```
Start coding or generate with AI.
```

```
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
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score,confusion_matrix,precision_score,recall_score
```

```
data=pd.read_csv("Iris.csv")
```

```
data.head()
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

```
data.shape
```

```
(150, 6)
```

```
data.columns
```

```
Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm',
       'Species'],
      dtype='object')
```

```
data.describe()
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	75.500000	5.843333	3.054000	3.758667	1.198667
std	43.445368	0.828066	0.433594	1.764420	0.763161
min	1.000000	4.300000	2.000000	1.000000	0.100000
25%	38.250000	5.100000	2.800000	1.600000	0.300000
50%	75.500000	5.800000	3.000000	4.350000	1.300000
75%	112.750000	6.400000	3.300000	5.100000	1.800000
max	150.000000	7.900000	4.400000	6.900000	2.500000

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             150 non-null    int64
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

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

```
Id              0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

```
data['Species'].nunique()
```

```
3
```

```
data['Species'].unique()
```

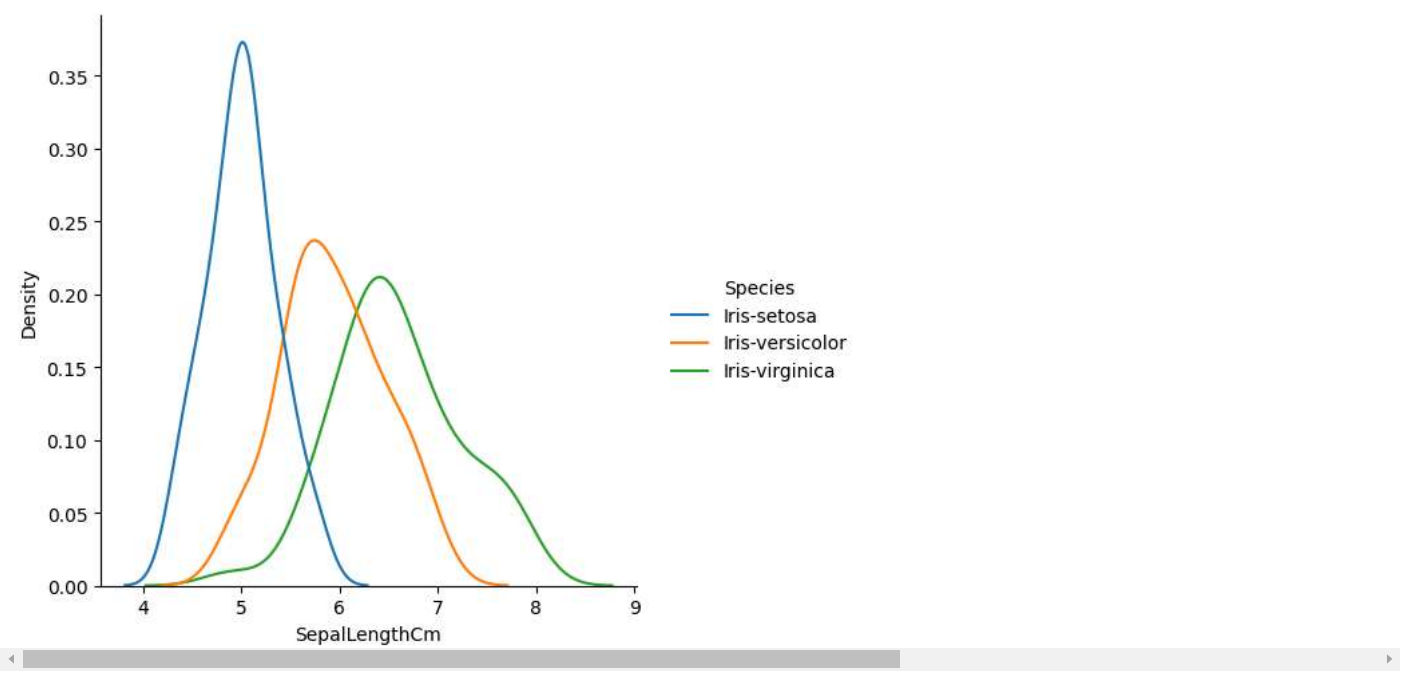
```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)
```

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

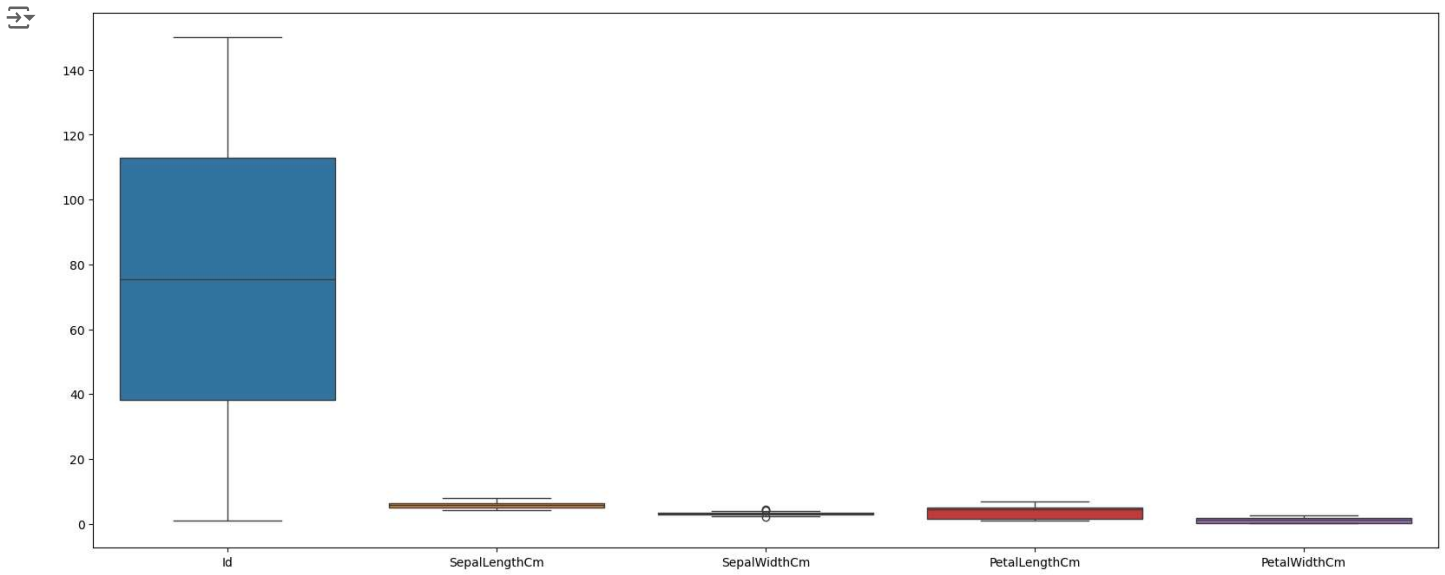
```
Species
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: count, dtype: int64
```

```
#No Missing values
#No outliers
#But our target is object data type,need to handle it
#Need to perform scaling
```

```
sns.displot(data=data,x="SepalLengthCm",kind="kde",hue="Species")
plt.show()
```




```
plt.figure(figsize =(20,8))
sns.boxplot(data=data,width=0.8)
plt.show()
```




```
#No outliers
#We can also drop some unwanted columns which are not useful for training our model like Id
#Divide the data into train and test


#But first divide it into x and y
x=data.drop(['Species','Id'],axis=1)
y=data['Species']


x.shape,y.shape
```

```
((150, 4), (150,))
```

```
#Divide the data into train and test


x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=1,stratify=y)


x_train.shape,y_train.shape
```

```
((120, 4), (120,))
```

```
x_test.shape,y_test.shape
```

```
((30, 4), (30,))
```

```
y_test.value_counts()
```

```
Species
Iris-virginica     10
Iris-setosa        10
Iris-versicolor    10
Name: count, dtype: int64
```

```
y_train.value_counts()
```

```
Species
Iris-setosa        40
Iris-virginica     40
Iris-versicolor    40
Name: count, dtype: int64
```

```
#Now we need to perform Standardization(scaling) and Encoding(dealing object data type)


ss=StandardScaler()
x_train_scaled=ss.fit_transform(x_train)
x_test_scaled=ss.transform(x_test)



type(x_train_scaled)
```

```
numpy.ndarray
```

```
x_train_scaled=pd.DataFrame(x_train_scaled,columns=x_train.columns)
x_test_scaled=pd.DataFrame(x_test_scaled,columns=x_test.columns)


#Our algorithm cannot take strings as input
#We'll have to convert the words into numbers
#There are two methods for this:
#     1)By using map func
#     2)LabelEncoder




#mapping_dict={"Iris-virginica":1,"Iris-setosa":2,"Iris-versicolor":3}
#y_train=y_train.map(mapping_dict)
#y_test=y_test.map(mapping_dict)
#y_train.value_counts()


from sklearn.preprocessing import OrdinalEncoder,LabelEncoder


Start coding or generate with AI.


oe=LabelEncoder() #it takes 1D array as input
y_train=oe.fit_transform(y_train)


Start coding or generate with AI.


y_train=pd.DataFrame(y_train,columns=['species'])
y_train.value_counts()
```

```
species
0          40
1          40
2          40
Name: count, dtype: int64
```

```
y_test=oe.transform(y_test)
y_test=pd.DataFrame(y_test,columns=['species'])
y_test.value_counts()
```

```
species
0          10
1          10
2          10
Name: count, dtype: int64
```

```
#Modelling

log=LogisticRegression()
log.fit(x_train,y_train)

pred=log.predict(x_test)
accuracy=accuracy_score(y_test,pred)
print(accuracy)
```

```
0.9666666666666667
/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py:1143: DataConversionWarning: A column-vector y was passed when a 1d
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
```

```
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
confusion_matrix(y_test,pred)
```

```
array([[10,  0,  0],
       [ 0, 10,  0],
       [ 0,  1,  9]])
```

```
precision_score(y_test,pred,average="weighted")
```

```
0.9696969696969696
```

```
recall_score(y_test,pred,average="weighted")
```

```
0.9666666666666667
```

Start coding or generate with AI.

```
confusion_matrix(y_test,pred)
```

```
array([[10,  0,  0],
       [ 0, 10,  0],
       [ 0,  1,  9]])
```

```
#Cross validation on the data

from sklearn.model_selection import cross_validate
cv=cross_validate(log,x_train,y_train,scoring="accuracy",cv=5)
cv['test_score']
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py:1143: DataConversionWarning: A column-vector y was passed when a 1d
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py:1143: DataConversionWarning: A column-vector y was passed when a 1d
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py:1143: DataConversionWarning: A column-vector y was passed when a 1d
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py:1143: DataConversionWarning: A column-vector y was passed when a 1d
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py:1143: DataConversionWarning: A column-vector y was passed when a 1d
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
array([0.91666667, 0.95833333, 0.95833333, 0.95833333, 0.91666667])
```