IRIS FLOWER CLASSIFICATION

Importing All The Necessary Libraries

In [101…

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split  #splits data into training and testing model
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

Data Collection, Loading And Pre Processing

In [102…

```
#Loading the CSV file into a Pandas Dataframe
data=pd.read_csv("C:/Users/NIdhi Aggarwal/Downloads/archive/IRIS.csv")
```

In [103…

```
data.head()
```

Out[103]:

	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 [104…

```
data.shape
```

Out[104]:

```
(150, 5)
```

In [105…

```
data.describe()
```

Out[105]:

	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 [106…

```
data.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   species       150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

In [107…

```
data.isnull().sum()
```

Out[107]:

```
sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
species         0
dtype: int64
```

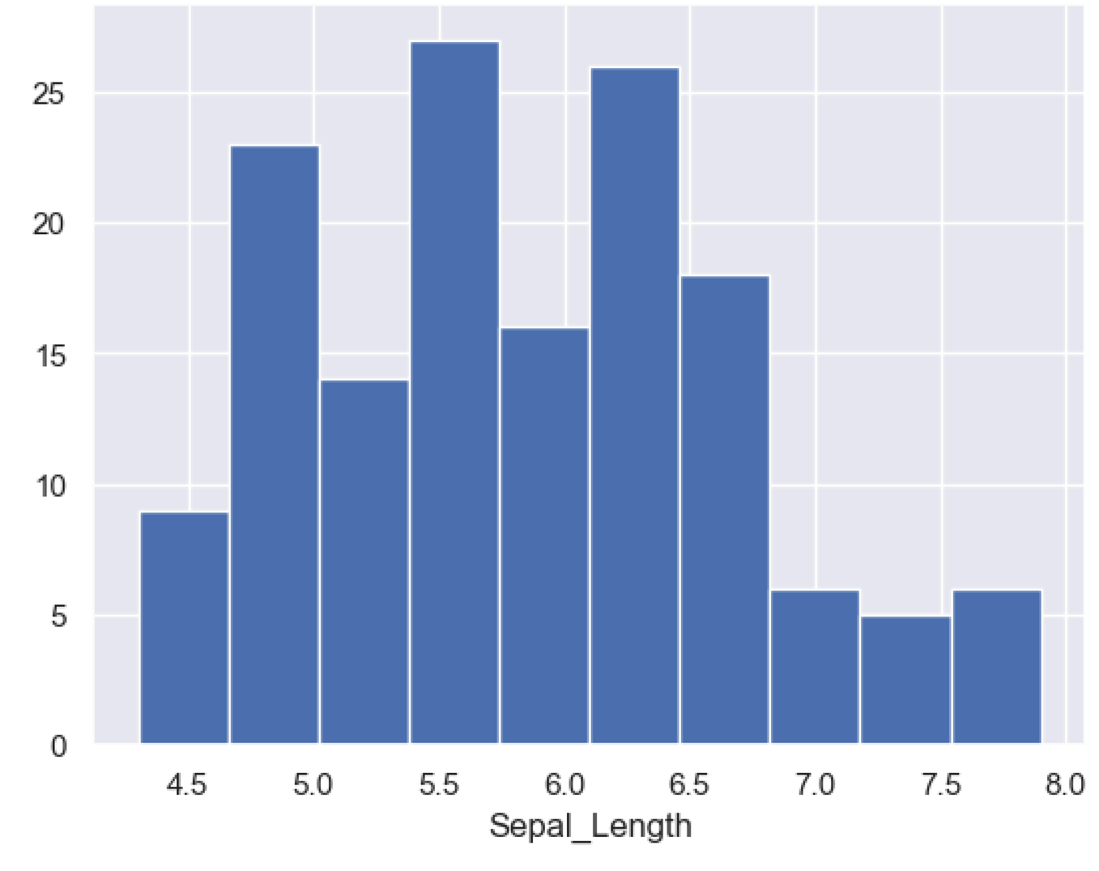
Data Visualization

In [139…

```
data['sepal_length'].hist()
plt.xlabel("Sepal_Length")
```

Out[139]:

