```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn import svm
from sklearn import metrics
from sklearn.tree import DecisionTreeClassifier


iris=pd.read_csv("iris.csv")
print(iris)
```

```
     Unnamed: 0  Sepal.Length  Sepal.Width  Petal.Length  Petal.Width  \
0             1           5.1          3.5           1.4          0.2
1             2           4.9          3.0           1.4          0.2
2             3           4.7          3.2           1.3          0.2
3             4           4.6          3.1           1.5          0.2
4             5           5.0          3.6           1.4          0.2
..          ...           ...          ...           ...          ...
145         146           6.7          3.0           5.2          2.3
146         147           6.3          2.5           5.0          1.9
147         148           6.5          3.0           5.2          2.0
148         149           6.2          3.4           5.4          2.3
149         150           5.9          3.0           5.1          1.8

       Species
0       setosa
1       setosa
2       setosa
3       setosa
4       setosa
..         ...
145  virginica
146  virginica
147  virginica
148  virginica
149  virginica

[150 rows x 6 columns]
```

```
print(iris.shape)

print(iris.describe())
```

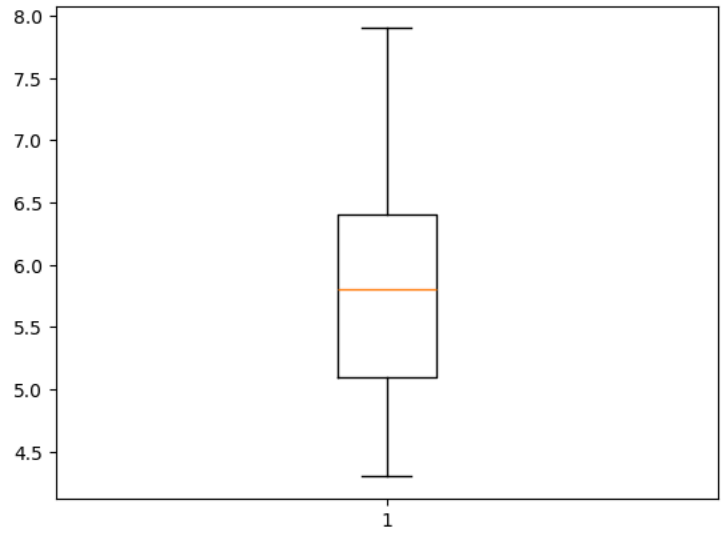
```
(150, 6)
       Unnamed: 0  Sepal.Length  Sepal.Width  Petal.Length  Petal.Width
count  150.000000    150.000000   150.000000    150.000000   150.000000
mean    75.500000      5.843333     3.057333      3.758000     1.199333
std     43.445368      0.828066     0.435866      1.765298     0.762238
min      1.000000      4.300000     2.000000      1.000000     0.100000
25%     38.250000      5.100000     2.800000      1.600000     0.300000
50%     75.500000      5.800000     3.000000      4.350000     1.300000
75%    112.750000      6.400000     3.300000      5.100000     1.800000
max    150.000000      7.900000     4.400000      6.900000     2.500000
```

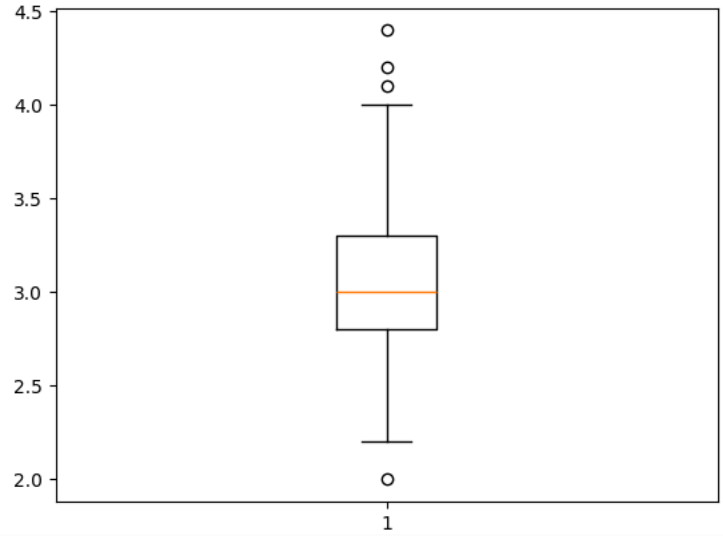
```
#Checking for null values
print(iris.isna().sum())
print(iris.describe())
```

```
Unnamed: 0      0
Sepal.Length    0
Sepal.Width     0
Petal.Length    0
Petal.Width     0
Species         0
dtype: int64
       Unnamed: 0  Sepal.Length  Sepal.Width  Petal.Length  Petal.Width
count  150.000000    150.000000   150.000000    150.000000   150.000000
mean    75.500000      5.843333     3.057333      3.758000     1.199333
std     43.445368      0.828066     0.435866      1.765298     0.762238
min      1.000000      4.300000     2.000000      1.000000     0.100000
25%     38.250000      5.100000     2.800000      1.600000     0.300000
50%     75.500000      5.800000     3.000000      4.350000     1.300000
75%    112.750000      6.400000     3.300000      5.100000     1.800000
max    150.000000      7.900000     4.400000      6.900000     2.500000
```

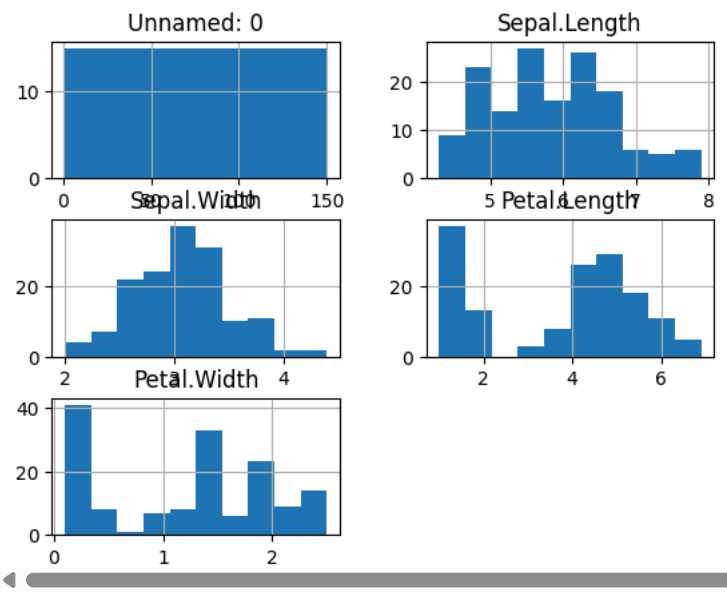
```
#Checking for outliars
import matplotlib.pyplot as plt
plt.figure(1)
plt.boxplot([iris['Sepal.Length']])
plt.figure(2)
```

```
plt.boxplot([iris['Sepal.Width']])
plt.show()
```






```
iris.hist()
plt.show()
```



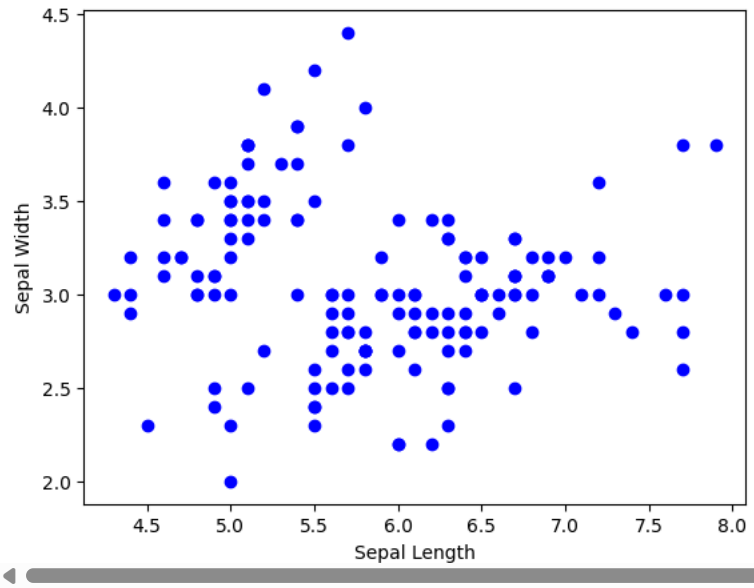

```
X = iris['Sepal.Length'].values.reshape(-1,1)
print(X)
```

```
[5.6]
[5.7]
[5.7]
[6.2]
[5.1]
[5.7]
[6.3]
[5.8]
[7.1]
[6.3]
[6.5]
[7.6]
[4.9]
[7.3]
[6.7]
[7.2]
[6.5]
[6.4]
[6.8]
[5.7]
[5.8]
[6.4]
[6.5]
[7.7]
[7.7]
[6. ]
[6.9]
[5.6]
[7.7]
[6.3]
[6.7]
[7.2]
[6.2]
[6.1]
[6.4]
[7.2]
[7.4]
[7.9]
[6.4]
[6.3]
[6.1]
[7.7]
[6.3]
[6.4]
[6. ]
[6.9]
[6.7]
[6.9]
[5.8]
[6.8]
[6.7]
[6.7]
[6.3]
[6.5]
[6.2]
[5.9]]
```

```
Y = iris['Sepal.Width'].values.reshape(-1,1)
print(Y)
```

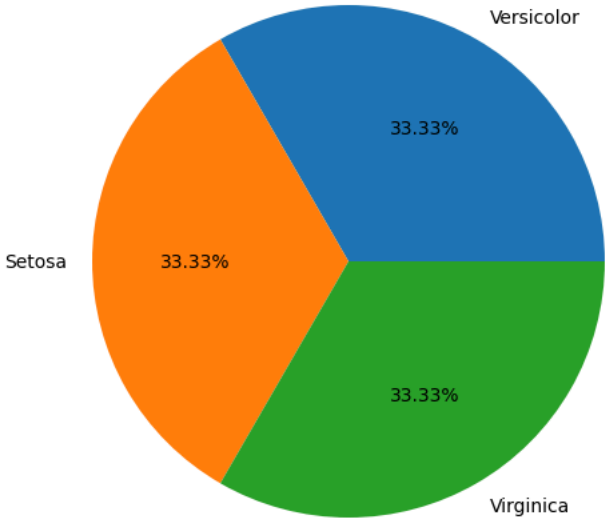
```
[2.8]
[2.8]
[2.7]
[3.3]
[3.2]
[2.8]
[3. ]
[2.8]
[3. ]
[2.8]
[3.8]
[2.8]
[2.8]
[2.6]
[3. ]
[3.4]
[3.1]
[3. ]
[3.1]
[3.1]
[3.1]
[2.7]
[3.2]
[3.3]
[3. ]
[2.5]
[3. ]
[3.4]
[3. ]]
```

```
plt.xlabel("Sepal Length")
plt.ylabel("Sepal Width")
plt.scatter(X,Y,color='b')
plt.show()
```




```
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
ax.axis('equal')
l = ['Versicolor', 'Setosa', 'Virginica']
s = [50,50,50]
ax.pie(s, labels = l,autopct='%1.2f%%')
plt.show()
```

```
WARNING:matplotlib.axes._base:Ignoring fixed x limits to fulfill fixed data aspect with adjustable data limits.
```




```
train, test = train_test_split(iris, test_size = 0.25)
print(train.shape)
print(test.shape)

train_X = train[['Sepal.Length', 'Sepal.Width', 'Petal.Length',
                 'Petal.Width']]
train_y = train.Species

test_X = test[['Sepal.Length', 'Sepal.Width', 'Petal.Length',
                 'Petal.Width']]
test_y = test.Species
train_X.head()
```

```
(112, 6)
(38, 6)
```

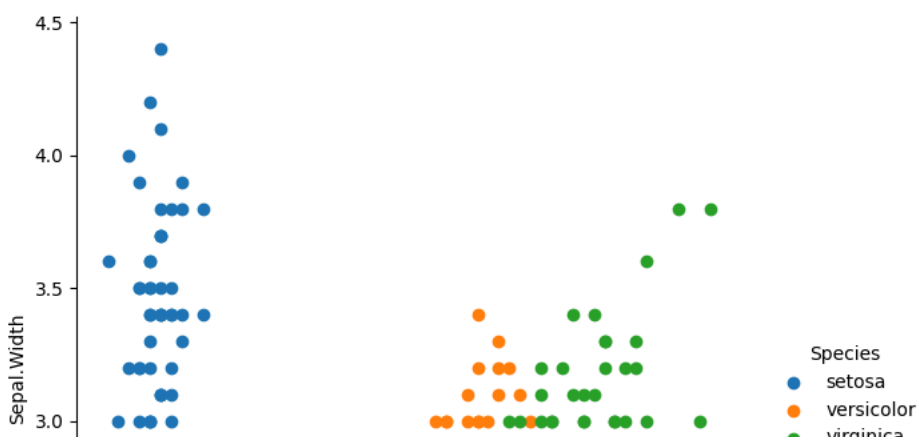
	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
71	6.1	2.8	4.0	1.3
14	5.8	4.0	1.2	0.2
145	6.7	3.0	5.2	2.3
49	5.0	3.3	1.4	0.2
122	7.7	2.8	6.7	2.0

Next steps: Generate code with train_X | View recommended plots | New interactive sheet

```
# Create a scatter plot using FacetGrid
g = sns.FacetGrid(iris, hue="Species", height=6)
g.map(plt.scatter, "Petal.Length", "Sepal.Width")

# Add legend
g.add_legend()

# Show the plot
plt.show()
```




```
model = LogisticRegression()
model.fit(train_X, train_y)
prediction = model.predict(test_X)
print('Accuracy:',metrics.accuracy_score(prediction,test_y))

#Confusion matrix
from sklearn.metrics import confusion_matrix
confusion_mat = confusion_matrix(test_y,prediction)
print("Confusion matrix: \n",confusion_mat)
```

```
Accuracy: 1.0
Confusion matrix:
 [[15  0  0]
 [ 0 10  0]
 [ 0  0 13]]
/usr/local/lib/python3.11/dist-packages/sklearn/linear_model/_logistic.py:465: ConvergenceWarning: lbfgs failed to converge (status=
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
results = pd.DataFrame({
    'Model': ['Logistic Regression','Support Vector Machines', 'Naive Bayes','KNN' ,'Decision Tree'],
    'Score': [0.947,0.947,0.947,0.947,0.921]})

result_df = results.sort_values(by='Score', ascending=False)
result_df = result_df.set_index('Score')
result_df.head(9)
```

Score	Model
0.947	Logistic Regression
0.947	Support Vector Machines
0.947	Naive Bayes
0.947	KNN
0.921	Decision Tree