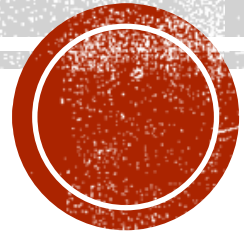
NEWS ARTICLES CLASSIFIER

P Manish

Vedha Krishna Velthapu

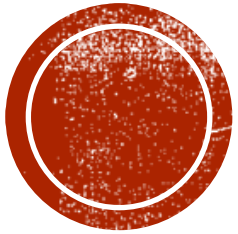


CONTENTS

- Planning Document
- Milestone 1 Report
- Milestone 2 Report
- Milestone 3 Report
- Milestone 4 Report
- Jmeter stress test screenshots
- Setup steps

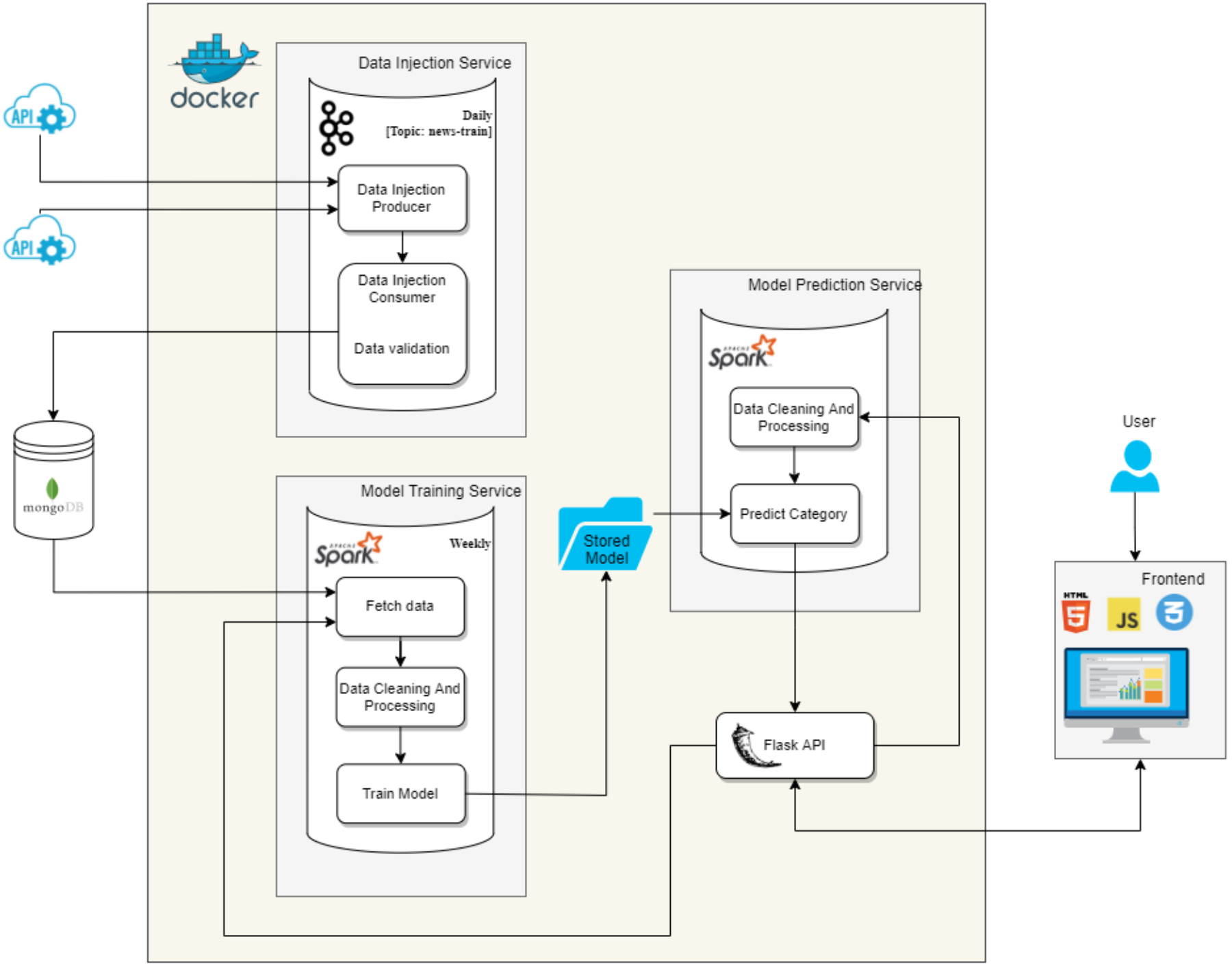
PLANNING DOCUMENT

https://docs.google.com/document/d/12yBr9iS_2Y7TUdLg-8Pu-fC3epiNBLcLnXRKi2ezRB4/edit?usp=sharing



MILESTONE 1

- Architectural Design
- Environment details
- Input and Output
- Challenges


docker
Data Injection Service
Daily
[Topic: news-train]
Data Injection Producer
Data Injection Consumer
Data validation
API
API
mongoDB
Model Training Service
Weekly
Fetch data
Data Cleaning And Processing
Train Model
Stored Model
Model Prediction Service
Data Cleaning And Processing
Predict Category
Flask API
User
Frontend
HTML
JS

ENVIRONMENT DETAILS

- **Database:** Free MongoDB database on cloud
- **Streaming:** Apache Kafka using queues

INPUT

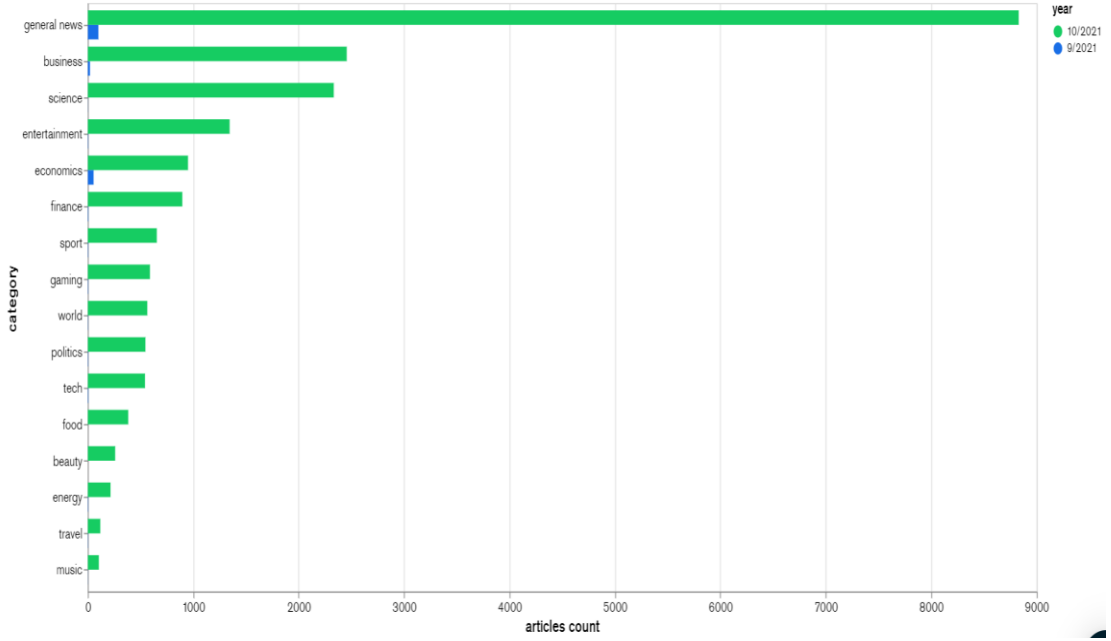
Input

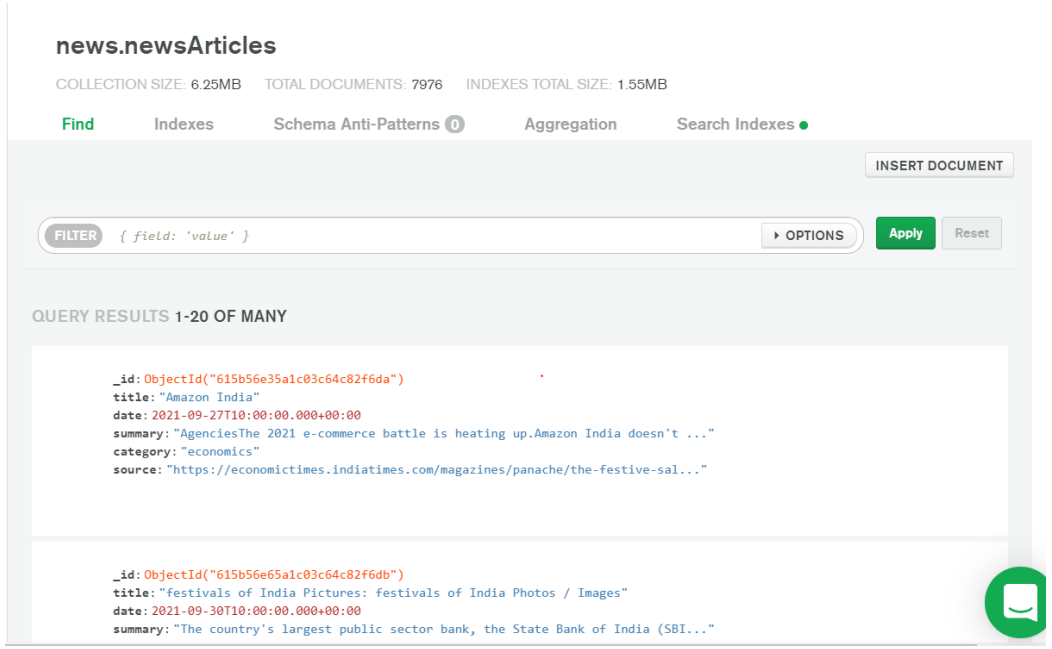
We get our data from 2 APIs

- Free news API
- Newscatcher API

Input Processed

- We validate that all the fields we need from the API response (published-date, topic/category, title, summary, and source).
- Both the APIs we used had a *news* category which we changed to *general news* before the article is stored in the database to reduce some ambiguity.
- The topic we used in Kafka is *news-train*
- We used a Kafka queue to pass the data periodically from the producer to the consumer.
- For the time being the producer fetches articles daily.





OUTPUT

- Structured data stored in our database with the following columns:
  - **Title:** News article's title
  - **Summary**: News article's summary
  - **Category**: News article's category, i.e sports, health
  - **Source:** News article's URL link
  - **Date:** News article's published date

CHALLENGES

Lack of understanding on Kafka and Kafka queues

Solution: Splitting the task so 1 person focus solely on Kafka until it is set up, instead of 2 people working on it together helped us. We managed to finish the other tasks along with Kafka in time.

Not saving duplicates on MongoDB and both the team members are not very experienced with MongoDB

Solution: We managed to created a unique index on the title, so any article with repeating titles will fail to save.

CHALLENGES

Finding a good source to get news articles from as most of the sources/APIs were paid

Solution: After a lot of research, we found 1 API we can use and the free news API provided, worked for us.

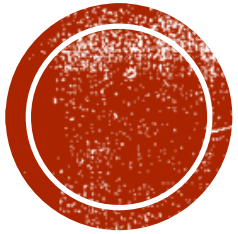
Coordinating and working together between different time zones

Solution: Communicating regularly and meeting every 2-3 days to have alignment on our tasks.

CHALLENGES

Saving large amounts of documents to the database was taking long and with bulk insert, if a document fails all documents after that won't save. With the unique title constraint if a document failed all the documents after that won't save

Solution: We decided to insert all the documents individually in a try-catch block. If there are any exceptions while saving the documents, it will not stop the application from running. Even though this is a bit slower, we would rather have a slow write than a lot of documents failing to save to the database.



MILESTONE 2

- Environment details
- Input and Output
- Challenges

ENVIRONMENT DETAILS

- **Database:** Free MongoDB database on cloud
- **Streaming:** Apache Kafka using queues
- **Data cleaning (to remove non-English words):** nltk
- **Data processing:** PySpark (Tokenizer, CountVectorizer, IDF)

INPUT

Input:

- The data from our database what we collected from milestone 1.

Input Processed:

1. Fetched data from database
2. Created a spark data frame
3. Concatenated the title and summary
4. Removed all the non-English words, spaces and special characters from the text. We used nltk and stop words for this
5. Pipeline with the following steps:
    1. Tokenize the title and summary
    2. Applied a CountVectorizer to convert the list of tokens above to vectors of token counts
    3. Applied term frequency–inverse document frequency (TF-IDF) to evaluate how relevant a words are in a set of documents.
6. Stored the pipeline so it can used by the predicting service to process the text input.

OUTPUT

- **Independent values (features):** The output is a matrix of token counts
- **Dependent values (target):** A list of integers that maps to categories
- Stored pipeline to be used by the predicting service

NOTE: The categories we already cleaned and validated before inserting into the database as we knew this was an important value.

OUTPUT

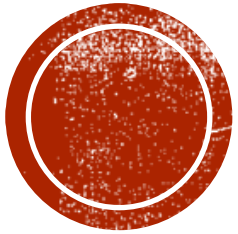
Independent values (features): The output is a matrix of token counts

Dependent values (target): A list of integers that maps to categories

CHALLENGES

Getting data from our mongoDb database using pyspark

Solution: Our mentor showed us a few examples which helped us get around the issue. Instead of fetching data using pyspark, we got data using pymongo and created a spark data frame with that data to process it.



MILESTONE 3

- Environment details
- Input and Output
- Challenges

ENVIRONMENT DETAILS

- **Database:** Free MongoDB database on cloud
- **Streaming:** Apache Kafka using queues
- **Data cleaning (to remove non-English words):** nltk
- **Data processing:** PySpark (Tokenizer, CountVectorizer, IDF)
- **Model training:** PySpark - Naïve bayes, random forest classifier, logistic regression with OneVsRest (Picked the model with the best accuracy)

INPUT AND OUTPUT

Input:

- Cleaned set of count vector of articles and integer as a category that maps to a string category from milestone 2.
- Stored pipeline to be used by the predicting service

Input Processed:

- Split the data into train and test using a build in spark data frame function, randomSplit().
- We used 3 models, NaiveBayes, RandomForestClassifier and LogisticRegression(1-to-rest) and picked the best model based on it's accuracy.
- We also used pySpark's CrossValidator and passed in different parameters to train the model and picked the best one for each.
- We finally saved the model using pySpark's save function.

Output:

- A folder with the saved model. This model can now be used by the predicting service.
- Stored pipeline to be used by the predicting service

CHALLENGES

Saving the model

Solution: We tried to save the model using mlflow and pickle, but both didn't work. We eventually settled to use pyspark's save and load to carry on with the next milestone

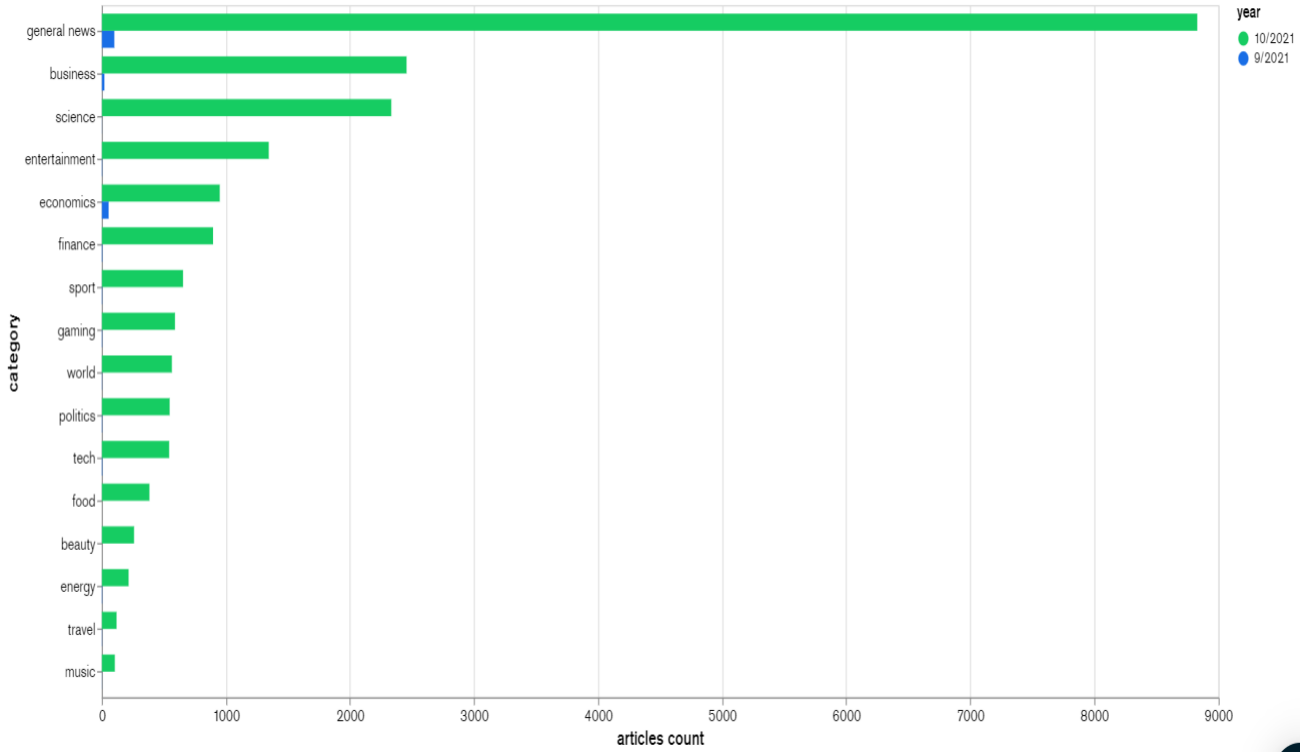
Training a model with at least 60% accuracy

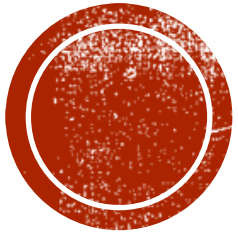
Solution: Instead of manually changing the hyper parameters of the models, run it and change it again and run the code we used pyspark's cross validator which was much faster

CHALLENGES

Imbalanced dataset

Solution: We can try hard to get a balanced data set but in future the data we accumulate could make it imbalanced. Therefore, we decided not to make our data balanced and used models that can have a relatively high accuracy with the data we have





MILESTONE 4

- Environment details
- Input and Output
- Challenges

ENVIRONMENT DETAILS

- **Database:** Free MongoDB database on cloud
- **Streaming:** Apache Kafka using queues
- **Data cleaning (to remove non-English words):** nltk
- **Data processing:** PySpark
- **Model training:** Naïve bayes, random forest classifier, logistic regression with OneVsRest (Picked the model with the best accuracy)
- **API:** (Flask api and uvicorn)
- **Frontend:** (HTM, css, javascript)

INPUT AND OUTPUT

Input:

- Saved model
- Stored pipeline to process article
- News article from frontend

Input Processed:

- Cleaned the text – removed all non-English words
- Passed it through the stored pipeline to get input for the model
- Loaded the model
- Called the predict function of that model to get the category of the news

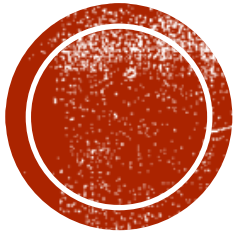
Output:

- Category of news

CHALLENGES

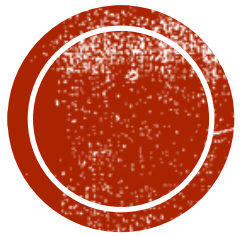
Not enough time and resources to setup docker

Solution: We tried to setup docker but due to our lack of understanding on it, we could not fix the error/issues we were getting. We also reached out to a few people on slack but mostly people could not do it either.



JMETER TESTING

Please look in the documents folder



SET UP (WITHOUT DOCKER)

Follow the steps on the next slides to set up the different environments appropriately

NOTE: Make sure using java 8 and python3

KAFKA (DATA INGESTION)

In terminal 1 (in the root folder):

- wget https://downloads.apache.org/kafka/3.0.0/kafka_2.12-3.0.0.tgz
- tar -xvf kafka_2.12-3.0.0.tgz
- sudo apt install openjdk-8-jdk -y
- java –version
- cd NewsArticlesClassifier
- pip3 install -r requirements.txt
- cd dataIngestionService
- ./kafka_2.12-3.0.0/bin/zookeeper-server-start.sh ./kafka_2.12-3.0.0/config/zookeeper.properties

In terminal 2 (in the root folder):

- ./kafka_2.12-3.0.0/bin/kafka-server-start.sh ./kafka_2.12-3.0.0/config/server.properties

KAFKA (DATA INGESTION)

In terminal 3 (run all the steps below on terminal 3 – in dataIngestionService folder):

- ./kafka_2.12-3.0.0/bin/kafka-topics.sh --list --bootstrap-server localhost:9092
- ./kafka_2.12-3.0.0/bin/zookeeper-server-start.sh -daemon ./kafka_2.12-3.0.0/config/zookeeper.properties
- ./kafka_2.12-3.0.0/bin/kafka-server-start.sh -daemon ./kafka_2.12-3.0.0/config/server.properties
- ps -ef | grep kafka
- ./kafka_2.12-3.0.0/bin/kafka-topics.sh --create --bootstrap-server 127.0.0.1:9092 --replication-factor 1 --partitions 1 --topic news-train
- ./kafka_2.12-3.0.0/bin/kafka-topics.sh --describe --bootstrap-server 127.0.0.1:9092 --topic news-train
- python3 consumer.py

In terminal 4 (In dataIngestionService folder):

- python3 producer.py

PYSPARK (DATA CLEANING + MODEL TRAINING)

Run the following:

- python3 -m nltk.downloader stopwords
- python3 -m nltk.downloader punkt
- wget https://dlcdn.apache.org/spark/spark-3.1.2/spark-3.1.2-bin-hadoop3.2.tgz
- tar -xvf spark-3.1.2-bin-hadoop3.2.tgz
- sudo apt install scala -y
- scala –version
- nano .bashrc
  - Add the following to the bash file

    export PATH=$PATH:/home/<USER>/spark-3.1.2-bin-hadoop3.2/bin

    export PYSPARK_PYTHON=python3
- Save and close the bash file
- Source .bashrc
- cd NewsArticlesClassifier/ModelPredictionService
- Spark-submit modelPrediction.py

NOTE: Make sure using java 8 and python3

FRONT END

On terminal 1:

- cd NewsArticlesClassifier
- Python3 main.py