Classification Models

1. Decision Tree Classification Model
2. Random Forest Classification Model
3. Support Vector Machine classification Model
4. KNN (K-Nearest Neighbor) classification Model
5. Gaussian Naive Bayes Classification Model

Decision Tree Analysis

- will be used primarily when response is categorical data
- part of supervised learning, response is already present
- is another option besides nominal, binomial, etc.
- no use of P-Value, because there is no regression equation
- uses DecisionTreeClassifier or Random Forest Classifier
- based on some questions on data set, answer is Yes/No only, no third option
- asking questions, and rows being split if they satisfy the conditions or not
- pureNode: will contain either Yes / No, but all records will have same value, so there is no uncertainity, so we stop splitting the further

-

```
                    7
            Y               N
            5               2
        6               1
```

```
Y       N       Y       N
5       1       0       1
```

- Deviation (or Variance) = `sqrt( prob(Y) * prob(N) )` = `sqrt( p * (1-p) )`
- variance indicates uncertaininty, lesser variance, lesser the uncertainity
- for first level of Decision Tree, uncertainity / variance is high and Variance goes on decreasing as you further steps into decision tree
- Deviation: S & p: proportion
    - S = p * (1-p)
    - S = p - p^2
    - dS/dy = 1-2p = 0
    - p = 1/2

On what basis questions are asked in Decision Tree analysis

- progress is based on questions
- quesion is asked on the basis of maximum information gain that can be achieved
- python should ask the question that makes pure split, means all yes responses are put together, and all No responses are put together
- S.D. should be minimum, to ensure least uncertainity, so that information gain is high
- gini is based on S.D., gini is maximum at first level, and will go on decreasing on subsequent questions/levels

Decision Tree Classifier

```
df = pd.read_excel('CDAC_DataBook.xlsx', sheet_name='iris')
df.head()
```

	Sepal_length	Sepal_width	Petal_length	Petal_width	Species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa

```
# x_train = df.drop('Species', axis=1)
# y_train = df.Species
x_train, x_test, y_train, y_test = train_test_split(df.drop('Species', axis=1), df['Species'], test_size=0.25)
```

import DecisionTreeClassifier

```
from sklearn.tree import DecisionTreeClassifier
```

DecisionTreeClassifier().fit(x_train, y_train)

```
mod1 = DecisionTreeClassifier().fit(x_train, y_train)
```

mod1.predict(x_test)

```
y_pred = mod1.predict(x_test)
```

confusion_matrix(y_test, y_pred)

```
print(confusion_matrix(y_test, y_pred))
```

```
[[12  0  0]
 [ 0 10  0]
```

```
 [ 0  1 15]]
```

import export_graphviz

```
from sklearn.tree import export_graphviz
import pydotplus
```

export_graphviz(model, out_file='fileName.dot')

```
export_graphviz(mod1, out_file='MyFile.dot')
```

```
x_train.head()
```

	Sepal_length	Sepal_width	Petal_length	Petal_width
120	6.9	3.2	5.7	2.3
140	6.7	3.1	5.6	2.4
6	4.6	3.4	1.4	0.3
91	6.1	3.0	4.6	1.4
28	5.2	3.4	1.4	0.2

```
x_train.shape
```

```
(112, 4)
```

Random Forest Classifier

- we create multiple trees as a forest of trees, and then ask the trees to generate the responses

- final decision will be taken on decisions/responses from number of trees

```
df = pd.read_excel('CDAC_DataBook.xlsx', sheet_name='iris')
df.head()
```

	Sepal_length	Sepal_width	Petal_length	Petal_width	Species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

```
x_train, x_test, y_train, y_test = train_test_split(df.drop('Species', axis=1), df['Species'], test_size=0.25)
```

import RandomForestClassifier

```
from sklearn.ensemble import RandomForestClassifier as rfc
```

DecisionTreeClassifier().fit(x_train, y_train)

```
mod1 = DecisionTreeClassifier().fit(x_train, y_train)
# using decision tree classifier

```

model.predict(x_test)

```
y_pred = mod1.predict(x_test)
```

RandomForestClassifier(n_estimators=20).fit(x_train, y_train)

```
mod2 = rfc(n_estimators=20).fit(x_train, y_train)
# rfc(n_estimators=20, max_depth=5, min_samples_leaf=2)

# using random forest classifier
# takes up more execution time,
# but produces accurate results than decision tree
y_pred = mod2.predict(x_test)
```

confusion_matrix(y_test, y_pred)

```
print(confusion_matrix(y_test, y_pred))
```

```
[[18  0  0]
 [ 0  8  0]
 [ 0  1 11]]
```

```

```

Naive Bayes Classifier

- uses Bayes Theorem
- uses conditional probability, P(A|B) = P(A ∪ B)/P(A)
- used to find probability of event A if probability of event B is already known
- e.g. supermarket case, if user buys bread and eggs, what is probability of user buying milk
- e.g. if a person is wearing a saree is a feature, and we know the probability of person wearing saree, what will be probability of person being female ?

To-Do: Scaling of Variables

- when there is huge difference in magnitude of predictors and responses, we have to scale up/down predictors / response
- e.g. age is two digits but salary is 5 digits, so wither we scale up the age or we scale down the salary to match up with age

Data Visualization

- presenting data in visual form using charts

```
import matplotlib
from matplotlib import pyplot as plt
import pylab
from pylab import plot, show
from pylab import legend, title, xlabel, ylabel
import numpy as np
import pandas as pd
```

Line Graph

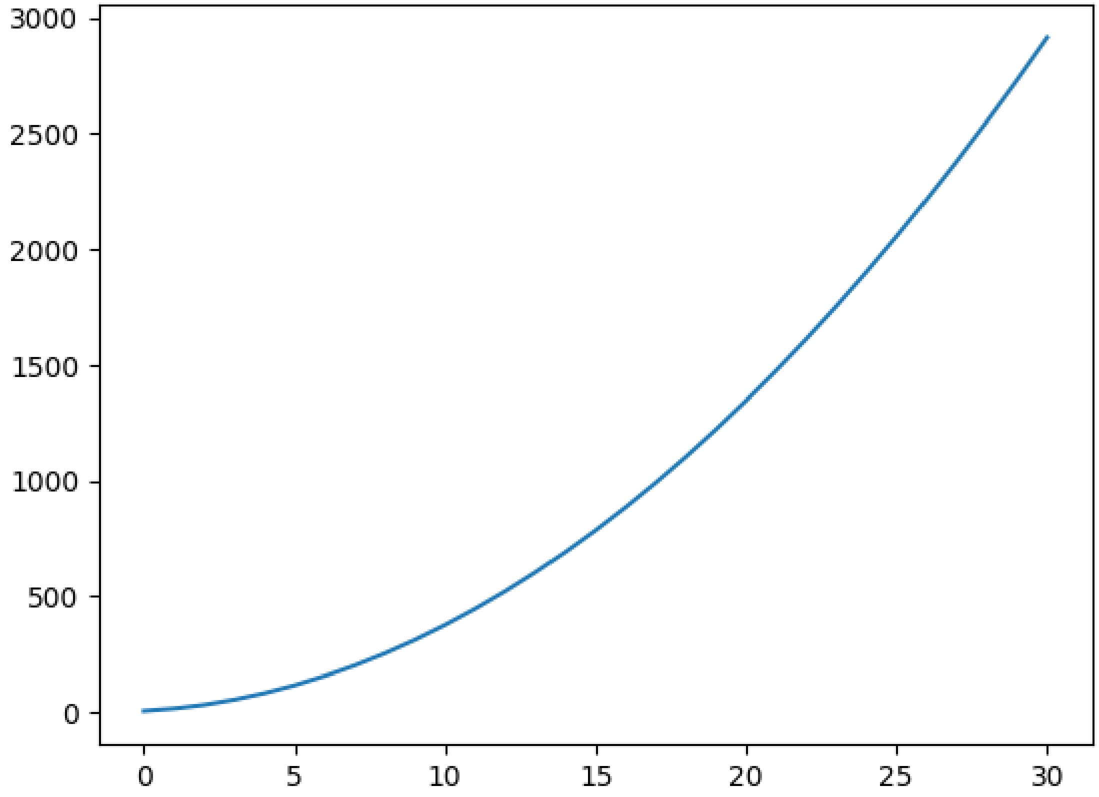
- line graph to be used to see relation between continuous or discrete data with categorical data

plot(x_pts, y_pts)

```
# y = 3x**2 + 7x + 5
x_pts = list(range(0, 31))
y_pts = []
for ctr in x_pts:
    y = 3*ctr**2 + 7*ctr + 5
```

```
    y_pts.append(y)
plot(x_pts, y_pts)
```

```
[<matplotlib.lines.Line2D at 0x7f441c9b79d0>]
```



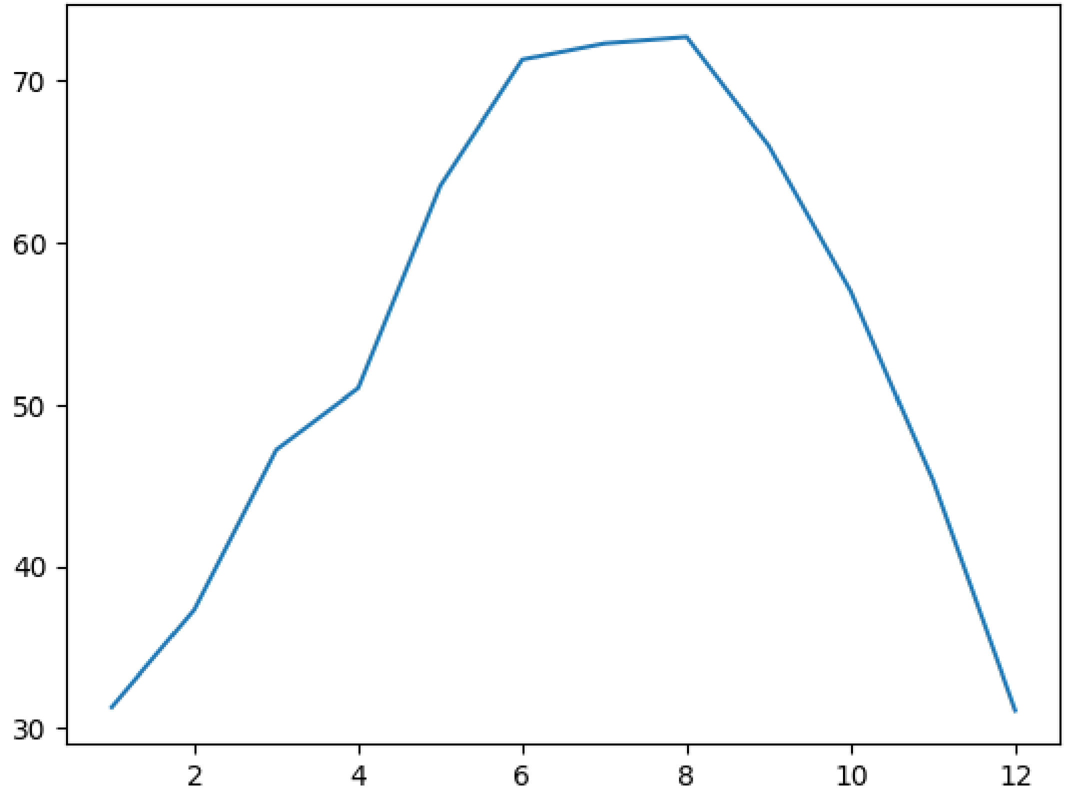

```
x1 = [31.3, 37.3, 47.2, 51.0, 63.5, 71.3, 72.3, 72.7, 66.0, 57.0, 45.3, 31.1]
len(x1)
```

```
12
```

```
months = list(range(1, 13))
```

```
plot(months, x1)
```

```
[<matplotlib.lines.Line2D at 0x7f441c6eab00>]
```

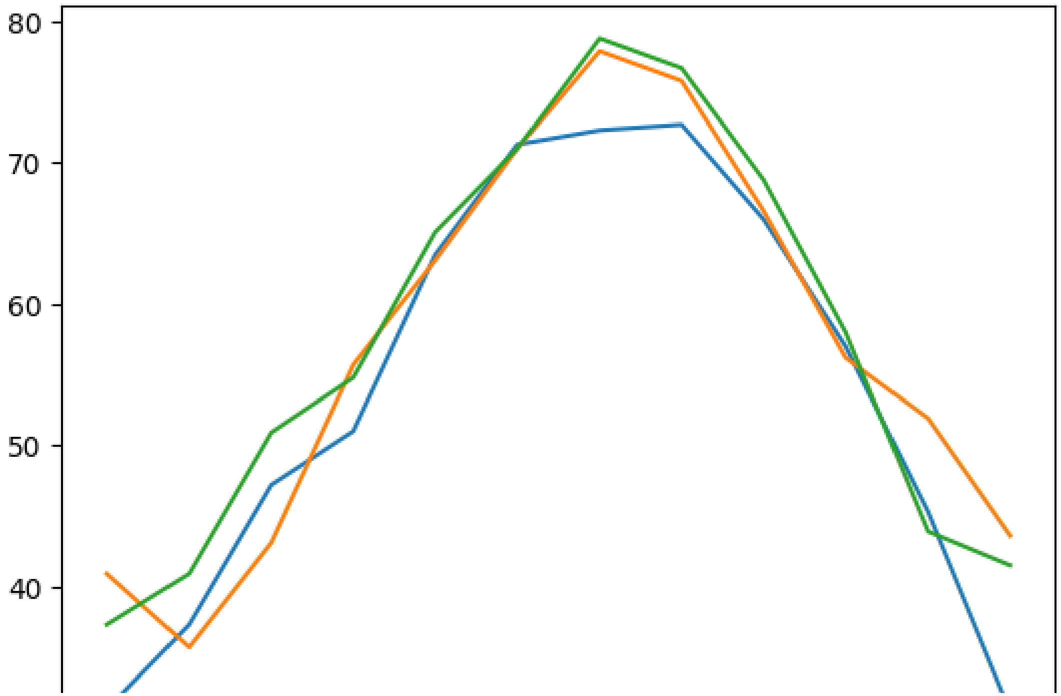



```
x1 = [31.3, 37.3, 47.2, 51.0, 63.5, 71.3, 72.3, 72.7, 66.0, 57.0, 45.3, 31.1]
x2 = [40.9, 35.7, 43.1, 55.7, 63.1, 71.0, 77.9, 75.8, 66.6, 56.2, 51.9, 43.6]
x3 = [37.3, 40.9, 50.9, 54.8, 65.1, 71.0, 78.8, 76.7, 68.8, 58.0, 43.9, 41.5]
```

plot(x, y1, x, y2, x, x3)

```
plot(months, x1, months, x2, months, x3)
```

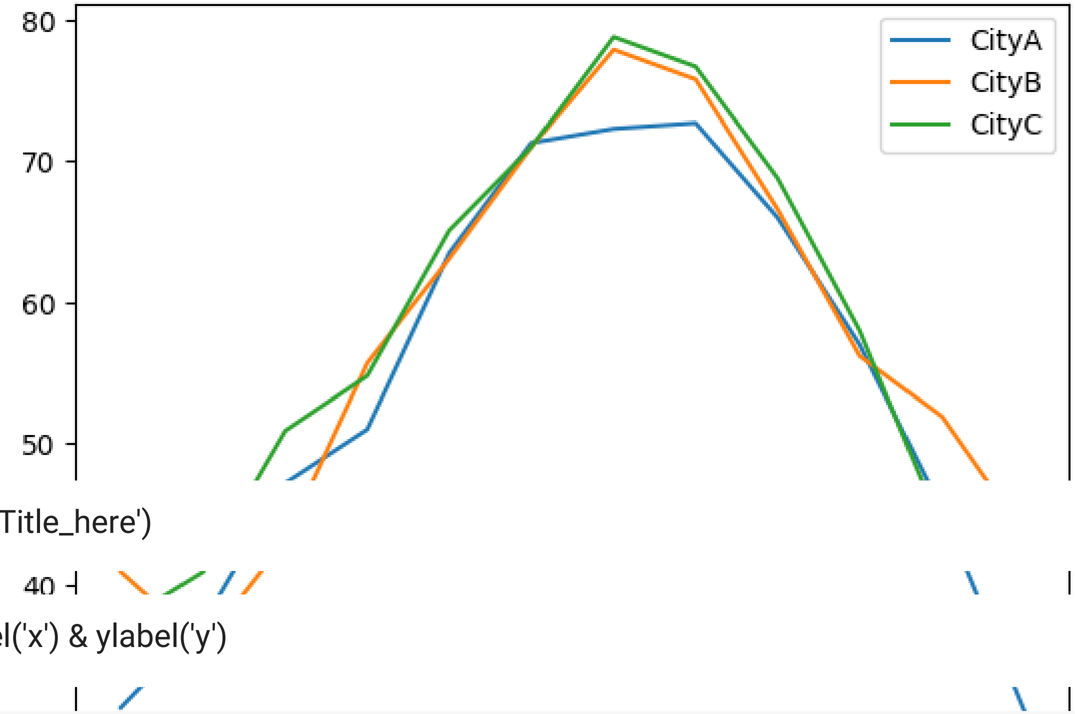
```
[<matplotlib.lines.Line2D at 0x7f441c558790>,
 <matplotlib.lines.Line2D at 0x7f441c5587f0>,
 <matplotlib.lines.Line2D at 0x7f441c558820>]
```



legend(['y1', 'y2', 'y3'])

```
1 plot(months, x1, months, x2, months, x3)
2 legend(['CityA', 'CityB', 'CityC'])
```

```
<matplotlib.legend.Legend at 0x7f4419d9bc40>
```

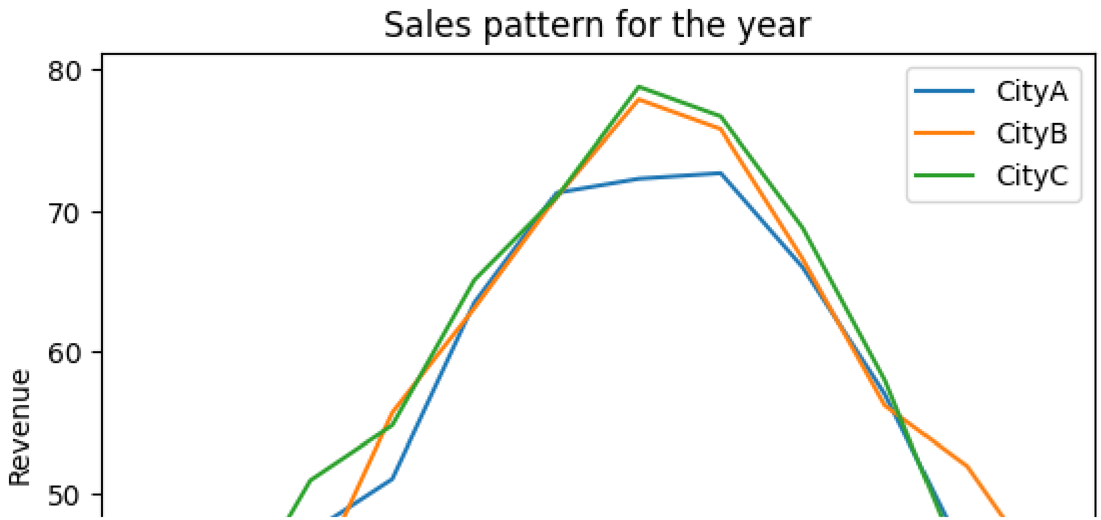


title('Title_here')

xlabel('x') & ylabel('y')

```
plot(months, x1, months, x2, months, x3)
legend(['CityA', 'CityB', 'CityC'])
title('Sales pattern for the year')
xlabel('Month')
ylabel('Revenue')
```

```
Text(0, 0.5, 'Revenue')
```



