Data Science Lab: Experiment 5

DS Lab	
Name: - Manan Shanghvi	**Batch: -** 2
Registration ID: - 211060019	**Date: -**

Experiment 5-
BAYESIAN CLASSIFIERS

Aim:-To Implement Bayesian Classifier.

Software Used:- Jupyter notebook

Output:-

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

```
ds=pd.read_csv('SALARY DATA.csv')
ds.head()
```

	UserID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0

GRADE: **Dated Signature:**

```
#input
x=ds.iloc[:,[2,3]].values #Age and EstimatedSalary
# output
y=ds.iloc[:,4].values #Purchased
y
```

```
array([0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1,
       0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0,
       1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0,
       1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1,
       0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1,
       1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1,
       0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0,
       1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1,
       0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1,
       1, 1, 0, 1], dtype=int64)
```

```
from sklearn.model_selection import train_test_split
xtrain, xtest, ytrain, ytest=train_test_split(x,y,test_size=0.25,random_state=0)
print(ytrain)
```

```
[0 1 0 1 1 1 0 0 0 0 0 0 1 1 1 0 1 0 0 1 0 1 0 1 0 0 1 1 1 1 0 1 0 1 0 0 1
 0 0 1 0 0 0 0 0 1 1 1 1 0 0 0 1 0 1 0 1 0 0 1 0 0 0 1 0 0 0 1 1 0 0 1 0 1
 1 1 0 0 1 1 0 0 1 1 0 1 0 0 1 1 0 1 1 1 0 0 0 0 0 1 0 0 1 1 1 1 1 0 1 1 0
 1 0 0 0 0 0 0 0 1 1 0 0 1 0 0 1 0 0 0 1 0 1 1 0 1 0 0 0 0 1 0 0 0 1 1 0 0
 0 0 1 0 1 0 0 0 1 0 0 0 0 1 1 1 0 0 0 0 0 0 0 1 1 1 1 1 0 1 0 0 0 0 0 1 0 0
 0 0 0 0 1 1 0 1 0 1 0 0 1 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 1 0 1 1 0 0 0 0 0
 0 1 1 0 0 0 0 1 0 0 0 0 1 0 1 0 1 0 0 0 1 0 0 0 1 0 1 0 0 0 0 0 1 1 1 0 0 0
 0 0 1 0 1 1 0 0 0 0 0 0 1 0 1 0 0 1 0 0 1 0 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 1
 0 0 0 0]
```

```
# standardization
from sklearn.preprocessing import StandardScaler
sc_x=StandardScaler()
xtrain=sc_x.fit_transform(xtrain)
xtest=sc_x.fit_transform(xtest)
print(xtrain[0:10])
```

```
[[ 0.58164944 -0.88670699]
 [-0.60673761  1.46173768]
 [-0.01254409 -0.5677824 ]
 [-0.60673761  1.89663484]
 [ 1.37390747 -1.40858358]
 [ 1.47293972  0.99784738]
 [ 0.08648817 -0.79972756]
 [-0.01254409 -0.24885782]
 [-0.21060859 -0.5677824 ]
 [-0.21060859 -0.19087153]]
```

```
#Model Implementation
from sklearn.linear_model import LogisticRegression
classifier =LogisticRegression (random_state=42)
classifier.fit(xtrain, ytrain)
#LogisticRegression(random_state=0)
```

```
LogisticRegression(random_state=42)
```

```
#Prediction
y_pred=y_pred= classifier.predict(xtest)
y_pred
```

```
array([0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1,
       0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1,
       0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1], dtype=int64)
```

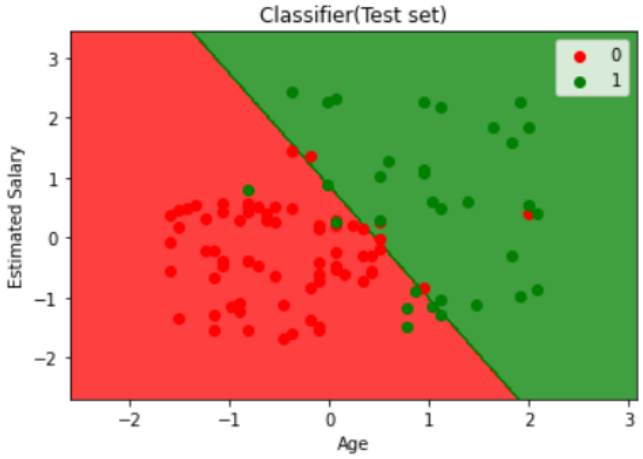
```
#confusion matrix
from sklearn.metrics import confusion_matrix
cm= confusion_matrix(ytest,y_pred)
print("Confusion Matrix: \n",cm)
```

```
Confusion Matrix:
 [[63  5]
 [ 8 24]]
```

```
from sklearn.metrics import accuracy_score
print("Accuracy:", accuracy_score(ytest, y_pred))
```

```
Accuracy: 0.87
```

```
from matplotlib.colors import ListedColormap
x_set,y_set = xtest, ytest
#creating a mesh grid start, stop , step for both Age and Estimate salary
x1, x2 =np.meshgrid(np.arange(start= x_set[:,0].min()-1,stop=x_set[:,0].max()+1, step=0.01), np.arange(start = x_set[:,1].min()-1
#contour plot
plt.contourf(x1, x2, classifier.predict(np.array([x1.ravel(), x2.ravel()]).T).reshape(x1.shape), alpha=0.75, cmap=ListedColormap(
plt.xlim(x1.min(),x1.max())
plt.ylim(x2.min(),x2.max())
for i,j in enumerate(np.unique(y_set)):
  plt.scatter(x_set[y_set==j,0], x_set[y_set==j,1],c=ListedColormap(('red','green')) (i), label=j)
plt.title('Classifier(Test set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
```




```
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
gnb.fit(xtrain,ytrain)
```

```
GaussianNB()
```

```
ypred =gnb.predict(xtest)
ypred
```

```
array([0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1,
       0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1,
       0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1,
       1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1], dtype=int64)
```

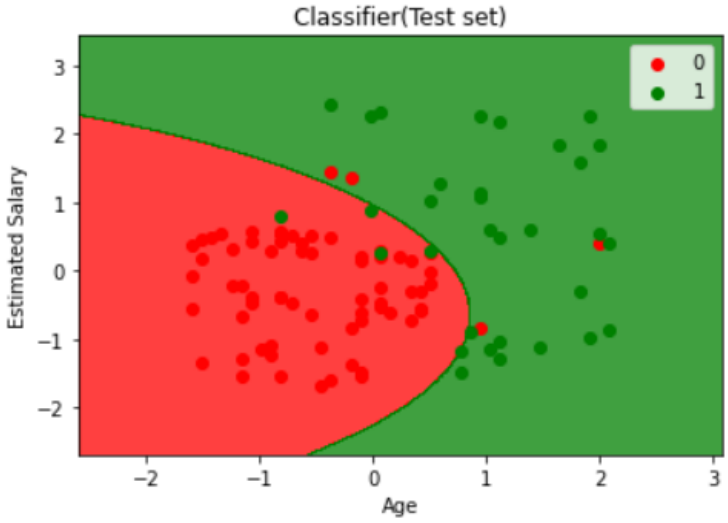
```
#confusion matrix
from sklearn.metrics import confusion_matrix
cm= confusion_matrix(ytest,ypred)
print("Confusion Matrix: \n",cm)
```

```
Confusion Matrix:
 [[64  4]
 [ 5 27]]
```

```
from sklearn.metrics import accuracy_score
print("Accuracy:", accuracy_score(ytest, ypred))
```

```
Accuracy: 0.91
```

```
from matplotlib.colors import ListedColormap
x_set,y_set = xtest, ytest
#creating a mesh grid start, stop , step for both Age and Estimate salary
x1, x2 =np.meshgrid(np.arange(start= x_set[:,0].min()-1,stop=x_set[:,0].max()+1, step=0.01), np.arange(start = x_set[:,1].min()-1
#contour plot
plt.contourf(x1, x2, gnb.predict(np.array([x1.ravel(), x2.ravel()]).T).reshape(x1.shape), alpha=0.75, cmap=ListedColormap(('red',
plt.xlim(x1.min(),x1.max())
plt.ylim(x2.min(),x2.max())
for i,j in enumerate(np.unique(y_set)):
  plt.scatter(x_set[y_set==j,0], x_set[y_set==j,1],c=ListedColormap(('red','green')) (i), label=j)
plt.title('Classifier(Test set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
```



Discussions:-

- Bayesian classifiers are a collection of classification algorithms based on 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 Naïve Bayes algorithm is used for classification problems. It is highly used in text classification. In text classification tasks, data contains high dimensions. It is used in spam filtering, sentiment detection, rating classification etc.
- The advantage of using naïve Bayes is its speed. It is fast and making predictions is easy with a high dimension of data. This model predicts the probability of an instance belonging to a class with a given set of feature values. In other words, each feature contributes to the predictions with no relation between each other. It uses Bayes theorem in the algorithm for training and prediction.
- The accuracy of the naïve bayes classifier as seen is 91% which is greater than the logistic regression.
- Naive Bayes learners and classifiers can be extremely fast compared to more sophisticated methods. The decoupling of the class conditional feature distributions means that each distribution can be independently estimated as a one dimensional distribution. This in turn helps to alleviate problems stemming from the curse of dimensionality.

Conclusion:-

In concluding the experiment involving the Bayesian Classifier we learnt the naïve bayes classification. How the data points are classified around the gaussian plane based on their probability of lying in that region or not. The accuracy of this model is 91%. We get the graph of age vs estimated salary as shown in the output using naïve bayes classifier.

GRADE: **Dated Signature:**