NAME : UMME AFSHAN DATA SCIENCE INTERN IN VIRTUAL INTERNSHIP PROGRAM AT LETSGROWMORE(LGMVIP) JAN 2022

TASK 2 BEGINNER LEVEL : IRIS FLOWER CLASSIFICATION ML PROJECT

In [ ]:
```
IMPORTING LIBRARIES AND LOADING THE DATA SET
```

In [1]:
```
# IMPORTING LIBRARIES
import numpy as np
import pandas as pd
import seaborn as sns
sns.set_palette('husl')
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
```

LOADING IRIS DATASET

In [2]:
```
url = 'https://raw.githubusercontent.com/jbrownlee/Datasets/master/iris.csv'
```

In [4]:
```
col_name = ['sepal-lenght','sepal-width','petal-lenght','petal-width','class']
```

In [5]:
```
dataset = pd.read_csv(url, names = col_name)
```

SUMMARIZE THE DATA

In [6]:
```
dataset.shape
```

Out[6]:
```
(150, 5)
```

In [7]:
```
dataset.head()
```

Out[7]:

	sepal-lenght	sepal-width	petal-lenght	petal-width	class
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

In [8]:

```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   sepal-lenght   150 non-null    float64
 1   sepal-width    150 non-null    float64
 2   petal-lenght   150 non-null    float64
 3   petal-width    150 non-null    float64
 4   class          150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

In [9]:

```
dataset.describe()
```

Out[9]:

	sepal-lenght	sepal-width	petal-lenght	petal-width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

In [10]:

```
dataset['class'].value_counts()
```

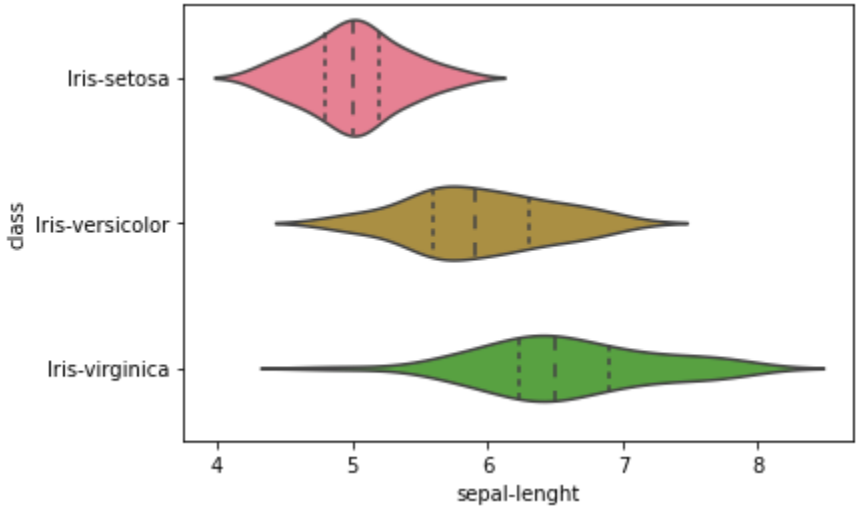
Out[10]:

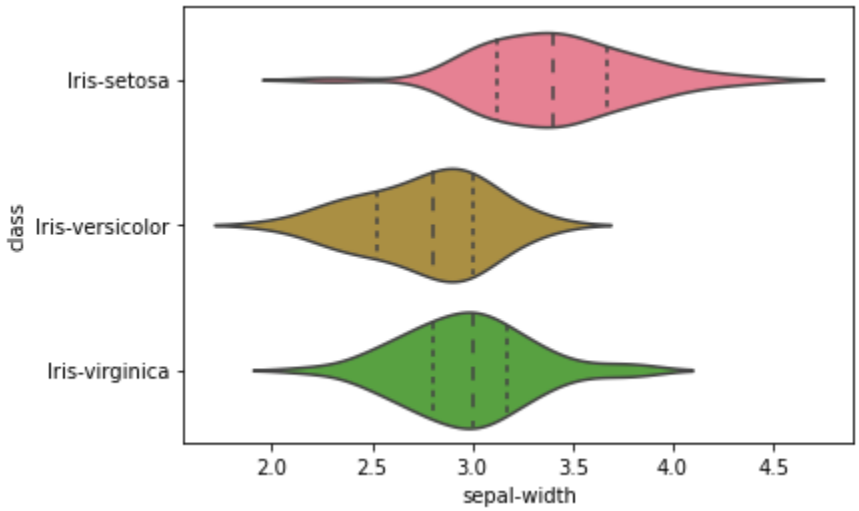
```
Iris-versicolor    50
Iris-setosa        50
Iris-virginica     50
Name: class, dtype: int64
```

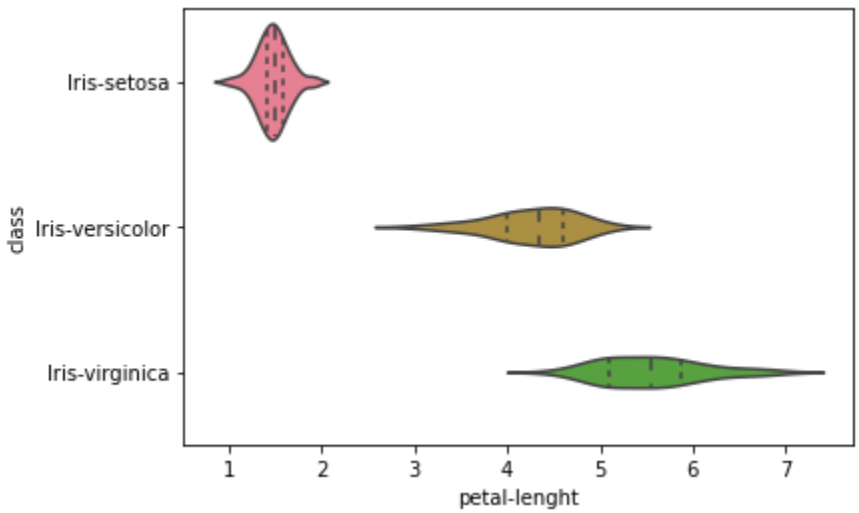
DATA VISUALIZATION

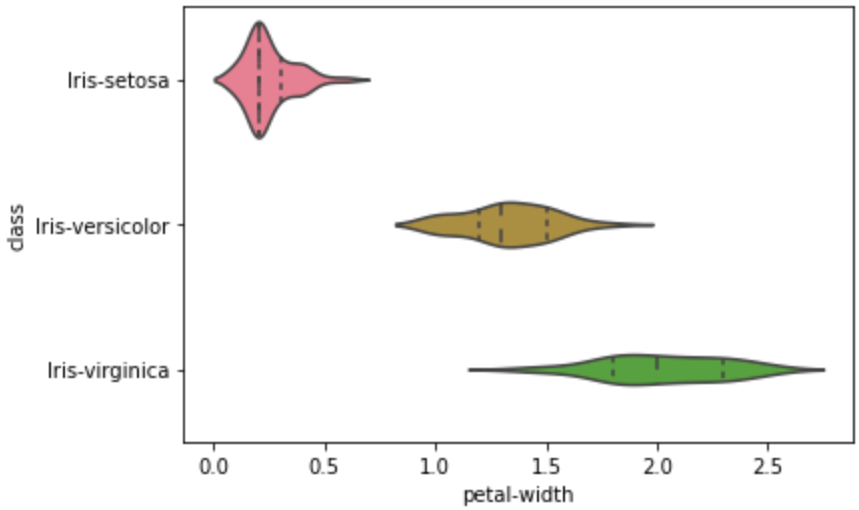
In [11]:

```
sns.violinplot(y='class', x='sepal-lenght', data=dataset, inner='quartile')
plt.show()
sns.violinplot(y='class', x='sepal-width', data=dataset, inner='quartile')
plt.show()
sns.violinplot(y='class', x='petal-lenght', data=dataset, inner='quartile')
plt.show()
sns.violinplot(y='class', x='petal-width', data=dataset, inner='quartile')
plt.show()
```


Iris-setosa
Iris-versicolor
Iris-virginica
class
4
5
6
7
8
sepal-lenght


Iris-setosa
Iris-versicolor
Iris-virginica
class
2.0
2.5
3.0
3.5
4.0
4.5
sepal-width

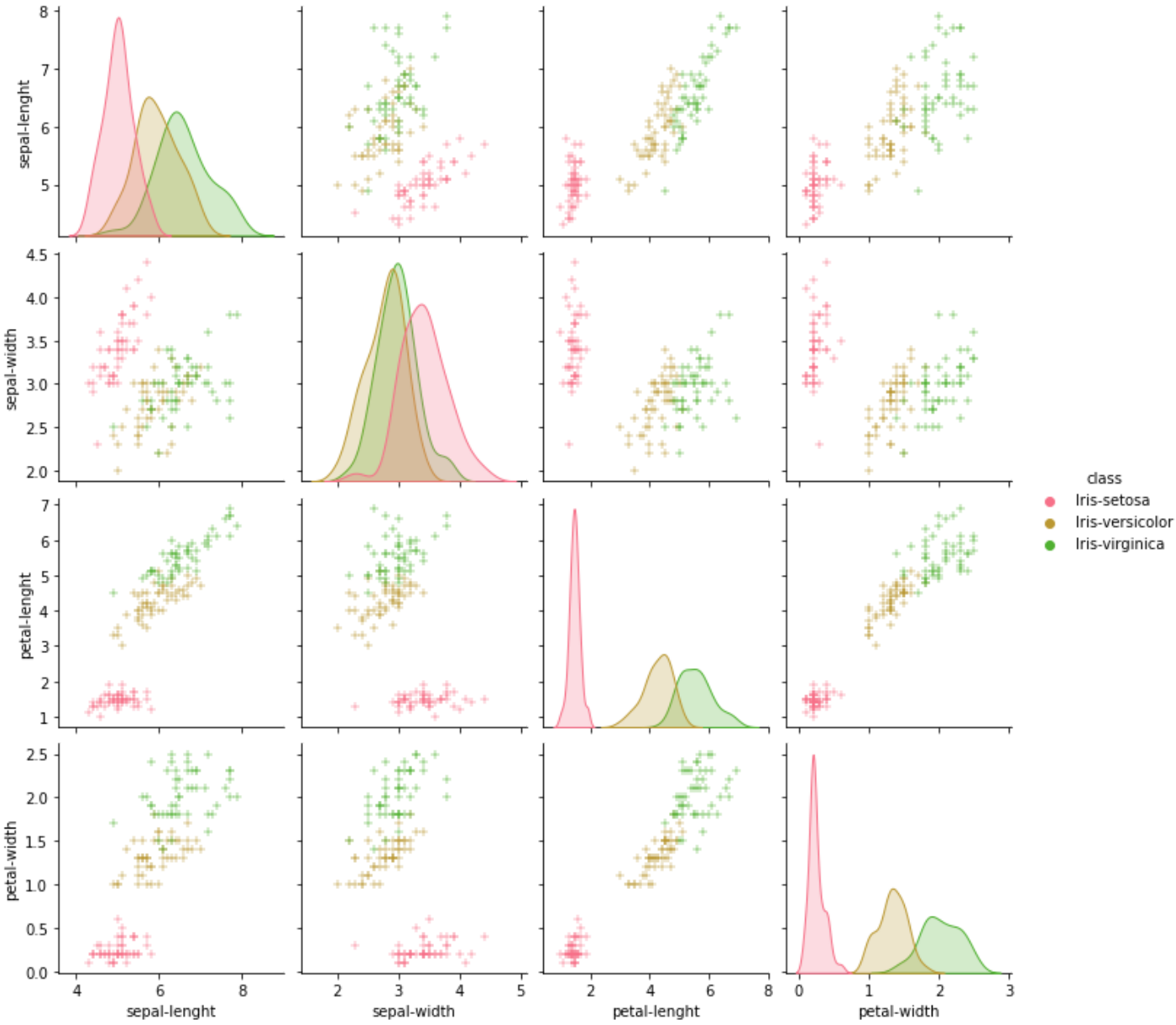

Iris-setosa
Iris-versicolor
Iris-virginica
class
1
2
3
4
5
6
7
petal-lenght



PAIR PLOT

In [13]:

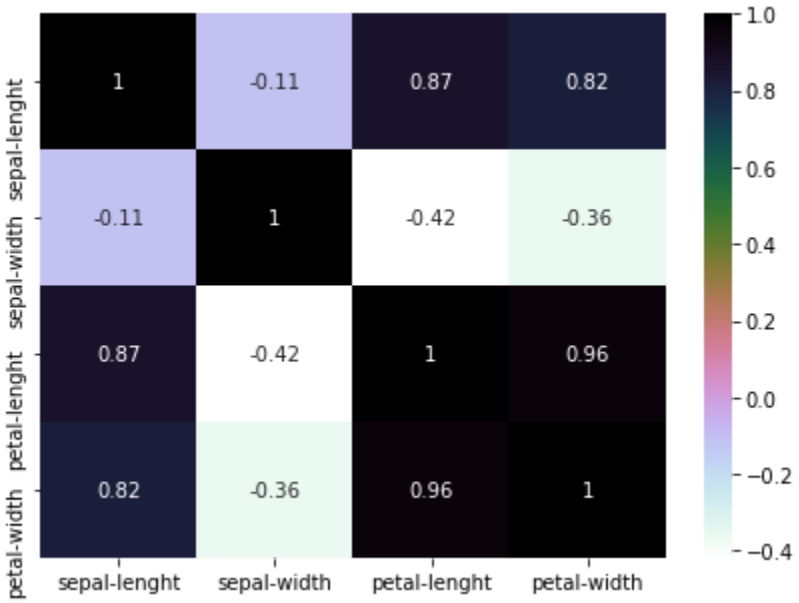
```
sns.pairplot(dataset, hue='class', markers='+')
plt.show()
```



HEAT MAP

In [14]:

