```
import numpy as np
import pandas as pd
import os
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans
from sklearn.metrics import confusion_matrix
import warnings
warnings.filterwarnings('ignore')
```

```
from google.colab import files
uploaded = files.upload()
```

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

```
Saving archive (2).zip to archive (2).zip
```

```
df = pd.read_csv('/content/archive (2).zip')
df.head()
```

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

```
df['species'].value_counts()
```

```
species
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: count, dtype: int64
```

```
df.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

```
df.info()
```

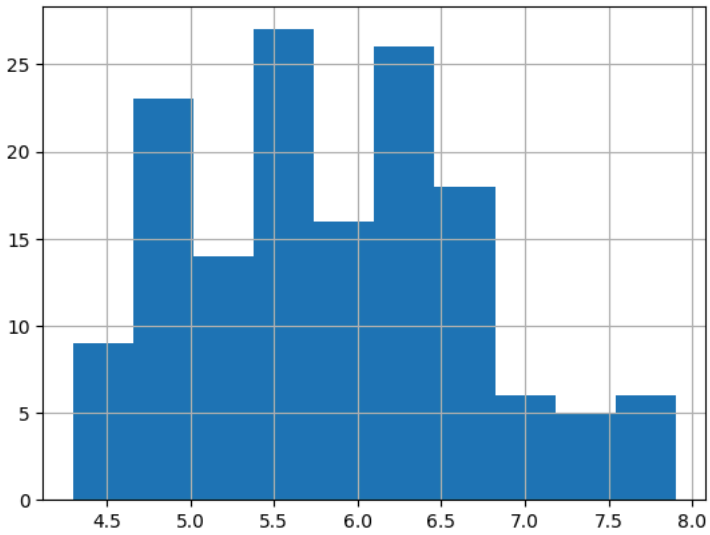
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

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

```
sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
species         0
dtype: int64
```

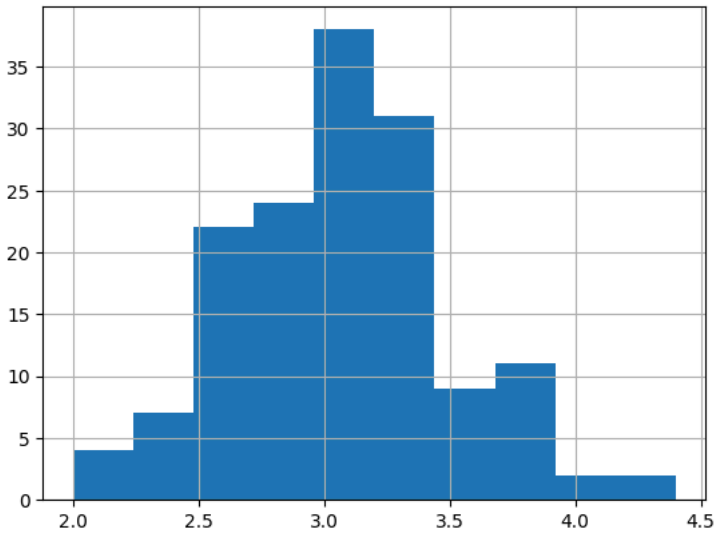
```
df['sepal_length'].hist()
```

```
<Axes: >
```



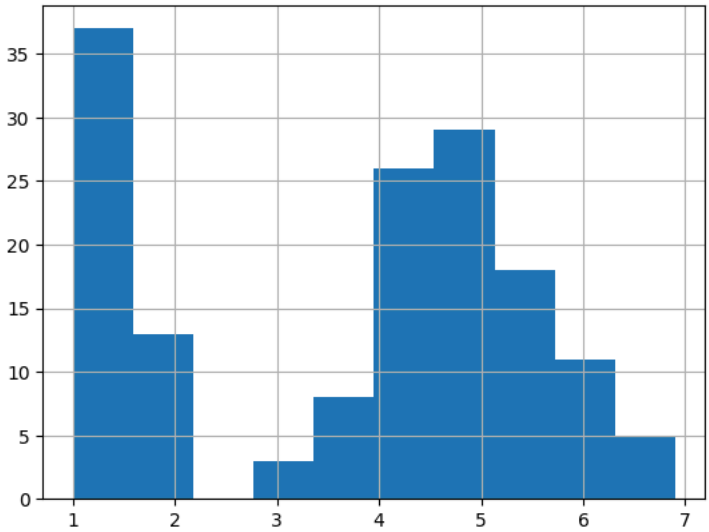

```
df['sepal_width'].hist()
```

```
<Axes: >
```



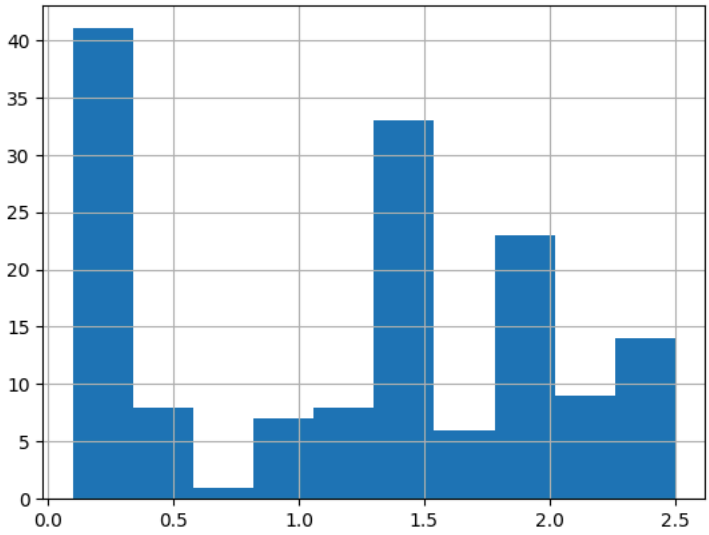

```
df['petal_length'].hist()
```

```
<Axes: >
```



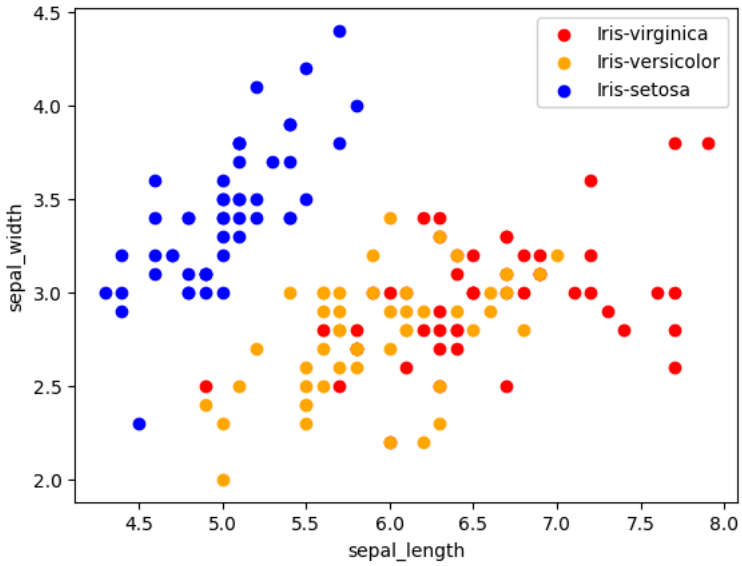

```
df['petal_width'].hist()
```

```
<Axes: >
```

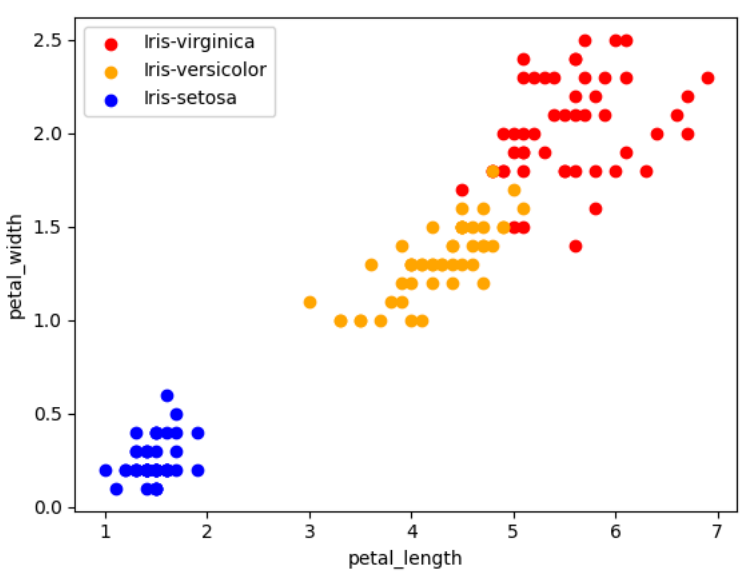



```
colors = ['red', 'orange', 'blue']
species = ['Iris-virginica', 'Iris-versicolor', 'Iris-setosa']
```

```
for i in range(3):
    x = df[df['species'] == species[i]]
    plt.scatter(x['sepal_length'], x['sepal_width'], c = colors[i], label = species[i])
    plt.xlabel('sepal_length')
    plt.ylabel('sepal_width')
    plt.legend()
```

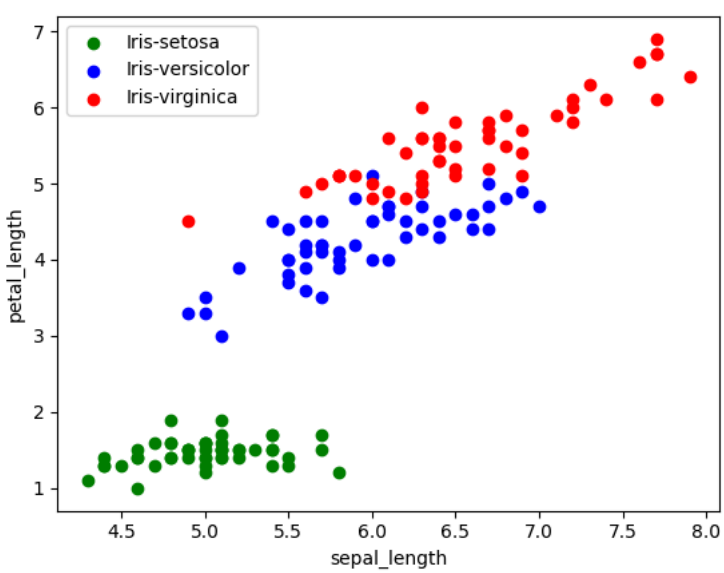



```
for i in range(3):
    x = df[df['species'] == species[i]]
    plt.scatter(x['petal_length'], x['petal_width'], c = colors[i], label = species[i])
    plt.xlabel('petal_length')
    plt.ylabel('petal_width')
    plt.legend()
```




```
colors = ['green','blue','red']
species = ['Iris-setosa','Iris-versicolor','Iris-virginica']
```

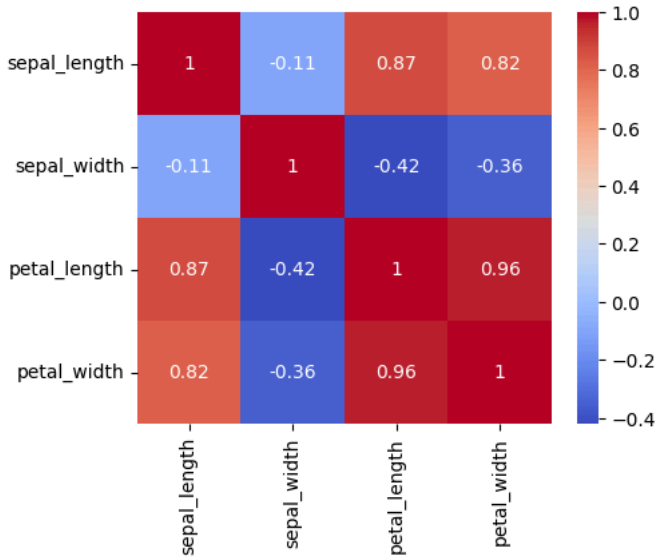
```
for i in range(3):
    x = df[df['species'] == species[i]]
    plt.scatter(x['sepal_length'], x['petal_length'], c = colors[i], label = species[i])
    plt.xlabel('sepal_length')
    plt.ylabel('petal_length')
    plt.legend()
```




```
corr = df.drop('species', axis = 1).corr()

fig, ax = plt.subplots(figsize = (5,4))
sns.heatmap(corr, annot = True, ax = ax, cmap = 'coolwarm')
```

```
<Axes: >
```




```
numeric_df = df.drop(columns=['species'])

correlation_matrix = numeric_df.corr()
```

```
correlation_matrix.corr()
```

	sepal_length	sepal_width	petal_length	petal_width
sepal_length	1.000000	-0.941225	0.975716	0.963204
sepal_width	-0.941225	1.000000	-0.992071	-0.994744
petal_length	0.975716	-0.992071	1.000000	0.997991
petal_width	0.963204	-0.994744	0.997991	1.000000

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
```

```
df['species']= le.fit_transform(df['species'])
df.head()
```

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

```
km = KMeans(n_clusters=3, random_state=0,)
y_predicted= km.fit_predict(df[['petal_length','petal_width']])
y_predicted
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 1, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1], dtype=int32)
```

```
from sklearn.model_selection import train_test_split
x = df.drop(columns=['species'])
y = df['species']
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.3)
```

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

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

```
▾ LogisticRegression
LogisticRegression()
```

```
print("Accuracy: ",model.score(X_test, y_test) * 100)
```

```
Accuracy:  88.88888888888889
```

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

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

```
▾ KNeighborsClassifier
KNeighborsClassifier()
```

```
print("Accuracy: ",model.score(X_test, y_test) * 100)
```

```
Accuracy:  91.11111111111111
```