assignment2

September 2, 2021

1 IN-STK 5000: Reproducibility assignment

[20]:
```
#importing libraries

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

[8]:
```
#uploading the data
df = pd.read_csv('https://archive.ics.uci.edu/ml/machine-learning-databases/
↪iris/iris.data')
```

[9]:
```
df.head(10)
```

[9]:

	5.1	3.5	1.4	0.2	Iris-setosa
0	4.9	3.0	1.4	0.2	Iris-setosa
1	4.7	3.2	1.3	0.2	Iris-setosa
2	4.6	3.1	1.5	0.2	Iris-setosa
3	5.0	3.6	1.4	0.2	Iris-setosa
4	5.4	3.9	1.7	0.4	Iris-setosa
5	4.6	3.4	1.4	0.3	Iris-setosa
6	5.0	3.4	1.5	0.2	Iris-setosa
7	4.4	2.9	1.4	0.2	Iris-setosa
8	4.9	3.1	1.5	0.1	Iris-setosa
9	5.4	3.7	1.5	0.2	Iris-setosa

The column names are missing in the dataframe. The column names are following: 1. sepal length in cm 2. sepal width in cm 3. petal length in cm 4. petal width in cm 5. class: – Iris Setosa – Iris Versicolour – Iris Virginica

[10]:
```
df = pd.read_csv('https://archive.ics.uci.edu/ml/machine-learning-databases/
↪iris/iris.data',
                 names=␣
↪["SepalLength","SepalWidth","PetalLength","PetalWidth","Species"])
```

[11]:
```
df.head(10)
```

[11]:

	SepalLength	SepalWidth	PetalLength	PetalWidth	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
5	5.4	3.9	1.7	0.4	Iris-setosa
6	4.6	3.4	1.4	0.3	Iris-setosa
7	5.0	3.4	1.5	0.2	Iris-setosa
8	4.4	2.9	1.4	0.2	Iris-setosa
9	4.9	3.1	1.5	0.1	Iris-setosa

[12]:
```
#to display stats about the data

df.describe()
```

[12]:

	SepalLength	SepalWidth	PetalLength	PetalWidth
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

[13]:
```
#to basic info about the datatype
df.info()
```

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

[14]:
```
#to display no. of samples on each class
df['Species'].value_counts()
```

[14]:
```
Iris-setosa        50
Iris-versicolor    50
```

