01. Creating a test bed environment

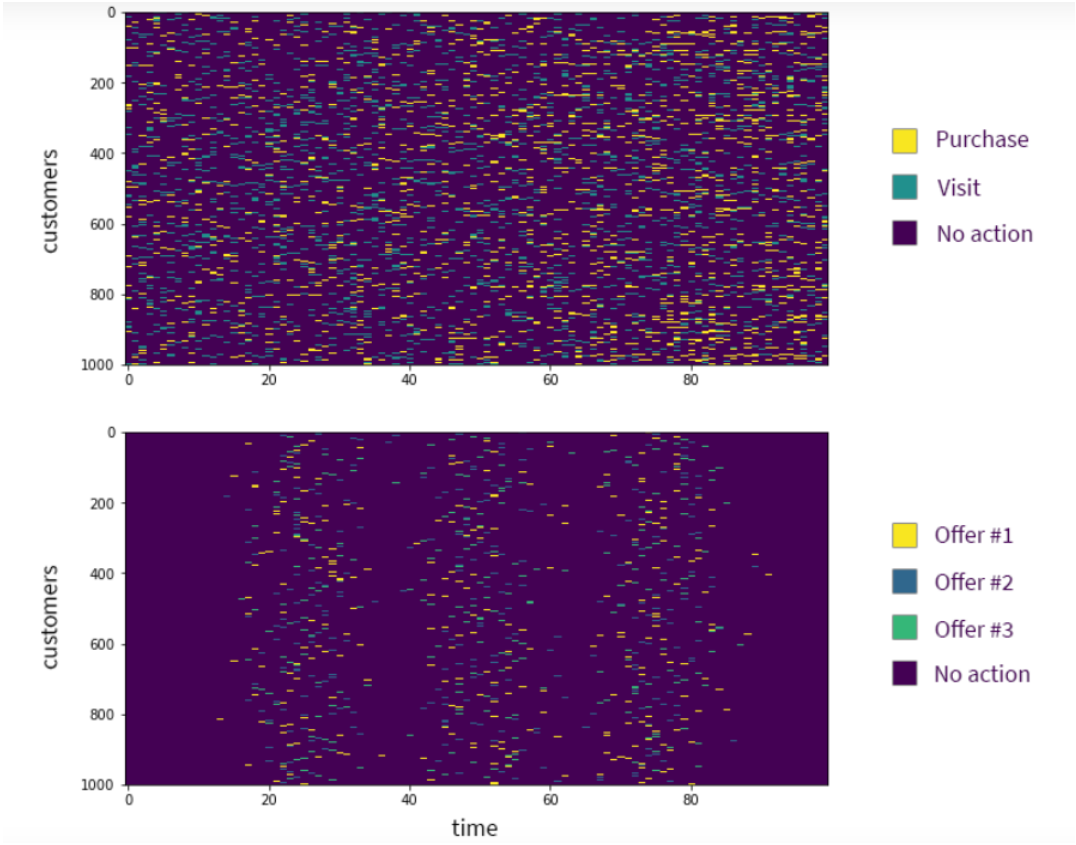
- There are three different events (**No action, Visit, Purchase**)
- Three different offers (**Advertisement, Small discount, Large discount**)
- Two different demographic classes (**Young person, Old person**)

```
events = [
    0, # no action
    1, # visit
    2  # purchase
]
offers = [
    1, # advertisement
    2, # small discount
    3  # large discount
]
demographic = [
    0, # young customers
    1  # old customers
]
```

- 1000 number of customer profiles are generated with 100 time samples and issue three offers at randomized points of time.
- **Customer behavior (event probabilities) changes over time as a function of previously received offers.**
- Customer profiles are generated according to the following condition,
  Probability of a purchase is boosted if and only if offer #1 (advertiesment) is followed by the offer #3 (large discount)

- For the sake of simplicity, we ignore demographic features when choosing the event probabilities. **But the design of the model we develop is fully capable of learning these dependencies as well.**

- Next, we can visualize customer profiles at 100 time points (1000 customers)



- We can see the frequency of purchases increase over time (the density of yellow dashes). Because some customers get the right sequence of offers that triggers an increase of purchase probability.

02. **Preparing trajectories for each customer**

- Next step is to learn offer targeting policy by using **Fitted Q Iteration (FQI)**
- FQI works with individual transitions
- Need to cut the trajectories into pieces as follows.

$$T = \{ (s_{u1}, a_{u1}, r_{u1}, s_{u2}),\ (s_{u2}, a_{u2}, r_{u2}, s_{u3}),\ (s_{u3}, a_{u3}, r_{u3}, s_{u4}) \}$$

- Each customer trajectory is consisted with a set of transition tuples
- **Number of purchases in each transition is called as reward**

- **state_features** function return state value for given customer in a time range
  state = [demographic_feature,
  number_of_vist_from_0_time_point_to_end_of_time_frame,
  first_time_point_offer_1_is_issued,
  first_time_point_offer_2_is_issued,
  first_time_point_offer_3_is_issued,
  ]

- Then trajectories are generated for 1000 customers in 100 time points

- Now data is ready!!!

