task1 OASIS

May 16, 2023

[1]:

```
from mpl_toolkits.mplot3d import Axes3D
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt # plotting
import numpy as np # linear algebra
import os # accessing directory structure
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import seaborn as sns
```

[2]:

```
nRowsRead = 1000 # specify 'None' if want to read whole file
# Iris_Data.csv may have more rows in reality, but we are only loading/
 ↪previewing the first 1000 rows
df = pd.read_csv('Iris.csv', delimiter=',', nrows = nRowsRead)
df.dataframeName = 'Iris.csv'
print(df)
nRow, nCol = df.shape
print(f'There are {nRow} rows and {nCol} columns')
```

```
      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]
There are 150 rows and 6 columns
```

[3]:

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

[3]:

```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

[4]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             150 non-null    int64
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

[5]:

```
df.head(5)   # before preprocessing
```

[5]:

```
   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
```

[6]:

```
df.dropna(how='any',inplace=True)
print(df)
```

```
      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]
```

[7]:

```
# drop column2 from the dataset
df = df.drop(columns='Id', axis=1)

# print the resulting dataset
print(df)
```

```
     SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm         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
..             …           …            …           …             …
145            6.7           3.0            5.2           2.3  Iris-virginica
146            6.3           2.5            5.0           1.9  Iris-virginica
147            6.5           3.0            5.2           2.0  Iris-virginica
148            6.2           3.4            5.4           2.3  Iris-virginica
149            5.9           3.0            5.1           1.8  Iris-virginica

[150 rows x 5 columns]
```

[8]:

```
df.head(5)
```

[8]:

```
   SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm      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
```

[9]:

```
df.tail(5)
```

[9]:

```
     SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm         Species
145            6.7           3.0            5.2           2.3  Iris-virginica
146            6.3           2.5            5.0           1.9  Iris-virginica
147            6.5           3.0            5.2           2.0  Iris-virginica
148            6.2           3.4            5.4           2.3  Iris-virginica
149            5.9           3.0            5.1           1.8  Iris-virginica
```

[10]:

```
df.dtypes
```

[10]:

```
SepalLengthCm    float64
SepalWidthCm     float64
PetalLengthCm    float64
PetalWidthCm     float64
Species           object
dtype: object
```

[11]:

```
df.index
```

[11]:

```
RangeIndex(start=0, stop=150, step=1)
```

[12]:

```
df.describe()
```

[12]:

```
       SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
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]:

```
df.nunique()
```

[13]:

```
SepalLengthCm    35
SepalWidthCm     23
```

```
PetalLengthCm    43
PetalWidthCm     22
Species           3
dtype: int64
```

```
[14]: df.shape
```

```
[14]: (150, 5)
```

```
[15]: X = df.drop(['Species'], axis=1)

      y = df['Species']
      print("okay")
```

```
okay
```

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

      X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.33,␣
       ↪random_state = 42)
```

```
[17]: X_train.shape, X_test.shape
```

```
[17]: ((100, 4), (50, 4))
```

```
[18]: X_train.dtypes
```

```
[18]: SepalLengthCm    float64
      SepalWidthCm     float64
      PetalLengthCm    float64
      PetalWidthCm     float64
      dtype: object
```

```
[19]: df.head(5)
```

```
[19]:    SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm      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
```

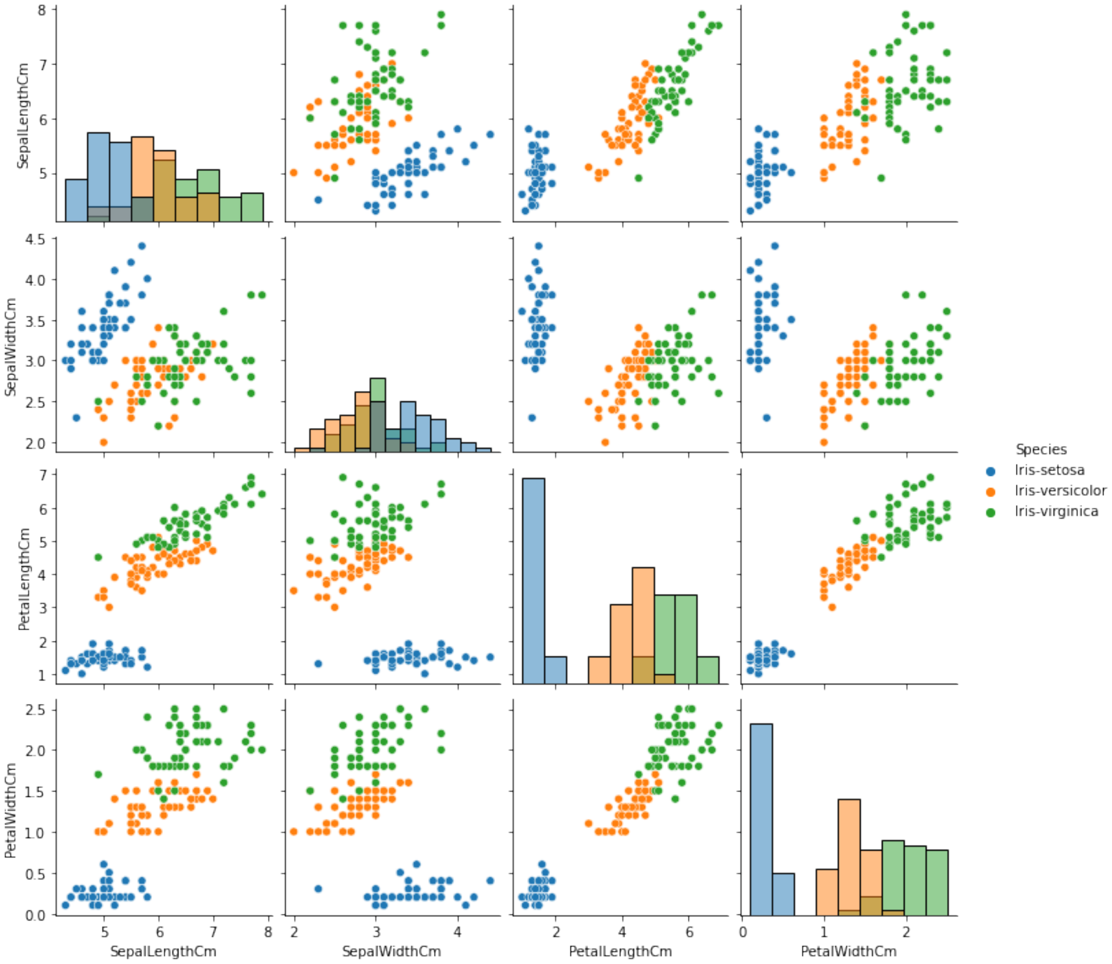
```
[20]: X_train.head(3)
```

```
[20]:      SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
      96             5.7           2.9            4.2           1.3
      105            7.6           3.0            6.6           2.1
```

```
66              5.6           3.0            4.5           1.5
```

[21]:
```python
sns.pairplot(df,hue="Species",diag_kind="hist") #before mapping
```

[21]:
```
<seaborn.axisgrid.PairGrid at 0x7faf4abb49a0>
```



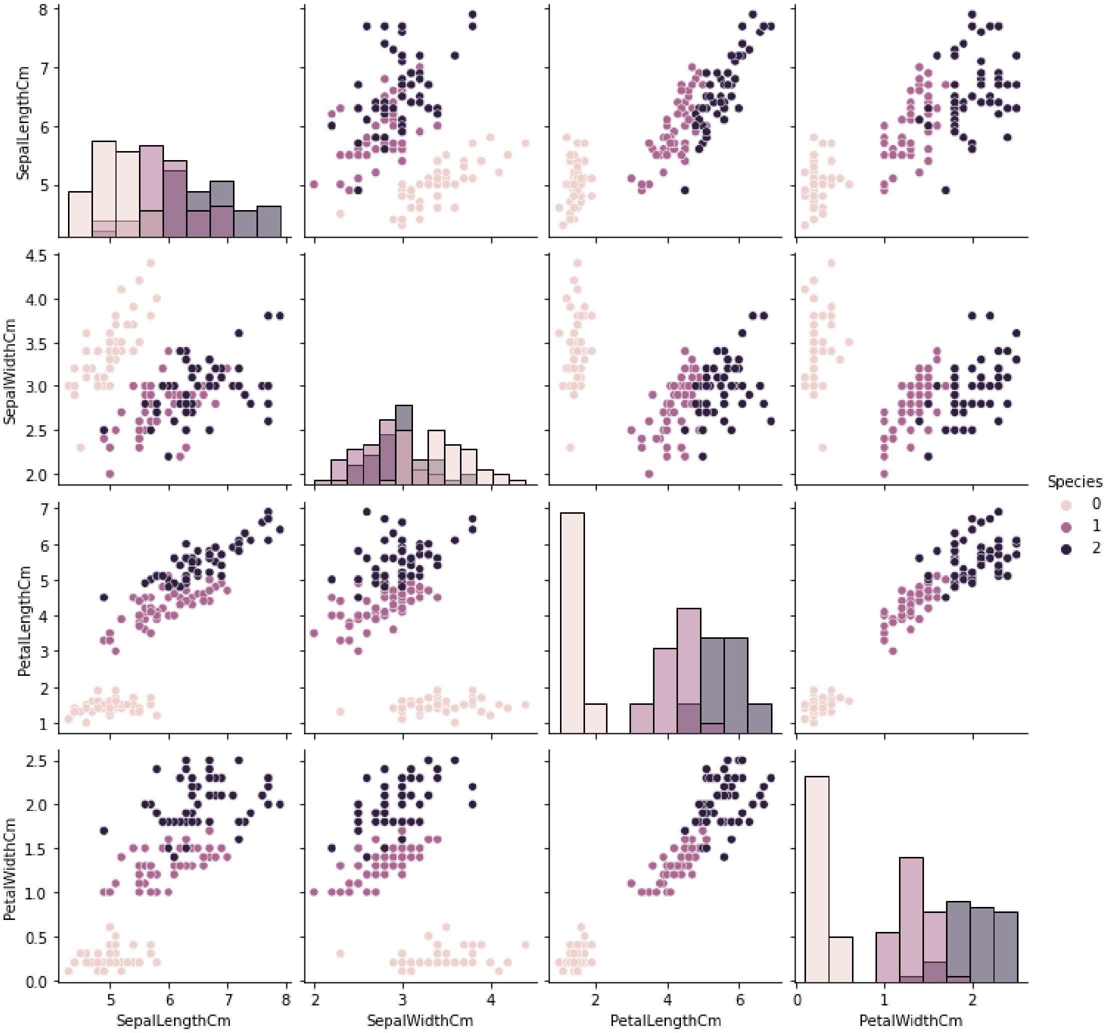
[22]:
```python
values={'Iris-setosa':0,'Iris-versicolor':1,'Iris-virginica':2}
df["Species"]=df["Species"].map(values)
print(df)
# using mapping function to convert categorical values to numerical values in the target variable SPECIES column
```

```
     SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm  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
```

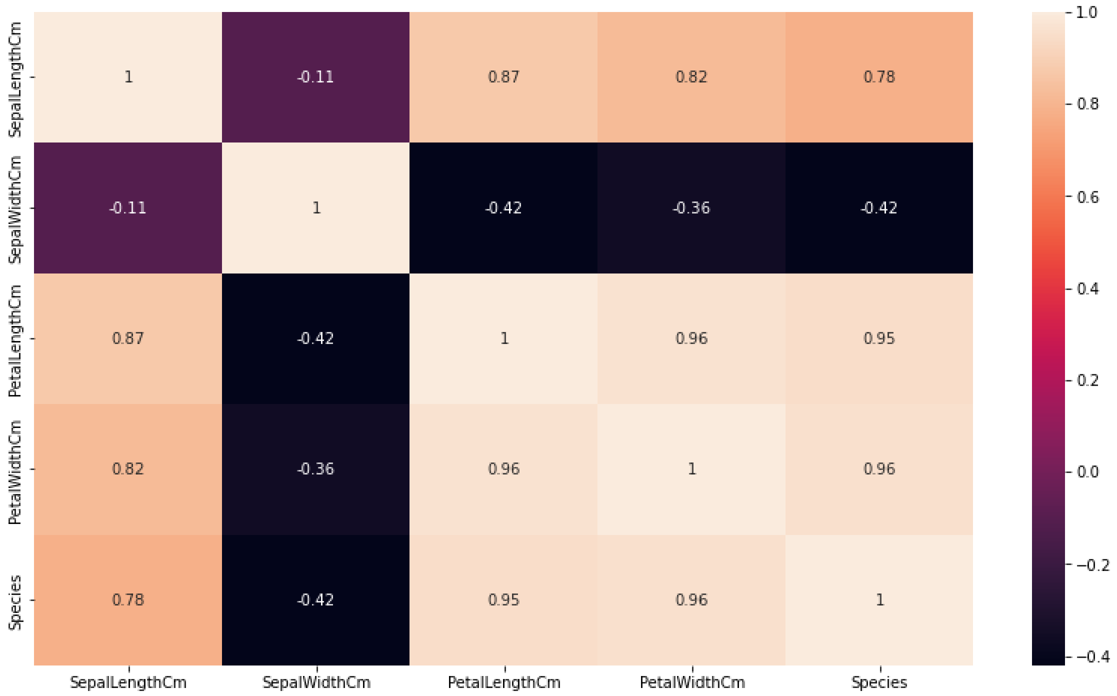
```
4            5.0          3.6          1.4          0.2         0
..           …            …            …            …           …
145          6.7          3.0          5.2          2.3         2
146          6.3          2.5          5.0          1.9         2
147          6.5          3.0          5.2          2.0         2
148          6.2          3.4          5.4          2.3         2
149          5.9          3.0          5.1          1.8         2

[150 rows x 5 columns]
```

[23]:
```
sns.pairplot(df,hue="Species",diag_kind="hist")   #after mapping
```

[23]:
```
<seaborn.axisgrid.PairGrid at 0x7faf4ec75e80>
```



[24]:
```
import seaborn
correlation = df.corr ()
fig=plt.figure(figsize=(14,8))
seaborn.heatmap(correlation,annot=True)
plt.show()
```



[25]:
```
correlation = df.corr ()
correlation.style.background_gradient (cmap = 'BrBG')
```

[25]:
```
<pandas.io.formats.style.Styler at 0x7faf4853e8b0>
```

[26]:
```
df.corr()
```

[26]:
```
               SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm  \
SepalLengthCm       1.000000     -0.109369       0.871754      0.817954
SepalWidthCm       -0.109369      1.000000      -0.420516     -0.356544
PetalLengthCm       0.871754     -0.420516       1.000000      0.962757
PetalWidthCm        0.817954     -0.356544       0.962757      1.000000
Species             0.782561     -0.419446       0.949043      0.956464

                Species
SepalLengthCm  0.782561
SepalWidthCm  -0.419446
PetalLengthCm  0.949043
```

```
PetalWidthCm    0.956464
Species         1.000000
```

[27]:

```
#Import Libraries file

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

from sklearn.model_selection import train_test_split #Train Test Split

from sklearn.naive_bayes import GaussianNB   # Naive Bayes Classifier

from sklearn import preprocessing            # Label Encoder

from sklearn.neighbors import KNeighborsClassifier  # KNN Classsifiers
```

[28]:

```
#Train Test split

x = df[['SepalLengthCm','SepalWidthCm','PetalLengthCm','PetalWidthCm']]
y= df['Species']

x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.30,␣
 ↪random_state=0)

x_train.shape
```

[28]:

```
(105, 4)
```

[29]:

```
#Import Gaussian Naive Bayes model
from sklearn.naive_bayes import GaussianNB
#Create a Gaussian Classifier
gnb = GaussianNB()
#Train the model using the training sets
gnb.fit(x_train, y_train)
```

[29]:

```
GaussianNB()
```

[30]:

```
#Predict the response for test dataset
y_pred = gnb.predict(x_test)
```

[31]:

```
# Evaluating model
#Import scikit-learn metrics module for accuracy calculation
from sklearn import metrics
# Model Accuracy
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 1.0
```

[32]:
```
# Evaluating model
#Import scikit-learn metrics module for accuracy calculation
from sklearn import metrics
# Model Accuracy
print("Accuracy:",metrics.classification_report(y_test, y_pred))
```

```
Accuracy:               precision    recall  f1-score   support

           0       1.00      1.00      1.00        16
           1       1.00      1.00      1.00        18
           2       1.00      1.00      1.00        11

    accuracy                           1.00        45
   macro avg       1.00      1.00      1.00        45
weighted avg       1.00      1.00      1.00        45
```

[ ]:
```

```

[ ]:
```

```