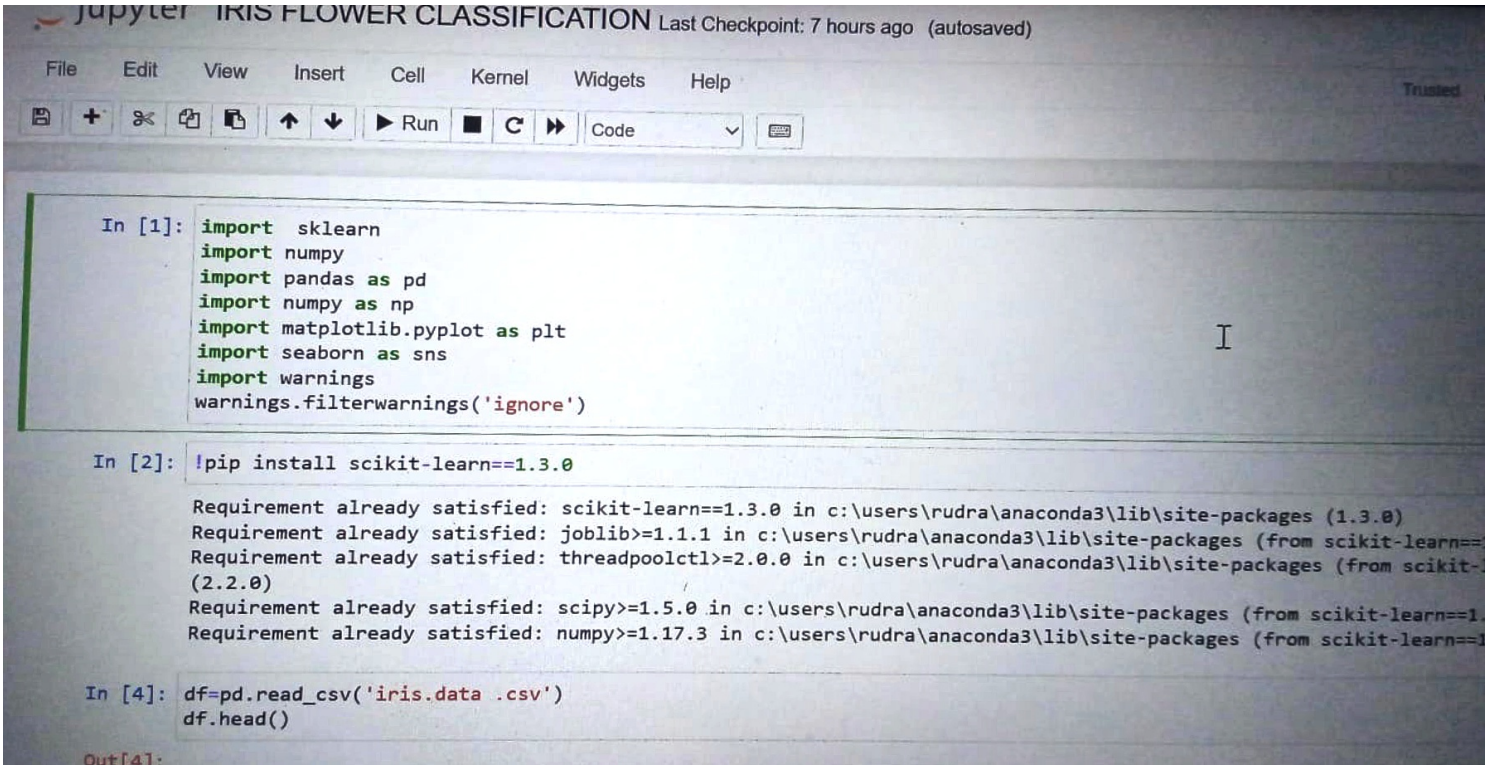
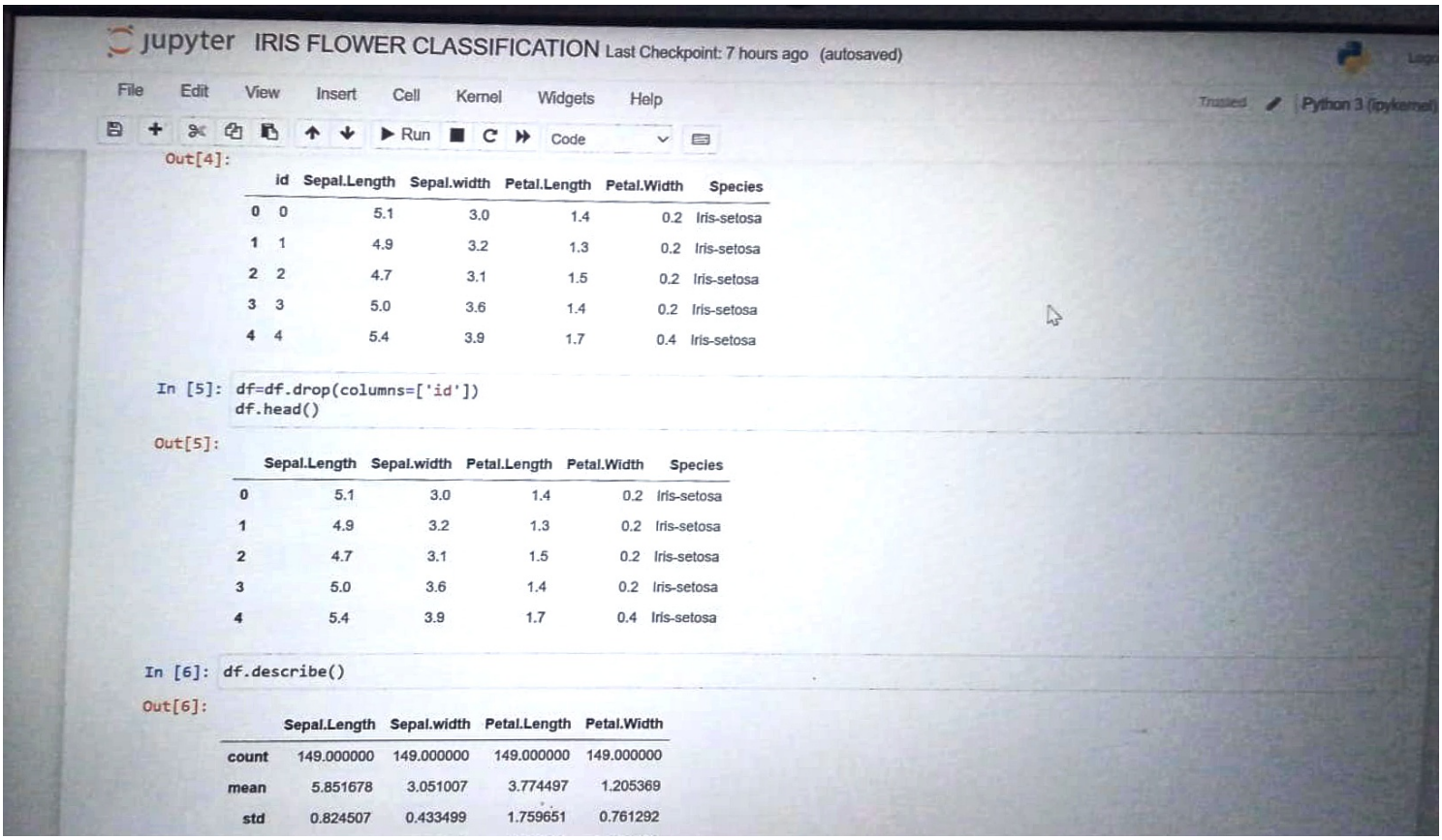
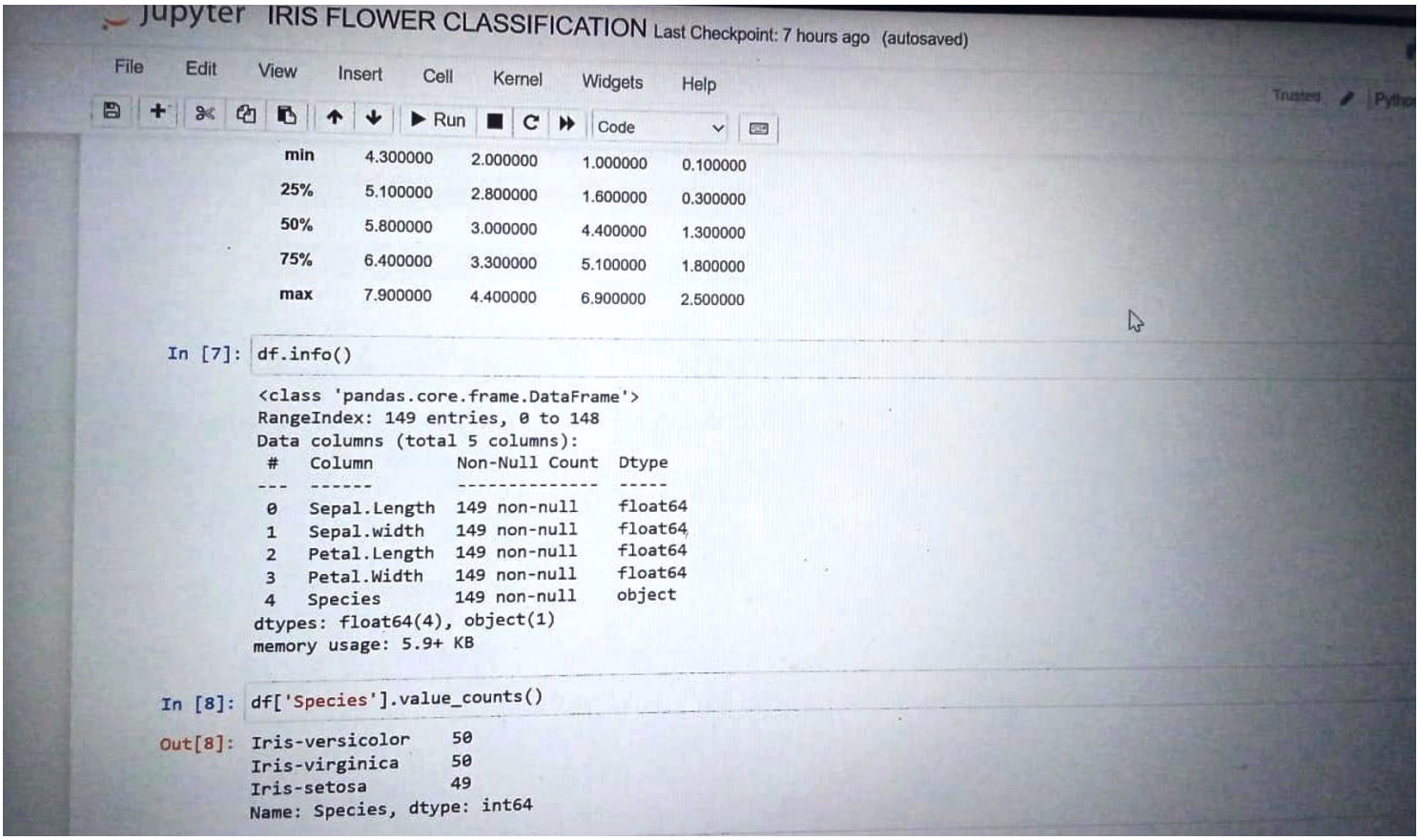

Jupyter IRIS FLOWER CLASSIFICATION Last Checkpoint: 7 hours ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help
Trusted
Run
Code
In [1]:
import sklearn
import numpy
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
In [2]:
!pip install scikit-learn==1.3.0
Requirement already satisfied: scikit-learn==1.3.0 in c:\users\rudra\anaconda3\lib\site-packages (1.3.0)
Requirement already satisfied: joblib>=1.1.1 in c:\users\rudra\anaconda3\lib\site-packages (from scikit-learn==
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\rudra\anaconda3\lib\site-packages (from scikit-
(2.2.0)
Requirement already satisfied: scipy>=1.5.0 in c:\users\rudra\anaconda3\lib\site-packages (from scikit-learn==1.
Requirement already satisfied: numpy>=1.17.3 in c:\users\rudra\anaconda3\lib\site-packages (from scikit-learn==
In [4]:
df=pd.read_csv('iris.data .csv')
df.head()
Out[4]:


jupyter IRIS FLOWER CLASSIFICATION Last Checkpoint: 7 hours ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help
Trusted Python 3 (ipykernel)
Run Code
Out[4]:
id Sepal.Length Sepal.width Petal.Length Petal.Width Species
0 0 5.1 3.0 1.4 0.2 Iris-setosa
1 1 4.9 3.2 1.3 0.2 Iris-setosa
2 2 4.7 3.1 1.5 0.2 Iris-setosa
3 3 5.0 3.6 1.4 0.2 Iris-setosa
4 4 5.4 3.9 1.7 0.4 Iris-setosa
In [5]: df=df.drop(columns=['id'])
df.head()
Out[5]:
Sepal.Length Sepal.width Petal.Length Petal.Width Species
0 5.1 3.0 1.4 0.2 Iris-setosa
1 4.9 3.2 1.3 0.2 Iris-setosa
2 4.7 3.1 1.5 0.2 Iris-setosa
3 5.0 3.6 1.4 0.2 Iris-setosa
4 5.4 3.9 1.7 0.4 Iris-setosa
In [6]: df.describe()
Out[6]:
Sepal.Length Sepal.width Petal.Length Petal.Width
count 149.000000 149.000000 149.000000 149.000000
mean 5.851678 3.051007 3.774497 1.205369
std 0.824507 0.433499 1.759651 0.761292


Jupyter IRIS FLOWER CLASSIFICATION Last Checkpoint: 7 hours ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help
Run Code
min 4.300000 2.000000 1.000000 0.100000
25% 5.100000 2.800000 1.600000 0.300000
50% 5.800000 3.000000 4.400000 1.300000
75% 6.400000 3.300000 5.100000 1.800000
max 7.900000 4.400000 6.900000 2.500000
In [7]: df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 149 entries, 0 to 148
Data columns (total 5 columns):
Column Non-Null Count Dtype
0 Sepal.Length 149 non-null float64
1 Sepal.width 149 non-null float64
2 Petal.Length 149 non-null float64
3 Petal.Width 149 non-null float64
4 Species 149 non-null object
dtypes: float64(4), object(1)
memory usage: 5.9+ KB
In [8]: df['Species'].value_counts()
Out[8]: Iris-versicolor 50
Iris-virginica 50
Iris-setosa 49
Name: Species, dtype: int64

File Edit View Insert Cell Kernel Widgets Help

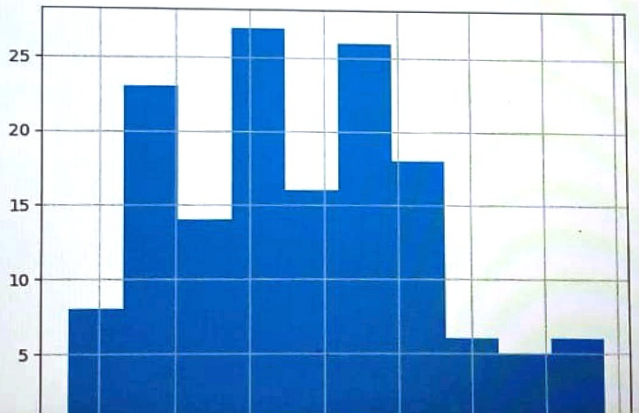
Run Code

Logout

Trusted Python 3 (ipykernel)

```
In [9]: df.isnull().sum()
Out[9]: Sepal.Length    0
        Sepal.width     0
        Petal.Length    0
        Petal.Width     0
        Species         0
        dtype: int64

In [10]: df['Sepal.Length'].hist()
Out[10]: <AxesSubplot:>
```



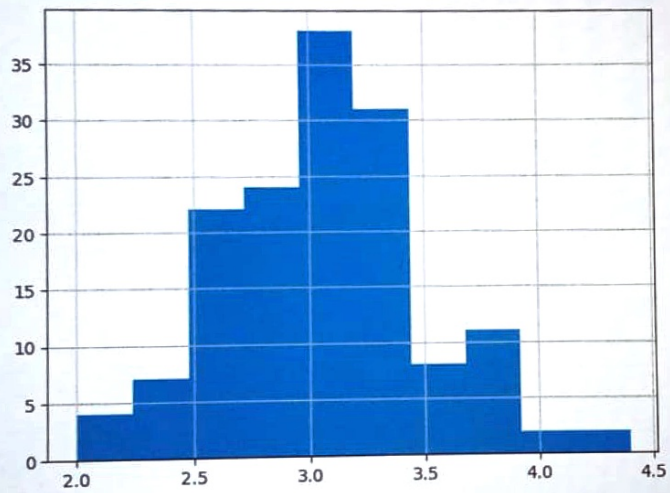
jupyter IRIS FLOWER CLASSIFICATION Last Checkpoint: 7 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Trusted Python 3 (ipykernel)

Run Code

```
In [11]: df['Sepal.width'].hist()
Out[11]: <AxesSubplot:>
```




```
In [12]: df['Petal.Length'].hist()
Out[12]: <AxesSubplot:>
```

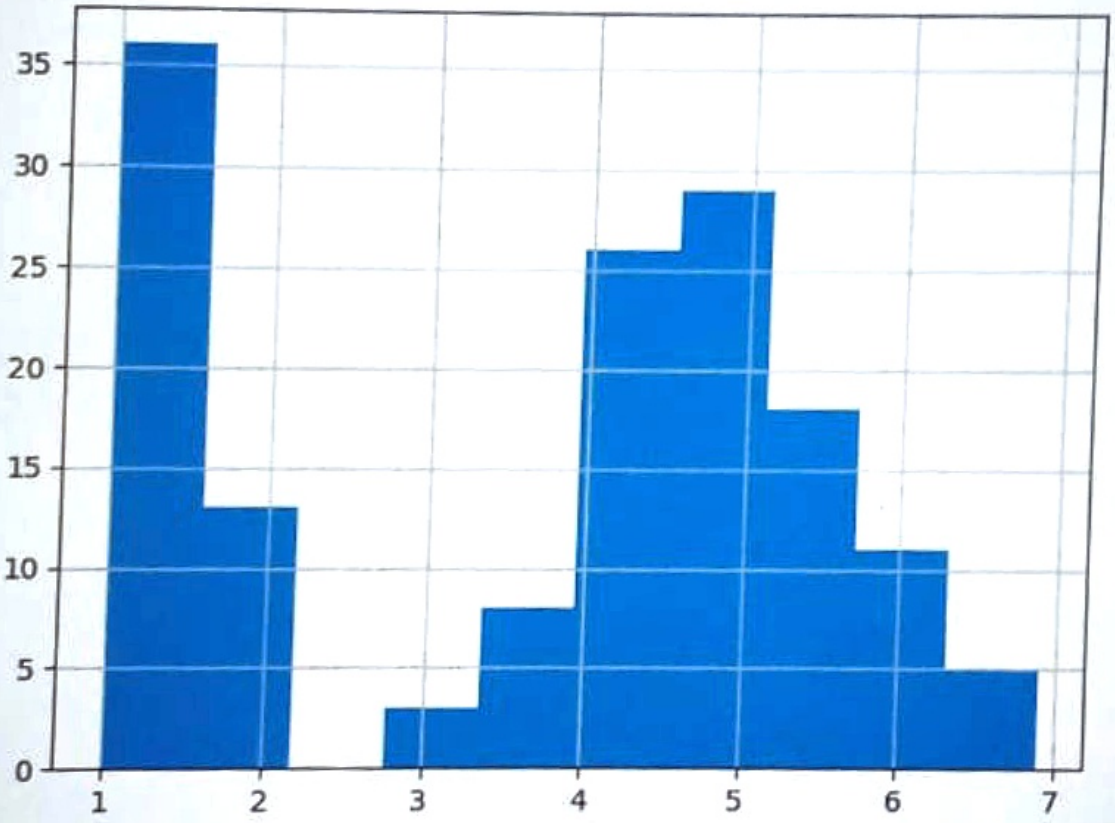
Jupyter IRIS FLOWER CLASSIFICATION Last Checkpoint: 7 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Run Code

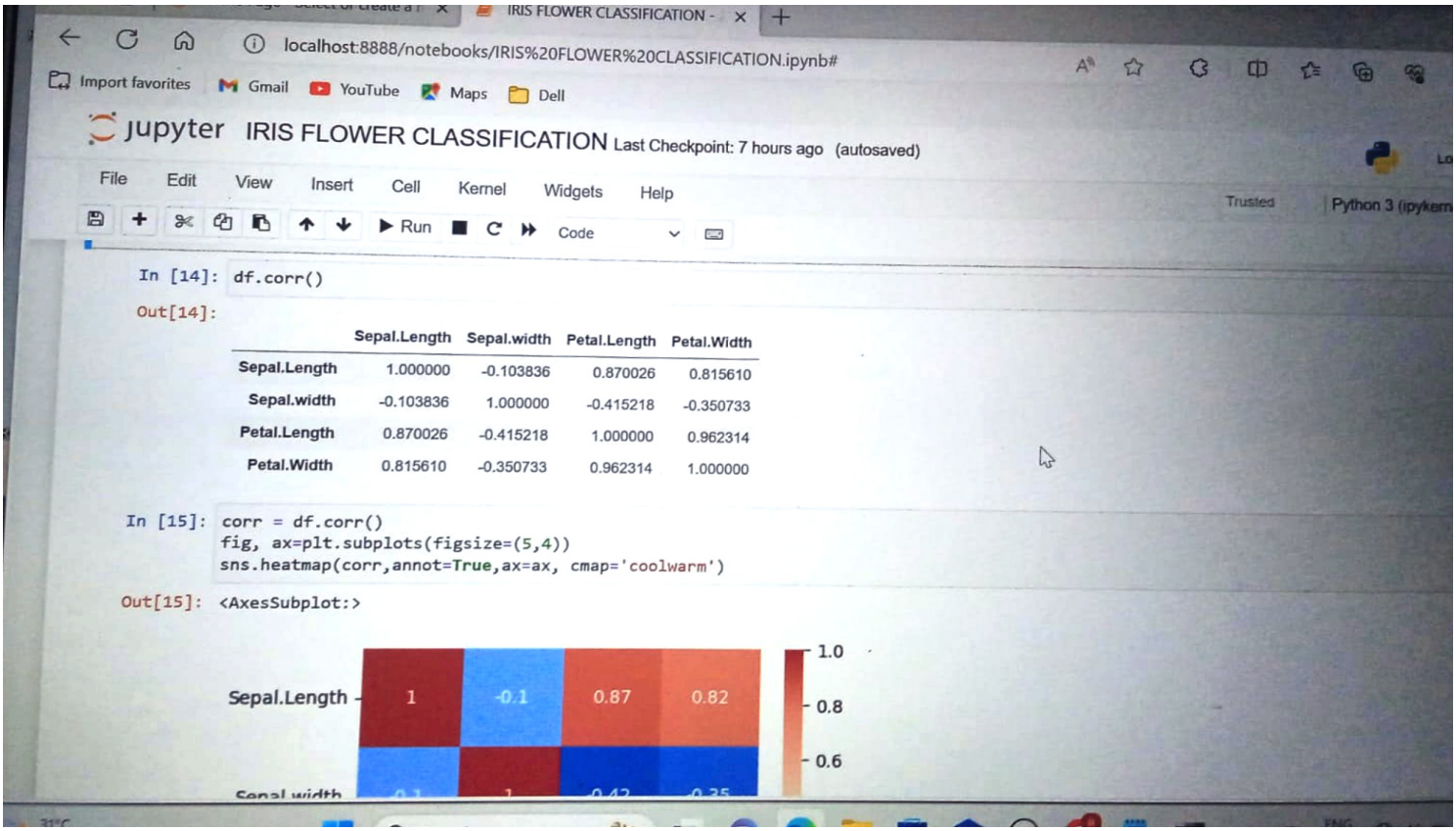
```
In [12]: df['Petal.Length'].hist()
```

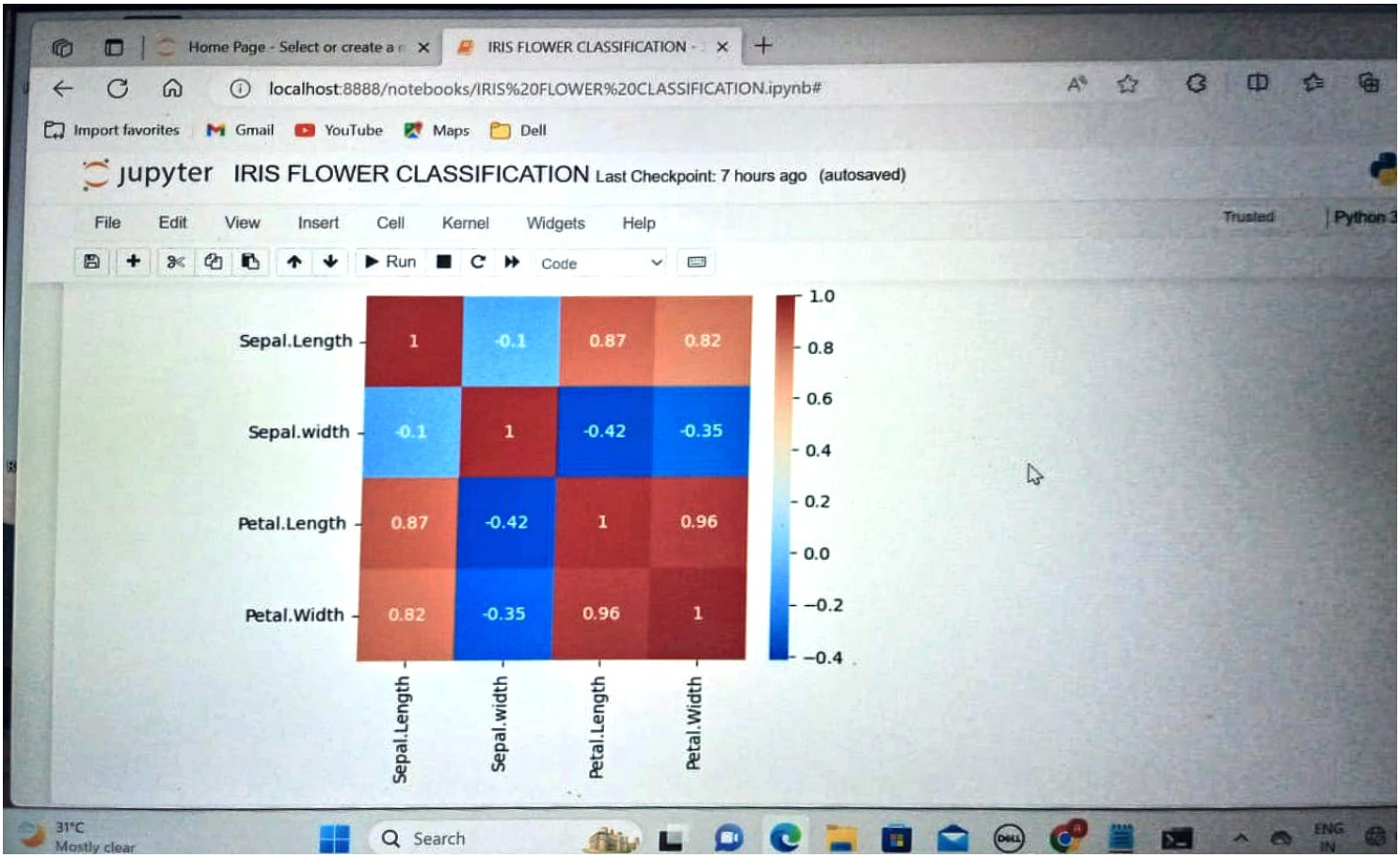
Out[12]: <AxesSubplot:>

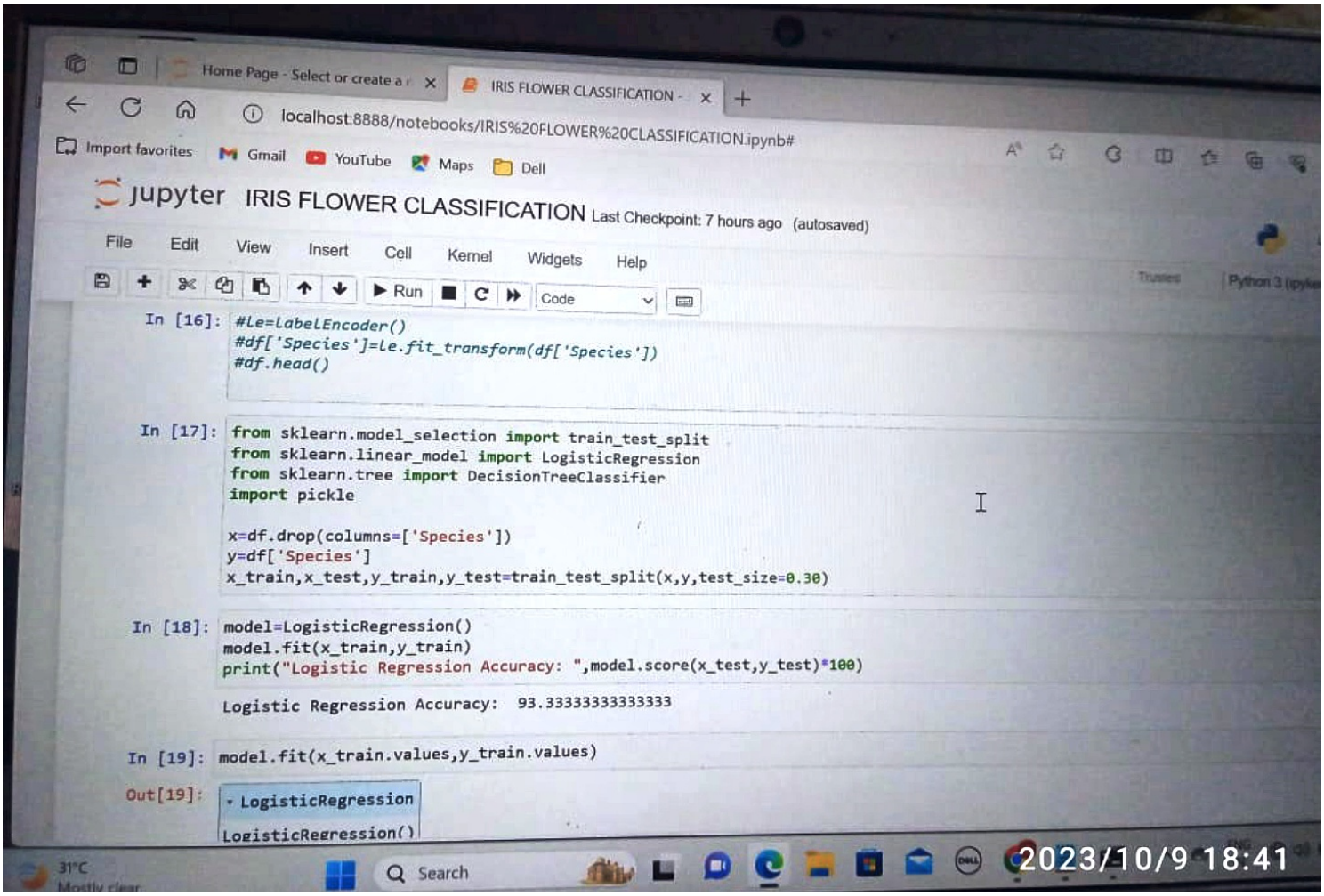


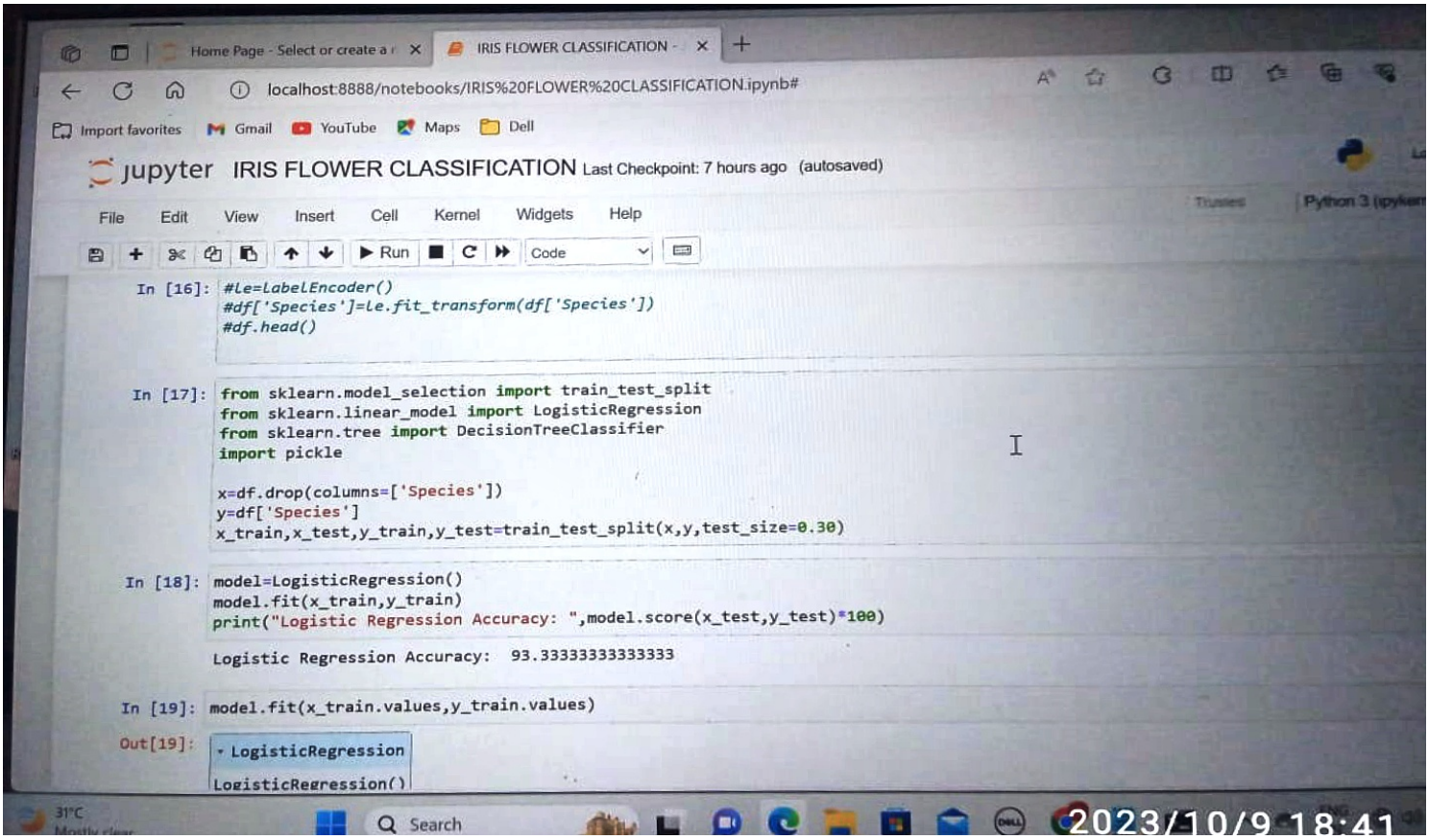

```
In [13]: df['Petal.Width'].hist()
```

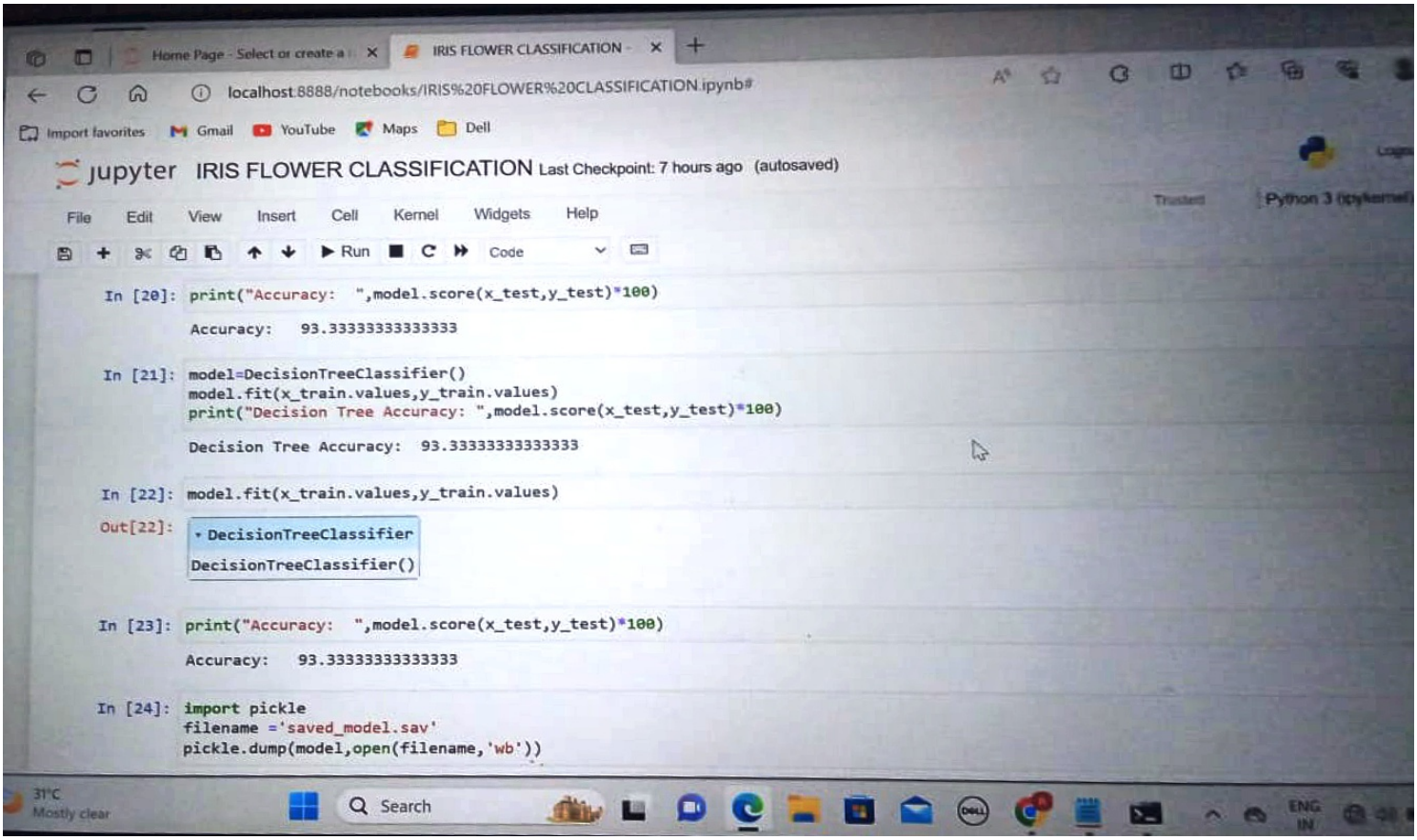
Out[13]: <AxesSubplot:>

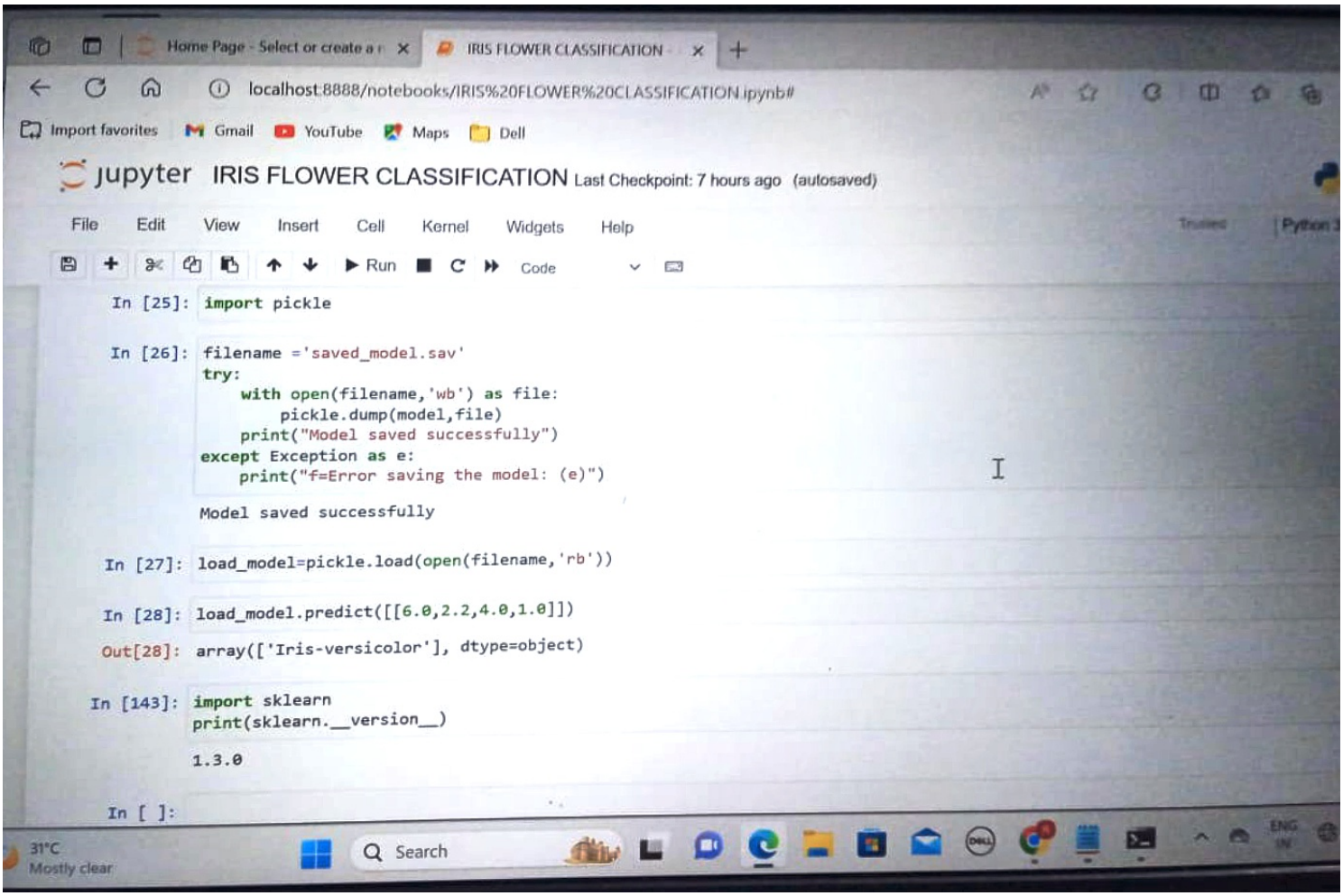

IRIS FLOWER CLASSIFICATION -
localhost:8888/notebooks/IRIS%20FLOWER%20CLASSIFICATION.ipynb#
Import favorites
Gmail
YouTube
Maps
Dell
jupyter IRIS FLOWER CLASSIFICATION Last Checkpoint: 7 hours ago (autosaved)
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Trusted
Python 3 (ipykern
Run
Code
In [14]: df.corr()
Out[14]:
Sepal.Length Sepal.width Petal.Length Petal.Width
Sepal.Length 1.000000 -0.103836 0.870026 0.815610
Sepal.width -0.103836 1.000000 -0.415218 -0.350733
Petal.Length 0.870026 -0.415218 1.000000 0.962314
Petal.Width 0.815610 -0.350733 0.962314 1.000000
In [15]: corr = df.corr()
fig, ax=plt.subplots(figsize=(5,4))
sns.heatmap(corr,annot=True,ax=ax, cmap='coolwarm')
Out[15]: <AxesSubplot:>
Sepal.Length
1
-0.1
0.87
0.82
1.0
0.8
0.6


Home Page - Select or create a
IRIS FLOWER CLASSIFICATION
localhost:8888/notebooks/IRIS%20FLOWER%20CLASSIFICATION.ipynb#
Import favorites
Gmail
YouTube
Maps
Dell
jupyter IRIS FLOWER CLASSIFICATION Last Checkpoint: 7 hours ago (autosaved)
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Trusted
Python 3
Run
Code
Sepal.Length
Sepal.width
Petal.Length
Petal.Width
1
-0.1
0.87
0.82
-0.1
1
-0.42
-0.35
0.87
-0.42
1
0.96
0.82
-0.35
0.96
1
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
Sepal.Length
Sepal.width
Petal.Length
Petal.Width
31°C
Mostly clear
Search
ENG
IN


Home Page - Select or create a
IRIS FLOWER CLASSIFICATION -
localhost:8888/notebooks/IRIS%20FLOWER%20CLASSIFICATION.ipynb#
Import favorites
Gmail
YouTube
Maps
Dell
jupyter IRIS FLOWER CLASSIFICATION Last Checkpoint: 7 hours ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help
Run
Code
In [16]:
#Le=LabelEncoder()
#df['Species']=Le.fit_transform(df['Species'])
#df.head()
In [17]:
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
import pickle
x=df.drop(columns=['Species'])
y=df['Species']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30)
In [18]:
model=LogisticRegression()
model.fit(x_train,y_train)
print("Logistic Regression Accuracy: ",model.score(x_test,y_test)*100)
Logistic Regression Accuracy: 93.33333333333333
In [19]:
model.fit(x_train.values,y_train.values)
Out[19]:
LogisticRegression
LogisticRegression()
31°C
Search
2023/10/9 18:41


Home Page - Select or create a
IRIS FLOWER CLASSIFICATION -
localhost:8888/notebooks/IRIS%20FLOWER%20CLASSIFICATION.ipynb#
Import favorites
Gmail
YouTube
Maps
Dell
jupyter IRIS FLOWER CLASSIFICATION Last Checkpoint: 7 hours ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help
Run
Code
In [16]:
#le=LabelEncoder()
#df['Species']=le.fit_transform(df['Species'])
#df.head()
In [17]:
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
import pickle
x=df.drop(columns=['Species'])
y=df['Species']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30)
In [18]:
model=LogisticRegression()
model.fit(x_train,y_train)
print("Logistic Regression Accuracy: ",model.score(x_test,y_test)*100)
Logistic Regression Accuracy: 93.33333333333333
In [19]:
model.fit(x_train.values,y_train.values)
Out[19]:
LogisticRegression
LogisticRegression()
31°C
Search
2023/10/9 18:41


Home Page - Select or create a
IRIS FLOWER CLASSIFICATION -
localhost:8888/notebooks/IRIS%20FLOWER%20CLASSIFICATION.ipynb#
Import favorites
Gmail
YouTube
Maps
Dell
jupyter IRIS FLOWER CLASSIFICATION Last Checkpoint: 7 hours ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help
Run
Code
In [20]: print("Accuracy: ",model.score(x_test,y_test)*100)
Accuracy: 93.33333333333333
In [21]: model=DecisionTreeClassifier()
model.fit(x_train.values,y_train.values)
print("Decision Tree Accuracy: ",model.score(x_test,y_test)*100)
Decision Tree Accuracy: 93.33333333333333
In [22]: model.fit(x_train.values,y_train.values)
Out[22]: DecisionTreeClassifier
DecisionTreeClassifier()
In [23]: print("Accuracy: ",model.score(x_test,y_test)*100)
Accuracy: 93.33333333333333
In [24]: import pickle
filename ='saved_model.sav'
pickle.dump(model,open(filename,'wb'))
31°C
Mostly clear
Search
ENG
IN


Home Page - Select or create a n
IRIS FLOWER CLASSIFICATION
localhost:8888/notebooks/IRIS%20FLOWER%20CLASSIFICATION.ipynb#
Import favorites
Gmail
YouTube
Maps
Dell
jupyter IRIS FLOWER CLASSIFICATION Last Checkpoint: 7 hours ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help
Run
Code
In [25]: import pickle
In [26]: filename ='saved_model.sav'
try:
with open(filename,'wb') as file:
pickle.dump(model,file)
print("Model saved successfully")
except Exception as e:
print("f=Error saving the model: (e)")
Model saved successfully
In [27]: load_model=pickle.load(open(filename,'rb'))
In [28]: load_model.predict([[6.0,2.2,4.0,1.0]])
Out[28]: array(['Iris-versicolor'], dtype=object)
In [143]: import sklearn
print(sklearn.__version__)
1.3.0
In [ ]:
31°C
Mostly clear
Search