096224: Distributed Database Management

Background

In this project you are called to help the Lukewarm^{TM} cable company in recommending channels to new clients.

After a client signs up for the cable / TV services, she will get a recommendation of 10 channels that she might like. **Your job is to help the Lukewarm^{TM} company figure out which stations are the best fit for her - without any prior knowledge on her viewing habits.**

You are requested to preform several tasks in order to achieve this goal. Please read all the instructions before starting your work, for your best interest.

Data

The data in this project is a variation of the data used in the previous project.

Program Viewing Data

A variation of the viewing data you saw in project 1.

- device_id - Globally unique device identifier.
- event_date - Date viewing event occurred (Local) YYYYMMDD.
- event_time - Time viewing event occurred (Local) HHMMSS.
- station_num - Station ID number
- prog_code - Unique program identifier.
- household_id - The unique and persistent identifier for the household that the device belongs to.

Sub Demographic Data

Contains a wide variety of household characteristics information (a smaller version of the data you saw in project 1)

- household_id - The unique and persistent identifier for the household.
- household_size - Number of people residing in the household.
- num_adults - Number of adults residing in the household.
- num_generations - Number of Generations in Household.
- marital_status - Code indicating marital status of the head of household.
- race_code - Code to indicate the race of the head of household.

- dwelling_type - Dwelling type indicator code.
- home_owner_status - Home owner/renter code.
- length_residence - Length of time in years that the head of household has lived in the household.
- home_market_value - Value of the household.
- net_worth - Net worth of the household.
- education_highest - Code that indicates the highest level of education.
- gender_individual - Gender of the individual used to identify the household.

Assignment

Static Data Analysis (70 Points)

Feature Extraction

In order to use the data we have more efficiently you are tasked to do several tasks.

First, we will explore the demographic data and analyze it.

We would like to adjust and normalize the features:

1. For numerical variables - normalize each column according to the maximum and minimum values in that column so the values in each numerical column would be in the range of [0,1] - for value x the normalized value will be $\frac{x-min}{max-min}$.
2. For categorical variables (non-ordinal) - use one hot encoding to change the values of the column to a binary vector representation.

Then, combine all adjusted and normalized variables into one feature vector column that will represent each household.

Notice! The household_id number is NOT a valid data feature.

Show 10 rows from the resulting dataframe, using *df.show(10 , truncate=False)* or *display(df.limit(10))* whatever looks better.

Answer in the PDF:

- Why we normalize the numerical columns? why do we need to change the categorical columns to vectors and it is not enough to change them to numerical values?

Visual Analysis

Now that we have one column that contains all our useful information, we can easily use many known and good algorithms to infer information from our data.

We want to see if the data is naturally divided into several groups of households. In order to do that we can use a dimensionality reduction algorithm, two famous algorithms are SVD and PCA. SVD divides the data into 3 matrices U, Σ, V such that $U\Sigma V^T$ equals our original data. The matrix that contains our reduced data is U. Use the SVD algorithm to project the feature vectors from the previous part on to 2 dimensional space and then plot the result in a scatter plot.

Use the already implemented SVD algorithm from PySpark MLLib, with k=2.

In order to plot, you may use:

svd_df.toPandas().plot.scatter(x='x_col_name', y='y_col_name')

- Show 10 rows from the resulting dataframe, using *df.show(10 , truncate=False)* or *display(df.limit(10))* whatever looks better.
- Show the resulting scatter plot.

Answer in the PDF:

- What do you see in the scatter plot? How many different clusters do you see in the plot?

Clustering

The SVD algorithm can help us get intuition if the data is truly divided into several clusters, however it is not a clustering algorithm and using it to cluster data can be problematic. Use the K-means algorithm to cluster the data into 8 clusters.

- Add a new column that states which household_id belongs to which cluster. The cluster names/ID-numbers do not matter.
- Add a new column that contains the distance of each household_id from it's centroid (Hint: use window functions).
- Use the already implemented K-means algorithm from PySpark MLLib, with k=8 and Set seed to 7.
- Show 10 rows from the resulting Dataframe, using *df.show(10 , truncate=False)* or *display(df.limit(10))* whatever looks better.

Visual Clustering

Now that we divided the data into clusters we will see if we can find a visual connection between data points in the same cluster and dissimilarity between data points from different clusters.

- Use the PCA algorithm to reduce the dimension of the data points you received from the previous section to size 2.
- Show 10 rows from the resulting Dataframe, using *df.show(10 , truncate=False)* or *display(df.limit(10))* whatever looks better.
- Plot the data points with dimension 2 on a scatter plot where the points of every cluster have different color.
- For the plotting (in this section only) you may use seaborn library.

Answer in the PDF:

- What do you see in the scatter plot? Do you think that if we change the number of clusters in the previous section we will get a better color division in the scatter plot? If so, to a higher or to a lower number of clusters?

Dividing households into subsets

For the calculation of the subsets, you MUST utilize PySpark Windows.

Order the households in each cluster by their distance from their cluster's centroid, in ascending order of distance - closest to centroid will be first.

For each cluster from the demographic data observe the subset that contains every seventh row in that cluster (the 7th, 14th, ... rows in that cluster). Remember - the rows are ordered in each cluster by the distance to their centroid.

We'll call that subset the *"7^{ths} subset"* Notice that we have 8 such subsets - one for each cluster.

