Workshop 3: Machine learning and Data streaming

Sebastian Belalcazar Mosquera

https://github.com/SEBASBELMOS/workshop-003

Overview

This project implements a machine learning pipeline to predict happiness scores for different countries using data from five CSV files, as part of Workshop 3: Machine Learning and Data Streaming.

Project Structure

Folder/File	Description
assets/	Static resources (images, documentation, etc.)
data/	Data used in the project (ignored in .gitignore)
├── database/	Database Script
├── raw/	World Happiness CSV files
├── processed/	World Happiness Report file
docs/	Documentation, Guides and workshop PDFs
env/	Environment variables (ignored in .gitignore)

Folder/File	Description
├── .env	Stores credentials and paths
kafka/	Python scripts for Apache Kafka
model/	AI Model
notebooks/	Jupyter Notebooks
├── 01_EDA.ipynb	Exploratory Data Analysis of CSV files
├── 02_model-training.ipynb	Model Selection and Training
├── 03_model-performance.ipynb	Model Performance / Metrics
utilities/	Python scripts for Data processing
docker-compose.yml	Docker configuration
pyproject.toml	Poetry dependency management file
README.md	This file

Tools and Libraries

- Python 3.13 → Download here
- PostgreSQL → Download here
- Power BI Desktop → Download here
- Jupyter Notebook → VSCode tool used
- Docker → Documentation here

All the libraries are included in the Poetry project config file (*pyproject.toml*).

Installation and Setup

1. **Clone the Repository:**

```
git clone <https://github.com/SEBASBELMOS/workshop-003.git>
cd workshop-001
```

2. **Installing the dependencies with *Poetry***

   - Windows:

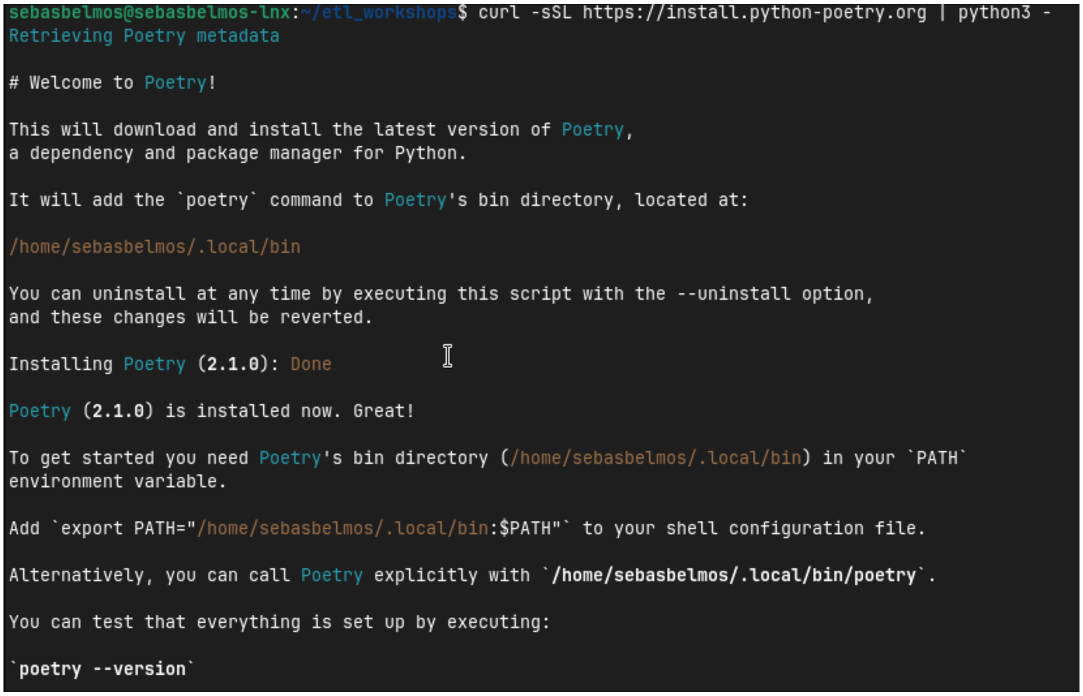
- In Powershell, execute this command:

```
PS C:\Users\sebas> (Invoke-WebRequest -Uri https://install.python-poetry.org -UseBasicParsing).Content | py -
Retrieving Poetry metadata

# Welcome to Poetry!

This will download and install the latest version of Poetry,
a dependency and package manager for Python.

It will add the `poetry` command to Poetry's bin directory, located at:

C:\Users\sebas\AppData\Roaming\Python\Scripts

You can uninstall at any time by executing this script with the --uninstall option,
and these changes will be reverted.

Installing Poetry (2.1.0)
Installing Poetry (2.1.0): Creating environment
Installing Poetry (2.1.0): Installing Poetry
Installing Poetry (2.1.0): Creating script
Installing Poetry (2.1.0): Done

Poetry (2.1.0) is installed now. Great!
```

```
(Invoke-WebRequest -Uri <https://install.python-poetry.org> -UseBasicParsing).Content | py -
```

