```
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import cross_val_score

import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.metrics import classification_report
import warnings
warnings.filterwarnings('ignore')
plt.style.use('ggplot')


from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

New Section

```
data=pd.read_csv('/content/Iris (1).csv')
```

```
data.sample(2)
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
112	113	6.8	3.0	5.5	2.1	Iris-virginica
125	126	7.2	3.2	6.0	1.8	Iris-virginica

```
data.shape
```

```
(150, 6)
```

```
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
```

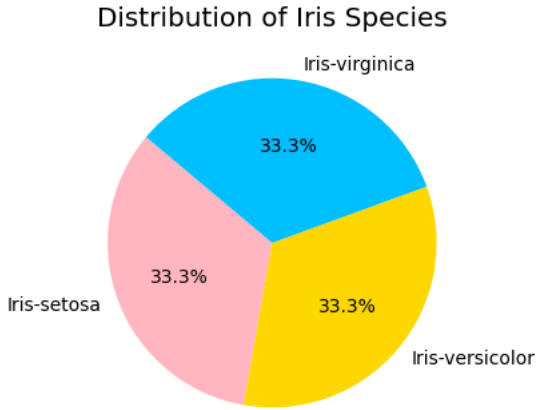
```
data.duplicated().sum()
```

```
0
```

```
colors = ['#FFB6C1', '#FFD700', '#00BFFF']

plt.figure(figsize=(4, 4))
data.groupby('Species').Species.value_counts().plot.pie(autopct='%1.1f%%',
    colors=colors,
    startangle=140)
plt.title('Distribution of Iris Species')
plt.ylabel('')
```

```
Text(0, 0.5, '')
```




```
data=data.drop(columns='Id')
```

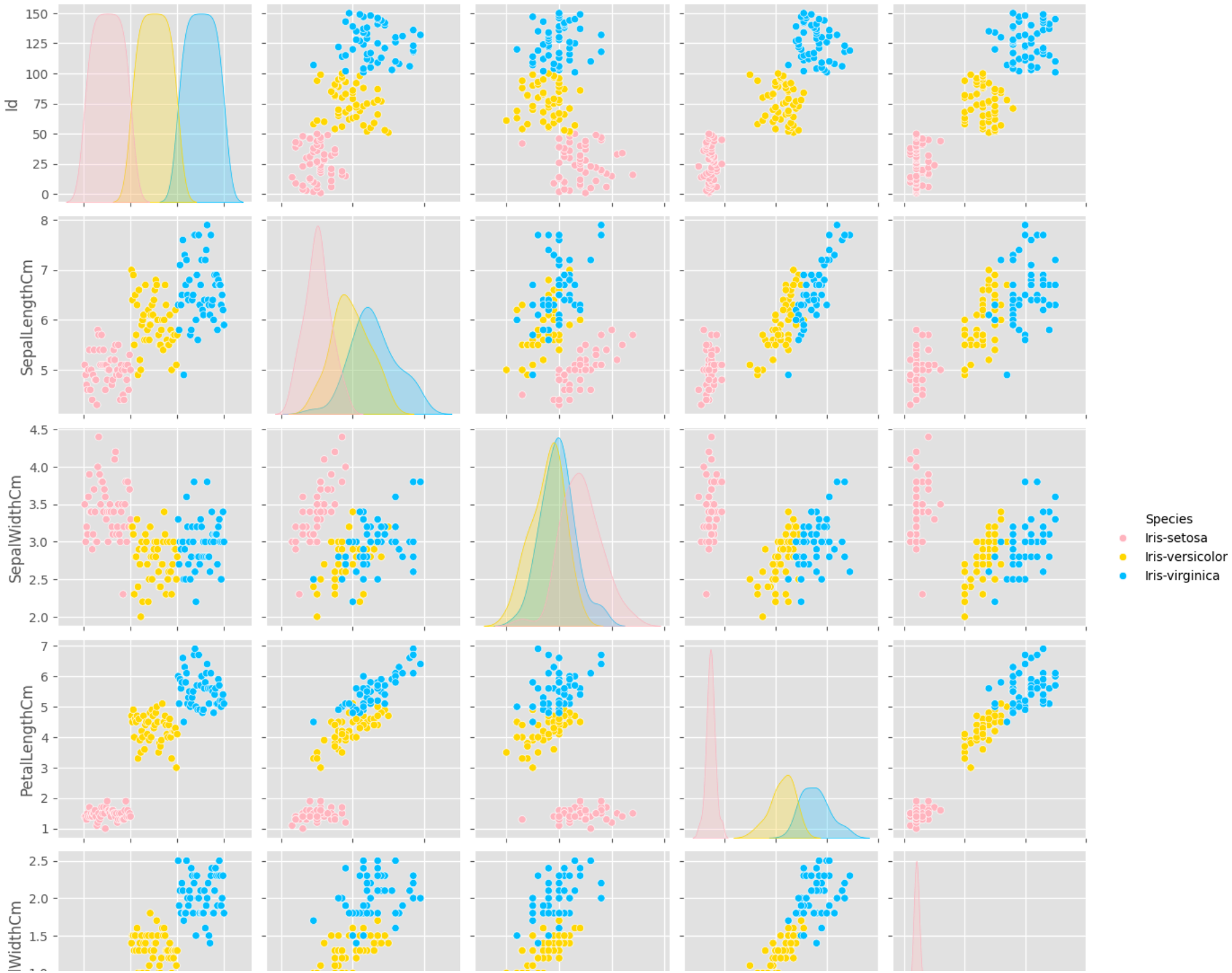
```
data['Species']=data['Species'].map({'Iris-setosa':0, 'Iris-versicolor':1, 'Iris-virginica':2 })
```

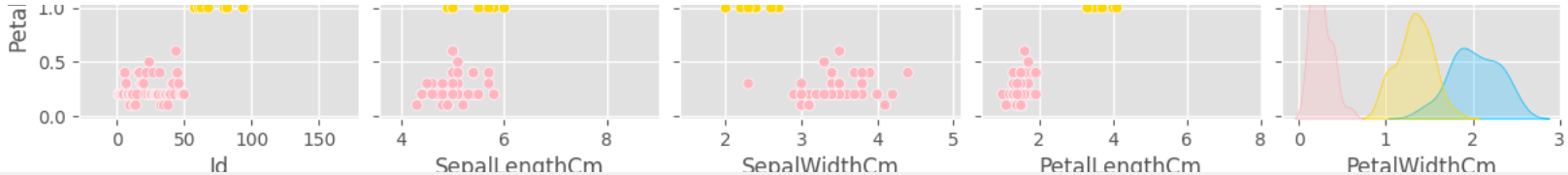
```
data.sample(2)
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
42	43	4.4	3.2	1.3	0.2	Iris-setosa
9	10	4.9	3.1	1.5	0.1	Iris-setosa

```
sns.pairplot(data,hue='Species', palette=['#FFB6C1', '#FFD700', '#00BFFF'])
```

```
<seaborn.axisgrid.PairGrid at 0x7b3f229f2ad0>
```



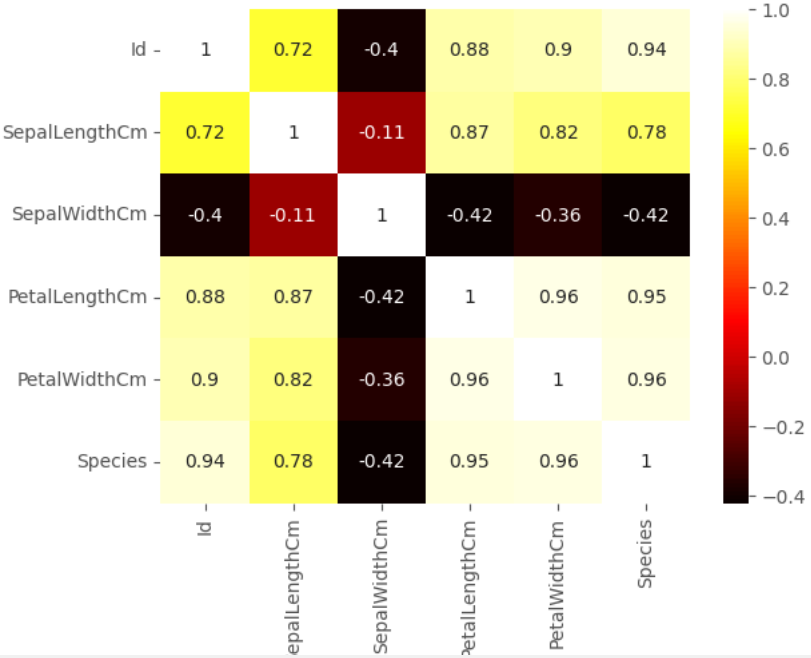



```
# Map species names to numerical labels
data['Species'] = data['Species'].map({'Iris-setosa': 0, 'Iris-versicolor': 1, 'Iris-virginica': 2})

# Verify the changes by inspecting data types
print(data.dtypes)

# Now try generating the heatmap
sns.heatmap(data.corr(), cmap='hot', annot=True)
```

```
Id               int64
SepalLengthCm    float64
SepalWidthCm     float64
PetalLengthCm    float64
PetalWidthCm     float64
Species            int64
dtype: object
<Axes: >
```

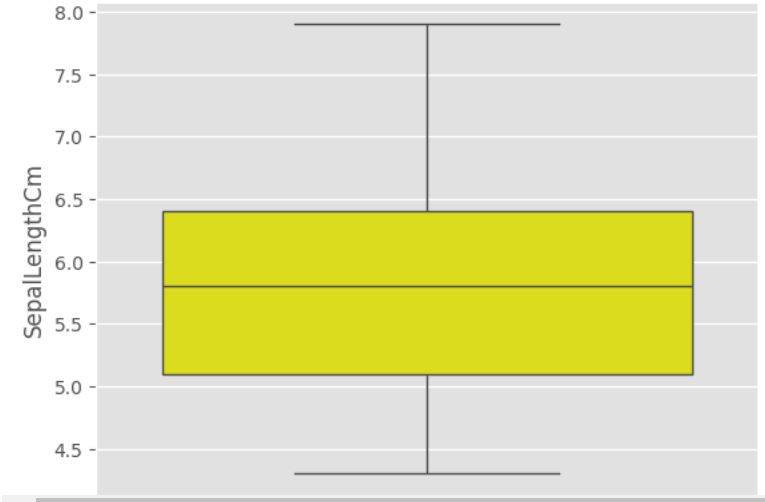



```
data.sample(2)
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
118	119	7.7	2.6	6.9	2.3	2
47	48	4.6	3.2	1.4	0.2	0

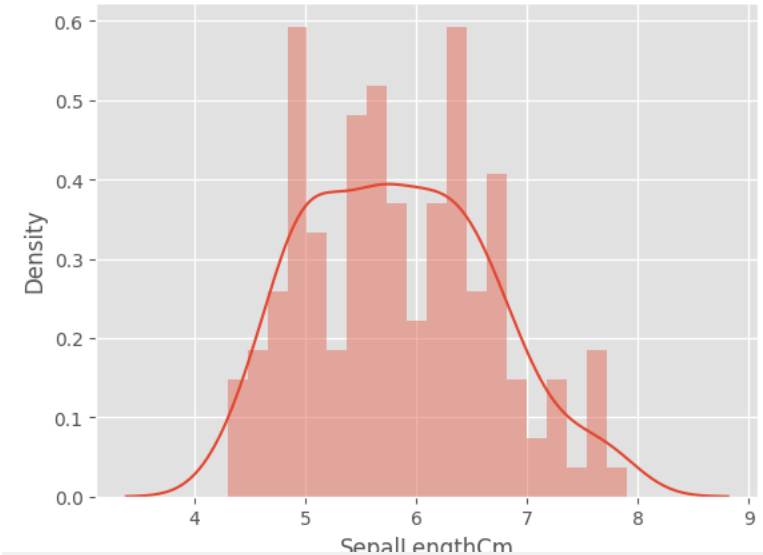
```
sns.boxplot(data, y='SepalLengthCm', color='yellow')
```

```
<Axes: ylabel='SepalLengthCm'>
```



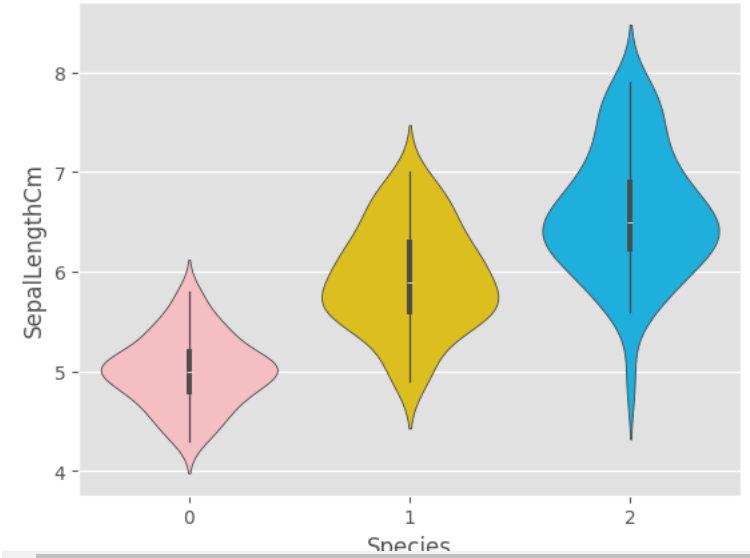

```
sns.distplot(data.SepalLengthCm,bins=20)
```

```
<Axes: xlabel='SepalLengthCm', ylabel='Density'>
```



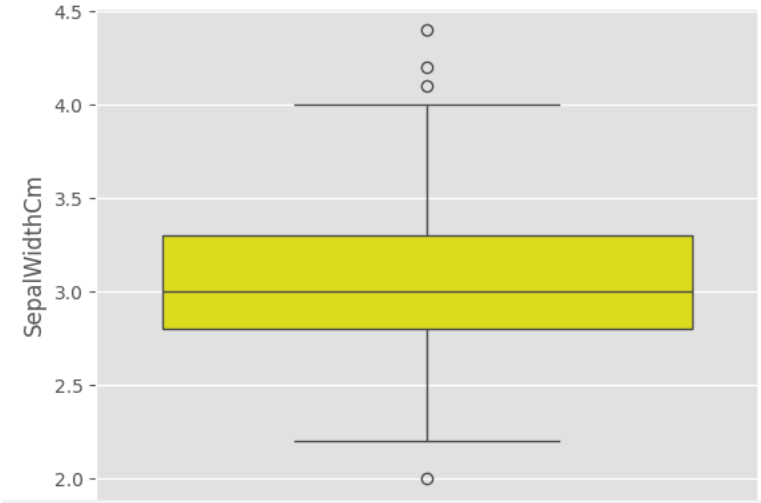

```
sns.violinplot(data,x='Species',y='SepalLengthCm', palette=['#FFB6C1', '#FFD700', '#00BFFF'])
```

```
<Axes: xlabel='Species', ylabel='SepalLengthCm'>
```



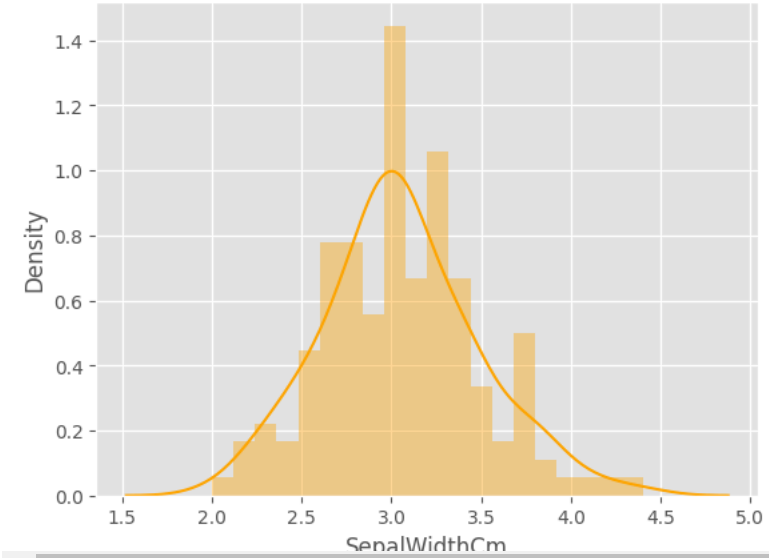

```
sns.boxplot(data, y='SepalWidthCm', color='yellow')
```

```
<Axes: ylabel='SepalWidthCm'>
```



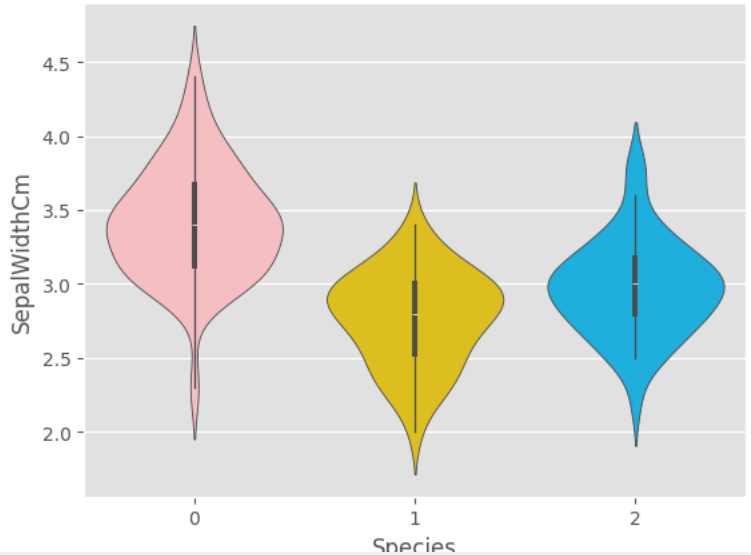

```
sns.distplot(data.SepalWidthCm,bins=20, color='Orange')
```

```
<Axes: xlabel='SepalWidthCm', ylabel='Density'>
```



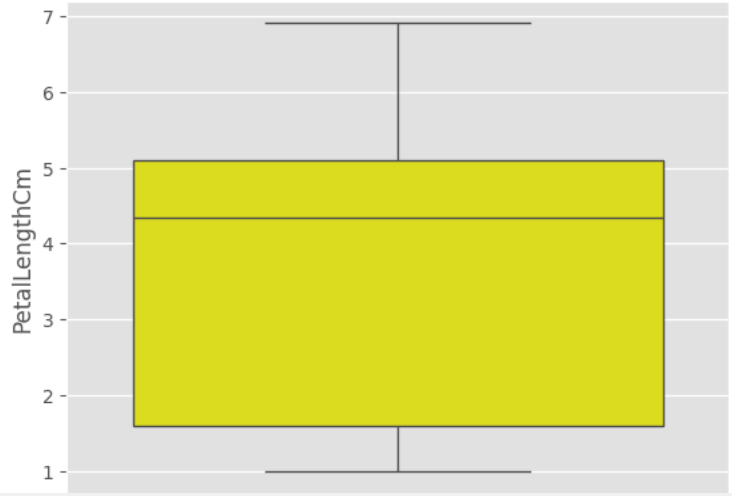

```
sns.violinplot(data,x='Species',y='SepalWidthCm', palette=['#FFB6C1', '#FFD700', '#00BFFF'])
```

```
<Axes: xlabel='Species', ylabel='SepalWidthCm'>
```



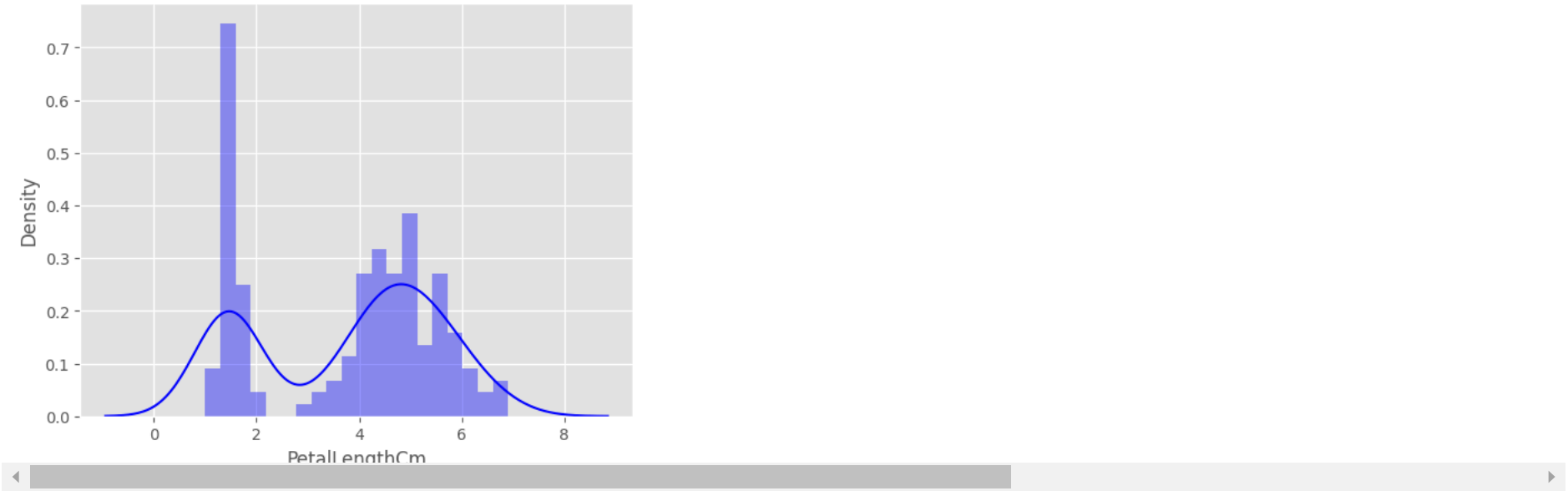

```
sns.boxplot(data, y='PetalLengthCm', color='yellow')
```

```
<Axes: ylabel='PetalLengthCm'>
```



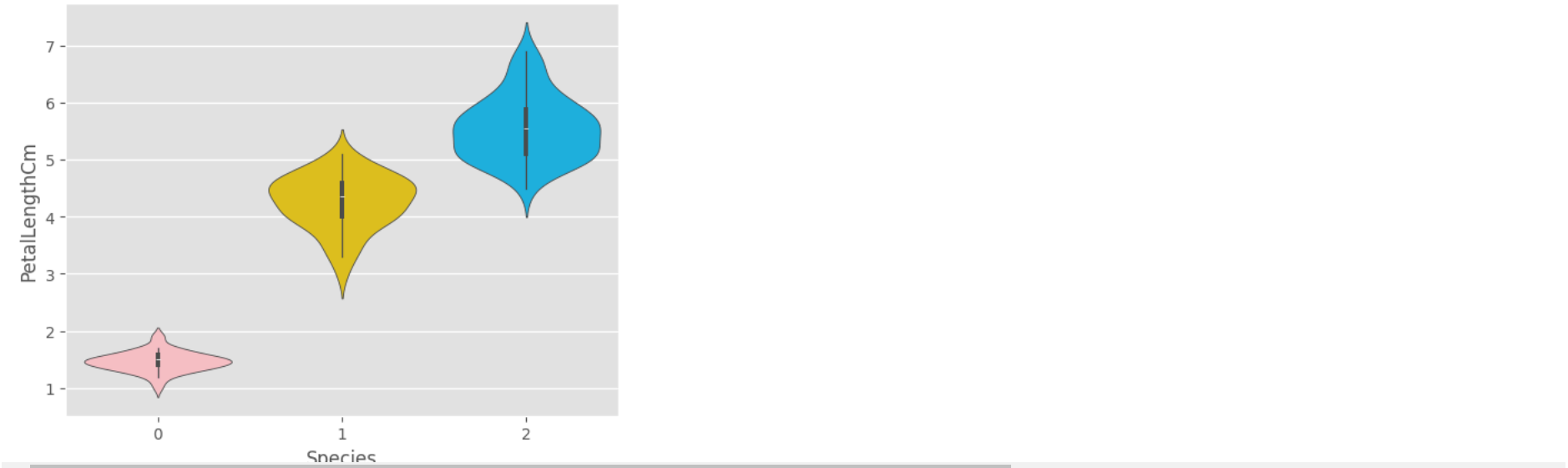

```
sns.distplot(data.PetalLengthCm,bins=20, color='blue')
```

```
<Axes: xlabel='PetalLengthCm', ylabel='Density'>
```




```
sns.violinplot(data,x='Species',y='PetalLengthCm', palette=['#FFB6C1', '#FFD700', '#00BFFF'])
```

```
<Axes: xlabel='Species', ylabel='PetalLengthCm'>
```

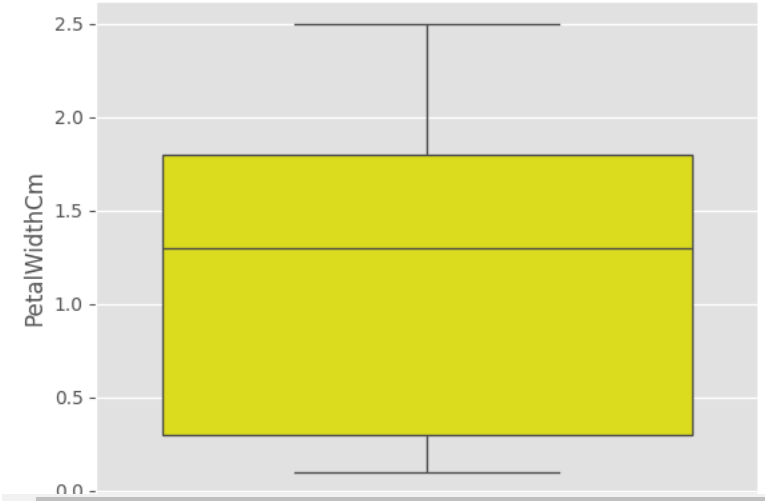



```
data.PetalWidthCm.describe()
```

	PetalWidthCm
count	150.000000
mean	1.198667
std	0.763161
min	0.100000
25%	0.300000
50%	1.300000
75%	1.800000
max	2.500000

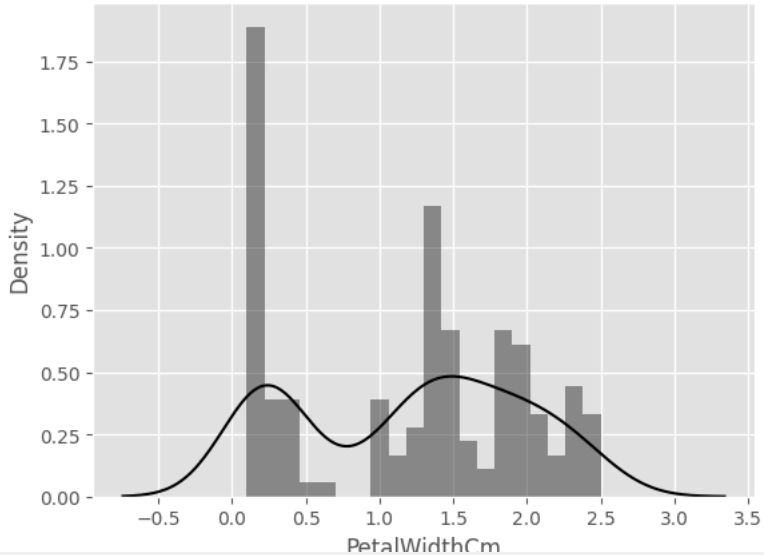
```
sns.boxplot(data, y='PetalWidthCm', color='yellow')
```

```
<Axes: ylabel='PetalWidthCm'>
```



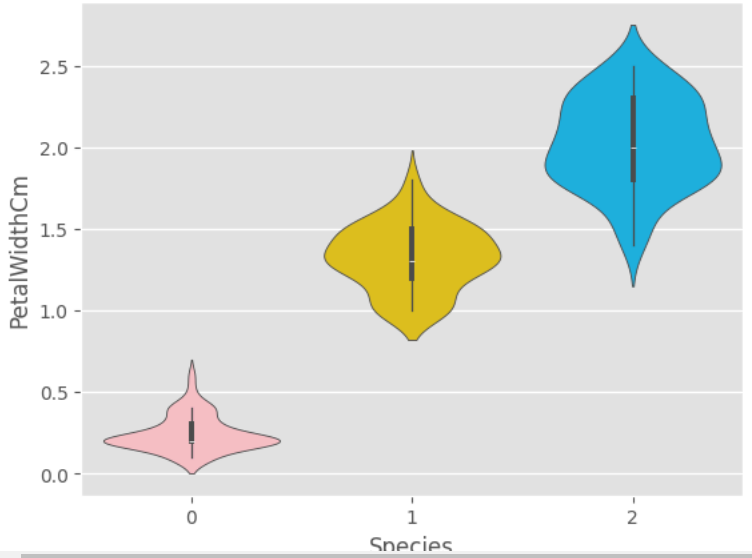

```
sns.distplot(data.PetalWidthCm,bins=20, color='black')
```

```
<Axes: xlabel='PetalWidthCm', ylabel='Density'>
```




```
sns.violinplot(data,x='Species',y='PetalWidthCm', palette=['#FFB6C1', '#FFD700', '#00BFFF'])
```

```
<Axes: xlabel='Species', ylabel='PetalWidthCm'>
```




```
data=data.sample(frac=1, random_state=48)
```

```
data.head(2)
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	0
1	2	4.9	3.0	1.4	0.2	0

Next steps: Generate code with data | View recommended plots | New interactive sheet

```
Y=data['Species']
X=data.drop(columns='Species')


X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size=0.2, random_state=13)


model =DecisionTreeClassifier()
model.fit(X_train, Y_train)
```

```
DecisionTreeClassifier
DecisionTreeClassifier()
```

```
models = []


models.append(('Logistic Regression',LogisticRegression(solver='lbfgs', max_iter=1000)))
models.append(('Naive Bayes',GaussianNB()))
models.append(('Decision Tree',DecisionTreeClassifier()))
models.append(('KNN',KNeighborsClassifier()))
models.append(('SVM',SVC()))


for name, model in models:
    result = cross_val_score(model,X,Y,cv=10,verbose=0) # Use X and Y instead of x and y

    print(f'{name}: {result.mean()}')
```

```
Logistic Regression: 1.0
Naive Bayes: 0.9933333333333334
Decision Tree: 0.9933333333333334
KNN: 1.0
SVM: 0.9933333333333334
```

```
y_pred=model.predict(X_test)


accuracy_score(Y_test, y_pred)
```

```
0.9666666666666667
```

```
print(classification_report(Y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        11
```