```
Text(0.5, 0, 'Sepal_Length')
```

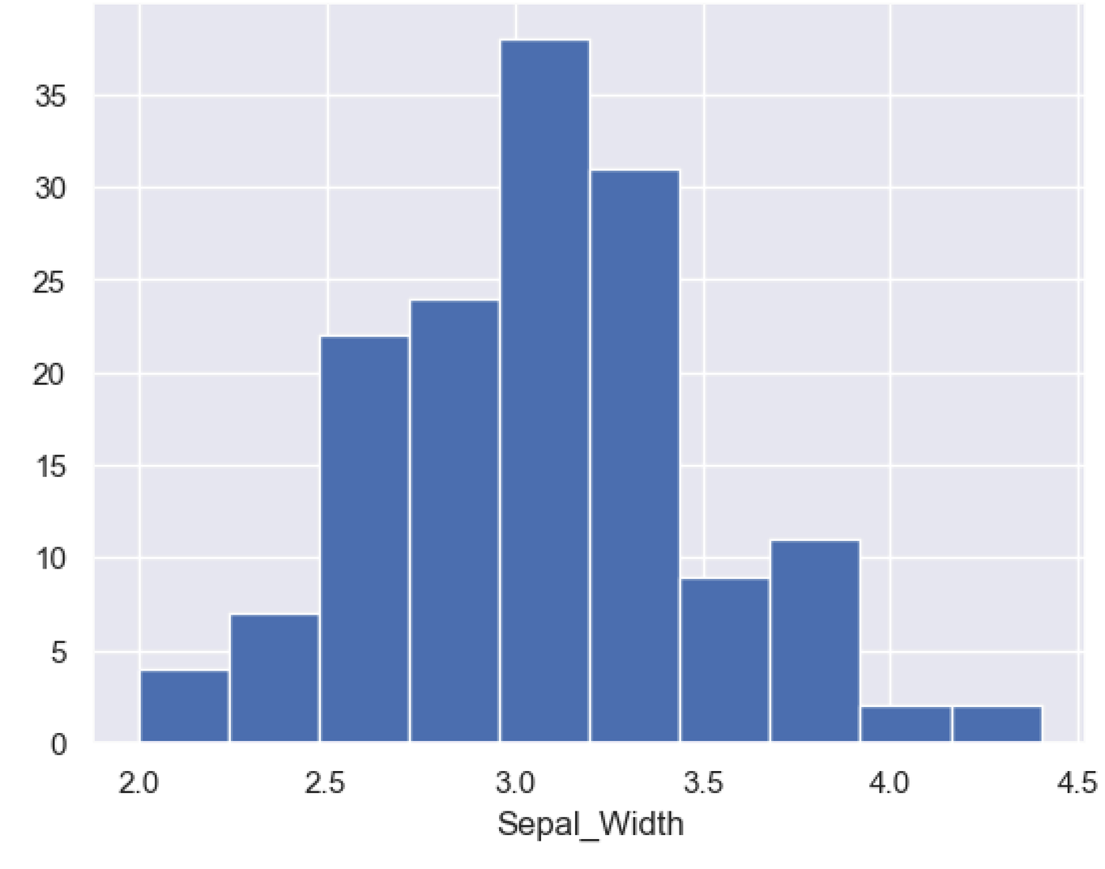


In [140…

```
data['sepal_width'].hist()
plt.xlabel('Sepal_Width')
```

Out[140]:

