**Big Data Platforms Final Project: will TuringBots replace human software developers?**

A number of assistants have been developed recently to augment human software developers and engineers.  Some of the examples include: [GitHub CoPilot XLinks to an external site.](https://github.com/features/preview/copilot-x), [StarCoderLinks to an external site.](https://huggingface.co/blog/starcoder), [Code LlamaLinks to an external site.](https://about.fb.com/news/2023/08/code-llama-ai-for-coding/),  [Amazon Code WhispererLinks to an external site.](https://aws.amazon.com/codewhisperer/) and [ChatGPTLinks to an external site.](https://openai.com/blog/chatgpt).  Can these AI solutions provide meaningful improvement in developers’ productivity and eventually replace human software engineers and data scientists?

In order to conduct this analysis, you have been provided access to GitHub Archive that is stored in Google Cloud Storage, located in gs://msca-bdp-data-open/final\_project\_git folder.  Overall data size is ~ 1.36 TiB.  This data represents content of publicly available GitHub repositories and commit history.

You will see 5 sub-folders, each containing a collection of parquet files.  A single folder can be read into Spark Dataframe:

* **Commits (**gs://msca-bdp-data-open/final\_project\_git/commits**)**: This contains information about the commits made to repositories. Each commit has metadata such as the author, committer, commit date, SHA, parent commit(s), and commit message.
* **Contents (**gs://msca-bdp-data-open/final\_project\_git/contents**)**: Provides the content of the files in the repositories. This is useful if you're looking to analyze source code or documents within repositories.
* **Files (**gs://msca-bdp-data-open/final\_project\_git/files**)**: This contains metadata about the files in the repositories such as the file path, the mode, and the blob ID which links back to the content.
* **Languages (**gs://msca-bdp-data-open/final\_project\_git/languages**)**: Each repository often has code written in one or more languages. This table provides an aggregation of the number of bytes of code for each language in a repository.
* **Licenses (**gs://msca-bdp-data-open/final\_project\_git/licenses**)**: Contains information on the licenses used by repositories.

To complete the project, you will need to perform the following steps:

* Discard irrelevant or obviously erroneous data
  + Most of the variable names should be self-explanatory, however data is deeply nested and will require detailed review in order to select the most appropriate data elements
* Complete thorough EDA to identify which variables you can use to complete your analysis
  + Any poorly populated or duplicate variables should be discarded
* What is the timeline of the data?  Do you see significant peaks and valleys?
  + Do you see any data collection gaps?
  + Do you see any outliers?  Remove obvious outliers before plotting the timeline
  + Do you see any spikes?  Are these spikes caused by real activities / events?
* What are the most popular programming languages on GitHub?
  + Did the trend of most popular programming languages change over time?
* What is the distribution of licenses across GitHub repositories?
  + Any certain programming languages that are more likely to be associated with a particular license?
* What can you tell about the most popular and most rapidly growing repositories?
  + Is there certain technology that is driving popularity or explosive growth?
  + Are these associated with [Big TechLinks to an external site.](https://en.wikipedia.org/wiki/Big_Tech), who are open sourcing the technology?
  + Are there any technological breakthroughs that are driving this brisk adoption?
* Identify what technologies are most frequently associated with Data Science or AI projects
  + Did these technologies change over time?
* What are the most frequent reasons for committing into GitHub repositories?
  + Is this new technology development, bug fix, etc.
* Identify the most prolific / influential Committers
  + By commit volume
  + Visualize the distribution of these commits
* How unique are the “subject” and “message” values?
  + Are they mostly unique? Or are people usually just copy-pasting the same text?
  + You can use LSH to measure uniqueness / similarity
  + Visualize “subject” and “message” duplication **across all**programming languages
  + Visualize “subject” and “message” duplication **for each of the top 5** programming languages
  + Please note: this is not a topic modeling (LDA / LSA) – but text similarity analysis

Submission guidance:

* Please submit your actual program codes (Jupyter notebooks) along with your PowerPoint
* Please ensure your PowerPoint presentation (in PPTX or PDF format) is submitted to the course module as-is (not zipped).  Otherwise, we are unable to use Canvas SpeedGrader.
  + Executive Summary
  + Methodology and source data overview
  + Conclusions and actionable recommendations
  + Roughly 8-12 pages is reasonable for this kind of project but there are no strict restrictions.
  + The presentation should look professional – not a collection of screenshots from your analytical software
  + On your slides you will want to provide:
* The presentation slides:
  + Should be self-sufficient and after reading them, there should not be any need to read the notebook (we are still asking you to provide the notebooks as a proof or work though)
  + Should clearly answer all the questions and the answers should be supported with the plots/tables/numbers produced in the notebook based on the actual data.
  + Should contain the **right**amount of supporting material for each question, putting too much supporting material is as bad as putting too little.  With too much - you would not be able to keep the audience attention and your presentation would be a mess, with too little - your statements would not look convincing
  + Everything should be clear, logical, well organized, as simple as possible; use proper English and run spell check
  + All the plots should be of production quality and easily readable.  We will be deducting points for fuzzy plots, untitled plots, unreadable labels, overlapping labels, etc.  If you formatting somehow gets corrupted when you put your slides into Canvas (sometimes it happens), it is your responsibility to fix formatting and re-upload your presentation.  For example, try saving it in some other format like PDF, HTML
  + All statements, conclusions, recommendations should be 100% supported by the data analysis

Some suggested steps:

1. GitHub data comes in nested format. You will need to understand the structure / fields in order to parse each element correctly and to assign them to different variables.
2. It is perfectly fine to run *some of* your analysis on a sample of the GitHub, as long as you can generalize your findings to entire data set
3. The data will be noisy, you are encouraged to eliminate obviously erroneous records.  You can use a combination of exploratory data analysis and web search to come up with best set criteria.
4. By doing this pre-processing, eliminating unnecessary fields and bad data, you can significantly reduce the data volume and simplifying the analysis.  Think about the right format to store your intermediate results in the Google Cloud Storage bucket, to maximize the speed / efficiency of consecutive operations (i.e. Parquet, CSV, JSON, etc.) based on what you have learned in this class.
   * This is Big Data, please do not try to cache it all in Spark memory - your environment will freeze and crush and we will be deducting points for such behavior
   * Before submitting the jobs on entire Big Data, dry run all your notebooks on small cached samples to ensure you don't have any Cartesian products, not losing records, don't have infinite loops, bad joins, etc.
   * Do not create a single notebook that runs everything from reading data to conducting an analysis.  In fact, I would very much recommend saving intermediate steps of your analysis, so you can continue without constantly going back to Big Data.  The filtered results file should be much smaller and will be processed quickly

Project execution options:

**Option 1:** setting-up your own Dataproc Instance on GCP.  This is a preferred option, as you will learn how to work with a variety of cloud services on Google Cloud, including setting-up and managing your own cluster, storage and watching for resource utilization and cost.  You should be able to use Google credits to cover most of the cloud usage costs.  In order to run the analysis, you will need to set-up Interactive Session Templates

and then read data directly from gs://msca-bdp-data-open/final\_project\_git bucket.  Some useful links:

* [GCP Dataproc ServiceLinks to an external site.](https://docs.google.com/document/d/1lSyxhH0_jUwg6sJMiP2J4XtL6y60QFjRAHvespT1yUw/edit?usp=sharing)
* [Starter Jupyter Notebook](https://canvas.uchicago.edu/courses/60116/files/12509339?wrap=1)[Download Starter Jupyter Notebook](https://canvas.uchicago.edu/courses/60116/files/12509339/download?download_frd=1)

**Option 2:** if you do not want to set-up Dataproc Instance on GCP  (for whatever reasons), you will be able to use my Dataproc instance "adsp-bdp-serverless-students-1".  If you are running on my Dataproc cluster, you can still read data directly from gs://msca-bdp-data-open/final\_project\_gitbucket using Spark.  This cluster is provided on as-is basis (without any kind of support or uptime guarantee), therefore you must back-up your notebooks frequently (on your laptops, in Box, in Google Drive, etc.)

* [Working on adsp-bdp-serverless-students-1Links to an external site.](https://docs.google.com/document/d/1Dev22rC22Q6bgf3PRYwKWcMbcsAD4EuaFTQoI8YeY2U/edit?usp=sharing)
* [Starter Jupyter Notebook](https://canvas.uchicago.edu/courses/60116/files/12509339?wrap=1)[Download Starter Jupyter Notebook](https://canvas.uchicago.edu/courses/60116/files/12509339/download?download_frd=1)

Other notes: these are Big files, so please don’t wait until the last day to start working on them!  Start early to avoid being affected by any performance issues.

**Grading Rubric:**

|  |  |
| --- | --- |
| **Rubric** | **Points** |
| Executive Summary with meaningful insights | 10 |
| Methodology and source data overview | 10 |
| Data clean-up and filtering | 10 |
| EDA and extensive usage of available variables | 10 |
| Timeline analysis | 10 |
| Programming Language and License analysis | 10 |
| Most popular technology and repositories analysis | 10 |
| Subject and Message uniqueness analysis | 10 |
| Conclusions and actionable recommendations | 20 |
| **Total Grade** | 100 |