Scattered Graph

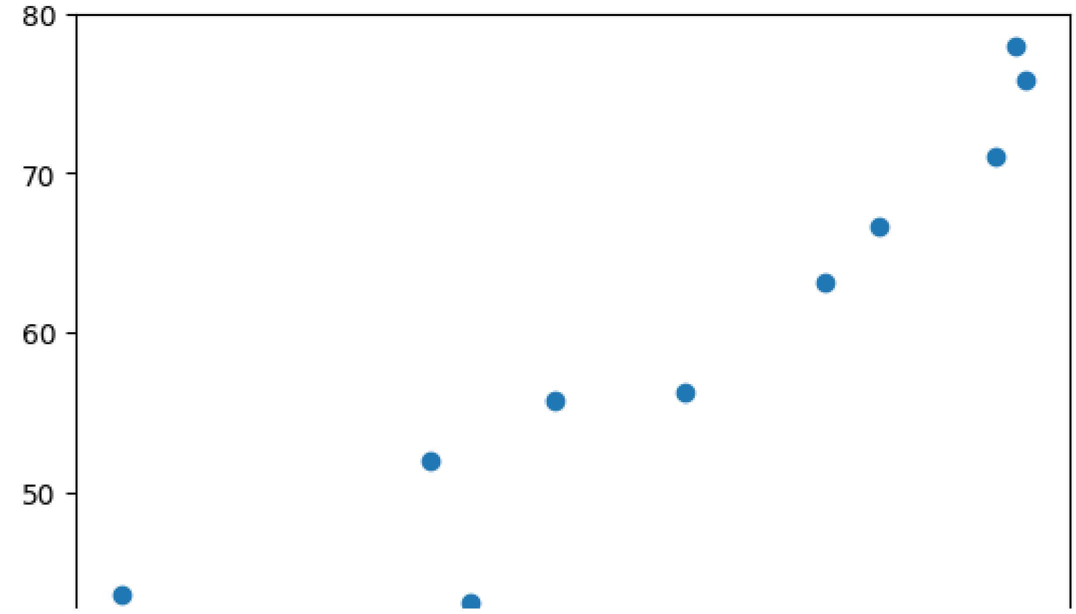
- used when both x-axis and y-axis value are continuous data
- each point represent the value of x-value & y-value

plt.scatter(x1, x2)

```
x1 = [31.3, 37.3, 47.2, 51.0, 63.5, 71.3, 72.3, 72.7, 66.0, 57.0, 45.3, 31.1]
x2 = [40.9, 35.7, 43.1, 55.7, 63.1, 71.0, 77.9, 75.8, 66.6, 56.2, 51.9, 43.6]
```

```
plt.scatter(x1, x2)
```

```
<matplotlib.collections.PathCollection at 0x7f441ca09ab0>
```



Q. plot scatter graph between wt & mpg from mtcars sheet

```
from google.colab import files
uploaded=files.upload()
# CDAC_DataBook.xlsx
# to be used with google colab

# import os
# os.chdir(r'C:\Users\surya\Downloads\PG-DBDA-Mar23\Datasets')
# os.getcwd()
# to change current working directory to specified path
# to be used while running on local system
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving CDAC_DataBook.xlsx to CDAC_DataBook.xlsx
```

```
df = pd.read_excel('CDAC_DataBook.xlsx', sheet_name='mtcars')
# reading excelfile with specifying the sheet name
df.head()
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
0	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
1	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
2	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
3	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
4	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2

```
plt.scatter(df.wt, df.mpg)
```

```
<matplotlib.collections.PathCollection at 0x7f44188d78b0>
```

Bar-Chart & Histogram

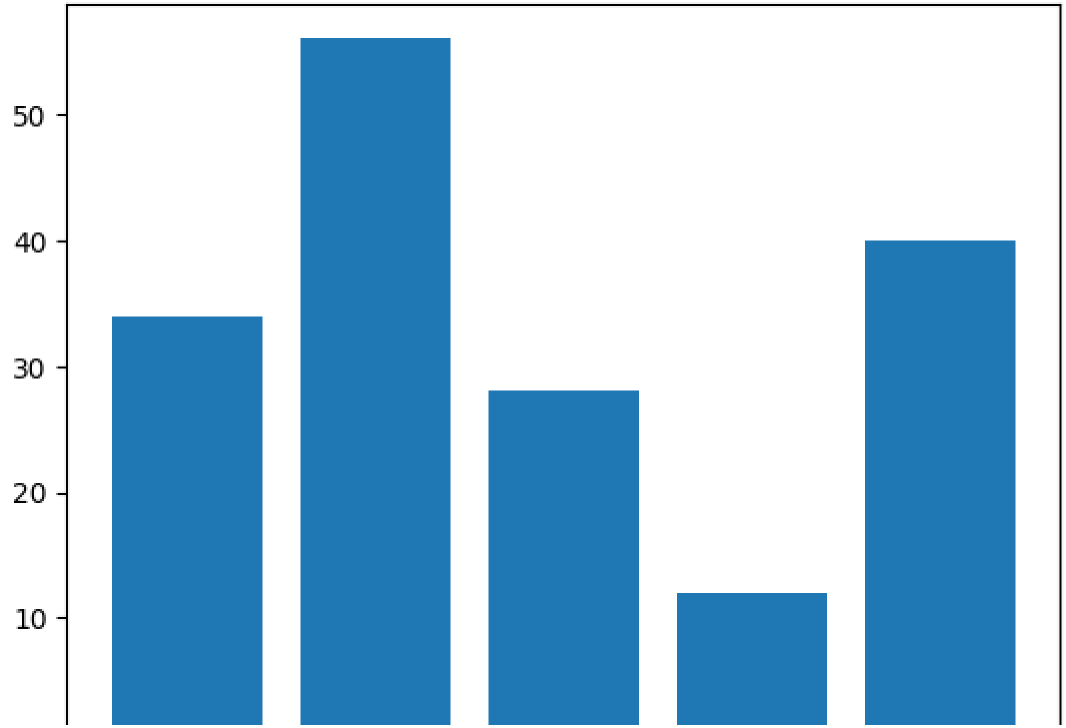
- in barchart
  - x-axis is categorical data which can have multiple values
  - since no continuous values are present on x-axis, values do not touch each other
  - y-axis will always be discrete data
  - width is always same
- in histogram
  - x-axis is continuous or interval data
  - since continuous values are present on x-axis, values touch each other at the terminal values
  - y-axis will always be discrete data
  - width will vary as per binsize

```
cts = [34, 56, 28, 12, 40]
languages = ['Marathi', 'Hindi', 'English', 'Kannada', 'Tamil']
```

plt.bar(x, y)

```
plt.bar(languages, cts)
```

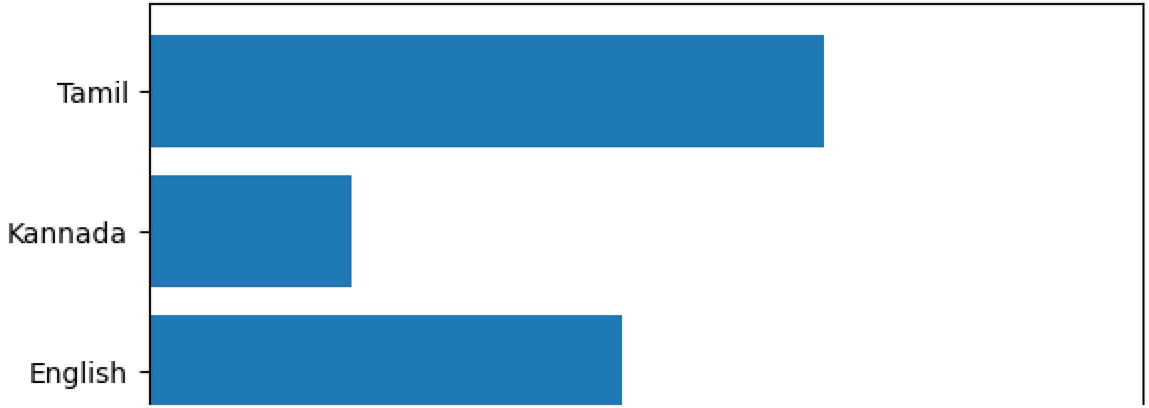
```
<BarContainer object of 5 artists>
```



plt.barh(x, y)

```
plt.barh(languages, cts)
```

```
<BarContainer object of 5 artists>
```



plt.pie(pie_share, labels=languages)


```
plt.pie(cts, labels=languages)
```

```
([<matplotlib.patches.Wedge at 0x7f44188bb1f0>,
  <matplotlib.patches.Wedge at 0x7f44188b8f10>,
  <matplotlib.patches.Wedge at 0x7f44188b9de0>,
  <matplotlib.patches.Wedge at 0x7f44188b90c0>,
  <matplotlib.patches.Wedge at 0x7f44188ba5f0>],
 [Text(0.8899186877588753, 0.6465637858537406, 'Marathi'),
  Text(-0.7259170302424207, 0.8264650417313637, 'Hindi'),
  Text(-0.8397380409695981, -0.7105209515197577, 'English'),
  Text(-0.14189900211144976, -1.0908091827628583, 'Kannada'),
  Text(0.8129099109773235, -0.7410650960845749, 'Tamil')])
```

Q. plot histogram for glucose column on

```
from google.colab import files
uploaded=files.upload()
# CDAC_DataBook.xlsx
# to be used with google colab

# import os
# os.chdir(r'C:\Users\surya\Downloads\PG-DBDA-Mar23\Datasets')
# os.getcwd()
# to change current working directory to specified path
# to be used while running on local system
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
df = pd.read_excel('CDAC_DataBook.xlsx', sheet_name='diabetes')
# reading excelfile with specifying the sheet name
df.head()
```

	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Out
0	148	72	35	0	33.6	0.63	50	
1	85	66	29	0	26.6	0.35	31	
2	183	64	0	0	23.3	0.67	32	

plt.hist(x)

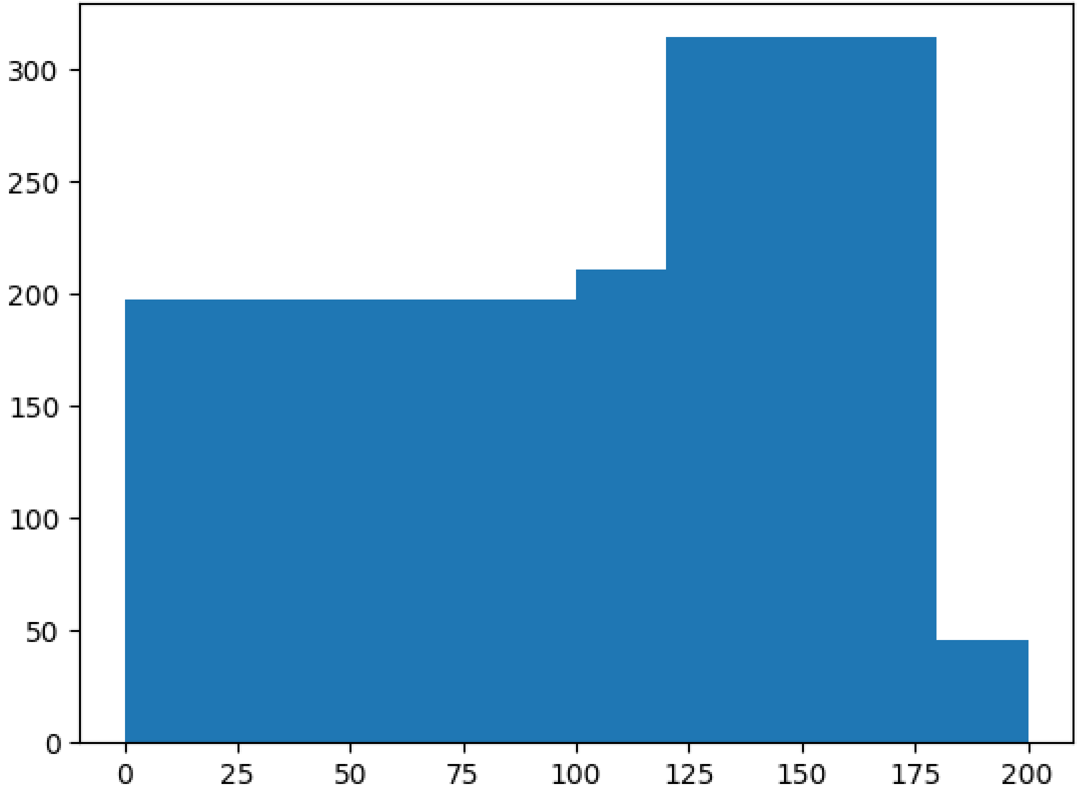
```
plt.hist(df.Glucose)
```

plt.hist(x, bins=list)

```
 <BarContainer object of 10 artists>)
```

```
1 plt.hist(df.Glucose, bins=[0, 100, 120, 180, 200])
```

```
(array([197., 211., 314.,  46.]),
 array([  0., 100., 120., 180., 200.]),
 <BarContainer object of 4 artists>)
```



Q. plot boxplot between write & math from nominal sheet

```
from google.colab import files
uploaded=files.upload()
# CDAC_DataBook.xlsx
# to be used with google colab

# import os
# os.chdir(r'C:\Users\surya\Downloads\PG-DBDA-Mar23\Datasets')
# os.getcwd()
# to change current working directory to specified path
# to be used while running on local system
```

```
df = pd.read_excel('CDAC_DataBook.xlsx', sheet_name='nominal')
# reading excelfile with specifying the sheet name
df.head()
```

	ses	write	math	prog
0	1	35	41	1
1	2	33	41	2
2	3	39	44	3
3	1	37	42	1
4	2	31	40	2

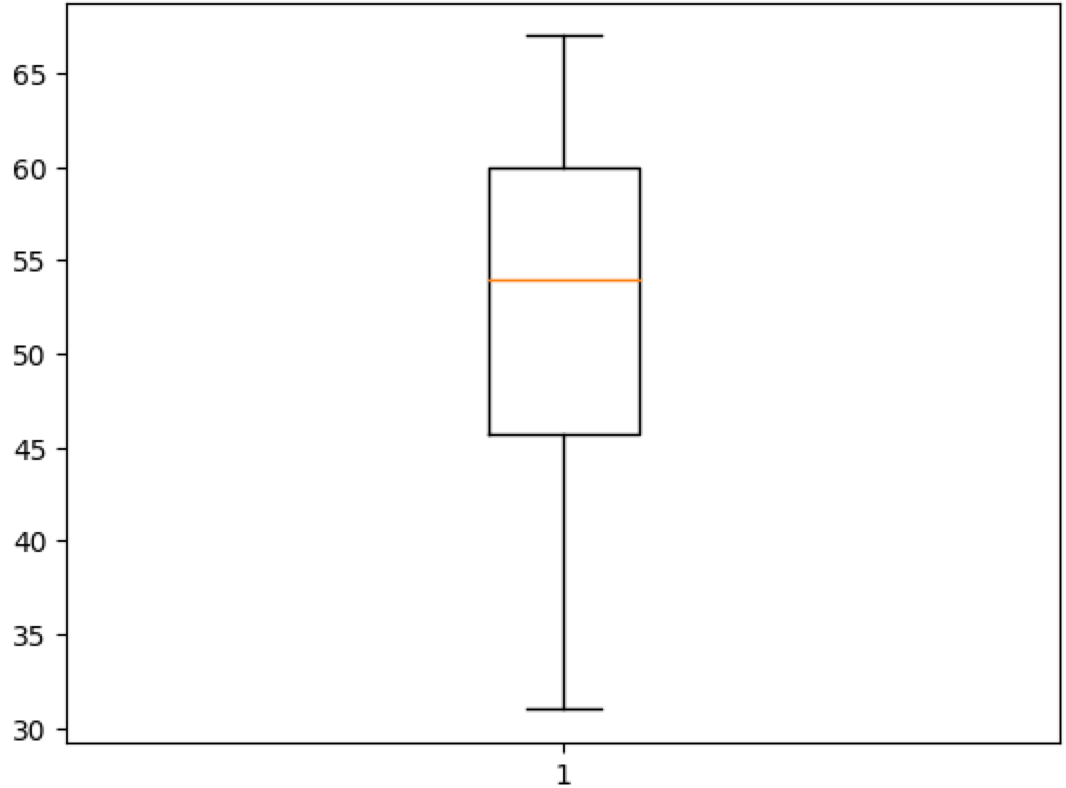
plt.boxplot(x)

