LAB 10 : Naive Bayes Classifier

1. Binary Classification using Naive Bayes Classifier
2. Sentiment Analysis using Naive Bayes

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
```

Binary Classification using Naive Bayes Classifier

Useful References :

1. https://machinelearningmastery.com/naive-bayes-classifier-scratch-python/
2. https://www.analyticsvidhya.com/blog/2021/01/a-guide-to-the-naive-bayes-algorithm/
3. https://towardsdatascience.com/implementing-naive-bayes-algorithm-from-scratch-python-c6880cfc9c41

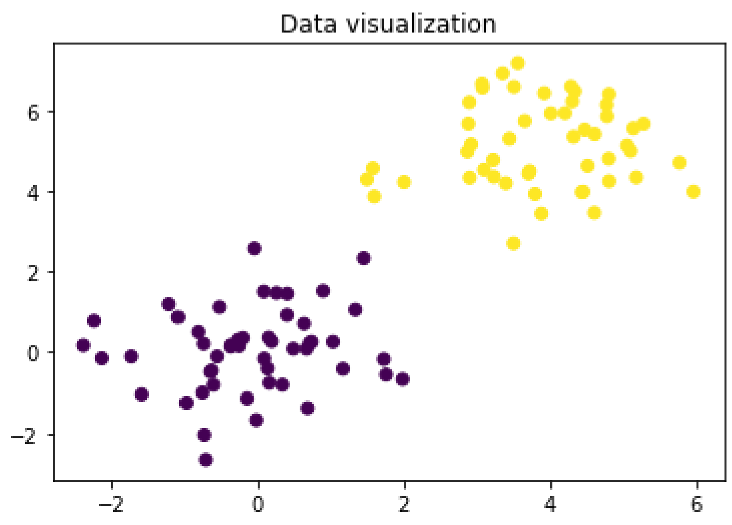
Note : The goal of this experiment is to perform and understand Naive Bayes classification by applying it on the below dataset, you can either fill in the below functions to get the result or you can create a class of your own using the above references to perform classification

1. Generation of 2D training data

In [ ]:
```
mean1=np.array([0,0])
mean2=np.array([4,5])
var=np.array([[1,0.1],[0.1,1]])
np.random.seed(0)
data1=np.random.multivariate_normal(mean1,var,50)
data2=np.random.multivariate_normal(mean2,var,50)
data=np.concatenate((data1,data2))
label=np.concatenate((np.zeros(data1.shape[0]),np.ones(data2.shape[0])))

plt.figure()
plt.scatter(data[:,0],data[:,1],c=label)
plt.title('Data visualization')
```

Out[ ]:
```
Text(0.5, 1.0, 'Data visualization')
```



In [ ]:

```
class NaiveBayes:
        def __init__(self):
                self.mean1=None
                self.mean2=None
                self.var1=None
                self.var2=None
                self.prior1=None
                self.prior2=None
                self.min = 1e-30

        def fit(self,data,label):
                self.mean1=np.mean(data[label==0],axis=0)
                self.mean2=np.mean(data[label==1],axis=0)
                self.var1=np.var(data[label==0],axis=0)
                self.var2=np.var(data[label==1],axis=0)
                self.prior1=(np.sum(label==0))/data.shape[0]
                self.prior2=(np.sum(label==1))/data.shape[0]

        def predict(self,data):
                p1=np.exp(-np.sum((data-self.mean1)**2/(self.var1+self.min)
                p2=np.exp(-np.sum((data-self.mean2)**2/(self.var2+self.min)
                p1*=self.prior1
                p2*=self.prior2
                return (p1<p2).astype(int), (p1, p2)
```

1. Test the model using some samples

In [ ]:

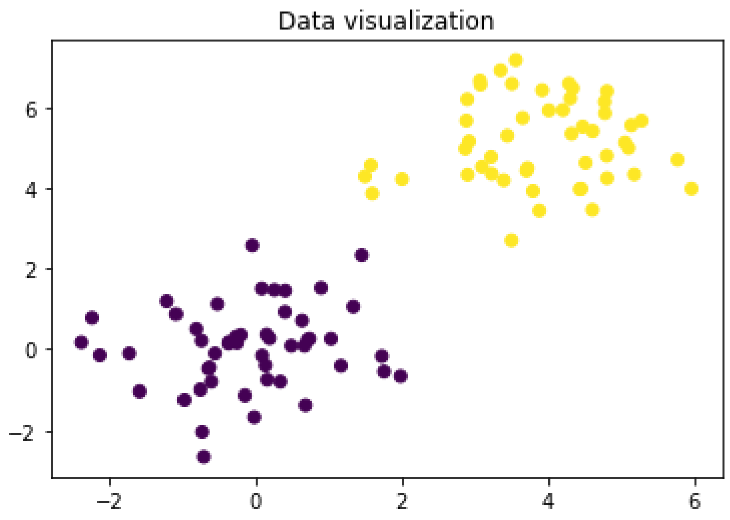
```
## Test Data Generation

mean1=np.array([0,0])
mean2=np.array([4,5])
var=np.array([[1,0.1],[0.1,1]])
np.random.seed(0)
data1=np.random.multivariate_normal(mean1,var,10)
data2=np.random.multivariate_normal(mean2,var,10)
test_data=np.concatenate((data1,data2))
y_test=np.concatenate((np.zeros(data1.shape[0]),np.ones(data2.shape[0])))
print('Test Data Size : ',test_data.shape[0])
plt.figure()
plt.scatter(data[:,0],data[:,1],c=label)
plt.title('Data visualization')
```

```
Test Data Size :  20
```

Out[ ]:

```
Text(0.5, 1.0, 'Data visualization')
```



Testing for a sample point

```
In [ ]: naiveBayes = NaiveBayes()
        naiveBayes.fit(data,label)
        _, (p1, p2) = naiveBayes.predict(test_data[0,:].reshape(1,-1))

        print(f'Class Probabilites for the first sample of test dataset: 0 -> {p1} a
```

```
Class Probabilites for the first sample of test dataset: 0 -> [0.03558711]
and 1 -> [2.0887671e-15]
```

As seen above the class probability for the 1st sample is given, we can observe that probability is higher for class 0 than 1 and hence imply that this datapoint belongs to class 0

Now Calculate the class probabilities for all the data points in the test dataset and calculate the accuracy by comparing the predicted labels with the true test labels

```
In [ ]: preds, _ = naiveBayes.predict(test_data)

        print(f'Accuracy: {np.sum(preds == y_test)/len(y_test)*100:.2f}%')
```

```
Accuracy: 100.00%
```

1. Use the Sci-kit Learn library to perform Gaussian Naive Bayes classifier on the above dataset, also report the accuracy and confusion matrix for the same

```
In [ ]: # Use gaussian naive bayes from sklearn
        from sklearn.naive_bayes import GaussianNB

        gnb = GaussianNB()
        gnb.fit(data, label)
        preds = gnb.predict(test_data)

        print(f'Accuracy: {np.sum(preds == y_test)/len(y_test)*100:.2f}%')
```

```
Accuracy: 100.00%
```

Sentiment Analysis using Naive Bayes Classifier

Go through the following article and implement the same

Keypoints :

1. The link to the dataset is given in the above article, download the same to perform sentiment analysis
2. Understanding how to deal with text data is very important since it requires a lot of preprocessing, you can go through this article if you are interested in learning more about it
3. Split the dataset into train-test and train the model
4. Report the accuracy metrics and try some sample prediction outside of those present in the dataset

Note : The goal of this experiment is to explore a practical use case of Naive bayes classifier as well as to understand how to deal with textual data, you can follow any other open source implemetations of sentiment analysis using naive bayes also

Other References :

1. https://towardsdatascience.com/sentiment-analysis-introduction-to-naive-bayes-algorithm-96831d77ac91
2. https://gist.github.com/CateGitau/6608912ca92733036c090676c61c13cd

In [ ]:
```
# Get data (only run once)
# !wget https://github.com/Hrd2D/Sentiment-analysis-on-Google-Play-store-ap
```

In [ ]:
```
import pandas as pd
```

In [ ]:
```
# Load data
data = pd.read_csv('google_play_store_apps_reviews_training.csv')

# Drop package name
data = data.drop('package_name', axis=1)

# Remove empty space and convert to lower case
data['review'] = data['review'].str.strip().str.lower()

data.head()
```

Out[ ]:

	review	polarity
0	privacy at least put some option appear offlin...	0
1	messenger issues ever since the last update, i...	0
2	profile any time my wife or anybody has more t...	0
3	the new features suck for those of us who don'...	0
4	forced reload on uploading pic on replying com...	0

In [ ]:
```
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer

# Train test split
X_train, X_test, y_train, y_test = train_test_split(data['review'], data['po
```

```python
# Vectorize the data as count vectors
countVectorizer = CountVectorizer(stop_words='english')

X_train = countVectorizer.fit_transform(X_train).toarray()
X_test = countVectorizer.transform(X_test).toarray()
```

In [ ]:
```python
# Using my code
naiveBayes = NaiveBayes()
naiveBayes.fit(X_train, y_train)

preds, _ = naiveBayes.predict(X_test)

print(f'Accuracy: {np.sum(preds == y_test)/len(y_test)*100:.2f}%')
```

```
Accuracy: 77.09%
```

In [ ]:
```python
# Using sklearn
naiveBayes = GaussianNB()
naiveBayes.fit(X_train, y_train)

preds = naiveBayes.predict(X_test)

print(f'Accuracy: {np.sum(preds == y_test)/len(y_test)*100:.2f}%')
```

```
Accuracy: 77.65%
```