```
csv_url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data'
```

```
import pandas as pd
```

```
iris = pd.read_csv(csv_url, header = None)
```

```
col_names =['Sepal_Length','Sepal_Width','Petal_Length','Petal_Width','Species']
```

```
iris = pd.read_csv(csv_url, names = col_names)
```

```
df1=df=iris
```

```
iris.head(8)
```

	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
5	5.4	3.9	1.7	0.4	Iris-setosa
6	4.6	3.4	1.4	0.3	Iris-setosa
7	5.0	3.4	1.5	0.2	Iris-setosa

```
iris.tail()
```

	Sepal_Length	Sepal_Width	Petal_Length	Petal_Width	Species
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

```
iris.index
```

```
RangeIndex(start=0, stop=150, step=1)
```

```
iris.columns
```

```
Index(['Sepal_Length', 'Sepal_Width', 'Petal_Length', 'Petal_Width',
       'Species'],
      dtype='object')
```

```
iris.shape
```

```
(150, 5)
```

```
iris.dtypes
```

```
Sepal_Length    float64
Sepal_Width     float64
Petal_Length    float64
Petal_Width     float64
Species          object
dtype: object
```

```
iris.describe()
```

	Sepal_Length	Sepal_Width	Petal_Length	Petal_Width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

```
iris.columns.values
```

```
array(['Sepal_Length', 'Sepal_Width', 'Petal_Length', 'Petal_Width',
       'Species'], dtype=object)
```

```
iris.iloc[5]
```

```
Sepal_Length            5.4
Sepal_Width             3.9
Petal_Length            1.7
Petal_Width             0.4
Species         Iris-setosa
Name: 5, dtype: object
```

```
iris[47:51]
```

	Sepal_Length	Sepal_Width	Petal_Length	Petal_Width	Species
47	4.6	3.2	1.4	0.2	Iris-setosa
48	5.3	3.7	1.5	0.2	Iris-setosa
49	5.0	3.3	1.4	0.2	Iris-setosa
50	7.0	3.2	4.7	1.4	Iris-versicolor

```
iris.loc[:,["Sepal_Length","Sepal_Width"]]
```

	Sepal_Length	Sepal_Width
0	5.1	3.5
1	4.9	3.0
2	4.7	3.2
3	4.6	3.1
4	5.0	3.6
...	...	...
145	6.7	3.0
146	6.3	2.5
147	6.5	3.0
148	6.2	3.4
149	5.9	3.0

150 rows × 2 columns

```
cols_2_4=iris.columns[2:4]
iris[cols_2_4]
```

	Petal_Length	Petal_Width
0	1.4	0.2
1	1.4	0.2
2	1.3	0.2
3	1.5	0.2
4	1.4	0.2
...	...	...
145	5.2	2.3
146	5.0	1.9
147	5.2	2.0
148	5.4	2.3
149	5.1	1.8

150 rows × 2 columns

```
iris.isnull().any()
```

```
Sepal_Length    False
Sepal_Width     False
Petal_Length    False
Petal_Width     False
Species         False
dtype: bool
```

```
iris.isnull().sum()
```

```
Sepal_Length    0
Sepal_Width     0
Petal_Length    0
Petal_Width     0
Species         0
dtype: int64
```

```
iris.dtypes
```

```
Sepal_Length    float64
Sepal_Width     float64
Petal_Length    float64
Petal_Width     float64
Species          object
dtype: object
```

```
df=iris
df['petal Length(cm)']=iris['Petal_Length'].astype("int")
```

```
df1=df
```

```
df
```

	Sepal_Length	Sepal_Width	Petal_Length	Petal_Width	Species	petal Length(cm)
0	5.1	3.5	1.4	0.2	Iris-setosa	1
1	4.9	3.0	1.4	0.2	Iris-setosa	1
2	4.7	3.2	1.3	0.2	Iris-setosa	1
3	4.6	3.1	1.5	0.2	Iris-setosa	1
4	5.0	3.6	1.4	0.2	Iris-setosa	1
...	...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Iris-virginica	5
146	6.3	2.5	5.0	1.9	Iris-virginica	5
147	6.5	3.0	5.2	2.0	Iris-virginica	5
148	6.2	3.4	5.4	2.3	Iris-virginica	5
149	5.9	3.0	5.1	1.8	Iris-virginica	5

150 rows × 6 columns

```
from sklearn import preprocessing
min_max_scaler = preprocessing.MinMaxScaler()
```

```
X=iris.iloc[:,:4]
```

```
X
```

	Sepal_Length	Sepal_Width	Petal_Length	Petal_Width
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

150 rows × 4 columns

```
X_scaled = min_max_scaler.fit_transform(X)
```

```
df_normalized = pd.DataFrame(X_scaled)
```

```
df_normalized
```

	0	1	2	3
0	0.222222	0.625000	0.067797	0.041667
1	0.166667	0.416667	0.067797	0.041667
2	0.111111	0.500000	0.050847	0.041667
3	0.083333	0.458333	0.084746	0.041667
4	0.194444	0.666667	0.067797	0.041667
...	...	...	...	...
145	0.666667	0.416667	0.711864	0.916667
146	0.555556	0.208333	0.677966	0.750000
147	0.611111	0.416667	0.711864	0.791667
148	0.527778	0.583333	0.745763	0.916667
149	0.444444	0.416667	0.694915	0.708333

150 rows × 4 columns

```
df2=df
df2['Species'].unique()
```

```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)
```

```
df_normalized = pd.DataFrame(X_scaled)
```

```
df_normalized
```

	0	1	2	3
0	0.222222	0.625000	0.067797	0.041667
1	0.166667	0.416667	0.067797	0.041667
2	0.111111	0.500000	0.050847	0.041667
3	0.083333	0.458333	0.084746	0.041667
4	0.194444	0.666667	0.067797	0.041667
...	...	...	...	...
145	0.666667	0.416667	0.711864	0.916667
146	0.555556	0.208333	0.677966	0.750000
147	0.611111	0.416667	0.711864	0.791667
148	0.527778	0.583333	0.745763	0.916667
149	0.444444	0.416667	0.694915	0.708333

150 rows × 4 columns

```
df2=df
df2['Species'].unique()
```

```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)
```

```
from sklearn import preprocessing
enc = preprocessing.OneHotEncoder()
```

```
features_df=df2.drop(columns=['Species'])
```

```
features_df
```

	Sepal_Length	Sepal_Width	Petal_Length	Petal_Width	petal Length(cm)
0	5.1	3.5	1.4	0.2	1
1	4.9	3.0	1.4	0.2	1
2	4.7	3.2	1.3	0.2	1
3	4.6	3.1	1.5	0.2	1
4	5.0	3.6	1.4	0.2	1
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	5
146	6.3	2.5	5.0	1.9	5
147	6.5	3.0	5.2	2.0	5
148	6.2	3.4	5.4	2.3	5
149	5.9	3.0	5.1	1.8	5

150 rows × 5 columns

```
enc_df=(enc.fit_transform(df2[['Species']])).toarray()
```

```
enc_df = pd.DataFrame(enc_df, columns = ['Iris-Setosa','Iris-Versicolor','Iris-Virginica'])
```

```
df_encode = features_df.join(enc_df)
```

```
df_encode
```

	Sepal_Length	Sepal_Width	Petal_Length	Petal_Width	petal Length(cm)	Iris-Setosa	Iris-Versicolor	Iris-Virginica
0	5.1	3.5	1.4	0.2	1	1.0	0.0	0.0
1	4.9	3.0	1.4	0.2	1	1.0	0.0	0.0
2	4.7	3.2	1.3	0.2	1	1.0	0.0	0.0
3	4.6	3.1	1.5	0.2	1	1.0	0.0	0.0
4	5.0	3.6	1.4	0.2	1	1.0	0.0	0.0
...	...	...	...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	5	0.0	0.0	1.0
146	6.3	2.5	5.0	1.9	5	0.0	0.0	1.0
147	6.5	3.0	5.2	2.0	5	0.0	0.0	1.0
148	6.2	3.4	5.4	2.3	5	0.0	0.0	1.0

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
from scipy import stats
```

```
df=pd.read_csv("D:\StudentPerformance.csv")
```

```
df
```

	gender	math score	reading score	writing score	placement score	club join year	placement offer
0	female	63.0	84.0	64	84.0	2020	2
1	female	71.0	80.0	76	86.0	2018	3
2	female	64.0	81.0	66	81.0	2020	2
3	male	71.0	85.0	77	96.0	2018	1
4	male	68.0	86.0	76	NaN	2021	3
5	female	94.0	86.0	61	100.0	2019	1
6	male	75.0	79.0	66	-99.0	2020	1
7	female	NaN	NaN	66	95.0	2019	3
8	male	66.0	88.0	66	88.0	2020	3
9	male	70.0	79.0	61	87.0	2021	2
10	female	-99.0	80.0	65	85.0	2021	1
11	male	76.0	84.0	-99	NaN	2020	2
12	female	74.0	79.0	79	98.0	2019	2

```
df.isnull()
```

	gender	math score	reading score	writing score	placement score	club join year	placement offer
0	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False
4	False	False	False	False	True	False	False
5	False	False	False	False	False	False	False
6	False	False	False	False	False	False	False
7	False	True	True	False	False	False	False
8	False	False	False	False	False	False	False
9	False	False	False	False	False	False	False
10	False	False	False	False	False	False	False
11	False	False	False	False	True	False	False
12	False	False	False	False	False	False	False

```
series = pd.isnull(df["reading score"])
df[series]
```

	gender	math score	reading score	writing score	placement score	club join year	placement offer
7	female	NaN	NaN	66	95.0	2019	3

```
df.notnull()
```

	gender	math score	reading score	writing score	placement score	club join year	placement offer
0	True	True	True	True	True	True	True
1	True	True	True	True	True	True	True
2	True	True	True	True	True	True	True
3	True	True	True	True	True	True	True
4	True	True	True	True	False	True	True
5	True	True	True	True	True	True	True
6	True	True	True	True	True	True	True
7	True	False	False	True	True	True	True
8	True	True	True	True	True	True	True
9	True	True	True	True	True	True	True
10	True	True	True	True	True	True	True
11	True	True	True	True	False	True	True
12	True	True	True	True	True	True	True

```
seseries = pd.notnull(df["reading score"])
df[series]
```

	gender	math score	reading score	writing score	placement score	club join year	placement offer
7	female	NaN	NaN	66	95.0	2019	3

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder ()
df['gender'] = le.fit_transform(df['gender'])
newdf=df
df
```

	gender	math score	reading score	writing score	placement score	club join year	placement offer
0	0	63.0	84.0	64	84.0	2020	2
1	0	71.0	80.0	76	86.0	2018	3
2	0	64.0	81.0	66	81.0	2020	2
3	1	71.0	85.0	77	96.0	2018	1
4	1	68.0	86.0	76	NaN	2021	3
5	0	94.0	86.0	61	100.0	2019	1
6	1	75.0	79.0	66	-99.0	2020	1
7	0	NaN	NaN	66	95.0	2019	3
8	1	66.0	88.0	66	88.0	2020	3
9	1	70.0	79.0	61	87.0	2021	2
10	0	-99.0	80.0	65	85.0	2021	1
11	1	76.0	84.0	-99	NaN	2020	2
12	0	74.0	79.0	79	98.0	2019	2

```
df['gender'].replace({1: "male", 0: "female"}, inplace=True)
df
```

	gender	math score	reading score	writing score	placement score	club join year	placement offer
0	female	63.0	84.0	64	84.0	2020	2
1	female	71.0	80.0	76	86.0	2018	3
2	female	64.0	81.0	66	81.0	2020	2
3	male	71.0	85.0	77	96.0	2018	1
4	male	68.0	86.0	76	NaN	2021	3
5	female	94.0	86.0	61	100.0	2019	1
6	male	75.0	79.0	66	-99.0	2020	1
7	female	NaN	NaN	66	95.0	2019	3
8	male	66.0	88.0	66	88.0	2020	3
9	male	70.0	79.0	61	87.0	2021	2
10	female	-99.0	80.0	65	85.0	2021	1
11	male	76.0	84.0	-99	NaN	2020	2
12	female	74.0	79.0	79	98.0	2019	2

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

	gender	math score	reading score	writing score	placement score	club join year	placement offer
0	female	63.0	84.0	64	84.0	2020	2
1	female	71.0	80.0	76	86.0	2018	3
2	female	64.0	81.0	66	81.0	2020	2
3	male	71.0	85.0	77	96.0	2018	1
4	male	68.0	86.0	76	0.0	2021	3
5	female	94.0	86.0	61	100.0	2019	1
6	male	75.0	79.0	66	-99.0	2020	1
7	female	0.0	0.0	66	95.0	2019	3
8	male	66.0	88.0	66	88.0	2020	3
9	male	70.0	79.0	61	87.0	2021	2
10	female	-99.0	80.0	65	85.0	2021	1
11	male	76.0	84.0	-99	0.0	2020	2
12	female	74.0	79.0	79	98.0	2019	2

```
m_v=df['reading score'].mean()
df['reading score'].fillna(value=m_v, inplace=True)
df
```

	gender	math score	reading score	writing score	placement score	club join year	placement offer
0	female	63.0	84.000000	64	84.0	2020	2
1	female	71.0	80.000000	76	86.0	2018	3
2	female	64.0	81.000000	66	81.0	2020	2
3	male	71.0	85.000000	77	96.0	2018	1
4	male	68.0	86.000000	76	NaN	2021	3
5	female	94.0	86.000000	61	100.0	2019	1
6	male	75.0	79.000000	66	-99.0	2020	1
7	female	NaN	82.583333	66	95.0	2019	3
8	male	66.0	88.000000	66	88.0	2020	3
9	male	70.0	79.000000	61	87.0	2021	2
10	female	-99.0	80.000000	65	85.0	2021	1
11	male	76.0	84.000000	-99	NaN	2020	2
12	female	74.0	79.000000	79	98.0	2019	2

```
ndf.replace(to_replace = np.nan, value = -99)
```

	gender	math score	reading score	writing score	placement score	club join year	placement offer
0	female	63.0	84.000000	64	84.0	2020	2
1	female	71.0	80.000000	76	86.0	2018	3
2	female	64.0	81.000000	66	81.0	2020	2
3	male	71.0	85.000000	77	96.0	2018	1
4	male	68.0	86.000000	76	-99.0	2021	3
5	female	94.0	86.000000	61	100.0	2019	1
6	male	75.0	79.000000	66	-99.0	2020	1
7	female	-99.0	82.583333	66	95.0	2019	3
8	male	66.0	88.000000	66	88.0	2020	3
9	male	70.0	79.000000	61	87.0	2021	2
10	female	-99.0	80.000000	65	85.0	2021	1
11	male	76.0	84.000000	-99	-99.0	2020	2
12	female	74.0	79.000000	79	98.0	2019	2

```
ndf.dropna()
```

	gender	math score	reading score	writing score	placement score	club join year	placement offer
0	female	63.0	84.0	64	84.0	2020	2
1	female	71.0	80.0	76	86.0	2018	3
2	female	64.0	81.0	66	81.0	2020	2
3	male	71.0	85.0	77	96.0	2018	1
5	female	94.0	86.0	61	100.0	2019	1
6	male	75.0	79.0	66	-99.0	2020	1
8	male	66.0	88.0	66	88.0	2020	3
9	male	70.0	79.0	61	87.0	2021	2
10	female	-99.0	80.0	65	85.0	2021	1
12	female	74.0	79.0	79	98.0	2019	2

```
ndf.dropna(how = 'all')
```

	gender	math score	reading score	writing score	placement score	club join year	placement offer
0	female	63.0	84.000000	64	84.0	2020	2
1	female	71.0	80.000000	76	86.0	2018	3
2	female	64.0	81.000000	66	81.0	2020	2
3	male	71.0	85.000000	77	96.0	2018	1
4	male	68.0	86.000000	76	NaN	2021	3
5	female	94.0	86.000000	61	100.0	2019	1
6	male	75.0	79.000000	66	-99.0	2020	1
7	female	NaN	82.583333	66	95.0	2019	3
8	male	66.0	88.000000	66	88.0	2020	3
9	male	70.0	79.000000	61	87.0	2021	2
10	female	-99.0	80.000000	65	85.0	2021	1
11	male	76.0	84.000000	-99	NaN	2020	2
12	female	74.0	79.000000	79	98.0	2019	2

```
ndf.dropna(axis = 1)
```

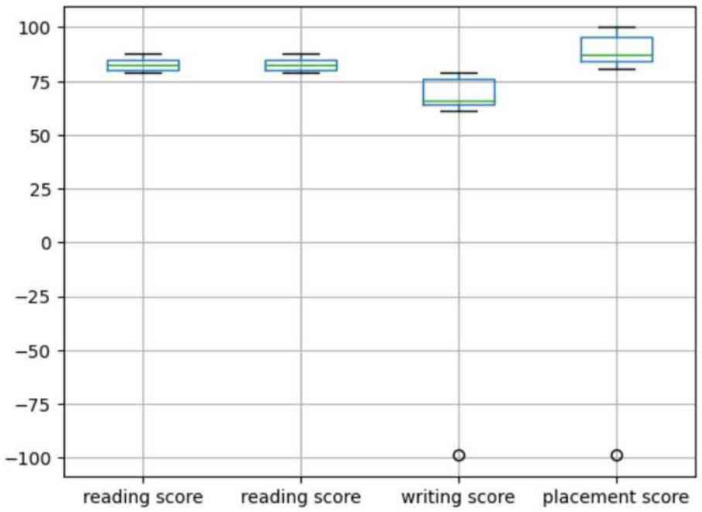
	gender	reading score	writing score	club join year	placement offer
0	female	84.000000	64	2020	2
1	female	80.000000	76	2018	3
2	female	81.000000	66	2020	2
3	male	85.000000	77	2018	1
4	male	86.000000	76	2021	3
5	female	86.000000	61	2019	1
6	male	79.000000	66	2020	1
7	female	82.583333	66	2019	3
8	male	88.000000	66	2020	3
9	male	79.000000	61	2021	2
10	female	80.000000	65	2021	1
11	male	84.000000	-99	2020	2
12	female	79.000000	79	2019	2

```
new_data =ndf.dropna (axis = 0, how ='any')
new_data
```

	gender	math score	reading score	writing score	placement score	club join year	placement offer
0	female	63.0	84.0	64	84.0	2020	2
1	female	71.0	80.0	76	86.0	2018	3
2	female	64.0	81.0	66	81.0	2020	2
3	male	71.0	85.0	77	96.0	2018	1
5	female	94.0	86.0	61	100.0	2019	1
6	male	75.0	79.0	66	-99.0	2020	1
8	male	66.0	88.0	66	88.0	2020	3
9	male	70.0	79.0	61	87.0	2021	2
10	female	-99.0	80.0	65	85.0	2021	1
12	female	74.0	79.0	79	98.0	2019	2

```
col =['reading score', 'reading score', 'writing score', 'placement score']
df.boxplot(col)
```

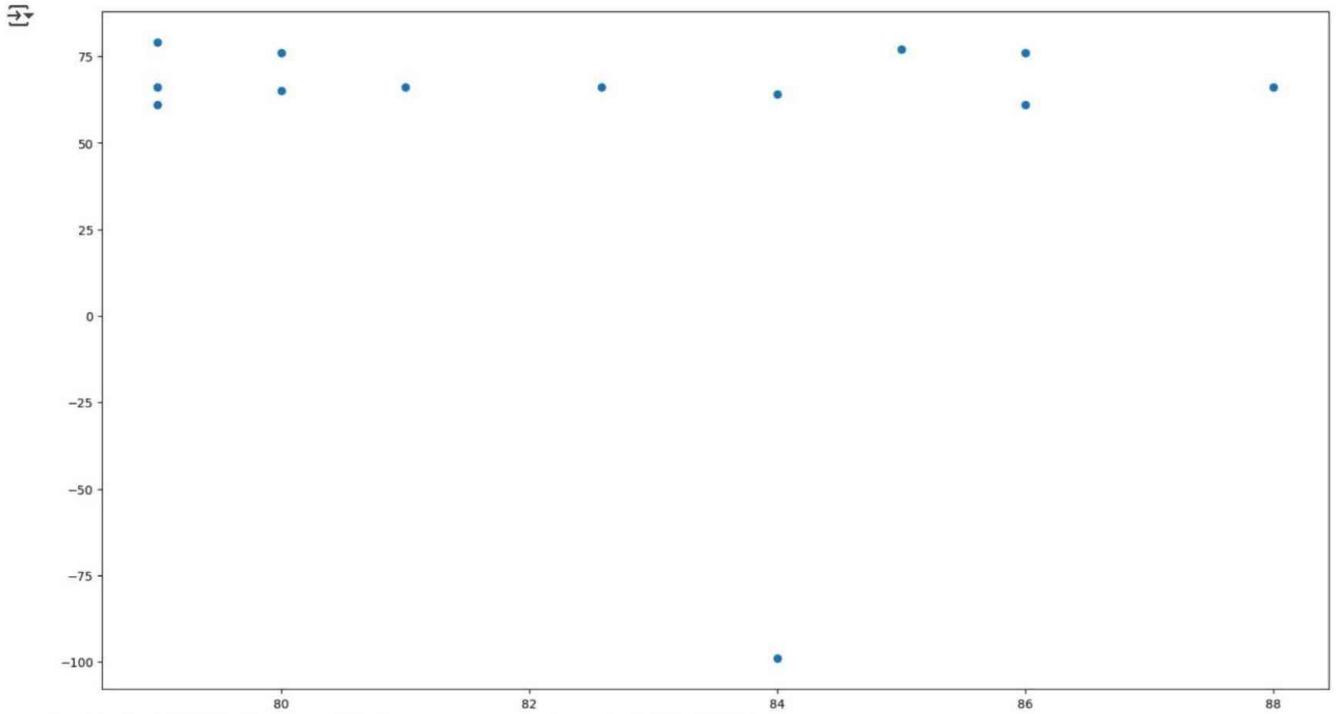
```
<AxesSubplot:>
```




```
print(np.where(df['reading score']>90))
print(np.where(df['writing score']>90))
```

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

```
fig, ax =plt.subplots(figsize = (18,10))
ax.scatter(df['reading score'], df['writing score'])
plt.show()
ax.set_xlabel('(Proportion non-retail business acres)/(town)')
ax.set_ylabel('(Full-value property-tax rate)/($10,000)')
```




```
Text(4.444444444444452, 0.5, '(Full-value property-tax rate)/($10,000)')
```

```
print(np.where((df['reading score']<50) & (df['writing score']>1)))
print(np.where((df['reading score']>85) & (df['writing score']<3)))
```

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

```
z = np.abs(stats.zscore(df['reading score']))
```

```
print(z)
```

```
0     0.472390
1     0.861418
2     0.527966
3     0.805843
4     1.139295
5     1.139295
6     1.194870
7     0.000000
8     1.806199
9     1.194870
10    0.861418
11    0.472390
12    1.194870
Name: reading score, dtype: float64
```

```
threshold = 0.18
```

```
sample_outliers = np.where(z <threshold)
```

```
sample_outliers
```

```
(array([7], dtype=int64),)
```

```
sorted_rscore= sorted(df['reading score'])
```

```
sorted_rscore
```

```
[79.0,
 79.0,
 79.0,
 80.0,
 80.0,
 81.0,
 82.58333333333333,
 84.0,
 84.0,
 85.0,
 86.0,
 86.0,
 88.0]
```

```
q1 = np.percentile(sorted_rscore, 25)
q3 = np.percentile(sorted_rscore, 75)
print(q1,q3)
```

```
80.0 85.0
```

```
IQR = q3-q1
```

```
lwr_bound = q1-(1.5*IQR)
upr_bound = q3+(1.5*IQR)
print(lwr_bound, upr_bound)
```

```
72.5 92.5
```

```
new_df=df
for i in sample_outliers:new_df.drop(i,inplace=True)
new_df
```

	gender	math score	reading score	writing score	placement score	club join year	placement offer
0	female	63.0	84.0	64	84.0	2020	2
1	female	71.0	80.0	76	86.0	2018	3
2	female	64.0	81.0	66	81.0	2020	2
3	male	71.0	85.0	77	96.0	2018	1
4	male	68.0	86.0	76	NaN	2021	3
5	female	94.0	86.0	61	100.0	2019	1
6	male	75.0	79.0	66	-99.0	2020	1
8	male	66.0	88.0	66	88.0	2020	3
9	male	70.0	79.0	61	87.0	2021	2
10	female	-99.0	80.0	65	85.0	2021	1
11	male	76.0	84.0	-99	NaN	2020	2
12	female	74.0	79.0	79	98.0	2019	2

```
df=pd.read_csv("D:\StudentPerformance.csv")
```

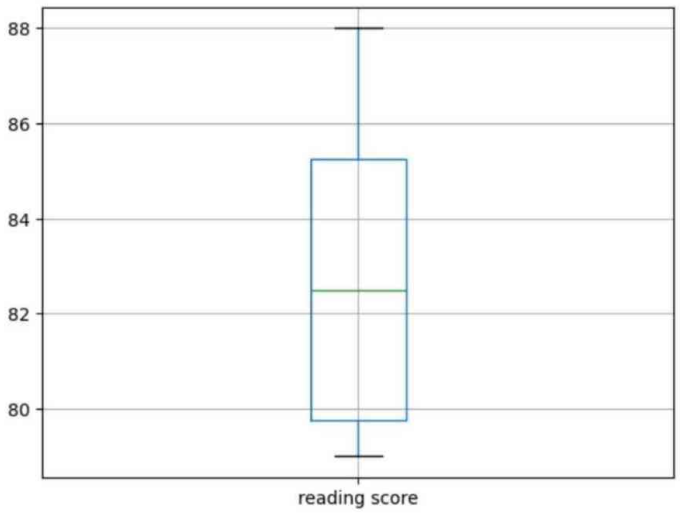
```
df_stud=df
ninetieth_percentile = np.percentile(df_stud['reading score'], 90)
b = np.where(df_stud['reading score']>ninetieth_percentile,
ninetieth_percentile, df_stud['reading score'])
print("New array:",b)
```

```
New array: [84. 80. 81. 85. 86. 86. 79. nan 88. 79. 80. 84. 79.]
```

```
col = ['reading score']

df.boxplot(col)
```

<AxesSubplot:>




```
median=np.median(sorted_rscore)
median
```

82.58333333333333

```
refined_df=df
refined_df['reading score'] = np.where(refined_df['reading score'] >upr_bound, median,refined_df['reading score'])
```

```
df
```

	gender	math score	reading score	writing score	placement score	club join year	placement offer
0	female	63.0	84.0	64	84.0	2020	2
1	female	71.0	80.0	76	86.0	2018	3
2	female	64.0	81.0	66	81.0	2020	2
3	male	71.0	85.0	77	96.0	2018	1
4	male	68.0	86.0	76	NaN	2021	3
5	female	94.0	86.0	61	100.0	2019	1
6	male	75.0	79.0	66	-99.0	2020	1
7	female	NaN	NaN	66	95.0	2019	3
8	male	66.0	88.0	66	88.0	2020	3
9	male	70.0	79.0	61	87.0	2021	2
10	female	-99.0	80.0	65	85.0	2021	1
11	male	76.0	84.0	-99	NaN	2020	2
12	female	74.0	79.0	79	98.0	2019	2

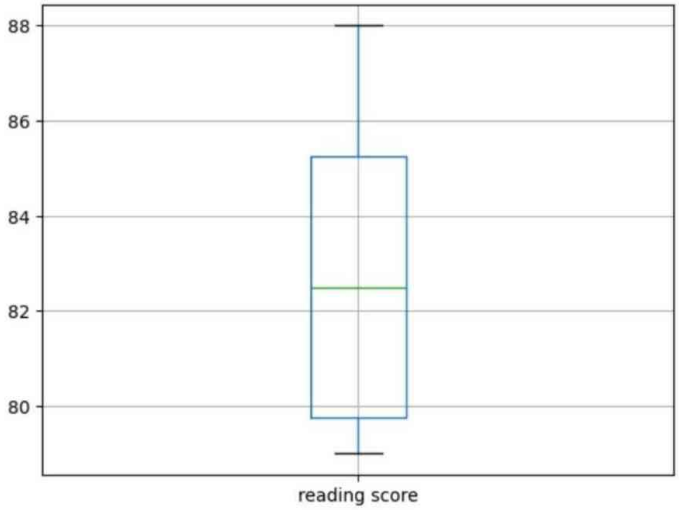
```
refined_df['reading score'] = np.where(refined_df['reading score'] <lwr_bound, median,refined_df['reading score'])
```

```
df
```

	gender	math score	reading score	writing score	placement score	club join year	placement offer
0	female	63.0	84.0	64	84.0	2020	2
1	female	71.0	80.0	76	86.0	2018	3
2	female	64.0	81.0	66	81.0	2020	2
3	male	71.0	85.0	77	96.0	2018	1
4	male	68.0	86.0	76	NaN	2021	3
5	female	94.0	86.0	61	100.0	2019	1
6	male	75.0	79.0	66	-99.0	2020	1
7	female	NaN	NaN	66	95.0	2019	3
8	male	66.0	88.0	66	88.0	2020	3
9	male	70.0	79.0	61	87.0	2021	2
10	female	-99.0	80.0	65	85.0	2021	1
11	male	76.0	84.0	-99	NaN	2020	2
12	female	74.0	79.0	79	98.0	2019	2

```
col = ['reading score']
refined_df.boxplot(col)
```

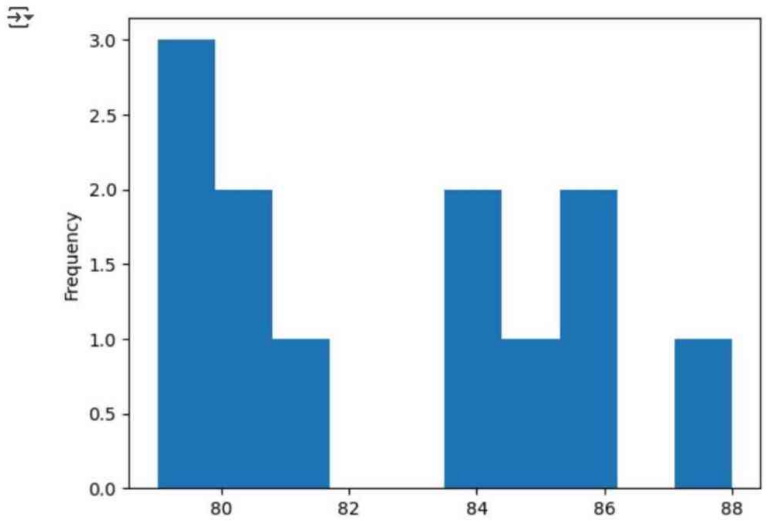
```
<AxesSubplot:>
```




```
df
```

	gender	math score	reading score	writing score	placement score	club join year	placement offer
0	female	63.0	84.0	64	84.0	2020	2
1	female	71.0	80.0	76	86.0	2018	3
2	female	64.0	81.0	66	81.0	2020	2
3	male	71.0	85.0	77	96.0	2018	1
4	male	68.0	86.0	76	NaN	2021	3
5	female	94.0	86.0	61	100.0	2019	1
6	male	75.0	79.0	66	-99.0	2020	1
7	female	NaN	NaN	66	95.0	2019	3
8	male	66.0	88.0	66	88.0	2020	3
9	male	70.0	79.0	61	87.0	2021	2
10	female	-99.0	80.0	65	85.0	2021	1
11	male	76.0	84.0	-99	NaN	2020	2
12	female	74.0	79.0	79	98.0	2019	2

```
import matplotlib.pyplot as plt
new_df['reading score'].plot(kind ='hist')
df['log_math'] = np.log10(df['reading score'])
```



```
import pandas as pd
import numpy as np
import requests
```

```
df=pd.read_csv("D:\DSBDA\Assign_3_Mall_Customers.csv")
```

```
df
```

	CustomerID	Genre	Age Group	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	Teen	19	15	39
1	2	Male	Middle Age	21	15	81
2	3	Female	Middle Age	20	16	6
3	4	Female	Middle Age	23	16	77
4	5	Female	Middle Age	31	17	40
...	...	...	...	...	...	...
195	196	Female	Middle Age	35	120	79
196	197	Female	Elder	45	126	28
197	198	Male	Middle Age	32	126	74
198	199	Male	Middle Age	32	137	18
199	200	Male	Middle Age	30	137	83

200 rows × 6 columns

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 6 columns):
 #   Column                  Non-Null Count  Dtype 
---  ------                  --------------  ----- 
 0   CustomerID              200 non-null    int64 
 1   Genre                   200 non-null    object
 2   Age Group               200 non-null    object
 3   Age                     200 non-null    int64 
 4   Annual Income (k$)      200 non-null    int64 
 5   Spending Score (1-100)  200 non-null    int64 
dtypes: int64(4), object(2)
memory usage: 9.5+ KB
```

```
df.shape
```

```
(200, 6)
```

```
df.head
```

```
<bound method NDFrame.head of      CustomerID   Genre   Age Group  Age  Annual Income (k$)  \
0             1    Male        Teen   19                  15   
1             2    Male  Middle Age   21                  15   
2             3  Female  Middle Age   20                  16   
3             4  Female  Middle Age   23                  16   
4             5  Female  Middle Age   31                  17   
..          ...     ...         ...  ...                 ...   
195         196  Female  Middle Age   35                 120   
196         197  Female       Elder   45                 126   
197         198    Male  Middle Age   32                 126   
198         199    Male  Middle Age   32                 137   
199         200    Male  Middle Age   30                 137   

     Spending Score (1-100)
0                        39
1                        81
2                         6
3                        77
4                        40
..                      ...
195                      79
196                      28
197                      74
198                      18
199                      83

[200 rows x 6 columns]>
```

```
df.tail
```

```
<bound method NDFrame.tail of      CustomerID   Genre   Age Group  Age  Annual Income (k$)  \
0             1    Male        Teen   19                  15
1             2    Male  Middle Age   21                  15
2             3  Female  Middle Age   20                  16
3             4  Female  Middle Age   23                  16
4             5  Female  Middle Age   31                  17
..          ...     ...         ...  ...                 ...
195         196  Female  Middle Age   35                 120
196         197  Female       Elder   45                 126
197         198    Male  Middle Age   32                 126
198         199    Male  Middle Age   32                 137
199         200    Male  Middle Age   30                 137

     Spending Score (1-100)
0                        39
1                        81
2                         6
3                        77
4                        40
..                      ...
195                      79
196                      28
197                      74
198                      18
199                      83

[200 rows x 6 columns]>
```

```
df.describe()
```

	CustomerID	Age	Annual Income (k$)	Spending Score (1-100)
count	200.000000	200.000000	200.000000	200.000000
mean	100.500000	38.850000	60.560000	50.200000
std	57.879185	13.969007	26.264721	25.823522
min	1.000000	18.000000	15.000000	1.000000
25%	50.750000	28.750000	41.500000	34.750000
50%	100.500000	36.000000	61.500000	50.000000
75%	150.250000	49.000000	78.000000	73.000000
max	200.000000	70.000000	137.000000	99.000000

```
df.Age.mean()
```

```
38.85
```

```
df.Age.mode()
```

```
0    32
Name: Age, dtype: int64
```

```
df.Age.median()
```

```
36.0
```

```
df.groupby(['Age']).count()
```

Age	CustomerID	Genre	Age Group	Annual Income (k$)	Spending Score (1-100)
18	4	4	4	4	4
19	8	8	8	8	8
20	5	5	5	5	5
21	5	5	5	5	5
22	3	3	3	3	3
23	6	6	6	6	6
24	4	4	4	4	4
25	3	3	3	3	3
26	2	2	2	2	2
27	6	6	6	6	6
28	4	4	4	4	4
29	5	5	5	5	5
30	7	7	7	7	7
31	8	8	8	8	8
32	11	11	11	11	11
33	3	3	3	3	3
34	5	5	5	5	5
35	9	9	9	9	9
36	6	6	6	6	6
37	3	3	3	3	3
38	6	6	6	6	6
39	3	3	3	3	3
40	6	6	6	6	6
41	2	2	2	2	2
42	2	2	2	2	2
43	3	3	3	3	3
44	2	2	2	2	2
45	3	3	3	3	3
46	3	3	3	3	3
47	6	6	6	6	6
48	5	5	5	5	5
49	7	7	7	7	7
50	5	5	5	5	5
51	2	2	2	2	2
52	2	2	2	2	2
53	2	2	2	2	2
54	4	4	4	4	4
55	1	1	1	1	1
56	1	1	1	1	1
57	2	2	2	2	2
58	2	2	2	2	2
59	4	4	4	4	4
60	3	3	3	3	3
63	2	2	2	2	2
64	1	1	1	1	1
65	2	2	2	2	2
66	2	2	2	2	2
67	4	4	4	4	4

68	3	3	3	3	3
69	1	1	1	1	1
70	2	2	2	2	2

```
df.groupby(['Genre']).count()
```

Genre	CustomerID	Age Group	Age	Annual Income (k$)	Spending Score (1-100)
Female	112	112	112	112	112
Male	88	88	88	88	88

```
df.Age.std()
```

```
13.969007331558883
```

```
df[['Age' , 'Annual Income (k$)', 'Spending Score (1-100)']].mean()
```

```
Age                     38.85
Annual Income (k$)      60.56
Spending Score (1-100)  50.20
dtype: float64
```

```
df[['Age' , 'Annual Income (k$)', 'Spending Score (1-100)']].mode()
```

	Age	Annual Income (k$)	Spending Score (1-100)
0	32.0	54	42.0
1	NaN	78	NaN

```
df[['Age' , 'Annual Income (k$)', 'Spending Score (1-100)']].median()
```

```
Age                     36.0
Annual Income (k$)      61.5
Spending Score (1-100)  50.0
dtype: float64
```

```
df[['Age' , 'Annual Income (k$)', 'Spending Score (1-100)']].max()
```

```
Age                      70
Annual Income (k$)      137
Spending Score (1-100)   99
dtype: int64
```

```
df[['Age' , 'Annual Income (k$)', 'Spending Score (1-100)']].std()
```

```
Age                     13.969007
Annual Income (k$)      26.264721
Spending Score (1-100)  25.823522
dtype: float64
```

```
df2 = df.groupby('Genre')
```

```
df
```

	CustomerID	Genre	Age Group	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	Teen	19	15	39
1	2	Male	Middle Age	21	15	81
2	3	Female	Middle Age	20	16	6
3	4	Female	Middle Age	23	16	77
4	5	Female	Middle Age	31	17	40
...	...	...	...	...	...	...
195	196	Female	Middle Age	35	120	79
196	197	Female	Elder	45	126	28
197	198	Male	Middle Age	32	126	74
198	199	Male	Middle Age	32	137	18
199	200	Male	Middle Age	30	137	83

200 rows × 6 columns

```
for Genre, Genre_f in df2:
    print(Genre)
    print(Genre_f)
```

```
Female
     CustomerID   Genre   Age Group  Age  Annual Income (k$)  \
2             3  Female  Middle Age   20                  16
3             4  Female  Middle Age   23                  16
4             5  Female  Middle Age   31                  17
5             6  Female  Middle Age   22                  17
6             7  Female  Middle Age   35                  18
..          ...     ...         ...  ...                 ...
191         192  Female  Middle Age   32                 103
193         194  Female  Middle Age   38                 113
194         195  Female       Elder   47                 120
195         196  Female  Middle Age   35                 120
196         197  Female       Elder   45                 126

     Spending Score (1-100)
2                         6
3                        77
4                        40
5                        76
6                         6
..                      ...
191                      69
193                      91
194                      16
195                      79
196                      28

[112 rows x 6 columns]
Male
     CustomerID Genre   Age Group  Age  Annual Income (k$)  \
0             1  Male        Teen   19                  15
1             2  Male  Middle Age   21                  15
8             9  Male       Elder   64                  19
10           11  Male       Elder   67                  19
14           15  Male  Middle Age   37                  20
..          ...   ...         ...  ...                 ...
187         188  Male  Middle Age   28                 101
192         193  Male  Middle Age   33                 113
197         198  Male  Middle Age   32                 126
198         199  Male  Middle Age   32                 137
199         200  Male  Middle Age   30                 137

     Spending Score (1-100)
0                        39
1                        81
8                         3
10                       14
14                       13
..                      ...
187                      68
192                       8
197                      74
198                      18
199                      83

[88 rows x 6 columns]
```

```
df2.get_group('Male')
```

	CustomerID	Genre	Age Group	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	Teen	19	15	39
1	2	Male	Middle Age	21	15	81
8	9	Male	Elder	64	19	3
10	11	Male	Elder	67	19	14
14	15	Male	Middle Age	37	20	13
...	...	...	...	...	...	...
187	188	Male	Middle Age	28	101	68
192	193	Male	Middle Age	33	113	8
197	198	Male	Middle Age	32	126	74
198	199	Male	Middle Age	32	137	18
199	200	Male	Middle Age	30	137	83

88 rows × 6 columns

```
df2.get_group('Female')
```

	CustomerID	Genre	Age Group	Age	Annual Income (k$)	Spending Score (1-100)
2	3	Female	Middle Age	20	16	6
3	4	Female	Middle Age	23	16	77
4	5	Female	Middle Age	31	17	40
5	6	Female	Middle Age	22	17	76
6	7	Female	Middle Age	35	18	6
...	...	...	...	...	...	...
191	192	Female	Middle Age	32	103	69
193	194	Female	Middle Age	38	113	91
194	195	Female	Elder	47	120	16
195	196	Female	Middle Age	35	120	79
196	197	Female	Elder	45	126	28

112 rows × 6 columns

```
df2[['Age' , 'Annual Income (k$)', 'Spending Score (1-100)']].median()
```

Genre	Age	Annual Income (k$)	Spending Score (1-100)
Female	35.0	60.0	50.0
Male	37.0	62.5	50.0

```
df2[['Age' , 'Annual Income (k$)', 'Spending Score (1-100)']].mean()
```

Genre	Age	Annual Income (k$)	Spending Score (1-100)
Female	38.098214	59.250000	51.526786
Male	39.806818	62.227273	48.511364

```
df2[['Age' , 'Annual Income (k$)', 'Spending Score (1-100)']].max()
```

Genre	Age	Annual Income (k$)	Spending Score (1-100)
Female	68	126	99
Male	70	137	97

```
df2[['Age' , 'Annual Income (k$)', 'Spending Score (1-100)']].min()
```

Genre	Age	Annual Income (k$)	Spending Score (1-100)
Female	18	16	5
Male	18	15	1

```
df2[['Age' , 'Annual Income (k$)', 'Spending Score (1-100)']].std()
```

Genre	Age	Annual Income (k$)	Spending Score (1-100)
Female	12.644095	26.011952	24.11495
Male	15.514812	26.638373	27.89677

```
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
r = requests.get(url)
```

```
df3 = pd.read_csv(url)
```

```
df3
```

	5.1	3.5	1.4	0.2	Iris-setosa
0	4.9	3.0	1.4	0.2	Iris-setosa
1	4.7	3.2	1.3	0.2	Iris-setosa
2	4.6	3.1	1.5	0.2	Iris-setosa
3	5.0	3.6	1.4	0.2	Iris-setosa
4	5.4	3.9	1.7	0.4	Iris-setosa
...	...	...	...	...	...
144	6.7	3.0	5.2	2.3	Iris-virginica
145	6.3	2.5	5.0	1.9	Iris-virginica
146	6.5	3.0	5.2	2.0	Iris-virginica
147	6.2	3.4	5.4	2.3	Iris-virginica
148	5.9	3.0	5.1	1.8	Iris-virginica

149 rows × 5 columns

```
df3.columns=("A" , "B" , " C " , "D" , "E")
```

```
df3
```

	A	B	C	D	E
0	4.9	3.0	1.4	0.2	Iris-setosa
1	4.7	3.2	1.3	0.2	Iris-setosa
2	4.6	3.1	1.5	0.2	Iris-setosa
3	5.0	3.6	1.4	0.2	Iris-setosa
4	5.4	3.9	1.7	0.4	Iris-setosa
...	...	...	...	...	...
144	6.7	3.0	5.2	2.3	Iris-virginica
145	6.3	2.5	5.0	1.9	Iris-virginica
146	6.5	3.0	5.2	2.0	Iris-virginica
147	6.2	3.4	5.4	2.3	Iris-virginica
148	5.9	3.0	5.1	1.8	Iris-virginica

149 rows × 5 columns

```
df4 =df3.groupby("E")
```

```
df4
```

```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x000001EF351D8D30>
```

```
df4.get_group("Iris-setosa")
```

	A	B	C	D	E
0	4.9	3.0	1.4	0.2	Iris-setosa
1	4.7	3.2	1.3	0.2	Iris-setosa
2	4.6	3.1	1.5	0.2	Iris-setosa
3	5.0	3.6	1.4	0.2	Iris-setosa
4	5.4	3.9	1.7	0.4	Iris-setosa
5	4.6	3.4	1.4	0.3	Iris-setosa
6	5.0	3.4	1.5	0.2	Iris-setosa
7	4.4	2.9	1.4	0.2	Iris-setosa
8	4.9	3.1	1.5	0.1	Iris-setosa
9	5.4	3.7	1.5	0.2	Iris-setosa
10	4.8	3.4	1.6	0.2	Iris-setosa
11	4.8	3.0	1.4	0.1	Iris-setosa
12	4.3	3.0	1.1	0.1	Iris-setosa
13	5.8	4.0	1.2	0.2	Iris-setosa
14	5.7	4.4	1.5	0.4	Iris-setosa
15	5.4	3.9	1.3	0.4	Iris-setosa
16	5.1	3.5	1.4	0.3	Iris-setosa
17	5.7	3.8	1.7	0.3	Iris-setosa
18	5.1	3.8	1.5	0.3	Iris-setosa
19	5.4	3.4	1.7	0.2	Iris-setosa
20	5.1	3.7	1.5	0.4	Iris-setosa
21	4.6	3.6	1.0	0.2	Iris-setosa
22	5.1	3.3	1.7	0.5	Iris-setosa
23	4.8	3.4	1.9	0.2	Iris-setosa
24	5.0	3.0	1.6	0.2	Iris-setosa
25	5.0	3.4	1.6	0.4	Iris-setosa
26	5.2	3.5	1.5	0.2	Iris-setosa
27	5.2	3.4	1.4	0.2	Iris-setosa
28	4.7	3.2	1.6	0.2	Iris-setosa
29	4.8	3.1	1.6	0.2	Iris-setosa
30	5.4	3.4	1.5	0.4	Iris-setosa
31	5.2	4.1	1.5	0.1	Iris-setosa
32	5.5	4.2	1.4	0.2	Iris-setosa
33	4.9	3.1	1.5	0.1	Iris-setosa
34	5.0	3.2	1.2	0.2	Iris-setosa
35	5.5	3.5	1.3	0.2	Iris-setosa
36	4.9	3.1	1.5	0.1	Iris-setosa
37	4.4	3.0	1.3	0.2	Iris-setosa
38	5.1	3.4	1.5	0.2	Iris-setosa
39	5.0	3.5	1.3	0.3	Iris-setosa
40	4.5	2.3	1.3	0.3	Iris-setosa
41	4.4	3.2	1.3	0.2	Iris-setosa
42	5.0	3.5	1.6	0.6	Iris-setosa
43	5.1	3.8	1.9	0.4	Iris-setosa
44	4.8	3.0	1.4	0.3	Iris-setosa
45	5.1	3.8	1.6	0.2	Iris-setosa
46	4.6	3.2	1.4	0.2	Iris-setosa
47	5.3	3.7	1.5	0.2	Iris-setosa
48	5.0	3.3	1.4	0.2	Iris-setosa

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

```
from sklearn.datasets import load_boston
boston =load_boston()
```

```
data = pd.DataFrame(boston.data)
```

```
data.columns = boston.feature_names
data.head()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33

```
data['PRICE'] = boston.target
```

```
data
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	PRICE
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33	36.2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
501	0.06263	0.0	11.93	0.0	0.573	6.593	69.1	2.4786	1.0	273.0	21.0	391.99	9.67	22.4
502	0.04527	0.0	11.93	0.0	0.573	6.120	76.7	2.2875	1.0	273.0	21.0	396.90	9.08	20.6
503	0.06076	0.0	11.93	0.0	0.573	6.976	91.0	2.1675	1.0	273.0	21.0	396.90	5.64	23.9
504	0.10959	0.0	11.93	0.0	0.573	6.794	89.3	2.3889	1.0	273.0	21.0	393.45	6.48	22.0
505	0.04741	0.0	11.93	0.0	0.573	6.030	80.8	2.5050	1.0	273.0	21.0	396.90	7.88	11.9

506 rows × 14 columns

```
data.isnull().sum()
```

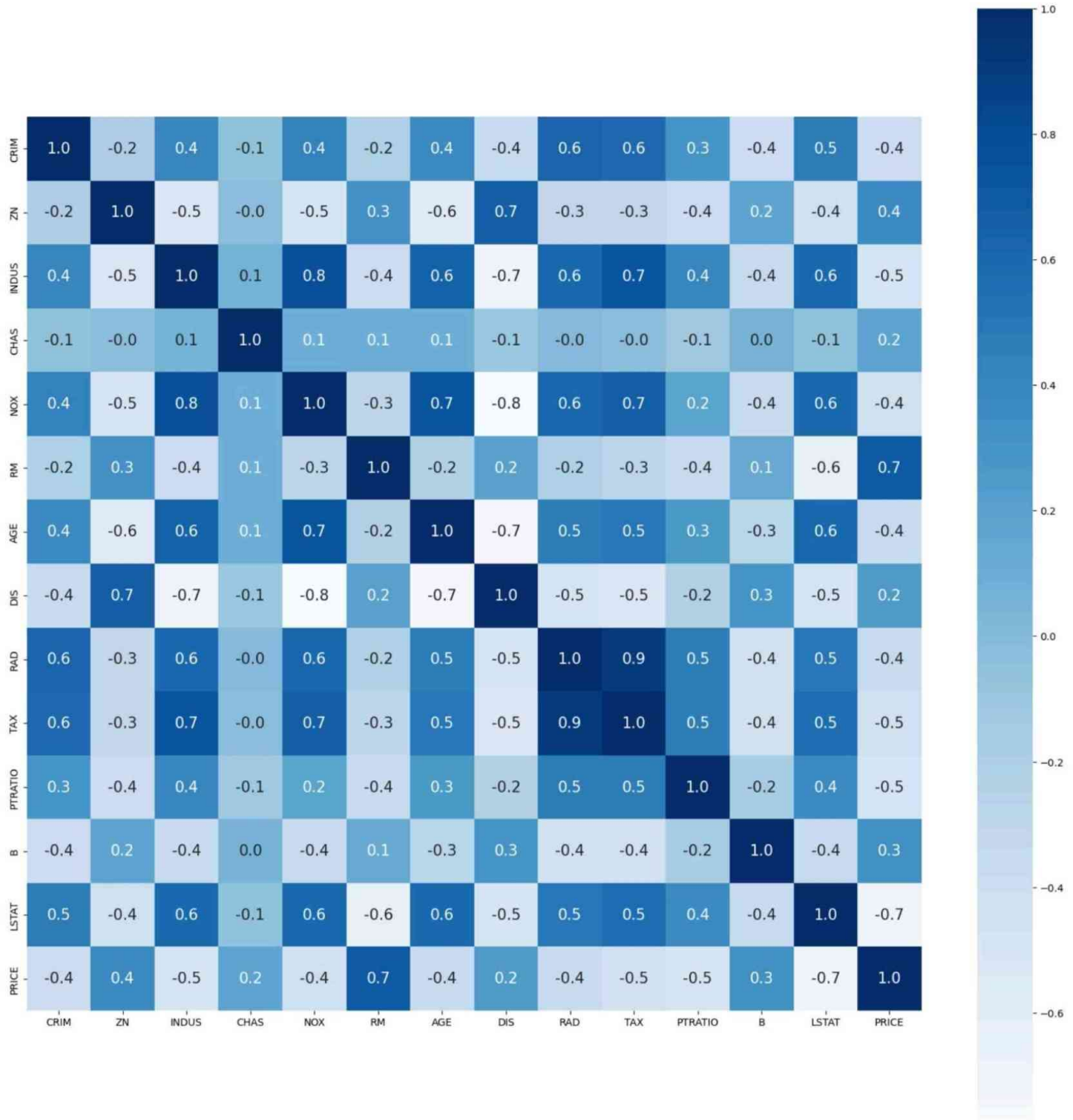
```
CRIM       0
ZN         0
INDUS      0
CHAS       0
NOX        0
RM         0
AGE        0
DIS        0
RAD        0
TAX        0
PTRATIO    0
B          0
LSTAT      0
PRICE      0
dtype: int64
```

```
corr = data.corr()
corr.shape
```

```
(14, 14)
```

```
plt.figure(figsize=(20,20))
sns.heatmap(corr, cbar=True, square= True, fm='.1f', annot=True, annot_kws={'size':15}, cmap='Blues')
```

```
plt.show()
```


CRIM 1.0 -0.2 0.4 -0.1 0.4 -0.2 0.4 -0.4 0.6 0.6 0.3 -0.4 0.5 -0.4
ZN -0.2 1.0 -0.5 -0.0 -0.5 0.3 -0.6 0.7 -0.3 -0.3 -0.4 0.2 -0.4 0.4
INDUS 0.4 -0.5 1.0 0.1 0.8 -0.4 0.6 -0.7 0.6 0.7 0.4 -0.4 0.6 -0.5
CHAS -0.1 -0.0 0.1 1.0 0.1 0.1 0.1 -0.1 -0.0 -0.0 -0.1 0.0 -0.1 0.2
NOX 0.4 -0.5 0.8 0.1 1.0 -0.3 0.7 -0.8 0.6 0.7 0.2 -0.4 0.6 -0.4
RM -0.2 0.3 -0.4 0.1 -0.3 1.0 -0.2 0.2 -0.2 -0.3 -0.4 0.1 -0.6 0.7
AGE 0.4 -0.6 0.6 0.1 0.7 -0.2 1.0 -0.7 0.5 0.5 0.3 -0.3 0.6 -0.4
DIS -0.4 0.7 -0.7 -0.1 -0.8 0.2 -0.7 1.0 -0.5 -0.5 -0.2 0.3 -0.5 0.2
RAD 0.6 -0.3 0.6 -0.0 0.6 -0.2 0.5 -0.5 1.0 0.9 0.5 -0.4 0.5 -0.4
TAX 0.6 -0.3 0.7 -0.0 0.7 -0.3 0.5 -0.5 0.9 1.0 0.5 -0.4 0.5 -0.5
PTRATIO 0.3 -0.4 0.4 -0.1 0.2 -0.4 0.3 -0.2 0.5 0.5 1.0 -0.2 0.4 -0.5
B -0.4 0.2 -0.4 0.0 -0.4 0.1 -0.3 0.3 -0.4 -0.4 -0.2 1.0 -0.4 0.3
LSTAT 0.5 -0.4 0.6 -0.1 0.6 -0.6 0.6 -0.5 0.5 0.5 0.4 -0.4 1.0 -0.7
PRICE -0.4 0.4 -0.5 0.2 -0.4 0.7 -0.4 0.2 -0.4 -0.5 -0.5 0.3 -0.7 1.0
CRIM ZN INDUS CHAS NOX RM AGE DIS RAD TAX PTRATIO B LSTAT PRICE
1.0 0.8 0.6 0.4 0.2 0.0 −0.2 −0.4 −0.6

```
x = data.drop(['PRICE'], axis = 1)
y = data['PRICE']
```

```
from sklearn.model_selection import train_test_split
xtrain, xtest, ytrain, ytest =train_test_split(x, y, test_size =0.2,random_state = 0)
```

```
import sklearn
from sklearn.linear_model import LinearRegression
lm = LinearRegression()
# Train the model using the training sets
model=lm.fit(xtrain,ytrain)
```

```
xtrain
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
220	0.35809	0.0	6.20	1.0	0.507	6.951	88.5	2.8617	8.0	307.0	17.4	391.70	9.71
71	0.15876	0.0	10.81	0.0	0.413	5.961	17.5	5.2873	4.0	305.0	19.2	376.94	9.88
240	0.11329	30.0	4.93	0.0	0.428	6.897	54.3	6.3361	6.0	300.0	16.6	391.25	11.38
6	0.08829	12.5	7.87	0.0	0.524	6.012	66.6	5.5605	5.0	311.0	15.2	395.60	12.43
417	25.94060	0.0	18.10	0.0	0.679	5.304	89.1	1.6475	24.0	666.0	20.2	127.36	26.64
...	...	...	...	...	...	...	...	...	...	...	...	...	...
323	0.28392	0.0	7.38	0.0	0.493	5.708	74.3	4.7211	5.0	287.0	19.6	391.13	11.74
192	0.08664	45.0	3.44	0.0	0.437	7.178	26.3	6.4798	5.0	398.0	15.2	390.49	2.87
117	0.15098	0.0	10.01	0.0	0.547	6.021	82.6	2.7474	6.0	432.0	17.8	394.51	10.30
47	0.22927	0.0	6.91	0.0	0.448	6.030	85.5	5.6894	3.0	233.0	17.9	392.74	18.80
172	0.13914	0.0	4.05	0.0	0.510	5.572	88.5	2.5961	5.0	296.0	16.6	396.90	14.69

404 rows × 13 columns

```
ytrain_pred=lm.predict(xtrain)
ytest_pred=lm.predict(xtest)
```

```
testdata=[[0.00632,18.0,2.31,0.0,0.538,6.575,65.2,4.0900,1.0,296.0,15.3,396.90,4.98]]
```

```
test_pred = lm.predict(testdata)
test_pred
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature names, but LinearRegressi
  warnings.warn(
array([30.49949836])
```

```
df1=pd.DataFrame(ytrain_pred,ytrain)
df2=pd.DataFrame(ytest_pred,ytest)
df1
```

PRICE	0
26.7	32.556927
21.7	21.927095
22.0	27.543826
22.9	23.603188
10.4	6.571910
...	...
18.5	19.494951
36.4	33.326364
19.2	23.796208
16.6	18.458353
23.1	23.249181

404 rows × 1 columns

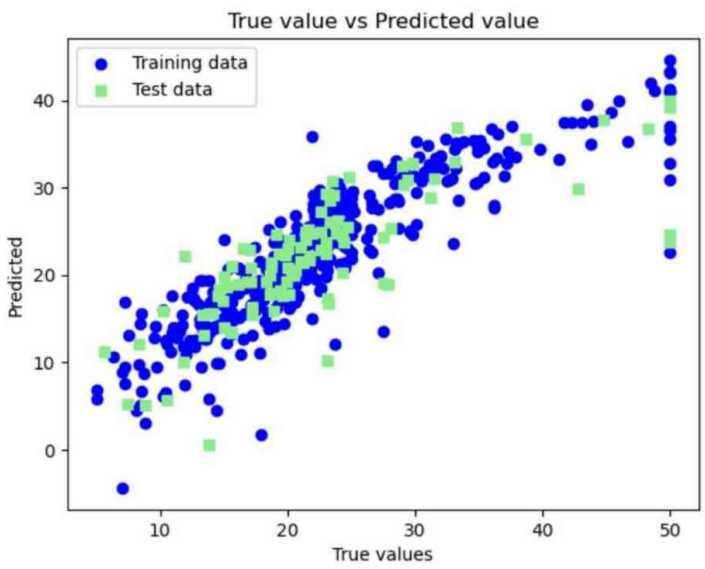
```
from sklearn.metrics import mean_squared_error, r2_score
mse = mean_squared_error(ytest, ytest_pred)
print('MSE on test data:',mse)
mse1 = mean_squared_error(ytrain_pred, ytrain)
print('MSE on training data:',mse1)
```

```
MSE on test data: 33.44897999767653
MSE on training data: 19.326470203585725
```

```
#from sklearn.metrics import mean_squared_error
#def linear_metrics():
r2 = lm.score(xtest, ytest)
rmse = (np.sqrt(mean_squared_error(ytest, ytest_pred)))
print('r-squared: {}' .format(r2))
print('----------------------------------------')
print('root mean squared error: {}'.format(rmse))
```

```
r-squared: 0.5892223849182507
----------------------------------------
root mean squared error: 5.783509315085135
```

```
#plotting the linear regression model
plt.scatter(ytrain ,ytrain_pred,c='blue',marker='o',label='Training data')
plt.scatter(ytest,ytest_pred ,c='lightgreen',marker='s',label='Test data')
plt.xlabel('True values')
plt.ylabel('Predicted')
plt.title("True value vs Predicted value")
plt.legend(loc= 'upper left') #plt.hlines(y=0,xmin=0,xmax=50)
plt.plot()
plt.show()
```




```
testdata=[[0.00632,18.0,2.31,0.0,0.538,6.575,65.2,4.0900,1.0,296.0,15.3,396.90,4.98]]
```

```
test_pred = lm.predict(testdata)
test_pred
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature names, but LinearRegressi
  warnings.warn(
array([30.49949836])
```

```
Start coding or generate with AI.
```

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

```
dataset = pd.read_csv('https://raw.githubusercontent.com/shivang98/Social-Network-ads-Boost/master/Social_Network_Ads.csv')
```

```
dataset
```

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0
...	...	...	...	...	...
395	15691863	Female	46	41000	1
396	15706071	Male	51	23000	1
397	15654296	Female	50	20000	1
398	15755018	Male	36	33000	0
399	15594041	Female	49	36000	1

400 rows × 5 columns

```
dataset.head()
```

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0

```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   User ID          400 non-null    int64
 1   Gender           400 non-null    object
 2   Age              400 non-null    int64
 3   EstimatedSalary  400 non-null    int64
 4   Purchased        400 non-null    int64
dtypes: int64(4), object(1)
memory usage: 15.8+ KB
```

```
X = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:, 4].values

print(X[:3, :])
print('-'*15)
print(y[:3])
```

```
[[   19 19000]
 [   35 20000]
 [   26 43000]]
---------------
[0 0 0]
```

```
dataset.tail()
```

	User ID	Gender	Age	EstimatedSalary	Purchased
395	15691863	Female	46	41000	1
396	15706071	Male	51	23000	1
397	15654296	Female	50	20000	1
398	15755018	Male	36	33000	0
399	15594041	Female	49	36000	1

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)

print(X_train[:3])
print('-'*15)
print(y_train[:3])
print('-'*15)
print(X_test[:3])
print('-'*15)
print(y_test[:3])
```

```
[[    44  39000]
 [    32 120000]
 [    38  50000]]
---------------
[0 1 0]
---------------
[[   30 87000]
 [   38 50000]
 [   35 75000]]
---------------
[0 0 0]
```

```
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)
```

```
print(X_train[:3])
print('-'*15)
print(X_test[:3])
```

```
[[ 0.58164944 -0.88670699]
 [-0.60673761  1.46173768]
 [-0.01254409 -0.5677824 ]]
---------------
[[-0.80480212  0.50496393]
 [-0.01254409 -0.5677824 ]
 [-0.30964085  0.1570462 ]]
```

```
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0, solver='lbfgs' )
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)

print(X_test[:10])
print('-'*15)
print(y_pred[:10])
```

```
[[-0.80480212  0.50496393]
 [-0.01254409 -0.5677824 ]
 [-0.30964085  0.1570462 ]
 [-0.80480212  0.27301877]
 [-0.30964085 -0.5677824 ]
 [-1.10189888 -1.43757673]
 [-0.70576986 -1.58254245]
 [-0.21060859  2.15757314]
 [-1.99318916 -0.04590581]
 [ 0.8787462  -0.77073441]]
---------------
[0 0 0 0 0 0 0 1 0 1]
```

```
print(y_pred[:20])
print(y_test[:20])
```

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

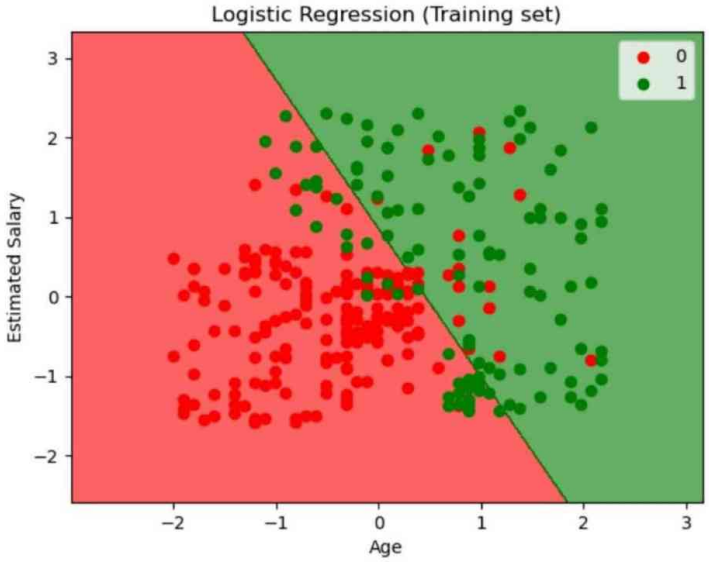
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
```

```
print(cm)
```

```
[[65  3]
 [ 8 24]]
```

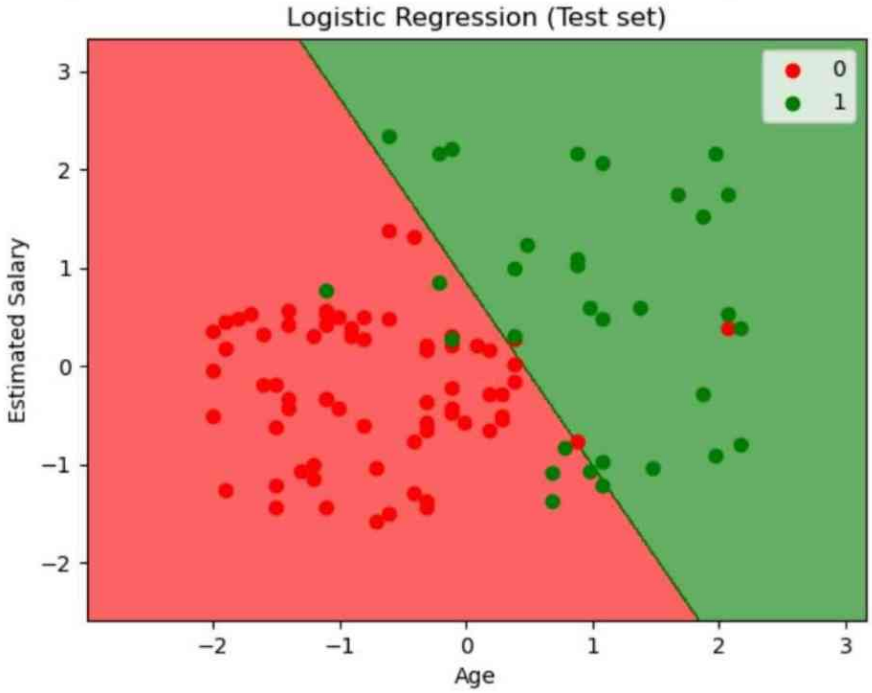
```
from matplotlib.colors import ListedColormap
X_set, y_set = X_train, y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.6, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('Logistic Regression (Training set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

```
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have precedence in case
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have precedence in case
```




```
from matplotlib.colors import ListedColormap
X_set, y_set = X_test, y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.6, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('Logistic Regression (Test set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

```
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have
```




```
#Prashant Jagtap
```

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix,ConfusionMatrixDisplay,classification_report,accuracy_score, precision_score, recall_score,
from sklearn.preprocessing import LabelEncoder
```

```
data =  pd.read_csv("https://raw.githubusercontent.com/venky14/Machine-Learning-with-Iris-Dataset/master/Iris.csv")
```

```
data
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 6 columns

```
data.head(5)
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

```
data.tail()
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

```
data.describe(include = 'all')
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
count	150.000000	150.000000	150.000000	150.000000	150.000000	150
unique	NaN	NaN	NaN	NaN	NaN	3
top	NaN	NaN	NaN	NaN	NaN	Iris-setosa
freq	NaN	NaN	NaN	NaN	NaN	50
mean	75.500000	5.843333	3.054000	3.758667	1.198667	NaN
std	43.445368	0.828066	0.433594	1.764420	0.763161	NaN
min	1.000000	4.300000	2.000000	1.000000	0.100000	NaN
25%	38.250000	5.100000	2.800000	1.600000	0.300000	NaN
50%	75.500000	5.800000	3.000000	4.350000	1.300000	NaN
75%	112.750000	6.400000	3.300000	5.100000	1.800000	NaN
max	150.000000	7.900000	4.400000	6.900000	2.500000	NaN

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             150 non-null    int64
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

```
print(data.shape)
data['Species'].unique()
```

```
(150, 6)
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)
```

```
data.isnull().sum()
```

```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

```
x = data.iloc[:,1:5]
y = data.iloc[:,5:]
```

```
encode = LabelEncoder()
y = encode.fit_transform(y)
```

```
C:\Users\coeco\anaconda3\lib\site-packages\sklearn\preprocessing\_label.py:115: DataConversionWarning: A column-vector y was passed
  y = column_or_1d(y, warn=True)
```

```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.3,random_state = 0)
```

```
naive_bayes = GaussianNB()
naive_bayes.fit(x_train,y_train)
pred = naive_bayes.predict(x_test)
```

```
pred
```

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

```
y_test
```

```
array([2, 1, 0, 2, 0, 2, 0, 1, 1, 1, 2, 1, 1, 1, 1, 0, 1, 1, 0, 0, 2, 1,
       0, 0, 2, 0, 0, 1, 1, 0, 2, 1, 0, 2, 2, 1, 0, 1, 1, 1, 2, 0, 2, 0,
```

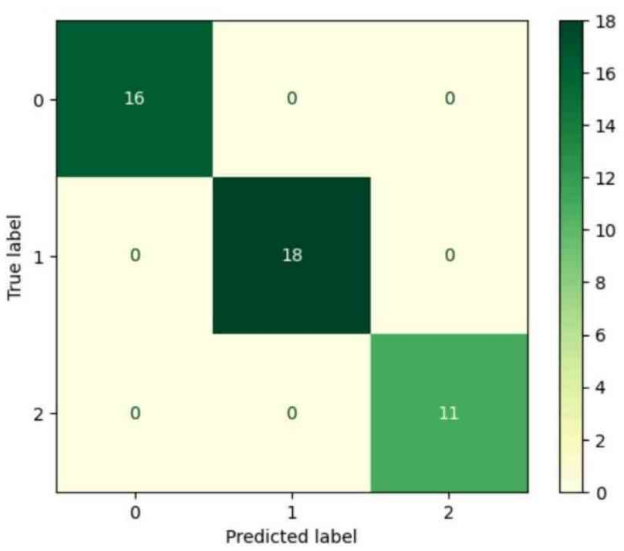
```
       0])
```

```
matrix =  confusion_matrix(y_test,pred,labels = naive_bayes.classes_)
print(matrix)

tp, fn, fp, tn = confusion_matrix(y_test,pred,labels=[1,0]).reshape(-1)
```

```
[[16  0  0]
 [ 0 18  0]
 [ 0  0 11]]
```

```
conf_matrix = ConfusionMatrixDisplay(confusion_matrix=matrix,display_labels=naive_bayes.classes_)
conf_matrix.plot(cmap=plt.cm.YlGn)
plt.show()
```




```
print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        16
           1       1.00      1.00      1.00        18
           2       1.00      1.00      1.00        11

    accuracy                           1.00        45
   macro avg       1.00      1.00      1.00        45
weighted avg       1.00      1.00      1.00        45
```

```
print('\nAccuracy: {:.2f}'.format(accuracy_score(y_test,pred)))
print('Error Rate: ',(fp+fn)/(tp+tn+fn+fp))
print('Sensitivity (Recall or True positive rate) :',tp/(tp+fn))
print('Specificity (True negative rate) :',tn/(fp+tn))
print('Precision (Positive predictive value) :',tp/(tp+fp))
print('False Positive Rate :',fp/(tn+fp))
```

```
Accuracy: 1.00
Error Rate:  0.0
Sensitivity (Recall or True positive rate) : 1.0
Specificity (True negative rate) : 1.0
Precision (Positive predictive value) : 1.0
False Positive Rate : 0.0
```

```
Start coding or generate with AI.
```

```
Start coding or generate with AI.
```

```
import nltk
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\pcoec\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
True
```

```
from nltk import word_tokenize, sent_tokenize
sent = "Sachin is considered to be one of the greatest cricket players. Virat is the captain of the Indian cricket team"
print(word_tokenize(sent))
print(sent_tokenize(sent))
```

```
['Sachin', 'is', 'considered', 'to', 'be', 'one', 'of', 'the', 'greatest', 'cricket', 'players', '.', 'Virat', 'is', 'the', 'captair
['Sachin is considered to be one of the greatest cricket players.', 'Virat is the captain of the Indian cricket team']
```

```
from nltk.corpus import stopwords
import nltk
nltk.download('stopwords')
stop_words = stopwords.words('english')
print(stop_words)
```

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'yours', 'yourse
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\pcoec\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

```
token = word_tokenize(sent)
cleaned_token = []
for word in token:
 if word not in stop_words:
    cleaned_token.append(word)

print("This is the unclean version : ",token)
print("This is the cleaned version : ",cleaned_token)
```

```
This is the unclean version :  ['Sachin', 'is', 'considered', 'to', 'be', 'one', 'of', 'the', 'greatest', 'cricket', 'players', '.',
This is the cleaned version :  ['Sachin', 'considered', 'one', 'greatest', 'cricket', 'players', '.', 'Virat', 'captain', 'Indian',
```

```
words = [cleaned_token.lower() for cleaned_token in cleaned_token if cleaned_token.isalpha()]
```

```
print(words)
```

```
['sachin', 'considered', 'one', 'greatest', 'cricket', 'players', 'virat', 'captain', 'indian', 'cricket', 'team']
```

```
from nltk.stem import PorterStemmer
stemmer = PorterStemmer()
port_stemmer_output = [stemmer.stem(words) for words in words]
print(port_stemmer_output)
```

```
['sachin', 'consid', 'one', 'greatest', 'cricket', 'player', 'virat', 'captain', 'indian', 'cricket', 'team']
```

```
from nltk.stem import WordNetLemmatizer
nltk.download('wordnet')
lemmatizer = WordNetLemmatizer()
lemmatizer_output = [lemmatizer.lemmatize(words) for words in words]
print(lemmatizer_output)
```

```
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\pcoec\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
['sachin', 'considered', 'one', 'greatest', 'cricket', 'player', 'virat', 'captain', 'indian', 'cricket', 'team']
```

```
from nltk import pos_tag
import nltk
nltk.download('averaged_perceptron_tagger')
token = word_tokenize(sent)
cleaned_token = []
for word in token:
 if word not in stop_words:
    cleaned_token.append(word)
```

```
tagged = pos_tag(cleaned_token)
print(tagged)
```

```
[('Sachin', 'NNP'), ('considered', 'VBD'), ('one', 'CD'), ('greatest', 'JJS'), ('cricket', 'NN'), ('players', 'NNS'), ('.', '.'), (
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     C:\Users\pcoec\AppData\Roaming\nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
```

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
import pandas as pd
```

```
docs = [ "Sachin is considered to be one of the greatest cricket players",
 "Federer is considered one of the greatest tennis players",
 "Nadal is considered one of the greatest tennis players",
 "Virat is the captain of the Indian cricket team"]
```

```
vectorizer = TfidfVectorizer(analyzer = "word", norm = None , use_idf = True , smooth_idf=True)
Mat = vectorizer.fit(docs)
print(Mat.vocabulary_)
```

```
{'sachin': 12, 'is': 7, 'considered': 2, 'to': 16, 'be': 0, 'one': 10, 'of': 9, 'the': 15, 'greatest': 5, 'cricket': 3, 'players': 1
```

```
tfidfMat = vectorizer.fit_transform(docs)
```

```
print(tfidfMat)
```

```
  (0, 11)       1.2231435513142097
  (0, 3)        1.5108256237659907
  (0, 5)        1.2231435513142097
  (0, 15)       1.0
  (0, 9)        1.0
  (0, 10)       1.2231435513142097
  (0, 0)        1.916290731874155
  (0, 16)       1.916290731874155
  (0, 2)        1.2231435513142097
  (0, 7)        1.0
  (0, 12)       1.916290731874155
  (1, 14)       1.5108256237659907
  (1, 4)        1.916290731874155
  (1, 11)       1.2231435513142097
  (1, 5)        1.2231435513142097
  (1, 15)       1.0
  (1, 9)        1.0
  (1, 10)       1.2231435513142097
  (1, 2)        1.2231435513142097
  (1, 7)        1.0
  (2, 8)        1.916290731874155
  (2, 14)       1.5108256237659907
  (2, 11)       1.2231435513142097
  (2, 5)        1.2231435513142097
  (2, 15)       1.0
  (2, 9)        1.0
  (2, 10)       1.2231435513142097
  (2, 2)        1.2231435513142097
  (2, 7)        1.0
  (3, 13)       1.916290731874155
  (3, 6)        1.916290731874155
  (3, 1)        1.916290731874155
  (3, 17)       1.916290731874155
  (3, 3)        1.5108256237659907
  (3, 15)       2.0
  (3, 9)        1.0
  (3, 7)        1.0
```

```
features_names = vectorizer.get_feature_names_out()
print(features_names)
```

```
['be' 'captain' 'considered' 'cricket' 'federer' 'greatest' 'indian' 'is'
 'nadal' 'of' 'one' 'players' 'sachin' 'team' 'tennis' 'the' 'to' 'virat']
```

```
dense = tfidfMat.todense()
denselist = dense.tolist()
df = pd.DataFrame(denselist , columns = features_names)
```

```
df
```

	be	captain	considered	cricket	federer	greatest	indian	is	nadal	of	one	players	sachin	team	te
0	1.916291	0.000000	1.223144	1.510826	0.000000	1.223144	0.000000	1.0	0.000000	1.0	1.223144	1.223144	1.916291	0.000000	0.0(
1	0.000000	0.000000	1.223144	0.000000	1.916291	1.223144	0.000000	1.0	0.000000	1.0	1.223144	1.223144	0.000000	0.000000	1.51
2	0.000000	0.000000	1.223144	0.000000	0.000000	1.223144	0.000000	1.0	1.916291	1.0	1.223144	1.223144	0.000000	0.000000	1.51
3	0.000000	1.916291	0.000000	1.510826	0.000000	0.000000	1.916291	1.0	0.000000	1.0	0.000000	0.000000	0.000000	1.916291	0.0(

```
features_names = sorted(vectorizer.get_feature_names())
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_16340\2669239957.py in <module>
----> 1 features_names = sorted(vectorizer.get_feature_names())

AttributeError: 'TfidfVectorizer' object has no attribute 'get_feature_names'
```

```
docList = ['Doc 1','Doc 2','Doc 3','Doc 4']
skDocsIfIdfdf = pd.DataFrame(tfidfMat.todense(),index = sorted(docList), columns=features_names)
print(skDocsIfIdfdf)
```

```
             be   captain  considered   cricket   federer  greatest    indian  \
Doc 1  1.916291  0.000000    1.223144  1.510826  0.000000  1.223144  0.000000
Doc 2  0.000000  0.000000    1.223144  0.000000  1.916291  1.223144  0.000000
Doc 3  0.000000  0.000000    1.223144  0.000000  0.000000  1.223144  0.000000
Doc 4  0.000000  1.916291    0.000000  1.510826  0.000000  0.000000  1.916291

        is     nadal   of       one   players    sachin      team    tennis  \
Doc 1  1.0  0.000000  1.0  1.223144  1.223144  1.916291  0.000000  0.000000
Doc 2  1.0  0.000000  1.0  1.223144  1.223144  0.000000  0.000000  1.510826
Doc 3  1.0  1.916291  1.0  1.223144  1.223144  0.000000  0.000000  1.510826
Doc 4  1.0  0.000000  1.0  0.000000  0.000000  0.000000  1.916291  0.000000

       the        to     virat
Doc 1  1.0  1.916291  0.000000
Doc 2  1.0  0.000000  0.000000
Doc 3  1.0  0.000000  0.000000
Doc 4  2.0  0.000000  1.916291
```

```
csim = cosine_similarity(tfidfMat,tfidfMat)
```

```
csimDf = pd.DataFrame(csim,index=sorted(docList),columns=sorted(docList))
```

```
print(csimDf)
```

```
          Doc 1     Doc 2     Doc 3     Doc 4
Doc 1  1.000000  0.492416  0.492416  0.277687
Doc 2  0.492416  1.000000  0.754190  0.215926
Doc 3  0.492416  0.754190  1.000000  0.215926
Doc 4  0.277687  0.215926  0.215926  1.000000
```

```
import seaborn as sns
import pandas as pd
titanic = sns.load_dataset("titanic")
```

```
titanic
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	True
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	True
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
886	0	2	male	27.0	0	0	13.0000	S	Second	man	True	NaN	Southampton	no	True
887	1	1	female	19.0	0	0	30.0000	S	First	woman	False	B	Southampton	yes	True
888	0	3	female	NaN	1	2	23.4500	S	Third	woman	False	NaN	Southampton	no	False
889	1	1	male	26.0	0	0	30.0000	C	First	man	True	C	Cherbourg	yes	True
890	0	3	male	32.0	0	0	7.7500	Q	Third	man	True	NaN	Queenstown	no	True

891 rows × 15 columns

```
titanic.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   survived     891 non-null    int64
 1   pclass       891 non-null    int64
 2   sex          891 non-null    object
 3   age          714 non-null    float64
 4   sibsp        891 non-null    int64
 5   parch        891 non-null    int64
 6   fare         891 non-null    float64
 7   embarked     889 non-null    object
 8   class        891 non-null    category
 9   who          891 non-null    object
 10  adult_male   891 non-null    bool
 11  deck         203 non-null    category
 12  embark_town  889 non-null    object
 13  alive        891 non-null    object
 14  alone        891 non-null    bool
dtypes: bool(2), category(2), float64(2), int64(4), object(5)
memory usage: 80.7+ KB
```

```
x=titanic["fare"]
```

```
x
```

```
0       7.2500
1      71.2833
2       7.9250
3      53.1000
4       8.0500
        ...
886    13.0000
887    30.0000
888    23.4500
889    30.0000
890     7.7500
Name: fare, Length: 891, dtype: float64
```

```
titanic.describe()
```

	survived	pclass	age	sibsp	parch	fare
count	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

```
titanic.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype   
---  ------       --------------  -----   
 0   survived     891 non-null    int64   
 1   pclass       891 non-null    int64   
 2   sex          891 non-null    object  
 3   age          714 non-null    float64 
 4   sibsp        891 non-null    int64   
 5   parch        891 non-null    int64   
 6   fare         891 non-null    float64 
 7   embarked     889 non-null    object  
 8   class        891 non-null    category
 9   who          891 non-null    object  
 10  adult_male   891 non-null    bool    
 11  deck         203 non-null    category
 12  embark_town  889 non-null    object  
 13  alive        891 non-null    object  
 14  alone        891 non-null    bool    
dtypes: bool(2), category(2), float64(2), int64(4), object(5)
memory usage: 80.7+ KB
```

```
titanic_cleaned=titanic.drop(['pclass','embarked','deck','embark_town'],axis=1)
titanic_cleaned.head(15)
```

	survived	sex	age	sibsp	parch	fare	class	who	adult_male	alive	alone
0	0	male	22.0	1	0	7.2500	Third	man	True	no	False
1	1	female	38.0	1	0	71.2833	First	woman	False	yes	False
2	1	female	26.0	0	0	7.9250	Third	woman	False	yes	True
3	1	female	35.0	1	0	53.1000	First	woman	False	yes	False
4	0	male	35.0	0	0	8.0500	Third	man	True	no	True
5	0	male	NaN	0	0	8.4583	Third	man	True	no	True
6	0	male	54.0	0	0	51.8625	First	man	True	no	True
7	0	male	2.0	3	1	21.0750	Third	child	False	no	False
8	1	female	27.0	0	2	11.1333	Third	woman	False	yes	False
9	1	female	14.0	1	0	30.0708	Second	child	False	yes	False
10	1	female	4.0	1	1	16.7000	Third	child	False	yes	False
11	1	female	58.0	0	0	26.5500	First	woman	False	yes	True
12	0	male	20.0	0	0	8.0500	Third	man	True	no	True
13	0	male	39.0	1	5	31.2750	Third	man	True	no	False
14	0	female	14.0	0	0	7.8542	Third	child	False	no	True

```
titanic_cleaned.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 11 columns):
 #   Column      Non-Null Count  Dtype   
---  ------      --------------  -----   
 0   survived    891 non-null    int64   
 1   sex         891 non-null    object  
 2   age         714 non-null    float64 
 3   sibsp       891 non-null    int64   
```

```
 4   parch       891 non-null    int64
 5   fare        891 non-null    float64
 6   class       891 non-null    category
 7   who         891 non-null    object
 8   adult_male  891 non-null    bool
 9   alive       891 non-null    object
 10  alone       891 non-null    bool
dtypes: bool(2), category(1), float64(2), int64(3), object(3)
memory usage: 58.6+ KB
```

```
titanic_cleaned.isnull().sum()
```

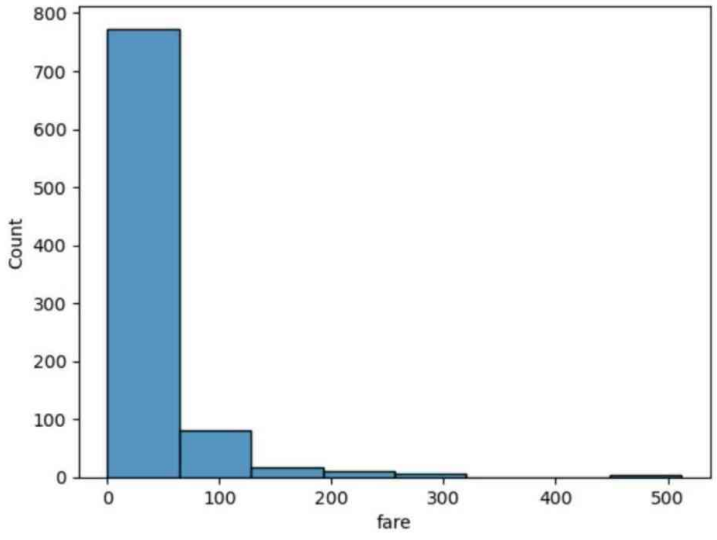
```
survived        0
sex             0
age           177
sibsp           0
parch           0
fare            0
class           0
who             0
adult_male      0
alive           0
alone           0
dtype: int64
```

```
titanic_cleaned.corr(method='pearson')
```

	survived	age	sibsp	parch	fare	adult_male	alone
survived	1.000000	-0.077221	-0.035322	0.081629	0.257307	-0.557080	-0.203367
age	-0.077221	1.000000	-0.308247	-0.189119	0.096067	0.280328	0.198270
sibsp	-0.035322	-0.308247	1.000000	0.414838	0.159651	-0.253586	-0.584471
parch	0.081629	-0.189119	0.414838	1.000000	0.216225	-0.349943	-0.583398
fare	0.257307	0.096067	0.159651	0.216225	1.000000	-0.182024	-0.271832
adult_male	-0.557080	0.280328	-0.253586	-0.349943	-0.182024	1.000000	0.404744
alone	-0.203367	0.198270	-0.584471	-0.583398	-0.271832	0.404744	1.000000

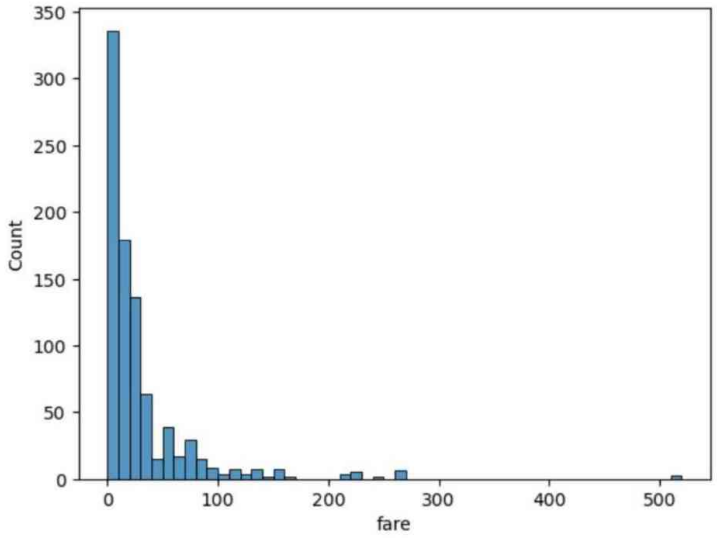
```
sns.histplot(data=titanic,x="fare",bins=8)
```

```
<AxesSubplot:xlabel='fare', ylabel='Count'>
```



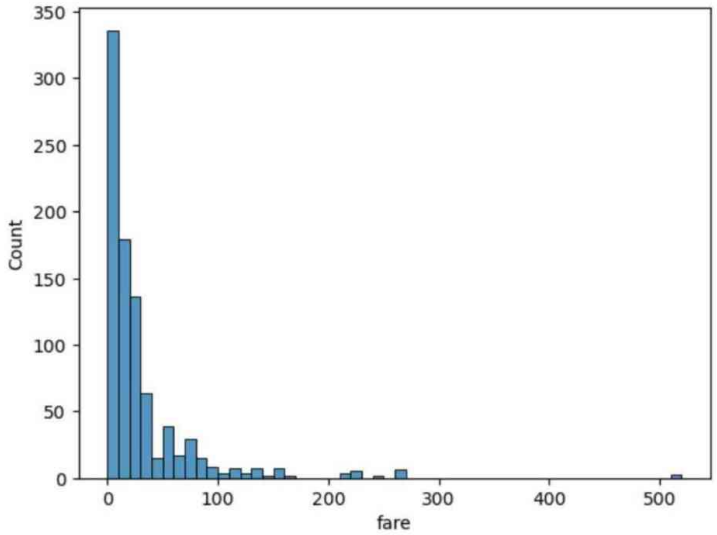

```
sns.histplot(data=titanic,x="fare",binwidth=10)
```

```
<AxesSubplot:xlabel='fare', ylabel='Count'>
```



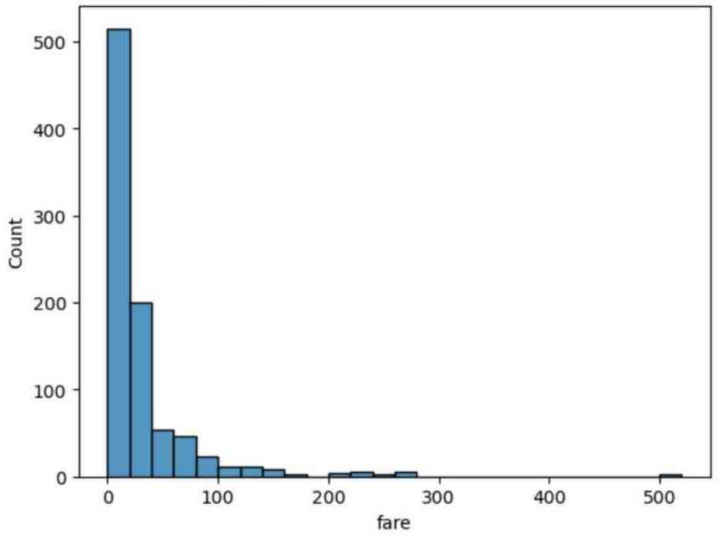

```
sns.histplot(data=titanic,x="fare",bins=20,binwidth=10)
```

```
<AxesSubplot:xlabel='fare', ylabel='Count'>
```



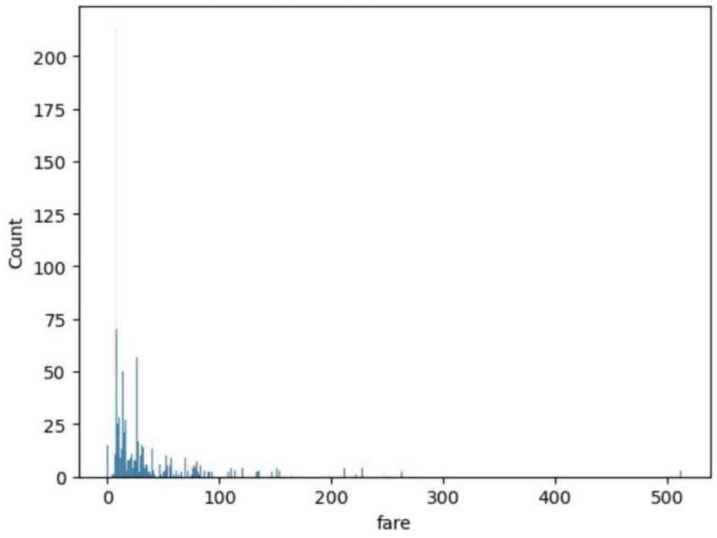

```
sns.histplot(data=titanic,x="fare",binwidth=20)
```

```
<AxesSubplot:xlabel='fare', ylabel='Count'>
```



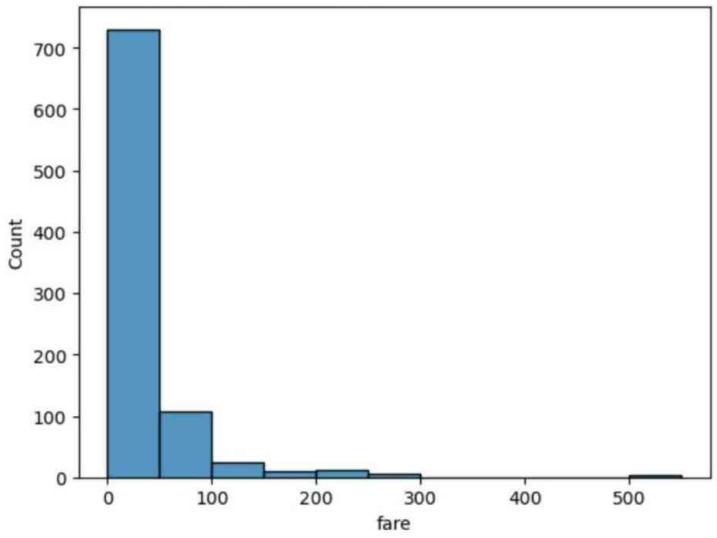

```
sns.histplot(data=titanic,x="fare",binwidth=1)
```

```
<AxesSubplot:xlabel='fare', ylabel='Count'>
```




```
sns.histplot(data=titanic,x="fare",bins=20,binwidth=50)
```

```
<AxesSubplot:xlabel='fare', ylabel='Count'>
```




```
Start coding or generate with AI.
```

```
import seaborn as sns
titanic = sns.load_dataset("titanic")
```

```
titanic
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	True
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	True
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
886	0	2	male	27.0	0	0	13.0000	S	Second	man	True	NaN	Southampton	no	True
887	1	1	female	19.0	0	0	30.0000	S	First	woman	False	B	Southampton	yes	True
888	0	3	female	NaN	1	2	23.4500	S	Third	woman	False	NaN	Southampton	no	False
889	1	1	male	26.0	0	0	30.0000	C	First	man	True	C	Cherbourg	yes	True
890	0	3	male	32.0	0	0	7.7500	Q	Third	man	True	NaN	Queenstown	no	True

891 rows × 15 columns

```
titanic.head(10)
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	True
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	True
5	0	3	male	NaN	0	0	8.4583	Q	Third	man	True	NaN	Queenstown	no	True
6	0	1	male	54.0	0	0	51.8625	S	First	man	True	E	Southampton	no	True
7	0	3	male	2.0	3	1	21.0750	S	Third	child	False	NaN	Southampton	no	False
8	1	3	female	27.0	0	2	11.1333	S	Third	woman	False	NaN	Southampton	yes	False
9	1	2	female	14.0	1	0	30.0708	C	Second	child	False	NaN	Cherbourg	yes	False

```
titanic.info
```

```
<bound method DataFrame.info of      survived  pclass     sex   age  sibsp  parch     fare embarked   class  \
0           0       3    male  22.0      1      0   7.2500        S   Third
1           1       1  female  38.0      1      0  71.2833        C   First
2           1       3  female  26.0      0      0   7.9250        S   Third
3           1       1  female  35.0      1      0  53.1000        S   First
4           0       3    male  35.0      0      0   8.0500        S   Third
..        ...     ...     ...   ...    ...    ...      ...      ...     ...
886         0       2    male  27.0      0      0  13.0000        S  Second
887         1       1  female  19.0      0      0  30.0000        S   First
888         0       3  female   NaN      1      2  23.4500        S   Third
889         1       1    male  26.0      0      0  30.0000        C   First
890         0       3    male  32.0      0      0   7.7500        Q   Third

       who  adult_male deck  embark_town alive  alone
0      man        True  NaN  Southampton    no  False
1    woman       False    C    Cherbourg   yes  False
2    woman       False  NaN  Southampton   yes   True
3    woman       False    C  Southampton   yes  False
4      man        True  NaN  Southampton    no   True
..     ...         ...  ...          ...   ...    ...
886    man        True  NaN  Southampton    no   True
887  woman       False    B  Southampton   yes   True
888  woman       False  NaN  Southampton    no  False
889    man        True    C    Cherbourg   yes   True
890    man        True  NaN   Queenstown    no   True

[891 rows x 15 columns]>
```

```
titanic.describe()
```

	survived	pclass	age	sibsp	parch	fare
count	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

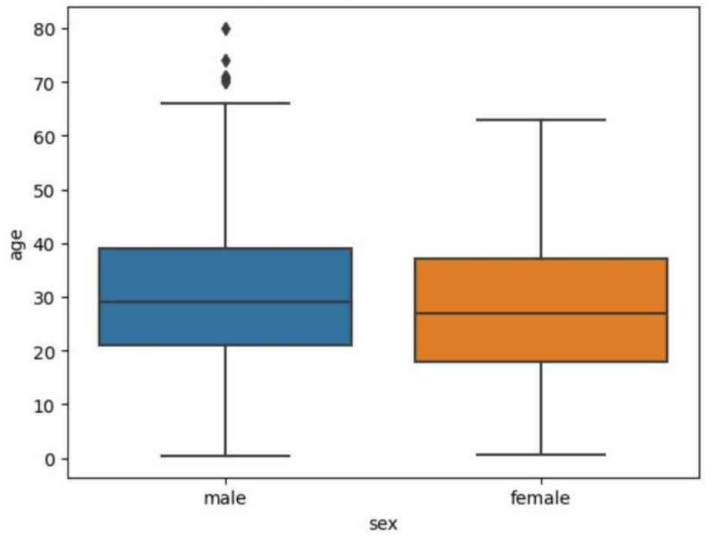
```
titanic.loc[:,["survived","alive"]]
```

	survived	alive
0	0	no
1	1	yes
2	1	yes
3	1	yes
4	0	no
...	...	...
886	0	no
887	1	yes
888	0	no
889	1	yes
890	0	no

891 rows × 2 columns

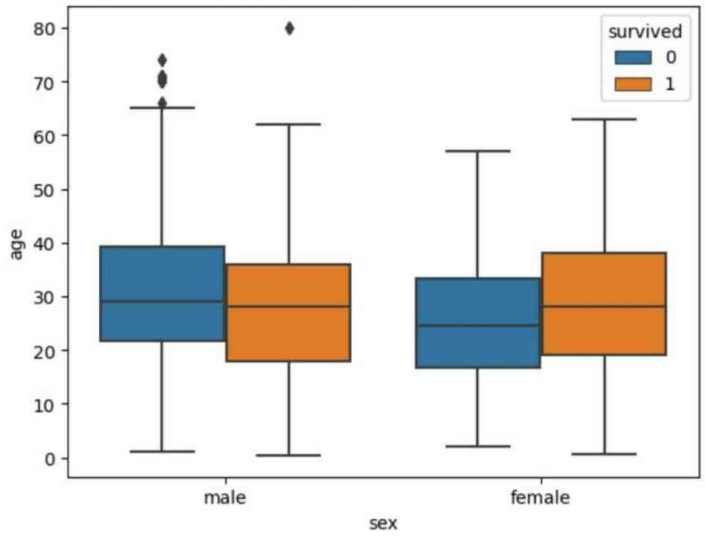
```
sns.boxplot(x="sex",y="age",data=titanic)
```

```
<AxesSubplot:xlabel='sex', ylabel='age'>
```




```
sns.boxplot(x="sex",y="age",data=titanic,hue="survived")
```

```
<AxesSubplot:xlabel='sex', ylabel='age'>
```




```
Start coding or generate with AI.
```

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

Reading the CSV file

```
df = pd.read_csv("https://raw.githubusercontent.com/shrikant-temburwar/Iris-Dataset/master/Iris.csv")
df.head()
```

```
df.describe()
```

```
df.shape
```

```
(150, 6)
```

```
df["Species"].unique()
```

```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)
```

```
df.groupby("Species").size()
```

```
Species
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
dtype: int64
```

List down the features and their types

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
Id               150 non-null int64
SepalLengthCm    150 non-null float64
SepalWidthCm     150 non-null float64
PetalLengthCm    150 non-null float64
PetalWidthCm     150 non-null float64
Species          150 non-null object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.1+ KB
```

Create a heatmap for each feature in the dataset.

```
corr = df.corr()
plt.subplots(figsize=(10,6))
sns.heatmap(corr, annot=True)
```

Create a box plot for each feature in the dataset

```
def graph(y):
 sns.boxplot(x="Species", y=y, data=df)

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

# Adding the subplot at the specified
# grid position
plt.subplot(221)
graph('SepalLengthCm')

plt.subplot(222)
graph('SepalWidthCm')

plt.subplot(223)
graph('PetalLengthCm')
```

```
plt.subplot(224)
graph('PetalWidthCm')

plt.show()
```

Compare distributions and identify outliers.

```
sns.boxplot(x='SepalWidthCm', data=df)
plt.show()
sns.boxplot(x='SepalLengthCm', data=df)
plt.show()
sns.boxplot(x='PetalWidthCm', data=df)
plt.show()
sns.boxplot(x='PetalLengthCm', data=df)
plt.show()
```

Start coding or generate with AI.

Start coding or generate with AI.

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

Reading the CSV file

```
df = pd.read_csv("https://raw.githubusercontent.com/shrikant-temburwar/Iris-Dataset/master/Iris.csv")
df.head()
```

```
df.describe()
```

```
df.shape
```

```
(150, 6)
```

```
df["Species"].unique()
```

```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)
```

```
df.groupby("Species").size()
```

```
Species
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
dtype: int64
```

List down the features and their types

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
Id               150 non-null int64
SepalLengthCm    150 non-null float64
SepalWidthCm     150 non-null float64
PetalLengthCm    150 non-null float64
PetalWidthCm     150 non-null float64
Species          150 non-null object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.1+ KB
```

Create a heatmap for each feature in the dataset.

```
corr = df.corr()
plt.subplots(figsize=(10,6))
sns.heatmap(corr, annot=True)
```

Create a box plot for each feature in the dataset

```
def graph(y):
 sns.boxplot(x="Species", y=y, data=df)

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

# Adding the subplot at the specified
# grid position
plt.subplot(221)
graph('SepalLengthCm')

plt.subplot(222)
graph('SepalWidthCm')

plt.subplot(223)
graph('PetalLengthCm')
```

```
plt.subplot(224)
graph('PetalWidthCm')

plt.show()
```

Compare distributions and identify outliers.

```
sns.boxplot(x='SepalWidthCm', data=df)
plt.show()
sns.boxplot(x='SepalLengthCm', data=df)
plt.show()
sns.boxplot(x='PetalWidthCm', data=df)
plt.show()
sns.boxplot(x='PetalLengthCm', data=df)
plt.show()
```

Start coding or generate with AI.

Start coding or generate with AI.