



MAJOR PROJECT

NAME

P SASI KUMAR

SUB

TEACHNOOK DATASCIENCE
MAJOR PROJECT

TOPICS

TAKE ANY DATASET OF YOUR CHOICE
,PERFORM EDA(EXPLORATORY DATA
ANALYSIS) AND APPLY A
SUITABLE CLASSIFIER,REGRESSOR
OR CLUSTERER AND CALCULATE THE
ACCURACY OF THE MODEL.

SUBMITTED TO

TEACHNOOK
IIT BHUVANESHWAR



```
#IRIS FLOWER DATASET - Logistic Regression
#dataset - IRIS FLOWER DATASET
#dataset -
https://raw.githubusercontent.com/ameenmanna8824/DATASETS/main/IRIS.csv
v
```

```
#1. Take the data and create dataframe
```

```
import pandas as pd
df =
pd.read_csv('https://raw.githubusercontent.com/ameenmanna8824/DATASETS/main/IRIS.csv')
df
```

	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					
..	
...					
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					

```
[150 rows x 5 columns]
```

```
#2. step no 2 not required
```

```
fsize = df.groupby('species', sort = False).size()
fsize
```

```
species
Iris-setosa      50
Iris-versicolor  50
Iris-virginica   50
dtype: int64
```

```

#3.Step no 3 not required
#We cannot apply visualisation for CLASSIFICATION MODEL

#4.Divide the data into input and output

#input - sepal length,sepal width, petal length,petal width
#output - species

x = df.iloc[:,4].values
y = df.iloc[:,4].values

#5.Train and Test variables
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,random_state =0)
#Random_state is used to avoid duplicates

#6.Normalisation or Scaling
#here step no 6 is not required because our inputs are already scaled

#7.Apply a classifier regressor or clusterer
from sklearn.linear_model import LogisticRegression
model = LogisticRegression

#8.Fit the model
from sklearn.linear_model import LogisticRegression

# Initialize the model
model = LogisticRegression()

# Fit the model to the training data
model.fit(x_train, y_train)

LogisticRegression()

#10.predict the output
y_pred = model.predict(x_test) #using the input testing values , we
predict the output.
y_pred

array(['Iris-virginica', 'Iris-versicolor', 'Iris-setosa',
       'Iris-virginica', 'Iris-setosa', 'Iris-virginica', 'Iris-
setosa',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-virginica', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-setosa',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-setosa', 'Iris-
setosa',
       'Iris-virginica', 'Iris-versicolor', 'Iris-setosa', 'Iris-
setosa',
       'Iris-virginica', 'Iris-setosa', 'Iris-setosa', 'Iris-
versicolor',
       'Iris-versicolor', 'Iris-setosa', 'Iris-virginica',

```

```

        'Iris-versicolor', 'Iris-setosa', 'Iris-virginica',
        'Iris-virginica', 'Iris-versicolor', 'Iris-setosa',
        'Iris-virginica'], dtype=object)

y_test #Actual oupit

array(['Iris-virginica', 'Iris-versicolor', 'Iris-setosa',
        'Iris-virginica', 'Iris-setosa', 'Iris-virginica', 'Iris-
        setosa',
        'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor',
        'Iris-virginica', 'Iris-versicolor', 'Iris-versicolor',
        'Iris-versicolor', 'Iris-versicolor', 'Iris-setosa',
        'Iris-versicolor', 'Iris-versicolor', 'Iris-setosa', 'Iris-
        setosa',
        'Iris-virginica', 'Iris-versicolor', 'Iris-setosa', 'Iris-
        setosa',
        'Iris-virginica', 'Iris-setosa', 'Iris-setosa', 'Iris-
        versicolor',
        'Iris-versicolor', 'Iris-setosa', 'Iris-virginica',
        'Iris-versicolor', 'Iris-setosa', 'Iris-virginica',
        'Iris-virginica', 'Iris-versicolor', 'Iris-setosa',
        'Iris-versicolor'], dtype=object)

```

#10.Accuracy

```

from sklearn.metrics import accuracy_score
accuracy_score(y_pred,y_test)*100

```

97.36842105263158

#INDIVIDUAL PREDICTION

```

model.predict([[5.1,3.1,1.4,0.2]])

array(['Iris-setosa'], dtype=object)

model.predict([[5.7,2.8,4.5,1.3]])

array(['Iris-versicolor'], dtype=object)

model.predict([[7.7,3.8,6.7,2.2]])

array(['Iris-virginica'], dtype=object)

```

#Custom values

```

model.predict([[6.8,3.3,7.9,4.5]])

array(['Iris-virginica'], dtype=object)

```

#EXPLORATORY DATA ANALYSIS(EDA) - PRE MACHINE LEARNING

#EDA - WE FIND DIFFERENT INSIGHTS and WE COME TO CONCLUSIONS using our DATA

```
#Dataset:https://raw.githubusercontent.com/ameenmanna8824/DATASETS/main/IRIS.csv

#import pandas as pd
#df =
pd.read_csv('https://raw.githubusercontent.com/ameenmanna8824/DATASETS/main/IRIS.csv')
#df
```

```
import pandas as pd
df =
pd.read_csv('https://raw.githubusercontent.com/ameenmanna8824/DATASETS/main/IRIS.csv')
df
```

	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					
..	
...					
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					

[150 rows x 5 columns]

df.head()

	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

```
df.size
```

```
750
```

```
df.shape
```

```
(150, 5)
```

```
df.info
```

```
<bound method DataFrame.info of
petal_length  petal_width  sepal_length  sepal_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
..            ...          ...           ...          ...
...
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
```

```
[150 rows x 5 columns]>
```

```
df.sample(6)
```

```
      sepal_length  sepal_width  petal_length  petal_width
species
30            4.8          3.1           1.6          0.2  Iris-
setosa
104           6.5          3.0           5.8          2.2  Iris-
virginica
14            5.8          4.0           1.2          0.2  Iris-
setosa
148           6.2          3.4           5.4          2.3  Iris-
virginica
77            6.7          3.0           5.0          1.7  Iris-
versicolor
```

```
53          5.5          2.3          4.0          1.3  Iris-  
versicolor
```

```
df.species.unique()
```

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

```
df.describe(include='all')
```

	sepal_length	sepal_width	petal_length	petal_width
species				
count	150.000000	150.000000	150.000000	150.000000
150				
unique	NaN	NaN	NaN	NaN
3				
top	NaN	NaN	NaN	NaN
setosa				
freq	NaN	NaN	NaN	NaN
50				
mean	5.843333	3.054000	3.758667	1.198667
NaN				
std	0.828066	0.433594	1.764420	0.763161
NaN				
min	4.300000	2.000000	1.000000	0.100000
NaN				
25%	5.100000	2.800000	1.600000	0.300000
NaN				
50%	5.800000	3.000000	4.350000	1.300000
NaN				
75%	6.400000	3.300000	5.100000	1.800000
NaN				
max	7.900000	4.400000	6.900000	2.500000
NaN				

```
df.corr()
```

```
<ipython-input-31-2f6f6606aa2c>:1: FutureWarning: The default value of  
numeric_only in DataFrame.corr is deprecated. In a future version, it  
will default to False. Select only valid columns or specify the value  
of numeric_only to silence this warning.
```

```
df.corr()
```

	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

```
df.isnull().sum()
```

```

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

```

```

df_setosa = df.loc[df['species']=='setosa']
df_setosa.describe()

```

	sepal_length	sepal_width	petal_length	petal_width
count	0.0	0.0	0.0	0.0
mean	NaN	NaN	NaN	NaN
std	NaN	NaN	NaN	NaN
min	NaN	NaN	NaN	NaN
25%	NaN	NaN	NaN	NaN
50%	NaN	NaN	NaN	NaN
75%	NaN	NaN	NaN	NaN
max	NaN	NaN	NaN	NaN

```

df_virginica = df.loc[df['species']=='virginica']
df_virginica.describe()

```

	sepal_length	sepal_width	petal_length	petal_width
count	0.0	0.0	0.0	0.0
mean	NaN	NaN	NaN	NaN
std	NaN	NaN	NaN	NaN
min	NaN	NaN	NaN	NaN
25%	NaN	NaN	NaN	NaN
50%	NaN	NaN	NaN	NaN
75%	NaN	NaN	NaN	NaN
max	NaN	NaN	NaN	NaN

```

#Slice row indexes from 25 to 43 and column indexes 0 and 1
#var.iloc[row slicing,column slicing]
df.iloc[40:64,2:5]

```

	petal_length	petal_width	species
40	1.3	0.3	Iris-setosa
41	1.3	0.3	Iris-setosa
42	1.3	0.2	Iris-setosa
43	1.6	0.6	Iris-setosa
44	1.9	0.4	Iris-setosa
45	1.4	0.3	Iris-setosa
46	1.6	0.2	Iris-setosa
47	1.4	0.2	Iris-setosa
48	1.5	0.2	Iris-setosa
49	1.4	0.2	Iris-setosa
50	4.7	1.4	Iris-versicolor
51	4.5	1.5	Iris-versicolor
52	4.9	1.5	Iris-versicolor

53	4.0	1.3	Iris-versicolor
54	4.6	1.5	Iris-versicolor
55	4.5	1.3	Iris-versicolor
56	4.7	1.6	Iris-versicolor
57	3.3	1.0	Iris-versicolor
58	4.6	1.3	Iris-versicolor
59	3.9	1.4	Iris-versicolor
60	3.5	1.0	Iris-versicolor
61	4.2	1.5	Iris-versicolor
62	4.0	1.0	Iris-versicolor
63	4.7	1.4	Iris-versicolor

#Let us consider the column species

#Now I want to know the exact count of the unique flower names

```
df['species'].nunique()
```

3

#Now I want the unique fruit names

```
fname = df['species'].unique()
```

```
fname
```

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

#Now I want to find out the exact count of each and every flower

```
#Iris-setosa      - 50
```

```
#Iris-versicolor  - 50
```

```
#Iris-virginica   - 50
```

```
fsize = df.groupby('species', sort = False).size()
```

```
fsize
```

#By default ,.groupby sets the values in Alphabetical order

```
species
```

```
Iris-setosa      50
```

```
Iris-versicolor  50
```

```
Iris-virginica   50
```

```
dtype: int64
```

```
import seaborn as sns
```

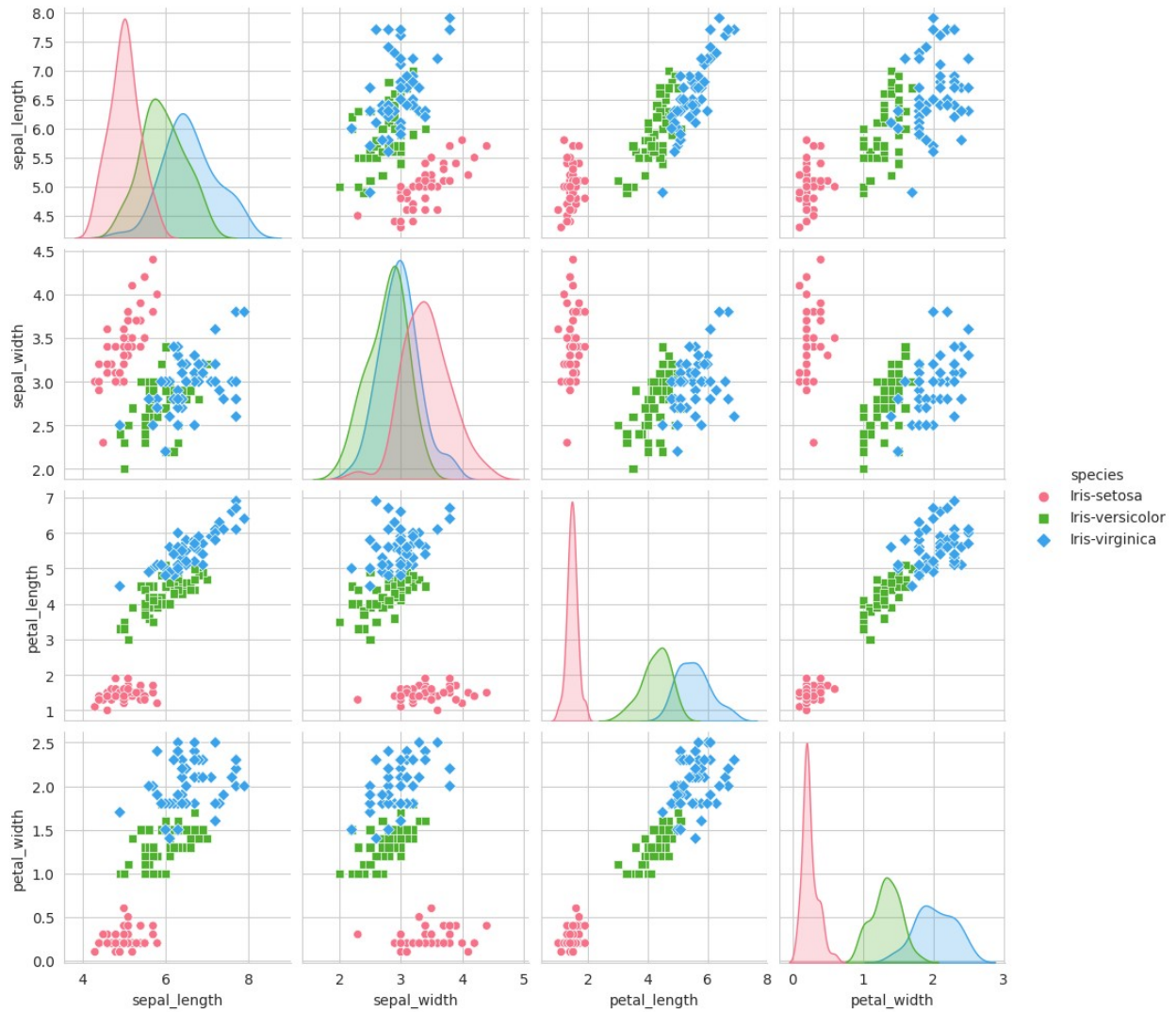
```
import matplotlib.pyplot as plt
```

```
plt.close()
```

```
sns.set_style('whitegrid')
```

```
sns.pairplot(df, hue="species", markers=["o", "s", "D"],
             palette="husl")
```

```
<seaborn.axisgrid.PairGrid at 0x7c8c8f2755a0>
```



```
plt.close()
sns.set_style('whitegrid')
sns.pairplot(df, hue='species', size=4)
```

```
/usr/local/lib/python3.10/dist-packages/seaborn/axisgrid.py:2095:
UserWarning: The `size` parameter has been renamed to `height`; please
update your code.
  warnings.warn(msg, UserWarning)
```

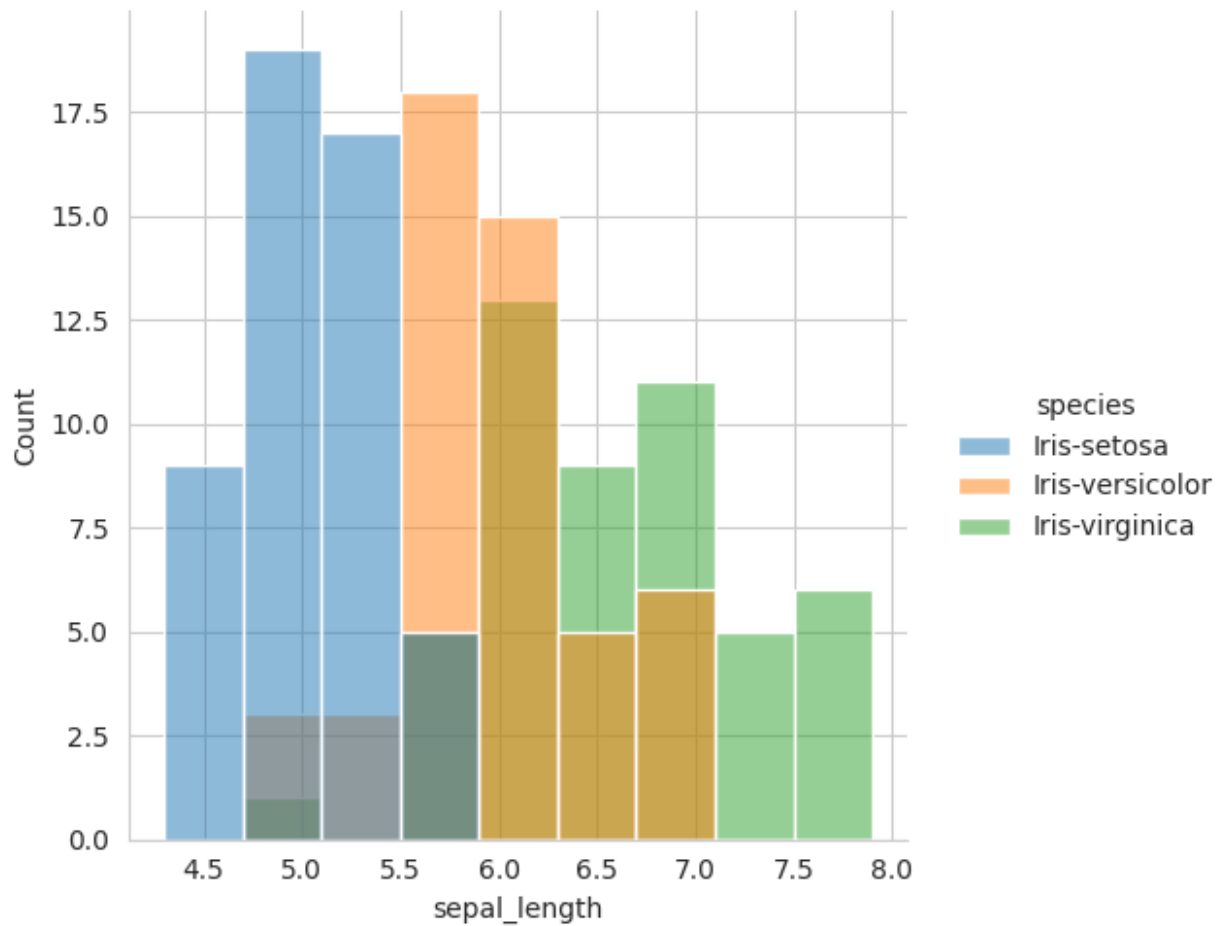
```
<seaborn.axisgrid.PairGrid at 0x7c8c81b75150>
```



```
import seaborn as sns
import matplotlib.pyplot as plt

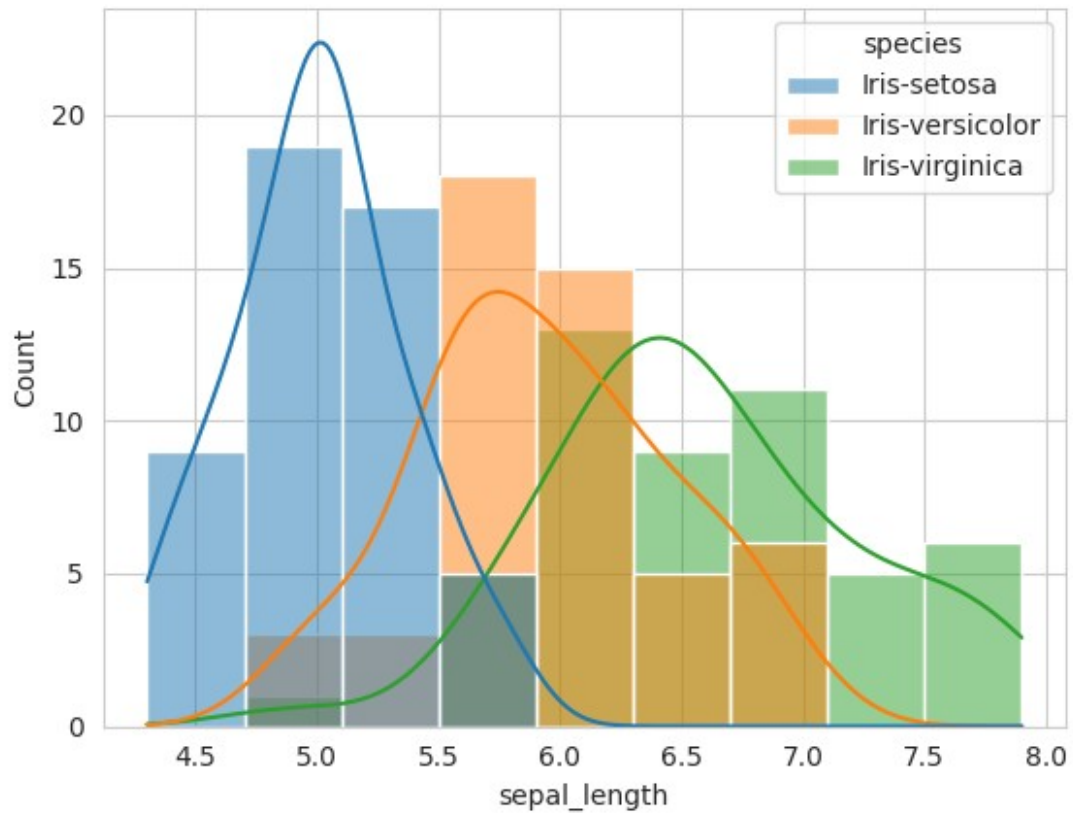
import seaborn as sns
import matplotlib.pyplot as plt

# Use displot (figure-level function) to plot histograms
sns.displot(df, x='sepal_length', hue='species', height=5)
plt.show()
```



```
import seaborn as sns
import matplotlib.pyplot as plt

# Use histplot (axes-level function) to plot histograms
sns.histplot(data=df, x='sepal_length', hue='species', kde=True)
plt.show()
```



```
sns.displot(df, x="petal_length", hue="species", height=5)  
plt.show()
```

