Machine Learning: Please choose and solve TWO questions below

Question A1:

- Assume that you have a communication graph of large number of users u. Denote all users by set U
- Interaction at time t between two users u1, u2 is represented by r12(t)=r(u1,u2,t)
- For each user u, you have information such as
  - The main location of that user (e.g. district)
  - The time series of phone balance and top up
- For interaction r12(t)=r(u1,u2,t)
  - [A01] If users interactions are via phone calls, there is information such as call time, call duration, rough locations for both two users. The content of the phone call is not available though.
  - [A02] In the case of Facebook data, there is information such as the public message user u1 post in u2's home, message time, number of likes, etc.
- There are some labeled data where the income of users in set P is observed.
  - Note that set P is much smaller than set U. P is a subet of U.
  - Income of user u1 at time is recorded and denoted by q(u1,t)
    - Assume each distinct user only has one income observation at a specific timeP

a) Please consider the case [A01] and derive the features which are important to predict user income
b) Please do the same thing for the case [A02]

Question A2:

- Please read Ping Li, Trevor Hastie, and Kenneth Church, Very sparse random projections (http://ww.web.stanford.edu/~hastie/Papers/Ping/KDD06_rp.pdf)

a) Lemmas 1 to 3 summarize the main results and contributions of this work. Please explain these results in words. How does this work extend what is known from previous work by Achlioptas on constructing Johnson-Lindenstrauss embeddings? What are the pros and cons of using big values of parameter s in very sparse random projection methods?
b) Please consider the following scenario. You have a location summary data frame of a large number of users, denoted df01. Denote all users by set U. This data frame summarizes which site, at a specific time, each user appears at. The schema of the data frame is as follow

    user_id: long,
    date_time: string (format YYYY-MM-dd:hh-mm-ss)
    site_id: long

   Denote the set of all site ids S. Let A denote large matrix of size |U|x|S|, where components A(u,s) denote the number of times user u appears at site s. Consider projecting A to a lower dimension |U|xd (d << |S|), where he resulting matrix B can be used as machine-generated location-related features for users in set U.

- Please explain why we might want to consider using very sparse random projections for this embedding. What parameter s you may want to use?
- Write pseudocode to transform df01 to df02 with schema as follow.

    user_id: long
    ft_01: float
    ft_02: float
    ...
    --
    ft_20: float

  where values df02(u,ft_i) (i = 1, 2,..., 20) are projected values of matrix A on space |U| x20, using very sparse random projections.

Question A3:

a) What are the fundamental differences between mean and median? Consider the mean and median of housing prices in District 4, Ho Chi Minh City. Among the two values (mean and median), which do you think would be higher and why?
b) Assume you are measuring the temperature of a laboratory room. There is a special thermometer installed in the room, which logs data to a database with hundreds of data

points every second. Assume that you store the temperature data in a data frame df01, whose schema is as follow.

date: string (format YYYY-MM-dd)
time: string (format hh-mm-ss)
temperature: integer (measured in Celsius)

Assume, also, that you know the range of all values in the temperature column a priori (because, for example, you know the thermometer can only withstand a certain range of temperatures.) Suppose the range is [-1000,1000].

The goal is to find the median temperature of the room each day. Suppose that for each date, there is a large number of data points that you can't sort the temperatures. Your task [T] in this exercise is to transform df01 into df02 with the following schema

date: string (format YYYY-MM-dd)
median_temperature: integer (measured in Celsius)

where df02 indicates the median temperature of the lab room each day.

- Write pseudocode to perform [T]
- Write pseudocode to perform [T] considering that the schema of df01 is as follow.

date: string (format YYYY-MM-dd)
time: string (format hh-mm-ss)
temperature: float(measured in Celsius)

Big Data coding: Please choose and solve from one to two questions below

Question B1:

- Assume that we have the activation/deactivation data of telco users as below:

phone_number: long
activation_date: format YYYY-MM-dd
deactivation_date: format YYYY-MM-dd

- Example data:

```
0987000001,2016-03-01,2016-05-01
0987000002,2016-02-01,2016-03-01
0987000001,2016-01-01,2016-03-01
0987000001,2016-12-01,
0987000002,2016-03-01,2016-05-01
0987000003,2016-01-01,2016-01-10
0987000001,2016-09-01,2016-12-01
0987000002,2016-05-01,
0987000001,2016-06-01,2016-09-0
```

- As a user may switch between prepaid and postpaid or stop using phone_number for a while, he/she can have multiple activation/deactivation records. Also, there when a user A stop using a phone_number 1 for too long, telco can assign that phone_number 1 to a new user B.
- Example of user switch by the above example data:
  - The prepaid phone number 0987000001 was used by A from 2016-01-01 to 2016-03-01, then it was changed to postpaid. A continued using it until 2016-05-01 and stopped using this number. After 1 month, on 2016-06-01, this phone number was reused by B 1 with prepaid plan. B used it until, 2016-09-01 then changed to postpaid, and finally changed back to prepaid on, 2016-12-01 and he's still using it until now. In this case, the actual activation date of current owner B of 0987000001 that we want to find is 2016-06-01.
- Let's assume that if a user A stops using a phone_number for 7 days, then that phone number would be associated with a new user B.
- From the given data, we want to find a list of unique phone numbers together with the **actual activation date when its current owner started using it**. Note that what we need is the first activation date of **current owner B**, not previous owner A, and not the date when current owner changes prepaid/postpaid plans.

a) Please implement the evaluation of actual activation date by **Python (pandas, numpy)**

b) **[If the candidate is familiar with Pyspark]** Assume that you are dealing with big data (700 mil unique users), please implement the above problem by **Pyspark**

Question B2:

- Assume that you have two parquet data frames:
  - df01_graph: (from_number: long, to_number: long, number_calls: int)

- df02_feature: (to_number: long, f001: float, f002: float, ..., f999: float)

where df01_graph represents the connection/relation between two phone numbers (from_number and to_number).

- Some notes about the data:
  - number is always non-null and non-negative
  - any fxxx can be null: a phone number df01_graph.to_number can have a non-null f001 and null f002
  - df02_feature.to_number is the key (unique row)
  - set_difference(df01_graph.to_number, df02_feature.to_number) is not empty: unique(df02_feature.to_number) may not cover all unique(df01_graph.to_number)
  - set_difference(df02_feature.to_number, df01_graph.to_number) is not empty: unique(df02_feature.to_number) may cover more than unique(df01_graph.to_number)
- We want to propagate features from df02_feature.to_number to df01_graph.from_number, i.e. a user A may not have a specific information f001 but his/her friend may have

a) Please implement **Python (numpy, pandas)** code to do the propagation. The candidate should choose to to weigh the propagation of features from users B1, B2 to user A
b) If the candidate is familiar with **Pyspark**, please implement this problem in **Pyspark**
c) Assume that we are dealing with big data where the number of records/edges of df01_graph is about 1 billion and the number of users in df02_feature is about 500 millions. In that environment, Please **speculate some big data issues by yourself** and propose solutions to such issues (for both cases a and b)

Question B3: (Pyspark)

- Please describe how we handle/control memory for executors/drivers in spark environment.
- Please describe how you debug pyspark error (memory error, networking/connection error, data size)
- Please describe how you find/debug data issues (missing, NaN, duplicates)
- For the above question, please speculate the issues by yourself and propose solutions

Framework and design

Question C1:

- Assume that you already have big data system which consists several computing modules A01, ..., A10 already. These modules need to run at specific dates (daily, end of the week, end of the month – depending on the specific module). There is a dependency between these modules: for example, at the end of a week, daily module A01 needs to run first, then weekly module A02 runs, then A03, etc.
- We want to design several extra modules (SMLV)
  - Scheduler: scheduling the modules AXX according to the dependency and target dates. Re-run module AXX if needed (the error would be recorded by the monitor module below)
  - Monitor: After a module AXX completes its run, it would report the error code and the list of output hdfs parquet files. We want to monitor/record both the error code, time and evaluate some metrics on the output parquet files (for example, evaluate the size (Mb/Gb) of the output, the number of records of the output, the mean, standard devation, median of some columns of the output)
  - Logger: Log the above monitoring results
  - Visualization: As we keep the monitoring results over time, we would like to visualize these results over time so that we can see any temporal trend/error (input data error/mising at a specific data, temporal trend of metrics such as mean/standard deviation, run time error code(
- Please design the framework and the interface for modules SMLV above in **Python**. The candidate can also design the interface between modules AXX and SMLV. Assume that modules AXX have been implemented in Pyspark
- Requirements
  - Implement the design in Python (no bash script, java, etc.). The candidate may not need to do full coding Just the interface/signature is enough
  - Easy to add additional computing modules to the system (Requirement R01: the user can modify the configuration of SMLV modules to handle this change – **without modifying SMLV code**)

- Easy to add arbitrary metrics or metric python code on the monitoring of output list (the user can add new python metric code to handle this change – **without modifying SMLV code**)
- Even if we need to modify a module of SMLV, for example, the logger module (to allow logging to different storage), we don't need to modify other modules to accommodate such a modification (of logger)
- The candidate is allowed to use any pure python package (conda/pip) and pyspark. However, the candidate is not allowed to use complicated big data modules such as Hue/Ambari – as we want to avoid the dependency on such complex modules and do it as generalized as possible)

Additional requirements

- The candidate must be very familiar with Ubuntu system and Python
- Experience with big data open source software is a plus
- Can install/deploy/fix/monitor Ubuntu/big data/python system quickly
- Data Crawling: The candidate is expected to be able to do data crawling on web.