

10 Academy Intensive Training Week-0 Challenge Document

Pre-training Assessment   
Date: 08 Apr - 12 Apr 2023

# Challenge Overview

This week’s challenge focuses on the sentiment, topic, and reporting correlation among various global media agencies. The challenge aims to evaluate candidates for the 12-week training program in Data Engineering (DE), Generative AI Engineer (AIE), and Machine Learning Engineering (MLE). It provides applicants with a real-world experience of tasks in these domains and helps 10 Academy select suitable candidates

Applicants who proceed to the next level by attaining an excellent performance in this week's challenge will have a clear picture of the required discipline, resilience, proactivity, talent diversity, and other essential elements of the 10 Academy training. Those that can not make it to the limited spots available will gain a clear understanding of the direction they should improve to prepare for AIE, DE & MLE job positions in the future. Everyone will gain project experience to showcase in their professional profile.

This week, therefore, is a **win-win** for everyone. We advise you to put your best effort to complete as many tasks as possible. We know that the number of tasks you are required to complete are a lot, and you **will not** have time to build intuition or to be comfortable with the new concepts and skills you are exposed to with this week’s challenge. Please **note that** building a deeper understanding is not the purpose for this week's project. Moreover, you may have never done or attempted to do some of the tasks before this training. If you are confused and **overwhelmed**, know that it **is expected**.

The tutors, community managers, and all other teams are there to support you as best as they can. Be proactive to ask questions, provide resources that may help others, and above all **persist**!

# Dataset Overview: Importance of News Headline Analysis

The data for this week's challenge is [Global News Dataset](https://www.kaggle.com/datasets/everydaycodings/global-news-dataset). This dataset comprises news articles collected over the past few months using the NewsAPI

You can download the global news data [here](https://drive.google.com/file/d/1ju5i8qZ---K1FYEV5Co0PYvPhW9t8gRb/view?usp=drive_link). The structure of the data is as follows

data.csv:

* **article\_id:** Unique article id
* **source\_id**
* **source\_name:** Source name
* **author:** The author of the article
* **title:** The headline or title of the article.
* **description:** A description or snippet from the article.
* **url:** The direct URL to the article.
* **url\_to\_image**: The URL to a relevant image for the article.
* **published\_at:** The date and time that the article was published, in UTC
* **content:** The unformatted content of the article, where available. This is truncated to 200 chars
* **category:** Search query used to fetch data
* **article:** Full content of that article
* **title\_sentiment:** Sentiment of the title

You can download the domain’s location (the country where the news media headquarter that owns the domain/website is located) data [here](https://drive.google.com/file/d/1JCf4CeqNMfSBvMDrghDs744KnMoYWxxz/view?usp=drive_link). The structure is as follows

domains\_location.csv:

* **SourceCommonName:** Common Domain Name (bbc.co.uk, cnn.com)
* **location:** Country short code (US, UK, CA)
* **Country:** Country name

You can download the 1million global website traffic data [here](https://drive.google.com/file/d/1pHcKCVOrXtjtjNcazBpzKuLvnqPRCqlI/view?usp=drive_link). The structure is as follows

traffic\_data.csv:

* **GlobalRank:** Rank of the domain globally
* **TldRank:** Rank of the TLD (Top-Level Domain [.com, .org…..etc]) among other similar TLD (Top-Level Domain [.com, .org…..etc])
* **Domain:**
* **TLD:** TLD (Top-Level Domain [.com, .org…..etc])
* **RefSubNets:** The number of Referring Subnets found for this domain in the [Fresh Index.](https://raven.zendesk.com/hc/en-us/articles/202108474-What-is-the-difference-between-Fresh-Index-and-Historic-Index#:~:text=The%20Fresh%20Index%20is%20updated,found%20in%20the%20Historic%20Index.)
* **RefIPs:** The number of Referring IPs found for this domain in the [Fresh Index](https://raven.zendesk.com/hc/en-us/articles/202108474-What-is-the-difference-between-Fresh-Index-and-Historic-Index#:~:text=The%20Fresh%20Index%20is%20updated,found%20in%20the%20Historic%20Index.).
* **IDN\_Domain:** [Internationalized Domain Name](https://en.wikipedia.org/wiki/Internationalized_domain_name)
* **IDN\_TLD:** [Internationalized Domain Name Top-Level Domain.](https://en.wikipedia.org/wiki/Internationalized_country_code_top-level_domain#:~:text=IDN%20ccTLDs%20are%20specially%20encoded,system%2C%20such%20as%20Chinese%20characters.)
* **PrevGlobalRank:** Previous Global Rank.
* **PrevTldRank:** Previous TLD Rank.
* **PrevRefSubNets:** Previous number of referring subnets.
* **PrevRefIPs:** Previous number of referring IPs.

# Data Handling Instructions

* Do not push the data to GitHub. Add the data path to your .gitignore file to prevent accidental uploads.
* Ensure compliance with data privacy and confidentiality standards when working with the dataset
* The data for the project is in the public domain, so feel free to share your exciting findings in your preferred social media.

# Guidance for Applicants

* **Win-Win Situation:** This week's challenge is designed for mutual benefit. Excellent performance can lead to selection for the training program, while others gain valuable insights for future improvement.
* **Task Completion:** Focus on completing as many tasks as possible. The goal is exposure to various aspects, not necessarily deep understanding.
* **Support System:** The 10 Academy team, including tutors and community managers, is here to support you. Don't hesitate to ask questions, share resources, and persist in your efforts.

# Week's Topics Covered

1. **Python and Javascript Programming:**
   * Task-specific programming assignments.
2. **GitHub Commands:**
   * Continuous committing and repository management.
3. **Frameworks, Processes, and Workflows:**
   * Utilising CRISP-DM methodology.
4. **Databases:**
   * Working with SQL and NoSQL databases.
5. **Data Understanding and Exploration:**
   * Applying exploratory data analysis techniques.
6. **CI/CD:**
   * Understanding continuous integration and continuous deployment.
7. **MLOps:**
   * ML system process design.
8. **Modelling:**
   * Topic modelling and sentiment analysis.
9. **Web App Development:**
   * Building dashboards.
10. **Full Stack Development:**
    * Exposure to full-stack concepts.
11. **Server and Serverless Deployment:**
    * Deployment architecture and strategies.

# Note to Applicants

* **Overwhelm is Expected:** The challenge is designed to cover a broad range of topics. Feeling confused or overwhelmed is expected, and support is available.
* **Proactivity is Key:** Be proactive in asking questions, sharing resources, and collaborating with others.
* **Persistence is Valued:** Persistence in tackling challenges is highly valued in this learning environment.

# Starter python package

We provide [a starter python package](https://github.com/10xac/week0_starter_network_analysis) that you may use to start working on the challenge. It has (at the time of writing this document) the following structure

Network\_analysis

├── .vscode

│ └── settings.json

├── .github

│ └── workflows

│ ├── flake8\_check.yml

│ ├── unittests.yml

│ └── docstring\_tests.yml

├── .gitignore

├── .flake8

├── .pre-commit-config.yaml

├── setup.cfg

├── Makefile

├── pyproject.toml

├── requirements.txt

├── style\_guide.md

├── README.md

├── view\_tree.py

├── docs

├── notebooks

│ ├── parse\_slack\_data.ipynb

│ └── README.md

├── tests

│ ├── \_\_init\_\_.py

└── src

├── config.py

├── \_\_init\_\_.py

├── utils.py

└── loader.py

Here's a breakdown of the different components (**WARNING: we may add, edit or remove some files without updating this document. Also note that, code or config files may not be working - you should fix it by yourself or remove it if you don’t need it**):

**.flake8:** Configuration file for the Flake8 tool, which checks for code style and quality.

**requirements.txt:** Lists the dependencies required to run the project. This file is commonly used with tools like pip to install project dependencies.

**.pre-commit-config.yaml:** Configuration file for pre-commit hooks, which are checks that run before commits are allowed.

**Makefile:** A Makefile for defining and running project tasks. It's a convenient way to encapsulate complex or repetitive tasks.

**pyproject.toml:** Configuration file for build tools and project metadata. It's commonly used for projects using poetry.

**tests/:** A directory for storing unit tests. The \_\_init\_\_.py file indicates that this directory should be treated as a Python package.

**docs/:** A directory to store project documentation.

**style\_guide.md:** A markdown file containing the style guide for the project. Good documentation for coding styles can be useful for maintaining consistency.

**README.md:** A README file containing essential information about the project. It serves as the first point of contact for anyone exploring the project.

**.gitignore:** A file specifying which files and directories to ignore when pushing to version control (Git). This often includes files that are generated during runtime, build artifacts, and sensitive information.

**.github/workflows/:** GitHub Actions workflow files for automating tasks like code formatting, unit testing, and docstring testing.

**setup.cfg:** Configuration file for the setuptools package, often used for packaging and distribution.

**.vscode/settings.json:** Configuration settings for Visual Studio Code, an integrated development environment (IDE).

**notebooks/:** A directory for storing Jupyter notebooks, which are often used for exploratory data analysis (EDA) and documenting code.

**view\_tree.py:** A Python script or module for viewing the project tree structure.

**src/:** A directory containing the source code of the project.

**config.py:** Configuration file for the project.

**\_\_init\_\_.py:** Marks the src directory as a Python package.

**utils.py:** A module containing utility functions.

loader.py: A module for loading Slack data

Feel free to adopt or alter the structure as you see fit.

# 

# Work Plan

The following is the summary of the 5 tasks you are expected to complete this week. **A step-by-step guideline will be provided during the week.**

In all of the tasks below the following are required performance indicators

* Ability to help others.
* Efficient and modular coding technique.
* Proactivity to self-learn - sharing references.

## Task 1:

### Git and GitHub

* Tasks:
  + Setting up Python environment
  + Git version control
  + CI/CD
* Key Performance Indicators (KPIs):
  + Dev Environment Setup.
  + Relevant skill in the area demonstrated.

### Project Planning - EDA & Stats

* Tasks:
  + Understanding CRISP-DM Framework
  + Data Understanding
  + Exploratory Data Analysis (EDA)
  + Statistical thinking
* KPIs:
  + Proactivity to self-learn - sharing references.
  + EDA techniques to understand data and discover insights,
  + Demonstrating Stats understanding by using suitable statistical distributions and plots to provide evidence to actionable insights gained from EDA.

**Minimum Essential To Do (if you do it better, don’t worry about sticking with this guideline) :**

* Create a github repository that you will be using to host all the code for this week. You can name it like ***news\_correlation\_10ac\_week0/***
* Create at least one new branch called ”task-1” for your analysis of day 1
  + Commit your work at least three times a day with descriptive commit message
* If you are using the starter code (which is prepared for a slack data analysis project), make sure to change every occurrence of ***slack*** (filename, function names, class names, etc.) to ***news***. Moreover, make sure to update the code to be useful for the news data. For example
  + Change *SlackDataLoader* to *NewsDataLoader* so that you can load the current data
  + Move all non-plotting functions from *notebooks/parse\_slack\_data.ipynb* into *src/loader.py* and *src/utils.py* such that in the notebook you use the *NewsDataLoader* from *src/loader.py* and functions from *src/utils.py* for all your data loading needs.
  + Write new methods and new classes to help you explore the data better
* Perform EDA analysis to answer the following questions
  + Who are the top and bottom 10
    - Websites that have the largest count of news articles
    - Websites with the highest numbers of visitors traffic
    - Countries with the highest number of news media organisations (represented by domains in the data)
    - Countries that have many articles written about them - the content of the news is about that country
    - Websites that reported (the news content) about Africa, US, China, EU, Russia, Ukraine, Middle East? Note that you will need to group countries together to form the African, EU, and Middle East continents/regions.
    - Websites with the highest count of positive, neutral, and negative sentiment? To do this you will need to group the data by website domain and apply descriptive statistics such as mean, median, and variance
      * compare the impact of using mean/average and median,
      * check the distribution of sentiments for a particular domain (select the top 10 domains by visitors traffic) amount of news reported or vs the global news sentiment distribution)
  + Compare the content metadata across sites
    - How similar are the raw message lengths across sites? Check the distribution among sites
    - How similar are the number of words in the title across sites? Check the distribution among sites
  + What is the impact of frequent news reporting and sentiment to the website’s global ranking?
    - Do a 2D scatter plot where x-axis is the total number of reports by a website, y-axis is the global ranking of the site, and the color representing average/median sentiment.

## Task 2:

### Data Science Component Building

* Tasks:
  + MLOps components
  + Analysis pipeline designs
  + Time Series Analysis
  + Classification of Headlines into the following tags
    - Breaking News
    - Politics
    - World News
    - Business/Finance
    - Technology
    - Science
    - Health
    - Entertainment
    - Sports
    - Environment
    - Crime
    - Education
    - Weather
    - Other
  + Topic Modelling & Sentiment Analysis.
  + Predictive analysis and modelling
  + Network analysis
* KPIs:
  + Understanding DS components.
  + Pipeline and process-centric thinking.
  + Ability to understand basic requirements and translate them into codes.
  + Demonstrating ML skills and knowledge

### ML Engineering

* Tasks:
  + Understanding MLOps components
    - Feature Store
    - Model versioning
    - Model monitoring,
  + Unit Testing
  + CI Implementation with Github Actions
  + Dockerization
  + Building python packages.
* KPIs:
  + Understanding of Software Engineering concepts.
  + Understanding of MLOps concepts
  + Demonstrating Code, Data, and Model CI/CD skill

**Minimum Essential To Do:**

* Merge your day 1 branches (e.g. task-1 branch) to the main branch
* From the main branch, create at least one new branch called “task-2” to develop your analysis of Task 2
* Create at least one unit test into tests/ folder
* Summarise the different MLOps components and their use
* Answer the following questions
  + Perform Keyword extraction/modelling using [TF-IDF](https://kavita-ganesan.com/python-keyword-extraction/), [KeyBert](https://github.com/MaartenGr/KeyBERT), [YAKE](https://liaad.github.io/yake/), or [other similar algorithms](https://towardsdatascience.com/keyword-extraction-python-tf-idf-textrank-topicrank-yake-bert-7405d51cd839)
    - How similar are keywords in the headline/title compared to keywords in the news body across sites? To do this you may need to perform TF-IDF or similar keyword identification
  + Perform topic modelling. Checkout [ref1](https://github.com/MaartenGr/BERTopic) [ref2](https://github.com/maha-prathamesh/Topic-Modeling-on-News-Articles) [ref3](https://github.com/crypter70/Topic-Extraction)
    - categorise the title/content into a known set of topic categories
    - Analyse topics and trends
      * Which websites reported the most diverse topics?
      * Analyse the topic trends. For example, plot a 2D scatter plot such that x-axis is date, y-axis is the topics, and the color represents the count of the topic in that particular date. What are the observed trends?
  + Model the events that the news articles are written about (this is the most challenging part of this project). You may follow similar methodology as [this one](https://github.com/parkervg/news-article-clustering) or [this one](https://github.com/SU-NLP/Event-Clustering-within-News-Articles/tree/master)
    - you will require to associate/model the event that the article is covering. For example 500 of the news articles by 60 news media could be about the event of a global disruption of the Meta company Apps such as Instagram).
    - Cluster news articles by events
      * How many events are covered in the data?
      * Analyse which news sites report events the earliest?
      * Which events have the highest reporting?
      * What is the correlation between news sites reporting events?
* Version your ML models and their artefacts using MLFlow

## Task 3:

### PostgreSQL

* Tasks:
  + Database Technologies (SQL vs NOSQL),
  + Database schema design.
  + Using PostgreSQL as Feature Store and load into it ML model features
  + Loading data into PostgresSQL
* KPIs:
  + Schema design for SQL
  + Demonstrating SQL skills

**Minimum Essential To Do:**

* Merge the necessary branches from task-2 into the main branch using a Pull Request (PR)
* Create at least 1-branch called task-3 to develop your work for Task 3
* Design a PostgreSQL table schema to store your ML features - the data used to create the Topic & Event Models
  + Create a schema using [dbdiagram.io](https://dbdiagram.io/home/) or [DbSchema](https://dbschema.com/) to host the data features you prepared in Task 1 & 2
  + Create your tables using python
  + Load all the relevant features into the database

## Task 4: (Optional)

### Dashboards

* Tasks:
  + Streamlit-based dashboard design
  + Programming a simple React app or component (streamlit)
* KPIs
  + Wireframing tools familiarity.
  + Dashboard Design.
  + Design thinking.
  + Building dashboards with Streamlit.
  + Fullstack concepts understanding using React and Python

**Minimum Essential To Do:**

* Merge the necessary branches from task-3 into the main branch using a Pull Request (PR)
* Create at least 1-branch called task-4 to develop your work for Task 4
* Design a dashboard to show results from Task 1 & 2 in Figma or similar tools
* Implement the dashboard you designed using Streamlit
* Write a basic React App or Streamlit component to demonstrate your understanding of Fullstack (frontend using React, database using SQL, and backend using Python) development
  + Streamlit components are built using React - see [this reference](https://streamlit-components-tutorial.netlify.app/introduction/streamlit-react-python/)

## Task 5: (Optional)

### Deployment

* Tasks:
  + **Streamlit or React App**
    - Writing at least one github action for continuous deployment (CD)
  + **Python Backend**
    - Writing at least one github action for continuous deployment (CD)
    - Configure environment variables for database credentials and other sensitive information.
  + **PostgreSQL Database**
    - Initialize the PostgreSQL database by running the necessary setup scripts. Typically involves creating the initial database cluster and superuser.
  + Understanding deployment
  + Understanding Kubernetes (Docker Swarm or other distributed docker container based deployments)
* KPIs
  + Understanding of deployment technologies

**Minimum Essential To Do:**

* Merge the necessary branches from task-4 into the main branch using a Pull Request (PR)
* Create at least 1-branch called task-5 to develop your work for Task 5
* Create at least one github action file in the .github/workflow that will package your code as docker image and push it to docker hub.
  + In the future you may add a step in your github actions to trigger serverless cloud deployment e.g. in AWS Lambda or other docker based deployment infrastructure
* Write a summary of your understanding of the different deployment tech stack.

### 

# Deliverables:

#### **Monday:**

* Your public Github repository link
  + You should make sure that all of your code related work for this week is going to be aggregated in this repository. We will analyse your repository link to evaluate your skills and work progress.

#### **Tuesday:**

* 1-page summary of the project as you understand it.
* Github link to your main branch
  + You should create at least one new branch called `task-1` for task 1 and merge your latest work with main branch.

#### **Wednesday:**

#### **Thursday:**

* 3-pages report on insights and work done in the previous days.

#### **Friday: Changed to Saturday 8:00 PM (UTC)**

* Link to deployed dashboard (or detailed screenshots). (Optional but a plus if you submit)
* GitHub Link to your main branch
* Final report - covering all week-0 work.
  + Write it in the format that you could post it as a Blog in Medium.

### 

### **Other Considerations:**

* **Documentation:** Encourage detailed documentation in code and report writing.
* **Collaboration:** Emphasise collaboration through Github issues and projects.
* **Communication**: Regular check-ins, Q&A sessions, and a supportive community atmosphere.
* **Flexibility:** Acknowledge potential challenges and encourage proactive communication.
* **Professionalism:** Emphasise work ethics and professional behaviour.
* **Time Management:** Stress the importance of punctuality and managing time effectively.

# Tutorials Schedule

In the following, the colour **purple** indicates morning sessions, and non-purple indicates afternoon sessions.

* Day 1:
  + Introduction to the Challenge
  + Python Environment Setup, Git and Github, CI/CD
  + Project Planning & EDA - Data Science Workflow using CRISP-DM and EDA techniques
* Day 2:
  + Data Science Component Building (Architecture Designs, Wifeframing, logic flow)
  + Topic Modeling, Sentiment Analysis, Time Series Analysis
  + ML Engineering components
* Day 3:
* Day 4:
  + Working with SQL
  + Database schema design
* Day 5:
* Building Dashboards using Streamlit
* Introduction to Fullstack programming using React and Python

# Leaderboard Updates

* Wednesday end of day
* Friday end of day

# 

# References

Highly recommended references are highlighted in **bold.**

Relevant academic papers

* [The drivers of global news spreading patterns | Scientific Reports (nature.com)](https://www.nature.com/articles/s41598-024-52076-6)

CRISP-DM, topic modelling, and sentiment analysis:

* <https://www.datascience-pm.com/crisp-dm-2/>
* [**Gensim Topic Modeling - A Guide to Building Best LDA models (machinelearningplus.com)**](https://www.machinelearningplus.com/topic-modeling-gensim-python/)
* [Beginners Guide to Topic Modeling in Python and Feature Selection (analyticsvidhya.com)](https://www.analyticsvidhya.com/blog/2016/08/beginners-guide-to-topic-modeling-in-python/)

Database & Dashboard design and implementation

* [Get started - Streamlit Docs](https://docs.streamlit.io/library/get-started)
* [Turn Python Scripts into Beautiful ML Tools | by Adrien Treuille | Towards Data Science](https://towardsdatascience.com/coding-ml-tools-like-you-code-ml-models-ddba3357eace)
* [Streamlit 101: An in-depth introduction | by Shail Deliwala | Towards Data Science](https://towardsdatascience.com/streamlit-101-an-in-depth-introduction-fc8aad9492f2)
* [StreamLit - Data Scientists tool for developing web apps | by INSAID | Medium](https://insaid.medium.com/streamlit-data-scientists-tool-for-developing-web-apps-a4c23d98798f)

Python Programming:

* [Object Oriented Programming](https://realpython.com/python3-object-oriented-programming/#:~:text=Object-oriented%20programming%20(OOP)%20is%20a%20method%20of%20structuring,the%20components%20of%20a%)
* [Python Courses and Tutorials: Online and On Site (python-course.eu)](https://python-course.eu/)

Data Engineering

* [What is Data Engineer: Role Description, Skills, and Background | AltexSoft](https://www.altexsoft.com/blog/what-is-data-engineer-role-skills/)

Version control – Git

* [What is version control | Atlassian](https://www.atlassian.com/git/tutorials/what-is-version-control)
* [Learn Git branching -- interactive way to learn Git](https://learngitbranching.js.org/)
* [Git with large files](https://stackoverflow.com/a/19494211)
* [Which files to not track and how to not track them? | Atlassian](https://www.atlassian.com/git/tutorials/saving-changes/gitignore)
* [.gitignore docs](https://git-scm.com/docs/gitignore)
* [Conventional commits -- lightweight convention on top of commit messages.](https://www.conventionalcommits.org/en/v1.0.0/)

CI/CD

* [**What is Continuous Integration | Atlassian**](https://www.atlassian.com/continuous-delivery/continuous-integration)
* [DevOps Pipeline | Atlassian](https://www.atlassian.com/devops/devops-tools/devops-pipeline)
* [7 Popular Open Source CI/CD Tools - DevOps.com](https://devops.com/7-popular-open-source-ci-cd-tools/)
* [**Setting up a CI/CD pipeline on Github**](https://blog.travis-ci.com/2019-05-30-setting-up-a-ci-cd-process-on-github)

MLOps

* [**MLOps: Continuous delivery and automation pipelines in machine learning**](https://cloud.google.com/architecture/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning)
* [MLwatcher - monitoring model performanc](https://www.anodot.com/blog/mlwatcher/)

Python Testing

* <https://machinelearningmastery.com/a-gentle-introduction-to-unit-testing-in-python/>
* <https://docs.python-guide.org/writing/tests/>
* <https://realpython.com/python-testing/>