LETSGROWMORE DATA SCIENCE

Iris Flower Classification ML Project

PRANJAL JAIN

TASK - 3

In [39]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import datasets
%matplotlib inline
```

In [40]:
```
df = pd.read_csv('D:\\Iris.csv')
```

In [41]:
```
df_columns = ["sepal length", "sepal width", "petal length", "petal width", "Class"]
df.head()
```

Out[41]:

	sepal length	sepal width	petal length	petal width	Class
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 [42]:
```
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   Class         150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

In [43]:
```
df.describe()
```

Out[43]:

	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 [44]:
```
df.isnull().sum()   #to check for null values
```

Out[44]:
```
sepal length    0
sepal width     0
petal length    0
petal width     0
Class           0
dtype: int64
```

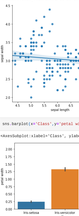
In [45]:
```
df.value_counts()     #returns a Series containing the counts of unique values.
```

Out[45]:
```
sepal length  sepal width  petal length  petal width  Class
4.9           3.1          1.5           0.1          Iris-setosa       3
5.0           3.4          1.5           0.2          Iris-setosa       1
              3.6          1.4           0.2          Iris-setosa       1
5.8           3.9          4.2           1.5          Iris-versicolor   1
6.2           3.4          5.4           2.3          Iris-virginica    1
              ...
5.5           2.3          4.0           1.3          Iris-versicolor   1
              2.4          3.8           1.1          Iris-versicolor   1
              2.5          4.0           1.3          Iris-versicolor   1
7.9           3.8          6.4           2.0          Iris-virginica    1
Length: 147, dtype: int64
```

In [46]:
```
sns.displot(df['sepal length'],kde=True, color="y")
plt.title("Distribution of Sepal length", fontsize=20, color = 'brown')
plt.show()
```

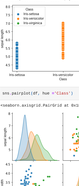
In [47]:
```
sns.lmplot(x='sepal length',y='sepal width',data=df) #linerscatter graph
```

Out[47]:
```
<seaborn.axisgrid.FacetGrid at 0x19ba4116790>
```



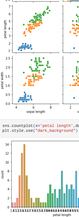
In [48]:
```
sns.barplot(x='Class',y='petal width',data=df)
```

Out[48]:
```
<AxesSubplot:xlabel='Class', ylabel='petal width'>
```



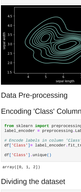
In [49]:
```
sns.scatterplot(x='Class', y='sepal length',s = 100, data=df , hue="Class")
```

Out[49]:
```
<AxesSubplot:xlabel='Class', ylabel='sepal length'>
```



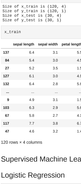
In [50]:
```
sns.pairplot(df, hue ='Class')
```

Out[50]:
```
<seaborn.axisgrid.PairGrid at 0x19ba413b160>
```



In [51]:
```
sns.countplot(df['petal length'],data=df)
plt.style.use('dark_background')
```

In [52]:
```
sns.kdeplot(df['sepal length'],df['sepal width'],data=df)
plt.style.use("grayscale")
```



Data Pre-processing

Encoding 'Class' Column

In [53]:
```
from sklearn import preprocessing
label_encoder = preprocessing.LabelEncoder()

# Encode labels in column 'Class'.
df['Class']= label_encoder.fit_transform(df['Class'])
```

In [54]:
```
df['Class'].unique()
```

Out[54]:
```
array([0, 1, 2])
```

Dividing the dataset

In [55]:
```
# divide the dataset into dependent and independent  features
x = df.iloc[:, 0:4]
x.head()
```

Out[55]:

	sepal length	sepal width	petal length	petal width
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 [56]:
```
y = df['Class'] # 20
y
```

Out[56]:

	Class
0	0
1	0
2	0
3	0
4	0
...	...
145	2
146	2
147	2
148	2
149	2

150 rows × 1 columns

Splitting dataset into Training and Testing

In [57]:
```
# train-test split
from sklearn.model_selection import train_test_split
```

In [58]:
```
x_train,x_test,y_train,y_test=train_test_split (x, y, test_size = 20,random_state=0)
print("Size of x_train :",x_train.shape)
print("Size of y_train :",y_train.shape)
print("Size of x_test :",x_test.shape)
print("Size of y_test :",y_test.shape)
```

```
Size of x_train is (130, 4)
Size of y_train is (130, 1)
Size of x_test is (20, 4)
Size of y_test is (20, 1)
```

In [59]:
```
x_train
```

Out[59]:

	sepal length	sepal width	petal length	petal width
137	6.4	3.1	5.5	1.8
84	5.4	3.0	4.5	1.5
27	5.2	3.5	1.5	0.2
127	6.1	3.0	4.9	1.8
132	6.4	2.8	5.6	2.2
...	...	...	...	...
9	4.9	3.1	1.5	0.1
103	6.3	2.9	5.6	1.8
67	5.8	2.7	4.1	1.0
117	7.7	3.8	6.7	2.2
47	4.6	3.2	1.4	0.2

120 rows × 4 columns

Supervised Machine Learning Models

Logistic Regression

In [60]:
```
from sklearn.linear_model import LogisticRegression
lgr = LogisticRegression(max_iter=1000)
lgr.fit (x_train, y_train.values.ravel())

# Predict the values

prediction = lgr.predict (x_test)
print(prediction)
```

```
[2 1 0 2 0 2 0 1 1 1 2 1 1 1 1 0 1 1 0 0 2 1 0 0 2 0 0 1 1 0]
```

In [61]:
```
# Confusion Matrix
from sklearn.metrics import confusion_matrix
confusionMatrix = confusion_matrix (y_test, prediction)
confusionMatrix
```

Out[61]:
```
array([[5,  0,  0],
       [0, 11,  1],
       [0,  0,  3]], dtype=int64)
```

In [62]:
```
from sklearn.metrics import accuracy_score

print("Accuracy is",accuracy_score(y_test, prediction))
```

```
Accuracy is 0.9
```

KNN

In [63]:
```
from sklearn.neighbors import KNeighborsClassifier

knn=KNeighborsClassifier(n_neighbors=19)
knn.fit(x_train,y_train.values.ravel())

y_pred = knn.predict(x_test)

from sklearn.metrics import accuracy_score
print("Accuracy is",accuracy_score(y_test, y_pred))
```

```
Accuracy is 0.9666666666666667
```

In [64]:
```
# Confusion matrix to evaluate performance of data
from sklearn.metrics import confusion_matrix
confusionMatrix = confusion_matrix (y_test, y_pred)
values = ['Iris-setosa', 'Iris-versicolor', 'Iris-virginica']
confusionMatrix_eval = pd.DataFrame(confusionMatrix, columns = values, index = values)
confusionMatrix_eval
```

Out[64]:

	Iris-setosa	Iris-versicolor	Iris-virginica
Iris-setosa	11	0	0
Iris-versicolor	0	12	1
Iris-virginica	0	0	6

Decision Tree

In [65]:
```
from sklearn import tree
dt = tree.DecisionTreeClassifier()
dt.tree.fit(x_train,y_train)
```

Out[65]:
```
DecisionTreeClassifier()
```

In [66]:
```
pred_tree = dt.tree.predict(x_test)
```

In [67]:
```
accuracy = accuracy_score(y_test, pred_tree)*100
```

In [68]:
```
print("Accuracy is", accuracy)
```

```
Accuracy is 100.0
```

In [ ]:
```

```