DAV_Python

April 24, 2024

1 Simple Linear Regression

[1]:
```python
# Essential imports
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.model_selection import train_test_split
import pandas as pd
import seaborn as sns
```

[2]:
```python
# Iris dataset
iris = sns.load_dataset('iris')
iris
```

[2]:
```
     sepal_length  sepal_width  petal_length  petal_width    species
0             5.1          3.5           1.4          0.2     setosa
1             4.9          3.0           1.4          0.2     setosa
2             4.7          3.2           1.3          0.2     setosa
3             4.6          3.1           1.5          0.2     setosa
4             5.0          3.6           1.4          0.2     setosa
..              …            …             …            …          …
145           6.7          3.0           5.2          2.3  virginica
146           6.3          2.5           5.0          1.9  virginica
147           6.5          3.0           5.2          2.0  virginica
148           6.2          3.4           5.4          2.3  virginica
149           5.9          3.0           5.1          1.8  virginica

[150 rows x 5 columns]
```

[3]:
```python
# Performing Simple LR on petal_length and petal_width
# Extract features
X = iris['petal_length']
Y = iris['petal_width']
X, Y
```

[3]:
```
(0      1.4
 1      1.4
 2      1.3
```

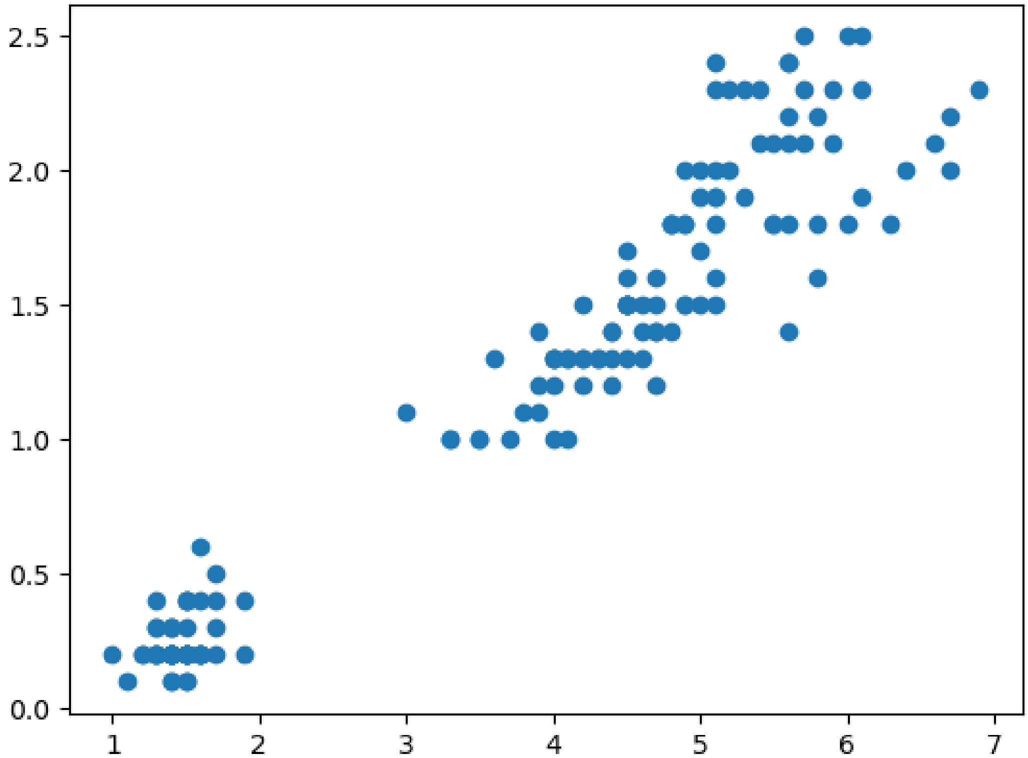
```
3      1.5
4      1.4
      …
145    5.2
146    5.0
147    5.2
148    5.4
149    5.1
Name: petal_length, Length: 150, dtype: float64,
0      0.2
1      0.2
2      0.2
3      0.2
4      0.2
      …
145    2.3
146    1.9
147    2.0
148    2.3
149    1.8
Name: petal_width, Length: 150, dtype: float64)
```

[4]:

```
# Visualise data
plt.scatter(X, Y)
```

[4]:

```
<matplotlib.collections.PathCollection at 0x1ef42191d90>
```



[5]:
```
# Split into training and testing sets
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, random_state=42,␣
↪test_size=0.1)
```

[6]:
```
# Print training set
X_train, Y_train
```

[6]:
```
(56     4.7
 104    5.8
 69     3.9
 55     4.5
 132    5.6
       …
 71     4.0
 106    4.5
 14     1.2
 92     4.0
 102    5.9
 Name: petal_length, Length: 135, dtype: float64,
 56     1.6
 104    2.2
 69     1.1
```

```
55     1.3
132    2.2
      …
71     1.3
106    1.7
14     0.2
92     1.2
102    2.1
Name: petal_width, Length: 135, dtype: float64)
```

[7]:
```
# Print testing set
X_test, Y_test
```

[7]:
```
(73     4.7
 18     1.7
 118    6.9
 78     4.5
 76     4.8
 31     1.5
 64     3.6
 141    5.1
 68     4.5
 82     3.9
 110    5.1
 12     1.4
 36     1.3
 9      1.5
 19     1.5
 Name: petal_length, dtype: float64,
 73     1.2
 18     0.3
 118    2.3
 78     1.5
 76     1.4
 31     0.4
 64     1.3
 141    2.3
 68     1.5
 82     1.2
 110    2.0
 12     0.1
 36     0.2
 9      0.1
 19     0.3
 Name: petal_width, dtype: float64)
```

[8]:
```python
# Import Linear Regression model
from sklearn.linear_model import LinearRegression
slr_model = LinearRegression()
```

[9]:
```python
# Reshape inputs
X_train, X_test = np.array(X_train).reshape(-1, 1), np.array(X_test).
 ↪reshape(-1, 1)

# Train model
slr_model.fit(X_train, Y_train)
```

