OASIS Infobyte Data Science Intern

Topic = Iris Flowers

By = NARESH KUMAR

Dataset Information

. Iris dataset contains 6 columns and 150 rows
. Columns : id, SepalLengthCm, SepalWidthCm, PetalLengthCm, PetalWidthCm, Species
. Species column has 3 classes (Iris-setosa, Iris-virginica, Iris-versicolor)

Dataset : https://storage.googleapis.com/kaggle-data-sets/4247/6570/bundle/archive.zip?X-Goog-Algorithm=GOOG4-RSA-SHA256&X-Goog-Credential=gcp-kaggle-com%40kaggle-161607.iam.gserviceaccount.com%2F20240102%2Fauto%2Fstorage%2Fgoog4_request&X-Goog-Date=20240102T134624Z&X-Goog-Expires=259200&X-Goog-SignedHeaders=host&X-Goog-Signature=697501a368a85d9cea2bc996bd0e74ad60fa9b85765202263f14d2fbe61357946f (https://storage.googleapis.com/kaggle-data-sets/4247/6570/bundle/archive.zip?X-Goog-Algorithm=GOOG4-RSA-SHA256&X-Goog-Credential=gcp-kaggle-com%40kaggle-161607.iam.gserviceaccount.com%2F20240102%2Fauto%2Fstorage%2Fgoog4_request&X-Goog-Date=20240102T134624Z&X-Goog-Expires=259200&X-Goog-SignedHeaders=host&X-Goog-Signature=697501a368a85d9cea2bc996bd0e74ad60fa9b85765202263f14d2fbe61357946f

Iris Versicolor | Iris Setosa | Iris Virginica

Importing Libaries

In [1]:
```
import numpy as np
import pandas as pd
import warnings
warnings.filterwarnings("ignore")
```

Import Iris Dataset

In [2]:
```
df=pd.read_csv(r'C:\Users\hp\Desktop\Iris.csv')
```

In [3]:
```
df
```

Out[3]:

	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
...	...	...	...	...	...	...
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

150 rows × 6 columns

In [4]:
```
df.head(5)
```

Out[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

In [5]:
```
df.shape
```

Out[5]:
```
(150, 6)
```

6 columns and 150 rows

In [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
```

In [7]:
```
df.drop('Id',axis=1,inplace=True)
```

drop id column because id is not necessary

In [8]:
```
df.sample(5)
```

Out[8]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
8	4.4	2.9	1.4	0.2	Iris-setosa
142	5.8	2.7	5.1	1.9	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
56	6.3	3.3	4.7	1.6	Iris-versicolor
21	5.1	3.7	1.5	0.4	Iris-setosa

In [9]:
```
df.isnull().sum()
```

Out[9]:
```
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

tno null values in the dataset

In [10]:
```
df.duplicated().sum()
```

Out[10]:
```
3
```

we have 3 duplicates rows in the dataset

In [11]:
```
df.drop_duplicates(inplace=True)
```

In [12]:
```
df.duplicated().sum()
```

Out[12]:
```
0
```

In [13]:
```
df
```

Out[13]:

	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

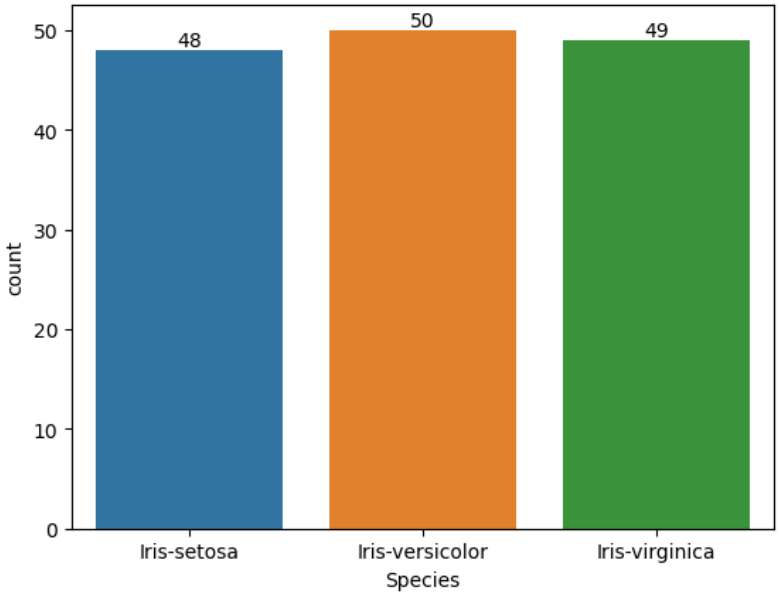
147 rows × 5 columns

EDA (Exploratory data analysis)

Data visualization

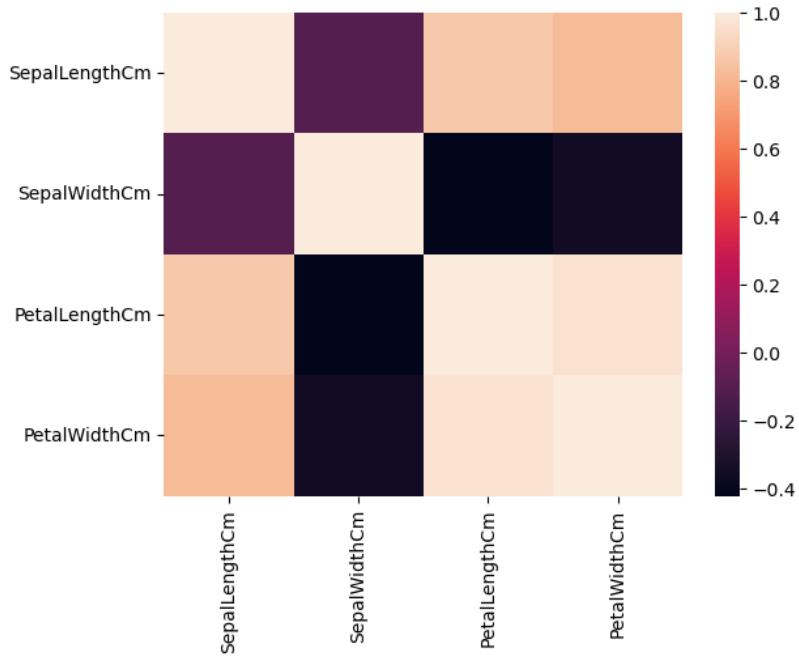
In [14]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

In [15]:
```
x=sns.countplot(x='Species', data=df)
for bars in x.containers:
    x.bar_label(bars)
```

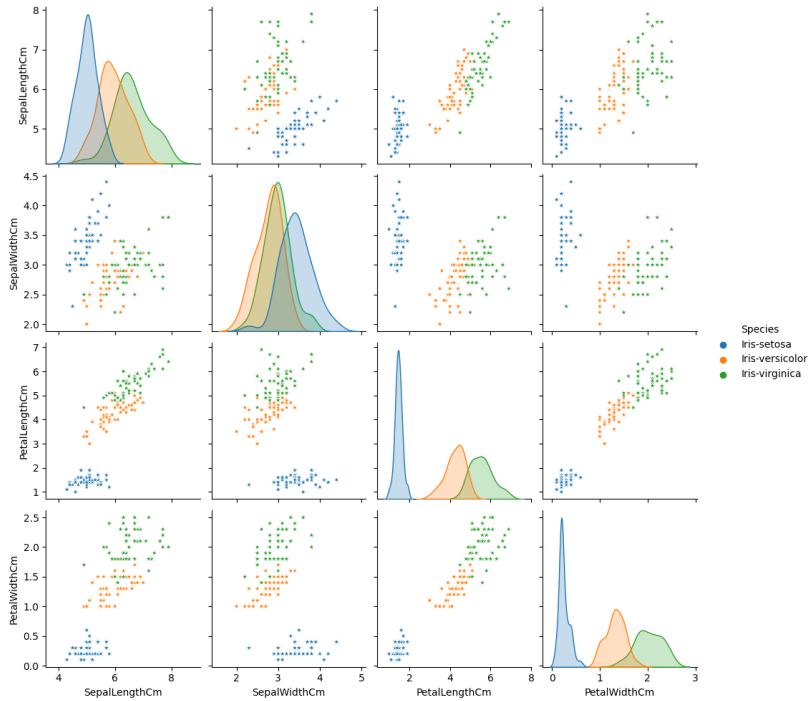


In [16]:
```
sns.heatmap(df.corr())
```

Out[16]: <AxesSubplot:>



In [17]:
```
sns.pairplot(df,hue='Species',markers='*')
plt.show()
```

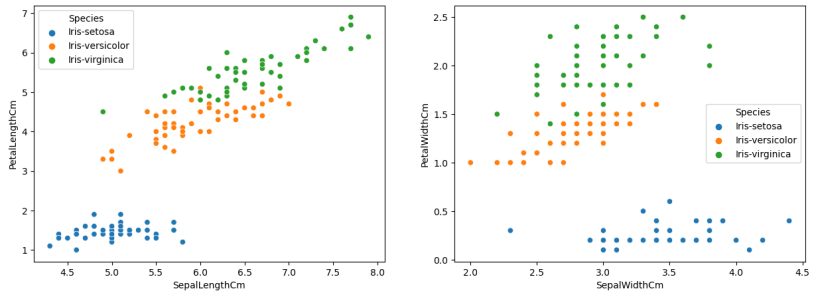


In [18]:
```
plt.figure(figsize=(15, 5))

# Creating the first subplot
plt.subplot(1, 2, 1)
sns.scatterplot(x='SepalLengthCm', y='PetalLengthCm', data=df, hue='Species'

# Creating the second subplot
plt.subplot(1, 2, 2)
sns.scatterplot(x='SepalWidthCm', y='PetalWidthCm', data=df, hue='Species')

plt.show()
```



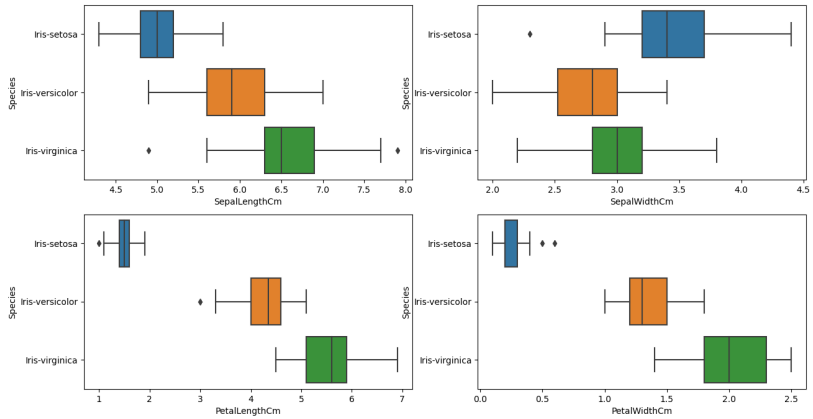
In [19]:
```
plt.figure(figsize=(15,8))
# Creating the first subplot
plt.subplot(2, 2, 1)
sns.boxplot(x='SepalLengthCm', y='Species', data=df)

# Creating the second subplot
plt.subplot(2, 2, 2)
sns.boxplot(x='SepalWidthCm', y='Species', data=df)

# Creating the third subplot
plt.subplot(2, 2, 3)
sns.boxplot(x='PetalLengthCm', y='Species', data=df)

# Creating the fourth subplot
plt.subplot(2, 2, 4)
sns.boxplot(x='PetalWidthCm', y='Species', data=df)

plt.show()
```



Label Encoder

In [20]:
```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
df['Species']=le.fit_transform(df['Species'])
df
```

Out[20]:

	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

147 rows × 5 columns

In [21]:
```
df['Species'].unique()
```

Out[21]:
```
array([0, 1, 2])
```

In [22]:
```
x=df.drop('Species',axis=1)
y=df.Species
```

Model Training

In [23]:
```
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,random_stat
```

In [24]:
```
print("x_train",x_train.shape)
print('x_test',x_test.shape)
print('y_train',y_train.shape)
print('y_test',y_test.shape)
```

```
x_train (102, 4)
x_test (45, 4)
y_train (102,)
y_test (45,)
```

In [25]:
```
from sklearn.metrics import *
def eval_model(model,x_train,y_train,x_test,y_test,model_name):
    model.fit(x_train,y_train)
    ypred=model.predict(x_test)
    train_2=model.score(x_train,y_train)
    test_2=model.score(x_test,y_test)
    MSE=mean_squared_error(y_test,ypred)
    MAE=mean_absolute_error(y_test,ypred)
    RMSE=np.sqrt(MSE)
    res=pd.DataFrame({"Train_2":train_2,'Test_2':test_2,'MSE': MSE,'MAE':MAE
    return res
```

Devlop Machine learning model

In [26]:
```
from sklearn.linear_model import LogisticRegression

# Assuming lr is your Logistic Regression model
lr = LogisticRegression(max_iter=1000)  # You can adjust the number based or


# Assuming you have the 'eval_model' function defined
# Replace it with your actual evaluation function
Logistic = eval_model(lr, x_train, y_train, x_test, y_test, 'lr')
Logistic
```

Out[26]:

	Train_2	Test_2	MSE	MAE	RMSE
lr	0.95098	1.0	0.0	0.0	0.0

In [27]:
```
from sklearn.neighbors import KNeighborsClassifier
knr=KNeighborsClassifier()
KNeighborsClassifier=eval_model(knr, x_train, y_train, x_test, y_test, 'knr'
KNeighborsClassifier
```

Out[27]:

	Train_2	Test_2	MSE	MAE	RMSE
knr	0.95098	0.955556	0.044444	0.044444	0.210819

In [28]:
```
res=pd.concat([Logistic,KNeighborsClassifier])
res
```

Out[28]:

	Train_2	Test_2	MSE	MAE	RMSE
lr	0.95098	1.000000	0.000000	0.000000	0.000000
knr	0.95098	0.955556	0.044444	0.044444	0.210819