Using ML for Predicting Socio-Economic Trends: Leveraging Google Trends, GDELT, and Eurostat Data

Cristina Jiménez, Ayah Dahmani, Ricardo González Otal and Juan Miguel López Piñero

TABLE OF CONTENTS

1.OVERVIEW

This project aims to develop ML models using advanced machine learning techniques, such as Now casting, Long Short Term Memory(LSTM), Random Forests, X gradient boosting, and Neural Networks. These models will be employed to predict a set of socio-economic variables in relation to labor market integration, providing valuable insights for decision-making processes. Moreover, this project will serve as a tool for new graduates from the Data Science program 2023-2024 of The Bridge School, allowing them to apply and showcase their skills in machine learning and data analysis.

2.INTRODUCTION

In today's rapidly evolving world, the ability to predict and understand socio-economic trends has become increasingly valuable. This project, created by the company Rambee and assigned to a team of four graduates from The Bridge School, aims to develop a nowcasting model based on machine learning techniques to forecast a specific socio-economic variable using big data obtained from Google Trends, the Global Database on Events, Language, and Tone (GDELT), and Eurostat.

Nowcasting,("now" and "forecasting"), refers to the process of predicting the present or the very near future by analyzing current and historical data. This approach is particularly useful in situations where traditional forecasting methods may not be as effective due to the lack of timely data or the need for more immediate insights.

The Team will test and evaluate the performance of a Nowcasting, LSTM neural network architecture, which is well-suited for sequential data and capturing long-term dependencies, making it a promising candidate for nowcasting tasks.

The main goal of this project is to develop and validate a ML model for sequential data analysis. The model will utilize three key data sources: Google Trends for search volume trends across countries, GDELT for sentiment indicators and topic popularity from news articles, and Eurostat for official socio-economic statistics from the European Union. By combining these diverse data sources, the model aims to provide insights into consumer behavior, sentiment trends, and socio-economic indicators for forecasting purposes.

3.AIMS AND OBJECTIVES

The primary tasks involved in this project are as follows:

1. Extract information on search volume queries from Google Trends for a predefined set of categories, with weekly frequency, across different countries.
2. Retrieve sentiment indicators and topic popularity rates from GDELT for a series of relevant socio-economic themes, in the form of "Article Tone" and "Topic Popularity Rate" data.
3. Obtain relevant socio-economic data from Eurostat for the variable of interest.
4. Construct and validate an ML model, including a LSTM model for a specific socio-economic variable of interest, using the extracted data from Google Trends, GDELT, and Eurostat as predictors.

The final model will enable a deeper understanding of current trends and facilitate proactive decision-making processes.

4. PYTRENDS API

4.1. API Overview

Google Trends is a powerful tool that provides insights into the popularity of search queries over time. To check the potential of this data, the team turned their focus to the PyTrends API, a Python library that offered an interface for accessing and analyzing Google Trends data.

4.2. Accessing the Gtrends

To create an API that extracts data from the Google Trends platform, we used the unofficial PYTRENDS API as a base. Our goal was to download the data in CSV format and preprocess it for further use. We chose to build the API using FastAPI, a framework we are familiar with thanks to The Bridge.

This API facilitates access to Google Trends data through the unofficial Pytrends API and includes specialized endpoints to extract the average search interest for a defined topic over a specific time period. Initially, we created endpoints with predefined date and country parameters. However, we are considering the option to include these parameters as part of the endpoint calls or the extraction function, allowing for more flexibility in specifying the desired countries or date ranges.

4.3. Challenges and Solutions

We developed two versions of the same API, one with the cleaning function integrated within each endpoint, and another with an independent endpoint for final data cleaning. One of the challenges we faced was ensuring that multiple API calls and corresponding CSV files do not overwrite each other. To address this, we plan to implement a timestamp or a similar mechanism to prevent data overwriting. Additionally, we need to verify that the data we obtain is monthly rather than weekly, as weekly data tends to become null. Furthermore, we implemented a function to clean the retrieved DataFrame by removing rows with null values and duplicates, ensuring data quality and consistency.

To handle potential errors during the data retrieval process, we created a function that includes a try-except block that retries the request after a brief delay if an error occurs, preventing system saturation and the dreaded 429 error.

Accessing Google Trends data through the Pytrends library presented many challenges, including rate limiting errors (429) due to excessive requests in a short period. Various solutions were explored, such as implementing time delays using time.sleep() and batch processing, searching for topics by their numeric codes, trying alternative code snippets from public repositories, upgrading to the latest Pytrends version, and creating custom headers from the Google Trends website. However, none of these approaches proved successful. Ultimately, the issue was resolved by incorporating a try-except block with a time delay between requests, effectively spacing out the searches and avoiding rate limiting errors.

However, It is crucial to note that Google Trends imposes rate limits on the number of requests that can be made within a certain time frame. If these limits are exceeded, the Pytrends library may encounter errors or receive incomplete or inaccurate data.

In addition to that, we also encountered further issues while working with the Google Trends API. These issues included inefficiency, with the API functioning only one out of every six attempts, and failures in cases of multiple retries. If more than four retry attempts were made, we had to wait until the following day to try again. Furthermore, we experienced excessively long wait times, requests exceeding the timeout limit, and further HTTP errors (400, 500, etc.). There was also a risk of our IP address being blocked by Google due to excessive requests or potential misuse of the API.

These challenges highlighted the limitations and uncertainties associated with relying on third-party APIs, particularly those provided by large tech companies with strict usage policies. Therefore, the resources and effort required to fully integrate and operationalize this data source were not justified by the marginal value it would add to the final analysis.

4.4 Final considerations on Google trend

Despite the initial plan to integrate the API from Google Trends,the team had to make strategic decisions due to the limited time available. Google Trends data obtained through the PyTrends library was initially considered, the team determined that productionizing the Google Trends API or recommending it to customers for data extraction would not be advisable due to the effort required and the limited additional value it would provide. However, to leverage the potential insights from the Google Trends data, the team opted to load this data locally for the specific purpose of training their models. This approach allowed us to incorporate the Google Trends data into their analysis without the overhead of productionizing the API or recommending it as a production-ready solution.

The integrated local dataset, comprising data from Eurostat, GDELT, and Google Trends data, is going to form the foundation for chapter 9 on modeling and enable efficient deployment, through a Model API.

5. GDELT API

5.1. API Overview

The Global Database of Events, Language, and Tone (GDELT) is a massive open-source database that monitors, captures and codes events from news articles and classifies them into a structured database.
These attributes make GDELT a valuable resource for analysts, and data scientists interested in studying global events. In the following section, we will explore how the team managed to access the GDELT database, extract relevant data, transform and format it for analysis, and address common challenges encountered while working with this data source.

5.2. Accessing Extracting and Transforming the GDELT Database

The team objective was building an API that could extract and analyze data from GDELT , accessing and working with such a vast and complex dataset has certainly been a challenge. After familiarizing ourselves with the GDELT documentation and understanding the different types of data available, we decided to focus our efforts on extracting sentiment indicators (tone) and topic popularity data. These two aspects would provide valuable insights. We chose to build our API using the FastAPI framework, because it offered an efficient approach. The first step was to set up the project structure and install the necessary dependencies, including the GDELT doc library, which would serve as our gateway to the GDELT database.

With the project setup complete, we began designing the API endpoints. We wanted to provide users with the flexibility to extract data at different time intervals – daily, monthly, and quarterly. This would cater to various use cases and allow for more granular or aggregated analysis as needed. The first endpoint we built was */diary/extraction*, which allowed users to extract daily data for a specific keyword and country. We implemented filters to narrow down the search results based on the provided parameters. Users could also choose to download the extracted data as a CSV file for further analysis or storage.

Next, we tackled the */monthly/extraction and /quarterly/*extraction endpoints. These endpoints would aggregate the daily data into monthly and quarterly summaries, respectively. We used pandas' grouping and aggregation to achieve this, ensuring that the data was correctly grouped and summed based on the specified time intervals. To streamline the data extraction process, we created a */project* endpoint that would automate the extraction and aggregation of data for a predefined list of countries. This endpoint would generate CSV files for each country, containing the tone and popularity data at monthly and quarterly intervals.

As our API grew more complex, we recognized the need for data cleaning and preprocessing. We implemented a /clean and a /EDA ,endpoint that would take the extracted data, handle missing dates, remove duplicates, and ensure that the data was in a consistent format for further analysis. Finally, we added a /mean endpoint that would calculate the mean values for a specified column across multiple CSV files. This would allow users to easily obtain the average tone or popularity for a given time period, providing a high-level overview of the data.

5.3. Challenges and Solutions

As we progressed, we faced several challenges. During the development of our project, we encountered an issue with the GDELT API, due to an excessive number of requests made by one of our team members, our IP address was temporarily banned from accessing the GDELT API. A formal email was drafted to the GDELT team, explaining the circumstances surrounding the excessive requests and emphasizing the academic nature of our project. We acknowledged the unintentional violation of the API usage guidelines and assured the GDELT owner that appropriate measures had been taken to prevent such incidents from occurring in the future.

Throughout the whole development process, we encountered numerous challenges and learned valuable lessons. We had to adapt our approach based on the limitations and quirks of the GDELT API, and we had to be creative in our solutions to overcome obstacles. In the end, we were proud to have built a robust and feature-rich API that could extract, aggregate, clean, and analyze data from the GDELT database. Our API would serve as a valuable tool for researchers, data scientists and analysts

6. Eurostat API

6.1. API Overview

Once again, we chose to build our API using the FastAPI framework, as it had proven to be a reliable and efficient choice for our previous project with the GDELT database. We set up the project structure and installed the necessary dependencies, including the pandas library for data manipulation and the requests library for making HTTP requests.

The Eurostat API functionality involves retrieving socio-economic data from the Eurostat API, The processed data is then grouped by country and gender, and an aggregated "EU" serie is generated by summing the data across countries, and then the data is organized into separate folders for raw, preprocessed, and processed data, as well as the "EU" series.

The API provides endpoints such as */eurostat/* to initiate the data retrieval and processing, */eurostat/head* to preview the raw data, and */eurostat/serie* to access the aggregated "EU" series. These endpoints facilitate further analysis or visualization of the socio-economic indicators.

6.2. Authenticating, Retrieving Socio-Economic Indicators, Data Cleaning and Handling Missing Values

Upon receiving the response from the Eurostat API, we saved the compressed data to a local file and then decompressed it using the gzip module. This decompressed data was then read into a pandas DataFrame for further processing. We realized that the raw data from Eurostat required extensive cleaning and preprocessing. We had to create functions to perform data cleaning, handling missing values, converting time periods, and filtering for specific criteria. Additionally, we had to map the time periods from quarters to specific months, as our analysis required monthly data.

Next, we needed to split the data by country and gender, as our analysis required separate datasets for each combination of country and gender. We created a dictionary of DataFrames, with each key representing a country and the corresponding value being a list of two DataFrames (one for males and one for females). Once the data was cleaned and preprocessed, we saved the resulting DataFrames to separate folders for raw, preprocessed, and processed data. This organization would make it easier to track the different stages of data processing and facilitate future analysis or updates.

To further enhance our analysis, we implemented functions to complete missing time series data and generate a combined series representing the European Union as a whole. This would allow us to analyze trends and patterns not only at the country level but also at the broader regional level.

6.3 Challenges and Solutions

One of the challenges we faced was filtering the data based on specific criteria. We wanted to focus on the employment rates of foreign citizens who were actively employed. To achieve this, we applied multiple filters to the DataFrame, selecting only the relevant rows based on citizenship status and employment status.

There was an issue with the relative paths being used to save the data. The code assumed that the files would be saved in a directory relative to the location of the Eurostat module itself. However, when the Eurostat module was called from a different directory (e.g., the root directory of the project), the files were being saved in an unintended location, specifically in the desktop, resulting in the creation of a "Data" folder outside the intended directory structure.

The issue stemmed from the fact that the relative paths used in the Eurostat module's functions were not consistent with the actual directory from which the module was being called. This discrepancy led to the files being saved in an unintended location, causing confusion and conflicts with the API endpoints responses.

In summary, the problem was related to the handling of relative paths within the Eurostat module's functions, which did not account for the possibility of the module being called from different directories, leading to the creation of folders outside the intended directory structure.

The next phase of the project involved integrating the datasets obtained from Google Trends and GDELT, which were provided in JSON format, into a single unified DataFrame. This data integration process aimed to combine the information from these diverse sources, enabling comprehensive analysis and modeling.

7. DATA INTEGRATION

7.1 Overview

The objective of the data integration phase was to incorporate the datasets from Gtrends and Gdelt, provided in JSON format, into csv data frames that we could use for modeling. After extracting the timestamps and values from the JSON data using regular expressions, each dictionary was converted into a pandas DataFrame. This involved creating an inner dictionary with timestamps as keys and corresponding values. The resulting list of DataFrames, one for each dictionary in the original JSON data, provided a structured and tabular representation of the data. This format facilitated further analysis and manipulation to calculate the means using pandas.

The next critical phase of the project was preparing the data for modeling. Three main objectives were, firstly, to calculate monthly averages for the dataset. These averages would serve as inputs for prediction models. Secondly, calculate the quarterly averages from the data. The third objective was to create a consolidated DataFrame that would serve as the foundation to train a model.

7.2 Merging Data from Multiple Source

One of the key challenges of the project was combining data from different sources, such as Google Trends and GDELT, and preparing them for modeling. These sources had different column formats and names, especially for date and time columns. It was necessary to normalize these data, ensuring that the date and time columns had a consistent format and could be used as indexes. In the next chapter we will explain how this process of combining and transforming data was performed at the API level, where the data was initially obtained and processed. After this data preparation stage, the combined and transformed datasets were ready for use in subsequent modeling and analysis.

8. MODELING

8.1 Model API

To streamline the modeling process and enable efficient deployment,this Model API provides endpoints for various modeling tasks, including model setup, metrics, prediction, and evaluation.

The API encompasses several endpoints . The */setupv0* endpoint handles the initial setup, splitting the historical data into training and testing sets, with the option to apply dimensionality reduction using SelectKBest. The */v1/train* endpoint allows users to train various models, including Random Forest, XGBoost, and LSTM, on the prepared data. Once trained, the models are saved locally for future use. The */v1/metrics* endpoint evaluates the performance of the trained models on the test data, providing metrics.

To make predictions, users can leverage the */v1/predic*t endpoint, which accepts the model type, dimensionality reduction flag, and input data. The API loads the corresponding trained model and returns the predicted value(s). Through this API, we aimed to create a user-friendly and efficient interface for training, evaluating, and deploying machine learning models, streamlining the entire process from data preparation to prediction.

8.2 Testing

After performing data cleaning, preprocessing, and creating lagged features, the team explored dimensionality reduction techniques. Specifically, the team implemented Principal Component Analysis (PCA) to reduce the number of feature columns while retaining most of the variance in the data. The team experimented with different numbers of principal components, visualizing the components and their explained variance ratios. To further analyze the relationship between the original features and the principal components, the team calculated the loadings (coefficients). However, the team encountered challenges with the visualization and recognized the need for alternative dimensionality reduction methods.

The team then attempted to build models using the reduced feature set. The team first tried an LSTM (Long Short-Term Memory) neural network model, which required reshaping the data into a 3D format suitable for the model's input. The team split the data into training and testing sets and trained the LSTM model, evaluating its performance on the test set. While the LSTM model seemed to work in the Jupyter notebook environment, it encountered issues when integrated into the API, potentially due to indexing problems. Further research is needed to resolve these challenges, but the lack of time has been a significant constraint during this project.

Additionally, the team experimented with a Random Forest Regression model using the original feature set, and evaluated its performance using mean squared error (MSE), root mean squared error (RMSE), and the coefficient of determination (R^2). However, the results indicated a high error and a negative R^2 value, suggesting poor model performance.
Despite their efforts, the initial results from the Random Forest Regression model were unsatisfactory. The model exhibited a high mean squared error of 1616948913.4 and a root mean squared error (RMSE) of 40211.30, indicating significant deviations between the predicted and actual values. Moreover, the coefficient of determination (R^2) was -46.587276550682944, which is a negative value.These poor performance metrics highlighted the need for further refinement and exploration of alternative modeling techniques to improve the model's accuracy and predictive capabilities.

In an attempt to improve the model's performance, the team explored the XGBoost. Initially, the team trained an XGBoost Regression model on the original feature set, which yielded a Root Mean Squared Error (RMSE) of 32156.26320094021 and, a coefficient of determination (R^2) of -29.431664070155676. While the RMSE showed a slight improvement compared to the Random Forest model, the negative R^2 value indicated that the model performed worse than simply predicting the mean of the target variable.

Undeterred, the team tried training the XGBoost model on the reduced feature set obtained from the SelectKBest method. This approach resulted in a further reduction in RMSE to 24343.95766166112. However, the R^2 value remained negative at -16.441222392319958, suggesting that the model still underperformed compared to a random prediction.

Regardless of the improvements, the team concluded that the XGBoost models were not adequately capturing the patterns in the data, and further alternative modeling techniques might be necessary to achieve better predictive performance.

9. SUGGESTIONS

Even with the team's diligent efforts in data preprocessing, feature engineering, and experimenting with various machine learning models, the results obtained so far have been unsatisfactory. The consistently negative coefficients of determination (R^2) across different models, such as Random Forest and XGBoost, indicate that the models are performing worse than simply predicting the mean of the target variable. While the Root Mean Squared Errors (RMSE) showed some improvements with certain techniques, the overall predictive performance remains poor. One potential contributing factor could be the lack of sufficient data after removing countries and many rows with missing values, as per the company's instructions due to the lack of time.

The decision to remove data from many countries could be a significant contributing factor to the poor performance of the models. By reducing the available data, the models may not have sufficient information to capture the underlying patterns and relationships between the features and the target variable. Anyway, It is worth considering alternative approaches in the future, if possible. Exploring imputation techniques or adjusting the thresholds for data removal could potentially preserve more data points and improve the representation of underlying patterns and relationships. Moreover, it is advisable to explore alternative data sources or APIs that provide more reliable, consistent, and timely data related to the variables of interest.

While the team developed an API for training machine learning models and making predictions, there is still room for improvement and future enhancements. One area that could be explored further is containerizing the models using Docker. If given more time, the team would have liked to package the trained models and the API into Docker containers, making it easier to deploy and scale the application across different environments. This approach would have fostered a more collaborative and efficient development workflow, allowing for faster iterations and improvements to the machine learning models. Overall, while the team accomplished its primary goals, incorporating Docker containerization would have been a valuable addition, enhancing the scalability, portability, and maintainability of the machine learning models and the API.

10. CONCLUSIONS

As recent graduates from the Data Science bootcamp program at The Bridge School, this project has provided us with invaluable hands-on experience in dealing with real-world challenges associated with working with complex data sources and APIs. We have learned that relying on third-party APIs, especially those provided by large tech companies like Google, can be challenging and uncertain due to strict usage policies, rate limiting, and potential API changes or failures.

Accessing and working with large, complex datasets like Eurostat, Google Trends and GDELT required significant effort from our team in data extraction, transformation, and cleaning, as well as handling various errors and challenges. Building APIs was crucial for us to extract, aggregate, clean, and analyze data from multiple sources. Integrating data from different sources, such as Google Trends, GDELT, and Eurostat, was also a complex and time-consuming process that required us to manipulate data formats, handle missing values, and ensure consistency in date and time columns. Developing a Model API streamlined the modeling process and enabled deployment, providing endpoints for model setup, prediction, and evaluation, making it easier for us to leverage the integrated data for forecasting purposes.

Throughout this project, we not only developed technical skills acquired during the bootcamp but also collaboration, adaptation, and creative problem-solving. These skills were essential as we encountered numerous challenges and had to adjust our approach based on limitations and quirks of the APIs.With that being said, this experience has reinforced the importance of robust data engineering and API knowledge which we acquired during this bootcamp.

We would like to express our sincere gratitude to our lecturers, Carlos Sevilla, Fernando Carrasco and teaching assistant, Josè Prado at The Bridge School, whose guidance and support throughout the Data Science bootcamp have been invaluable in preparing us for this project. Additionally, we extend our appreciation to the staff at Rambee, Juan Arévalo and Beatriz Martín for providing us with this opportunity and for their continuous support, which made this project possible.

Overall, this project has been an invaluable learning experience for us as graduates from The Bridge School, allowing us to apply and showcase the skills we acquired during the Data Science bootcamp while gaining practical experience in tackling real-world challenges in data integration, API development, and machine learning.

APPENDIX

Web Sources

1. Towards Data Science (https://towardsdatascience.com/)
2. GitHub (https://github.com/MajesticKhan/Nowcasting-Python)
3. KDnuggets (https://www.kdnuggets.com/)
4. Analytics Vidhya (https://www.analyticsvidhya.com/)
5. Data Science Central (https://www.datasciencecentral.com/)
6. Machine Learning Mastery (https://machinelearningmastery.com/)
7. Data Camp (https://www.datacamp.com/community/blog)
8. Kaggle (https://www.kaggle.com/code/theoviel/deep-learning-starter-simple-lstm)
9. Data Science Renee (https://datasciencerenee.com/)
10. Data Science Plus (https://datascience.plus/)
11. Data Science Society (https://datasciencesociety.net/)
12. Data Science Stack Exchange (https://datascience.stackexchange.com/)
13. Data Science Weekly Newsletter (https://www.datascienceweekly.org/)
14. Data Science Central (https://www.datasciencecentral.com/)
15. Data Science Dojo (https://datasciencedojo.com/)