



CAPSTONE PROJECT

NEWS ARTICLES CLASSIFIER

Group-25

Gnana Venkata Ram Mohan and Swapna M

**Table of Contents**

**PROJECT OBJECTIVE3**

**PROBLEM STATEMENT3**

**PROJECT SETUP 3**

**CODING5**

**INFRASTRUCTURE & DEPLOYMENT8**

**1. OBJECTIVE:**

Design and implement a project that classifies the news feeds into a category using below 3 services.

1. Design and implement a service that collects the news feeds from different sources and store them in Database.
2. Design and implement a service that creates the data frames from data in Database and retrain the model to improve the accuracy
3. Design and implement a service that predicts the news category from the model.

**2. PROBLEM STATEMENT:**

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. Design a system which can classify incoming news articles and appropriately tag the corresponding category. Develop a data pipeline which includes the 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

**3. PROJECT SETUP**

**Stages are elaborated as follows –**

1. **DATA INGESTION**

Project Name - data-ingestion-service

The objective of this project is to collect news feeds and store them in MongoDB.

This data will be used as raw data to retrain the modal in later stages of the project.

We have collected news feeds from two different sources mentioned below.

1) Using rapidAPI URL :

<https://rapidapi.com/newscatcher-api-newscatcher-api-default/api/free-news/>

2) Using newsApi.

The collected data from these two sources has been translated into below 5 fields while inserting them in MongoDB

• title

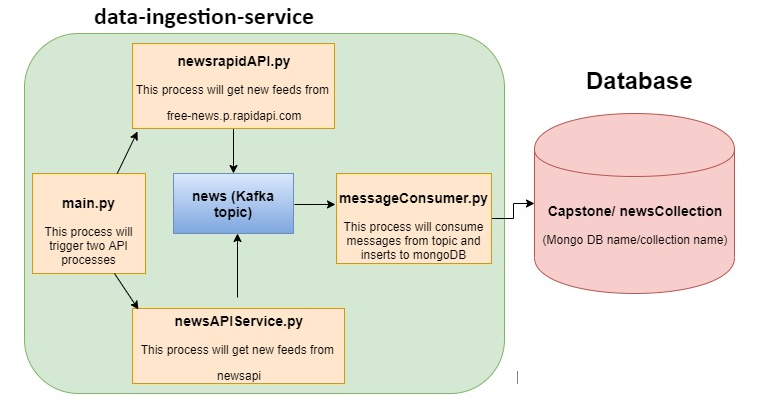
• date/ time

• summary

• topic/ category

• source

Below is the data-ingestion-service project data flow diagram:

****

**Code for Data Ingestion:**

**Main.py**

import newsrapidAPI

import newsAPIService

if \_\_name\_\_ == '\_\_main\_\_':

    title=["Finance","Sports","Politics","Religion","Education","Entertainment","Health","Business","cryptocurrency"]

    for i in title:

        print(i)

        newsrapidAPI.getNews(i)

        newsAPIService.getNews(i)

**messageconsumer.py**

from typing import Any

from kafka import KafkaConsumer

from json import loads

import json

import time

from pymongo import MongoClient

consumer = KafkaConsumer('news', bootstrap\_servers=['localhost:9092'])

client = MongoClient("mongodb://localhost:27017/")

mydb = client["Capstone"]

mycoll = mydb["newsCollection"]

for message in consumer:

    record = json.loads(message.value)

    news\_record=json.loads(record)

    mycoll.insert\_one(news\_record)

    print (news\_record)

**newsAPIService.py**

from newsapi import NewsApiClient

from kafka import KafkaProducer

import json

import time

def getNews(title):

    newsapi = NewsApiClient(api\_key='934545bbdbc14506a2624791a03d6fb9')

    top\_headlines = newsapi.get\_top\_headlines(q='business',

                                              category='business',

                                              language='en',

                                              country='us')

    def json\_serializer(newsDist):

        return json.dumps(newsDist).encode("utf-8")

    newsDist = {}

    i=0

    for i in range(len(top\_headlines['articles'])):

        newsDist.update(title=top\_headlines['articles'][i]['title'],date=top\_headlines['articles'][i]['publishedAt'],summary=top\_headlines['articles'][i]['content'],category=title,source=top\_headlines['articles'][i]['url'])

        producer = KafkaProducer(bootstrap\_servers=['localhost:9092'], value\_serializer=json\_serializer)

        producer.send('news', json.dumps(newsDist))

        time.sleep(5)

        print (json.dumps(newsDist))

**newsrapidAPI.py**

def json\_serializer(newsDist):

        return json.dumps(newsDist).encode("utf-8")

    newsDist = {}

    i=0

    for i in range(len(response['articles'])):

        newsDist.update(title=response['articles'][i]['title'], date=response['articles'][i]['published\_date'],

                    summary=response['articles'][i]['summary'], category=response['articles'][i]['topic'],

                    source=response['articles'][i]['link'])

        producer = KafkaProducer(bootstrap\_servers=['localhost:9092'], value\_serializer=json\_serializer)

        producer.send('news', json.dumps(newsDist))

        i=i+1

        time.sleep(5)

        print (json.dumps(newsDist)

**2. DATA PREPARATION (PRE-PROCESSING), SEGREGATION AND MODEL TRAINING**

Project Name – model-training-service

The objective of this project is to trigger model re-training and deployment on-demand.

Before using the raw data can be used for model training/ retraining it needs to be preprocessed to relevant structure.

• Load the data from “raw\_data” source (MongoDB) into Spark by using relevant connector for PySpark

• Perform data cleaning and preprocessing, followed by segregation to train and test datasets. (Necessary pre-training steps to be shared with participants)

• Perform model re-training (a pre-trained model along with code to retrain the model could be provided to participants).

• Highlight the explainability of the AIML model by white boxing which patterns learnt by the training data arrives to a particular class [ News topic ].

• Serialise the model and save it to a location (or push the model to a model registry like MLFlow), which can later be used for model retrieval model deployment.

**3.** **MODEL PREDICTION**

Project Name – model-prediction-service

A separate classifier project picks up the trained model either from a location or from the model registry, and exposes it for prediction in following modes –

• Real Time REST (flask) API – allows users to initiate a classification request in JSON or any other format using a REST client and returns the classified category in real time. This API should be a POST request and should accept JSON in request and return JSON as response. Request body should contain 2 fields - the title and description/summary of a single news article

Response should contain the predicted category as a field

• User Interface - A simple Streamlit app/a basic HTML page containing a form can be exposed in this project.

1. It takes the title and the description/ summary of the article, calls the flask API for prediction, and shows the predicted classification.

2. It provides an option to upload a batch file and response can be processed and stored at a folder location

**Project architecture diagram**

Diagram

Description automatically generated

**4. INFRASTRUCTURE & DEPLOYMENT** –

Additional guidelines

1. Docker images may be created for all the 3 projects. These images can then be used for deployment as containers.

2. Deployment can be orchestrated by using docker-compose (optional)

3. Flask APIs (wherever required) should use Gunicorn/ Bjoern/ CherryPy as a WSGI server.

4. Use PEP guidelines for python code standard.

**ML Tools**

Following tools to be used for project setup –

1. PyCharm as Python IDE

2. Virtual Environment – use venv or virtualenvwrapper to setup separate environments for all projects described.

3. MySQL/ MongoDB as datastore.

4. PySpark for stream processing.

5. POSTMAN for testing Flask APIs

. 6. Apache Zookeeper + Kafka for message queue/ streams.

7. Tensorboard for monitoring the progress of model retraining.

8. MLFlow for model versioning + hyper-parameters versioning.

9. Python cookiecutter templates may be used for setting up the project