Department of Computer Science and Engineering (Data Science)

Subject: Machine Learning – I (DJ19DSC402)

AY: 2021-22

Experiment 4 (Naïve Bayes Classifier)

Name: Dev Patel **SAP ID: 60009200016**

Batch: K/K1 **Date: 11/04/22**

Aim: Implement Naïve Bayes Classifier on a given Dataset.

Theory:

Naïve Bayes Classifier Algorithm

- Naïve Bayes algorithm is a supervised learning algorithm, which is based on **Bayes theorem** and used for solving classification problems.
- It is mainly used in *text classification* that includes a high-dimensional training dataset.
- Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions.
- **It is a probabilistic classifier, which means it predicts on the basis of the probability of an object**.
- Some popular examples of Naïve Bayes Algorithm are **spam filtration, Sentimental analysis, and classifying articles**.

The Naïve Bayes algorithm is comprised of two words Naïve and Bayes, which can be described as:

- **Naïve**: It is called Naïve because it assumes that the occurrence of a certain feature is independent of the occurrence of other features. Such as if the fruit is identified on the bases of colour, shape, and taste, then red, spherical, and sweet fruit is recognized as an apple. Hence each feature individually contributes to identify that it is an apple without depending on each other.
- **Bayes**: It is called Bayes because it depends on the principle of Bayes' Theorem.

Bayes' Theorem:

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

$$P(A|B)=\frac{P(B|A)P(A)}{P(B)}$$

Where,

P(A|B) is Posterior probability: Probability of hypothesis A on the observed event B.

P(B|A) is Likelihood probability: Probability of the evidence given that the probability of a hypothesis is true.

P(A) is Prior Probability: Probability of hypothesis before observing the evidence.
P(B) is Marginal Probability: Probability of Evidence.
Types of Naïve Bayes Model:
There are three types of Naive Bayes Model, which are given below:

Gaussian: The Gaussian model assumes that features follow a normal distribution. This means if predictors take continuous values instead of discrete, then the model assumes that these values are sampled from the Gaussian distribution.

- **Multinomial**: 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.
  The classifier uses the frequency of words for the predictors.
- **Bernoulli**: The Bernoulli classifier works similar to the Multinomial classifier, but the predictor variables are the independent Booleans variables. Such as if a particular word is present or not in a document. This model is also famous for document classification tasks.

Lab Assignments to complete in this session:

Use the given dataset and perform the following tasks:
Dataset 1: Breastcancer.csv
Dataset 2: Social_Network_Ads.csv
Dataset 3: emails.csv
Dataset 4: German_credit_score.csv

1. Perform required pre-processing on Dataset 1 and fit a Naïve Bayes classifier built from scratch. Evaluate the f1 score of the classifier.
2. Using sklearn library fit a Naïve Bayes classifier on Dataset 2 and Dataset 4.

Code and Output:

In [1]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import sklearn.preprocessing
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score , confusion_matrix, f1_score
%matplotlib inline
```

1. Perform required preprocessing on Dataset 1 and fit a Naïve Bayes classifier built from scratch and evaluate the f1 score of classifier.

In [2]:

```
brc = pd.read_csv('/content/sample_data/Breast_cancer_data.csv')
brc.head()
```

Out[2]:

	mean_radius	mean_texture	mean_perimeter	mean_area	mean_smoothness	diagnosis
0	17.99	10.38	122.80	1001.0	0.11840	0
1	20.57	17.77	132.90	1326.0	0.08474	0
2	19.69	21.25	130.00	1203.0	0.10960	0
3	11.42	20.38	77.58	386.1	0.14250	0
4	20.29	14.34	135.10	1297.0	0.10030	0

In [3]:

```
brc.isnull().sum()
```

Out[3]:

```
mean_radius        0
mean_texture       0
mean_perimeter     0
mean_area          0
mean_smoothness    0
diagnosis          0
dtype: int64
```

In [4]:

```
brc.duplicated().sum()
```

Out[4]:

```
0
```

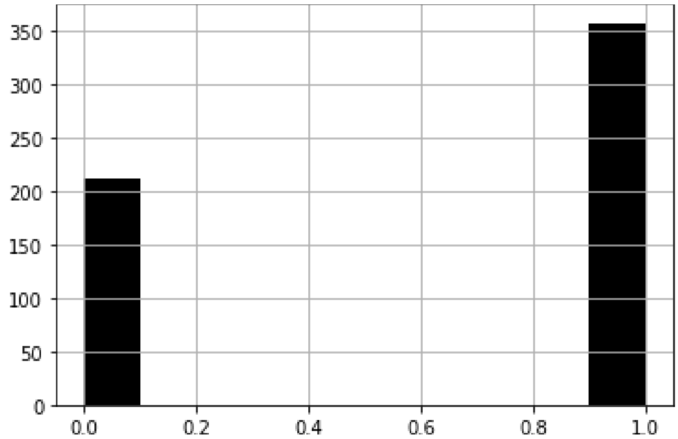
EDA

In [5]:

```
brc["diagnosis"].hist(color = 'black')
```

Out[5]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0cefd1ced0>
```

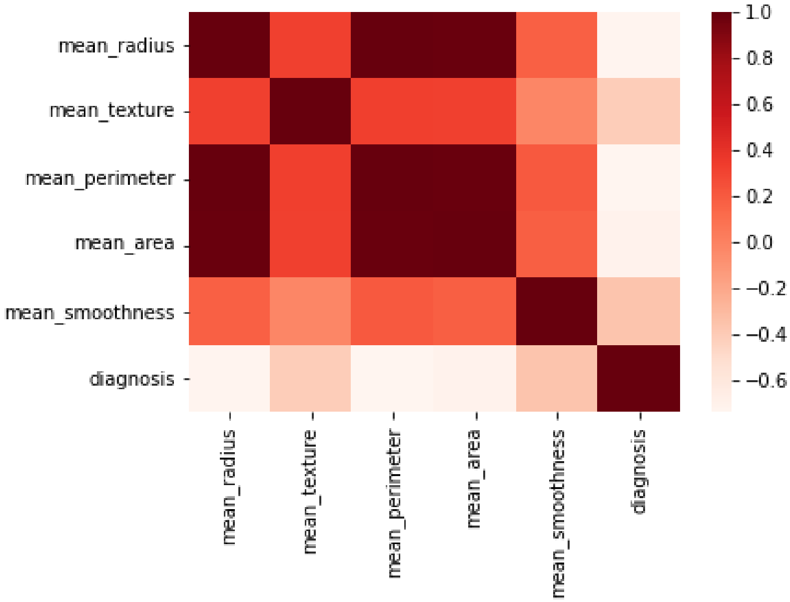


In [6]:

```
sns.heatmap(brc.corr(),cmap = 'Reds')
```

Out[6]:

<matplotlib.axes._subplots.AxesSubplot at 0x7f0cefc1dc90>



In [7]:

```
brc = brc[["mean_radius", "mean_texture", "mean_smoothness", "diagnosis"]] #dropping corr
elated columns
brc.head()
```

Out[7]:

	mean_radius	mean_texture	mean_smoothness	diagnosis
0	17.99	10.38	0.11840	0
1	20.57	17.77	0.08474	0
2	19.69	21.25	0.10960	0
3	11.42	20.38	0.14250	0
4	20.29	14.34	0.10030	0

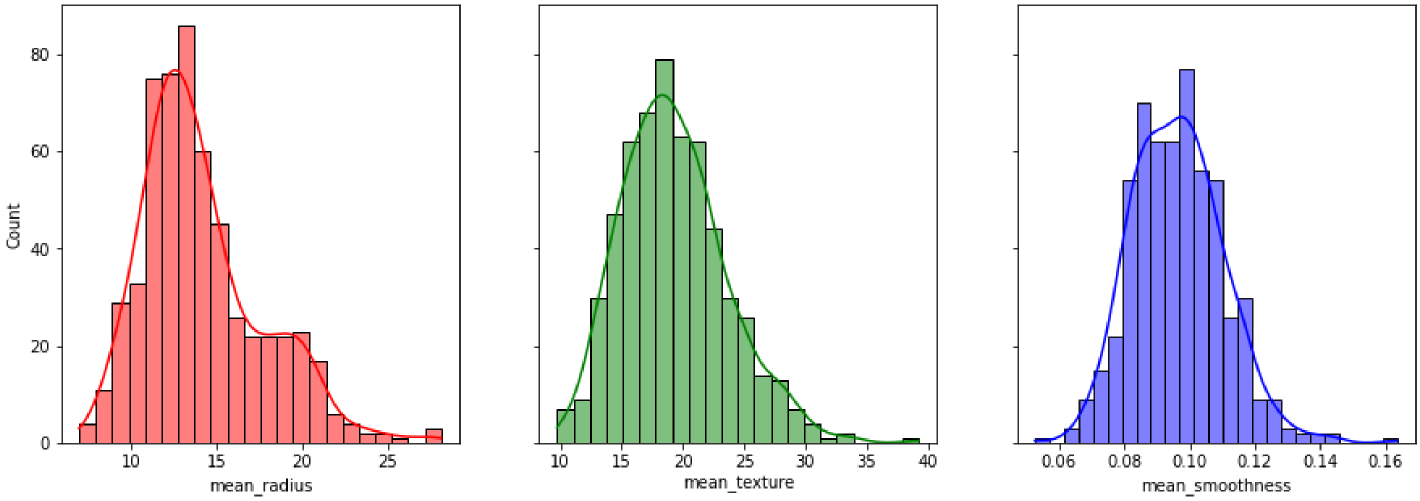
In [8]:

```
fig, axes = plt.subplots(1, 3, figsize=(15, 5), sharey=True)
sns.histplot(brc, ax=axes[0], x="mean_radius", color='r', kde=True)
sns.histplot(brc, ax=axes[1], x="mean_texture", color ='g', kde=True)
sns.histplot(brc, ax=axes[2], x="mean_smoothness", color='b', kde=True)
```

Out[8]:

<matplotlib.axes._subplots.AxesSubplot at 0x7f0ceceb6e10>

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0eeeeb0e10>
```



In [9]:

```python
# Calculate P(Y=y) for all possible y

def calculate_prior(df, Y):
    classes = sorted(list(df[Y].unique()))
    prior = []
    for i in classes:
        prior.append(len(df[df[Y]==i])/len(df))
    return prior
```

In [10]:

```python
# To Calculate P(X=x|Y=y) using Gaussian distribution

def calculate_likelihood_gaussian(df, feat_name, feat_val, Y, label):
    feat = list(df.columns)
    df = df[df[Y]==label]
    mean, std = df[feat_name].mean(), df[feat_name].std()
    p_x_given_y = (1 / (np.sqrt(2 * np.pi) * std)) *  np.exp(-((feat_val-mean)**2 / (2 *
std**2 )))
    return p_x_given_y
```

In [11]:

```python
# To calculate P(X=x1|Y=y)P(X=x2|Y=y)...P(X=xn|Y=y) * P(Y=y) for all y and to find the ma
ximum of all

def naive_bayes_gaussian(df, X, Y):
    features = list(df.columns)[:-1]
    prior = calculate_prior(df, Y)

    Y_pred = []
    for x in X:
        labels = sorted(list(df[Y].unique()))
        likelihood = [1]*len(labels)
        for j in range(len(labels)):
            for i in range(len(features)):
                likelihood[j] *= calculate_likelihood_gaussian(df, features[i], x[i], Y,
labels[j])

        post_prob = [1]*len(labels)
        for j in range(len(labels)):
            post_prob[j] = likelihood[j] * prior[j]
        Y_pred.append(np.argmax(post_prob))

    return np.array(Y_pred)
```

In [12]:

```python
# Test Gaussian model
train, test = train_test_split(brc, test_size=.25, random_state=41)
```

```
X_test = test.iloc[:,:-1].values
Y_test = test.iloc[:,-1].values
Y_pred = naive_bayes_gaussian(train, X=X_test, Y="diagnosis")

print("Confusion matrix:\n",confusion_matrix(Y_test, Y_pred))
print("\nF1 score of the model =",f1_score(Y_test, Y_pred))
```

```
Confusion matrix:
 [[48  5]
 [ 0 90]]

F1 score of the model = 0.972972972972973
```

2. Using sklearn library fit a Naïve Bayes classifier on Social_Network_Ads.csv and German_credit_score.csv

Social_Network_Ads.csv

In [13]:

```
df = pd.read_csv('/content/sample_data/Social_Network_Ads.csv')
df.head()
```

Out[13]:

	User ID	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

In [14]:

```
df.isnull().sum()
```

Out[14]:

```
User ID            0
Gender             0
Age                0
EstimatedSalary    0
Purchased          0
dtype: int64
```

In [15]:

```
df.duplicated().sum()
```

Out[15]:

```
0
```

In [16]:

```
le = sklearn.preprocessing.LabelEncoder()

df['Gender']= le.fit_transform(df['Gender'])
df.head()
```

Out[16]:

	User ID	Gender	Age	EstimatedSalary	Purchased

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

In [17]:

```
X = df.iloc[:, :-1]
y = df.iloc[:,-1]
df = df.drop("User ID",axis =1)
```

In [18]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 4 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Gender           400 non-null    int64
 1   Age              400 non-null    int64
 2   EstimatedSalary  400 non-null    int64
 3   Purchased        400 non-null    int64
dtypes: int64(4)
memory usage: 12.6 KB
```

In [19]:

```
X_train, X_test, y_train, y_test = train_test_split (X,y, test_size = 0.2, random_state
= 41)
```

In [20]:

```
se = sklearn.preprocessing.StandardScaler()
X_train = se.fit_transform(X_train)
X_test = se.fit_transform(X_test)
X_train
```

Out[20]:

```
array([[-1.53743998, -0.95118973, -0.06599205,  2.21413759],
       [ 1.14481726, -0.95118973, -0.64988486,  0.0381701 ],
       [ 1.07684764,  1.05131497,  0.42058529, -0.1359073 ],
       ...,
       [-1.15890907, -0.95118973,  1.49105544,  0.35731199],
       [-1.37407341,  1.05131497, -0.74720033,  0.2992862 ],
       [ 0.17766241,  1.05131497,  1.29642451, -1.3544491 ]])
```

In [21]:

```
gnb = GaussianNB()
gnb.fit(X_train, y_train)
```

Out[21]:

```
GaussianNB()
```

In [22]:

```
y_pred = gnb.predict(X_test)
y_trainpred = gnb.predict(X_train)
```

In [23]:

```
print("Accuracy in train_set= ",accuracy_score(y_train,y_trainpred))
print("Accuracy in test_set= ",accuracy_score(y_test,y_pred))
```

```
Accuracy in train_set=  0.884375
```

```
Accuracy in train_set=  0.884375
Accuracy in test_set=  0.8625
```

In [24]:

```
print(confusion_matrix(y_test,y_pred))
```

```
[[46  4]
 [ 7 23]]
```

German_credit_score.csv

In [25]:

```
gc = pd.read_csv('/content/sample_data/german_credit_data.csv')
```

In [26]:

```
gc.head()
```

Out[26]:

	Unnamed: 0	Age	Sex	Job	Housing	Saving accounts	Checking account	Credit amount	Duration	Purpose	Risk
0	0	67	male	2	own	NaN	little	1169	6	radio/TV	good
1	1	22	female	2	own	little	moderate	5951	48	radio/TV	bad
2	2	49	male	1	own	little	NaN	2096	12	education	good
3	3	45	male	2	free	little	little	7882	42	furniture/equipment	good
4	4	53	male	2	free	little	little	4870	24	car	bad

In [27]:

```
gc.isnull().sum()
```

Out[27]:

```
Unnamed: 0            0
Age                   0
Sex                   0
Job                   0
Housing               0
Saving accounts     183
Checking account    394
Credit amount         0
Duration              0
Purpose               0
Risk                  0
dtype: int64
```

In [28]:

```
gc.shape
```

Out[28]:

```
(1000, 11)
```

In [29]:

```
gc.duplicated().sum()
```

Out[29]:

```
0
```

In [30]:

```
gc = gc.drop("Unnamed: 0",axis = 1)
```

In [31]:

```
gc.drop("Checking account",axis =1)
sav = gc["Saving accounts"].value_counts()
print(sav)
```

```
little         603
moderate       103
quite rich      63
rich            48
Name: Saving accounts, dtype: int64
```

In [32]:

```
sav = "little"
gc["Saving accounts"] = gc["Saving accounts"].fillna(sav)
```

In [33]:

```
le = sklearn.preprocessing.LabelEncoder()

gc['Sex']= le.fit_transform(gc['Sex'])
gc['Housing']= le.fit_transform(gc['Housing'])
gc['Saving accounts']= le.fit_transform(gc['Saving accounts'])
gc['Checking account']= le.fit_transform(gc['Checking account'])
gc['Purpose']= le.fit_transform(gc['Purpose'])
gc['Risk']= le.fit_transform(gc['Risk'])
gc.head()
```

Out[33]:

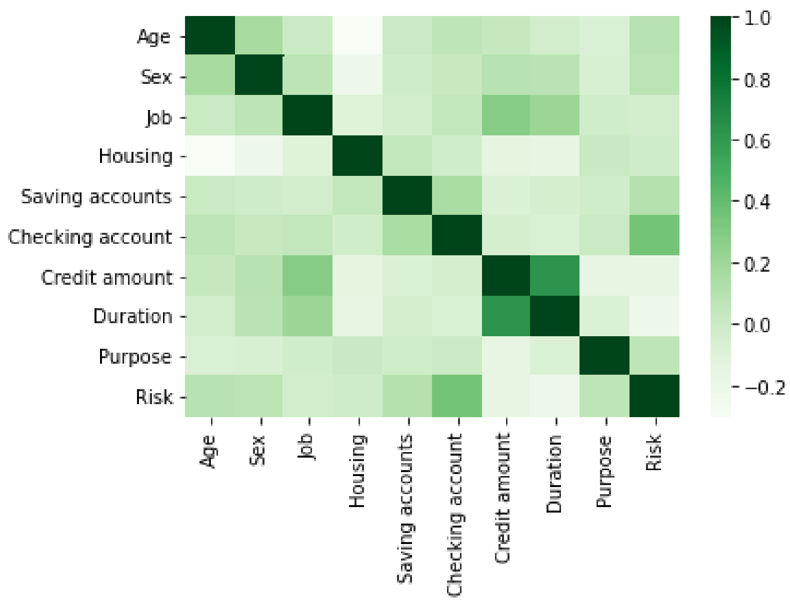
	Age	Sex	Job	Housing	Saving accounts	Checking account	Credit amount	Duration	Purpose	Risk
0	67	1	2	1	0	0	1169	6	5	1
1	22	0	2	1	0	1	5951	48	5	0
2	49	1	1	1	0	3	2096	12	3	1
3	45	1	2	0	0	0	7882	42	4	1
4	53	1	2	0	0	0	4870	24	1	0

In [34]:

```
sns.heatmap(gc.corr(),cmap = 'Greens')
```

Out[34]:

<matplotlib.axes._subplots.AxesSubplot at 0x7f0ceaca2d90>



In [35]:

```
gc.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 10 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Age               1000 non-null   int64
 1   Sex               1000 non-null   int64
 2   Job               1000 non-null   int64
 3   Housing           1000 non-null   int64
 4   Saving accounts   1000 non-null   int64
 5   Checking account  1000 non-null   int64
 6   Credit amount     1000 non-null   int64
 7   Duration          1000 non-null   int64
 8   Purpose           1000 non-null   int64
 9   Risk              1000 non-null   int64
dtypes: int64(10)
memory usage: 78.2 KB
```

In [36]:

```
X = gc.iloc[:, :-1]
y = gc.iloc[:,-1]
```

In [37]:

```
X_train, X_test, y_train, y_test = train_test_split (X,y, test_size = 0.2, random_state
= 41)
```

In [38]:

```
se = sklearn.preprocessing.StandardScaler()
X_train = se.fit_transform(X_train)
X_test = se.fit_transform(X_test)
X_train
```

Out[38]:

```
array([[ 2.56212222, -1.52299116,  0.14200319, ..., -0.88762883,
        -1.25200183,  1.07086157],
       [ 0.68347115,  0.65660263,  0.14200319, ...,  0.55321125,
        -0.74210799, -0.95082399],
       [-1.2846395 , -1.52299116,  0.14200319, ..., -0.8407751 ,
        -0.74210799,  1.07086157],
       ...,
       [ 0.95184988, -1.52299116, -1.41419617, ...,  0.11320899,
        -0.48716107, -0.95082399],
       [-1.19517992, -1.52299116,  0.14200319, ..., -0.56458473,
        -0.99705491,  1.07086157],
       [-0.47950332,  0.65660263,  0.14200319, ..., -0.53992488,
        -1.25200183,  1.07086157]])
```

In [39]:

```
gnb = GaussianNB()
gnb.fit(X_train, y_train)
```

Out[39]:

```
GaussianNB()
```

In [40]:

```
y_pred = gnb.predict(X_test)
y_trainpred = gnb.predict(X_train)
```

In [41]:

```
print("Accuracy in train set= ",accuracy_score(y_train,y_trainpred))
```

```
print("Accuracy in test_set= ",accuracy_score(y_test,y_pred))
```

```
Accuracy in train_set=  0.7325
Accuracy in test_set=  0.72
```

In [42]:

```
print(confusion_matrix(y_test,y_pred))
```

```
[[ 27  42]
 [ 14 117]]
```