LetsGrowMore

Begineer Level Task : Data Science

Name : Akshata Gawali

Begineer Level Task 01 : Iris Flowers Classification ML Project

Task Description :

This particular ML project is usually referred to as the “Hello World” of Machine Learning. The iris flowers dataset contains numeric attributes, and it is perfect for beginners to learn about supervised ML algorithms, mainly how to load and handle data. Also, since this is a small dataset, it can easily fit in memory without requiring special transformations or scaling capabilities.

DataSetLink : [http://archive.ics.uci.edu/ml/datasets/Iris (http://archive.ics.uci.edu/ml/datasets/Iris)](http://archive.ics.uci.edu/ml/datasets/Iris)

Importing Libraries

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
import statsmodels.api as sm
from sklearn.linear_model import LogisticRegression
```

Importing Iris Dataset

In [2]:
```python
iris = pd.read_csv('iris_flowers.csv')
iris.head()  #Show top 5 values
```

Out[2]:

	sepal_length	sepal_width	petal_length	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 [3]:
```python
iris.tail() #show last 5 values
```

Out[3]:

	sepal_length	sepal_width	petal_length	petal_width	class
145	6.7	3.0	5.2	2.3	iris_virginica
146	6.3	2.5	5.0	1.9	iris_virginica
147	6.5	3.0	5.2	2.0	iris_virginica
148	6.2	3.4	5.4	2.3	iris_virginica
149	5.9	3.0	5.1	1.8	iris_virginica

In [4]:
```python
print(iris)
```

```
     sepal_length  sepal_width  petal_length  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
..            ...          ...           ...          ...             ...
145           6.7          3.0           5.2          2.3  iris_virginica
146           6.3          2.5           5.0          1.9  iris_virginica
147           6.5          3.0           5.2          2.0  iris_virginica
148           6.2          3.4           5.4          2.3  iris_virginica
149           5.9          3.0           5.1          1.8  iris_virginica

[150 rows x 5 columns]
```

In [5]:
```python
print(iris.shape)  #no. of rows and columns
```

```
(150, 5)
```

```
In [6]: iris.describe()  #used to view stastical details
```

Out[6]:

	sepal_length	sepal_width	petal_length	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 [7]: iris.nunique()   # returns unique elements
```

```
Out[7]: sepal_length    35
        sepal_width     23
        petal_length    43
         petal_width    22
        class            3
        dtype: int64
```

Checking Null Values

```
In [9]: iris.isnull()  #Returns dataframe object where all value are replaced with boolea
```

Out[9]:

	sepal_length	sepal_width	petal_length	petal_width	class
0	False	False	False	False	False
1	False	False	False	False	False
2	False	False	False	False	False
3	False	False	False	False	False
4	False	False	False	False	False
...	...	...	...	...	...
145	False	False	False	False	False
146	False	False	False	False	False
147	False	False	False	False	False
148	False	False	False	False	False
149	False	False	False	False	False

150 rows × 5 columns

In [10]:
```
iris.head(10).sum()     #returns no. of missing values
```

Out[10]:
```
sepal_length                                                 48.6
sepal_width                                                  33.1
petal_length                                                 14.5
 petal_width                                                  2.2
class           iris_setosairis_setosairis_setosairis_setosair...
dtype: object
```

Checking For Duplicate Values

In [11]:
```
iris[iris.duplicated()]
```

Out[11]:

	sepal_length	sepal_width	petal_length	petal_width	class
34	4.9	3.1	1.5	0.1	iris_setosa
37	4.9	3.1	1.5	0.1	iris_setosa
142	5.8	2.7	5.1	1.9	iris_virginica

Renaming the Name of Column

In [12]:
```
iris.rename(columns={'sepal_length': 'sepal length', 'sepal_width': 'sepal width'
```

Out[12]:

	sepal length	sepal width	petal length	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
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	iris_virginica
146	6.3	2.5	5.0	1.9	iris_virginica
147	6.5	3.0	5.2	2.0	iris_virginica
148	6.2	3.4	5.4	2.3	iris_virginica
149	5.9	3.0	5.1	1.8	iris_virginica

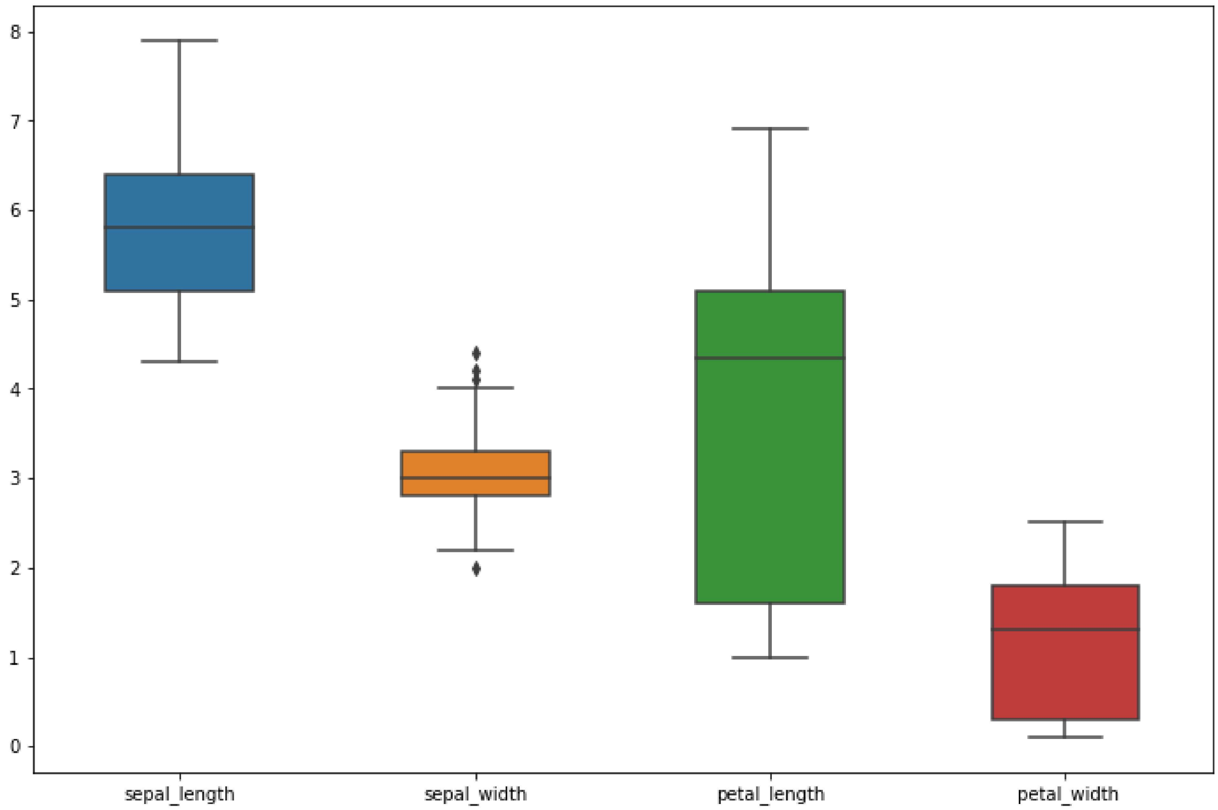
150 rows × 5 columns

Data Visualization

Box Plot

In [14]:
```
plt.figure(figsize=(12,8))
sns.boxplot(data = iris, width= 0.5, fliersize = 5)
```

Out[14]: <AxesSubplot:>



Exploring Co-relation Between between different columns

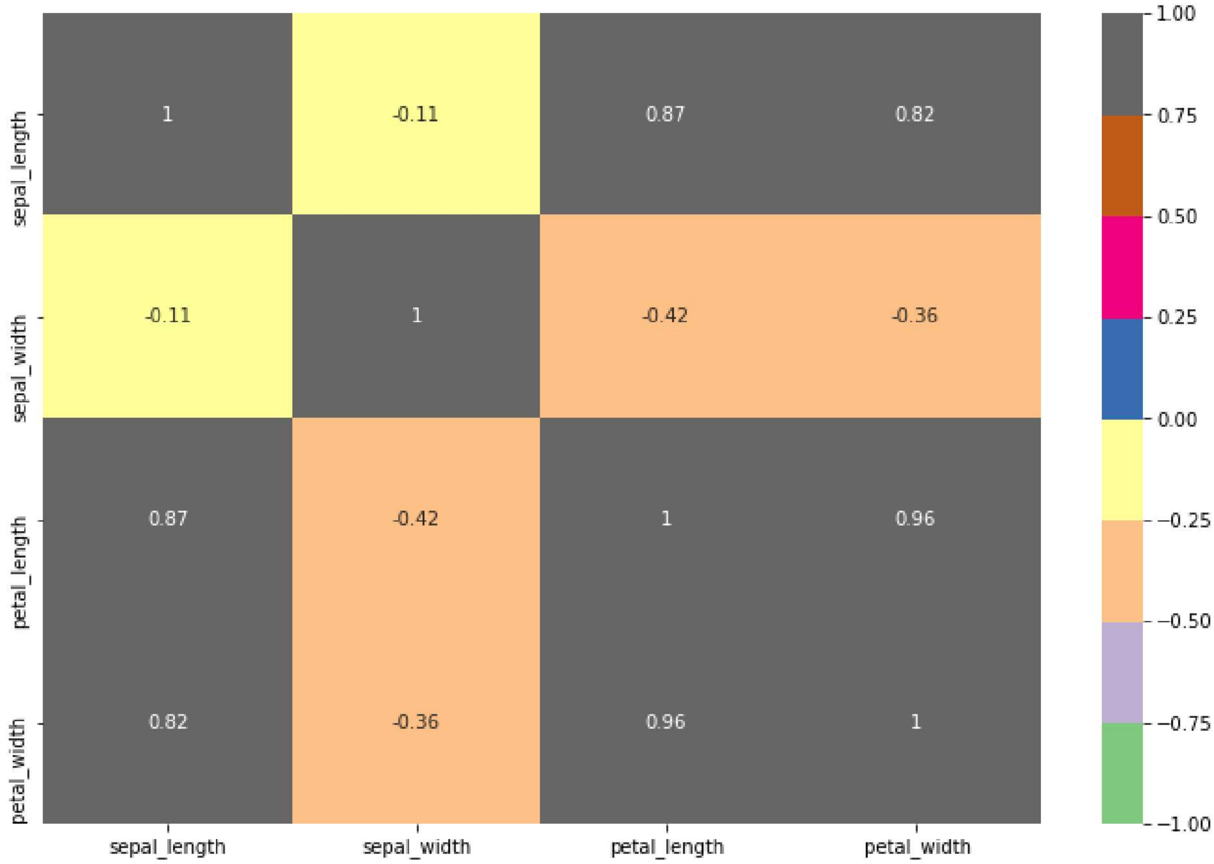
In [15]:
```
iris.corr(method='pearson')
```

Out[15]:

	sepal_length	sepal_width	petal_length	petal_width
sepal_length	1.000000	-0.109369	0.871754	0.817954
sepal_width	-0.109369	1.000000	-0.420516	-0.356544
petal_length	0.871754	-0.420516	1.000000	0.962757
petal_width	0.817954	-0.356544	0.962757	1.000000

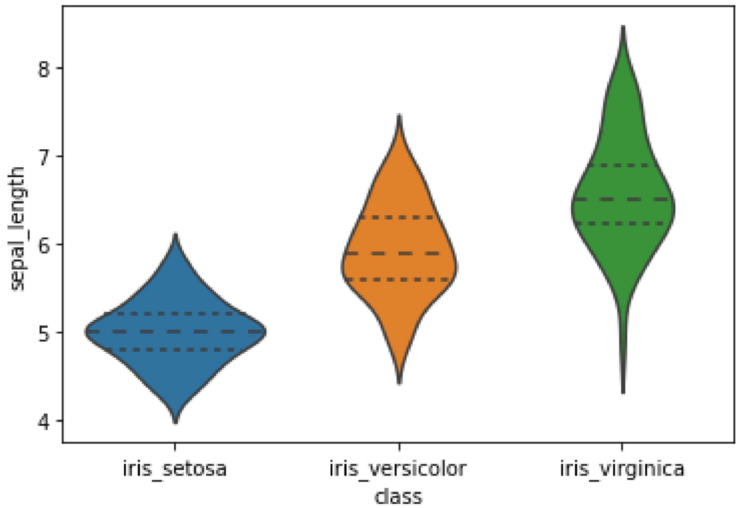
```
In [16]: plt.figure(figsize=(12,8))   #data preprocessing or corelation matrix
         sns.heatmap(iris.corr(),annot=True,cmap='Accent',vmin=-1,vmax=1)
```

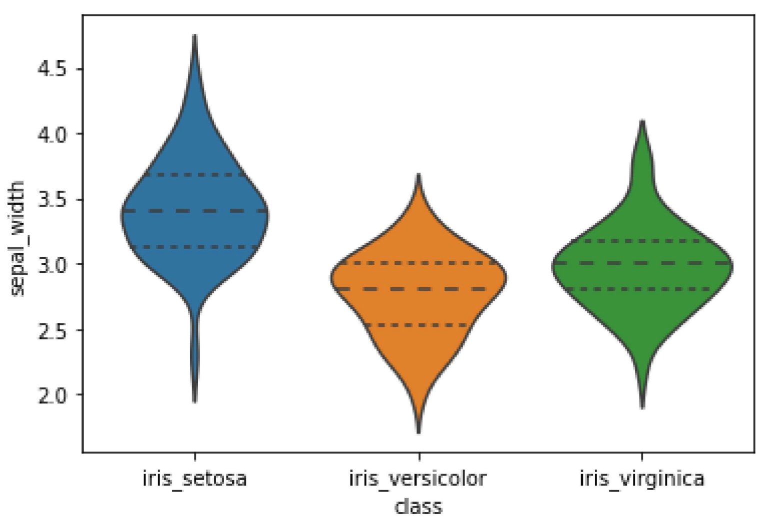
```
Out[16]: <AxesSubplot:>
```

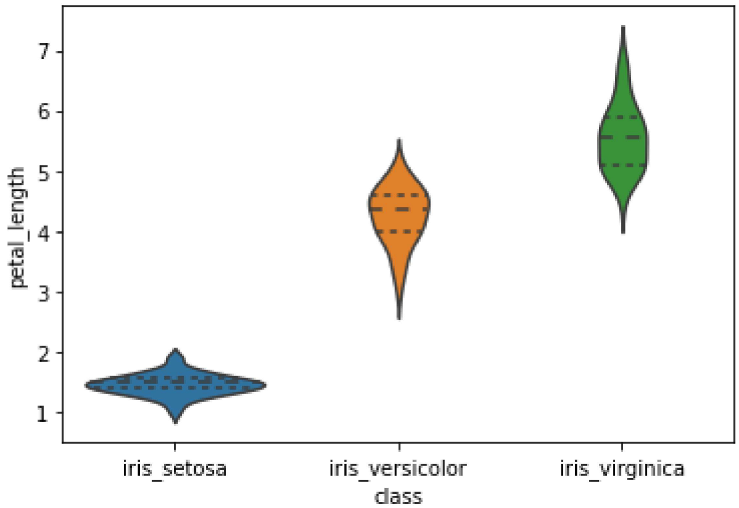


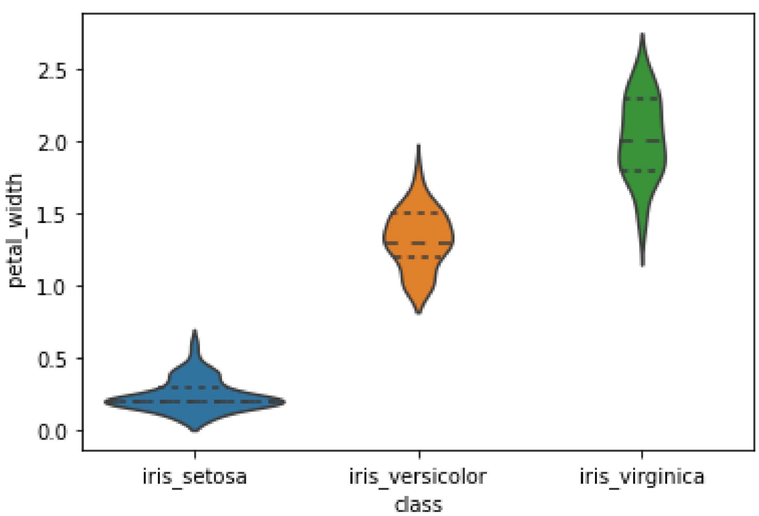
Violin Plot

In [17]:
```
for col in iris.columns[:4]:
    sns.violinplot(x='class',y=col,data=iris,inner='quartile')
    plt.show()
```



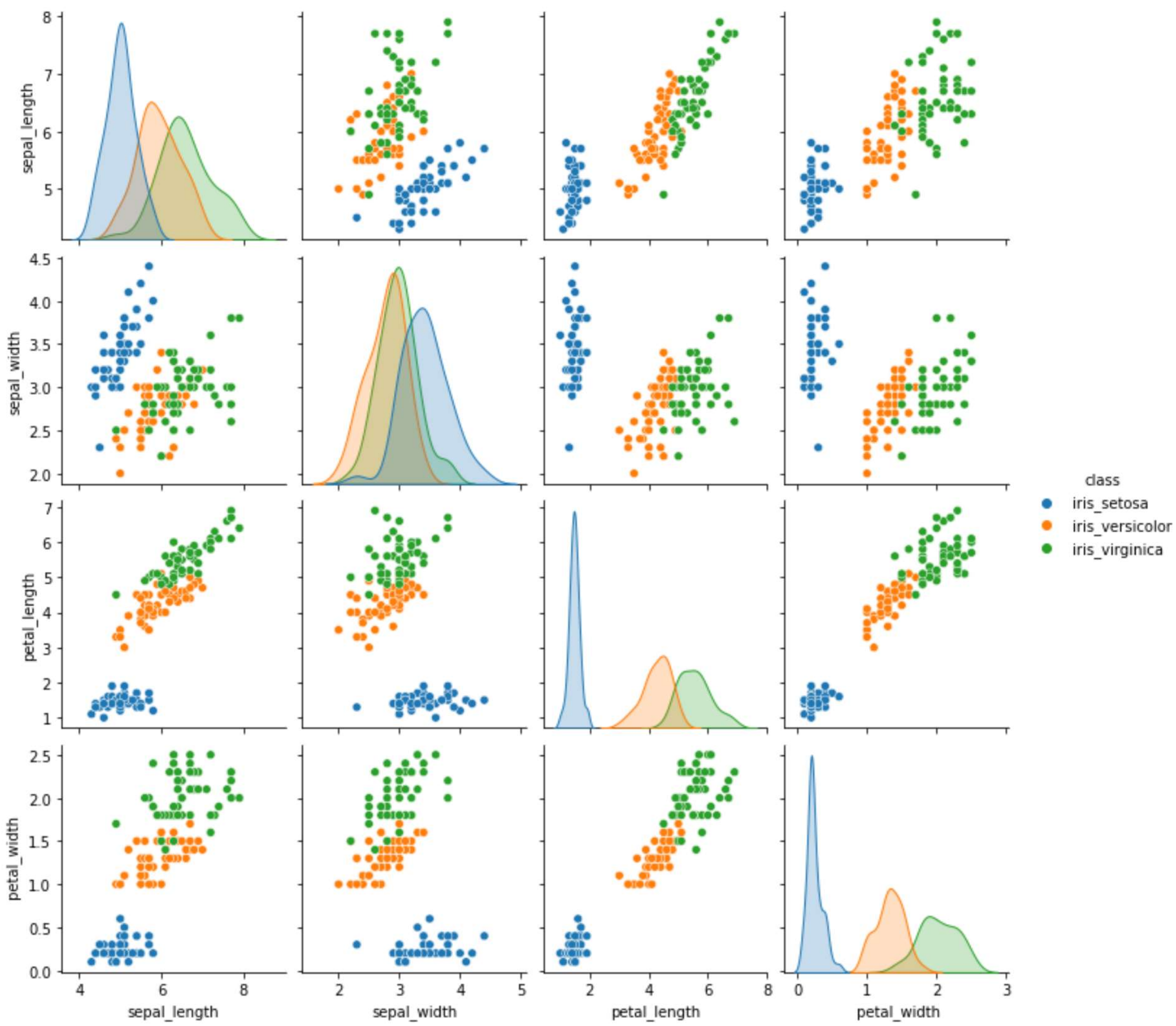



7
6
5
4
3
2
1
petal_length
iris_setosa
iris_versicolor
iris_virginica
class


2.5
2.0
1.5
1.0
0.5
0.0
petal_width
iris_setosa
iris_versicolor
iris_virginica
class

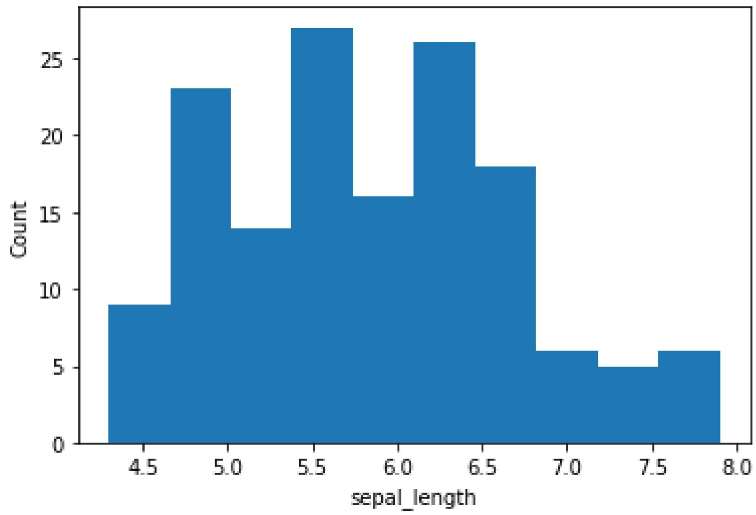
Pair Plot

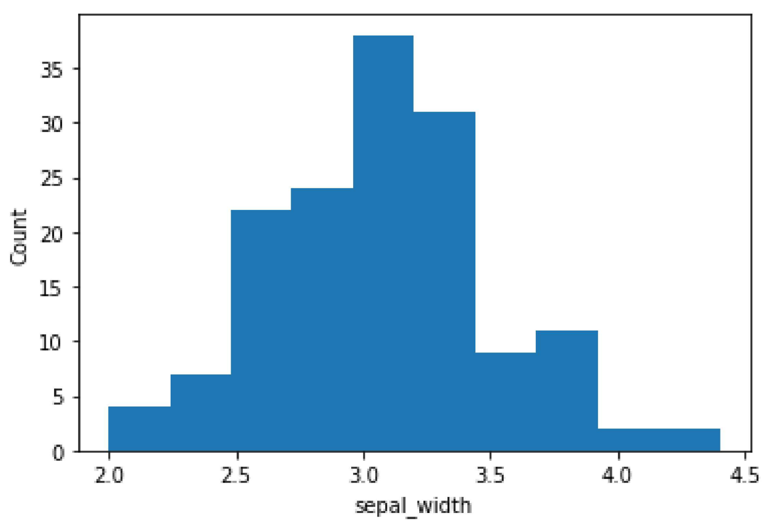
In [18]:
```
sns.pairplot(iris,hue='class');
```

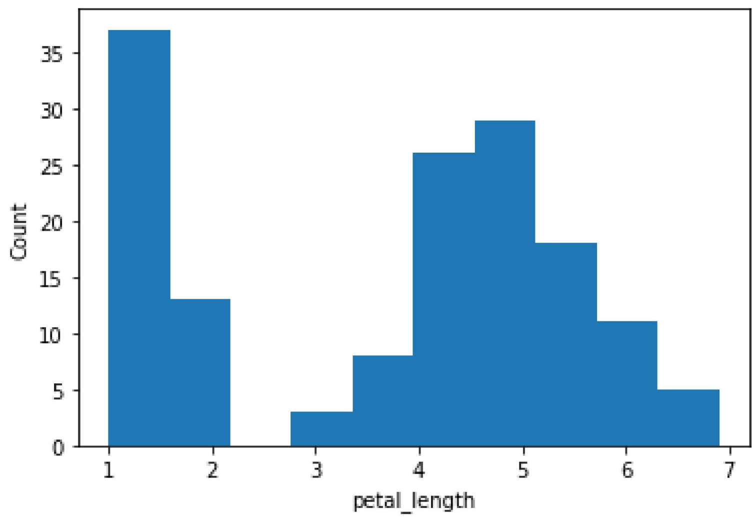


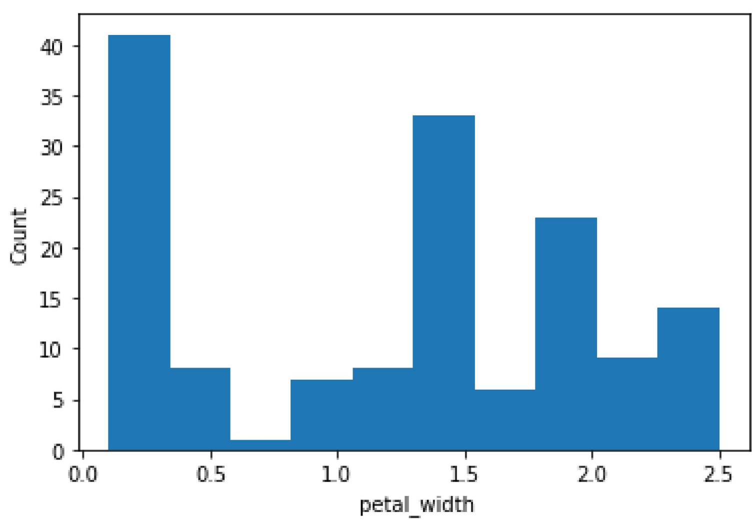
Histogram

In [19]:
```
for col in iris.columns[:4]:
    plt.hist(iris[col])
    plt.xlabel(col)
    plt.ylabel('Count')
    plt.show()
```



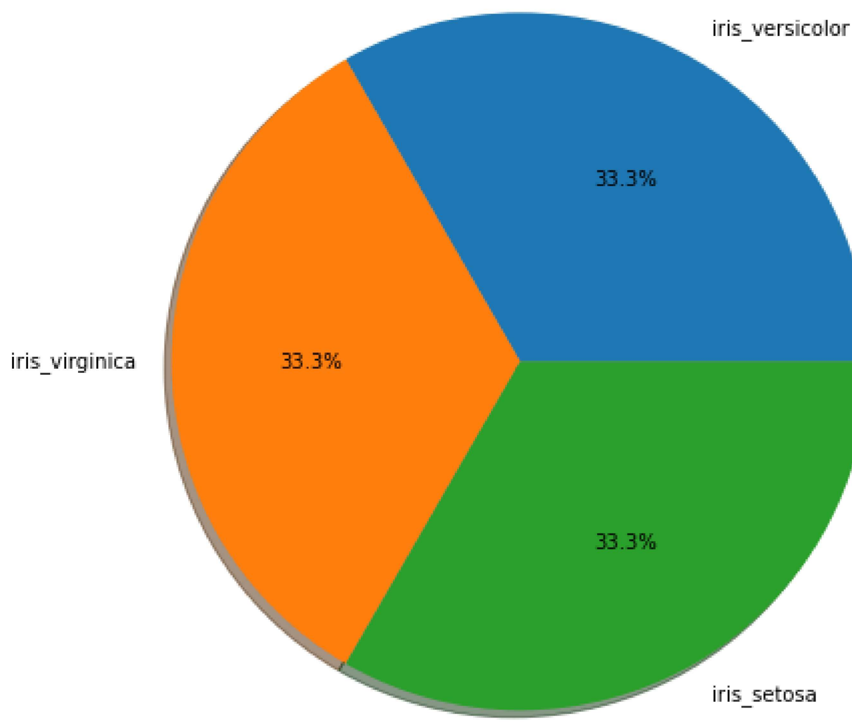






Plotting Pie Chart

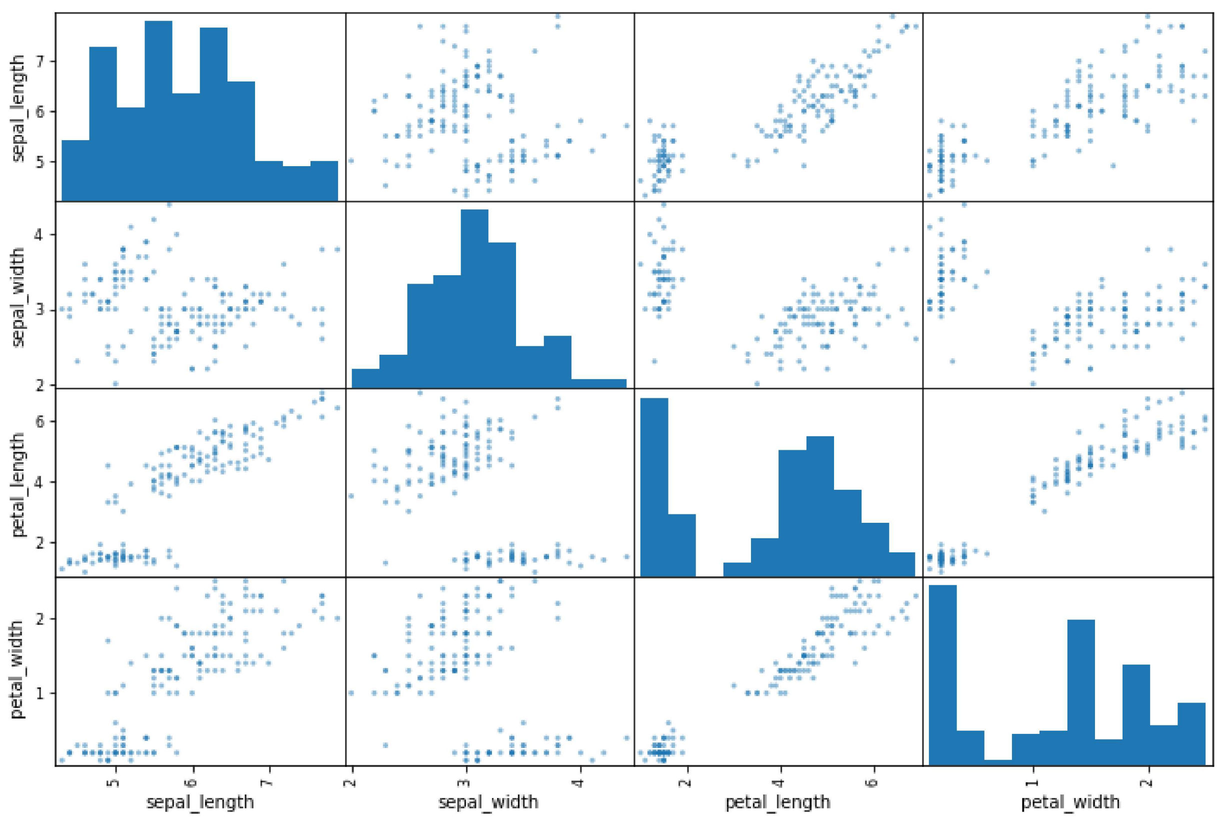
In [20]:
```
plt.figure(figsize=(8,8))
plt.pie(iris['class'].value_counts().values,labels=iris['class'].value_counts().k
plt.show()
```



Matrix Scatterplot

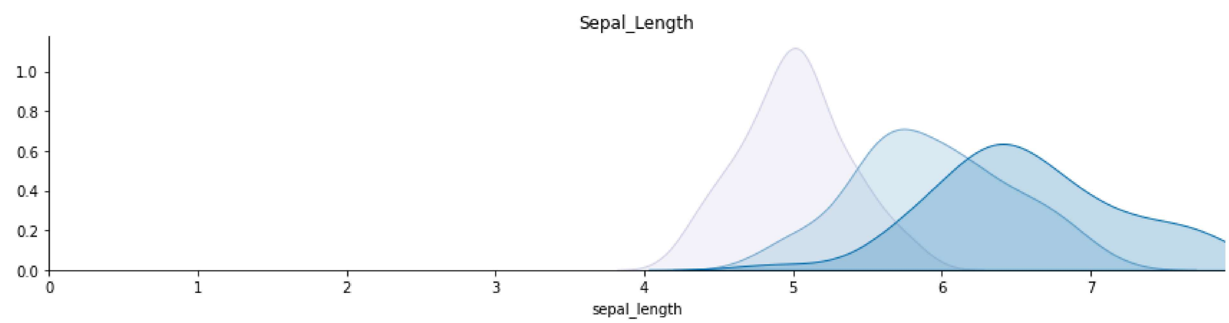
In [21]:
```
pd.plotting.scatter_matrix(iris,figsize=(12,8))
```

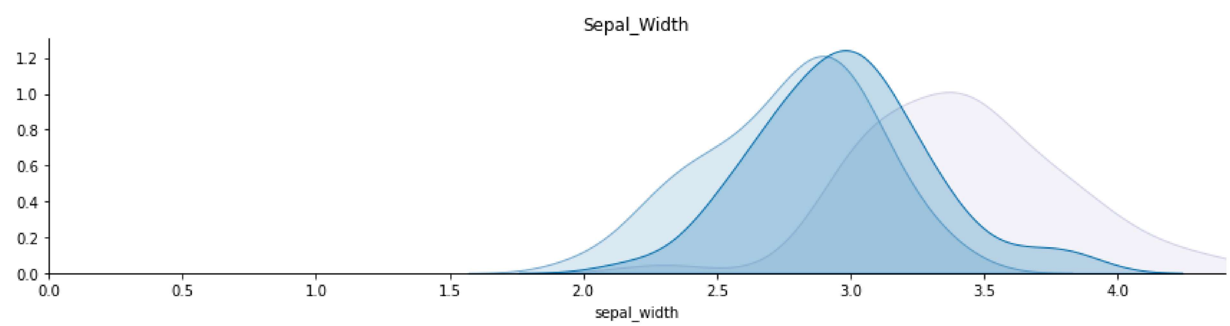
Out[21]:
```
array([[<AxesSubplot:xlabel='sepal_length', ylabel='sepal_length'>,
        <AxesSubplot:xlabel='sepal_width', ylabel='sepal_length'>,
        <AxesSubplot:xlabel='petal_length', ylabel='sepal_length'>,
        <AxesSubplot:xlabel=' petal_width', ylabel='sepal_length'>],
       [<AxesSubplot:xlabel='sepal_length', ylabel='sepal_width'>,
        <AxesSubplot:xlabel='sepal_width', ylabel='sepal_width'>,
        <AxesSubplot:xlabel='petal_length', ylabel='sepal_width'>,
        <AxesSubplot:xlabel=' petal_width', ylabel='sepal_width'>],
       [<AxesSubplot:xlabel='sepal_length', ylabel='petal_length'>,
        <AxesSubplot:xlabel='sepal_width', ylabel='petal_length'>,
        <AxesSubplot:xlabel='petal_length', ylabel='petal_length'>,
        <AxesSubplot:xlabel=' petal_width', ylabel='petal_length'>],
       [<AxesSubplot:xlabel='sepal_length', ylabel=' petal_width'>,
        <AxesSubplot:xlabel='sepal_width', ylabel=' petal_width'>,
        <AxesSubplot:xlabel='petal_length', ylabel=' petal_width'>,
        <AxesSubplot:xlabel=' petal_width', ylabel=' petal_width'>]],
      dtype=object)
```

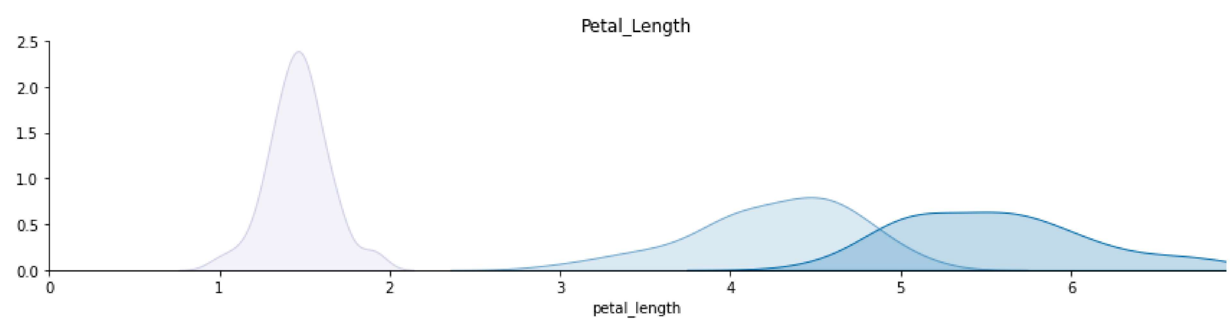


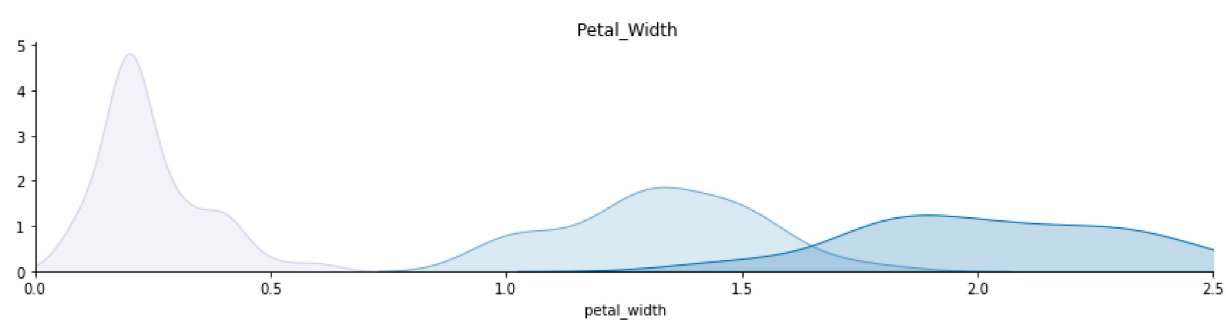
In [22]:
```
def plot_kde(a):
    facet=sns.FacetGrid(iris,hue='class',aspect=4,palette='PuBu')
    facet.map(sns.kdeplot,a,shade=True)
    facet.set(xlim=(0,iris[a].max()))
    plt.title(a.title())
    plt.show()
```

In [23]:
```
for col in iris.columns[:4]:
    plot_kde(col)
```









Training the Model

In [25]:
```
x = iris.drop(columns='class',axis=1)
y = iris['class']
x.head(), y.head()
```

Out[25]:
```
(   sepal_length  sepal_width  petal_length  petal_width
 0           5.1          3.5           1.4          0.2
 1           4.9          3.0           1.4          0.2
 2           4.7          3.2           1.3          0.2
 3           4.6          3.1           1.5          0.2
 4           5.0          3.6           1.4          0.2,
 0    iris_setosa
 1    iris_setosa
 2    iris_setosa
 3    iris_setosa
 4    iris_setosa
 Name: class, dtype: object)
```

split dataset into training and testing

In [26]:
```
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.4,random_stat
```

Selecting the models and metrics (Supervised ML Models)

In [27]:
```
lr = LogisticRegression()
knn = KNeighborsClassifier()
svm = SVC()
nb = GaussianNB()
dt = DecisionTreeClassifier()
rf = RandomForestClassifier()
```

Prediction and performance metrics

In [28]:
```
#training and evaluationg models
models = [lr,knn,svm,nb,dt,rf]
scores = []

for model in models:
    model.fit(x_train,y_train)
    y_pred = model.predict(x_test)
    scores.append(accuracy_score(y_test,y_pred))
    print("Accuracy of " + type(model).__name__ + " is", np.round(accuracy_score(
    print("Confusion Matrix of " + type(model).__name__ + " : ")
    print(confusion_matrix(y_test,y_pred))
    print("Classification Report of " + type(model).__name__ + " : ")
    print(classification_report(y_test,y_pred))
    print('----------')
```

```
Accuracy of LogisticRegression is 0.967
Confusion Matrix of LogisticRegression :
[[19  0  0]
 [ 0 20  1]
 [ 0  1 19]]
Classification Report of LogisticRegression :
                 precision    recall  f1-score   support

    iris_setosa       1.00      1.00      1.00        19
iris_versicolor       0.95      0.95      0.95        21
 iris_virginica       0.95      0.95      0.95        20

       accuracy                           0.97        60
      macro avg       0.97      0.97      0.97        60
   weighted avg       0.97      0.97      0.97        60

----------
Accuracy of KNeighborsClassifier is 0.983
Confusion Matrix of KNeighborsClassifier :
[[19  0  0]
 [ 0 21  0]
 [ 0  1 19]]
Classification Report of KNeighborsClassifier :
                 precision    recall  f1-score   support

    iris_setosa       1.00      1.00      1.00        19
iris_versicolor       0.95      1.00      0.98        21
 iris_virginica       1.00      0.95      0.97        20

       accuracy                           0.98        60
      macro avg       0.98      0.98      0.98        60
   weighted avg       0.98      0.98      0.98        60

----------
Accuracy of SVC is 0.983
Confusion Matrix of SVC :
[[19  0  0]
 [ 0 20  1]
 [ 0  0 20]]
Classification Report of SVC :
                 precision    recall  f1-score   support
```

```
    iris_setosa       1.00      1.00      1.00        19
iris_versicolor       1.00      0.95      0.98        21
 iris_virginica       0.95      1.00      0.98        20

       accuracy                           0.98        60
      macro avg       0.98      0.98      0.98        60
   weighted avg       0.98      0.98      0.98        60

----------
Accuracy of GaussianNB is 0.95
Confusion Matrix of GaussianNB :
[[19  0  0]
 [ 0 19  2]
 [ 0  1 19]]
Classification Report of GaussianNB :
                 precision    recall  f1-score   support

    iris_setosa       1.00      1.00      1.00        19
iris_versicolor       0.95      0.90      0.93        21
 iris_virginica       0.90      0.95      0.93        20

       accuracy                           0.95        60
      macro avg       0.95      0.95      0.95        60
   weighted avg       0.95      0.95      0.95        60

----------
Accuracy of DecisionTreeClassifier is 0.967
Confusion Matrix of DecisionTreeClassifier :
[[19  0  0]
 [ 0 20  1]
 [ 0  1 19]]
Classification Report of DecisionTreeClassifier :
                 precision    recall  f1-score   support

    iris_setosa       1.00      1.00      1.00        19
iris_versicolor       0.95      0.95      0.95        21
 iris_virginica       0.95      0.95      0.95        20

       accuracy                           0.97        60
      macro avg       0.97      0.97      0.97        60
   weighted avg       0.97      0.97      0.97        60

----------
Accuracy of RandomForestClassifier is 0.967
Confusion Matrix of RandomForestClassifier :
[[19  0  0]
 [ 0 20  1]
 [ 0  1 19]]
Classification Report of RandomForestClassifier :
                 precision    recall  f1-score   support

    iris_setosa       1.00      1.00      1.00        19
iris_versicolor       0.95      0.95      0.95        21
 iris_virginica       0.95      0.95      0.95        20

       accuracy                           0.97        60
```

```
   macro avg       0.97      0.97      0.97        60
weighted avg       0.97      0.97      0.97        60

----------
```

In [29]:
```python
results = pd.DataFrame({
    'Models': ['Logistic Regression', 'K-Nearest Neighbors', 'Support Vector Mach
               'Random Forest'],'Accuracy': scores})

results = results.sort_values(by='Accuracy', ascending=False)
print(results)
```

```
                    Models  Accuracy
1      K-Nearest Neighbors  0.983333
2   Support Vector Machine  0.983333
0      Logistic Regression  0.966667
4            Decision Tree  0.966667
5            Random Forest  0.966667
3              Naive Bayes  0.950000
```

Thus we learned how to load, handle & train the dataset using various supervised ML algorithms. also We learned K-Nearest Neighbors and Support Vector Machine models have predicted the result to a high level of accuracy, while Naive Bayes has predicted to the least level of accuracy.

Thank You!!!

In [ ]: