LEKKALA GANESH

MACHINE LEARNING INTERN

IRIS FLOWER CLASSIFICATION ML PROJECT

Description- 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.

Data Set Link - http://archive.ics.uci.edu/ml/datasets/Iris

IMPORTING LIBRARIES

In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib .pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [38]:
```
columns = ['Sepal length' , 'Sepal width' , 'Petal length' , 'Petal width' , 'Species']
df = pd.read_csv('iris.data',names=columns)
```

In [39]:
```
df.head()
```

Out[39]:

	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 [40]:
```
df.shape
```

Out[40]:
```
(150, 5)
```

In [41]:
```
df.isnull()
```

Out[41]:

	Sepal length	Sepal width	Petal length	Petal width	Species
0	False	False	False	False	False
1	False	False	False	False	False
2	False	False	False	False	False
3	False	False	False	False	False
4	False	False	False	False	False
...	...	...	...	...	...
145	False	False	False	False	False
146	False	False	False	False	False
147	False	False	False	False	False
148	False	False	False	False	False
149	False	False	False	False	False

150 rows × 5 columns

In [42]:
```
df.describe()
```

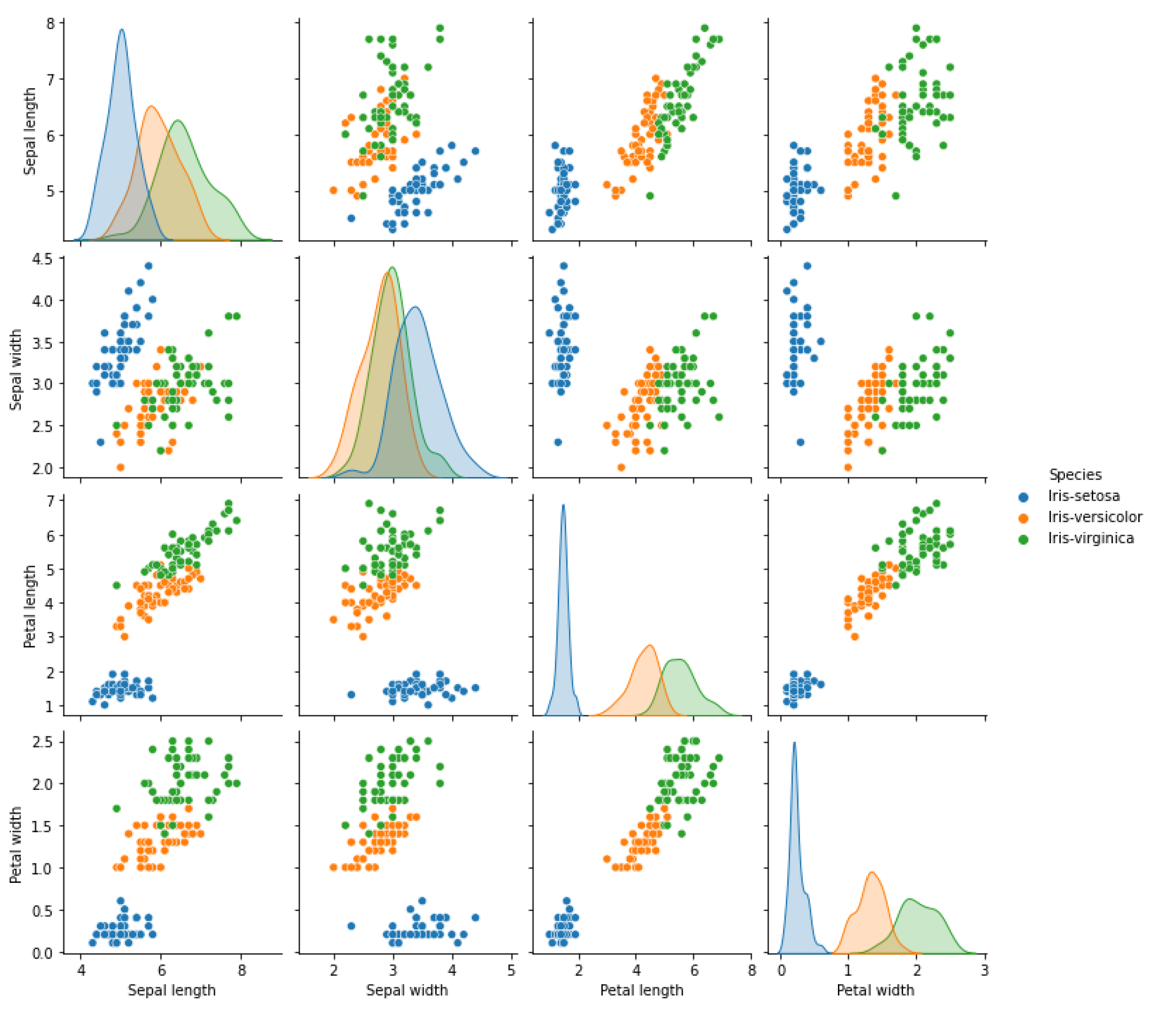
Out[42]:

	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

VISUALISE THE WHOLE DATASET

In [44]:
```
sns.pairplot(df, hue='Species')
```

Out[44]:
```
<seaborn.axisgrid.PairGrid at 0x250e0875e80>
```



In [46]:
```
df.columns
```

Out[46]:
```
Index(['Sepal length', 'Sepal width', 'Petal length', 'Petal width',
       'Species'],
      dtype='object')
```

In [47]:
```
df.nunique()
```

Out[47]:
```
Sepal length    35
Sepal width     23
Petal length    43
Petal width     22
Species          3
dtype: int64
```

In [48]:
```
df.Species.nunique()
```

Out[48]:
```
3
```

In [49]:
```
df.Species.value_counts()
```

Out[49]:
```
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: Species, dtype: int64
```

In [50]:
```
df.max()
```

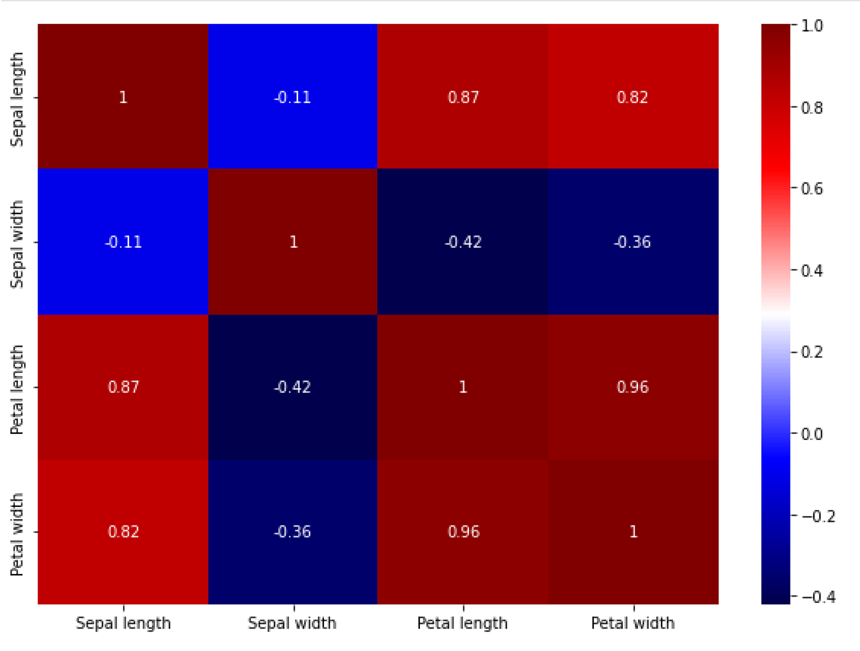
Out[50]:
```
Sepal length                7.9
Sepal width                 4.4
Petal length                6.9
Petal width                 2.5
Species          Iris-virginica
dtype: object
```

In [51]:
```
df.min()
```

Out[51]:
```
Sepal length             4.3
Sepal width              2.0
Petal length             1.0
Petal width              0.1
Species          Iris-setosa
dtype: object
```

DATA PREPROCESSING/CORRELATIONAL MATRIX

In [ ]:
```
plt.figure(figsize=(10,7))
sns.heatmap(df.corr(),annot=True,cmap='seismic')
plt.show()
```



LABEL ENCODER

In [76]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
```

In [81]:
```
df['Species'] = le.fit_transform(df['Species'])
df.head()
```

Out[81]:

	Sepal length	Sepal width	Petal length	Petal width	Species	Sprecies
0	5.1	3.5	1.4	0.2	0	0
1	4.9	3.0	1.4	0.2	0	0
2	4.7	3.2	1.3	0.2	0	0
3	4.6	3.1	1.5	0.2	0	0
4	5.0	3.6	1.4	0.2	0	0

In [82]:
```
X = df.drop(columns=['Species'])
Y = df['Species']
X[:5]
```

Out[82]:

	Sepal length	Sepal width	Petal length	Petal width	Sprecies
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

SEPERATE FEATURES AND TARGET

In [56]:
```
data = df.values
X = data[:,0:4]
Y = data[:,4]
```

CALCULATE AVERAGE OF EACH FEATURES FOR ALL CLASSES

In [67]:
```
Y_Data = np.array([np.average(X[:, i][Y==j].astype('float32')) for i in range (X.shape[1])
 for j in (np.unique(Y))])
Y_Data_reshaped = Y_Data.reshape(4, 3)
Y_Data_reshaped = np.swapaxes(Y_Data_reshaped, 0, 1)
X_axis = np.arange(len(columns)-1)
width = 0.25
```

PLOT THE AVERAGE

In [68]:
```
plt.bar(X_axis, Y_Data_reshaped[0], width, label = 'Setosa')
plt.bar(X_axis+width, Y_Data_reshaped[1], width, label = 'Versicolour')
plt.bar(X_axis+width*2, Y_Data_reshaped[2], width, label = 'Virginica')
plt.xticks(X_axis, columns[:4])
plt.xlabel("Features")
plt.ylabel("Value in cm.")
plt.legend(bbox_to_anchor=(1.3,1))
plt.show()
```

MODEL TRAINING

In [69]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.3)
```

In [70]:
```
from sklearn.svm import SVC
svm = SVC()
svm.fit(X_train, y_train)
```

Out[70]:
```
SVC()
```

MODEL EVALUATION

In [71]:
```
predictions = svm.predict(X_test)

from sklearn.metrics import accuracy_score
accuracy_score(y_test, predictions)
```

Out[71]:
```
0.9555555555555556
```

CLASSIFICATION REPORT

In [72]:
```
from sklearn.metrics import classification_report
print(classification_report(y_test, predictions))
```

```
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        11
Iris-versicolor       1.00      0.90      0.95        21
 Iris-virginica       0.87      1.00      0.93        13

       accuracy                           0.96        45
      macro avg       0.96      0.97      0.96        45
   weighted avg       0.96      0.96      0.96        45
```

TESTING THE MODEL

In [73]:
```
X_new = np.array([[3, 2, 1, 0.2], [  4.9, 2.2, 3.8, 1.1 ], [  5.3, 2.5, 4.6, 1.9 ]])
prediction = svm.predict(X_new)
print("Prediction of Species: {}".format(prediction))
```

```
Prediction of Species: ['Iris-setosa' 'Iris-versicolor' 'Iris-virginica']
```

The model is predicting correctly because the setosa is shortest and virginica is the longest and versicolor is in between these two as we saw this in above graph.

Save the model using pickle

In [74]:
```
import pickle
with open('Model.pickle', 'wb') as f:
    pickle.dump(svm, f)
```

LOAD THE MODEL

In [75]:
```
with open('Model.pickle', 'rb') as f:
    model = pickle.load(f)
model.predict(X_new)
```

Out[75]:
```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)
```