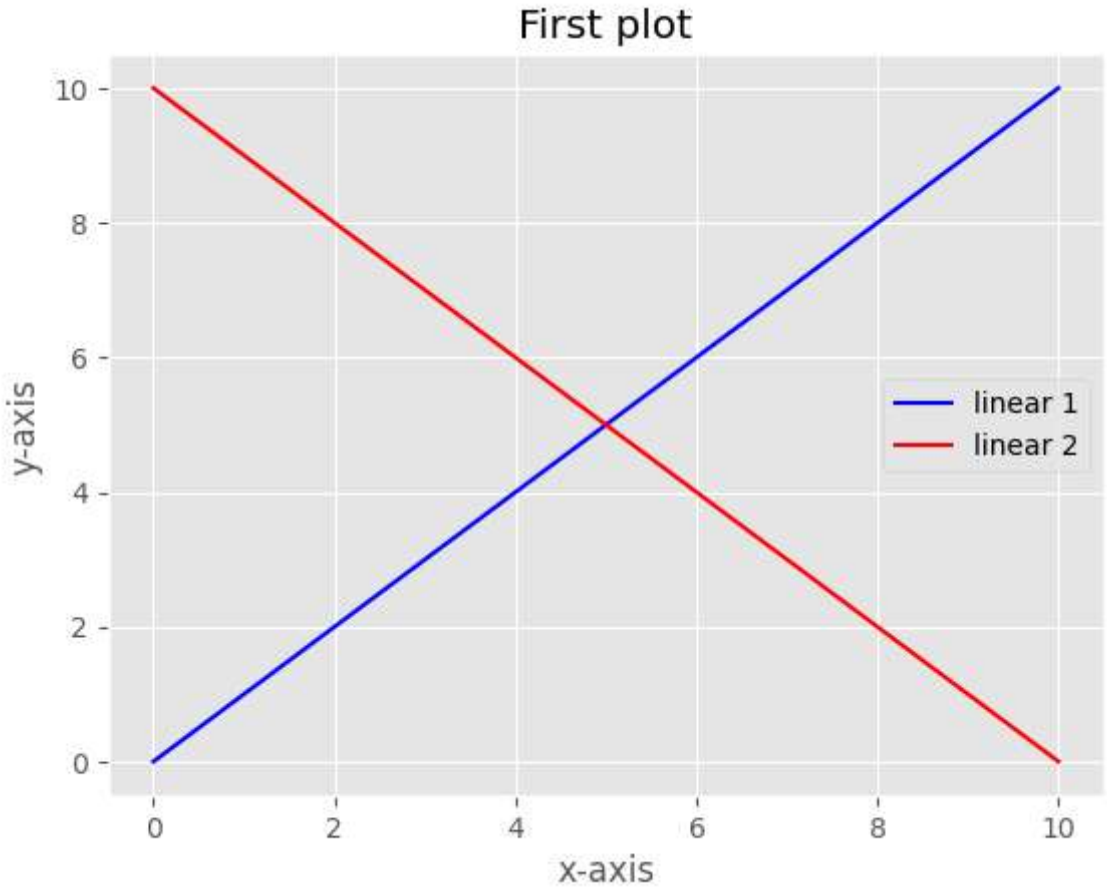
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
x=np.linspace(10,0,100)
y=np.linspace(0,10,100)

plt.plot(x,x,label='linear 1',color='b')
plt.plot(x,y,label='linear 2',color='r')
plt.legend()
plt.xlabel("x-axis")
plt.ylabel("y-axis")
plt.title("First plot")
plt.show
```

```
<function matplotlib.pyplot.show(close=None, block=None)>
```



```
import numpy as np
import pandas as pd
df = pd.read_csv("test.csv")
print(df)
print("\nhead\n")
print(df.head())
print("\ntail\n")
print(df.tail())
print("\ntail 2\n")
print(df.tail(2))
print("\nindex\n")
print(df.index)
print("\ncolumn\n")
print(df.columns)
print("\ndescribe\n")
print(df.describe)
print("\nsort\n")
print(df.sort_values(by="Period",ascending=False))
print("\nT\n")
print(df.T)
```

```
              1  Period
     1503  1505    1504
```

```
1500  1502    1501
...    ...     ...
4        6       5
3        5       4
2        4       3
1        3       2
0        2       1

[1505 rows x 2 columns]
```

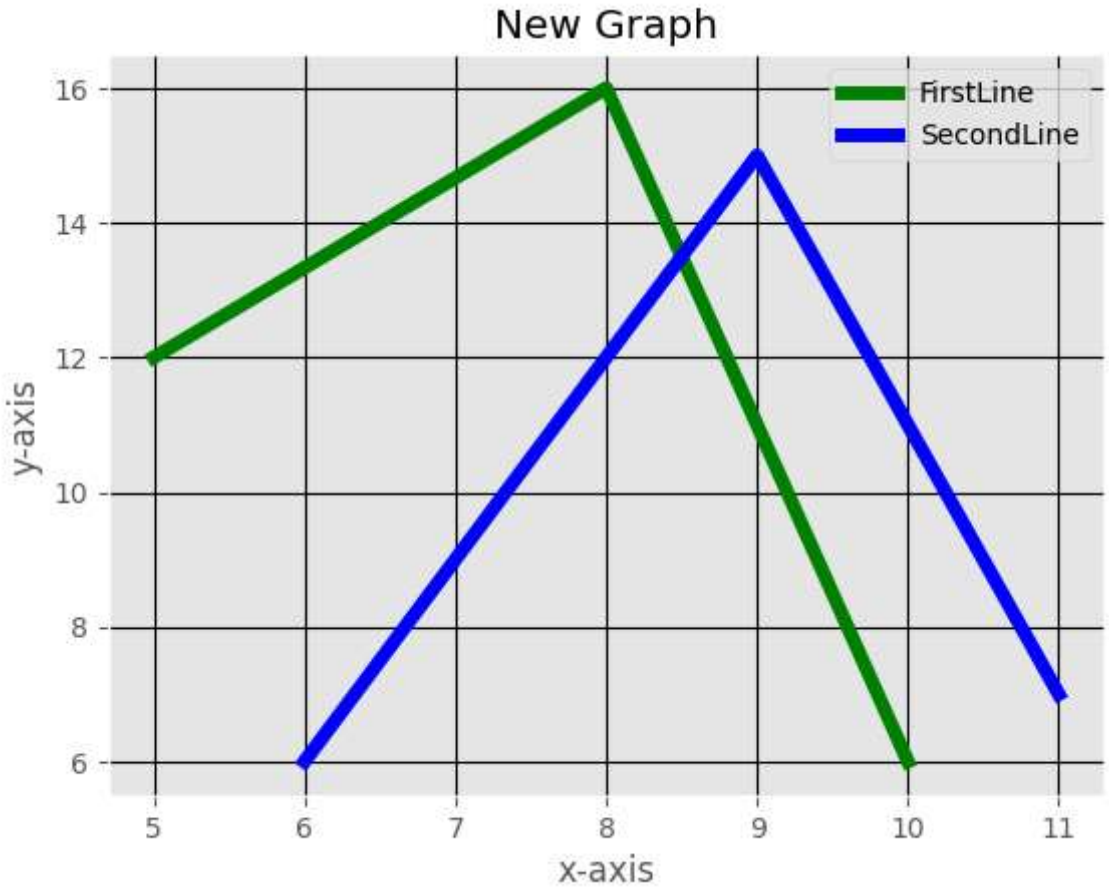
T

```
        0     1     2     3     4     5     6     7     8     9     ...  1495  \
1          2     3     4     5     6     7     8     9    10    11  ...  1497
Period     1     2     3     4     5     6     7     8     9    10  ...  1496

        1496  1497  1498  1499  1500  1501  1502  1503  1504
1       1498  1499  1500  1501  1502  1503  1504  1505  1506
Period  1497  1498  1499  1500  1501  1502  1503  1504  1505

[2 rows x 1505 columns]
```

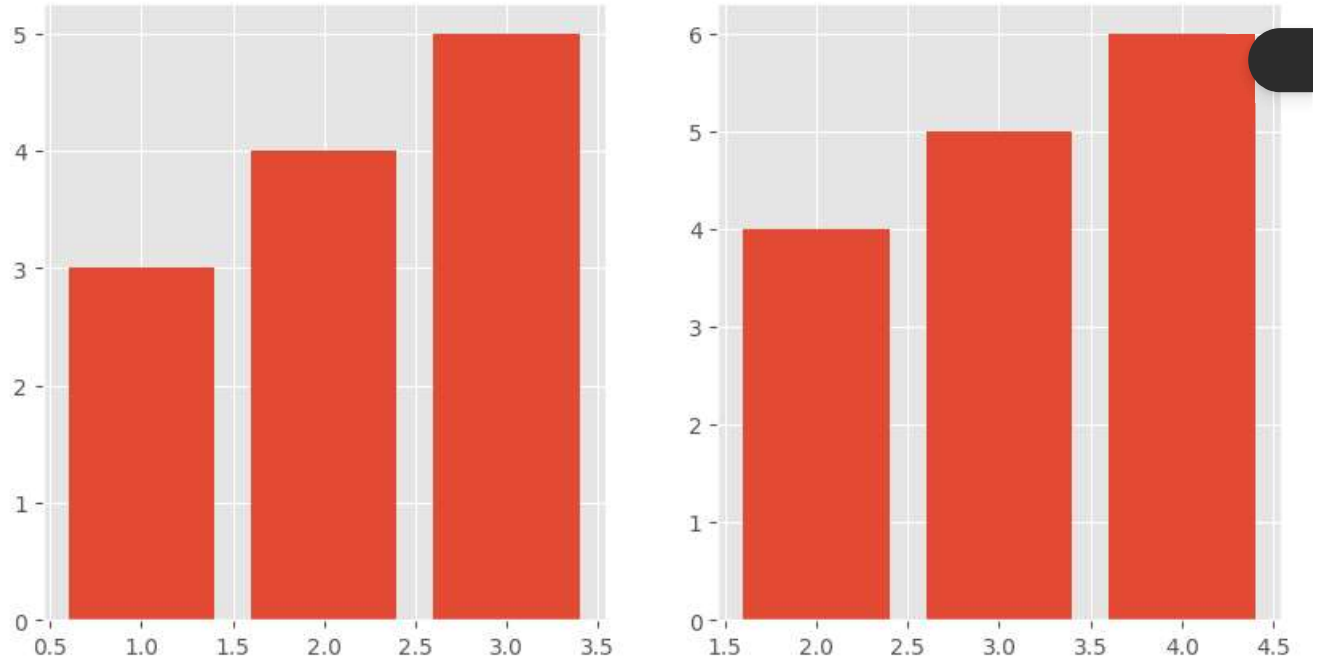
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
plt.bar([1, 2, 3, 4, 5],[88, 4, 56, 1, 23], label="car1", color="b",width=5)
plt.bar([1, 2, 3, 4, 5], [ 5, 4, 3, 2, 1] ,label="car2",color="g",width=5)
plt.legend()
plt.title("new graph")
plt.xlabel("x axis")
plt.ylabel("y axis")
plt.show()
```