Do the same again for each cluster, now for each 11^{th} row (11th, 22th, ...). We'll call this subset the *"11^{ths} subset"*. Notice that you also have 8 such subsets.

In addition for the 2 kinds of subsets (7^{ths} and 11^{ths}), you also have the full data for each cluster.

In total, you should have 24 sets of household_ids - 3 sets per cluster.

Notice! - The division of households into clusters and then into subsets is done ONLY ONCE! Any further mention of these subsets refers to the initial division done in this part.

Each household_id is tied to a cluster (only one - same household cannot belong to 2 clusters) and then to subsets - one or more.

For example, the 77^{st} household in distance to its cluster belongs in all 3 subsets of that cluster - as 77 is divisible by 7 and by 11, and also all subsets are included inside the "Full" subset.

Cluster's Viewing Analysis

Next we want to see if the clustering on the households that we've done holds any value. For that we will use the viewing data - to see whether the viewing habits of people from different demographic clusters are actually different.

In order to asses the uniqueness of each cluster's viewing habits, we will do the following procedure - that will yield us the 'diff_rank' measure:

1. For each cluster/subset inside each cluster, count the amount of viewing events (rows in the viewing_data) for each station.

   The output of this stage (per each subset of each cluster) - is a count of viewing rows, aggregated per station number.

2. Divide the amount of viewings per station in the total amount of views for that subset of that cluster, and then multiply by 100.

   The output of this stage is the percent of viewing events per station in this subset of each cluster. This can be treated as the cluster's/subset's popularity rating of a station.

3. Now, repeat steps 1 and 2 - but for the entire data, **not** on a per cluster/subset basis. This is the popularity rating of each station among the general population.

   The output of this stage is the popularity ratings (percentages) of each station among the general public.

4. For each station and each subset of each cluster, subtract the popularity rating (percentage) of the general public from the popularity of that station for that specific subset of each cluster.

   We will name the output of this stage - the 'diff_rank' of a station among a cluster.

- **For each subset** calculate the top-10 highest 'diff_rank' stations. Overall you need to return 24 groups of top ranked stations, one for each subset - 3 sets of results per cluster.
- Print / Show these results, with clear titles that differentiate between subsets.
- Each cluster's 3 sets' results should be shown close to one another.
- You may take artsy liberty in choosing the exact way of presentation, while conforming to the previous bullets.

Answer in the PDF:

- What does a positive / negative 'diff_rank' of a station for a cluster mean? What insights can we infer from the 'diff_rank' of a station & cluster about the viewing habits of people in that cluster, compared to an 'average household'?
- Compare the results of the same subset kind (7ths, 11ths, Full) between clusters, and also compare between different subset kinds inside the clusters. Do we see any value in the clustering we've done earlier? Describe the differences between the results, if there are any.

Dynamic Data Analysis - Streaming (30 Points)

Use Spark Streaming to extract the viewing data from Kafka. Use the code that was provided to you in the Starter notebook in order to connect and read from the stream.

While streaming, repeat the "Cluster's Viewing Analysis" task from the previous task, for each trigger's data.

You should get outputs for each batch of viewing data that you've read from the stream.

(In case your code does not produce any prints to the cell output - check the Driver Logs - accessible via a click on your running cluster menu. You should copy the results from the streaming process to a cell in the appropriate place.)

- Read and process at least 3 batches/triggers. Show the results per batch.
- *Notice! In this part - you should only analyze the "7^{ths} subset" per each cluster*

  That means that for each trigger you would have 8 groups of top ranked stations - only 1 group of stations per cluster instead of 3.

  We are interested only in the 7^{ths} subset output in this part.
- In the PDF - Compare the results you got each time between each cluster. Are there any similarities? (i.e. is there a connection between two households in the same cluster to their liked stations?)

In this part you should NOT use any viewing data from the static part, but you must use the other demographic data and division to subsets (etc.) from the previous parts. It is possible that some code you've written will require changes and adjustments for the streaming part.

The results of each batch should include the data from all the previous batches. For you - watch out for what data do you aggregate from all the batches - Think - Is collecting all raw viewing data entries feasible in the long run? Will it affect runtime while streaming?

What to submit?

1. **Jupyter notebook (.ipynb)** containing your code, the analysis and code explanations and cell outputs. File name: **PROJECT2_WET_[ID1]_[ID2].ipynb**
2. You should submit the **3 versions** of your notebook: a **.ipynb, PDF and HTML INCLUDING your outputs.**
3. **PDF file** containing your analysis and **answers to questions** throughout the instructions. File name: **PROJECR2_DRY_[ID1]_[ID2].pdf**

[ID1] and [ID2] are the ids of the students doing the project.

General Guidelines

- Use *Spark 3* and above.
- Use only pyspark and matplotlib (when plotting is needed).
- Write clean code and document functions when necessary.
- Questions related to the project will be answered in the forum, via Moodle, exclusively.
- Only one of the team members need to submit.
- Submission is due to August 24, any delay in submission will result in a reduction of 20 points from the final score of this part.
- The use of generative AI tools such as ChatGPT is permitted. However, we recommend to use it sporadically and only after you have given the code a try yourself. Keep in mind that ChatGPT is not a legitimate helper during your final exam. Should you choose to use generative AI, please indicate clearly where it was used, in what manner did you decide to use it (e.g., which prompts were used) and to what extent was it helpful (e.g., what was the amount of work you needed to invest to correct it). Note that reporting on the use of said tool will not affect your grade in any way.

Good luck,
Course staff.