THE SPARKS FOUNDATION - GRIP November 2022

Data Science & Business Analytics

Name: Roshni Verma

Task 6: Prediction Using Decision Tree Algorithm

Dataset is available at: https://bit.ly/3kXTdox

In [1]:
```
# Importing the required libraries
import pandas as pd
import numpy as np
import seaborn as sb
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics
```

In [3]:
```
df = pd.read_csv("Iris.csv")
df.head()
```

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

In [4]:
```
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: 6.5+ KB
```

As seen from the above table, except for the column named "Species" which consists of object entries, rest of the columns have entries having data type has float.

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

Out[5]:

	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

From the above table we can find out the values like the count mean, standard deviation , minimum and maximum values and so on for each of the columns.

In [6]:
```
df.isna().sum()
```

Out[6]:
```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

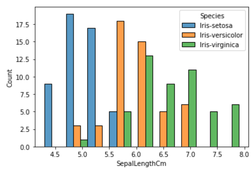
As seen from above, there are no missing values in our dataset. Nor do we have any garbage values. Hence no data cleaning is required.

Visualizing the dataset.

Seeing the frequency of each species having various values for each of the attributes- Sepal Length, Sepal Width, Petal Length and Petal width.

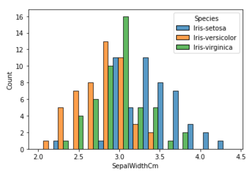
In [9]:
```
sb.histplot(data=df, x="SepalLengthCm", hue="Species", multiple="dodge")
```

Out[9]:
```
<AxesSubplot:xlabel='SepalLengthCm', ylabel='Count'>
```



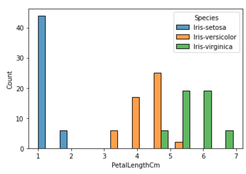
In [10]:
```
sb.histplot(data=df, x="SepalWidthCm", hue="Species", multiple="dodge")
```

Out[10]:
```
<AxesSubplot:xlabel='SepalWidthCm', ylabel='Count'>
```



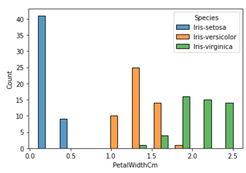
In [11]:
```
sb.histplot(data=df, x="PetalLengthCm", hue="Species", multiple="dodge")
```

Out[11]:
```
<AxesSubplot:xlabel='PetalLengthCm', ylabel='Count'>
```



In [12]:
```
sb.histplot(data=df, x="PetalWidthCm", hue="Species", multiple="dodge")
```

Out[12]:
```
<AxesSubplot:xlabel='PetalWidthCm', ylabel='Count'>
```



Now we will get the target values and feature values.

In [14]:
```
feature_cols = ['SepalLengthCm','SepalWidthCm','PetalLengthCm','PetalWidthCm']
X = df[feature_cols]
X.head()
```

Out[14]:

	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
4	5.0	3.6	1.4	0.2

In [15]:
```
y = df.Species
y.unique()
```

Out[15]:
```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)
```

Further, we will split the dataset into the training and test dataset.

In [16]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=2)
```

Decision Tree Classifier

In [17]:
```
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier()
```

Fitting the model to our train dataset.

In [18]:
```
model.fit(X_train,y_train)
```

Out[18]:
```
DecisionTreeClassifier()
```

Predicting the values for the test dataset.

In [19]:
```
y_pred = model.predict(X_test)
```

Calculating the Accuracy of our trained model.

In [22]:
```
metrics.accuracy_score(y_test,y_pred)
```

Out[22]:
```
0.9473684210526315
```