Part 1:

1. Suppose we have Car data provided on Page 2 collected and the dataset contains three features. The first feature is the color, the second feature is Type, and the third feature is Origin. The target attribute is marked Stolen, which indicates whether a specific car is stolen or not. Suppose we have the following training data including 14 training samples or examples. Using Naive Bayes Classifier to classify a new instance which follows a condition New Instance = (Blue, SUV, Domestic) into (Yes or No). Please include the detailed calculation process

Firstly we need to calculate:

1. **P** (Stolen = Yes | Blue, SUV, Domestic)
2. **P** (Stolen = No | Blue, SUV, Domestic)

First we calculate:

P(Stolen = Yes) = 6/14

P(Stolen = No) = 8/14

Color	Stolen = Yes	Stolen = No
Red	3/6	4/6
Yellow	2/6	2/8
Blue	1/6	2/8

Type	Stolen = Yes	Stolen = No
SUV	2/6	5/6
Sports	4/6	3/8

Origin	Stolen = Yes	Stolen = No
Domestic	2/6	5/6
Imported	4/6	3/8

Now:

- P(Blue, SUV, Domestic) = P (Blue, SUV, Domestic | Stolen = Yes) + P (Blue, SUV, Domestic | Stolen = No)

= 6/14 * (1/6 * 2/6 * 2/6) + 8/14 (2/8 * 5/8 * 5/8) = 257/4023

- P(Blue, SUV, Domestic | Stolen = Yes)

= P(Blue | Stolen = Yes) * P(SUV | Stolen = Yes) * P(Domestic | Stolen = Yes)

= 1/6 * 2/6 * 2/6

- P(Blue, SUV, Domestic | Stolen = No)

= P(Blue | Stolen = No) * P(SUV | Stolen = No) * P(Domestic | Stolen = No)

= 2/8 * 5/8 * 5/8

1. P (Stolen = Yes | Blue, SUV, Domestic) =

(P(Stolen = Yes) * P(Blue, SUV, Domestic | Stolen = Yes)) / P(Blue, SUV, Domestic)

= (6/14 * 1/6 / 2/6 * 2/6) / (257/4023) = 0.1245

2. P (Stolen = No | Blue, SUV, Domestic) =

(P(Stolen = No) * P(Blue, SUV, Domestic | Stolen = No)) / P(Blue, SUV, Domestic)

= (8/14 * 2/8 * 5/8 * 5/8) / (257/4023) = 0.8735

P (Stolen = No | Blue, SUV, Domestic) > P (Stolen = Yes | Blue, SUV, Domestic)

So our final Decision will be not stolen

Part 1:

2. Consider the following loss table, which contains three actions and two classes. Calculate the expected risk of three actions, and determine the rejection area of P(Class1| x).

Page
Date

Part(1)

2. Let us calculate the expected risks of the three actions:

$R(\alpha_1|X) = 0 \cdot P(C_1|X) + 6\,P(C_2|X)$
$= 6(1 - P(C_1|X))$

$R(\alpha_2|X) = 3P(C_1|X) + 0\,P(C_2|X)$
$= 3P(C_1|X)$

$R(\alpha_r|X) = 2$

We choose α_1 if

$R(\alpha_1|X) < 2 \Rightarrow 6(1 - P(C_1|X)) < 2$
$1 - P(C_1|X) < \frac{2}{6}\ (\frac{2}{6})$
$P(C_1|X) > 1 - \frac{2}{6}$
$P(C_1|X) > \frac{4}{6}$
$P(C_1|X) > \frac{2}{3}$

We choose α_2 IF

$R(\alpha_2|X) < 2 \Rightarrow 3P(C_1|X) < 2$
$P(C_1|X) < \frac{2}{3}$
$P(C_1|X) < \frac{2}{3}$

We reject if $\frac{2}{3} < P(C_1|X) < \frac{4}{6}$

Life`s made up of little things.
No great sacrifice of duty

Part 2:

1) This function loads the spambase dataset which contains email messages labeled as spam or not spam. The function returns the dataset split into features and labels, as well as the list of class names.

We uses the read_csv() function from the pandas library to load the dataset from the URL into a pandas DataFrame object. The iloc function is used to split the dataset into features X and labels y, where X contains all rows and all columns except for the last column, and y contains all rows and only the last column.

the function uses the unique() function from pandas to get the list of unique values in the 'spam' column, which corresponds to the two class names: '0' for not spam and '1' for spam. The function returns the features X, labels y, and class names as a tuple.

a) we split the data to calculates the sizes of the subsets based on the total number of samples in the dataset. It uses array slicing to extract the corresponding subsets of X and y.

we use the train_test_split function from Scikit-Learn to split the data randomly into training and testing sets. This function takes care of shuffling the data if needed and ensures that the split is done in a randomized way, which can help prevent any biases that may arise from having the data sorted in a specific way. we train and evaluates two different Naive Bayes classifiers - Gaussian Naive Bayes and Multinomial Naive Bayes - using the scikit-learn library.

we create instances of the two classifiers and fits them to the training data using the fit method. Then, the code uses the predict method to make predictions on the test data, and computes the confusion matrix and accuracy score for each classifier using the confusion_matrix and accuracy_score functions.

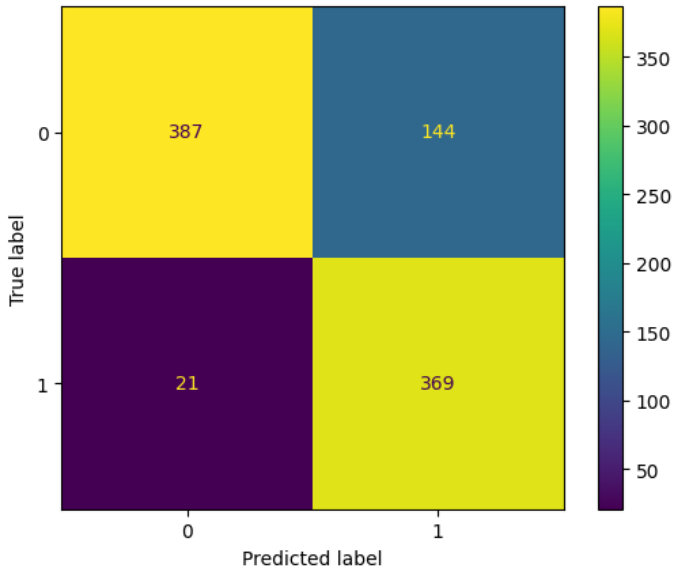
```
Confusion Matrix (Gaussian Naive Bayes):
 [[569 352]
 [  0   0]]
Accuracy (Gaussian Naive Bayes): 0.6178067318132465
Confusion Matrix (Multinomial Naive Bayes):
 [[666 255]
 [  0   0]]
Accuracy (Multinomial Naive Bayes): 0.7231270358306189
Classification Report (Bernoulli Naive Bayes):
              precision    recall  f1-score   support

           0       1.00      0.62      0.76       921
           1       0.00      0.00      0.00         0

    accuracy                           0.62       921
   macro avg       0.50      0.31      0.38       921
weighted avg       1.00      0.62      0.76       921
```

```
Classification Report (Bernoulli Naive Bayes):
              precision    recall  f1-score   support

           0       1.00      0.62      0.76       921
           1       0.00      0.00      0.00         0

    accuracy                           0.62       921
   macro avg       0.50      0.31      0.38       921
weighted avg       1.00      0.62      0.76       921
```

b) we use the train_test_split function . The test set size is set to 20% of the total dataset, and a random state of 42 is used to ensure reproducibility.
, creates instances of the two classifiers and fits them to the training data using the fit method. The predict method is used to make predictions on the test data, and the confusion_matrix and accuracy_score functions are used to compute the confusion matrix and accuracy score for each classifier.
The confusion matrix provides a detailed breakdown of the number of true positive, true negative, false positive, and false negative predictions made by the classifier. The accuracy score indicates the proportion of correctly classified instances among all instances in the test set, uses the ConfusionMatrixDisplay class from scikit-learn to plot the confusion matrix for each classifier as a heatmap.

```
Confusion Matrix (Gaussian Naive Bayes):
 [[387 144]
 [ 21 369]]
Accuracy (Gaussian Naive Bayes): 0.8208469055374593
Confusion Matrix (Multinomial Naive Bayes):
 [[445  86]
 [111 279]]
Accuracy (Multinomial Naive Bayes): 0.7861020629750272
Classification Report (Bernoulli Naive Bayes):
              precision    recall  f1-score   support

           0       0.95      0.73      0.82       531
           1       0.72      0.95      0.82       390

    accuracy                           0.82       921
   macro avg       0.83      0.84      0.82       921
weighted avg       0.85      0.82      0.82       921

Classification Report (Bernoulli Naive Bayes):
              precision    recall  f1-score   support

           0       0.95      0.73      0.82       531
           1       0.72      0.95      0.82       390

    accuracy                           0.82       921
   macro avg       0.83      0.84      0.82       921
weighted avg       0.85      0.82      0.82       921
```

```
Out[19]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x231d9e7faf0>
```

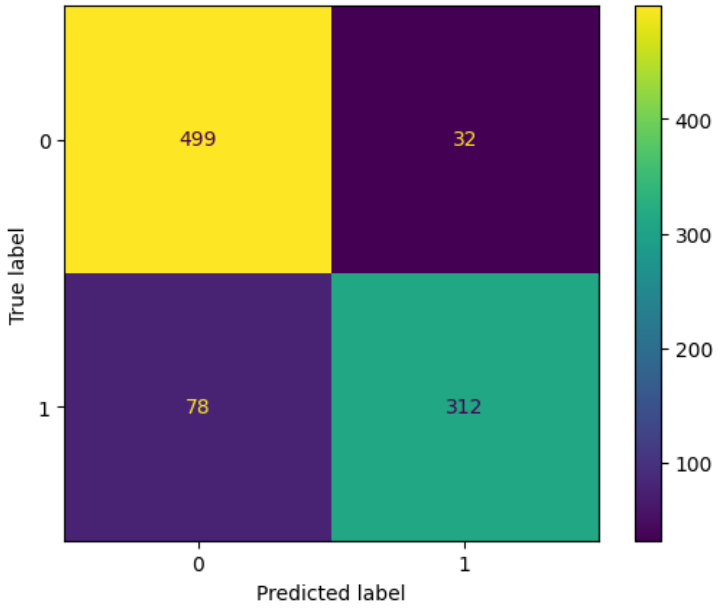


c) Use another Naive Bayes classifier

we use the ConfusionMatrixDisplay class to plot the confusion matrix for the Bernoulli Naive Bayes classifier as a heatmap.

```
Confusion Matrix (Bernoulli Naive Bayes):
 [[499  32]
 [ 78 312]]
Accuracy (Bernoulli Naive Bayes): 0.8805646036916395
Classification Report (Bernoulli Naive Bayes):
              precision    recall  f1-score   support

           0       0.86      0.94      0.90       531
           1       0.91      0.80      0.85       390

    accuracy                           0.88       921
   macro avg       0.89      0.87      0.88       921
weighted avg       0.88      0.88      0.88       921
```

```
Out[20]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x231d99e84c0>
```



d) train a classifier on multiple subsets of a training set and evaluate the accuracy of each subset.

compute the accuracy score for each subset using the accuracy_score function . It creates a bar chart using matplotlib to visualize the accuracy of each subset. The x-axis of the chart shows the names of each subset, and the y-axis shows the accuracy score.