NEWS ARTICLES CLASSIFIER

**Table of Contents**

Contents

[Release History 3](#_Toc86685024)

[**1** **Project Scope** 4](#_Toc86685025)

[1.1 Terminology 4](#_Toc86685026)

[**2** **Environment** 5](#_Toc86685027)

[2.1 Pre - Requisites 5](#_Toc86685028)

[2.2 Pre – Development Steps 5](#_Toc86685029)

[2.3 Proposed Architecture 6](#_Toc86685030)

[**3** **Development Phases** 7](#_Toc86685031)

[3.1 Data Ingestion 7](#_Toc86685032)

[3.1.1 Data Collection 7](#_Toc86685033)

[3.1.2 Data Storage 8](#_Toc86685034)

[3.2 Data Cleansing 8](#_Toc86685035)

[3.2.1 Data Processing 8](#_Toc86685036)

[3.3 Model Training 8](#_Toc86685037)

[3.3.1 Model Training 8](#_Toc86685038)

[3.4 Model Prediction and Deployment 9](#_Toc86685039)

[3.4.1 Model Prediction 9](#_Toc86685040)

[3.4.2 Web Application development 9](#_Toc86685041)

[3.4.3 Model Deployment 9](#_Toc86685042)

[**4** **Appendix** 10](#_Toc86685043)

[4.1 Screenshots 10](#_Toc86685044)

[4.1.1 Data Collection & Storage 10](#_Toc86685045)

[4.1.2 Data Processing 12](#_Toc86685046)

[4.1.3 Model Training 13](#_Toc86685047)

[4.2 References 15](#_Toc86685048)

# Release History

The following table outlines the history of releases and versions for this document.

|  |  |  |  |
| --- | --- | --- | --- |
| **Author** | **Release Date** | **Release** | **Description of Release or Revision** |
| Krishna Chaitanya | 06-10-2021 | V0.0 | Initial draft for week1 milestone |
| Krishna Chaitanya | 09-10-2021 | V0.1 | Week1 milestone updated |
| Krishna Chaitanya | 17-10-2021 | V0.2 | Week2 milestone updated |
| Krishna Chaitanya | 23-10-2021 | V0.3 | Week3 milestone updated |
| Krishna Chaitanya | 31-10-2021 | V0.4 | Week4 milestone and final submission |

# **Project Scope**

Classify News Articles into categories - With information overload today users are inundated with news articles of all topics, even the ones which may not be relevant to users. So, a system which can classify incoming news articles and appropriately tag the corresponding category is required

Developing a system as required involves all the following stages of Machine Learning Project Life Cycle –

1. Data Ingestion
2. Data Preparation
3. Data segregation & Model Training
4. Model Deployment
5. Model Prediction

## Terminology

|  |  |
| --- | --- |
| Terminology | Description |
| Source system | RSS feeds and web pages |
| Target System | Database in data ingestion phase and HTML / WEB UI in model prediction phase |
| WebUI in the final deployment |

# **Environment**

Prepare the system with all the necessary software for the development, testing and deployment.

## Pre - Requisites

We must ensure the following requirements are available and installed:

|  |  |  |
| --- | --- | --- |
|  | **Software** | **Remarks** |
| language | Python, HTML&CSS | To write necessary code |
| data store | MongoDB/MySQL | To store the data |
| IDE | PyCharm / Visual code, jupyter notebook | To enable faster coding |
| Packages/API | Kafka, Flask, PySpark, sklearn, BeautifulSoup, Docker, Git | Software packages and APIs required for development |
| Operating System | Windows/Ubuntu |  |
|  | **Hardware** | **Remarks** |
| RAM | Minimum:8gb |  |
| Recommended:16gb |  |
| HDD | Minimum:20gb |  |

## Pre – Development Steps

* Installing the softwares
  + Download and install the corresponding executable packages for python, mongodb/mysql, pycharm/vs code, docker, git etc.
  + Using python pip install flask, pyspark, flask, etc
  + For Kafka
    - Download and extract the corresponding archive file.
    - Configure kafka and zookeeper properties by editing the corresponding properties files located in kafka/config directory.
    - To start kafka server, start the zookeeper process first
  + For kafka-mongodb sink configuration:
    - Download mongo-kafka-connect jar file and place it to kafka\libs folder
    - Create new properties files containing parameters required to connect kafka broker and mongodb database and save the files to kafka\config folder. These properties files are passed as parameters to connect-standalone script file to establish the sink connection
  + For mongodb-spark connection:
    - Download bson, mongodb driver core, mongo java driver, mongo spark connector jars and place them in a folder in the project working directory
    - Provide the path of the above jar files location in the python code to load the data from “raw\_data” from MongoDB into Spark

## Proposed Architecture

Proposed architecture is given in the following diagram:

Graphical user interface

Description automatically generated

# **Development Phases**

## Data Ingestion

### Data Collection

* Data collection is done using Free News API from RapidAPI.com
  + Input details:

|  |  |  |
| --- | --- | --- |
| **Service** | **Input** | **Response** |
| NewsApi  (rapidapi.com)  &  (mediastack.com) | Secret key as a parameter as part of headers | Response is a dictionary with a articles list with various features |
| Key\_words to search news articles as a query string |
| Kafka broker | Articles data in dictionary(json) format published to topic | With successful kafka-mongodb sink configuration, messages published to topic is inserted to configured mongodb database-collection |

* To start the data collection, run the rapidapi\_news.py which collects the data and publish the articles to kafka topic

### Data Storage

* Data collected is stored in database

|  |  |  |
| --- | --- | --- |
| **Service** | **Input** | **Response** |
| Kafka-mongo sink connection | Articles data in dictionary(json) format published to topic | Messages published to topic is inserted to configured mongodb database-collection |

## Data Cleansing

### Data Processing

* After reading the raw\_data from mongodb following the pre-development steps, data is loaded into Spark via SparkSession object as a DataFrame
* The DataFrame is now used to perform feature engineering i.e.,
  + extracting features from data by extracting relevant features and dropping irrelevant features, and
  + transforming them into formats that are suitable for model training
  + filling missing data or dropping row data according to requirement

## Model Training

### Model Training

* The data extracted from the Data Processing stage is being used for the training of the model
* Text pre-processing is performed on the text data and the cleaned data is loaded for text classification.
* Text Vectorization will be implemented on text data using TfidfVectorizer and encode the tag labels using MultilabelBinarizer
* Model Classical classifiers (SGC classifier, MultiNomial Naive Bayes Classifier, Random Forest Classifier, etc.) for modelling and compare the results.
* The trained model is saved as a pickle file which will be later used for model deployment and predictions
* The vectorizer is also saved to a pickle file to be use for later during predictions

## Model Prediction and Deployment

### Model Prediction

* For predictions, load the previously saved vectorizer and model pickle files
* The text that’s to be classified is first transformed using the vectorizer and the predictions are returned

### Web Application development

* Using HTML and CSS a home page template is developed
* Using flaskapi the html template is rendered, and the app end points are coded
* Once the user enters the text to be classified and submits in the homepage the data is classified and predicted and displayed as Result

### Model Deployment

* Using docker python3.8-slim image as base, created the newsapp:latest docker image
* Copied all the required files like model, vectorizer, pyhton scripts, html template and css for the web app to the container image
* All the pyhton libraries required for application execution are mentioned in the requirements.txt and are installed while the docker image is built
* On successful docker image creation, the container is run and can be used for prediction

# **Appendix**

## Screenshots

### Data Collection & Storage

* Python : calling rapid api and publishing data to kafka topic

Text

Description automatically generated

* Kafka : kafka broker and kafka sink properties

A picture containing text

Description automatically generated

* MongoDB: data stored in database

Graphical user interface, text, application, email

Description automatically generated

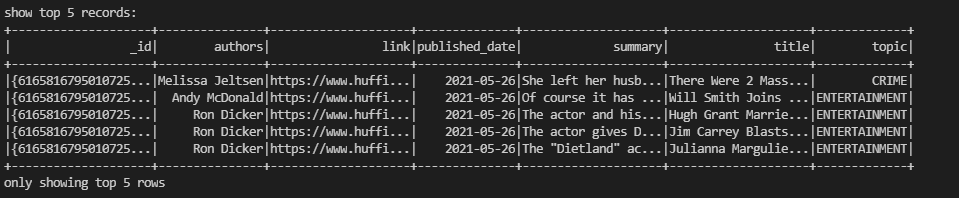
### Data Processing

* Python: output showing features selected after dropping irrelevant features

Text

Description automatically generated

* Python: output showing data(top 5 records)



* Python: output showing data after selecting only features on which we want to focus model training on

A screenshot of a computer

Description automatically generated with medium confidence

### Model Training

* UI : the user enter the article and clicks the submit and the prediction is displayed

A screenshot of a computer

Description automatically generated with medium confidence

* UI : the user enter the article and clicks the submit and the prediction is displayed

A screenshot of a computer

Description automatically generated with medium confidence

* UI : display the result

A screenshot of a computer

Description automatically generated with medium confidence

## References

* Kafka windows installation:
  + <https://towardsdatascience.com/running-zookeeper-kafka-on-windows-10-14fc70dcc771>
  + <https://www.onlinetutorialspoint.com/kafka/install-apache-kafka-on-windows-10.html>
* Kafka mongo sink config:
  + <https://stackoverflow.com/questions/56880527/how-to-stream-data-from-kafka-to-mongodb-by-kafka-connector>
  + <https://repo1.maven.org/maven2/org/mongodb/kafka/mongo-kafka-connect/>
* Mongo Spark configuration:
  + <https://docs.mongodb.com/spark-connector/current/python-api/>
  + <https://www.mongodb.com/blog/post/getting-started-with-mongodb-pyspark-and-jupyter-notebook>
* TfidVectorizer
  + <https://stackoverflow.com/questions/44193154/notfittederror-tfidfvectorizer-vocabulary-wasnt-fitted/44194026>