

Name- Sandipan Mukherjee

LETSGROWMORE INTERNSHIP :

01) BEGINNER LEVEL TASK -

Iris Flowers Classification ML Project :

Import library

```
In [2]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
```

Load Data and Understanding the Data Info

```
In [3]: i_data = pd.read_csv('Iris.csv')
i_data.head()
```

```
Out[3]:   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 [4]: i_data.tail()
```

```
Out[4]:   sepal length  sepal width  petal length  petal width  class
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
```

```
In [5]: i_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   class              150 non-null    object  
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

```
In [6]: i_data.describe()
```

```
Out[6]:   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      0.829066    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
```

Data Preprocessing

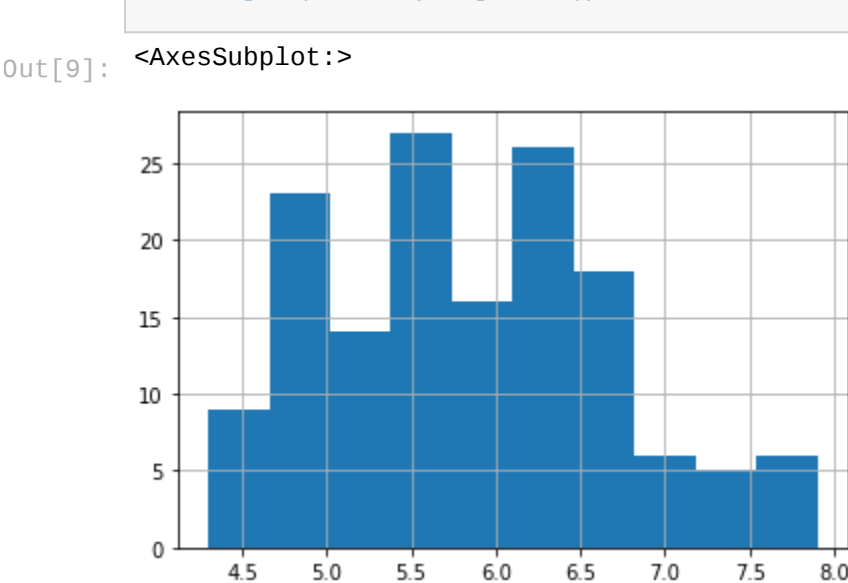
```
In [7]: i_data['class'].value_counts()
```

```
Out[7]: Iris-setosa      50
Iris-versicolor      50
Iris-virginica       50
Name: class, dtype: int64
```

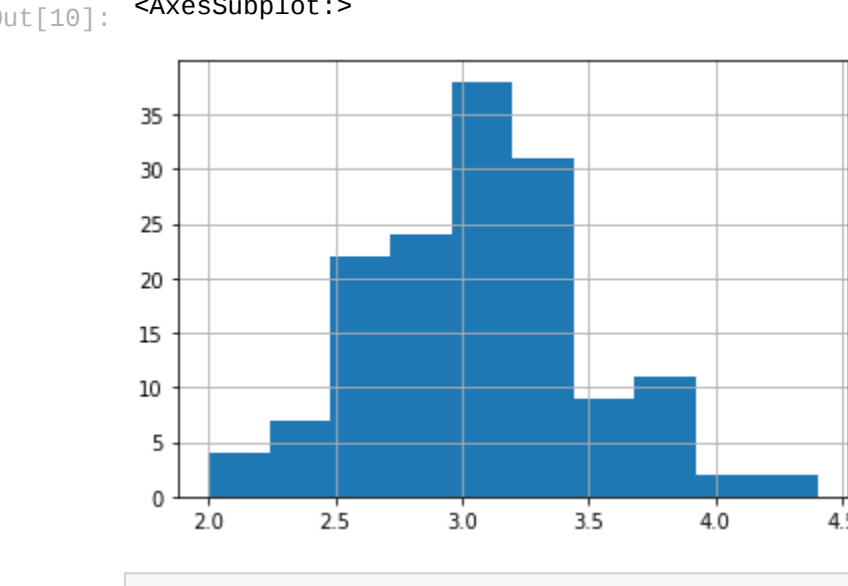
```
In [8]: i_data.isnull().sum()
```

```
Out[8]: sepal length    0
sepal width    0
petal length    0
petal width    0
class          0
dtype: int64
```

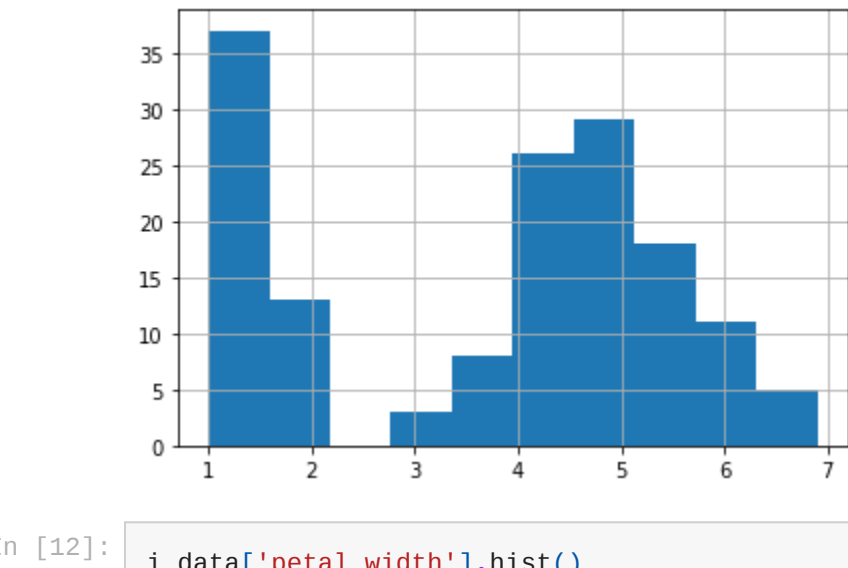
```
In [9]: i_data['sepal length'].hist()
```



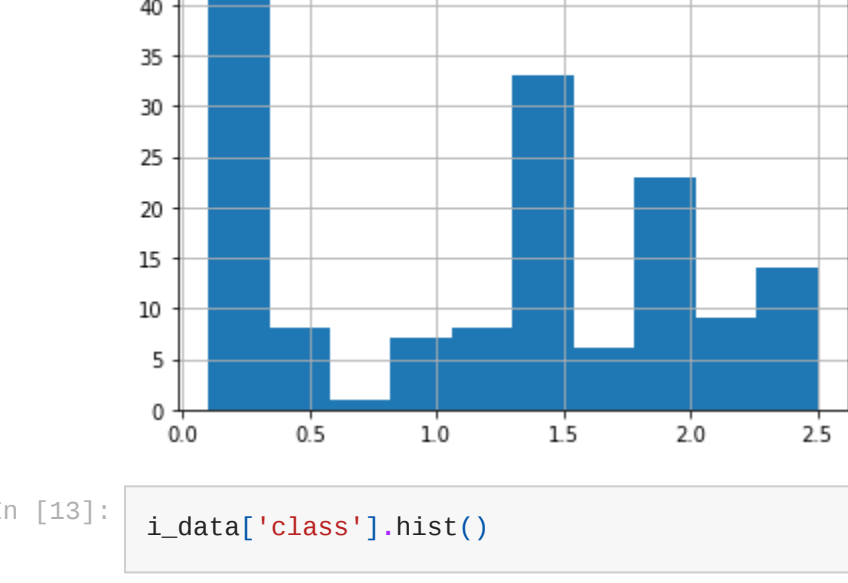
```
In [10]: i_data['sepal width'].hist()
```



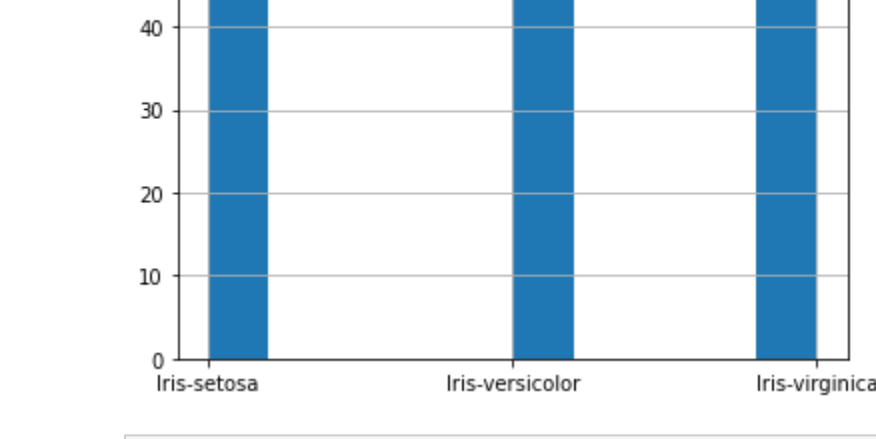
```
In [11]: i_data['petal length'].hist()
```



```
In [12]: i_data['petal width'].hist()
```



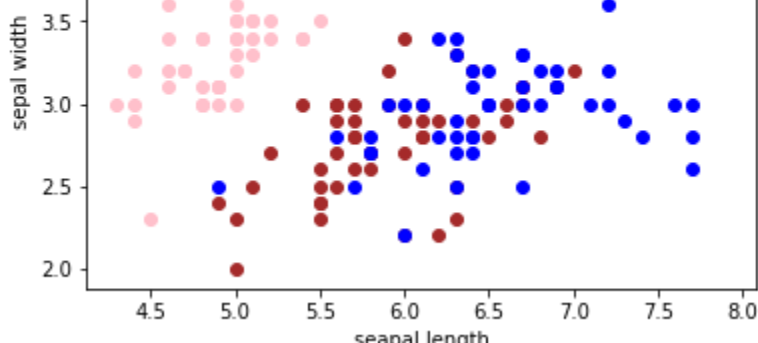
```
In [13]: i_data['class'].hist()
```



```
In [14]: colors = ['pink', 'brown', 'blue']
Class = ['Iris-setosa', 'Iris-versicolor', 'Iris-virginica']
```

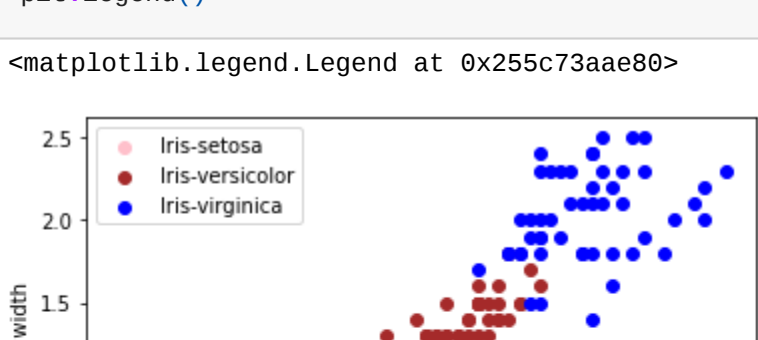
```
In [15]: for i in range(3):
x = i_data[i_data['class'] == Class[i]]
plt.scatter(x['sepal length'], x['sepal width'], c = colors[i], label=Class[i])
plt.xlabel("sepal length")
plt.ylabel("sepal width")
plt.legend()
```

Out[15]: <matplotlib.legend.Legend at 0x255c731ae00>



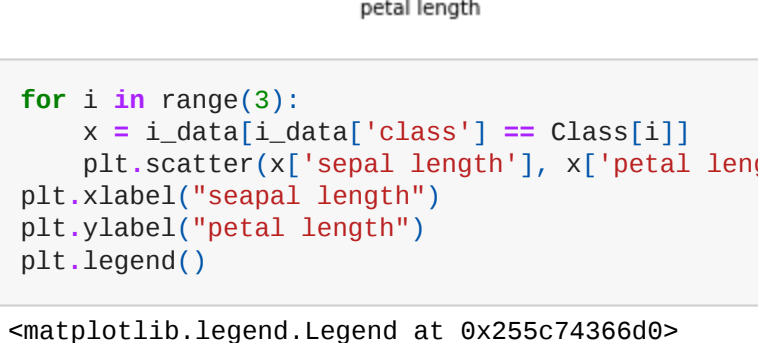
```
In [16]: for i in range(3):
x = i_data[i_data['class'] == Class[i]]
plt.scatter(x['petal length'], x['petal width'], c = colors[i], label=Class[i])
plt.xlabel("petal length")
plt.ylabel("petal width")
plt.legend()
```

Out[16]: <matplotlib.legend.Legend at 0x255c73aae80>



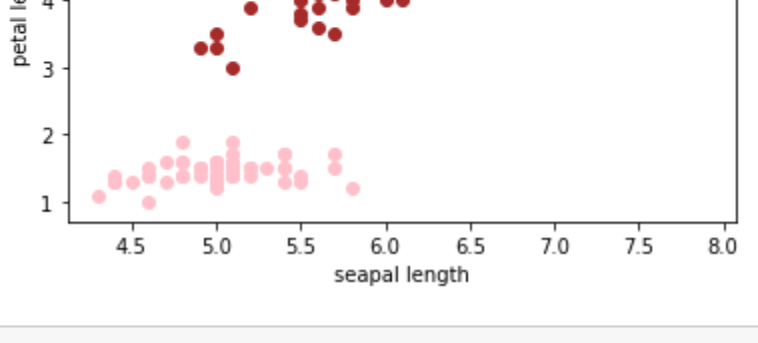
```
In [17]: for i in range(3):
x = i_data[i_data['class'] == Class[i]]
plt.scatter(x['sepal length'], x['petal length'], c = colors[i], label=Class[i])
plt.xlabel("sepal length")
plt.ylabel("petal length")
plt.legend()
```

Out[17]: <matplotlib.legend.Legend at 0x255c7436d0>

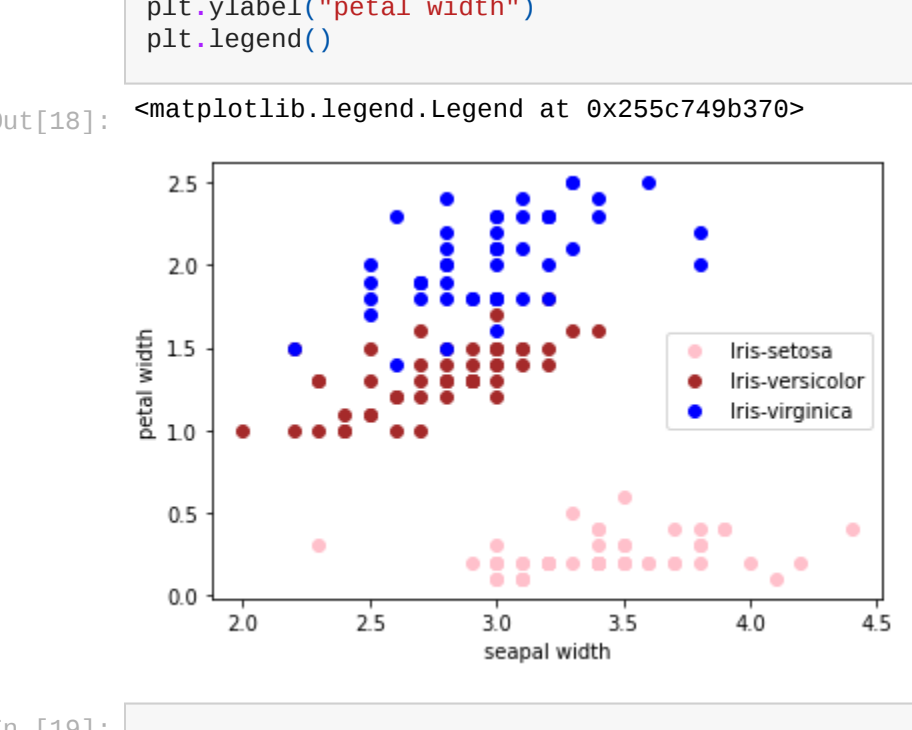


```
In [18]: for i in range(3):
x = i_data[i_data['class'] == Class[i]]
plt.scatter(x['sepal width'], x['petal width'], c = colors[i], label=Class[i])
plt.xlabel("sepal width")
plt.ylabel("petal width")
plt.legend()
```

Out[18]: <matplotlib.legend.Legend at 0x255c749b370>

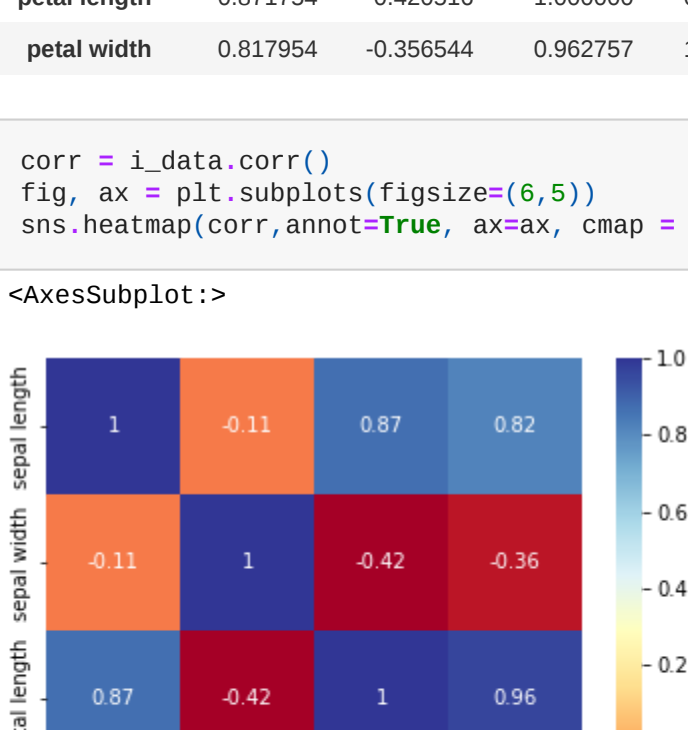


```
In [19]: i_data.corr()
```



```
In [20]: corr = i_data.corr()
fig, ax = plt.subplots(figsize=(6,5))
sns.heatmap(corr,annot=True,ax=ax, cmap = 'RdYlBu' )
```

Out[20]: <AxesSubplot:>



```
In [21]: le = LabelEncoder()
i_data['class'] = le.fit_transform(i_data['class'])
i_data.head()
```

```
Out[21]:   sepal length  sepal width  petal length  petal width  class
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 [22]: i_data.tail()
```

```
Out[22]:   sepal length  sepal width  petal length  petal width  class
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
```

Model Building

```
In [23]: #train data = 80
#test data = 20
x = i_data.drop(columns=['class'])
y = i_data['class']
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.20)
```

```
In [24]: sc=StandardScaler()
x_train=sc.fit_transform(x_train)
x_test=sc.fit_transform(x_test)
```

Logistic Regression

```
In [25]: model = LogisticRegression()
```

```
In [26]: model.fit(x_train, y_train)
```

```
Out[26]: LogisticRegression()
```

```
In [27]: print("Accuracy:", model.score(x_test, y_test)*100)
```

Accuracy: 98.0

Prediction on Given Input

```
In [37]: pre=[[4.6,3.4,1.4,0.3]]
pre
```

Out[37]: [[4.6, 3.4, 1.4, 0.3]]

```
In [38]: num=sc.fit_transform(pre)
```

```
In [39]: result=model.predict(pre)
result[0]
```

Out[39]: 1

```
In [40]: pre=[[6.9,3.1,4.9,1.5]]
pre
num=sc.fit_transform(pre)
result=model.predict(pre)
result[0]
```

Out[40]: 2

```
In [ ]:
```