new graph

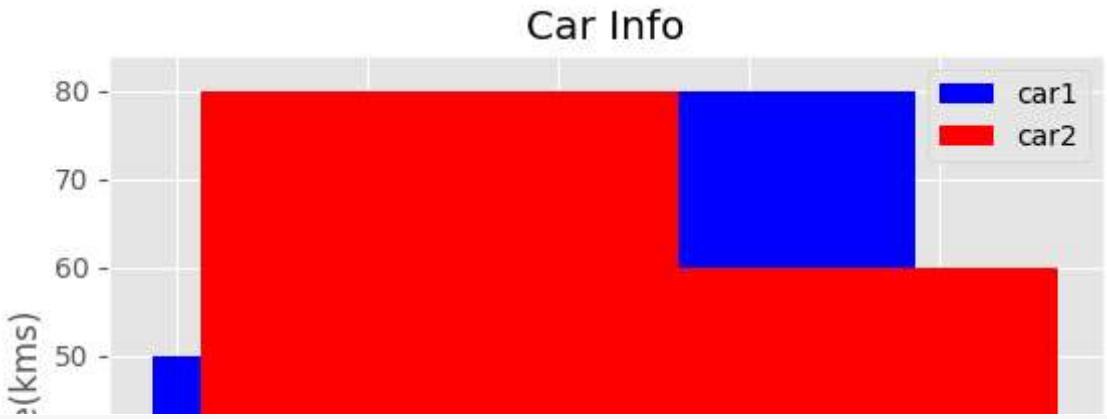
```
#18. plotting multiline with background
from matplotlib import style
style.use('ggplot')
x1=[5,8,10]
y1=[12,16,6]
x2=[6,9,11]
y2=[6,15,7]
plt.plot(x1,y1,'g',label="FirstLine",linewidth=5)
plt.plot(x2,y2,'b',label="SecondLine",linewidth=5)
plt.title("New Graph")
plt.ylabel("y-axis")
plt.xlabel("x-axis")
plt.legend()
plt.grid(True,color='k')
plt.show()
```



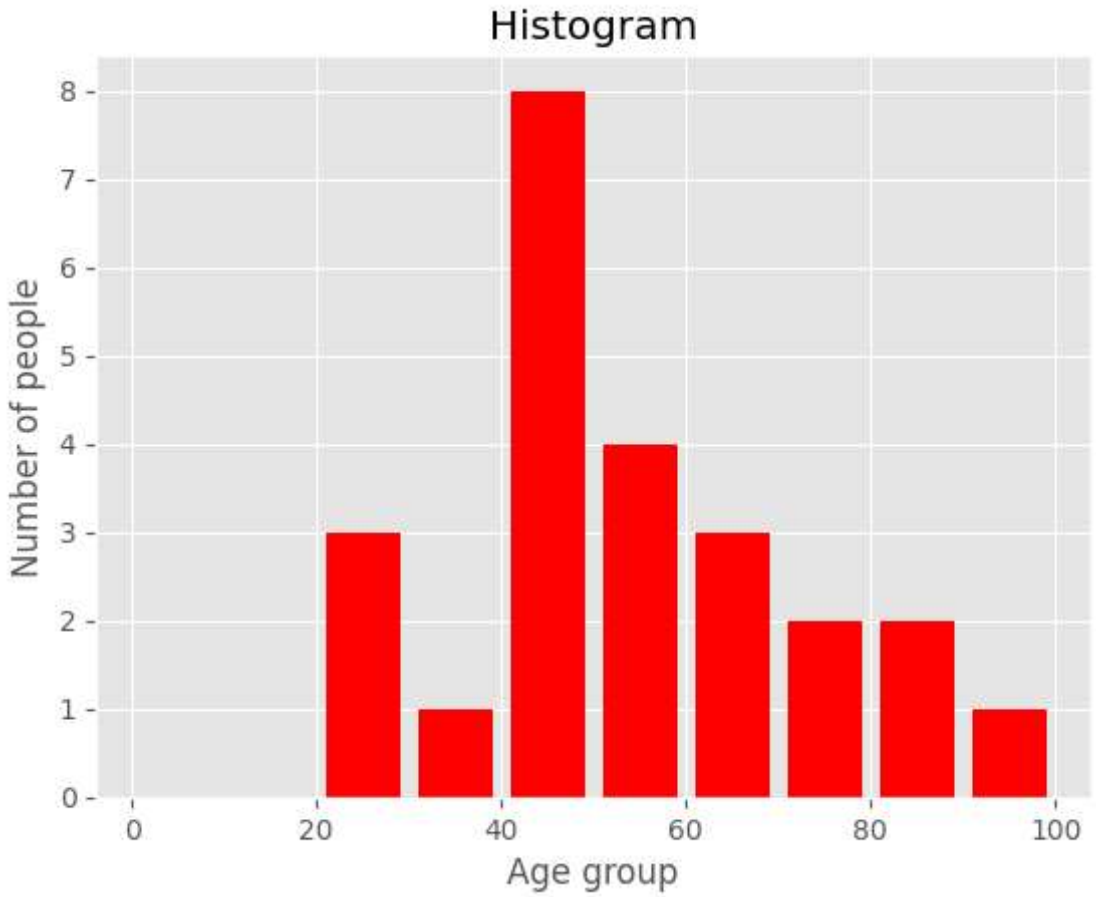
```
#19. Plotting bar graph
import matplotlib.pyplot as plt

fig = plt.figure(figsize=(10, 5))

ax1 = fig.add_subplot(121)
ax2 = fig.add_subplot(122)

ax1.bar([1, 2, 3], [3, 4, 5])
ax2.bar([2, 3, 4], [4, 5, 6])

plt.show()
```




```
#20. Plot Multibar
plt.bar([0.25,1.25,2.25,3.25,4.25],[50,40,70,80,20],label='car1',color='b',width=5)
plt.bar([0.75,1.75,2.75,3.75,4.75],[80,20,20,50,60],label='car2',color='r',width=5)
plt.legend()
plt.xlabel('Days')
plt.ylabel('Distance(kms)')
plt.title('Car Info')
plt.show()
```




```
#21. Plot histogram
population_age=[22,55,62,45,21,22,34,42,42,42,102,95,85,55,110,120,70,65,55,111,115,80,75
bins=[0,10,20,30,40,50,60,70,80,90,100]
plt.hist(population_age,bins,histtype='bar',color='r',rwidth=0.8)
plt.xlabel("Age group")
plt.ylabel("Number of people")
plt.title('Histogram')
plt.show()
```



ML Model housing

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

```
df = pd.read_csv("/content/housing.csv")
df.head()
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population
0	-122.23	37.88	41.0	880.0	129.0	322.0
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0
2	-122.24	37.85	52.0	1467.0	190.0	496.0
3	-122.25	37.85	52.0	1274.0	235.0	558.0
4	-122.25	37.85	52.0	1627.0	280.0	565.0

```
from sklearn.model_selection import train_test_split
df.dropna(inplace=True)
train_set,test_set=train_test_split(df,test_size=0.2,random_state=42)
X_train=train_set.iloc[:,:-2]
X_train
# y1_train=train_set.iloc[:,-2:]
# y1_train
y_train=train_set.iloc[:,-2:-1]
y_train
X_test=test_set.iloc[:,:-2]
X_test
# y1_test=test_set.iloc[:,-2:]
# y1_test
y_test=test_set.iloc[:,-2:-1]
y_test
```

	median_house_value
14425	80100.0
16398	500001.0
7721	352100.0
1411	187500.0
1336	361000.0
...	...
8285	163100.0
6264	229200.0
2999	327500.0
13452	58300.0
14809	500001.0

4087 rows × 1 columns

```
# from sklearn.model_selection import train_test_split
# df.dropna(inplace=True)
# train_set,test_set=train_test_split(df,test_size=0.2,random_state=42)
# X_train=train_set.iloc[:,:-2]
# X_train
# y1_train=train_set.iloc[:,-2:]
# y1_train
# y_train=y1_train.iloc[:,:1]
# y_train
# X_test=test_set.iloc[:,:-2]
# X_test
# y1_test=test_set.iloc[:,-2:]
# y1_test
# y_test=y1_test.iloc[:,:1]
# y_test
```

```
from sklearn.linear_model import LinearRegression
#print(train_X)
model=LinearRegression()
model.fit(X_train,y_train)
```

▾ LinearRegression

LinearRegression()

```
#testing model
y_pred=model.predict(X_test)
print(y_pred)
```

```
[[ 62720.63712627]
 [414459.89124757]
 [238028.6841184 ]
 ...
 [282366.97531626]
 [ 77825.94834712]
 [387233.21922771]]
```

```
from sklearn.metrics import mean_squared_error
rmse=mean_squared_error(y_pred,y_test,squared=False)
rmse
```

```
70025.94402055604
```

```
X_sample=[[-122.23,37.88,41.0,880.0,129.0,322.0,126.0,8.3252]]
y_pred=model.predict(X_sample)
print(y_pred)
```

```
[[411052.90996556]]
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not
  warnings.warn(
```

```
# longitude = input("longitude\n")
# latitude = input("latitude\n")
# housing_median_age = input("housing_median_age\n")
# total_rooms = input("total_rooms\n")
# total_bedrooms = input("total_bedrooms\n")
# population = input("population\n")
# households = input("households\n")
# median_income = input("median_income\n")
# X_sample1=[[longitude,latitude,housing_median_age,total_rooms,total_bedrooms,population
# y_pred=model.predict(X_sample)
# print("house price =")
# print(y_pred)
```

```
#checking Decision tree Regressor

from sklearn.tree import DecisionTreeRegressor
model2 = DecisionTreeRegressor()
model2.fit(X_train,y_train)
```

```
▾ DecisionTreeRegressor
DecisionTreeRegressor()
```

```
#testing model
y_pred=model2.predict(X_test)
print(y_pred)
```

```
[ 78500. 500001. 235500. ... 342500.  60900. 374000.]
```

```
from sklearn.metrics import mean_squared_error
rmse=mean_squared_error(y_pred,y_test,squared=False)
rmse
```

```
69501.52492301987
```

```
#Using Cross Validation

from sklearn.model_selection import cross_val_score
model2_rmses = -cross_val_score(model2, X_train, y_train, scoring = "neg_root_mean_square
model2_rmses
pd.Series(model2_rmses).describe()
```

```
count       10.000000
mean     70278.875928
std       1891.063101
min      66722.422826
25%      69387.047113
50%      70306.525198
75%      71507.897823
max      73235.984580
dtype: float64
```

```
#checking Random Forest Regressor

from sklearn.ensemble._forest import RandomForestRegressor
model3 = RandomForestRegressor()
model3.fit(X_train,y_train)
```