- Press Win + R, type *sysdm.cpl*, and press **Enter**.
- Go to the *Advanced* tab, select *environment variable*.
- Under System variables, select Path → Click Edit.
- Click *Edit* and set the path provided during the installation in **PATH** so that the `poetry` command works. ("C:\Users\username\AppData\Roaming\Python\Scripts")
- Restart Powershell and execute *poetry --version*.

* Linux
  - In a terminal, execute this command:



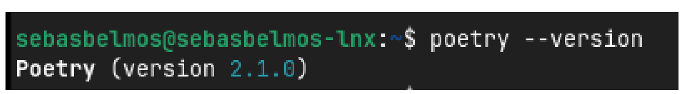
```
sebasbelmos@sebasbelmos-lnx:~/etl_workshops$ curl -sSL https://install.python-poetry.org | python3 -
Retrieving Poetry metadata

# Welcome to Poetry!

This will download and install the latest version of Poetry,
a dependency and package manager for Python.

It will add the `poetry` command to Poetry's bin directory, located at:

/home/sebasbelmos/.local/bin

You can uninstall at any time by executing this script with the --uninstall option,
and these changes will be reverted.

Installing Poetry (2.1.0): Done

Poetry (2.1.0) is installed now. Great!

To get started you need Poetry's bin directory (/home/sebasbelmos/.local/bin) in your `PATH`
environment variable.

Add `export PATH="/home/sebasbelmos/.local/bin:$PATH"` to your shell configuration file.

Alternatively, you can call Poetry explicitly with `/home/sebasbelmos/.local/bin/poetry`.

You can test that everything is set up by executing:

`poetry --version`
```

```
curl -sSL <https://install.python-poetry.org> | python3 -
```

- Now, execute:

```
export PATH = "/home/user/.locar/bin:$PATH"
```

- Finally, restart the terminal and execute *poetry --version*.



```
sebasbelmos@sebasbelmos-lnx:~$ poetry --version
Poetry (version 2.1.0)
```

3. **Poetry Shell**

- Enter the Poetry shell with *poetry shell*.
- Then, execute *poetry init*, it will create a file called *pyproject.toml*
- To add all the dependencies, execute this:

```
poetry add pandas matplotlib psycopg2-binary sqlalchemy python-do
tenv seaborn ipykernel dotenv kafka-python
```

- Install the dependencies with:
  In case of error with the .lock file, just execute
  poetry lock to fix it.

  ```
  poetry install
  ```

- Create the kernel with this command (You must choose this kernel when running the notebooks):

  ```
  poetry run python -m ipykernel install --user --name workshop-003 --display-name "Python (workshop-003)"
  ```

4. **Enviromental variables**

   > Realise this in VS Code.

   1. Inside the cloned repository, create a new directory named *env/*.
   2. Within that directory, create a file called *.env*.
   3. In the *.env file*, define the following six environment variables (without double quotes around values):

      ```
      PG_HOST = #host address, e.g. localhost or 127.0.0.1
      PG_PORT = #PostgreSQL port, e.g. 5432

      PG_USER = #your PostgreSQL user
      PG_PASSWORD = #your user password

      PG_DATABASE = #your database name, e.g. postgres
      ```

   4. Create the database with this command:

      ```
      psql -U your_username -c "CREATE DATABASE happiness_db;"
      ```

5. **Execution**

a. Run all the notebooks to create the EDA, transformations and model.

b. Run this command to start the Docker Containers for Kafka and Zookeeper.

```
docker-compose up -d
```

c. To check if the containers are correctly running, use this command:

```
docker ps
```

```
PS D:\U\FIFTH SEMESTER\ETL\workshop-003> docker ps
CONTAINER ID   IMAGE                             COMMAND                  CREATED          STATUS          PORTS                                        NAMES
7f1a6bb4db88   confluentinc/cp-kafka:latest      "/etc/confluent/dock…"   19 minutes ago   Up 19 minutes   0.0.0.0:9092->9092/tcp, 9093/tcp             kafka_w3
38e0542b340d   confluentinc/cp-zookeeper:latest  "/etc/confluent/dock…"   19 minutes ago   Up 19 minutes   2888/tcp, 0.0.0.0:2181->2181/tcp, 3888/tcp   zookeeper_w3
PS D:\U\FIFTH SEMESTER\ETL\workshop-003>
```