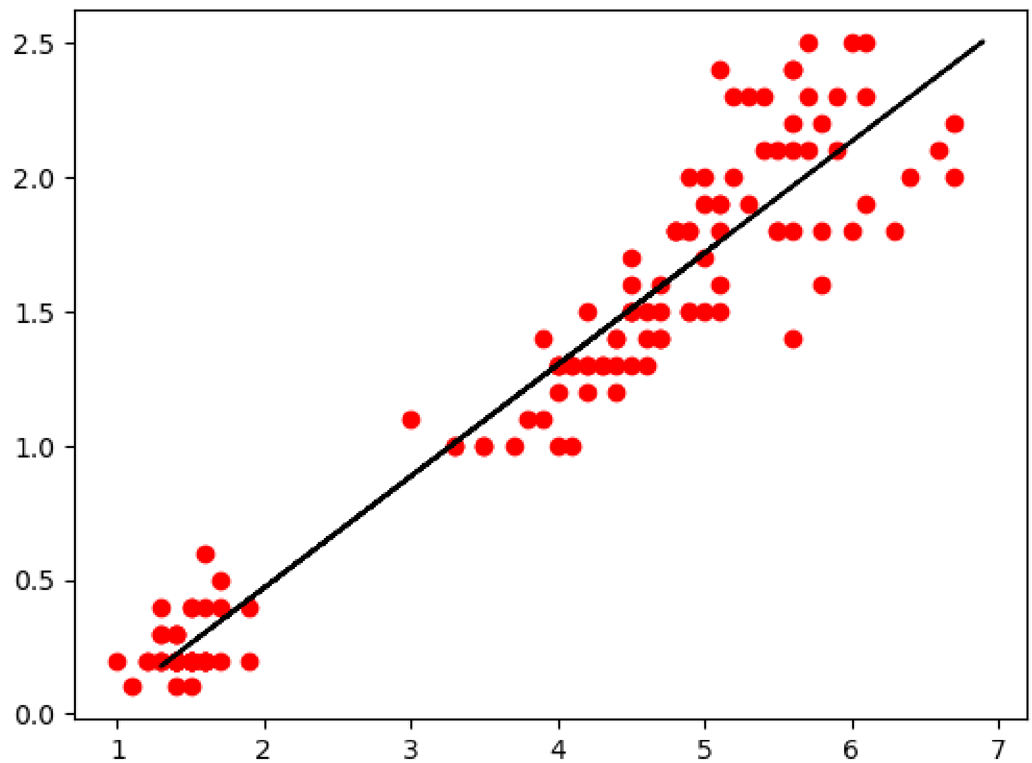
[9]:
```
LinearRegression()
```

[10]:
```python
# Make predictions
Y_pred = slr_model.predict(X_test)
```

[11]:
```python
# Evaluate model
from sklearn.metrics import mean_squared_error, mean_absolute_error

print(f"Mean Squared Error: {mean_squared_error(Y_test, Y_pred)}")
print(f"Mean Absolute Error: {mean_absolute_error(Y_test, Y_pred)}")
```

```
Mean Squared Error: 0.046489206286248065
Mean Absolute Error: 0.15873640858391244
```

[12]:
```python
# Final predictions
plt.scatter(X_train, Y_train, color='red')
plt.plot(X_test, Y_pred, color='black')
plt.show();
```



2 Multiple Linear Regression

[13]:
```
# Essential imports
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.model_selection import train_test_split
import pandas as pd
import seaborn as sns
```

[14]:
```
# Iris dataset
iris = sns.load_dataset('iris')
iris
```

[14]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
..	…	…	…	…	…

```
145           6.7          3.0           5.2          2.3  virginica
146           6.3          2.5           5.0          1.9  virginica
147           6.5          3.0           5.2          2.0  virginica
148           6.2          3.4           5.4          2.3  virginica
149           5.9          3.0           5.1          1.8  virginica

[150 rows x 5 columns]
```

[15]:
```python
# For example, predict petal_width using sepal_length, sepal_width and␣
 ↪petal_length

# Drop species
iris.drop('species', axis=1, inplace=True)
```

[16]:
```python
# Set features
X = iris[['sepal_length', 'sepal_width', 'petal_length']]
Y = iris['petal_width']
X, Y
```

[16]:
```
(     sepal_length  sepal_width  petal_length
 0             5.1          3.5           1.4
 1             4.9          3.0           1.4
 2             4.7          3.2           1.3
 3             4.6          3.1           1.5
 4             5.0          3.6           1.4
 ..            …           …           …
 145           6.7          3.0           5.2
 146           6.3          2.5           5.0
 147           6.5          3.0           5.2
 148           6.2          3.4           5.4
 149           5.9          3.0           5.1

 [150 rows x 3 columns],
 0      0.2
 1      0.2
 2      0.2
 3      0.2
 4      0.2
       …
 145    2.3
 146    1.9
 147    2.0
 148    2.3
 149    1.8
 Name: petal_width, Length: 150, dtype: float64)
```

```
[17]: # Training and testing splits
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, random_state=42,␣
 ↪test_size=0.2)
```

```
[18]: # Training data
X_train, Y_train
```

```
[18]: (     sepal_length  sepal_width  petal_length
 22            4.6          3.6           1.0
 15            5.7          4.4           1.5
 65            6.7          3.1           4.4
 11            4.8          3.4           1.6
 42            4.4          3.2           1.3
 ..            …          …           …
 71            6.1          2.8           4.0
 106           4.9          2.5           4.5
 14            5.8          4.0           1.2
 92            5.8          2.6           4.0
 102           7.1          3.0           5.9

 [120 rows x 3 columns],
 22     0.2
 15     0.4
 65     1.4
 11     0.2
 42     0.2
       …
 71     1.3
 106    1.7
 14     0.2
 92     1.2
 102    2.1
 Name: petal_width, Length: 120, dtype: float64)
```

```
[19]: # Testing data
X_test, Y_test
```

```
[19]: (     sepal_length  sepal_width  petal_length
 73            6.1          2.8           4.7
 18            5.7          3.8           1.7
 118           7.7          2.6           6.9
 78            6.0          2.9           4.5
 76            6.8          2.8           4.8
 31            5.4          3.4           1.5
 64            5.6          2.9           3.6
 141           6.9          3.1           5.1
 68            6.2          2.2           4.5
```

```
82          5.8         2.7         3.9
110         6.5         3.2         5.1
12          4.8         3.0         1.4
36          5.5         3.5         1.3
9           4.9         3.1         1.5
19          5.1         3.8         1.5
56          6.3         3.3         4.7
104         6.5         3.0         5.8
69          5.6         2.5         3.9
55          5.7         2.8         4.5
132         6.4         2.8         5.6
29          4.7         3.2         1.6
127         6.1         3.0         4.9
26          5.0         3.4         1.6
128         6.4         2.8         5.6
131         7.9         3.8         6.4
145         6.7         3.0         5.2
108         6.7         2.5         5.8
143         6.8         3.2         5.9
45          4.8         3.0         1.4
30          4.8         3.1         1.6,
73      1.2
18      0.3
118     2.3
78      1.5
76      1.4
31      0.4
64      1.3
141     2.3
68      1.5
82      1.2
110     2.0
12      0.1
36      0.2
9       0.1
19      0.3
56      1.6
104     2.2
69      1.1
55      1.3
132     2.2
29      0.2
127     1.8
26      0.4
128     2.1
131     2.0
145     2.3
```

```
108    1.8
143    2.3
45     0.3
30     0.2
Name: petal_width, dtype: float64)
```

[20]:
```
# Import Linear Regression model
from sklearn.linear_model import LinearRegression

mlr = LinearRegression()
```

[21]:
```
# Train model
mlr.fit(X_train, Y_train)
```

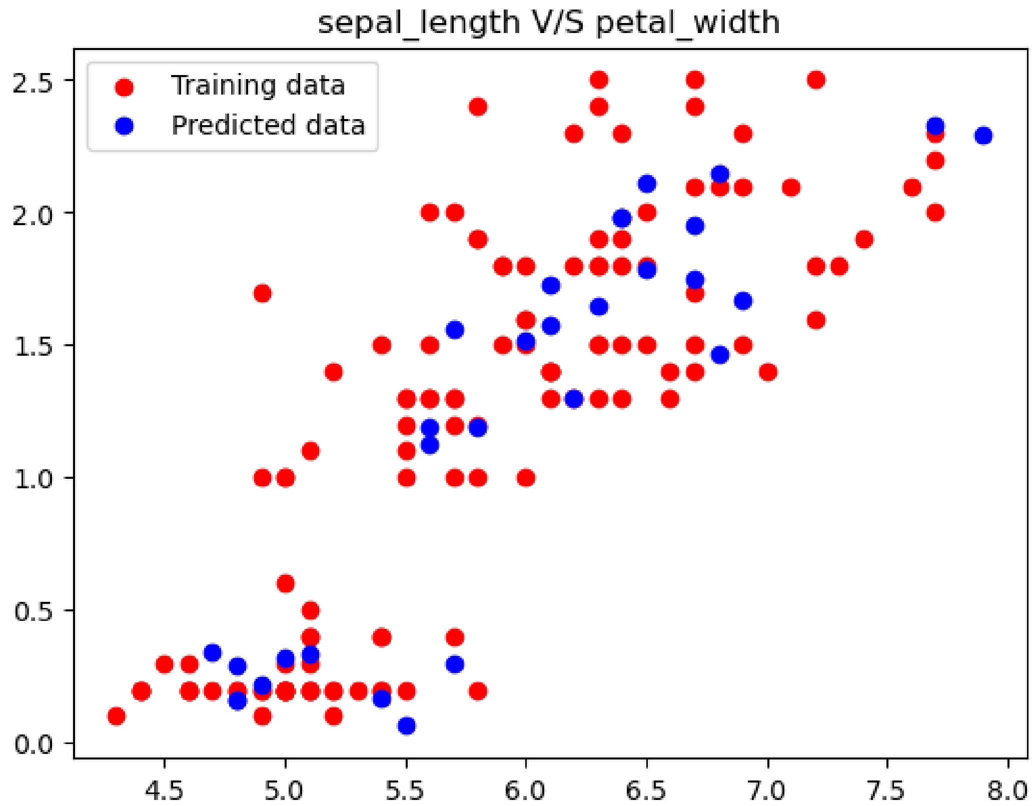
[21]:
```
LinearRegression()
```

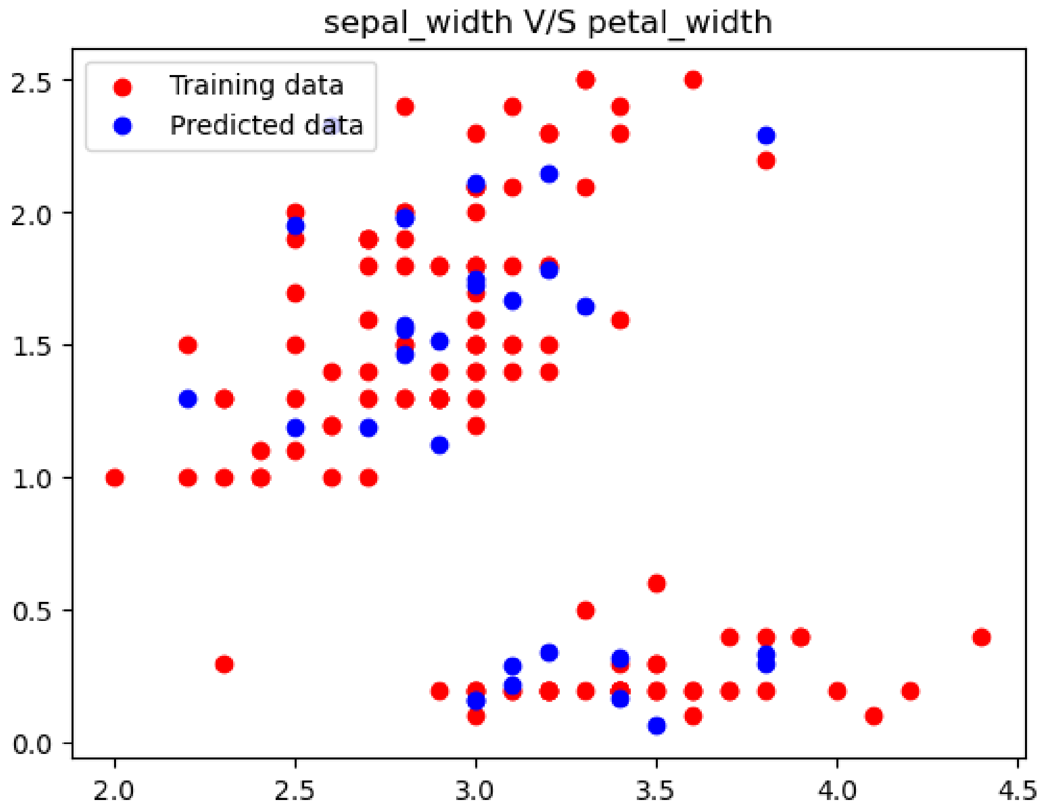
[22]:
```
# Make predictions
Y_pred = mlr.predict(X_test)
```

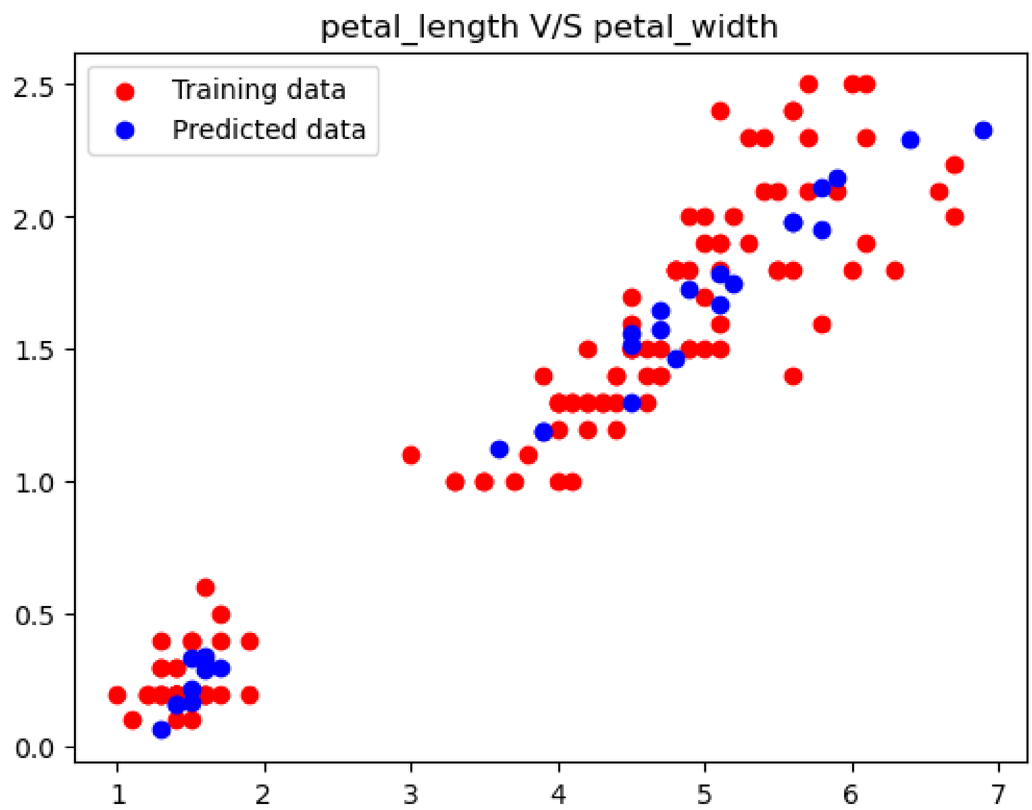
[23]:
```
# Evaluate model
from sklearn.metrics import mean_squared_error, mean_absolute_error

print(f"Mean Squared Error: {mean_squared_error(Y_test, Y_pred)}")
print(f"Mean Absolute Error: {mean_absolute_error(Y_test, Y_pred)}")
```

```
Mean Squared Error: 0.046332603325764436
Mean Absolute Error: 0.15905558437496845
```

[24]:
```
# Visualisations
for input_feature in X.columns:
  plt.scatter(X_train[input_feature], Y_train, color='red', label="Training␣
↪data")
  plt.scatter(X_test[input_feature], Y_pred, color='blue', label="Predicted␣
↪data")
  plt.title(f"{input_feature} V/S {Y.name}")
  plt.legend(loc="upper left")
  plt.show();
```


sepal_length V/S petal_width
Training data
Predicted data
2.5
2.0
1.5
1.0
0.5
0.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0


sepal_width V/S petal_width
Training data
Predicted data
2.5
2.0
1.5
1.0
0.5
0.0
2.0
2.5
3.0
3.5
4.0
4.5



3 Logistic Regression

[25]:
```
# Essential imports
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.model_selection import train_test_split
import pandas as pd
import seaborn as sns
```

[26]:
```
# Penguins dataset
penguins = sns.load_dataset('penguins')
penguins
```

[26]:
```
    species     island  bill_length_mm  bill_depth_mm  flipper_length_mm  \
0    Adelie  Torgersen            39.1           18.7              181.0
1    Adelie  Torgersen            39.5           17.4              186.0
2    Adelie  Torgersen            40.3           18.0              195.0
3    Adelie  Torgersen             NaN            NaN                NaN
4    Adelie  Torgersen            36.7           19.3              193.0
```

```
..      …         …               …             …              …
339  Gentoo    Biscoe             NaN           NaN              NaN
340  Gentoo    Biscoe            46.8          14.3            215.0
341  Gentoo    Biscoe            50.4          15.7            222.0
342  Gentoo    Biscoe            45.2          14.8            212.0
343  Gentoo    Biscoe            49.9          16.1            213.0

     body_mass_g     sex
0         3750.0    Male
1         3800.0  Female
2         3250.0  Female
3            NaN     NaN
4         3450.0  Female
..           …       …
339          NaN     NaN
340       4850.0  Female
341       5750.0    Male
342       5200.0  Female
343       5400.0    Male

[344 rows x 7 columns]
```

[27]:
```python
# Remove NaN values
penguins.dropna(inplace=True)
penguins
```

[27]:
```
    species     island  bill_length_mm  bill_depth_mm  flipper_length_mm  \
0    Adelie  Torgersen            39.1           18.7              181.0
1    Adelie  Torgersen            39.5           17.4              186.0
2    Adelie  Torgersen            40.3           18.0              195.0
4    Adelie  Torgersen            36.7           19.3              193.0
5    Adelie  Torgersen            39.3           20.6              190.0
..      …         …               …             …              …
338  Gentoo    Biscoe            47.2           13.7              214.0
340  Gentoo    Biscoe            46.8           14.3              215.0
341  Gentoo    Biscoe            50.4           15.7              222.0
342  Gentoo    Biscoe            45.2           14.8              212.0
343  Gentoo    Biscoe            49.9           16.1              213.0

     body_mass_g     sex
0         3750.0    Male
1         3800.0  Female
2         3250.0  Female
4         3450.0  Female
5         3650.0    Male
..           …       …
338       4925.0  Female
```

```
340        4850.0  Female
341        5750.0    Male
342        5200.0  Female
343        5400.0    Male

[333 rows x 7 columns]
```

[28]:
```
# Predict sex using bill_length_mm, bill_depth_mm, flipper_length_mm and␣
↪body_mass_g

# Encode sex
penguins.replace("Male", 0, inplace=True)
penguins.replace("Female", 1, inplace=True)
penguins
```

[28]:
```
    species     island  bill_length_mm  bill_depth_mm  flipper_length_mm  \
0    Adelie  Torgersen            39.1           18.7              181.0
1    Adelie  Torgersen            39.5           17.4              186.0
2    Adelie  Torgersen            40.3           18.0              195.0
4    Adelie  Torgersen            36.7           19.3              193.0
5    Adelie  Torgersen            39.3           20.6              190.0
..      …         …             …             …                  …
338  Gentoo     Biscoe            47.2           13.7              214.0
340  Gentoo     Biscoe            46.8           14.3              215.0
341  Gentoo     Biscoe            50.4           15.7              222.0
342  Gentoo     Biscoe            45.2           14.8              212.0
343  Gentoo     Biscoe            49.9           16.1              213.0

     body_mass_g  sex
0         3750.0    0
1         3800.0    1
2         3250.0    1
4         3450.0    1
5         3650.0    0
..           …   …
338       4925.0    1
340       4850.0    1
341       5750.0    0
342       5200.0    1
343       5400.0    0

[333 rows x 7 columns]
```

[29]:
```
# Features
X = penguins[['bill_length_mm', 'bill_depth_mm', 'flipper_length_mm',␣
↪'body_mass_g']]
Y = penguins['sex']
```

```
X, Y
```

[29]:
```
(     bill_length_mm  bill_depth_mm  flipper_length_mm  body_mass_g
 0              39.1           18.7              181.0       3750.0
 1              39.5           17.4              186.0       3800.0
 2              40.3           18.0              195.0       3250.0
 4              36.7           19.3              193.0       3450.0
 5              39.3           20.6              190.0       3650.0
 ..              …              …                  …           …
 338            47.2           13.7              214.0       4925.0
 340            46.8           14.3              215.0       4850.0
 341            50.4           15.7              222.0       5750.0
 342            45.2           14.8              212.0       5200.0
 343            49.9           16.1              213.0       5400.0

 [333 rows x 4 columns],
 0      0
 1      1
 2      1
 4      1
 5      0
       ..
 338    1
 340    1
 341    0
 342    1
 343    0
 Name: sex, Length: 333, dtype: int64)
```

[30]:
```
# Training and testing split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, random_state=42,␣
 ↪test_size=0.1)
```

[31]:
```
# Training set
X_train, Y_train
```

[31]:
```
(     bill_length_mm  bill_depth_mm  flipper_length_mm  body_mass_g
 285            49.8           16.8              230.0       5700.0
 296            47.5           14.2              209.0       4600.0
 188            47.6           18.3              195.0       3850.0
 260            42.7           13.7              208.0       3950.0
 52             35.0           17.9              190.0       3450.0
 ..              …              …                  …           …
 194            50.9           19.1              196.0       3550.0
 77             37.2           19.4              184.0       3900.0
 112            39.7           17.7              193.0       3200.0
 277            45.5           15.0              220.0       5000.0
```

```
108              38.1           17.0              181.0        3175.0

[299 rows x 4 columns],
285    0
296    1
188    1
260    1
52     1
      ..
194    0
77     0
112    1
277    0
108    1
Name: sex, Length: 299, dtype: int64)
```

[32]:
```
# Testing set
X_test, Y_test
```

[32]:
```
(     bill_length_mm  bill_depth_mm  flipper_length_mm  body_mass_g
 30             39.5           16.7              178.0       3250.0
 317            46.9           14.6              222.0       4875.0
 79             42.1           19.1              195.0       4000.0
 201            49.8           17.3              198.0       3675.0
 63             41.1           18.2              192.0       4050.0
 304            44.9           13.8              212.0       4750.0
 289            50.7           15.0              223.0       5550.0
 186            49.7           18.6              195.0       3600.0
 217            49.6           18.2              193.0       3775.0
 203            51.4           19.0              201.0       3950.0
 81             42.9           17.6              196.0       4700.0
 14             34.6           21.1              198.0       4400.0
 328            43.3           14.0              208.0       4575.0
 132            36.8           18.5              193.0       3500.0
 272            45.1           14.4              210.0       4400.0
 138            37.0           16.5              185.0       3400.0
 120            36.2           17.2              187.0       3150.0
 152            46.5           17.9              192.0       3500.0
 82             36.7           18.8              187.0       3800.0
 282            45.7           13.9              214.0       4400.0
 115            42.7           18.3              196.0       4075.0
 143            40.7           17.0              190.0       3725.0
 323            49.1           15.0              228.0       5500.0
 205            50.7           19.7              203.0       4050.0
 6              38.9           17.8              181.0       3625.0
 116            38.6           17.0              188.0       2900.0
 268            44.9           13.3              213.0       5100.0
```

```
332           43.5          15.2              213.0      4650.0
169           58.0          17.8              181.0      3700.0
331           49.8          15.9              229.0      5950.0
174           43.2          16.6              187.0      2900.0
309           52.1          17.0              230.0      5550.0
69            41.8          19.4              198.0      4450.0
90            35.7          18.0              202.0      3550.0,
30     1
317    1
79     0
201    1
63     0
304    1
289    0
186    0
217    0
203    0
81     0
14     0
328    1
132    1
272    1
138    1
120    1
152    1
82     1
282    1
115    0
143    0
323    0
205    0
6      1
116    1
268    1
332    1
169    1
331    0
174    1
309    0
69     0
90     1
Name: sex, dtype: int64)
```

[33]:

```python
# Import Logistic Regression model
from sklearn.linear_model import LogisticRegression

lr_model = LogisticRegression()
```

[34]:
```python
# Train model
lr_model.fit(X_train, Y_train)
```

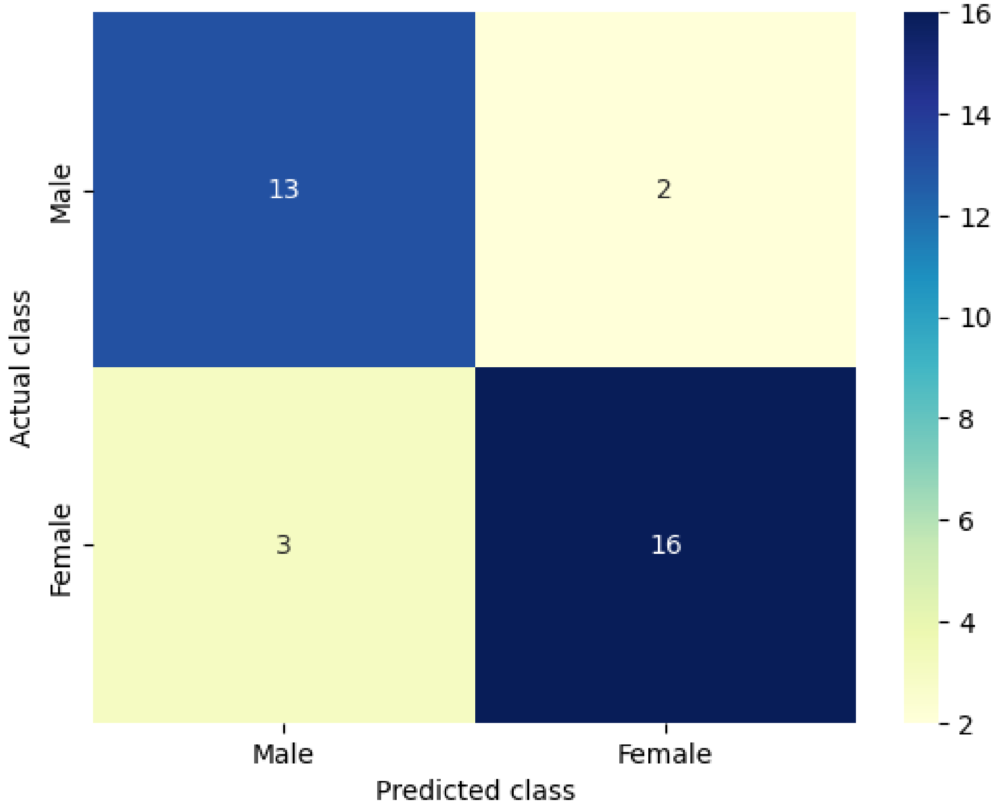
[34]:
```
LogisticRegression()
```

[35]:
```python
# Make predicitons
Y_pred = lr_model.predict(X_test)
```

[36]:
```python
# Evaluate model
from sklearn.metrics import confusion_matrix, classification_report, roc_curve,␣
↪roc_auc_score

# Confusion matrix
cnf_matrix = confusion_matrix(Y_test, Y_pred)
cnf_matrix
```

[36]:
```
array([[13,  2],
       [ 3, 16]], dtype=int64)
```

[37]:
```python
# Prettier confusion matrix (heatmap)
sns.heatmap(pd.DataFrame(cnf_matrix), annot=True, cmap="YlGnBu",␣
↪xticklabels=["Male", "Female"], yticklabels=["Male", "Female"])
plt.xlabel("Predicted class")
plt.ylabel("Actual class")
plt.show();
```



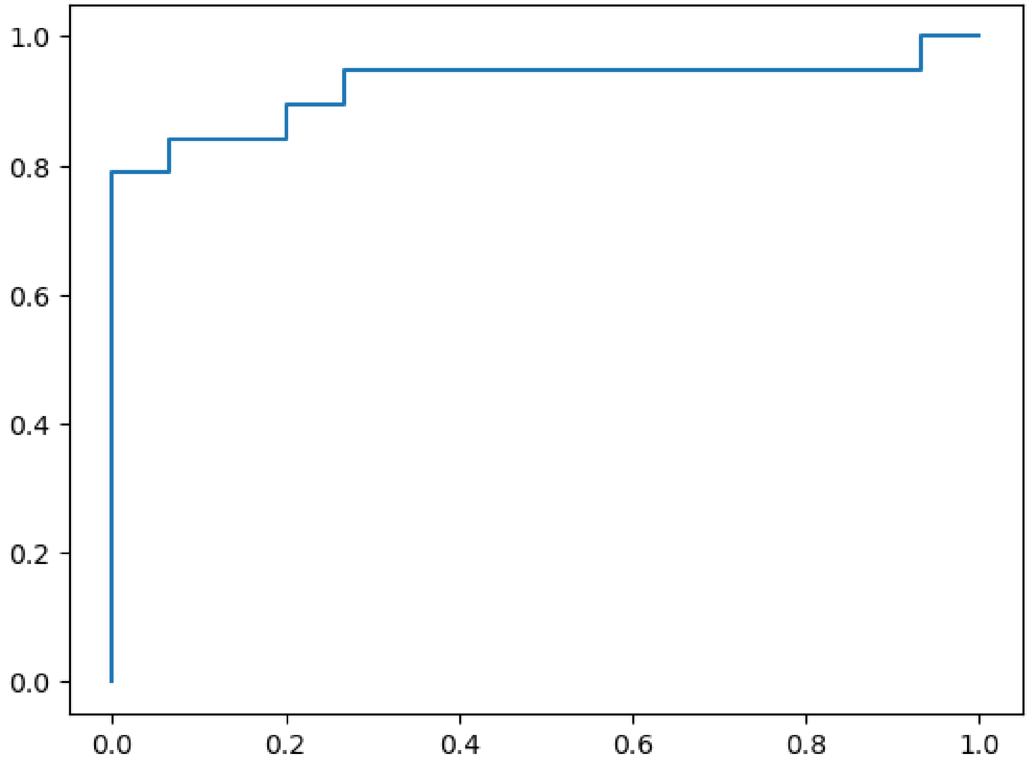
[38]:

```
# Classification report
print(classification_report(Y_test, Y_pred, target_names=["Male", "Female"]))
```

```
              precision    recall  f1-score   support

        Male       0.81      0.87      0.84        15
      Female       0.89      0.84      0.86        19

    accuracy                           0.85        34
   macro avg       0.85      0.85      0.85        34
weighted avg       0.86      0.85      0.85        34
```

[39]:

```
# ROC curve
Y_pred_proba = lr_model.predict_proba(X_test)[::, 1]
fpr, tpr, _ = roc_curve(Y_test, Y_pred_proba)
auc = roc_auc_score(Y_test, Y_pred_proba)
plt.plot(fpr, tpr, label=f"R2 score: {auc}")
plt.show();
```

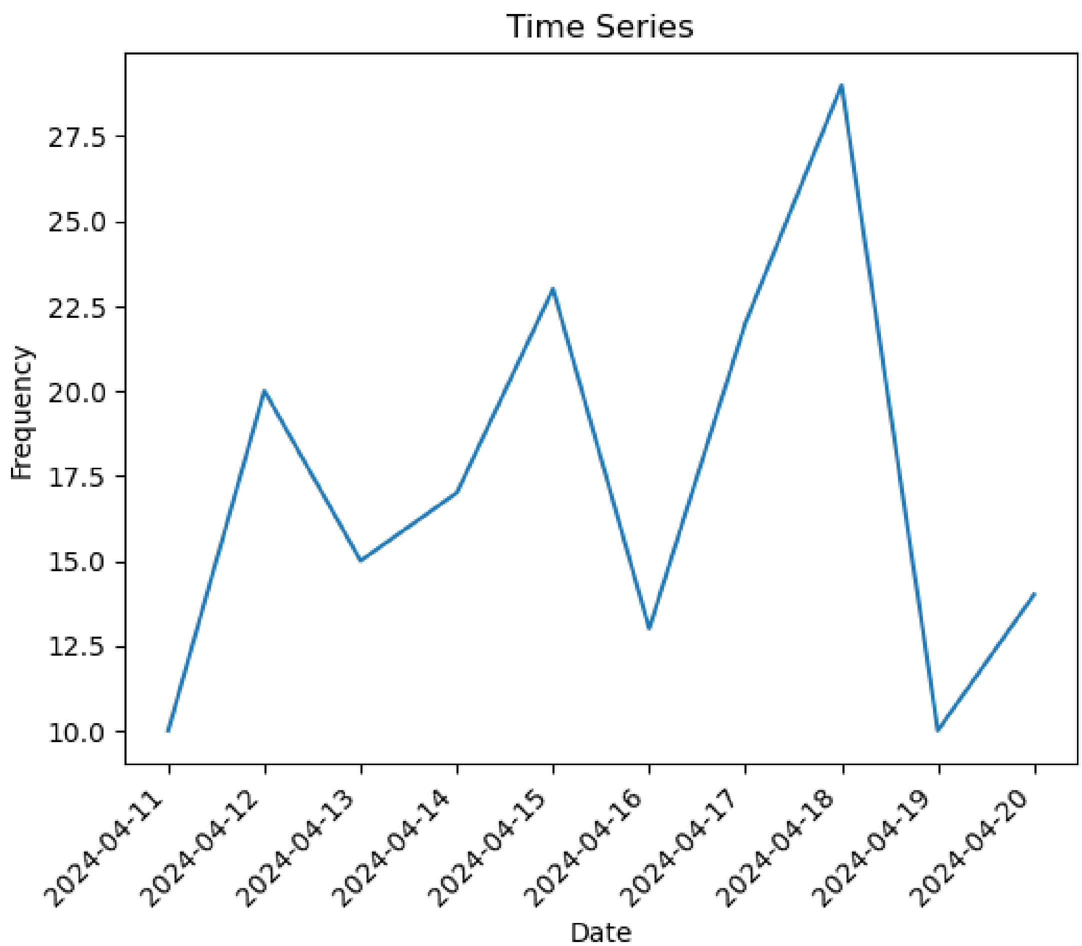


4 Time Series Analysis

- This example is generating dummy data

[40]:
```
# Essential imports
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

[41]:
```
import datetime

df = pd.DataFrame({'date': np.array([datetime.datetime(2024, 4, i + 1) for i in␣
 ↪range(10, 20)]),
                   'frequency': [10, 20, 15, 17, 23, 13, 22, 29, 10, 14]})

plt.plot(df.date, df.frequency)
plt.title("Time Series")
plt.xticks(rotation=45, ha="right")
plt.xlabel("Date")
plt.ylabel("Frequency")
plt.show();
```



5 Data Visualisation (Python)

- This example generates random data for plotting using NumPy
- Of course, you can choose to make your own small sample dataset (recommended)

[42]:
```
# Essetial libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

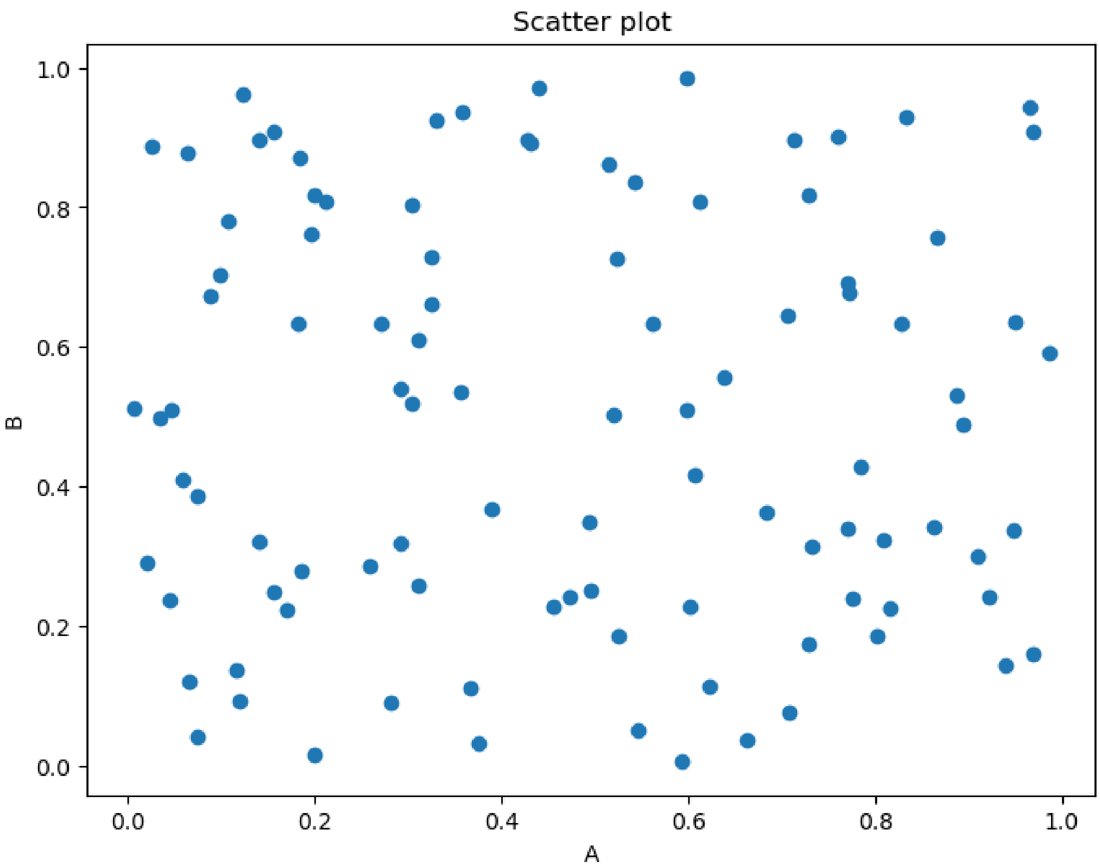
[43]:
```
# Generate a dummy dataset
np.random.seed(42) # Set random seed for reproducibility
data = pd.DataFrame({
    'A': np.random.rand(100),
    'B': np.random.rand(100),
    'C': np.random.rand(100),
    'Category': np.random.choice(['X', 'Y', 'Z'], 100)
```

```
})

# Display it
data.head()
```

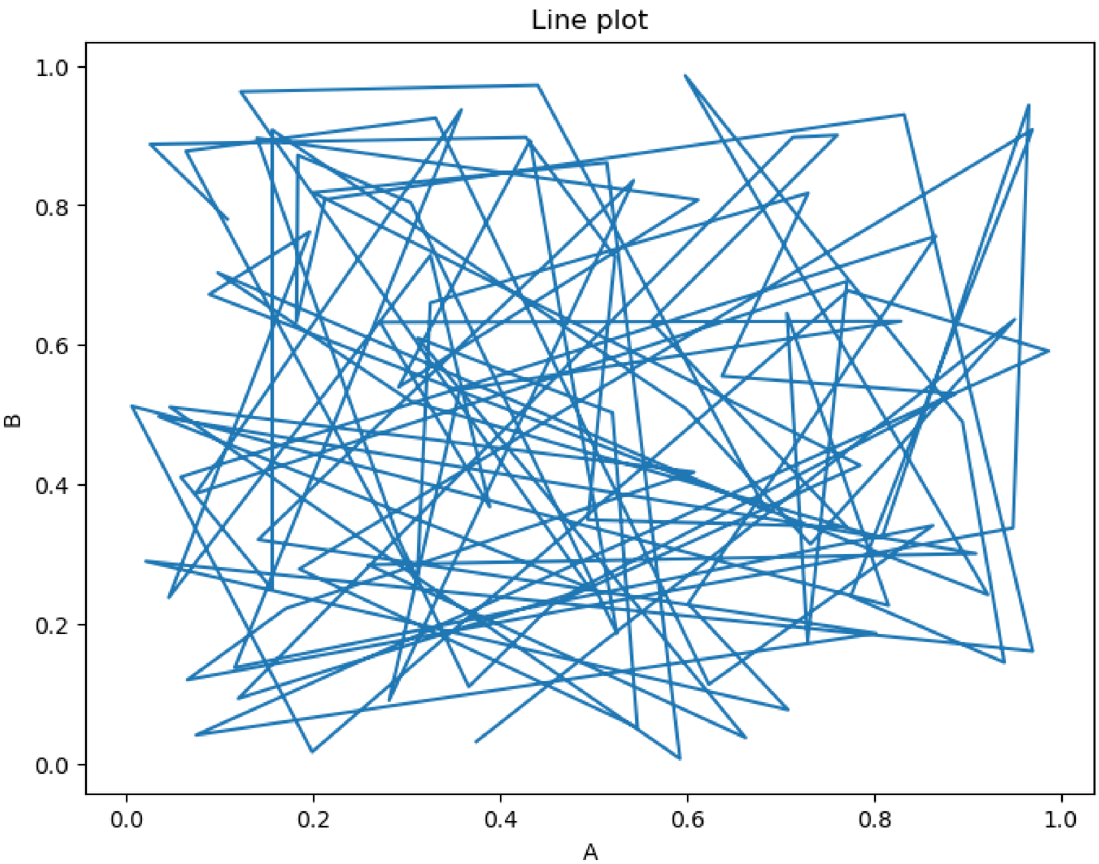
[43]:

	A	B	C	Category
0	0.374540	0.031429	0.642032	X
1	0.950714	0.636410	0.084140	Y
2	0.731994	0.314356	0.161629	Z
3	0.598658	0.508571	0.898554	Z
4	0.156019	0.907566	0.606429	Z

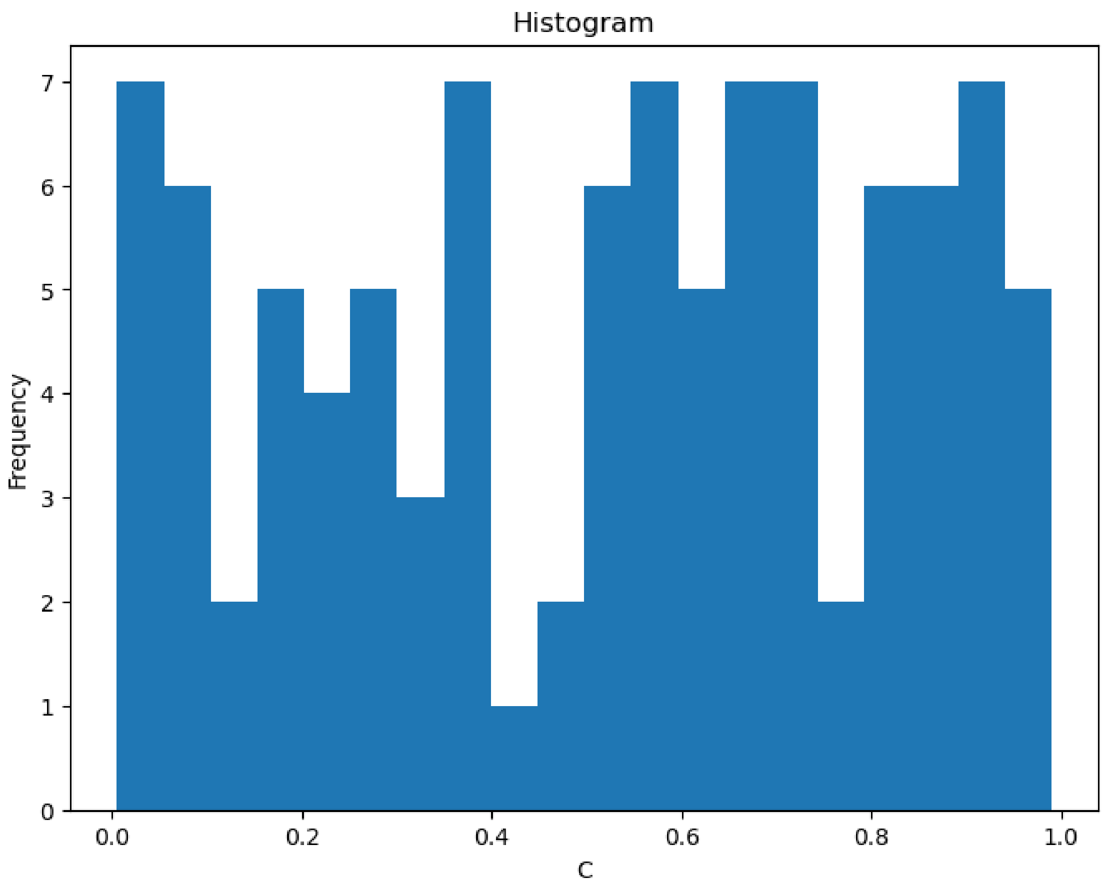
[44]:
```
# Scatter plot
plt.figure(figsize=(8, 6))
plt.scatter(data['A'], data['B'])
plt.title('Scatter plot')
plt.xlabel('A')
plt.ylabel('B')
plt.show();
```



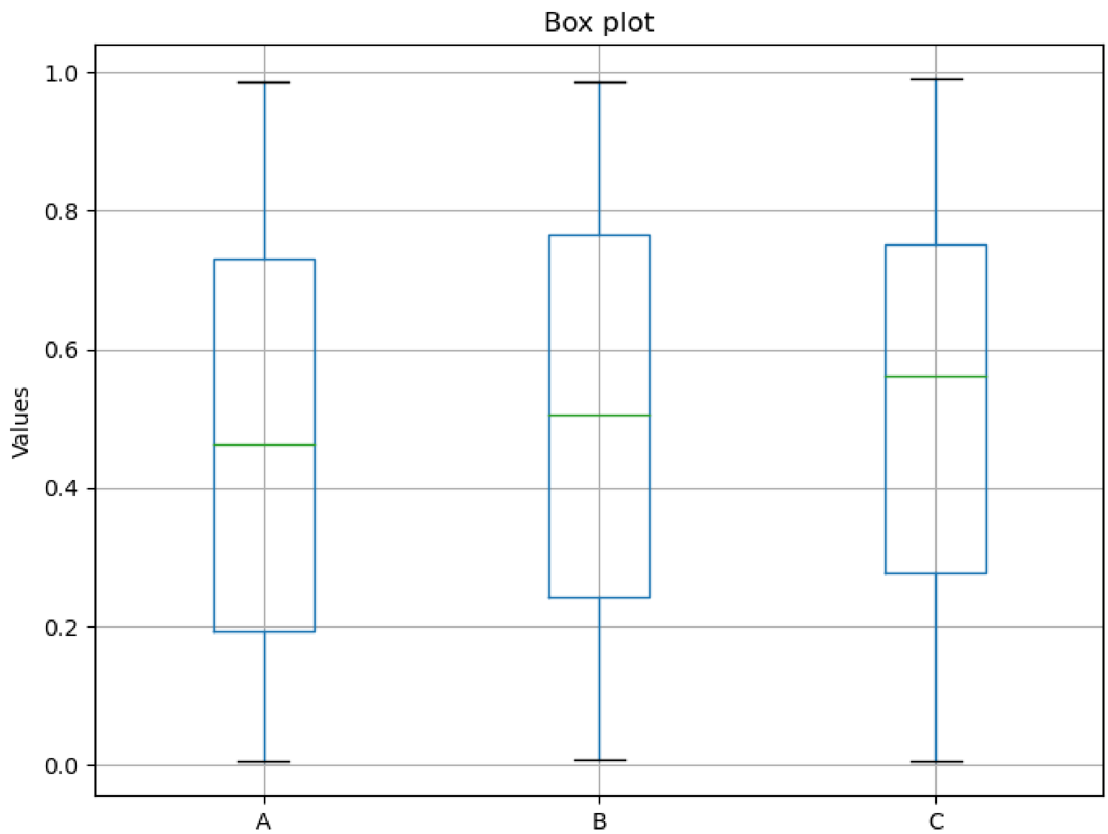
[45]:
```
# Line plot
plt.figure(figsize=(8, 6))
plt.plot(data['A'], data['B'])
plt.title('Line plot')
plt.xlabel('A')
plt.ylabel('B')
plt.show();
```



[46]:
```
# Histogram
plt.figure(figsize=(8, 6))
plt.hist(data['C'], bins=20)
plt.title('Histogram')
plt.xlabel('C')
plt.ylabel('Frequency')
plt.show();
```



[47]:
```
# Box plot
plt.figure(figsize=(8, 6))
data.boxplot(column=['A', 'B', 'C'])
plt.title('Box plot')
plt.ylabel('Values')
plt.show();
```


Box plot
1.0
0.8
0.6
0.4
0.2
0.0
Values
A
B
C