In [1]:

```
import pandas as pd
```

In [3]:

```
titanic = pd.read_csv('https://github.com/ybifoundation/Dataset/raw/main/Titanic.csv')
```

In [4]:

```
titanic.head()
```

Out[4]:

	pclass	survived	name	sex	age	sibsp	parch	ticket	fare	cabin	embarked	boat	body	home.dest
0	1	1	Allen, Miss. Elisabeth Walton	female	29.00	0	0	24160	211.3375	B5	S	2	NaN	St Louis, MO
1	1	1	Allison, Master. Hudson Trevor	male	0.92	1	2	113781	151.5500	C22 C26	S	11	NaN	Montreal, PQ / Chesterville, ON
2	1	0	Allison, Miss. Helen Loraine	female	2.00	1	2	113781	151.5500	C22 C26	S	NaN	NaN	Montreal, PQ / Chesterville, ON
3	1	0	Allison, Mr. Hudson Joshua Creighton	male	30.00	1	2	113781	151.5500	C22 C26	S	NaN	135.0	Montreal, PQ / Chesterville, ON
4	1	0	Allison, Mrs. Hudson J C (Bessie Waldo Daniels)	female	25.00	1	2	113781	151.5500	C22 C26	S	NaN	NaN	Montreal, PQ / Chesterville, ON

In [5]:

```
titanic.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1309 entries, 0 to 1308
Data columns (total 14 columns):
 #  Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0  pclass    1309 non-null   int64
 1  survived  1309 non-null   int64
 2  name      1309 non-null   object
 3  sex       1309 non-null   object
 4  age       1046 non-null   float64
 5  sibsp     1309 non-null   int64
 6  parch     1309 non-null   int64
 7  ticket    1309 non-null   object
 8  fare      1308 non-null   float64
 9  cabin     295 non-null    object
 10 embarked  1307 non-null   object
 11 boat      486 non-null    object
 12 body      121 non-null    float64
 13 home.dest 745 non-null    object
dtypes: float64(3), int64(4), object(7)
memory usage: 143.3+ KB
```

In [6]:

```
titanic.describe()
```

Out[6]:

	pclass	survived	age	sibsp	parch	fare	body
count	1309.000000	1309.000000	1046.000000	1309.000000	1309.000000	1308.000000	121.000000
mean	2.294882	0.381971	29.881138	0.498854	0.385027	33.295479	160.809917
std	0.837836	0.486055	14.413493	1.041658	0.865560	51.758668	97.696922
min	1.000000	0.000000	0.170000	0.000000	0.000000	0.000000	1.000000
25%	2.000000	0.000000	21.000000	0.000000	0.000000	7.895800	72.000000
50%	3.000000	0.000000	28.000000	0.000000	0.000000	14.454200	155.000000
75%	3.000000	1.000000	39.000000	1.000000	0.000000	31.275000	256.000000
max	3.000000	1.000000	80.000000	8.000000	9.000000	512.329200	328.000000

In [7]:

```
titanic.columns
```

Out[7]:

```
Index(['pclass', 'survived', 'name', 'sex', 'age', 'sibsp', 'parch', 'ticket',
       'fare', 'cabin', 'embarked', 'boat', 'body', 'home.dest'],
      dtype='object')
```

In [8]:

```
titanic.shape
```

Out[8]:

```
(1309, 14)
```

In [9]:

```
titanic.corr()
```

Out[9]:

	pclass	survived	age	sibsp	parch	fare	body
pclass	1.000000	-0.312469	-0.408106	0.060832	0.018322	-0.558629	-0.034642
survived	-0.312469	1.000000	-0.055512	-0.027825	0.082660	0.244265	NaN
age	-0.408106	-0.055512	1.000000	-0.243699	-0.150917	0.178740	0.058809
sibsp	0.060832	-0.027825	-0.243699	1.000000	0.373587	0.160238	-0.099961
parch	0.018322	0.082660	-0.150917	0.373587	1.000000	0.221539	0.051099
fare	-0.558629	0.244265	0.178740	0.160238	0.221539	1.000000	-0.043110
body	-0.034642	NaN	0.058809	-0.099961	0.051099	-0.043110	1.000000