```
Iris-virginica     50
Name: Species, dtype: int64
```

1.1 Preprocessing the dataset

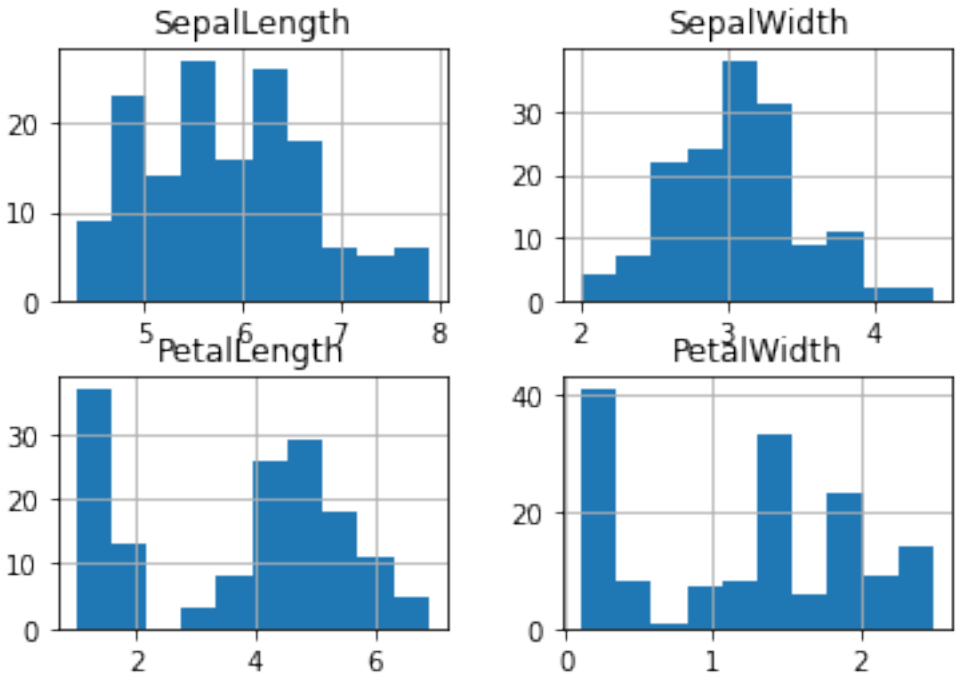
[15]:
```
df.isnull().sum()
```

[15]:
```
SepalLength    0
SepalWidth     0
PetalLength    0
PetalWidth     0
Species        0
dtype: int64
```

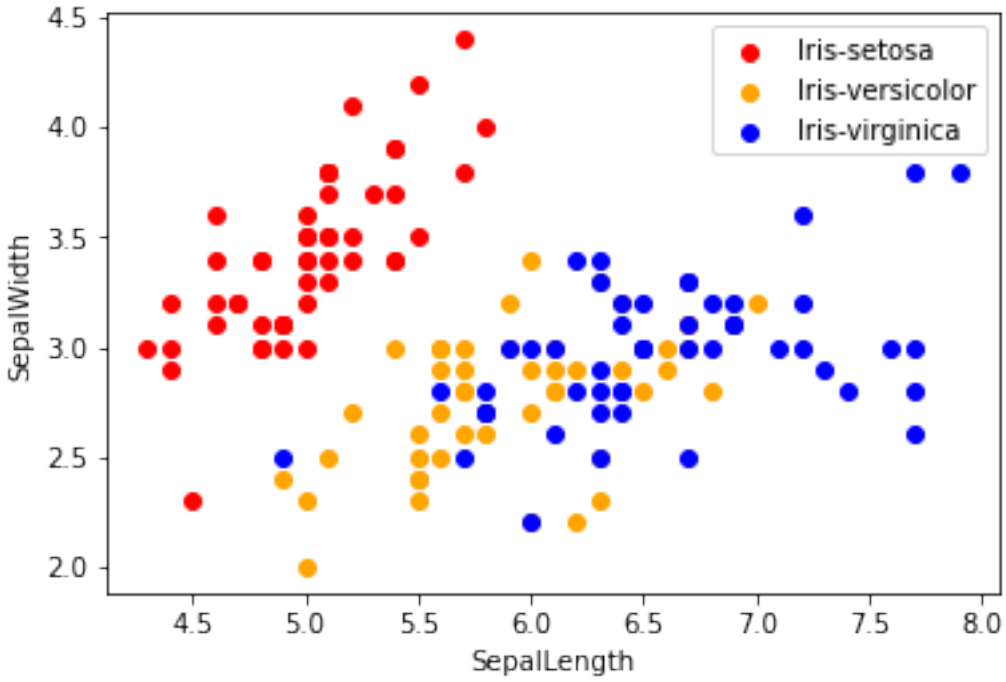
1.2 Exploratory Data Analysis

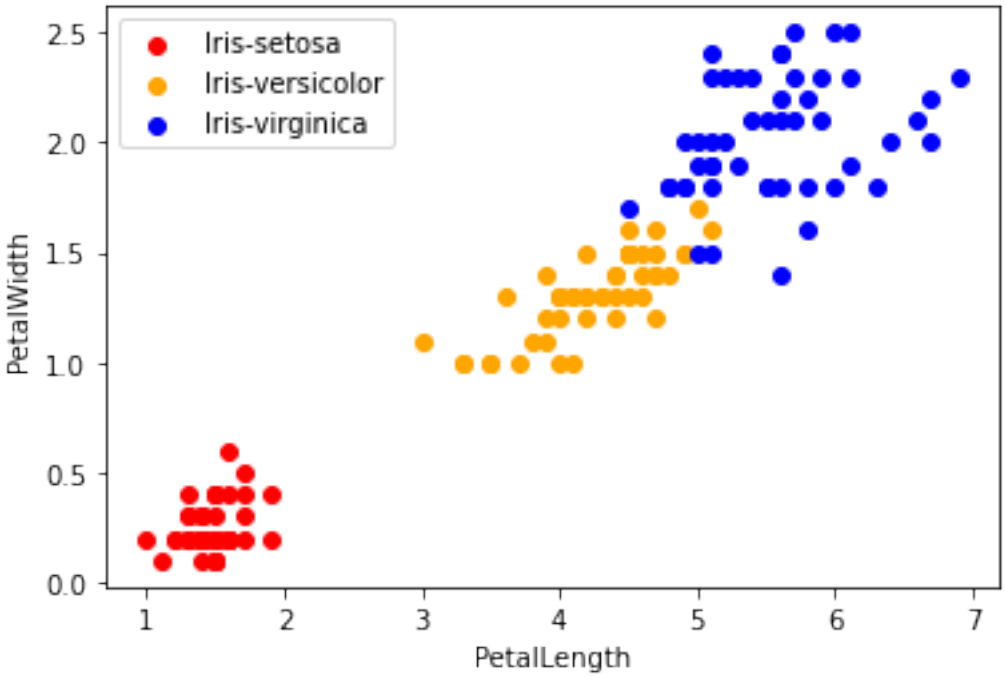
[16]:
```
df.hist()
```

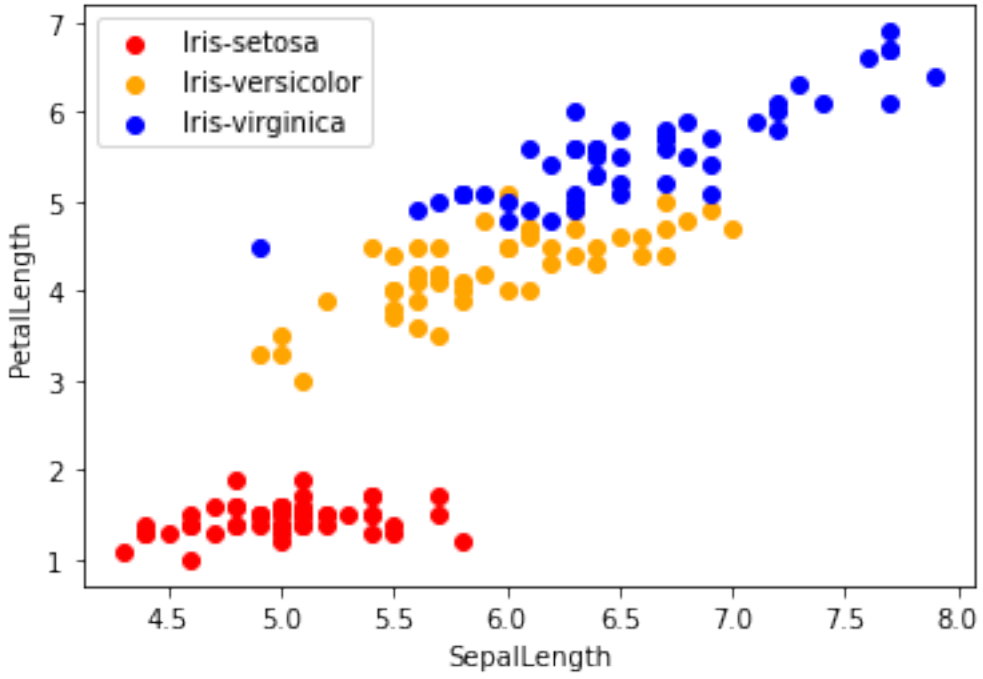
[16]:
```
array([[<AxesSubplot:title={'center':'SepalLength'}>,
        <AxesSubplot:title={'center':'SepalWidth'}>],
       [<AxesSubplot:title={'center':'PetalLength'}>,
        <AxesSubplot:title={'center':'PetalWidth'}>]], dtype=object)
```

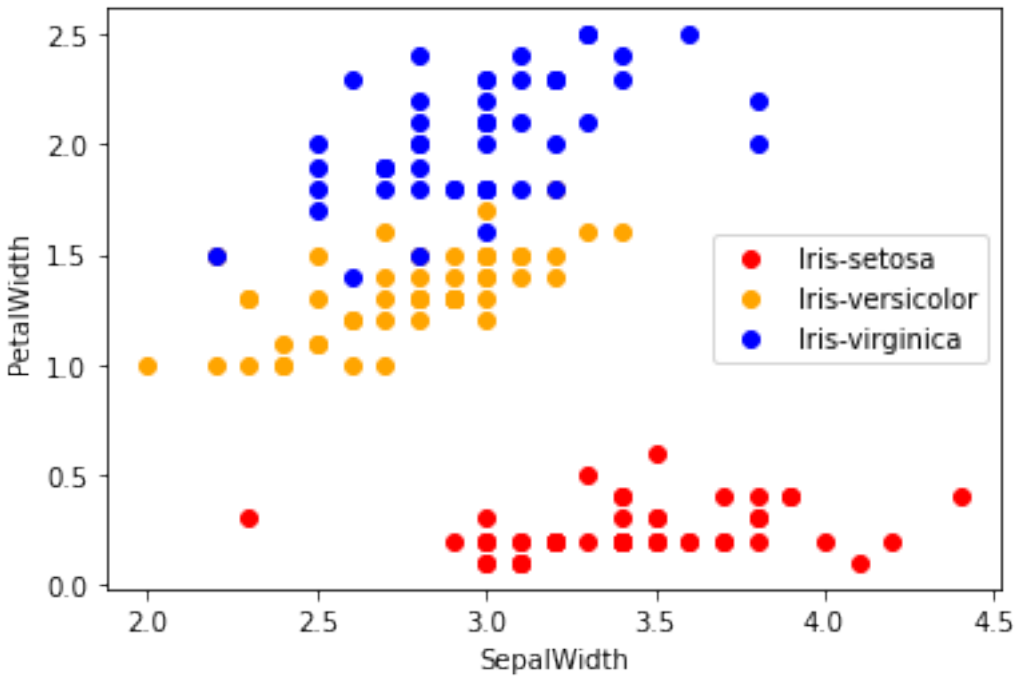


[17]:
```
#scatterplots

def plotting(sLength, sWidth):
    colors = ['red','orange','blue']
    species = ['Iris-setosa','Iris-versicolor','Iris-virginica']

    for i in range(3):
        x = df[df['Species'] == species[i]]
        plt.scatter(x[sLength],x[sWidth],c=colors[i],label=species[i])

    plt.xlabel(sLength)
    plt.ylabel(sWidth)
    plt.legend()
    plt.figure()

plotting("SepalLength","SepalWidth")
plotting("PetalLength","PetalWidth")
plotting("SepalLength","PetalLength")
plotting("SepalWidth","PetalWidth")
```




Iris-setosa
Iris-versicolor
Iris-virginica
PetalWidth
PetalLength
0.0
0.5
1.0
1.5
2.0
2.5
1
2
3
4
5
6
7


Iris-setosa
Iris-versicolor
Iris-virginica
PetalLength
SepalLength
1
2
3
4
5
6
7
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0




```
<Figure size 432x288 with 0 Axes>
```

1.3 Coorelation matrix

We can use the coorelation matrix to neglect the variables with high correlation. Each cell in the table shows the correlation between two variables. The value of target is in the range of -1 to 1.

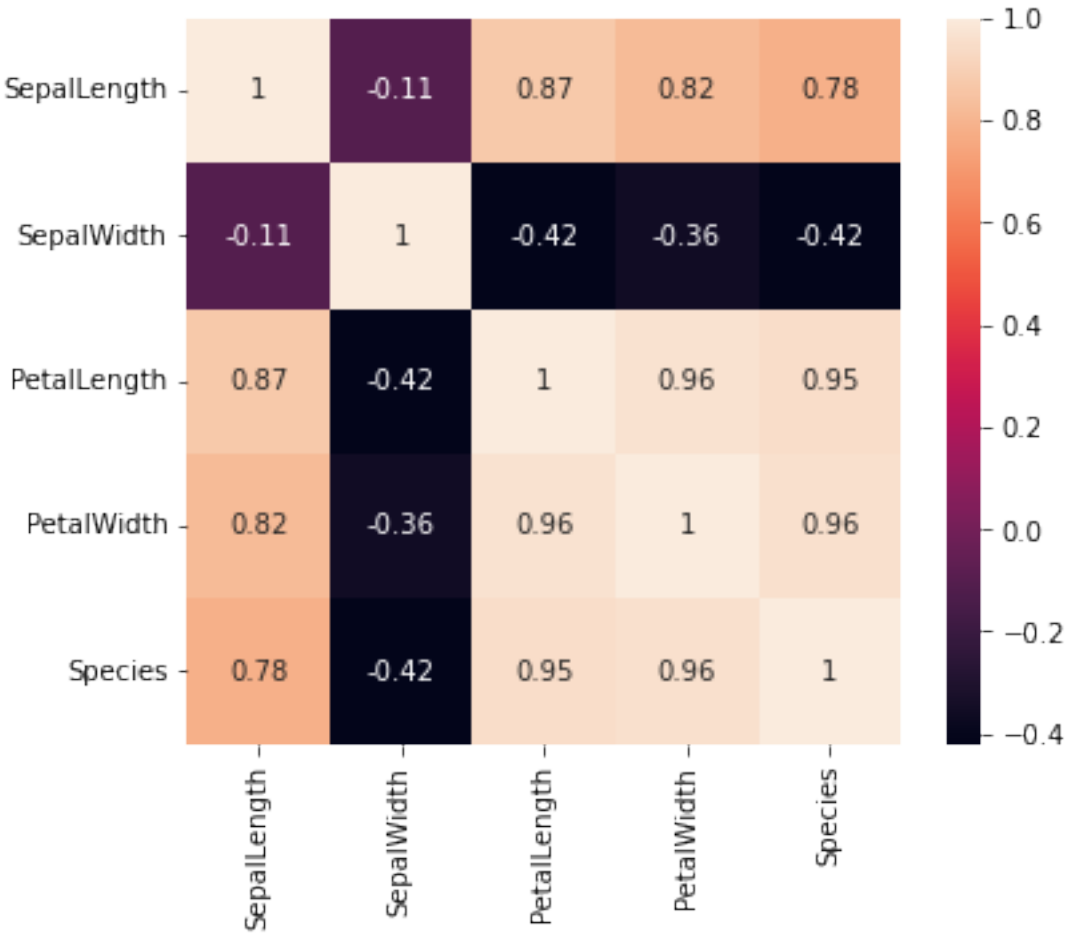
[18]:
```
df.corr()
```

[18]:

	SepalLength	SepalWidth	PetalLength	PetalWidth
SepalLength	1.000000	-0.109369	0.871754	0.817954
SepalWidth	-0.109369	1.000000	-0.420516	-0.356544
PetalLength	0.871754	-0.420516	1.000000	0.962757
PetalWidth	0.817954	-0.356544	0.962757	1.000000

[89]:
```
corr = df.corr()
fig,ax = plt.subplots(figsize=(6,5))
sn.heatmap(corr,annot=True,ax=ax,)
```

[89]:
```
<AxesSubplot:>
```



We can see both from the matrix and the heatmap that the value of the correlation between the variables are not very extreme, and therefore we do not need to neglect any of them.

1.4 Label encoder

[32]:
```
#converting the class from the label name to 0,1,2
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()

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

[32]:

	SepalLength	SepalWidth	PetalLength	PetalWidth	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

1.5 Spliting the data

[72]:
```
from sklearn.model_selection import train_test_split

#Spliting the data into: training- 70% and testing- 40%

X=df.drop(columns=['Species'])
Y = df['Species']
x_train,x_test,y_train,y_test = train_test_split(X,Y,test_size=0.45)
```

1.6 Logistic regression classifier

[82]:
```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
```

[83]:
```
model.fit(x_train,y_train)
```

[83]:
```
LogisticRegression()
```

[84]:
```
print(f"Accuracy: {model.score(x_test,y_test)*100:3.3f} %")
```

```
Accuracy: 95.588 %
```

1.7 kNN classifier

[85]:
```
from sklearn.neighbors import KNeighborsClassifier
modelkNN = KNeighborsClassifier()
```

[86]:
```
modelkNN.fit(x_train,y_train)
print(f"Accuracy: {modelkNN.score(x_test,y_test)*100:3.3f} %")
```

```
Accuracy: 95.588 %
```