Oasis Infobyte Internship_June-2024

Machine Intelligence Data Science Intern

Iris Flowers Classification ML Project:

Dataset Information:

- This dataset contains 150 rows of Iris blossom data, with three species of iris (setosa, versicolor, and virginica) represented in each row. Each row lists the length and width of the petals and sepals in centimeters, as well as the species of the blossom.
- There are no missing values in the dataset.

Attribute Information:

- sepal length in cm
- sepal width in cm
- petal length in cm
- petal width in cm
- class:
  - Iris-Setosa
  - Iris-Versicolour
  - Iris-Virginica

Dataset : http://archive.ics.uci.edu/ml/datasets/Iris

Importing Libraries

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn as sk
%matplotlib inline
%matplotlib inline
%matplotlib inline
%matplotlib inline
```

Import Dataset

In [2]:
```
iris_df = pd.read_csv("Iris.csv")
print("Dataset has been successfully loaded")
```

```
Dataset has been successfully loaded
```

In [3]:
```
iris_df
```

Out[3]:

	Id	SepalLength	SepalWidth	PetalLength	PetalWidth	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

Data Exploration

In [4]:
```
iris_df.shape
iris_df.info()
```

```
(150, 6)
<class 'DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 Id             150 non-null int64
 SepalLength    150 non-null float64
 SepalWidth     150 non-null float64
 PetalLength    150 non-null float64
 PetalWidth     150 non-null float64
 Species        150 non-null object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.0 KB
```

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

Out[5]:

	Id	SepalLength	SepalWidth	PetalLength	PetalWidth
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

In [6]:
```
iris_df.isnull().sum()
```

```
Id             0
SepalLength    0
SepalWidth     0
PetalLength    0
PetalWidth     0
Species        0
dtype: int64
```

- No Null values in dataset

In [7]:
```
print("Species count of values in dataset:\n",iris_df["Species"].value_counts())
print("Species Species in iris dataset",iris_df["Species"].unique())
```

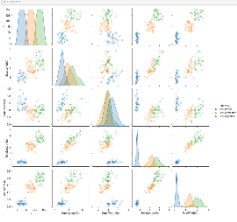
```
Species count of values in dataset:
 Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Species Species in iris dataset ['Iris-setosa' 'Iris-versicolor' 'Iris-virginica']
```

- There are 3 species with equal count in the whole iris dataset in the original

Exploratory Data Analysis

Data Visualization

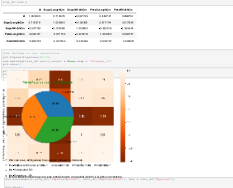
In [8]:
```
sns.pairplot(iris_df, hue = "Species", palette = "Set1")
plt.show()
```



- It shows that the Iris-setosa is separable and not better separable at the Iris-virs.

In [9]:
```
print("Species Species 75%")
print("Species Species 75%")
sns.swarmplot(x="Species", y="PetalLength", data=iris_df, hue="Species")
plt.show()
sns.swarmplot(x="Species", y="PetalWidth", data=iris_df, hue="Species")
plt.show()
```

