Experiment – 8

Aim: Write a program in python to implement Naive Bayes Algorithm.

1. Show the distribution curve
2. Show accuracy of the classifier

Theory:

Naive Bayes classifiers are a collection of classification algorithms based Bayes'Theorem. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e., every pair of features being classified is independent of each other.

The fundamental Naive Bayes assumption is that each feature makes an:

- independent
- equal

contribution to the outcome.

Naïve bayes uses the concept of Bayes theorem:

Bayes' theorem is also known as **Bayes' Rule** or **Bayes' law**, is used to determine the probability of a hypothesis with prior knowledge. It depends on the conditional probability.

P(A|B) = (P(B|A) * P(A)) / P(B)

P(A|B) = probability of A on the observed event B

P(B|A) = probability of B on the observed event A

Types of Naïve Bayes classifiers:

1) **Gaussian Naïve Bayes**: This type of Naive Bayes is used when variables are continuous in nature. It assumes that all the variables have a normal distribution.
2) **Multinomial Naïve Bayes**: The Multinomial Naïve Bayes classifier is used when the data is multinomial distributed. It is primarily used for document classification problems, it means a particular document belongs to which category such as Sports, Politics, education, etc.
3) **Bernoulli Naïve Bayes**: This is used when features are binary. Instead of using the frequency of the word, 1s and 0s are used to represent the presence or absence of a feature. In that case, the features will be binary and Bernoulli Naive Bayes will be used.

Dataset used: Titanic survivability dataset which provides the binary output which signifies whether the a particular passenger with certain characteristics was able to survive or not.

https://www.kaggle.com/c/titanic/data

Code

```
import pandas as pd

from sklearn.naive_bayes import GaussianNB, BernoulliNB, MultinomialNB

from sklearn.model_selection import train_test_split

from sklearn import metrics

import matplotlib.pyplot as plt

from scipy.stats import norm

import statistics
data = pd.read_csv('train.csv')
data
columns = data.columns
columns
data=data[['PassengerId', 'Pclass', 'Age', 'SibSp',
       'Parch', 'Survived']]
Data
data = data.dropna()
data
x = data.values[:,1:5]
y = data.values[:,5:6]
x_train, x_test, y_train, y_test = train_test_split(x,y)
y_train = y_train.flatten()
type(y_train)
axis = data.values[:,2:3]
axis = axis.flatten()
# axis
print("Distribution curve for age : ")
mean = statistics.mean(axis)
sd = statistics.stdev(axis)

plt.plot(axis, norm.pdf(axis, mean, sd))
plt.show()
axis = data.values[:,1:2]
axis = axis.flatten()
print("Distribution curve for Pclass (it has only 3 values: 1,2 and 3) : ")
mean = statistics.mean(axis)
sd = statistics.stdev(axis)

plt.plot(axis, norm.pdf(axis, mean, sd))
plt.show()
clf = GaussianNB()
clf.fit(x_train, y_train)
y_pred = clf.predict(x_test)
score = metrics.accuracy_score(y_test, y_pred)*100
print("Score using Gaussian naive bayes : " , score)
clf2 = BernoulliNB()
clf2.fit(x_train, y_train)
y_pred = clf2.predict(x_test)
score = metrics.accuracy_score(y_test, y_pred)*100
print("Score using Bernoulli naive bayes : " , score)
```

```
clf3 = MultinomialNB()
clf3.fit(x_train, y_train)
y_pred = clf3.predict(x_test)
score = metrics.accuracy_score(y_test, y_pred)*100
print("Score using Multinomial naive bayes : " , score)
```

Result

```
In [20]: data=data[['PassengerId', 'Pclass', 'Age', 'SibSp',
                    'Parch', 'Survived']]
         data
```

Out[20]:

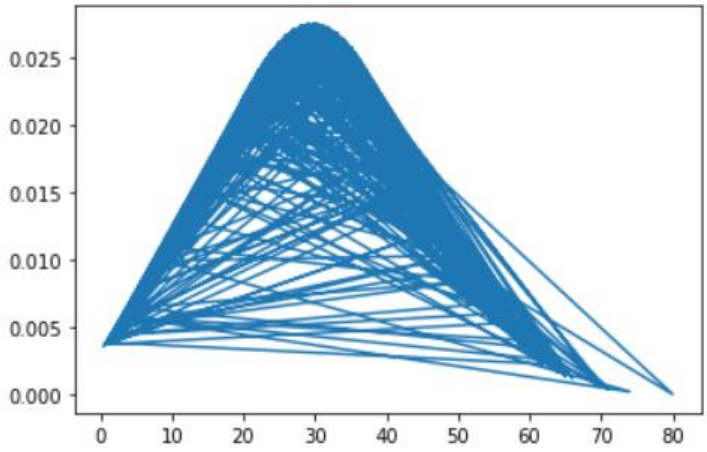
	PassengerId	Pclass	Age	SibSp	Parch	Survived
0	1	3	22.0	1	0	0
1	2	1	38.0	1	0	1
2	3	3	26.0	0	0	1
3	4	1	35.0	1	0	1
4	5	3	35.0	0	0	0
...	...	...	...	...	...	...
886	887	2	27.0	0	0	0
887	888	1	19.0	0	0	1
888	889	3	NaN	1	2	0
889	890	1	26.0	0	0	1
890	891	3	32.0	0	0	0

891 rows × 6 columns

```
In [22]: x = data.values[:,1:5]
         y = data.values[:,5:6]
         x_train, x_test, y_train, y_test = train_test_split(x,y)

In [23]: x_train

Out[23]: array([[ 3., 32.,  0.,  0.],
                [ 2., 29.,  1.,  0.],
                [ 3., 18.,  1.,  0.],
                ...,
                [ 3., 29.,  0.,  4.],
                [ 2., 25.,  0.,  0.],
                [ 3., 22.,  0.,  0.]])
```

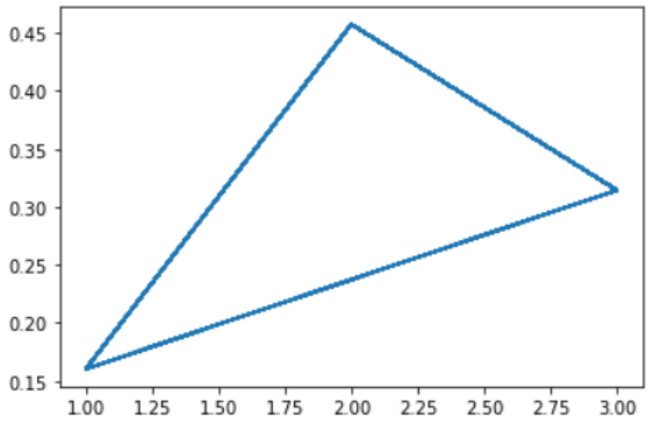
In [26]:
```
print("Distribution curve for age : ")
mean = statistics.mean(axis)
sd = statistics.stdev(axis)

plt.plot(axis, norm.pdf(axis, mean, sd))
plt.show()
```

```
Distribution curve for age :
```



In [27]:
```
axis = data.values[:,1:2]
axis = axis.flatten()
print("Distribution curve for Pclass (it has only 3 values: 1,2 and 3) : ")
mean = statistics.mean(axis)
sd = statistics.stdev(axis)

plt.plot(axis, norm.pdf(axis, mean, sd))
plt.show()
```

```
Distribution curve for Pclass (it has only 3 values: 1,2 and 3) :
```



In [28]:
```
clf = GaussianNB()
clf.fit(x_train, y_train)
y_pred = clf.predict(x_test)
score = metrics.accuracy_score(y_test, y_pred)*100
print("Score using Gaussian naive bayes : " , score)
```

```
Score using Gaussian naive bayes :  67.59776536312849
```

In [29]:
```
clf2 = BernoulliNB()
clf2.fit(x_train, y_train)
y_pred = clf2.predict(x_test)
score = metrics.accuracy_score(y_test, y_pred)*100
print("Score using Bernoulli naive bayes : " , score)
```

```
Score using Bernoulli naive bayes :  55.3072625698324
```

In [30]:
```
clf3 = MultinomialNB()
clf3.fit(x_train, y_train)
y_pred = clf3.predict(x_test)
score = metrics.accuracy_score(y_test, y_pred)*100
print("Score using Multinomial naive bayes : " , score)
```

```
Score using Multinomial naive bayes :  59.77653631284916
```

Conclusion

In this program, Gaussian, Bernoulli and Multinomial Naïve Bayes were used to successfully calculate the predictions based on the provided input. The program successfully provided the accuracy scores of the three different naïve bayes techniques. Distribution curve was successfully printed using matplotlib library.