```
<ipython-input-250-aee46ef53514>:5: DataConversionWarning: A column-vector y was pass
  model3.fit(X_train,y_train)
▾ RandomForestRegressor
RandomForestRegressor()
```

```
#testing model
y_pred=model3.predict(X_test)
print(y_pred)
```

```
[ 71916.   475829.58 253150.   ... 296846.05  73683.   432412.32]
```

```
from sklearn.metrics import mean_squared_error
rmse=mean_squared_error(y_pred,y_test,squared=False)
rmse
```

```
49636.368617575325
```

```
#using Cross Validation
from sklearn.model_selection import cross_val_score
model3_rmses = -cross_val_score(model2, X_train, y_train, scoring = "neg_root_mean_square
model3_rmses
pd.Series(model3_rmses).describe()
```

```
count       10.000000
mean     69720.629159
std       1472.164911
min      66317.550887
25%      69340.766212
50%      70004.823119
75%      70800.943073
max      71196.083567
dtype: float64
```

ML Model for Salary Dataset

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

```
df = pd.read_csv("/content/Salary_dataset.csv")
df.head()
```

	Unnamed: 0	YearsExperience	Salary
0	0	1.2	39344.0
1	1	1.4	46206.0
2	2	1.6	37732.0
3	3	2.1	43526.0
4	4	2.3	39892.0

```
from sklearn.model_selection import train_test_split
df.dropna(inplace=True)
train_set,test_set=train_test_split(df,test_size=0.2,random_state=42)
X_train=train_set.iloc[:,:-1]
X_train
y1_train=train_set.iloc[:,-2:]
y1_train
y_train=y1_train.iloc[:,:1]
y_train
X_test=test_set.iloc[:,-2:-1]
X_test
y1_test=test_set.iloc[:,-2:]
y1_test
y_test=y1_test.iloc[:,:1]
y_test
```

	YearsExperience
27	9.7
15	5.0
23	8.3
17	5.4
8	3.3
9	3.8

```
df.shape
```

```
(30, 3)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 3 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   Unnamed: 0       30 non-null     int64  
 1   YearsExperience  30 non-null     float64
 2   Salary           30 non-null     float64
dtypes: float64(2), int64(1)
memory usage: 848.0 bytes
```

```
df.describe()
```

	Unnamed: 0	YearsExperience	Salary
count	30.000000	30.000000	30.000000
mean	14.500000	5.413333	76004.000000
std	8.803408	2.837888	27414.429785
min	0.000000	1.200000	37732.000000
25%	7.250000	3.300000	56721.750000
50%	14.500000	4.800000	65238.000000
75%	21.750000	7.800000	100545.750000
max	29.000000	10.600000	122392.000000

```
df.isnull().sum()
```

```
Unnamed: 0          0
YearsExperience     0
Salary              0
dtype: int64
```

```
df[['YearsExperience','Salary']].cov()
```

	YearsExperience	Salary
YearsExperience	8.053609	7.610630e+04
Salary	76106.303448	7.515510e+08

```
X=df.drop('Salary',axis=1)
```

```
y=df.Salary
```

```
X.head()
```

	Unnamed: 0	YearsExperience
0	0	1.2
1	1	1.4
2	2	1.6
3	3	2.1
4	4	2.3

```
y.head()
```

```
0    39344.0
1    46206.0
2    37732.0
3    43526.0
4    39892.0
Name: Salary, dtype: float64
```

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,random_state=0,test_size=0.30)
print(X_train.shape)
X_test.shape
```

```
(21, 2)
(9, 2)
```

```
from sklearn.linear_model import LinearRegression
#print(train_X)
model=LinearRegression()
model.fit(X_train,y_train)
```

```
▾ LinearRegression
LinearRegression()
```

```
#testing model
y_pred=model.predict(X_test)
print(y_pred)
```

```
[ 41068.96469651 124994.63666745  63523.55124173  63316.24056634
 117108.4126212  109222.18857494 117564.91702613  63772.74497127
  74647.18917665]
```

```
from sklearn.metrics import mean_squared_error
rmse=mean_squared_error(y_pred,y_test,squared=False)
rmse
```

```
4929.255926250668
```

```
y_pred=model.predict(X_test)
print(y_pred)
```

```
[ 41068.96469651 124994.63666745  63523.55124173  63316.24056634
 117108.4126212  109222.18857494 117564.91702613  63772.74497127
  74647.18917665]
```

```
#checking Decision tree Regressor

from sklearn.tree import DecisionTreeRegressor
model2 = DecisionTreeRegressor()
model2.fit(X_train,y_train)
```

```
▾ DecisionTreeRegressor
DecisionTreeRegressor()
```

```
#testing model
y_pred=model2.predict(X_test)
print(y_pred)
```

```
[ 46206. 121873.  56958.  57190. 105583. 105583. 105583.  56958.  66030.]
```

```
from sklearn.metrics import mean_squared_error
rmse=mean_squared_error(y_pred,y_test,squared=False)
rmse
```

```
8131.063091625842
```

```
#Using Cross Validation

from sklearn.model_selection import cross_val_score
model2_rmses = -cross_val_score(model2, X_train, y_train, scoring = "neg_root_mean_square
model2_rmses
pd.Series(model2_rmses).describe()
```

```
count       10.000000
mean      7702.111486
std       3476.187428
min       3232.645743
25%       4618.423468
50%       7631.134269
75%      10460.976139
max      12499.022842
dtype: float64
```

```
#checking Random Forest Regressor

from sklearn.ensemble._forest import RandomForestRegressor
model3 = RandomForestRegressor()
model3.fit(X_train,y_train)
```

```
▾ RandomForestRegressor
RandomForestRegressor()
```

```
#testing model
y_pred=model3.predict(X_test)
print(y_pred)
```

```
[ 44241.59 116046.6   58549.96  58206.66 109943.8  108748.42 110508.
  57779.96  69409.95]
```

```
from sklearn.metrics import mean_squared_error
rmse=mean_squared_error(y_pred,y_test,squared=False)
rmse
```

```
6283.461087346685
```

```
#using Cross Validation
from sklearn.model_selection import cross_val_score
model3_rmses = -cross_val_score(model2, X_train, y_train, scoring = "neg_root_mean_square
model3_rmses
pd.Series(model3_rmses).describe()
```

```
count       10.000000
mean      7424.631240
std       3097.082095
min       2893.969938
25%       5828.508358
50%       6886.732808
75%       9625.547162
max      12499.022842
dtype: float64
```

weight-height

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

```
df = pd.read_csv("/content/weight-height.csv")
df
```

	Gender	Height	Weight

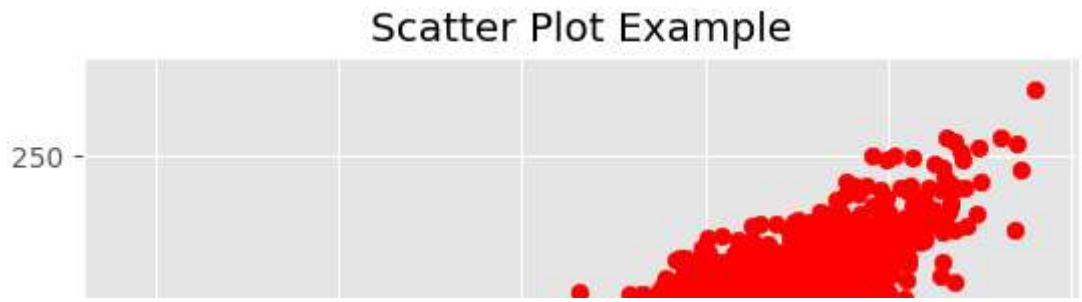
```
x = df['Height']
#x = df.iloc[:1]
x
```

```
0       73.847017
1       68.781904
2       74.110105
3       71.730978
4       69.881796
          ...
9995    66.172652
9996    67.067155
9997    63.867992
9998    69.034243
9999    61.944246
Name: Height, Length: 10000, dtype: float64
```

```
y = df['Weight']
#y = df.iloc[:2]
y
```

```
0       241.893563
1       162.310473
2       212.740856
3       220.042470
4       206.349801
           ...
9995    136.777454
9996    170.867906
9997    128.475319
9998    163.852461
9999    113.649103
Name: Weight, Length: 10000, dtype: float64
```

```
plt.scatter(x, y, marker = 'o', color = 'r')
plt.title("Scatter Plot Example")
plt.show()
```



Binary Classification using Logistic Reggression

```
#import
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model  import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
```

```
dataset = pd.read_csv('/content/diabetes2.csv')
dataset.head()
dataset.isnull().any()
dataset=dataset.fillna(method='ffill')
```

```
#dividing data into x & y
X = dataset.drop(columns = ['Outcome'])
y = dataset['Outcome']
```

```
#split data into test & train dataset
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2,random_state=0)
```

```
logreg = LogisticRegression(solver='liblinear')
```

```
#fitting model with data
logreg.fit(X_train,y_train)
```

```
LogisticRegression
LogisticRegression(solver='liblinear')
```

```
#predict for X_test
y_pred=logreg.predict(X_test)
```

```
from sklearn.metrics import mean_squared_error
rmse=mean_squared_error(y_pred,y_test,squared=False)
rmse
```

```
0.4264014327112209
```

```
#check the model accuracy
from sklearn import metrics
print("Accuracy:", metrics.accuracy_score(y_test,y_pred))
```

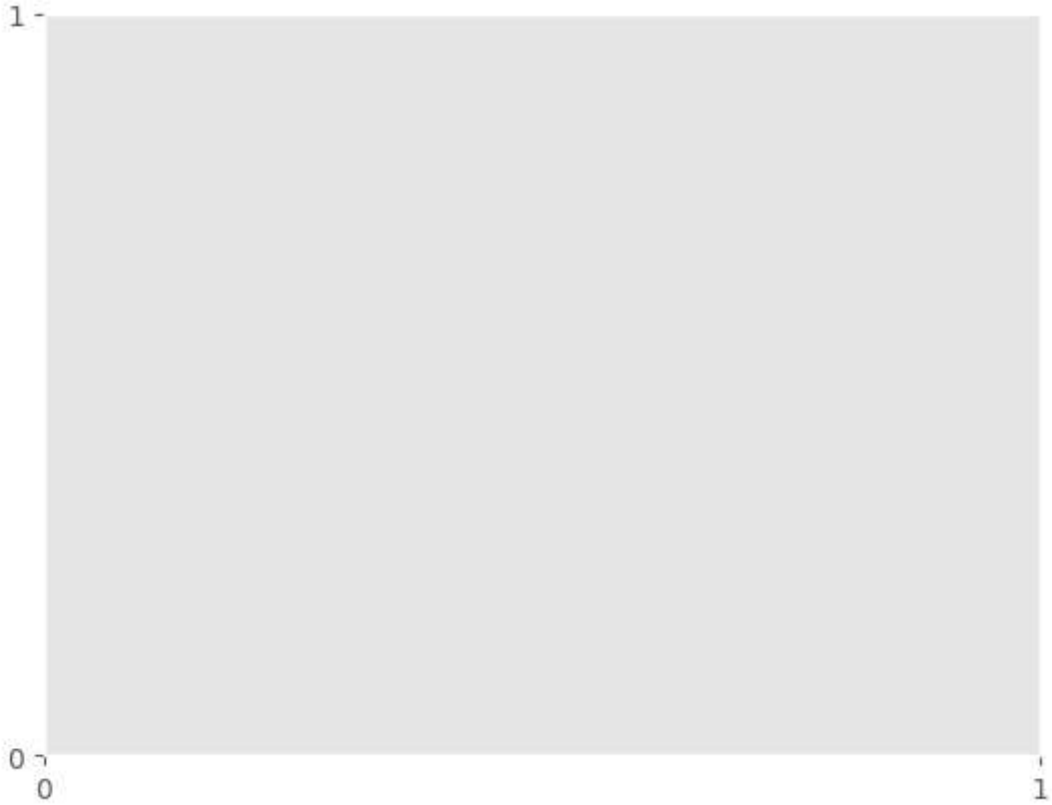
```
Accuracy: 0.8181818181818182
```

```
confusion_matrix=metrics.confusion_matrix(y_test,y_pred)
print(confusion_matrix)
```

```
[[98  9]
 [19 28]]
```

```
class_names=[0 , 1]
fig,ax=plt.subplots()
ticks_marks =  np.arange(len(class_names))
plt.xticks(ticks_marks,class_names)
plt.yticks(ticks_marks,class_names)
```

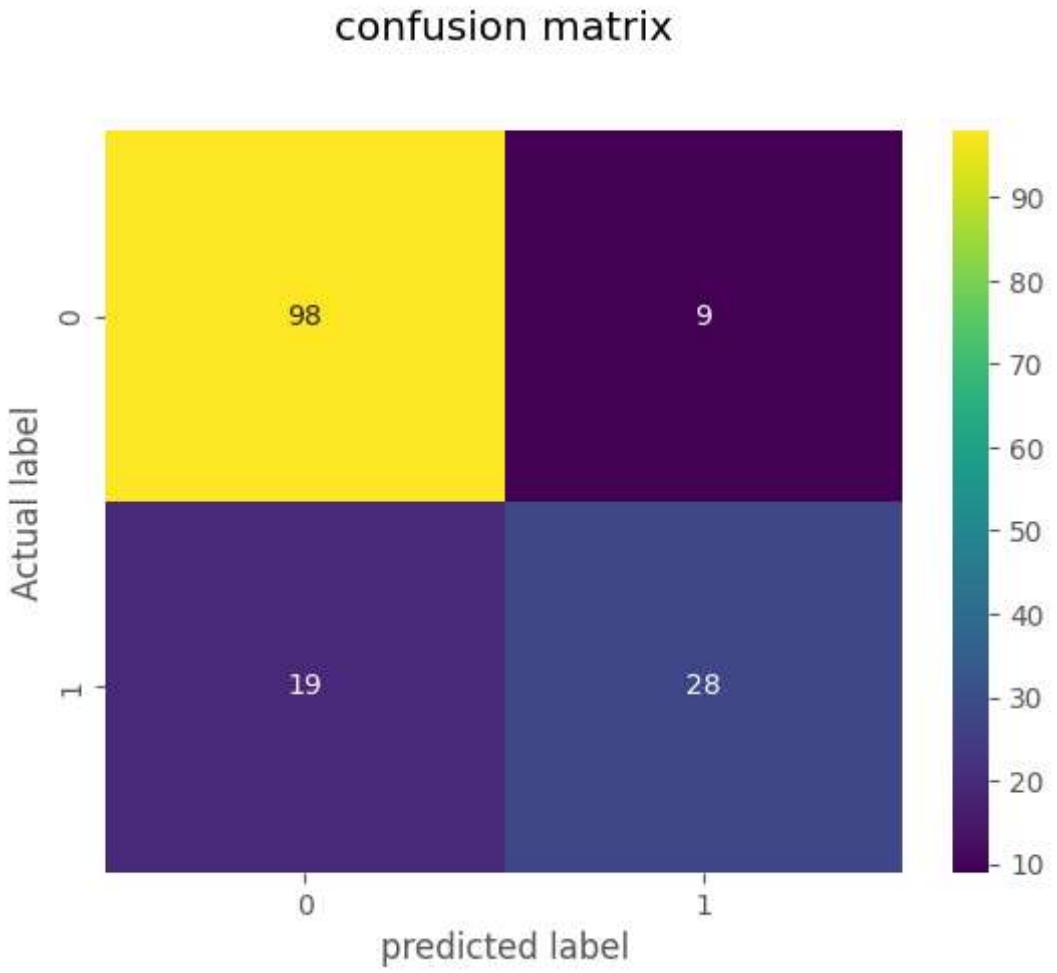
```
([<matplotlib.axis.YTick at 0x79e5446a8910>,
  <matplotlib.axis.YTick at 0x79e5446a8c70>],
 [Text(0, 0, '0'), Text(0, 1, '1')])
```




```
#create heatmap
sns.heatmap(pd.DataFrame(confusion_matrix),annot=True,cmap="viridis",fmt='g')
```

```
ax.xaxis.set_label_position("bottom")
plt.title('confusion matrix',y=1.1)
plt.ylabel('Actual label')
plt.xlabel('predicted label')
```

```
Text(0.5, 23.52222222222222, 'predicted label')
```



SVM Multicast

```
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn import datasets
from sklearn import svm
```

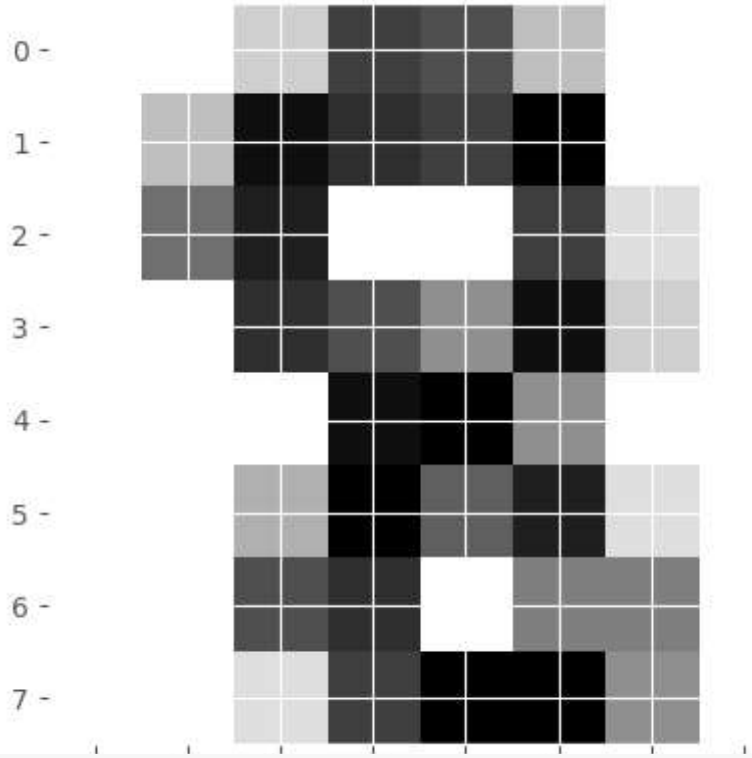
```
#load datasets
digits = datasets.load_digits()
dir(digits)
print(digits.data.shape)
print(digits.DESCR)
```

```
(1797, 64)
.. _digits_dataset:

Optical recognition of handwritten digits dataset
--------------------------------------------------
```

```
**Data Set Characteristics:**

    :Number of Instances: 1797
    :Number of Attributes: 64
    :Attribute Information: 8x8 image of integer pixels in the range 0..16.
    :Missing Attribute Values: None
    :Creator: E. Alpaydin (alpaydin '@' boun.edu.tr)
    :Date: July; 1998

This is a copy of the test set of the UCI ML hand-written digits datasets
https://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits

The data set contains images of hand-written digits: 10 classes where
each class refers to a digit.

Preprocessing programs made available by NIST were used to extract
normalized bitmaps of handwritten digits from a preprinted form. From a
total of 43 people, 30 contributed to the training set and different 13
to the test set. 32x32 bitmaps are divided into nonoverlapping blocks of
4x4 and the number of on pixels are counted in each block. This generates
an input matrix of 8x8 where each element is an integer in the range
0..16. This reduces dimensionality and gives invariance to small
distortions.

For info on NIST preprocessing routines, see M. D. Garris, J. L. Blue, G.
T. Candela, D. L. Dimmick, J. Geist, P. J. Grother, S. A. Janet, and C.
L. Wilson, NIST Form-Based Handprint Recognition System, NISTIR 5469,
1994.

.. topic:: References

  - C. Kaynak (1995) Methods of Combining Multiple Classifiers and Their
    Applications to Handwritten Digit Recognition, MSc Thesis, Institute of
    Graduate Studies in Science and Engineering, Bogazici University.
  - E. Alpaydin, C. Kaynak (1998) Cascading Classifiers, Kybernetika.
  - Ken Tang and Ponnuthurai N. Suganthan and Xi Yao and A. Kai Qin.
    Linear dimensionalityreduction using relevance weighted LDA. School of
    Electrical and Electronic Engineering Nanyang Technological University.
    2005.
  - Claudio Gentile. A New Approximate Maximal Margin Classification
    Algorithm. NIPS. 2000.
```

```
#examine data
x = digits.images.reshape((len(digits.images), -1))
x.shape
y = digits.target
y_names = digits.target_names
print(y_names)
idx = 722
print(digits.images[idx])
print(digits.target[idx])
plt.imshow(digits.images[idx],cmap='binary')
plt.show()
```

```
[0 1 2 3 4 5 6 7 8 9]
[[ 0.  0.  3. 12. 11.  4.  0.  0.]
 [ 0.  4. 15. 13. 12. 16.  0.  0.]
 [ 0.  9. 14.  0.  0. 12.  2.  0.]
 [ 0.  0. 13. 11.  7. 15.  3.  0.]
 [ 0.  0.  0. 15. 16.  7.  0.  0.]
 [ 0.  0.  5. 16. 10. 14.  2.  0.]
 [ 0.  0. 11. 13.  0.  8.  8.  0.]
 [ 0.  0.  2. 12. 16. 16.  7.  0.]]
8
```




```
from scipy.sparse import random
X_train,x_test,y_train,y_test = train_test_split(x, y, test_size=0.2, random_state=42)
```

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

```
model = svm.SVC(gamma = 0.001)
```

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

```
▾ SVC
SVC(gamma=0.001)
```

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