```
Text(0.5, 0, 'Sepal_Width')
```

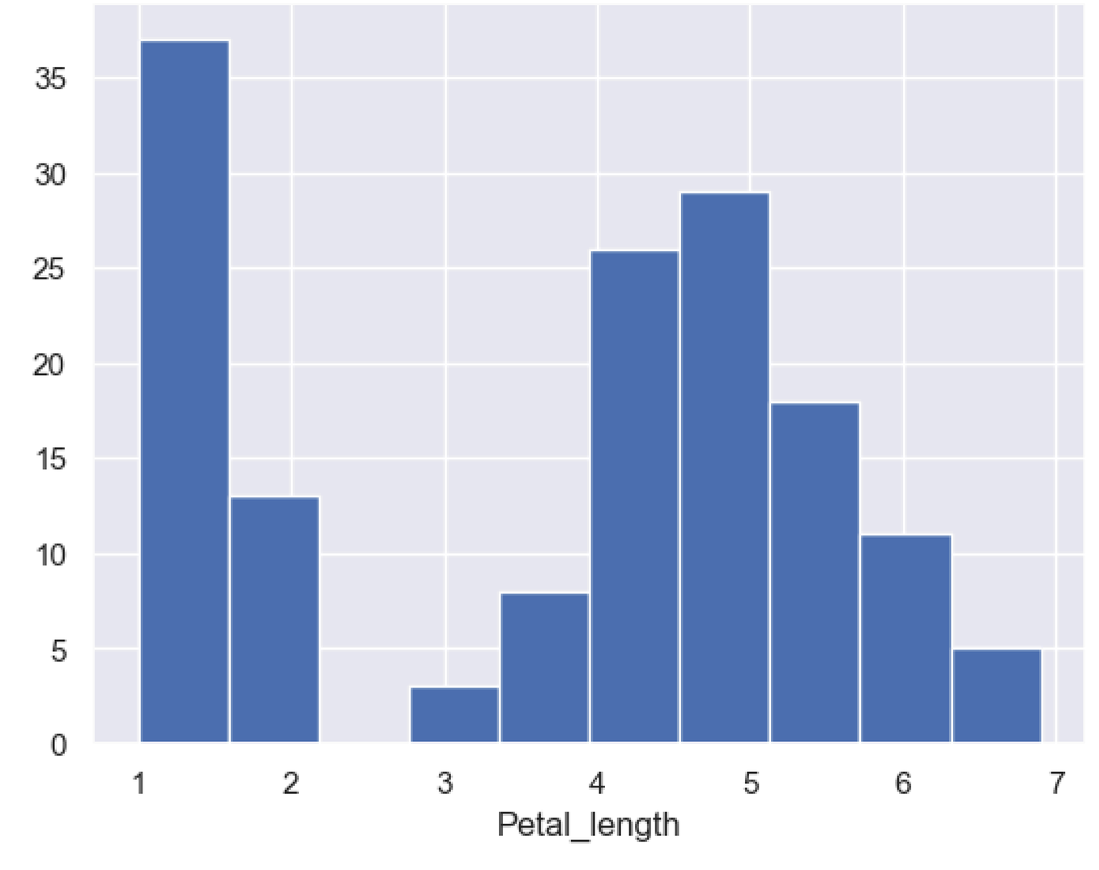


In [142…

```
data['petal_length'].hist()
plt.xlabel('Petal_length')
```

Out[142]:

```
Text(0.5, 0, 'Petal_length')
```

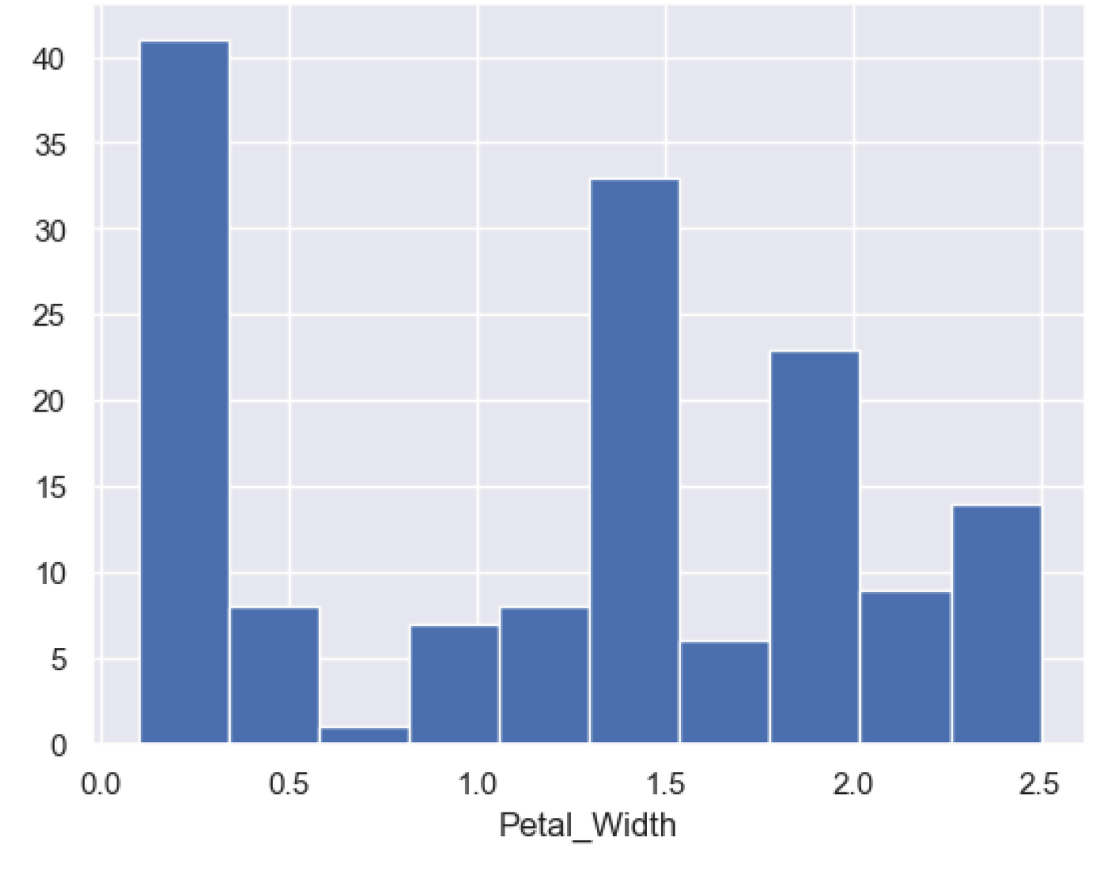


In [143…