```
Species Species 75%
Species Species 75%
```



In [10]:
```
Species Species 75%
```

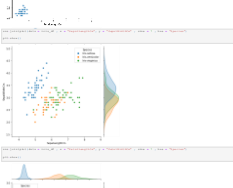
```
Species Species 75%
```

In [11]:
```
iris_df.corr()
```

Out[11]:

	Id	SepalLength	SepalWidth	PetalLength	PetalWidth
Id	1.000000	0.716676	-0.276755	0.882747	0.899759
SepalLength	0.716676	1.000000	-0.109369	0.871754	0.817954
SepalWidth	-0.276755	-0.109369	1.000000	-0.420516	-0.356544
PetalLength	0.882747	0.871754	-0.420516	1.000000	0.962757
PetalWidth	0.899759	0.817954	-0.356544	0.962757	1.000000

In [12]:
```
plot_corr = iris_df.corr()
plot_corr.index = plot_corr.columns
sns.heatmap(data=plot_corr, annot=True)
plt.show()
```

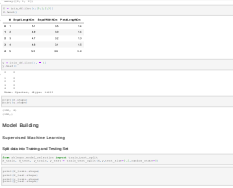


In [13]:
```
plot_corr = iris_df.corr()
plot_corr.index = plot_corr.columns
sns.heatmap(data=plot_corr, annot=True, cmap="cubehelix")
plt.show()
```

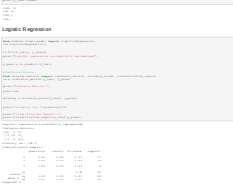
In [14]:
```
sns.jointplot(x="Id", y="SepalLength", data=iris_df, hue="Species")
plt.show()
```

In [15]:
```
sns.jointplot(x="SepalWidth", y="SepalLength", data=iris_df, hue="Species")
plt.show()
```

In [16]:
```
sns.jointplot(x="PetalLength", y="SepalLength", data=iris_df, hue="Species")
plt.show()
```



In [17]:
```
sns.jointplot(x="PetalWidth", y="SepalLength", data=iris_df, hue="Species")
plt.show()
```

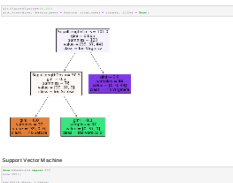


Scatterplot for Petal Length and Petal Width

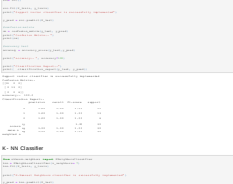
In [18]:
```
sns.scatterplot(x="petal length (cm)",y="petal width (cm)", data=iris_df,hue="Species")
plt.show()
```

In [19]:
```
sns.jointplot(x="PetalLength", y="PetalWidth", data = iris_df, hue = "Species")
plt.show()
```

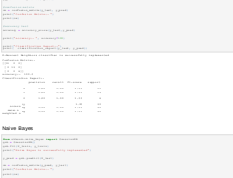
In [20]:
```
sns.jointplot(x="SepalLength", y="PetalLength", data = iris_df, hue = "Species")
plt.show()
```



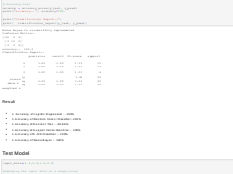
In [21]:
```
sns.jointplot(x="SepalLength", y="PetalWidth", data = iris_df, hue = "Species")
plt.show()
```



In [22]:
```
sns.jointplot(x="SepalWidth", y="PetalLength", data = iris_df, hue = "Species")
plt.show()
```



In [23]:
```
sns.jointplot(x="SepalWidth", y="PetalWidth", data = iris_df, hue = "Species")
plt.show()
```



In [24]:
```
print("Species Species 75%")
print("Species Species 75%")
sns.violinplot(x="Species", y="PetalLength", data=iris_df, palette="Spectral")
plt.show()
sns.violinplot(x="Species", y="PetalWidth", data=iris_df, palette="Spectral")
plt.show()
sns.violinplot(x="Species", y="SepalLength", data=iris_df, palette="Spectral")
plt.show()
sns.violinplot(x="Species", y="SepalWidth", data=iris_df, palette="Spectral")
plt.show()
```

```
Species Species 75%
Species Species 75%
```

In [25]:
```
print("Species Species 75%")
print("Species Species 75%")
sns.boxplot(x="Species", y="PetalLength", data=iris_df, palette="Spectral")
plt.show()
sns.boxplot(x="Species", y="PetalWidth", data=iris_df, palette="Spectral")
plt.show()
sns.boxplot(x="Species", y="SepalLength", data=iris_df, palette="Spectral")
plt.show()
sns.boxplot(x="Species", y="SepalWidth", data=iris_df, palette="Spectral")
plt.show()
```

```
Species Species 75%
Species Species 75%
```


In [26]:
```
print("Species Species 75%")
print("Species Species 75%")
sns.boxplot(x="Species", y="PetalLength", data=iris_df, palette="Pastel1")
plt.show()
sns.boxplot(x="Species", y="PetalWidth", data=iris_df, palette="Pastel1")
plt.show()
sns.boxplot(x="Species", y="SepalLength", data=iris_df, palette="Pastel1")
plt.show()
sns.boxplot(x="Species", y="SepalWidth", data=iris_df, palette="Pastel1")
plt.show()
```

```
Species Species 75%
Species Species 75%
```


In [27]:
```
sns.boxplot(x="Species", y="PetalLength", data=iris_df)
sns.stripplot(x="Species", y="PetalLength", data=iris_df, jitter=True, edgecolor="gray")
plt.show()
sns.boxplot(x="Species", y="PetalWidth", data=iris_df)
sns.stripplot(x="Species", y="PetalWidth", data=iris_df, jitter=True, edgecolor="gray")
plt.show()
sns.boxplot(x="Species", y="SepalLength", data=iris_df)
sns.stripplot(x="Species", y="SepalLength", data=iris_df, jitter=True, edgecolor="gray")
plt.show()
sns.boxplot(x="Species", y="SepalWidth", data=iris_df)
sns.stripplot(x="Species", y="SepalWidth", data=iris_df, jitter=True, edgecolor="gray")
plt.show()
```


In [28]:
```
sns.boxplot(x="Species", y="PetalLength", data=iris_df)
sns.boxplot(x="Species", y="PetalWidth", data=iris_df)
sns.boxplot(x="Species", y="SepalLength", data=iris_df)
sns.boxplot(x="Species", y="SepalWidth", data=iris_df)
plt.show()
```


In [29]:
```
sns.boxplot(x="Species", y="PetalLength", data=iris_df)
sns.boxplot(x="Species", y="PetalWidth", data=iris_df)
sns.boxplot(x="Species", y="SepalLength", data=iris_df)
sns.boxplot(x="Species", y="SepalWidth", data=iris_df)
plt.show
```


In [30]:
```
sns.boxplot(x="Species", y="PetalLength", data=iris_df)
sns.boxplot(x="Species", y="PetalWidth", data=iris_df)
sns.boxplot(x="Species", y="SepalLength", data=iris_df)
sns.boxplot(x="Species", y="SepalWidth", data=iris_df)
plt.show
```


In [31]:
```
sns.boxplot(x="Species", y="PetalLength", data=iris_df)
sns.boxplot(x="Species", y="PetalWidth", data=iris_df)
sns.boxplot(x="Species", y="SepalLength", data=iris_df)
sns.boxplot(x="Species", y="SepalWidth", data=iris_df)
plt.show
```

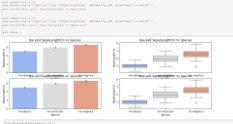

In [32]:
```
sns.boxplot(x="Species", y="PetalLength", data=iris_df)
sns.boxplot(x="Species", y="PetalWidth", data=iris_df)
sns.boxplot(x="Species", y="SepalLength", data=iris_df)
sns.boxplot(x="Species", y="SepalWidth", data=iris_df)
plt.show
```


In [33]:
```
sns.boxplot(x="Species", y="PetalLength", data=iris_df)
sns.boxplot(x="Species", y="PetalWidth", data=iris_df)
sns.boxplot(x="Species", y="SepalLength", data=iris_df)
sns.boxplot(x="Species", y="SepalWidth", data=iris_df)
plt.show
```


In [34]:
```
sns.boxplot(x="Species", y="PetalLength", data=iris_df)
sns.boxplot(x="Species", y="PetalWidth", data=iris_df)
sns.boxplot(x="Species", y="SepalLength", data=iris_df)
sns.boxplot(x="Species", y="SepalWidth", data=iris_df)
plt.show
```


In [35]:
```
sns.boxplot(x="Species", y="PetalLength", data=iris_df)
sns.boxplot(x="Species", y="PetalWidth", data=iris_df)
sns.boxplot(x="Species", y="SepalLength", data=iris_df)
sns.boxplot(x="Species", y="SepalWidth", data=iris_df)
plt.show
```


In [36]:
```
sns.boxplot(x="Species", y="PetalLength", data=iris_df)
sns.boxplot(x="Species", y="PetalWidth", data=iris_df)
sns.boxplot(x="Species", y="SepalLength", data=iris_df)
sns.boxplot(x="Species", y="SepalWidth", data=iris_df)
plt.show
```

Data Cleaning

In [37]:
```
# Drop the column 'Id' since it is not needed
from sklearn.model_selection import train_test_split
iris = iris_df.drop('Id', axis=1)

print("Species",iris["Species"].unique())
iris_df.sample(5)
```

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

Out[37]:

	SepalLength	SepalWidth	PetalLength	PetalWidth	Species
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
5	5.4	3.9	1.7	0.4	Iris-setosa
6	4.6	3.4	1.4	0.3	Iris-setosa
7	5.0	3.4	1.5	0.2	Iris-setosa

In [38]:
```
iris.sample(5)
```

Out[38]:

	SepalLength	SepalWidth	PetalLength	PetalWidth
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
5	5.4	3.9	1.7	0.4
6	4.6	3.4	1.4	0.3
7	5.0	3.4	1.5	0.2

In [39]:
```
iris.shape[0] - 5
```

Out[39]:
```
145
```

In [40]:
```
iris.shape[0] - 10
```

Out[40]:
```
140
```

In [41]:
```
iris.shape[0] - 15
```

Out[41]:
```
135
```

In [42]:
```
iris.shape[0] - 20
```

Out[42]:
```
130
```

Model Building

Supervised Machine Learning

Split data into Training and Testing Set

In [43]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(iris[iris.iloc[:,0:4].columns.values], iris['Species'], test_size=0.3, random_state=0)
```

In [44]:
```
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
```

```
(105, 4)
(105,)
(45, 4)
(45,)
```

Logistic Regression

In [45]:
```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()

print("Fitting Fitting in progress...")
lr.fit(X_train,y_train)
print("Fitting Fitting is successfully implemented")

y_pred = lr.predict(X_test)

from sklearn.metrics import confusion_matrix, classification_report
print("Confusion Matrix:\n",
confusion_matrix(y_test, y_pred))

print("Report:\n",
classification_report(y_test,y_pred))
```

```
Fitting Fitting in progress...
Fitting Fitting is successfully implemented
Confusion Matrix:
 [[ 19   0   0]
 [  0  13   0]
 [  0   0  13]]
Report:
              precision    recall  f1-score   support

 Iris-setosa       1.00      1.00      1.00        19
Iris-versicolor    1.00      1.00      1.00        13
 Iris-virginica    1.00      1.00      1.00        13

       accuracy                           1.00        45
      macro avg       1.00      1.00      1.00        45
   weighted avg       1.00      1.00      1.00        45
```

Random Forest Classifier

In [46]:
```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier()

print("Fitting Fitting in progress...")
rfc.fit(X_train,y_train)
print("Fitting Fitting is successfully implemented")

y_pred = rfc.predict(X_test)

from sklearn.metrics import confusion_matrix, classification_report
print("Confusion Matrix:\n",
confusion_matrix(y_test, y_pred))

print("Report:\n",
classification_report(y_test,y_pred))
```

```
Fitting Fitting in progress...
Fitting Fitting is successfully implemented
Confusion Matrix:
 [[ 19   0   0]
 [  0  13   0]
 [  0   0  13]]
Report:
              precision    recall  f1-score   support

 Iris-setosa       1.00      1.00      1.00        19
Iris-versicolor    1.00      1.00      1.00        13
 Iris-virginica    1.00      1.00      1.00        13

       accuracy                           1.00        45
      macro avg       1.00      1.00      1.00        45
   weighted avg       1.00      1.00      1.00        45
```

Decision Tree

In [47]:
```
from sklearn.tree import DecisionTreeClassifier
dtc = DecisionTreeClassifier()

print("Fitting Fitting in progress...")
dtc.fit(X_train,y_train)
print("Fitting Fitting is successfully implemented")

y_pred = dtc.predict(X_test)

from sklearn.metrics import confusion_matrix, classification_report
print("Confusion Matrix:\n",
confusion_matrix(y_test, y_pred))

print("Report:\n",
classification_report(y_test,y_pred))
```

```
Fitting Fitting in progress...
Fitting Fitting is successfully implemented
Confusion Matrix:
 [[ 19   0   0]
 [  0  12   1]
 [  0   0  13]]
Report:
              precision    recall  f1-score   support

 Iris-setosa       1.00      1.00      1.00        19
Iris-versicolor    1.00      0.92      0.96        13
 Iris-virginica    0.93      1.00      0.96        13

       accuracy                           0.98        45
      macro avg       0.98      0.97      0.98        45
   weighted avg       0.98      0.98      0.98        45
```

In [48]:
```
# Plot the decision tree
from sklearn.tree import plot_tree
```

In [49]:
```
plt.figure(figsize=(20,20))
plot_tree(dtc, filled = True, rounded = True, class_names = classes, fontsize = 8pt)
```


Support Vector Machine

In [50]:
```
from sklearn.svm import SVC
svm = SVC()

print("Fitting Fitting in progress...")
svm.fit(X_train,y_train)
print("Fitting Fitting is successfully implemented")

y_pred = svm.predict(X_test)

from sklearn.metrics import confusion_matrix, classification_report
print("Confusion Matrix:\n",
confusion_matrix(y_test, y_pred))

print("Report:\n",
classification_report(y_test,y_pred))
```

```
Fitting Fitting in progress...
Fitting Fitting is successfully implemented
Confusion Matrix:
 [[ 19   0   0]
 [  0  13   0]
 [  0   0  13]]
Report:
              precision    recall  f1-score   support

 Iris-setosa       1.00      1.00      1.00        19
Iris-versicolor    1.00      1.00      1.00        13
 Iris-virginica    1.00      1.00      1.00        13

       accuracy                           1.00        45
      macro avg       1.00      1.00      1.00        45
   weighted avg       1.00      1.00      1.00        45
```

K- NN Classifier

In [51]:
```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=3)

print("Fitting Fitting in progress...")
knn.fit(X_train,y_train)
print("Fitting Fitting is successfully implemented")

y_pred = knn.predict(X_test)

from sklearn.metrics import confusion_matrix, classification_report
print("Confusion Matrix:\n",
confusion_matrix(y_test, y_pred))

print("Report:\n",
classification_report(y_test,y_pred))
```

```
Fitting Fitting in progress...
Fitting Fitting is successfully implemented
Confusion Matrix:
 [[ 19   0   0]
 [  0  13   0]
 [  0   0  13]]
Report:
              precision    recall  f1-score   support

 Iris-setosa       1.00      1.00      1.00        19
Iris-versicolor    1.00      1.00      1.00        13
 Iris-virginica    1.00      1.00      1.00        13

       accuracy                           1.00        45
      macro avg       1.00      1.00      1.00        45
   weighted avg       1.00      1.00      1.00        45
```

Naive Bayes

In [52]:
```
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()

print("Fitting Fitting in progress...")
gnb.fit(X_train,y_train)
print("Fitting Fitting is successfully implemented")

y_pred = gnb.predict(X_test)

from sklearn.metrics import confusion_matrix, classification_report
print("Confusion Matrix:\n",
confusion_matrix(y_test, y_pred))

print("Report:\n",
classification_report(y_test,y_pred))
```

```
Fitting Fitting in progress...
Fitting Fitting is successfully implemented
Confusion Matrix:
 [[ 19   0   0]
 [  0  13   0]
 [  0   0  13]]
Report:
              precision    recall  f1-score   support

 Iris-setosa       1.00      1.00      1.00        19
Iris-versicolor    1.00      1.00      1.00        13
 Iris-virginica    1.00      1.00      1.00        13

       accuracy                           1.00        45
      macro avg       1.00      1.00      1.00        45
   weighted avg       1.00      1.00      1.00        45
```

Result

- 1. Accuracy Logistic Regression - 100%
- 2. Accuracy Support Vector Machine - 100%
- 3. Accuracy Decision Tree - 100%
- 4. Accuracy Random Forest Machine - 100%
- 5. Accuracy K-NN Classifier - 100%
- 6. Accuracy Naive Bayes - 100%

Test Model

In [53]:
```
# Input features for a new flower
input_features = [5.1, 3.5, 1.4, 0.2]

# Convert the input features to a DataFrame
input_df = pd.DataFrame([input_features], columns=['SepalLength', 'SepalWidth', 'PetalLength', 'PetalWidth'])

# Predict the species using the Logistic Regression model
prediction = logreg.predict(input_df)

print("Predicted species:", prediction[0])
```

```
Predicted species: Iris-setosa
```

Suggestions are always welcome.

Thank You !