Challenge 2: Analysis of comments on Glassdoor

Name: **María Paloma Prado Durán**

Student code: **224807991**

Email: maria.prado0799@alumnos.udg.mx

Subject: *Programming II*

April 29, 2025

Development:

Dataset Extraction.

To begin the process, the first step is to load the data you'll be working with. To do this, the *load_data_frame* function was created, which reads the CSV file and assigns it as *df*.

It's important to note that the dataset to be loaded will be glassdoor_reviews.csv, The dataframe compiled by David Gauthier contains a large number of employee reviews of their employers. Each review includes job descriptions and ratings on various criteria such as work-life balance, income, culture, and more. This dataframe is valuable for analyzing how employees perceive different aspects of their jobs and for developing models that predict overall ratings based on textual comments. For example, it has been used in projects that apply language models like BERT to predict a company's overall rating from employee reviews.

Model Construction.

Once the dataframe is loaded, the *preprocessing* function is used to create a copy of the dataset with the column of interest to be analyzed. By default, the *pros* column is used. This column will be transformed by removing punctuation marks, extra spaces, and other elements. A column containing the text without stopwords will be added to this new dataframe, as well as another column containing the lemmatized text. Finally, the text's n-grams will be created. By default, they will be created and graphed for a range of 2 to 5. It is important to mention that these plots will be saved in the *plots* folder.

Once the dataframe is ready, the new module called ***nlp*** is used. This creates an *lda model* that classifies the words into four themes (this value can be modified). Sentiment analysis is also performed, allowing the generation of word cloud charts by theme. Just like the other charts, these will be saved in the "plots" folder.

Once this is done, the information is divided into testing and training (20%-80%) with which it will be used to train 5 classification models Naive Bayes, Logistic Regression, Random Forest, SVM and Decision Tree, these models were selected arbitrarily and you can always choose to change. With these models, predictions will be made and performance metrics will be calculated, such as accuracy, precision, recall and f1-score, without leaving aside the calculation of the ROC accuracy to make the ROC curve graph, finally, the module graphs the confusion matrix of each model.

This calculated metrics are important because:

- *accuracy*: Measures the proportion of correct predictions out of the total number of predictions made, i.e., whether the model has good overall performance or not.
- *precision*: This measure tells us how many times the model actually predicted malignancy. A high value indicates that the model makes few false positives (FPs), meaning that few benign cases were misclassified as malignant. This is useful when the cost of a false positive is high (for example, if an unnecessary biopsy is expensive or risky).
- *recall:* It measures how many of all the truly malignant cases the model correctly identified. A high value indicates that the model makes few false negatives (FNs), meaning that few malignant tumors were misclassified as benign. This is key in medical problems where failing to detect a malignant case can be dangerous (as is the case with our dataset).
- *F1-score*: This is the harmonic average of Precision and Recall, giving a single value that balances both aspects. A high value indicates that the model has a good balance between avoiding false positives and false negatives.

Likewise, the confusion matrix will be graphed, which, using a 2x2 table, shows how many of the model's predictions were correct and incorrect in each category, in our case detecting cancer or not. Finally, the ROC Curve shows the model's performance for different classification thresholds, that is, how the ratio of true positives (TPR) to false positives (FPR) changes. An *auc_score* greater than 0.9 indicates a very good model.

Additionally, it generates a classification report summarizing the performance of a classification model on a test dataset. The report includes several important evaluation metrics used to measure the effectiveness of a classification model, such as precision, recall, f1-score, etc. The report will be saved as a .txt file

MLOps.

Once the metrics are calculated, we want to save them. Therefore, in the ***mlops_pipline.py*** script, which contains the aforementioned functions, we will use the `mlflow` library. This is an open-source platform purpose-built to assist machine learning practitioners and teams in handling the complexities of the machine learning process. MLflow[1] focuses on the full lifecycle for machine learning projects, ensuring that each phase is manageable, traceable, and reproducible. The following functions will be used in this challenge:

- ***set_tracking***: Defines the location where MLflow will store experiments and models.
- ***set_experiment***: Defines or creates an experiment in MLflow to organize runs.
- ***log_metric***: Saves numerical metrics of the model, such as precision, recall, F1-score.
- ***log_text***: Saves a text to a file within the MLflow log.
- ***log_artifact***: Save files as images, graphs or reports in MLflow, in this case it will be the confusion matrix and the ROC curve generated and saved.
- ***log_model***: Save the trained model in MLflow for future predictions or deployments.

Execution Guide.

To execute the script, the user is invited to follow the following steps::

1. Clone the repository https://github.com/Paloma-PD/Challenges-Progra2.git using a git clone.

[1] mlflow: https://pypi.org/project/mlflow/

<u>NOTE:</u> The repository has information from challenge 1, in case you have already cloned it, you must run the following command in the terminal: `git pull origin main`

2. Install the minimum dependencies for execution using the requirements_2.txt file
3. 3. Run the mlops_pipline.py script, which is the one that concentrates all the functions.
4. Run the upload_results.py script to update the results/changes directly to GitHub

It is important to mention that to execute the scripts you must be in the directory where they are located, this in case you execute them from the terminal or git bash.

Extra: Architecture diagram..

As support, the project architecture diagram is included

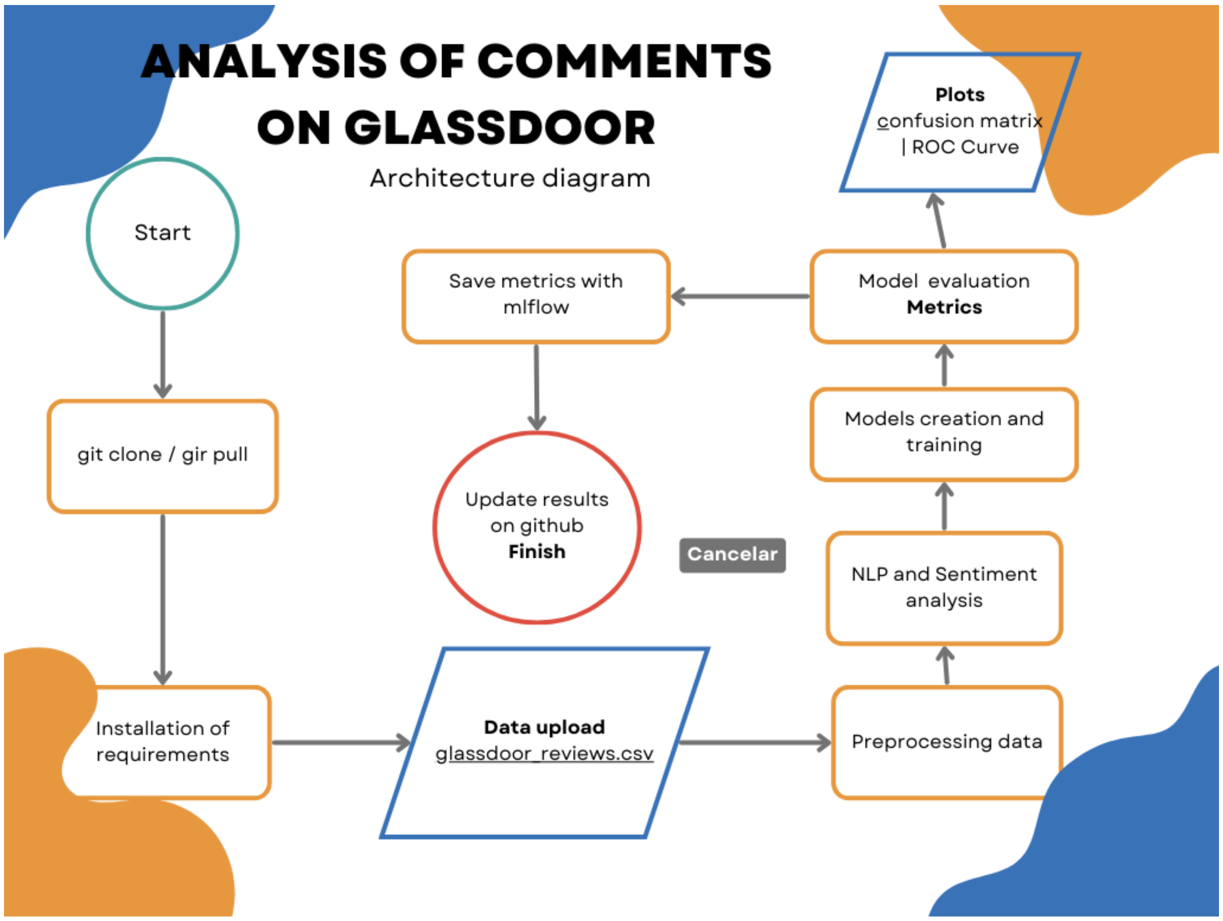
ANALYSIS OF COMMENTS ON GLASSDOOR
Architecture diagram
Start
git clone / gir pull
Installation of requirements
Data upload glassdoor_reviews.csv
Preprocessing data
NLP and Sentiment analysis
Models creation and training
Model evaluation Metrics
Plots confusion matrix | ROC Curve
Save metrics with mlflow
Update results on github Finish
Cancelar

Figure 1. Architecture diagram (Canva.com)