# Architecture and Design - News Article Category Prediction (Week 4 - Milestones)

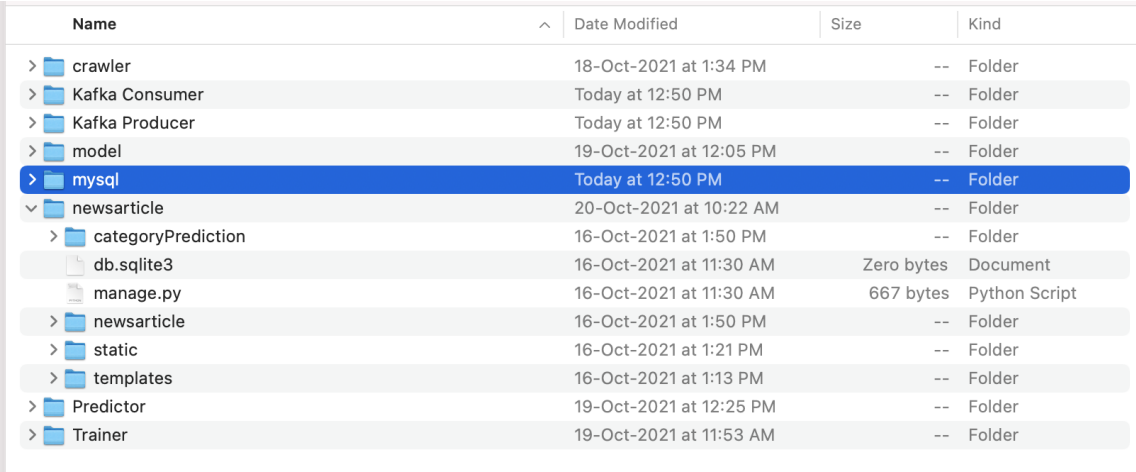
## Objective

This document is about classifying 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.

## Main Software Components

- html, css, js
- django
- python FastAPI
- Crawler(Beautiful Soup)
- pyspark
- Kafka
- mysql
- NLP libraries

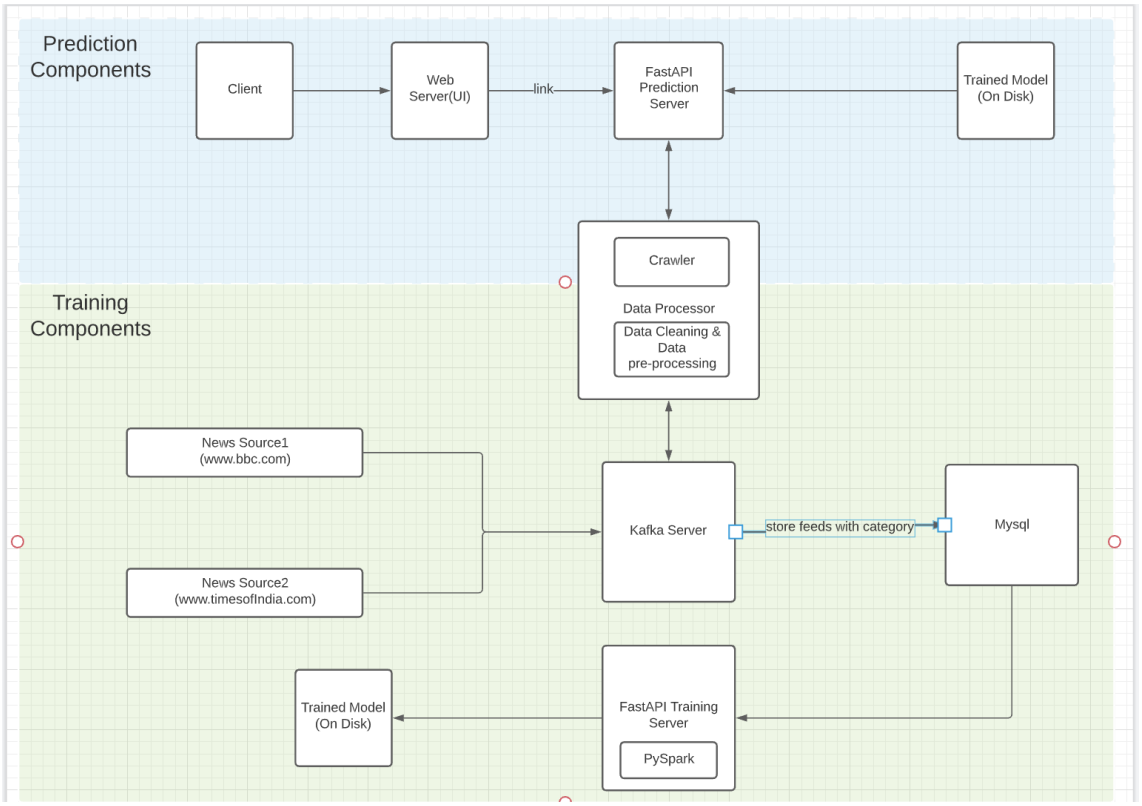
## Project Setup



| Name | Date Modified | Size | Kind |
|---|---|---|---|
| crawler | 18-Oct-2021 at 1:34 PM | -- | Folder |
| Kafka Consumer | Today at 12:50 PM | -- | Folder |
| Kafka Producer | Today at 12:50 PM | -- | Folder |
| model | 19-Oct-2021 at 12:05 PM | -- | Folder |
| mysql | Today at 12:50 PM | -- | Folder |
| newsarticle | 20-Oct-2021 at 10:22 AM | -- | Folder |
| categoryPrediction | 16-Oct-2021 at 1:50 PM | -- | Folder |
| db.sqlite3 | 16-Oct-2021 at 11:30 AM | Zero bytes | Document |
| manage.py | 16-Oct-2021 at 11:30 AM | 667 bytes | Python Script |
| newsarticle | 16-Oct-2021 at 1:50 PM | -- | Folder |
| static | 16-Oct-2021 at 1:21 PM | -- | Folder |
| templates | 16-Oct-2021 at 1:13 PM | -- | Folder |
| Predictor | 19-Oct-2021 at 12:25 PM | -- | Folder |
| Trainer | 19-Oct-2021 at 11:53 AM | -- | Folder |

- All the packages and services are accumulated in the docker-compose(yaml) file.

## Architecture



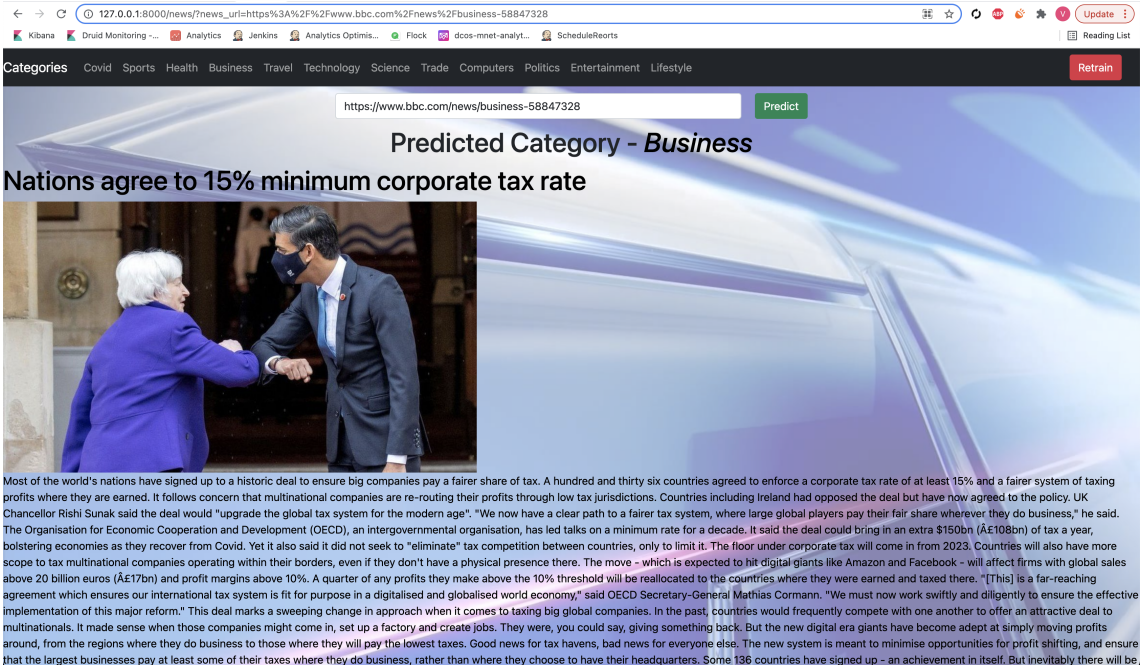

## UI Server(django)

### Technologies

- We are using django server as a frontend web server.
- HTML, CSS, bootstrap for rendering webpages.
- Javascript for input validations.

### Functionality

- User can enter url or article in the input text box.
- Then he can click on `Predict` button for category predections.
- It will get the article from the crawler and display it on the page.
- It will also the predictor api which returns the predicted category.
- `Retrain` button is also provided to retrain the model, it will call the Trainer API for the same.




## Categories Supported

```
{'business':1,'computers':2,'covid':3,'entertainment':4,'health':5,'lifestyle':6
,'politics':7,'science':8,'sport':9,'technology':10,'trade':11,'travel':12}
```

## Docker Compose

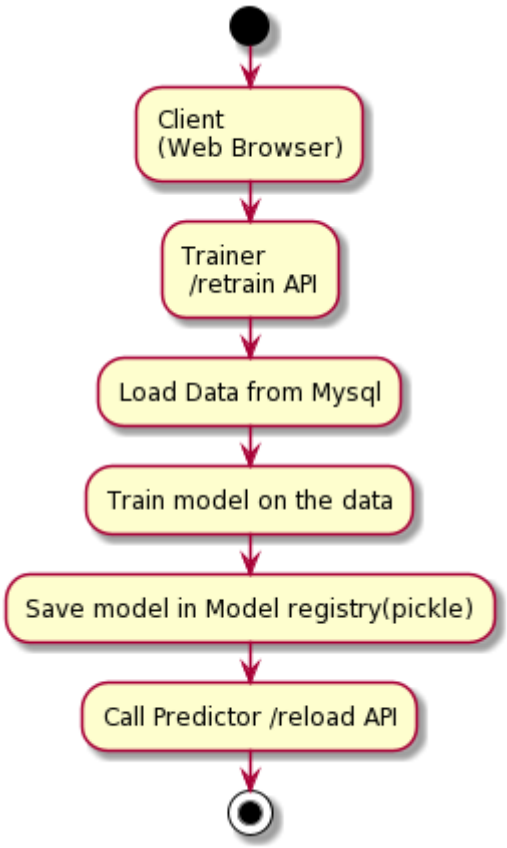
```
version: '3'

services:
  crawler:
    image: python
    command: bash -c "pip install -r /home/requirements.txt && python3
/home/Crawler.py"
    volumes:
      - ./Crawler:/home
      - ./model:/home/model
    ports:
      - "8355:8355"
  trainer:
    image: ecoron/python36-sklearn
    command: bash -c "pip install -r /home/requirements.txt && python3
/home/trainer.py"
    volumes:
      - ./Trainer:/home
      - ./model:/home/model
    ports:
      - "8354:8354"
  predictor:
```

```
    image: ecoron/python36-sklearn
    command: bash -c "pip install pip install -r /home/requirements.txt && python3
/home/predictor.py"
    volumes:
      - ./Predictor:/home
      - ./model:/home/model
    ports:
      - "8818:8818"
  ui:
    image: python
    command: bash -c "pip install -r /home/requirements.txt && python3
/home/newsarticle/manage.py runserver 0.0.0.0:8001"
    volumes:
      - ./UI:/home
    ports:
      - "8001:8001"
  mysql:
    image: mysql
    environment:
      MYSQL_DATABASE: 'newsdb'
      # So you don't have to use root, but you can if you like
      #MYSQL_USER: 'root'
      # Password for root access
      #MYSQL_PASSWORD: 'Murthy@007'
      MYSQL_ROOT_PASSWORD: 'Murthy@007'
    ports:
      - "3306:3306"
  pymysql:
    image: python
    command: bash -c "pip install -r /home/requirements.txt && python3 /home/main.py"
    volumes:
      - ./Mysql:/home
    ports:
      - "9999:9999"
  zookeeper:
    image: wurstmeister/zookeeper
    environment:
      ZOOKEEPER_CLIENT_PORT: 2181
      ZOOKEEPER_TICK_TIME: 2000
    ports:
      - "2181:2181"
  kafka:
    image: wurstmeister/kafka
    depends_on:
      - zookeeper
    ports:
      - "29092:29092"
    environment:
      KAFKA_BROKER_ID: 17
      KAFKA_ZOOKEEPER_CONNECT: zookeeper:2181
      KAFKA_LISTENERS: PLAINTEXT://kafka:9092,PLAINTEXT_HOST://kafka:29092
      KAFKA_ADVERTISED_LISTENERS: PLAINTEXT://kafka:9092,PLAINTEXT_HOST://kafka:29092
```

```
      KAFKA_LISTENER_SECURITY_PROTOCOL_MAP:
PLAINTEXT:PLAINTEXT,PLAINTEXT_HOST:PLAINTEXT
      KAFKA_INTER_BROKER_LISTENER_NAME: PLAINTEXT
      KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR: 1
  producer:
      image: python
      command: bash -c "pip install -r /home/requirements.txt && python3 -u
/home/producer.py"
      volumes:
        - ./Producer:/home
      depends_on:
        - kafka
      ports:
        - "9998:9998"
  consumer:
    image: python
    command: bash -c "pip install -r /home/requirements.txt && python3 -u
/home/cons.py"
    volumes:
      - ./Consumer:/home
    depends_on:
      - kafka
  spark-master:
    image: bde2020/spark-master:3.1.1-hadoop3.2
    ports:
      - "8080:8080"
      - "7077:7077"
    environment:
      - INIT_DAEMON_STEP=setup_spark
  spark-worker:
    image: bde2020/spark-worker:3.1.1-hadoop3.2
    depends_on:
      - spark-master
    ports:
      - "8081:8081"
    environment:
      - "SPARK_MASTER=spark://spark-master:7077"
```

**Trainer and Predictor API's**

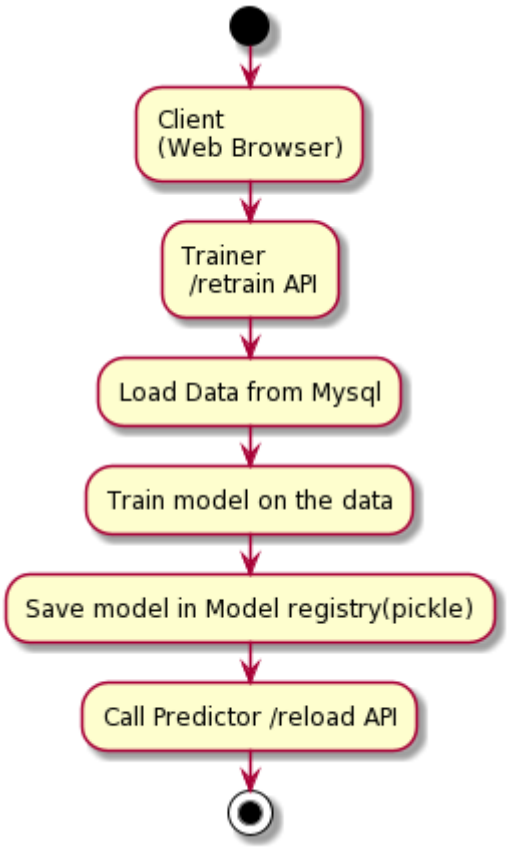
- `Retrain` API is created on the Trainer service.
- The service will get the articles from the mysql db and train the model.
- This model is then logged into the model registry by MLflow.
- Trainer will then inform the predictor to reload the model.




{.callout}

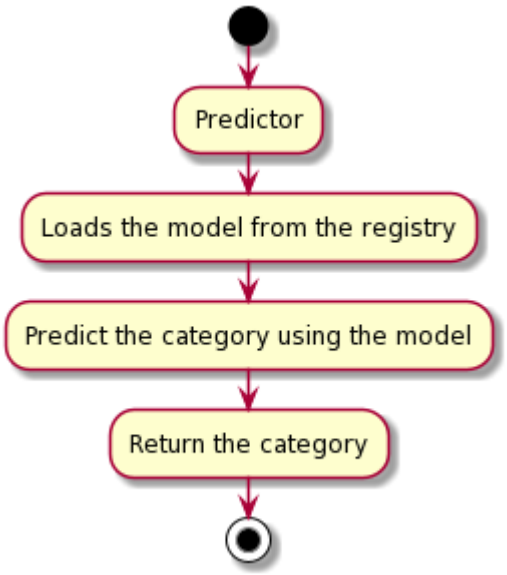
**Trainer**

- Trainer first loads the articles from mysql db.
- Then train the model using MLflow and pyspark.
- MLflow saves the model in the modele-registry as the artifacts.




**Predictor**

- Predictor loads the model from the registry.
- Predicts the category, given the article.



## Group Details

- Group-Name: VK Learners
- - Member1: Vinod Adwani
- - Member2: KSRC Murthy