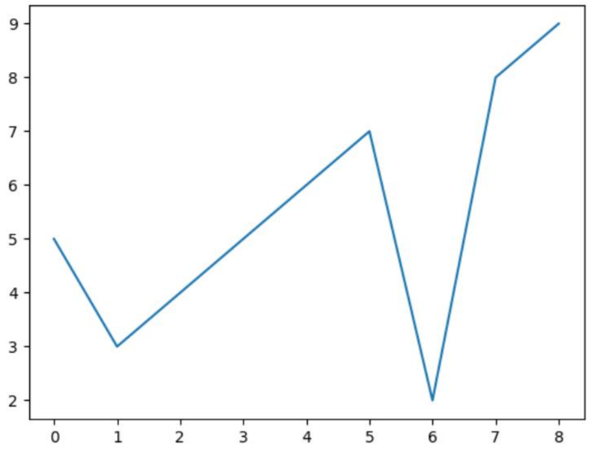
EXPERIMENT NO :4

AIM: Data Visualization

CODE

```
y = [5,3,4,5,6,7,2,8,9]
plt.plot (y)
plt.show()
```

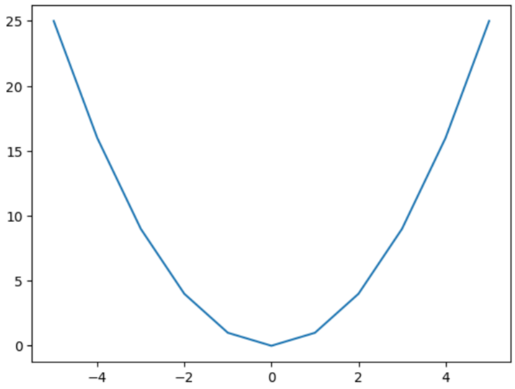
OUTPUT



CODE

```
x = [-5,-4,-3,-2,-1,0,1,2,3,4,5]
#y = [25,16,9,4,1,0,1,4,9,16,25]
y = [i**2 for i in x]
plt.plot(x,y)
plt.show()
```

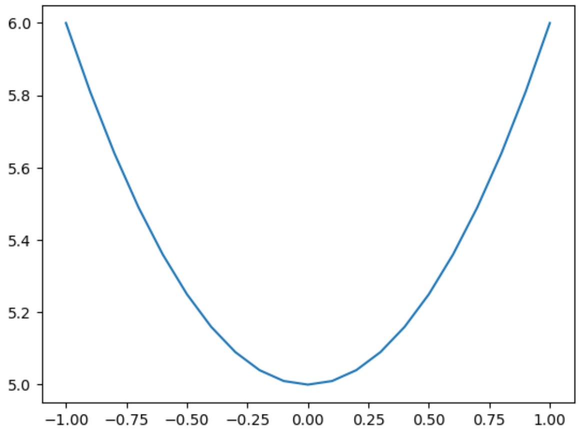
OUTPUT



CODE

```
import numpy as np
import math

x = np.arange(-1,1.1,0.1).tolist()
y = [i**2 + 5 for i in x]

plt.plot(x,y)
plt.show()
```

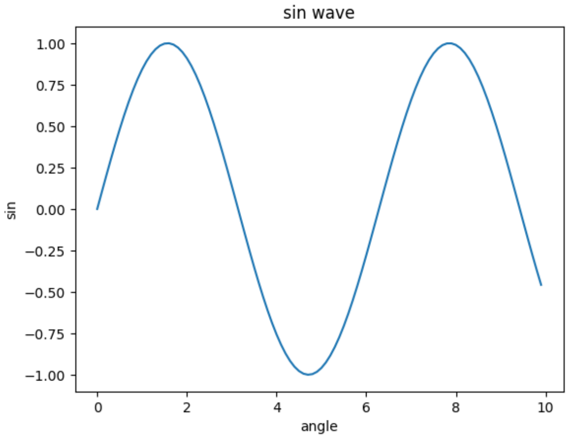
OUTPUT



CODE

```
import numpy as np
x = np.arange(0,10,0.1)
y = np.sin(x)
plt.plot(x,y)
plt.xlabel('angle')
plt.ylabel('sin')
plt.title('sin wave')
plt.show()
```

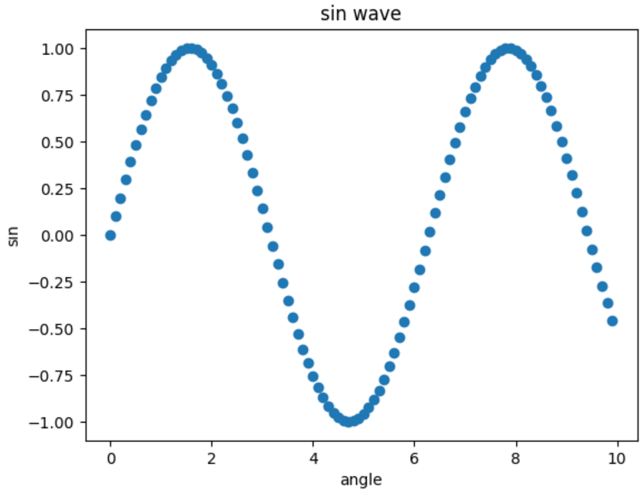
OUTPUT



CODE

```
plt.scatter(x,y)
plt.xlabel('angle')
plt.ylabel('sin')
plt.title('sin wave')
plt.show()
```

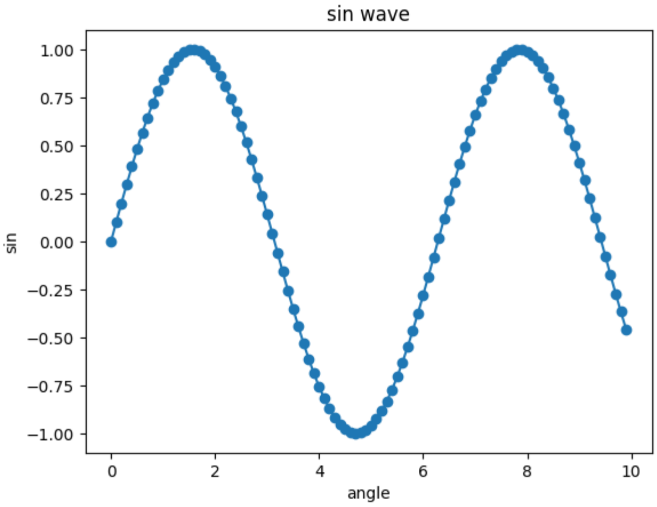
OUTPUT



CODE

```
plt.plot(x,y)
plt.scatter(x,y)
plt.xlabel('angle')
plt.ylabel('sin')
plt.title('sin wave')
plt.show()
```

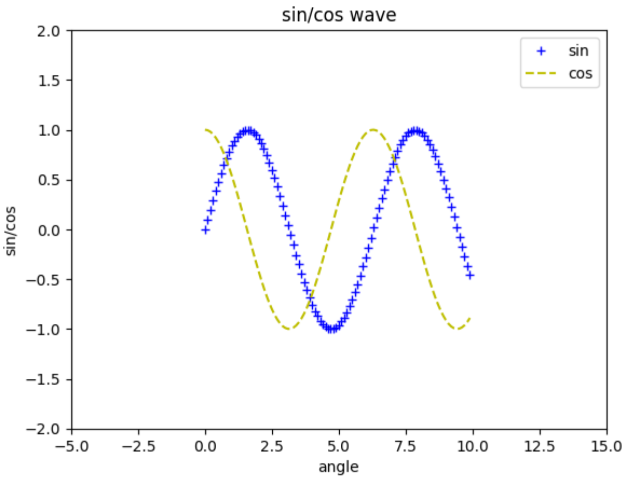
OUTPUT



CODE

```
plt.plot(x,np.sin(x), 'b+', label='sin')

plt.plot(x,np.cos(x) ,'y--', label='cos') #Scatter connections

plt.xlabel('angle')

plt.ylabel('sin/cos')

plt.title('sin/cos wave')

plt.ylim(-2,2)

plt.xlim(-5,15)

plt.legend()

plt.show()
```

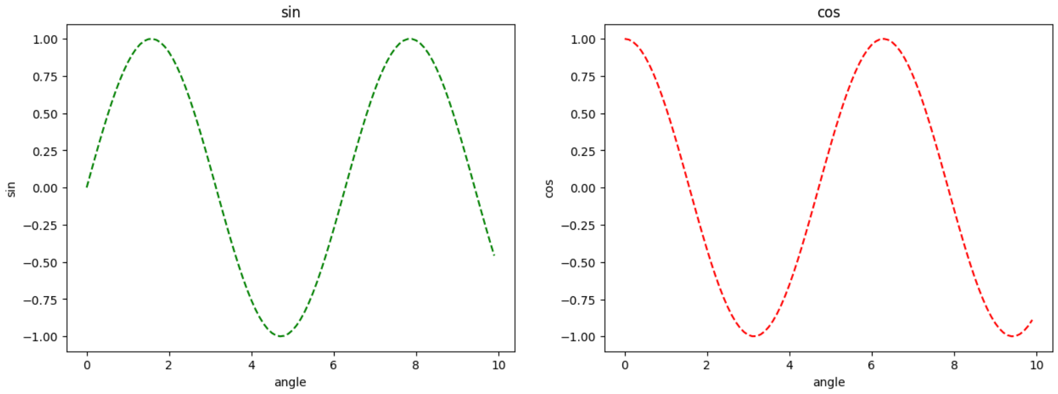
OUTPUT



CODE

```
fig, axis = plt.subplots(1,2, figsize=(15,5))
x = np.arange(0,10,0.1)
axis[0].plot(x,np.sin(x), 'g--')
axis[0].set_title('sin')
axis[0].set_xlabel('angle')
axis[0].set_ylabel('sin')
axis[1].plot(x,np.cos(x), 'r--')
axis[1].set_title('cos')
axis[1].set_xlabel('angle')
axis[1].set_ylabel('cos')
plt.show()
```

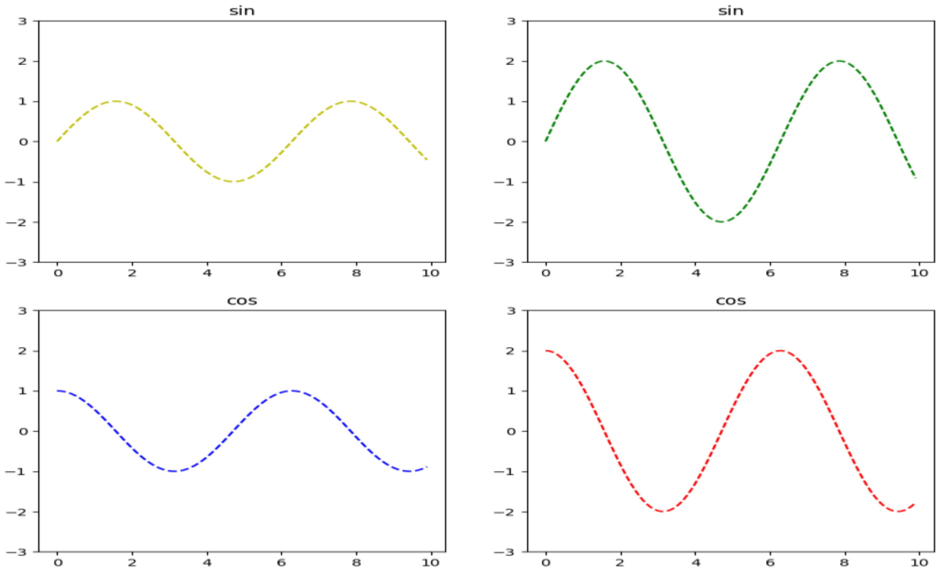
OUTPUT



CODE

```
fig, axis = plt.subplots(2,2, figsize=(10,10))
x = np.arange(0,10,0.1)
axis[0][0].plot(x,np.sin(x), 'y--')
axis[0][0].set_title('sin')
axis[0][0].set_ylim(-3,3)
axis[0][1].plot(x,2*np.sin(x), 'g--')
axis[0][1].set_title('sin')
axis[0][1].set_ylim(-3,3)
axis[1][0].plot(x,np.cos(x), 'b--')
axis[1][0].set_title('cos')
axis[1][0].set_ylim(-3,3)
axis[1][1].plot(x,2*np.cos(x), 'r--')
axis[1][1].set_title('cos')
axis[1][1].set_ylim(-3,3)
plt.show()
```

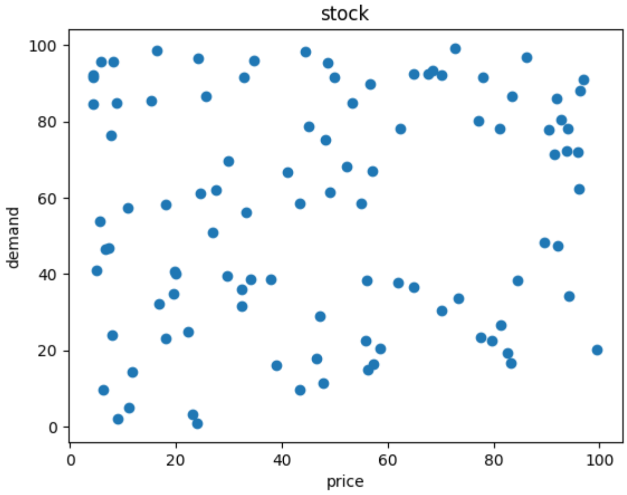
OUTPUT



CODE

```
x = np.random.random(100)*100
y = np.random.random(100)*100
plt.scatter(x,y)
plt.xlabel('price')
plt.ylabel('demand')
plt.title('stock')
plt.show()
```

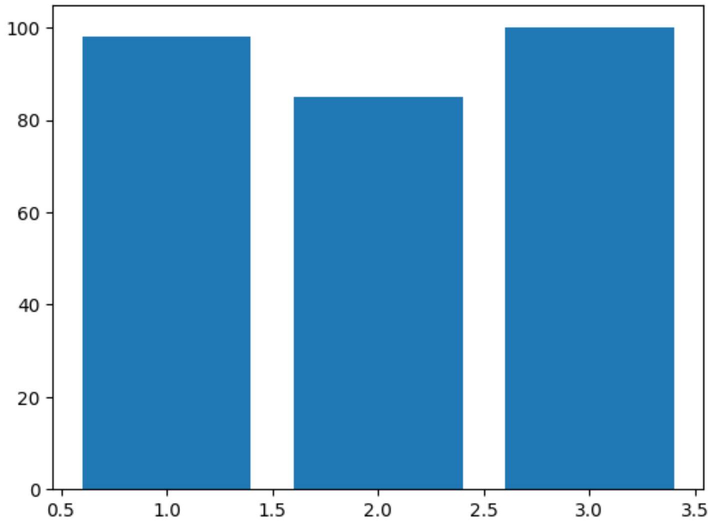
OUTPUT



CODE

```
x = np.array([1,2,3])
y = [98,85,100]
plt.bar(x,y)
plt.show()
```

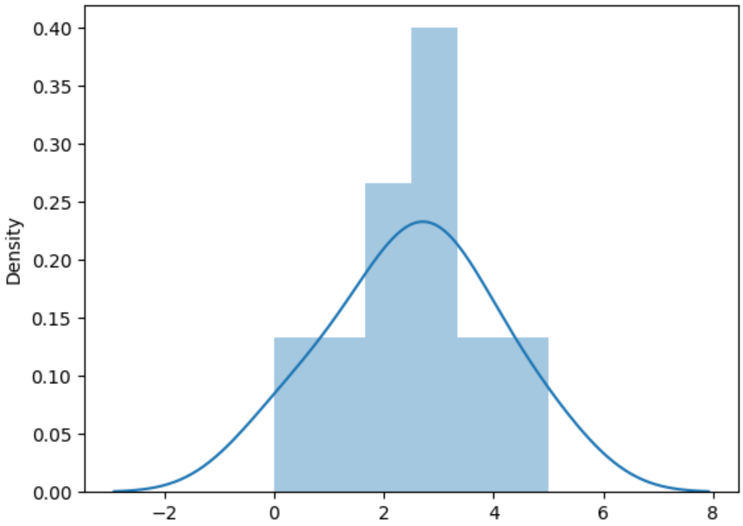
OUTPUT



CODE

```
import matplotlib.pyplot as plt
import seaborn as sns
sns.distplot([0, 1, 2, 3, 4, 5,3,2,3])
plt.show()
```

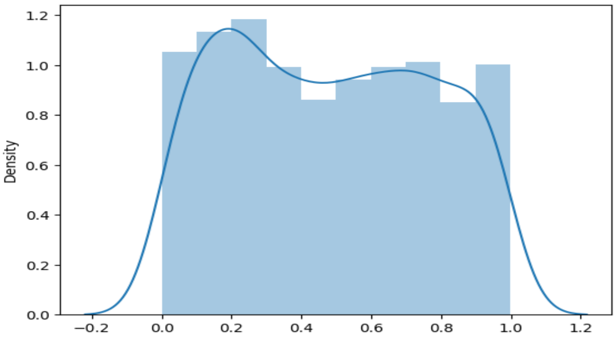
OUTPUT

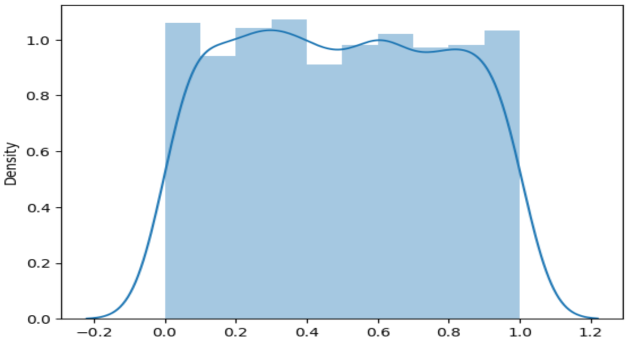


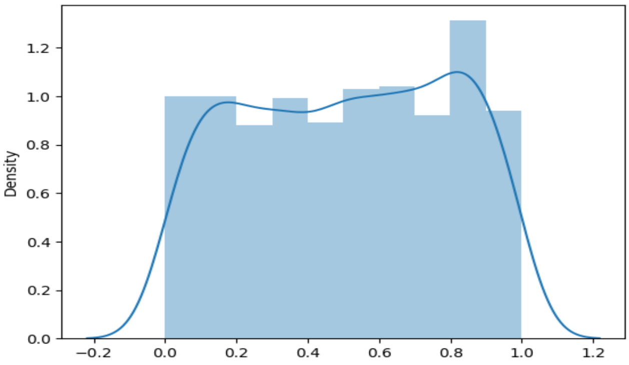
CODE

```
from numpy import random
import matplotlib.pyplot as plt
import seaborn as sns
x = np.random.random((1000,3))
sns.distplot(x[:,0], hist=True)
plt.show()
sns.distplot(x[:,1], hist=True)
plt.show()
sns.distplot(x[:,2], hist=True)
plt.show()
```

OUTPUT



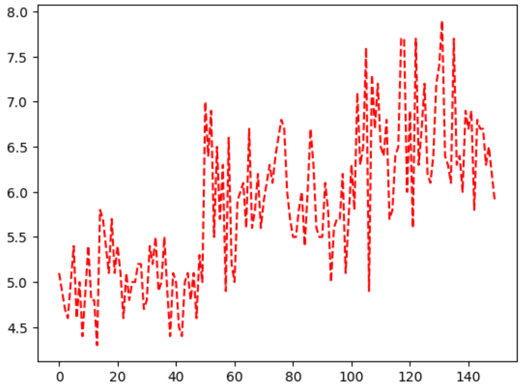




CODE

```
import matplotlib.pyplot as plt
plt.plot(iris["sepal.length"], "r--")
plt.show
```

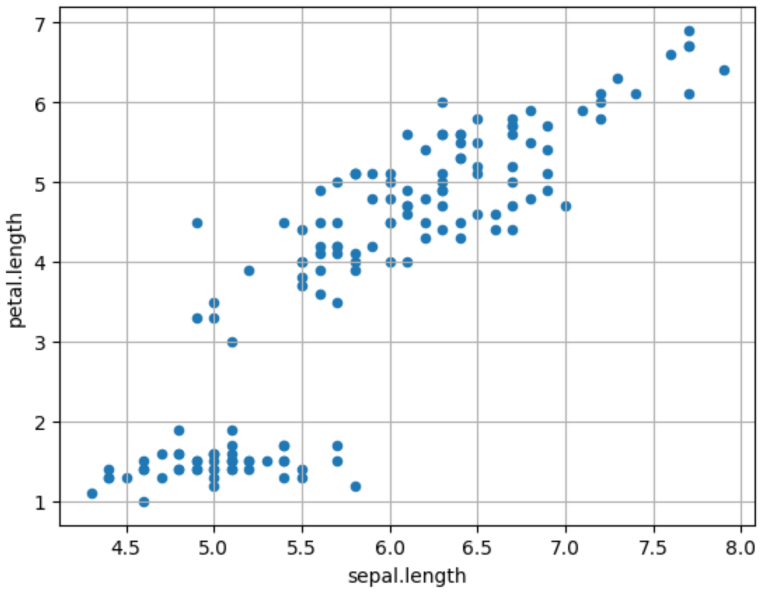
OUTPUT



CODE

```
iris.plot(kind ="scatter",
          x ='sepal.length',
          y ='petal.length')
plt.grid()
```

