REVEAL
Data Scientist Intern Tech Case Documentation

Description : Reveal develops a solution that aims at finding companies present in multiple CRMs, to build a bunch of different features for our customers. Two datasets of companies and some of their attributes were provided. They represent the data available in the CRMs of two of their users

Objectives: The main goal was to write an algorithm that will use the data from the 'companies' table and compute matches based on the company attributes

Deliverables: A python code and two csv file with found matches

- The first CSV file named **matched_companiesF.csv** containing the match using only the name as attribute
- The second CSV file named **matched_companiesFF.csv** containing the match using not only the name as attribute but also other parameters such as postal code and country

Coding part

First Step: I first downloaded the two datasets (Dataset_A and Dataset_B), then I had a look at each dataset in order to see what type of data I am dealing with. After that i tried to guess the description of each dataset. The first dataset, Dataset_A contained 8723 rows and 5 unlabeled columns and the Dataset_B contained 8795 rows and 8 unlabeled columns, so i tried to guess the meaning of each information specially the description of each column, so I finally came up with this description of columns:

Dataset_A:	Company ID
	Company Name
	Phone Number
	Postal Code
	Country
Dataset_B:	Company ID
	Company Name
	Website
	Phone Number
	Address
	Postal Code
	City
	Country

I then restructured the two datasets by created two new datasets called **restructured_datasetA.csv** and **restructured_datasetB.csv** having the columns labelled accordingly. So that it will be easier for me to access information of the two datasets for manipulation and finding matches.

Second Step: I then starting my algorithm but the most important thing to keep in mind is that since we want to find similarities among the two companies; I tried three different approaches but the main one I kept for the sake of this project leverage on the **Cosine similarity** to find the similarity among the two datasets. In support to my approach I also used two main libraries **TfidfVectorizer** and **linear_kernel**:

1. TfidfVectorizer

- TfidfVectorizer is used for text feature extraction and vectorization. TF-IDF stands for "Term Frequency-Inverse Document Frequency," and it is a numerical representation of text data.
- The purpose of TfidfVectorizer is to convert a collection of text documents into numerical feature vectors. It tokenizes the text, counts the frequency of each word (term) in each document, and then scales the count with respect to the term's importance in the entire corpus (collection of documents) to account for commonly occurring terms.
- The resulting vectors represent the importance of each term in a document relative to the entire corpus, and they are suitable for various machine learning tasks such as clustering, classification, and similarity calculations.

2. linear_kernel

- linear_kernel is a function used to calculate the linear kernel between two sets of samples. In the context of text similarity, it is used to compute the cosine similarity between vectors representing text data.
- The cosine similarity is a measure of similarity between two non-zero vectors in an inner product space. For text data represented using TF-IDF vectors, the cosine similarity measures the cosine of the angle between the two vectors, providing a value between -1 and 1.
- A cosine similarity of 1 means the vectors are pointing in the same direction and are perfectly similar, while a value of 0 means the vectors are orthogonal (no similarity), and a value of -1 means they point in opposite directions.
- In the matching context, the cosine similarity is used to determine how similar the company names (and potentially other attributes) are between the two datasets, enabling the identification of potential matches.

Overall, for this project, The TfidfVectorizer is used to convert the company names, postal codes, and countries into TF-IDF vectors. Then, the linear_kernel is used to calculate the cosine similarity between these vectors, allowing the code to measure the similarity between companies in the two datasets and identify potential matches based on the similarity scores.

So, diving into the coding part, I first defined the preprocess function taking name as parameter. This function takes a company name as input, removes any non-alphanumeric characters, and converts the name to lowercase. The purpose of this function is to standardize the company names so that they can be compared accurately.

Next, I defined the match_companies(dataset_A, dataset_B, output_file): This is the main function that performs the company matching process. It takes two CSV file paths as input (representing the two restructured datasets to be matched) and an output file path to save the results. Here's how the function works:
a. Load the datasets into pandas DataFrames.
b. Drop any rows with missing 'Company Name' values from both datasets.
c. Preprocess the company names using the preprocess_name function.
d. Create TF-IDF vectors for the preprocessed company names in both datasets using the TfidfVectorizer.
e. Calculate cosine similarity between the TF-IDF vectors of dataset A and B using linear_kernel.
f. Iterate through each company in dataset A and find its best match in dataset B based on the highest cosine similarity score. A threshold is used to decide whether two companies are a match or not (adjustable by the user).
g. Store the matched pairs and relevant information (Company IDs, Names, and Similarity Scores) in a new DataFrame.
h. Save the matched pairs to a new CSV file specified by the output_file.
The script also includes an if __name__ == "__main__": block, which ensures that the match_companies function is executed only if the script is run directly (not imported as a module). This allows you to run the script from the command line, and the company matching process will be executed using the specified input datasets and save the results to the output file.

Without forgetting that Set a threshold for similarity score in this case I took **0.8** for a good matching accuracy but we could tune it depending on our needs or purposes.

Output: I compute two matches: **matched_companiesF.csv** only using names and it is not good enough I mean it does not really filter the matches enough compared to the **matched_companiesFF.csv** that I did not only using name but using postal code and country and that one is more filtered and it contains less matches. And the purpose of doing two filtering was to see the influence of the other attributes.

If I had more time, there are several areas in which I could potentially improve the matching process and the overall code:

- **Handling Missing Data:** The current code drops rows with missing company names in both datasets. Instead, we could explore imputation techniques to handle missing data more effectively, especially if other attributes (e.g., phone number, postal code) are available and could be used for better matching

- **Multiple Attribute Matching**: The current implementation focuses solely on matching based on company names. However, incorporating other relevant attributes (e.g., postal code, country, phone number, website) could improve the matching accuracy. Different weights could be assigned to each attribute based on their importance in the matching process.

- **Data Preprocessing:** Additional data preprocessing steps, such as standardization, normalization, or stemming, could be explored to enhance the quality of the matching process. For instance, removing stop words or applying lemmatization might improve the feature representation.

- **Duplicate Handling**: The code currently does not handle duplicates well. If a company name is repeated in either dataset, the current approach might lead to incorrect matches. Addressing this issue would require additional logic to handle duplicates appropriately.

- **Evaluation Metrics:** Implementing evaluation metrics such as precision, recall, and F1-score could provide a quantitative measure of the matching performance and help assess the effectiveness of the matching process.