JP Morgan Quant Challenge, 2018
Explanation Document

Name: Gurpreet Singh
Team: $$$$
Date: September 30, 2018

QUESTION 1

1.1. Method

I have simply used Multi-Layer Perceptron to predict the ticket rate for each customer. This is implemented using Scikit-Learn library for python.

1.2. Features

For each customer, features are computed using his/her DOB, Flight Time, Flight Date, Booking Date, Place of Departure, Place of Arrival and Ticket Class (Business/Economy).

1. From DOB, the customer's age is extracted, based on the intuition that the age of a customer might affect the price he/she is willing to pay for a flight.
2. As Flight Times affect the rate of a flight, the time is used to compute the difference of minutes from 0:00, i.e. for a flight at 0:20, the feature value would be (0x60 + 20 = 20).
3. The Flight Date, Flight Time and Flight Day are used together to compute the number of days before the flight's actual departure has the customer booked the ticket, as the ticket rate often increases towards the departure date.
4. Since Ticket Class would obviously affect the rate of ticket, a binary variable is computed which, if one, means that the ticket is a business class ticket.
5. The place of arrival and place of departure would decide the distance travelled by a flight, and therefore would affect the ticket charge. Since the distances are not directly available, I compute 21 features (as there are 7 places and therefore 7c2 = 21 possible tuples, ignoring the order), each binary and representing whether a flight is between the places denoted by the respective tuple/feature.

1.3. Model

Using these features, I compute the feature set, and train it against the actual fares a customer is willing to give.

Upon observing the performance on training data, decision trees worked the best, but seemed to overfit by a large amount (Training R2   .99, Test R2   0.60). Based on this, I decided to use Neural Networks (I did not linear or polynomial methods of regression).

Another observation was the better performance if two separate networks are trained for different customers based on different ticket classes. That is, the tickets/customers are divided into two classes, one for economy and another for business. For these two sets of features and labels, separate and independent networks are trained. Using this method gives a marginal improvement over a single network. The sizes of the networks that seemed to work the best were (50, 20, 5) for both the networks (multiple different combinations were tried).

1.4. Code

The code is divided into three parts -

1.4.1. Reading the Data

This simply involved reading the data using the urls for the training and testing data, and splitting each lime into different fields.

This is handled using the function read_data (string url, string split_string [default:',''], boolean test [default: False])

1.4.2. Processing Features

This step essentially computes the features set, based on the features described in the previous section.

This is handled using the function process_features(2D-array X, dictionary tuple_cities)

Note: The tuple_cities attribute is computed beforehand, which simply computes an index for each tuple (of departure and arrival cities) and stores equal index for both orders of the cities in any tuple.

1.4.3. Regressing

This step fits the training data and predicts the test labels, and puts the predicted output into the clipboard (for my ease of testing).

1.4.4. Required Libraries

For running the code, one needs to have the following python libraries installed

1. numpy
2. datetime
3. scikit-learn
4. pyperclip (for copying to clipboard)
5. urllib2

JP Morgan Quant Challenge, 2018
Explanation Document

Name: Gurpreet Singh
Team: $$$$
Date: September 30, 2018

QUESTION 2

We are given a continous random process X. In order to model this process, we break the continous time interval into discrete time steps, as given in the hint. The number of steps, again as suggested by the hint, is taken to be 100. Therefore, the state of the random process X is computed at 100 discrete points, uniformly spread over the interval $[0, T]$ where $T = 2$.

Using the forementioned method, we can obtain an estimate of X_T. However, to compute the expectation, we use Monte Carlo method, and therefore sample multiple values of X_T and average over them to compute the expectations required by the question.

To do this, 10000 estimates (again suggested in the hints) of X_T were generated using the process mentioned earlier, and these estimates are used to compute a Monte Carlo estimation of the expectations as follows

1. $\mathbb{E}[X_T] = \sum_{i=1}^{10000} X_T^{(i)}$
2. $\mathbb{E}[\max\{X_T - 100, 0\}] = \sum_{i=1}^{10000} \max\left\{X_T^{(i)} - 100, 0\right\}$

For the third part, only using Monte Carlo would not give us a solution, as there is a differential. In order to compute the partial derivative, I have used central difference technique, wherein, I compute the expectation (term inside the partial differntial function), for values at $\sigma + d\sigma$ and at $\sigma - d\sigma$, and approximate the differential as

$$\frac{\delta(\mathbb{E}[\max\{X_T - 100, 0\}])}{\delta(\sigma)} = \frac{\mathbb{E}[\max\{X_T(\sigma + d\sigma) - 100, 0\}] - \mathbb{E}[\max\{X_T(\sigma - d\sigma) - 100, 0\}]}{2\ d\sigma}$$

Using these techniques, I have estimated all three quantities as required by the question.

2.1. Code

For each path, I compute three estimates of X_T corresponding to σ, $\sigma + d\sigma$ and $\sigma - d\sigma$. All these estimates are computed using the same samples of dW in order to save time (the expectations remain unbiased even after this, and therefore still provide good estimations of the quantities required).

The random normal sampling is done using Numpy's random sampling function, and all the random values required for the next 100 steps are obtained simultaneously, to reduce the number of calls to numpy's functions and therefore save time.

Using these 100 random samples, dX is computed at each time step, and X is updated as $X = X + dX$, where X is the current state of the random process given. After 100 steps, we obtain $X_T = X$ as the estimate for the current path. The same procedure is repeated for $\sigma + d\sigma$ and $\sigma - d\sigma$, using the same random normal samples.

This is then repeated for 10000 samples, and all three values corresponding to σ, $\sigma + d\sigma$ and $\sigma - d\sigma$ are saved in an array/.

Post sampling, the 10000 samples are used as in the equations given previously to compute the expectations and derrivative.

JP Morgan Quant Challenge, 2018
Explanation Document

QUESTION 3

Name: Gurpreet Singh
Team: $$$$
Date: September 30, 2018

This question is similar to the second question in terms of discretizing the time interval and using Monte Carlo for computing expectations. However, in this case, we need to use the historical data in order to compute parameters of the random process. These parameters are σ_X, σ_Y and ρ.

As pointed out by the question, σ_X and σ_Y are computed as the expected log difference between consecutive time steps. This is estimated using the SPOT Data for X and Y, that is, by using the spot samples to compute a Monte Carlo esitmate of the σ_X and σ_Y. The quantity ρ is also computed similarly, by assuming the spot samples to be tuples of X and Y.

Similar to the second quesiton, we are given two continous random processes X and Y. In order to model these processes, we break the continous time interval into discrete time steps, as given in the hint. The number of steps, again as suggested by the hint, is taken to be 252. Therefore, the state of the random processes X and Y are computed at 252 discrete points, uniformly spread over the interval $[0, T]$ where, in this case, $T = 252 * N$.

Using the forementioned method, we can obtain one estimate of each X_T and Y_N. However, to compute the expectation, we use Monte Carlo method, and therefore sample multiple values of X_T and Y_N and average over them to compute the expectations required by the question.

To do this, 10000 estimates (again suggested in the hints) of X_T and Y_N were generated using the process mentioned earlier, and these estimates are used to compute a Monte Carlo estimation of the payoff as follows

1. $\mathbb{E}\left[\max\left\{\frac{1}{2}\frac{X_T}{X_0} + \frac{1}{2}\frac{Y_T}{Y_0} - 1, 0\right\}\right] = \sum_{i=1}^{10000} \mathbb{I}\left[accept(i)\right] \max\left\{\frac{1}{2}\frac{X_T^{(i)}}{X_0} + \frac{1}{2}\frac{Y_T^{(i)}}{Y_0} - 1, 0\right\}$

The function accept(i) is used to satisfy the constraint mentioned in the question, that for all $k = 0 \cdots N$, $X_t \in [0.75X_0, 1.25X_0]$ and $Y_t \in [0.75Y_0, 1.25Y_0]$ where $t = t_0 + 0.25 * k * 252$ days

3.1. Code

Using the historical data, first the values of $sigma_X$, $sigma_Y$ and rho are computed, using the formulas provided in the question.

For each path, estimates of X_T and Y_T are computed. These estimates are computed using samples of $\boldsymbol{\epsilon}$, which is distributed according to a bivariate normal distribution with correlation rho. The random normal sampling is done using Numpy's random multivariate_normal sampling function, and all the random values required for the next 100 steps are obtained simultaneously, to reduce the number of calls to numpy's functions and therefore save time.

Using these 100 random samples, dX and dY are computed at each time step, and X is updated as $X = X + dX$, where X is the current state of the random process given (likewise for Y). After 100 steps, we obtain $X_T = X$ and $Y_T = Y$ as the estimates for the current path.

This is then repeated for 10000 samples, and all paths satisfy the acceptance criteria are saved in an array. Post sampling, the 10000 samples are used as in the equations given previously to compute the expectation for the payoff.