In [ ]:

```
import pandas as pd
import numpy as np
import os
import matplotlib.pyplot as plt
import plotly.express as px
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [ ]:

```
df = pd.read_csv('Iris.csv')
df.head()
```

In [ ]:

```
df.shape
```

In [ ]:

```
df = df.drop(columns = ['Id'])
df.head()
```

In [ ]:

```
# to display stats about data
df.describe()
```

In [ ]:

```
# to basic info about datatype
df.info()
```

In [ ]:

```
df['Species'].value_counts()
```

In [ ]:

```
df.isna().sum()
```

In [ ]:

```
df['SepalLengthCm'].hist()
```

In [ ]:

```
df['SepalWidthCm'].hist()
```

In [ ]:

```
corr=df.corr()
fig,ax=plt.subplots(figsize=(5,4))
sns.heatmap(corr,annot=True,ax=ax,cmap='coolwarm')
```

In [ ]:

```
px.scatter(df, x='Species', y='PetalWidthCm')
```

In [ ]:

```
px.line(df, x='Species', y='PetalWidthCm')
```

In [ ]:

```
px.scatter(df, x='Species', y='PetalLengthCm')
```

In [ ]:

```
px.scatter(df, x='Species', y='SepalLengthCm')
```

In [ ]:

```
px.scatter(df, x='Species', y='SepalWidthCm')
```

In [ ]:

```
px.scatter_matrix(df, color='Species', title='Iris', dimensions=['SepalLengthCm','SepalWidt
                                                                 'PetalLengthCm','PetalWid
```

In [ ]:

```
# scatterplot
colors = ['red', 'orange', 'blue']
species = ['Iris-virginica','Iris-versicolor','Iris-setosa']
```

In [ ]:

```
for i in range(3):
    x = df[df['Species'] == species[i]]
    plt.scatter(x['SepalLengthCm'], x['SepalWidthCm'], c = colors[i], label=species[i])
plt.xlabel("Sepal Length")
plt.ylabel("Sepal Width")
plt.legend()
```

In [ ]:

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
```

In [ ]:

```
df['Species'] = le.fit_transform(df['Species'])
df.head()
```

In [ ]:

```
from sklearn.model_selection import train_test_split
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 [ ]:

```
# logistic regression
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
```

In [ ]:

```
# model training
model.fit(x_train, y_train)
```

In [ ]:

```
# print metric to get performance
print("Accuracy: ",model.score(x_test, y_test) * 100)
```

In [ ]:

```
# decision tree
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report,f1_score

model = DecisionTreeClassifier()
```

In [ ]:

```
model.fit(x_train, y_train)
```

In [ ]:

```
# print metric to get performance
print("Accuracy: ",model.score(x_test, y_test) * 100)
```

In [ ]:

```
data = 8,3.755,7,2.1
```

In [ ]:

```
data_array = np.array([data])
data_array
```

In [ ]:

```
predic = model.predict(data_array)
```

In [ ]:

```
predic
```

In [ ]:

```
catagory = ['Iris-Satosa','Iris-Versicolor','Iris-Virginica']
```

In [ ]:

```
print(catagory[int(predic[0])])
```

In [ ]:

```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier()
# training
model.fit(x_train, y_train)
# testing
y_pred = model.predict(x_test)
print(classification_report(y_test, y_pred))
print(f1_score(y_test, y_pred))
```

In [ ]:

```

```

In [ ]:

```

```