GLOBAL ACADEMY OF MATHEMATICAL AND ECONOMIC SCIENCES

March 2022

GUIDE TO PORTFOLIO PROJECT

Probabilistic Sales Forecasting using Discrete Distribution Models

Project Creator/Supervisor: Zion Pibowei

Course: Probabilistic Modelling

This guide is a run-through of the approach and techniques expected in fulfilling the technical objectives of the portfolio project.

Here is a glance at the first 5 rows of the data. The columns City, Customer type, Gender, gross margin percentage, gross income, and Rating have been removed for the sake of brevity.

Branch	Product line	Unit price	Qty	Tax 5%	Total	Date	Time	Payment	cogs
A	Health and beauty	74.69	7	26.1415	548.9715	1/5/2019	13:08	Ewallet	522.83
C	Electronic accessories	15.28	5	3.8200	80.2200	3/8/2019	10:29	Cash	76.40
A	Home and lifestyle	46.33	7	16.2155	340.5255	3/3/2019	13:23	Credit card	324.31
A	Health and beauty	58.22	8	23.2880	489.0480	1/27/2019	20:33	Ewallet	465.76
A	Sports and travel	86.31	7	30.2085	634.3785	2/8/2019	10:37	Ewallet	604.17

The first thing you should do is convert the Date field to a datetime data type after replacing "2019" with "2022". Yes, we want to treat the data as historical records for Q1 2022 and make forecasts for Q2.

Now, let's see how to approach the objectives, one by one:

1. Binomial Experiments:

(a) Use a binomial experiment to simulate the number of times sales will be recorded in Branch A in Q2 daily. Repeat the simulation for branches B and C, and aggregate the results.

Notes:

A binomial experiment is a sequence of trials where each trial has 2 possible outcomes and can only result in one of them. The results of a binomial experiment follow the binomial probability distribution, & the probability of obtaining any of the outcomes of the binomial distribution is given by the binomial probability mass function (PMF) as follows:

$$P(X = k) = \binom{n}{k} p^k (1-p)^{n-k} \quad (1)$$

Where X is the random variable of interest, k is a specific value of X at an instance, n is the fixed number of trials in the experiment and p is the probability of realising each successful outcome. This probability is known or calculated beforehand.

From the objective, we are interested in predicting successful outcomes, so we need to simulate a binomial distribution $X \sim B(n,p)$, then generate a draw (random outcome k) from that distribution. To simulate a binomial distribution in Python, use numpy's random module and run binomial(n,p) which simulates the $B(n,p)$ distribution and returns a random outcome. Since objective 1(a) is focused on simulating number of times sales were recorded each day, rather than number of quantity sold, the number of trials n will be total number of times sales were made each day. So we are running different simulations for each of the days. To get random draws from the binomial distribution for each day, here's a possible algorithm:

Algorithm 1.1

- Get array of total number of times sales were recorded each day:
  $N = \{n_i \text{ for } i \in D\}$
- Get array of daily probabilities of success for branch A:
  $P(A) = \left[\frac{n(A_i)}{n_i} \text{for } i \in D\right]$
- Run element-wise binomial simulation for elements in $N \times P(A)$:
  $[B(n,p) \text{ for } (n,p) \in N \times P(A)]$

This results in an array containing outcomes simulated for each day. Daily results will vary for each implementation of the algorithm. To make results a bit more stable you can generate multiple draws (e.g., $M = 1000$) for each day and get the daily mean. This requires a slight change to step 3 in Algorithm 1.1, as follows:

$$\left[\frac{1}{M}\sum B(n,p,M) \text{ for } (n,p) \in N \times P(A)\right]$$

A binomial distribution doesn't take account of time, you can only simulate possible outcomes that can occur but not for a specific time in the future. A good way to make realistic projections for each day in Q2 *could be* to combine the simulation over the entire data with the simulation for that specific day. This brings us to Algorithm 1.2 as follows:

Algorithm 1.2

- Compute the following prior information for entire data:
  - Total number of times n_E sales were recorded throughout entire sales history
  - Total number of times n_A sales were recorded in Branch A throughout history
  - Probability of success for Branch A for the entire history: $p_A = n_A/n_E$
- Compute prior information for each day:
  - Array of total number of times sales were recorded each day:

    $N = [n_i \text{ for } i \in D]$
  - Array of daily probabilities of success for branch A:

    $P(A) = \left[\frac{n(A_i)}{n_i} \text{for } i \in D\right]$
- For each element in $N \times P(A)$, generate a draw from binomial simulation for that instance, add it to the outcome from binomial simulation of entire data, & subtract historical number of sales events in branch A:

  $\left[B\left(n_E, p_E\right) + B(n,p) - n_A \text{ for } (n,p) \in N \times P(A)\right]$

Here's the intuition behind the algorithm: Say there are 90 days in the data and we want to simulate for the 91st day (and given stationarity of the time series, we expect the 91st day to be similar to the 1st). We can simulate over 90 days, add it to a single day simulation based on Day 1, so that the resulting outcome is the total for 91 days. Then we subtract the total historical number of outcomes for 90 days. This leaves us with the outcome for 1 day, namely the 91st day of interest. We can stabilise the results by taking the mean of multiple draws from the 90-day simulation, using $\frac{1}{M}\sum B\left(n_E, p_E, M\right)$, and round off to the next integer since outcomes must be discrete.

Objective 1(b): Simulate the quantity of products that will be sold in each of the branches in Q2 daily.

This is quite similar to objective a), only difference being that you are to take account of each quantity sold and where they were sold, rather than just the number of times bulk sales were recorded.

2. Multinomial Modelling:

(a) Multinomial Simulation

The multinomial distribution is a generalisation of the binomial distribution for a discrete variable with K outcomes. Rather than simulating for each branch one at a time to check whether or not a sale will occur in that branch (binomial), the multinomial experiment can treat all three branches simultaneously to predict where each event would occur.

For this experiment, the sequence is to first learn the probability distribution of quantity sold each day from the data, then simulate quantities that will be sold in the future, and finally assign branches where those sales would be made. This is a simple and direct solution that will not make use of conditional probabilities. We will adopt the estimation approach used in our binomial simulations as follows:

$$x_i = M(n,p)_D \ + \ M(n,p)_i \ - \ x_{Br} \tag{2}$$

where:

- x_i is the estimated result containing K outcomes corresponding to K branches for a specific day i,
- $M(n,p)_D$ is a draw from the multinomial distribution simulated for the entire data D,
- $M(n,p)_i$ is a draw from the multinomial distribution simulated for the specific day of interest i,
- x_{Br} is a sequence of the total outcomes from each of the branches of interest throughout the entire history of the data.

To simulate a multinomial distribution using numpy.random, run multinomial(n,p) and pass a sequence of probabilities for p. This simulates the $M(n,p)$ distribution and returns a random sequence of K outcomes.

Algorithm 2.1

- Compute the following prior information for entire data:
  - Total quantity of products q_E sold throughout entire sales history
  - Total quantities sold in each branch throughout history: q_A, q_B, q_C

  - Probability of success for each branch throughout history: p_A, p_B, p_C
- Compute prior information for each day:
  - Array of total quantities of products sold each day:

$$Q = \left[\sum_{j \in i} q_j \text{ for } i \in D \right]$$

  - Arrays of daily probabilities of success for each branch, A, B, C:

$$P_A = \left[\left(\frac{\sum q_{A_j}}{\sum q_j} \right)_{j \in i} \text{ for } i \in D \right]$$

  - Take Cartesian product of the 3 probability arrays to get compact probability array where each element is a 3-tuple representing a single day:

$$P = P_A \times P_B \times P_C$$

- For each element in the Cartesian product $Q \times P$, implement element-wise multinomial simulation according to the estimation approach in Equation (2):

$$[M(n_E, [p_A, p_B, p_C]) + M(i,j) - [n_A, n_B, n_C] \text{ for } (i,j) \in Q \times P]$$

  Again, you can stabilise the results using multiple simulations of the first term and taking the mean on the row axis:

$$Q_{pred} = \left[\frac{1}{m} \sum M(n_E, [p_A, p_B, p_C], m) + M(i,j) - [n_A, n_B, n_C] \text{ for } i,j \in Q \times P \right]$$

- Finally, retrieve maximum quantity from each daily 3-tuple result $\{q\}$, and using the index of the max, assign the branch label corresponding to that index.
  - $Q_{max} = \left[\mathbf{max} \{q\} \textbf{ for } \{q\} \in Q_{pred} \right]$
  - $Br_{max} = \left[Br_i \textbf{ for } i \in [3] \text{ such that } q_i == \mathbf{max} \{q\} \textbf{ for } \{q\} \in Q_{pred} \right]$

(2b) Multinomial Classification – Naïve Bayes

The Naive Bayes algorithm uses Bayes theorem to predict the conditional probability of one event given the occurrence of anther event. The model assumes conditional independence amongst input variables. For Objective (2b), we wish to estimate the

conditional probability of an event occurring in a branch Br given a specific quantity sold Q, without using the joint probability of these 2 outcomes:

$$P(Br|Q) = \frac{P(Q|Br) \times P(Br)}{P(Q)} \tag{3}$$

where:

- $P(Br|Q)$ is the ***posterior probability*** of any branch Br getting predicted given a specified or predicted quantity Q,
- $P(Br)$ is the ***prior probability*** of any branch Br occurring in the historical data,
- $P(Q)$, the ***evidence***, is the probability of a specific quantity Q sold daily, and,
- $P(Q|Br)$, the ***likelihood***, is the probability of a specific quantity sold given the branch Br.

The posterior probability given by Bayes' Theorem in Equation (3) is not what we are interested in, in itself. Instead we are interested in the branch that maximises this probability. That is, we are looking for:

$$\max_{Br \in Branch} P(Br|Q) \ = P(Q|Br) \times P(Br) \tag{4}$$

Notice the denominator $P(Q)$ has been dropped. Because $P(Q)$ is constant for all the calculations of posterior probability for each branch and so does not affect the results.

For now, you do not need to take account of other input variables but you can add them later on. So, we start with the case of just one univariate – Quantity.

Algorithm 2.2: Univariate Case

- Compute prior information from data required for binomial simulation
- Aggregate quantity sold by branch each day, so each day has 3 events
- Run binomial simulation to predict 3 outcomes of quantity each day
- Fit naïve Bayes model (Eq. 4) on the data, i.e., learn the conditional probability distribution $P(Q|Br)$ from the data. $P(Br)$ has been previously computed. To calculate $P(Q|Br)$ for each branch, the binomial PMF is required:

$$P(X = q_j) = \binom{q_i}{q_j} p^{q_j}(1-p)^{q_i - q_j}$$

  - q_j is the specific quantity to be classified which can be taken from the simulated data
  - q_i is the total quantity for a single day in historical data
  - p is the probability of success for the branch of interest for that day in the data
- Make probabilistic predictions
  - Compute $P(Br|Q)$ for each branch using the results for $P(Q|Br)$.
  - Output maximum probability across the branches as the predicted probability.
  - Output branch corresponding to maximum probability as the predicted class.