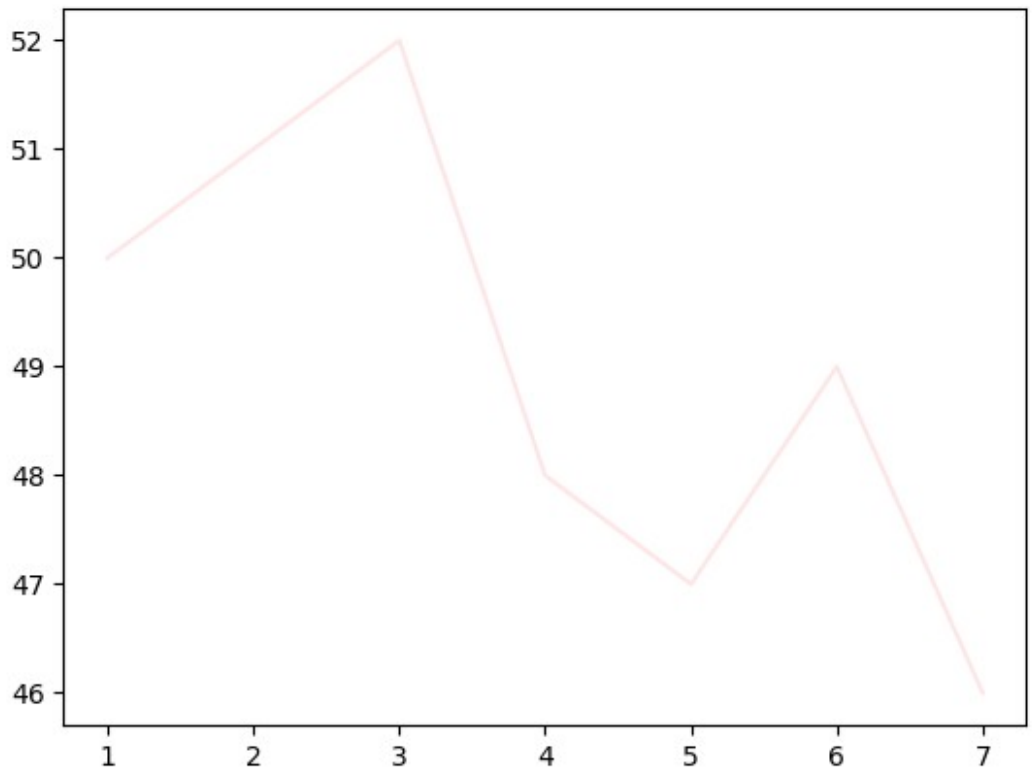

```
plt.plot(x,y,color='#099997',marker='p',linestyle='',markersize=20)
```

```
[<matplotlib.lines.Line2D at 0x26d8e6ab250>]
```




```
plt.plot(x,y,color='red',alpha=.1)
```

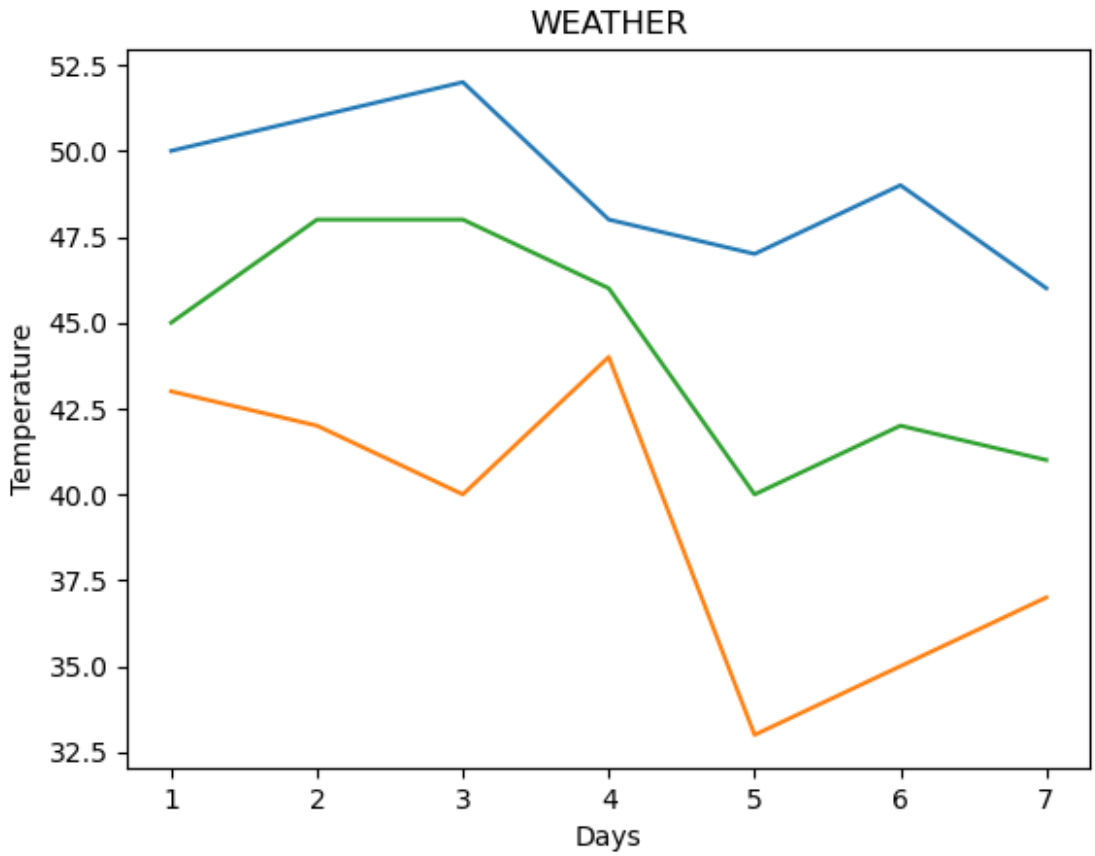
```
[<matplotlib.lines.Line2D at 0x26d8e3ad4e0>]
```



Alpha is transparency factor ,at alpha=0,its not visible and at alpha =1 its opaque

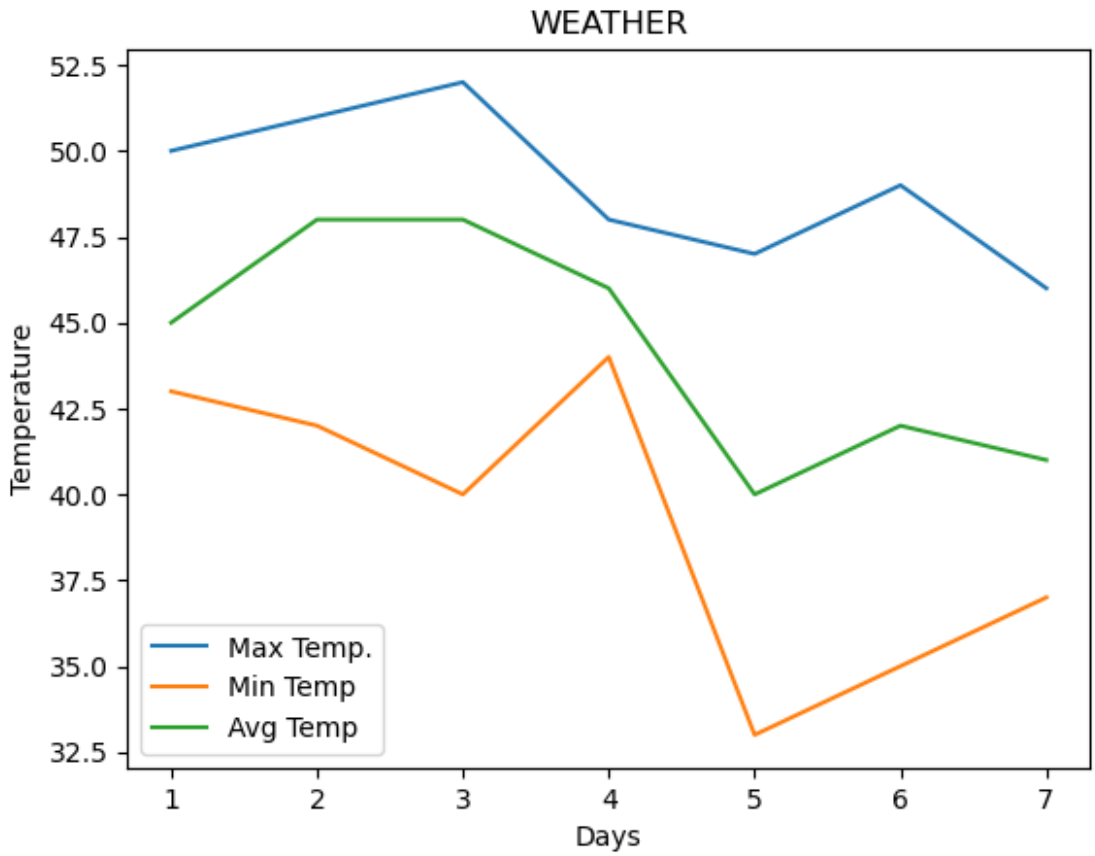
```
days=[1,2,3,4,5,6,7]
max_t=[50,51,52,48,47,49,46]
min_t=[43,42,40,44,33,35,37]
avg_t=[45,48,48,46,40,42,41]

plt.xlabel("Days")
plt.ylabel("Temperature")
plt.title("WEATHER")
plt.plot(days,max_t,label="Max Temp.")
plt.plot(days,min_t,label="Min Temp")
plt.plot(days,avg_t,label="Avg Temp")
```

```
[<matplotlib.lines.Line2D at 0x26d8df63fa0>]
```




```
plt.xlabel("Days")
plt.ylabel("Temperature")
plt.title("WEATHER")
plt.plot(days,max_t,label="Max Temp.")
plt.plot(days,min_t,label="Min Temp")
plt.plot(days,avg_t,label="Avg Temp")
plt.legend()
```

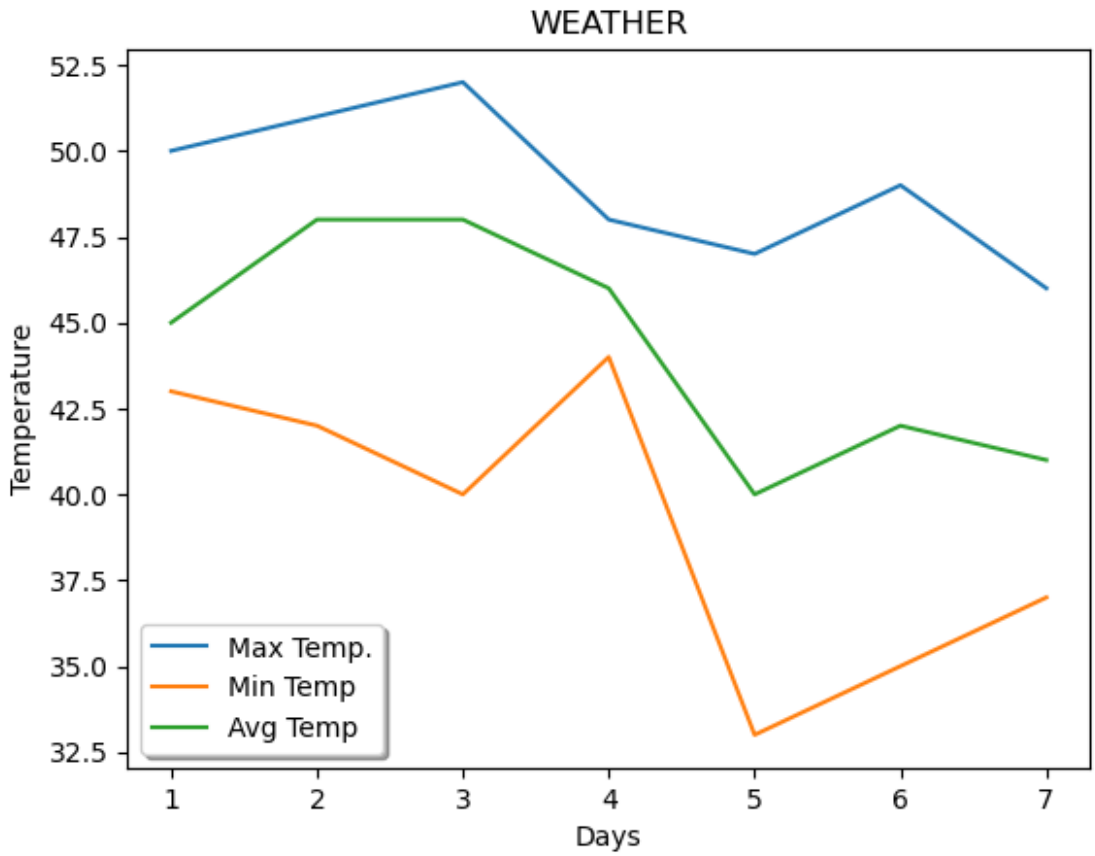
```
<matplotlib.legend.Legend at 0x26d8fe0ba30>
```



By default legend moves smartly to place where its finds a space for itself.

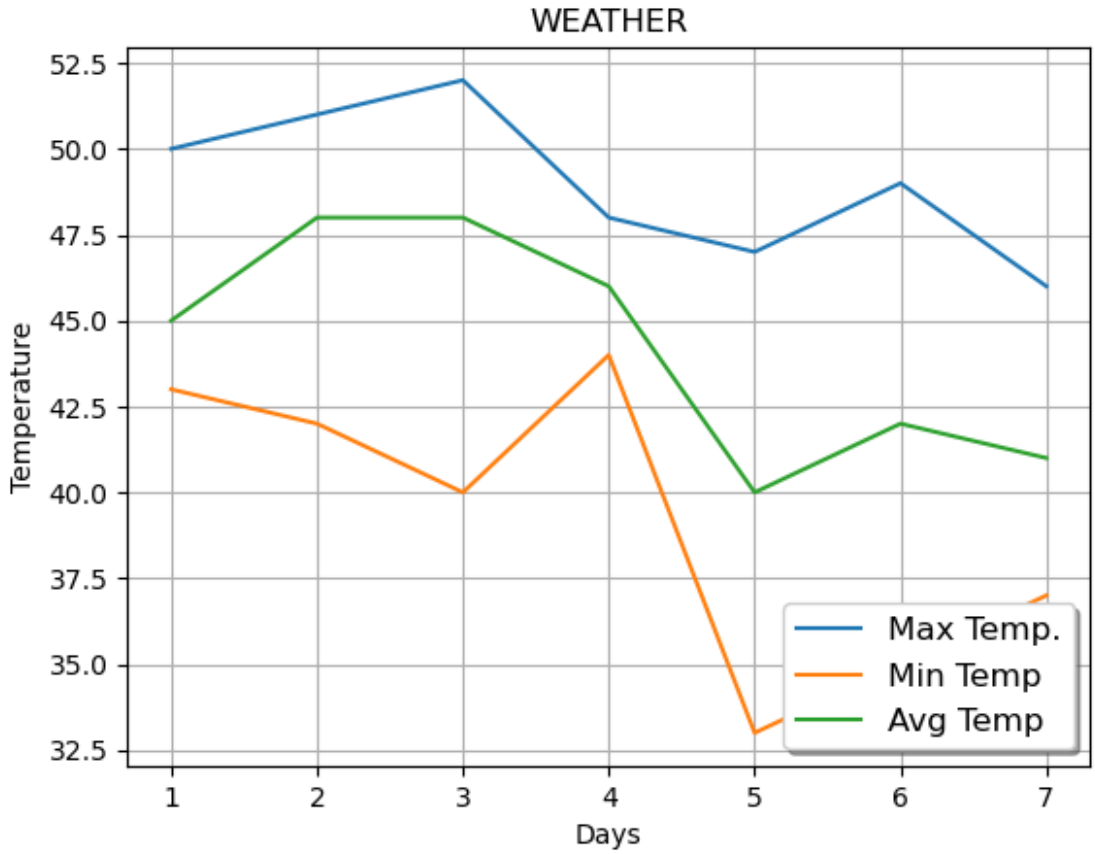
```
plt.xlabel("Days")
plt.ylabel("Temperature")
plt.title("WEATHER")
plt.plot(days,max_t,label="Max Temp.")
plt.plot(days,min_t,label="Min Temp")
plt.plot(days,avg_t,label="Avg Temp")
plt.legend(loc="best",shadow=True)
```

```
<matplotlib.legend.Legend at 0x1d105f1fe80>
```



#we can put loc=0,1,2,3,4,5,6,7,8,9,10 as every digit gives legend a certain location

```
plt.xlabel("Days")
plt.ylabel("Temperature")
plt.title("WEATHER")
plt.plot(days,max_t,label="Max Temp.")
plt.plot(days,min_t,label="Min Temp")
plt.plot(days,avg_t,label="Avg Temp")

plt.legend(loc=4,shadow=True,fontsize="large")
plt.grid()
```


WEATHER
52.5
50.0
47.5
45.0
42.5
40.0
37.5
35.0
32.5
Temperature
1
2
3
4
5
6
7
Days
Max Temp.
Min Temp
Avg Temp

#python programming

Display Calculations

```
print("DJ")
```

```
DJ
```

Display Calculations

```
print(2*3)
print(3*4)
```

```
6
12
```

SKLearn Train Test Split Method

Using full dta set for training is not a good practise ,so now we will used a part of data set for training and rest for testing

```
import pandas as pd
df=pd.read_csv(r"C:\Users\Ayush\OneDrive\Documents\map.csv")
df
```

```
    mileage  age  sellprice
0     69000    6      18000
1     35000    3      34000
2     57000    5      26100
3     22500    2      40000
4     46000    4      31500
5     59000    5      29400
6     52000    5      32000
7     72000    6      19300
8     91000    8      12000
9     67000    6      22000
10    83000    7      20000
11    79000    7      21000
12    59000    5      33000
```

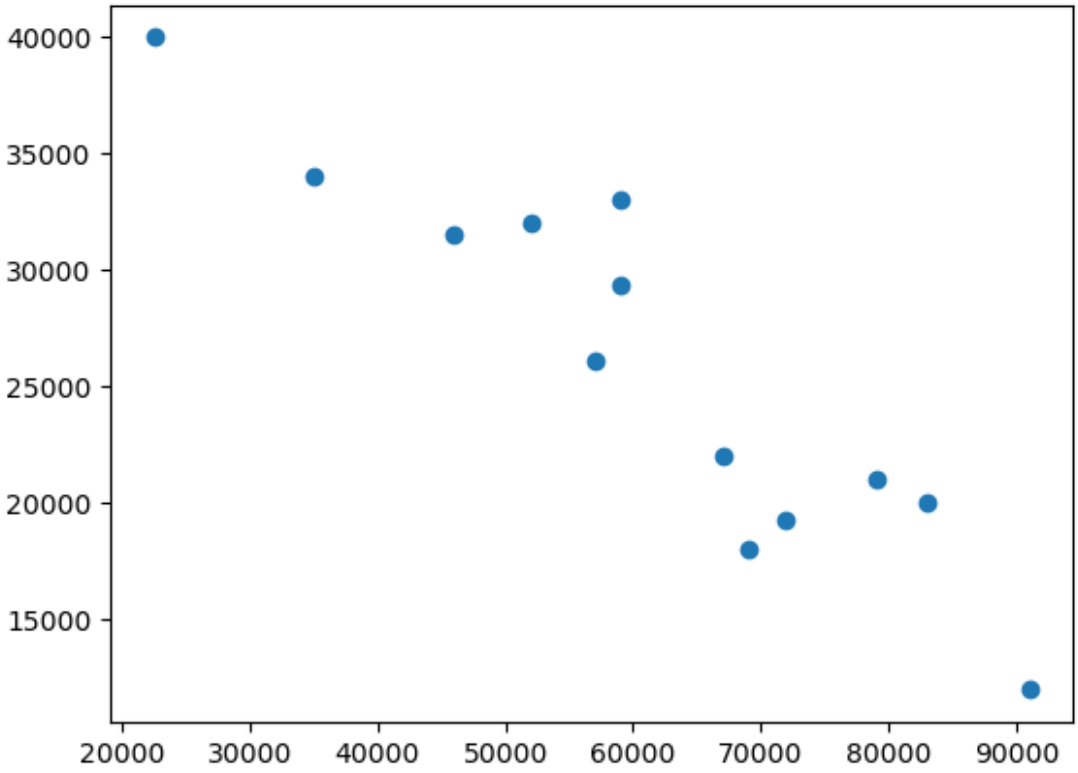
```
df.head()
```

```
   mileage  age  sellprice
0    69000    6      18000
1    35000    3      34000
2    57000    5      26100
3    22500    2      40000
4    46000    4      31500
```

```
import matplotlib.pyplot as plt
%matplotlib inline
```

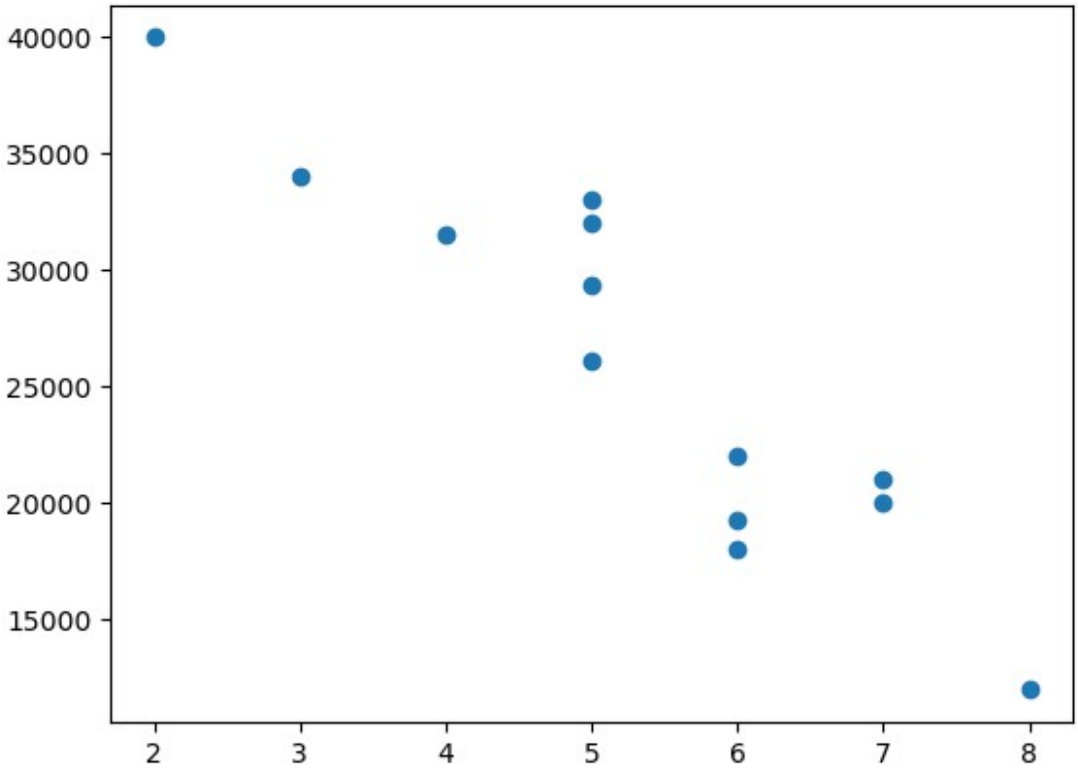
```
plt.scatter(df['mileage'],df['sellprice'])
```

```
<matplotlib.collections.PathCollection at 0x225def2ab30>
```




```
plt.scatter(df['age'],df['sellprice'])
```

```
<matplotlib.collections.PathCollection at 0x225df0da2c0>
```




```
x=df[['mileage','age']]
y=df['sellprice']
```

```
x
```

	mileage	age
0	69000	6
1	35000	3
2	57000	5
3	22500	2
4	46000	4
5	59000	5
6	52000	5
7	72000	6
8	91000	8
9	67000	6
10	83000	7
11	79000	7
12	59000	5

```
y
```

```
0     18000
1     34000
2     26100
3     40000
```

```
4     31500
5     29400
6     32000
7     19300
8     12000
9     22000
10    20000
11    21000
12    33000
Name: sellprice, dtype: int64
```

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

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
```

```
len(x_train)
```

```
10
```

this lenght is 80% of length of total data set

```
len(x_test)
```

```
3
```

```
x_train
```

```
    mileage  age
5     59000    5
2     57000    5
4     46000    4
10    83000    7
1     35000    3
6     52000    5
12    59000    5
8     91000    8
7     72000    6
9     67000    6
```

it selected 80% of dta set randomly as training model

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=10) for Random state value of 10 it will not change X_train set in different runs

```
from sklearn.linear_model import LinearRegression
clf=LinearRegression()
```

```
clf.fit(x_train,y_train)
```

```
LinearRegression()
```

```
clf.predict(x_test)
```

```
array([18979.82697601, 23148.56468738, 41847.12937475])
```

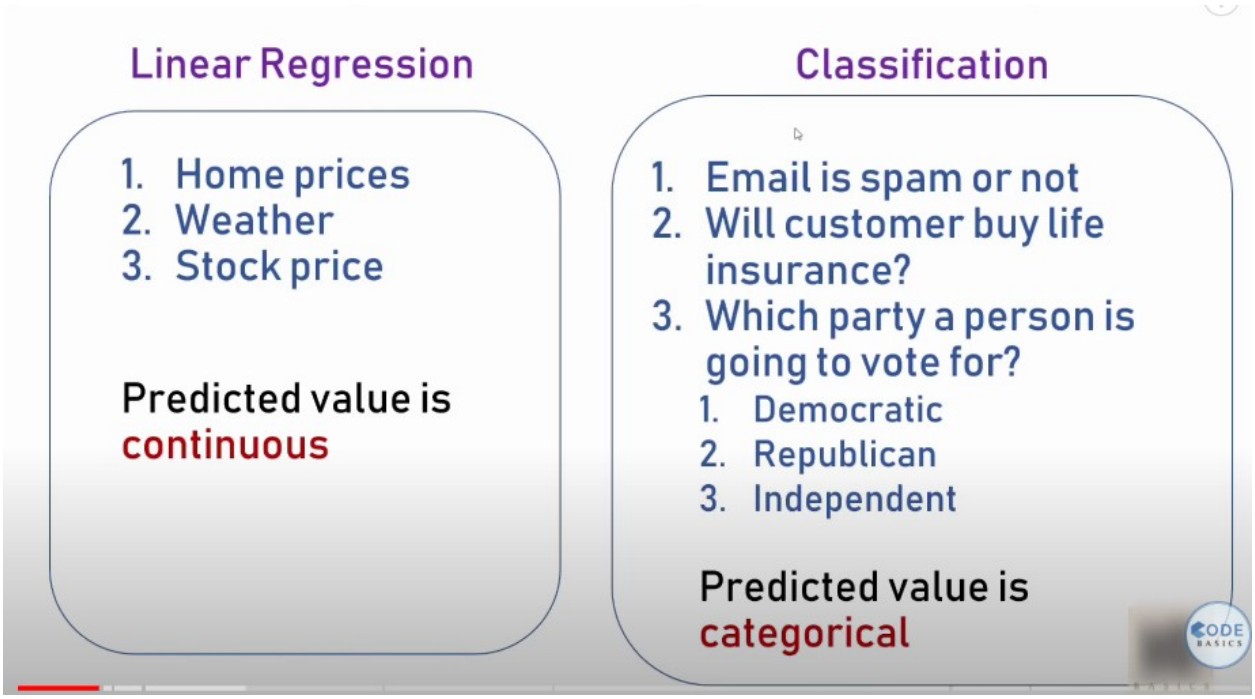
```
y_test
```

```
11    21000
0     18000
3     40000
Name: sellprice, dtype: int64
```

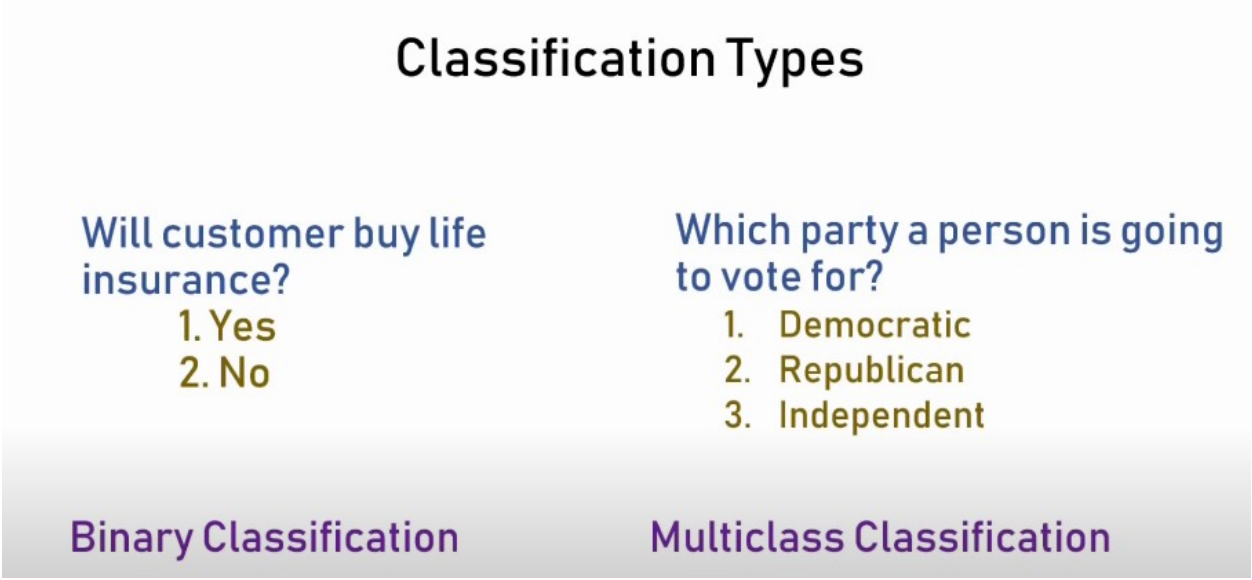
```
clf.score(x_test,y_test)
```

```
0.8805595867188535
```

Logistic Regression

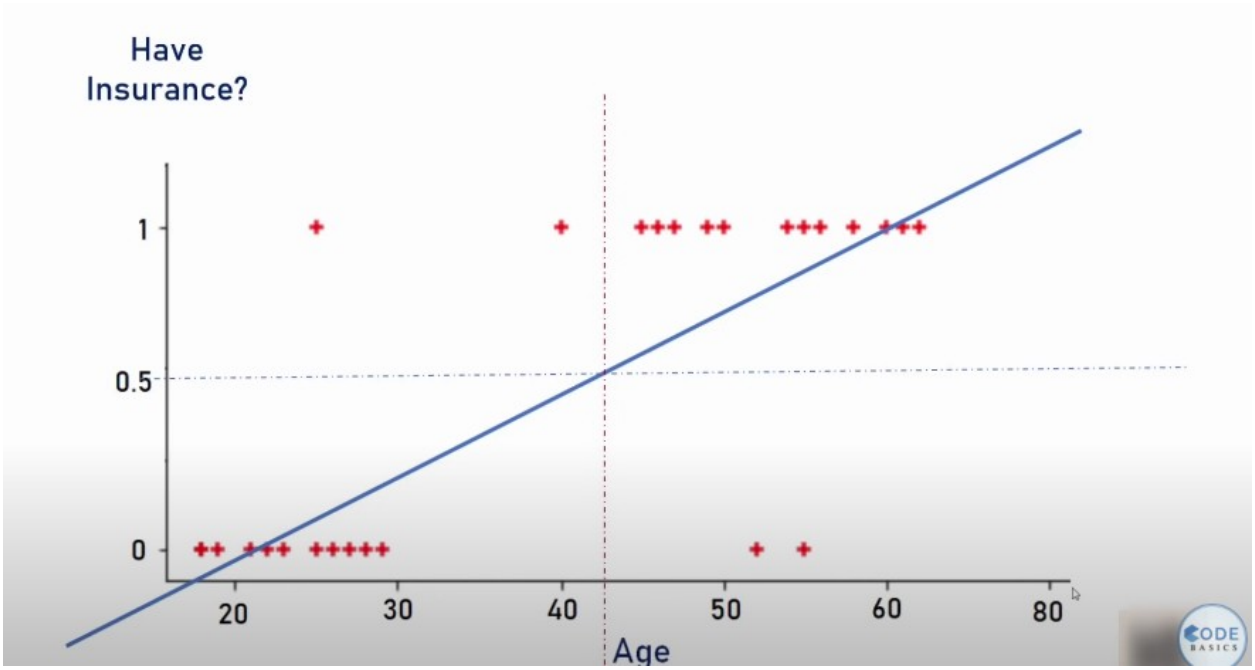


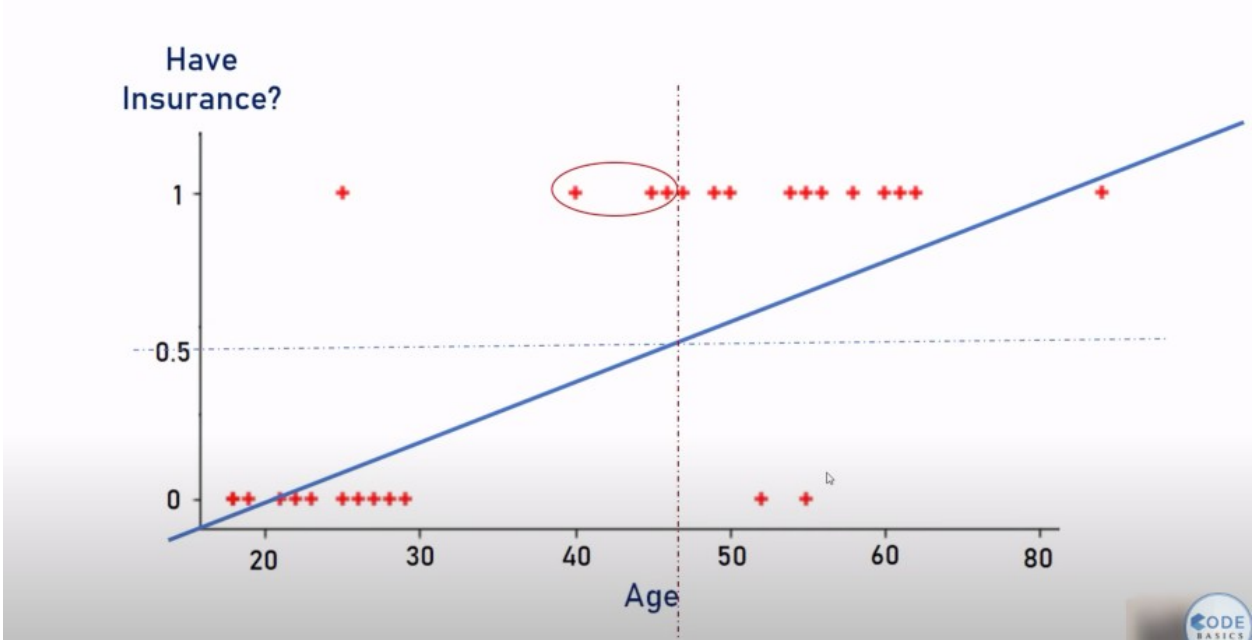
Logistic Regression is a technique that is used to solve these classification problems

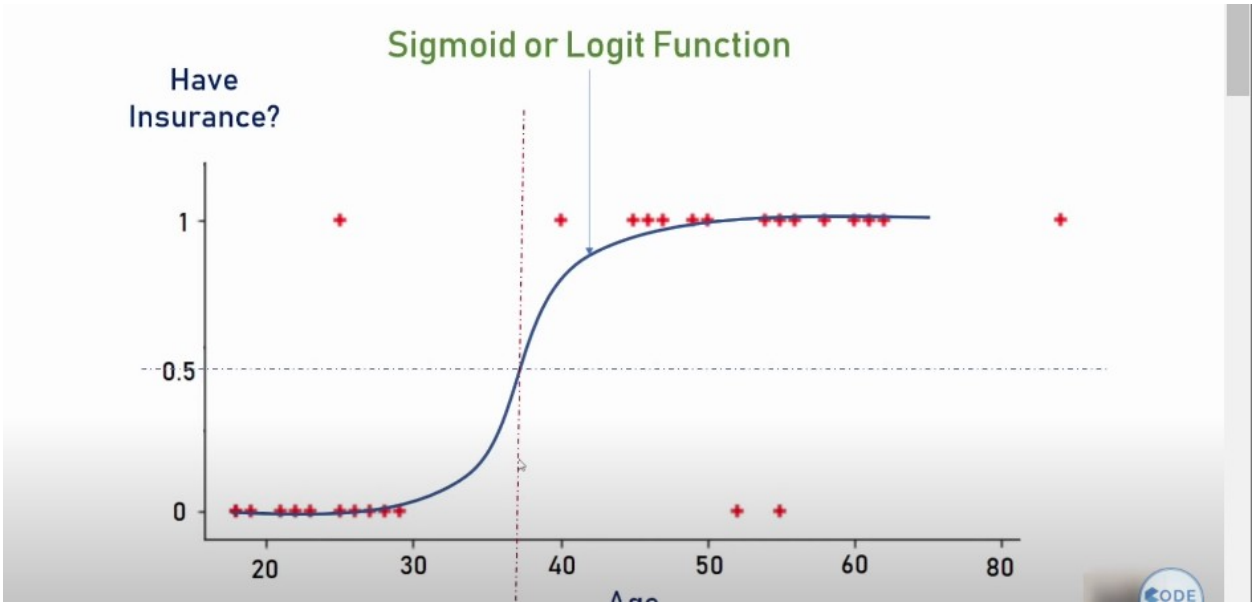


#

Binary classification is when you have more than one categories and when you have two categories then its binary classificatio




Have
Insurance?
1
0.5
0
20
30
40
50
60
80
Age


Sigmoid or Logit Function
Have
Insurance?
1
0.5
0
20
30
40
50
60
80

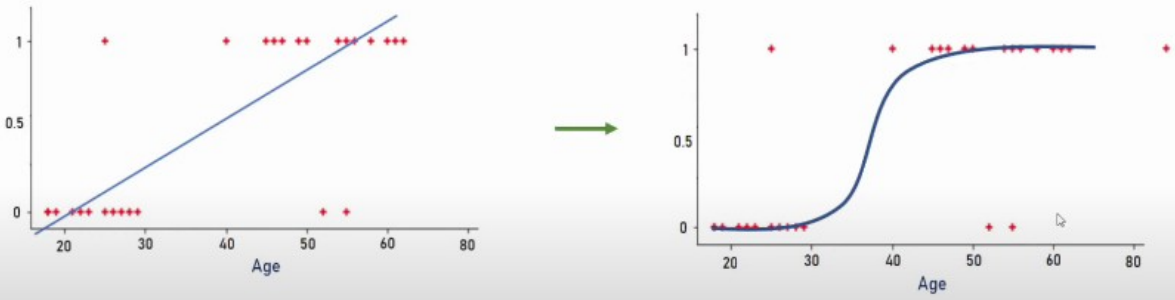
$$sigmoid(z) = \frac{1}{1 + e^{-z}}$$

e = Euler's number ~ 2.71828

Sigmoid function converts input into range 0 to 1

$$y = m * x + b$$

$$y = \frac{1}{1 + e^{-(m*x+b)}}$$



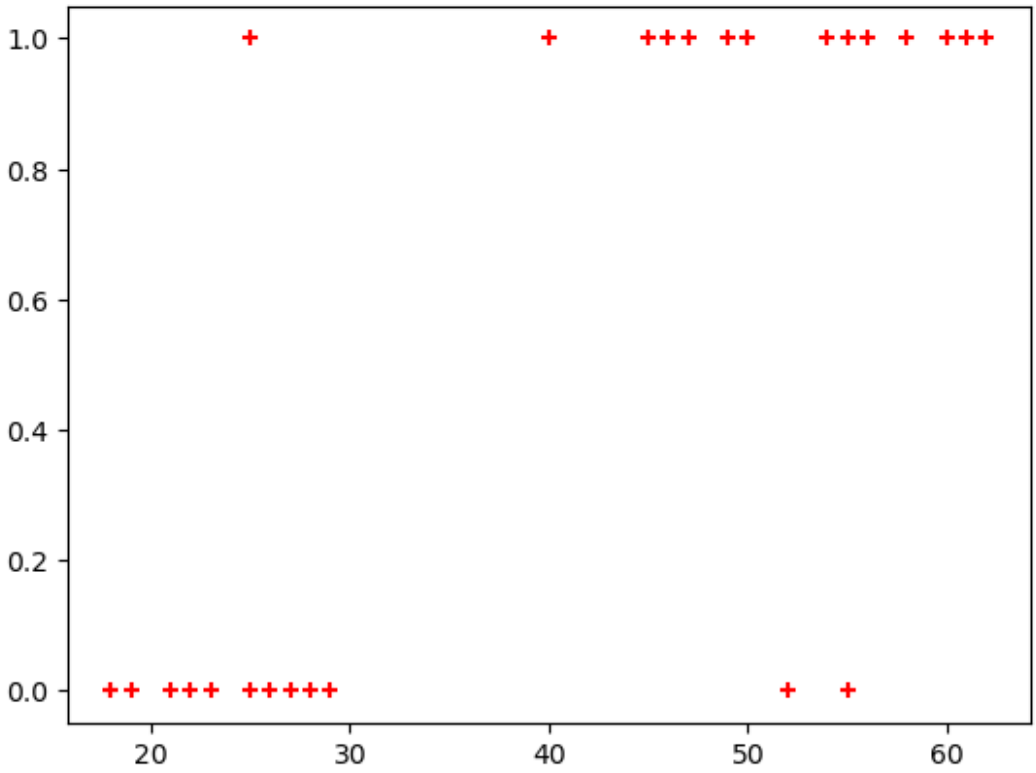

```
import pandas as pd
from matplotlib import pyplot as plt
%matplotlib inline

df=pd.read_csv(r"C:\Users\Ayush\OneDrive\Documents\insurance.csv")
df.head()
```

	age	insurance
0	22	0

```
1   25           0
2   47           1
3   52           0
4   46           1
```

```
plt.scatter(df.age,df.insurance,marker='+',color='red')
```

```
<matplotlib.collections.PathCollection at 0x225e2b46f50>
```




```
df.shape
```

```
(27, 2)
```

```
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test =
train_test_split(df[['age']],df.insurance,train_size=0.9)
```

```
x_test
```

```
    age
15   49
22   26
3    52
```

```
x_train
```

```
    age
4    46
25   50
7    60
17   25
8    23
26   54
16   55
0    22
19   19
5    56
9    62
24   45
12   28
6    55
21   21
10   61
1    25
2    47
20   18
13   27
23   40
11   18
18   58
14   29
```

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

```
model=LogisticRegression()
```

```
model.fit(x_train,y_train)
```

```
LogisticRegression()
```

```
model.predict(x_test)
```

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

```
model.predict([[38]])
```

```
C:\ProgramData\anaconda3\lib\site-packages\sklearn\base.py:420:
UserWarning: X does not have valid feature names, but
LogisticRegression was fitted with feature names
  warnings.warn(
```

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

```
y_test
```

```
15    1
22    0
```

```
3     0
Name: insurance, dtype: int64
```

```
model.score(x_test,y_test)
```

```
0.6666666666666666
```

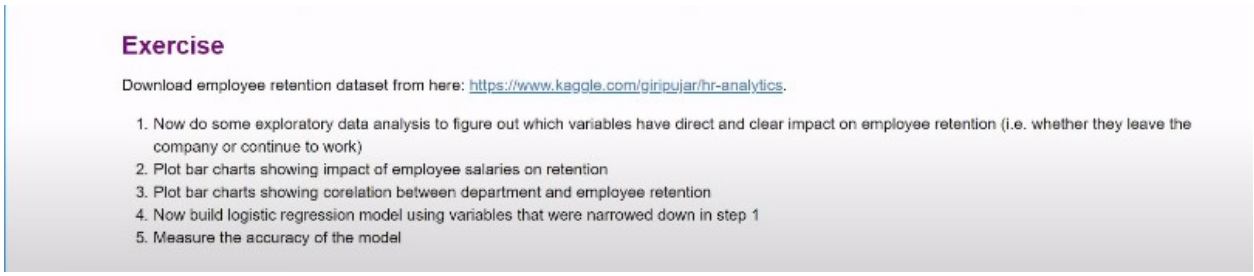
```
model.predict_proba(x_test)
```

```
array([[0.12664319, 0.87335681],
       [0.8379084 , 0.1620916 ],
       [0.08339351, 0.91660649]])
```

```
model.predict([[25]])
```

```
C:\ProgramData\anaconda3\lib\site-packages\sklearn\base.py:420:
UserWarning: X does not have valid feature names, but
LogisticRegression was fitted with feature names
  warnings.warn(
```

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




```
import matplotlib.pyplot as plt
%matplotlib inline
import pandas as pd
```

```
df=pd.read_csv(r"C:\Users\Ayush\OneDrive\Documents\hranalysis.csv")
df
```

```
       satisfaction_level  last_evaluation  number_project  \
0                    0.38             0.53               2
1                    0.80             0.86               5
2                    0.11             0.88               7
3                    0.72             0.87               5
4                    0.37             0.52               2
...                   ...              ...             ...
14994                0.40             0.57               2
14995                0.37             0.48               2
14996                0.37             0.53               2
14997                0.11             0.96               6
14998                0.37             0.52               2
```

```
       average_montly_hours  time_spend_company  Work_accident
left  \
0                       157                   3              0     1

1                       262                   6              0     1

2                       272                   4              0     1

3                       223                   5              0     1

4                       159                   3              0     1

...                     ...                 ...            ...   ...

14994                   151                   3              0     1

14995                   160                   3              0     1

14996                   143                   3              0     1

14997                   280                   4              0     1

14998                   158                   3              0     1


       promotion_last_5years Department  salary
0                          0      sales     low
1                          0      sales  medium
2                          0      sales  medium
3                          0      sales     low
4                          0      sales     low
...                      ...        ...     ...
14994                      0    support     low
14995                      0    support     low
14996                      0    support     low
14997                      0    support     low
14998                      0    support     low

[14999 rows x 10 columns]

dummies=pd.get_dummies(df.Department)
dummies

       IT  RandD  accounting  hr  management  marketing  product_mng
sales  \
0       0      0           0   0           0          0            0
1
1       0      0           0   0           0          0            0
1
2       0      0           0   0           0          0            0
1
```

```
3       0      0           0   0           0          0             0
1
4       0      0           0   0           0          0             0
1
...    ..    ...         ...  ..         ...        ...           ...
...
14994   0      0           0   0           0          0             0
0
14995   0      0           0   0           0          0             0
0
14996   0      0           0   0           0          0             0
0
14997   0      0           0   0           0          0             0
0
14998   0      0           0   0           0          0             0
0
```

```
       support  technical
0            0          0
1            0          0
2            0          0
3            0          0
4            0          0
...        ...        ...
14994        1          0
14995        1          0
14996        1          0
14997        1          0
14998        1          0

[14999 rows x 10 columns]
```

```
dummies1=pd.get_dummies(df.salary)
dummies1
```

```
       high  low  medium
0         0    1       0
1         0    0       1
2         0    0       1
3         0    1       0
4         0    1       0
...     ...  ...     ...
14994     0    1       0
14995     0    1       0
14996     0    1       0
14997     0    1       0
14998     0    1       0

[14999 rows x 3 columns]
```

```
merged=pd.concat([df,dummies,dummies1],axis='columns')
merged
```

```
       satisfaction_level  last_evaluation  number_project  \
0                    0.38             0.53               2
1                    0.80             0.86               5
2                    0.11             0.88               7
3                    0.72             0.87               5
4                    0.37             0.52               2
...                   ...              ...             ...
14994                0.40             0.57               2
14995                0.37             0.48               2
14996                0.37             0.53               2
14997                0.11             0.96               6
14998                0.37             0.52               2

       average_montly_hours  time_spend_company  Work_accident
left  \
0                       157                   3              0     1

1                       262                   6              0     1

2                       272                   4              0     1

3                       223                   5              0     1

4                       159                   3              0     1

...                     ...                 ...            ...   ...

14994                   151                   3              0     1

14995                   160                   3              0     1

14996                   143                   3              0     1

14997                   280                   4              0     1

14998                   158                   3              0     1


       promotion_last_5years Department  salary  ...  hr
management  \
0                          0      sales     low  ...   0
0

1                          0      sales  medium  ...   0
0

2                          0      sales  medium  ...   0
0

3                          0      sales     low  ...   0
0
```

```
4                          0      sales     low  ...   0           0

...                      ...        ...     ...  ...  ..         ...

14994                      0    support     low  ...   0           0

14995                      0    support     low  ...   0           0

14996                      0    support     low  ...   0           0

14997                      0    support     low  ...   0           0

14998                      0    support     low  ...   0           0


       marketing  product_mng  sales  support  technical  high  low
medium
0              0            0      1        0          0     0    1
0
1              0            0      1        0          0     0    0
1
2              0            0      1        0          0     0    0
1
3              0            0      1        0          0     0    1
0
4              0            0      1        0          0     0    1
0
...          ...          ...    ...      ...        ...   ...  ...
...
14994          0            0      0        1          0     0    1
0
14995          0            0      0        1          0     0    1
0
14996          0            0      0        1          0     0    1
0
14997          0            0      0        1          0     0    1
0
14998          0            0      0        1          0     0    1
0

[14999 rows x 23 columns]

final=merged.drop(['Department','salary'],axis='columns')
final

       satisfaction_level  last_evaluation  number_project  \
0                    0.38             0.53               2
1                    0.80             0.86               5
2                    0.11             0.88               7
3                    0.72             0.87               5
4                    0.37             0.52               2
```

```
...                    ...                ...               ...
14994                 0.40               0.57                 2
14995                 0.37               0.48                 2
14996                 0.37               0.53                 2
14997                 0.11               0.96                 6
14998                 0.37               0.52                 2

       average_montly_hours  time_spend_company  Work_accident
left  \
0                       157                   3              0     1

1                       262                   6              0     1

2                       272                   4              0     1

3                       223                   5              0     1

4                       159                   3              0     1

...                     ...                 ...            ...   ...

14994                   151                   3              0     1

14995                   160                   3              0     1

14996                   143                   3              0     1

14997                   280                   4              0     1

14998                   158                   3              0     1


       promotion_last_5years  IT  RandD  ...  hr  management
marketing  \
0                          0   0      0  ...   0           0
0
1                          0   0      0  ...   0           0
0
2                          0   0      0  ...   0           0
0
3                          0   0      0  ...   0           0
0
4                          0   0      0  ...   0           0
0
...                      ...  ..    ...  ...  ..         ...        ..
.
14994                      0   0      0  ...   0           0
0
14995                      0   0      0  ...   0           0
0
```

```
14996                     0   0      0  ...   0           0
0
14997                     0   0      0  ...   0           0
0
14998                     0   0      0  ...   0           0
0

       product_mng  sales  support  technical  high  low  medium
0                0      1        0          0     0    1       0
1                0      1        0          0     0    0       1
2                0      1        0          0     0    0       1
3                0      1        0          0     0    1       0
4                0      1        0          0     0    1       0
...            ...    ...      ...        ...   ...  ...     ...
14994            0      0        1          0     0    1       0
14995            0      0        1          0     0    1       0
14996            0      0        1          0     0    1       0
14997            0      0        1          0     0    1       0
14998            0      0        1          0     0    1       0

[14999 rows x 21 columns]
```

```
x=final.drop(['low','sales'],axis='columns')
x
```

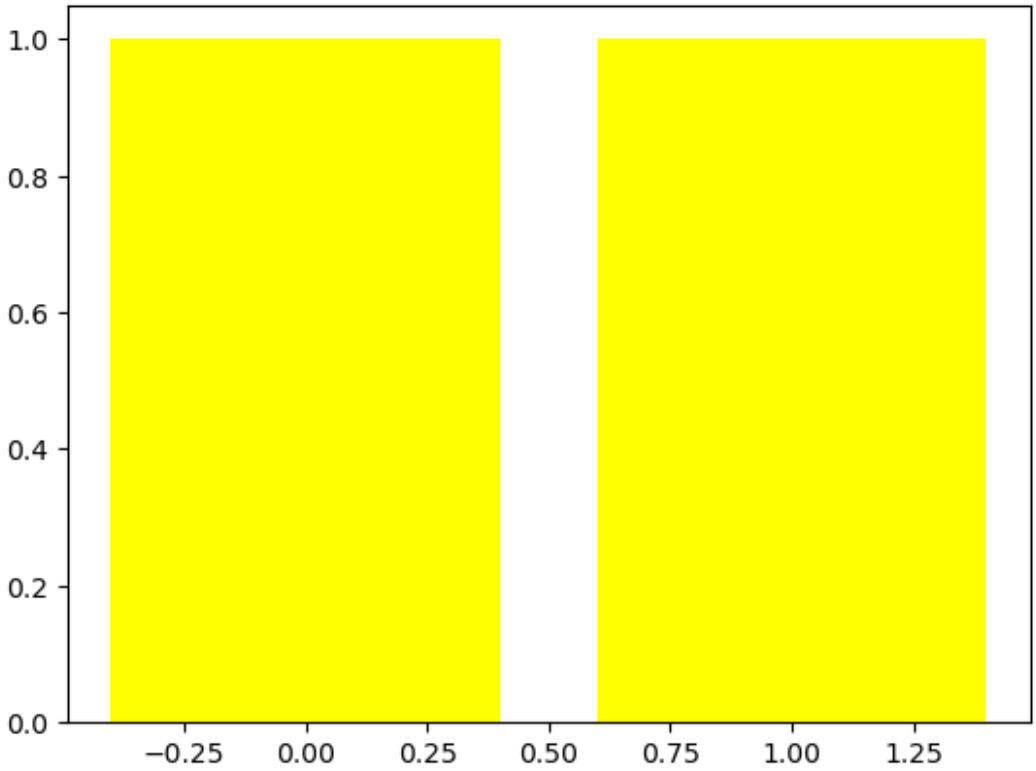
```
       satisfaction_level  last_evaluation  number_project  \
0                    0.38             0.53               2
1                    0.80             0.86               5
2                    0.11             0.88               7
3                    0.72             0.87               5
4                    0.37             0.52               2
...                   ...              ...             ...
14994                0.40             0.57               2
14995                0.37             0.48               2
14996                0.37             0.53               2
14997                0.11             0.96               6
14998                0.37             0.52               2

       average_montly_hours  time_spend_company  Work_accident
left  \
0                       157                   3              0     1

1                       262                   6              0     1

2                       272                   4              0     1

3                       223                   5              0     1

4                       159                   3              0     1
```

```
...                     ...                  ...            ...   ...
14994                   151                    3              0     1
14995                   160                    3              0     1
14996                   143                    3              0     1
14997                   280                    4              0     1
14998                   158                    3              0     1
```

```
       promotion_last_5years  IT  RandD  accounting  hr  management  \
0                          0   0      0           0   0           0
1                          0   0      0           0   0           0
2                          0   0      0           0   0           0
3                          0   0      0           0   0           0
4                          0   0      0           0   0           0
...                      ...  ..    ...         ...  ..         ...
14994                      0   0      0           0   0           0
14995                      0   0      0           0   0           0
14996                      0   0      0           0   0           0
14997                      0   0      0           0   0           0
14998                      0   0      0           0   0           0

       marketing  product_mng  support  technical  high  medium
0              0            0        0          0     0       0
1              0            0        0          0     0       1
2              0            0        0          0     0       1
3              0            0        0          0     0       0
4              0            0        0          0     0       0
...          ...          ...      ...        ...   ...     ...
14994          0            0        1          0     0       0
14995          0            0        1          0     0       0
14996          0            0        1          0     0       0
14997          0            0        1          0     0       0
14998          0            0        1          0     0       0

[14999 rows x 19 columns]
```

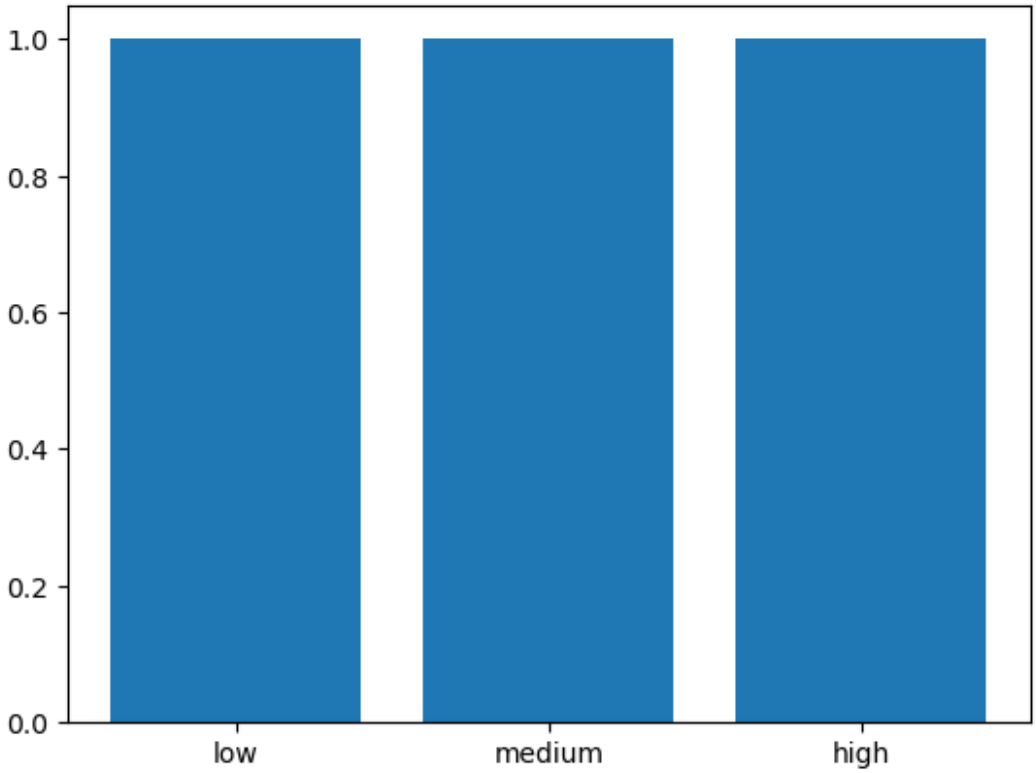
```
import matplotlib.pyplot as plt
%matplotlib inline

plt.bar(final.left,final.low,color='red',label='low')
plt.bar(final.left,final.high,color='blue',label='high')
plt.bar(final.left,final.medium,color='yellow',label='medium')
```

```
<BarContainer object of 14999 artists>
```



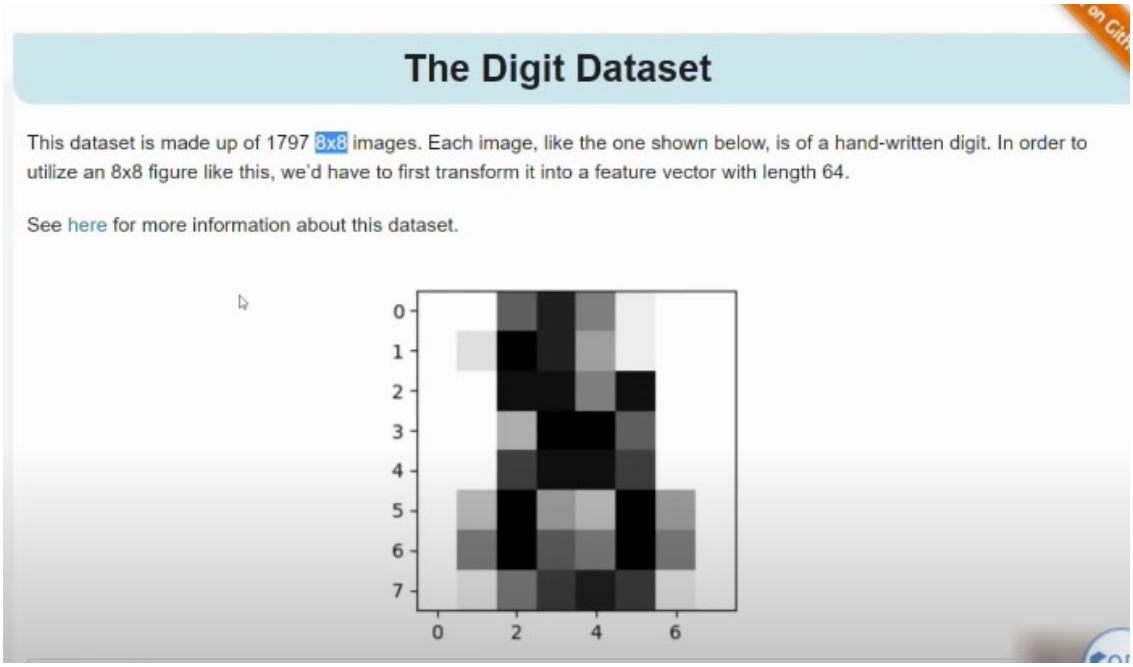

```
plt.bar(df.salary,df.left)
```

```
<BarContainer object of 14999 artists>
```



Multiclass Classification

```
%matplotlib inline
import matplotlib.pyplot as plt
from sklearn.datasets import load_digits
```




```
digits=load_digits()
```

```
dir(digits)
```

```
['DESCR', 'data', 'feature_names', 'frame', 'images', 'target',
'target_names']
```

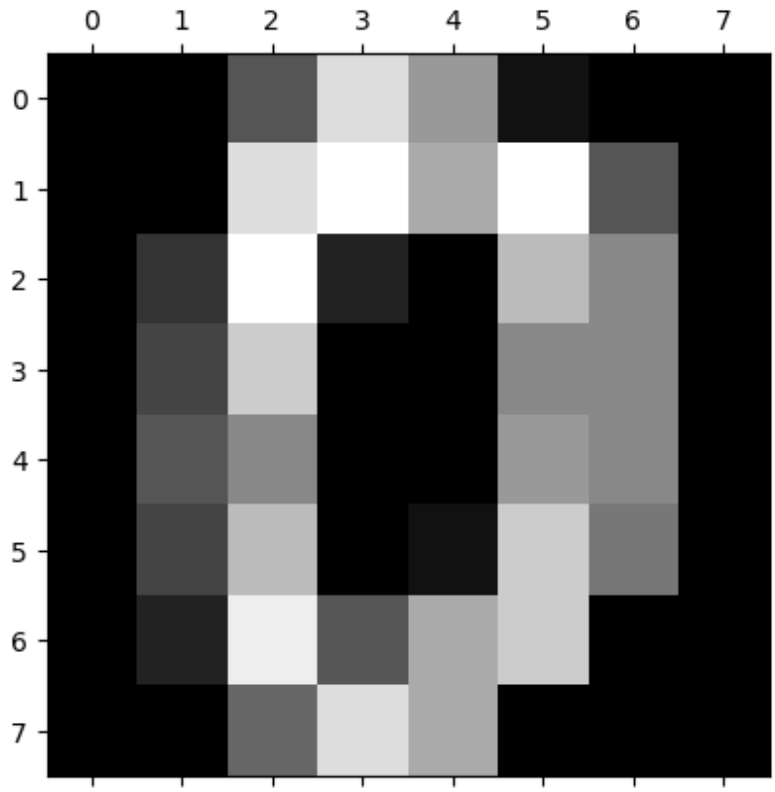
```
digits.data[0]
```

```
array([ 0.,  0.,  5., 13.,  9.,  1.,  0.,  0.,  0.,  0., 13., 15.,
10.,
       15.,  5.,  0.,  0.,  3., 15.,  2.,  0., 11.,  8.,  0.,  0.,
4.,
       12.,  0.,  0.,  8.,  8.,  0.,  0.,  5.,  8.,  0.,  0.,  9.,
8.,
        0.,  0.,  4., 11.,  0.,  1., 12.,  7.,  0.,  0.,  2., 14.,
5.,
       10., 12.,  0.,  0.,  0.,  0.,  6., 13., 10.,  0.,  0.,  0.])
```

```
plt.gray()
plt.matshow(digits.images[0])
```

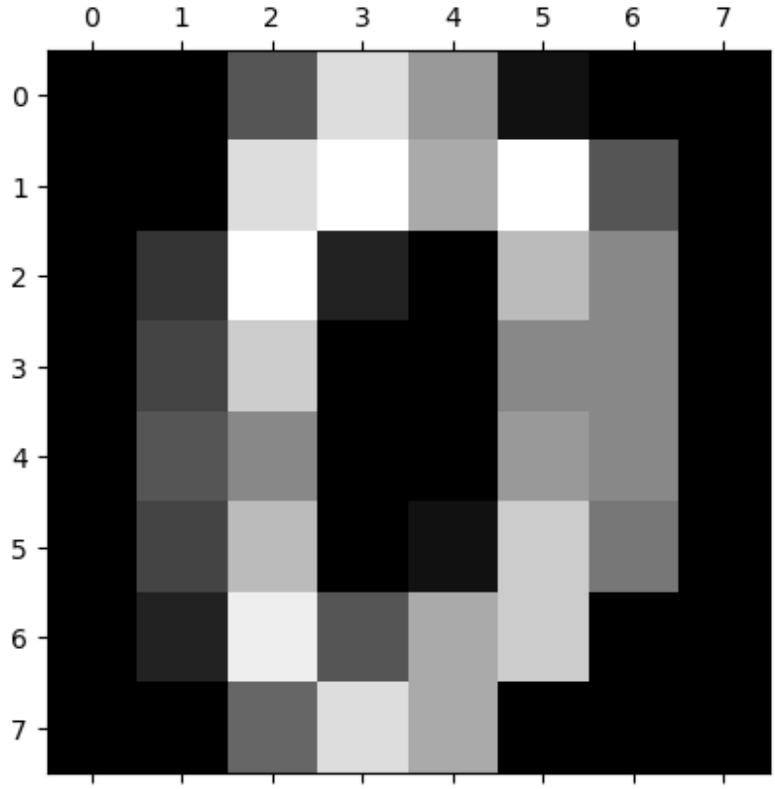
```
<matplotlib.image.AxesImage at 0x22638919120>
```

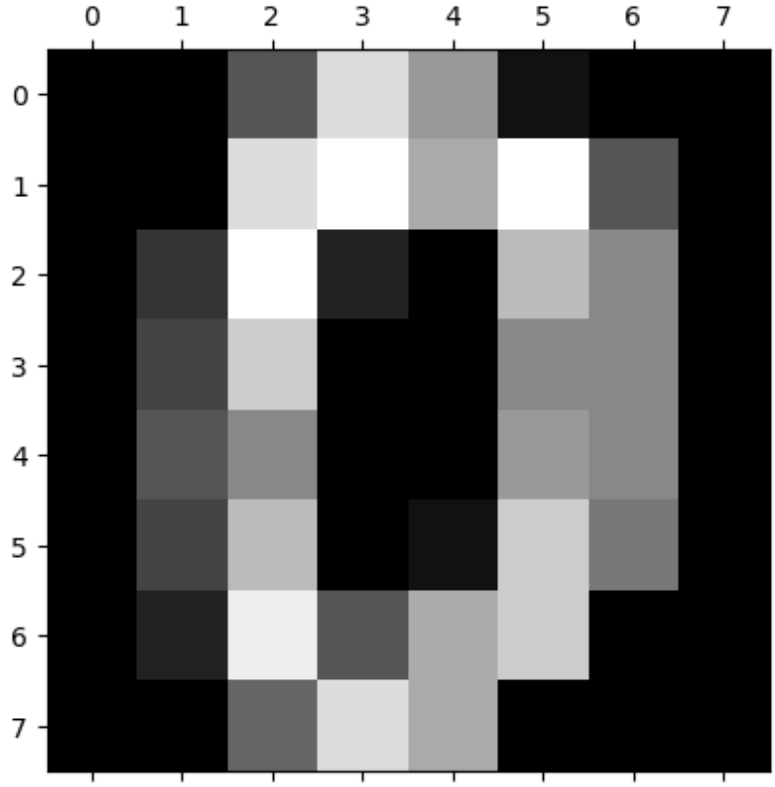
```
<Figure size 640x480 with 0 Axes>
```

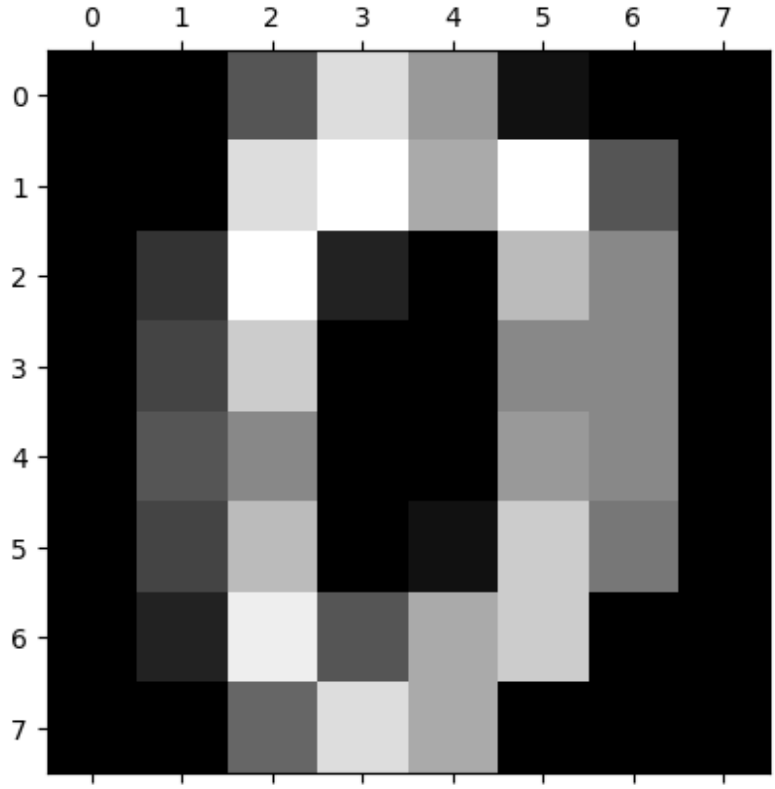


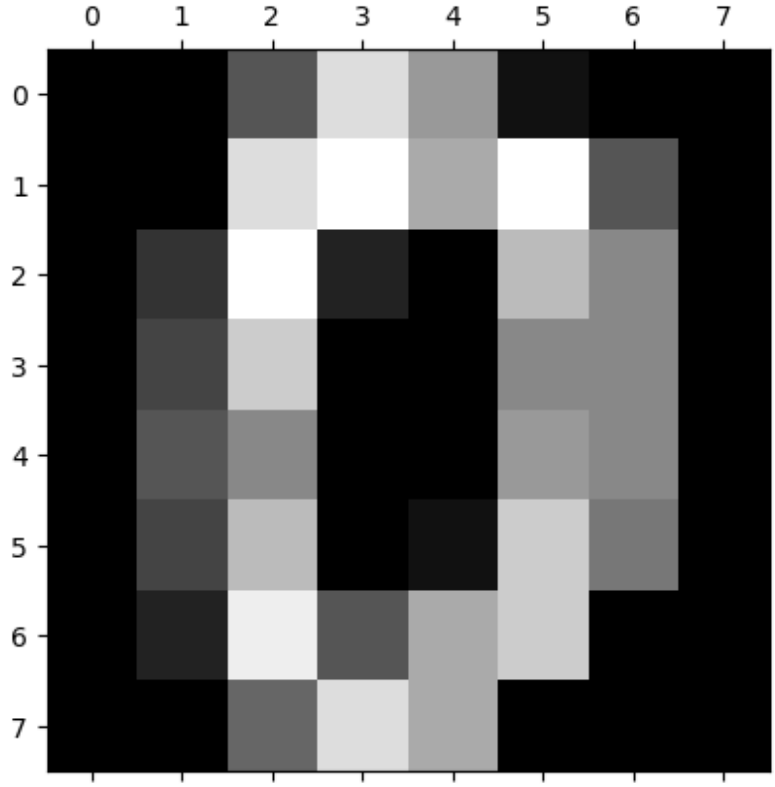

```
plt.gray()
for i in range(5):
   plt.matshow(digits.images[0])
```

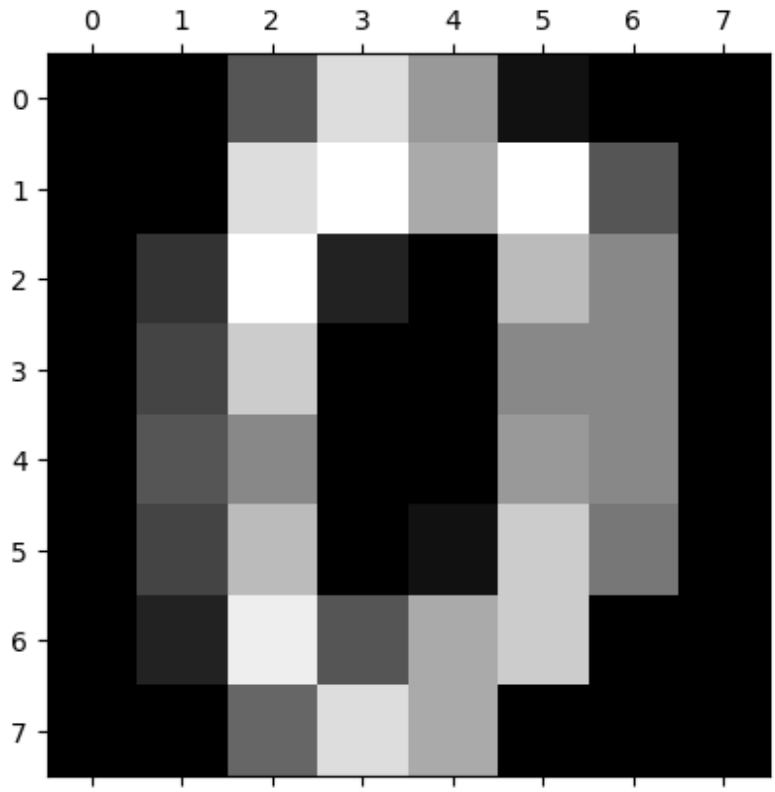
```
<Figure size 640x480 with 0 Axes>
```


0 1 2 3 4 5 6 7
0 1 2 3 4 5 6 7


0 1 2 3 4 5 6 7
0 1 2 3 4 5 6 7


0 1 2 3 4 5 6 7
0 1 2 3 4 5 6 7


0 1 2 3 4 5 6 7
0 1 2 3 4 5 6 7




```
digits.target[0:5]
```

```
array([0, 1, 2, 3, 4])
```

```
from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test=train_test_split(digits.data,digits.targ
et,test_size=0.2)
```

```
len(x)
```

```
14999
```

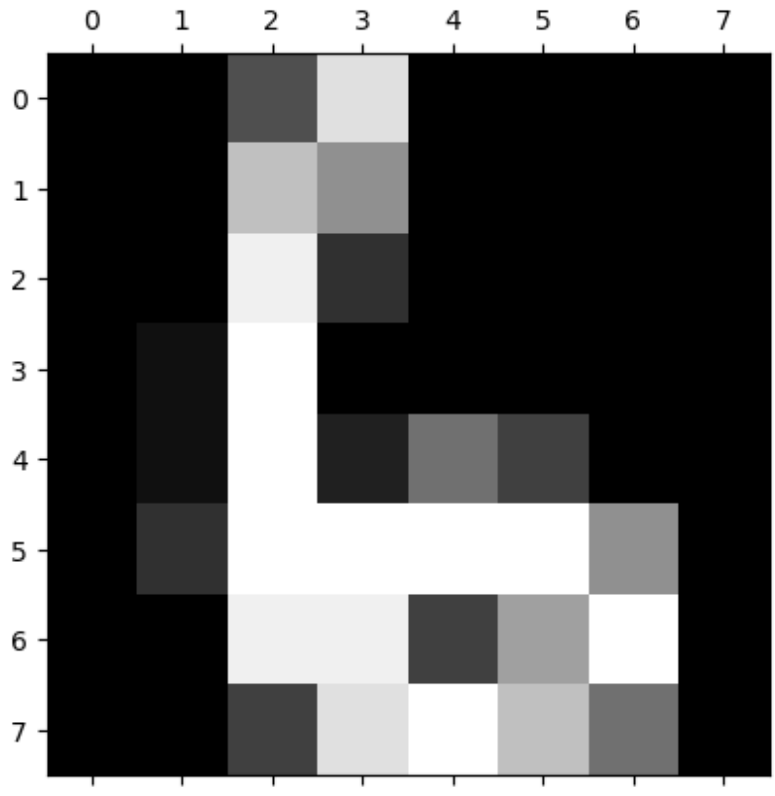
```
len(x_test)
```

```
450
```

```
from sklearn.linear_model import LogisticRegression
model=LogisticRegression()

model.fit(x_train,y_train)
```

```
C:\ProgramData\anaconda3\lib\site-packages\sklearn\linear_model\
_logistic.py:458: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

```
Increase the number of iterations (max_iter) or scale the data as
shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(

LogisticRegression()

model.score(x_test,y_test)

0.9644444444444444

plt.matshow(digits.images[67])

<matplotlib.image.AxesImage at 0x2263a1367a0>
```




```
digits.target[67]

6

model.predict([digits.data[67]])

array([6])
```

```
model.predict(digits.data[0:5])
```

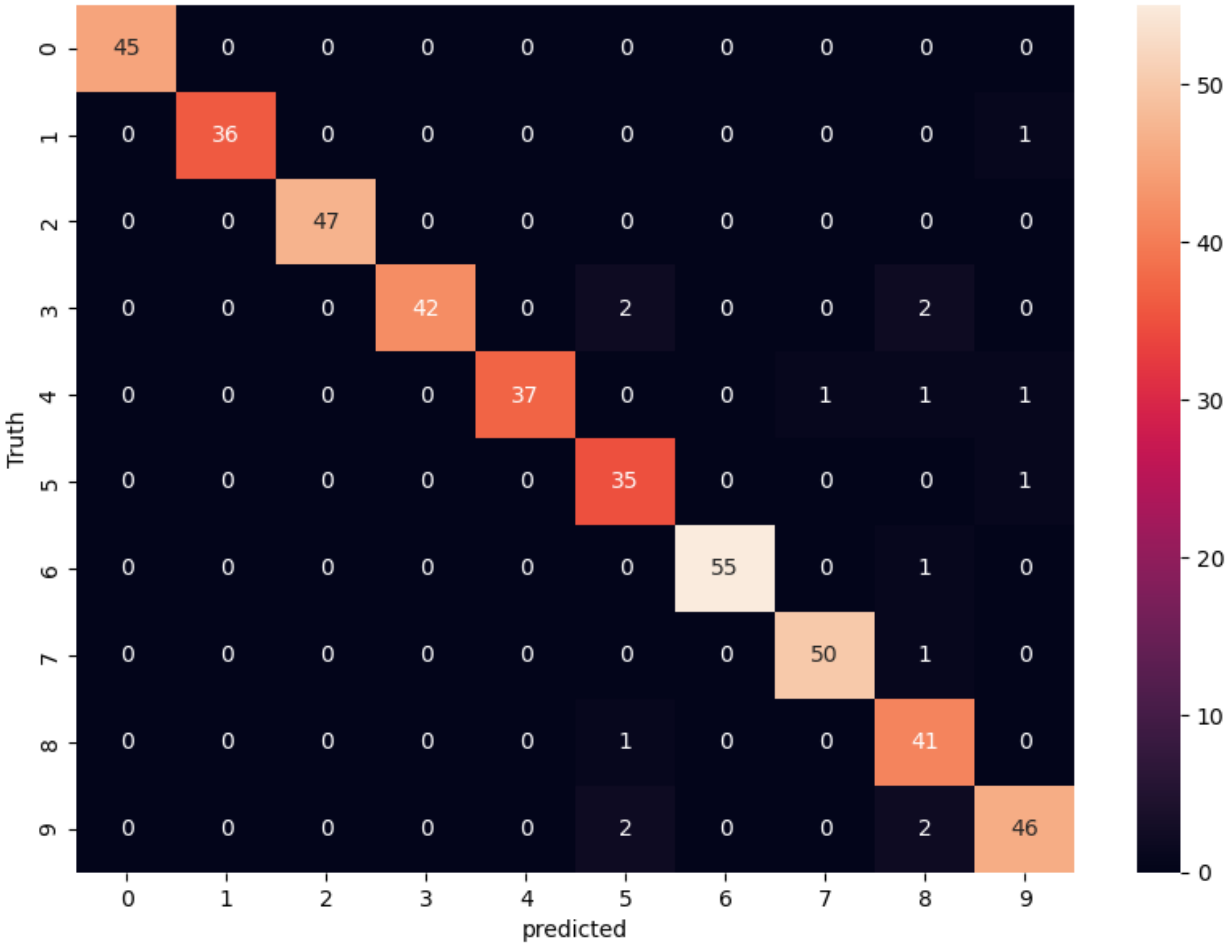
```
array([0, 1, 2, 3, 4])
```

```
y_predicted=model.predict(x_test)
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,y_predicted)
cm
```

```
array([[45,  0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0, 36,  0,  0,  0,  0,  0,  0,  0,  1],
       [ 0,  0, 47,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0, 42,  0,  2,  0,  0,  2,  0],
       [ 0,  0,  0,  0, 37,  0,  0,  1,  1,  1],
       [ 0,  0,  0,  0,  0, 35,  0,  0,  0,  1],
       [ 0,  0,  0,  0,  0,  0, 55,  0,  1,  0],
       [ 0,  0,  0,  0,  0,  0,  0, 50,  1,  0],
       [ 0,  0,  0,  0,  0,  1,  0,  0, 41,  0],
       [ 0,  0,  0,  0,  0,  2,  0,  0,  2, 46]], dtype=int64)
```

```
import seaborn as sn
plt.figure(figsize=(10,7))
sn.heatmap(cm,annot=True)
plt.xlabel('predicted')
plt.ylabel('Truth')
```

```
Text(95.72222222222221, 0.5, 'Truth')
```



Use sklearn.datasets iris flower dataset to train your model using logistic regression. You need to figure out accuracy of your model and use that to predict different samples in your test dataset. In iris dataset there are 150 samples containing following features,

1. Sepal Length
2. Sepal Width
3. Petal Length
4. Petal Width

Using above 4 features you will clasify a flower in one of the three categories,

1. Setosa
2. Versicolour
3. Virginica

```
!pip install pandas
```

```
Requirement already satisfied: pandas in c:\programdata\anaconda3\lib\
site-packages (1.5.3)
Requirement already satisfied: numpy>=1.21.0 in c:\programdata\
anaconda3\lib\site-packages (from pandas) (1.23.5)
Requirement already satisfied: pytz>=2020.1 in c:\programdata\
anaconda3\lib\site-packages (from pandas) (2022.7)
Requirement already satisfied: python-dateutil>=2.8.1 in c:\
programdata\anaconda3\lib\site-packages (from pandas) (2.8.2)
Requirement already satisfied: six>=1.5 in c:\programdata\anaconda3\
lib\site-packages (from python-dateutil>=2.8.1->pandas) (1.16.0)
```

```
import pandas as pd
```

```
data=
```

DummyVariable and OneHotEncoding

```
import pickle
from sklearn.linear_model import LinearRegression

# Create a simple Linear Regression model
model = LinearRegression()

# Sample data for training
X_train = [[1], [2], [3], [4], [5]]
y_train = [2, 4, 6, 8, 10]

# Train the model
model.fit(X_train, y_train)

# Save the model to a file using pickle
with open('model_pickle', 'wb') as f:
    pickle.dump(model, f)

##the above will create a model named "pickle "in your folder

import pandas as pd
df=pd.read_csv(r"C:\Users\Ayush\OneDrive\Documents\townships.csv")
df
```

	town	area	price
0	monroe township	2600	550000
1	monroe township	3000	565000
2	monroe township	3200	610000
3	monroe township	3600	680000
4	monroe township	4000	725000
5	west windsor	2600	585000
6	west windsor	2800	615000
7	west windsor	3300	650000
8	west windsor	3600	710000
9	robinsville	2600	575000
10	robinsville	2900	600000
11	robinsville	3100	620000
12	robinsville	3600	695000

Now we would create DUMMY VARIABLES

```
dummies=pd.get_dummies(df.town)
dummies
```

	monroe township	robinsville	west windsor
0	1	0	0

```
1                  1             0              0
2                  1             0              0
3                  1             0              0
4                  1             0              0
5                  0             0              1
6                  0             0              1
7                  0             0              1
8                  0             0              1
9                  0             1              0
10                 0             1              0
11                 0             1              0
12                 0             1              0
```

since our category column is "town,now we will concatenate this data frame with original data frame

```
merged=pd.concat([df,dummies],axis='columns')
merged
```

```
                town  area   price  monroe township  robinsville  west
windsor
0   monroe township  2600  550000                1            0
0
1   monroe township  3000  565000                1            0
0
2   monroe township  3200  610000                1            0
0
3   monroe township  3600  680000                1            0
0
4   monroe township  4000  725000                1            0
0
5      west windsor  2600  585000                0            0
1
6      west windsor  2800  615000                0            0
1
7      west windsor  3300  650000                0            0
1
8      west windsor  3600  710000                0            0
1
9       robinsville  2600  575000                0            1
0
10      robinsville  2900  600000                0            1
0
11      robinsville  3100  620000                0            1
0
12      robinsville  3600  695000                0            1
0
```

Now we need to drop original township columns as now we have dummy variables in the name of townships and along with this we need to drop one township dummy variable also inorder to prevent "dummy variable trap".Lets drop west windsor column.

The Dummy Variable Trap **occurs when two or more dummy variables created by one-hot encoding are highly correlated (multi-collinear)**. This means that one variable can be predicted from the others, making it difficult to interpret predicted coefficient variables in regression models.

https://www.learndatasci.com › dum...

Dummy Variable Trap - LearnDataSci

About featured snippets Feedback

People also ask

What is dummy variable trap and how do you avoid it?

To avoid dummy variable trap we should always **add one less (n-1) dummy variable then the total**

```
final=merged.drop(['town','west windsor'],axis='columns')
final
```

	area	price	monroe township	robinsville
0	2600	550000	1	0
1	3000	565000	1	0
2	3200	610000	1	0
3	3600	680000	1	0
4	4000	725000	1	0
5	2600	585000	0	0
6	2800	615000	0	0
7	3300	650000	0	0
8	3600	710000	0	0
9	2600	575000	0	1
10	2900	600000	0	1
11	3100	620000	0	1
12	3600	695000	0	1

when we are using SKLEARN linear regression then our dummy variable(any one)would we dropped automatically as it is aware of dummy variable trap buts its always a good practice to drop it by your own. Now we will create linear regression model object ,

```
from sklearn.linear_model import LinearRegression
model=LinearRegression()

x=final.drop(['price'],axis='columns')
x
```

	area	monroe township	robinsville
0	2600	1	0
1	3000	1	0
2	3200	1	0
3	3600	1	0
4	4000	1	0
5	2600	0	0
6	2800	0	0
7	3300	0	0
8	3600	0	0
9	2600	0	1
10	2900	0	1
11	3100	0	1
12	3600	0	1

```
y=final.price
y
```

```
0     550000
1     565000
2     610000
3     680000
```

```
4     725000
5     585000
6     615000
7     650000
8     710000
9     575000
10    600000
11    620000
12    695000
Name: price, dtype: int64
```

```
model.fit(x,y)
```

```
LinearRegression()
```

```
model.predict([[2800,0,1]])
```

```
C:\ProgramData\anaconda3\lib\site-packages\sklearn\base.py:420:
UserWarning: X does not have valid feature names, but LinearRegression
was fitted with feature names
  warnings.warn(
```

```
array([590775.63964739])
```

```
model.predict([[3400,0,0]])
```

```
C:\ProgramData\anaconda3\lib\site-packages\sklearn\base.py:420:
UserWarning: X does not have valid feature names, but LinearRegression
was fitted with feature names
  warnings.warn(
```

```
array([681241.66845839])
```

```
model.score(x,y)
```

```
0.9573929037221873
```

This tells us that our model is 95.73% accurate

```
df
```

	town	area	price
0	monroe township	2600	550000
1	monroe township	3000	565000
2	monroe township	3200	610000
3	monroe township	3600	680000
4	monroe township	4000	725000
5	west windsor	2600	585000
6	west windsor	2800	615000
7	west windsor	3300	650000
8	west windsor	3600	710000

```
9       robinsville  2600  575000
10      robinsville  2900  600000
11      robinsville  3100  620000
12      robinsville  3600  695000
```

In order to use ONE HOT ENCODER,first we need to do label encoding on the TOWN column

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()

dfle=df
dfle.town=le.fit_transform(dfle.town)
dfle
```

```
    town  area   price
0      0  2600  550000
1      0  3000  565000
2      0  3200  610000
3      0  3600  680000
4      0  4000  725000
5      2  2600  585000
6      2  2800  615000
7      2  3300  650000
8      2  3600  710000
9      1  2600  575000
10     1  2900  600000
11     1  3100  620000
12     1  3600  695000
```

```
x=df[['town','area']].values
x
```

```
array([[   0, 2600],
       [   0, 3000],
       [   0, 3200],
       [   0, 3600],
       [   0, 4000],
       [   2, 2600],
       [   2, 2800],
       [   2, 3300],
       [   2, 3600],
       [   1, 2600],
       [   1, 2900],
       [   1, 3100],
       [   1, 3600]], dtype=int64)
```

```
y=dfle.price
y
```

```
0     550000
1     565000
2     610000
3     680000
4     725000
5     585000
6     615000
7     650000
8     710000
9     575000
10    600000
11    620000
12    695000
Name: price, dtype: int64
```

```
from sklearn.preprocessing import OneHotEncoder
ohe = OneHotEncoder(categorical_features=[0])
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[62], line 2
      1 from sklearn.preprocessing import OneHotEncoder
----> 2 ohe = OneHotEncoder(categorical_features=[0])

TypeError: OneHotEncoder.__init__() got an unexpected keyword argument 'categorical_features'
```

```
ohe.fit_transform(x).toarray()
```

```
array([[1., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0.],
       [1., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0.],
       [1., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0.],
       [1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0.],
       [1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1.],
       [0., 0., 1., 1., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 1., 0., 1., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 1., 0., 0., 0., 0., 0., 0., 1., 0., 0.],
       [0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 1., 0.],
       [0., 1., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 1., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0.],
       [0., 1., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0.],
       [0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0.]])
```

```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
import numpy as np

# Sample data with categorical and numerical features
data = x
```

```
# Specify which columns need to be one-hot encoded (in this example, column 0)
categorical_features = [0]

# Create a ColumnTransformer to apply the OneHotEncoder to the specified columns
preprocessor = ColumnTransformer(
    transformers=[('cat', OneHotEncoder(), categorical_features)],
    remainder='passthrough'  # Keep the remaining columns as they are
)

# Fit and transform the data using the ColumnTransformer
transformed_data = preprocessor.fit_transform(data)

print(transformed_data)
```

```
[[1.0e+00 0.0e+00 0.0e+00 2.6e+03]
 [1.0e+00 0.0e+00 0.0e+00 3.0e+03]
 [1.0e+00 0.0e+00 0.0e+00 3.2e+03]
 [1.0e+00 0.0e+00 0.0e+00 3.6e+03]
 [1.0e+00 0.0e+00 0.0e+00 4.0e+03]
 [0.0e+00 0.0e+00 1.0e+00 2.6e+03]
 [0.0e+00 0.0e+00 1.0e+00 2.8e+03]
 [0.0e+00 0.0e+00 1.0e+00 3.3e+03]
 [0.0e+00 0.0e+00 1.0e+00 3.6e+03]
 [0.0e+00 1.0e+00 0.0e+00 2.6e+03]
 [0.0e+00 1.0e+00 0.0e+00 2.9e+03]
 [0.0e+00 1.0e+00 0.0e+00 3.1e+03]
 [0.0e+00 1.0e+00 0.0e+00 3.6e+03]]
```

```
x=transformed_data
```

in order to avoid dummy variable trap ,drop one dummy variable .The next step eould drop zeroth column and take all rows

```
x
```

```
array([[1.0e+00, 0.0e+00, 0.0e+00, 2.6e+03],
       [1.0e+00, 0.0e+00, 0.0e+00, 3.0e+03],
       [1.0e+00, 0.0e+00, 0.0e+00, 3.2e+03],
       [1.0e+00, 0.0e+00, 0.0e+00, 3.6e+03],
       [1.0e+00, 0.0e+00, 0.0e+00, 4.0e+03],
       [0.0e+00, 0.0e+00, 1.0e+00, 2.6e+03],
       [0.0e+00, 0.0e+00, 1.0e+00, 2.8e+03],
       [0.0e+00, 0.0e+00, 1.0e+00, 3.3e+03],
       [0.0e+00, 0.0e+00, 1.0e+00, 3.6e+03],
       [0.0e+00, 1.0e+00, 0.0e+00, 2.6e+03],
       [0.0e+00, 1.0e+00, 0.0e+00, 2.9e+03],
```

```
       [0.0e+00, 1.0e+00, 0.0e+00, 3.1e+03],
       [0.0e+00, 1.0e+00, 0.0e+00, 3.6e+03]])


x = x[:, 1:]
x
```

```
array([[0.0e+00, 0.0e+00, 2.6e+03],
       [0.0e+00, 0.0e+00, 3.0e+03],
       [0.0e+00, 0.0e+00, 3.2e+03],
       [0.0e+00, 0.0e+00, 3.6e+03],
       [0.0e+00, 0.0e+00, 4.0e+03],
       [0.0e+00, 1.0e+00, 2.6e+03],
       [0.0e+00, 1.0e+00, 2.8e+03],
       [0.0e+00, 1.0e+00, 3.3e+03],
       [0.0e+00, 1.0e+00, 3.6e+03],
       [1.0e+00, 0.0e+00, 2.6e+03],
       [1.0e+00, 0.0e+00, 2.9e+03],
       [1.0e+00, 0.0e+00, 3.1e+03],
       [1.0e+00, 0.0e+00, 3.6e+03]])
```

```
model.fit(x,y)
```

```
LinearRegression()
```

```
model.predict([[1,0,2800]])
```

```
array([590775.63964739])
```

```
model.predict([[ 0,1,3400]])
```

```
array([681241.6684584])
```

```
model.score(x,y)
```

```
0.9573929037221873
```

So,it is giving same output as earlier

Now we have to predict price of Mercedes benz that is 4 yrs old with run of 45000,bmw x5 that is 7 yr old with run of 86000

```
import pandas as pd
df=pd.read_csv(r"C:\Users\Ayush\OneDrive\Documents\cars.csv")
df
```

	carmodel	mileage	sellprice	age
0	BMW X5	69000	18000	6
1	BMW X5	35000	34000	3
2	BMW X5	57000	26100	5
3	BMW X5	22500	40000	2
4	BMW X5	46000	31500	4

```
5    AUDI A5    59000      29400    5
6    AUDI A5    52000      32000    5
7    AUDI A5    72000      19300    6
8    AUDI A5    91000      12000    8
9   mercedez    67000      22000    6
10  mercedez    83000      20000    7
11  mercedez    79000      21000    7
12  mercedez    59000      33000    5
```

```
dummies=pd.get_dummies(df.carmodel)
dummies
```

```
    AUDI A5  BMW X5  mercedez
0         0       1         0
1         0       1         0
2         0       1         0
3         0       1         0
4         0       1         0
5         1       0         0
6         1       0         0
7         1       0         0
8         1       0         0
9         0       0         1
10        0       0         1
11        0       0         1
12        0       0         1
```

```
merged=pd.concat([df,dummies],axis='columns')
merged
```

```
    carmodel  mileage  sellprice  age  AUDI A5  BMW X5  mercedez
0     BMW X5    69000      18000    6        0       1         0
1     BMW X5    35000      34000    3        0       1         0
2     BMW X5    57000      26100    5        0       1         0
3     BMW X5    22500      40000    2        0       1         0
4     BMW X5    46000      31500    4        0       1         0
5    AUDI A5    59000      29400    5        1       0         0
6    AUDI A5    52000      32000    5        1       0         0
7    AUDI A5    72000      19300    6        1       0         0
8    AUDI A5    91000      12000    8        1       0         0
9   mercedez    67000      22000    6        0       0         1
10  mercedez    83000      20000    7        0       0         1
11  mercedez    79000      21000    7        0       0         1
12  mercedez    59000      33000    5        0       0         1
```

```
final=merged.drop(['carmodel','AUDI A5','sellprice'],axis='columns')
final
```

```
    mileage  age  BMW X5  mercedez
0     69000    6       1         0
1     35000    3       1         0
```

```
2     57000    5       1          0
3     22500    2       1          0
4     46000    4       1          0
5     59000    5       0          0
6     52000    5       0          0
7     72000    6       0          0
8     91000    8       0          0
9     67000    6       0          1
10    83000    7       0          1
11    79000    7       0          1
12    59000    5       0          1
```

```
y=df.sellprice
y
```

```
0     18000
1     34000
2     26100
3     40000
4     31500
5     29400
6     32000
7     19300
8     12000
9     22000
10    20000
11    21000
12    33000
Name: sellprice, dtype: int64
```

```
from sklearn.linear_model import LinearRegression
model=LinearRegression()
```

```
model.fit(final,y)
```

```
LinearRegression()
```

```
model.predict([[45000,4,0,1]])
```

```
C:\ProgramData\anaconda3\lib\site-packages\sklearn\base.py:420:
UserWarning: X does not have valid feature names, but LinearRegression
was fitted with feature names
  warnings.warn(
```

```
array([36991.31721061])
```

```
model.predict([[86000,7,1,0]])
```

```
C:\ProgramData\anaconda3\lib\site-packages\sklearn\base.py:420:
UserWarning: X does not have valid feature names, but LinearRegression
```

```
was fitted with feature names
  warnings.warn(

array([11080.74313219])

model.score(final,y)

0.9417050937281083
```

Let's Plot BAR

```
import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline

company=["GOOGL","AMZN","MSFT","FB"]
revenue=[90,136,89,27]
profit=[30,20,45,10]
```

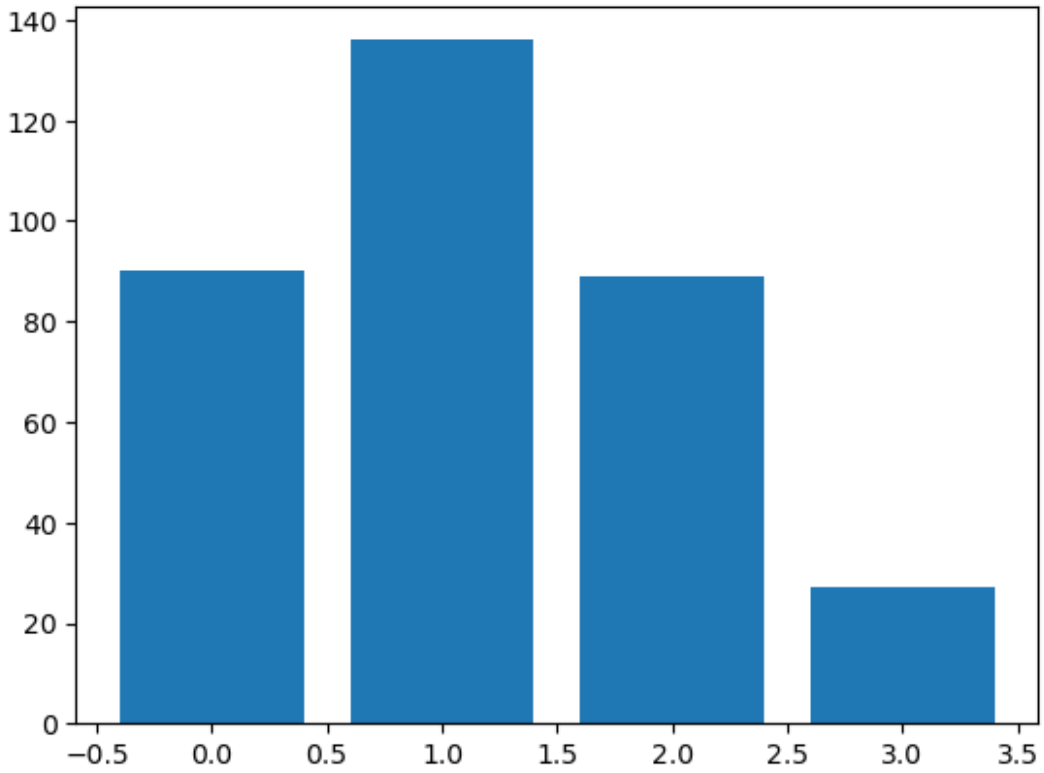
Bar doesnot take string input ,it has to be made list and lets name it YPOS

```
ypos=np.arange(len(company))
ypos
```

```
array([0, 1, 2, 3])
```

```
plt.bar(ypos,revenue)
```

```
<BarContainer object of 4 artists>
```



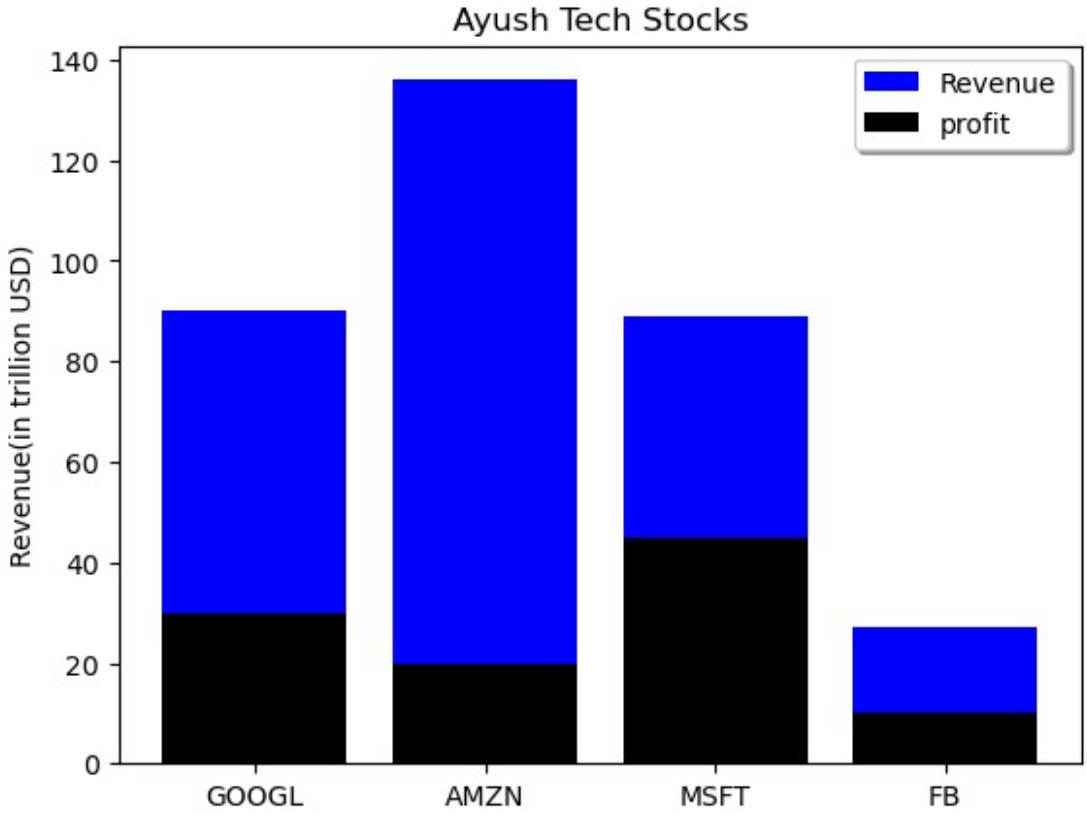
xticks(ypos,company) replaces 0 by GOOGL,1 by AMZN and so on

```
xpos=np.arange(len(company))
xpos
```

```
array([0, 1, 2, 3])
```

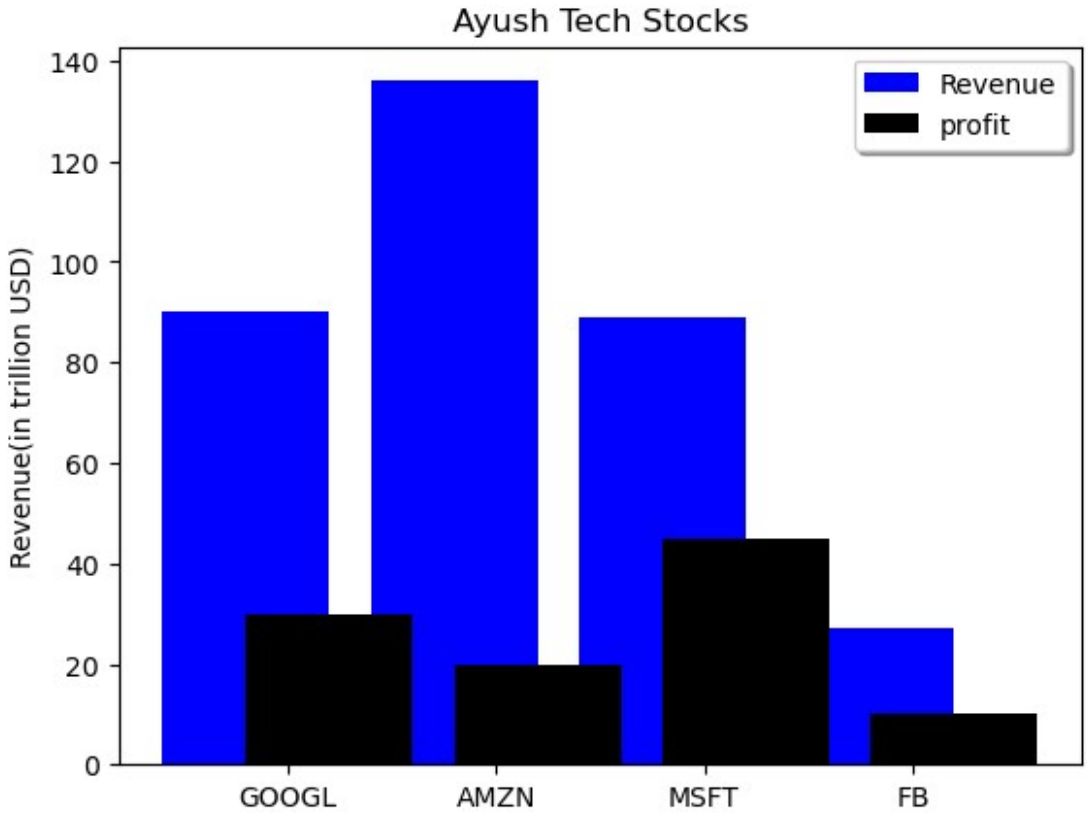
```
plt.xticks(ypos,company)
plt.ylabel("Revenue(in trillion USD)")
plt.title("Ayush Tech Stocks")
plt.bar(xpos,revenue,label="Revenue",color="blue")
plt.bar(xpos,profit,label="profit",color="black")
plt.legend(shadow=True)
```

```
<matplotlib.legend.Legend at 0x1da002c7850>
```



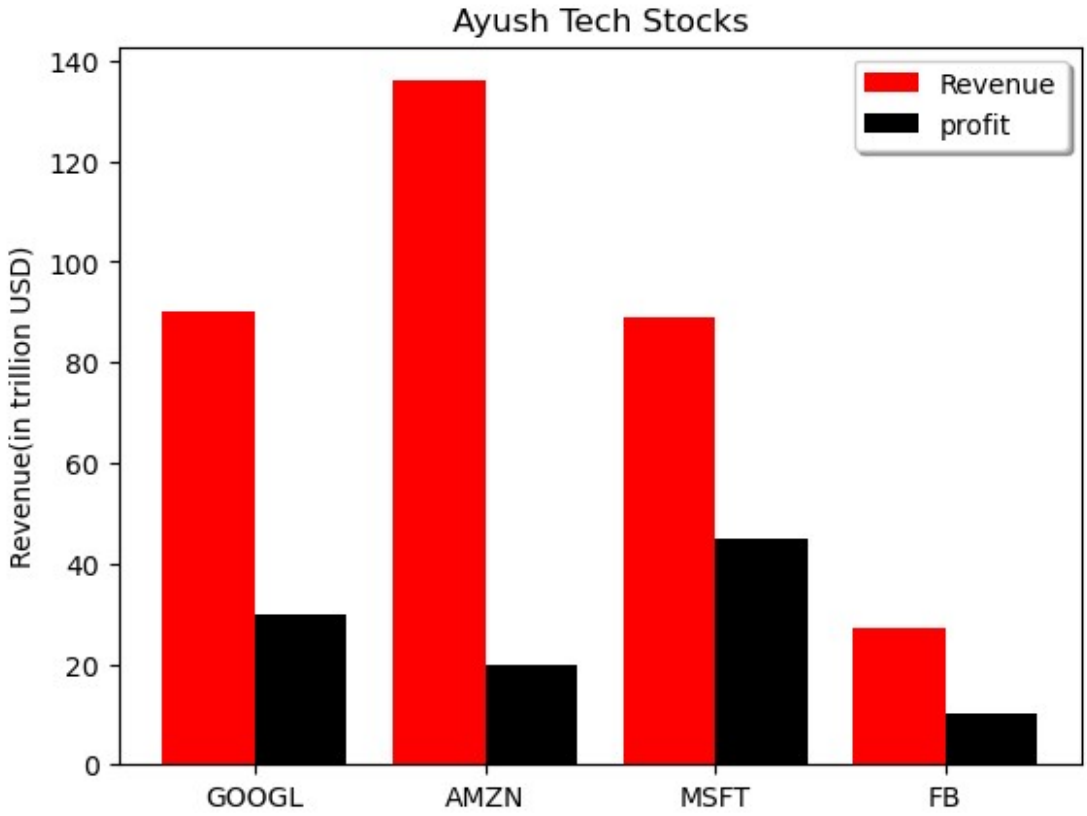

```
plt.xticks(ypos,company)
plt.ylabel("Revenue(in trillion USD)")
plt.title("Ayush Tech Stocks")
plt.bar(xpos-0.2,revenue,label="Revenue",color="blue")
plt.bar(xpos+0.2,profit,label="profit",color="black")
plt.legend(shadow=True)
```

```
<matplotlib.legend.Legend at 0x1da002c4880>
```



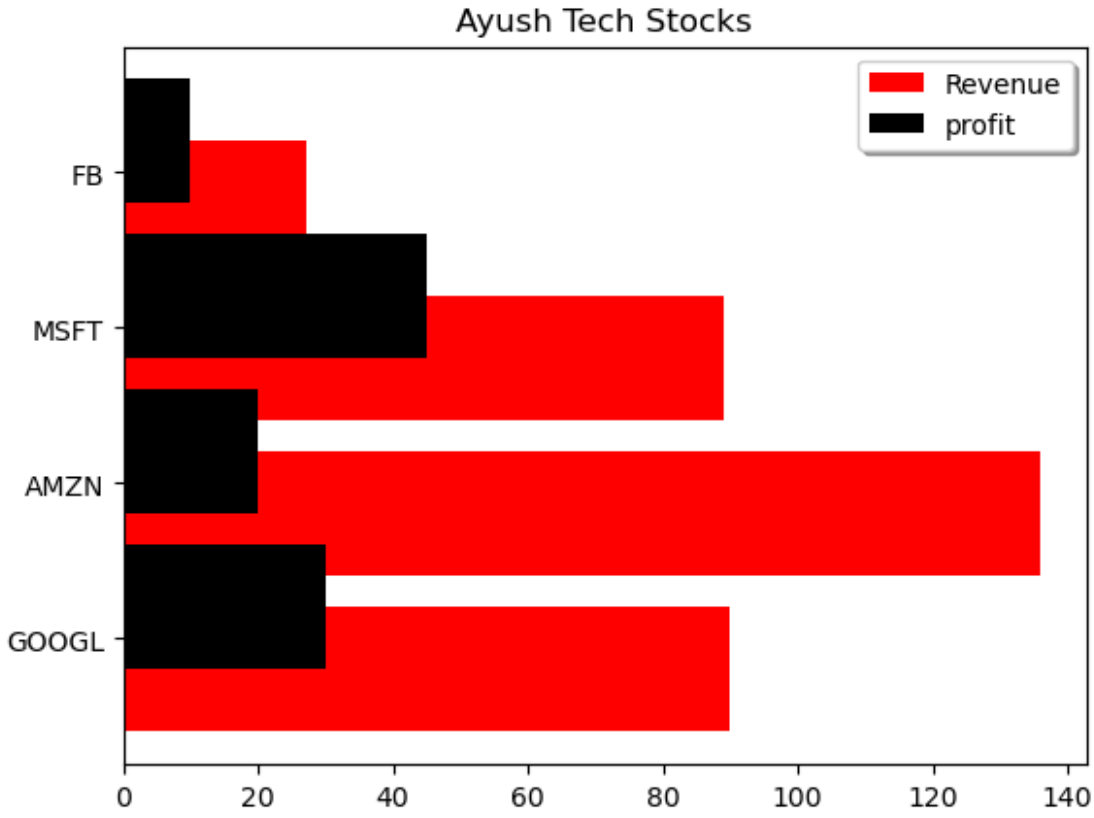

```
plt.xticks(ypos,company)
plt.ylabel("Revenue(in trillion USD)")
plt.title("Ayush Tech Stocks")
plt.bar(xpos-0.2,revenue,width=0.4,label="Revenue",color="red")
plt.bar(xpos+0.2,profit,width=0.4,label="profit",color="black")
plt.legend(shadow=True)
```

```
<matplotlib.legend.Legend at 0x238933cc700>
```




```
plt.yticks(xpos,company)

plt.title("Ayush Tech Stocks")
plt.barh(xpos-0.2,revenue,label="Revenue",color="red")
plt.barh(xpos+0.2,profit,label="profit",color="black")
plt.legend(shadow=True)
```

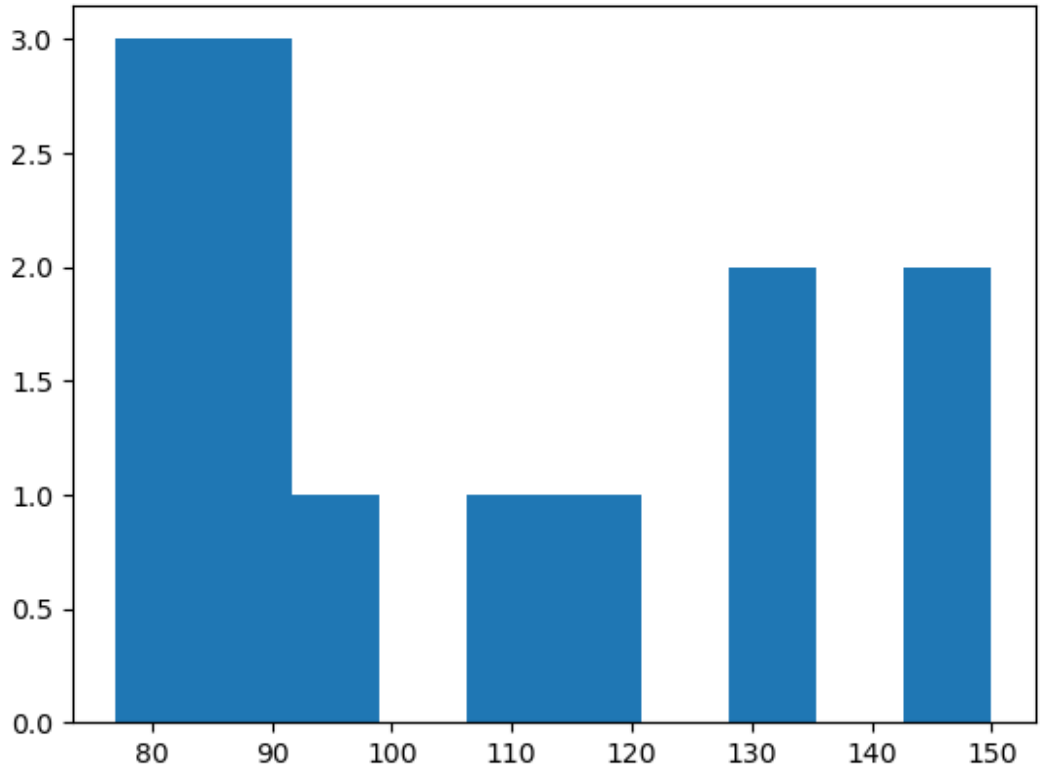
```
<matplotlib.legend.Legend at 0x23894a7f9a0>
```



Let's Plot HISTOGRAM

```
blood_sugar=[113,85,90,150,149,88,93,115,135,80,77,82,129]
plt.hist(blood_sugar)
```

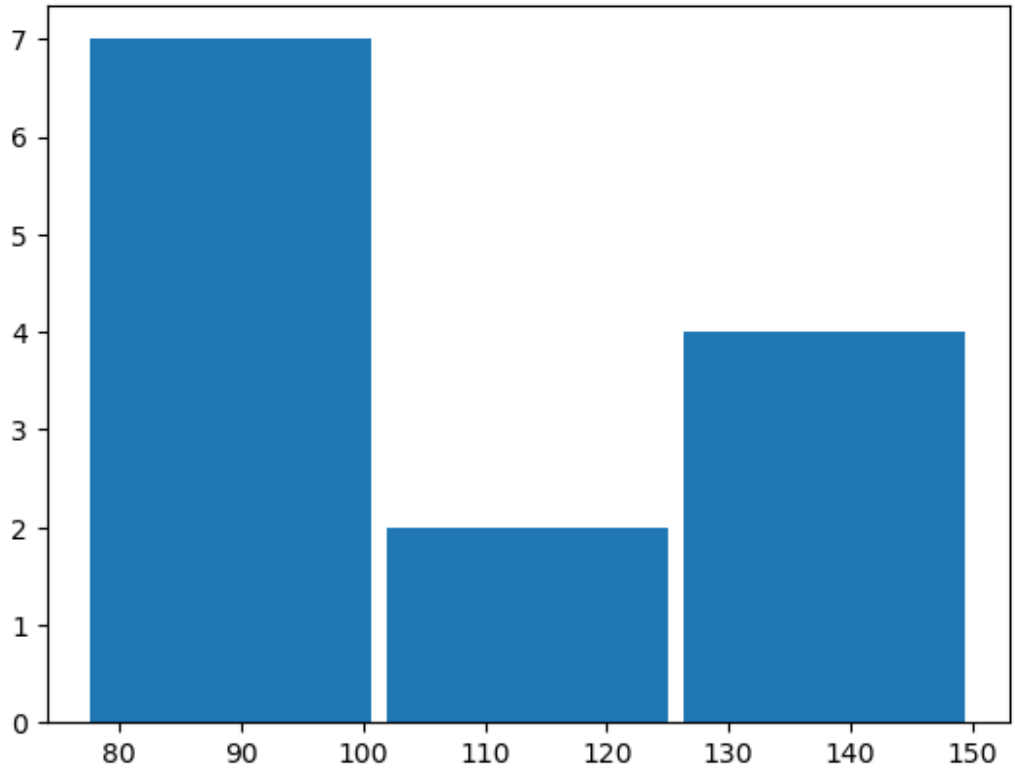
```
(array([3., 3., 1., 0., 1., 1., 0., 2., 0., 2.]),
 array([ 77. ,  84.3,  91.6,  98.9, 106.2, 113.5, 120.8, 128.1, 135.4,
        142.7, 150. ]),
 <BarContainer object of 10 artists>)
```



These are called Bins or buckets

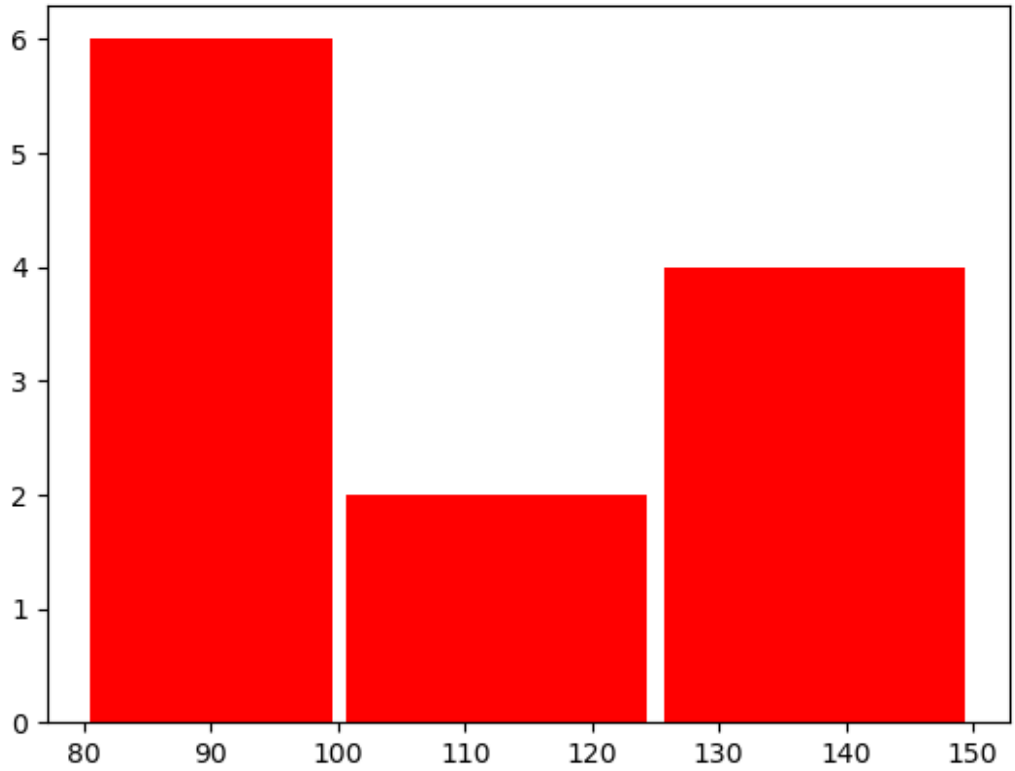
```
blood_sugar=[113,85,90,150,149,88,93,115,135,80,77,82,129]
plt.hist(blood_sugar,bins=3,rwidth=0.95)
```

```
(array([7., 2., 4.]),
 array([ 77.        , 101.33333333, 125.66666667, 150.        ]),
 <BarContainer object of 3 artists>)
```



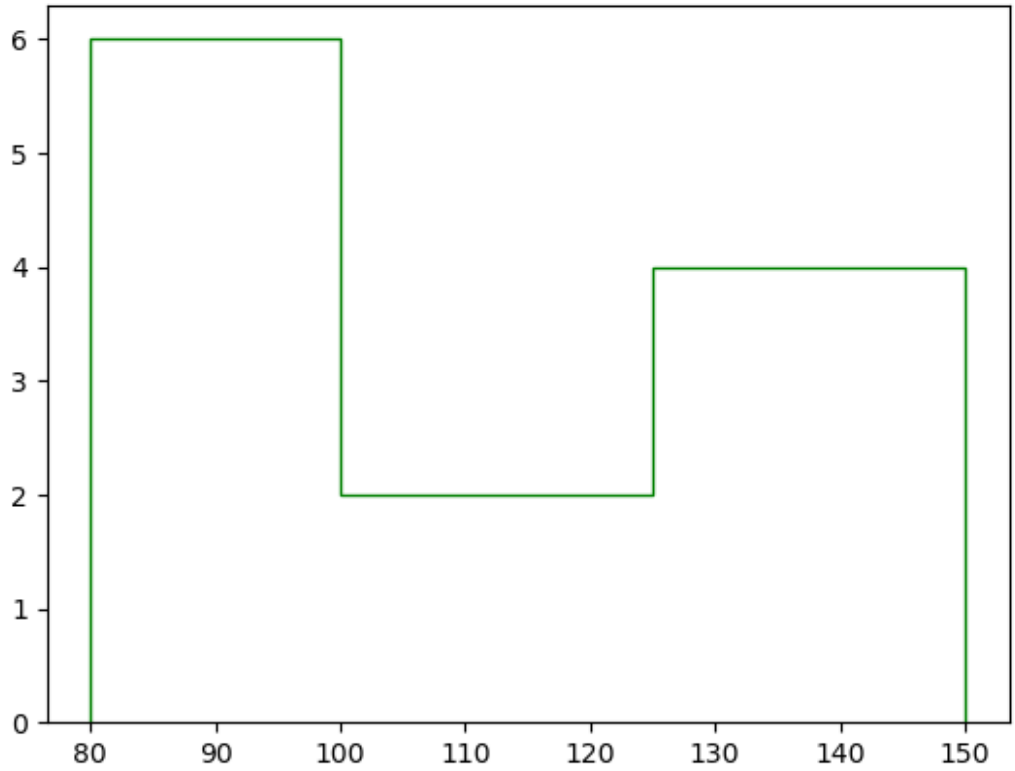

```
blood_sugar=[113,85,90,150,149,88,93,115,135,80,77,82,129]
plt.hist(blood_sugar,bins=[80,100,125,150],rwidth=0.95,color="r")
```

```
(array([6., 2., 4.]),
 array([ 80., 100., 125., 150.]),
 <BarContainer object of 3 artists>)
```

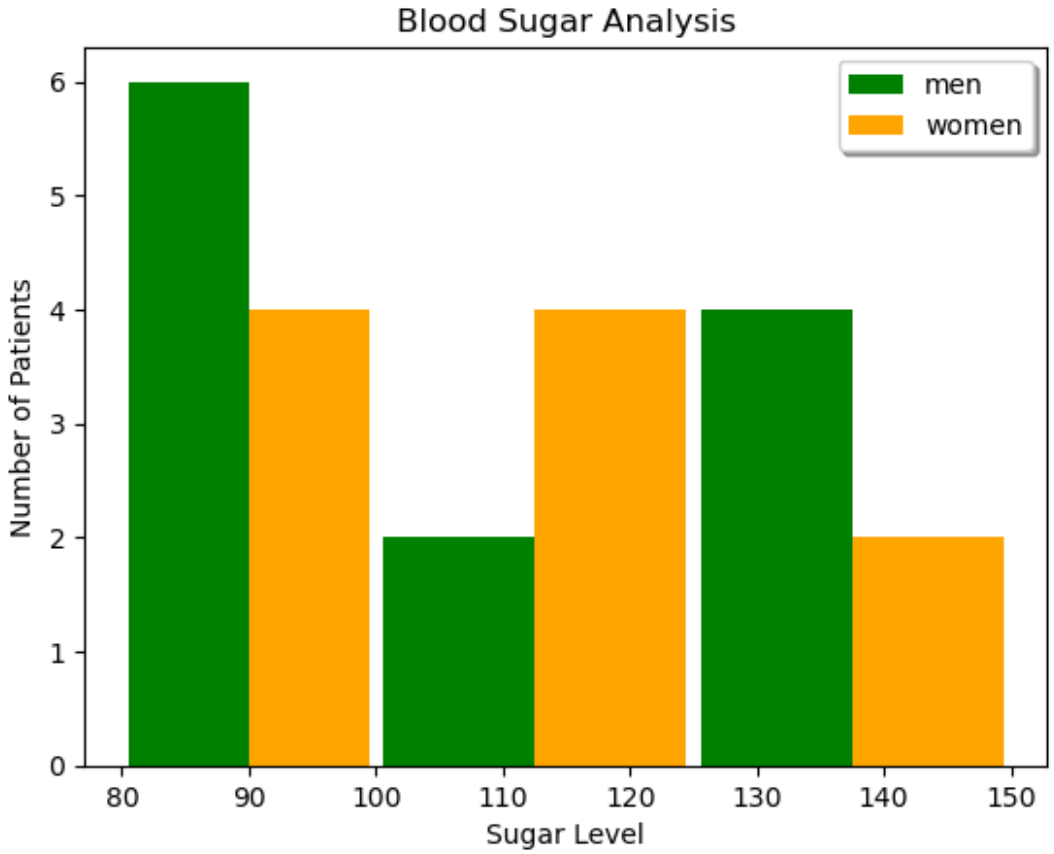



```
blood_sugar=[113,85,90,150,149,88,93,115,135,80,77,82,129]
plt.hist(blood_sugar,bins=[80,100,125,150],rwidth=0.95,color="g",histt
ype="step")
```

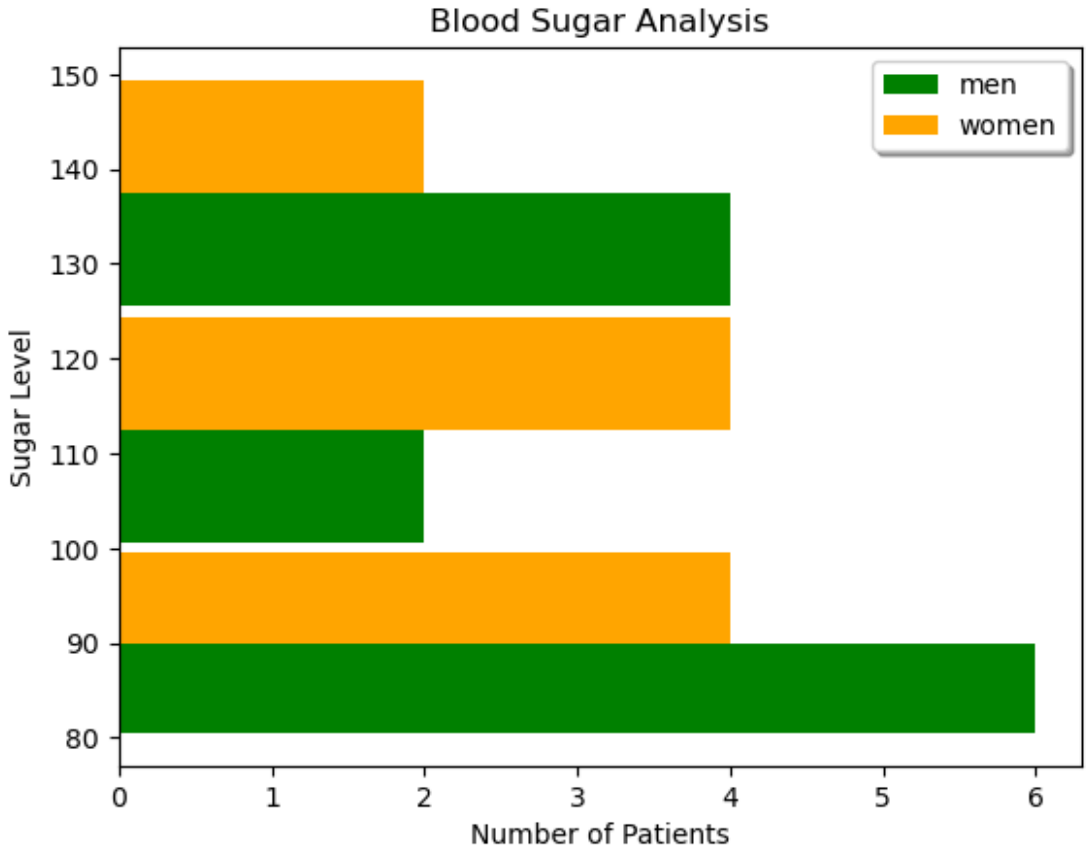
```
(array([6., 2., 4.]),
 array([ 80., 100., 125., 150.]),
 [<matplotlib.patches.Polygon at 0x23895f47910>])
```




```
blood_sugar_men=[113,85,90,150,149,88,93,115,135,80,77,82,129]
blood_sugar_women=[67,98,89,120,133,150,84,69,89,79,120,112,100]
plt.hist([blood_sugar_men,blood_sugar_women],bins=[80,100,125,150],
         rwidth=0.95,color=["green","orange"],label=["men","women"])
plt.legend(shadow=True)
plt.xlabel("Sugar Level")
plt.ylabel("Number of Patients")
plt.title("Blood Sugar Analysis")


Text(0.5, 1.0, 'Blood Sugar Analysis')
```



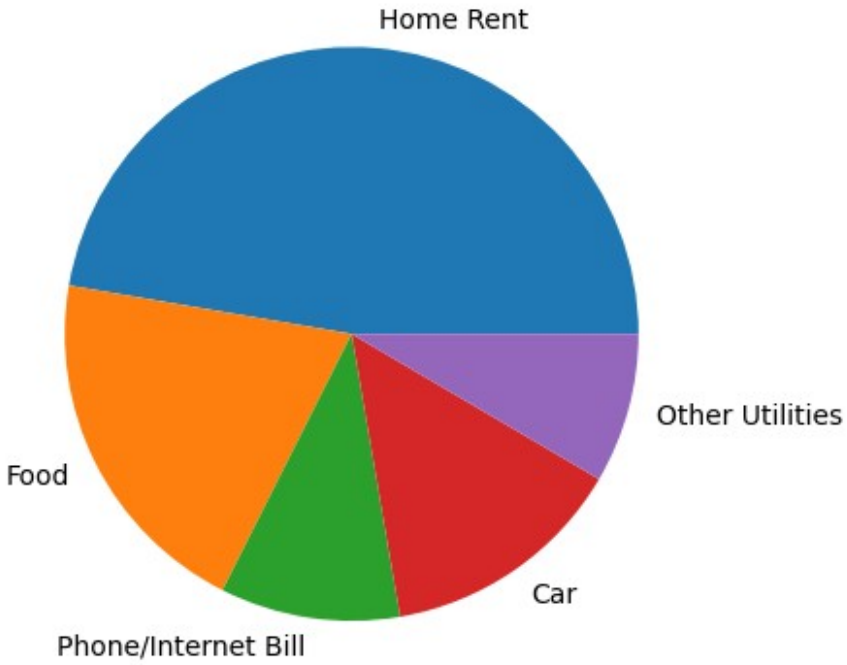

```
blood_sugar_men=[113,85,90,150,149,88,93,115,135,80,77,82,129]
blood_sugar_women=[67,98,89,120,133,150,84,69,89,79,120,112,100]
plt.hist([blood_sugar_men,blood_sugar_women],bins=[80,100,125,150],
        rwidth=0.95,color=["green","orange"],label=["men","women"]
        ,orientation="horizontal")
plt.legend(shadow=True)
plt.ylabel("Sugar Level")
plt.xlabel("Number of Patients")
plt.title("Blood Sugar Analysis")


Text(0.5, 1.0, 'Blood Sugar Analysis')
```



Lets Look Now at PIE CHART

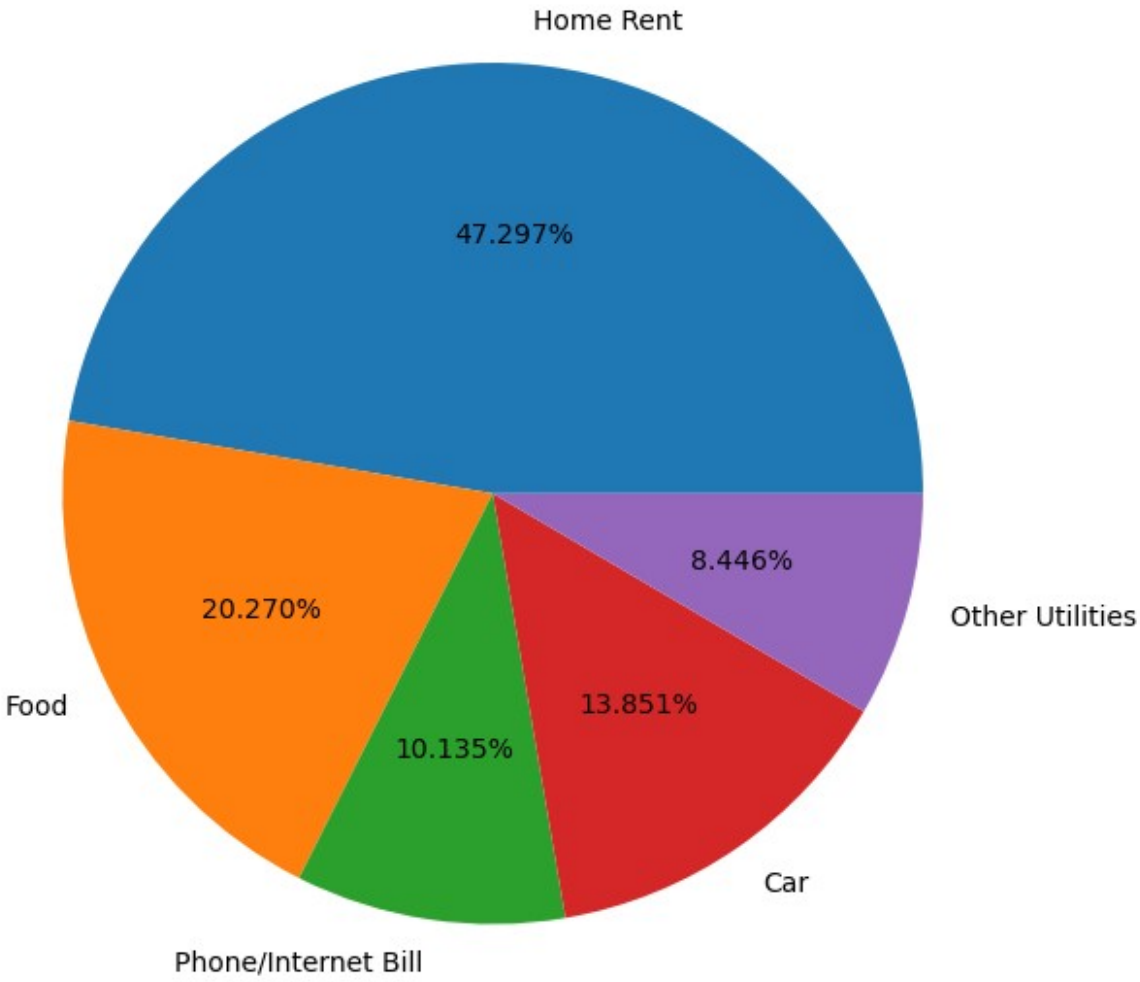
```
%matplotlib inline
import matplotlib.pyplot as plt
exp_vals = [1400, 600, 300, 410, 250]
exp_labels = ["Home Rent", "Food", "Phone/Internet Bill", "Car",
"Other Utilities"]
plt.axis("equal")
plt.pie(exp_vals,labels=exp_labels)
```

```
([<matplotlib.patches.Wedge at 0x238966f4430>,
  <matplotlib.patches.Wedge at 0x238966f4340>,
  <matplotlib.patches.Wedge at 0x238966f4af0>,
  <matplotlib.patches.Wedge at 0x238966f4f70>,
  <matplotlib.patches.Wedge at 0x238966f53c0>],
 [Text(0.09328656407206024, 1.0960372333838069, 'Home Rent'),
  Text(-0.9822184890776084, -0.4952240298229684, 'Food'),
  Text(-0.16284704617934698, -1.0878790555712807, 'Phone/Internet
Bill'),
  Text(0.6256100334857941, -0.9047718419590123, 'Car'),
  Text(1.0615045230766318, -0.28845822485734873, 'Other Utilities')])
```

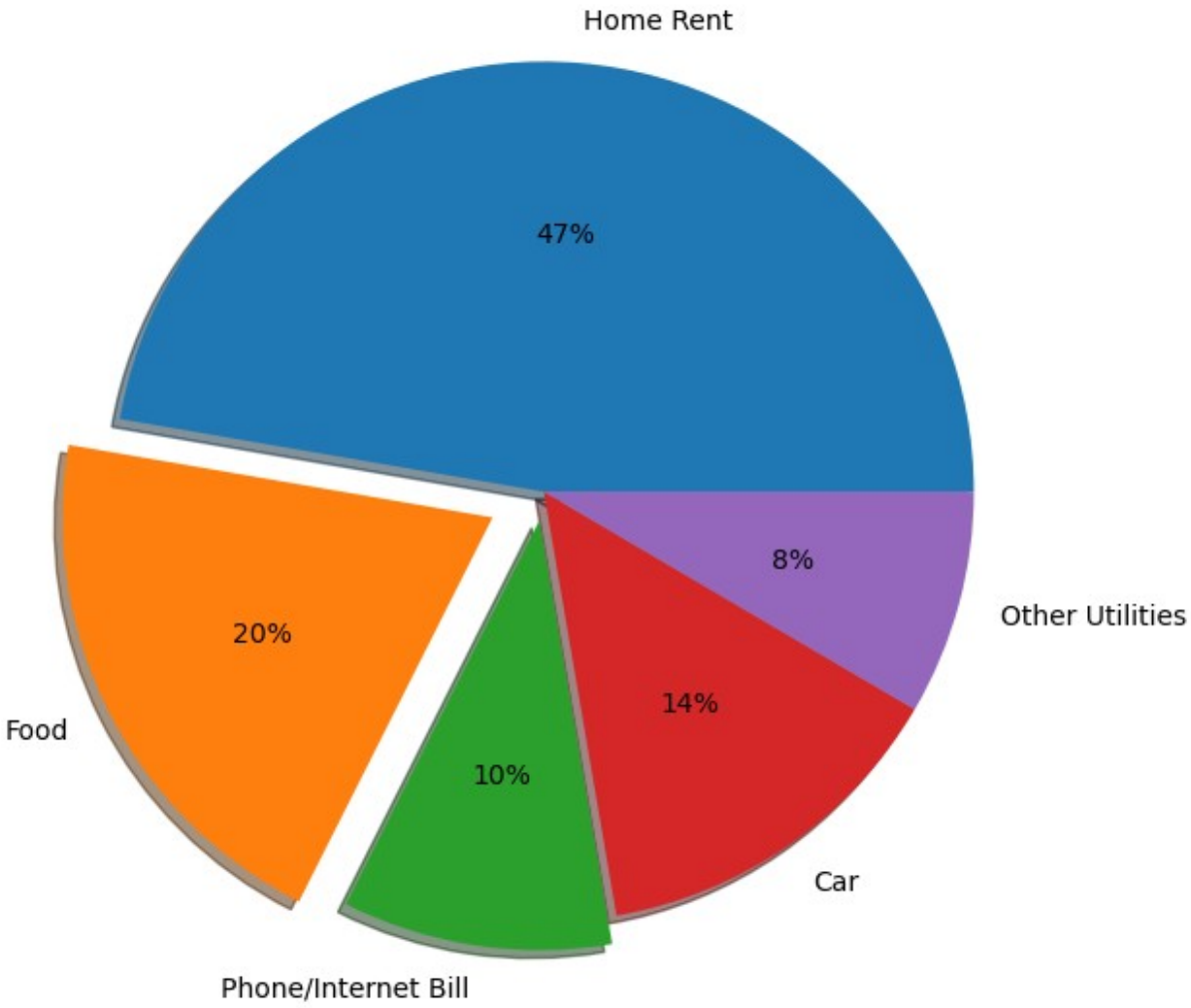


plt.axis("equal")makes it perfect circle

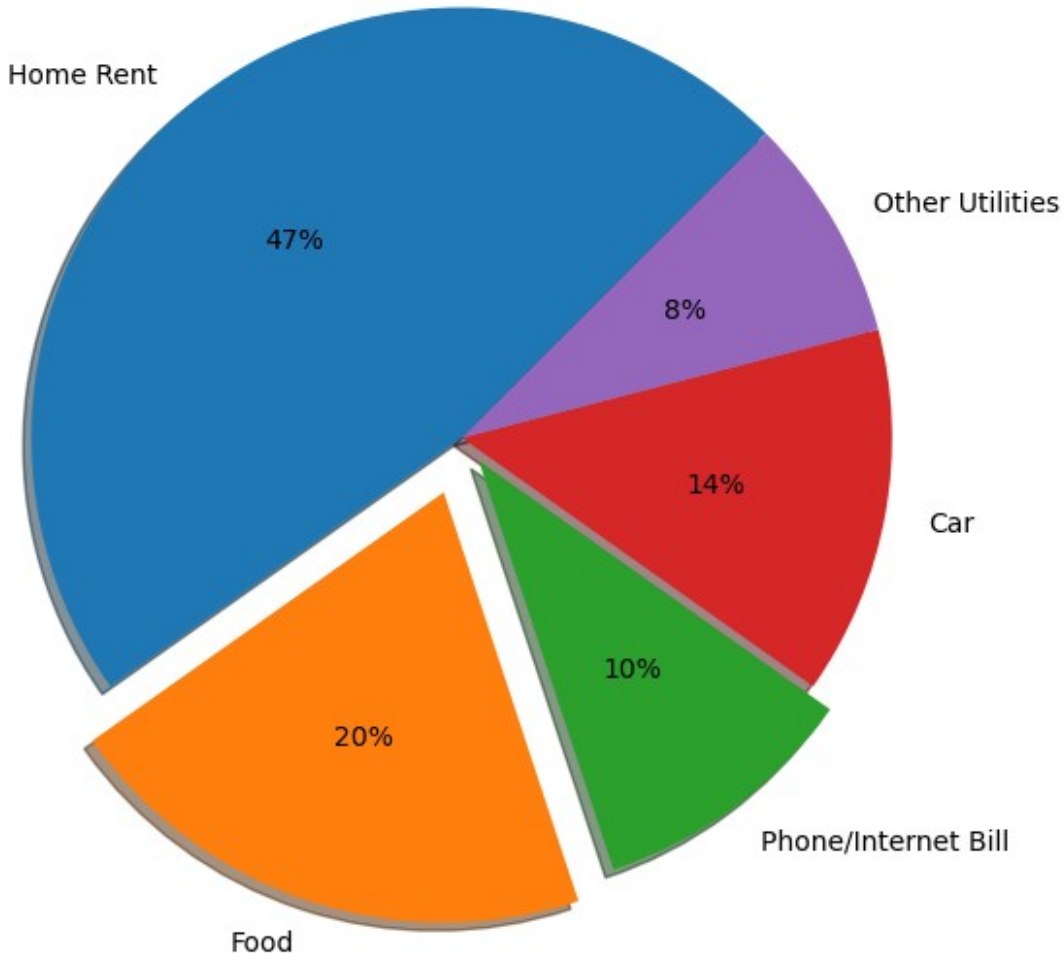
```
%matplotlib inline
import matplotlib.pyplot as plt

exp_vals = [1400, 600, 300, 410, 250]
exp_labels = ["Home Rent", "Food", "Phone/Internet Bill", "Car",
"Other Utilities"]
plt.axis("equal")
plt.pie(exp_vals, labels=exp_labels,radius=1.5,autopct="%0.3f%%")
plt.show()
```




```
%matplotlib inline
import matplotlib.pyplot as plt

exp_vals = [1400, 600, 300, 410, 250]
exp_labels = ["Home Rent", "Food", "Phone/Internet Bill", "Car",
"Other Utilities"]
plt.axis("equal")
plt.pie(exp_vals, labels=exp_labels,radius=1.5,autopct="%0.0f%
%",shadow=True,explode=[0,0.2,0.1,0,0])
plt.show()
```




```
%matplotlib inline
import matplotlib.pyplot as plt

exp_vals = [1400, 600, 300, 410, 250]
exp_labels = ["Home Rent", "Food", "Phone/Internet Bill", "Car",
"Other Utilities"]
plt.axis("equal")
plt.pie(exp_vals,
labels=exp_labels,startangle=45,radius=1.5,autopct="%0.0f%
%",shadow=True,explode=[0,0.2,0.1,0,0])
plt.show()
```


Home Rent
47%
Other Utilities
8%
14%
Car
10%
20%
Phone/Internet Bill
Food

Functions,Calendar,List,Tuple and Dictionary in Python

```
def addition(a,b=0):
    return a+b;

t=addition(2,3)

print(t+3)
```

```
8
```

```
import calendar as c

dir(calendar)
```

```
['Calendar',
 'EPOCH',
 'FRIDAY',
 'February',
 'HTMLCalendar',
 'IllegalMonthError',
 'IllegalWeekdayError',
 'January',
 'LocaleHTMLCalendar',
 'LocaleTextCalendar',
 'MONDAY',
 'SATURDAY',
 'SUNDAY',
 'THURSDAY',
 'TUESDAY',
 'TextCalendar',
 'WEDNESDAY',
 '_EPOCH_ORD',
 '__all__',
 '__builtins__',
 '__cached__',
 '__doc__',
 '__file__',
 '__loader__',
 '__name__',
 '__package__',
 '__spec__',
 '_colwidth',
 '_locale',
 '_localized_day',
 '_localized_month',
 '_monthlen',
```

```
 '_nextmonth',
 '_prevmonth',
 '_spacing',
 'c',
 'calendar',
 'datetime',
 'day_abbr',
 'day_name',
 'different_locale',
 'error',
 'firstweekday',
 'format',
 'formatstring',
 'isleap',
 'leapdays',
 'main',
 'mdays',
 'month',
 'month_abbr',
 'month_name',
 'monthcalendar',
 'monthrange',
 'prcal',
 'prmonth',
 'prweek',
 'repeat',
 'setfirstweekday',
 'sys',
 'timegm',
 'week',
 'weekday',
 'weekheader']
```

```
print(c.month(2023,10))
```

```
    October 2023
Mo Tu We Th Fr Sa Su
                   1
 2  3  4  5  6  7  8
 9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29
30 31
```

```
c.isleap(2023)
```

```
False
```

```
dict={"ayush":3,"shaurya":5,"jha":2}
```

```
print(dict["ayush"])
dict["ayush"]=-9
print(dict["ayush"])
```

```
3
-9
```

```
t=("ayush",10123,"New York")
print(t[0]+t[2])
print(t[0:])
```

```
ayushNew York
('ayush', 10123, 'New York')
```

```
t[0]="king ayush"
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[23], line 1
----> 1 t[0]="king ayush"

TypeError: 'tuple' object does not support item assignment
```

```
list=[[1,23,3],2,3]
```

```
print(list)
```

```
[[1, 23, 3], 2, 3]
```

```
list[0]
```

```
[1, 23, 3]
```

```
list[0][1]
```

```
23
```

```
dir(numpy)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[30], line 1
----> 1 dir(numpy)

NameError: name 'numpy' is not defined
```

```
import numpy
dir(numpy)
```

```
['ALLOW_THREADS',
 'AxisError',
 'BUFSIZE',
 'CLIP',
 'ComplexWarning',
 'DataSource',
 'ERR_CALL',
 'ERR_DEFAULT',
 'ERR_IGNORE',
 'ERR_LOG',
 'ERR_PRINT',
 'ERR_RAISE',
 'ERR_WARN',
 'FLOATING_POINT_SUPPORT',
 'FPE_DIVIDEBYZERO',
 'FPE_INVALID',
 'FPE_OVERFLOW',
 'FPE_UNDERFLOW',
 'False_',
 'Inf',
 'Infinity',
 'MAXDIMS',
 'MAY_SHARE_BOUNDS',
 'MAY_SHARE_EXACT',
 'ModuleDeprecationWarning',
 'NAN',
 'NINF',
 'NZERO',
 'NaN',
 'PINF',
 'PZERO',
 'RAISE',
 'RankWarning',
 'SHIFT_DIVIDEBYZERO',
 'SHIFT_INVALID',
 'SHIFT_OVERFLOW',
 'SHIFT_UNDERFLOW',
 'ScalarType',
 'Tester',
 'TooHardError',
 'True_',
 'UFUNC_BUFSIZE_DEFAULT',
 'UFUNC_PYVALS_NAME',
 'VisibleDeprecationWarning',
 'WRAP',
 '_CopyMode',
 '_NoValue',
 '_UFUNC_API',
 '__NUMPY_SETUP__',
 '__all__',
```

```
'__builtins__',
'__cached__',
'__config__',
'__deprecated_attrs__',
'__dir__',
'__doc__',
'__expired_functions__',
'__file__',
'__getattr__',
'__git_version__',
'__loader__',
'__mkl_version__',
'__name__',
'__package__',
'__path__',
'__spec__',
'__version__',
'_add_newdoc_ufunc',
'_distributor_init',
'_financial_names',
'_globals',
'_mat',
'_pyinstaller_hooks_dir',
'_pytesttester',
'_version',
'abs',
'absolute',
'add',
'add_docstring',
'add_newdoc',
'add_newdoc_ufunc',
'all',
'allclose',
'alltrue',
'amax',
'amin',
'angle',
'any',
'append',
'apply_along_axis',
'apply_over_axes',
'arange',
'arccos',
'arccosh',
'arcsin',
'arcsinh',
'arctan',
'arctan2',
'arctanh',
'argmax',
```

```
'argmin',
'argpartition',
'argsort',
'argwhere',
'around',
'array',
'array2string',
'array_equal',
'array_equiv',
'array_repr',
'array_split',
'array_str',
'asanyarray',
'asarray',
'asarray_chkfinite',
'ascontiguousarray',
'asfarray',
'asfortranarray',
'asmatrix',
'atleast_1d',
'atleast_2d',
'atleast_3d',
'average',
'bartlett',
'base_repr',
'binary_repr',
'bincount',
'bitwise_and',
'bitwise_not',
'bitwise_or',
'bitwise_xor',
'blackman',
'block',
'bmat',
'bool8',
'bool_',
'broadcast',
'broadcast_arrays',
'broadcast_shapes',
'broadcast_to',
'busday_count',
'busday_offset',
'busdaycalendar',
'byte',
'byte_bounds',
'bytes0',
'bytes_',
'c_',
'can_cast',
'cast',
```

```
'cbrt',
'cdouble',
'ceil',
'cfloat',
'char',
'character',
'chararray',
'choose',
'clip',
'clongdouble',
'clongfloat',
'column_stack',
'common_type',
'compare_chararrays',
'compat',
'complex128',
'complex64',
'complex_',
'complexfloating',
'compress',
'concatenate',
'conj',
'conjugate',
'convolve',
'copy',
'copysign',
'copyto',
'core',
'corrcoef',
'correlate',
'cos',
'cosh',
'count_nonzero',
'cov',
'cross',
'csingle',
'ctypeslib',
'cumprod',
'cumproduct',
'cumsum',
'datetime64',
'datetime_as_string',
'datetime_data',
'deg2rad',
'degrees',
'delete',
'deprecate',
'deprecate_with_doc',
'diag',
'diag_indices',
```

```
'diag_indices_from',
'diagflat',
'diagonal',
'diff',
'digitize',
'disp',
'divide',
'divmod',
'dot',
'double',
'dsplit',
'dstack',
'dtype',
'e',
'ediff1d',
'einsum',
'einsum_path',
'emath',
'empty',
'empty_like',
'equal',
'errstate',
'euler_gamma',
'exp',
'exp2',
'expand_dims',
'expm1',
'extract',
'eye',
'fabs',
'fastCopyAndTranspose',
'fft',
'fill_diagonal',
'find_common_type',
'finfo',
'fix',
'flatiter',
'flatnonzero',
'flexible',
'flip',
'fliplr',
'flipud',
'float16',
'float32',
'float64',
'float_',
'float_power',
'floating',
'floor',
'floor_divide',
```

```
'fmax',
'fmin',
'fmod',
'format_float_positional',
'format_float_scientific',
'format_parser',
'frexp',
'from_dlpack',
'frombuffer',
'fromfile',
'fromfunction',
'fromiter',
'frompyfunc',
'fromregex',
'fromstring',
'full',
'full_like',
'gcd',
'generic',
'genfromtxt',
'geomspace',
'get_array_wrap',
'get_include',
'get_printoptions',
'getbufsize',
'geterr',
'geterrcall',
'geterrobj',
'gradient',
'greater',
'greater_equal',
'half',
'hamming',
'hanning',
'heaviside',
'histogram',
'histogram2d',
'histogram_bin_edges',
'histogramdd',
'hsplit',
'hstack',
'hypot',
'i0',
'identity',
'iinfo',
'imag',
'in1d',
'index_exp',
'indices',
'inexact',
```

```
 'inf',
 'info',
 'infty',
 'inner',
 'insert',
 'int0',
 'int16',
 'int32',
 'int64',
 'int8',
 'int_',
 'intc',
 'integer',
 'interp',
 'intersect1d',
 'intp',
 'invert',
 'is_busday',
 'isclose',
 'iscomplex',
 'iscomplexobj',
 'isfinite',
 'isfortran',
 'isin',
 'isinf',
 'isnan',
 'isnat',
 'isneginf',
 'isposinf',
 'isreal',
 'isrealobj',
 'isscalar',
 'issctype',
 'issubclass_',
 'issubdtype',
 'issubsctype',
 'iterable',
 'ix_',
 'kaiser',
 'kron',
 'lcm',
 'ldexp',
 'left_shift',
 'less',
 'less_equal',
 'lexsort',
 'lib',
 'linalg',
 'linspace',
 'little_endian',
```

```
'load',
'loadtxt',
'log',
'log10',
'log1p',
'log2',
'logaddexp',
'logaddexp2',
'logical_and',
'logical_not',
'logical_or',
'logical_xor',
'logspace',
'longcomplex',
'longdouble',
'longfloat',
'longlong',
'lookfor',
'ma',
'mask_indices',
'mat',
'math',
'matmul',
'matrix',
'matrixlib',
'max',
'maximum',
'maximum_sctype',
'may_share_memory',
'mean',
'median',
'memmap',
'meshgrid',
'mgrid',
'min',
'min_scalar_type',
'minimum',
'mintypecode',
'mkl',
'mod',
'modf',
'moveaxis',
'msort',
'multiply',
'nan',
'nan_to_num',
'nanargmax',
'nanargmin',
'nancumprod',
'nancumsum',
```

```
'nanmax',
'nanmean',
'nanmedian',
'nanmin',
'nanpercentile',
'nanprod',
'nanquantile',
'nanstd',
'nansum',
'nanvar',
'nbytes',
'ndarray',
'ndenumerate',
'ndim',
'ndindex',
'nditer',
'negative',
'nested_iters',
'newaxis',
'nextafter',
'nonzero',
'not_equal',
'numarray',
'number',
'obj2sctype',
'object0',
'object_',
'ogrid',
'oldnumeric',
'ones',
'ones_like',
'os',
'outer',
'packbits',
'pad',
'partition',
'percentile',
'pi',
'piecewise',
'place',
'poly',
'poly1d',
'polyadd',
'polyder',
'polydiv',
'polyfit',
'polyint',
'polymul',
'polynomial',
'polysub',
```

```
'polyval',
'positive',
'power',
'printoptions',
'prod',
'product',
'promote_types',
'ptp',
'put',
'put_along_axis',
'putmask',
'quantile',
'r_',
'rad2deg',
'radians',
'random',
'ravel',
'ravel_multi_index',
'real',
'real_if_close',
'rec',
'recarray',
'recfromcsv',
'recfromtxt',
'reciprocal',
'record',
'remainder',
'repeat',
'require',
'reshape',
'resize',
'result_type',
'right_shift',
'rint',
'roll',
'rollaxis',
'roots',
'rot90',
'round',
'round_',
'row_stack',
's_',
'safe_eval',
'save',
'savetxt',
'savez',
'savez_compressed',
'sctype2char',
'sctypeDict',
'sctypes',
```

```
 'searchsorted',
 'select',
 'set_numeric_ops',
 'set_printoptions',
 'set_string_function',
 'setbufsize',
 'setdiff1d',
 'seterr',
 'seterrcall',
 'seterrobj',
 'setxor1d',
 'shape',
 'shares_memory',
 'short',
 'show_config',
 'sign',
 'signbit',
 'signedinteger',
 'sin',
 'sinc',
 'single',
 'singlecomplex',
 'sinh',
 'size',
 'sometrue',
 'sort',
 'sort_complex',
 'source',
 'spacing',
 'split',
 'sqrt',
 'square',
 'squeeze',
 'stack',
 'std',
 'str0',
 'str_',
 'string_',
 'subtract',
 'sum',
 'swapaxes',
 'sys',
 'take',
 'take_along_axis',
 'tan',
 'tanh',
 'tensordot',
 'test',
 'testing',
 'tile',
```

```
'timedelta64',
'trace',
'tracemalloc_domain',
'transpose',
'trapz',
'tri',
'tril',
'tril_indices',
'tril_indices_from',
'trim_zeros',
'triu',
'triu_indices',
'triu_indices_from',
'true_divide',
'trunc',
'typecodes',
'typename',
'ubyte',
'ufunc',
'uint',
'uint0',
'uint16',
'uint32',
'uint64',
'uint8',
'uintc',
'uintp',
'ulonglong',
'unicode_',
'union1d',
'unique',
'unpackbits',
'unravel_index',
'unsignedinteger',
'unwrap',
'use_hugepage',
'ushort',
'vander',
'var',
'vdot',
'vectorize',
'version',
'void',
'void0',
'vsplit',
'vstack',
'warnings',
'where',
'who',
```

```
 'zeros',
 'zeros_like']
```

```
book={}
```

```
book['tim']={
    'name':'tim',
    'address':'gh',
    'phone':123
}
```

```
book['tom']={
    'name':'tom',
    'address':'hg',
    'phone':456
}
```

```
import json
s=json.dumps(book)
print(s)
```

```
{"tim": {"name": "tim", "address": "gh", "phone": 123}, "tom":
{"name": "tom", "address": "hg", "phone": 456}}
```

```
with open(r"C:\Users\Ayush\AppData\Local\Programs\Python\Python311\
p2.py", "r") as f:

    f.read()
```

it convert date string to actual date ,mainly in csv file as in xlxs its already converted

```
import pandas as pd
df=pd.read_excel(r"C:\Users\Ayush\Downloads\
newexcel2.xlsx",parse_dates=["day"])
df
```

	day	MinTemp	MaxTemp	Rainfall	Evaporation
0	2017-01-01	19.0	24.3	NaN	3.4
1	2017-01-04	17.0	NaN	3.6	4.4
2	2017-01-05	13.7	NaN	-1.0	5.8
3	2017-01-06	15.0	15.5	39.8	7.2
4	2017-01-07	NaN	16.1	-1.0	5.6
5	2017-01-08	NaN	16.9	0.0	5.8
6	2017-01-09	6.1	18.2	0.2	4.2

```
df.columns
```

```
Index(['day', 'MinTemp', 'MaxTemp', 'Rainfall', 'Evaporation'],
dtype='object')
```

```
df.set_index('day',inplace=True)
```

```
df
```

day	MinTemp	MaxTemp	Rainfall	Evaporation
2017-01-01	19.0	24.3	NaN	3.4
2017-01-04	17.0	NaN	3.6	4.4
2017-01-05	13.7	NaN	-1.0	5.8
2017-01-06	15.0	15.5	39.8	7.2
2017-01-07	NaN	16.1	-1.0	5.6
2017-01-08	NaN	16.9	0.0	5.8
2017-01-09	6.1	18.2	0.2	4.2

```
new_df=df.fillna(0)
new_df
```

day	MinTemp	MaxTemp	Rainfall	Evaporation
2017-01-01	19.0	24.3	0.0	3.4
2017-01-04	17.0	0.0	3.6	4.4
2017-01-05	13.7	0.0	-1.0	5.8
2017-01-06	15.0	15.5	39.8	7.2
2017-01-07	0.0	16.1	-1.0	5.6
2017-01-08	0.0	16.9	0.0	5.8
2017-01-09	6.1	18.2	0.2	4.2

```
new_df=df.fillna({
    'MinTemp':0,
     'MaxTemp':0
```

```
}
)
new_df
```

day	MinTemp	MaxTemp	Rainfall	Evaporation
2017-01-01	19.0	24.3	NaN	3.4
2017-01-04	17.0	0.0	3.6	4.4
2017-01-05	13.7	0.0	-1.0	5.8
2017-01-06	15.0	15.5	39.8	7.2
2017-01-07	0.0	16.1	-1.0	5.6
2017-01-08	0.0	16.9	0.0	5.8
2017-01-09	6.1	18.2	0.2	4.2

carry forward the same value of last upper cell if value is ZERO

```
new_df=df.fillna(method="ffill")
new_df
```

day	MinTemp	MaxTemp	Rainfall	Evaporation
2017-01-01	19.0	24.3	NaN	3.4
2017-01-04	17.0	24.3	3.6	4.4
2017-01-05	13.7	24.3	-1.0	5.8
2017-01-06	15.0	15.5	39.8	7.2
2017-01-07	15.0	16.1	-1.0	5.6
2017-01-08	15.0	16.9	0.0	5.8
2017-01-09	6.1	18.2	0.2	4.2

but we can see upper element of trainfall is still NAN,so use bfill(backward fill,i.e.,lower cell value)

```
new_df=df.fillna(method="bfill")
new_df
```

day	MinTemp	MaxTemp	Rainfall	Evaporation
2017-01-01	19.0	24.3	3.6	3.4
2017-01-04	17.0	15.5	3.6	4.4
2017-01-05	13.7	15.5	-1.0	5.8
2017-01-06	15.0	15.5	39.8	7.2
2017-01-07	6.1	16.1	-1.0	5.6
2017-01-08	6.1	16.9	0.0	5.8
2017-01-09	6.1	18.2	0.2	4.2

```
new_df
```

day	MinTemp	MaxTemp	Rainfall	Evaporation
2017-01-01	19.0	24.3	3.6	3.4

```
2017-01-04     17.0     15.5       3.6          4.4
2017-01-05     13.7     15.5      -1.0          5.8
2017-01-06     15.0     15.5      39.8          7.2
2017-01-07      6.1     16.1      -1.0          5.6
2017-01-08      6.1     16.9       0.0          5.8
2017-01-09      6.1     18.2       0.2          4.2
```

now it will copy the value horizontally as axis=columns

```
new_df=df.fillna(method="bfill",axis="columns")
new_df
```

```
             MinTemp  MaxTemp  Rainfall  Evaporation
day
2017-01-01     19.0     24.3       3.4          3.4
2017-01-04     17.0      3.6       3.6          4.4
2017-01-05     13.7     -1.0      -1.0          5.8
2017-01-06     15.0     15.5      39.8          7.2
2017-01-07     16.1     16.1      -1.0          5.6
2017-01-08     16.9     16.9       0.0          5.8
2017-01-09      6.1     18.2       0.2          4.2
```

```
df
```

```
             MinTemp  MaxTemp  Rainfall  Evaporation
day
2017-01-01     19.0     24.3       NaN          3.4
2017-01-04     17.0      NaN       3.6          4.4
2017-01-05     13.7      NaN      -1.0          5.8
2017-01-06     15.0     15.5      39.8          7.2
2017-01-07      NaN     16.1      -1.0          5.6
2017-01-08      NaN     16.9       0.0          5.8
2017-01-09      6.1     18.2       0.2          4.2
```

```
new_df=df.fillna(method="ffill",limit=1)
new_df
```

```
             MinTemp  MaxTemp  Rainfall  Evaporation
day
2017-01-01     19.0     24.3       NaN          3.4
2017-01-04     17.0     24.3       3.6          4.4
2017-01-05     13.7      NaN      -1.0          5.8
2017-01-06     15.0     15.5      39.8          7.2
2017-01-07     15.0     16.1      -1.0          5.6
2017-01-08      NaN     16.9       0.0          5.8
2017-01-09      6.1     18.2       0.2          4.2
```

```
new_df=df.interpolate()
new_df
```

```
              MinTemp    MaxTemp  Rainfall  Evaporation
day
2017-01-01  19.000000  24.300000       NaN          3.4
2017-01-04  17.000000  21.366667       3.6          4.4
2017-01-05  13.700000  18.433333      -1.0          5.8
2017-01-06  15.000000  15.500000      39.8          7.2
2017-01-07  12.033333  16.100000      -1.0          5.6
2017-01-08   9.066667  16.900000       0.0          5.8
2017-01-09   6.100000  18.200000       0.2          4.2
```

The value in place of copied 13.3 had a better guess now,i.e.,mid value that is 12.033 and 9.066

```
new_df=df.interpolate(method="time")
new_df
```

```
              MinTemp  MaxTemp  Rainfall  Evaporation
day
2017-01-01  19.000000    24.30       NaN          3.4
2017-01-04  17.000000    19.02       3.6          4.4
2017-01-05  13.700000    17.26      -1.0          5.8
2017-01-06  15.000000    15.50      39.8          7.2
2017-01-07  12.033333    16.10      -1.0          5.6
2017-01-08   9.066667    16.90       0.0          5.8
2017-01-09   6.100000    18.20       0.2          4.2
```

above should give a value of minTemp of 1/4/17 near 13.7 to be more accurate replacing actual value of 17,but its not working as of now

Dropped all rows with NA data,etc.

```
new_df=df.dropna()
new_df
```

```
            MinTemp  MaxTemp  Rainfall  Evaporation
day
2017-01-06     15.0     15.5      39.8          7.2
2017-01-09      6.1     18.2       0.2          4.2
```

```
new_df=df
new_df
```

```
            MinTemp  MaxTemp  Rainfall  Evaporation
day
2017-01-01     19.0     24.3       NaN          3.4
2017-01-04     17.0      NaN       3.6          4.4
2017-01-05     13.7      NaN      -1.0          5.8
2017-01-06     15.0     15.5      39.8          7.2
2017-01-07      NaN     16.1      -1.0          5.6
2017-01-08      NaN     16.9       0.0          5.8
2017-01-09      6.1     18.2       0.2          4.2
```

it will drop only row where each columns are NA

```
new_df=df.dropna(how="all")
new_df
```

```
            MinTemp  MaxTemp  Rainfall  Evaporation
day
2017-01-01     19.0     24.3       NaN          3.4
2017-01-04     17.0      NaN       3.6          4.4
2017-01-05     13.7      NaN      -1.0          5.8
2017-01-06     15.0     15.5      39.8          7.2
2017-01-07      NaN     16.1      -1.0          5.6
2017-01-08      NaN     16.9       0.0          5.8
2017-01-09      6.1     18.2       0.2          4.2
```

the below means that if more than one NA value then drop it

```
new_df=df.dropna(thresh=1)
new_df
```

```
            MinTemp  MaxTemp  Rainfall  Evaporation
day
2017-01-01     19.0     24.3       NaN          3.4
2017-01-04     17.0      NaN       3.6          4.4
2017-01-05     13.7      NaN      -1.0          5.8
2017-01-06     15.0     15.5      39.8          7.2
2017-01-07      NaN     16.1      -1.0          5.6
2017-01-08      NaN     16.9       0.0          5.8
2017-01-09      6.1     18.2       0.2          4.2
```

```
dt=pd.date_range("01-01-2017","01-09-2017")
idx=pd.DatetimeIndex(dt)
df=df.reindex(idx)
df
```

```
            MinTemp  MaxTemp  Rainfall  Evaporation
2017-01-01     19.0     24.3       NaN          3.4
2017-01-02      NaN      NaN       NaN          NaN
2017-01-03      NaN      NaN       NaN          NaN
2017-01-04     17.0      NaN       3.6          4.4
2017-01-05     13.7      NaN      -1.0          5.8
2017-01-06     15.0     15.5      39.8          7.2
2017-01-07      NaN     16.1      -1.0          5.6
2017-01-08      NaN     16.9       0.0          5.8
2017-01-09      6.1     18.2       0.2          4.2
```

```
df
```

```
            MinTemp  MaxTemp  Rainfall  Evaporation
2017-01-01     19.0     24.3       NaN          3.4
2017-01-02      NaN      NaN       NaN          NaN
```

```
2017-01-03      NaN      NaN       NaN          NaN
2017-01-04     17.0      NaN       3.6          4.4
2017-01-05     13.7      NaN      -1.0          5.8
2017-01-06     15.0     15.5      39.8          7.2
2017-01-07      NaN     16.1      -1.0          5.6
2017-01-08      NaN     16.9       0.0          5.8
2017-01-09      6.1     18.2       0.2          4.2
```

lets convert all -1 to NaN

```
import pandas as pd
import numpy as np

new_df = df.replace(-1, np.NaN)
new_df
```

```
            MinTemp  MaxTemp  Rainfall  Evaporation
2017-01-01     19.0     24.3       NaN          3.4
2017-01-02      NaN      NaN       NaN          NaN
2017-01-03      NaN      NaN       NaN          NaN
2017-01-04     17.0      NaN       3.6          4.4
2017-01-05     13.7      NaN       NaN          5.8
2017-01-06     15.0     15.5      39.8          7.2
2017-01-07      NaN     16.1       NaN          5.6
2017-01-08      NaN     16.9       0.0          5.8
2017-01-09      6.1     18.2       0.2          4.2
```

```
import pandas as pd
df=pd.read_excel(r"C:\Users\Ayush\Downloads\weatherfinal.xlsx")
df
```

```
   MinTemp        MaxTemp  Rainfall  Evaporation
0      8.0           24.3       NaN          3.4
1     14.0  Not available       3.6          4.4
2     13.7           23.4      -1.0          5.8
3     13.3           15.5      39.8          7.2
4      7.6           16.1      -1.0         -2.0
5      6.2           16.9       0.0          5.8
6      6.1           18.2       0.2          4.2
```

-1 and -2 both replaced by NaN

```
new_df = df.replace([-1,-2], np.NaN)
new_df
```

```
   MinTemp        MaxTemp  Rainfall  Evaporation
0      8.0           24.3       NaN          3.4
1     14.0  Not available       3.6          4.4
2     13.7           23.4       NaN          5.8
3     13.3           15.5      39.8          7.2
```

```
4      7.6           16.1       NaN          NaN
5      6.2           16.9       0.0          5.8
6      6.1           18.2       0.2          4.2
```

```
df
```

```
   MinTemp        MaxTemp  Rainfall  Evaporation
0      8.0           24.3       NaN          3.4
1     14.0  Not available       3.6          4.4
2     13.7           23.4      -1.0          5.8
3     13.3           15.5      39.8          7.2
4      7.6           16.1      -1.0         -2.0
5      6.2           16.9       0.0          5.8
6      6.1           18.2       0.2          4.2
```

```
new_df = df.replace({
    "MaxTemp":"Not available",
    "Rainfall":-1,
},np.NaN)
new_df
```

```
   MinTemp  MaxTemp  Rainfall  Evaporation
0      8.0     24.3       NaN          3.4
1     14.0      NaN       3.6          4.4
2     13.7     23.4       NaN          5.8
3     13.3     15.5      39.8          7.2
4      7.6     16.1       NaN         -2.0
5      6.2     16.9       0.0          5.8
6      6.1     18.2       0.2          4.2
```

```
df
```

```
   MinTemp        MaxTemp  Rainfall  Evaporation
0      8.0           24.3       NaN          3.4
1     14.0  Not available       3.6          4.4
2     13.7           23.4      -1.0          5.8
3     13.3           15.5      39.8          7.2
4      7.6           16.1      -1.0         -2.0
5      6.2           16.9       0.0          5.8
6      6.1           18.2       0.2          4.2
```

```
new_df=df.replace({
    "Not available":np.NaN,
   -1:"no rain"
})
new_df
```

```
   MinTemp  MaxTemp Rainfall  Evaporation
0      8.0     24.3      NaN          3.4
1     14.0      NaN      3.6          4.4
2     13.7     23.4  no rain          5.8
```

```
3     13.3     15.5     39.8          7.2
4      7.6     16.1  no rain         -2.0
5      6.2     16.9      0.0          5.8
6      6.1     18.2      0.2          4.2
```

```
import pandas as pd
df=pd.read_excel(r"C:\Users\Ayush\Downloads\weatherfinal.xlsx")
df
```

```
   MinTemp         MaxTemp Rainfall  Evaporation     event
0      8.0            24.3      NaN        3.400     sunny
1     14.0  Not available   3.6 MM        4.400  snowfall
2     13.7         23.4 C       -1        5.800      rain
3     13.3            15.5     39.8        0.072       fog
4      7.6         16.1 F       -1       -2.000    cloudy
5      6.2            16.9        0        5.800      mist
6      6.1            18.2      0.2        4.200       dew
```

now we want to remove units such as MM,C,%,etc. but this will also remove events as that is also alphabets

```
new_df = df.replace('[a-zA-Z]', '', regex=True)
new_df
```

```
   MinTemp MaxTemp Rainfall  Evaporation event
0      8.0    24.3      NaN        3.400
1     14.0              3.6        4.400
2     13.7   23.4         -1        5.800
3     13.3    15.5     39.8        0.072
4      7.6   16.1         -1       -2.000
5      6.2    16.9         0        5.800
6      6.1    18.2      0.2        4.200
```

```
new_df = df.replace({
    'MaxTemp':'[a-zA-Z]',
    "Rainfall":'[a-zA-Z]'
},'',regex=True)
new_df
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call
last)
Cell In[77], line 5
      1 new_df = df.replace({
      2     'MaxTemp':'[a-zA-Z]',
      3     "Rainfall":'[a-zA-Z]'
      4 },'',regex=True)
----> 5 new_df(inplace=True)
```

```
TypeError: 'DataFrame' object is not callable
```

```
new_df=new_df.replace( '',np.NaN, regex=True)
new_df
```

```
   MinTemp        MaxTemp Rainfall  Evaporation     event
0      8.0           24.3      NaN        3.400     sunny
1     14.0  Not available   3.6 MM        4.400  snowfall
2     13.7         23.4 C       -1        5.800      rain
3     13.3           15.5     39.8        0.072       fog
4      7.6         16.1 F       -1       -2.000    cloudy
5      6.2           16.9        0        5.800      mist
6      6.1           18.2      0.2        4.200       dew
```

Replacing a list of values with another list

```
df=pd.DataFrame({
    'score':
['Exceptional','Average',"Good",'Poor','Average',"Exceptional"],
    'Student':["Rob","Maya","Parthiv","Tom","Julian","Erica"]
})
df
```

```
         score  Student
0  Exceptional      Rob
1      Average     Maya
2         Good  Parthiv
3         Poor      Tom
4      Average   Julian
5  Exceptional    Erica
```

```
new_df=df.replace(['Exceptional','Average',"Good",'Poor'],[1,2,3,4])
new_df
```

```
   score  Student
0      1      Rob
1      2     Maya
2      3  Parthiv
3      4      Tom
4      2   Julian
5      1    Erica
```

```
import pandas as pd
df=pd.read_excel(r"C:\Users\Ayush\Downloads\Book3.xlsx")
df
```

```
          day     city  temperature  windspeed   event
0  2017-01-01  newyork           32          6    rain
1  2017-01-02  newyork           36          7   sunny
2  2017-01-03  newyork           28         12    snow
```

```
3  2017-01-04  newyork           33           7   sunny
4  2017-01-01   mumbai           90           5   sunny
5  2017-01-02   mumbai           85          12     fog
6  2017-01-03   mumbai           87          15     fog
7  2017-01-04   mumbai           92           5    rain
8  2017-01-01    paris           45          20   sunny
9  2017-01-02    paris           50          13  cloudy
10 2017-01-03    paris           54           8  cloudy
11 2017-01-04    paris           42          10  cloudy
```

```
g =df.groupby('city')
g
```

```
<pandas.core.groupby.generic.DataFrameGroupBy object at
0x00000265A988DA20>
```

```
for city,city_df in g:
    print(city)
    print(city_df)
```

```
mumbai
         day    city  temperature  windspeed  event
4 2017-01-01  mumbai           90          5  sunny
5 2017-01-02  mumbai           85         12    fog
6 2017-01-03  mumbai           87         15    fog
7 2017-01-04  mumbai           92          5   rain
newyork
         day     city  temperature  windspeed  event
0 2017-01-01  newyork           32          6   rain
1 2017-01-02  newyork           36          7  sunny
2 2017-01-03  newyork           28         12   snow
3 2017-01-04  newyork           33          7  sunny
paris
          day   city  temperature  windspeed   event
8  2017-01-01  paris           45         20   sunny
9  2017-01-02  paris           50         13  cloudy
10 2017-01-03  paris           54          8  cloudy
11 2017-01-04  paris           42         10  cloudy
```

```
g.get_group('mumbai')
```

```
         day    city  temperature  windspeed  event
4 2017-01-01  mumbai           90          5  sunny
5 2017-01-02  mumbai           85         12    fog
6 2017-01-03  mumbai           87         15    fog
7 2017-01-04  mumbai           92          5   rain
```

```
g.mean()
```

```
C:\Users\Ayush\AppData\Local\Temp\ipykernel_16584\2978112660.py:1:
FutureWarning: The default value of numeric_only in
```

```
DataFrameGroupBy.mean is deprecated. In a future version, numeric_only
will default to False. Either specify numeric_only or select only
columns which should be valid for the function.
  g.mean()
```

city	temperature	windspeed
mumbai	88.50	9.25
newyork	32.25	8.00
paris	47.75	12.75

```
g.max()
```

city	day	temperature	windspeed	event
mumbai	2017-01-04	92	15	sunny
newyork	2017-01-04	36	12	sunny
paris	2017-01-04	54	20	sunny

the above three steps was SPLIT,APPLY and COMBINE,as we splitted in citied ,applied operation for max. on each and then combined the results from three in one df

```
g.describe()
```

	temperature							
	count	mean	std	min	25%	50%	75%	max
city								
mumbai	4.0	88.50	3.109126	85.0	86.50	88.5	90.50	92.0
newyork	4.0	32.25	3.304038	28.0	31.00	32.5	33.75	36.0
paris	4.0	47.75	5.315073	42.0	44.25	47.5	51.00	54.0

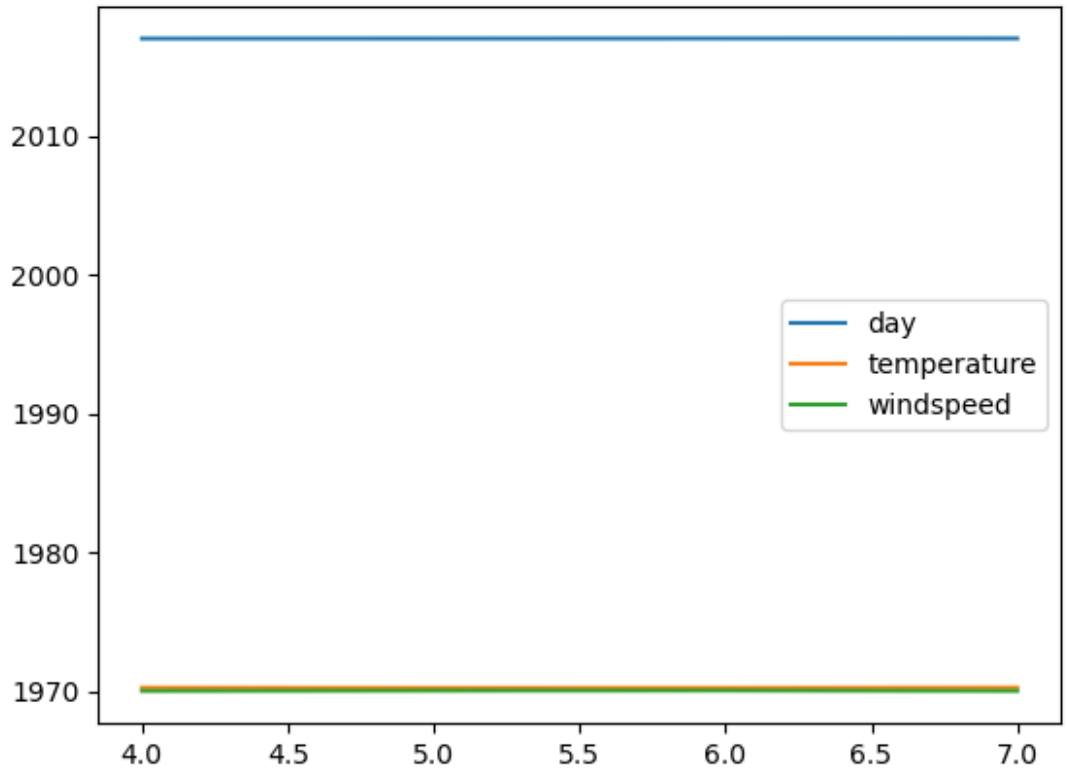
	windspeed							
	count	mean	std	min	25%	50%	75%	max
city								
mumbai	4.0	9.25	5.057997	5.0	5.00	8.5	12.75	15.0
newyork	4.0	8.00	2.708013	6.0	6.75	7.0	8.25	12.0
paris	4.0	12.75	5.251984	8.0	9.50	11.5	14.75	20.0

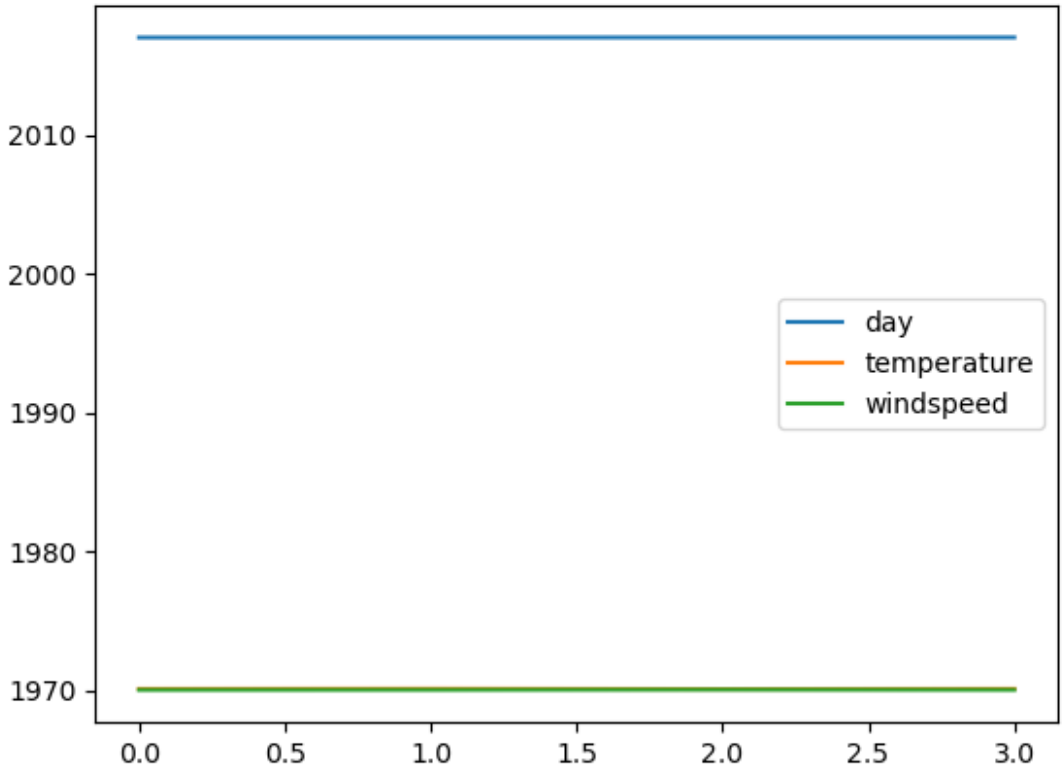
```
g
```

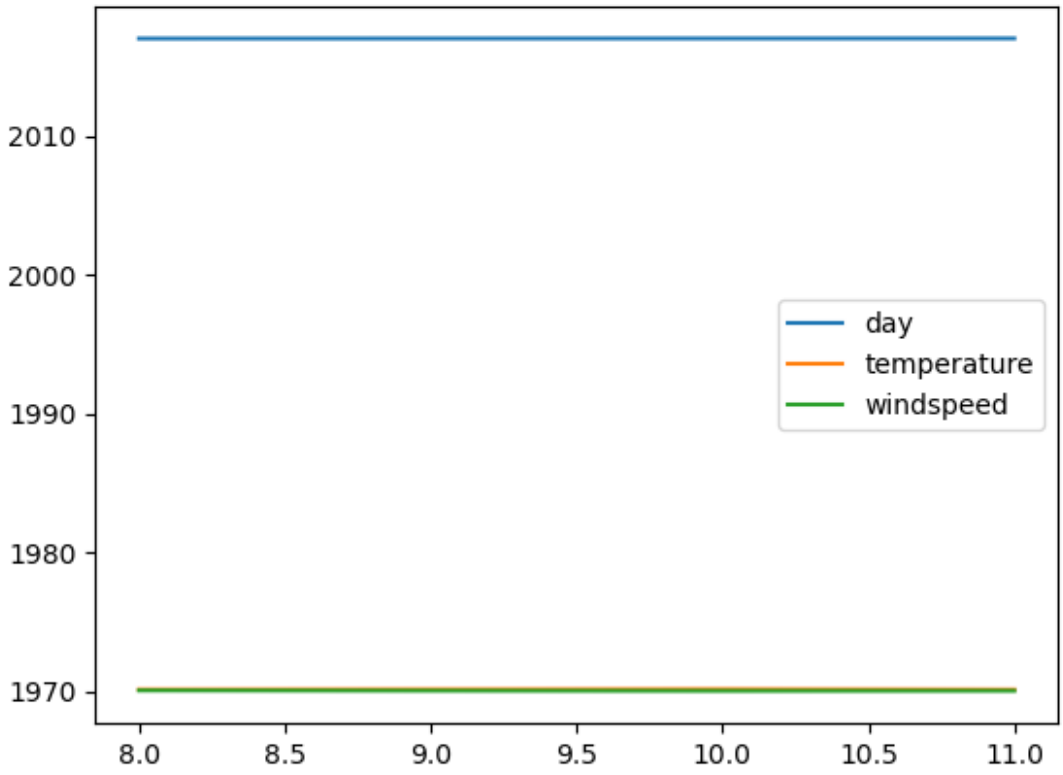
```
<pandas.core.groupby.generic.DataFrameGroupBy object at
0x00000265A988DA20>
```

```
%matplotlib inline
g.plot()
```

```
city
mumbai     Axes(0.125,0.11;0.775x0.77)
newyork    Axes(0.125,0.11;0.775x0.77)
paris      Axes(0.125,0.11;0.775x0.77)
dtype: object
```







the above plotting is wrong but code is correct

```
import pandas as pd

india=pd.DataFrame({
    "city":["mumbai","delhi","bangalore"],
    "temperature":[32,45,30],
    "humidity":[80,60,78]
})
india
```

	city	temperature	humidity
0	mumbai	32	80
1	delhi	45	60
2	bangalore	30	78

```
us=pd.DataFrame({
    "city":["new york","chicago","orlando"],
    "temperature":[21,14,35],
    "humidity":[68,65,75]
})
us
```

	city	temperature	humidity
0	new york	21	68
1	chicago	14	65
2	orlando	35	75

```
df = pd.concat([india,us])
df
```

	city	temperature	humidity
0	mumbai	32	80
1	delhi	45	60
2	bangalore	30	78
0	new york	21	68
1	chicago	14	65
2	orlando	35	75

```
df = pd.concat([india,us],ignore_index=True)
df
```

	city	temperature	humidity
0	mumbai	32	80
1	delhi	45	60
2	bangalore	30	78
3	new york	21	68
4	chicago	14	65
5	orlando	35	75

```
df = pd.concat([india,us], keys=["india","us"])
df
```

```
              city  temperature  humidity
india 0     mumbai           32        80
      1      delhi           45        60
      2  bangalore           30        78
us    0   new york           21        68
      1    chicago           14        65
      2    orlando           35        75
```

```
df.loc["india"]
```

```
        city  temperature  humidity
0     mumbai           32        80
1      delhi           45        60
2  bangalore           30        78
```

```
df.loc["us"]
```

```
       city  temperature  humidity
0  new york           21        68
1   chicago           14        65
2   orlando           35        75
```

```
temperature_df=pd.DataFrame({
    "city":["mumbai","delhi","bangalore"],
    "temperature":[32,45,30],

})
temperature_df
```

```
        city  temperature
0     mumbai           32
1      delhi           45
2  bangalore           30
```

```
windspeed_df=pd.DataFrame({
    "city":["mumbai","delhi","bangalore"],
    "windspeed":[7,12,9],

})
windspeed_df
```

```
        city  windspeed
0     mumbai          7
1      delhi         12
2  bangalore          9
```

```
df=pd.concat([temperature_df,windspeed_df])
df
```

```
        city  temperature  windspeed
0     mumbai         32.0        NaN
```

```
1      delhi         45.0         NaN
2  bangalore         30.0         NaN
0     mumbai          NaN         7.0
1      delhi          NaN        12.0
2  bangalore          NaN         9.0
```

```
df=pd.concat([temperature_df,windspeed_df],axis=1)
df
```

```
        city  temperature       city  windspeed
0     mumbai           32     mumbai          7
1      delhi           45      delhi         12
2  bangalore           30  bangalore          9
```

since axis=1,it will append it as column and not as rows

```
windspeed_df=pd.DataFrame({
    "city":["delhi","mumbai"],
    "windspeed":[7,12],

})
windspeed_df
```

```
     city  windspeed
0   delhi          7
1  mumbai         12
```

```
df=pd.concat([temperature_df,windspeed_df],axis=1)
df
```

```
        city  temperature    city  windspeed
0     mumbai           32   delhi        7.0
1      delhi           45  mumbai       12.0
2  bangalore           30     NaN        NaN
```

```
windspeed_df=pd.DataFrame({
    "city":["delhi","mumbai"],
    "windspeed":[7,12],

},index=[1,0])
windspeed_df
```

```
     city  windspeed
1   delhi          7
0  mumbai         12
```

```
temperature_df=pd.DataFrame({
    "city":["mumbai","delhi","bangalore"],
    "temperature":[32,45,30],
```

```
},index=[0,1,2])
temperature_df
```

```
        city  temperature
0     mumbai           32
1      delhi           45
2  bangalore           30
```

```
df=pd.concat([temperature_df,windspeed_df],axis=1)
df
```

```
        city  temperature    city  windspeed
0     mumbai           32  mumbai       12.0
1      delhi           45   delhi        7.0
2  bangalore           30     NaN        NaN
```

```
temperature_df
```

```
        city  temperature
0     mumbai           32
1      delhi           45
2  bangalore           30
```

```
s=pd.Series(["humid","dry","rain"],name="event")
s
```

```
0    humid
1      dry
2     rain
Name: event, dtype: object
```

```
df=pd.concat([temperature_df,s],axis=1)
df
```

```
        city  temperature  event
0     mumbai           32  humid
1      delhi           45    dry
2  bangalore           30   rain
```

```
def addition(a,b=0):
    return a+b;

t=addition(2,3)

print(t+3)
```

```
8
```

```
import calendar as c

dir(calendar)
```

```
['Calendar',
 'EPOCH',
 'FRIDAY',
 'February',
 'HTMLCalendar',
 'IllegalMonthError',
 'IllegalWeekdayError',
 'January',
 'LocaleHTMLCalendar',
 'LocaleTextCalendar',
 'MONDAY',
 'SATURDAY',
 'SUNDAY',
 'THURSDAY',
 'TUESDAY',
 'TextCalendar',
 'WEDNESDAY',
 '_EPOCH_ORD',
 '__all__',
 '__builtins__',
 '__cached__',
 '__doc__',
 '__file__',
 '__loader__',
 '__name__',
 '__package__',
 '__spec__',
 '_colwidth',
 '_locale',
 '_localized_day',
 '_localized_month',
 '_monthlen',
 '_nextmonth',
 '_prevmonth',
 '_spacing',
 'c',
 'calendar',
 'datetime',
```

```
 'day_abbr',
 'day_name',
 'different_locale',
 'error',
 'firstweekday',
 'format',
 'formatstring',
 'isleap',
 'leapdays',
 'main',
 'mdays',
 'month',
 'month_abbr',
 'month_name',
 'monthcalendar',
 'monthrange',
 'prcal',
 'prmonth',
 'prweek',
 'repeat',
 'setfirstweekday',
 'sys',
 'timegm',
 'week',
 'weekday',
 'weekheader']
```

```
print(c.month(2023,10))
```

```
    October 2023
Mo Tu We Th Fr Sa Su
                   1
 2  3  4  5  6  7  8
 9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29
30 31
```

```
c.isleap(2023)
```

```
False
```

```
dict={"ayush":3,"shaurya":5,"jha":2}
```

```
print(dict["ayush"])
dict["ayush"]=-9
print(dict["ayush"])
```

```
3
-9
```

```
t=("ayush",10123,"New York")
print(t[0]+t[2])
print(t[0:])
```

```
ayushNew York
('ayush', 10123, 'New York')
```

```
t[0]="king ayush"
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[23], line 1
----> 1 t[0]="king ayush"

TypeError: 'tuple' object does not support item assignment
```

```
list=[[1,23,3],2,3]

print(list)
```

```
[[1, 23, 3], 2, 3]
```

```
list[0]
```

```
[1, 23, 3]
```

```
list[0][1]
```

```
23
```

```
dir(numpy)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[30], line 1
----> 1 dir(numpy)

NameError: name 'numpy' is not defined
```

```
import numpy
dir(numpy)
```

```
['ALLOW_THREADS',
 'AxisError',
 'BUFSIZE',
 'CLIP',
 'ComplexWarning',
 'DataSource',
 'ERR_CALL',
```

```
 'ERR_DEFAULT',
 'ERR_IGNORE',
 'ERR_LOG',
 'ERR_PRINT',
 'ERR_RAISE',
 'ERR_WARN',
 'FLOATING_POINT_SUPPORT',
 'FPE_DIVIDEBYZERO',
 'FPE_INVALID',
 'FPE_OVERFLOW',
 'FPE_UNDERFLOW',
 'False_',
 'Inf',
 'Infinity',
 'MAXDIMS',
 'MAY_SHARE_BOUNDS',
 'MAY_SHARE_EXACT',
 'ModuleDeprecationWarning',
 'NAN',
 'NINF',
 'NZERO',
 'NaN',
 'PINF',
 'PZERO',
 'RAISE',
 'RankWarning',
 'SHIFT_DIVIDEBYZERO',
 'SHIFT_INVALID',
 'SHIFT_OVERFLOW',
 'SHIFT_UNDERFLOW',
 'ScalarType',
 'Tester',
 'TooHardError',
 'True_',
 'UFUNC_BUFSIZE_DEFAULT',
 'UFUNC_PYVALS_NAME',
 'VisibleDeprecationWarning',
 'WRAP',
 '_CopyMode',
 '_NoValue',
 '_UFUNC_API',
 '__NUMPY_SETUP__',
 '__all__',
 '__builtins__',
 '__cached__',
 '__config__',
 '__deprecated_attrs__',
 '__dir__',
 '__doc__',
```

```
'__expired_functions__',
'__file__',
'__getattr__',
'__git_version__',
'__loader__',
'__mkl_version__',
'__name__',
'__package__',
'__path__',
'__spec__',
'__version__',
'_add_newdoc_ufunc',
'_distributor_init',
'_financial_names',
'_globals',
'_mat',
'_pyinstaller_hooks_dir',
'_pytesttester',
'_version',
'abs',
'absolute',
'add',
'add_docstring',
'add_newdoc',
'add_newdoc_ufunc',
'all',
'allclose',
'alltrue',
'amax',
'amin',
'angle',
'any',
'append',
'apply_along_axis',
'apply_over_axes',
'arange',
'arccos',
'arccosh',
'arcsin',
'arcsinh',
'arctan',
'arctan2',
'arctanh',
'argmax',
'argmin',
'argpartition',
'argsort',
'argwhere',
'around',
```

```
'array',
'array2string',
'array_equal',
'array_equiv',
'array_repr',
'array_split',
'array_str',
'asanyarray',
'asarray',
'asarray_chkfinite',
'ascontiguousarray',
'asfarray',
'asfortranarray',
'asmatrix',
'atleast_1d',
'atleast_2d',
'atleast_3d',
'average',
'bartlett',
'base_repr',
'binary_repr',
'bincount',
'bitwise_and',
'bitwise_not',
'bitwise_or',
'bitwise_xor',
'blackman',
'block',
'bmat',
'bool8',
'bool_',
'broadcast',
'broadcast_arrays',
'broadcast_shapes',
'broadcast_to',
'busday_count',
'busday_offset',
'busdaycalendar',
'byte',
'byte_bounds',
'bytes0',
'bytes_',
'c_',
'can_cast',
'cast',
'cbrt',
'cdouble',
'ceil',
'cfloat',
```

```
'char',
'character',
'chararray',
'choose',
'clip',
'clongdouble',
'clongfloat',
'column_stack',
'common_type',
'compare_chararrays',
'compat',
'complex128',
'complex64',
'complex_',
'complexfloating',
'compress',
'concatenate',
'conj',
'conjugate',
'convolve',
'copy',
'copysign',
'copyto',
'core',
'corrcoef',
'correlate',
'cos',
'cosh',
'count_nonzero',
'cov',
'cross',
'csingle',
'ctypeslib',
'cumprod',
'cumproduct',
'cumsum',
'datetime64',
'datetime_as_string',
'datetime_data',
'deg2rad',
'degrees',
'delete',
'deprecate',
'deprecate_with_doc',
'diag',
'diag_indices',
'diag_indices_from',
'diagflat',
'diagonal',
```

```
'diff',
'digitize',
'disp',
'divide',
'divmod',
'dot',
'double',
'dsplit',
'dstack',
'dtype',
'e',
'ediff1d',
'einsum',
'einsum_path',
'emath',
'empty',
'empty_like',
'equal',
'errstate',
'euler_gamma',
'exp',
'exp2',
'expand_dims',
'expm1',
'extract',
'eye',
'fabs',
'fastCopyAndTranspose',
'fft',
'fill_diagonal',
'find_common_type',
'finfo',
'fix',
'flatiter',
'flatnonzero',
'flexible',
'flip',
'fliplr',
'flipud',
'float16',
'float32',
'float64',
'float_',
'float_power',
'floating',
'floor',
'floor_divide',
'fmax',
'fmin',
```

```
'fmod',
'format_float_positional',
'format_float_scientific',
'format_parser',
'frexp',
'from_dlpack',
'frombuffer',
'fromfile',
'fromfunction',
'fromiter',
'frompyfunc',
'fromregex',
'fromstring',
'full',
'full_like',
'gcd',
'generic',
'genfromtxt',
'geomspace',
'get_array_wrap',
'get_include',
'get_printoptions',
'getbufsize',
'geterr',
'geterrcall',
'geterrobj',
'gradient',
'greater',
'greater_equal',
'half',
'hamming',
'hanning',
'heaviside',
'histogram',
'histogram2d',
'histogram_bin_edges',
'histogramdd',
'hsplit',
'hstack',
'hypot',
'i0',
'identity',
'iinfo',
'imag',
'in1d',
'index_exp',
'indices',
'inexact',
'inf',
```

```
'info',
'infty',
'inner',
'insert',
'int0',
'int16',
'int32',
'int64',
'int8',
'int_',
'intc',
'integer',
'interp',
'intersect1d',
'intp',
'invert',
'is_busday',
'isclose',
'iscomplex',
'iscomplexobj',
'isfinite',
'isfortran',
'isin',
'isinf',
'isnan',
'isnat',
'isneginf',
'isposinf',
'isreal',
'isrealobj',
'isscalar',
'issctype',
'issubclass_',
'issubdtype',
'issubsctype',
'iterable',
'ix_',
'kaiser',
'kron',
'lcm',
'ldexp',
'left_shift',
'less',
'less_equal',
'lexsort',
'lib',
'linalg',
'linspace',
'little_endian',
```

```
'load',
'loadtxt',
'log',
'log10',
'log1p',
'log2',
'logaddexp',
'logaddexp2',
'logical_and',
'logical_not',
'logical_or',
'logical_xor',
'logspace',
'longcomplex',
'longdouble',
'longfloat',
'longlong',
'lookfor',
'ma',
'mask_indices',
'mat',
'math',
'matmul',
'matrix',
'matrixlib',
'max',
'maximum',
'maximum_sctype',
'may_share_memory',
'mean',
'median',
'memmap',
'meshgrid',
'mgrid',
'min',
'min_scalar_type',
'minimum',
'mintypecode',
'mkl',
'mod',
'modf',
'moveaxis',
'msort',
'multiply',
'nan',
'nan_to_num',
'nanargmax',
'nanargmin',
'nancumprod',
```

```
'nancumsum',
'nanmax',
'nanmean',
'nanmedian',
'nanmin',
'nanpercentile',
'nanprod',
'nanquantile',
'nanstd',
'nansum',
'nanvar',
'nbytes',
'ndarray',
'ndenumerate',
'ndim',
'ndindex',
'nditer',
'negative',
'nested_iters',
'newaxis',
'nextafter',
'nonzero',
'not_equal',
'numarray',
'number',
'obj2sctype',
'object0',
'object_',
'ogrid',
'oldnumeric',
'ones',
'ones_like',
'os',
'outer',
'packbits',
'pad',
'partition',
'percentile',
'pi',
'piecewise',
'place',
'poly',
'poly1d',
'polyadd',
'polyder',
'polydiv',
'polyfit',
'polyint',
'polymul',
```

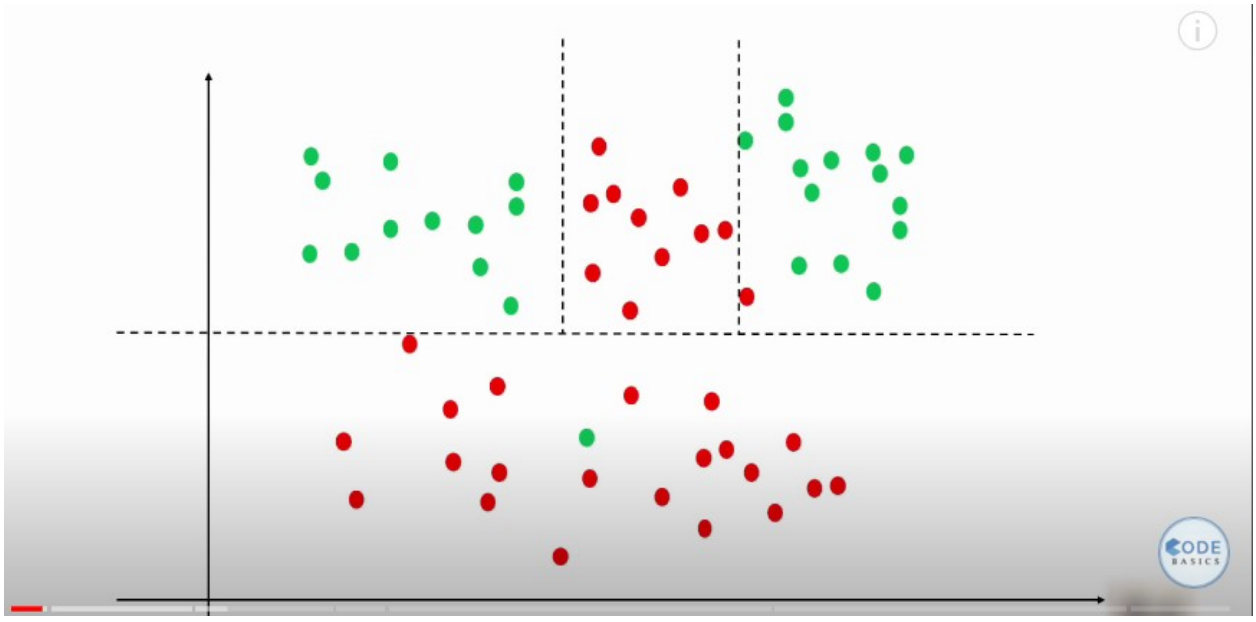
```
'polynomial',
'polysub',
'polyval',
'positive',
'power',
'printoptions',
'prod',
'product',
'promote_types',
'ptp',
'put',
'put_along_axis',
'putmask',
'quantile',
'r_',
'rad2deg',
'radians',
'random',
'ravel',
'ravel_multi_index',
'real',
'real_if_close',
'rec',
'recarray',
'recfromcsv',
'recfromtxt',
'reciprocal',
'record',
'remainder',
'repeat',
'require',
'reshape',
'resize',
'result_type',
'right_shift',
'rint',
'roll',
'rollaxis',
'roots',
'rot90',
'round',
'round_',
'row_stack',
's_',
'safe_eval',
'save',
'savetxt',
'savez',
'savez_compressed',
'sctype2char',
```

```
 'sctypeDict',
 'sctypes',
 'searchsorted',
 'select',
 'set_numeric_ops',
 'set_printoptions',
 'set_string_function',
 'setbufsize',
 'setdiff1d',
 'seterr',
 'seterrcall',
 'seterrobj',
 'setxor1d',
 'shape',
 'shares_memory',
 'short',
 'show_config',
 'sign',
 'signbit',
 'signedinteger',
 'sin',
 'sinc',
 'single',
 'singlecomplex',
 'sinh',
 'size',
 'sometrue',
 'sort',
 'sort_complex',
 'source',
 'spacing',
 'split',
 'sqrt',
 'square',
 'squeeze',
 'stack',
 'std',
 'str0',
 'str_',
 'string_',
 'subtract',
 'sum',
 'swapaxes',
 'sys',
 'take',
 'take_along_axis',
 'tan',
 'tanh',
 'tensordot',
```

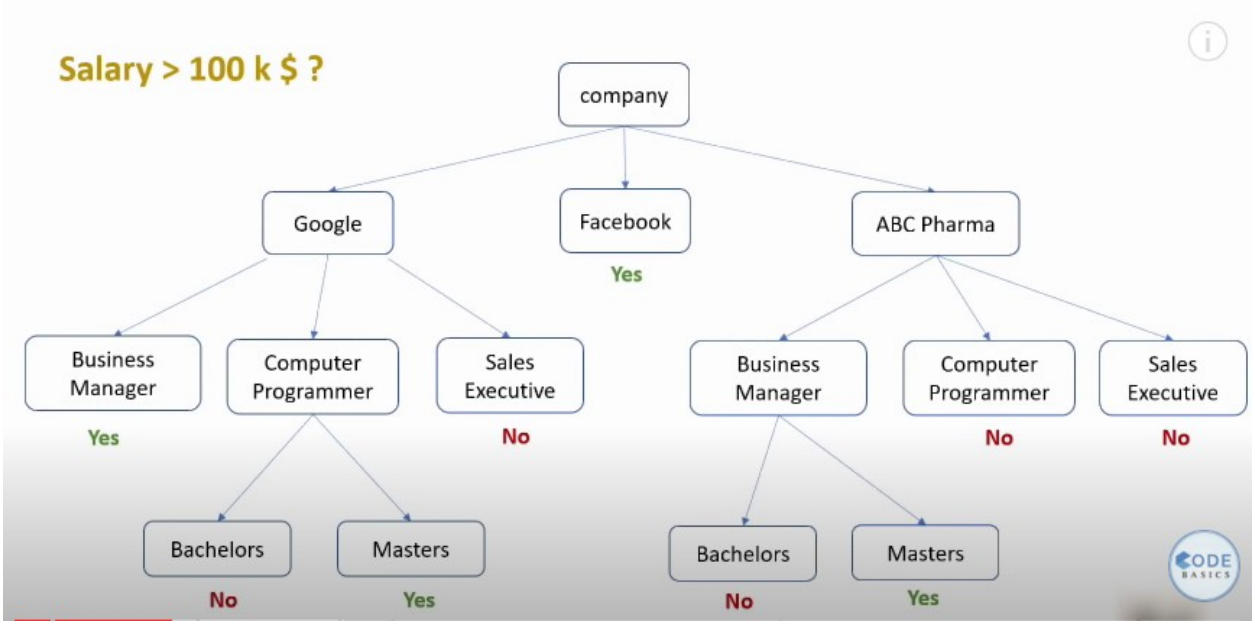
```
'test',
'testing',
'tile',
'timedelta64',
'trace',
'tracemalloc_domain',
'transpose',
'trapz',
'tri',
'tril',
'tril_indices',
'tril_indices_from',
'trim_zeros',
'triu',
'triu_indices',
'triu_indices_from',
'true_divide',
'trunc',
'typecodes',
'typename',
'ubyte',
'ufunc',
'uint',
'uint0',
'uint16',
'uint32',
'uint64',
'uint8',
'uintc',
'uintp',
'ulonglong',
'unicode_',
'union1d',
'unique',
'unpackbits',
'unravel_index',
'unsignedinteger',
'unwrap',
'use_hugepage',
'ushort',
'vander',
'var',
'vdot',
'vectorize',
'version',
'void',
'void0',
'vsplit',
'vstack',
```

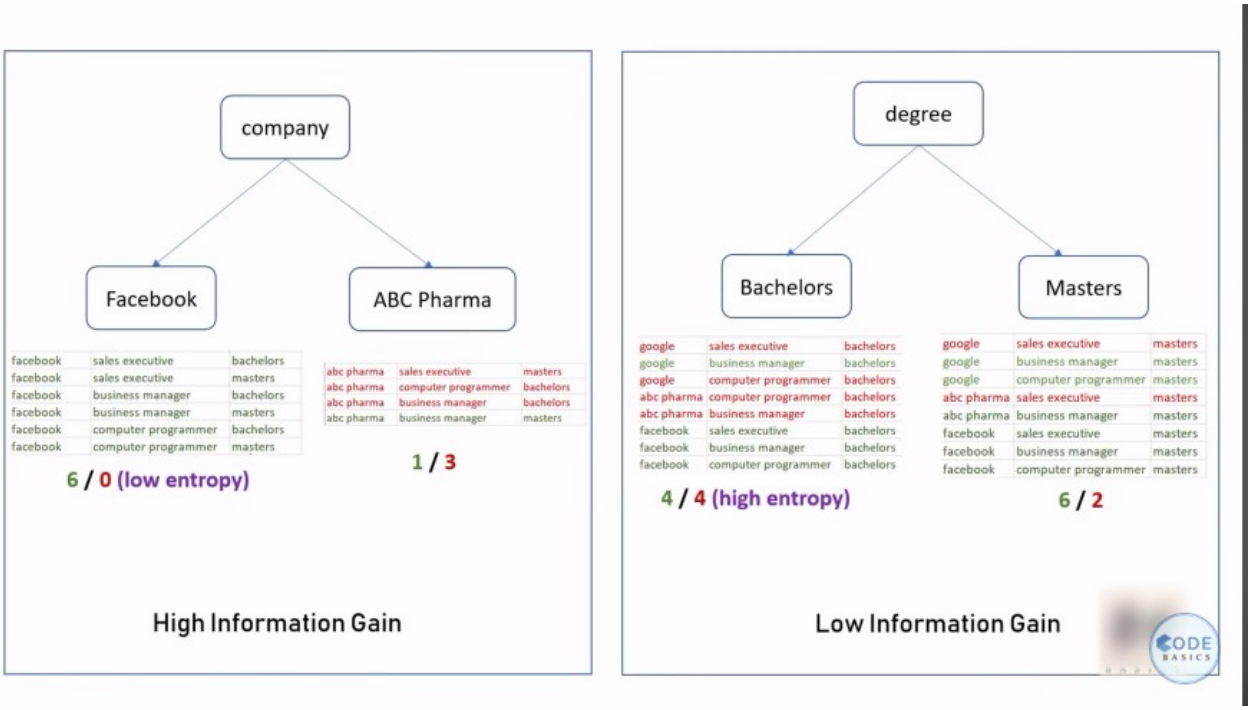
```
 'warnings',
 'where',
 'who',
 'zeros',
 'zeros_like']
```

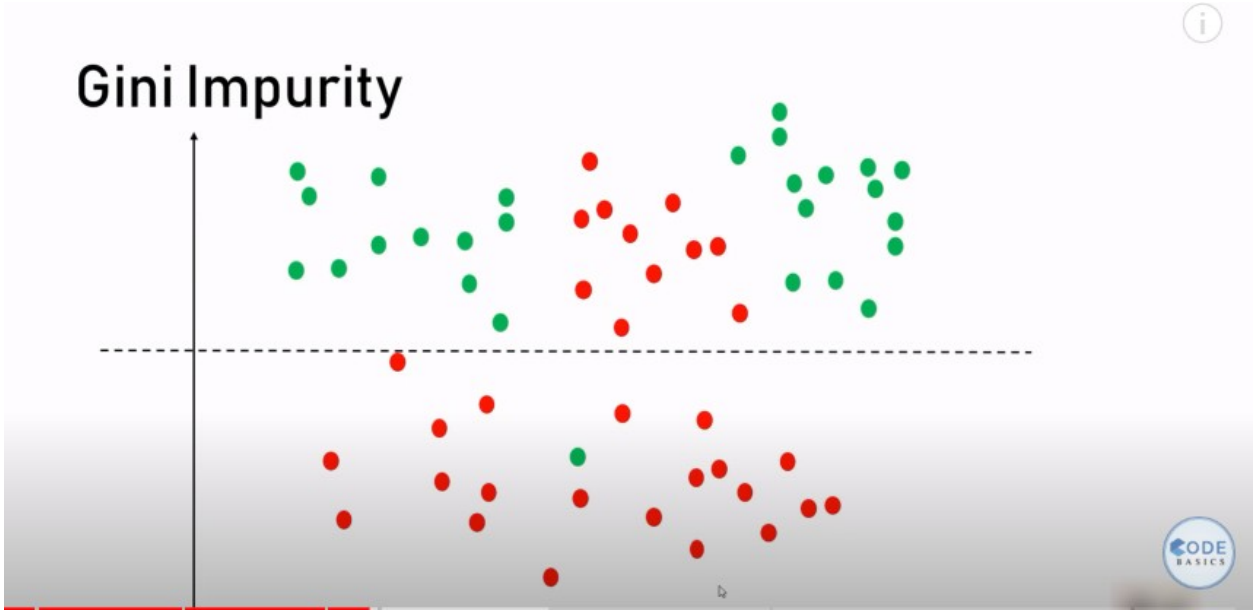
DECISION TREE



Company	Job	Degree	Salary_more_then_100k
google	sales executive	bachelors	0
google	sales executive	masters	0
google	business manager	bachelors	1
google	business manager	masters	1
google	computer programmer	bachelors	0
google	computer programmer	masters	1
abc pharma	sales executive	masters	0
abc pharma	computer programmer	bachelors	0
abc pharma	business manager	bachelors	0
abc pharma	business manager	masters	1
facebook	sales executive	bachelors	1
facebook	sales executive	masters	1
facebook	business manager	bachelors	1
facebook	business manager	masters	1


Salary > 100 k $?
company
Google
Facebook
ABC Pharma
Yes
Business Manager
Computer Programmer
Sales Executive
Business Manager
Computer Programmer
Sales Executive
Yes
No
No
No
Bachelors
Masters
Bachelors
Masters
No
Yes
No
Yes


company
Facebook
ABC Pharma
facebook sales executive bachelors
facebook sales executive masters
facebook business manager bachelors
facebook business manager masters
facebook computer programmer bachelors
facebook computer programmer masters
6 / 0 (low entropy)
abc pharma sales executive masters
abc pharma computer programmer bachelors
abc pharma business manager bachelors
abc pharma business manager masters
1 / 3
High Information Gain
degree
Bachelors
Masters
google sales executive bachelors
google business manager bachelors
google computer programmer bachelors
abc pharma computer programmer bachelors
abc pharma business manager bachelors
facebook sales executive bachelors
facebook business manager bachelors
facebook computer programmer bachelors
4 / 4 (high entropy)
google sales executive masters
google business manager masters
google computer programmer masters
abc pharma sales executive masters
abc pharma business manager masters
facebook sales executive masters
facebook business manager masters
facebook computer programmer masters
6 / 2
Low Information Gain




```
import pandas as pd

df=pd.read_csv(r"C:\Users\Ayush\OneDrive\Documents\company.csv")
df.head()
```

	company	job	degree	salary
0	google	sales executive	bachelors	0
1	google	sales executive	masters	0
2	google	businessmanager	bachelors	1
3	google	businessmanager	masters	1
4	google	computer programmer	bachelors	0

```
inputs=df.drop('salary',axis='columns')
target=df.salary

target
```

```
0     0
1     0
2     1
3     1
4     0
5     1
6     0
7     0
8     0
9     1
10    1
11    1
12    1
13    1
14    1
```

```
15    1
Name: salary, dtype: int64
```

```
from sklearn.preprocessing import LabelEncoder
```

```
le_company=LabelEncoder()
le_job=LabelEncoder()
le_degree=LabelEncoder()
```

```
inputs['company_n']=le_company.fit_transform(inputs['company'])
inputs['job_n']=le_company.fit_transform(inputs['job'])
inputs['degree_n']=le_company.fit_transform(inputs['degree'])
inputs.head()
```

	company	job	degree	company_n	job_n	degree_n
0	google	sales executive	bachelors	2	2	0
1	google	sales executive	masters	2	2	1
2	google	businessmanager	bachelors	2	0	0
3	google	businessmanager	masters	2	0	1
4	google	computer programmer	bachelors	2	1	0

```
inputs_n=inputs.drop(['company','job','degree'],axis='columns')
inputs_n
```

	company_n	job_n	degree_n
0	2	2	0
1	2	2	1
2	2	0	0
3	2	0	1
4	2	1	0
5	2	1	1
6	0	2	1
7	0	1	0
8	0	0	0
9	0	0	1
10	1	2	0
11	1	2	1
12	1	0	0
13	1	0	1
14	1	1	0
15	1	1	1

```
from sklearn import tree
```

```
model=tree.DecisionTreeClassifier()
```

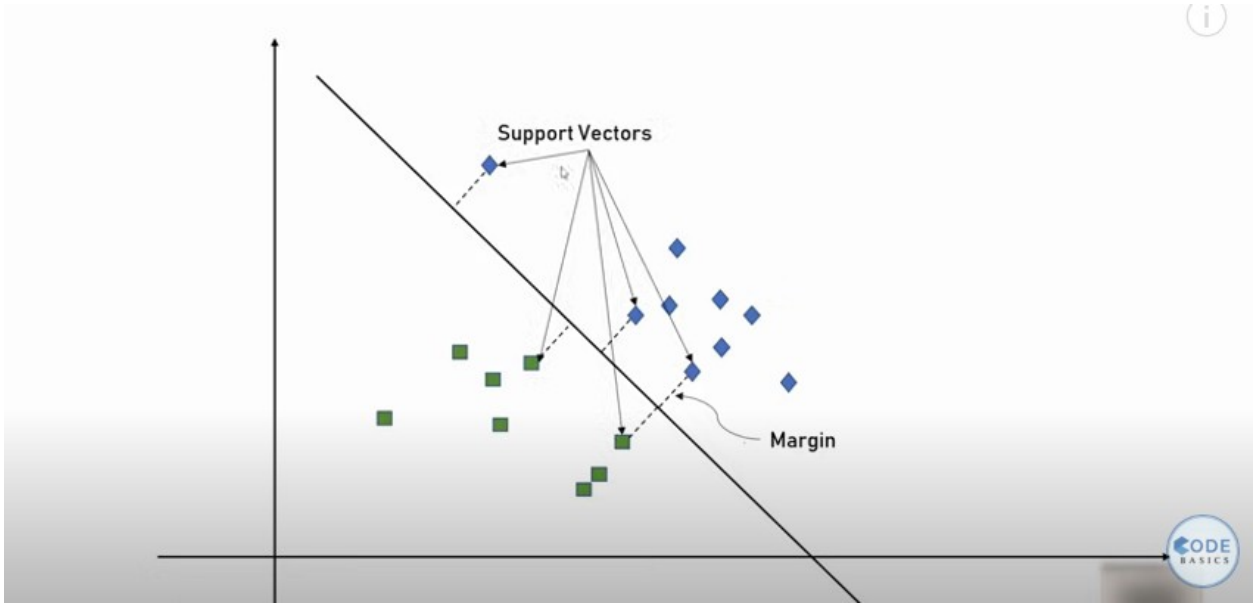
```
model.fit(inputs_n,target)
```

```
DecisionTreeClassifier()
```

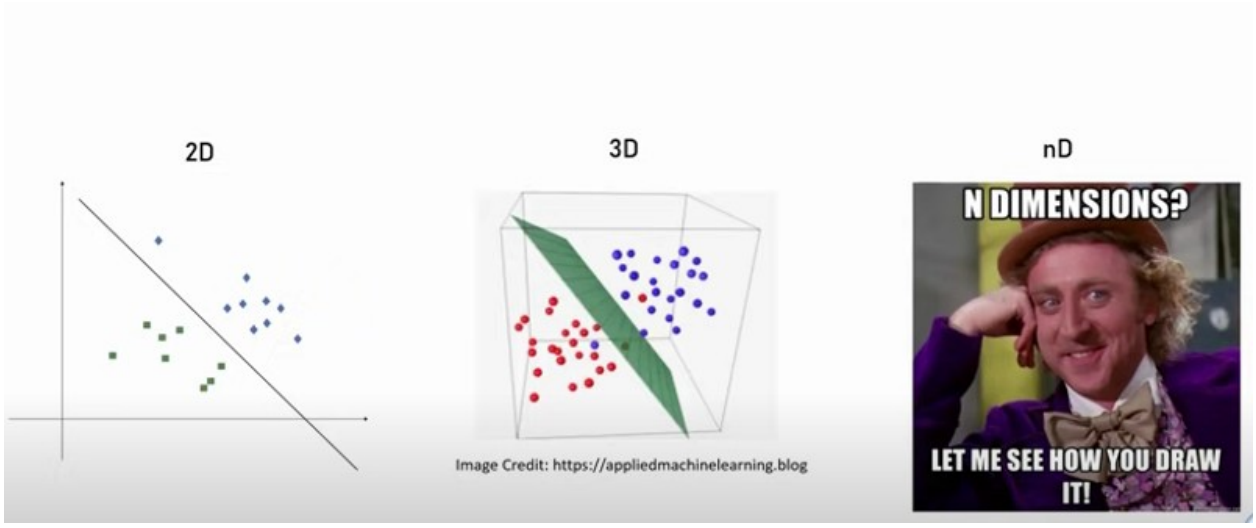
```
model.score(inputs_n,target)
```

```
1.0

model.predict([[2,2,1]])

C:\ProgramData\anaconda3\lib\site-packages\sklearn\base.py:420:
UserWarning: X does not have valid feature names, but
DecisionTreeClassifier was fitted with feature names
  warnings.warn(

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

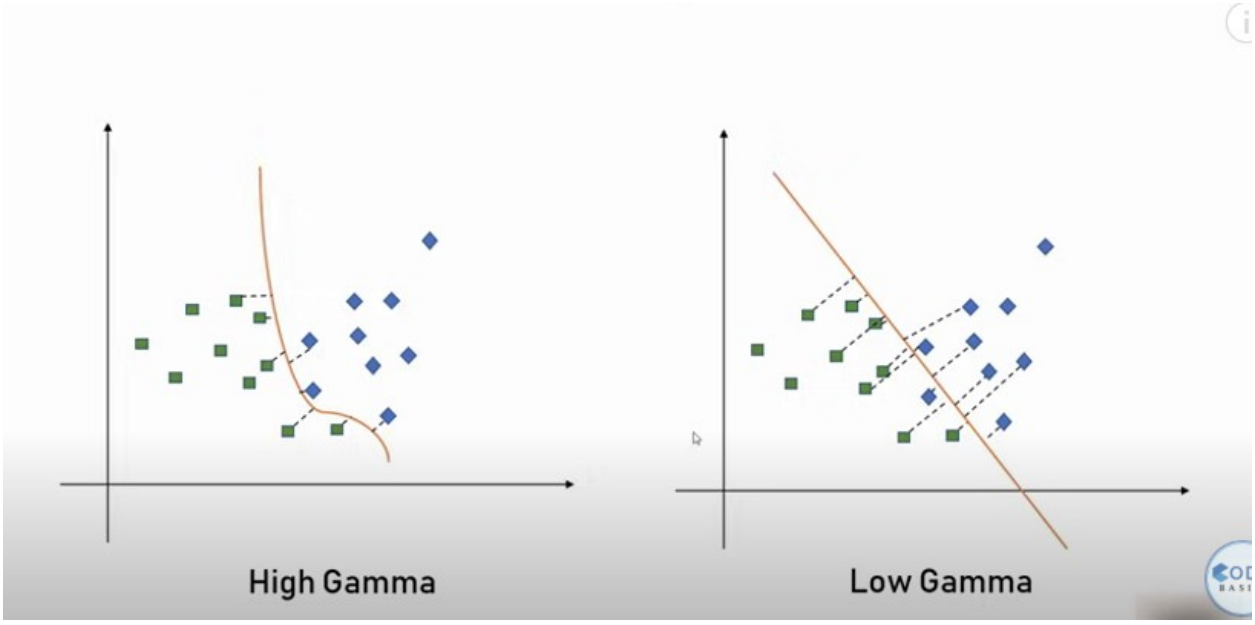
SUPPORT VECTOR MACHINE(SVM)

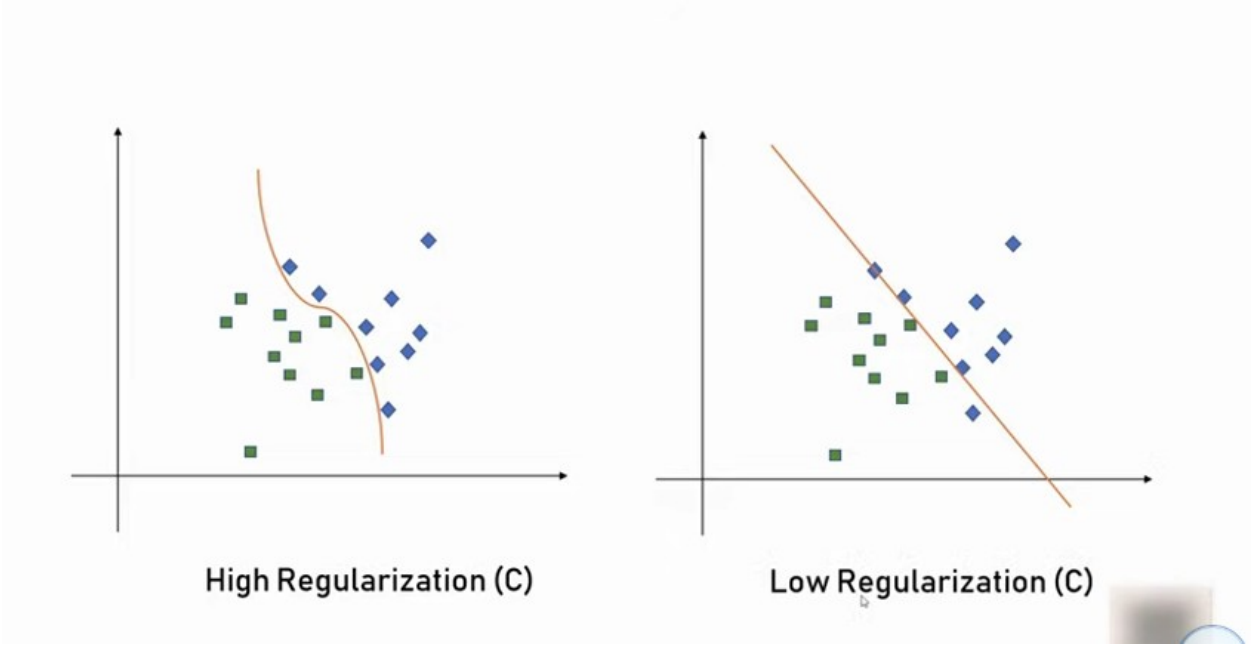


SVM tries to maximise the margin lenght inorder to better classify objects

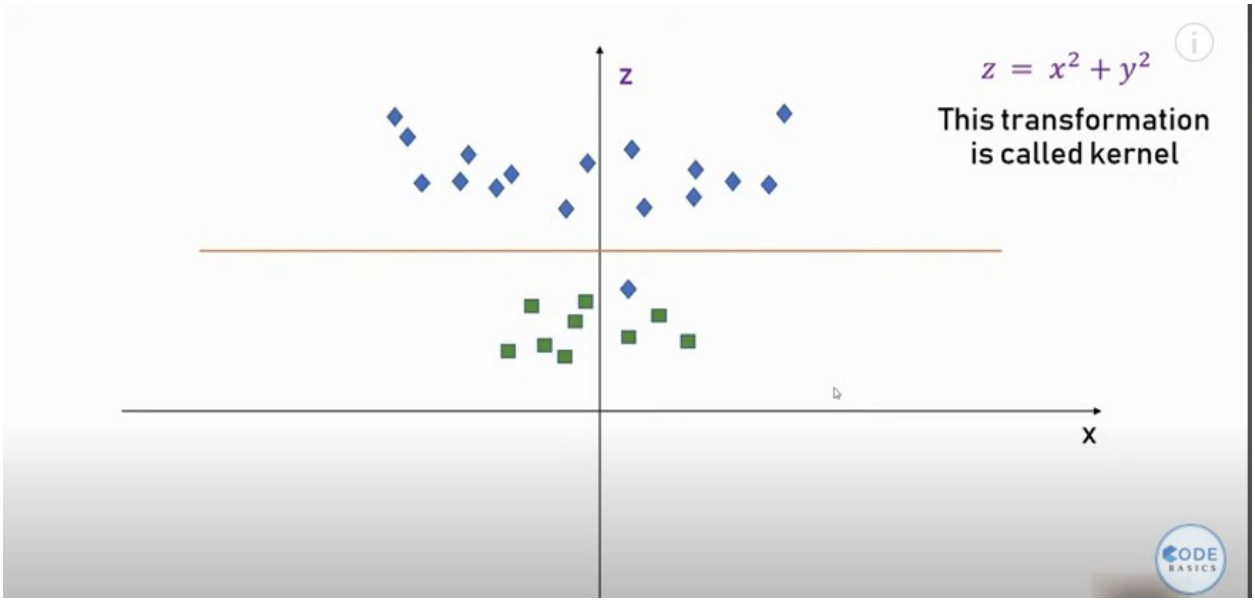


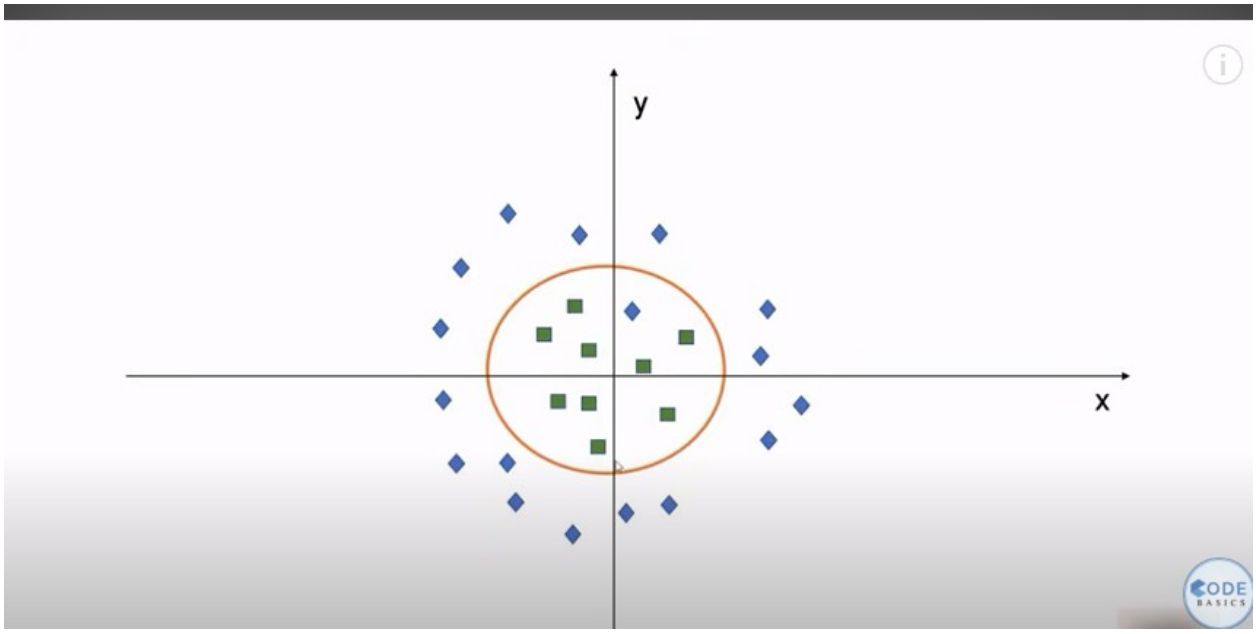
Support vector machine draws a hyper plane in n dimensional space such that it maximizes margin between classification groups





"C" means "REGULARISATION"






```
import pandas as pd
from sklearn.datasets import load_iris
iris=load_iris()

dir(iris)
```

```
['DESCR',
 'data',
 'data_module',
 'feature_names',
 'filename',
 'frame',
 'target',
 'target_names']
```

```
iris.feature_names
```

```
['sepal length (cm)',
 'sepal width (cm)',
 'petal length (cm)',
 'petal width (cm)']
```

```
df=pd.DataFrame(iris.data,columns=iris.feature_names)
df.head()
```

```
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width
(cm)
0                5.1               3.5                1.4
0.2
1                4.9               3.0                1.4
0.2
2                4.7               3.2                1.3
```

```
0.2
3                4.6               3.1                1.5
0.2
4                5.0               3.6                1.4
0.2
```

```
df['target']=iris.target
df.head()
```

```
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width
(cm)  \
0                5.1               3.5                1.4
0.2
1                4.9               3.0                1.4
0.2
2                4.7               3.2                1.3
0.2
3                4.6               3.1                1.5
0.2
4                5.0               3.6                1.4
0.2

   target
0       0
1       0
2       0
3       0
4       0
```

```
iris.target_names
```

```
array(['setosa', 'versicolor', 'virginica'], dtype='<U10')
```

```
df[df.target==1].head()
```

```
    sepal length (cm)  sepal width (cm)  petal length (cm)  petal
width (cm)  \
50                7.0               3.2                4.7
1.4
51                6.4               3.2                4.5
1.5
52                6.9               3.1                4.9
1.5
53                5.5               2.3                4.0
1.3
54                6.5               2.8                4.6
1.5

    target
50       1
51       1
```

```
52       1
53       1
54       1
```

```
df[df.target==2].head()
```

```
     sepal length (cm)  sepal width (cm)  petal length (cm)  petal
width (cm)  \
100                6.3               3.3                6.0
2.5
101                5.8               2.7                5.1
1.9
102                7.1               3.0                5.9
2.1
103                6.3               2.9                5.6
1.8
104                6.5               3.0                5.8
2.2

     target
100       2
101       2
102       2
103       2
104       2
```

```
df['flower_name']=df.target.apply(lambda x: iris.target_names[x])
df.head()
```

```
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width
(cm)  \
0                5.1               3.5                1.4
0.2
1                4.9               3.0                1.4
0.2
2                4.7               3.2                1.3
0.2
3                4.6               3.1                1.5
0.2
4                5.0               3.6                1.4
0.2

   target flower_name
0       0      setosa
1       0      setosa
2       0      setosa
3       0      setosa
4       0      setosa
```

```
from matplotlib import pyplot as plt
%matplotlib inline
```

```
df0=df[df.target==0]
df1=df[df.target==1]
df2=df[df.target==2]

df0.head()
```

```
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width
(cm)  \
0                5.1               3.5                1.4
0.2
1                4.9               3.0                1.4
0.2
2                4.7               3.2                1.3
0.2
3                4.6               3.1                1.5
0.2
4                5.0               3.6                1.4
0.2

   target flower_name
0       0      setosa
1       0      setosa
2       0      setosa
3       0      setosa
4       0      setosa
```

```
df1.head()
```

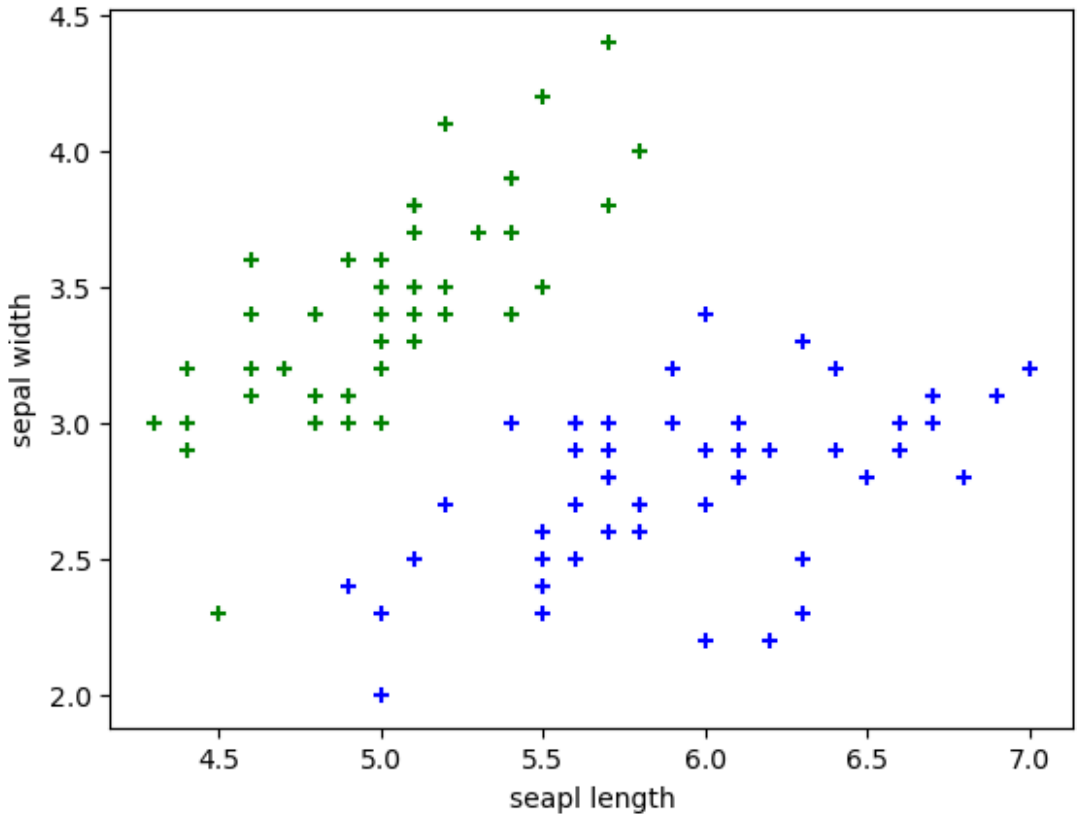
```
    sepal length (cm)  sepal width (cm)  petal length (cm)  petal
width (cm)  \
50                7.0               3.2                4.7
1.4
51                6.4               3.2                4.5
1.5
52                6.9               3.1                4.9
1.5
53                5.5               2.3                4.0
1.3
54                6.5               2.8                4.6
1.5

    target flower_name
50       1  versicolor
51       1  versicolor
52       1  versicolor
53       1  versicolor
54       1  versicolor
```

```
df2.head()
```

```
     sepal length (cm)  sepal width (cm)  petal length (cm)  petal
width (cm)  \
100                6.3               3.3                6.0
2.5
101                5.8               2.7                5.1
1.9
102                7.1               3.0                5.9
2.1
103                6.3               2.9                5.6
1.8
104                6.5               3.0                5.8
2.2

     target flower_name
100       2   virginica
101       2   virginica
102       2   virginica
103       2   virginica
104       2   virginica
```

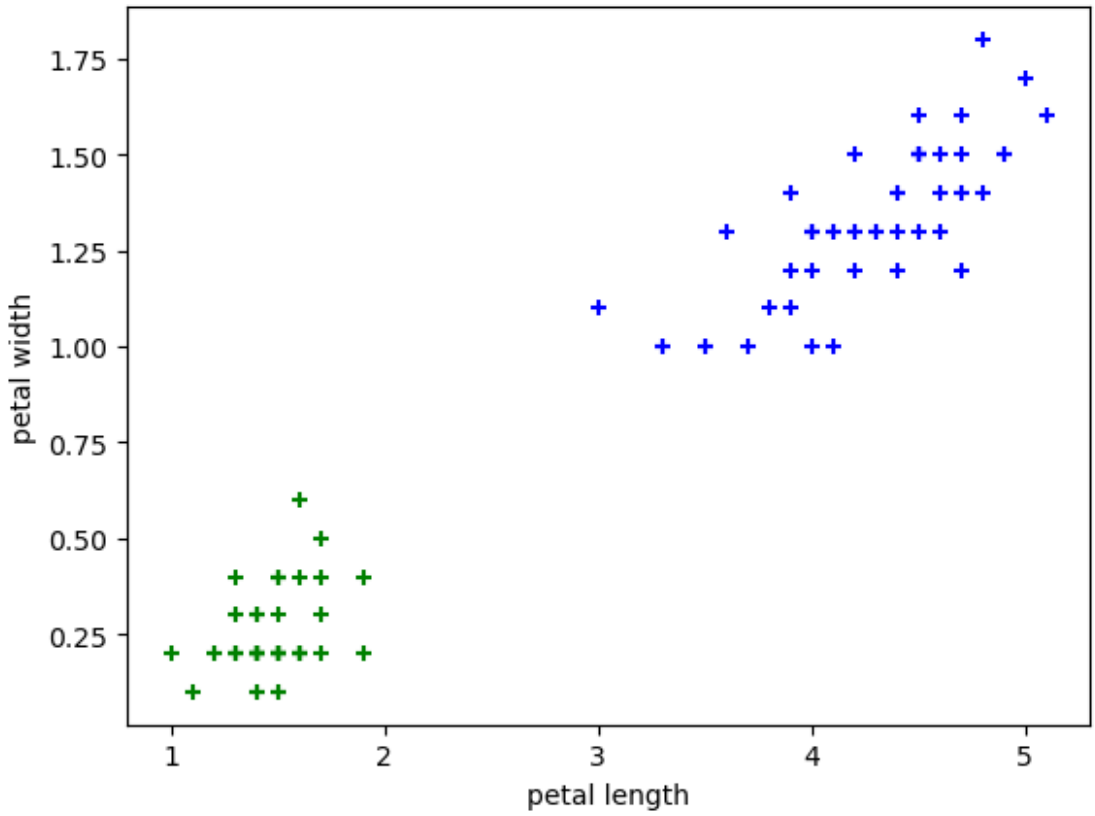
```
plt.scatter(df0['sepal length (cm)'],df0['sepal width
(cm)'],color='green',marker='+')
plt.scatter(df1['sepal length (cm)'],df1['sepal width
(cm)'],color='blue',marker='+')
plt.xlabel('seapl length')
plt.ylabel('sepal width')
```

```
Text(0, 0.5, 'sepal width')
```




```
plt.scatter(df0['petal length (cm)'],df0['petal width
(cm)'],color='green',marker='+')
plt.scatter(df1['petal length (cm)'],df1['petal width
(cm)'],color='blue',marker='+')
plt.xlabel('petal length')
plt.ylabel('petal width')
```

```
Text(0, 0.5, 'petal width')
```




```
from sklearn.model_selection import train_test_split

x=df.drop(['target','flower_name'],axis='columns')
x.head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

```
y=df.target
y.head()
```

```
0    0
1    0
2    0
3    0
```

```
4    0
Name: target, dtype: int32
```

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
```

```
len(x_train)
```

```
120
```

```
len(x_test)
```

```
30
```

```
from sklearn.svm import SVC
model=SVC()
```

```
model.fit(x_train,y_train)
```

```
SVC()
```

```
model.score(x_test,y_test)
```

```
0.9666666666666667
```

```
model=SVC(C=10)
model.fit(x_train,y_train)
```

```
SVC(C=10)
```

```
model.score(x_test,y_test)
```

```
0.9333333333333333
```

So,increasing REGULARISATION decreases model ACCURACY and same with GAMMA

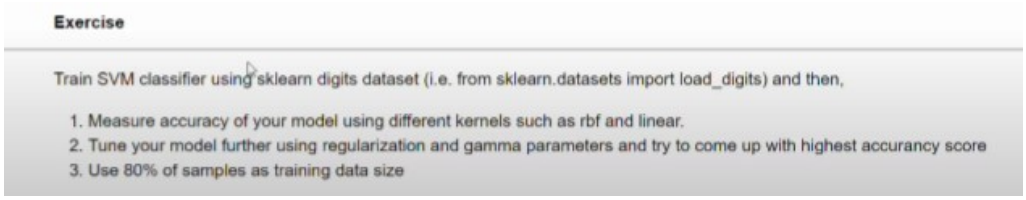
```
model=SVC(gamma=100)
model.fit(x_train,y_train)
```

```
SVC(gamma=100)
```

```
model.score(x_test,y_test)
```

```
0.4666666666666667
```

```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape=None, degree=3, gamma=100, kernel='rbf',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
```




```
import pandas as pd
from sklearn.datasets import load_digits
digits = load_digits()

digits.target
```

```
array([0, 1, 2, ..., 8, 9, 8])
```

```
dir(digits)
```

```
['DESCR', 'data', 'feature_names', 'frame', 'images', 'target',
'target_names']
```

```
digits.target_names
```

```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

```
df = pd.DataFrame(digits.data,digits.target)
df.head()
```

```
    0    1    2     3     4     5    6    7    8    9   ...   54   55
56  \
0  0.0  0.0  5.0  13.0   9.0   1.0  0.0  0.0  0.0  0.0  ...  0.0  0.0
0.0
1  0.0  0.0  0.0  12.0  13.0   5.0  0.0  0.0  0.0  0.0  ...  0.0  0.0
0.0
2  0.0  0.0  0.0   4.0  15.0  12.0  0.0  0.0  0.0  0.0  ...  5.0  0.0
0.0
3  0.0  0.0  7.0  15.0  13.0   1.0  0.0  0.0  0.0  8.0  ...  9.0  0.0
0.0
4  0.0  0.0  0.0   1.0  11.0   0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0
0.0

    57   58    59    60    61   62   63
0  0.0  6.0  13.0  10.0   0.0  0.0  0.0
1  0.0  0.0  11.0  16.0  10.0  0.0  0.0
2  0.0  0.0   3.0  11.0  16.0  9.0  0.0
3  0.0  7.0  13.0  13.0   9.0  0.0  0.0
4  0.0  0.0   2.0  16.0   4.0  0.0  0.0

[5 rows x 64 columns]
```

```
df['target'] = digits.target
df.head(20)
```

```
     0    1     2     3     4     5     6    7    8     9  ...   55
56   57  \
0  0.0  0.0   5.0  13.0   9.0   1.0   0.0  0.0  0.0   0.0  ...  0.0
0.0  0.0
1  0.0  0.0   0.0  12.0  13.0   5.0   0.0  0.0  0.0   0.0  ...  0.0
0.0  0.0
2  0.0  0.0   0.0   4.0  15.0  12.0   0.0  0.0  0.0   0.0  ...  0.0
0.0  0.0
3  0.0  0.0   7.0  15.0  13.0   1.0   0.0  0.0  0.0   8.0  ...  0.0
0.0  0.0
4  0.0  0.0   0.0   1.0  11.0   0.0   0.0  0.0  0.0   0.0  ...  0.0
0.0  0.0
5  0.0  0.0  12.0  10.0   0.0   0.0   0.0  0.0  0.0   0.0  ...  0.0
0.0  0.0
6  0.0  0.0   0.0  12.0  13.0   0.0   0.0  0.0  0.0   0.0  ...  0.0
0.0  0.0
7  0.0  0.0   7.0   8.0  13.0  16.0  15.0  1.0  0.0   0.0  ...  0.0
0.0  0.0
8  0.0  0.0   9.0  14.0   8.0   1.0   0.0  0.0  0.0   0.0  ...  0.0
0.0  0.0
9  0.0  0.0  11.0  12.0   0.0   0.0   0.0  0.0  0.0   2.0  ...  0.0
0.0  0.0
0  0.0  0.0   1.0   9.0  15.0  11.0   0.0  0.0  0.0   0.0  ...  0.0
0.0  0.0
1  0.0  0.0   0.0   0.0  14.0  13.0   1.0  0.0  0.0   0.0  ...  0.0
0.0  0.0
2  0.0  0.0   5.0  12.0   1.0   0.0   0.0  0.0  0.0   0.0  ...  2.0
0.0  0.0
3  0.0  2.0   9.0  15.0  14.0   9.0   3.0  0.0  0.0   4.0  ...  0.0
0.0  2.0
4  0.0  0.0   0.0   8.0  15.0   1.0   0.0  0.0  0.0   0.0  ...  0.0
0.0  0.0
5  0.0  5.0  12.0  13.0  16.0  16.0   2.0  0.0  0.0  11.0  ...  0.0
0.0  4.0
6  0.0  0.0   0.0   8.0  15.0   1.0   0.0  0.0  0.0   0.0  ...  2.0
0.0  0.0
7  0.0  0.0   1.0   8.0  15.0  10.0   0.0  0.0  0.0   3.0  ...  0.0
0.0  0.0
8  0.0  0.0  10.0   7.0  13.0   9.0   0.0  0.0  0.0   0.0  ...  0.0
0.0  0.0
9  0.0  0.0   6.0  14.0   4.0   0.0   0.0  0.0  0.0   0.0  ...  2.0
0.0  0.0

     58    59    60    61    62   63  target
0   6.0  13.0  10.0   0.0   0.0  0.0       0
1   0.0  11.0  16.0  10.0   0.0  0.0       1
2   0.0   3.0  11.0  16.0   9.0  0.0       2
3   7.0  13.0  13.0   9.0   0.0  0.0       3
4   0.0   2.0  16.0   4.0   0.0  0.0       4
5   9.0  16.0  16.0  10.0   0.0  0.0       5
```

```
6   1.0   9.0  15.0  11.0   3.0  0.0       6
7  13.0   5.0   0.0   0.0   0.0  0.0       7
8  11.0  16.0  15.0  11.0   1.0  0.0       8
9   9.0  12.0  13.0   3.0   0.0  0.0       9
0   1.0  10.0  13.0   3.0   0.0  0.0       0
1   0.0   1.0  13.0  16.0   1.0  0.0       1
2   3.0  11.0   8.0  13.0  12.0  4.0       2
3  12.0  12.0  13.0  11.0   0.0  0.0       3
4   0.0  10.0  15.0   4.0   0.0  0.0       4
5  15.0  16.0   2.0   0.0   0.0  0.0       5
6   0.0   7.0  15.0  16.0  11.0  0.0       6
7   0.0  11.0   9.0   0.0   0.0  0.0       7
8  11.0  14.0   5.0   0.0   0.0  0.0       8
9   7.0  16.0  16.0  13.0  11.0  1.0       9

[20 rows x 65 columns]
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test =
train_test_split(df.drop('target',axis='columns'), df.target,
test_size=0.3)
```

```
from sklearn.svm import SVC
rbf_model = SVC(kernel='rbf')
```

```
from sklearn.svm import SVC
rbf_model = SVC(kernel='rbf')
```

```
len(X_train)
```

```
1257
```

```
len(X_test)
```

```
540
```

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

```
SVC()
```

```
rbf_model.score(X_test,y_test)
```

```
0.9888888888888889
```

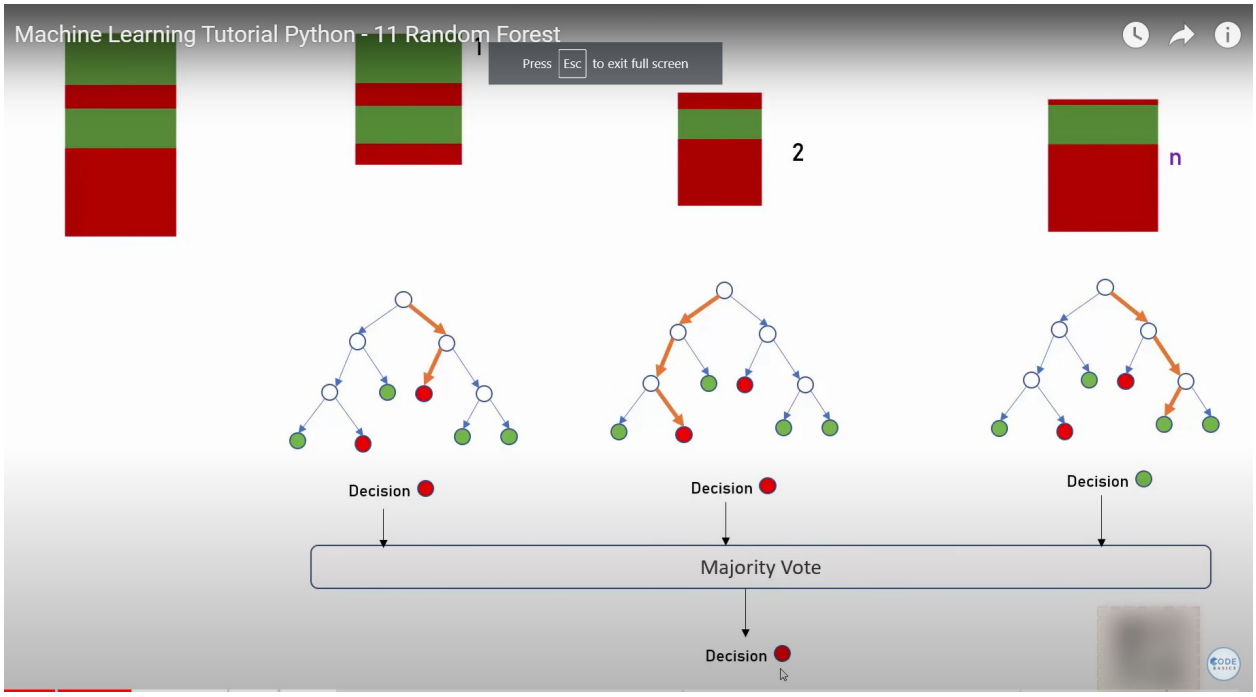
```
linear_model = SVC(kernel='linear')
linear_model.fit(X_train,y_train)
```

```
SVC(kernel='linear')
```

```
linear_model.score(X_test,y_test)
```

```
0.9814814814814815
```

Random Forest Algorithm

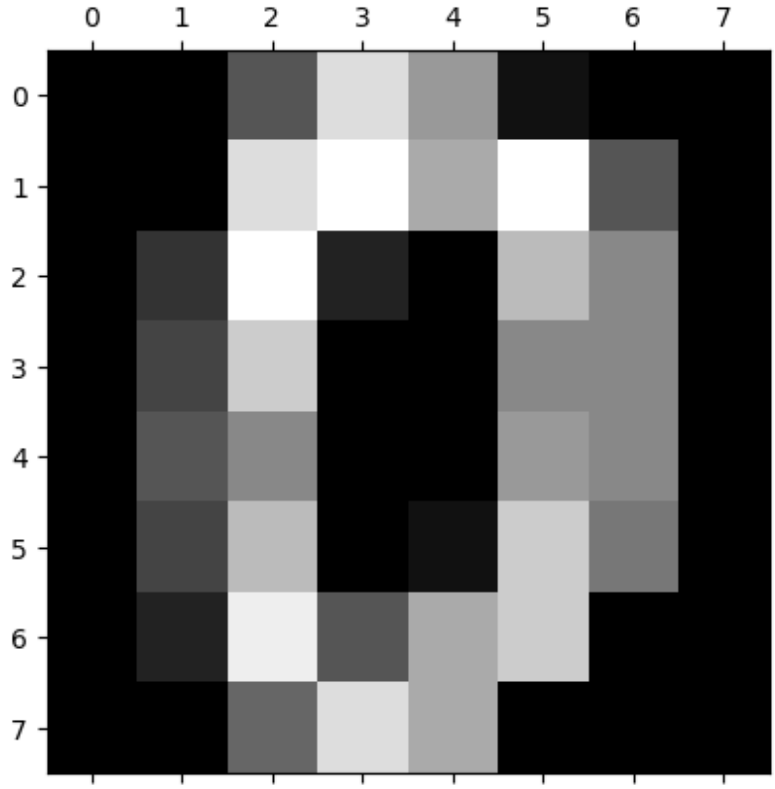


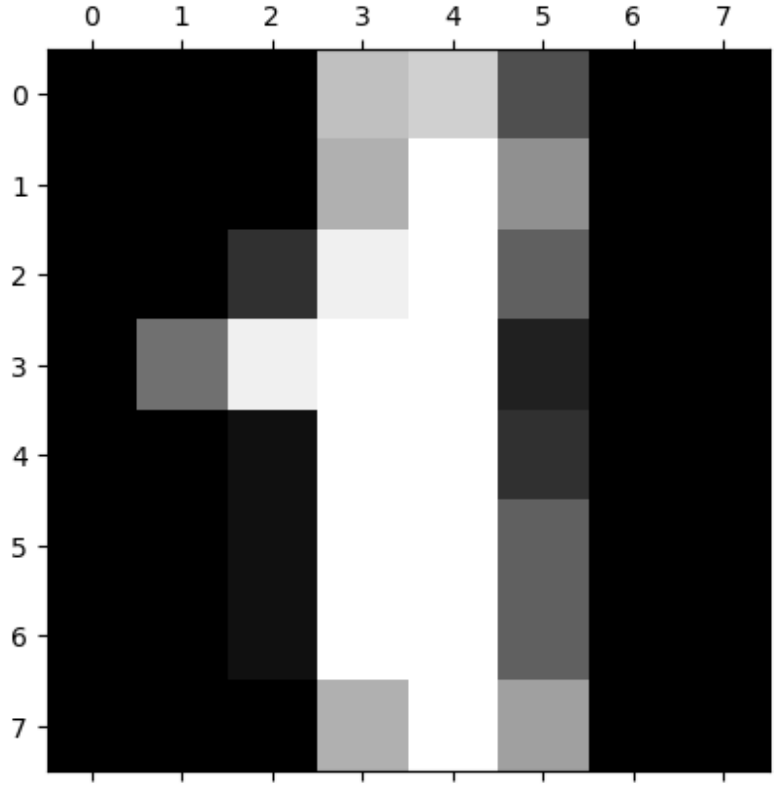

```
import pandas as pd
from sklearn.datasets import load_digits
digits=load_digits()

dir(digits)
```

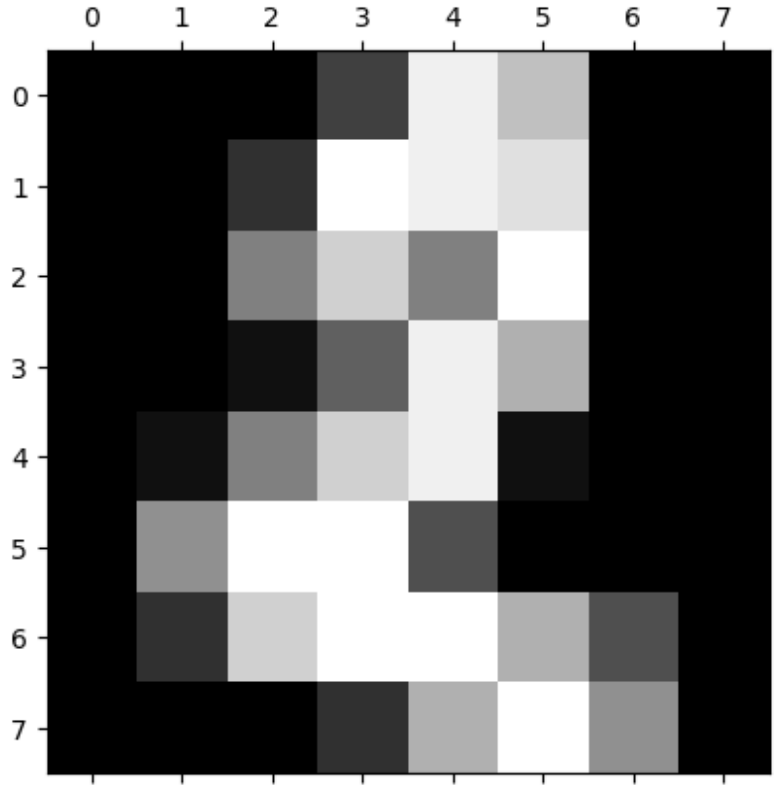
```
['DESCR', 'data', 'feature_names', 'frame', 'images', 'target',
'target_names']
```

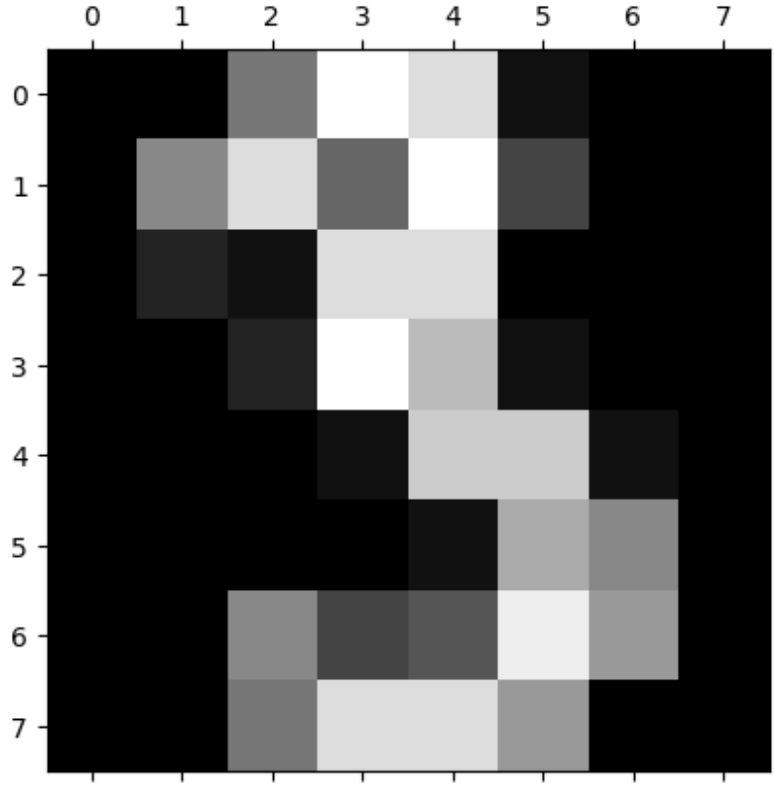
```
%matplotlib inline
import matplotlib.pyplot as plt
plt.gray()
for i in range(4):
    plt.matshow(digits.images[i])
```

```
<Figure size 640x480 with 0 Axes>
```


0 1 2 3 4 5 6 7
0 1 2 3 4 5 6 7


0 1 2 3 4 5 6 7
0 1 2 3 4 5 6 7


0 1 2 3 4 5 6 7
0 1 2 3 4 5 6 7


0 1 2 3 4 5 6 7
0 1 2 3 4 5 6 7

```
df=pd.DataFrame(digits.data)
df.head()
```

```
    0    1    2     3     4     5    6    7    8    9   ...   54   55
56  \
0  0.0  0.0  5.0  13.0   9.0   1.0  0.0  0.0  0.0  0.0  ...  0.0  0.0
0.0
1  0.0  0.0  0.0  12.0  13.0   5.0  0.0  0.0  0.0  0.0  ...  0.0  0.0
0.0
2  0.0  0.0  0.0   4.0  15.0  12.0  0.0  0.0  0.0  0.0  ...  5.0  0.0
0.0
3  0.0  0.0  7.0  15.0  13.0   1.0  0.0  0.0  0.0  8.0  ...  9.0  0.0
0.0
4  0.0  0.0  0.0   1.0  11.0   0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0
0.0

    57   58    59    60    61   62   63
0  0.0  6.0  13.0  10.0   0.0  0.0  0.0
1  0.0  0.0  11.0  16.0  10.0  0.0  0.0
2  0.0  0.0   3.0  11.0  16.0  9.0  0.0
3  0.0  7.0  13.0  13.0   9.0  0.0  0.0
4  0.0  0.0   2.0  16.0   4.0  0.0  0.0

[5 rows x 64 columns]
```

```
digits.target
```

```
array([0, 1, 2, ..., 8, 9, 8])
```

```
df['target']=digits.target
df.head()
```

```
     0    1    2     3     4     5    6    7    8    9  ...   55   56
57  \
0  0.0  0.0  5.0  13.0   9.0   1.0  0.0  0.0  0.0  0.0  ...  0.0  0.0
0.0
1  0.0  0.0  0.0  12.0  13.0   5.0  0.0  0.0  0.0  0.0  ...  0.0  0.0
0.0
2  0.0  0.0  0.0   4.0  15.0  12.0  0.0  0.0  0.0  0.0  ...  0.0  0.0
0.0
3  0.0  0.0  7.0  15.0  13.0   1.0  0.0  0.0  0.0  8.0  ...  0.0  0.0
0.0
4  0.0  0.0  0.0   1.0  11.0   0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0
0.0

    58    59    60    61   62   63  target
0  6.0  13.0  10.0   0.0  0.0  0.0       0
1  0.0  11.0  16.0  10.0  0.0  0.0       1
2  0.0   3.0  11.0  16.0  9.0  0.0       2
3  7.0  13.0  13.0   9.0  0.0  0.0       3
4  0.0   2.0  16.0   4.0  0.0  0.0       4
```

```
[5 rows x 65 columns]
```

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(df.drop(['target'],axis
='columns'),digits.target,test_size=0.2)
```

```
len(x_train)
```

```
1437
```

```
len(x_test)
```

```
360
```

```
from sklearn.ensemble import RandomForestClassifier
model=RandomForestClassifier()
model.fit(x_train,y_train)
```

```
RandomForestClassifier()
```

```
model.score(x_test,y_test)
```

```
0.975
```

```
model=RandomForestClassifier(n_estimators=10)
model.fit(x_train,y_train)
```

```
RandomForestClassifier(n_estimators=10)
```

```
model.score(x_test,y_test)
```

```
0.9333333333333333
```

```
model=RandomForestClassifier(n_estimators=950)
model.fit(x_train,y_train)
```

```
RandomForestClassifier(n_estimators=950)
```

```
model.score(x_test,y_test)
```

```
0.9833333333333333
```

```
y_predicted=model.predict(x_test)
```

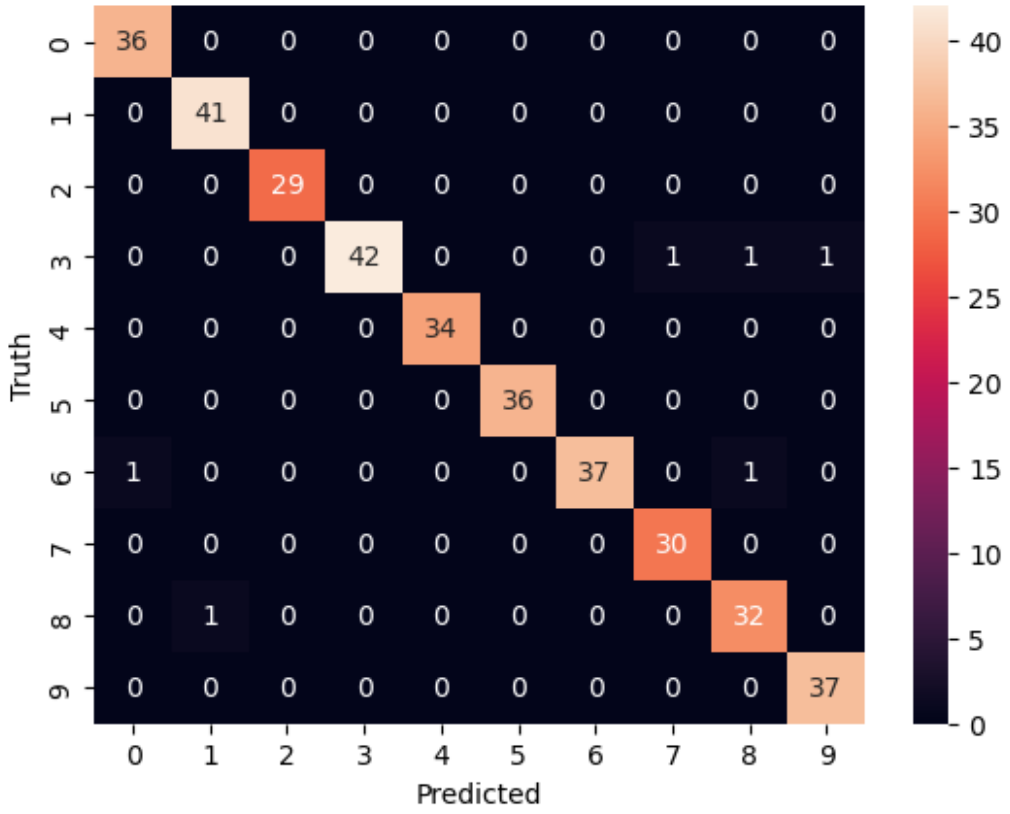
```
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,y_predicted)
cm
```

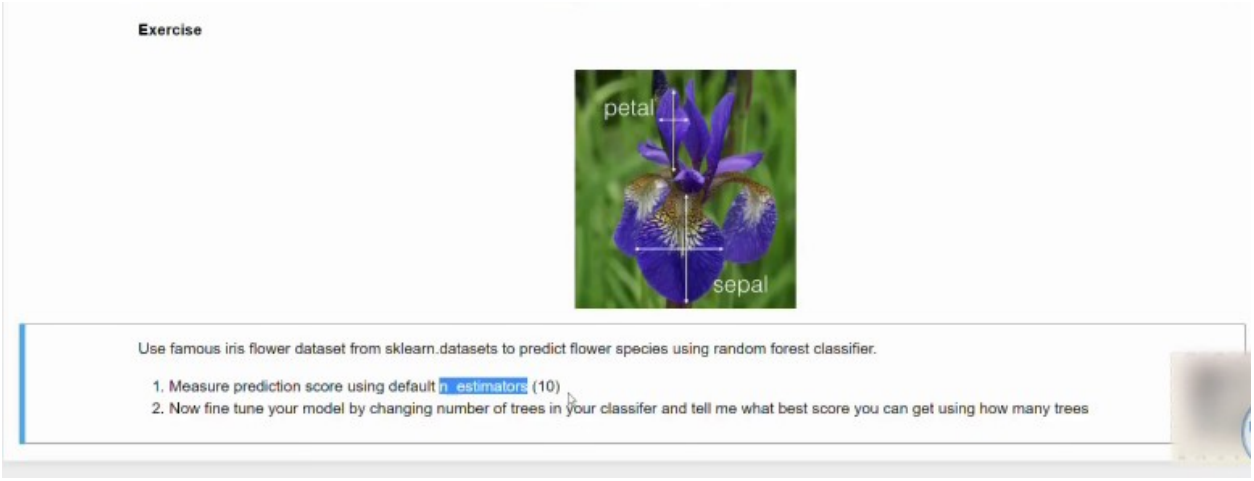
```
array([[36,  0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0, 41,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0, 29,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0, 42,  0,  0,  0,  1,  1,  1],
       [ 0,  0,  0,  0, 34,  0,  0,  0,  0,  0],
```

```
       [ 0,  0,  0,  0,  0, 36,  0,  0,  0,  0],
       [ 1,  0,  0,  0,  0,  0, 37,  0,  1,  0],
       [ 0,  0,  0,  0,  0,  0,  0, 30,  0,  0],
       [ 0,  1,  0,  0,  0,  0,  0,  0, 32,  0],
       [ 0,  0,  0,  0,  0,  0,  0,  0,  0, 37]], dtype=int64)
```

```
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sn
sn.heatmap(cm,annot=True)
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

```
Text(50.722222222222214, 0.5, 'Truth')
```






```
import pandas as pd
from sklearn.datasets import load_iris
iris=load_iris()

iris.feature_names
```

```
['sepal length (cm)',
 'sepal width (cm)',
 'petal length (cm)',
 'petal width (cm)']
```

```
dir(iris)
```

```
['DESCR',
 'data',
 'data_module',
 'feature_names',
 'filename',
 'frame',
 'target',
 'target_names']
```

```
df=pd.DataFrame(iris.data,columns=iris.feature_names)
df.head()
```

```
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width
(cm)
0                5.1               3.5                1.4
0.2
1                4.9               3.0                1.4
0.2
2                4.7               3.2                1.3
0.2
3                4.6               3.1                1.5
0.2
```

```
4                 5.0                3.6                 1.4
0.2

df['target']=iris.target
df

     sepal length (cm)  sepal width (cm)  petal length (cm)  petal
width (cm)  \
0                  5.1               3.5                 1.4
0.2
1                  4.9               3.0                 1.4
0.2
2                  4.7               3.2                 1.3
0.2
3                  4.6               3.1                 1.5
0.2
4                  5.0               3.6                 1.4
0.2
..                 ...               ...                 ...
...
145                6.7               3.0                 5.2
2.3
146                6.3               2.5                 5.0
1.9
147                6.5               3.0                 5.2
2.0
148                6.2               3.4                 5.4
2.3
149                5.9               3.0                 5.1
1.8

     target
0         0
1         0
2         0
3         0
4         0
..      ...
145       2
146       2
147       2
148       2
149       2

[150 rows x 5 columns]

df[df.target==2].head()

     sepal length (cm)  sepal width (cm)  petal length (cm)  petal
width (cm)  \
```

```
100                6.3               3.3                6.0
2.5
101                5.8               2.7                5.1
1.9
102                7.1               3.0                5.9
2.1
103                6.3               2.9                5.6
1.8
104                6.5               3.0                5.8
2.2

     target
100       2
101       2
102       2
103       2
104       2
```

```
df['flowername']=df.target.apply(lambda x: iris.target_names[x])
df.head()
```

```
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width
(cm)  \
0                5.1               3.5                1.4
0.2
1                4.9               3.0                1.4
0.2
2                4.7               3.2                1.3
0.2
3                4.6               3.1                1.5
0.2
4                5.0               3.6                1.4
0.2

   target flowername
0       0     setosa
1       0     setosa
2       0     setosa
3       0     setosa
4       0     setosa
```

```
x=df.drop(['target','flowername'],axis='columns')
x.head()
```

```
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width
(cm)
0                5.1               3.5                1.4
0.2
1                4.9               3.0                1.4
0.2
```

```
2                 4.7               3.2                1.3
0.2
3                 4.6               3.1                1.5
0.2
4                 5.0               3.6                1.4
0.2
```

```
y=df.target
y.head()
```

```
0    0
1    0
2    0
3    0
4    0
Name: target, dtype: int32
```

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

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
```

```
len(x_test)
```

```
30
```

```
len(x_train)
```

```
120
```

```
from sklearn.ensemble import RandomForestClassifier
model=RandomForestClassifier(n_estimators=100)
model.fit(x_train,y_train)
```

```
RandomForestClassifier()
```

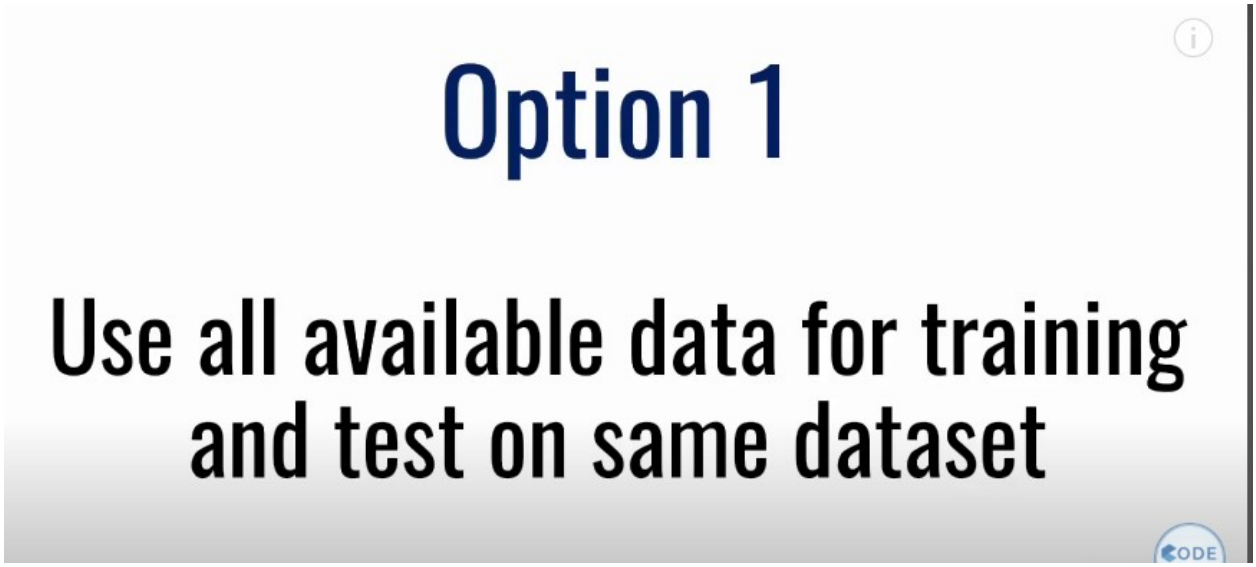
```
model.score(x_test,y_test)
```

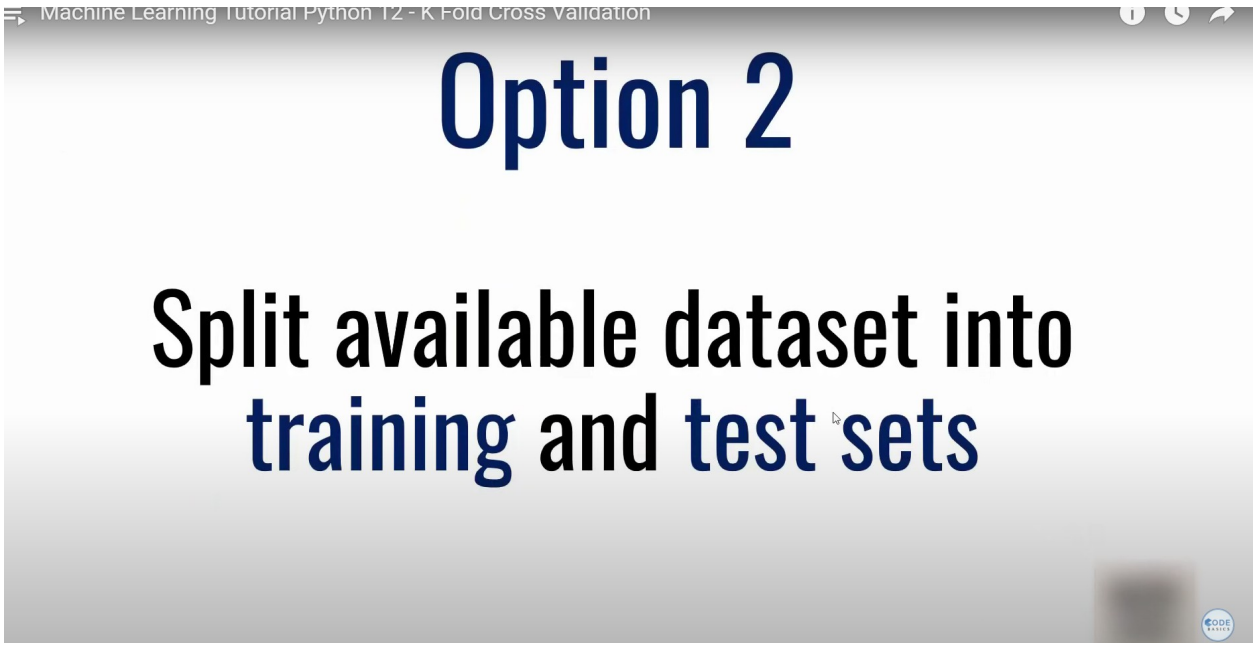
```
0.9666666666666667
```

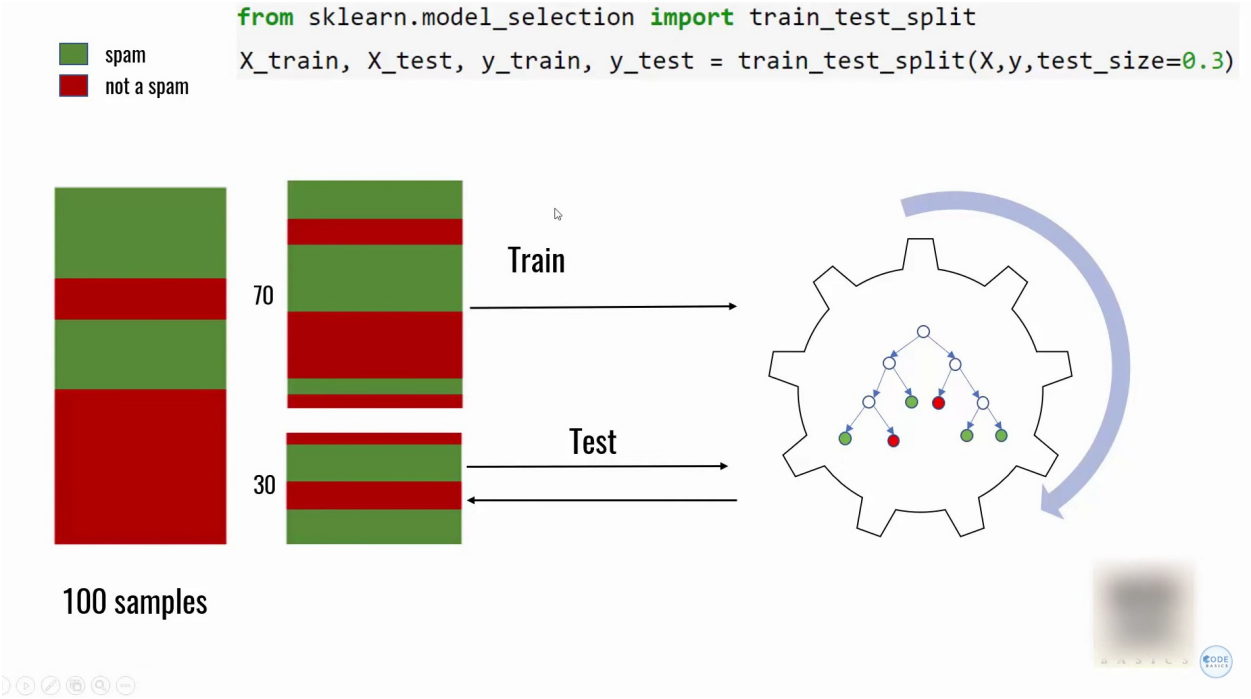
K FOLD CROSS VALIDATION

we can check that which of the above model is most efficient,fast,correct,etc.

We have some ways to check out model score of above models.OPTION 1 is to train and test model on same data set.OPTION 2 is to train amodel on a part of dataset and then test it on another part of same dataset.this is not perfect as the pattern of data encountered in training set may be totally different from testing set.so,THIRD OPTION is to first train data on a particular fold of dataset and test with rest part,thus changing the testing and training fold and then taking out average of all scores to get final score of model used.

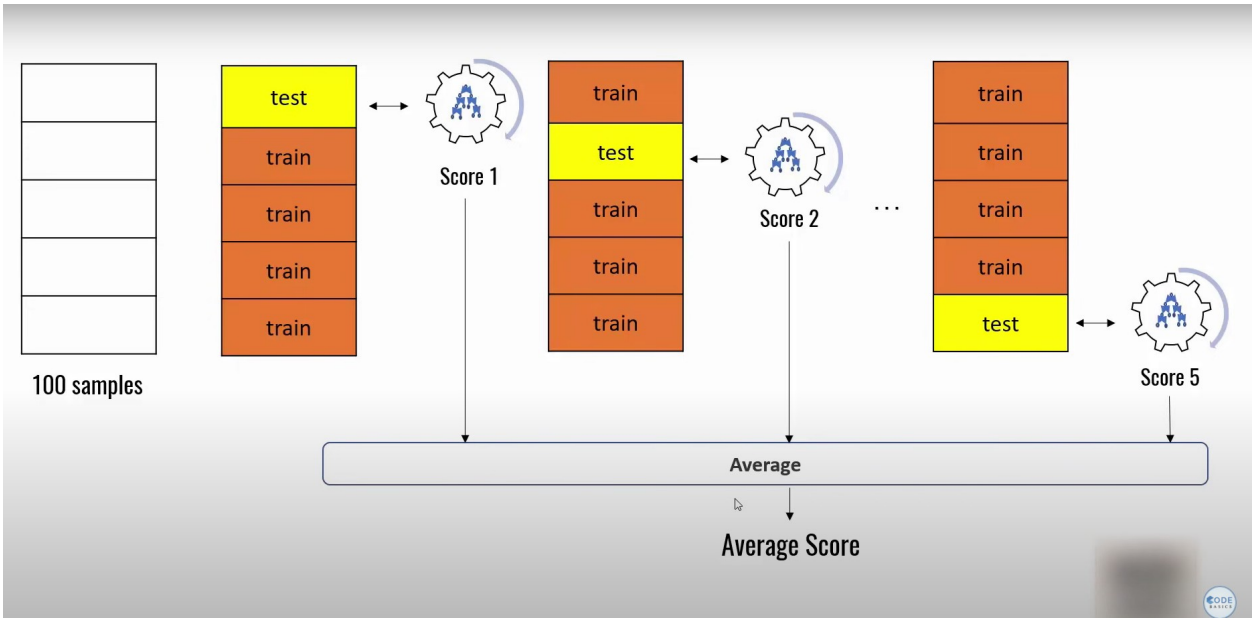

Option 1
Use all available data for training
and test on same dataset


Machine Learning Tutorial Python 12 - K Fold Cross Validation
Option 2
Split available dataset into
training and test sets



Option 3

K fold cross validation




```
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
import numpy as np
from sklearn.datasets import load_digits

digits=load_digits()

from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(digits.data,digits.targ
et,test_size=0.3)

lr=LogisticRegression()
lr.fit(x_train,y_train)
lr.score(x_test,y_test)
```

```
C:\ProgramData\anaconda3\lib\site-packages\sklearn\linear_model\
_logistic.py:458: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as
shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(

0.9685185185185186
```

```
svm=SVC()
svm.fit(x_train,y_train)
svm.score(x_test,y_test)
```

```
0.9925925925925926
```

```
rf=RandomForestClassifier()
rf.fit(x_train,y_train)
rf.score(x_test,y_test)
```

```
0.9833333333333333
```

we can't decide in a single run score that which model is more accurate because inn each run train and test data sets changes which may affect the score of a model severely.

```
from sklearn.model_selection import KFold
kf=KFold(n_splits=3)
kf
```

```
KFold(n_splits=3, random_state=None, shuffle=False)
```

```
for train_index,test_index in kf.split([1,2,3,4,5,6,7,8,9]):
    print(train_index,test_index)
```

```
[3 4 5 6 7 8] [0 1 2]
[0 1 2 6 7 8] [3 4 5]
[0 1 2 3 4 5] [6 7 8]
```

```
def get_score(model,x_train,x_test,y_train,y_test):
    model.fit(x_train,y_train)
    return model.score(x_test,y_test)
```

```
 get_score(LogisticRegression(),x_train,x_test,y_train,y_test)
```

```
C:\ProgramData\anaconda3\lib\site-packages\sklearn\linear_model\
_logistic.py:458: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as
shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

```
0.9685185185185186
```

```
get_score(SVC(),x_train,x_test,y_train,y_test)
```

```
0.9925925925925926
```

```
from sklearn.model_selection import StratifiedKFold
folds=StratifiedKFold(n_splits=3)

scores_l=[]
scores_svm=[]
scores_rf=[]
for train_index,test_index in kf.split(digits.data):

x_train,x_test,y_train,y_test=digits.data[train_index],digits.data[tes
t_index],digits.target[train_index],digits.target[test_index]

scores_l.append(get_score(LogisticRegression(),x_train,x_test,y_train,
y_test))
    scores_svm.append(get_score(SVC(),x_train,x_test,y_train,y_test))

scores_rf.append(get_score(RandomForestClassifier(),x_train,x_test,y_t
rain,y_test))
```

```
C:\ProgramData\anaconda3\lib\site-packages\sklearn\linear_model\
_logistic.py:458: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as
shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
C:\ProgramData\anaconda3\lib\site-packages\sklearn\linear_model\
_logistic.py:458: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as
shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
C:\ProgramData\anaconda3\lib\site-packages\sklearn\linear_model\
_logistic.py:458: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

```
Increase the number of iterations (max_iter) or scale the data as
shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(

scores_l

[0.9232053422370617, 0.9415692821368948, 0.9148580968280468]

scores_svm

[0.9666110183639399, 0.9816360601001669, 0.9549248747913188]

scores_rf

[0.9449081803005008, 0.9482470784641068, 0.9265442404006677]
```

The above comparisison can be done in one line using Cross_val_score

```
from sklearn.model_selection import cross_val_score

cross_val_score (LogisticRegression(),digits.data,digits.target)

C:\ProgramData\anaconda3\lib\site-packages\sklearn\linear_model\
_logistic.py:458: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as
shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
C:\ProgramData\anaconda3\lib\site-packages\sklearn\linear_model\
_logistic.py:458: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as
shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
```

```
https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
C:\ProgramData\anaconda3\lib\site-packages\sklearn\linear_model\
_logistic.py:458: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as
shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
C:\ProgramData\anaconda3\lib\site-packages\sklearn\linear_model\
_logistic.py:458: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as
shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
C:\ProgramData\anaconda3\lib\site-packages\sklearn\linear_model\
_logistic.py:458: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as
shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(

array([0.92222222, 0.86944444, 0.94150418, 0.93871866, 0.89693593])

cross_val_score (SVC(),digits.data,digits.target)

array([0.96111111, 0.94444444, 0.98328691, 0.98885794, 0.93871866])

cross_val_score (RandomForestClassifier(),digits.data,digits.target)
```

```
array([0.93888889, 0.91111111, 0.96657382, 0.9637883 , 0.9275766 ])

cross_val_score
(RandomForestClassifier(n_estimators=15),digits.data,digits.target)

array([0.91111111, 0.875     , 0.94150418, 0.95821727, 0.90529248])
```

Thus we can do cross validation for different methods and even we can do parameter tuning in same method to acheive max score.

```
pip install nbconvert[webpdf]

Defaulting to user installation because normal site-packages is not
writeable
Requirement already satisfied: nbconvert[webpdf] in c:\programdata\
anaconda3\lib\site-packages (6.5.4)
Requirement already satisfied: jupyterlab-pygments in c:\programdata\
anaconda3\lib\site-packages (from nbconvert[webpdf]) (0.1.2)
Requirement already satisfied: mistune<2,>=0.8.1 in c:\programdata\
anaconda3\lib\site-packages (from nbconvert[webpdf]) (0.8.4)
Requirement already satisfied: nbclient>=0.5.0 in c:\programdata\
anaconda3\lib\site-packages (from nbconvert[webpdf]) (0.5.13)
Requirement already satisfied: pandocfilters>=1.4.1 in c:\programdata\
anaconda3\lib\site-packages (from nbconvert[webpdf]) (1.5.0)
Requirement already satisfied: lxml in c:\programdata\anaconda3\lib\
site-packages (from nbconvert[webpdf]) (4.9.1)
Requirement already satisfied: bleach in c:\programdata\anaconda3\lib\
site-packages (from nbconvert[webpdf]) (4.1.0)
Requirement already satisfied: MarkupSafe>=2.0 in c:\programdata\
anaconda3\lib\site-packages (from nbconvert[webpdf]) (2.1.1)
Requirement already satisfied: defusedxml in c:\programdata\anaconda3\
lib\site-packages (from nbconvert[webpdf]) (0.7.1)
Requirement already satisfied: nbformat>=5.1 in c:\programdata\
anaconda3\lib\site-packages (from nbconvert[webpdf]) (5.7.0)
Requirement already satisfied: tinycss2 in c:\programdata\anaconda3\
lib\site-packages (from nbconvert[webpdf]) (1.2.1)
Requirement already satisfied: pygments>=2.4.1 in c:\programdata\
anaconda3\lib\site-packages (from nbconvert[webpdf]) (2.11.2)
Requirement already satisfied: beautifulsoup4 in c:\programdata\
anaconda3\lib\site-packages (from nbconvert[webpdf]) (4.11.1)
Requirement already satisfied: packaging in c:\programdata\anaconda3\
lib\site-packages (from nbconvert[webpdf]) (22.0)
Requirement already satisfied: traitlets>=5.0 in c:\programdata\
anaconda3\lib\site-packages (from nbconvert[webpdf]) (5.7.1)
Requirement already satisfied: entrypoints>=0.2.2 in c:\programdata\
anaconda3\lib\site-packages (from nbconvert[webpdf]) (0.4)
Requirement already satisfied: jinja2>=3.0 in c:\programdata\
anaconda3\lib\site-packages (from nbconvert[webpdf]) (3.1.2)
Requirement already satisfied: jupyter-core>=4.7 in c:\programdata\
anaconda3\lib\site-packages (from nbconvert[webpdf]) (5.2.0)
```

```
Collecting pyppeteer<1.1,>=1
  Downloading pyppeteer-1.0.2-py3-none-any.whl (83 kB)
     ---------------------------------------- 83.4/83.4 kB 1.6 MB/s
eta 0:00:00
Requirement already satisfied: platformdirs>=2.5 in c:\programdata\
anaconda3\lib\site-packages (from jupyter-core>=4.7-
>nbconvert[webpdf]) (2.5.2)
Requirement already satisfied: pywin32>=1.0 in c:\programdata\
anaconda3\lib\site-packages (from jupyter-core>=4.7-
>nbconvert[webpdf]) (305.1)
Requirement already satisfied: nest-asyncio in c:\programdata\
anaconda3\lib\site-packages (from nbclient>=0.5.0->nbconvert[webpdf])
(1.5.6)
Requirement already satisfied: jupyter-client>=6.1.5 in c:\
programdata\anaconda3\lib\site-packages (from nbclient>=0.5.0-
>nbconvert[webpdf]) (7.3.4)
Requirement already satisfied: fastjsonschema in c:\programdata\
anaconda3\lib\site-packages (from nbformat>=5.1->nbconvert[webpdf])
(2.16.2)
Requirement already satisfied: jsonschema>=2.6 in c:\programdata\
anaconda3\lib\site-packages (from nbformat>=5.1->nbconvert[webpdf])
(4.17.3)
Requirement already satisfied: urllib3<2.0.0,>=1.25.8 in c:\
programdata\anaconda3\lib\site-packages (from pyppeteer<1.1,>=1-
>nbconvert[webpdf]) (1.26.14)
Collecting pyee<9.0.0,>=8.1.0
  Downloading pyee-8.2.2-py2.py3-none-any.whl (12 kB)
Requirement already satisfied: tqdm<5.0.0,>=4.42.1 in c:\programdata\
anaconda3\lib\site-packages (from pyppeteer<1.1,>=1-
>nbconvert[webpdf]) (4.64.1)
Requirement already satisfied: certifi>=2021 in c:\programdata\
anaconda3\lib\site-packages (from pyppeteer<1.1,>=1-
>nbconvert[webpdf]) (2023.5.7)
Requirement already satisfied: importlib-metadata>=1.4 in c:\
programdata\anaconda3\lib\site-packages (from pyppeteer<1.1,>=1-
>nbconvert[webpdf]) (4.11.3)
Collecting websockets<11.0,>=10.0
  Downloading websockets-10.4-cp310-cp310-win_amd64.whl (101 kB)
     ---------------------------------------- 101.4/101.4 kB ? eta
0:00:00
Requirement already satisfied: appdirs<2.0.0,>=1.4.3 in c:\
programdata\anaconda3\lib\site-packages (from pyppeteer<1.1,>=1-
>nbconvert[webpdf]) (1.4.4)
Requirement already satisfied: soupsieve>1.2 in c:\programdata\
anaconda3\lib\site-packages (from beautifulsoup4->nbconvert[webpdf])
(2.3.2.post1)
Requirement already satisfied: six>=1.9.0 in c:\programdata\anaconda3\
lib\site-packages (from bleach->nbconvert[webpdf]) (1.16.0)
Requirement already satisfied: webencodings in c:\programdata\
```

```
anaconda3\lib\site-packages (from bleach->nbconvert[webpdf]) (0.5.1)
Requirement already satisfied: zipp>=0.5 in c:\programdata\anaconda3\
lib\site-packages (from importlib-metadata>=1.4->pyppeteer<1.1,>=1-
>nbconvert[webpdf]) (3.11.0)
Requirement already satisfied: attrs>=17.4.0 in c:\programdata\
anaconda3\lib\site-packages (from jsonschema>=2.6->nbformat>=5.1-
>nbconvert[webpdf]) (22.1.0)
Requirement already satisfied: pyrsistent!=0.17.0,!=0.17.1,!
=0.17.2,>=0.14.0 in c:\programdata\anaconda3\lib\site-packages (from
jsonschema>=2.6->nbformat>=5.1->nbconvert[webpdf]) (0.18.0)
Requirement already satisfied: pyzmq>=23.0 in c:\programdata\
anaconda3\lib\site-packages (from jupyter-client>=6.1.5-
>nbclient>=0.5.0->nbconvert[webpdf]) (23.2.0)
Requirement already satisfied: tornado>=6.0 in c:\programdata\
anaconda3\lib\site-packages (from jupyter-client>=6.1.5-
>nbclient>=0.5.0->nbconvert[webpdf]) (6.1)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\
programdata\anaconda3\lib\site-packages (from jupyter-client>=6.1.5-
>nbclient>=0.5.0->nbconvert[webpdf]) (2.8.2)
Requirement already satisfied: colorama in c:\programdata\anaconda3\
lib\site-packages (from tqdm<5.0.0,>=4.42.1->pyppeteer<1.1,>=1-
>nbconvert[webpdf]) (0.4.6)
Installing collected packages: pyee, websockets, pyppeteer
Successfully installed pyee-8.2.2 pyppeteer-1.0.2 websockets-10.4
Note: you may need to restart the kernel to use updated packages.

  WARNING: The script pyppeteer-install.exe is installed in 'C:\Users\
Ayush\AppData\Roaming\Python\Python310\Scripts' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress
this warning, use --no-warn-script-location.
```

```
jupyter nbconvert --to webpdf --allow-chromium-download your-notebook-
file.ipynb
```

```
  Cell In[2], line 1
    jupyter nbconvert --to webpdf --allow-chromium-download your-
notebook-file.ipynb
            ^
SyntaxError: invalid syntax


jupyter nbconvert --to webpdf --allow-chromium-download your-notebook-
file.ipynb
```

```
  Cell In[4], line 1
    jupyter nbconvert --to webpdf --allow-chromium-download your-
notebook-file.ipynb
            ^
SyntaxError: invalid syntax
```

```
import pandas as pd

df = pd.read_csv(r"C:\Users\Ayush\Downloads\pandaswheather.Zip")
df
```

```
     MinTemp  MaxTemp  Rainfall  Evaporation  Sunshine WindGustDir  \
0        8.0     24.3       0.0          3.4       6.3          NW
1       14.0     26.9       3.6          4.4       9.7         ENE
2       13.7     23.4       3.6          5.8       3.3          NW
3       13.3     15.5      39.8          7.2       9.1          NW
4        7.6     16.1       2.8          5.6      10.6         SSE
..       ...      ...       ...          ...       ...         ...
361      9.0     30.7       0.0          7.6      12.1         NNW
362      7.1     28.4       0.0         11.6      12.7           N
363     12.5     19.9       0.0          8.4       5.3         ESE
364     12.5     26.9       0.0          5.0       7.1          NW
365     12.3     30.2       0.0          6.0      12.6          NW

     WindGustSpeed WindDir9am WindDir3pm  WindSpeed9am  ...
Humidity3pm  \
0             30.0         SW         NW           6.0  ...
29
1             39.0          E          W           4.0  ...
36
2             85.0          N        NNE           6.0  ...
69
3             54.0        WNW          W          30.0  ...
56
4             50.0        SSE        ESE          20.0  ...
49
..             ...        ...        ...           ...  ...           .
..
361           76.0        SSE         NW           7.0  ...
15
362           48.0        NNW        NNW           2.0  ...
22
363           43.0        ENE        ENE          11.0  ...
47
364           46.0        SSW        WNW           6.0  ...
39
365           78.0         NW        WNW          31.0  ...
13

     Pressure9am  Pressure3pm  Cloud9am  Cloud3pm  Temp9am  Temp3pm  \
0         1019.7       1015.0         7         7     14.4     23.6
1         1012.4       1008.4         5         3     17.5     25.7
2         1009.5       1007.2         8         7     15.4     20.2
3         1005.5       1007.0         2         7     13.5     14.1
4         1018.3       1018.5         7         7     11.1     15.4
..           ...          ...       ...       ...      ...      ...
```

```
361       1016.1       1010.8         1         3     20.4     30.0
362       1020.0       1016.9         0         1     17.2     28.2
363       1024.0       1022.8         3         2     14.5     18.3
364       1021.0       1016.2         6         7     15.8     25.9
365       1009.6       1009.2         1         1     23.8     28.6

     RainToday  RISK_MM RainTomorrow
0           No      3.6          Yes
1          Yes      3.6          Yes
2          Yes     39.8          Yes
3          Yes      2.8          Yes
4          Yes      0.0           No
..         ...      ...          ...
361         No      0.0           No
362         No      0.0           No
363         No      0.0           No
364         No      0.0           No
365         No      0.0           No

[366 rows x 22 columns]
```

```
df['MaxTemp'].max()
```

```
35.8
```

give minimum temp of all days when it rained

```
df['MinTemp'][df['RainToday']=='Yes']
```

```
1      14.0
2      13.7
3      13.3
4       7.6
9       8.4
       ...
327     7.8
338    14.4
339    10.3
341     0.3
348    11.9
Name: MinTemp, Length: 66, dtype: float64
```

average pressure.This may not be exact because some of cells of excel sheet may be empty due to non availability of data of that particular date.so we need to clean the data to get exact result,the process known as "Data Cleaning" or "Data Munging".so,we place zero in empty places.

```
df["Pressure9am"].mean()
```

```
1019.7090163934428
```

```
df.fillna(0,inplace=True)
df["Pressure9am"].mean()
```

```
1019.7090163934428
```

```
whether_data={
    'day':['1/1/2017','1/2/2017','1/3/2017','1/4/2017','1/5/2017'],
    'Temperature':["32.4","56.9","32.9","24","45"],
    'windspeed':["102.3","303.9","1","2","3"]
}
df=pd.DataFrame(whether_data)
df
```

```
        day Temperature windspeed
0  1/1/2017        32.4     102.3
1  1/2/2017        56.9     303.9
2  1/3/2017        32.9         1
3  1/4/2017          24         2
4  1/5/2017          45         3
```

```
df.shape
```

```
(5, 3)
```

```
 rows,columns=df.shape
```

```
rows
```

```
5
```

```
df.head()
```

```
        day Temperature windspeed
0  1/1/2017        32.4     102.3
1  1/2/2017        56.9     303.9
2  1/3/2017        32.9         1
3  1/4/2017          24         2
4  1/5/2017          45         3
```

```
df.head(2)
```

```
        day Temperature windspeed
0  1/1/2017        32.4     102.3
1  1/2/2017        56.9     303.9
```

```
df.tail()
```

```
        day Temperature windspeed
0  1/1/2017        32.4     102.3
1  1/2/2017        56.9     303.9
2  1/3/2017        32.9         1
```

```
3  1/4/2017          24         2
4  1/5/2017          45         3
```

```
df.tail(1)
```

```
        day Temperature windspeed
4  1/5/2017          45         3
```

```
df[2:5]
```

```
        day Temperature windspeed
2  1/3/2017        32.9         1
3  1/4/2017          24         2
4  1/5/2017          45         3
```

```
df[:]
```

```
        day Temperature windspeed
0  1/1/2017        32.4     102.3
1  1/2/2017        56.9     303.9
2  1/3/2017        32.9         1
3  1/4/2017          24         2
4  1/5/2017          45         3
```

```
df.columns
```

```
Index(['day', 'Temperature', 'windspeed'], dtype='object')
```

```
df.day
```

```
0    1/1/2017
1    1/2/2017
2    1/3/2017
3    1/4/2017
4    1/5/2017
Name: day, dtype: object
```

```
df.Temperature
```

```
0    32.4
1    56.9
2    32.9
3      24
4      45
Name: Temperature, dtype: object
```

```
type(df['day'])
```

```
pandas.core.series.Series
```

```
df[["day","windspeed"]]
```

```
        day windspeed
0  1/1/2017     102.3
1  1/2/2017     303.9
2  1/3/2017         1
3  1/4/2017         2
4  1/5/2017         3
```

```
df["Temperature"].max()
```

```
'56.9'
```

```
df["Temperature"].min()
```

```
'24'
```

```
df["Temperature"].mean()
```

```
---------------------------------------------------------------------------
-----
ValueError                                Traceback (most recent call
last)
File C:\ProgramData\anaconda3\lib\site-packages\pandas\core\
nanops.py:1630, in _ensure_numeric(x)
   1629 try:
-> 1630     x = float(x)
   1631 except (TypeError, ValueError):
   1632     # e.g. "1+1j" or "foo"

ValueError: could not convert string to float: '32.456.932.92445'

During handling of the above exception, another exception occurred:

ValueError                                Traceback (most recent call
last)
File C:\ProgramData\anaconda3\lib\site-packages\pandas\core\
nanops.py:1634, in _ensure_numeric(x)
   1633 try:
-> 1634     x = complex(x)
   1635 except ValueError as err:
   1636     # e.g. "foo"

ValueError: complex() arg is a malformed string

The above exception was the direct cause of the following exception:

TypeError                                 Traceback (most recent call
last)
Cell In[32], line 1
----> 1 df["Temperature"].mean()

File C:\ProgramData\anaconda3\lib\site-packages\pandas\core\
```

```
generic.py:11847, in
NDFrame._add_numeric_operations.<locals>.mean(self, axis, skipna,
level, numeric_only, **kwargs)
  11829 @doc(
  11830     _num_doc,
  11831     desc="Return the mean of the values over the requested
axis.",
   (...)
  11845     **kwargs,
  11846 ):
> 11847     return NDFrame.mean(self, axis, skipna, level,
numeric_only, **kwargs)

File C:\ProgramData\anaconda3\lib\site-packages\pandas\core\
generic.py:11401, in NDFrame.mean(self, axis, skipna, level,
numeric_only, **kwargs)
  11393 def mean(
  11394     self,
  11395     axis: Axis | None | lib.NoDefault = lib.no_default,
   (...)
  11399     **kwargs,
  11400 ) -> Series | float:
> 11401     return self._stat_function(
  11402         "mean", nanops.nanmean, axis, skipna, level,
numeric_only, **kwargs
  11403     )

File C:\ProgramData\anaconda3\lib\site-packages\pandas\core\
generic.py:11353, in NDFrame._stat_function(self, name, func, axis,
skipna, level, numeric_only, **kwargs)
  11343     warnings.warn(
  11344         "Using the level keyword in DataFrame and Series
aggregations is "
  11345         "deprecated and will be removed in a future version.
Use groupby "
   (...)
  11348         stacklevel=find_stack_level(),
  11349     )
  11350     return self._agg_by_level(
  11351         name, axis=axis, level=level, skipna=skipna,
numeric_only=numeric_only
  11352     )
> 11353 return self._reduce(
  11354     func, name=name, axis=axis, skipna=skipna,
numeric_only=numeric_only
  11355 )

File C:\ProgramData\anaconda3\lib\site-packages\pandas\core\
series.py:4816, in Series._reduce(self, op, name, axis, skipna,
numeric_only, filter_type, **kwds)
```

```
   4812     raise NotImplementedError(
   4813         f"Series.{name} does not implement {kwd_name}."
   4814     )
   4815 with np.errstate(all="ignore"):
-> 4816     return op(delegate, skipna=skipna, **kwds)

File C:\ProgramData\anaconda3\lib\site-packages\pandas\core\
nanops.py:93, in disallow.__call__.<locals>._f(*args, **kwargs)
     91 try:
     92     with np.errstate(invalid="ignore"):
---> 93         return f(*args, **kwargs)
     94 except ValueError as e:
     95     # we want to transform an object array
     96     # ValueError message to the more typical TypeError
     97     # e.g. this is normally a disallowed function on
     98     # object arrays that contain strings
     99     if is_object_dtype(args[0]):

File C:\ProgramData\anaconda3\lib\site-packages\pandas\core\
nanops.py:155, in bottleneck_switch.__call__.<locals>.f(values, axis,
skipna, **kwds)
    153         result = alt(values, axis=axis, skipna=skipna, **kwds)
    154 else:
--> 155     result = alt(values, axis=axis, skipna=skipna, **kwds)
    157 return result

File C:\ProgramData\anaconda3\lib\site-packages\pandas\core\
nanops.py:418, in _datetimelike_compat.<locals>.new_func(values, axis,
skipna, mask, **kwargs)
    415 if datetimelike and mask is None:
    416     mask = isna(values)
--> 418 result = func(values, axis=axis, skipna=skipna, mask=mask,
**kwargs)
    420 if datetimelike:
    421     result = _wrap_results(result, orig_values.dtype,
fill_value=iNaT)

File C:\ProgramData\anaconda3\lib\site-packages\pandas\core\
nanops.py:706, in nanmean(values, axis, skipna, mask)
    703     dtype_count = dtype
    705 count = _get_counts(values.shape, mask, axis,
dtype=dtype_count)
--> 706 the_sum = _ensure_numeric(values.sum(axis, dtype=dtype_sum))
    708 if axis is not None and getattr(the_sum, "ndim", False):
    709     count = cast(np.ndarray, count)

File C:\ProgramData\anaconda3\lib\site-packages\pandas\core\
nanops.py:1637, in _ensure_numeric(x)
   1634             x = complex(x)
   1635         except ValueError as err:
```

```
   1636             # e.g. "foo"
-> 1637             raise TypeError(f"Could not convert {x} to
numeric") from err
   1638 return x

TypeError: Could not convert 32.456.932.92445 to numeric
```

```
df["Temperature"].std()
```

```
---------------------------------------------------------------------
-----
TypeError                                 Traceback (most recent call
last)
Cell In[34], line 1
----> 1 df["Temperature"].std()

File C:\ProgramData\anaconda3\lib\site-packages\pandas\core\
generic.py:11717, in
NDFrame._add_numeric_operations.<locals>.std(self, axis, skipna,
level, ddof, numeric_only, **kwargs)
  11697 @doc(
  11698     _num_ddof_doc,
  11699     desc="Return sample standard deviation over requested
axis."
   (...)
  11715     **kwargs,
  11716 ):
> 11717     return NDFrame.std(self, axis, skipna, level, ddof,
numeric_only, **kwargs)

File C:\ProgramData\anaconda3\lib\site-packages\pandas\core\
generic.py:11305, in NDFrame.std(self, axis, skipna, level, ddof,
numeric_only, **kwargs)
  11296 def std(
  11297     self,
  11298     axis: Axis | None = None,
   (...)
  11303     **kwargs,
  11304 ) -> Series | float:
> 11305     return self._stat_function_ddof(
  11306         "std", nanops.nanstd, axis, skipna, level, ddof,
numeric_only, **kwargs
  11307     )

File C:\ProgramData\anaconda3\lib\site-packages\pandas\core\
generic.py:11266, in NDFrame._stat_function_ddof(self, name, func,
axis, skipna, level, ddof, numeric_only, **kwargs)
  11256     warnings.warn(
  11257         "Using the level keyword in DataFrame and Series
aggregations is "
```

```
  11258         "deprecated and will be removed in a future version.
Use groupby "
   (...)
  11261         stacklevel=find_stack_level(),
  11262     )
  11263     return self._agg_by_level(
  11264         name, axis=axis, level=level, skipna=skipna, ddof=ddof
  11265     )
> 11266 return self._reduce(
  11267     func, name, axis=axis, numeric_only=numeric_only,
skipna=skipna, ddof=ddof
  11268 )

File C:\ProgramData\anaconda3\lib\site-packages\pandas\core\
series.py:4816, in Series._reduce(self, op, name, axis, skipna,
numeric_only, filter_type, **kwds)
   4812     raise NotImplementedError(
   4813         f"Series.{name} does not implement {kwd_name}."
   4814     )
   4815 with np.errstate(all="ignore"):
-> 4816     return op(delegate, skipna=skipna, **kwds)

File C:\ProgramData\anaconda3\lib\site-packages\pandas\core\
nanops.py:155, in bottleneck_switch.__call__.<locals>.f(values, axis,
skipna, **kwds)
    153         result = alt(values, axis=axis, skipna=skipna, **kwds)
    154 else:
--> 155     result = alt(values, axis=axis, skipna=skipna, **kwds)
    157 return result

File C:\ProgramData\anaconda3\lib\site-packages\pandas\core\
nanops.py:906, in nanstd(values, axis, skipna, ddof, mask)
    903 orig_dtype = values.dtype
    904 values, mask, _, _, _ = _get_values(values, skipna, mask=mask)
--> 906 result = np.sqrt(nanvar(values, axis=axis, skipna=skipna,
ddof=ddof, mask=mask))
    907 return _wrap_results(result, orig_dtype)

File C:\ProgramData\anaconda3\lib\site-packages\pandas\core\
nanops.py:93, in disallow.__call__.<locals>._f(*args, **kwargs)
     91 try:
     92     with np.errstate(invalid="ignore"):
---> 93         return f(*args, **kwargs)
     94 except ValueError as e:
     95     # we want to transform an object array
     96     # ValueError message to the more typical TypeError
     97     # e.g. this is normally a disallowed function on
     98     # object arrays that contain strings
     99     if is_object_dtype(args[0]):
```

```
File C:\ProgramData\anaconda3\lib\site-packages\pandas\core\
nanops.py:155, in bottleneck_switch.__call__.<locals>.f(values, axis,
skipna, **kwds)
    153         result = alt(values, axis=axis, skipna=skipna, **kwds)
    154 else:
--> 155     result = alt(values, axis=axis, skipna=skipna, **kwds)
    157 return result

File C:\ProgramData\anaconda3\lib\site-packages\pandas\core\
nanops.py:966, in nanvar(values, axis, skipna, ddof, mask)
    964 if axis is not None:
    965     avg = np.expand_dims(avg, axis)
--> 966 sqr = _ensure_numeric((avg - values) ** 2)
    967 if mask is not None:
    968     np.putmask(sqr, mask, 0)

TypeError: unsupported operand type(s) for -: 'float' and 'str'
```

the above functions aren't working because they are float and not integers

```
whether_data={
    'day':['1/1/2017','1/2/2017','1/3/2017','1/4/2017','1/5/2017'],
    'Temperature':["32","56","32","24","45"],
    'windspeed':["14","30","19","28","38"]
}
df=pd.DataFrame(whether_data)
df
```

	day	Temperature	windspeed
0	1/1/2017	32	14
1	1/2/2017	56	30
2	1/3/2017	32	19
3	1/4/2017	24	28
4	1/5/2017	45	38

conditional data selection

```
df[df.windspeed>='32']
```

	day	Temperature	windspeed
4	1/5/2017	45	38

```
df[df.windspeed==df.windspeed.max()]
```

	day	Temperature	windspeed
4	1/5/2017	45	38

```
df[df.windspeed==df['windspeed'].max()]
```

```
        day Temperature windspeed
4  1/5/2017          45        38
```

printing day and temperature when temp. was maximum

```
df[["day","Temperature"]][df.windspeed==df['windspeed'].max()]
```

```
        day Temperature
4  1/5/2017          45
```

```
df.index
```

```
RangeIndex(start=0, stop=5, step=1)
```

```
df.set_index("day")
```

```
         Temperature windspeed
day
1/1/2017          32        14
1/2/2017          56        30
1/3/2017          32        19
1/4/2017          24        28
1/5/2017          45        38
```

```
df
```

```
        day Temperature windspeed
0  1/1/2017          32        14
1  1/2/2017          56        30
2  1/3/2017          32        19
3  1/4/2017          24        28
4  1/5/2017          45        38
```

```
df.loc[1]
```

```
day            1/2/2017
Temperature          56
windspeed            30
Name: 1, dtype: object
```

to change actual data set with index changed data set

```
df.set_index("day",inplace=True)
```

```
df.loc["1/2/2017"]
```

```
Temperature    56
windspeed      30
Name: 1/2/2017, dtype: object
```

Now lets reset index as it was

```
df.reset_index(inplace=True)

df
```

```
        day Temperature windspeed
0  1/1/2017          32        14
1  1/2/2017          56        30
2  1/3/2017          32        19
3  1/4/2017          24        28
4  1/5/2017          45        38
```

If two rows has same index "x" ,then on printing df.loc[x] both rows would be printed

```
weather_data=[
    ("1/1/2017","32" ,6,"rain"),
     ("1/2/2017",35,7,"sunny"),
     ("1/3/2017",28,2,"snow"),
]
df=pd.DataFrame (weather_data,
columns=["days","temperature","windspeed","event"])
df
```

```
       days temperature  windspeed  event
0  1/1/2017          32          6   rain
1  1/2/2017          35          7  sunny
2  1/3/2017          28          2   snow
```

```
weather_data=[

{"day":"1/1/2017","temperature":"32" ,"windspeed":6,"event":"rain"},

{"day":"1/2/2017","temperature":"35" ,"windspeed":7,"event":"sunny"},

{"day":"1/3/2017","temperature":"28" ,"windspeed":2,"event":"snow"}
]
df=pd.DataFrame(weather_data)
df
```

```
        day temperature  windspeed  event
0  1/1/2017          32          6   rain
1  1/2/2017          35          7  sunny
2  1/3/2017          28          2   snow
```

```
weather_data=[
    {"day":"1/1/2017"},

{"day":"1/2/2017","temperature":"35" ,"windspeed":7,"event":"sunny"},

{"day":"1/3/2017","temperature":"28" ,"windspeed":2,"event":"snow"}
```

```
]
df=pd.DataFrame(weather_data)
df
```

```
        day temperature  windspeed  event
0  1/1/2017         NaN        NaN    NaN
1  1/2/2017          35        7.0  sunny
2  1/3/2017          28        2.0   snow
```

```
df = pd.read_excel(r"C:\Users\Ayush\AppData\Roaming\Microsoft\AddIns\
weather.xlsx")
df
```

```
        row1    empty Unnamed: 2   Unnamed: 3 Unnamed: 4   Unnamed: 5
\
0    MinTemp  MaxTemp   Rainfall  Evaporation   Sunshine  WindGustDir

1          8     24.3          0          3.4        6.3           NW

2         14     26.9        3.6          4.4        9.7          ENE

3       13.7     23.4        3.6          5.8        3.3           NW

4       13.3     15.5       39.8          7.2        9.1           NW

..       ...      ...        ...          ...        ...          ...

362        9     30.7          0          7.6       12.1          NNW

363      7.1     28.4          0         11.6       12.7            N

364     12.5     19.9          0          8.4        5.3          ESE

365     12.5     26.9          0            5        7.1           NW

366     12.3     30.2          0            6       12.6           NW


        Unnamed: 6  Unnamed: 7  Unnamed: 8    Unnamed: 9  ...
Unnamed: 12  \
0    WindGustSpeed  WindDir9am  WindDir3pm  WindSpeed9am  ...
Humidity3pm
1               30          SW          NW             6  ...
29
2               39           E           W             4  ...
36
3               85           N         NNE             6  ...
69
4               54         WNW           W            30  ...
56
..             ...         ...         ...           ...  ...
```

```
...
362             76          SSE           NW              7  ...
15
363             48          NNW          NNW              2  ...
22
364             43          ENE          ENE             11  ...
47
365             46          SSW          WNW              6  ...
39
366             78           NW          WNW             31  ...
13

     Unnamed: 13  Unnamed: 14 Unnamed: 15 Unnamed: 16 Unnamed: 17
Unnamed: 18  \
0    Pressure9am  Pressure3pm    Cloud9am    Cloud3pm     Temp9am
Temp3pm
1         1019.7         1015           7           7        14.4
23.6
2         1012.4       1008.4           5           3        17.5
25.7
3         1009.5       1007.2           8           7        15.4
20.2
4         1005.5         1007           2           7        13.5
14.1
..           ...          ...         ...         ...         ...
...
362       1016.1       1010.8           1           3        20.4
30
363         1020       1016.9           0           1        17.2
28.2
364         1024       1022.8           3           2        14.5
18.3
365         1021       1016.2           6           7        15.8
25.9
366       1009.6       1009.2           1           1        23.8
28.6

    Unnamed: 19 Unnamed: 20   Unnamed: 21
0     RainToday     RISK_MM  RainTomorrow
1            No         3.6           Yes
2           Yes         3.6           Yes
3           Yes        39.8           Yes
4           Yes         2.8           Yes
..          ...         ...           ...
362          No           0            No
363          No           0            No
364          No           0            No
365          No           0            No
366          No           0            No
```

```
[367 rows x 22 columns]
```

Removing the toppest row as it dont contain data

```
df = pd.read_excel(r"C:\Users\Ayush\AppData\Roaming\Microsoft\AddIns\
weather.xlsx",skiprows=1)
df
```

```
     MinTemp  MaxTemp  Rainfall  Evaporation  Sunshine WindGustDir  \
0        8.0     24.3       0.0          3.4       6.3          NW
1       14.0     26.9       3.6          4.4       9.7         ENE
2       13.7     23.4       3.6          5.8       3.3          NW
3       13.3     15.5      39.8          7.2       9.1          NW
4        7.6     16.1       2.8          5.6      10.6         SSE
..       ...      ...       ...          ...       ...         ...
361      9.0     30.7       0.0          7.6      12.1         NNW
362      7.1     28.4       0.0         11.6      12.7           N
363     12.5     19.9       0.0          8.4       5.3         ESE
364     12.5     26.9       0.0          5.0       7.1          NW
365     12.3     30.2       0.0          6.0      12.6          NW

     WindGustSpeed WindDir9am WindDir3pm  WindSpeed9am  ...
Humidity3pm  \
0             30.0         SW         NW           6.0  ...
29
1             39.0          E          W           4.0  ...
36
2             85.0          N        NNE           6.0  ...
69
3             54.0        WNW          W          30.0  ...
56
4             50.0        SSE        ESE          20.0  ...
49
..             ...        ...        ...           ...  ...          .
..
361           76.0        SSE         NW           7.0  ...
15
362           48.0        NNW        NNW           2.0  ...
22
363           43.0        ENE        ENE          11.0  ...
47
364           46.0        SSW        WNW           6.0  ...
39
365           78.0         NW        WNW          31.0  ...
13

     Pressure9am  Pressure3pm  Cloud9am  Cloud3pm  Temp9am  Temp3pm  \
0         1019.7       1015.0         7         7     14.4     23.6
```

```
1          1012.4       1008.4          5          3     17.5     25.7
2          1009.5       1007.2          8          7     15.4     20.2
3          1005.5       1007.0          2          7     13.5     14.1
4          1018.3       1018.5          7          7     11.1     15.4
..            ...          ...        ...        ...      ...      ...
361        1016.1       1010.8          1          3     20.4     30.0
362        1020.0       1016.9          0          1     17.2     28.2
363        1024.0       1022.8          3          2     14.5     18.3
364        1021.0       1016.2          6          7     15.8     25.9
365        1009.6       1009.2          1          1     23.8     28.6

     RainToday  RISK_MM RainTomorrow
0           No      3.6          Yes
1          Yes      3.6          Yes
2          Yes     39.8          Yes
3          Yes      2.8          Yes
4          Yes      0.0           No
..         ...      ...          ...
361         No      0.0           No
362         No      0.0           No
363         No      0.0           No
364         No      0.0           No
365         No      0.0           No

[366 rows x 22 columns]
```

```
df = pd.read_excel(r"C:\Users\Ayush\AppData\Roaming\Microsoft\AddIns\
weather.xlsx",header=None,names=["new1","new2","new1","new2","new1","n
ew2","new1","new2","new1","new2","new1","new2","new1","new2","new1","n
ew2","new1","new2","new1","new2","new1"])
df
```

```
            new1      new2       new1.1    new2.1       new1.2
new2.2  \
row1       empty       NaN          NaN       NaN          NaN
NaN
MinTemp  MaxTemp  Rainfall  Evaporation  Sunshine  WindGustDir
WindGustSpeed
8           24.3         0          3.4       6.3           NW
30
14          26.9       3.6          4.4       9.7          ENE
39
13.7        23.4       3.6          5.8       3.3           NW
85
...          ...       ...          ...       ...          ...
...
9           30.7         0          7.6      12.1          NNW
76
7.1         28.4         0         11.6      12.7            N
48
```

```
12.5        19.9          0           8.4        5.3           ESE
43
12.5        26.9          0             5        7.1            NW
46
12.3        30.2          0             6       12.6            NW
78

              new1.3      new2.3         new1.4         new2.4  ...
new2.5  \
row1             NaN         NaN            NaN            NaN  ...
NaN
MinTemp  WindDir9am  WindDir3pm  WindSpeed9am  WindSpeed3pm  ...
Humidity3pm
8                 SW          NW             6             20  ...
29
14                 E           W             4             17  ...
36
13.7               N         NNE             6              6  ...
69
...              ...         ...           ...            ...  ...
...
9                SSE          NW             7             50  ...
15
7.1              NNW         NNW             2             19  ...
22
12.5             ENE         ENE            11              9  ...
47
12.5             SSW         WNW             6             28  ...
39
12.3              NW         WNW            31             35  ...
13

               new1.6       new2.6    new1.7    new2.7   new1.8
new2.8  \
row1             NaN          NaN       NaN       NaN      NaN
NaN
MinTemp  Pressure9am  Pressure3pm  Cloud9am  Cloud3pm  Temp9am
Temp3pm
8             1019.7         1015         7         7     14.4
23.6
14            1012.4       1008.4         5         3     17.5
25.7
13.7          1009.5       1007.2         8         7     15.4
20.2
...              ...          ...       ...       ...      ...      ..
.
9             1016.1       1010.8         1         3     20.4
30
7.1             1020       1016.9         0         1     17.2
```

```
28.2
12.5            1024       1022.8         3         2     14.5
18.3
12.5            1021       1016.2         6         7     15.8
25.9
12.3          1009.6       1009.2         1         1     23.8
28.6

            new1.9   new2.9       new1.10
row1           NaN      NaN           NaN
MinTemp  RainToday  RISK_MM  RainTomorrow
8               No      3.6           Yes
14             Yes      3.6           Yes
13.7           Yes     39.8           Yes
...            ...      ...           ...
9               No        0            No
7.1             No        0            No
12.5            No        0            No
12.5            No        0            No
12.3            No        0            No

[368 rows x 21 columns]
```

```
df = pd.read_excel(r"C:\Users\Ayush\AppData\Roaming\Microsoft\AddIns\
weather.xlsx",nrows=3)
df
```

```
      row1    empty Unnamed: 2   Unnamed: 3 Unnamed: 4   Unnamed: 5  \
0  MinTemp  MaxTemp   Rainfall  Evaporation   Sunshine  WindGustDir
1        8     24.3          0          3.4        6.3           NW
2       14     26.9        3.6          4.4        9.7          ENE

      Unnamed: 6  Unnamed: 7  Unnamed: 8    Unnamed: 9  ...  Unnamed:
12  \
0  WindGustSpeed  WindDir9am  WindDir3pm  WindSpeed9am  ...
Humidity3pm
1             30          SW          NW             6  ...
29
2             39           E           W             4  ...
36

   Unnamed: 13  Unnamed: 14 Unnamed: 15 Unnamed: 16 Unnamed: 17
Unnamed: 18  \
0  Pressure9am  Pressure3pm    Cloud9am    Cloud3pm     Temp9am
Temp3pm
1       1019.7         1015           7           7        14.4
23.6
2       1012.4       1008.4           5           3        17.5
25.7
```

```
  Unnamed: 19 Unnamed: 20   Unnamed: 21
0   RainToday     RISK_MM  RainTomorrow
1          No         3.6           Yes
2         Yes         3.6           Yes

[3 rows x 22 columns]
```

```
import pandas as pd

df = pd.read_excel(r"C:\Users\Ayush\Downloads\weatherfinal.xlsx")
df
```

```
   MinTemp        MaxTemp  Rainfall  Evaporation
0      8.0           24.3       NaN          3.4
1     14.0  Not available       3.6          4.4
2     13.7           23.4      -1.0          5.8
3     13.3           15.5      39.8          7.2
4      7.6           16.1      -1.0          5.6
5      6.2           16.9       0.0          5.8
6      6.1           18.2       0.2          4.2
```

Not avilable, NAN and -1(rainfall cant be negative) were replaced by NAN in below

```
df = pd.read_excel(r"C:\Users\Ayush\Downloads\
weatherfinal.xlsx",na_values=["Not available","NaN",-1])
df
```

```
   MinTemp  MaxTemp  Rainfall  Evaporation
0      8.0     24.3       NaN          3.4
1     14.0      NaN       3.6          4.4
2     13.7     23.4       NaN          5.8
3     13.3     15.5      39.8          7.2
4      7.6     16.1       NaN          5.6
5      6.2     16.9       0.0          5.8
6      6.1     18.2       0.2          4.2
```

```
df
```

```
   MinTemp        MaxTemp  Rainfall  Evaporation
0      8.0           24.3       NaN          3.4
1     14.0  Not available       3.6          4.4
2     13.7           23.4      -1.0          5.8
3     13.3           15.5      39.8          7.2
4      7.6           16.1      -1.0          5.6
5      6.2           16.9       0.0          5.8
6      6.1           18.2       0.2          4.2
```

```
df = pd.read_excel(r"C:\Users\Ayush\Downloads\
weatherfinal.xlsx",na_values={
    "MaxTemp":["Not available"],
```

```
    "Rainfall":["NAN",-1],
})
df
```

	MinTemp	MaxTemp	Rainfall	Evaporation
0	8.0	24.3	NaN	3.4
1	14.0	NaN	3.6	4.4
2	13.7	23.4	NaN	5.8
3	13.3	15.5	39.8	7.2
4	7.6	16.1	NaN	5.6
5	6.2	16.9	0.0	5.8
6	6.1	18.2	0.2	4.2

lets copy this df

```
df.to_csv("new.csv")

df
```

	MinTemp	MaxTemp	Rainfall	Evaporation
0	8.0	24.3	NaN	3.4
1	14.0	NaN	3.6	4.4
2	13.7	23.4	NaN	5.8
3	13.3	15.5	39.8	7.2
4	7.6	16.1	NaN	5.6
5	6.2	16.9	0.0	5.8
6	6.1	18.2	0.2	4.2

```
df.to_csv("new.csv",index=False)
this will give no index inside  new.csv,
so now if you open new.csv in excel it will give no index

def convert_MAXTEMP_cell(cell):
    if cell=="Not available":
        return "340"
    return cell

df = pd.read_excel(r"C:\Users\Ayush\Downloads\weatherfinal.xlsx",
                   converters={"MaxTemp":convert_MAXTEMP_cell})
df
```

	MinTemp	MaxTemp	Rainfall	Evaporation
0	8.0	24.3	NaN	3.4
1	14.0	340	3.6	4.4
2	13.7	23.4	-1.0	5.8
3	13.3	15.5	39.8	7.2
4	7.6	16.1	-1.0	5.6
5	6.2	16.9	0.0	5.8
6	6.1	18.2	0.2	4.2

```
df
```

```
   MinTemp MaxTemp  Rainfall  Evaporation
0      8.0    24.3       NaN          3.4
1     14.0     340       3.6          4.4
2     13.7    23.4      -1.0          5.8
3     13.3    15.5      39.8          7.2
4      7.6    16.1      -1.0          5.6
5      6.2    16.9       0.0          5.8
6      6.1    18.2       0.2          4.2
```

```
df.to_excel("weatherfinal.xlsx",sheet_name="sheet1",startrow=1,startco
l=2)
checkit,it converts dataform to excel format
```

```
with pd.ExcelWriter('stocks_weather.Xlsx') as writer:
    df_stocks.to_excel(writer,sheet_name="stocks")
    df_weather.to_excel(writer,sheet_name="weather")
```

```
---------------------------------------------------------------------
-----
KeyError                                  Traceback (most recent call
last)
File C:\ProgramData\anaconda3\lib\site-packages\pandas\io\excel\
_base.py:1146, in ExcelWriter.__new__(cls, path, engine, date_format,
datetime_format, mode, storage_options, if_sheet_exists,
engine_kwargs, **kwargs)
   1145     if engine == "auto":
-> 1146         engine = get_default_engine(ext, mode="writer")
   1147 except KeyError as err:

File C:\ProgramData\anaconda3\lib\site-packages\pandas\io\excel\
_util.py:85, in get_default_engine(ext, mode)
     84         _default_writers["xlsx"] = "xlsxwriter"
---> 85     return _default_writers[ext]
     86 else:

KeyError: 'Xlsx'

The above exception was the direct cause of the following exception:

ValueError                                Traceback (most recent call
last)
Cell In[112], line 1
----> 1 with pd.ExcelWriter('stocks_weather.Xlsx') as writer:
      2     df_stocks.to_excel(writer,sheet_name="stocks")
      3     df_weather.to_excel(writer,sheet_name="weather")

File C:\ProgramData\anaconda3\lib\site-packages\pandas\io\excel\
_base.py:1148, in ExcelWriter.__new__(cls, path, engine, date_format,
datetime_format, mode, storage_options, if_sheet_exists,
engine_kwargs, **kwargs)
```

```
   1146             engine = get_default_engine(ext, mode="writer")
   1147     except KeyError as err:
-> 1148         raise ValueError(f"No engine for filetype: '{ext}'") 
from err
   1150 if engine == "xlwt":
   1151     xls_config_engine = config.get_option(
   1152         "io.excel.xls.writer", silent=True
   1153     )

ValueError: No engine for filetype: 'Xlsx'
```

we can make two df into two sheets in a single xlsx file

```
df
```

	MinTemp	MaxTemp	Rainfall	Evaporation
0	8.0	24.3	NaN	3.4
1	14.0	340	3.6	4.4
2	13.7	23.4	-1.0	5.8
3	13.3	15.5	39.8	7.2
4	7.6	16.1	-1.0	5.6
5	6.2	16.9	0.0	5.8
6	6.1	18.2	0.2	4.2

EXCEPTION HANDLING IN PYTHON

```
x=input("Enter First Number: ")
y=input("Enter Second Number: ")
z=int(x)/int(y)
print("Division is : ",z)
```

```
Enter First Number: 3
Enter Second Number: 0

---------------------------------------------------------------------------
-----
ZeroDivisionError                         Traceback (most recent call
last)
Cell In[1], line 3
      1 x=input("Enter First Number: ")
      2 y=input("Enter Second Number: ")
----> 3 z=int(x)/int(y)
      4 print("Division is : ",z)

ZeroDivisionError: division by zero
```

our program got terminated at the point division become invalid and didn't ran further to give "print" statement output.so that's why we need exception handling to prevent our program from terminating in middle and to make it to reach the destination.

```
x=input("Enter First Number: ")
y=input("Enter Second Number: ")
try:
   z=int(x)/int(y)
except Exception as e:
   print('exception ocuured : ',e)
   print(e)
   z=None
print("Division is : ",z)
print("task done master!!!")
```

```
Enter First Number: 4
Enter Second Number: 0
exception ocuured :  division by zero
division by zero
Division is :  None
task done master!!!
```

here,program didn't crashed in between...

```
x=input("Enter First Number: ")
y=input("Enter Second Number: ")
try:
   z=x/int(y)
except Exception as e:
   z=None
print("Division is : ",z)
print("task done master!!!")
```

```
Enter First Number: 1
Enter Second Number: 0
Division is :  None
task done master!!!
```

this is handling multiple exceptions ,no typecast of "x" to int and other is division by zero.

```
x=input("Enter First Number: ")
y=input("Enter Second Number: ")
try:
   z=x/int(y)
except Exception as e:
   print('exception type : ',type(e).__name__)
   z=None
print("Division is : ",z)
print("task done master!!!")
```

```
Enter First Number: 1
Enter Second Number: 0
exception type :  TypeError
Division is :  None
task done master!!!
```

```
x=input("Enter First Number: ")
y=input("Enter Second Number: ")
try:
```

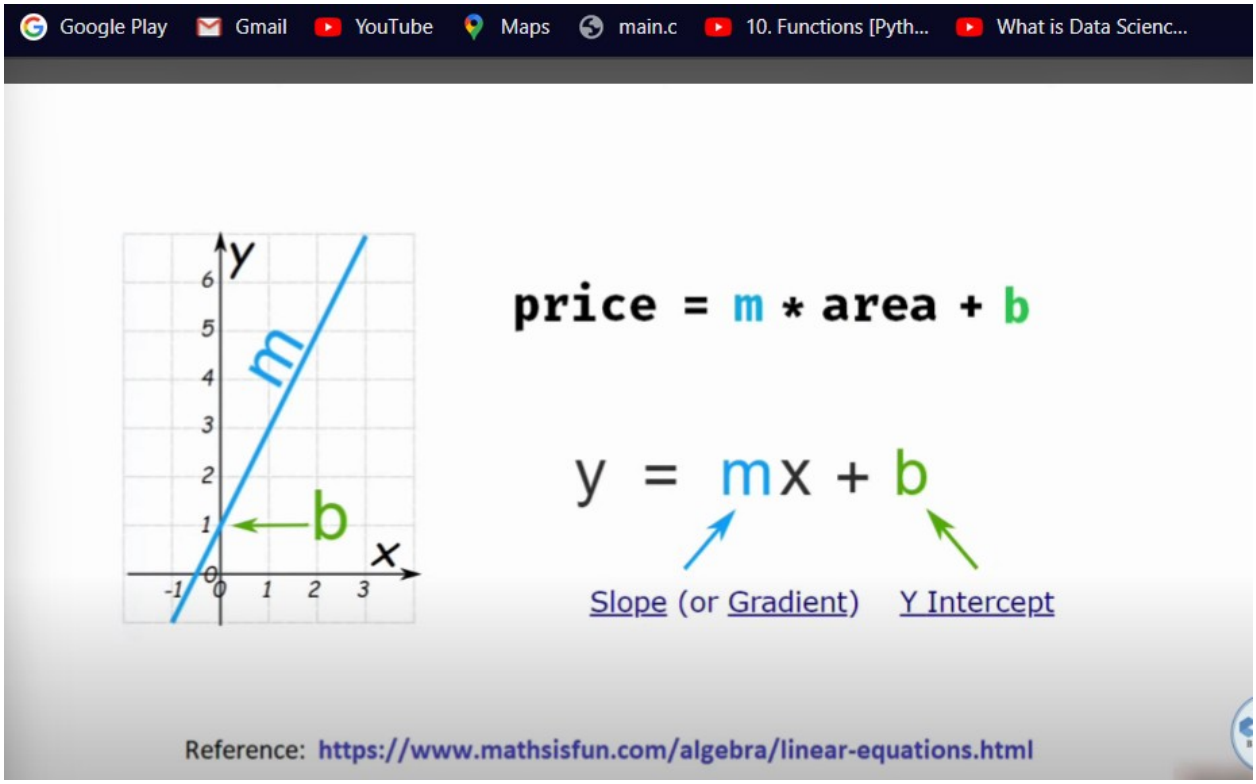
```
   z=x/int(y)
except ZeroDivisionError as e:
   print('division by zero ')
   z=None
except Exception as e:
    print("exception type: ",type(e).__name__)
    z=None
print("Division is : ",z)
print("task done master!!!")
```

```
Enter First Number: 1
Enter Second Number: 0
exception type:  TypeError
Division is :  None
task done master!!!
```

```
x=input("Enter First Number: ")
y=input("Enter Second Number: ")
try:
   z=x/int(y)
except ZeroDivisionError as e:
   print('division by zero ')
   z=None
except TypeError as e:
    print('type error exception')
    z=None
print("Division is : ",z)
print("task done master!!!")
```

```
Enter First Number: 1
Enter Second Number: 0
type error exception
Division is :  None
task done master!!!
```

Machine Learning (Linear Regression)

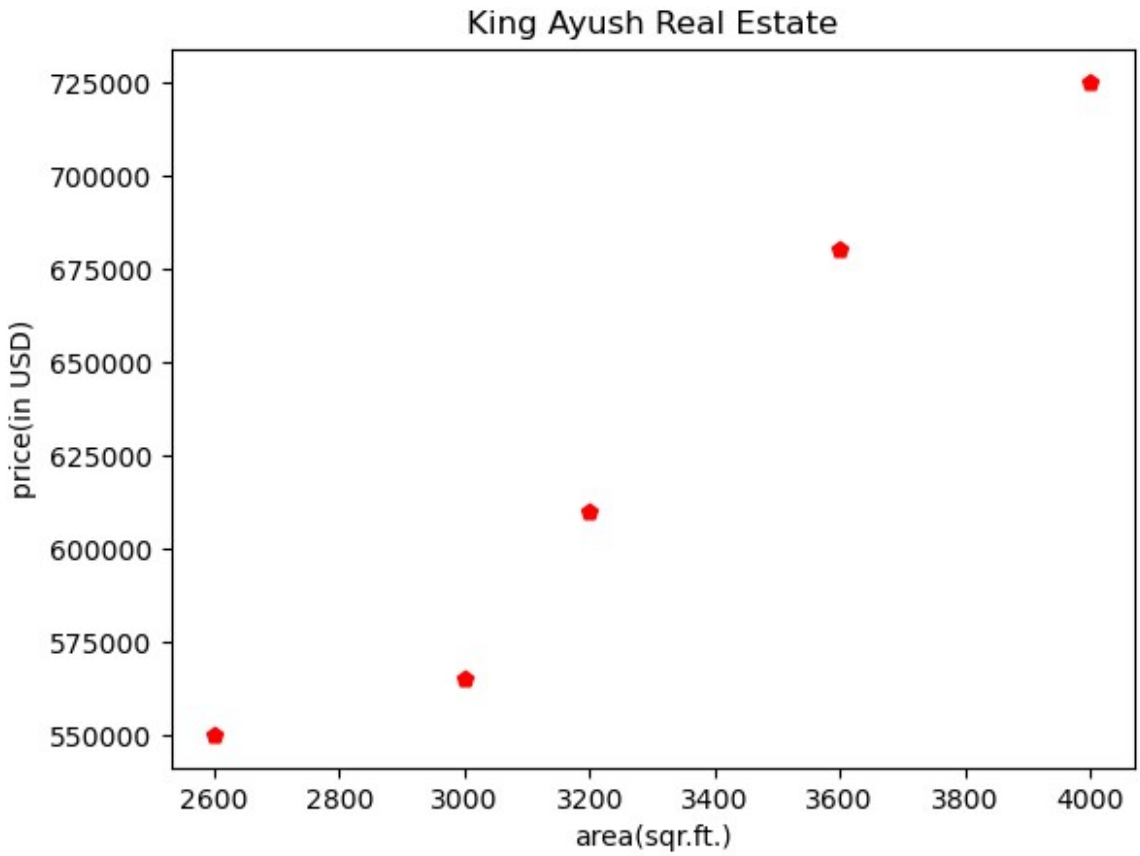



```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import linear_model

df=pd.read_csv(r"C:\Users\Ayush\OneDrive\Documents\araerprice.csv")
df
```

	area	price
0	2600	550000
1	3000	565000
2	3200	610000
3	3600	680000
4	4000	725000

```
%matplotlib inline
plt.title('King Ayush Real Estate')
plt.xlabel('area(sqr.ft.)')
plt.ylabel('price(in USD)')

plt.scatter(df.area,df.price,color='red',marker='p')
```

```
<matplotlib.collections.PathCollection at 0x17866569960>
```



first we need to create linear regression object,from sklearn(scikit learn) python module we have alredy imported linear model

```
reg = linear_model.LinearRegression()
reg.fit(df[['area']],df.price)
```

```
LinearRegression()
```

```
reg.predict([[3300]])
```

```
C:\ProgramData\anaconda3\lib\site-packages\sklearn\base.py:420:
UserWarning: X does not have valid feature names, but LinearRegression
was fitted with feature names
  warnings.warn(
```

```
array([628715.75342466])
```

```
reg.coef_
```

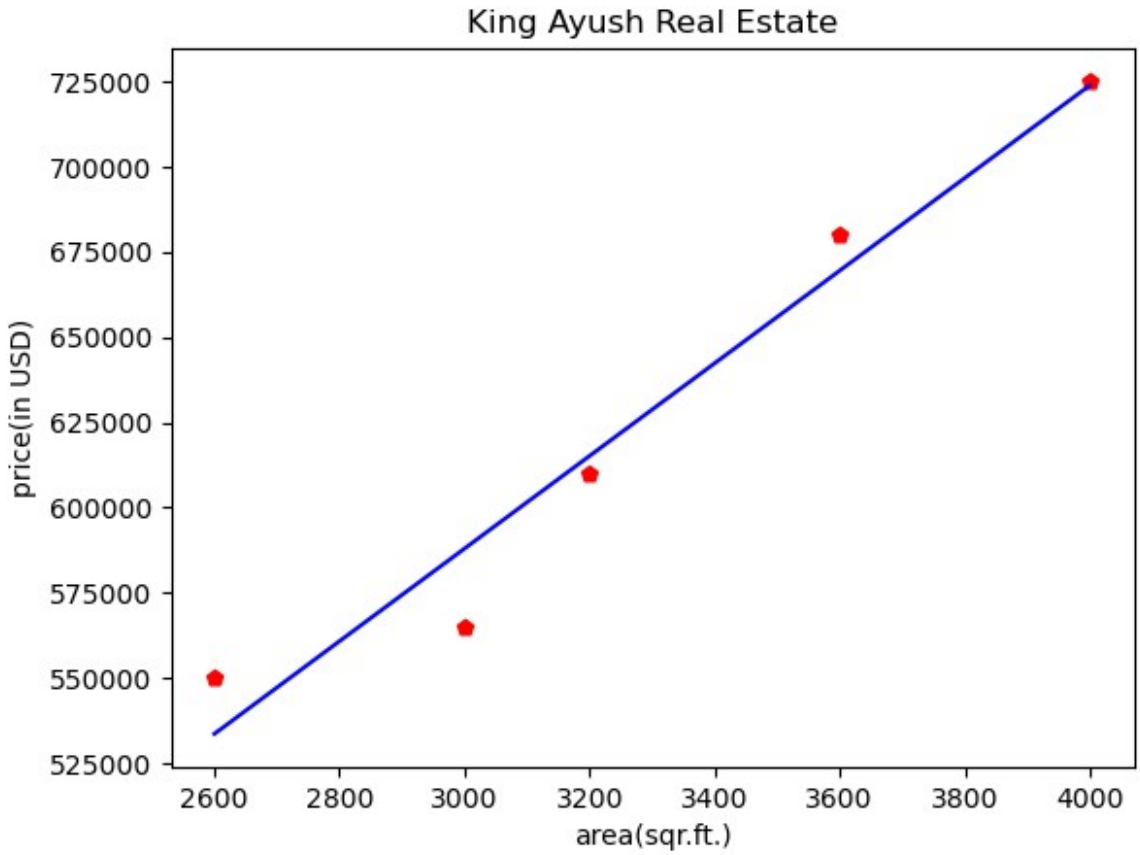
```
array([135.78767123])
```

```
reg.intercept_
```

```
180616.43835616432
```

```
%matplotlib inline
plt.title('King Ayush Real Estate')
plt.xlabel('area(sqr.ft.)')
plt.ylabel('price(in USD)')
plt.scatter(df.area,df.price,color='red',marker='p')
plt.plot(df.area,reg.predict(df[['area']]),color='blue')
```

```
[<matplotlib.lines.Line2D at 0x178665a9bd0>]
```




```
135.78767123*3300+180616.43835616432
```

```
628715.7534151643
```

```
reg.predict([[3200]])
```

```
C:\ProgramData\anaconda3\lib\site-packages\sklearn\base.py:420:
UserWarning: X does not have valid feature names, but LinearRegression
was fitted with feature names
  warnings.warn(
```

```
array([615136.98630137])
```

```
d = pd.read_csv(r"C:\Users\Ayush\OneDrive\Documents\areas.csv")
d
```

```
    area
0   4586
1   1524
2   7856
3   2456
4   4896
5   2456
6   7854
7   2365
8   1200
9   3245
10  4800
```

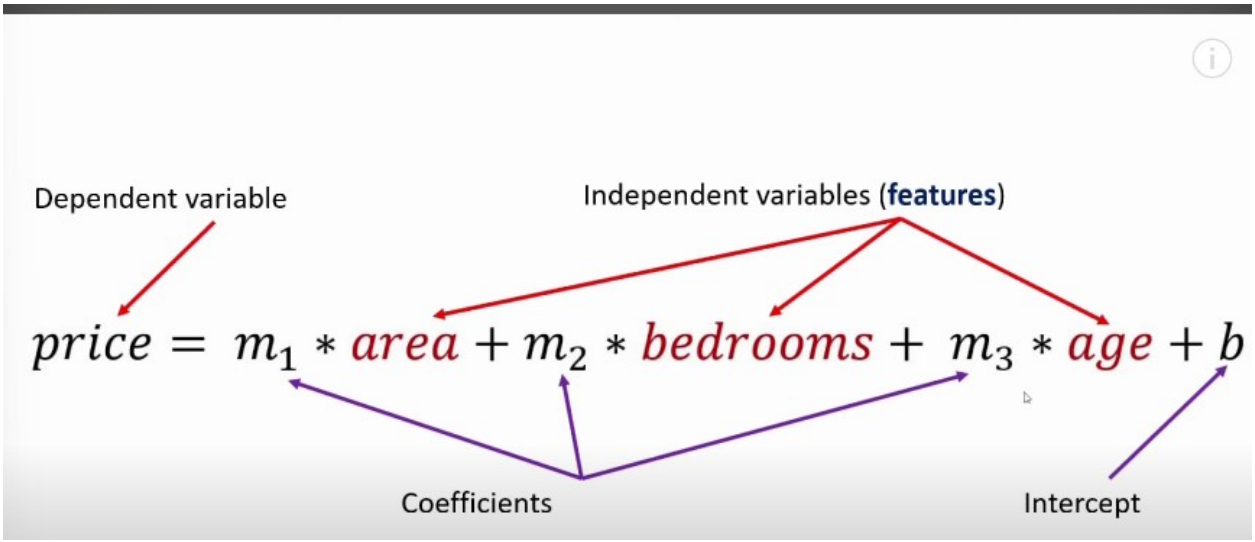
```
p=reg.predict(d)
d['prices']=p
```

```
d
```

```
d.to_csv("prediction.csv")
```

the above will export prices and areas to prediction csv file

LINEAR REG> WITH MULTIPLE VARIABLE OR MULTIVARIATE REGRESSION



Y=m1*x1+m2*x2+m3*x3+b

```
import pandas as pd
import numpy as np
from sklearn import linear_model

df=pd.read_csv(r"C:\Users\Ayush\OneDrive\Documents\areaageprice.csv")
df
```

	area	bedrooms	age	price
0	2600	3.0	20	550000
1	3000	4.0	15	565000
2	3200	NaN	18	610000
3	3600	3.0	30	595000
4	4000	5.0	8	760000

```
import math
median_bedrooms=math.floor(df.bedrooms.median())
median_bedrooms
```

```
3
```

```
df.bedrooms=df.bedrooms.fillna(median_bedrooms)
df
```

	area	bedrooms	age	price
0	2600	3.0	20	550000
1	3000	4.0	15	565000
2	3200	3.0	18	610000
3	3600	3.0	30	595000
4	4000	5.0	8	760000

```
reg=linear_model.LinearRegression()
reg.fit(df[['area','bedrooms','age']],df.price)
```

```
LinearRegression()
```

```
reg.coef_
```

```
array([   137.25, -26025.  ,  -6825.  ])
```

```
reg.intercept_
```

```
383724.9999999998
```

```
reg.predict([[3000,3,40]])
```

```
C:\ProgramData\anaconda3\lib\site-packages\sklearn\base.py:420:
UserWarning: X does not have valid feature names, but LinearRegression
was fitted with feature names
  warnings.warn(
```

```
array([444400.])
```

```
df=pd.read_csv(r"C:\Users\Ayush\Downloads\salarysheet2.csv")
df
```

	experience	testscore	interview	salary
0	NaN	8.0	9	50000
1	NaN	8.0	6	45000
2	five	6.0	7	60000
3	two	10.0	10	65000
4	seven	9.0	6	70000
5	three	7.0	10	62000
6	ten	NaN	7	72000
7	eleven	7.0	8	80000

```
from word2number import w2n
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
Cell In[125], line 1
----> 1 from word2number import w2n

ModuleNotFoundError: No module named 'word2number'
```

```
pip install word2number
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: word2number in c:\users\ayush\appdata\roaming\python\python310\site-packages (1.1)
Note: you may need to restart the kernel to use updated packages.
```

```
from word2number import w2n
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
Cell In[127], line 1
----> 1 from word2number import w2n

ModuleNotFoundError: No module named 'word2number'
```

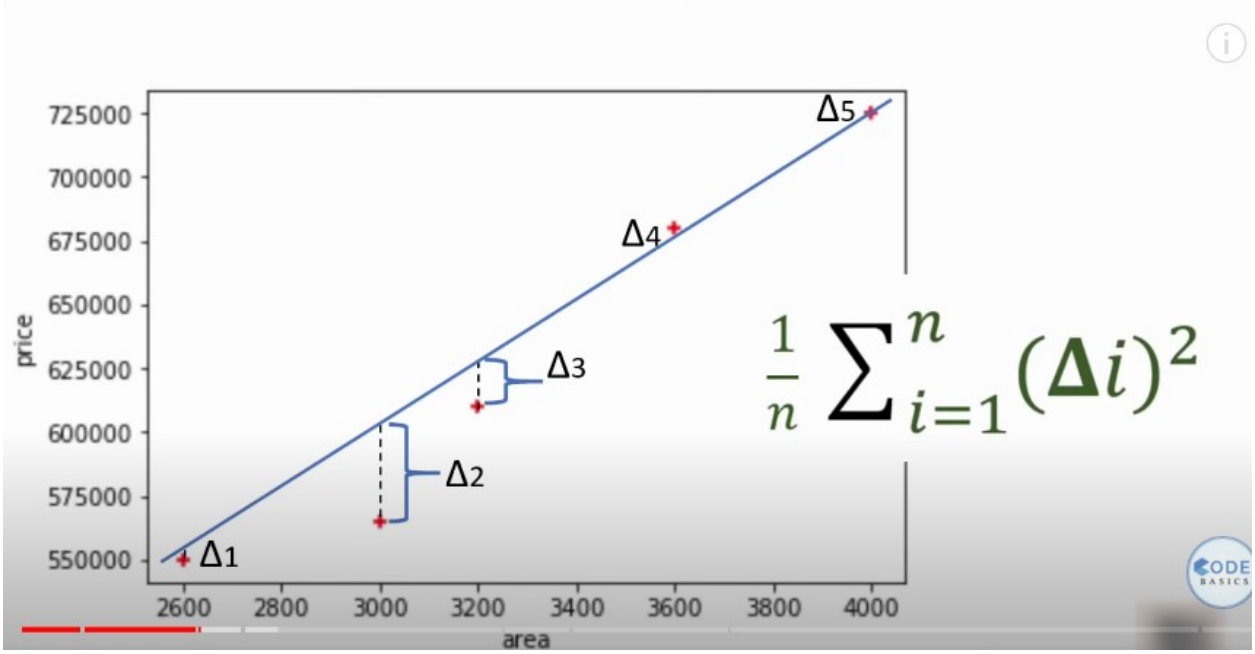
```
import word2number
import sys
print(sys.executable)
```

```
!pip install word2number
import word2number
import sys
print(sys.executable)
```

```
!pip uninstall word2number
!pip install word2number

from word2number import w2n

t='one'
d=w2n.word_to_num(t)
print(d)

from word2number import w2n
t = 'one'
d = w2n.word_to_num(t)
print(d)

word_to_num_dict = {
    'zero': 0, 'one': 1, 'two': 2, 'three': 3, 'four': 4,
    'five': 5, 'six': 6, 'seven': 7, 'eight': 8, 'nine': 9,
    'ten': 10,  # Add more numbers as needed
}

t = 'one'
d = word_to_num_dict.get(t.lower(), None)
print(d)

pip install inflect
import inflect

p = inflect.engine()
t = 'one'
d = p.number_to_words(t)
print(d)
```

the above statements aren't working(conversion of words to number),so please try it later

gradient descendent and cost function## #gradient descendent and cost function



Mean Squared Error

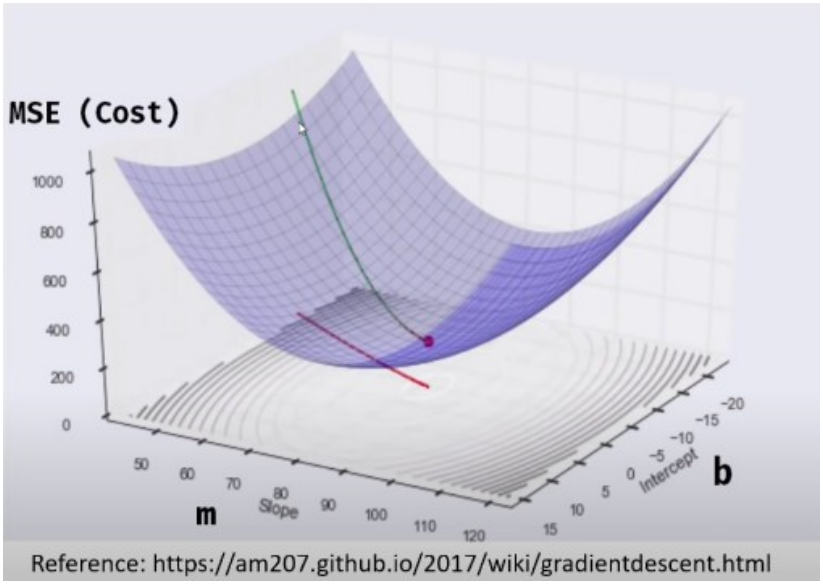
$$mse = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - y_{predicted}\right)^2$$

Mean Squared Error

$$mse = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - (mx_i + b)\right)^2$$

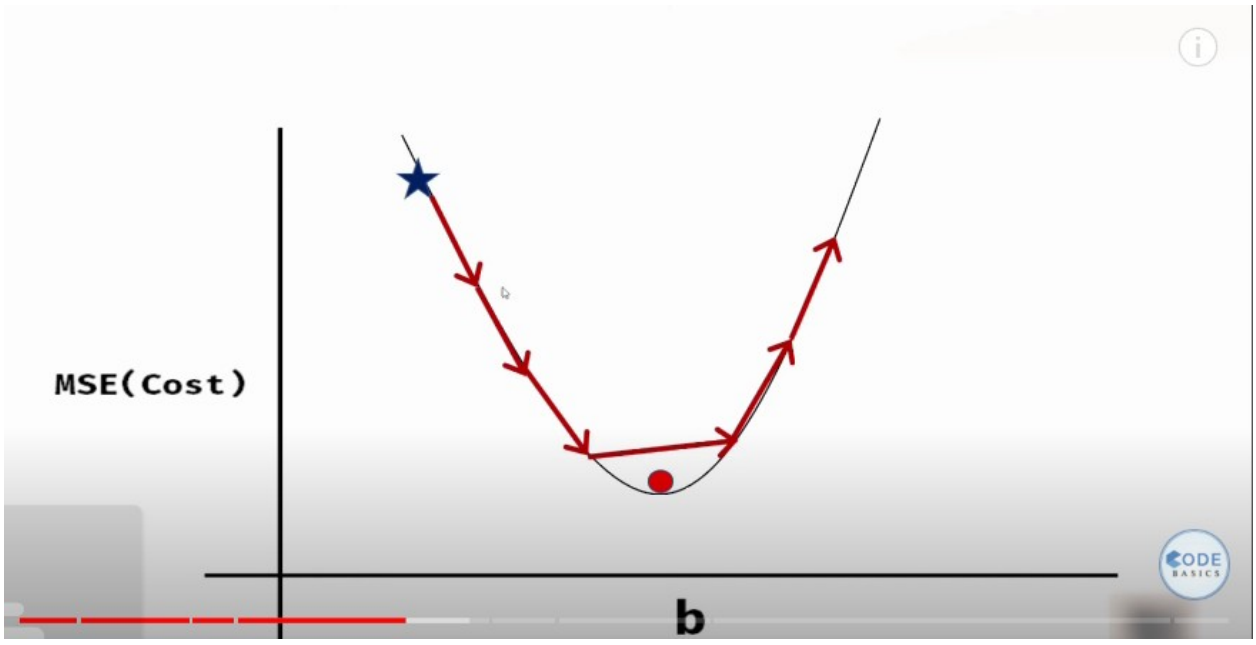
Cost Function

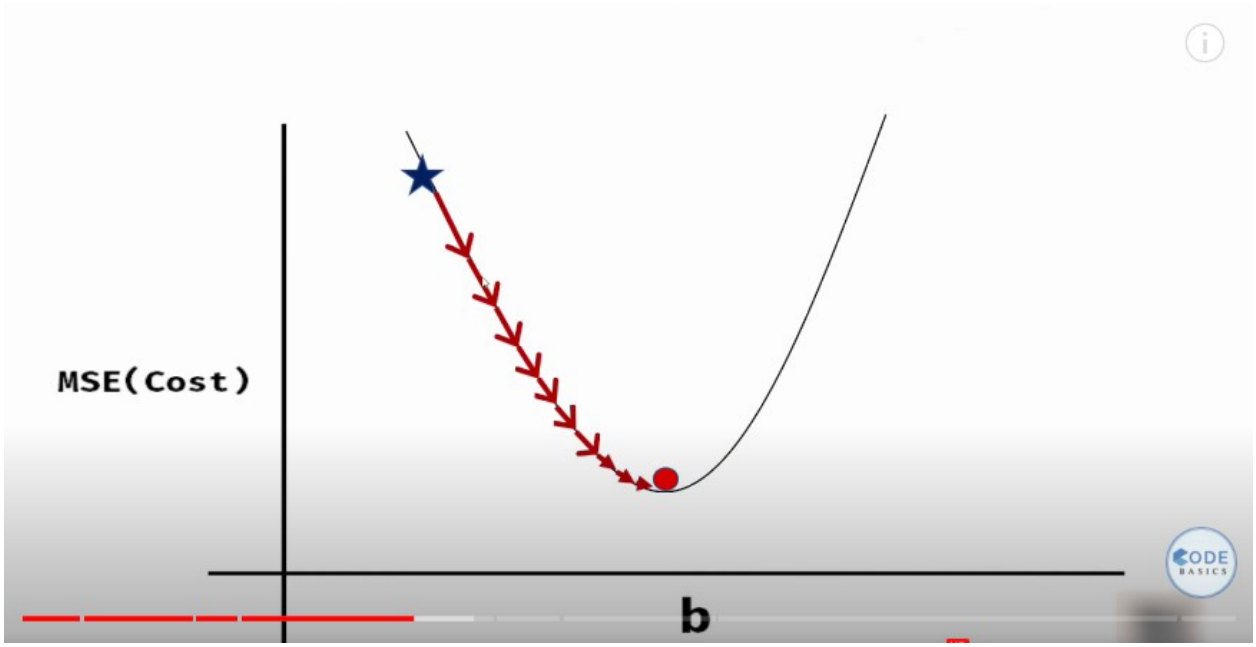
```
"GRADIENT DESCENT"is an algorithm that helps us find
best fit line for given training data set.
```



Reference: https://am207.github.io/2017/wiki/gradientdescent.html

The way to reach min. MSE is in two ways,the first is to take a fixed size step but in this we can miss the minima /min.error point and another step is to take the small steps so that we can reach the min. error point



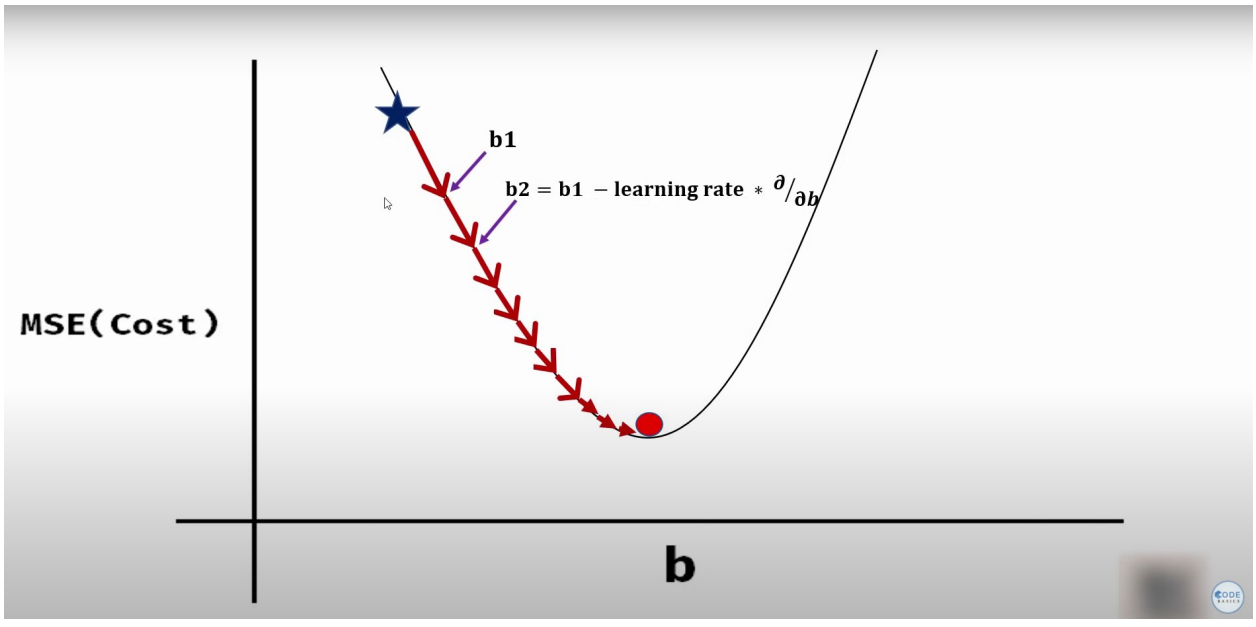


```
stepsize is reducing in above figure,and like this we can reach
min.error point
```

$$mse = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - (mx_i + b)\right)^2$$

$$\partial/\partial m = \frac{2}{n}\sum_{i=1}^{n} -x_i\left(y_i - (mx_i + b)\right)$$

$$\partial/\partial b = \frac{2}{n}\sum_{i=1}^{n} -\left(y_i - (mx_i + b)\right)$$




```
conda update conda
conda update anaconda

pip install ipykernel --upgrade
```

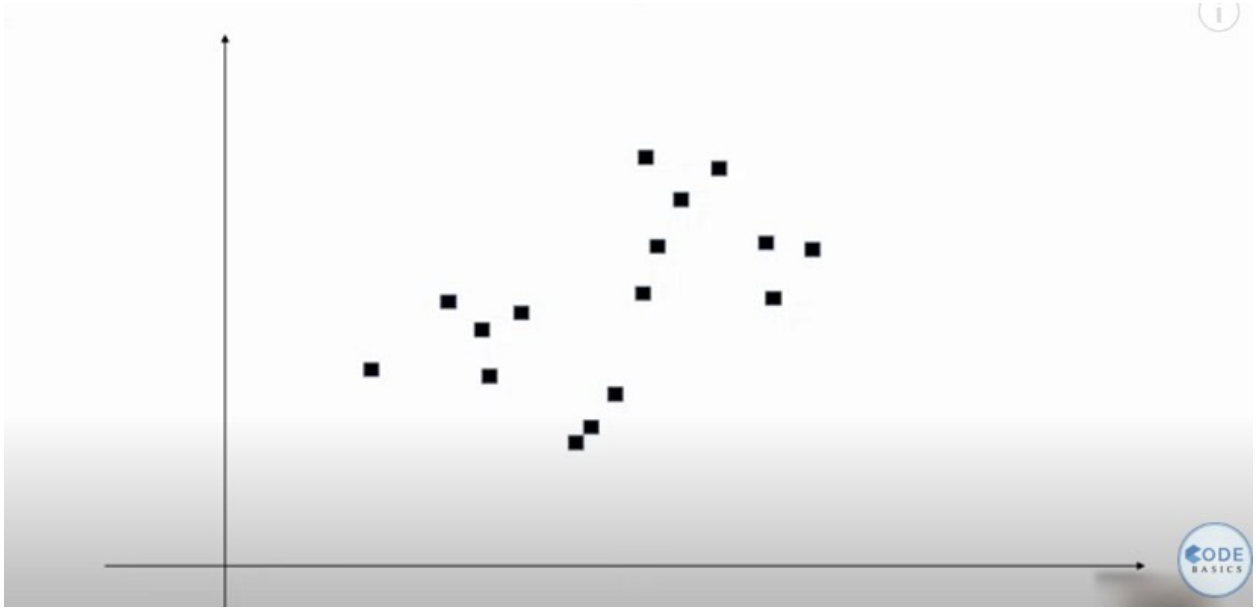
```
import numpy as np

def gradient_descent(x, y):
    m_curr = b_curr = 0
    iterations = 1000
    n = len(x)
    learning_rate = 0.001

    for i in range(iterations):
        y_predicted = m_curr * x + b_curr
        cost=1/n*sum([val**2 for val in (y-y_predicted)])
        md = -(2/n) * sum(x * (y - y_predicted))
        bd = -(2/n) * sum(y - y_predicted)
        m_curr = m_curr - learning_rate * md
        b_curr = b_curr - learning_rate * bd
        print("m: {}, b: {}, iteration: {},cost :{}".format(m_curr,
b_curr, i,cost))

x = np.array([1, 2, 3, 4, 5])
y = np.array([5, 7, 9, 11, 13])
gradient_descent(x, y)
```

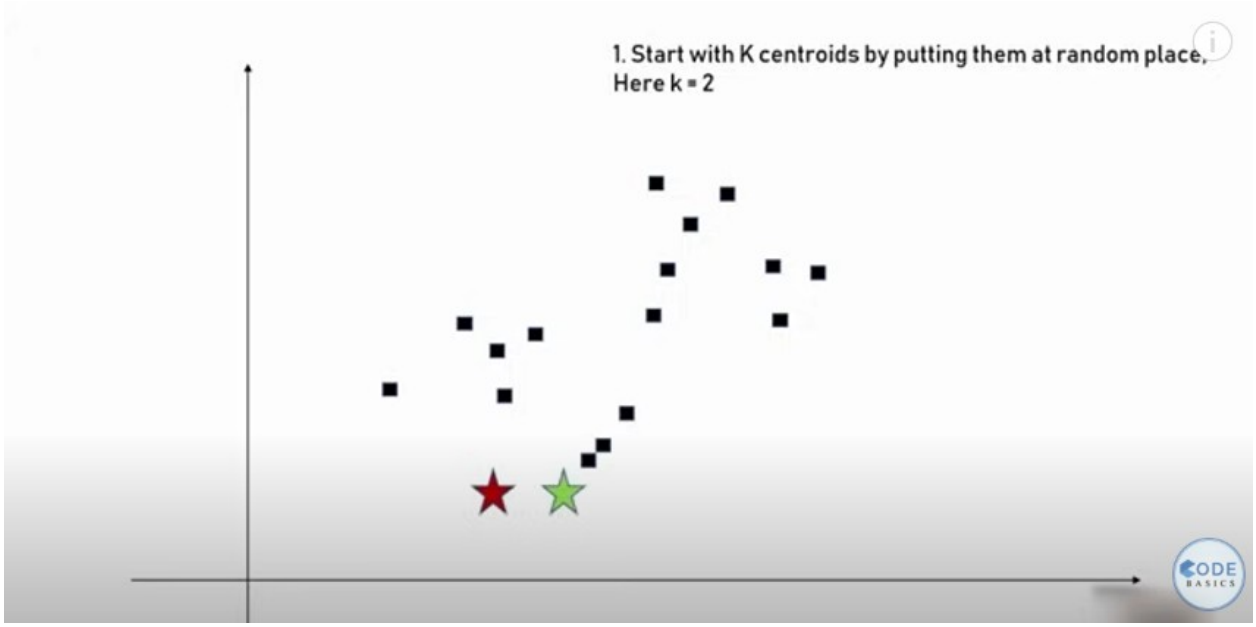
```
m: 0.062, b: 0.018000000000000002, iteration: 0,cost :89.0m: 0.122528,
b: 0.035592000000000006, iteration: 1,cost :84.881304
m: 0.181618832, b: 0.052785648000000004, iteration:
2,cost :80.955185108544
m: 0.239306503808, b: 0.069590363712, iteration:
3,cost :77.21263768455901m: 0.29562421854195203, b: 0.086015343961728,
iteration: 4,cost :73.64507722605434m: 0.35060439367025875, b:
0.10206956796255283, iteration: 5,cost :70.2443206760065m:
0.40427867960173774, b: 0.11776180246460617, iteration:
6,cost :67.00256764921804
m: 0.4566779778357119, b: 0.13310060678206653, iteration:
7,cost :63.912382537082294
m: 0.5078324586826338, b: 0.14809433770148814, iteration:
8,cost :60.966677449199324
m: 0.5577715785654069, b: 0.16275115427398937, iteration:
9,cost :58.15869595270883
```

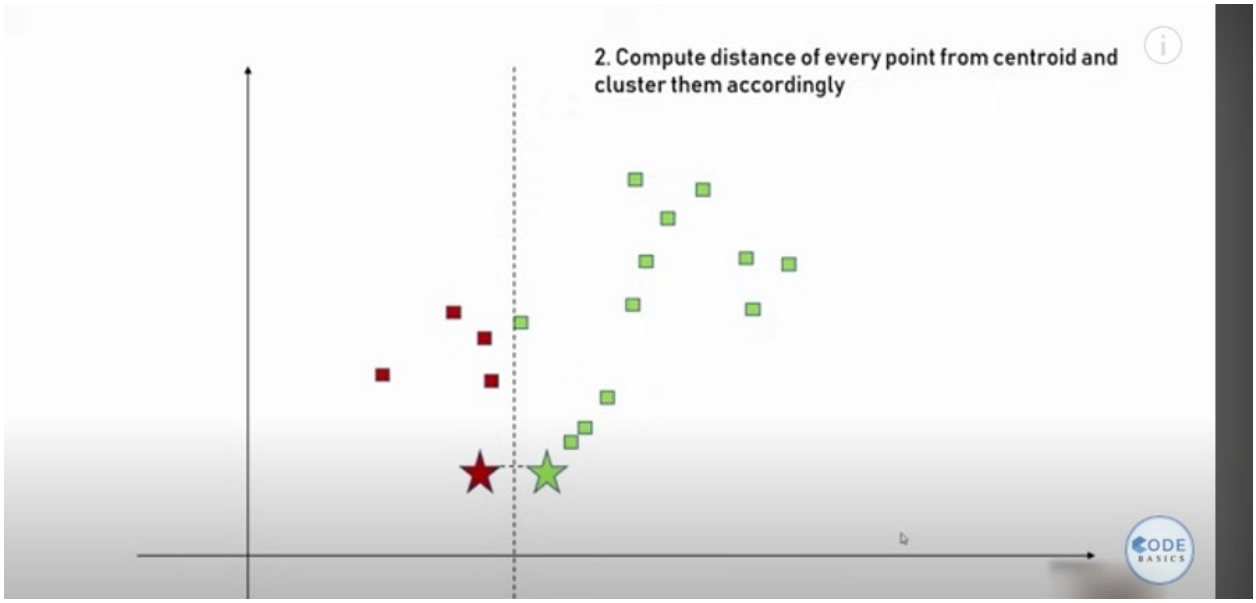
```
import pickle

with open('model_pickle', 'wb') as f:
    pickle.dump(model, f)

import pickle
from sklearn.linear_model import LinearRegression

# Create a simple Linear Regression model
model = LinearRegression()

# Sample data for training
```

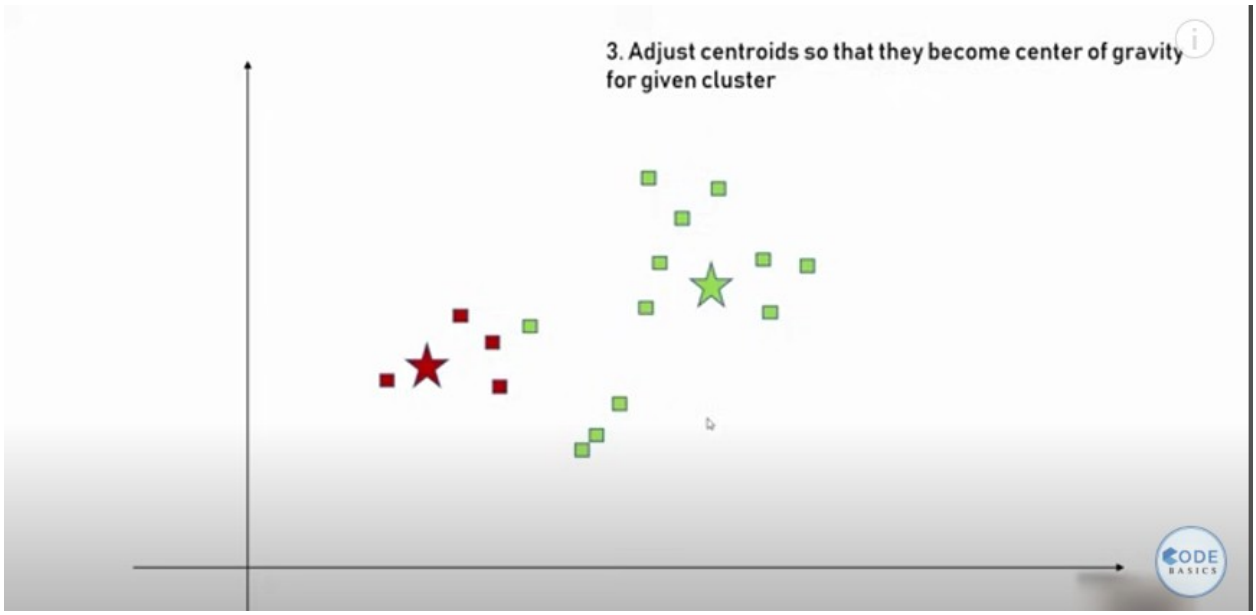
```
X_train = [[1], [2], [3], [4], [5]]
y_train = [2, 4, 6, 8, 10]

# Train the model
model.fit(X_train, y_train)

# Save the model to a file using pickle
with open('model_pickle', 'wb') as f:
    pickle.dump(model, f)
```

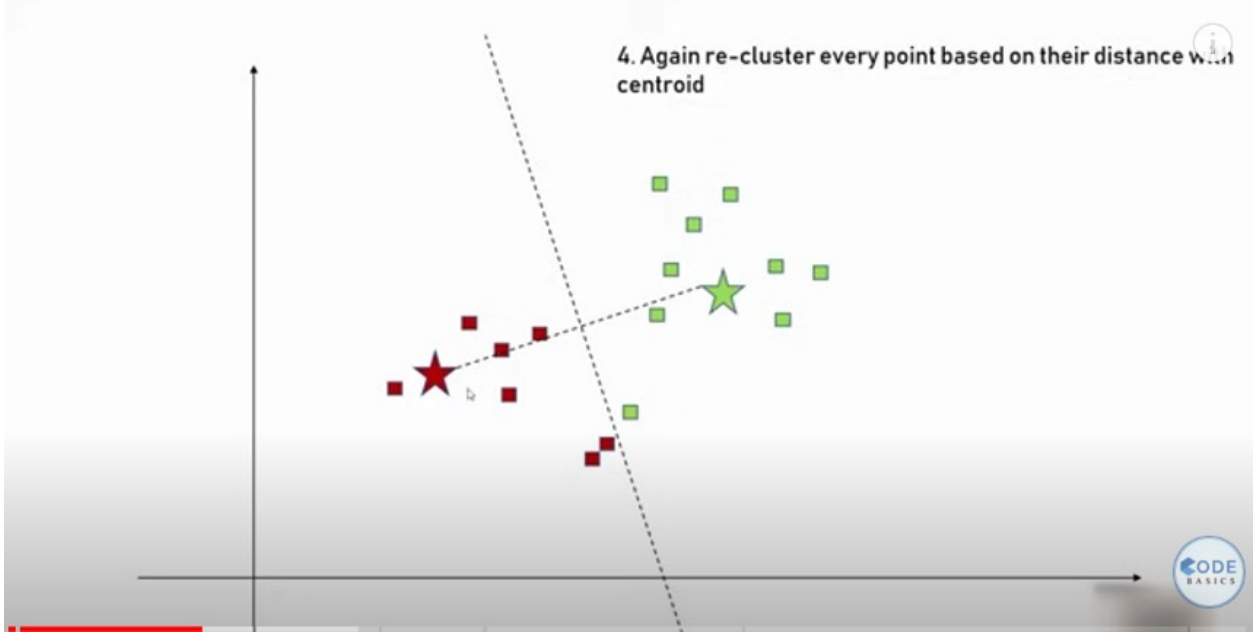
K Means Clustering Algorithm

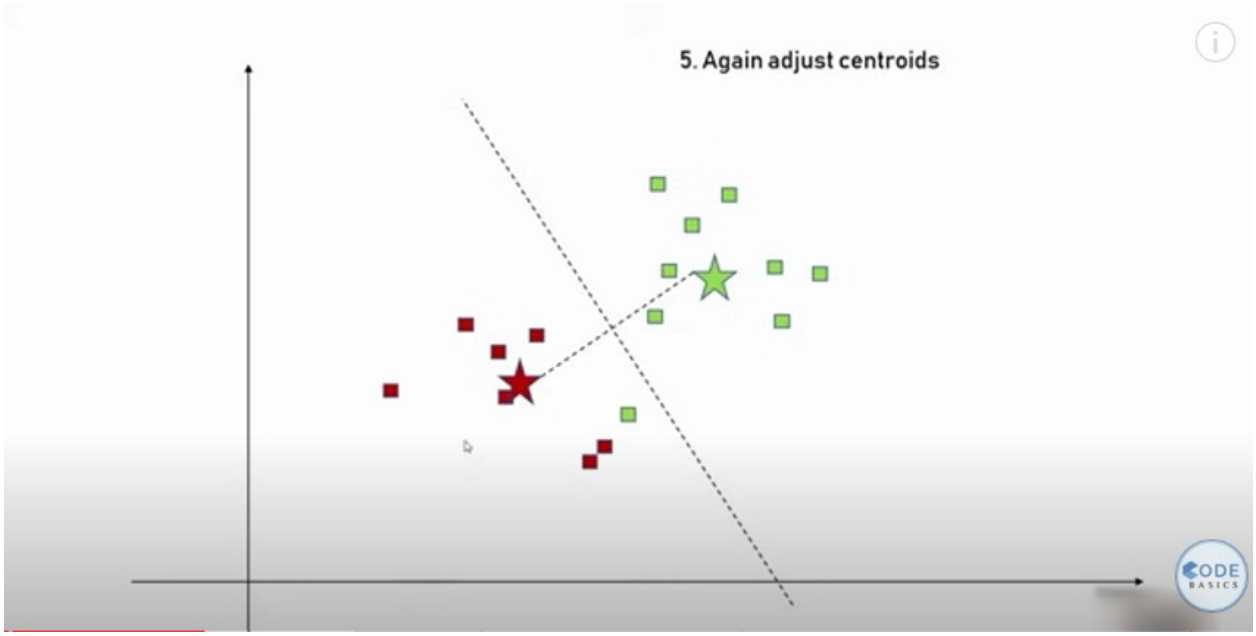


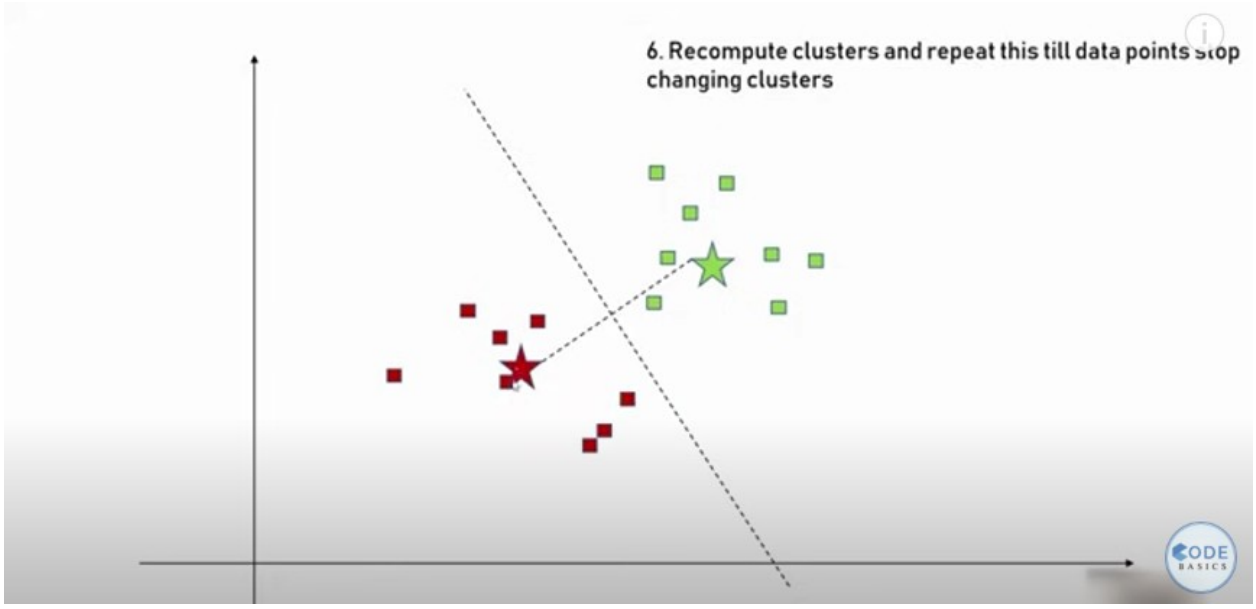


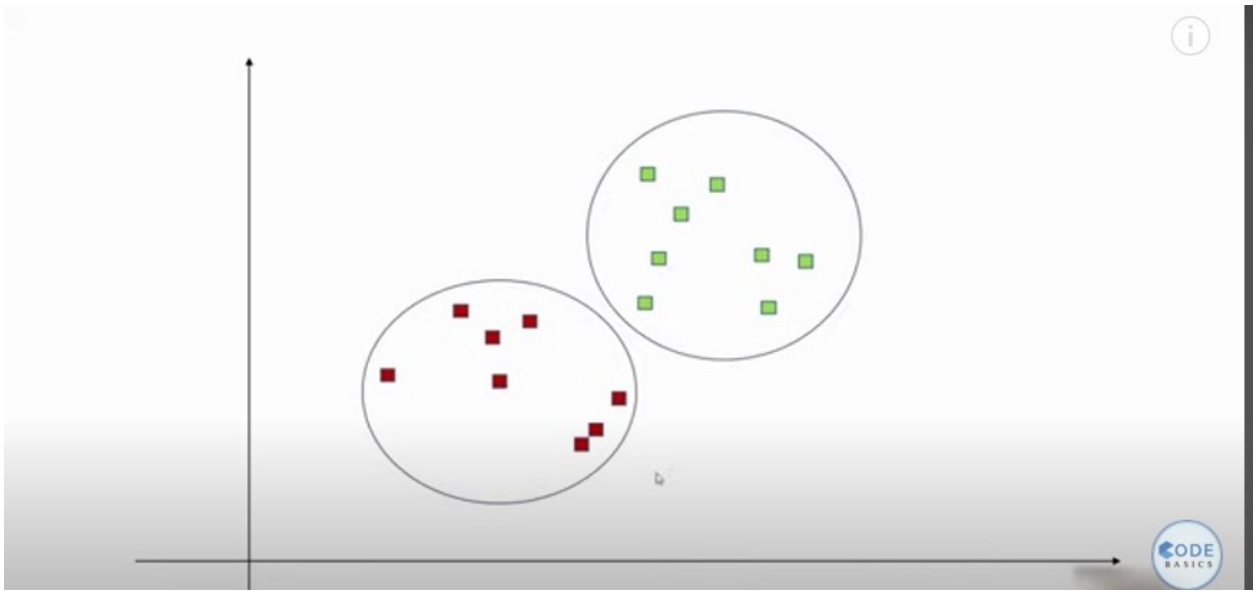

2. Compute distance of every point from centroid and cluster them accordingly

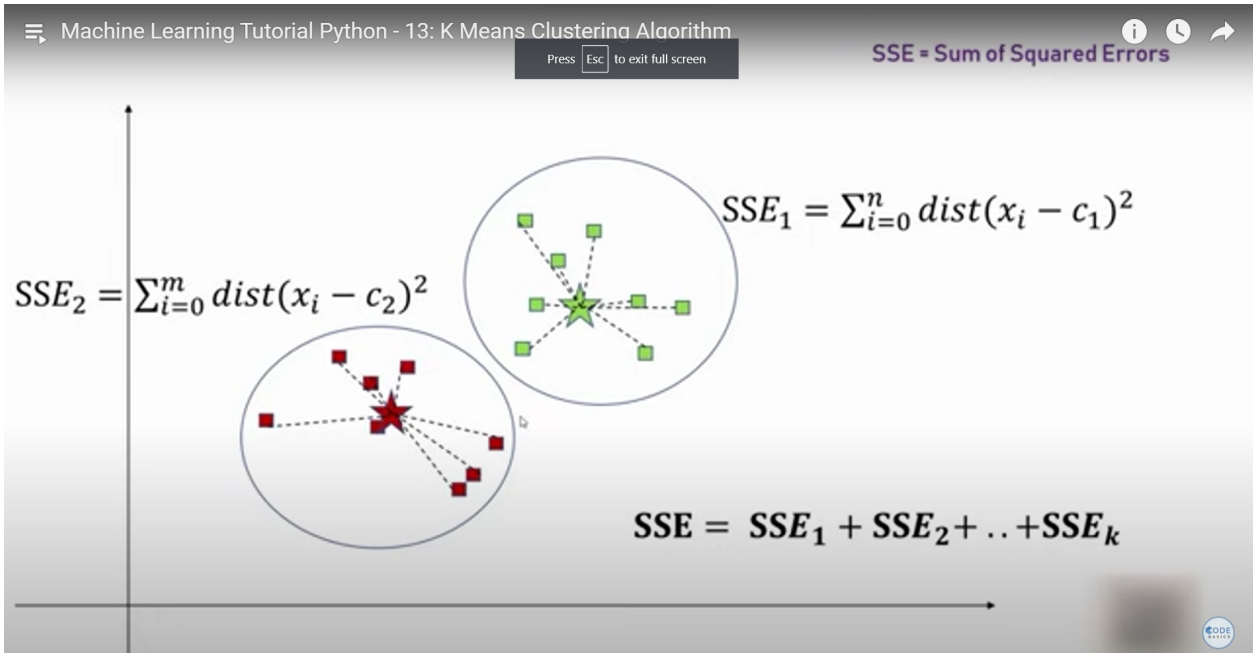

3. Adjust centroids so that they become center of gravity for given cluster

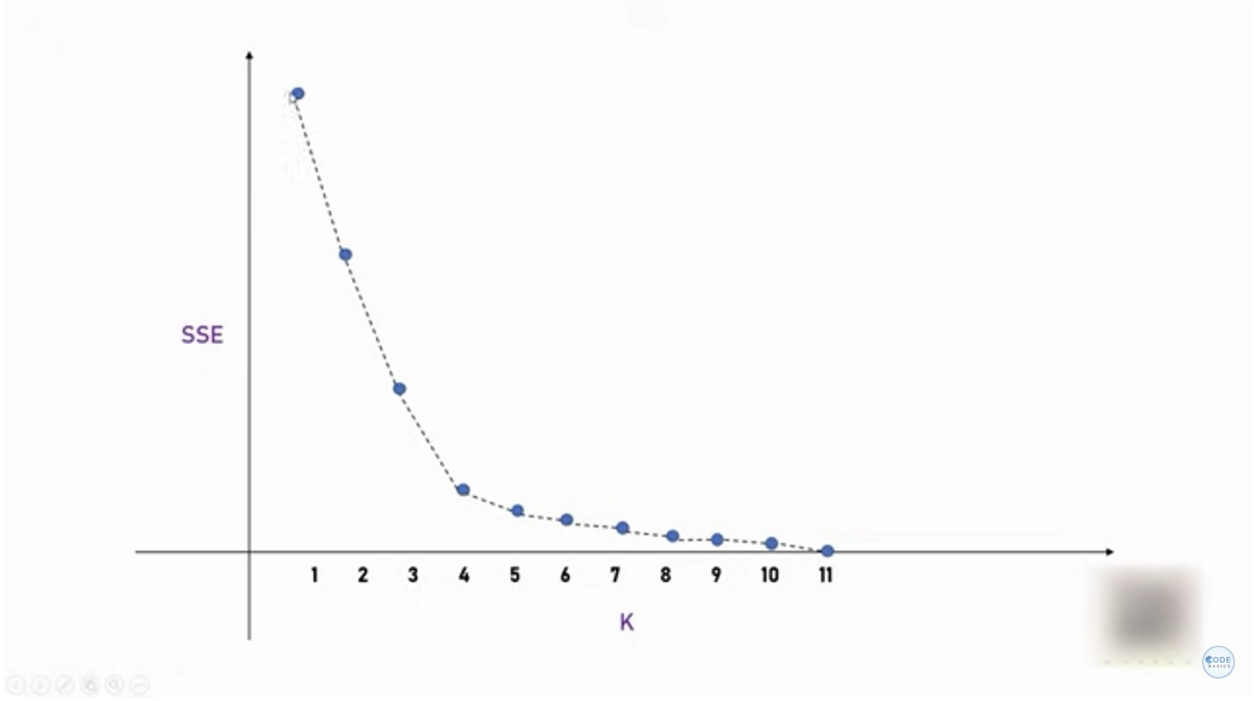

4. Again re-cluster every point based on their distance with centroid
CODE BASICS

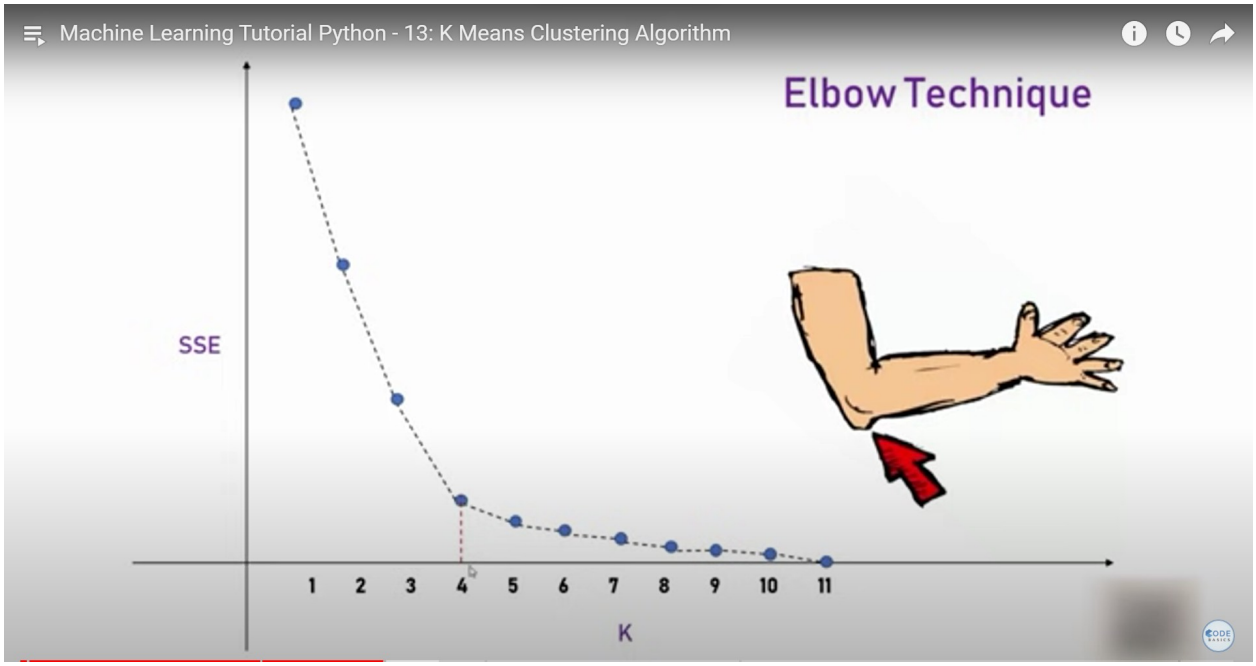

5. Again adjust centroids
CODE BASICS


6. Recompute clusters and repeat this till data points stop changing clusters
CODE BASICS


CODE BASICS


Machine Learning Tutorial Python - 13: K Means Clustering Algorithm
Press Esc to exit full screen
SSE = Sum of Squared Errors
$SSE_1 = \sum_{i=0}^{n} dist(x_i - c_1)^2$
$SSE_2 = \sum_{i=0}^{m} dist(x_i - c_2)^2$
$SSE = SSE_1 + SSE_2 + .. + SSE_k$


SSE
1
2
3
4
5
6
7
8
9
10
11
K



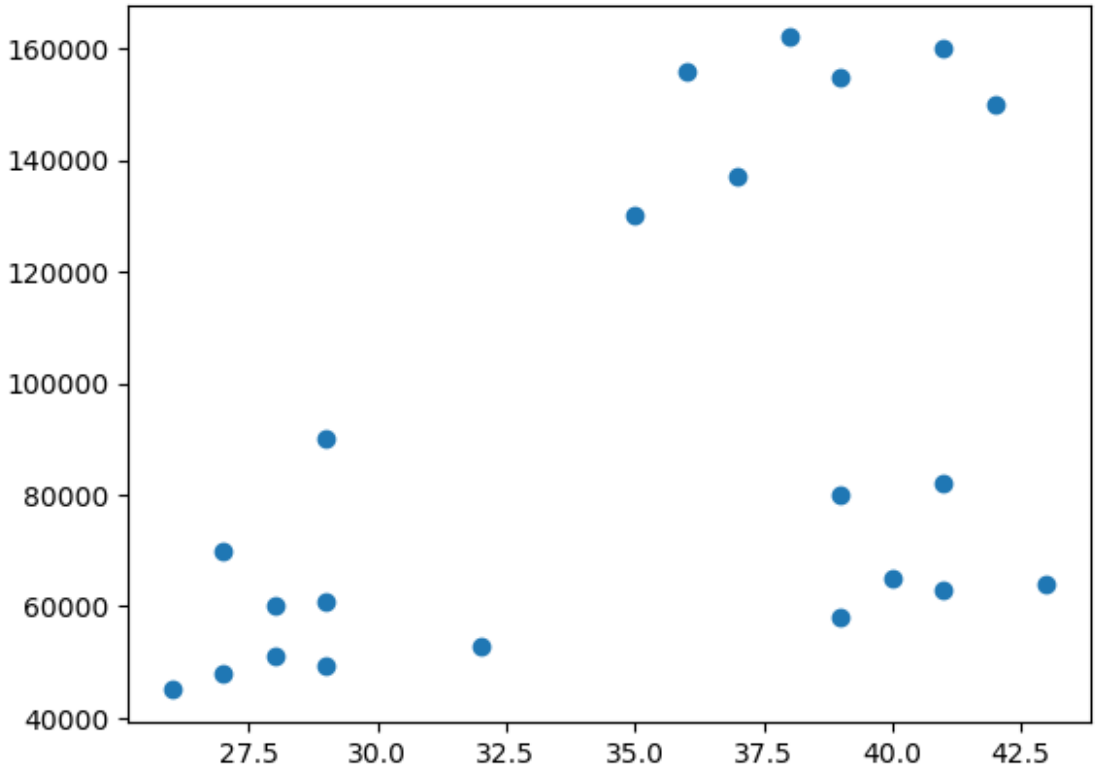
We will consider the K for which SSE is towards minimum and using graph we can get good K value at elbow.

```
from sklearn.cluster import KMeans
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from matplotlib import pyplot as plt
%matplotlib inline

df=pd.read_csv(r"C:\Users\Ayush\OneDrive\Documents\income.csv")
df.head()
```

	Name	Age	Income($)
0	Rob	27	70000
1	Michael	29	90000
2	Mohan	29	61000
3	Ismail	28	60000
4	Kory	42	150000

```
plt.scatter(df['Age'],df['Income($)'])
```

```
<matplotlib.collections.PathCollection at 0x28cca8606d0>
```




```
km=KMeans(n_clusters=3)
km
```

```
KMeans(n_clusters=3)
```

```
y_predicted=km.fit_predict(df[['Age','Income($)']])
y_predicted
```

```
C:\ProgramData\anaconda3\lib\site-packages\sklearn\cluster\
_kmeans.py:870: FutureWarning: The default value of `n_init` will
change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly
to suppress the warning
  warnings.warn(
C:\ProgramData\anaconda3\lib\site-packages\sklearn\cluster\
_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on
Windows with MKL, when there are less chunks than available threads.
You can avoid it by setting the environment variable
OMP_NUM_THREADS=1.
  warnings.warn(

array([2, 2, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2,
0])
```

It made three clusters assigning them three different values i.e,0,1,2

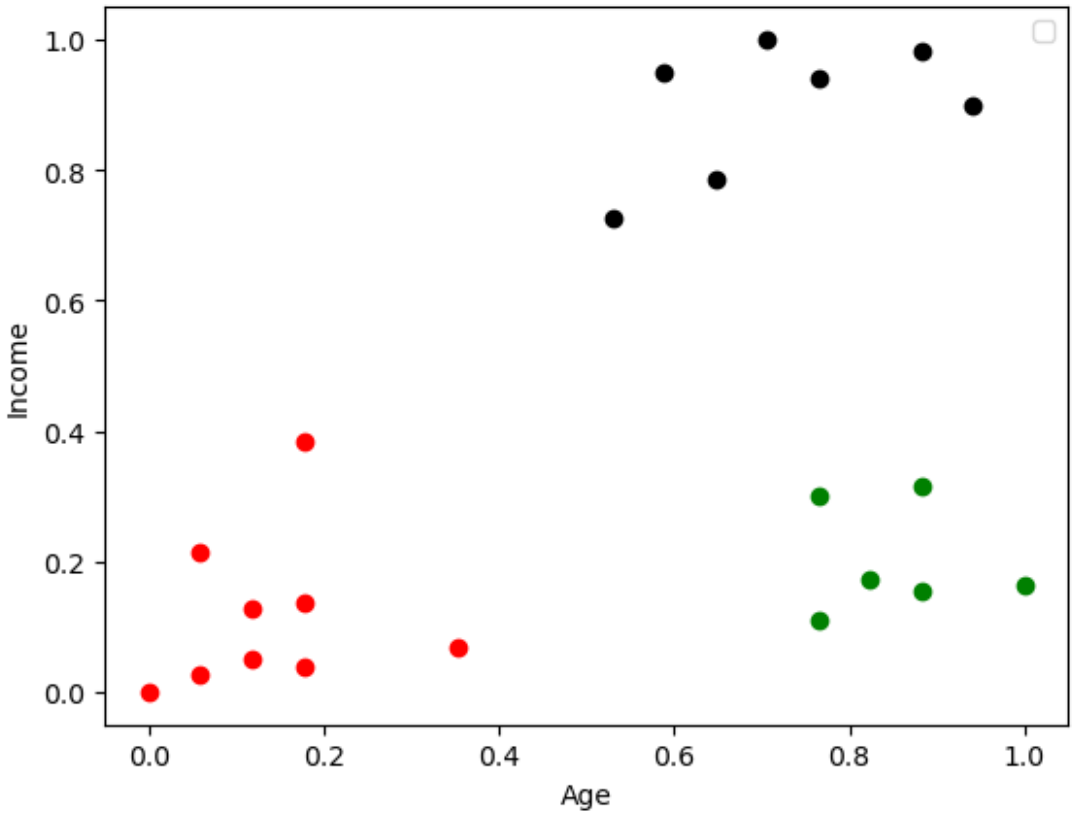
```
df['cluster']=y_predicted
df.head()
```

```
      Name  Age  Income($)  cluster
0      Rob   27      70000        2
1  Michael   29      90000        2
2    Mohan   29      61000        0
3   Ismail   28      60000        0
4     Kory   42     150000        1
```

```
df1=df[df.cluster==0]
df2=df[df.cluster==1]
df3=df[df.cluster==2]

plt.scatter(df1['Age'],df1['Income($)'],color='green')
plt.scatter(df2['Age'],df2['Income($)'],color='red')
plt.scatter(df3['Age'],df3['Income($)'],color='black')

plt.xlabel('Age')
plt.ylabel('Income')
plt.legend()
```

```
No artists with labels found to put in legend.  Note that artists
whose label start with an underscore are ignored when legend() is
called with no argument.
```

```
<matplotlib.legend.Legend at 0x28ccb1cf340>
```




```
scaler = MinMaxScaler()
scaler.fit(df[['Income($)']])
df['Income($)'] = scaler.transform(df[['Income($)']])

scaler.fit(df[['Age']])
df['Age'] = scaler.transform(df[['Age']])

df
```

	Name	Age	Income($)	cluster
0	Rob	0.058824	0.213675	1
1	Michael	0.176471	0.384615	1
2	Mohan	0.176471	0.136752	1
3	Ismail	0.117647	0.128205	1
4	Kory	0.941176	0.897436	2
5	Gautam	0.764706	0.940171	2
6	David	0.882353	0.982906	2
7	Andrea	0.705882	1.000000	2
8	Brad	0.588235	0.948718	2
9	Angelia	0.529412	0.726496	2
10	Donald	0.647059	0.786325	2
11	Tom	0.000000	0.000000	1
12	Arnold	0.058824	0.025641	1
13	Jared	0.117647	0.051282	1

```
14     Stark  0.176471   0.038462        1
15    Ranbir  0.352941   0.068376        1
16    Dipika  0.823529   0.170940        0
17  Priyanka  0.882353   0.153846        0
18      Nick  1.000000   0.162393        0
19      Alia  0.764706   0.299145        0
20       Sid  0.882353   0.316239        0
21     Abdul  0.764706   0.111111        0
```

some black and green dots are mixed but this is due to improper range of X (small range)and Y(very large range) axis.Now we will scale AGE and INCOME from 0 to 1 to overcome this issue.

```
km=KMeans(n_clusters=3)
y_predicted=km.fit_predict(df[['Age','Income($)']])
y_predicted
```

```
C:\ProgramData\anaconda3\lib\site-packages\sklearn\cluster\
_kmeans.py:870: FutureWarning: The default value of `n_init` will
change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly
to suppress the warning
  warnings.warn(
C:\ProgramData\anaconda3\lib\site-packages\sklearn\cluster\
_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on
Windows with MKL, when there are less chunks than available threads.
You can avoid it by setting the environment variable
OMP_NUM_THREADS=1.
  warnings.warn(
```

```
array([1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2,
2])
```

```
df['cluster']=y_predicted
df.drop('cluster',axis='columns',inplace=True)
```

```
df
```

```
        Name       Age  Income($)  clusters
0        Rob  0.058824   0.213675         1
1    Michael  0.176471   0.384615         1
2      Mohan  0.176471   0.136752         1
3     Ismail  0.117647   0.128205         1
4       Kory  0.941176   0.897436         2
5     Gautam  0.764706   0.940171         2
6      David  0.882353   0.982906         2
7     Andrea  0.705882   1.000000         2
8       Brad  0.588235   0.948718         2
9    Angelia  0.529412   0.726496         2
10    Donald  0.647059   0.786325         2
11       Tom  0.000000   0.000000         1
```

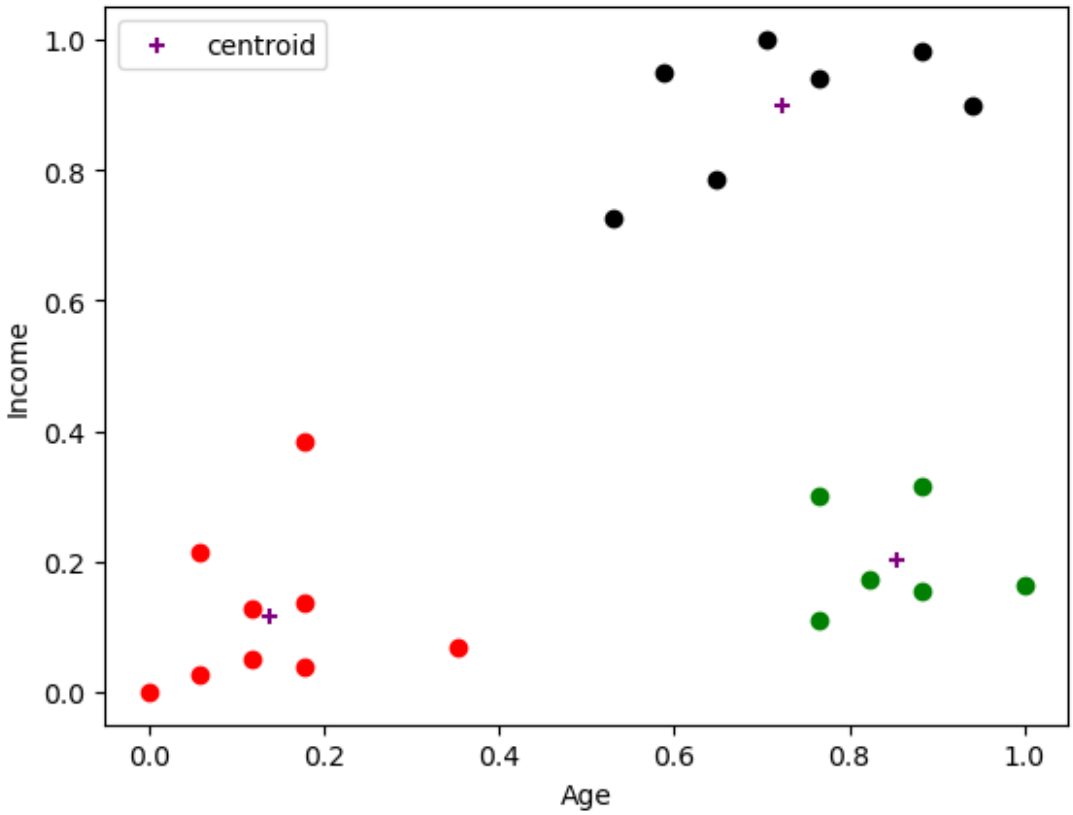
```
12    Arnold  0.058824   0.025641         1
13     Jared  0.117647   0.051282         1
14     Stark  0.176471   0.038462         1
15    Ranbir  0.352941   0.068376         1
16    Dipika  0.823529   0.170940         0
17  Priyanka  0.882353   0.153846         0
18      Nick  1.000000   0.162393         0
19      Alia  0.764706   0.299145         0
20       Sid  0.882353   0.316239         0
21     Abdul  0.764706   0.111111         0
```

```
km.cluster_centers_
```

```
array([[0.72268908, 0.8974359 ],
       [0.1372549 , 0.11633428],
       [0.85294118, 0.2022792 ]])
```

```
df1=df[df.clusters==0]
df2=df[df.clusters==1]
df3=df[df.clusters==2]

plt.scatter(df1['Age'],df1['Income($)'],color='green')
plt.scatter(df2['Age'],df2['Income($)'],color='red')
plt.scatter(df3['Age'],df3['Income($)'],color='black')
plt.scatter(km.cluster_centers_[:,0],km.cluster_centers_[:,1],color='p
urple',marker='+',label='centroid')
plt.xlabel('Age')
plt.ylabel('Income')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x28cd3abd930>
```




```
k_rng=range(1,10)
sse=[]
for k in k_rng:
    km=KMeans(n_clusters=k)
    km.fit(df[['Age','Income($)']])
    sse.append(km.inertia_)
```

```
C:\ProgramData\anaconda3\lib\site-packages\sklearn\cluster\
_kmeans.py:870: FutureWarning: The default value of `n_init` will
change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly
to suppress the warning
  warnings.warn(
C:\ProgramData\anaconda3\lib\site-packages\sklearn\cluster\
_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on
Windows with MKL, when there are less chunks than available threads.
You can avoid it by setting the environment variable
OMP_NUM_THREADS=1.
  warnings.warn(
C:\ProgramData\anaconda3\lib\site-packages\sklearn\cluster\
_kmeans.py:870: FutureWarning: The default value of `n_init` will
change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly
to suppress the warning
  warnings.warn(
C:\ProgramData\anaconda3\lib\site-packages\sklearn\cluster\
```

```
_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on
Windows with MKL, when there are less chunks than available threads.
You can avoid it by setting the environment variable
OMP_NUM_THREADS=1.
  warnings.warn(
C:\ProgramData\anaconda3\lib\site-packages\sklearn\cluster\
_kmeans.py:870: FutureWarning: The default value of `n_init` will
change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly
to suppress the warning
  warnings.warn(
C:\ProgramData\anaconda3\lib\site-packages\sklearn\cluster\
_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on
Windows with MKL, when there are less chunks than available threads.
You can avoid it by setting the environment variable
OMP_NUM_THREADS=1.
  warnings.warn(
C:\ProgramData\anaconda3\lib\site-packages\sklearn\cluster\
_kmeans.py:870: FutureWarning: The default value of `n_init` will
change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly
to suppress the warning
  warnings.warn(
C:\ProgramData\anaconda3\lib\site-packages\sklearn\cluster\
_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on
Windows with MKL, when there are less chunks than available threads.
You can avoid it by setting the environment variable
OMP_NUM_THREADS=1.
  warnings.warn(
C:\ProgramData\anaconda3\lib\site-packages\sklearn\cluster\
_kmeans.py:870: FutureWarning: The default value of `n_init` will
change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly
to suppress the warning
  warnings.warn(
C:\ProgramData\anaconda3\lib\site-packages\sklearn\cluster\
_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on
Windows with MKL, when there are less chunks than available threads.
You can avoid it by setting the environment variable
OMP_NUM_THREADS=1.
  warnings.warn(
C:\ProgramData\anaconda3\lib\site-packages\sklearn\cluster\
_kmeans.py:870: FutureWarning: The default value of `n_init` will
change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly
to suppress the warning
  warnings.warn(
C:\ProgramData\anaconda3\lib\site-packages\sklearn\cluster\
_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on
Windows with MKL, when there are less chunks than available threads.
You can avoid it by setting the environment variable
OMP_NUM_THREADS=1.
  warnings.warn(
```

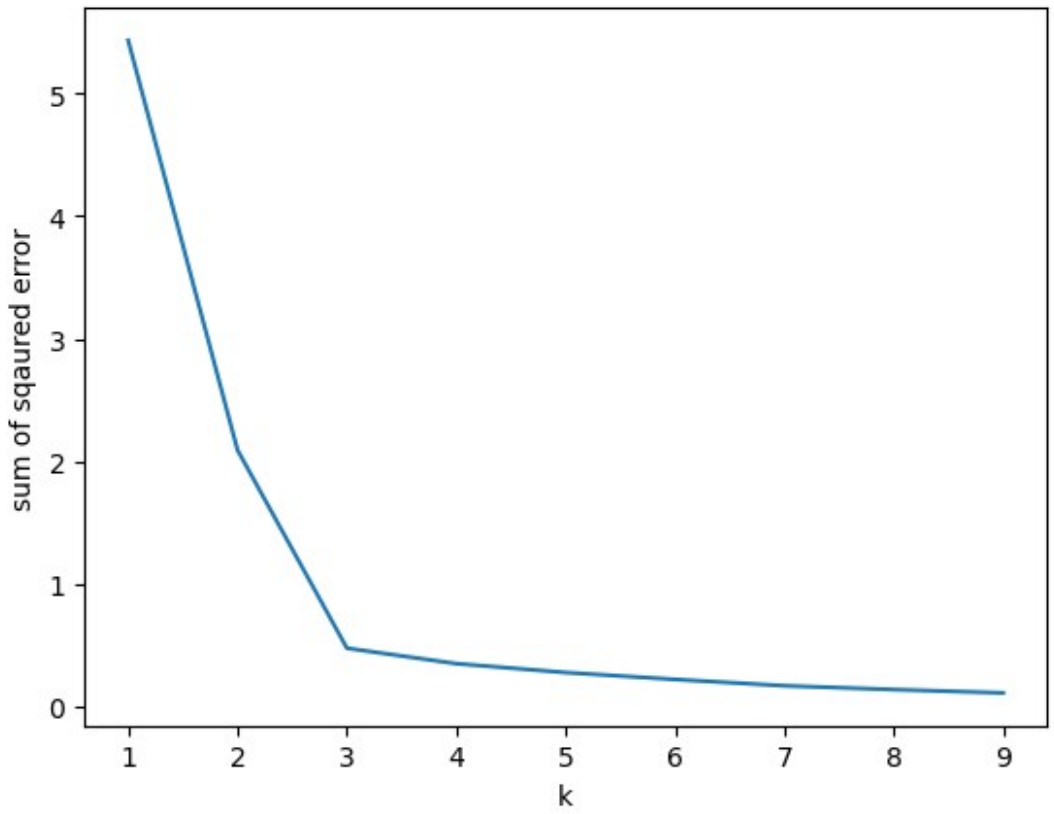
```
C:\ProgramData\anaconda3\lib\site-packages\sklearn\cluster\
_kmeans.py:870: FutureWarning: The default value of `n_init` will
change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly
to suppress the warning
  warnings.warn(
C:\ProgramData\anaconda3\lib\site-packages\sklearn\cluster\
_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on
Windows with MKL, when there are less chunks than available threads.
You can avoid it by setting the environment variable
OMP_NUM_THREADS=1.
  warnings.warn(
C:\ProgramData\anaconda3\lib\site-packages\sklearn\cluster\
_kmeans.py:870: FutureWarning: The default value of `n_init` will
change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly
to suppress the warning
  warnings.warn(
C:\ProgramData\anaconda3\lib\site-packages\sklearn\cluster\
_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on
Windows with MKL, when there are less chunks than available threads.
You can avoid it by setting the environment variable
OMP_NUM_THREADS=1.
  warnings.warn(
C:\ProgramData\anaconda3\lib\site-packages\sklearn\cluster\
_kmeans.py:870: FutureWarning: The default value of `n_init` will
change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly
to suppress the warning
  warnings.warn(
C:\ProgramData\anaconda3\lib\site-packages\sklearn\cluster\
_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on
Windows with MKL, when there are less chunks than available threads.
You can avoid it by setting the environment variable
OMP_NUM_THREADS=1.
  warnings.warn(
```

```
sse
```

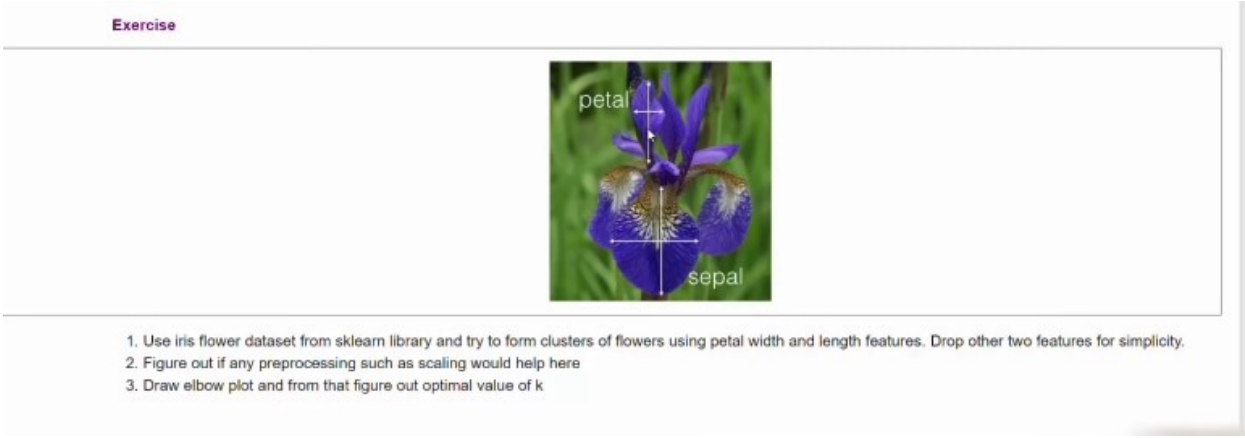
```
[5.434011511988176,
 2.0911363886990766,
 0.4750783498553095,
 0.34910470944195643,
 0.27669362763002775,
 0.2203764169077066,
 0.16869711728567785,
 0.1376250414652804,
 0.11056185254866235]
```

```
plt.xlabel("k")
plt.ylabel("sum of sqaured error")
plt.plot(k_rng,sse)
```

```
[<matplotlib.lines.Line2D at 0x28cd3b26d70>]
```



Therefore k=3 is suitable value as its our elbow



Naive Bayes Classifier Algorithm

```
import pandas as pd
df=pd.read_csv(r"C:\Users\Ayush\Downloads\Titanic-Dataset.csv")
df.head()
```

```
   PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3

                                                Name     Sex   Age
SibSp  \
0                            Braund, Mr. Owen Harris    male  22.0
1
1  Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0
1
2                             Heikkinen, Miss. Laina  female  26.0
0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0
1
4                           Allen, Mr. William Henry    male  35.0
0

   Parch            Ticket     Fare Cabin Embarked
0      0         A/5 21171   7.2500   NaN        S
1      0          PC 17599  71.2833   C85        C
2      0  STON/O2. 3101282   7.9250   NaN        S
3      0            113803  53.1000  C123        S
4      0            373450   8.0500   NaN        S
```

```
df.drop(['PassengerId','Name','SibSp','Parch','Ticket','Cabin','Embark
ed'],axis='columns',inplace=True)
df.head()
```

```
   Survived  Pclass     Sex   Age     Fare
0         0       3    male  22.0   7.2500
1         1       1  female  38.0  71.2833
2         1       3  female  26.0   7.9250
3         1       1  female  35.0  53.1000
4         0       3    male  35.0   8.0500
```

```
target=df.Survived
inputs=df.drop(['Survived'],axis='columns')

dummies=pd.get_dummies(inputs.Sex)
dummies.head(3)
```

```
   female  male
0       0     1
1       1     0
2       1     0
```

```
inputs=pd.concat([inputs,dummies],axis='columns')
inputs.head(3)
```

```
   Pclass     Sex   Age     Fare  female  male
0       3    male  22.0   7.2500       0     1
1       1  female  38.0  71.2833       1     0
2       3  female  26.0   7.9250       1     0
```

```
inputs.drop(['Sex'],axis='columns',inplace=True)
inputs.head(3)
```

```
   Pclass   Age     Fare  female  male
0       3  22.0   7.2500       0     1
1       1  38.0  71.2833       1     0
2       3  26.0   7.9250       1     0
```

```
inputs.columns[inputs.isna().any()]
```

```
Index(['Age'], dtype='object')
```

This tells us that there is NA value in age

```
inputs.Age[:10]
```

```
0    22.0
1    38.0
2    26.0
3    35.0
4    35.0
5     NaN
6    54.0
7     2.0
8    27.0
9    14.0
Name: Age, dtype: float64
```

```
inputs.Age=inputs.Age.fillna(inputs.Age.mean())
inputs.head(10)
```

```
   Pclass        Age     Fare  female  male
0       3  22.000000   7.2500       0     1
1       1  38.000000  71.2833       1     0
2       3  26.000000   7.9250       1     0
3       1  35.000000  53.1000       1     0
4       3  35.000000   8.0500       0     1
5       3  29.699118   8.4583       0     1
6       1  54.000000  51.8625       0     1
7       3   2.000000  21.0750       0     1
8       3  27.000000  11.1333       1     0
9       2  14.000000  30.0708       1     0
```

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(inputs,target,test_size
=0.2)
```

```
len(x_train)
```

```
712
```

```
len(x_test)
```

```
179
```

```
from sklearn.naive_bayes import GaussianNB
model=GaussianNB()
```

```
model.fit(x_train,y_train)
```

```
GaussianNB()
```

```
model.score(x_test,y_test)
```

```
0.7821229050279329
```

```
model.predict(x_test[:10])
```

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

```
y_test[:10]
```

```
792    0
590    0
277    0
266    0
770    0
565    0
318    1
805    0
722    0
861    0
Name: Survived, dtype: int64
```

```
model.predict_proba(x_test[:10])
```

```
array([[2.12627759e-02, 9.78737224e-01],
       [9.91220621e-01, 8.77937918e-03],
       [9.79089545e-01, 2.09104553e-02],
       [9.85242121e-01, 1.47578790e-02],
       [9.90341238e-01, 9.65876210e-03],
       [9.90255099e-01, 9.74490096e-03],
       [2.12443720e-07, 9.99999788e-01],
       [9.91055451e-01, 8.94454854e-03],
       [9.82085960e-01, 1.79140399e-02],
       [9.78754064e-01, 2.12459360e-02]])
```

Left side is probability of person to not survive and right is to survive in particular class

```
import pandas as pd
df=pd.read_csv(r"C:\Users\Ayush\Downloads\spam.csv")
df.head()
```

```
  Category                                            Message
0      ham  Go until jurong point, crazy.. Available only ...
1      ham                      Ok lar... Joking wif u oni...
2     spam  Free entry in 2 a wkly comp to win FA Cup fina...
3      ham  U dun say so early hor... U c already then say...
4      ham  Nah I don't think he goes to usf, he lives aro...
```

```
df.groupby('Category').describe()
```

```
         Message
\
           count unique
top
Category

ham         4825   4516                                Sorry, I'll call
later
spam         747    641  Please call our customer service
representativ...


         freq
Category
ham        30
spam        4
```

```
df['spam']=df['Category'].apply(lambda x:1 if x== 'spam' else 0)
df.head()
```

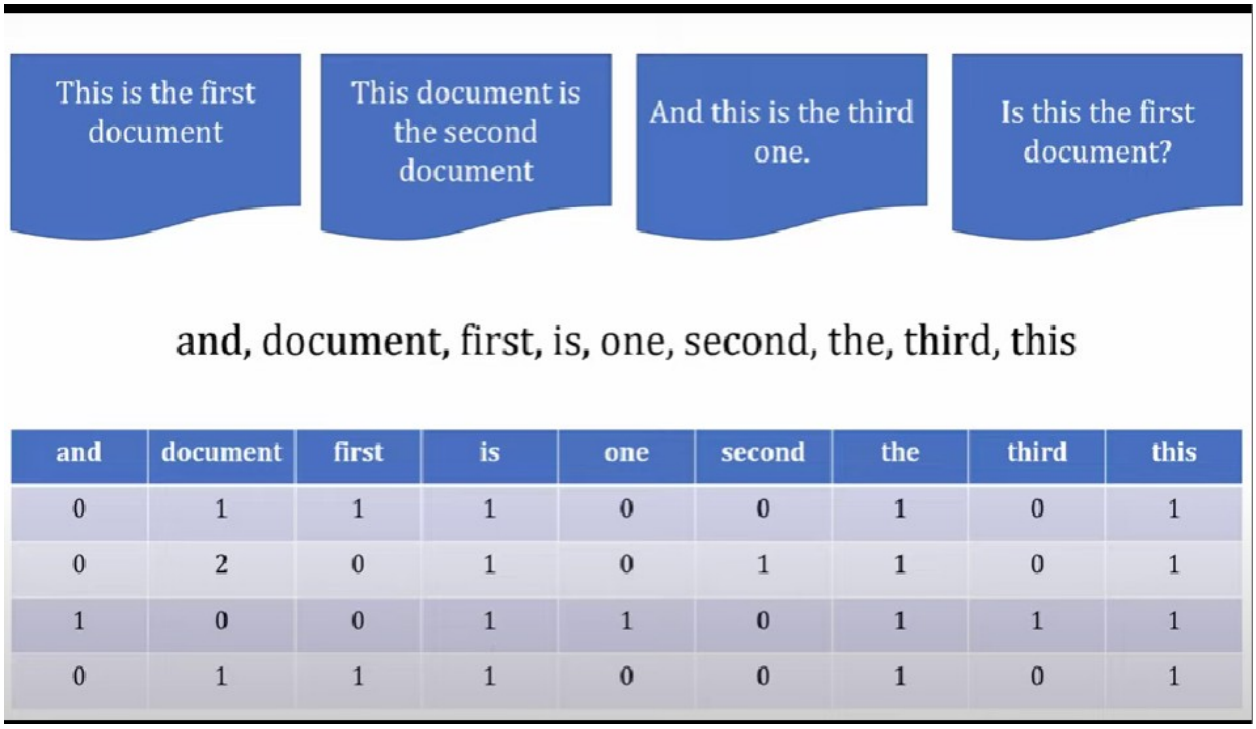
```
  Category                                            Message  spam
0      ham  Go until jurong point, crazy.. Available only ...     0
1      ham                      Ok lar... Joking wif u oni...     0
2     spam  Free entry in 2 a wkly comp to win FA Cup fina...     1
3      ham  U dun say so early hor... U c already then say...     0
4      ham  Nah I don't think he goes to usf, he lives aro...     0
```

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(df.Message,df.spam,test
_size=0.25)
```

```
---------------------------------------------------------------------
-----
NameError                                 Traceback (most recent call
last)
Cell In[1], line 2
      1 from sklearn.model_selection import train_test_split
----> 2
```

```
x_train,x_test,y_train,y_test=train_test_split(df.Message,df.spam,test
_size=0.25)

NameError: name 'df' is not defined
```

Now we need to need to convert message text to numbers as we know that machine learning model can handle numbers and not alphabets and this we would do with help of COUNT VECTOR TECHNIQUE

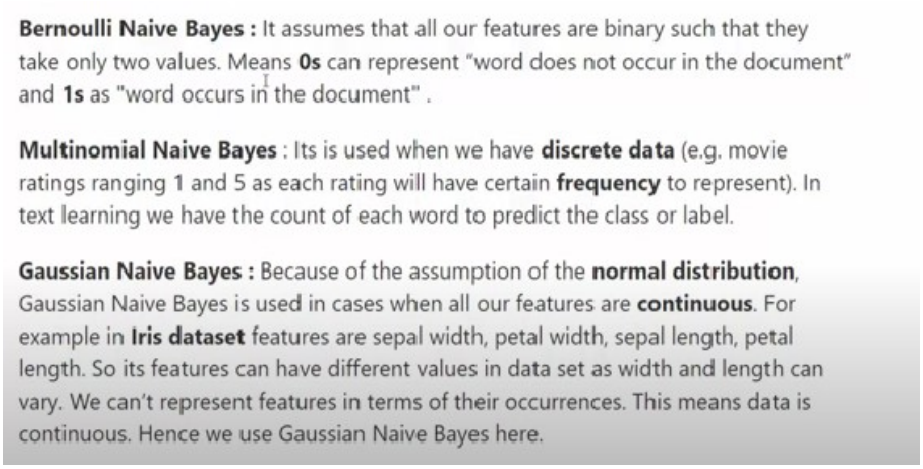


and	document	first	is	one	second	the	third	this
0	1	1	1	0	0	1	0	1
0	2	0	1	0	1	1	0	1
1	0	0	1	1	0	1	1	1
0	1	1	1	0	0	1	0	1

```
>>> from sklearn.feature_extraction.text import CountVectorizer
>>> corpus = [
...     'This is the first document.',
...     'This document is the second document.',
...     'And this is the third one.',
...     'Is this the first document?',
... ]
>>> vectorizer = CountVectorizer()
>>> X = vectorizer.fit_transform(corpus)
>>> print(vectorizer.get_feature_names())
['and', 'document', 'first', 'is', 'one', 'second', 'the', 'third', 'this']
>>> print(X.toarray())
[[0 1 1 1 0 0 1 0 1]
 [0 2 0 1 0 1 1 0 1]
 [1 0 0 1 1 0 1 1 1]
 [0 1 1 1 0 0 1 0 1]]
```

```
from sklearn.feature_extraction.text import CountVectorizer
v=CountVectorizer()
x_train_count=v.fit_transform(x_train.values)
x_train_count.toarray()[:3]
```

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




```
from sklearn.naive_bayes import MultinomialNB
model=MultinomialNB()
model.fit(x_train_count,y_train)
```

```
MultinomialNB()
```

```
emails=[
    'Hey mohan,can we get together to watch football game tomorrow?',
    'Upto 20% discount on parking,exclusive offer just for you'
]
emails_count=v.transform(emails)
model.predict(emails_count)
```

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

```
x_test_count=v.transform(x_test)
model.score(x_test_count,y_test)
```

```
0.9798994974874372
```

Using sklearn pipeline for transformation

```
from sklearn.pipeline import Pipeline
clf=Pipeline([
    ('vectorizer',CountVectorizer()),
```

```
    ('nb',MultinomialNB())
])

clf.fit(x_train,y_train)
```

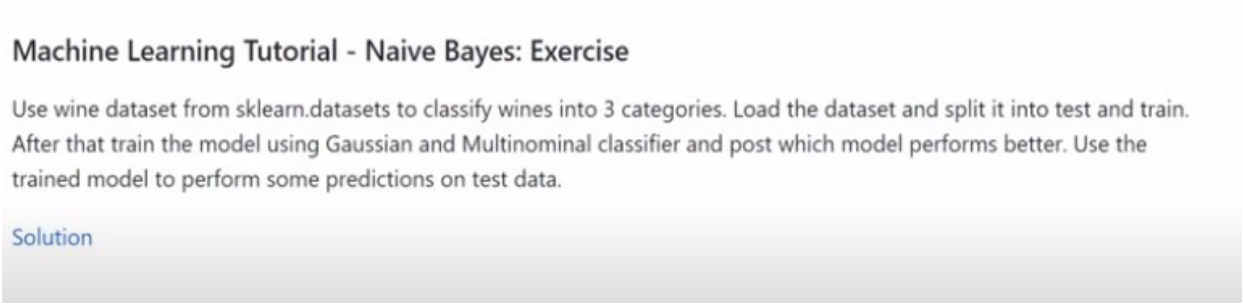
```
Pipeline(steps=[('vectorizer', CountVectorizer()), ('nb',
MultinomialNB())])
```

```
clf.score(x_test,y_test)
```

```
0.9798994974874372
```

```
clf.predict(emails)
```

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

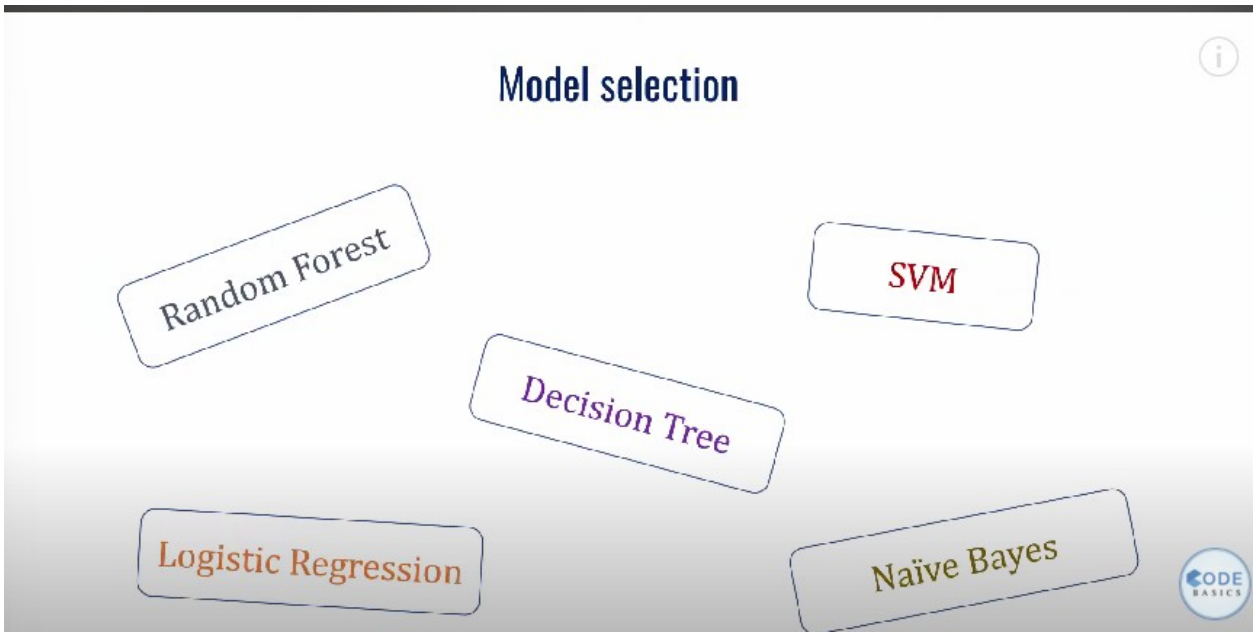


We have learnt so many models till now,they all have some parameters.The process of tuning the optimal parameter is called HYPERPARAMETER TUNING.

```
from sklearn import svm,datasets
iris=datasets.load_iris()

import pandas as pd
df=pd.DataFrame(iris.data,columns=iris.feature_names)
df['flower']=iris.target
df['flower']=df['flower'].apply(lambda x: iris.target_names[x])
df[47:52]
```

```
    sepal length (cm)  sepal width (cm)  petal length (cm)  petal
width (cm)  \
47                 4.6               3.2                1.4
0.2
48                 5.3               3.7                1.5
0.2
49                 5.0               3.3                1.4
0.2
50                 7.0               3.2                4.7
1.4
51                 6.4               3.2                4.5
1.5
```

```
        flower
47      setosa
48      setosa
49      setosa
50  versicolor
51  versicolor
```




```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(iris.data,iris.target,test_size=0.3)

model = svm.SVC(kernel='rbf',C=30,gamma='auto')
model.fit(x_train,y_train)
model.score(x_test,y_test)
```

```
0.9777777777777777
```

since score is changing depending on training and testing data set so we cant rely on this method.so we would do k fold method

```
 from sklearn.model_selection import cross_val_score

cross_val_score(svm.SVC(kernel='linear',C=10,gamma='auto'),iris.data,iris.target,cv=5)
```

```
array([1.        , 1.        , 0.9       , 0.96666667, 1.        ])
```

```
cross_val_score(svm.SVC(kernel='rbf',C=10,gamma='auto'),iris.data,iris.target,cv=5)
```

```
array([0.96666667, 1.        , 0.96666667, 0.96666667, 1.        ])
```

```
cross_val_score(svm.SVC(kernel='rbf',C=20,gamma='auto'),iris.data,iris
.target,cv=5)
```

```
array([0.96666667, 1.        , 0.9       , 0.96666667, 1.        ])
```

```
import numpy as np
kernels=['rbf','linear']
C=[1,10,20]
avg_scores={}
for kval in kernels:
    for cval in C:

cv_scores=cross_val_score(svm.SVC(kernel=kval,C=cval,gamma='auto'),iri
s.data,iris.target,cv=5)
        avg_scores[kval+'-'+str(cval)]=np.average(cv_scores)
avg_scores
```

```
{'rbf-1': 0.9800000000000001,
 'rbf-10': 0.9800000000000001,
 'rbf-20': 0.9666666666666668,
 'linear-1': 0.9800000000000001,
 'linear-10': 0.9733333333333334,
 'linear-20': 0.9666666666666666}
```

```
from sklearn.model_selection import GridSearchCV
clf=GridSearchCV(svm.SVC(gamma='auto') ,{
    'C':[1,10,20],
    'kernel':['rbf','linear']
},cv=5,return_train_score=False)
clf.fit(iris.data,iris.target)
clf.cv_results_
```

```
{'mean_fit_time': array([0.00168085, 0.00140595, 0.        ,
0.00159941, 0.00165677,
        0.00190835]),
 'std_fit_time': array([0.0033617 , 0.00202084, 0.        ,
0.00319881, 0.00215055,
        0.0013924 ]),
 'mean_score_time': array([0.00167813, 0.00020185, 0.00160131,
0.00160112, 0.00080175,
        0.00100136]),
 'std_score_time': array([0.00335627, 0.00040369, 0.00320263,
0.00320225, 0.00160351,
        0.00200272]),
 'param_C': masked_array(data=[1, 1, 10, 10, 20, 20],
              mask=[False, False, False, False, False, False],
        fill_value='?',
             dtype=object),
 'param_kernel': masked_array(data=['rbf', 'linear', 'rbf', 'linear',
'rbf', 'linear'],
```

```
              mask=[False, False, False, False, False, False],
       fill_value='?',
            dtype=object),
 'params': [{'C': 1, 'kernel': 'rbf'},
  {'C': 1, 'kernel': 'linear'},
  {'C': 10, 'kernel': 'rbf'},
  {'C': 10, 'kernel': 'linear'},
  {'C': 20, 'kernel': 'rbf'},
  {'C': 20, 'kernel': 'linear'}],
 'split0_test_score': array([0.96666667, 0.96666667, 0.96666667, 1.
, 0.96666667,
       1.        ]),
 'split1_test_score': array([1., 1., 1., 1., 1., 1.]),
 'split2_test_score': array([0.96666667, 0.96666667, 0.96666667, 0.9
, 0.9       ,
       0.9       ]),
 'split3_test_score': array([0.96666667, 0.96666667, 0.96666667,
0.96666667, 0.96666667,
       0.93333333]),
 'split4_test_score': array([1., 1., 1., 1., 1., 1.]),
 'mean_test_score': array([0.98      , 0.98      , 0.98      ,
0.97333333, 0.96666667,
       0.96666667]),
 'std_test_score': array([0.01632993, 0.01632993, 0.01632993,
0.03887301, 0.03651484,
       0.0421637 ]),
 'rank_test_score': array([1, 1, 1, 4, 5, 6])}
```

```
df=pd.DataFrame(clf.cv_results_)
df
```

```
   mean_fit_time  std_fit_time  mean_score_time  std_score_time
param_C  \
0       0.001681      0.003362         0.001678        0.003356
1
1       0.001406      0.002021         0.000202        0.000404
1
2       0.000000      0.000000         0.001601        0.003203
10
3       0.001599      0.003199         0.001601        0.003202
10
4       0.001657      0.002151         0.000802        0.001604
20
5       0.001908      0.001392         0.001001        0.002003
20

  param_kernel                           params  split0_test_score  \
0          rbf      {'C': 1, 'kernel': 'rbf'}           0.966667
1       linear   {'C': 1, 'kernel': 'linear'}           0.966667
2          rbf     {'C': 10, 'kernel': 'rbf'}           0.966667
```

```
3       linear  {'C': 10, 'kernel': 'linear'}           1.000000
4          rbf     {'C': 20, 'kernel': 'rbf'}           0.966667
5       linear  {'C': 20, 'kernel': 'linear'}           1.000000

   split1_test_score  split2_test_score  split3_test_score
split4_test_score  \
0                1.0           0.966667           0.966667
1.0
1                1.0           0.966667           0.966667
1.0
2                1.0           0.966667           0.966667
1.0
3                1.0           0.900000           0.966667
1.0
4                1.0           0.900000           0.966667
1.0
5                1.0           0.900000           0.933333
1.0

   mean_test_score  std_test_score  rank_test_score
0         0.980000        0.016330                1
1         0.980000        0.016330                1
2         0.980000        0.016330                1
3         0.973333        0.038873                4
4         0.966667        0.036515                5
5         0.966667        0.042164                6
```

```
df[['param_C','param_kernel','mean_test_score']]
```

```
  param_C param_kernel  mean_test_score
0       1          rbf         0.980000
1       1       linear         0.980000
2      10          rbf         0.980000
3      10       linear         0.973333
4      20          rbf         0.966667
5      20       linear         0.966667
```

```
dir(clf)
```

```
['__abstractmethods__',
 '__class__',
 '__delattr__',
 '__dict__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__getstate__',
```

```
'__gt__',
'__hash__',
'__init__',
'__init_subclass__',
'__le__',
'__lt__',
'__module__',
'__ne__',
'__new__',
'__reduce__',
'__reduce_ex__',
'__repr__',
'__setattr__',
'__setstate__',
'__sizeof__',
'__str__',
'__subclasshook__',
'__weakref__',
'_abc_impl',
'_check_feature_names',
'_check_n_features',
'_check_refit_for_multimetric',
'_estimator_type',
'_format_results',
'_get_param_names',
'_get_tags',
'_more_tags',
'_repr_html_',
'_repr_html_inner',
'_repr_mimebundle_',
'_required_parameters',
'_run_search',
'_select_best_index',
'_validate_data',
'_validate_params',
'best_estimator_',
'best_index_',
'best_params_',
'best_score_',
'classes_',
'cv',
'cv_results_',
'decision_function',
'error_score',
'estimator',
'fit',
'get_params',
'inverse_transform',
'multimetric_',
```

```
 'n_features_in_',
 'n_jobs',
 'n_splits_',
 'param_grid',
 'pre_dispatch',
 'predict',
 'predict_log_proba',
 'predict_proba',
 'refit',
 'refit_time_',
 'return_train_score',
 'score',
 'score_samples',
 'scorer_',
 'scoring',
 'set_params',
 'transform',
 'verbose']
```

```
clf.best_score_
```

```
0.9800000000000001
```

```
clf.best_params_
```

```
{'C': 1, 'kernel': 'rbf'}
```

Use RandomizedSearchCV to reduce number of iterations and with random combination of parameters. This is useful when you have too many parameters to try and your training time is longer. It helps reduce the cost of computation

```
from sklearn.model_selection import RandomizedSearchCV
rs = RandomizedSearchCV(svm.SVC(gamma='auto'), {
        'C': [1,10,20],
        'kernel': ['rbf','linear']
    },
    cv=5,
    return_train_score=False,
    n_iter=2
)
rs.fit(iris.data, iris.target)
pd.DataFrame(rs.cv_results_)
[['param_C','param_kernel','mean_test_score']]
```

```
  param_C param_kernel  mean_test_score
0      20       linear         0.966667
1       1          rbf         0.980000
```

How about different models with different hyperparameters?

```
from sklearn import svm
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression

model_params = {
    'svm': {
        'model': svm.SVC(gamma='auto'),
        'params' : {
            'C': [1,10,20],
            'kernel': ['rbf','linear']
        }
    },
    'random_forest': {
        'model': RandomForestClassifier(),
        'params' : {
            'n_estimators': [1,5,10]
        }
    },
    'logistic_regression' : {
        'model':
LogisticRegression(solver='liblinear',multi_class='auto'),
        'params': {
            'C': [1,5,10]
        }
    }
}

scores = []

for model_name, mp in model_params.items():
    clf =  GridSearchCV(mp['model'], mp['params'], cv=5,
return_train_score=False)
    clf.fit(iris.data, iris.target)
    scores.append({
        'model': model_name,
        'best_score': clf.best_score_,
        'best_params': clf.best_params_
    })

df = pd.DataFrame(scores,columns=['model','best_score','best_params'])
df
```

```
                model  best_score                best_params
0                 svm    0.980000  {'C': 1, 'kernel': 'rbf'}
1       random_forest    0.960000       {'n_estimators': 1}
2  logistic_regression    0.966667                   {'C': 5}
```

Based on above, I can conclude that SVM with C=1 and kernel='rbf' is the best model for solving my problem of iris flower classification

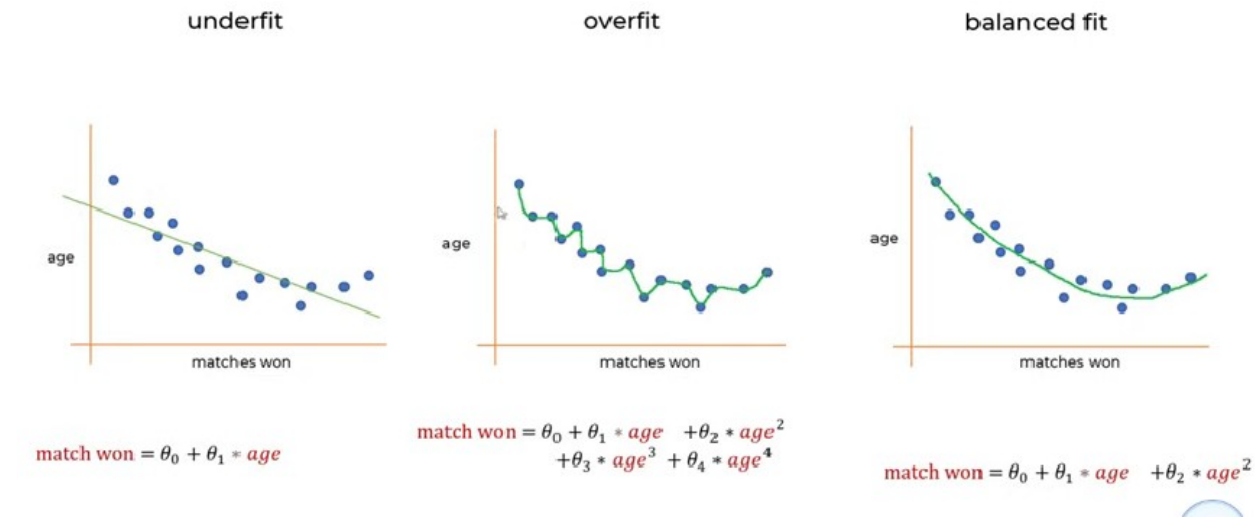
Exercise: Machine Learning Finding Optimal Model and Hyperparameters

For digits dataset in sklearn.dataset, please try following classifiers and find out the one that gives best performance. Also find the optimal parameters for that classifier.

```
from sklearn import svm
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.naive_bayes import MultinomialNB
from sklearn.tree import DecisionTreeClassifier
```

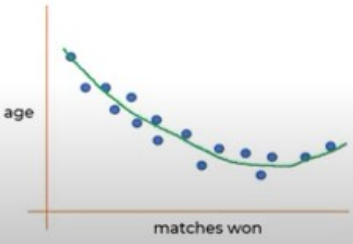
Solution

L1 and L2 Regularization | Lasso, Ridge Regression



$$\text{match won} = \theta_0 + \theta_1 * age + \theta_2 * age^2 + \theta_3 * age^3 + \theta_4 * age^4$$

Try to make θ_3 and $\theta_4$almost close to zero



$$\text{match won} = \theta_0 + \theta_1 * age + \theta_2 * age^2$$

Mean Squared Error

$$mse = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - y_{predicted}\right)^2$$

Mean Squared Error

$$mse = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - h_\theta(x_i)\right)^2$$

$$h_\theta(x_i) = \theta_0 + \theta_1 x_1 + \theta_2 x_2^2 + \theta_3 x_3^3$$

here X1,X2,X3 are features such as sepal lenth,width,area,etc

L2 Regularization

$$mse = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - h_\theta(x_i)\right)^2 + \lambda \sum_{i=1}^{n} \theta_i^2$$

$$h_\theta(x_i) = \theta_0 + \theta_1 x_1 + \theta_2 x_2^2 + \theta_3 x_3^3$$

therefore,We would have to keep THETA values small in order to keep MSE low

L1 Regularization

$$mse = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - h_\theta(x_i)\right)^2 + \lambda \sum_{i=1}^{n}|\theta_i|$$

$$h_\theta(x_i) = \theta_0 + \theta_1 x_1 + \theta_2 x_2^2 + \theta_3 x_3^3$$

```
import pandas as pd
dataset=pd.read_csv(r"C:\Users\Ayush\OneDrive\Documents\
melb_data.csv")
dataset.head()
```

```
       Suburb           Address  Rooms Type      Price Method SellerG
\
0  Abbotsford      85 Turner St      2    h  1480000.0      S  Biggin

1  Abbotsford   25 Bloomburg St      2    h  1035000.0      S  Biggin

2  Abbotsford      5 Charles St      3    h  1465000.0     SP  Biggin

3  Abbotsford  40 Federation La      3    h   850000.0     PI  Biggin

4  Abbotsford       55a Park St      4    h  1600000.0     VB  Nelson


         Date  Distance  Postcode  ...  Bathroom  Car  Landsize
BuildingArea  \
0  3/12/2016       2.5    3067.0  ...       1.0  1.0     202.0
NaN
1  4/02/2016       2.5    3067.0  ...       1.0  0.0     156.0
79.0
2  4/03/2017       2.5    3067.0  ...       2.0  0.0     134.0
150.0
3  4/03/2017       2.5    3067.0  ...       2.0  1.0      94.0
NaN
4  4/06/2016       2.5    3067.0  ...       1.0  2.0     120.0
```

```
142.0

   YearBuilt  CouncilArea Lattitude  Longtitude             Regionname
\
0        NaN        Yarra  -37.7996    144.9984  Northern Metropolitan

1     1900.0        Yarra  -37.8079    144.9934  Northern Metropolitan

2     1900.0        Yarra  -37.8093    144.9944  Northern Metropolitan

3        NaN        Yarra  -37.7969    144.9969  Northern Metropolitan

4     2014.0        Yarra  -37.8072    144.9941  Northern Metropolitan


  Propertycount
0        4019.0
1        4019.0
2        4019.0
3        4019.0
4        4019.0

[5 rows x 21 columns]
```

```
dataset.nunique()
```

```
Suburb             314
Address          13378
Rooms                9
Type                 3
Price             2204
Method               5
SellerG            268
Date                58
Distance           202
Postcode           198
Bedroom2            12
Bathroom             9
Car                 11
Landsize          1448
BuildingArea       602
YearBuilt          144
CouncilArea         33
Lattitude         6503
Longtitude        7063
Regionname           8
Propertycount      311
dtype: int64
```

```
dataset.shape
```

```
(12211, 613)
```

```
cols_to_use=['Suburb','Rooms','Type','Method','SellerG','Regionname','
Propertycount',

'Distance','CouncilArea','Bedroom2','Bathroom','Car','Landsize','Build
ingArea','Price']

dataset=dataset[cols_to_use]
dataset.head()
```

```
---------------------------------------------------------------------------
-----
KeyError                                  Traceback (most recent call
last)
Cell In[84], line 1
----> 1 dataset=dataset[cols_to_use]
      2 dataset.head()

File C:\ProgramData\anaconda3\lib\site-packages\pandas\core\
frame.py:3813, in DataFrame.__getitem__(self, key)
   3811     if is_iterator(key):
   3812         key = list(key)
-> 3813     indexer = self.columns._get_indexer_strict(key, "columns")
[1]
   3815 # take() does not accept boolean indexers
   3816 if getattr(indexer, "dtype", None) == bool:

File C:\ProgramData\anaconda3\lib\site-packages\pandas\core\indexes\
base.py:6070, in Index._get_indexer_strict(self, key, axis_name)
   6067 else:
   6068     keyarr, indexer, new_indexer =
self._reindex_non_unique(keyarr)
-> 6070 self._raise_if_missing(keyarr, indexer, axis_name)
   6072 keyarr = self.take(indexer)
   6073 if isinstance(key, Index):
   6074     # GH 42790 - Preserve name from an Index

File C:\ProgramData\anaconda3\lib\site-packages\pandas\core\indexes\
base.py:6133, in Index._raise_if_missing(self, key, indexer,
axis_name)
   6130     raise KeyError(f"None of [{key}] are in the
[{axis_name}]")
   6132 not_found = list(ensure_index(key)[missing_mask.nonzero()
[0]].unique())
-> 6133 raise KeyError(f"{not_found} not in index")

KeyError: "['Suburb', 'Type', 'Method', 'SellerG', 'Regionname',
'CouncilArea'] not in index"
```

```
dataset.shape
```

```
(12211, 613)
```

```
dataset.isna().sum()
```

```
Rooms                          0
Propertycount                  0
Distance                       0
Bedroom2                       0
Bathroom                       0
                              ..
CouncilArea_Whitehorse         0
CouncilArea_Whittlesea         0
CouncilArea_Wyndham            0
CouncilArea_Yarra              0
CouncilArea_Yarra Ranges       0
Length: 613, dtype: int64
```

```
cols_to_fill_zero=['Propertycount','Distance','Bedroom2','Bathroom','C
ar']
dataset[cols_to_fill_zero]=dataset[cols_to_fill_zero].fillna(0)
dataset.isna().sum()
```

```
Rooms                          0
Propertycount                  0
Distance                       0
Bedroom2                       0
Bathroom                       0
                              ..
CouncilArea_Whitehorse         0
CouncilArea_Whittlesea         0
CouncilArea_Wyndham            0
CouncilArea_Yarra              0
CouncilArea_Yarra Ranges       0
Length: 613, dtype: int64
```

```
dataset['Landsize']=dataset['Landsize'].fillna(dataset.Landsize.mean()
)
dataset['BuildingArea']=dataset['BuildingArea'].fillna(dataset.Buildin
gArea.mean())
```

```
dataset.isna().sum()
```

```
Rooms                          0
Propertycount                  0
Distance                       0
Bedroom2                       0
Bathroom                       0
                              ..
CouncilArea_Whitehorse         0
CouncilArea_Whittlesea         0
CouncilArea_Wyndham            0
```

```
CouncilArea_Yarra           0
CouncilArea_Yarra Ranges    0
Length: 613, dtype: int64
```

```
dataset.dropna(inplace=True)
dataset.isna().sum()
```

```
Rooms                       0
Propertycount               0
Distance                    0
Bedroom2                    0
Bathroom                    0
                           ..
CouncilArea_Whitehorse      0
CouncilArea_Whittlesea      0
CouncilArea_Wyndham         0
CouncilArea_Yarra           0
CouncilArea_Yarra Ranges    0
Length: 613, dtype: int64
```

```
dataset=pd.get_dummies(dataset,drop_first=True)
dataset.head()
```

```
   Rooms  Propertycount  Distance  Bedroom2  Bathroom  Car
Landsize  \
0      2         4019.0       2.5       2.0       1.0  1.0
202.0

1      2         4019.0       2.5       2.0       1.0  0.0
156.0

2      3         4019.0       2.5       3.0       2.0  0.0
134.0

3      3         4019.0       2.5       3.0       2.0  1.0
 94.0

4      4         4019.0       2.5       3.0       1.0  2.0
120.0


   BuildingArea      Price  Suburb_Aberfeldie  ...
CouncilArea_Moreland  \
0     151.96765  1480000.0                  0  ...
0
1      79.00000  1035000.0                  0  ...
0
2     150.00000  1465000.0                  0  ...
0
3     151.96765   850000.0                  0  ...
0
4     142.00000  1600000.0                  0  ...
0

   CouncilArea_Nillumbik  CouncilArea_Port Phillip
CouncilArea_Stonnington  \
```

```
0                         0                              0
0
1                         0                              0
0
2                         0                              0
0
3                         0                              0
0
4                         0                              0
0

   CouncilArea_Unavailable  CouncilArea_Whitehorse
CouncilArea_Whittlesea  \
0                          0                         0
0
1                          0                         0
0
2                          0                         0
0
3                          0                         0
0
4                          0                         0
0

   CouncilArea_Wyndham  CouncilArea_Yarra  CouncilArea_Yarra Ranges
0                    0                  1                         0
1                    0                  1                         0
2                    0                  1                         0
3                    0                  1                         0
4                    0                  1                         0

[5 rows x 613 columns]
```

Drop first to drop first column to prevent dummy data trap

```
x=dataset.drop('Price',axis=1)
y=dataset['Price']

from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,rando
m_state=2)

from sklearn.linear_model import LinearRegression
reg = LinearRegression().fit(x_train,y_train)

reg.score(x_test,y_test)

-998345369.4972584

reg.score(x_train,y_train)
```

```
0.712092741110876
```

```
from sklearn import linear_model

lasso_reg=linear_model.Lasso(alpha=50,max_iter=100,tol=0.1)

lasso_reg.fit(x_train,y_train)
```

```
C:\ProgramData\anaconda3\lib\site-packages\sklearn\linear_model\
_coordinate_descent.py:631: ConvergenceWarning: Objective did not
converge. You might want to increase the number of iterations, check
the scale of the features or consider increasing regularisation.
Duality gap: 5.272e+14, tolerance: 3.475e+14
  model = cd_fast.enet_coordinate_descent(
```

```
Lasso(alpha=50, max_iter=100, tol=0.1)
```

#Lasso is L1 reg.

```
lasso_reg.score(x_test,y_test)
```

```
0.6635378648697849
```

```
lasso_reg.score(x_train,y_train)
```

```
0.7075207761454996
```

```
from sklearn.linear_model import Ridge
ridge_reg=Ridge(alpha=50,max_iter=100,tol=0.1)
ridge_reg.fit(x_train,y_train)
```

```
Ridge(alpha=50, max_iter=100, tol=0.1)
```

```
ridge_reg.score(x_test,y_test)
```

```
0.6644072458301125
```

```
ridge_reg.score(x_train,y_train)
```

```
0.6767113841211165
```

Ridge is L2 regression

```
```