Walmart sponsored a Kaggle competition to predict shopping trip type (https://www.kaggle.com/c/walmart-recruiting-trip-type-classification). This problem challenges the participants to classify a set of items referred to as a market basket purchased by a customer. The classification system consists of a set of 38 classes or types. The class labels are referred to as *shopping trip type*, or *trip type*. Walmart provides no information on the characteristics of the types, presumably to protect commercial interests.

They have provided a training set comprised of 647,053 purchased items from 95,674 visits. This exercise involves using one categorical variable, the department description of each purchased item, to predict trip type. There are 69 departments and a customer's purchases are distributed across the departments. The question to be answered in this analysis is: how much predictive information about trip type can be extracted from the departments?

More information is available from Kaggle. Some guidance toward answering the question is provided below.

a. Get the data file from Kaggle. The first record contains the names of the variables.

b. The first segment of code should map each record of a purchase to a visit. It's best to build a dictionary in which the key is the visit number (second column) and the value is a list consisting of departments from which the items were purchased. For instance, the three records from visit 9 are

```
8,9,"Friday",1070080727,1,"IMPULSE MERCHANDISE",115
8,9,"Friday",3107,1,"PRODUCE",103
8,9,"Friday",4011,1,"PRODUCE",5501
```

The trip type is 8 and the first department from which an item was purchased from is IMPULSE MERCHANDISE.[8] The dictionary entry for visit 9 is

```
[3, ['IMPULSE MERCHANDISE', 'PRODUCE', 'PRODUCE']]
```

The trip type in our dictionary entry is 3, not 8, as it is in data file because we have relabeled trip type using the integers $0, 1, \ldots, 37$.
Iterate over the data file and build the visit dictionary by aggregating all of the purchases from a single trip. It's a good idea to relabel trip types as consecutive integers $0, 1, \ldots, 37$. If you relabel trip types, then you will need to build a dictionary in which the keys are the Walmart trip type codes and the values are the integer-valued labels. You can build both dictionaries as the script iterates over the data file.

c. Construct a dictionary, we'll call it `dataDict`, in which the keys are visits and values are lists of length $p = 69$ and contain the number of items purchased from each department. The lists correspond to the predictor vectors $\mathbf{x}_1, \ldots, \mathbf{x}_n$, where $n = 95{,}674$. To build `dataDict`, note that the keys are the same as for the visit dictionary. Iterate over the keys and for each entry in the visit dictionary, count the number of items from each department and store the count in a list. Save the list as the value associated with the key. You can count the number of times a particular department `d` occurred in a list `x` using the `count` function:

```
counts = [0]*p  # p is the number of departments.
for i,d in enumerate(depts):
    counts[i] += x.count(d)
```

The variable `depts` is the list of departments from which items were purchased in the course of the visit. Store the count list *and* the trip type as the value associated with visit.

[8] The UPS code 1070080727 is a barcode symbol that specifically identifies the item.

d. The frequency distributions of each trip type (similar to the `count` list) may be constructed using the same code structure. The difference is that the dictionary keys are the trips.

e. Build a dictionary that contains the log-relative frequencies of occurrence of each department for each trip (the $\log(\widehat{\pi}_{k,i})$'s where k indexes trip and i indexes department). Also compute the log-priors (the $\log(\widehat{\pi}_k)$'s).

f. Compute an estimate of the accuracy of the multinomial naïve Bayes prediction function. Iterate over `dataDict` and extract each test vector. Compute the sum of log-probabilities and determine the most-likely trip:

```
yhat = np.argmax(postProbList)
```

where the list `postProbList` contains the sums for each trip.

g. While iterating over `dataDict`, tabulate the prediction outcomes in a confusion matrix. The confusion matrix may be a `Numpy` array initialized according to

```
confusionMatrix = np.zeros(shape = (nTrips,nTrips))
```

If the visit in question has been labeled as trip `y0`, then increment the entry in the confusion matrix in row `y0` and column `yhat`. (Relabeling the trip classes as integers from 0 to $g-1$ has made it easy to build the confusion matrix in this manner.) While iterating, compute and print the overall estimate of accuracy:

```
acc = sum(np.diag(confusionMatrix))/sum(sum(confusionMatrix))
```

h. Report the estimated overall accuracy and the confusion matrix.