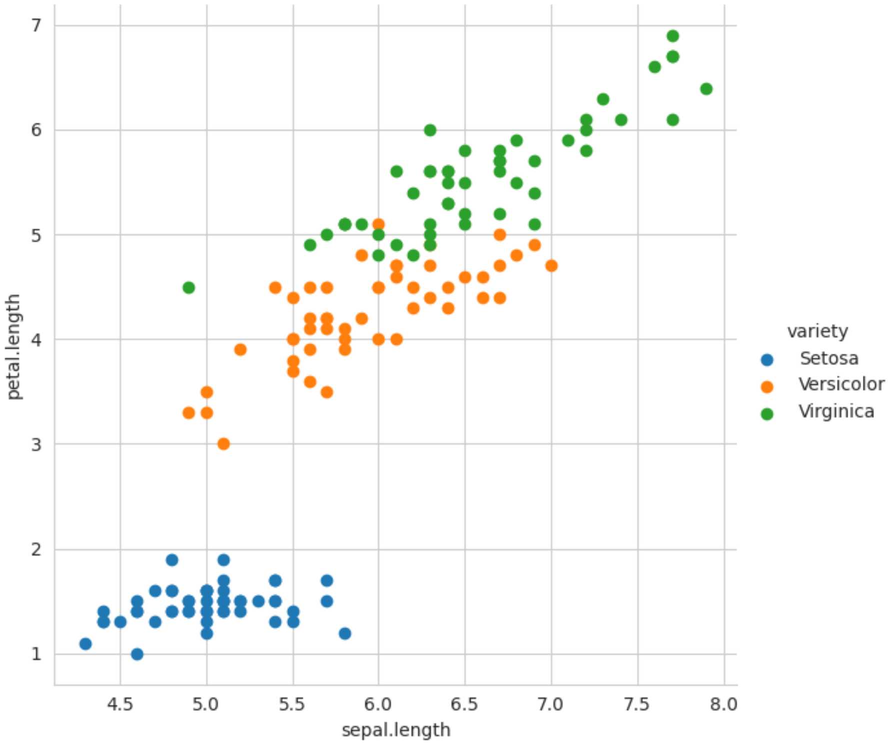
OUTPUT



CODE

```
import seaborn as sns
sns.set_style("whitegrid")
sns.FacetGrid(iris, hue ="variety",
            height = 6).map(plt.scatter,
                          'sepal.length',
                          'petal.length').add_legend()
```

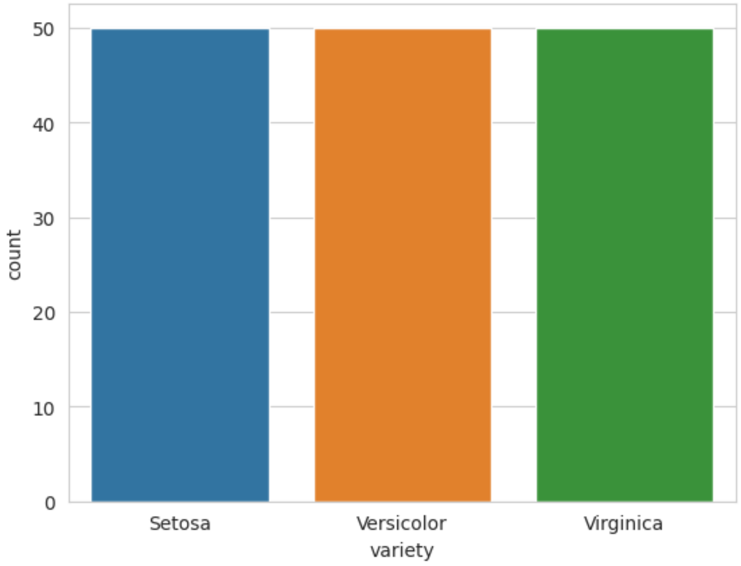
OUTPUT



CODE

```
sns.countplot(x='variety', data=iris, )
plt.show()
```

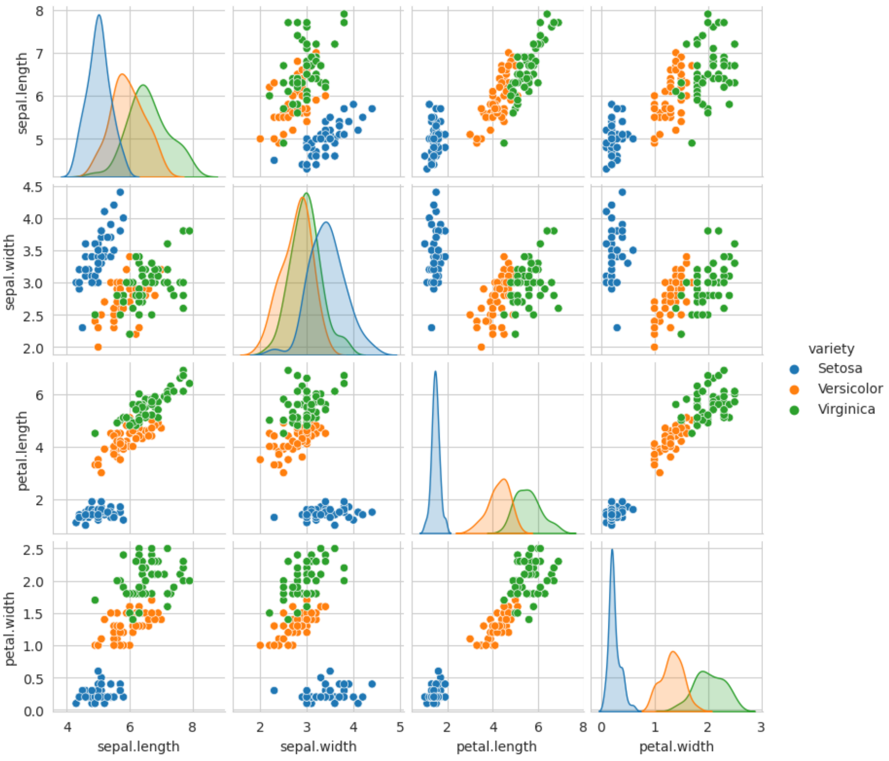
OUTPUT



CODE

```
sns.pairplot(iris,hue='variety', height=2)
```

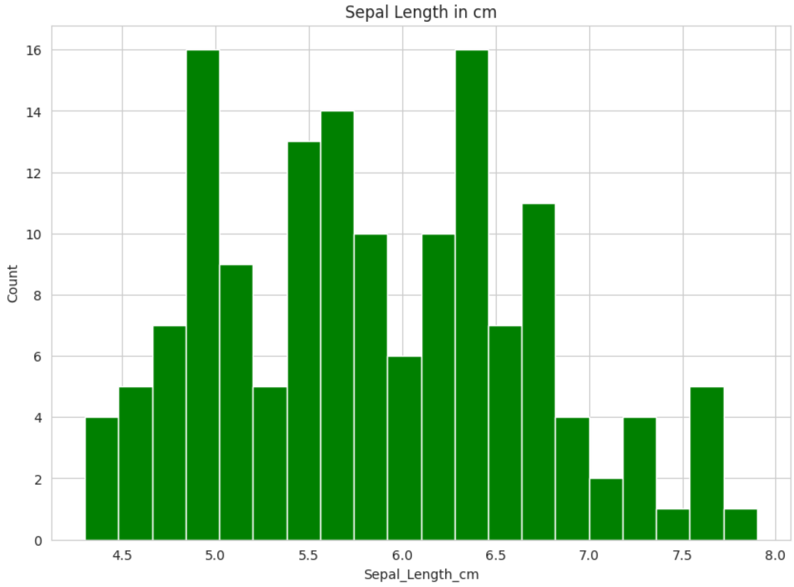
OUTPUT



CODE

```
plt.figure(figsize = (10, 7))
x = iris["sepal.length"]
plt.hist(x, bins = 20, color = "green")
plt.title("Sepal Length in cm")
plt.xlabel("Sepal_Length_cm")
plt.ylabel("Count")
```

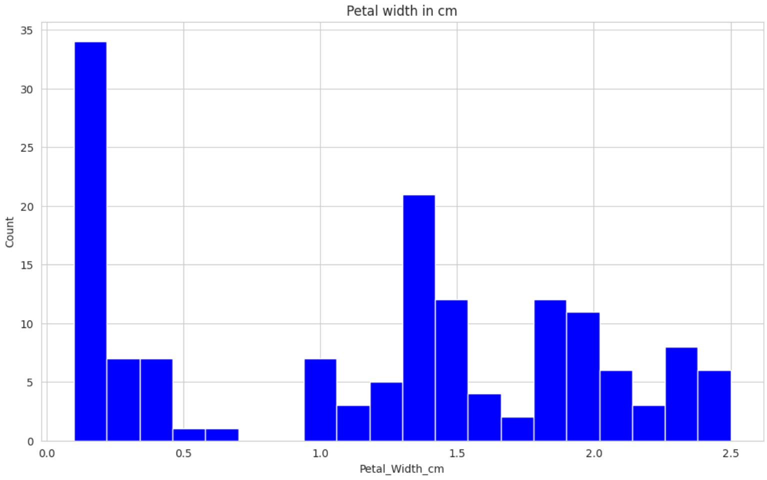
OUTPUT



CODE

```
plt.figure(figsize = (12, 7))
x = iris["petal.width"]
plt.hist(x, bins =20, color = "blue")
plt.title("Petal width in cm")
plt.xlabel("Petal_Width_cm")
plt.ylabel("Count")
```

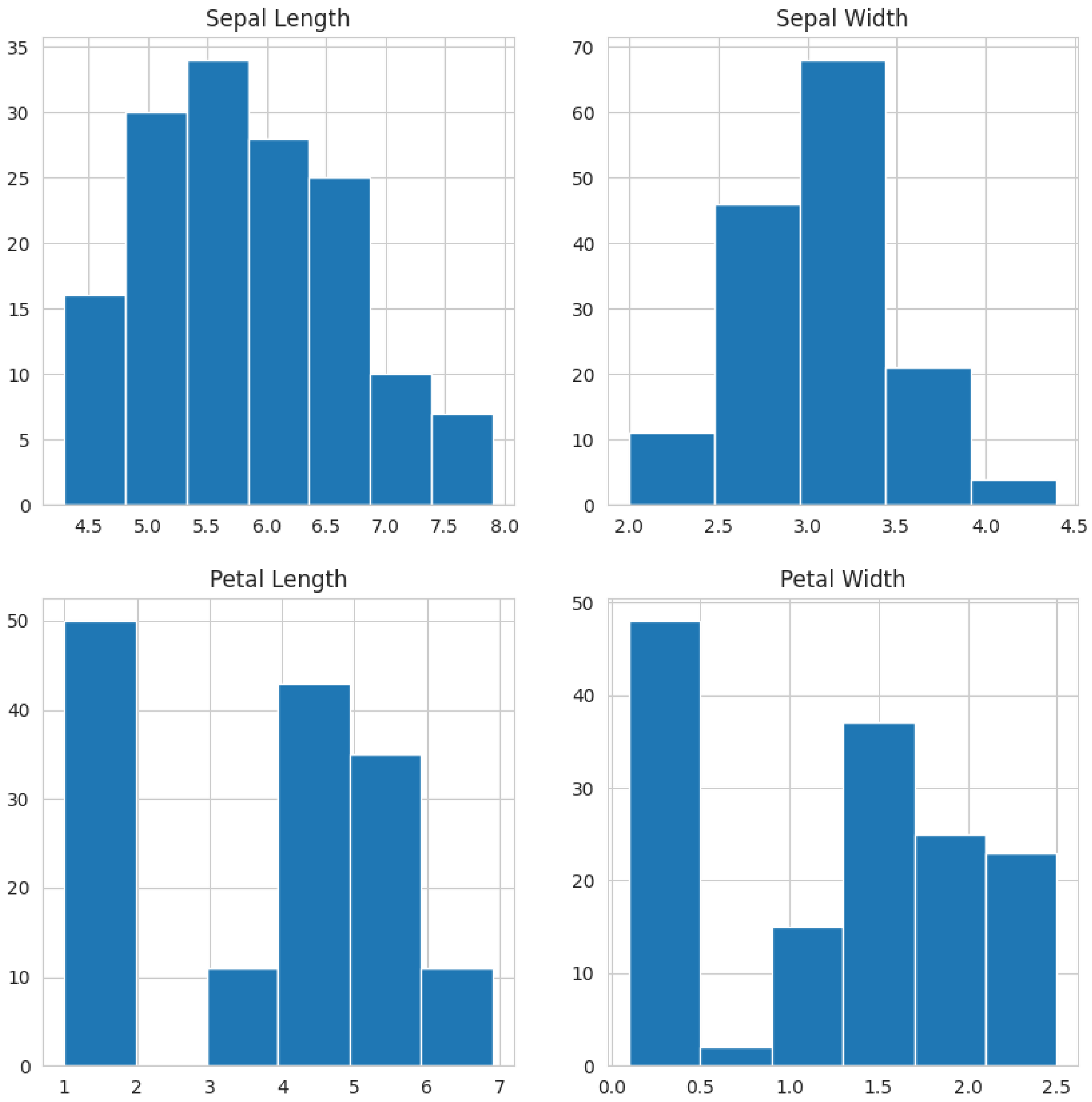
OUTPUT



CODE

```
fig, axes = plt.subplots(2, 2, figsize=(10,10))
axes[0,0].set_title("Sepal Length")
axes[0,0].hist(iris['sepal.length'], bins=7)
axes[0,1].set_title("Sepal Width")
axes[0,1].hist(iris['sepal.width'], bins=5);
axes[1,0].set_title("Petal Length")
axes[1,0].hist(iris['petal.length'], bins=6);
axes[1,1].set_title("Petal Width")
axes[1,1].hist(iris['petal.width'], bins=6);
```

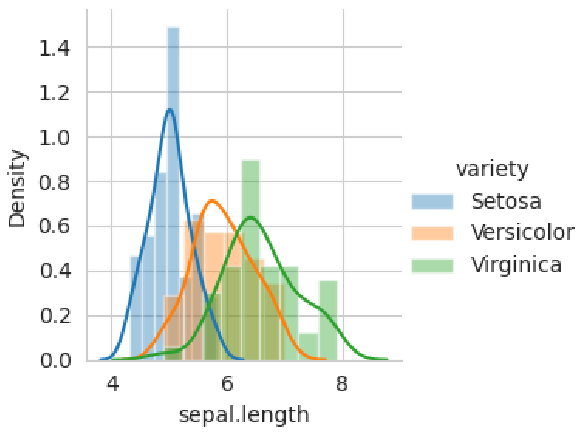
OUTPUT

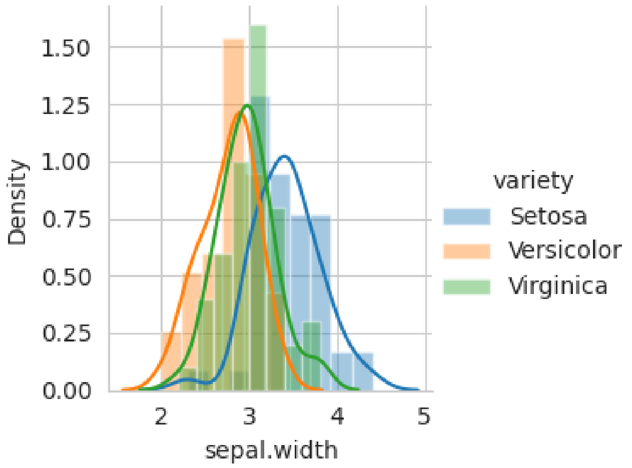


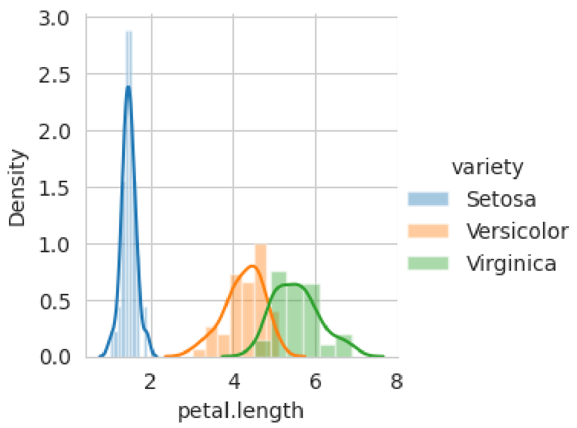
CODE

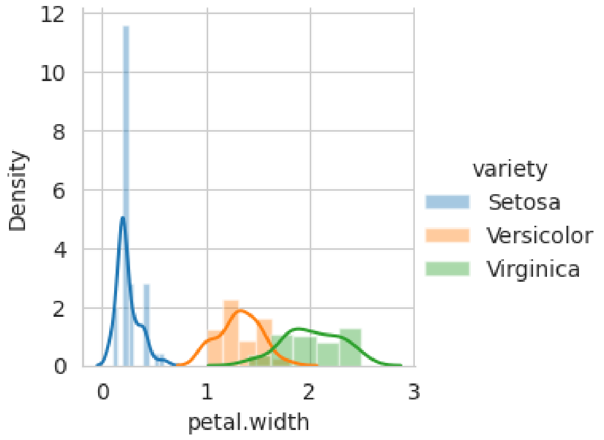
```
plot = sns.FacetGrid(iris, hue="variety")
plot.map(sns.distplot, "sepal.length").add_legend()
plot = sns.FacetGrid(iris, hue="variety")
plot.map(sns.distplot, "sepal.width").add_legend()
plot = sns.FacetGrid(iris, hue="variety")
plot.map(sns.distplot, "petal.length").add_legend()
plot = sns.FacetGrid(iris, hue="variety")
plot.map(sns.distplot, "petal.width").add_legend()
plt.show()
```

OUTPUT





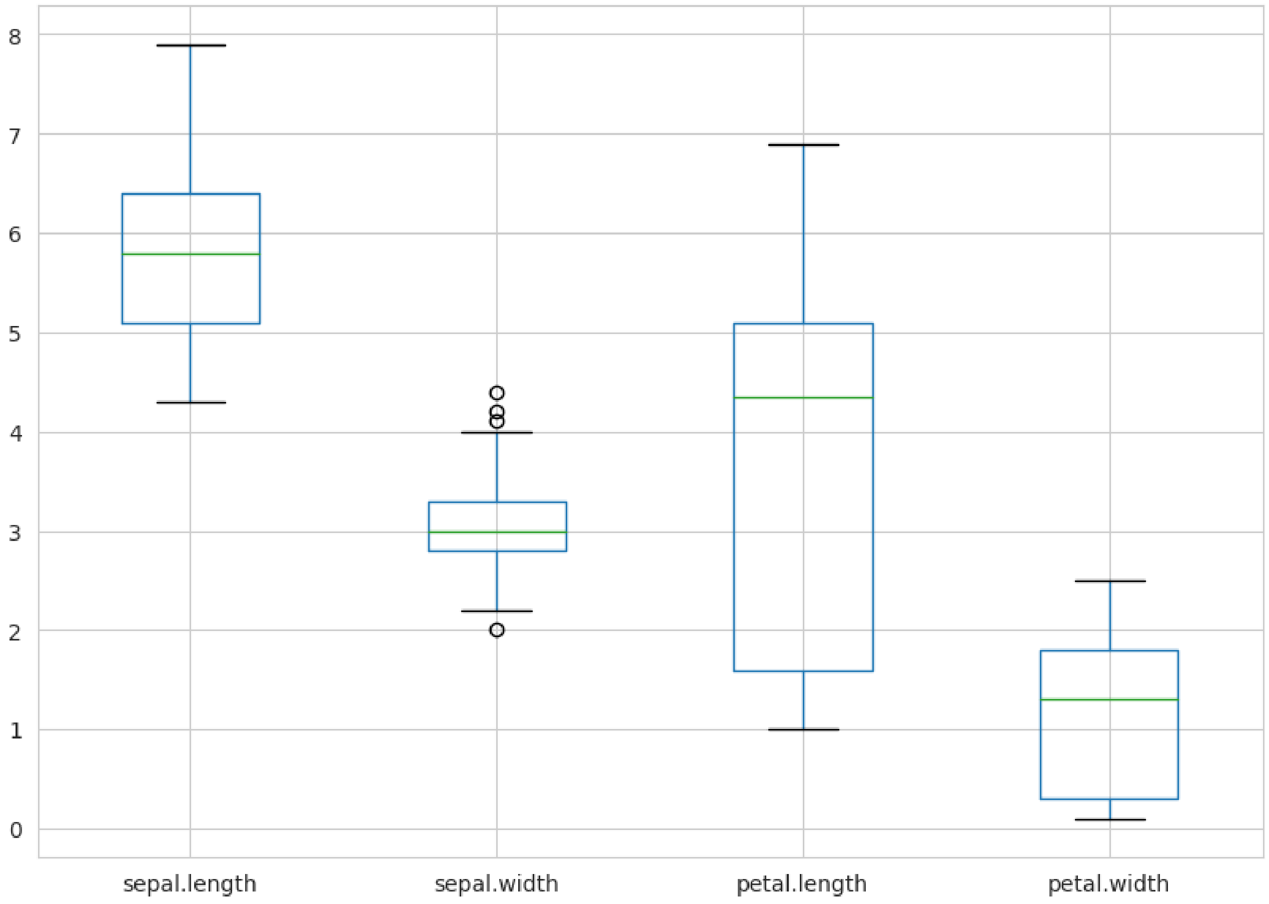




CODE

```
plt.figure(figsize = (10, 7))
iris.boxplot()
```

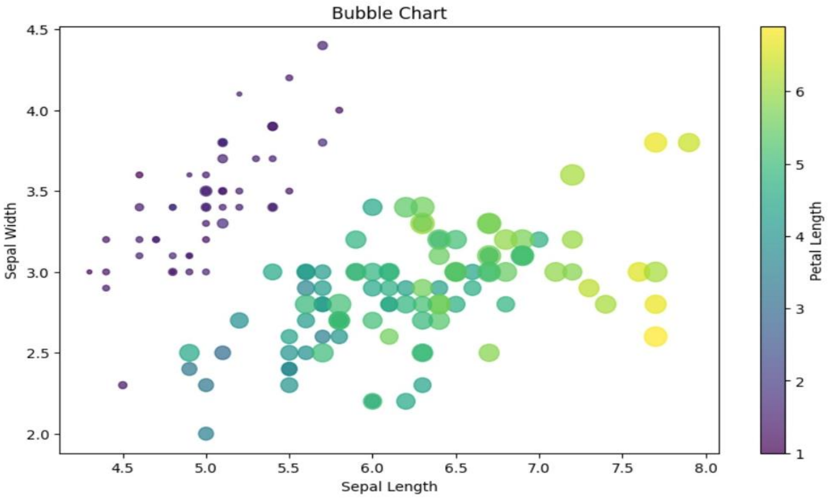
OUTPUT



CODE

```
import seaborn as sns

plt.figure(figsize=(10, 6))

plt.scatter(x=iris["sepal_length"], y=iris["sepal_width"], c=iris["petal_length"], s=iris["petal_width"] * 100, cmap="viridis", alpha=0.7)

plt.colorbar(label="Petal Length")

plt.xlabel("Sepal Length")

plt.ylabel("Sepal Width")

plt.title("Bubble Chart")

plt.show()
```

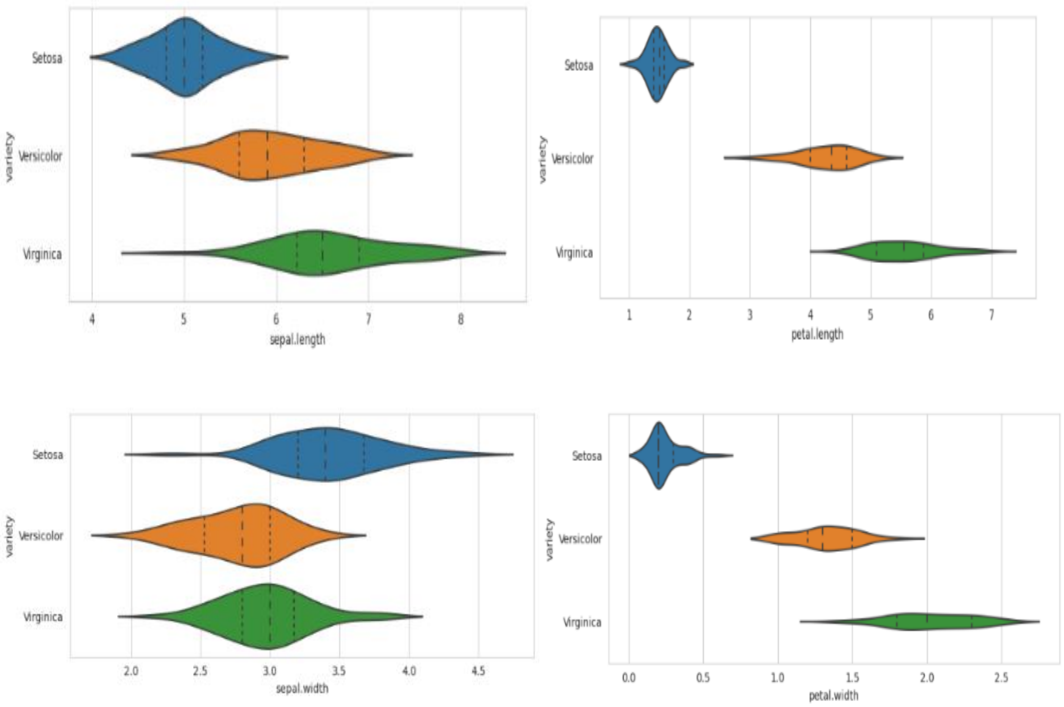
OUTPUT



CODE

```
import matplotlib.gridspec as gridspec

fig = plt.figure(figsize=(9, 40))

outer = gridspec.GridSpec(4, 1, wspace=0.2, hspace=0.2)
for i, col in enumerate(iris.columns[:-1]):
  inner = gridspec.GridSpecFromSubplotSpec(2, 1,subplot_spec=outer[i], wspace=0.2, hspace=0.4)
  ax = plt.Subplot(fig, inner[1])
  _ = sns.violinplot(y="variety", x=f"{col}", data=iris, inner='quartile', ax=ax)
  fig.add_subplot(ax)
fig.show()
```

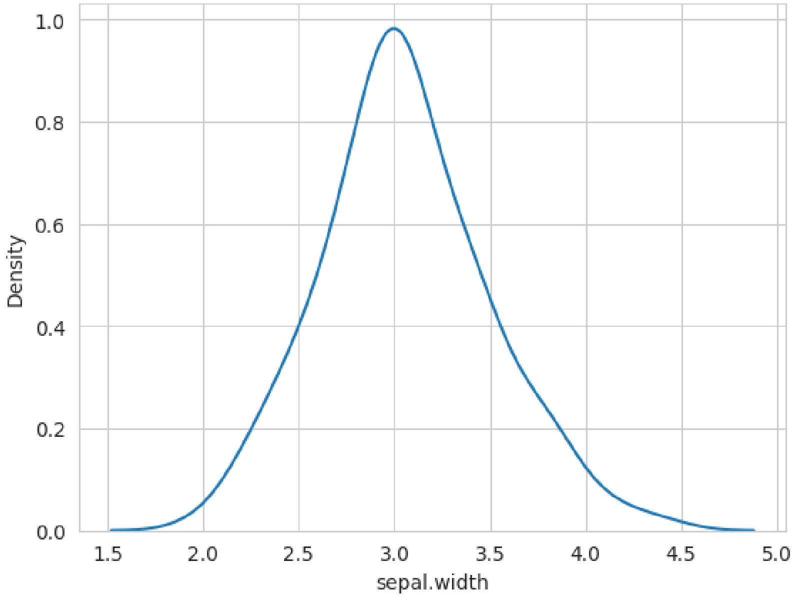
OUTPUT



CODE

```
sns.kdeplot(iris['sepal.width'])
```

OUTPUT



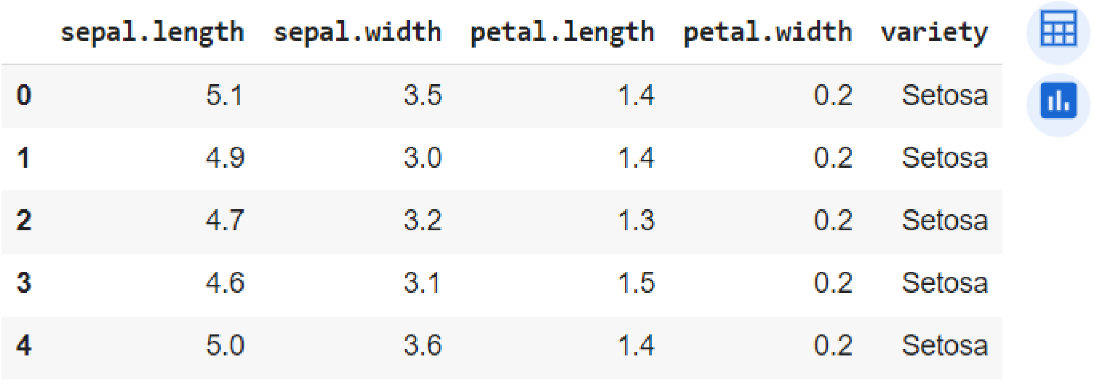
EXPERIMENT NO :10

AIM: Classification using Naïve Bayes with iris dataset

CODE

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

df = pd.read_csv("iris.csv")
X = df.iloc[:,:4].values
y = df['variety'].values
df.head(5)
```

OUTPUT



	sepal.length	sepal.width	petal.length	petal.width	variety
0	5.1	3.5	1.4	0.2	Setosa
1	4.9	3.0	1.4	0.2	Setosa
2	4.7	3.2	1.3	0.2	Setosa
3	4.6	3.1	1.5	0.2	Setosa
4	5.0	3.6	1.4	0.2	Setosa

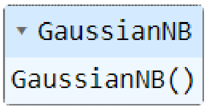
CODE

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)

from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.fit_transform(X_test)
```

CODE

```
from sklearn.naive_bayes import GaussianNB

classifier = GaussianNB()

classifier.fit(X_train, y_train)
```

OUTPUT



```
▾ GaussianNB
GaussianNB()
```

CODE

```
classifier.score(X_test,y_test)
```

OUTPUT

0.9

CODE

```
y_pred = classifier.predict(X_test)
y_pred
```

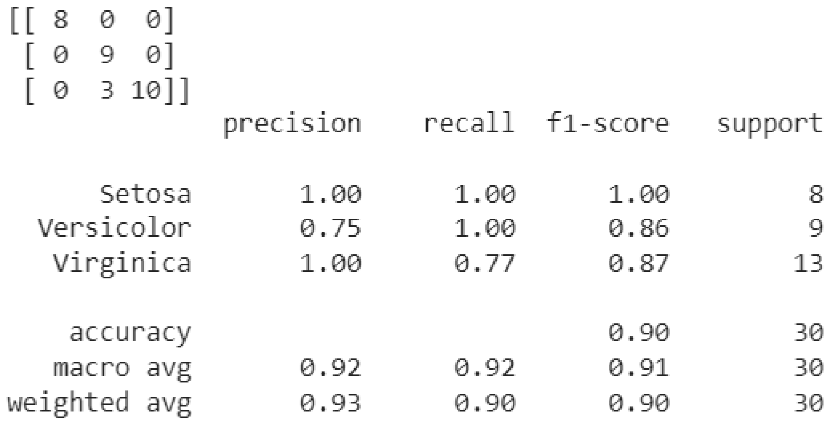
OUTPUT

array(['Virginica', 'Setosa', 'Versicolor', 'Virginica', 'Virginica', 'Versicolor', 'Virginica', 'Virginica', 'Setosa', 'Versicolor', 'Versicolor', 'Setosa', 'Versicolor', 'Versicolor', 'Virginica', 'Setosa', 'Virginica', 'Versicolor', 'Versicolor', 'Setosa', 'Versicolor', 'Virginica', 'Setosa', 'Virginica', 'Versicolor', 'Setosa', 'Versicolor', 'Setosa', 'Virginica', 'Versicolor'], dtype='<U10')

CODE

```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

OUTPUT



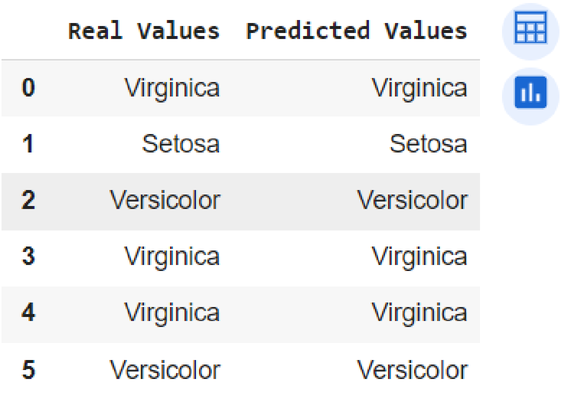
```
[[ 8  0  0]
 [ 0  9  0]
 [ 0  3 10]]
              precision    recall  f1-score   support

      Setosa       1.00      1.00      1.00         8
  Versicolor       0.75      1.00      0.86         9
   Virginica       1.00      0.77      0.87        13

    accuracy                           0.90        30
   macro avg       0.92      0.92      0.91        30
weighted avg       0.93      0.90      0.90        30
```

CODE

```
df_result = pd.DataFrame({'Real Values':y_test, 'Predicted Values':y_pred})
```

```
df_result
```

OUTPUT



	Real Values	Predicted Values
0	Virginica	Virginica
1	Setosa	Setosa
2	Versicolor	Versicolor
3	Virginica	Virginica
4	Virginica	Virginica
5	Versicolor	Versicolor

EXPERIMENT NO :14

AIM : Decision tree using titanic dataset

CODE

```
import pandas as pd
#df = pd.read_csv('titanic.csv', index_col='PassengerId')
df = pd.read_csv('titanic.csv')
print(df.head(2))
```

OUTPUT

```
PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1

                                               Name     Sex   Age  SibSp  \
0                            Braund, Mr. Owen Harris    male  22.0      1
1  Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0      1

   Parch     Ticket     Fare Cabin Embarked
0      0  A/5 21171   7.2500   NaN        S
1      0   PC 17599  71.2833   C85        C
```

CODE

```
df.shape
```

OUTPUT

```
(891, 12)
```

CODE

```
df = df[['Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare', 'Survived']]
df.shape
```

OUTPUT

```
(891, 7)
```

CODE

```
df['Sex'] = df['Sex'].map({'male': 0, 'female': 1})
df = df.dropna()
df.shape
```

OUTPUT

```
(714, 7)
```

CODE

```
X = df.drop('Survived', axis=1)
y = df['Survived']

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=1)

from sklearn import tree
model = tree.DecisionTreeClassifier()
model.fit(X_train, y_train)
model.score(X_test, y_test)
```

OUTPUT

```
0.8156424581005587
```

CODE

```
from sklearn.metrics import accuracy_score
y_predict = model.predict(X_test)
print("Accuracy:",accuracy_score(y_test, y_predict) )
```

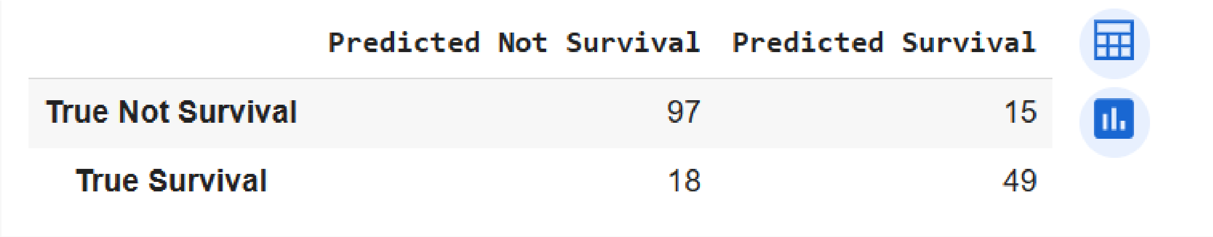
OUTPUT

```
Accuracy: 0.8156424581005587
```

CODE

```
from sklearn.metrics import confusion_matrix
pd.DataFrame(
    confusion_matrix(y_test, y_predict),
    columns=['Predicted Not Survival', 'Predicted Survival'],
    index=['True Not Survival', 'True Survival']
```

OUTPUT



	Predicted Not Survival	Predicted Survival
True Not Survival	97	15
True Survival	18	49

CODE

```
from sklearn import tree
tree.plot_tree(model,filled=True)
```

OUTPUT