```
data['petal_width'].hist()
plt.xlabel('Petal_Width')
```

Out[143]:

```
Text(0.5, 0, 'Petal_Width')
```



In [113…

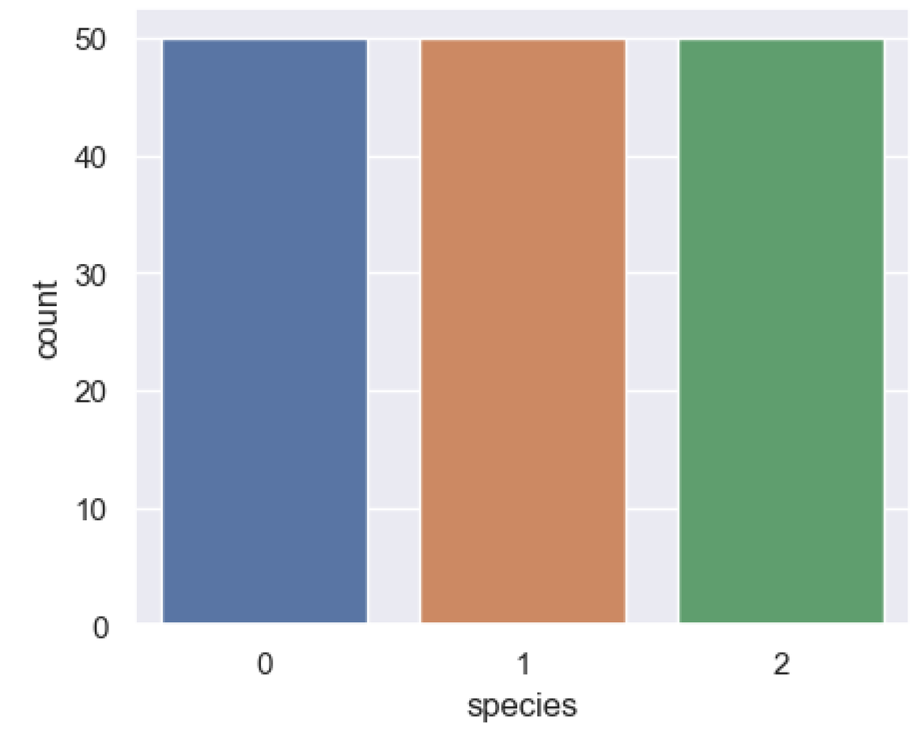
```
data['species'].value_counts()
```

Out[113]:

```
species
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: count, dtype: int64
```

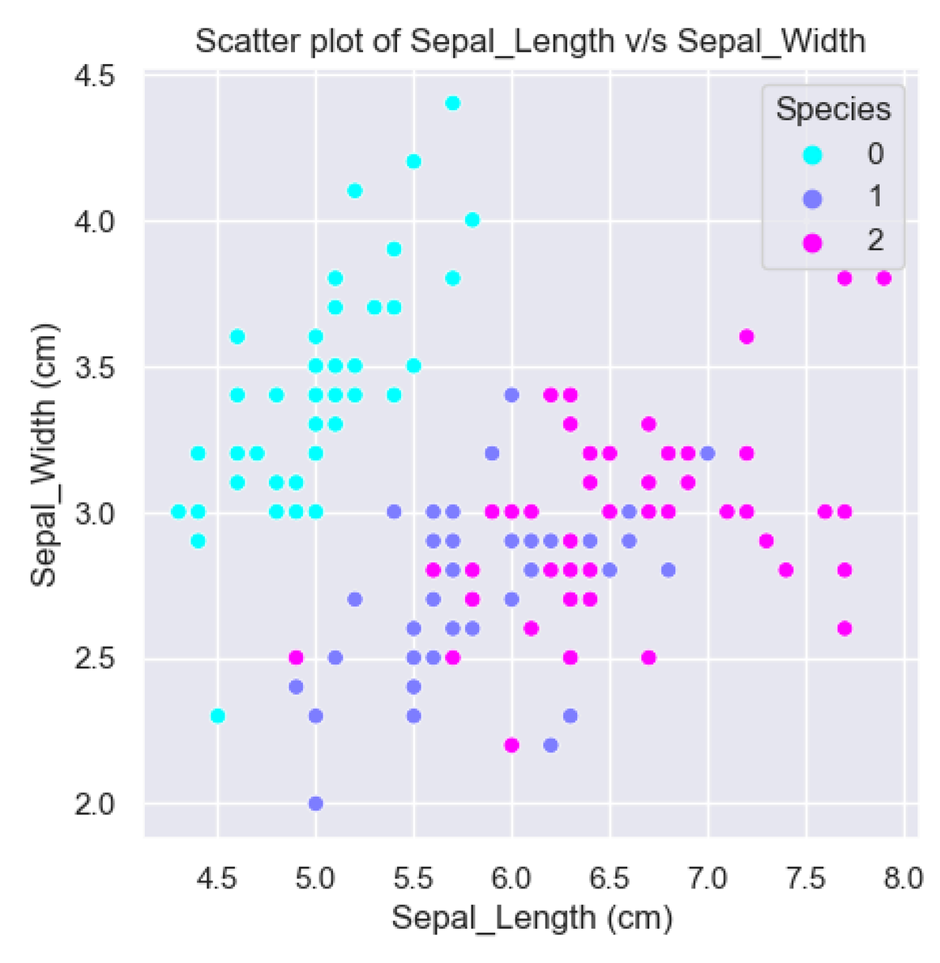
In [144…

```
sns.set()
plt.figure(figsize=(5,4))
sns.countplot(x='species',data=data)
plt.show()
```



In [154…

```
# Create the scatter plot with size and color by species
plt.figure(figsize=(5,5))
sns.scatterplot(data=data, x='sepal_length', y='sepal_width', hue='species',  palette='cool')
plt.title("Scatter plot of Sepal_Length v/s Sepal_Width")
plt.xlabel("Sepal_Length (cm)")
plt.ylabel("Sepal_Width (cm)")
plt.legend(title='Species')
plt.show()
```



In [146…

```
# Create the scatter plot with size and color by species
plt.figure(figsize=(5, 5))
sns.scatterplot(data=data, x='petal_length', y='petal_width', hue='species',  palette='cool')
plt.title("Scatter plot of Petal_Length v/s Petal_Width")
plt.xlabel("Petal_Length (cm)")
plt.ylabel("Petal_Width (cm)")
plt.legend(title='Species')
plt.show()
```

In [117…

```
X=data.drop('species',axis=1)
```

In [118…

```
X.corr()
```

Out[118]:

	sepal_length	sepal_width	petal_length	petal_width
sepal_length	1.000000	-0.109369	0.871754	0.817954
sepal_width	-0.109369	1.000000	-0.420516	-0.356544
petal_length	0.871754	-0.420516	1.000000	0.962757
petal_width	0.817954	-0.356544	0.962757	1.000000

In [147…

```
data.replace({'species':{'Iris-setosa':0,'Iris-versicolor':1,'Iris-virginica':2}},inplace=True)
data.head()
```

Out[147]:

	sepal_length	sepal_width	petal_length	petal_width	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

In [122…

```
#Here, X are the features and Y is the target i.e. Classification on the basis of the Species
X=data.drop(columns=['species'])
Y=data['species']
```

Splitting The Data into Training And Tesyting Data

In [123…

```
X_train, X_test, Y_train, Y_test=train_test_split(X, Y, test_size=0.2, random_state=42)
```

In [124…

```
X.shape, X_train.shape, X_test.shape
```

Out[124]:

```
((150, 4), (120, 4), (30, 4))
```

Training The Dataset

In [125…

```
model=LogisticRegression()
```

In [126…

```
model.fit(X_train,Y_train)
```

Out[126]:

```
▼ LogisticRegression
LogisticRegression()
```

In [148…

```
X_train_predict=model.predict(X_train)
print(X_train_predict)
```

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

In [149…

```
train_accuracy_score=accuracy_score(Y_train, X_train_predict)
print("The accuracy score of the Training Dataset is: ", train_accuracy_score)
```

```
The accuracy score of the Training Dataset is:  0.975
```

Evaluation Of Data

In [150…

```
X_test_predict=model.predict(X_test)
print(X_test_predict)
```

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

In [151…

```
test_accuracy_score=accuracy_score(Y_test, X_test_predict)
print("The accuracy score of the Testing Dataset is: ",test_accuracy_score)
```

```
The accuracy score of the Testing Dataset is:  1.0
```

In [ ]: