Q1. A company conducted a survey of its employees and found that 70% of the employees use the company's health insurance plan, while 40% of the employees who use the plan are smokers. What is the probability that an employee is a smoker given that he/she uses the health insurance plan?

```
A = Employee uses the health insurance plan
B = Employee is a smoker

We are given:
P(A) = 70% = 0.7 (probability that an employee uses the health insurance plan)
P(B|A) = 40% = 0.4 (probability that an employee is a smoker given that they use the
health insurance plan)

Bayes' theorem states:

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

We need to calculate P(B|A), the probability that an employee is a smoker given that
they use the health insurance plan.

Using Bayes' theorem:

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

P(B|A) = 0.40 * 0.70 / 0.70
       = 0.40
```

Q2. What is the difference between Bernoulli Naive Bayes and Multinomial Naive Bayes?

The difference between Bernoulli Naive Bayes and Multinomial Naive Bayes lies in their assumptions about the distribution of features and the way they handle feature presence or absence.

Bernoulli Naive Bayes: Bernoulli Naive Bayes is typically used when the features are binary or represent the presence or absence of certain characteristics. It assumes that each feature follows a Bernoulli distribution, where each feature is considered independently and has a binary outcome (0 or 1). In this model, the presence of a feature is represented by 1, and its absence is represented by 0. The probabilities are estimated based on the occurrence of features in each class. It is commonly used for text classification tasks, such as sentiment analysis, where the presence or absence of certain words is relevant.

Multinomial Naive Bayes: Multinomial Naive Bayes is suitable for discrete features that represent counts or frequencies, such as word counts in text data. It assumes that the features follow a multinomial distribution, which models the frequency of occurrences of each feature in each class. Instead of considering the presence or absence of features, it takes into account the frequencies or counts of the features. The probabilities are estimated based on the occurrence and frequency of features in each class. It is commonly used in text classification tasks, such as document categorization or spam filtering, where the frequency of words is important.

Q3. How does Bernoulli Naive Bayes handle missing values?

Bernoulli Naive Bayes assumes that each feature follows a Bernoulli distribution, where each feature is considered independently and has a binary outcome (0 or 1). When it comes to missing values, Bernoulli Naive Bayes typically treats them as a separate category or as a different value, depending on how the missing values are represented in the data.

There are a few common approaches to handling missing values in Bernoulli Naive Bayes:

Separate category: One approach is to treat missing values as a separate category or a distinct value for each feature. This means creating a separate class or value to represent the missingness of a feature. In this case, during the training phase, the model estimates the probabilities of each feature taking the value of 0, 1, or the missing category. During the classification phase, if a feature has a missing value, the model considers the missing category and incorporates it into the classification process.

Ignoring missing values: Another approach is to simply ignore the instances with missing values during the training and classification phases. In this case, the instances with missing values are excluded from the calculations of probabilities for each feature. When classifying a new instance with missing values, the model would assign equal probabilities to each class and not consider the missing features in the classification process.

The choice between these approaches depends on the specific problem, the amount of missing data, and the potential impact of missingness on the classification task. It's important to note that the treatment of missing values in Bernoulli Naive Bayes can vary based on the specific implementation or library used.

Q4. Can Gaussian Naive Bayes be used for multi-class classification?

Yes, Gaussian Naive Bayes can be used for multi-class classification. Gaussian Naive Bayes is a probabilistic algorithm that can be used for classification problems where the input features are continuous-valued. In the case of multi-class classification, Gaussian Naive Bayes assumes that the conditional probability distribution of the input features given the class label follows a Gaussian distribution. It then calculates the posterior probability of each class given the input features using Bayes' theorem and chooses the class with the highest probability as the predicted class. One way to handle multi-class classification using Gaussian Naive Bayes is to use the "one-vs-all" or "one-vs-rest" approach. In this approach, a separate binary classification model is trained for each class label. Each binary classifier predicts whether an input belongs to that class or not. The final prediction is made by selecting the class with the highest probability among all the binary classifiers.

In [3]:

```
import pandas as pd
df = pd.read_csv('spambase.csv')
```

In [4]:

```
df
```

Out[4]:

	0	0.64	0.64.1	0.1	0.32	0.2	0.3	0.4	0.5	0.6	...	0.41	0.42	0.43	0.778
0	0.21	0.28	0.50	0.0	0.14	0.28	0.21	0.07	0.00	0.94	...	0.000	0.132	0.0	0.372
1	0.06	0.00	0.71	0.0	1.23	0.19	0.19	0.12	0.64	0.25	...	0.010	0.143	0.0	0.276
2	0.00	0.00	0.00	0.0	0.63	0.00	0.31	0.63	0.31	0.63	...	0.000	0.137	0.0	0.137
3	0.00	0.00	0.00	0.0	0.63	0.00	0.31	0.63	0.31	0.63	...	0.000	0.135	0.0	0.135
4	0.00	0.00	0.00	0.0	1.85	0.00	0.00	1.85	0.00	0.00	...	0.000	0.223	0.0	0.000
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4595	0.31	0.00	0.62	0.0	0.00	0.31	0.00	0.00	0.00	0.00	...	0.000	0.232	0.0	0.000
4596	0.00	0.00	0.00	0.0	0.00	0.00	0.00	0.00	0.00	0.00	...	0.000	0.000	0.0	0.353
4597	0.30	0.00	0.30	0.0	0.00	0.00	0.00	0.00	0.00	0.00	...	0.102	0.718	0.0	0.000
4598	0.96	0.00	0.00	0.0	0.32	0.00	0.00	0.00	0.00	0.00	...	0.000	0.057	0.0	0.000
4599	0.00	0.00	0.65	0.0	0.00	0.00	0.00	0.00	0.00	0.00	...	0.000	0.000	0.0	0.125

4600 rows × 58 columns

In [21]:

```
x = df.drop('1', axis = 1)
y = df['1']
x.shape  ,y.shape
```

Out[21]:

```
((4600, 57), (4600,))
```

In [22]:

```
from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test = train_test_split(x,y,random_state = 42 ,test_size = 0.3)
```

In [27]:

```
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
gnb.fit(x_train,y_train)

y_pred = gnb.predict(x_test)

from sklearn.model_selection import KFold,cross_val_score
CV = KFold(n_splits= 10)

from sklearn.metrics import confusion_matrix,accuracy_score,classification_report
print(confusion_matrix(y_test,y_pred))
print(accuracy_score(y_test,y_pred))
print(classification_report(y_test,y_pred))

import numpy as np
cross_score = np.mean(cross_val_score(GaussianNB(),x_train,y_train,cv = CV,scoring = 'ac
print(f'Mean of Accuracy of k = 10 Cross validation is {cross_score}')
```

```
[[582 221]
 [ 30 547]]
0.8181159420289855
              precision    recall  f1-score   support

           0       0.95      0.72      0.82       803
           1       0.71      0.95      0.81       577

    accuracy                           0.82      1380
   macro avg       0.83      0.84      0.82      1380
weighted avg       0.85      0.82      0.82      1380

Mean of Accuracy of k = 10 Cross validation is 0.8201863354037267
```

In [ ]: