Task 1:

Which probability distributions would you use to model each of the three components: probability of category purchase $P(I_{ict}=1)$, the probability of product choice within a category $P(C_{it}=j \mid I_{ict}=1)$, and the probability for the purchased amount $P(Q_{ijt} \mid I_{ict}=1 \wedge C_{it}=j)$? In your opinion, what drives the category purchase incidence? What factors besides the three mentioned above could influence product choice?

1. Since there no assumptions regarding the categories, a possible probability distribution would be a discrete uniform distribution, $\mathrm{U}\{1,\mathrm{C}\}$ where C is the number of categories
2. For the product choice, and since only one product will be purchased, a possible distribution will be the categorical (multinomial) distribution, that describes the possible results of a random variable that can take on one of *K* possible products, with the probability of each product,i, separately specified, i.e, $\mathrm{p}_i > 0, \sum_i^K p_i = 1$
3. The probability of the quantities purchased will be a **Gaussian** distribution for each customer with its parameters, the mean and variance, being calculated based on the customer purchasing history.
4. In addition to the three factors that influence the product choice, one could think of: 1) being on sale, 2) history of product purchase, 3) if other products purchased (cereal needs milk), and 4) day of purchase (weekdays vs weekends).

Task 2:

- The task is implemented in Python. Imported modules: keras (tensoflow as backend), sklearn.metrics (roc_curve, auc), and json
- In the dataset, it is clear that the price for each product is fixed, independent of the week, so the price is removed all together.
- Also, if a product is advertised in one week, it is advertised for ALL customers, hence, the input at every timestep can only be a 40x1 boolean vector, with an entry for every product, instead of a 2000x40 matrix for every product-customer combination.
- The first 40 steps (out of 49) is used for training, and the rest was used for testing (the variable PredictionStep was used to implement this).
- A recurrent architecture is implemented. It is structured as:
  40 inputs, 200 LSTM nodes layer, and 2000x40 dense (sigmoidal) output layer. A 50% dropout is used for the recurrent layer to improve generalization. A window size of 20 was used for the input sequence.
  It should be noted that these hyperparameters were chosen empirically and there was no attempt to find out the optimal values.
- The minimized cost function is the mean squared error and an “adam” training protocol was used
- In the file, promotion_schedule.csv, the data for product j = 20 was missing, and thus had to be hard coded
- The resulting ROC for all products was equal to 1.00 (attached as “roc.txt”)
- The results for the promotion_schedule.csv has been generated into SO1Predictions.csv (values less than .00000000001 was set to zero).
- Attached files are:
  1. so1_task.py
  2. SO1Predictions.csv
  3. roc.txt