```
plt.figure(figsize=(7,5))
sns.heatmap(dataset.corr(), annot=True, cmap='cubehelix_r')
plt.show()
```



SPLITTING THE DATASET

In [15]:
```
X = dataset.drop(['class'], axis=1)
y = dataset['class']
print(f'X shape: {X.shape} | y shape: {y.shape} ')
```

```
X shape: (150, 4) | y shape: (150,)
```

In [17]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=1)
```

In [18]:
```
models = []
models.append(('LR', LogisticRegression()))
models.append(('LDA', LinearDiscriminantAnalysis()))
models.append(('KNN', KNeighborsClassifier()))
models.append(('CART', DecisionTreeClassifier()))
models.append(('NB', GaussianNB()))
models.append(('SVC', SVC(gamma='auto')))
# evaluate each model in turn
results = []
model_names = []
for name, model in models:
    kfold = StratifiedKFold(n_splits=10, random_state=1, shuffle=True)
cv_results = cross_val_score(model, X_train, y_train, cv=kfold, scoring='accuracy')
results.append(cv_results)
model_names.append(name)
print('%s: %f (%f)' % (name, cv_results.mean(), cv_results.std()))
```

```
SVC: 0.983333 (0.033333)
```

MODEL CREATION

In [19]:
```
model = SVC(gamma='auto')
model.fit(X_train, y_train)
prediction = model.predict(X_test)
```

In [20]:
```
print(f'Test Accuracy: {accuracy_score(y_test, prediction)}')
```

```
print(f'Classification Report: \n {classification_report(y_test, prediction)}')
```

```
Test Accuracy: 0.9666666666666667
Classification Report:
                  precision    recall  f1-score   support

     Iris-setosa       1.00      1.00      1.00        11
 Iris-versicolor       1.00      0.92      0.96        13
  Iris-virginica       0.86      1.00      0.92         6

        accuracy                           0.97        30
       macro avg       0.95      0.97      0.96        30
    weighted avg       0.97      0.97      0.97        30
```

THANK YOU!