LetsGrowMore Data Science Internship

Beginner Level - TASK 1

Iris Flowers Classification ML Project:

BY RUBA ROSHINI S

This particular ML project is usually referred to as the “Hello World” of Machine Learning. The iris flowers dataset contains numeric attributes, and it is perfect for beginners to learn about supervised ML algorithms, mainly how to load and handle data. Also, since this is a small dataset, it can easily fit in memory without requiring special transformations or scaling capabilities.

Importing all the libraries

In [3]:

```
import pandas as pd
import matplotlib.pyplot as plt
get_ipython().run_line_magic('matplotlib', 'inline')
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
```

Loading the dataset

In [4]:

```
df=pd.read_csv("Iris.csv")
df
```

Out[4]:

	sepal_length	sepal_width	petal_length	petal_width	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

	sepal_length	sepal_width	petal_length	petal_width	species
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 × 5 columns

In [3]:
```
df.head()
```

Out[3]:

	sepal_length	sepal_width	petal_length	petal_width	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

Getting the size of the dataset

In [4]:
```
data_size=df.shape
print(f"Number of rows :{data_size[0]}")
print(f"Number of columns :{data_size[1]}")
```

```
Number of rows :150
Number of columns :5
```

In [5]:
```
df.isnull().sum()
```

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

Analyzing and visualizing the dataset

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object 
```

```
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

In [5]:
```
df.describe()
```

Out[5]:

	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

In [6]:
```
df.tail()
```

Out[6]:

	sepal_length	sepal_width	petal_length	petal_width	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

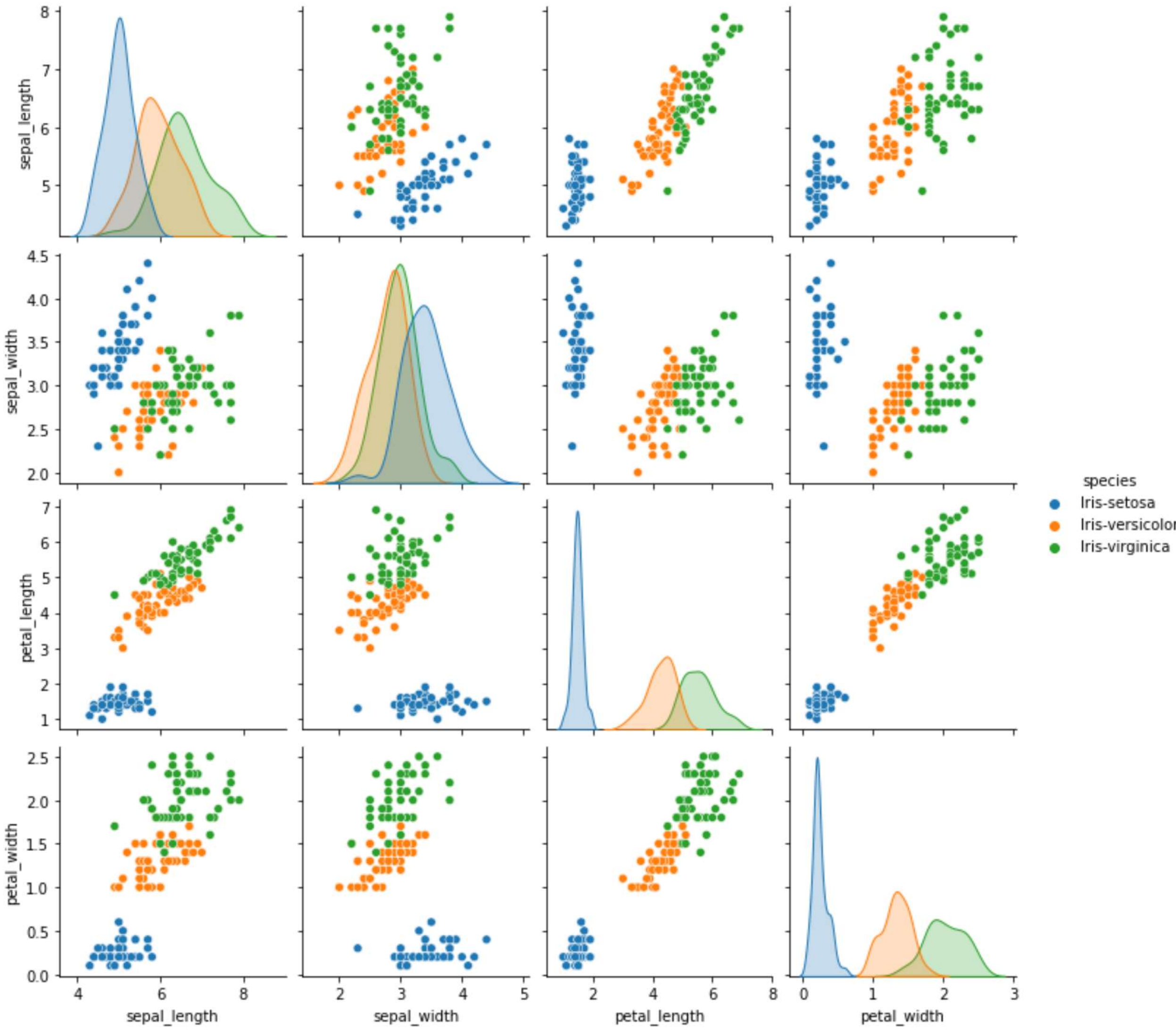
In [7]:
```
df.head()
```

Out[7]:

	sepal_length	sepal_width	petal_length	petal_width	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

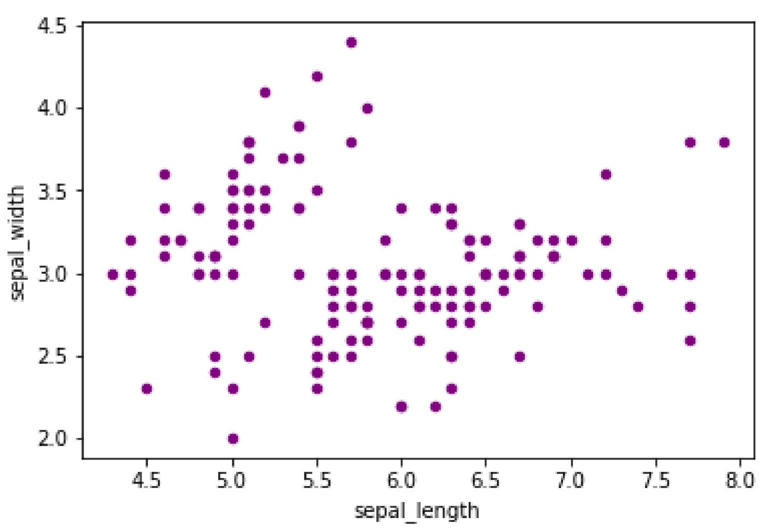
In [8]:
```
sns.pairplot(df, hue='species')
```

Out[8]:
```
<seaborn.axisgrid.PairGrid at 0x1596e6850a0>
```



In [9]:
```
df.plot(kind="scatter", x="sepal_length", y="sepal_width",color="purple", alpha=1)
```

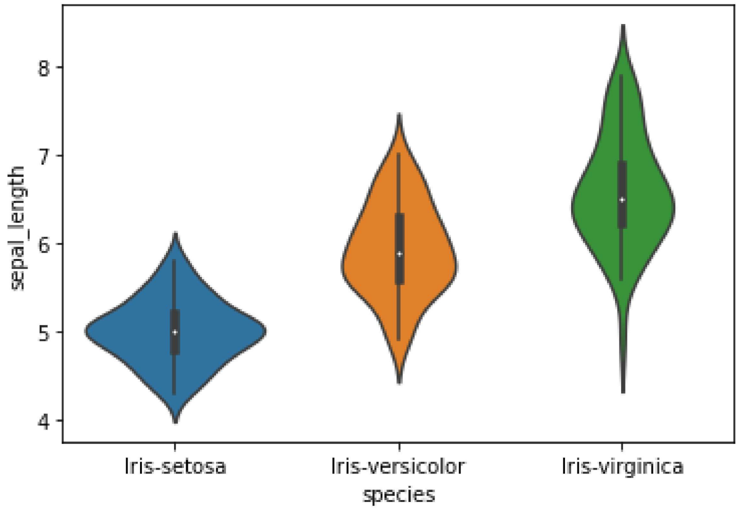
Out[9]: <AxesSubplot:xlabel='sepal_length', ylabel='sepal_width'>



2.Violin Plot

visualise the distribution of the data and its probability density.

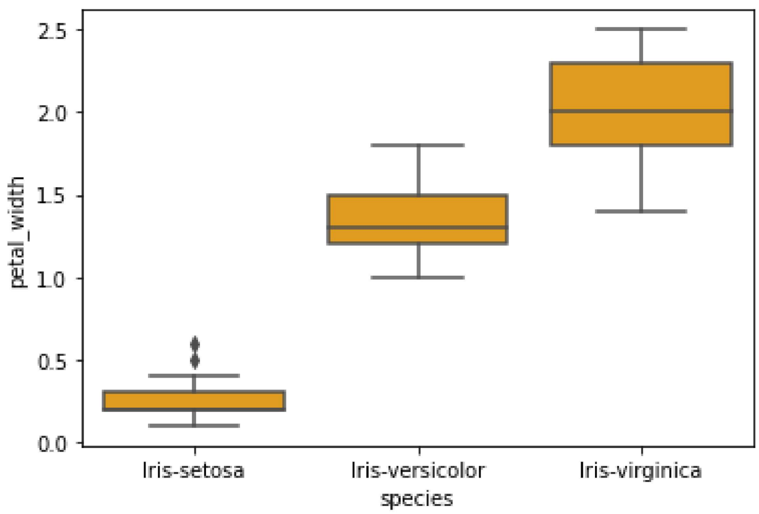
```
In [10]: sns.violinplot(x='species', y='sepal_length', data=df)
         plt.show()
```



3.Box Plot

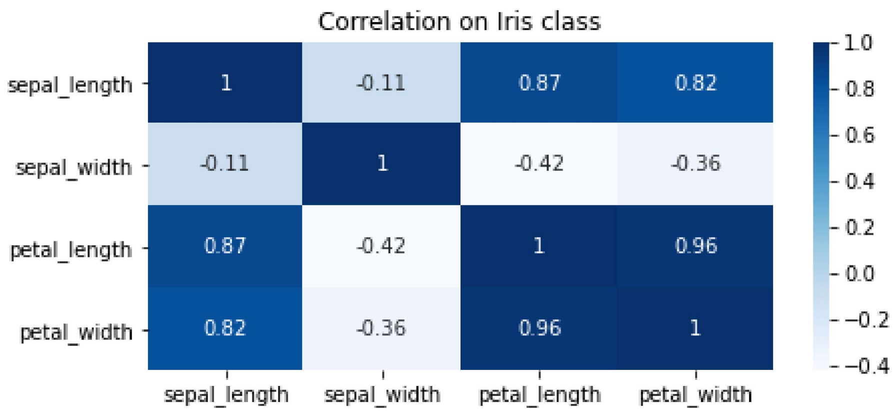
```
In [11]: sns.boxplot(x="species",y="petal_width",data=df,color="orange")
```

```
Out[11]: <AxesSubplot:xlabel='species', ylabel='petal_width'>
```



4.Heat map

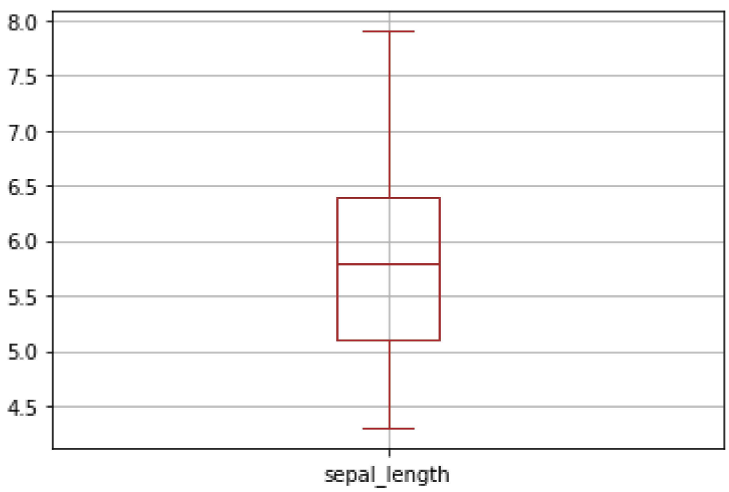
```
In [12]: plt.subplots(figsize = (7,3))
         sns.heatmap(df.corr(),annot=True,cmap="Blues").set_title("Correlation on Iris class")
         plt.show()
```



Checking for the Outliers

In [13]:
```
df.boxplot(column=['sepal_length'],color="brown")
```

Out[13]: <AxesSubplot:>



correlation of the Iris features

In [14]:
```
df.cov()
```

Out[14]:

	sepal_length	sepal_width	petal_length	petal_width
sepal_length	0.685694	-0.039268	1.273682	0.516904
sepal_width	-0.039268	0.188004	-0.321713	-0.117981
petal_length	1.273682	-0.321713	3.113179	1.296387
petal_width	0.516904	-0.117981	1.296387	0.582414

Spliting the dataset

In [15]:
```
x = df.drop(['species'], axis =1)
y = df['species']
```

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

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.4,random_state =0)
```

Logistic Regression

In [17]:
```
log_reg = LogisticRegression()
log_reg.fit(x_train, y_train)
predictions = log_reg.predict(x_test)
print ("Logistic Regression")
print ("The Accuracy Score ", accuracy_score(y_test, predictions))
print (confusion_matrix(y_test, predictions))
print (classification_report(y_test, predictions))
```

```
Logistic Regression
The Accuracy Score  0.9166666666666666
[[16  0  0]
 [ 0 22  1]
 [ 0  4 17]]
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        16
Iris-versicolor       0.85      0.96      0.90        23
 Iris-virginica       0.94      0.81      0.87        21

       accuracy                           0.92        60
      macro avg       0.93      0.92      0.92        60
   weighted avg       0.92      0.92      0.92        60
```

SVM

In [18]:
```
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
from sklearn import svm

model = SVC() # select the svm algorithm
clf = svm.SVC(gamma=0.001, C=100.)

# we train the algorithm with training data and training output
model.fit(x_train, y_train)
clf.fit(x_train, y_train)
# we pass the testing data to the stored algorithm to predict the outcome
prediction = model.predict(x_test)
print("Support Vector Machines")
print('Train-The accuracy of the SVM is: ', accuracy_score(prediction, y_test)) # we ch
#we pass the predicted output by the model and the actual output
```

```
Support Vector Machines
Train-The accuracy of the SVM is:  0.9333333333333333
```

In [19]:
```
# train
model = SVC() # select the svm algorithm

# we train the algorithm with training data and training output
model.fit(x_train, y_train)

prediction = model.predict(x_train)
```

```python
print("Support Vector Machines")
print ("Train-The accuracy of the SVM is:", accuracy_score(y_test, predictions))

#classification report
print (classification_report(y_test, predictions))
```

```
Support Vector Machines
Train-The accuracy of the SVM is: 0.9166666666666666
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        16
Iris-versicolor       0.85      0.96      0.90        23
 Iris-virginica       0.94      0.81      0.87        21

       accuracy                           0.92        60
      macro avg       0.93      0.92      0.92        60
   weighted avg       0.92      0.92      0.92        60
```

In [20]:

```python
#test
print ("Test - Accuracy :", accuracy_score(y_test, clf.predict
(x_test)))
print ("Test-Confusion matrix :\n",confusion_matrix(y_test, clf.
predict(x_test)))
print (classification_report(y_test, predictions))
```

```
Test - Accuracy : 0.9333333333333333
Test-Confusion matrix :
 [[16  0  0]
 [ 0 22  1]
 [ 0  3 18]]
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        16
Iris-versicolor       0.85      0.96      0.90        23
 Iris-virginica       0.94      0.81      0.87        21

       accuracy                           0.92        60
      macro avg       0.93      0.92      0.92        60
   weighted avg       0.92      0.92      0.92        60
```

In [ ]: