**Problem 2**:  
The problem is more into prescriptive analytics and recommendation system. If I was in this condition I will take a step by step approach to make the pilot batch of recommendation successful.  
I will start with minimal time approach to very data driven solution at the final step.  
  
Step 1:   
**Rule based approach/ Naïve approach:**I will try to give a score to client based on his/her previous viewing. This can be based on the location, type of house, the preferred budget, and the date before which they would like to close the deal, the click rate and so on.  
I will also have the real estate agents information with me with same features how well they were able to close deals of certain budget and location deals. Assigning the real estate agents to the clients in naïve manner.  
While doing this we can try first on a small population and check if that was successful with a A/B testing if our Rule based approach worked.  
  
Drawbacks:   
- Highly sparse data would be difficult to correlate and provide a solution.  
- Human Bias could be a huge problem here   
- Big data is very difficult to manage  
  
  
Step 2:   
**Cluster and Assign**:  
We can initially cluster all the real estate listing in the websites using some unsupervised algorithm it can be KNN , K-Means or anything for that matter.   
We can then cluster with the same information in agent’s historical deals.  
After this we can find the closest (distance can be from centroid to the client or it can be from the cluster centroid of agent to cluster centroid of clients) cluster between clusters of agents and cluster which the client belong to.  
This will be able to mostly give us the right agent assigned to the right client.

Drawbacks:   
- There might be many outlier clients which does not actually belong to a cluster, this must be done right by the manual method.  
- We might not end up using all the features provided to us as we are looking for similar features between real estate agents and clients. (we find the distance only if the dimensions of both the data set are similar)

Step 3:   
**Rule based approach/ Naïve approach:**I will try to give a score to client based on his/her previous viewing. This can be based on the location, type of house, the preferred budget, and the date before which they would like to close the deal, the click rate and so on.  
I will also have the real estate agents information with me with same features how well they were able to close deals of certain budget and location deals. Assigning the real estate agents to the clients in naïve manner.  
While doing this we can try first on a small population and check if that was successful with a A/B testing if our Rule based approach worked.  
  
Drawbacks:   
- Highly sparse data would be difficult to correlate and provide a solution.  
- Human Bias could be a huge problem here   
- Big data is very difficult to manage

Step 4:   
**Machine learning approach and assignment solution:**I will create few groups of customers like(should be created with the business data in hand this is just an assumption , to do this grouping also we have to sort out to ML algorithms) :  
- Immediate buyers  
- prospective future buyers  
- Initial login buyers   
- long term customers   
- high profile customers   
and so on   
We can find the real estate agents previously who performed best in each of this group of buyers in their past experiences   
- group 1 ( this group of real estate agents might be good at high profile and long term customers )  
- group 2( this group of real estate agents might be good at Initial login buyers and long term customers )  
- group 3( this group of real estate agents might be good at high profile and prospective future buyers)  
  
We can give a weights to each buyer (with the knowing who is having higher capability to convert into actual buyer). With all this information we can create an optimization algorithm to fulfil all the clients with the available agents in hand. (We can build rules like an agent should not have more than 6 clients, group 1 agents should have at least 3 high profile and 2 long term customer based on their historical success).This is completely automated way of assigning clients to the best agents.  
To monitor it we can use metrics like conversion rate (% growth), customer feedback (can be analysed using NLP techniques) and so on. If we get a negative feedback the real estate agents group would be shifted and we can re make the whole process with data driven solution.

Drawbacks:   
- Huge technical development time, maintain the model to predict the groups in regular intervals.  
- model should be challenged with multiple features to remain the state of art as a problem can collapse the groups .

**Problem 3**:

**1. How do you setup local python dev environment?**

* What IDE do you use? Visual studio and Jupyter Lab
* How do you setup python production environment in Linux?
  + List the cli commands if possible.
* check for updates in package – sudo apt update
* sudo apt install python3 python3-pip
* python3 -m venv house\_sigma
* source house\_sigma /bin/activate
* pip install numpy pandas … ( all packages needed )
* pip install -r requirements.txt ( in case we have to duplicate already existing virtual env )
* python app.py ( application that we run)
* open the browser local host and see the web page created for the model

### 2. Are you familiar using any linux distro?

I am familiar with SSH

### 3. Setup a RESTful API with python & nginx

**Python code :**  
from flask import Flask,jsonify,request,make\_response

import time

from collections import OrderedDict

app = Flask(\_\_name\_\_)

@app.errorhandler(404)

def resource\_not\_found(e):

return "(404)"

@app.route('/timestamp')

def hello\_world():

current\_timestamp = int(time.time())

return jsonify({'timestamp': current\_timestamp})

@app.route('/readdata', methods=['POST'])

def handle\_json():

content\_type = request.headers.get('Content-Type')

if (content\_type == 'application/json'):

json = request.get\_json()

response ={

'username':json.get('username'),

'password':json.get('password')

}

return make\_response(jsonify(response))

else:

return 'Content-Type not supported!'

if \_\_name\_\_ == '\_\_main\_\_':

app.run(host='127.0.0.1')  
 **Results:**(base)  
Hariharavarshan@LAPTOP-CSTQVURR MINGW64 /d/DataScience/House sigma

$ curl 127.0.0.1:5000/timestamp

% Total % Received % Xferd Average Speed Time Time Time Current

Dload Upload Total Spent Left Speed

100 25 100 25 0 0 10317 0 --:--:-- --:--:-- --:--:-- 25000{"timestamp":1686895271}  
  
(base)  
Hariharavarshan@LAPTOP-CSTQVURR MINGW64 /d/DataScience/House sigma

$ curl --header "Content-Type: application/json" --request POST --data '{"username":"abc","password":"xyz"}' http://127.0.0.1:5000/readdata

% Total % Received % Xferd Average Speed Time Time Time Current

Dload Upload Total Spent Left Speed

100 71 100 36 100 35 12072 11737 --:--:-- --:--:-- --:--:-- 35500{"password":"xyz","username":"abc"}

(base)

Hariharavarshan@LAPTOP-CSTQVURR MINGW64 /d/DataScience/House sigma

$ curl 127.0.0.1:5000/noexist

% Total % Received % Xferd Average Speed Time Time Time Current

Dload Upload Total Spent Left Speed

100 5 100 5 0 0 1389 0 --:--:-- --:--:-- --:--:-- 2500(404)