In [1]:

```
import pandas as pd
import numpy as np
from sklearn import datasets
from collections import Counter
```

In [2]:

```
iris=datasets.load_iris()
Species=iris.target
data=pd.DataFrame(np.c_[iris.data,Species.reshape((Species.shape[0],1))],columns=iris.fe
ature_names+['Species'])
data.head()
```

Out[2]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	Species
0	**5.1**	**3.5**	**1.4**	**0.2**	**0.0**
1	**4.9**	**3.0**	**1.4**	**0.2**	**0.0**
2	**4.7**	**3.2**	**1.3**	**0.2**	**0.0**
3	**4.6**	**3.1**	**1.5**	**0.2**	**0.0**
4	**5.0**	**3.6**	**1.4**	**0.2**	**0.0**

In [3]:

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

Out[3]:

```
Index(['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)',
       'petal width (cm)', 'Species'],
      dtype='object')
```

In [4]:

```
from sklearn.model_selection import train_test_split
train,test=train_test_split(data,test_size=0.2,random_state=123)
```

In [12]:

```
class Naive_Bayes():
    def __init__(self,data):
        self.data=data
        self.X_train=data.drop('Species',axis=1)
        self.y_train=data['Species']
        self.mean_std={}

    def fit(self):
        self.result=Counter(self.y_train)

        for target in self.result.keys():
            for col in self.X_train.columns:
                self.mean_std[target,col,"mean"]=self.data[self.data['Species']==target]
.mean()[col]
                self.mean_std[target,col,"std"]=self.data[self.data['Species']==target].
std()[col]

        for i in self.result:
            self.result[i]=round(self.result[i]/len(self.X_train.index),8)

    def predict(self,X_test):
        prediction=[]
        for i in X_test.index:
```

```
            prob_index={}
            for target in self.result:
                prob=self.result[target]
                for col in self.X_train:
                    l=1/(((2*np.pi)**0.5)*self.mean_std[target,col,"std"])
                    m=-((X_test[col][i]-self.mean_std[target,col,"mean"])**2)
                    n=2*(self.mean_std[target,col,"std"]**2)
                    prob=prob*l*np.exp(m/n)

                prob_index[target]=prob

            probability=0
            for target in prob_index:
                if prob_index[target]>probability:
                    pred=target
                    probability=prob_index[target]
            prediction.append(pred)
        return prediction
```

In [13]:

```
model=Naive_Bayes(train)
model.fit()
```

In [14]:

```
X_test=test.drop('Species',axis=1)
y_test=test['Species']
predictions=model.predict(X_test)
```

In [15]:

```
from sklearn.metrics import accuracy_score
accuracy_score(y_test,predictions)
```

Out[15]:

```
0.9666666666666667
```

In [16]:

```
from sklearn.naive_bayes import GaussianNB
model1=GaussianNB()
model1.fit(data.iloc[:,:4],data.iloc[:,4])
predictions=model.predict(data.iloc[:,:4])
from sklearn.metrics import accuracy_score
print(accuracy_score(data.iloc[:,4],predictions))
```

```
0.9666666666666667
```