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

```
from warnings import filterwarnings
filterwarnings(action='ignore')
```

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

```
        Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm  \
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       Iris-setosa
1       Iris-setosa
2       Iris-setosa
3       Iris-setosa
4       Iris-setosa
..              ...
145  Iris-virginica
146  Iris-virginica
147  Iris-virginica
148  Iris-virginica
149  Iris-virginica

[150 rows x 6 columns]
```

```
print(iris.shape)
```

```
(150, 6)
```

```
print(iris.describe())
```

```
                Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
count  150.000000     150.000000    150.000000     150.000000    150.000000
mean    75.500000       5.843333      3.054000       3.758667      1.198667
std     43.445368       0.828066      0.433594       1.764420      0.763161
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())
```

```
Id              0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
                Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
count  150.000000     150.000000    150.000000     150.000000    150.000000
mean    75.500000       5.843333      3.054000       3.758667      1.198667
std     43.445368       0.828066      0.433594       1.764420      0.763161
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
```

```
iris.head()
```

	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

```
iris.head(150)
```

	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

```
iris.tail(100)
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
50	51	7.0	3.2	4.7	1.4	Iris-versicolor
51	52	6.4	3.2	4.5	1.5	Iris-versicolor
52	53	6.9	3.1	4.9	1.5	Iris-versicolor
53	54	5.5	2.3	4.0	1.3	Iris-versicolor
54	55	6.5	2.8	4.6	1.5	Iris-versicolor
...	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica

```
n = len(iris[iris['Species'] == 'versicolor'])
print("No of Versicolor in Dataset:",n)
```

```
No of Versicolor in Dataset: 0
```

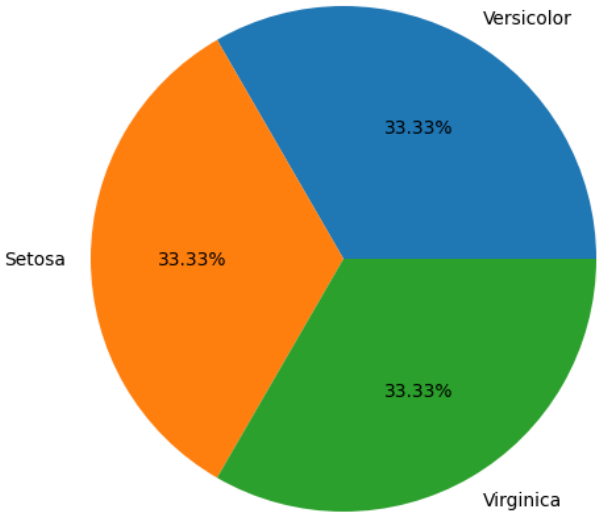
```
n1 = len(iris[iris['Species'] == 'virginica'])
print("No of Virginica in Dataset:",n1)
```

```
No of Virginica in Dataset: 0
```

```
n2 = len(iris[iris['Species'] == 'setosa'])
print("No of Setosa in Dataset:",n2)
```

```
No of Setosa in Dataset: 0
```

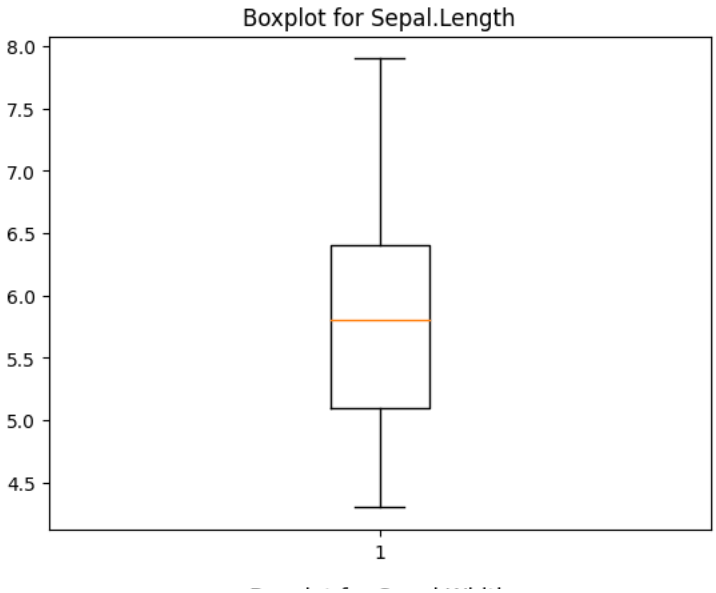
```
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
ax.axis('equal')
l = ['Versicolor', 'Setosa', 'Virginica']
s = [50,50,50]
ax.pie(s, labels = l,autopct='%1.2f%%')
plt.show()
```



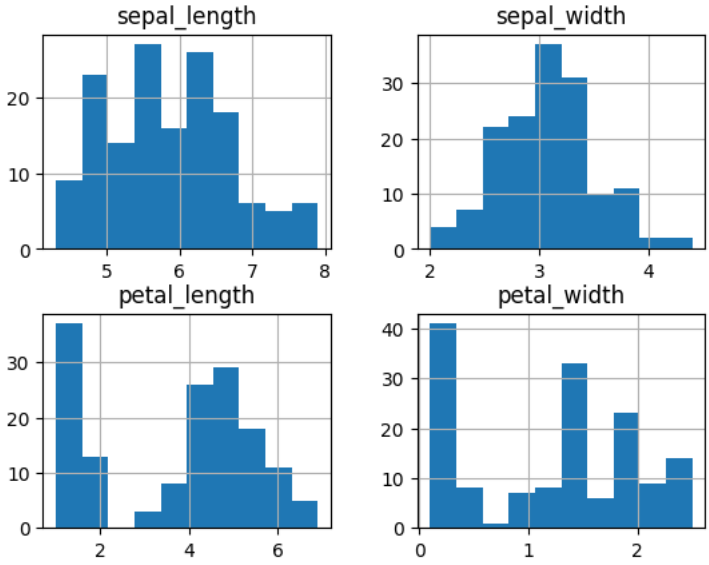

```
import seaborn as sns

# Load Iris dataset
iris = sns.load_dataset('iris')

# Print column names
print(iris.columns)
```

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

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

# Load Iris dataset
iris = sns.load_dataset('iris')

# Create boxplots for Sepal.Length and Sepal.Width
plt.figure(1)
plt.boxplot([iris['sepal_length']])
plt.title('Boxplot for Sepal.Length')

plt.figure(2)
plt.boxplot([iris['sepal_width']])
plt.title('Boxplot for Sepal.Width')

plt.show()
```

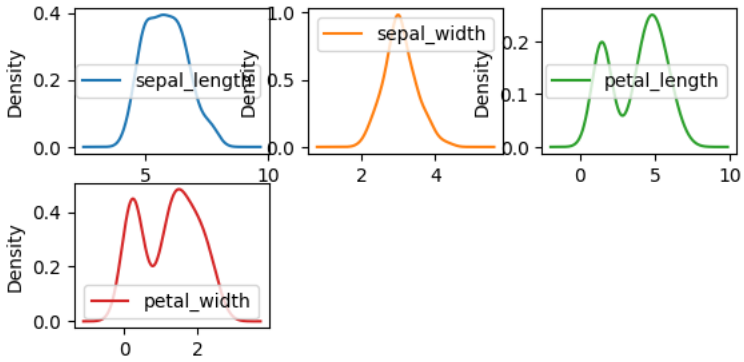



```
iris.hist()
plt.show()
```



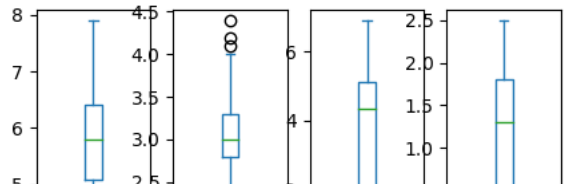

```
iris.plot(kind ='density',subplots = True, layout =(3,3),sharex = False)
```

```
array([[<Axes: ylabel='Density'>, <Axes: ylabel='Density'>,
        <Axes: ylabel='Density'>],
       [<Axes: ylabel='Density'>, <Axes: ylabel='Density'>,
        <Axes: ylabel='Density'>],
       [<Axes: ylabel='Density'>, <Axes: ylabel='Density'>,
        <Axes: ylabel='Density'>]], dtype=object)
```

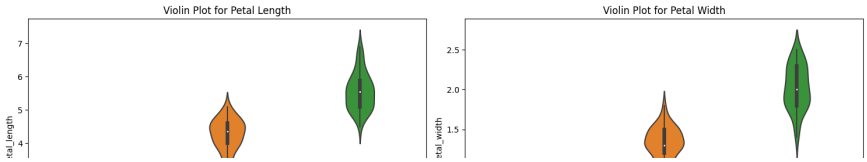



```
iris.plot(kind ='box',subplots = True, layout =(2,5),sharex = False)
```

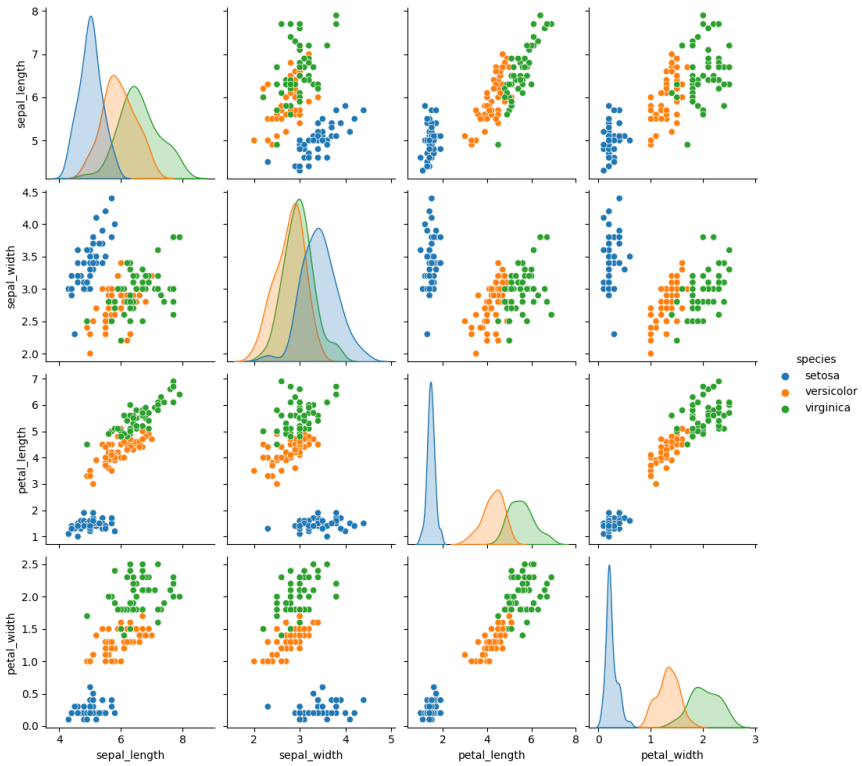
```
sepal_length       Axes(0.125,0.53;0.133621x0.35)
sepal_width     Axes(0.285345,0.53;0.133621x0.35)
petal_length     Axes(0.44569,0.53;0.133621x0.35)
petal_width     Axes(0.606034,0.53;0.133621x0.35)
dtype: object
```



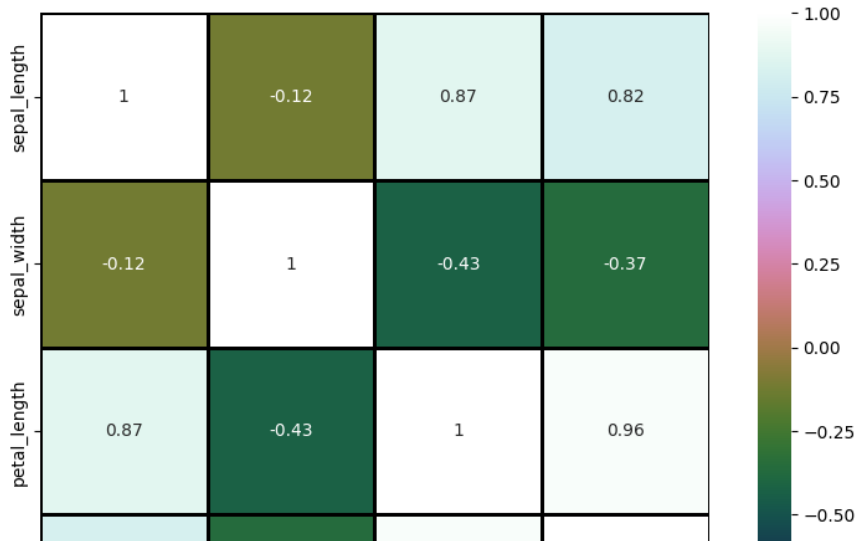
```
plt.figure(figsize=(15, 10))

# Replace 'Species' with the correct column name for species
plt.subplot(2, 2, 1)
sns.violinplot(x='species', y='petal_length', data=iris)
plt.title('Violin Plot for Petal Length')

plt.subplot(2, 2, 2)
sns.violinplot(x='species', y='petal_width', data=iris)
plt.title('Violin Plot for Petal Width')

plt.subplot(2, 2, 3)
sns.violinplot(x='species', y='sepal_length', data=iris)
plt.title('Violin Plot for Sepal Length')

plt.subplot(2, 2, 4)
sns.violinplot(x='species', y='sepal_width', data=iris)
plt.title('Violin Plot for Sepal Width')

plt.tight_layout()
plt.show()
```




```
# Create pair plot with hue='species'
sns.pairplot(iris, hue='species')

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




```
#Heat Maps
fig=plt.gcf()
fig.set_size_inches(10,7)
fig=sns.heatmap(iris.corr(),annot=True,cmap='cubehelix',linewidths=1,linecolor='k',square=True,mask=False, vmin=-1, vmax=1,c
```




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

```
[[5.1]
 [4.9]
 [4.7]
 [4.6]
 [5. ]
 [5.4]
 [4.6]
 [5. ]
 [4.4]
 [4.9]
 [5.4]
 [4.8]
 [4.8]
 [4.3]
 [5.8]
 [5.7]
 [5.4]
 [5.1]
 [5.7]
 [5.1]
 [5.4]
 [5.1]
 [4.6]
 [5.1]
 [4.8]
 [5. ]
 [5. ]
 [5.2]
 [5.2]
 [4.7]
 [4.8]
 [5.4]
 [5.2]
 [5.5]
 [4.9]
 [5. ]
 [5.5]
 [4.9]
 [4.4]
 [5.1]
 [5. ]
 [4.5]
 [4.4]
 [5. ]
 [5.1]
 [4.8]
 [5.1]
 [4.6]
 [5.3]
 [5. ]
 [7. ]
 [6.4]
 [6.9]
 [5.5]
 [6.5]
 [5.7]
 [6.3]
 [4.9]
```

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

```
      [2.6]
      [2.3]
      [2.7]
      [3. ]
      [2.9]
      [2.9]
      [2.5]
      [2.8]
      [3.3]
      [2.7]
      [3. ]
      [2.9]
      [3. ]
      [3. ]
      [2.5]
      [2.9]
      [2.5]
      [3.6]
      [3.2]
      [2.7]
      [3. ]
      [2.5]
      [2.8]
      [3.2]
      [3. ]
      [3.8]
      [2.6]
      [2.2]
      [3.2]
      [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()
```


```
#Correlation
corr_mat = iris.corr()
print(corr_mat)
```

```
                sepal_length  sepal_width  petal_length  petal_width
sepal_length        1.000000    -0.117570      0.871754     0.817941
sepal_width        -0.117570     1.000000     -0.428440    -0.366126
petal_length        0.871754    -0.428440      1.000000     0.962865
petal_width         0.817941    -0.366126      0.962865     1.000000
```

```
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
```



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

```
(112, 5)
(38, 5)
```

```
# Print column names of the training dataset
print(train.columns)

# Print column names of the testing dataset
print(test.columns)
```

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

```
# Assuming 'species' is the target variable
feature_columns = ['sepal_length', 'sepal_width', 'petal_length', 'petal_width']

train_X = train[feature_columns]
train_y = train['species']

test_X = test[feature_columns]
test_y = test['species']
```

```
train_X.head()
```

	sepal_length	sepal_width	petal_length	petal_width
16	5.4	3.9	1.3	0.4
103	6.3	2.9	5.6	1.8
24	4.8	3.4	1.9	0.2
129	7.2	3.0	5.8	1.6
76	6.8	2.8	4.8	1.4

```
test_y.head()
```

```
97     versicolor
146     virginica
108     virginica
9          setosa
131     virginica
Name: species, dtype: object
```

```
test_y.head()
```

```
97     versicolor
146     virginica
108     virginica
```

```
9          setosa
131     virginica
Name: species, dtype: object
```

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

```
Accuracy: 0.9473684210526315
```

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

```
Confusion matrix:
 [[14  0  0]
 [ 0 13  1]
 [ 0  1  9]]
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        14
  versicolor       0.93      0.93      0.93        14
   virginica       0.90      0.90      0.90        10

    accuracy                           0.95        38
   macro avg       0.94      0.94      0.94        38
weighted avg       0.95      0.95      0.95        38
```

```
#Using Support Vector
from sklearn.svm import SVC
model1 = SVC()
model1.fit(train_X,train_y)

pred_y = model1.predict(test_X)

from sklearn.metrics import accuracy_score
print("Acc=",accuracy_score(test_y,pred_y))
```

```
Acc= 0.9473684210526315
```

```
#Using KNN Neighbors
from sklearn.neighbors import KNeighborsClassifier
model2 = KNeighborsClassifier(n_neighbors=5)
model2.fit(train_X,train_y)
y_pred2 = model2.predict(test_X)

from sklearn.metrics import accuracy_score
print("Accuracy Score:",accuracy_score(test_y,y_pred2))
```

```
Accuracy Score: 0.9473684210526315
```

```
#Using GaussianNB
from sklearn.naive_bayes import GaussianNB
model3 = GaussianNB()
model3.fit(train_X,train_y)
y_pred3 = model3.predict(test_X)

from sklearn.metrics import accuracy_score
print("Accuracy Score:",accuracy_score(test_y,y_pred3))
```

```
Accuracy Score: 0.9736842105263158
```

```
#Using Decision Tree
from sklearn.tree import DecisionTreeClassifier
model4 = DecisionTreeClassifier(criterion='entropy',random_state=7)
model4.fit(train_X,train_y)
y_pred4 = model4.predict(test_X)

from sklearn.metrics import accuracy_score
print("Accuracy Score:",accuracy_score(test_y,y_pred4))
```

```
Accuracy Score: 0.9473684210526315
```

```
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

Start coding or generate with AI.