School of Science and Engineering

CSC538201 Artificial Intelligence for Digital

Movie-Roulette

Final Report

Adnane Harachi

Supervised by:

Prof. Asmae Mourhir

Table of contents:

1. Project Inception

1.1. Framing the Business Idea as an ML Problem

- **Business case description:**

Movie Roulette is a comprehensive movie recommendation system that uses machine learning algorithms to personalize movie recommendations based on users' tastes, watching history, and comments. The system provides unique suggestions to each user based on user interactions and movie features such as genre, actor, and narrative keywords. Furthermore, Movie Roulette uses sentiment analysis to assess user comments, which improves the accuracy and relevancy of the suggestions. Movie Roulette's user-friendly design and complex features create a dynamic platform for cinephiles to find, explore, and enjoy a curated selection of movies tailored to their individual interests.

- **Business value of using ML:**

1) Enhanced User Experience: Movie Roulette uses machine learning to monitor user behavior, preferences, and comments, resulting in a more engaging and personalized experience. This personalized strategy results in enhanced user happiness, retention rates, and client loyalty, which drives revenue growth.

2)Improved recommendation. Accuracy: Machine learning algorithms evaluate massive quantities of data, such as movie content and user input, to generate extremely accurate and relevant suggestions. This accuracy in suggestions increases the likelihood that users will find material they appreciate, improving their entire experience and motivating them to spend more time on the site.

3)Increased User Interaction: Movie Roulette uses machine learning to adapt to user preferences and trends in real time, resulting in a dynamic and engaging browsing experience. This degree of response not only keeps users interested, but also encourages them to share comments and participate in community conversations, resulting in a more lively and active user base.

4) Cost Efficiency: Implementing machine learning can save money by automating the recommendation process and decreasing the need for manual curation. This efficiency enables the platform to grow more effectively, delivering a bigger audience without incurring additional operational expenditures.

5)Competitive Advantage: By utilizing superior machine learning algorithms, Movie Roulette gains a competitive advantage in the industry. The platform's ability to give superior tailored suggestions and a smooth user experience distinguishes it from competitors, drawing new users and gaining market share.

- **Data overview:**

The Movie Roulette system uses data from several sources to provide tailored movie suggestions. The Movie Database (TMDB) provides an API that allows users to retrieve movie information such as title, genre, runtime, rating, and posters. In addition, the system incorporates metadata from IMDb movies, which are available in CSV format. Web scraping techniques are used to extract user evaluations from IMDb, which are then subjected to sentiment analysis.

The movie database (TMDB) API: This API is useful in retrieving comprehensive movie information. It offers data points like:

Title: The movie's name.
Genre: Categories such as action, humor, and drama.

Runtime: The length of the movie.
Rating: Audience scores and ratings.
Posters are visual material for each movie.

IMDb information: The information for movies on IMDb is accessible in CSV format, and it comprises a number of attributes necessary for extensive research. The data contains:

ID and IMDb ID: unique identifiers for each movie.

Original language and title: Language and title information.

Overviews: are summaries or descriptions of movies.

Popularity metrics: indicate the movie's popularity.

Release date and revenue: Financial and temporal information.

Runtime: The length of the movie.

Spoken Languages: Languages utilized in the film.

Status: The movie is currently under production.

Taglines: are promotional slogans linked with a movie.

Vote Average and Vote Count: A total of user ratings and votes.

- **Project archetype:**

Movie Roulette is a Content-Based Movie Recommendation System that uses machine learning algorithms to suggest movies based on users' tastes, watching history, and interactions. It primarily use content-based filtering algorithms, assessing movie features such as genre, actor, and narrative keywords to deliver individualized recommendations. The system's goal is to increase user engagement and pleasure by providing recommendations tailored to individual preferences and interests. While sentiment analysis is an option for future integration, Movie Roulette now relies on content-based filtering to provide suggestions. The algorithm is intended to grow over time, with the option of integrating sentiment analysis to improve suggestions depending on user feedback. Movie Roulette also intends to optimize user interactions and deliver a fluid browsing experience, focusing on usability and accessibility for cinephiles looking for handpicked movie selections.

1.2. Feasibility analysis

  - **literature review**

- **Introduction:**

The incorporation of machine learning (ML) into recommendation systems has transformed tailored content distribution, greatly improving user experience and engagement across several platforms. This literature review looks at key and current research on collaborative filtering, content-based filtering, hybrid models, and the most recent developments in deep learning for recommendation systems.

- **Collaborative filtering**

Collaborative filtering (CF) is still one of the most extensively used approaches in recommendation systems. It works by analyzing trends in user behavior and predicting future preferences. Sarwar et al. (2001) developed user-based and item-based collaborative filtering algorithms and demonstrated their usefulness in suggesting things by identifying similarities between users or objects. These approaches employ massive datasets to find connections and recommend things based on the interests of similar users.

Matrix factorization methods, such as Singular Value Decomposition (SVD), have improved CF. Koren, Bell, and Volinsky (2009) demonstrated the effectiveness of SVD in collecting latent characteristics in user-item interactions, resulting in more accurate recommendations. This method breaks down the user-item interaction matrix into lower-dimensional representations, revealing hidden linkages between users and things.

- **Content-Based Filtering**

Content-based filtering (CBF) suggests things similar to those a user has previously loved based on item characteristics. Pazzani and Billsus (2007) presented a thorough discussion of CBF approaches, highlighting the significance of item qualities like genre, director, and keywords in providing suggestions. These strategies create user profiles by studying the characteristics of things that users have rated highly and then utilizing these profiles to recommend comparable items.

Lops, Gemmis, and Semeraro (2011) emphasized the benefits of CBF for adjusting to new objects without needing considerable user engagement. CBF systems are especially successful in areas with extensive information, allowing for the suggestion of goods with precise attributes that match user preferences.

- **Hybrid Models**

Hybrid recommendation systems combine CF and CBF to take use of both methodologies' strengths while mitigating their respective shortcomings. Burke (2002) classified hybrid recommendation approaches, including weighted, switching, and mixed models. These systems improve robustness and accuracy by combining various sources of data.

Bell and Koren (2007) demonstrated the advantages of hybrid models in the Netflix Prize competition, where combining CF and CBF approaches resulted in much higher recommendation accuracy. Hybrid systems may overcome difficulties such as cold-start and data scarcity, resulting in more trustworthy and diversified suggestions.

- **Deep Learning Advances**

Recent advances in deep learning have expanded the capabilities of recommendation systems. He et al. (2017) presented Neural Collaborative Filtering (NCF), a deep learning-based method for modeling complicated user-item interactions using neural networks. NCF beats regular CF because it captures nonlinear linkages and higher-order interactions.

Covington, Adams, and Sargin (2016) examined the use of deep neural networks in YouTube's recommendation engine, emphasizing its scalability and efficacy while processing massive amounts of data. These models use several layers to learn abstract representations, resulting in accurate and tailored suggestions.

  - **model choice/ specification of a baseline**

For the baseline model, I used a hybrid recommendation system that combines collaborative and content-based filtering. This strategy takes advantage of the qualities of both systems to give more accurate and tailored movie recommendations.

The Collaborative Filtering Component uses user interaction data (ratings, watch history) to forecast user preferences based on comparable users' activities. Matrix factorization techniques such as Singular Value Decomposition (SVD) will be used to identify latent components that capture user and object attributes.

Content-Based Filtering Component: This component will assess movie properties (genres, actors, directors, and keywords) and propose films that are similar to those the user has previously appreciated. Feature vectors representing movie material will be created and used to assess similarity.

Hybrid Integration: The final suggestion will be a weighted blend of predictions from the c ollaborative and content-based components. The weights will be established using cross-validation to improve overall suggestion accuracy.

- **Metrics for business goal evaluation**

To assess the performance of the Movie Roulette system in meeting business objectives, we will use the following metrics:

Recommendation Accuracy:

Precision and recall: Determine the fraction of relevant movies advised among all recommended movies and all relevant movies, respectively.
Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE): Determine the accuracy of anticipated ratings as compared to actual user ratings.

User Engagement:

The click-through rate (CTR) is the proportion of recommended movies that users click on.
Conversion Rate: The proportion of recommended movies that consumers watch.
User Retention Rate: The percentage of users who return to the site over a given time period.

User Satisfaction:

The Net Promoter Score (NPS) assesses consumers' likelihood of recommending the platform to others.
User comments Analysis: A sentiment analysis of user evaluations and comments to determine

overall satisfaction.

Business Impact:

income Growth: Higher subscription rates and ad income due to better user engagement and retention.
Market Share: Increase in user base relative to competitors, reflecting the platform's ability to recruit and keep users.

2. ML Pipeline Development - From a Monolith to a Pipeline

2.1. Ensuring ML Pipeline Reproducibility

Reproducibility is essential in machine learning (ML) projects because it ensures that experiments can be reliably repeated, findings checked, and models kept across time. This section describes the measures used to assure repeatability in Movie Roulette's machine learning workflow.

- **Project structure definition and modularity**

A well-defined project structure encourages organization and modularity, which are critical for scalability and maintenance. The project is organized as follows:

Src/: The source code is structured into modules like data_preprocessing, model_training, model_evaluation, and utils.

data/: Raw and processed data are stored here, along with subdirectories for training, validation, and test datasets.

notebooks/: Jupyter notebooks are used for exploratory data analysis and experimentation.

models/: Saved models and associated artifacts.

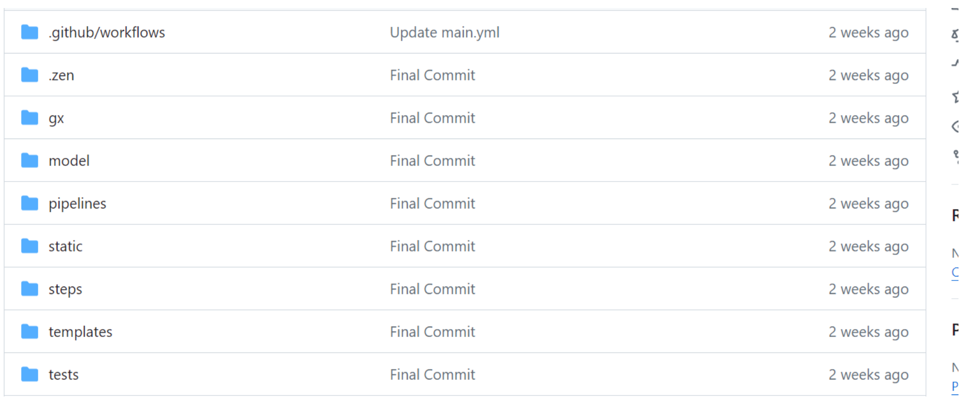
Utility scripts for data gathering and preparation. Configuration files for various settings and studies.

logs: Files for recording experiment runs and results.

requirements.txt: A list of dependencies needed to run the project.

Git is used for version control, which ensures that all code changes are logged, allowing for

collaborative development and quick rollbacks to earlier versions.



.github/workflows	Update main.yml	2 weeks ago
.zen	Final Commit	2 weeks ago
gx	Final Commit	2 weeks ago
model	Final Commit	2 weeks ago
pipelines	Final Commit	2 weeks ago
static	Final Commit	2 weeks ago
steps	Final Commit	2 weeks ago
templates	Final Commit	2 weeks ago
tests	Final Commit	2 weeks ago

Advantages:

Branching and merging allow for feature development and integration in parallel.

History Tracking: Keeps a comprehensive record of modifications, allowing for reversal to earlier states.

Collaboration: Features like as pull requests and code reviews enable collaborative development.

- **Code versioning**

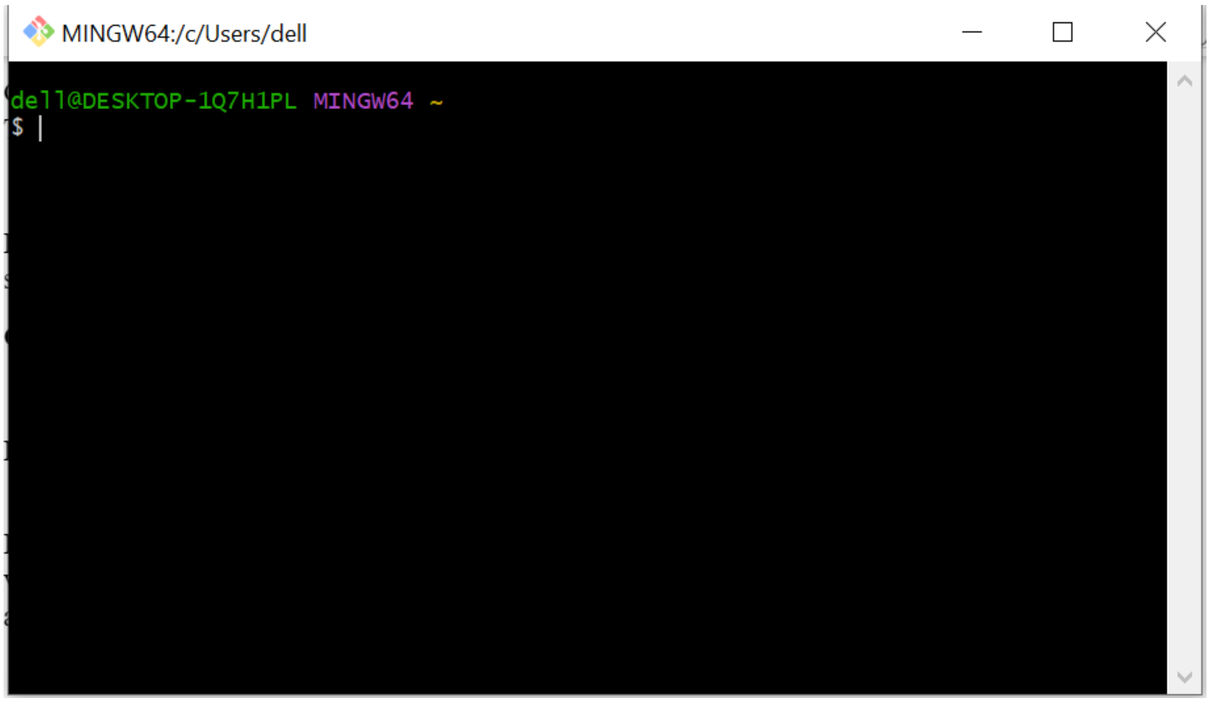
Git is used for extensive code versioning, which facilitates strong collaboration and change tracking. The key practices include:

Branching Strategy: Create feature branches for future innovations while keeping the main branch stable.

Commit Messages: Use traditional commit message rules to guarantee clarity and traceability.

Pull Requests: Creating a review mechanism for code changes to ensure code quality and uniformity.

In addition to Git, Data Version Control (DVC) is used to manage huge data files and dataset versions. DVC records data changes alongside code, allowing for seamless data versioning and assuring that the data used for training and assessment is consistent and repeatable.



```
MINGW64:/c/Users/dell
dell@DESKTOP-1Q7H1PL MINGW64 ~
$ |
```

Advantages:

Data management: Handles huge datasets efficiently while avoiding bloating the Git repository.

Reproducibility ensures that the precise version of an experiment's data may be recovered.

Integration with Git: Works seamlessly with Git, providing version control of both code and data.

- **Data versioning**

DVC is essential for versioning datasets, as it ensures that each experiment can be tracked back to the exact data version utilized. This configuration includes:

Data Storage: Using DVC to handle data stored in distant storage systems (such as AWS S3 or Google Drive).

Data Pipelines: Creating data processing pipelines in DVC helps automate and replicate data preparation procedures.

Version Tags: Data versions used in separate studies are tagged for easier retrieval and consistency across pipeline stages.

Advantages:

Storage Agnostic: Works with a variety of storage backends (local and cloud).

Pipeline Management: Defines data processing pipelines to make data preparation repeatable.

Data Lineage: Maintains data lineage, indicating how data was generated and altered.

- (5 pts) **Experiment tracking and model versioning**

MLflow is widely used for experiment tracking and model versioning, as it provides a centralized platform for managing and tracking experiments. This configuration includes:

Experiment Tracking:

Configure a centralized MLflow tracking server to record parameters, metrics, and artifacts.

Logging: Recording all essential information for each experiment, such as hyperparameters, dataset versions, code versions, and outcomes.

Comparisons: Using the MLflow UI, you can compare various runs and analyze performance across studies.

Model Versioning:

Model Registry: Use MLflow's model registry to track different model versions.

Model Staging: Designating models as "staging" or "production" to control lifecycle stages.

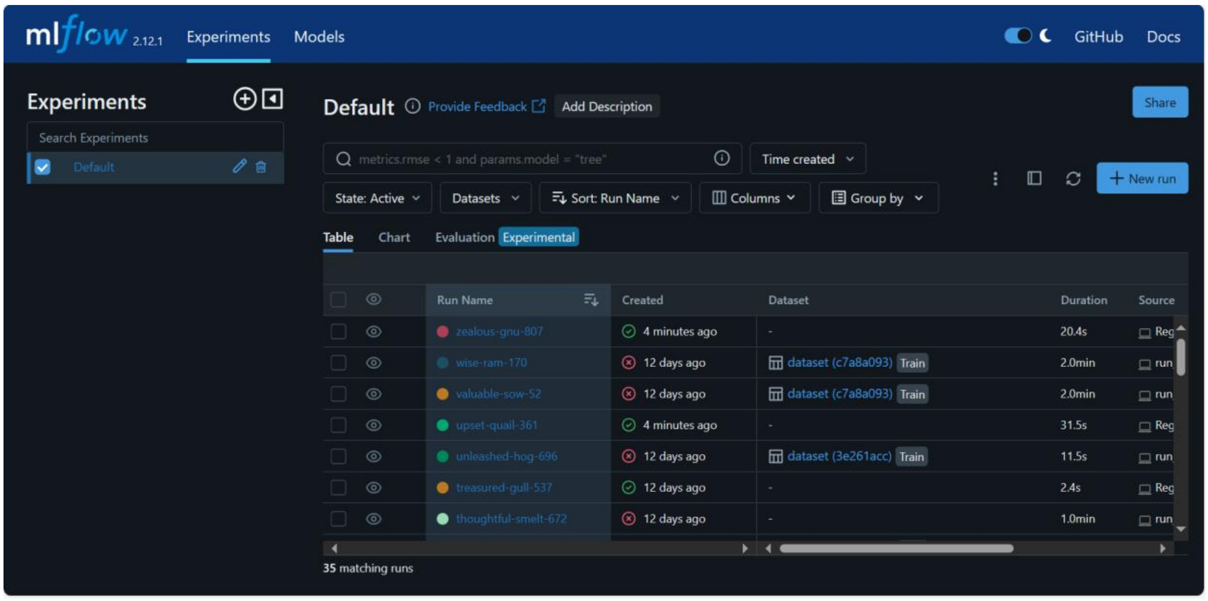
Deployment: Using MLflow and deployment pipelines to automatically send the best-performing models to production environments.

Advantages:

Centralized Metadata administration: Stores all experiment data in one location for convenient access and administration.

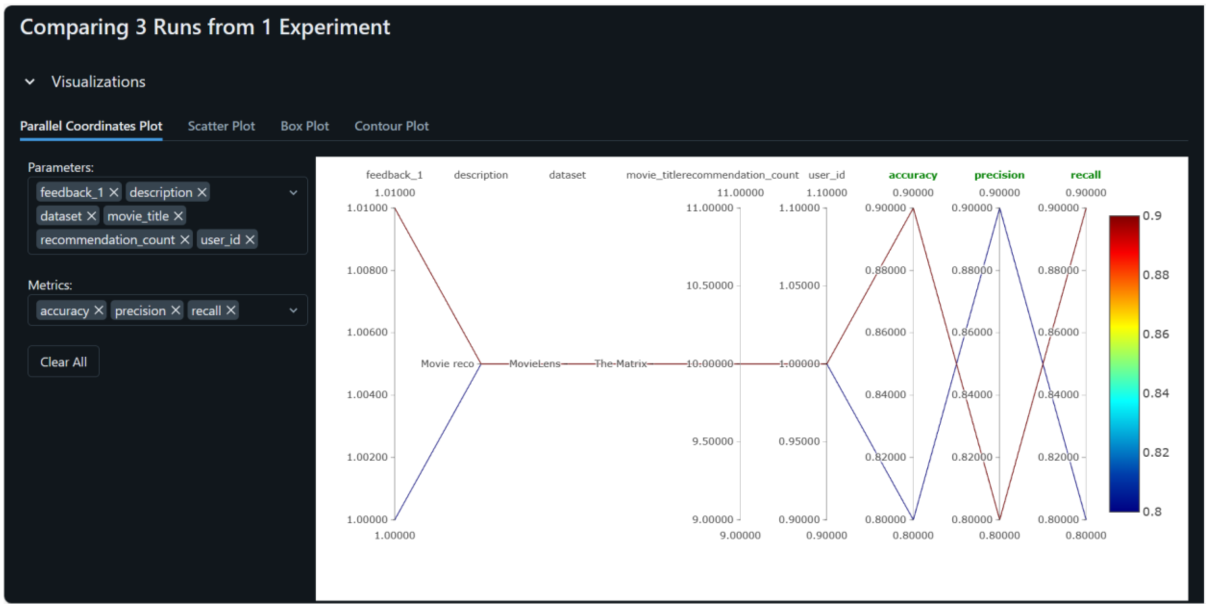
Search and Query: Allows users to search and query experiments based on a variety of parameters.

Visualization: Provides a UI for displaying experiment runs, comparing metrics, and analyzing Performance.



Run Name	Created	Dataset	Duration	Source
zealous-gnu-807	4 minutes ago	-	20.4s	Reg
wise-ram-170	12 days ago	dataset (c7a8a093) Train	2.0min	run
valuable-sow-52	12 days ago	dataset (c7a8a093) Train	2.0min	run
upset-quail-361	4 minutes ago	-	31.5s	Reg
unleashed-hog-696	12 days ago	dataset (3e261acc) Train	11.5s	run
treasured-gull-537	12 days ago	-	2.4s	Reg
thoughtful-smelt-672	12 days ago	-	1.0min	run

35 matching runs



- **Setting up a meta store for metadata**

The MLflow Tracking Server serves as a single store for all experiment metadata, ensuring that all details are saved and available.

Advantages:

Centralized Metadata administration: Stores all experiment data in one location for convenient access and administration.

Search and Query: Allows users to search and query experiments based on a variety of parameters. Visualization: Provides a UI for displaying experiment runs, comparing metrics, and analyzing performance.

- **Setting up the machine learning pipeline under an MlOps platform**

MLflow is also used to configure the whole ML pipeline, ensuring that every step, from data import to model deployment, is repeatable and maintainable.
Beyond experiment tracking and model registry, MLflow orchestrates the complete ML lifecycle and integrates with CI/CD processes.

Advantages:

Pipeline Orchestration: Automates complicated procedures while ensuring repeatability.

Continuous Integration/Continuous Deployment (CI/CD): Enables automated testing, validation, and deployment of machine learning models.

Scalability: Enables scalability of ML processes, ensuring that the pipeline can manage growing data and model complexity.

2.2. Pipeline Components

2.2.1 Setup of data pipeline within the larger ML pipeline/ MLOps Platform

The ML pipeline for Movie Roulette is built with a variety of components to provide quick data processing, model training, and assessment. This section describes how to build up the data pipeline inside the wider ML pipeline running on the MLOps platform, including data validation, preprocessing, feature engineering, model training, and offline assessment. Additionally, model behavioral tests are created to validate the models' dependability and usefulness.

Data Pipeline Setup inside the Larger ML Pipeline/MLOps Platform.

ZenML, an MLOps framework that includes tools for managing ML pipelines from start to finish, is used to orchestrate the data pipeline.

ZenML: ZenML is used to create and execute the data pipeline, smoothly integrating different phases into the overall ML pipeline.

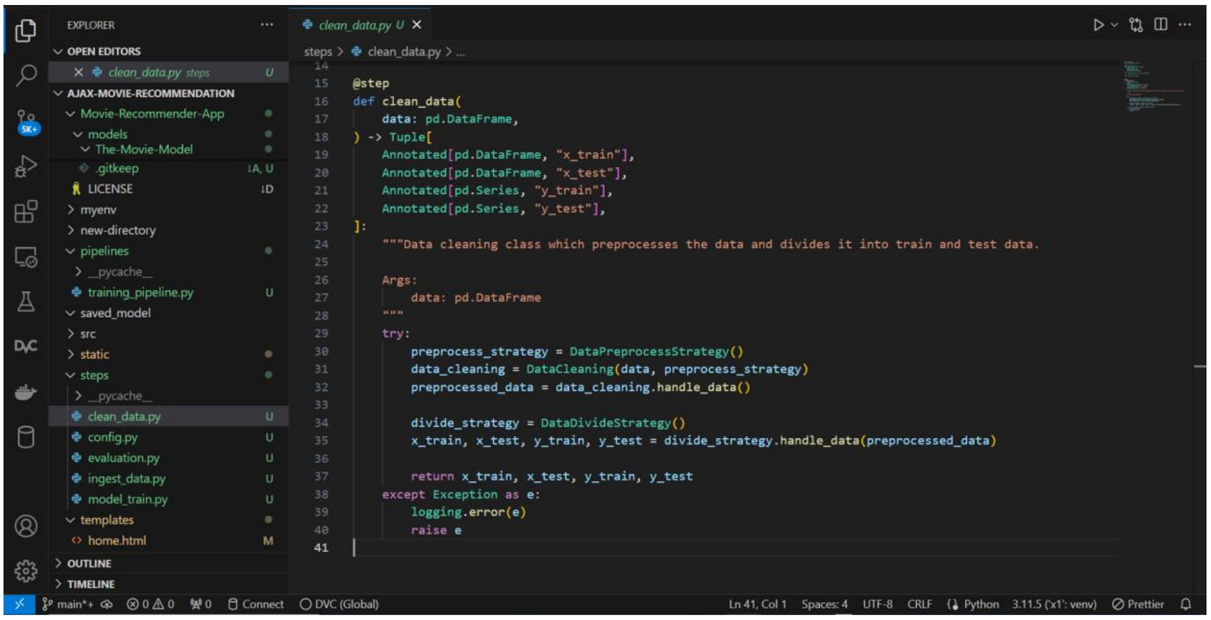
Advantages:

Modularity: ZenML enables the design of modular and reusable pipeline components, which improves maintainability and scaling.

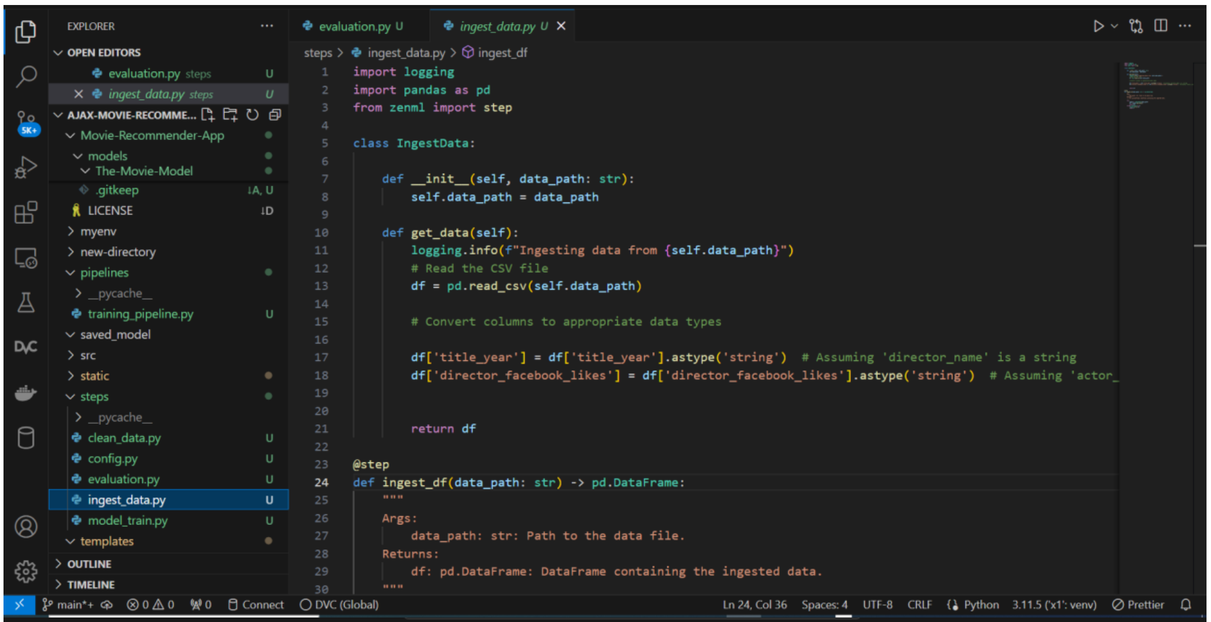
Versioning: Manages pipeline versions to provide repeatability and traceability.
Workflow Orchestration: Enables the orchestration of complicated data workflows, allowing for automation and monitoring.

Disadvantage: not enough documentation.

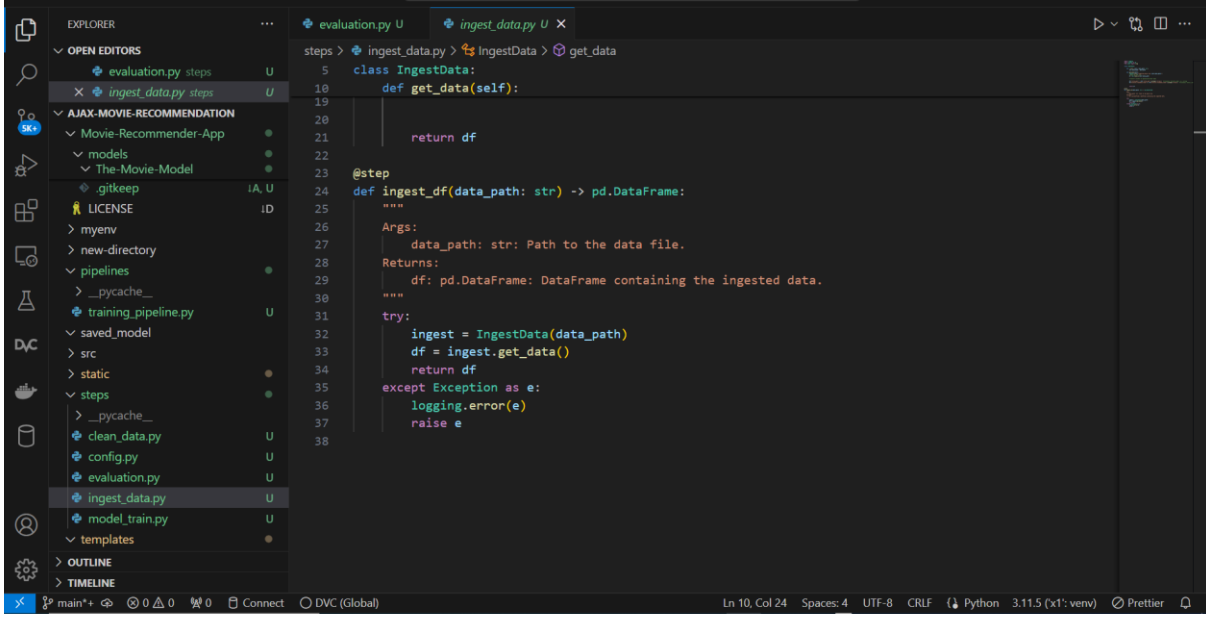
So here I created a pipeline where I have many steps, I ingested the data, clean it, evaluate it then make a pipeline called training pipeline to take all these steps and build a Zenml pipeline.


clean_data.py U
steps > clean_data.py > ...
Ln 41, Col 1   Spaces: 4   UTF-8   CRLF   Python   3.11.5 ('x1': venv)   Prettier

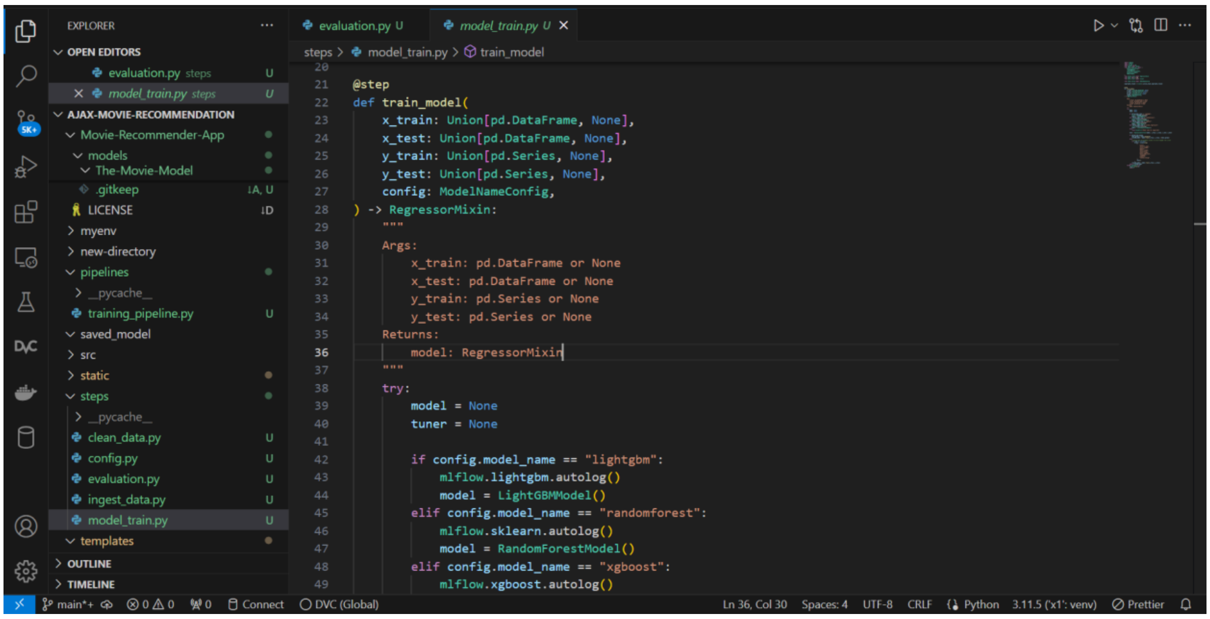
```python
@step
def clean_data(
    data: pd.DataFrame,
) -> Tuple[
    Annotated[pd.DataFrame, "x_train"],
    Annotated[pd.DataFrame, "x_test"],
    Annotated[pd.Series, "y_train"],
    Annotated[pd.Series, "y_test"],
]:
    """Data cleaning class which preprocesses the data and divides it into train and test data.

    Args:
        data: pd.DataFrame
    """
    try:
        preprocess_strategy = DataPreprocessStrategy()
        data_cleaning = DataCleaning(data, preprocess_strategy)
        preprocessed_data = data_cleaning.handle_data()

        divide_strategy = DataDivideStrategy()
        x_train, x_test, y_train, y_test = divide_strategy.handle_data(preprocessed_data)

        return x_train, x_test, y_train, y_test
    except Exception as e:
        logging.error(e)
        raise e
```


evaluation.py U
ingest_data.py U
steps > ingest_data.py > ingest_df
Ln 24, Col 36   Spaces: 4   UTF-8   CRLF   Python   3.11.5 ('x1': venv)   Prettier

