AKSHAT

+ Code + Text

Task 1 by Oasis Infobyte

```
#loading the required file on the colab server
#importing the required libraries

import pandas as pd
import numpy as np
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn. metrics import accuracy_score
from sklearn.metrics import classification_report


#read and load the csv file
raw_csv_data = pd.read_csv("Iris.csv")
df = raw_csv_data.copy()
```

Understand the data

```
df.head()
# displaying top 5 rows from the start
```

	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

```
df.tail()
# displaying bottom 5 rows
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

```
df.shape
```

```
(150, 6)
```

```
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
```

```
df.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

```
df.columns
```

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

```
df.nunique()
```

```
Id               150
SepalLengthCm     35
SepalWidthCm      23
PetalLengthCm     43
PetalWidthCm      22
Species            3
dtype: int64
```

Cleaning the Data

```
df.isna().sum()
```

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

```
#drop the unnecessary column
df_comp = df.drop('Id',axis = 1)
```

```
df_comp.shape
```

```
(150, 5)
```

```
df_comp.head()
```

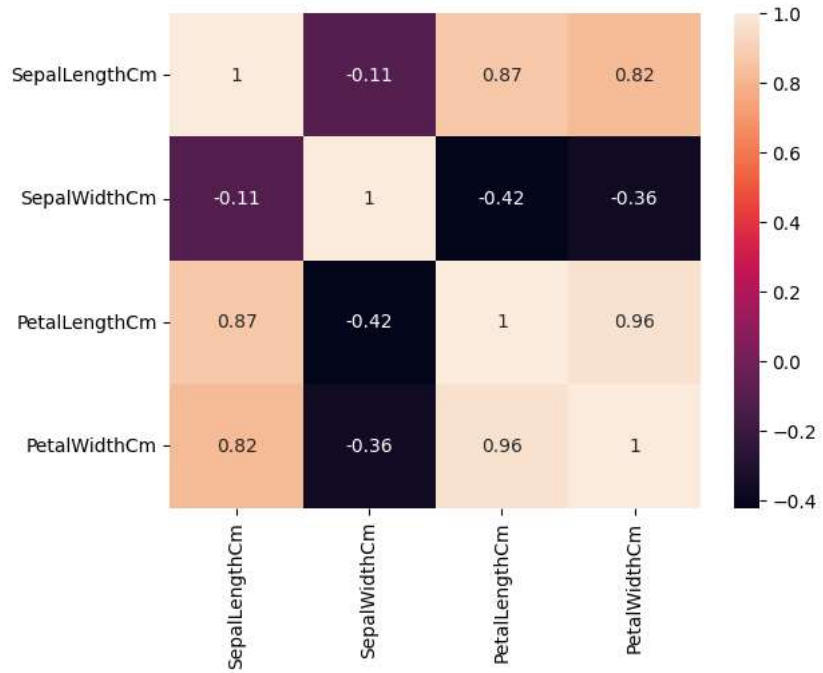
	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

Exploratory Data Analysis

```
#select only numeric columns
numeric_cols = df_comp.select_dtypes(include = ['float']).columns
correlation = df[numeric_cols].corr()
```

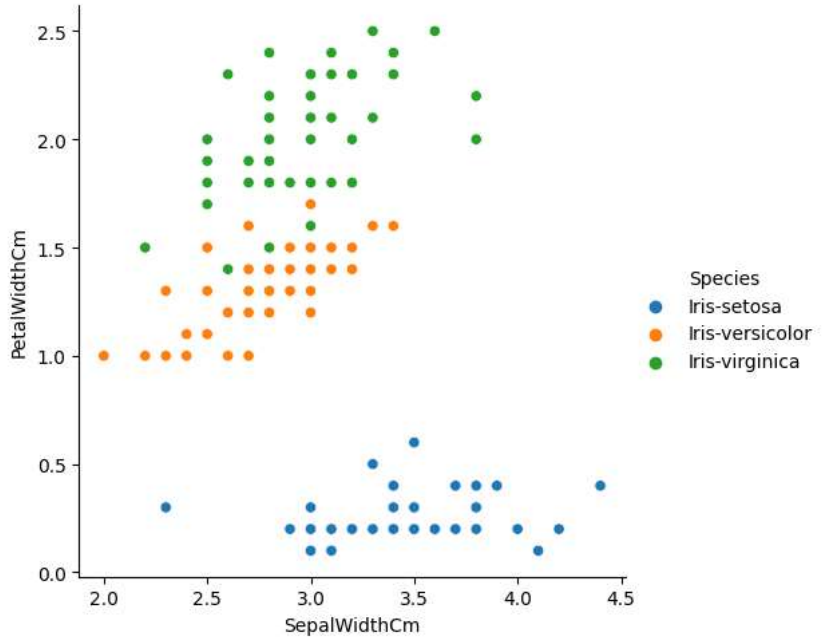
```
#plotting correlation heatmap
sns.heatmap(correlation, annot = True)
```

```
<Axes: >
```




```
sns.relplot(x='SepalWidthCm', y='PetalWidthCm', hue='Species', data= df_comp)
```

```
<seaborn.axisgrid.FacetGrid at 0x7f627597faf0>
```

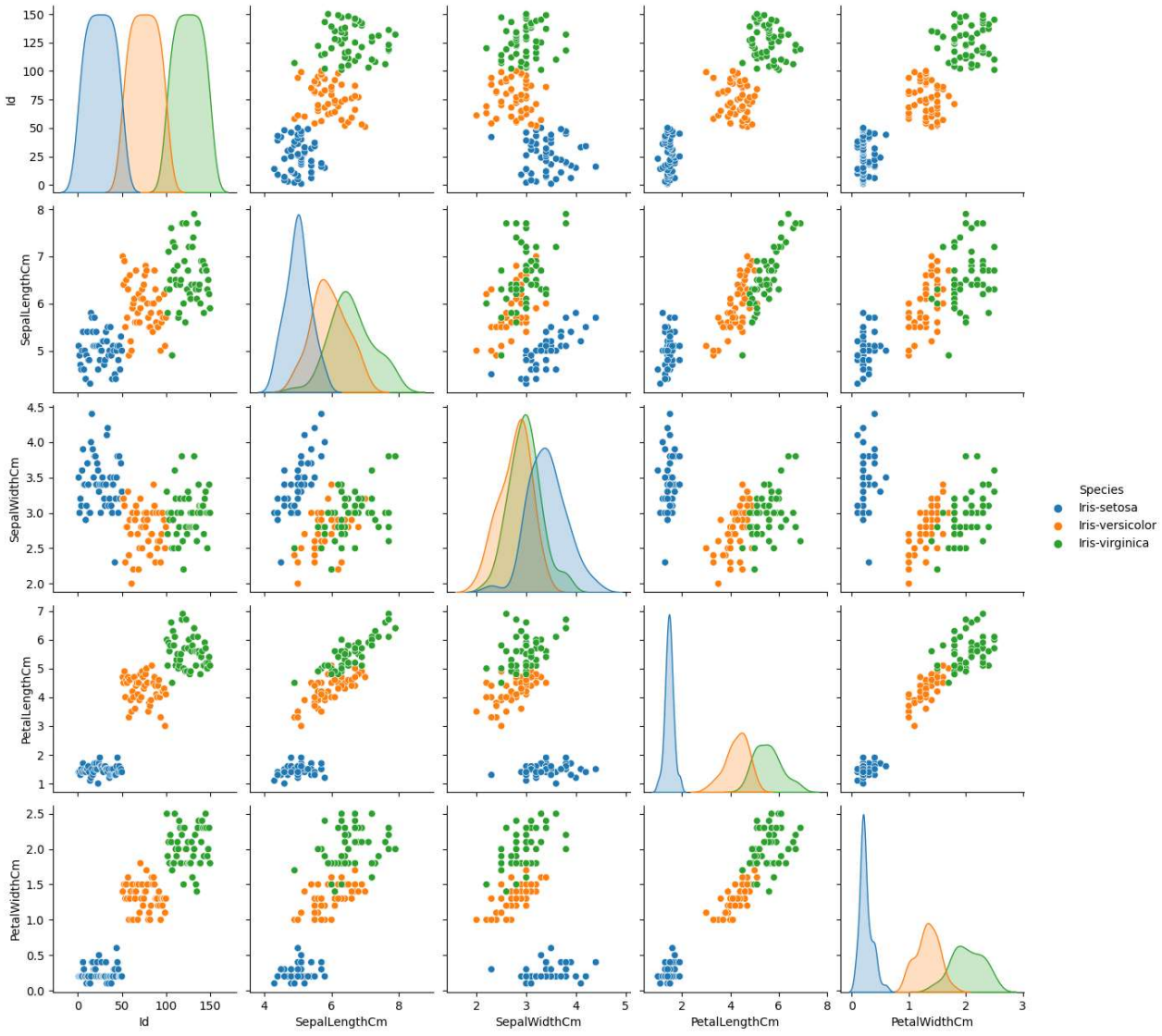



```
sns.relplot(x='SepalLengthCm', y='PetalLengthCm', hue='Species', data= df_comp)
```

```
<seaborn.axisgrid.FacetGrid at 0x7f62759323e0>
```


```
sns.pairplot(df, hue='Species')
```

```
<seaborn.axisgrid.PairGrid at 0x7f6271c1df00>
```



Training Models

```
#slicing
x=df_comp.iloc[:,0:4]
```

```
x.head()
```

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2

```
y = df_comp['Species']
y.head()
```

```
0    Iris-setosa
1    Iris-setosa
2    Iris-setosa
3    Iris-setosa
4    Iris-setosa
Name: Species, dtype: object
```

```
df_comp['Species'].unique()
```

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

```
#splitting data set into train and test
x_train, x_test, y_train, y_test = train_test_split(x,x,test_size=0.20, random_state=42)
```

```
#For training we are using Logistic regression and evaluating the prediction accuracy using accuracy_score
model = LogisticRegression()
model.fit(x,y)
```

```
▾ LogisticRegression
LogisticRegression()
```

```
#predictions
predictions = model.predict(x)
#now compare with the actual data
scores = pd.DataFrame({'Actual':y,'Predictions':predictions})
scores.head()
```

	Actual	Predictions
0	Iris-setosa	Iris-setosa
1	Iris-setosa	Iris-setosa
2	Iris-setosa	Iris-setosa
3	Iris-setosa	Iris-setosa
4	Iris-setosa	Iris-setosa

```
scores.tail()
```

	Actual	Predictions
145	Iris-virginica	Iris-virginica
146	Iris-virginica	Iris-virginica
147	Iris-virginica	Iris-virginica
148	Iris-virginica	Iris-virginica
149	Iris-virginica	Iris-virginica

```
pred_lr = model.predict(x_test)
```

```
print(accuracy_score(y_pred , pred_lr)*100,'%')
```

```
100.0 %
```

```
print(classification_report(y_pred, pred_lr))
```

	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	10
Iris-versicolor	1.00	1.00	1.00	9
Iris-virginica	1.00	1.00	1.00	11
accuracy			1.00	30
macro avg	1.00	1.00	1.00	30
weighted avg	1.00	1.00	1.00	30