Homework 2

Submission details:

1. Submission date: **06.03.24 at 23:59.**
2. Submission should be in English only.
3. Submission is in 3 person teams.
    a. Only Team Leader submits via Moodle.
    b. Submission should be in a zip file named "HW2_ID1_ID2_ID3.zip" where ID1, ID2, ID3 are the team members
        i. For example, for 3 students with ids – 123457800, 456789120, 124578890, the zip folder will be named: HW2_123457800_456789120_124578890.zip
    c. Notebook template:
        i. We provide you with a template for the submission notebook.
        ii. Make sure that you follow the guidelines and formats.
        iii. Make sure you keep it clean and organized.
4. Questions will be submitted to special Moodle forum, only by team leader, in English.
5. Questions about the assignment that will be sent by email will not be answered.
6. See adjusted points for **Miluim** in **green**. E.g. (**original pt.**) **(adjusted pt.)**
7. **Late submission** will result in 10 pt. reduction for each overdue day.

Submissions checklist:

@TeamLeader - Ensure that you submitted:

1. HW2_ID1_ID2_ID3.zip
    a. report_ID1_ID2_ID3.pdf - **OPTIONAL**
        i. Should include answers and plots where necessary.
        ii. Can include external links to visualization.
    b. notebook_ID1_ID2_ID3.ipynb
    c. notebook_ID1_ID2_ID3.pdf
    d. notebook_ID1_ID2_ID3.html

Background

Similarity and Embeddings

In data science, "similarity" refers to a quantitative measure that captures the degree of likeness between two entities. This likeness is often derived from attributes or features of the entities being compared.

An embedding is a transformation of data into a lower-dimensional space, often a vector, such that the geometric relations in this space mirror the semantic or conceptual similarities between the data points.

Spark NLP

Spark NLP is a natural language processing (NLP) library built on top of Apache Spark. Spark NLP leverages Spark's capabilities to enable high-performance, scalable text processing and machine learning (ML) applications. It provides a comprehensive suite of NLP functionalities, including tokenization, lemmatization, part-of-speech tagging, named entity recognition, sentiment analysis, and more, enabling developers and data scientists to build sophisticated NLP applications quickly and efficiently.

General guidelines:

Questions 1-4:

- All questions should be implemented in **pyspark**:
  - You may use *sparknlp* and MLlib (native spark packages) for machine learning tasks and also nltk and scipy for text preprocessing.
  - torch, sklearn or similar are not allowed.
- Pandas, NumPy, or similar can be used only for visualization and statistical analysis.
- Collab or similar is not allowed.

Questions:

Question 1: Data Processing (15 pt) (25 pt)

Your task involves working with the *linkedin/companies* dataset, specifically focusing on the 'similar' attribute, which will represent a **match**. The aim of this question is to flatten this attribute and create a new table. Follow the steps outlined below:

1. **Exploding**: Create a new table where each row symbolizes the similarity relationship between two companies - Company A and Company B.
    a. A row is present in this new table if Company X is found in the 'similar companies' list of Company Y or vice versa.
    b. A row is present only if both companies appear in the "companies".
2. **Naming Convention:** The columns from the original dataset should be prefixed with 'company_A' for the first company and 'company_B' for the second, and suffix with _[column].
    a. Company A is the one with a smaller lexicographical value of the column name.
    b. For example: company_A + about [column]= > company_A_about
3. **Columns order**: Ensure that the column sequence mirrors the original table, starting with **all** the "company_A" attributes followed by the attributes of "company_B".
    a. Make sure that the first column is company_A_name, and then all the rest of company_A columns' (by companies order)., and same for company_B.
    b. This should help you in the validation section.
4. **Similarity Assumption:** Consider similarity as symmetric. This means that if Company B is in the 'similar' companies' list of Company A, both are deemed similar even if the reverse is not listed.
    a. Company A and Company B will only pair up once. That is, if Company B is listed in the 'similar' attribute of Company A, and vice versa, they will be represented as a single row, regardless of their mutual listings in each other's 'similar' attribute.
    b. Ignore self-similarity. That is, if company A is listed in a similar column of itself, you should ignore it.
5. **Dropping Columns:** exclude "similar" columns.
6. **Final Task:** After performing the above transformations, display the schema of the dataframe using *df.printschema().*
7. **Validation:**
    a. Create a subset of the above by filtering on company_A_name from the following list:
        i. ['Altium Leadership', 'American College of Music - ACM Online', 'Arieca Inc.', 'Bango Bowls', 'Blue Llamas, Inc.']
        ii. order as (i) and then by company_B_name

b. Display the filtered df using *.display().*

Example:

Given a company and a subset of the columns, your output should look **similar** to:

a. Company_A_name : Athena Smart Cities
b. Company_A_country_code : MY
c. ….
d. Company_B_name : Eboss Group Holdings
e. Company_B_country_code - ....
f. …

Notes:

- Here, "Athena Smart Cities" is Company A because it starts with an "A", whereas "Eboss Group Holdings" starts with an "E".
- This is an example of a partial output **not** formatted as the actual output.
- Companies should be joined based on similar.Links/url field/column, using parsing when necessary.
    - Filter companies with null in the url column.

Question 2: Missingness Mechanisms (20 pt) (0 pt)

What is the missing values distribution in the “companies” table?

When answering, address the following points:

- Missing values are not necessarily “null”, define what is the correct definition of “null” value for each column.
- Classify the missingness mechanism of each column into one of - Missing Completely at Random (MCAR), Missing at Random (MAR), and Missing Not at Random (MNAR) or other (no clear answer, or an alternative mechanism). **Explain how you reached this conclusion.**
    - Only classify columns with at least 2% missing values and less than 95%.
    - For a column that contains:
        - an array of structs, it is sufficient to look at the array as a whole (and not at each exploded list)
        - a struct, you should look at each key (i.e. current_company:company_id)
- Add visualizations and summary tables where necessary.
    - Support your answer with statistical analysis.

Notes: There may be more than one correct answer for the question.

Question 3: Evaluating Similarity (25 pt) (35 pt)

In many machine learning scenarios, we often encounter ground truth labels and attempt to make predictions based on them. However, it is not uncommon for these labels to be imperfect representations of our intended target or have errors.

For this question, assume the role of a data scientist at a rival company to LinkedIn named "*ConnectSphere*" tasked with assessing the effectiveness of the "similar" column in expressing the similarity between two companies. In addition, try to reverse engineer the algorithm of LinkedIn and estimate the general pipeline.

General Guidelines:

- Exploration:
  - Give 2 examples of good and 2 examples of bad matches.
  - Generalize and identify the strengths and weaknesses of the LinkedIn matching algorithm.
- Reverse engineering:
  - *Try* to interpret and explain the labels using feature analysis and interpretability methods.
  - What is reverse engineering in our context?
    - Reverse engineering in our context refers to the process of deconstructing and analyzing a model or algorithm to understand how it works, what features it uses, and how it makes decisions.
    - You do not need to implement the actual **LinkedIn** algorithm, but you should use different methods to help you understand the general idea behind their algorithm.
    - You can also use any algorithm you learned in previous DS courses (e.g. unsupervised or supervised methods), algorithms presented in the Booking.com lecture, or any other interesting way that you can think of.
- Findings:
  - Summarize your findings and decide how good the algorithm is based on the data and labels.

Notes: There is more than 1 correct answer and multiple ways to address this question. We want to understand your thought process and see critical thinking.

Question 4: Supervised Learning (40 pt)

Learning Setting: Supervised Multiclass classification

Objective:

Develop a model to predict the **meta industry** of a given company and analyze its performance.
Include the following details in your analysis:

- Feature engineering and selection methods
- Preprocessing pipeline
- Model selection
- Error analysis and Class-wise Performance Metrics

Background:

- We will reference the **industry** of a company represented by the column "*industries*", which is also identical to the column "*sphere*"
    - Do not use the "*sphere"* column in your features.
- We map each **industry** not-None string value to a **meta industry.**
    - For example, the following is a mapping defined in your notebook:
        - Industry: 'Furniture and Home Furnishings Manufacturing'.
        - Meta industry: "Manufacturing".
- In total:
    - 147 **industries** -> 12 **meta industry**.
    - Ignore **industries** with None.
    - *The mapping is provided in the submission notebook as a dict.*

Labeled Data:

- You will be provided with ~1M rows of labeled data in the table *linkedin/companies.*

Prediction Task:

- You will be provided with ~300K rows of unlabeled data in the table *linkedin/companies_test*.
    - The data was sampled from the same distribution of the original companies' dataset.
- For each company you should predict its **meta industry.**
- Your prediction should be based only on **meta industry** classes seen at the *linkedin/companies*.

Output Specifications:

Save your predictions in a CSV format named: "test_ID1_ID2_ID3.csv" with the following columns:

- *id*
- *label*

All column values should be treated as strings.

Example of Output Format:

id, label
'Xxxx', 'Manufacturing'

'Aaa', 'Real Estate and Construction'

...

Notes:

- The predictions in your output should match the order of the column ***id*** *(not company_id)* as found in *linkedin/companies_test*.
- For example, in the provided example, id 'Xxxx' precedes 'Aaa' because, in *linkedin/companies_test*, 'Xxxx' is listed as the first company and 'Aaa' as the second.
- Save your output to **databricks** using:

df.write.csv("/mnt/lab94290/results/test_ID1_ID2_ID3.csv", header=True)

- Validate the output correctness using read:

df = spark.read.csv("/mnt/lab94290/results/test_ID1_ID2_ID3.csv", header=True)

- ***You do not need to upload the CSV to Moodle – save it to Databricks using the above instructions.***

Evaluation:

- **Metric:** weighted F1 score.
- **Baseline (10 pt.)** **(20 pt.):** 0.3 weighted F1 score on the validation set.
    - Perform a random split with a ratio of 80-20 with seed-specific for your group:
        - Seed Calculation: ID1+ID2+ ID3 last 2 digits.
            - If the last 2 digits begin with 0, only use the second digit or 0 if both are 0.
        - Example:
            - ID1 = 12345678, ID2=2345678, ID3=3456789

        - Sum=1814814**5**
        - Seed=45
    - Report the weighted F1 score on the train and validation sets.
- **Competitive (20 pt.)** **(10 pt.):** You will be evaluated in comparison to other groups on the test set.
- **Analysis (10 pt.)**: Report quality.