IRIS_DataAnalysis

July 11, 2021

1 Library Import

[1]:
```
#Importing relevant libraries

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import statsmodels.api as sm
from statsmodels.formula.api import ols
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
```

2 Reading the dataset

[2]:
```
#Reading the dataset and copying into a DataFrame

data = pd.read_csv("IRIS_Dataset.csv")
```

[3]:
```
#Columns heads in the dataset

data.head(0)
```

[3]:
```
Empty DataFrame
Columns: [sepal_length, sepal_width, petal_length, petal_width, species]
Index: []
```

[4]:
```
#Description of Data

data.describe()
```

[4]:
```
       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
```

[5]:
```
#Check for Null Values

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

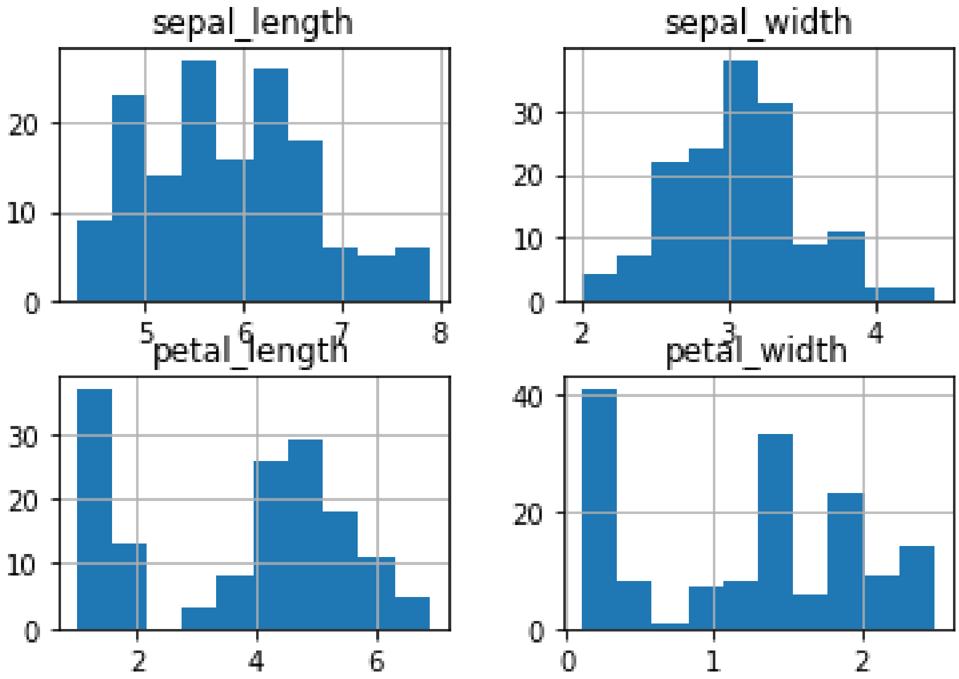
[5]:
```
sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
species         0
dtype: int64
```

3 Data Visualization

[6]:
```
#Data Visualization - Histograms

data.hist()
```

[6]:
```
array([[<AxesSubplot:title={'center':'sepal_length'}>,
        <AxesSubplot:title={'center':'sepal_width'}>],
       [<AxesSubplot:title={'center':'petal_length'}>,
        <AxesSubplot:title={'center':'petal_width'}>]], dtype=object)
```

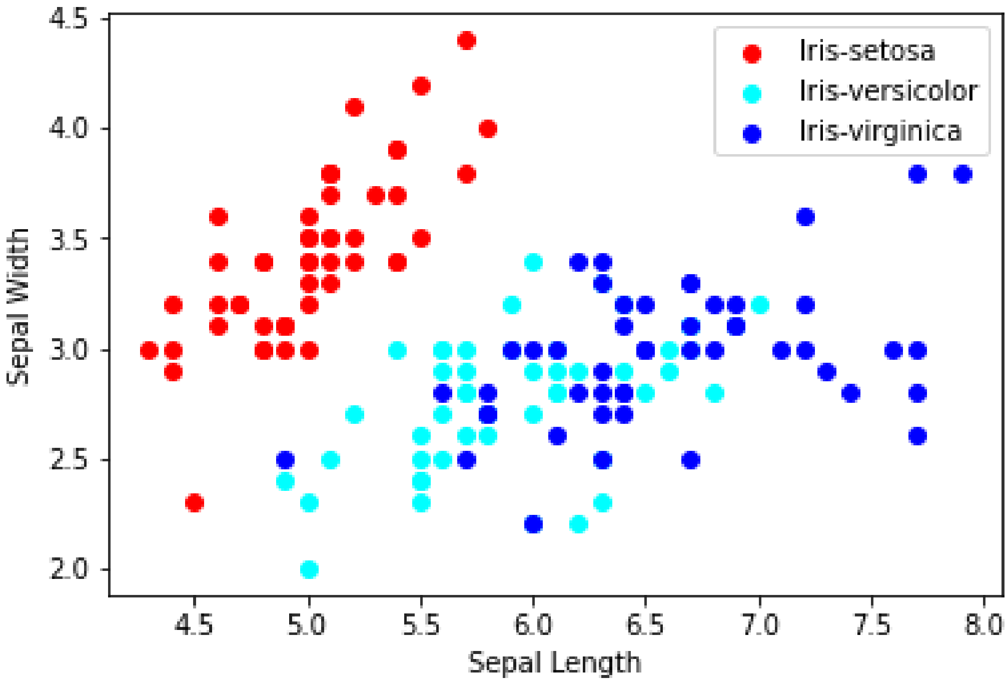


```
[7]: #Data Visualization - Scatterplots (Sepal)

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

     for i in range(3):
         x = data[data['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()
```

```
[7]: <matplotlib.legend.Legend at 0x2b84c68e850>
```



[8]:

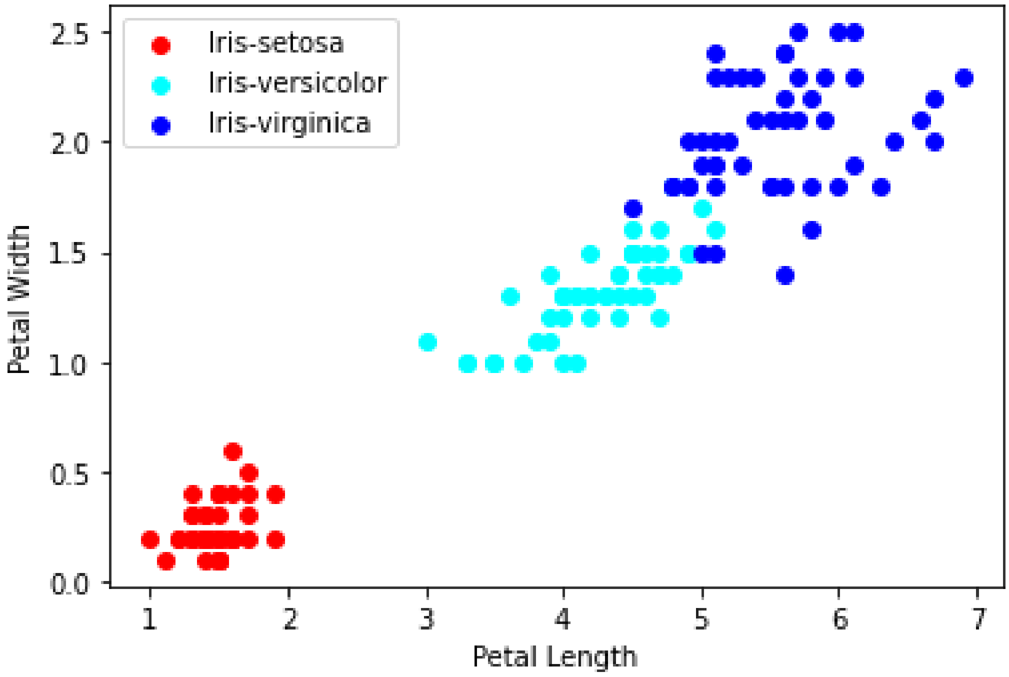
```
#Data Visualization - Scatterplots (Petal)

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

for i in range(3):
    x = data[data['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()
```

[8]:

```
<matplotlib.legend.Legend at 0x2b84c6ed700>
```

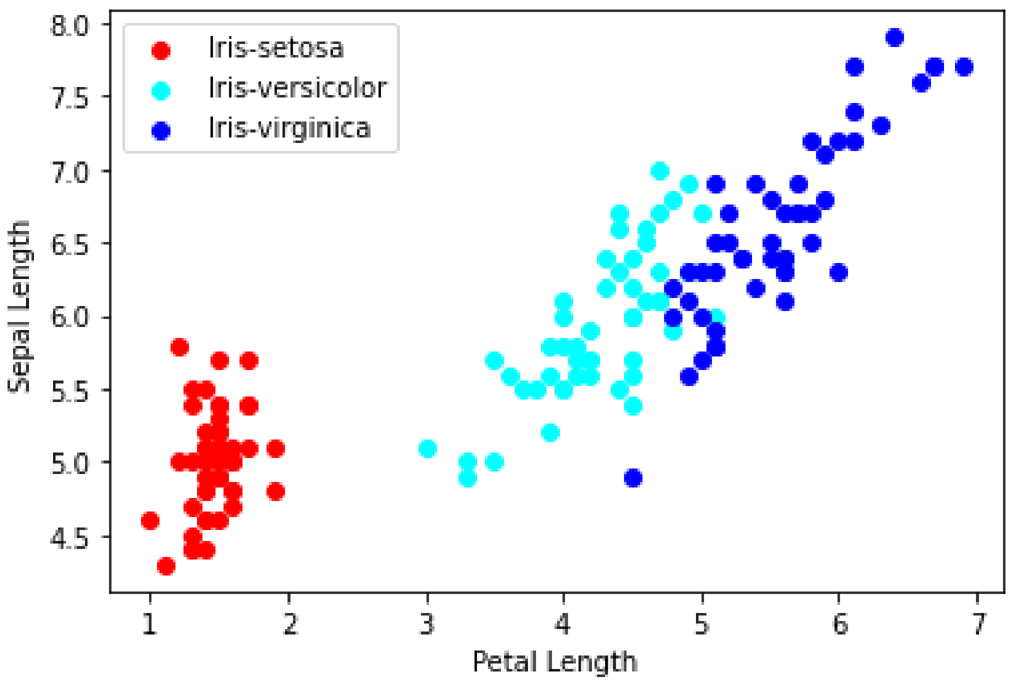


```
[9]: #Data Visualization - Scatterplots (Lengthwise)

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

     for i in range(3):
         x = data[data['species']==species[i]]
         plt.scatter(x['petal_length'], x['sepal_length'], c=colors[i],␣
      ↪label=species[i])
     plt.xlabel("Petal Length")
     plt.ylabel("Sepal Length")
     plt.legend()
```

```
[9]: <matplotlib.legend.Legend at 0x2b84d750550>
```

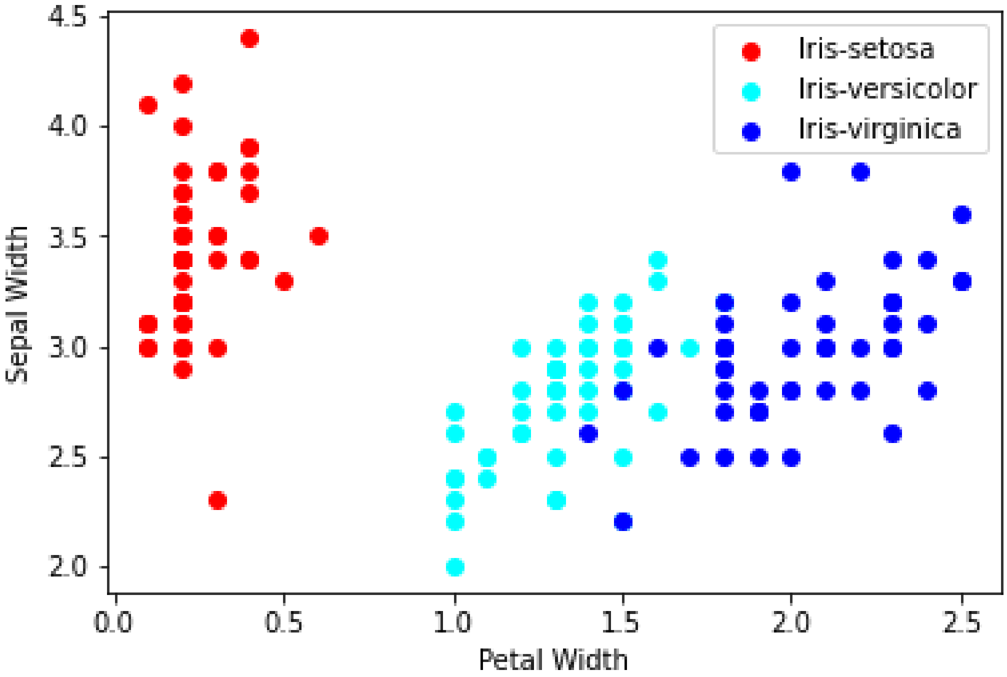


```
[10]: #Data Visualization - Scatterplots (Petal)

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

      for i in range(3):
          x = data[data['species']==species[i]]
          plt.scatter(x['petal_width'], x['sepal_width'], c=colors[i],␣
       ↪label=species[i])
      plt.xlabel("Petal Width")
      plt.ylabel("Sepal Width")
      plt.legend()
```

```
[10]: <matplotlib.legend.Legend at 0x2b84d7baa60>
```



4 Correlation Matrix & HeatMap

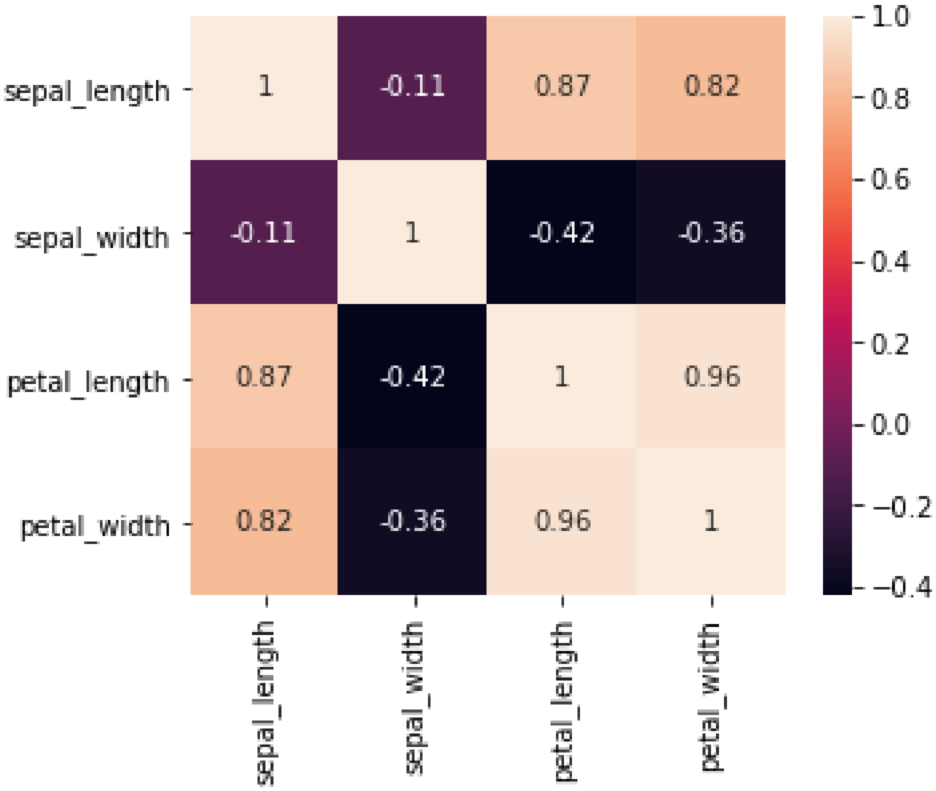
[11]:
```
#Correlation Matrix

data.corr()
```

[11]:
```
              sepal_length  sepal_width  petal_length  petal_width
sepal_length      1.000000    -0.109369      0.871754     0.817954
sepal_width      -0.109369     1.000000     -0.420516    -0.356544
petal_length      0.871754    -0.420516      1.000000     0.962757
petal_width       0.817954    -0.356544      0.962757     1.000000
```

[12]:
```
corr = data.corr()
fig, ax = plt.subplots(figsize=(5,4))
sns.heatmap(corr, annot=True, ax=ax, cmap='rocket')
```

[12]:
```
<AxesSubplot:>
```



5 Label Encoder (Preprocessing)

[13]:
```
#Label Encoder

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

[13]:
```
   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
```

6 Model Training

[14]:
```
#Model Training
#Train - 70%
#Test - 30%

X = data.drop(columns=['species'])
Y = data['species']
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size = 0.30)
```

6.0.1 Logistic Regression

[15]:
```
#Logistic Regression

model1 = LogisticRegression()
```

[16]:
```
#Model Training

model1.fit(x_train, y_train)
```

[16]:
```
LogisticRegression()
```

[17]:
```
#Print metric to get performance

print("Accuracy: ", model1.score(x_test, y_test)*100)
```

```
Accuracy:  100.0
```

6.0.2 K-Nearest Neighbors

[18]:
```
#KNN - K-Nearest Neighbors

model2 = KNeighborsClassifier()
```

[19]:
```
#Model Training

model2.fit(x_train, y_train)
```

[19]:
```
KNeighborsClassifier()
```

[20]:
```
#Print metric to get performance

print("Accuracy: ", model2.score(x_test, y_test)*100)
```

```
Accuracy:  100.0
```

6.0.3 Decision Tree

[21]:

```
#Decision Tree

model3 = DecisionTreeClassifier()
```

[22]:

```
#Model Training

model3.fit(x_train, y_train)
```

[22]:

```
DecisionTreeClassifier()
```

[23]:

```
#Print metric to get performance

print("Accuracy: ", model3.score(x_test, y_test)*100)
```

```
Accuracy:  95.55555555555556
```