In [11]:

```
titanic.nunique()
```

Out[11]:

```
pclass       3
survived     2
name      1307
sex          2
age         98
sibsp        7
parch        8
ticket     929
fare       281
cabin      186
embarked     3
boat        27
body       121
home.dest  369
dtype: int64
```

In [12]:

```
titanic.columns
```

Out[12]:

```
Index(['pclass', 'survived', 'name', 'sex', 'age', 'sibsp', 'parch', 'ticket',
       'fare', 'cabin', 'embarked', 'boat', 'body', 'home.dest'],
      dtype='object')
```

In [13]:

```
y = titanic['survived']
```

In [14]:

```
X = titanic[['pclass', 'name', 'sex', 'age', 'sibsp', 'parch', 'ticket',
       'fare', 'cabin', 'embarked', 'boat', 'body', 'home.dest']]
```

In [15]:

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

In [16]:

```
X_train, X_test, y_train, y_test = train_test_split(X,y, train_size=0.7, random_state=2529)
```

In [17]:

```
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

Out[17]:

```
((916, 13), (393, 13), (916,), (393,))
```

In [18]:

```
X_train
```

Out[18]:

	pclass	name	sex	age	sibsp	parch	ticket	fare	cabin	embarked	boat	body	home.dest
645	3	Asplund, Mr. Johan Charles	male	23.0	0	0	350054	7.7958	NaN	S	13	NaN	Oskarshamn, Sweden Minneapolis, MN
482	2	Lehmann, Miss. Bertha	female	17.0	0	0	SC 1748	12.0000	NaN	C	12	NaN	Berne, Switzerland / Central City, IA
521	2	Nye, Mrs. (Elizabeth Ramell)	female	29.0	0	0	C.A. 29395	10.5000	F33	S	11	NaN	Folkstone, Kent / New York, NY
1141	3	Rice, Master. Albert	male	10.0	4	1	382652	29.1250	NaN	Q	NaN	NaN	NaN
88	1	Daniels, Miss. Sarah	female	33.0	0	0	113781	151.5500	NaN	S	8	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...
380	2	Cook, Mrs. (Selena Rogers)	female	22.0	0	0	W./C. 14266	10.5000	F33	S	14	NaN	Pennsylvania
943	3	Laitinen, Miss. Kristina Sofia	female	37.0	0	0	4135	9.5875	NaN	S	NaN	NaN	NaN
740	3	Culumovic, Mr. Jeso	male	17.0	0	0	315090	8.6625	NaN	S	NaN	NaN	Austria-Hungary
399	2	Drew, Mr. James Vivian	male	42.0	1	1	28220	32.5000	NaN	S	NaN	NaN	Greenport, NY
828	3	Goodwin, Miss. Jessie Allis	female	10.0	5	2	CA 2144	46.9000	NaN	S	NaN	NaN	Wiltshire, England Niagara Falls, NY

916 rows × 13 columns

In [19]:

```
import pandas as pd
```

In [20]:

```
headphone = pd.read_csv('https://github.com/ybifoundation/Dataset/raw/main/Flipkart%20Headphones.csv')
```

In [21]:

```
headphone.head()
```

Out[21]:

	Model	Company	Color	Type	Average Rating	Number of Ratings	Selling Price	Maximum Retail Price	Discount
0	5PLUS 5PHP28 Wired without Mic Headset	5PLUS	Red	On the Ear	3.6	101	496	3399	2903
1	A R Wireless compatible with Headset Bluetooth...	A R	Red	Multicolor	3.9	35280	188	799	611
2	Aerizo Wireless Touch R100 Earbuds (Black) Blu...	Aerizo	Black	True Wireless	4.0	1934	589	1298	709
3	Allmusic powerful driven bass with dynamic bea...	Allmusic	Multicolor	In the Ear	4.0	15841	260	1599	1339
4	Allmusic OPP.O Ultra HD Sound Premium Bass Spo...	Allmusic	Black	In the Ear	3.8	10766	270	999	729

In [22]:

```
headphone.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 9 columns):
 #  Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0  Model             1000 non-null   object 
 1  Company           1000 non-null   object 
 2  Color             1000 non-null   object 
 3  Type              1000 non-null   object 
 4  Average Rating    1000 non-null   float64
 5  Number of Ratings 1000 non-null   int64  
 6  Selling Price     1000 non-null   int64  
 7  Maximum Retail Price  1000 non-null   int64  
 8  Discount          1000 non-null   int64  
dtypes: float64(1), int64(4), object(4)
memory usage: 70.4+ KB
```

In [23]:

```
headphone.describe()
```

Out[23]:

	Average Rating	Number of Ratings	Selling Price	Maximum Retail Price	Discount
count	1000.000000	1.000000e+03	1000.000000	1000.000000	1000.000000
mean	3.831000	5.004075e+04	832.875000	2423.043000	1590.168000
std	0.467459	1.572297e+05	812.535141	1774.025318	1341.379254
min	1.000000	0.000000e+00	88.000000	0.000000	-2991.000000
25%	3.600000	1.167500e+02	349.000000	1079.750000	697.500000
50%	3.900000	1.712000e+03	599.000000	1999.000000	1390.500000
75%	4.000000	1.332700e+04	999.000000	2999.000000	2250.000000
max	5.000000	1.299042e+06	7990.000000	16999.000000	12000.000000

In [24]:

```
headphone.columns
```

Out[24]:

```
Index(['Model', 'Company', 'Color', 'Type', 'Average Rating',
       'Number of Ratings', 'Selling Price', 'Maximum Retail Price',
       'Discount'],
      dtype='object')
```

In [25]:

```
headphone.shape
```

Out[25]:

```
(1000, 9)
```

In [26]:

```
headphone.corr()
```

Out[26]:

	Average Rating	Number of Ratings	Selling Price	Maximum Retail Price	Discount
Average Rating	1.000000	0.203486	0.013916	-0.025261	-0.041838
Number of Ratings	0.203486	1.000000	-0.038969	-0.033778	-0.021067
Selling Price	0.013916	-0.038969	1.000000	0.696545	0.315461
Maximum Retail Price	-0.025261	-0.033778	0.696545	1.000000	0.900609
Discount	-0.041838	-0.021067	0.315461	0.900609	1.000000

In [27]:

```
headphone.nunique()
```

Out[27]:

```
Model                591
Company              202
Color                 94
Type                  16
Average Rating        21
Number of Ratings    434
Selling Price        241
Maximum Retail Price 106
Discount             416
dtype: int64
```

In [28]:

```
headphone.columns
```

Out[28]:

```
Index(['Model', 'Company', 'Color', 'Type', 'Average Rating',
       'Number of Ratings', 'Selling Price', 'Maximum Retail Price',
       'Discount'],
      dtype='object')
```

In [29]:

```
y = headphone['Selling Price']
```

In [30]:

```
X = headphone[['Model', 'Company', 'Color', 'Type', 'Average Rating',
    'Number of Ratings', 'Maximum Retail Price',
    'Discount']]
```

In [31]:

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

In [32]:

```
X_train, X_test, y_train, y_test = train_test_split(X,y, train_size=0.7, random_state=2529)
```

In [33]:

```
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

Out[33]:

```
((700, 8), (300, 8), (700,), (300,))
```

In [34]:

```
X_train
```

Out[34]:

	Model	Company	Color	Type	Average Rating	Number of Ratings	Maximum Retail Price	Discount
669	POKRYT Hot SaleASAP charging Version 5.0 Bluet...	POKRYT	Black	In the Ear	4.0	333036	1999	1100
583	N2B MAGNET-02 PACK OF 2 Bluetooth Headset	N2B	Black	True Wireless	3.5	35	1998	1744
688	PunnkFunnk K20 Gaming Headset, Over Ear Gaming...	PunnkFunnk	Black	On the Ear	3.5	1713	2499	1550
422	Fire-Boltt BN1000 Bluetooth Headset	FIER	Black Red	In the Ear	4.0	1934	2499	1200
825	Titan N2 Bluetooth Headset	Titan	Black	In the Ear	3.6	1686	1599	1270
...	...	...	...	...	...	...	...	...
740	SE WORLD P47 WIRELESS HEADPHONE Bluetooth, Wir...	SE	Black	On the Ear	3.8	42	1999	1600
399	FIER MAGNET-K1-I7 Pack of 3 Bluetooth Bluetoot...	FIER	Black	White	3.6	980	1999	1630
828	Togkart TWS i12 Bluetooth Headset	Togkart	Black	True Wireless	3.9	0	1499	1044
562	Mivi DuoPods M20 True Wireless Bluetooth Headset	Mivi	Blue	True Wireless	3.8	1748	2999	2200
352	DigiClues Bass-Dops Bluetooth Headset	DigiClues	Black	True Wireless	3.3	85	2999	2000

700 rows × 8 columns

In [35]:

```
#pridict chance of admission

```

In [36]:

```
import pandas as pd
```

In [37]:

```
admission =pd.read_csv('https://github.com/ybifoundation/Dataset/raw/main/Admission%20Chance.csv')
```

In [38]:

```
admission.columns
```

Out[38]:

```
Index(['Serial No', 'GRE Score', 'TOEFL Score', 'University Rating', ' SOP',
       'LOR ', 'CGPA', 'Research', 'Chance of Admit '],
      dtype='object')
```

In [39]:

```
y = admission['Chance of Admit ']
```

In [40]:

```
X = admission[['GRE Score', 'TOEFL Score', 'University Rating', ' SOP',
    'LOR ', 'CGPA', 'Research']]
```

In [41]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y, train_size=0.7, random_state=2529)
```

In [42]:

```
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

Out[42]:

```
((280, 7), (120, 7), (280,), (120,))
```

In [43]:

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

In [44]:

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

Out[44]:

```
LinearRegression()
```

In [45]:

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

In [46]:

```
from sklearn.metrics import mean_absolute_error, mean_absolute_percentage_error, mean_squared_error
```

In [47]:

```
mean_absolute_error(y_test,y_pred)
```

Out[47]:

```
0.04400128934232654
```

In [48]:

```
mean_absolute_percentage_error(y_test,y_pred)
```

Out[48]:

```
0.07575278864605443
```

In [49]:

```
mean_squared_error(y_test,y_pred)
```

Out[49]:

```
0.004038263715495692 5
```

In [50]:

```
#house price prediction
```

In [51]:

```
import pandas as pd
```

In [52]:

```
house =pd.read_csv('https://github.com/ybifoundation/Dataset/raw/main/Boston.csv')
```

In [53]:

```
house.columns
```

Out[53]:

```
Index(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX',
       'PTRATIO', 'B', 'LSTAT', 'MEDV'],
      dtype='object')
```

In [54]:

```
y = house['MEDV']
```

In [55]:

```
X = house[['CRIM', 'ZN', 'INDUS', 'CHAS', 'NX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX',
       'PTRATIO', 'B', 'LSTAT']]
```

In [56]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y, train_size=0.7, random_state=2529)
```

In [57]:

```
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

Out[57]:

```
((354, 13), (152, 13), (354,), (152,))
```

In [58]:

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

In [59]:

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

Out[59]:

```
LinearRegression()
```

In [60]:

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

In [61]:

```
from sklearn.metrics import mean_absolute_error, mean_absolute_percentage_error, mean_squared_error
```

In [62]:

```
mean_absolute_error(y_test,y_pred)
```

Out[62]:

```
3.1550309276024904
```

In [63]:

```
mean_absolute_percentage_error(y_test,y_pred)
```

Out[63]:

```
0.1635593588221794
```

In [64]:

```
mean_squared_error(y_test,y_pred)
```

Out[64]:

```
20.718012877838454
```

In [65]:

```
#fish species prediction
```

In [66]:

```
import pandas as pd
```

In [67]:

```
fish =pd.read_csv('https://github.com/ybifoundation/Dataset/raw/main/Fish.csv')
```

In [68]:

```
fish.columns
```

Out[68]:

```
Index(['Category', 'Species', 'Weight', 'Height', 'Width', 'Length1',
       'Length2', 'Length3'],
      dtype='object')
```

In [69]:

```
y = fish['Category']
```

In [70]:

```
X = fish[['Weight', 'Height', 'Width', 'Length1',
       'Length2', 'Length3']]
```

In [71]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y, train_size=0.7, random_state=2529)
```

In [72]:

```
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

Out[72]:

```
((111, 6), (48, 6), (111,), (48,))
```

In [73]:

```
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier()
```

In [74]:

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

Out[74]:

```
KNeighborsClassifier()
```

In [75]:

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

In [76]:

```
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

In [77]:

```
accuracy_score(y_test,y_pred)
```

Out[77]:

```
0.5
```

In [78]:

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

Out[78]:

```
array([[ 7,  1,  0,  3,  0,  0,  0],
       [ 0,  0,  1,  0,  0,  0,  0],
       [ 6,  0, 10,  1,  0,  0,  0],
       [ 3,  2,  0,  0,  0,  0,  0],
       [ 1,  1,  4,  0,  1,  0,  0],
       [ 0,  0,  0,  0,  0,  6,  0],
       [ 0,  1,  0,  0,  0,  0,  0]], dtype=int64)
```

In [79]:

```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           1       0.41      0.64      0.50        11
           2       0.00      0.00      0.00         1
           3       0.67      0.59      0.62        17
           4       0.00      0.00      0.00         5
           5       1.00      0.14      0.25         7
           6       1.00      1.00      1.00         6
           7       0.00      0.00      0.00         1

    accuracy                           0.50        48
   macro avg       0.44      0.34      0.34        48
weighted avg       0.60      0.50      0.50        48

C:\Users\Suraj\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1318: UndefinedMetricWarning: Precision and F-score are ill-defined a
nd being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Suraj\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1318: UndefinedMetricWarning: Precision and F-score are ill-defined a
nd being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Suraj\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1318: UndefinedMetricWarning: Precision and F-score are ill-defined a
nd being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [81]:

```
#iris flower prediction
```

In [82]:

```
import pandas as pd
```

In [83]:

```
iris =pd.read_csv('https://github.com/ybifoundation/Dataset/raw/main/IRIS.csv')
```

In [84]:

```
iris.columns
```

Out[84]:

```
Index(['sepal_length', 'sepal_width', 'petal_length', 'petal_width',
       'species'],
      dtype='object')
```

In [85]:

```
y = iris['species']
```

In [86]:

```
X = iris[['sepal_length', 'sepal_width', 'petal_length', 'petal_width']]
```

In [87]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y, train_size=0.7, random_state=2529)
```

In [88]:

```
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

Out[88]:

```
((105, 4), (45, 4), (105,), (45,))
```

In [89]:

```
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier()
```

In [90]:

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

Out[90]:

```
DecisionTreeClassifier()
```

In [91]:

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

In [92]:

```
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

In [93]:

```
accuracy_score(y_test,y_pred)
```

Out[93]:

```
0.9777777777777777
```

In [94]:

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

Out[94]:

```
array([[14,  0,  0],
       [ 0,  9,  0],
       [ 0,  1, 21]], dtype=int64)
```

In [95]:

```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

 Iris-setosa       1.00      1.00      1.00        14
Iris-versicolor       0.90      1.00      0.95         9
Iris-virginica       1.00      0.95      0.98        22

    accuracy                           0.98        45
   macro avg       0.97      0.98      0.97        45
weighted avg       0.98      0.98      0.98        45
```

In [ ]:

```

```