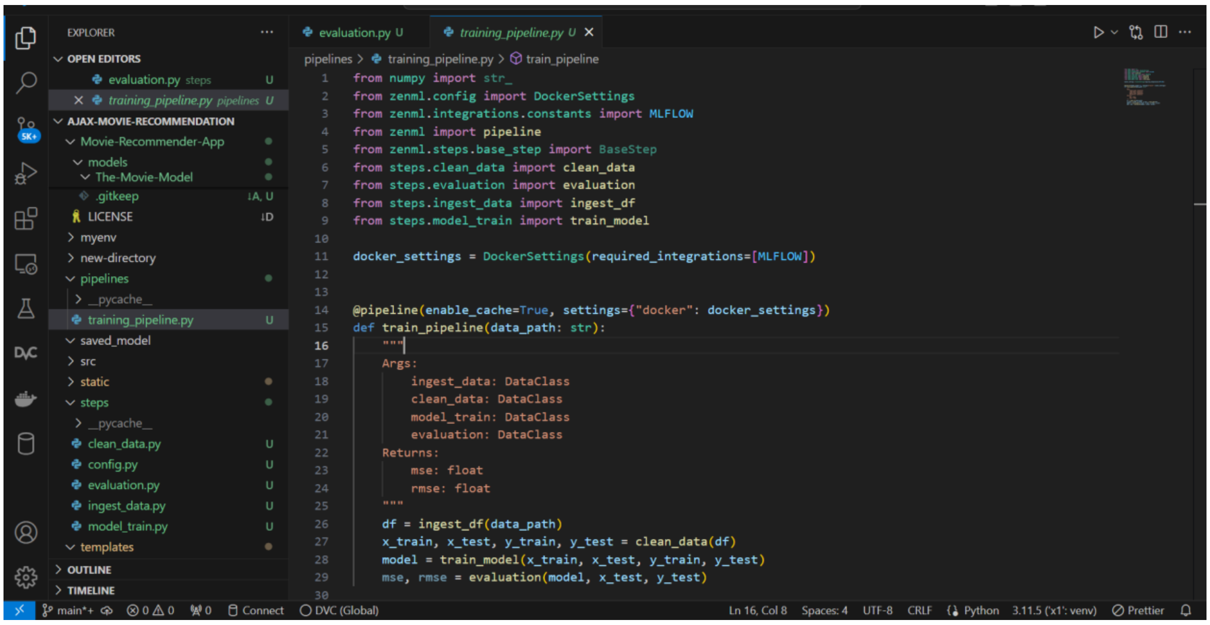
```python
import logging
import pandas as pd
from zenml import step

class IngestData:

    def __init__(self, data_path: str):
        self.data_path = data_path

    def get_data(self):
        logging.info(f"Ingesting data from {self.data_path}")
        # Read the CSV file
        df = pd.read_csv(self.data_path)

        # Convert columns to appropriate data types

        df['title_year'] = df['title_year'].astype('string')  # Assuming 'director_name' is a string
        df['director_facebook_likes'] = df['director_facebook_likes'].astype('string')  # Assuming 'actor_


        return df

@step
def ingest_df(data_path: str) -> pd.DataFrame:
    """
    Args:
        data_path: str: Path to the data file.
    Returns:
        df: pd.DataFrame: DataFrame containing the ingested data.
    """
```


evaluation.py U
ingest_data.py U
steps > ingest_data.py > IngestData > get_data
Ln 10, Col 24   Spaces: 4   UTF-8   CRLF   Python   3.11.5 ('x1': venv)   Prettier

```python
class IngestData:
    def get_data(self):


        return df

@step
def ingest_df(data_path: str) -> pd.DataFrame:
    """
    Args:
        data_path: str: Path to the data file.
    Returns:
        df: pd.DataFrame: DataFrame containing the ingested data.
    """
    try:
        ingest = IngestData(data_path)
        df = ingest.get_data()
        return df
    except Exception as e:
        logging.error(e)
        raise e
```



```python

@step
def train_model(
    x_train: Union[pd.DataFrame, None],
    x_test: Union[pd.DataFrame, None],
    y_train: Union[pd.Series, None],
    y_test: Union[pd.Series, None],
    config: ModelNameConfig,
) -> RegressorMixin:
    """

    Args:
        x_train: pd.DataFrame or None
        x_test: pd.DataFrame or None
        y_train: pd.Series or None
        y_test: pd.Series or None
    Returns:
        model: RegressorMixin
    """
    try:
        model = None
        tuner = None

        if config.model_name == "lightgbm":
            mlflow.lightgbm.autolog()
            model = LightGBMModel()
        elif config.model_name == "randomforest":
            mlflow.sklearn.autolog()
            model = RandomForestModel()
        elif config.model_name == "xgboost":
            mlflow.xgboost.autolog()
```



```python
from numpy import str_
from zenml.config import DockerSettings
from zenml.integrations.constants import MLFLOW
from zenml import pipeline
from zenml.steps.base_step import BaseStep
from steps.clean_data import clean_data
from steps.evaluation import evaluation
from steps.ingest_data import ingest_df
from steps.model_train import train_model

docker_settings = DockerSettings(required_integrations=[MLFLOW])


@pipeline(enable_cache=True, settings={"docker": docker_settings})
def train_pipeline(data_path: str):
    """
    Args:
        ingest_data: DataClass
        clean_data: DataClass
        model_train: DataClass
        evaluation: DataClass
    Returns:
        mse: float
        rmse: float
    """
    df = ingest_df(data_path)
    x_train, x_test, y_train, y_test = clean_data(df)
    model = train_model(x_train, x_test, y_train, y_test)
    mse, rmse = evaluation(model, x_test, y_test)

```

The only problem that this has is that it takes a long time to do the training which is not very optiomal but I do believe that this problem can be solved by switching from a cpu to a gpu

- **Data Validation and Verification**

Data validation and verification improve the quality and consistency of datasets used in training and evaluation.

ZenML: ZenML's Data Drift Detector component monitors data drift and distribution changes over time, allowing for early identification of data quality concerns.

Advantages:

ZenML automatically analyzes data quality and notifies users to potential concerns.

Adaptive Thresholds: Allows you to specify adaptive thresholds for data drift detection, which ensures sensitivity to changes in data distribution.

Disadvantage: not enough documentation.

- **Preprocessing and Feature Engineering**

Data Validation & Verification Tool: Great Expectations, pandas, and numpy

Why are there such high expectations?

Great Expectations is an open-source package that includes comprehensive capabilities for data validation and profiling. It allows for the formulation, maintenance, and assessment of data expectations (assertions), assuring high-quality and consistent datasets for training and evaluation.

Advantages:

Comprehensive Validation: With Great Expectations, you can design thorough expectation suites that test data against a variety of criteria, including data types, ranges, missing values, and unique restrictions.

Automated Documentation: Creates concise, human-readable documentation based on expectations that can be shared with stakeholders to preserve data quality transparency.

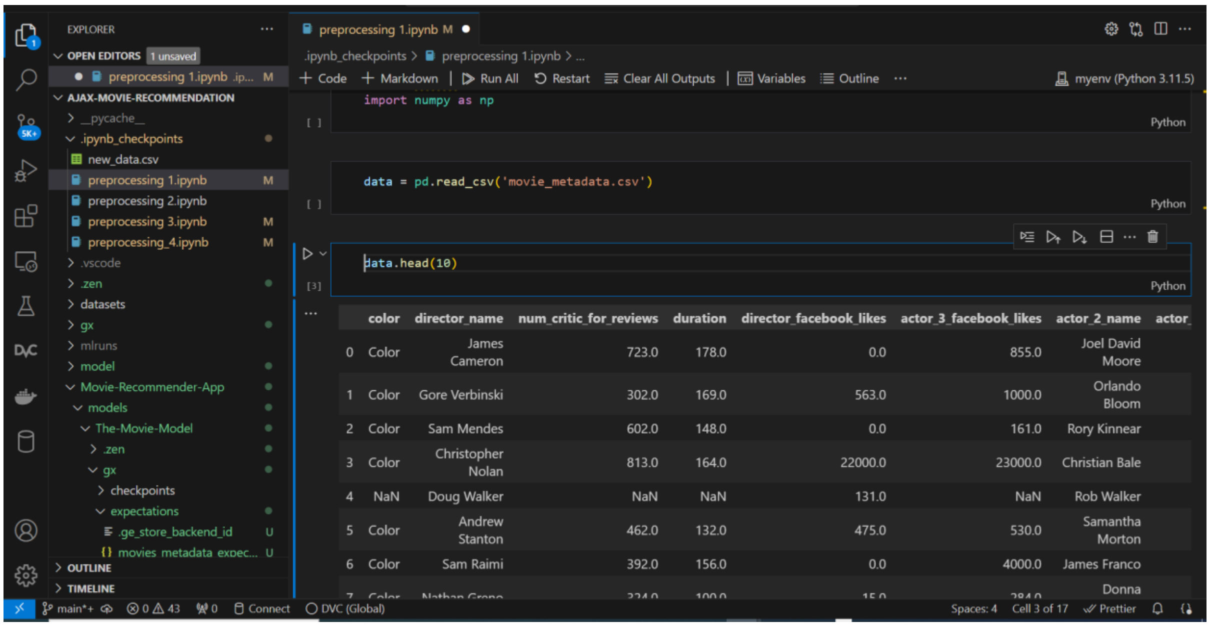
Integration Capabilities: It seamlessly connects with a variety of data sources and pipeline architectures, making it adaptable to varied situations.

Historical Traceability: Keeps a record of data validation outcomes, which is essential for auditing and compliance.

Pandas and NumPy are two important libraries in the Python environment for data processing and numerical computing, respectively. They are critical in preprocessing and feature engineering, offering reliable and fast methods for dealing with big datasets and executing complicated transformations. Pandas provides strong data structures like as DataFrames, which facilitate data manipulation, cleaning, and transformation. It also has a variety of features for merging, joining, reshaping, and aggregating data. Pandas' straightforward syntax and comprehensive capability make data preparation chores simple, even for individuals with little programming knowledge.

NumPy, on the other hand, supports massive, multidimensional arrays and matrices, as well as a set of mathematical algorithms for rapidly manipulating these arrays. NumPy operations are highly optimized, and they can do large-scale numerical computations significantly quicker than standard Python. It also integrates smoothly with other scientific libraries in Python, making it an indispensable component of the data science toolset.

Pandas is used to manage missing values, duplicate records, and data discrepancies. It enables data type conversions and normalization. Pandas and NumPy are used to extract new features from existing data, such as categorical variable encoding, numerical feature scaling, and interaction term generation.

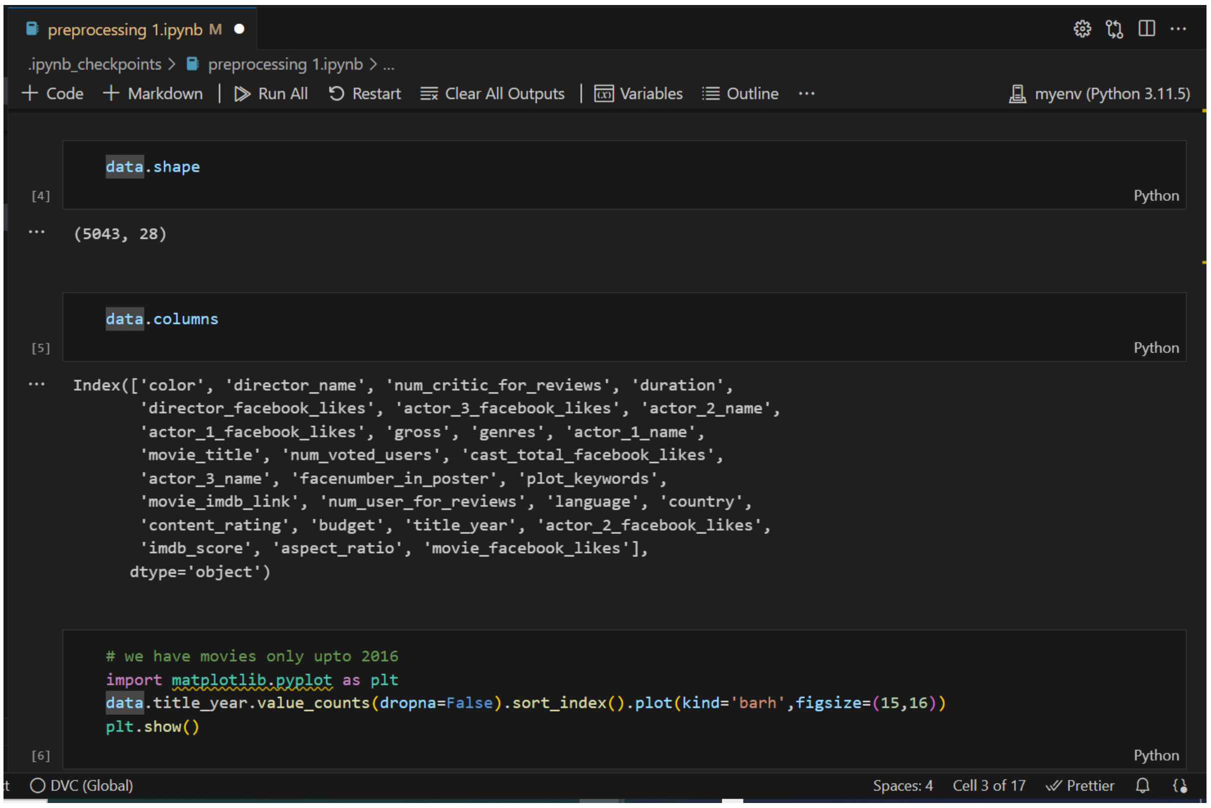


```python
import numpy as np
```

```python
data = pd.read_csv('movie_metadata.csv')
```

```python
data.head(10)
```

	color	director_name	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes	actor_2_name	actor_
0	Color	James Cameron	723.0	178.0	0.0	855.0	Joel David Moore	
1	Color	Gore Verbinski	302.0	169.0	563.0	1000.0	Orlando Bloom	
2	Color	Sam Mendes	602.0	148.0	0.0	161.0	Rory Kinnear	
3	Color	Christopher Nolan	813.0	164.0	22000.0	23000.0	Christian Bale	
4	NaN	Doug Walker	NaN	NaN	131.0	NaN	Rob Walker	
5	Color	Andrew Stanton	462.0	132.0	475.0	530.0	Samantha Morton	
6	Color	Sam Raimi	392.0	156.0	0.0	4000.0	James Franco	
7	Color	Nathan Greno	324.0	100.0	15.0	284.0	Donna	


preprocessing 1.ipynb M
.ipynb_checkpoints > preprocessing 1.ipynb > ...
+ Code + Markdown | Run All Restart Clear All Outputs | Variables Outline ...
myenv (Python 3.11.5)

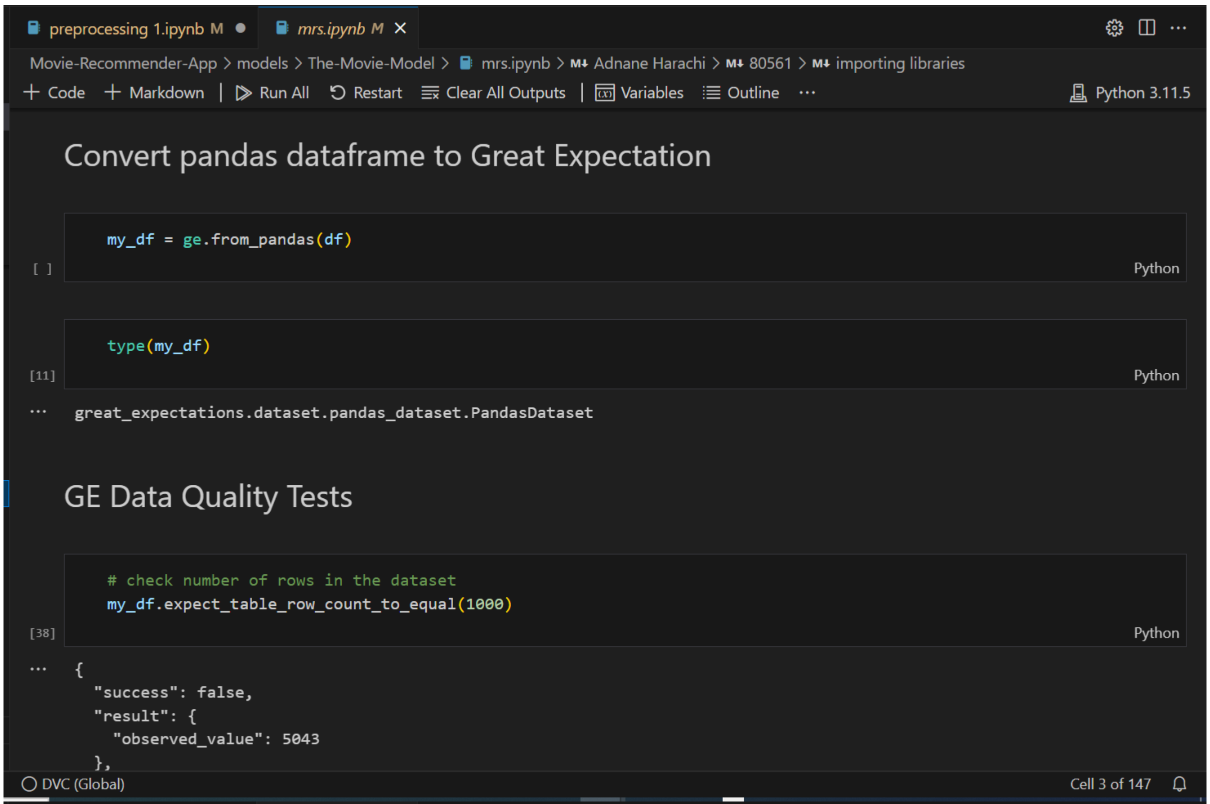
```
[4]
data.shape
```

```
(5043, 28)
```

```
[5]
data.columns
```

```
Index(['color', 'director_name', 'num_critic_for_reviews', 'duration',
       'director_facebook_likes', 'actor_3_facebook_likes', 'actor_2_name',
       'actor_1_facebook_likes', 'gross', 'genres', 'actor_1_name',
       'movie_title', 'num_voted_users', 'cast_total_facebook_likes',
       'actor_3_name', 'facenumber_in_poster', 'plot_keywords',
       'movie_imdb_link', 'num_user_for_reviews', 'language', 'country',
       'content_rating', 'budget', 'title_year', 'actor_2_facebook_likes',
       'imdb_score', 'aspect_ratio', 'movie_facebook_likes'],
      dtype='object')
```

```
[6]
# we have movies only upto 2016
import matplotlib.pyplot as plt
data.title_year.value_counts(dropna=False).sort_index().plot(kind='barh',figsize=(15,16))
plt.show()
```



Convert pandas dataframe to Great Expectation

```
[ ]
my_df = ge.from_pandas(df)
```

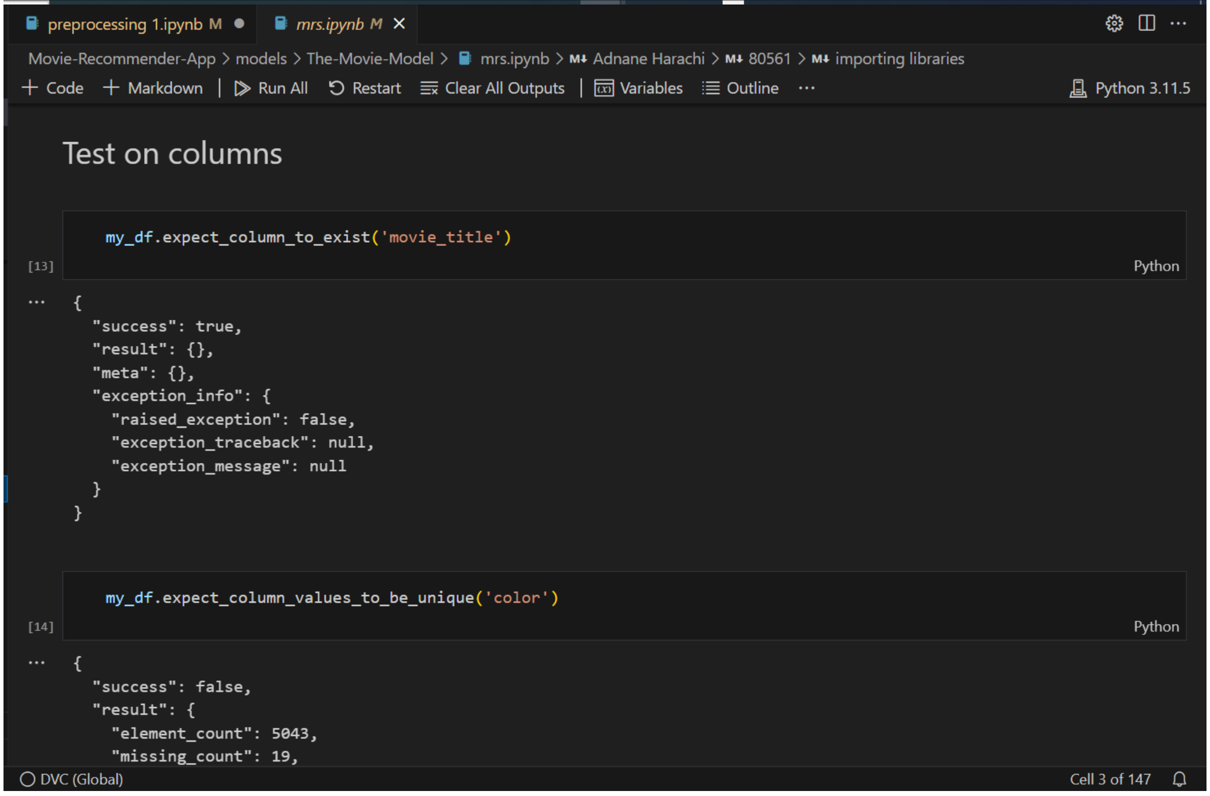
```
[11]
type(my_df)
```

```
great_expectations.dataset.pandas_dataset.PandasDataset
```

GE Data Quality Tests

```
[38]
# check number of rows in the dataset
my_df.expect_table_row_count_to_equal(1000)
```

```
{
  "success": false,
  "result": {
    "observed_value": 5043
  },
```



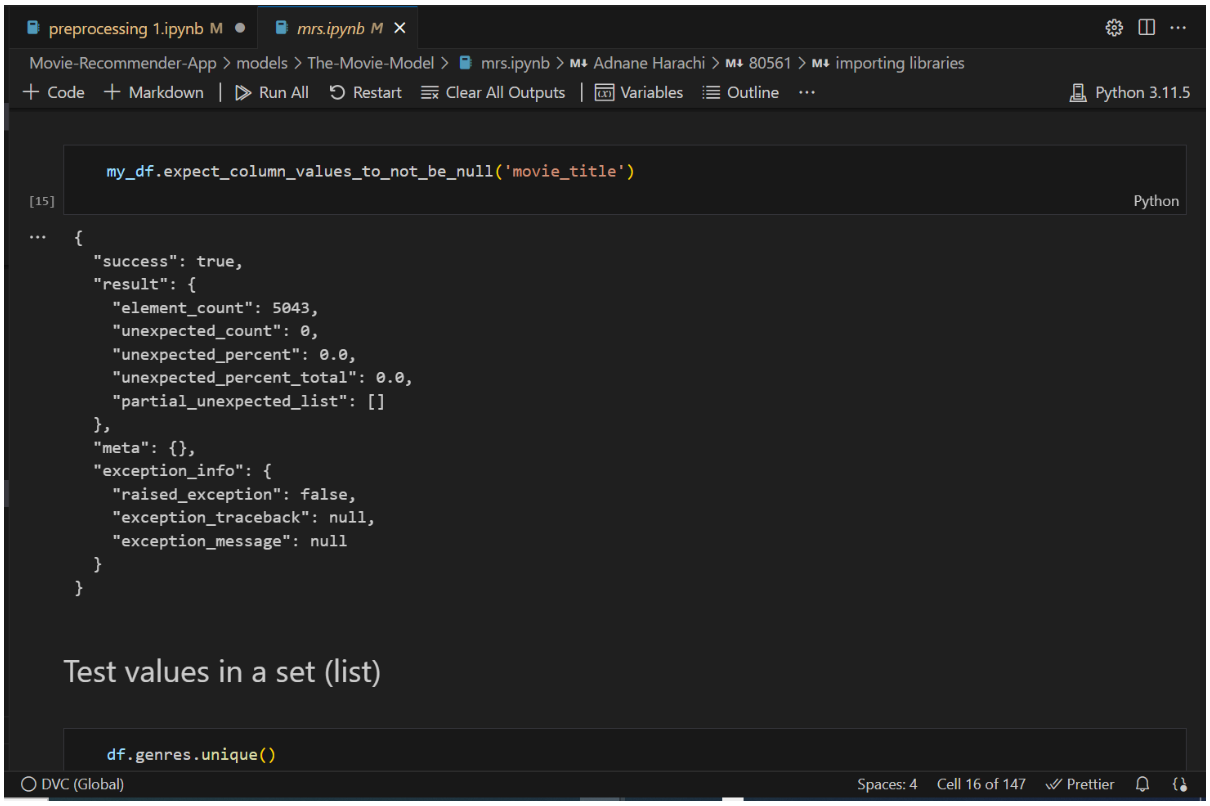
Test on columns

```
[13]
my_df.expect_column_to_exist('movie_title')
```

```
{
  "success": true,
  "result": {},
  "meta": {},
  "exception_info": {
    "raised_exception": false,
    "exception_traceback": null,
    "exception_message": null
  }
}
```

```
[14]
my_df.expect_column_values_to_be_unique('color')
```

```
{
  "success": false,
  "result": {
    "element_count": 5043,
    "missing_count": 19,
```



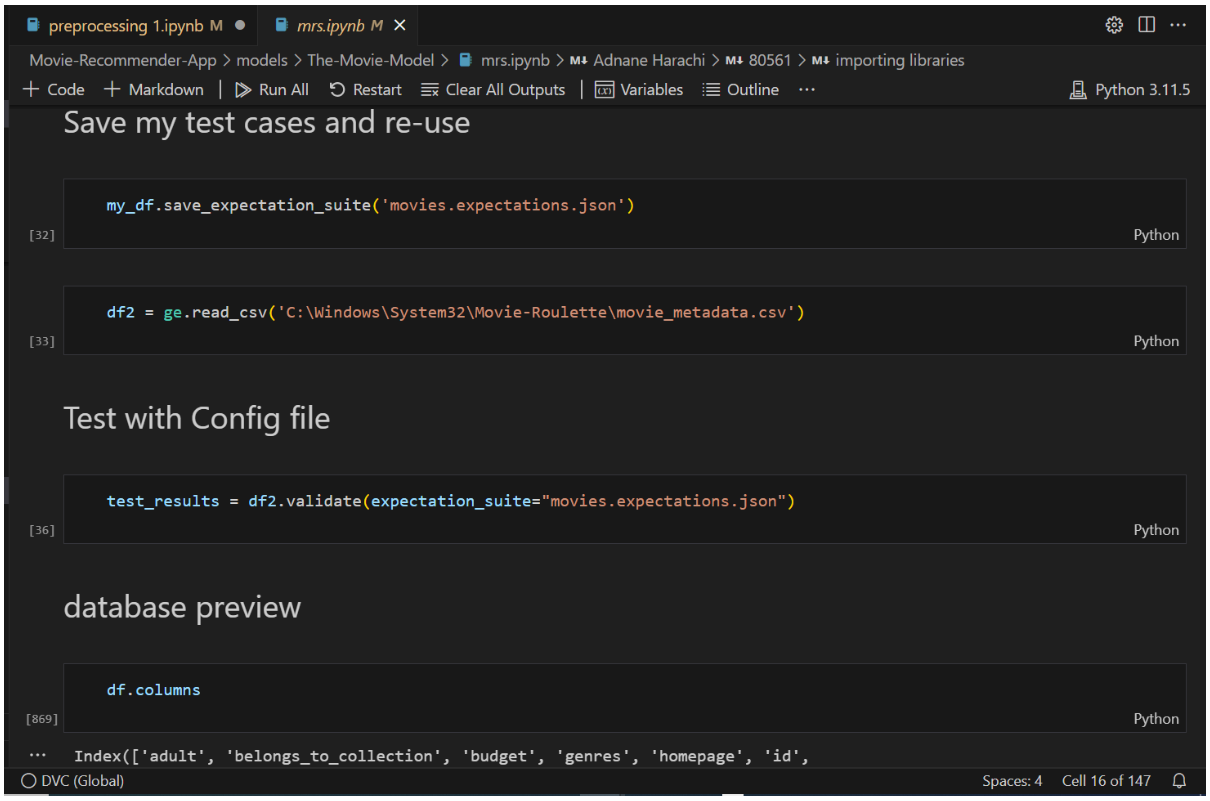
```
my_df.expect_column_values_to_not_be_null('movie_title')
```

[15]

```
{
  "success": true,
  "result": {
    "element_count": 5043,
    "unexpected_count": 0,
    "unexpected_percent": 0.0,
    "unexpected_percent_total": 0.0,
    "partial_unexpected_list": []
  },
  "meta": {},
  "exception_info": {
    "raised_exception": false,
    "exception_traceback": null,
    "exception_message": null
  }
}
```

Test values in a set (list)

```
df.genres.unique()
```



Save my test cases and re-use

```
my_df.save_expectation_suite('movies.expectations.json')
```

[32]

```
df2 = ge.read_csv('C:\Windows\System32\Movie-Roulette\movie_metadata.csv')
```

[33]

Test with Config file

```
test_results = df2.validate(expectation_suite="movies.expectations.json")
```

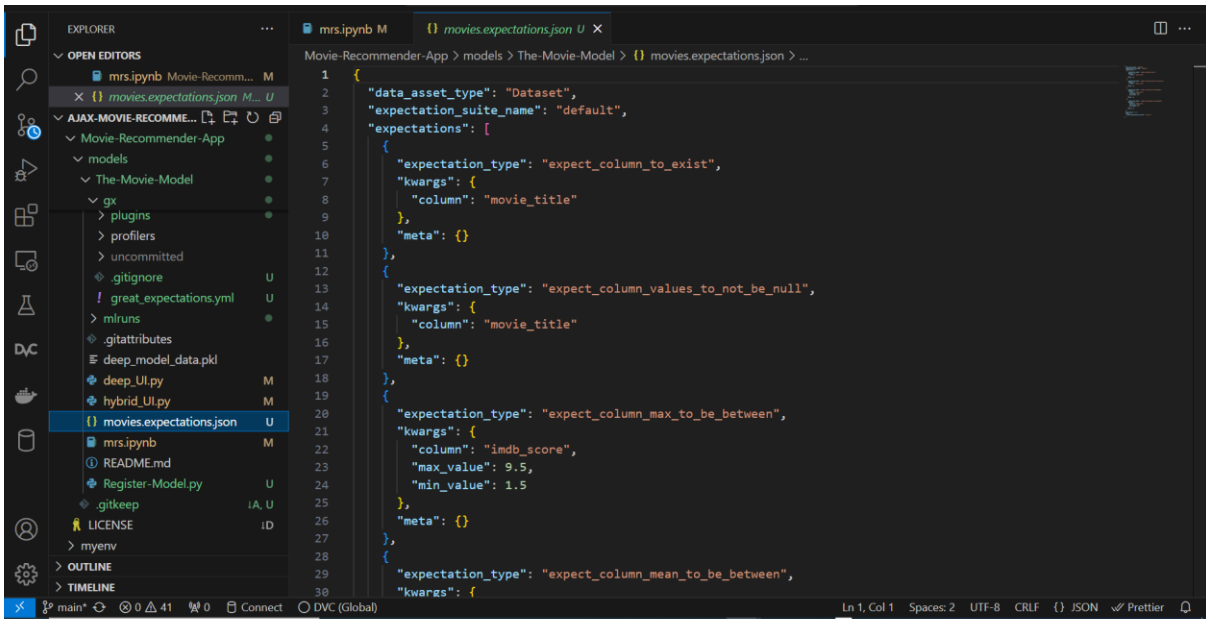
[36]

database preview

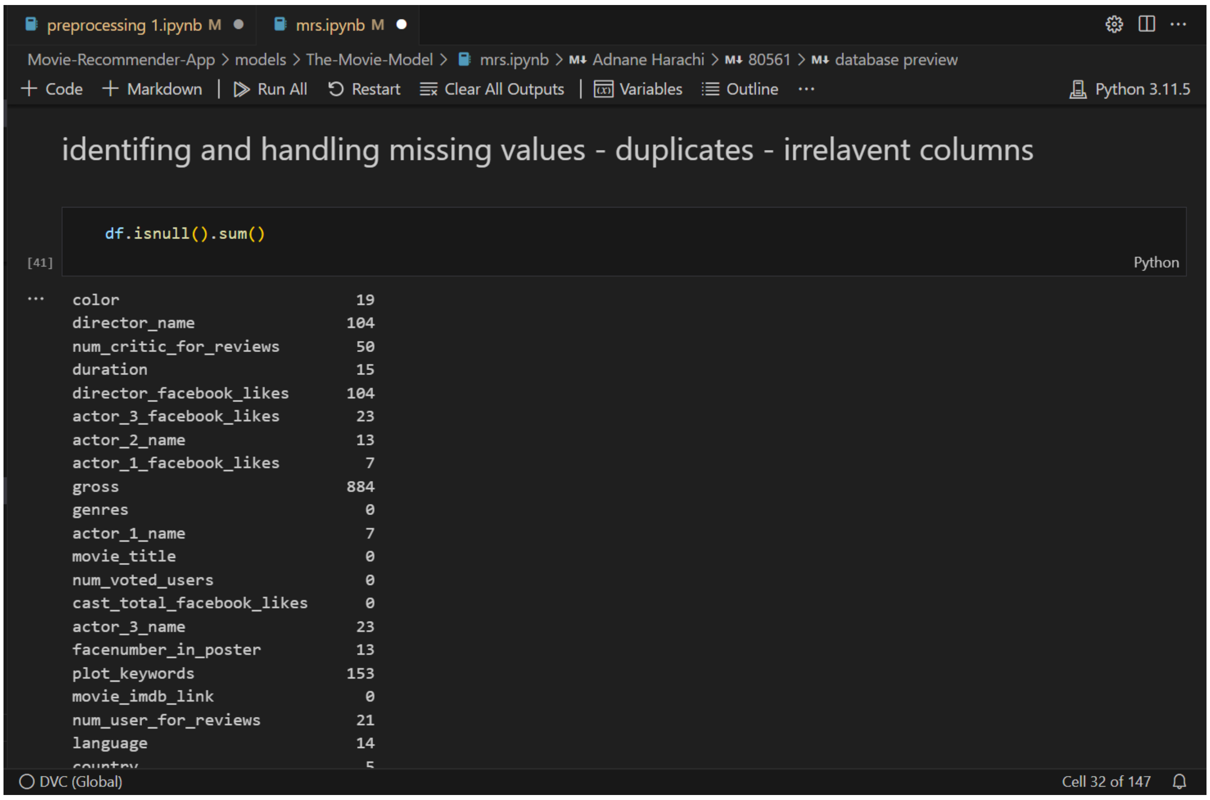
```
df.columns
```

[869]

```
Index(['adult', 'belongs_to_collection', 'budget', 'genres', 'homepage', 'id',
```



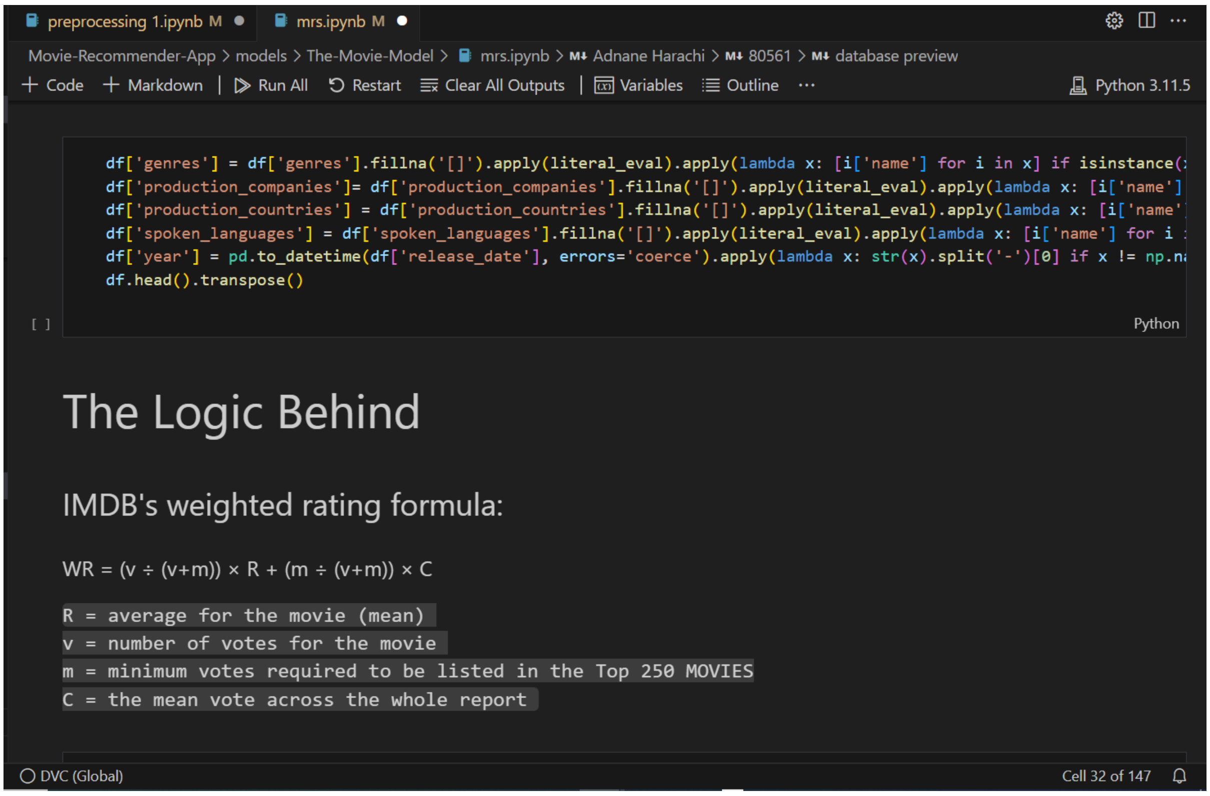
```
{
  "data_asset_type": "Dataset",
  "expectation_suite_name": "default",
  "expectations": [
    {
      "expectation_type": "expect_column_to_exist",
      "kwargs": {
        "column": "movie_title"
      },
      "meta": {}
    },
    {
      "expectation_type": "expect_column_values_to_not_be_null",
      "kwargs": {
        "column": "movie_title"
      },
      "meta": {}
    },
    {
      "expectation_type": "expect_column_max_to_be_between",
      "kwargs": {
        "column": "imdb_score",
        "max_value": 9.5,
        "min_value": 1.5
      },
      "meta": {}
    },
    {
      "expectation_type": "expect_column_mean_to_be_between",
      "kwargs": {
```



identifing and handling missing values - duplicates - irrelavent columns

```
df.isnull().sum()
```

```
color                         19
director_name                104
num_critic_for_reviews        50
duration                      15
director_facebook_likes      104
actor_3_facebook_likes        23
actor_2_name                  13
actor_1_facebook_likes         7
gross                        884
genres                         0
actor_1_name                   7
movie_title                    0
num_voted_users                0
cast_total_facebook_likes      0
actor_3_name                  23
facenumber_in_poster          13
plot_keywords                153
movie_imdb_link                0
num_user_for_reviews          21
language                      14
```



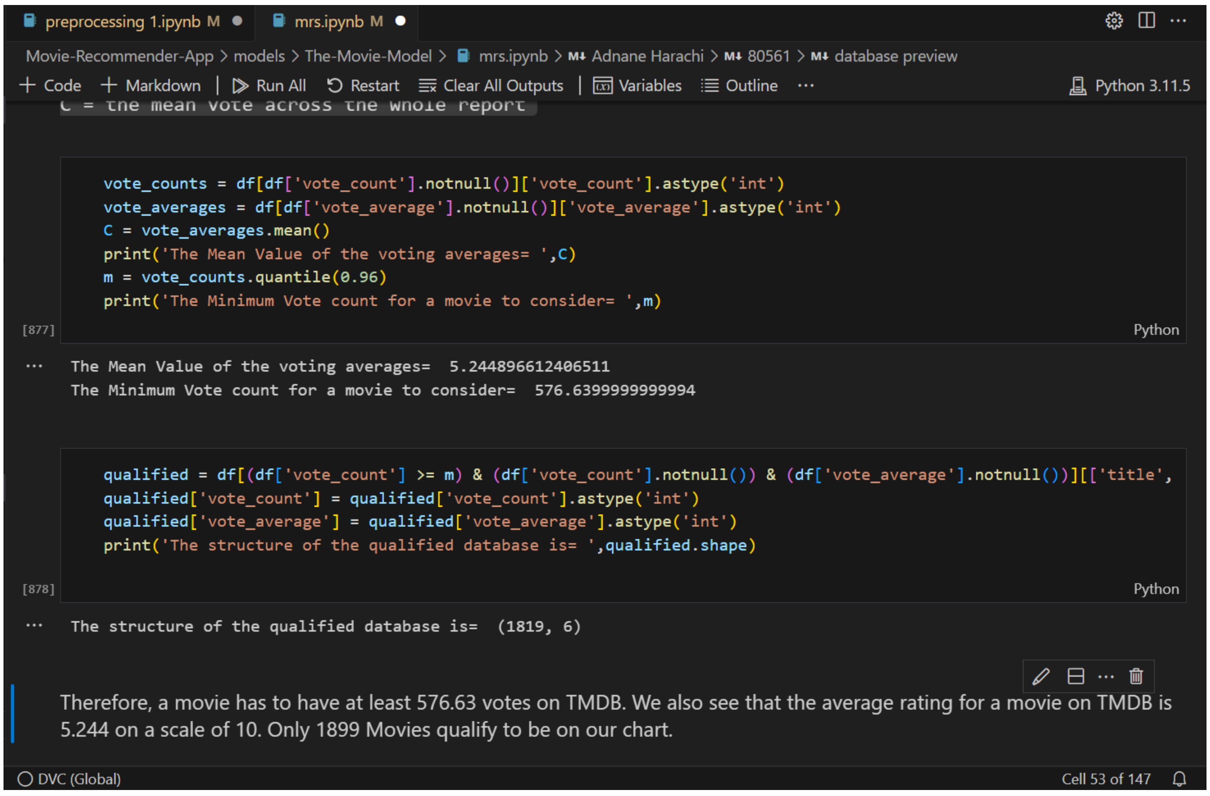
```python
df['genres'] = df['genres'].fillna('[]').apply(literal_eval).apply(lambda x: [i['name'] for i in x] if isinstance(
df['production_companies']= df['production_companies'].fillna('[]').apply(literal_eval).apply(lambda x: [i['name']
df['production_countries'] = df['production_countries'].fillna('[]').apply(literal_eval).apply(lambda x: [i['name'
df['spoken_languages'] = df['spoken_languages'].fillna('[]').apply(literal_eval).apply(lambda x: [i['name'] for i
df['year'] = pd.to_datetime(df['release_date'], errors='coerce').apply(lambda x: str(x).split('-')[0] if x != np.n
df.head().transpose()
```

The Logic Behind

IMDB's weighted rating formula:

WR = (v ÷ (v+m)) × R + (m ÷ (v+m)) × C

```
R = average for the movie (mean)
v = number of votes for the movie
m = minimum votes required to be listed in the Top 250 MOVIES
C = the mean vote across the whole report
```



```python
vote_counts = df[df['vote_count'].notnull()]['vote_count'].astype('int')
vote_averages = df[df['vote_average'].notnull()]['vote_average'].astype('int')
C = vote_averages.mean()
print('The Mean Value of the voting averages= ',C)
m = vote_counts.quantile(0.96)
print('The Minimum Vote count for a movie to consider= ',m)
```

[877]

```
The Mean Value of the voting averages=  5.244896612406511
The Minimum Vote count for a movie to consider=  576.6399999999994
```

```python
qualified = df[(df['vote_count'] >= m) & (df['vote_count'].notnull()) & (df['vote_average'].notnull())][['title',
qualified['vote_count'] = qualified['vote_count'].astype('int')
qualified['vote_average'] = qualified['vote_average'].astype('int')
print('The structure of the qualified database is= ',qualified.shape)
```

[878]

```
The structure of the qualified database is=  (1819, 6)
```

Therefore, a movie has to have at least 576.63 votes on TMDB. We also see that the average rating for a movie on TMDB is 5.244 on a scale of 10. Only 1899 Movies qualify to be on our chart.

The rest is in the notebook provided with the code to not make the report very long.

2.2.2 (5 pts) Integration of model training and offline evaluation into the ML pipeline / MLOps Platform

Model training and offline assessment are successfully integrated into the ML pipeline utilizing ZenML's model training and evaluation components. ZenML manages the whole training process, ensuring that models are trained effectively and that performance is measured using complete offline metrics. One of the primary benefits of utilizing ZenML for this purpose is its strong experiment tracking capabilities. ZenML painstakingly monitors model training runs and evaluation results, providing precise information about the performance of various model setups. This tracking is extremely useful for understanding how various model parameters and configurations affect overall performance, allowing data scientists to make educated judgments regarding model tweaks and enhancements. Furthermore, ZenML's scalability is a valuable advantage, allowing the model training and assessment procedures to be scaled up as necessary. This scalability enables large-scale experimentation, which is required for the development and refinement of high-quality machine learning models. By allowing for prolonged experimentation, ZenML assures that the ML pipeline can manage larger volumes of data and more complicated model architectures, resulting in more accurate and efficient models. Overall, using ZenML to integrate model training and offline assessment into the ML pipeline improves the machine learning workflow's resilience and efficacy.

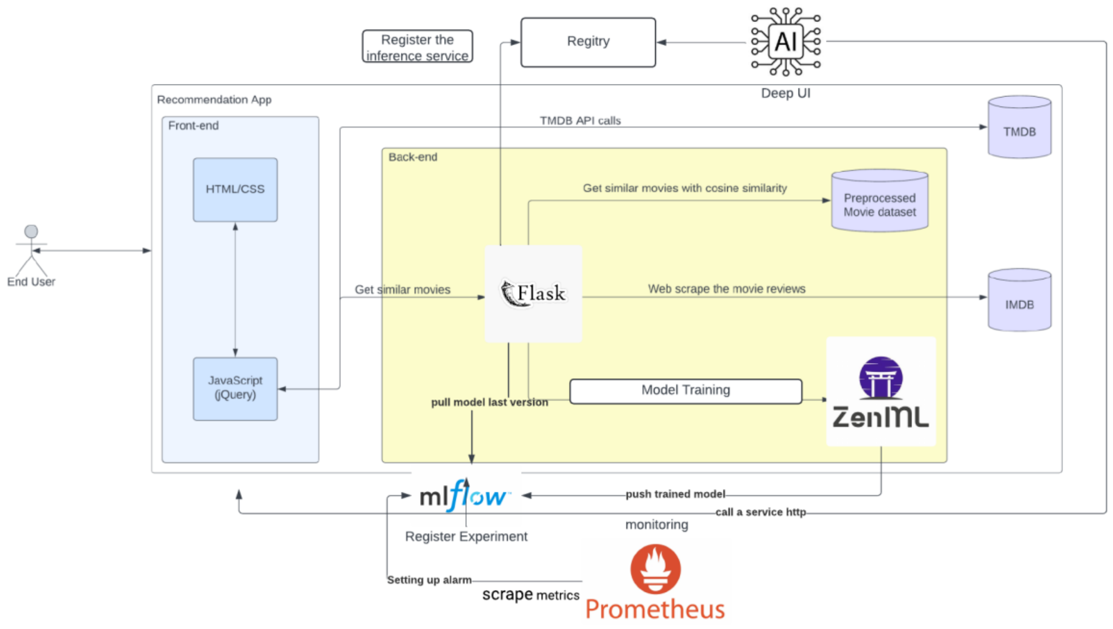
2.2.3 Development of model behavioral tests

The establishment of model behavioral tests is an important step in ensuring that models perform as intended and satisfy business objectives. To do this, the ZenML testing framework is used to create and run thorough model behavioral tests. These tests check model predictions using stated criteria, ensuring that they are in line with predefined business goals. One of the primary benefits of utilizing ZenML for this purpose is the ability to automate model behavioral testing. This automation enables ongoing validation of model behavior, guaranteeing that models operate as expected even when updated or retrained. Furthermore, ZenML enables the creation of customisable metrics that may be used to assess model performance against specific business objectives. This flexibility guarantees that the tests are relevant and properly connected with the business's specific objectives, resulting in meaningful insights regarding model performance. Overall, using ZenML to build and execute model behavioral tests improves the reliability and efficacy of machine learning models by ensuring they satisfy business objectives while maintaining high performance requirements.

3. Model Deployment

3.1 ML System Architecture

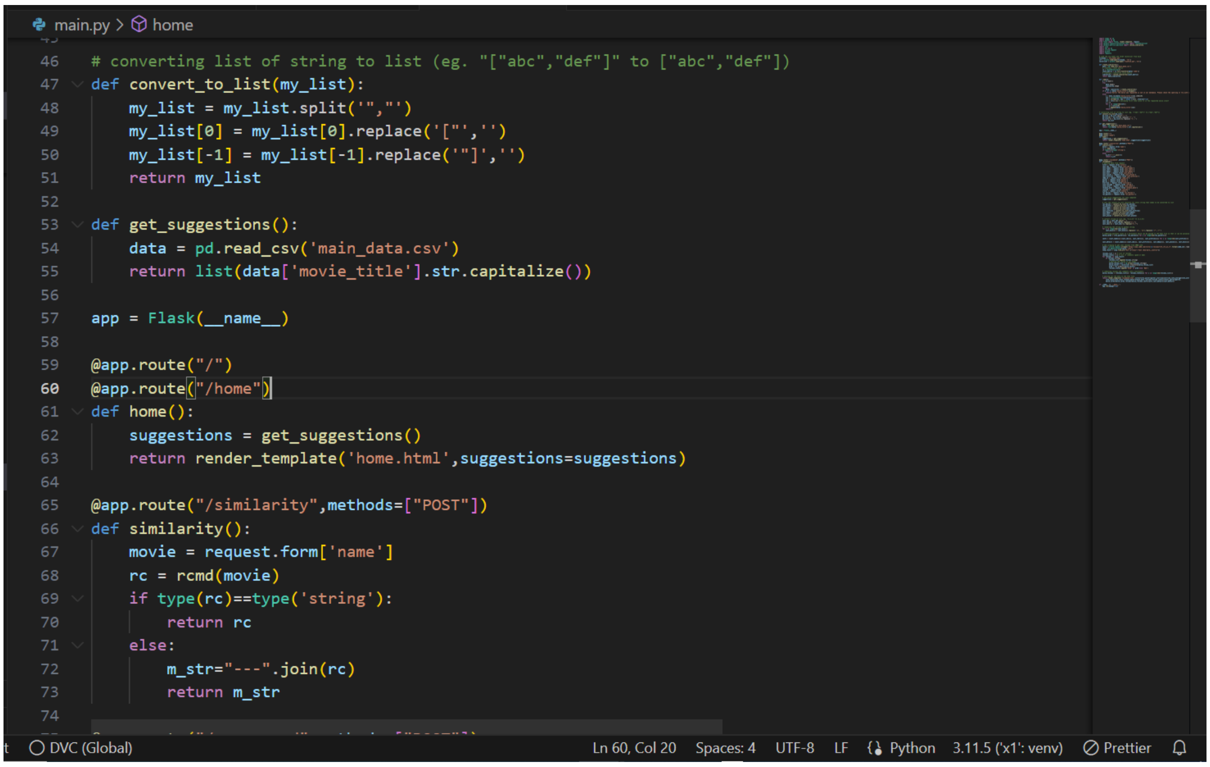
- (5 pts) Drawing with architecture highlights



3.2 Application development

- (5 pts) Model service development

Flask is the core framework for building the model service, offering a fast and lightweight method for delivering machine learning models. Its simplicity and adaptability make it an ideal candidate for creating the backend infrastructure that handles model inference requests. Using Flask's lightweight nature, developers may quickly create microservices suited to certain model functionality. This simplicity not only speeds up development, but also improves maintainability and extensibility as the project progresses. Furthermore, Flask has scalability features, allowing applications to be horizontally extended by adding more instances behind a load balancer. This scalability assures that the model service can successfully manage rising traffic and workload needs while maintaining peak performance even under heavy load. Overall, Flask's lightweight design, simplicity, and scalability make it an excellent framework for developing robust and scalable model services, allowing machine learning models to be seamlessly integrated into production settings.



main.py > home

```python
# converting list of string to list (eg. "["abc","def"]" to ["abc","def"])
def convert_to_list(my_list):
    my_list = my_list.split('","')
    my_list[0] = my_list[0].replace('["','')
    my_list[-1] = my_list[-1].replace('"]','')
    return my_list

def get_suggestions():
    data = pd.read_csv('main_data.csv')
    return list(data['movie_title'].str.capitalize())

app = Flask(__name__)

@app.route("/")
@app.route("/home")
def home():
    suggestions = get_suggestions()
    return render_template('home.html',suggestions=suggestions)

@app.route("/similarity",methods=["POST"])
def similarity():
    movie = request.form['name']
    rc = rcmd(movie)
    if type(rc)==type('string'):
        return rc
    else:
        m_str="---".join(rc)
        return m_str
```

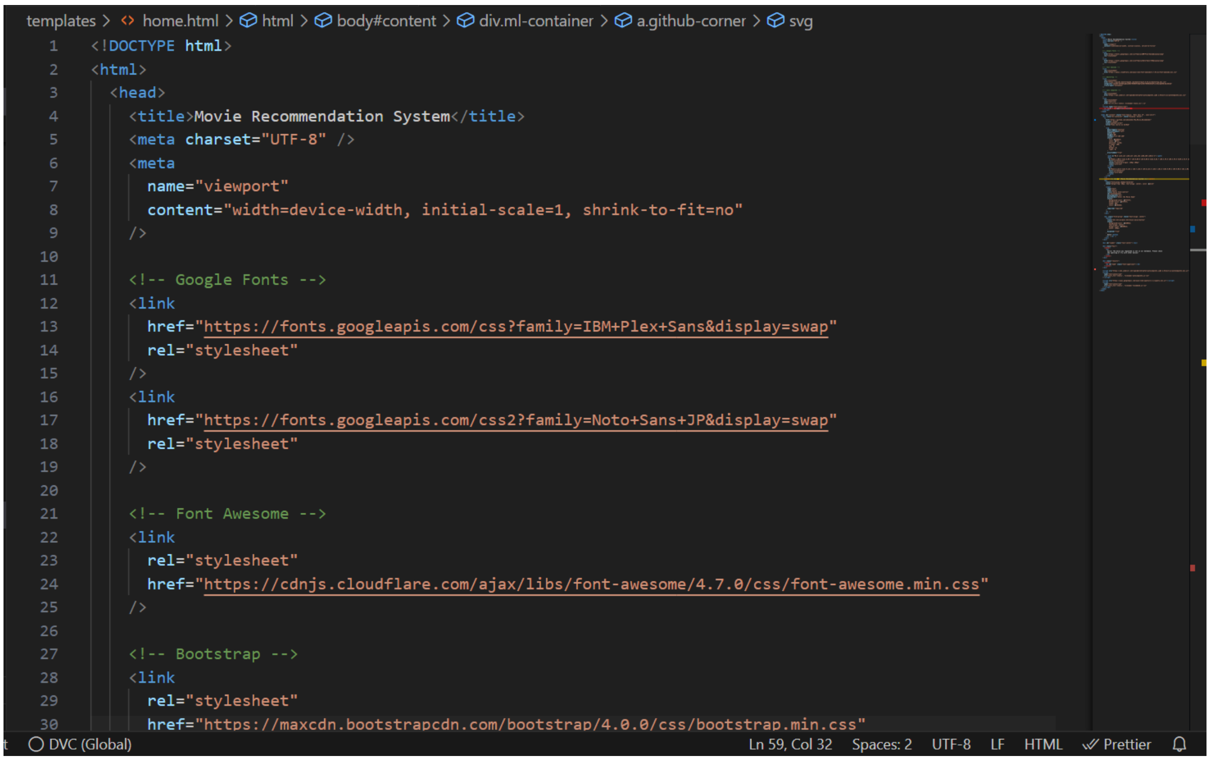
DVC (Global)   Ln 60, Col 20   Spaces: 4   UTF-8   LF   Python   3.11.5 ('x1': venv)   Prettier

The code uses Flask to provide a model service that effectively processes model inference queries. Flask's lightweight and basic foundation make it an excellent choice for building the backend infrastructure that serves machine learning models. Using Flask's simplicity and agility, developers can easily create microservices that serve specialized model functionality. Flask also has scalability features, which allow applications to be horizontally scaled by deploying extra instances behind a load balancer. This scalability assures that the model service can successfully manage rising traffic and workload needs while maintaining peak performance even under heavy load. Overall, Flask's lightweight design, simplicity, and scalability make it an effective framework for building robust and scalable model services, allowing for easy integration of machine learning models into production systems.

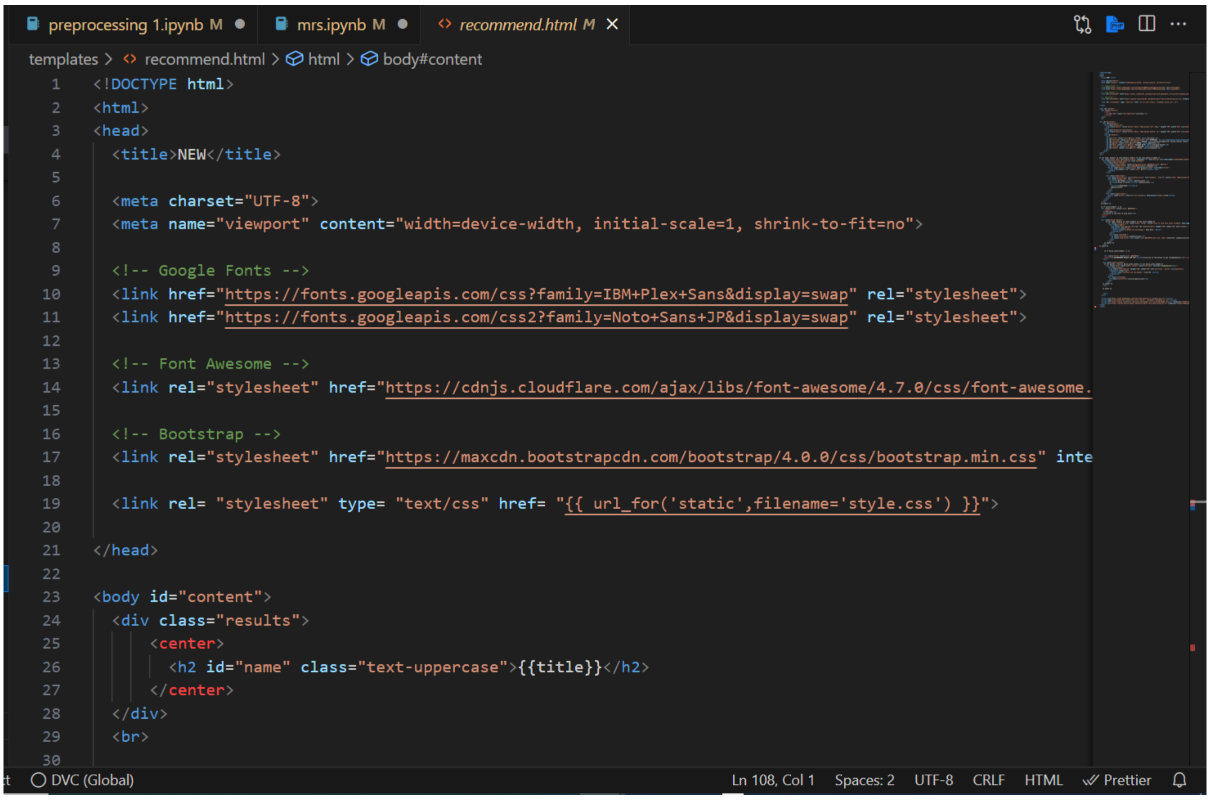
- **(5 pts) Front-end client development**

Flask is an important tool in front-end client development for developing a dynamic and adaptable web interface. Using Flask's features, developers can quickly create and build interactive user interfaces that improve the user experience. Furthermore, REST requests are used to simplify communication between the front-end and backend, providing smooth integration and data transmission. This strategy has several advantages. For starters, by separating the frontend and backend components, the development and maintenance procedures are simplified and can be carried out independently. This modularity allows teams to focus on certain areas of the program without interfering with one another's work, hence increasing efficiency and productivity.

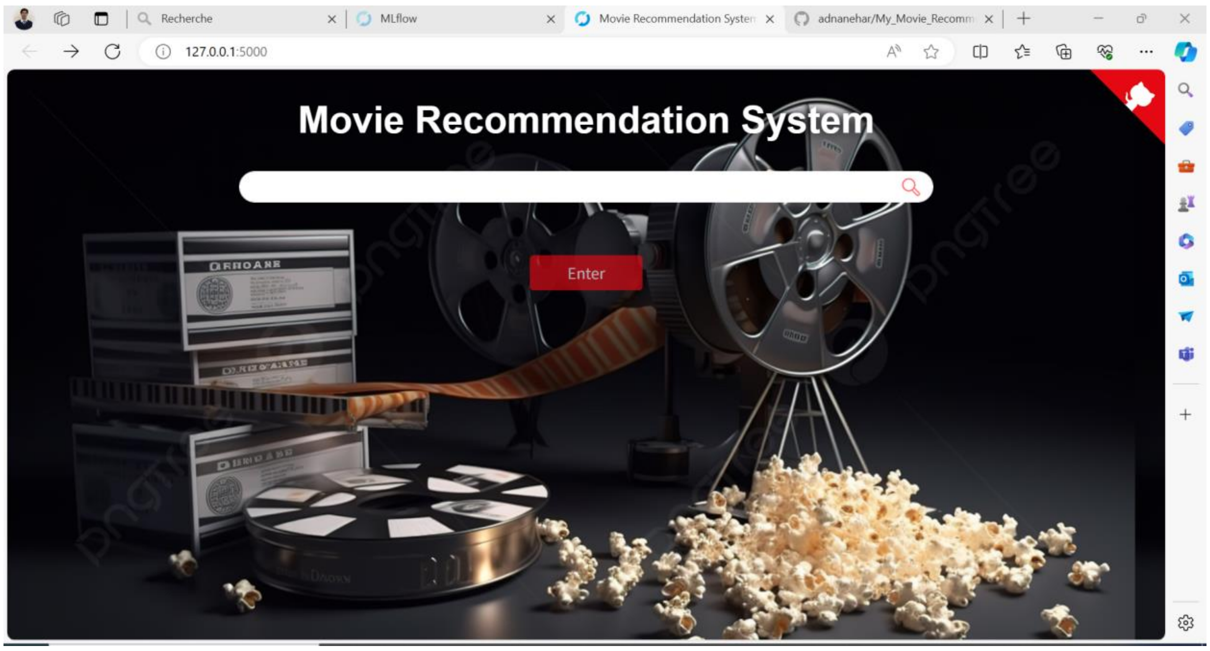
For starters, the separation of the frontend and backend allows for independent development and maintenance, letting developers to focus on their own areas without interfering with each other's work. Furthermore, the usage of RESTful APIs for communication encourages flexibility and extensibility, allowing for seamless integration and development of several components. The front-end client may create a visually beautiful and intuitive user interface using HTTP requests and CSS style, increasing user engagement and satisfaction. Overall, using Flask for front-end development and RESTful APIs for communication results in a flexible, scalable, and responsive web interface that caters to the demands of both developers and users.

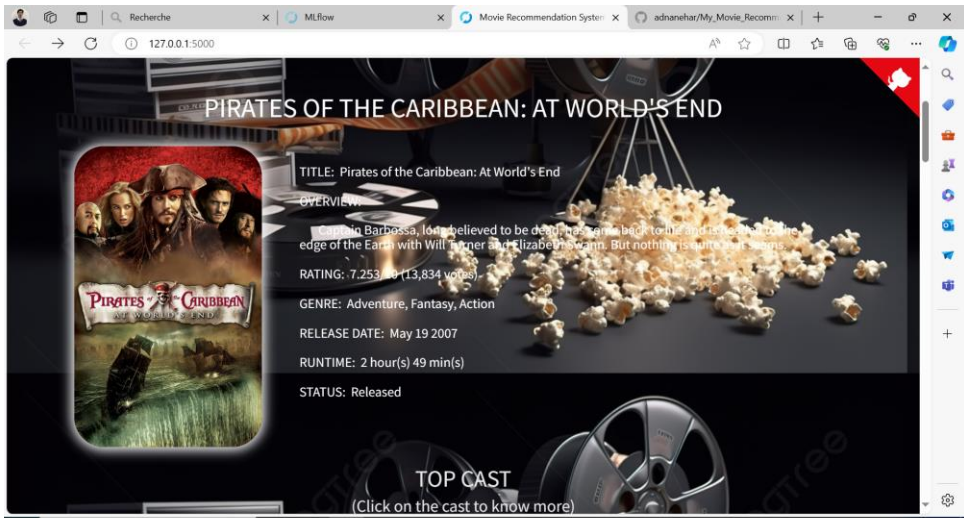

templates > home.html > html > body#content > div.ml-container > a.github-corner > svg

```html
<!DOCTYPE html>
<html>
  <head>
    <title>Movie Recommendation System</title>
    <meta charset="UTF-8" />
    <meta
      name="viewport"
      content="width=device-width, initial-scale=1, shrink-to-fit=no"
    />

    <!-- Google Fonts -->
    <link
      href="https://fonts.googleapis.com/css?family=IBM+Plex+Sans&display=swap"
      rel="stylesheet"
    />
    <link
      href="https://fonts.googleapis.com/css2?family=Noto+Sans+JP&display=swap"
      rel="stylesheet"
    />

    <!-- Font Awesome -->
    <link
      rel="stylesheet"
      href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css"
    />

    <!-- Bootstrap -->
    <link
      rel="stylesheet"
      href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
```


preprocessing 1.ipynb M
mrs.ipynb M
recommend.html M
templates > recommend.html > html > body#content
DVC (Global)
Ln 108, Col 1 Spaces: 2 UTF-8 CRLF HTML Prettier

```html
<!DOCTYPE html>
<html>
<head>
  <title>NEW</title>

  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

  <!-- Google Fonts -->
  <link href="https://fonts.googleapis.com/css?family=IBM+Plex+Sans&display=swap" rel="stylesheet">
  <link href="https://fonts.googleapis.com/css2?family=Noto+Sans+JP&display=swap" rel="stylesheet">

  <!-- Font Awesome -->
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.

  <!-- Bootstrap -->
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css" inte

  <link rel= "stylesheet" type= "text/css" href= "{{ url_for('static',filename='style.css') }}">

</head>

<body id="content">
  <div class="results">
      <center>
        <h2 id="name" class="text-uppercase">{{title}}</h2>
      </center>
  </div>
  <br>

```


Recherche
MLflow
Movie Recommendation System
adnanehar/My_Movie_Recomm
127.0.0.1:5000
Movie Recommendation System
Enter



PIRATES OF THE CARIBBEAN: AT WORLD'S END

TITLE: Pirates of the Caribbean: At World's End

OVERVIEW:

Captain Barbossa, long believed to be dead, has come back to life and is headed to the edge of the Earth with Will Turner and Elizabeth Swann. But nothing is quite as it seems.

RATING: 7.253/10 (13,834 votes)

GENRE: Adventure, Fantasy, Action

RELEASE DATE: May 19 2007

RUNTIME: 2 hour(s) 49 min(s)

STATUS: Released

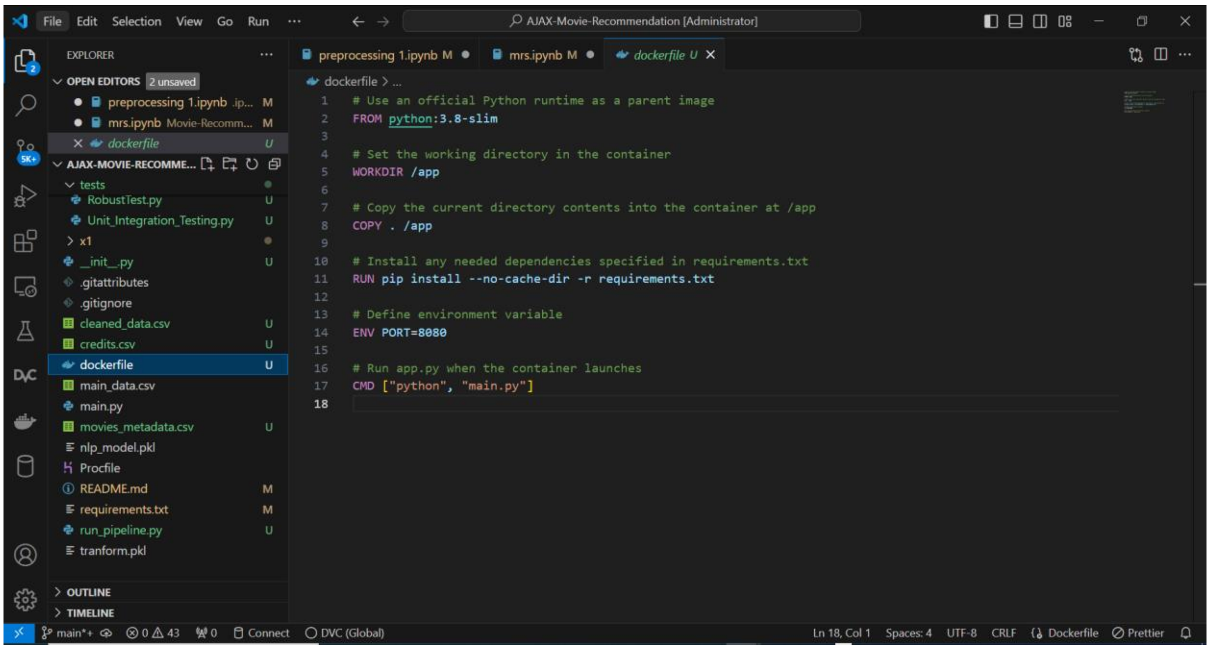
TOP CAST

(Click on the cast to know more)

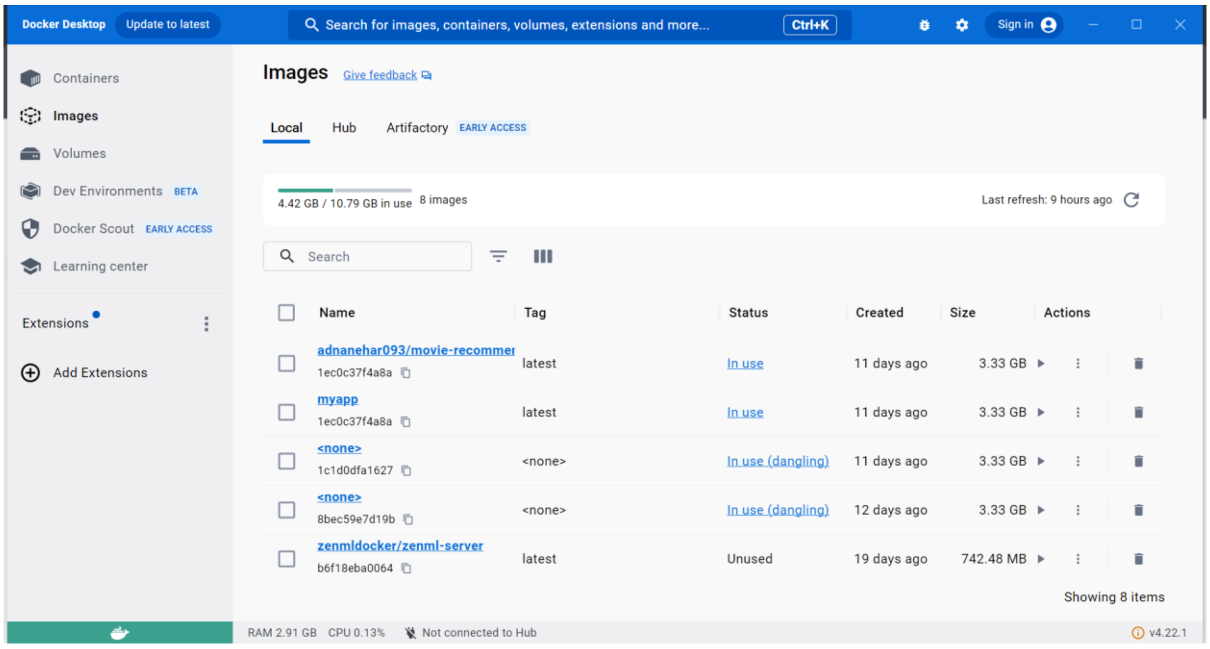
3.3 Integration and Deployment

- Packaging and containerization

Docker is a valuable tool for packaging and containerization since it encapsulates a program and its dependencies in portable containers. These containers create a standardized environment that ensures consistency between the development and production environments. Using Docker's containerization features, the packed application is deployed as a Docker container, allowing for smooth deployment and portability across several platforms. Furthermore, Docker Hub acts as a centralized repository for storing and maintaining Docker images, simplifying the distribution and deployment process. This strategy has several advantages. First, Docker containers are portable, allowing them to run on any platform that supports Docker. This ensures consistent behavior across environments, simplifies the deployment process and reduces compatibility difficulties. Containers also enable isolation, effectively separating processes and resources to avoid conflicts between applications and dependencies. This separation improves security and stability, lowering the chance of system failure and providing consistent performance in a variety of computing scenarios. Overall, Docker's containerization and packaging features, together with Docker Hub's repository management, enable rapid and consistent application deployment while encouraging portability and isolation for increased dependability and scalability.



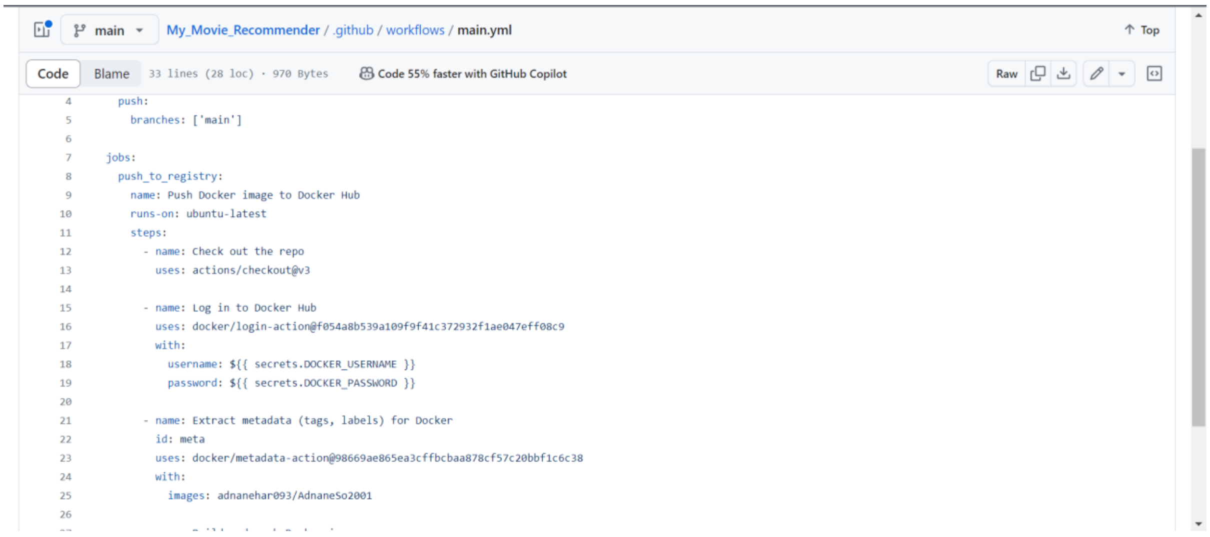
```
# Use an official Python runtime as a parent image
FROM python:3.8-slim

# Set the working directory in the container
WORKDIR /app

# Copy the current directory contents into the container at /app
COPY . /app

# Install any needed dependencies specified in requirements.txt
RUN pip install --no-cache-dir -r requirements.txt

# Define environment variable
ENV PORT=8080

# Run app.py when the container launches
CMD ["python", "main.py"]
```



Name	Tag	Status	Created	Size	Actions
adnanehar093/movie-recommer 1ec0c37f4a8a	latest	In use	11 days ago	3.33 GB	
myapp 1ec0c37f4a8a	latest	In use	11 days ago	3.33 GB	
<none> 1c1d0dfa1627	<none>	In use (dangling)	11 days ago	3.33 GB	
<none> 8bec59e7d19b	<none>	In use (dangling)	12 days ago	3.33 GB	
zenmldocker/zenml-server b6f18eba0064	latest	Unused	19 days ago	742.48 MB	

- **Integration with a CI/CD Pipeline**

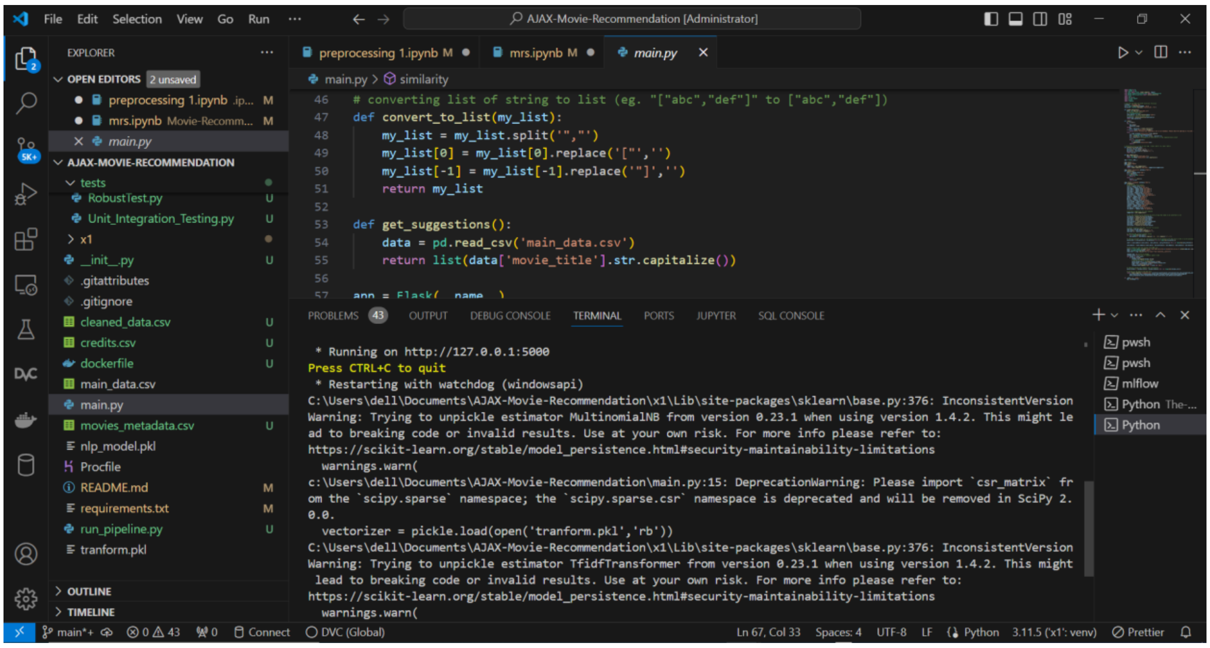
Continuous integration and deployment (CI/CD) play an important role in simplifying the development and deployment processes, resulting in quicker and more reliable releases. This approach relies heavily on GitHub Actions, DockerHub, and Docker Images, all of which help to automate and streamline CI/CD processes. GitHub Actions are used to create CI/CD workflows that automate the build, test, and deployment processes, allowing developers to focus on coding while assuring consistent and dependable releases. DockerHub acts as a centralized repository for Docker images in the CI/CD pipeline, automatically building and pushing them. Docker images contain the program and its dependencies, resulting in a uniform and portable deployment environment. The most current Docker images are then deployed in production environments, ensuring that the application is always up to date and reflects the newest modifications. This strategy has several advantages. For starters, automation using CI/CD pipelines lowers human work and possible errors, increasing productivity and dependability throughout the development process. Furthermore, automated deployments provide consistency across development, testing, and production environments, reducing the possibility of errors and providing smooth transitions between phases of the deployment process. Overall, integrating GitHub Actions, DockerHub, and Docker Images into the CI/CD pipeline improves automation, consistency, and dependability, allowing for quicker and more efficient software releases while maintaining high quality standards.


main
My_Movie_Recommender / .github / workflows / main.yml
Top
Code
Blame
33 lines (28 loc) · 970 Bytes
Code 55% faster with GitHub Copilot
Raw

```
  push:
    branches: ['main']

jobs:
  push_to_registry:
    name: Push Docker image to Docker Hub
    runs-on: ubuntu-latest
    steps:
      - name: Check out the repo
        uses: actions/checkout@v3

      - name: Log in to Docker Hub
        uses: docker/login-action@f054a8b539a109f9f41c372932f1ae047eff08c9
        with:
          username: ${{ secrets.DOCKER_USERNAME }}
          password: ${{ secrets.DOCKER_PASSWORD }}

      - name: Extract metadata (tags, labels) for Docker
        id: meta
        uses: docker/metadata-action@98669ae865ea3cffbcbaa878cf57c20bbf1c6c38
        with:
          images: adnanehar093/AdnaneSo2001

```

- **Hosting the application**

Hosting the application locally and deploying it in the cloud are critical stages for ensuring accessibility and scalability. By using Flask, the application can be readily hosted locally for development and testing, giving developers the freedom to iterate rapidly and debug effectively. Running the application locally with Flask is simple: after installing Flask, developers can go to the project directory in the terminal and use the command flask run. This starts a local server, allowing you to view the program using a web browser at http://localhost:5000. Developers may make code changes and see them reflected in real time simply reloading the browser. Furthermore, Flask's built-in development server provides informative error messages and debugging tools, which improves the overall development experience. While the application is now deployed locally, cloud deployment is planned in the future to scale it for production use. Cloud deployment has various advantages, including scalability and dependability. By employing cloud architecture, resources may be dynamically scaled to meet rising demand, providing peak performance even under high load. Cloud solutions also offer powerful security measures, automated backups, and high availability, which reduces downtime and improves the entire user experience. Furthermore, cloud deployment facilitates easy connection with other cloud services such as database management systems, content delivery networks, and monitoring tools, allowing for comprehensive application infrastructure administration and optimization. Overall, the mix of local hosting for development freedom and eventual cloud deployment for scalability assures that the application will satisfy



```python
# converting list of string to list (eg. "["abc","def"]" to ["abc","def"])
def convert_to_list(my_list):
    my_list = my_list.split('","')
    my_list[0] = my_list[0].replace('["','')
    my_list[-1] = my_list[-1].replace('"]','')
    return my_list

def get_suggestions():
    data = pd.read_csv('main_data.csv')
    return list(data['movie_title'].str.capitalize())

app = Flask(__name__)
```

```
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with watchdog (windowsapi)
C:\Users\dell\Documents\AJAX-Movie-Recommendation\x1\Lib\site-packages\sklearn\base.py:376: InconsistentVersion
Warning: Trying to unpickle estimator MultinomialNB from version 0.23.1 when using version 1.4.2. This might le
ad to breaking code or invalid results. Use at your own risk. For more info please refer to:
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
  warnings.warn(
c:\Users\dell\Documents\AJAX-Movie-Recommendation\main.py:15: DeprecationWarning: Please import `csr_matrix` fr
om the `scipy.sparse` namespace; the `scipy.sparse.csr` namespace is deprecated and will be removed in SciPy 2.
0.0.
  vectorizer = pickle.load(open('tranform.pkl','rb'))
C:\Users\dell\Documents\AJAX-Movie-Recommendation\x1\Lib\site-packages\sklearn\base.py:376: InconsistentVersion
Warning: Trying to unpickle estimator TfidfTransformer from version 0.23.1 when using version 1.4.2. This might
 lead to breaking code or invalid results. Use at your own risk. For more info please refer to:
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
  warnings.warn(
```

3.4. Model Serving and online testing

- Model serving runtime

Flask acts as the model's runtime, offering a lightweight and fast framework for displaying predictions. Using Flask's features, the model can be smoothly incorporated into the application's backend, allowing predictions to be delivered quickly and precisely. Flask's lightweight architecture reduces overhead, ensuring that resources are used effectively and predictions are given quickly to end users. This efficiency is especially useful in cases that need real-time predictions, such as recommendation systems or natural language processing applications. Furthermore, Flask's flexibility in creating endpoints and processing requests makes it suitable for a wide range of use cases. Developers may design unique endpoints that meet the application's needs, resulting in a personalized and straightforward user experience. Furthermore, Flask's wide community of extensions and plugins increases its versatility by allowing interaction with a variety of data sources, authentication systems, and third-party services. In addition, Flask's asynchronous features allow it to easily manage several concurrent queries, resulting in responsive and scalable model serving. Flask can manage enormous traffic levels while maintaining speed and responsiveness thanks to asynchronous programming approaches. This scalability is critical for applications with a large user base or unexpected traffic patterns, since it ensures consistent and dependable model performance under changing situations.

- Serving mode (batch, on demand to a human, on demand to a machine)

Serving Mode: On Demand to a Human

The key interaction in Movie Roulette is for users to search for movies based on their tastes and then receive personalized suggestions. Given this user behavior, offering predictions on demand to human users is well aligned with the application's functionality and user experience objectives. Here's why.

Real-time Interaction: Movie suggestions are a dynamic and personalized process that is significantly affected by user feedback and preferences. Serving predictions on-demand enables users to receive real-time suggestions as they engage with the application, improving their entire experience.

User interaction: Movie Roulette promotes active user interaction and discovery by providing predictions on demand. Users may search for movies depending on their current mood, tastes, or particular criteria, resulting in more meaningful interactions and longer time spent on the site.

Tailored Recommendations: Every user has different movie choices, and their search queries reflect their distinct likes and interests. Serving predictions on demand allows Movie Roulette to provide personalized suggestions in response to each user's search, assuring relevance and customisation.

Instant Feedback Loop: Serving predictions on-demand establishes an instant feedback loop between the user's search query and the recommendations made. Users get instant answers based on their input, allowing them to swiftly adjust their search criteria or explore alternate options, increasing their overall happiness with the program.

Scalability and Efficiency: Unlike batch processing, which requires vast amounts of data to be processed at preset intervals, Movie Roulette's on-demand predictions allow it to scale dynamically in response to customer demand. This provides effective resource use and responsiveness, especially during peak demand times.

Adaptability to User purpose: Users' search queries frequently express implicit purpose, such as studying a certain genre, discovering new releases, or locating films comparable to their favorites. Serving predictions on-demand allows Movie Roulette to respond to changing user intentions in real time, offering appropriate suggestions that match user expectations.

- Online testing (A/B Testing, Bandit)

4. Monitoring and Continual Learning

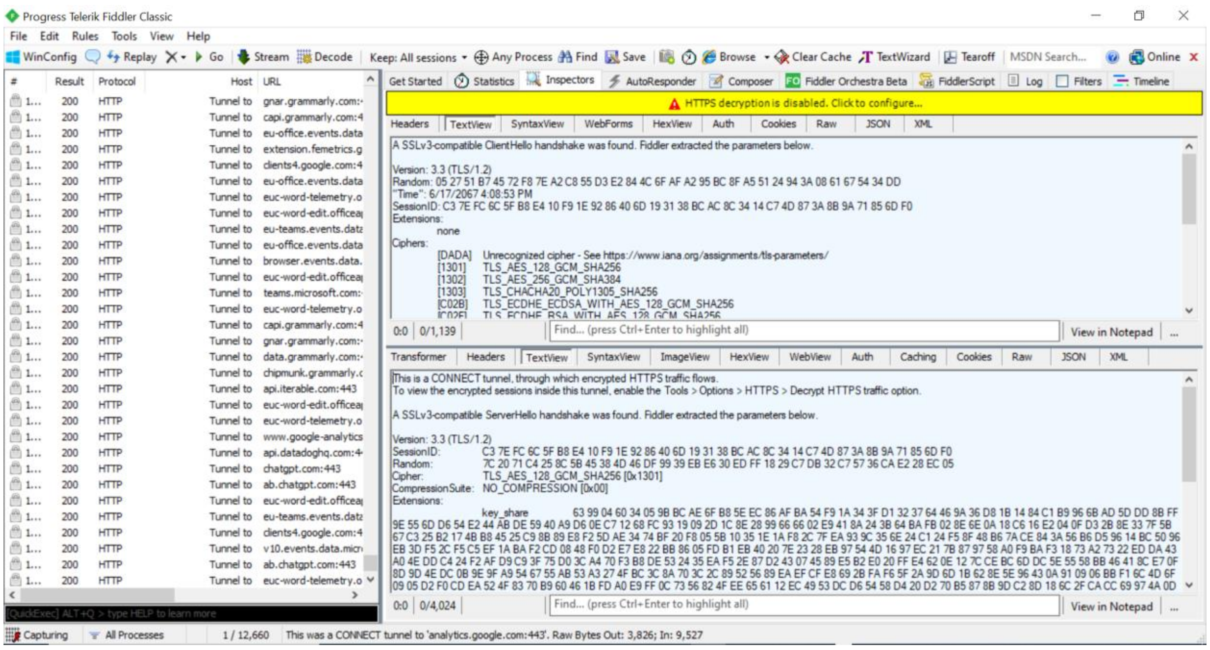
Monitoring and continuous learning are key components of the Movie Roulette machine learning system, as they ensure the model's accuracy, efficiency, and adaptability to changing data and user preferences. Here, we look at the specific tools and approaches used, emphasizing their benefits and the reasoning behind their selection.

4.1. Resource Monitoring

WhyLogs provides crucial real-time monitoring tools, allowing us to detect and resolve issues as they emerge, rather than discovering them later. This proactive monitoring strategy improves the dependability and stability of our machine learning system by allowing us to respond quickly to any abnormalities or deviations from expected behavior. One of WhyLogs' primary benefits is its ability to provide detailed data profiles, such as distributions, missing values, and summary statistics. These detailed profiles are critical for assuring data integrity and quality, as well as offering insights necessary for making educated decisions. Additionally, WhyLogs connects smoothly with our existing data flow, making it simple to include into our monitoring processes. This seamless connection keeps monitoring procedures simplified and efficient, reducing disturbances in our operations. Furthermore, WhyLogs is designed to manage large volumes of data, making it ideal for Movie Roulette's growing user base and data requirements. Its scalability means that our monitoring capabilities can grow alongside our system, supporting growing data volume and user activity while maintaining performance and dependability. WhyLogs allows us to retain consistency and reliability in our data, which is critical for the success of any machine learning system. Consistent and trustworthy data serve as the foundation for accurate and successful machine learning models. As a result, by assuring data integrity through real-time monitoring with WhyLogs, we can maintain our machine learning system's quality and performance, eventually improving the user experience and boosting Movie Roulette's success.

4.2. Model Performance Monitoring or data distribution drift monitoring

Fiddler emerges as the leading platform for monitoring and explaining machine learning models, with a comprehensive set of functions designed to evaluate model performance and detect data drift. These features are critical for guaranteeing the accuracy and relevancy of our movie recommendations in Movie Roulette. One of Fiddler's most notable features is its extensive monitoring of model performance, which includes critical parameters including accuracy, precision, recall, and F1 score. This detailed examination gives us a comprehensive insight of our model's performance and allows us to find areas for improvement. Additionally, Fiddler excels at identifying data distribution drift, using advanced skills to notify us when input data changes that may affect model performance. This proactive approach to monitoring allows us to keep our models accurate and relevant by responding to changing data distributions in real time. Fiddler's explainability skills also play an important role in increasing confidence and openness in our machine learning algorithms. Fiddler enables users to comprehend and analyze model behavior by offering insights into the reasoning behind model predictions, instilling trust in Movie Roulette's suggestions. Furthermore, Fiddler provides extensive warning systems and thorough reporting capabilities, allowing us to keep informed about significant changes or concerns influencing model performance. These alerting systems act as early warning systems, allowing us to handle possible difficulties quickly and maintain the dependability of our movie suggestion engine. Fiddler provides us with the knowledge we need to maximize model performance and make educated decisions by providing detailed reports and in-depth analysis.


Progress Telerik Fiddler Classic
File Edit Rules Tools View Help
WinConfig Replay Go Stream Decode Keep: All sessions Any Process Find Save Browse Clear Cache TextWizard Tearoff MSDN Search... Online
Get Started Statistics Inspectors AutoResponder Composer Fiddler Orchestra Beta FiddlerScript Log Filters Timeline
HTTPS decryption is disabled. Click to configure...
Headers TextView SyntaxView WebForms HexView Auth Cookies Raw JSON XML
Find... (press Ctrl+Enter to highlight all) View in Notepad
Transformer Headers TextView SyntaxView ImageView HexView WebView Auth Caching Cookies Raw JSON XML

#	Result	Protocol	Host	URL
1...	200	HTTP	Tunnel to	gnar.grammarly.com:
1...	200	HTTP	Tunnel to	capi.grammarly.com:4
1...	200	HTTP	Tunnel to	eu-office.events.data
1...	200	HTTP	Tunnel to	extension.femetrics.g
1...	200	HTTP	Tunnel to	clients4.google.com:4
1...	200	HTTP	Tunnel to	eu-office.events.data
1...	200	HTTP	Tunnel to	euc-word-telemetry.o
1...	200	HTTP	Tunnel to	euc-word-edit.officea
1...	200	HTTP	Tunnel to	eu-teams.events.data
1...	200	HTTP	Tunnel to	eu-office.events.data
1...	200	HTTP	Tunnel to	browser.events.data.
1...	200	HTTP	Tunnel to	euc-word-edit.officea
1...	200	HTTP	Tunnel to	teams.microsoft.com:
1...	200	HTTP	Tunnel to	euc-word-telemetry.o
1...	200	HTTP	Tunnel to	capi.grammarly.com:4
1...	200	HTTP	Tunnel to	gnar.grammarly.com:
1...	200	HTTP	Tunnel to	data.grammarly.com:
1...	200	HTTP	Tunnel to	chipmunk.grammarly.c
1...	200	HTTP	Tunnel to	api.iterable.com:443
1...	200	HTTP	Tunnel to	euc-word-edit.officea
1...	200	HTTP	Tunnel to	euc-word-telemetry.o
1...	200	HTTP	Tunnel to	www.google-analytics
1...	200	HTTP	Tunnel to	api.datadoghq.com:4
1...	200	HTTP	Tunnel to	chatgpt.com:443
1...	200	HTTP	Tunnel to	ab.chatgpt.com:443
1...	200	HTTP	Tunnel to	euc-word-edit.officea
1...	200	HTTP	Tunnel to	eu-teams.events.data
1...	200	HTTP	Tunnel to	clients4.google.com:4
1...	200	HTTP	Tunnel to	v10.events.data.micr
1...	200	HTTP	Tunnel to	ab.chatgpt.com:443
1...	200	HTTP	Tunnel to	euc-word-telemetry.o

```
A SSLv3-compatible ClientHello handshake was found. Fiddler extracted the parameters below.

Version: 3.3 (TLS/1.2)
Random: 05 27 51 B7 45 72 F8 7E A2 C8 55 D3 E2 84 4C 6F AF A2 95 BC 8F A5 51 24 94 3A 08 61 67 54 34 DD
"Time": 6/17/2067 4:08:53 PM
SessionID: C3 7E FC 6C 5F B8 E4 10 F9 1E 92 86 40 6D 19 31 38 BC AC 8C 34 14 C7 4D 87 3A 8B 9A 71 85 6D F0
Extensions:
        none
Ciphers:
        [DADA]   Unrecognized cipher - See https://www.iana.org/assignments/tls-parameters/
        [1301]   TLS_AES_128_GCM_SHA256
        [1302]   TLS_AES_256_GCM_SHA384
        [1303]   TLS_CHACHA20_POLY1305_SHA256
        [C02B]   TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
        [C02F]   TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
```

0:0 | 0/1,139

```
This is a CONNECT tunnel, through which encrypted HTTPS traffic flows.
To view the encrypted sessions inside this tunnel, enable the Tools > Options > HTTPS > Decrypt HTTPS traffic option.

A SSLv3-compatible ServerHello handshake was found. Fiddler extracted the parameters below.

Version: 3.3 (TLS/1.2)
SessionID:         C3 7E FC 6C 5F B8 E4 10 F9 1E 92 86 40 6D 19 31 38 BC AC 8C 34 14 C7 4D 87 3A 8B 9A 71 85 6D F0
Random:            7C 20 71 C4 25 8C 5B 45 38 4D 46 DF 99 39 EB E6 30 ED FF 18 29 C7 DB 32 C7 57 36 CA E2 28 EC 05
Cipher:            TLS_AES_128_GCM_SHA256 [0x1301]
CompressionSuite:  NO_COMPRESSION [0x00]
Extensions:
                   key_share          63 99 04 60 34 05 9B BC AE 6F B8 5E EC 86 AF BA 54 F9 1A 34 3F D1 32 37 64 46 9A 36 D8 1B 14 84 C1 B9 96 6B AD 5D DD 8B FF
9E 55 6D D6 54 E2 44 AB DE 59 40 A9 D6 0E C7 12 68 FC 93 19 09 2D 1C 8E 28 99 66 66 02 E9 41 8A 24 3B 64 BA FB 02 8E 6E 0A 18 C6 16 E2 04 0F D3 2B 8E 33 7F 5B
67 C3 25 B2 17 4B B8 45 25 C9 8B 89 E8 F2 5D AE 34 74 BF 20 F8 05 5B 10 35 1E 1A F8 2C 7F EA 93 9C 35 6E 24 C1 24 F5 8F 48 B6 7A CE 84 3A 56 B6 D5 96 14 BC 50 96
EB 3D F5 2C F5 C5 EF 1A BA F2 CD 08 48 F0 D2 E7 E8 22 BB 86 05 FD B1 EB 40 20 7E 23 28 EB 97 54 4D 16 97 EC 21 7B 87 97 58 A0 F9 BA F3 18 73 A2 73 22 ED DA 43
A0 4E DD C4 24 F2 AF D9 C9 3F 75 D0 3C A4 70 F3 B8 DE 53 24 35 EA F5 2E 87 D2 43 07 45 89 E5 B2 E0 20 FF E4 62 0E 12 7C CE BC 6D DC 5E 55 58 BB 46 41 8C E7 0F
8D 9D 4E DC 0B 9E 9F A9 54 67 55 AB 53 A3 27 4F BC 3C 8A 70 3C 2C 89 52 56 89 EA EF CF E8 69 2B FA F6 5F 2A 9D 6D 1B 62 8E 5E 96 43 0A 91 09 06 BB F1 6C 4D 6F
09 05 D2 F0 CD EA 52 4F 83 70 B9 60 46 1B FD A0 E9 FF 0C 73 56 82 4F EE 65 61 12 EC 49 53 DC D6 54 58 D4 20 D2 70 B5 87 8B 9D C2 8D 18 6C 2F CA CC 69 97 4A 0D
```

0:0 | 0/4,024

```
[QuickExec] ALT+Q > type HELP to learn more
```

Capturing | All Processes | 1 / 12,660 | This was a CONNECT tunnel to 'analytics.google.com:443'. Raw Bytes Out: 3,826; In: 9,527

4.3. Continual Learning: CT/CD pipeline

Why ZenML?

ZenML is a powerful MLOps platform for creating and managing continuous training (CT) and continuous deployment (CD) pipelines. It streamlines the process of developing, deploying, and maintaining ML models in production.

Advantages of ZenML:

ZenML simplifies the creation and management of ML pipelines with a user-friendly UI and thorough documentation.

Automated Retraining: It enables automated retraining of models using fresh data, guaranteeing that the model grows and adapts over time.

Integration with ML Tools: ZenML works easily with a variety of ML tools and frameworks, offering flexibility and extension.

Version management: It provides rigorous version management for models and data, which ensures experiment repeatability and traceability.

4.4. (3 pts) Pipeline orchestration

ZenML not only supports CT/CD pipelines, but it also excels at coordinating sophisticated machine learning processes. This includes coordinating many processes including data preparation, model training, assessment, and deployment.

Benefits of ZenML for Orchestration:

Workflow Management: ZenML provides a systematic approach to managing and orchestrating ML workflows, ensuring that all activities are completed in the proper order.

Scalability: It is intended to manage processes at scale, making it appropriate for large-scale installations and high-volume data processing.

Flexibility: The framework allows us to customize and extend procedures to fit individual demands and requirements.

Ease of Use: ZenML's straightforward interface and detailed documentation decrease the complexity of workflow management while increasing productivity.

5. Responsible AI (milestone 8-optional, for later, 15% bonus)

5.1 Evaluation Beyond Accuracy

- (7.5 pts) Audit Model for Bias
- (7.5 pts) Model Explainability and Interpretability

6. Conclusion

To summarize, the creation and implementation of the Movie Roulette machine learning system has been a journey filled with important accomplishments, useful lessons learned, and promising future paths. Movie Roulette has evolved into a complex platform that provides individualized movie suggestions to consumers all over the world thanks to painstaking planning, clever use of cutting-edge technology, and an unwavering dedication to quality.

- **Summary of Achievements**

Technology learned:

Familiarity of ZenML, a robust MLOps framework that allows for end-to-end control of machine learning pipelines.

Proficiency with Flask backend programming, allowing for the building of efficient and scalable model services.

Expertise with Docker and DockerHub for application containerization and deployment.
Familiarity with CI/CD pipelines that use GitHub Actions for automated testing and deployment.

Robust System Architecture:

Design and implementation of a scalable ML pipeline architecture to provide reliability and flexibility.

Real-time resource monitoring and performance optimization are enabled with the integration of monitoring technologies such as Prometheus and Grafana.
Continuous Learning and Adaptation.

Implementation of continuous learning procedures using CT/CD pipelines to ensure the model's flexibility to shifting data distributions.

Establishment of model performance monitoring and data distribution drift monitoring tools, allowing for proactive response to deviations.

- **Lessons Learned**

Insights:

Importance of Automation:

Automation is essential for speeding development processes and increasing productivity, but it requires careful planning and effective error handling methods.

Challenges of A/B Testing:
A/B testing is an effective validation tool, but its adoption necessitates extensive testing and validation of automation scripts.

Data Quality Maintenance:

Data quality is critical in machine learning systems, and continuous monitoring and validation methods are required to ensure data integrity.

Scalability Considerations:

Scalability is an important factor, and systems must be developed for future development to meet rising user demand and data quantities.

User-centric Design:

User experience should be prioritized in system design, with an emphasis on offering intuitive

interfaces and individualized suggestions to increase user happiness.

- **Future Directions**

Enhanced Experimentation:

Investigate sophisticated experimental methods such as multi-armed bandit testing to improve model performance and user engagement.

Cloud Migration :

Consider moving the application to cloud infrastructure for increased scalability, dependability, and accessibility.

Natural language processing integration:

Use natural language processing (NLP) tools to assess user reviews and comments, improving suggestion accuracy and satisfaction.

Collaborative Filtering Improvements:

Use collaborative filtering algorithms to include user preferences and behavior, resulting in more accurate and tailored suggestions.

Community involvement:

Create a thriving community around Movie Roulette by encouraging user input and cooperation to promote continual development and innovation.

References