Note :-

transition = (current_state, action, reward, new_state)

03. **Policy learning**

- Once data is ready, we can apply the **FQI algorithm** to learn **action value function (Q function)**

- Here the random forest algorithm from scikit-learn is used as a subroutine for **Q function estimation**

- `RandomForestRegressor(max_depth,random_state,number_of_decision_trees)`

- **Action value function is trained at 5 times as fitted Q iteration**

- Q function format – Q([[current_state, offer_type]])

- Now we have trained Action value function (Q function)

04. **Action value function on test data**

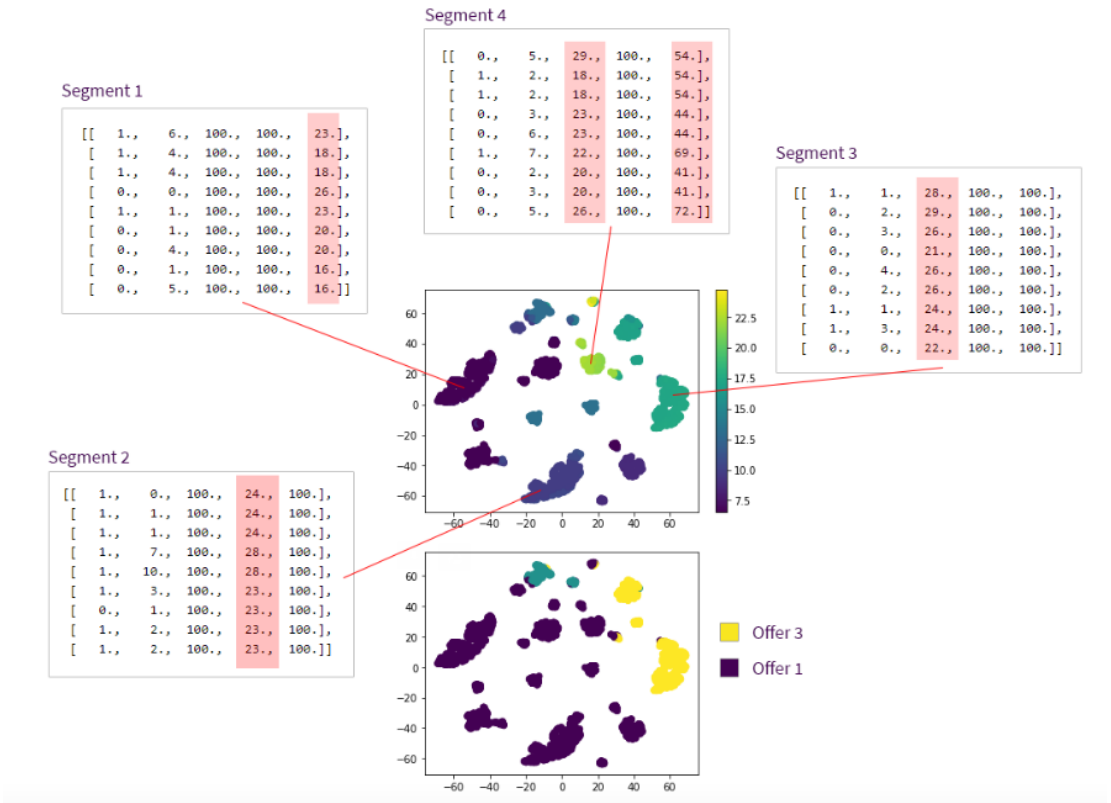
- Next, we can apply trained action value function (Q function) to test data set

- Visualization is done by using TSNE method. It is a higher dimensional data visualization tool. (Map to 2D field)

- Each status point is mapped into one point at TSNE 2D plane.

- Then two resulting values of maximum action value function are plotted in TSNE 2D plane. (Best action value, Best action in given three offers)

- Following shows two TSNE plots for best action value and best action.



- Two plots are identical except the color coding.

- We can see several clusters of customer states with different expected values and its recommended actions

- Each state of customer is a represented by a vector

  (demographic feature, number of visits, issue times of each three offers)

  <u>Explanation of above two plots</u>

- **Segment 1**. These are the customers who got offer #3 in the beginning of the trajectory. Our generative model boosts the purchase probability only if offer #1 precedes offer #3, so these customers are lost causes with minimum value.

- **Segment 2**. These customers got offer #2 first. The model rightly recommends to give them offer #1 next, but the predicted value for this segment is relatively low because the remaining time until the end of the trajectory is limited.
- **Segment 3**. These customers got offer #1 first, and the model recommends to give them offer #3 next. The expected value is relatively high because these customers are on the right track.
- **Segment 4**. These customers already got offer #1 followed by offer #3, and it boosted their purchase rates. They have the highest expected values. The system recommends providing them offer #1 as a default action, but the recommended action does not make any difference because the right offer combination has already been unlocked by that time.

05. **Evaluating next best action policy**