ks-task-2-classify-iris-flowers-1

January 27, 2024

[1]:
```
#Import Libraries/Packages
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

[2]:
```
import pandas as pd
# Load the csv file into a DataFrame
dataset = pd.read_csv("Iris.csv")
dataset
```

[2]:
```
      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]
```

[3]:
```
# Shape of Dataset
dataset.shape
```

[3]:
```
(150, 6)
```

[4]:
```
# Display the first few rows of the DataFrame
dataset.head()
```

[4]:
```
   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
```

[5]:
```
# Dataset Columns
dataset.columns
```

[5]:
```
Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm',
       'Species'],
      dtype='object')
```

[6]:
```
#Checking Null Values
dataset.isnull().sum()
```

[6]:
```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

[7]:
```
#Dataset Summary
dataset.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
```

[8]:
```
#Dataset Statistical Summary
dataset.describe()
```

[8]:

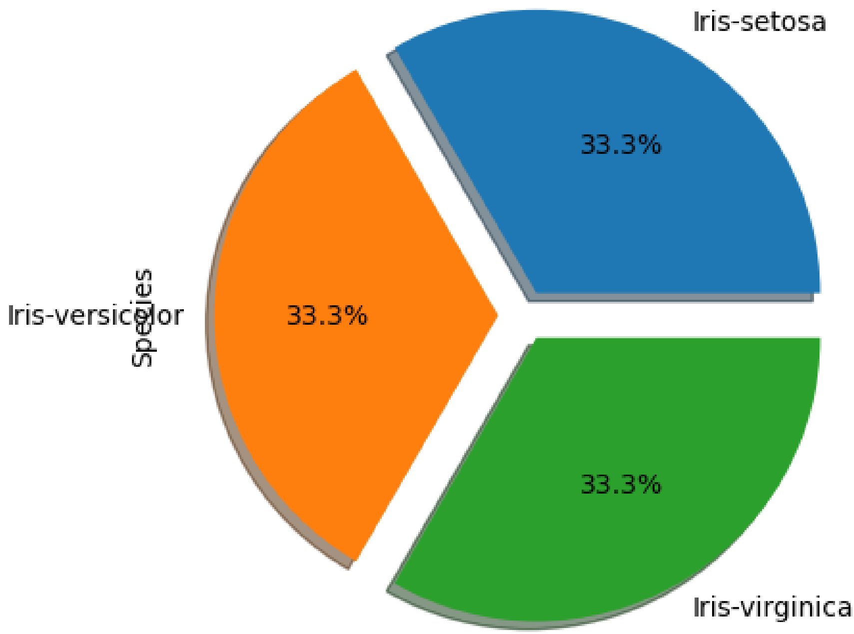
	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	75.500000	5.843333	3.054000	3.758667	1.198667
std	43.445368	0.828066	0.433594	1.764420	0.763161
min	1.000000	4.300000	2.000000	1.000000	0.100000
25%	38.250000	5.100000	2.800000	1.600000	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

[9]:
```
# To display no. of samples on each class.
dataset['Species'].value_counts()
```

[9]:
```
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: Species, dtype: int64
```

[13]:
```
dataset['Species'].value_counts().plot(kind = 'pie', autopct = '%1.1f%%',shadow␣
 ↪= True, explode = [0.09,0.09,0.09])
```

[13]:
```
<Axes: ylabel='Species'>
```

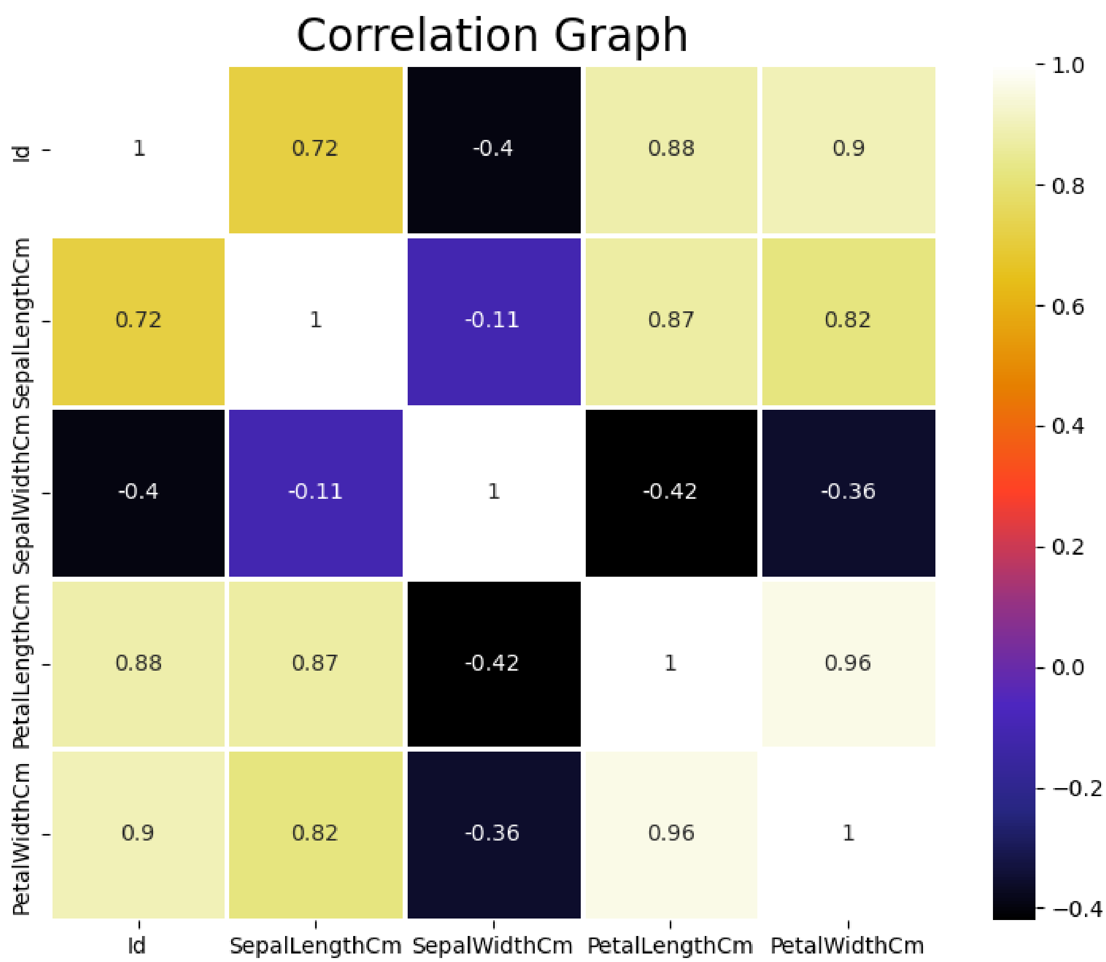


[12]:
```
dataset.corr()
```

[12]:
```
                     Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  \
Id             1.000000       0.716676     -0.397729       0.882747
SepalLengthCm  0.716676       1.000000     -0.109369       0.871754
SepalWidthCm  -0.397729      -0.109369      1.000000      -0.420516
PetalLengthCm  0.882747       0.871754     -0.420516       1.000000
PetalWidthCm   0.899759       0.817954     -0.356544       0.962757

               PetalWidthCm
Id                 0.899759
SepalLengthCm      0.817954
SepalWidthCm      -0.356544
PetalLengthCm      0.962757
PetalWidthCm       1.000000
```

[14]:
```
import seaborn as sns
import matplotlib.pyplot as plt
plt.figure(figsize=(9, 7))
sns.heatmap(dataset.corr(), cmap='CMRmap', annot=True, linewidths=2)
plt.title("Correlation Graph", size=20)
plt.show()
```



[15]:
```
#Label encoding for categorical variables
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
dataset['Species'] = le.fit_transform(dataset['Species'])
dataset.head()
```

[15]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	0
1	2	4.9	3.0	1.4	0.2	0
2	3	4.7	3.2	1.3	0.2	0
3	4	4.6	3.1	1.5	0.2	0
4	5	5.0	3.6	1.4	0.2	0

[16]:
```
dataset['Species'].unique()
```

[16]: array([0, 1, 2])

[18]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
# Load the CSV file into a DataFrame
df = pd.read_csv("Iris.csv")
features = ['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm']
X = df.loc[:, features].values
Y = df['Species'].values
# Split the dataset into training and test sets
X_Train, X_Test, Y_Train, Y_Test = train_test_split(X, Y, test_size=0.
 ↪2,random_state=0)
print(X_Train.shape)
```

```
(120, 4)
```

[19]:
```
Y_Train.shape
```

[19]:
```
(120,)
```

[20]:
```
X_Test.shape
```

[20]:
```
(30, 4)
```

[21]:
```
Y_Test.shape
```

[21]:
```
(30,)
```

[22]:
```
# Feature Scaling to bring all the variables in a single scale.
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_Train = sc.fit_transform(X_Train)
X_Test = sc.transform(X_Test)
# Importing some metrics for evaluating models.
from sklearn import metrics
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
```

[23]:
```
from sklearn.linear_model import LogisticRegression
log_model= LogisticRegression(random_state = 0)
log_model.fit(X_Train, Y_Train)
# model training
log_model.fit(X_Train, Y_Train)
# Predicting
Y_Pred_Test_log_res=log_model.predict(X_Test)
print(Y_Pred_Test_log_res)
log_model = LogisticRegression(random_state=0, max_iter=100)
```

```
['Iris-virginica' 'Iris-versicolor' 'Iris-setosa' 'Iris-virginica'
 'Iris-setosa' 'Iris-virginica' 'Iris-setosa' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-virginica' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor' 'Iris-setosa'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-setosa' 'Iris-setosa'
 'Iris-virginica' 'Iris-versicolor' 'Iris-setosa' 'Iris-setosa'
 'Iris-virginica' 'Iris-setosa' 'Iris-setosa' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-setosa']
```

[24]:
```
from sklearn import metrics
accuracy = metrics.accuracy_score(Y_Test, Y_Pred_Test_log_res)
print("Accuracy:", accuracy * 100)
```

```
Accuracy: 100.0
```

[25]:
```
from sklearn.metrics import classification_report
print(classification_report(Y_Test, Y_Pred_Test_log_res))
```

```
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        11
Iris-versicolor       1.00      1.00      1.00        13
 Iris-virginica       1.00      1.00      1.00         6

       accuracy                           1.00        30
      macro avg       1.00      1.00      1.00        30
   weighted avg       1.00      1.00      1.00        30
```

[26]:
```
from sklearn.metrics import confusion_matrix
confusion_matrix(Y_Test,Y_Pred_Test_log_res )
```

[26]:
```
array([[11,  0,  0],
       [ 0, 13,  0],
       [ 0,  0,  6]], dtype=int64)
```

[28]:
```
from sklearn.neighbors import KNeighborsClassifier
knn_model = KNeighborsClassifier(n_neighbors=3,␣
 ↪weights='distance',algorithm='auto')
# Importing KNeighborsClassifier
from sklearn.neighbors import KNeighborsClassifier
knn_model = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p = 2)
# model training
knn_model.fit(X_Train, Y_Train)
# Predicting
Y_Pred_Test_knn=knn_model.predict(X_Test)
# model training
log_model.fit(X_Train, Y_Train)
```

[28]: LogisticRegression(random_state=0)

[29]:
```
Y_Pred_Test_knn
```

[29]:
```
array(['Iris-virginica', 'Iris-versicolor', 'Iris-setosa',
       'Iris-virginica', 'Iris-setosa', 'Iris-virginica', 'Iris-setosa',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-virginica', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-setosa',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-setosa', 'Iris-setosa',
       'Iris-virginica', 'Iris-versicolor', 'Iris-setosa', 'Iris-setosa',
       'Iris-virginica', 'Iris-setosa', 'Iris-setosa', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-setosa'], dtype=object)
```

[30]:
```
print("Accuracy:",metrics.accuracy_score(Y_Test,Y_Pred_Test_knn)*100)
```

```
Accuracy: 100.0
```

[31]:
```
print(classification_report(Y_Test,Y_Pred_Test_knn))
```

```
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        11
Iris-versicolor       1.00      1.00      1.00        13
 Iris-virginica       1.00      1.00      1.00         6

       accuracy                           1.00        30
      macro avg       1.00      1.00      1.00        30
   weighted avg       1.00      1.00      1.00        30
```

[32]:
```
confusion_matrix(Y_Test, Y_Pred_Test_knn)
```

[32]:
```
array([[11,  0,  0],
       [ 0, 13,  0],
       [ 0,  0,  6]], dtype=int64)
```

[33]:
```
from sklearn.tree import DecisionTreeClassifier
dec_tree␣
↪=DecisionTreeClassifier(criterion='entropy',splitter='best',max_depth=6)
# model training
dec_tree.fit(X_Train, Y_Train)
# Predicting
Y_Pred_Test_dtr=dec_tree.predict(X_Test)
Y_Pred_Test_dtr
```

[33]:
```
array(['Iris-virginica', 'Iris-versicolor', 'Iris-setosa',
       'Iris-virginica', 'Iris-setosa', 'Iris-virginica', 'Iris-setosa',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-virginica', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-setosa',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-setosa', 'Iris-setosa',
       'Iris-virginica', 'Iris-versicolor', 'Iris-setosa', 'Iris-setosa',
       'Iris-virginica', 'Iris-setosa', 'Iris-setosa', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-setosa'], dtype=object)
```

[34]:
```
print("Accuracy:",metrics.accuracy_score(Y_Test, Y_Pred_Test_dtr)*100)
print(classification_report(Y_Test, Y_Pred_Test_dtr))
```

```
Accuracy: 100.0
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        11
Iris-versicolor       1.00      1.00      1.00        13
 Iris-virginica       1.00      1.00      1.00         6

       accuracy                           1.00        30
      macro avg       1.00      1.00      1.00        30
   weighted avg       1.00      1.00      1.00        30
```

[35]:
```
confusion_matrix(Y_Test, Y_Pred_Test_dtr)
```

[35]:
```
array([[11,  0,  0],
       [ 0, 13,  0],
       [ 0,  0,  6]], dtype=int64)
```

[36]:
```
from sklearn.naive_bayes import GaussianNB
nav_byes = GaussianNB()
# model training
nav_byes.fit(X_Train, Y_Train)
# Predicting
Y_Pred_Test_nvb=nav_byes.predict(X_Test)
Y_Pred_Test_nvb
```

[36]:
```
array(['Iris-virginica', 'Iris-versicolor', 'Iris-setosa',
       'Iris-virginica', 'Iris-setosa', 'Iris-virginica', 'Iris-setosa',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-setosa',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-setosa', 'Iris-setosa',
       'Iris-virginica', 'Iris-versicolor', 'Iris-setosa', 'Iris-setosa',
       'Iris-virginica', 'Iris-setosa', 'Iris-setosa', 'Iris-versicolor',
```

```
       'Iris-versicolor', 'Iris-setosa'], dtype='<U15')
```

[37]:
```
print("Accuracy:",metrics.accuracy_score(Y_Test, Y_Pred_Test_nvb)*100)
print(classification_report(Y_Test, Y_Pred_Test_nvb))
```

```
Accuracy: 96.66666666666667
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        11
Iris-versicolor       0.93      1.00      0.96        13
 Iris-virginica       1.00      0.83      0.91         6

       accuracy                           0.97        30
      macro avg       0.98      0.94      0.96        30
   weighted avg       0.97      0.97      0.97        30
```

[38]:
```
confusion_matrix(Y_Test,Y_Pred_Test_nvb )
```

[38]:
```
array([[11,  0,  0],
       [ 0, 13,  0],
       [ 0,  1,  5]], dtype=int64)
```

[39]:
```
from sklearn.svm import SVC
svm_model=SVC(C=500, kernel='rbf')
# model training
svm_model.fit(X_Train, Y_Train)
# Predicting
Y_Pred_Test_svm=svm_model.predict(X_Test)
Y_Pred_Test_svm
```

[39]:
```
array(['Iris-virginica', 'Iris-versicolor', 'Iris-setosa',
       'Iris-virginica', 'Iris-setosa', 'Iris-virginica', 'Iris-setosa',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-virginica', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-setosa',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-setosa', 'Iris-setosa',
       'Iris-virginica', 'Iris-versicolor', 'Iris-setosa', 'Iris-setosa',
       'Iris-virginica', 'Iris-setosa', 'Iris-setosa', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-setosa'], dtype=object)
```

[40]:
```
print("Accuracy:",metrics.accuracy_score(Y_Test,Y_Pred_Test_svm)*100)
print(classification_report(Y_Test, Y_Pred_Test_svm))
```

```
Accuracy: 100.0
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        11
```

```
Iris-versicolor       1.00      1.00      1.00        13
 Iris-virginica       1.00      1.00      1.00         6

       accuracy                           1.00        30
      macro avg       1.00      1.00      1.00        30
   weighted avg       1.00      1.00      1.00        30
```

[41]:
```
confusion_matrix(Y_Test,Y_Pred_Test_svm )
```

[41]:
```
array([[11,  0,  0],
       [ 0, 13,  0],
       [ 0,  0,  6]], dtype=int64)
```

[42]:
```
print("Accuracy of Logistic Regression Model:",metrics.
 ↪accuracy_score(Y_Test,Y_Pred_Test_log_res)*100)
print("Accuracy of KNN Model:",metrics.
 ↪accuracy_score(Y_Test,Y_Pred_Test_knn)*100)
print("Accuracy of Decision Tree Model:",metrics.
 ↪accuracy_score(Y_Test,Y_Pred_Test_dtr)*100)
print("Accuracy of Naive Bayes Model:",metrics.
 ↪accuracy_score(Y_Test,Y_Pred_Test_nvb)*100)
print("Accuracy of SVM Model:",metrics.
 ↪accuracy_score(Y_Test,Y_Pred_Test_svm)*100)
```

```
Accuracy of Logistic Regression Model: 100.0
Accuracy of KNN Model: 100.0
Accuracy of Decision Tree Model: 100.0
Accuracy of Naive Bayes Model: 96.66666666666667
Accuracy of SVM Model: 100.0
```

[ ]:
```

```