```
plt.boxplot()
```

- upper Fence:
- lower Fence:
- outlier:

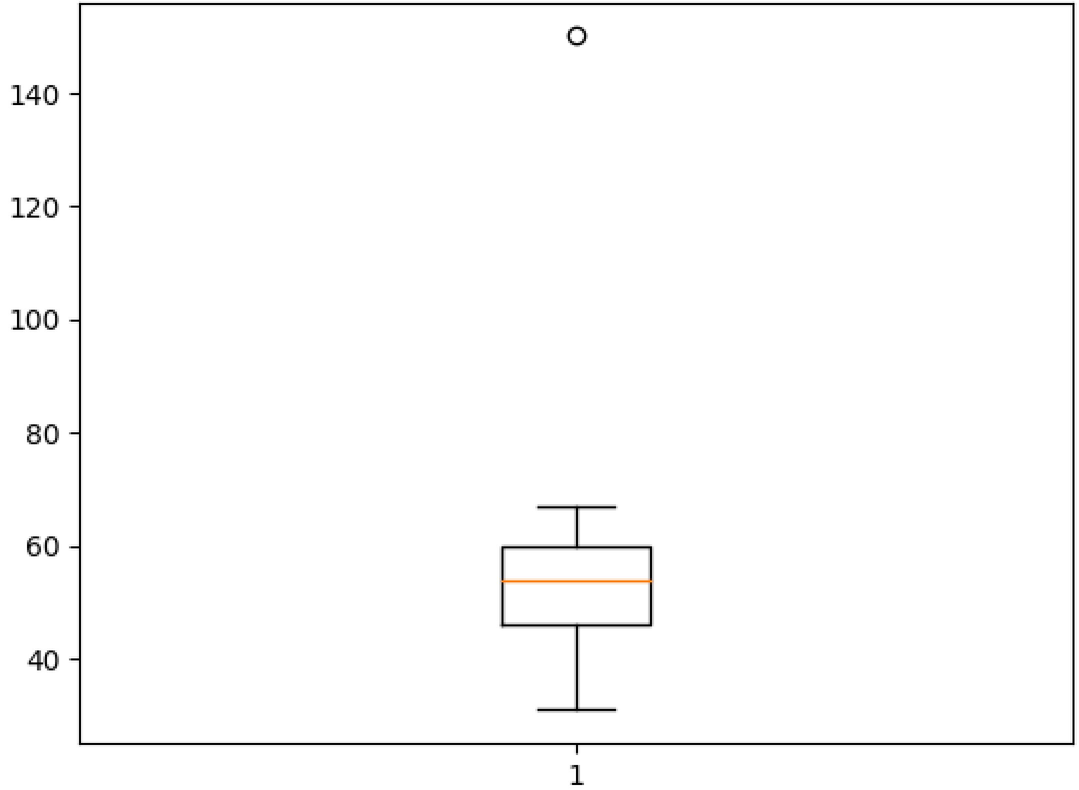
```
x1 = list(df.write)
x2 = list(df.math)
# x1.append(150)
plt.boxplot(x1)
```

```
{'whiskers': [<matplotlib.lines.Line2D at 0x7f44182d3970>,
  <matplotlib.lines.Line2D at 0x7f44182d3c10>],
 'caps': [<matplotlib.lines.Line2D at 0x7f44182d3d90>,
  <matplotlib.lines.Line2D at 0x7f4417d1c070>],
 'boxes': [<matplotlib.lines.Line2D at 0x7f44182d36d0>],
 'medians': [<matplotlib.lines.Line2D at 0x7f4417d1c310>],
 'fliers': [<matplotlib.lines.Line2D at 0x7f4417d1c5b0>],
 'means': []}
```



```
x1 = list(df.write)
x2 = list(df.math)
x1.append(150)
plt.boxplot(x1)
```

```
{'whiskers': [<matplotlib.lines.Line2D at 0x7f4417d43af0>,
  <matplotlib.lines.Line2D at 0x7f4417d43d90>],
 'caps': [<matplotlib.lines.Line2D at 0x7f4417da0070>,
  <matplotlib.lines.Line2D at 0x7f4417da0310>],
 'boxes': [<matplotlib.lines.Line2D at 0x7f4417d43970>],
 'medians': [<matplotlib.lines.Line2D at 0x7f4417da05b0>],
 'fliers': [<matplotlib.lines.Line2D at 0x7f4417da0850>],
 'means': []}
```




```
x1 = list(df.write)
x2 = list(df.math)
```

```
x1.append(150)
x = [x1, x2]
plt.boxplot(x)
```

```
{'whiskers': [<matplotlib.lines.Line2D at 0x7f4417ed4df0>,
  <matplotlib.lines.Line2D at 0x7f4417ed5090>,
  <matplotlib.lines.Line2D at 0x7f4417ed6050>,
```

Seaborn

```
  <matplotlib.lines.Line2D at 0x7f4417ed6590>,
```

```
import seaborn as sb
```

```
  <matplotlib.lines.Line2D at 0x7f4417ed5db0>],
```

```
df = pd.read_excel('CDAC_DataBook.xlsx', sheet_name='mtcars')
# reading excelfile with specifying the sheet name
df.head()
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
0	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
1	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
2	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
3	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
4	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2

```
df.columns
```

```
Index(['mpg', 'cyl', 'disp', 'hp', 'drat', 'wt', 'qsec', 'vs', 'am', 'gear',
       'carb'],
      dtype='object')
```

sb.heatmap(df.corr(), annot=True)

```
sb.heatmap(df.corr(), annot=True)
# provides correlation coefficient between different columns
```

```
<Axes: >
```