d. Now we can create a `Kafka Topic` with this command:

```
docker exec -it kafka_w3 kafka-topics --create --topic wh_kafka_topic --bootstrap-server localhost:9092
```

```
PS D:\U\FIFTH SEMESTER\ETL\workshop-003> docker exec -it kafka_w3 kafka-topics --create --topic wh_kafka_topic --bootstrap-server localhost:9092
WARNING: Due to limitations in metric names, topics with a period ('.') or underscore ('_') could collide. To avoid issues it is best to use either, but not both.
Created topic wh_kafka_topic.
PS D:\U\FIFTH SEMESTER\ETL\workshop-003>
```

e. To check if it was created, run this command:

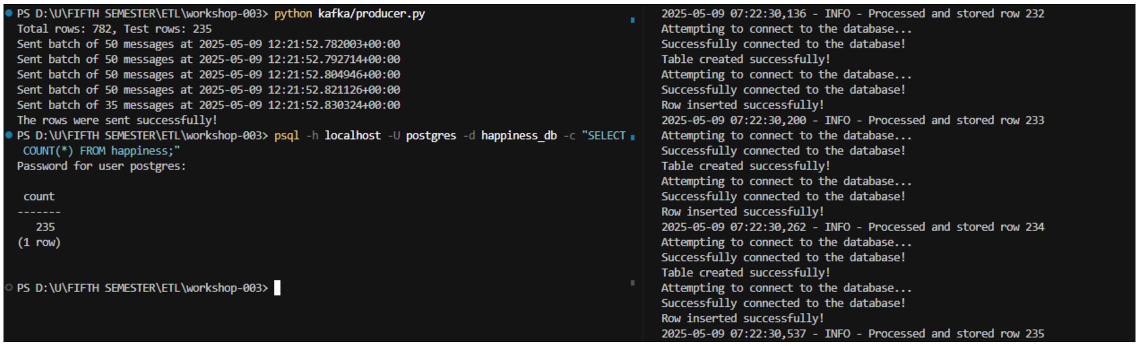
```
docker exec -it kafka_w3 kafka-topics --list --bootstrap-server localhost:9092
```

```
PS D:\U\FIFTH SEMESTER\ETL\workshop-003> docker exec -it kafka_w3 kafka-topics --list --bootstrap-server localhost:9092
wh_kafka_topic
PS D:\U\FIFTH SEMESTER\ETL\workshop-003>
```

f. Finally, run the files from the kafka directory (`producer.py` and `consumer.py`, in the same order) with the following commands:

```
python kafka/producer.py
```

```
python kafka/consumer.py
```



g. Optional Cleanup (After executing everything)

```
docker-compose down
psql -h localhost -U postgres -d happiness_db -c "DELETE FROM happiness;"
psql -h localhost -U postgres -d happiness_db -c "ALTER SEQUENCE happiness_id_seq RESTART WITH 1;"
```

Conclusions

This project successfully implemented a machine learning pipeline to predict happiness scores, fulfilling the objectives of Workshop 3: Machine Learning and Data Streaming. The pipeline integrated exploratory data analysis (EDA), model training, data streaming with Apache Kafka, and performance evaluation, with predictions stored in a PostgreSQL database.

Model Performance

- Four regression models were evaluated: Linear Regression, Random Forest Regressor, an Alternative Random Forest Regressor, and Gradient Boosting Regressor. The Alternative Random Forest Regressor, configured with 100 estimators and a random state of 0, achieved the best performance with a Mean Squared Error (MSE) of 0.1721, a Mean Absolute Error (MAE) of 0.3246, and a Coefficient of Determination (R^2) of 0.8639. This indicates that the model explains 86.39% of the variance in happiness scores, outperforming the other models and demonstrating the effectiveness of ensemble techniques with increased estimators. The Explained Variance Score of 0.8650 further confirms the model's ability to capture the variance in the target variable, with a Root Mean Squared Error (RMSE) of approximately 0.4149, suggesting an average prediction error of 0.4149 on a 0–10 scale.

- The Linear Regression model produced an MSE of 0.2103 and an R^2 of 0.8337, explaining 83.37% of the variance, which is a reasonable but improvable fit. The Random Forest Regressor (50 estimators) improved upon this with an MSE of 0.1755 and an R^2 of 0.8612, while the Gradient Boosting Regressor achieved an MSE of 0.1749 and an R^2 of 0.8617, both showing strong performance but falling short of the Alternative Random Forest Regressor.
- The expected model features include: ['year', 'gdp_per_capita', 'health_life_expectancy', 'social_support', 'freedom', 'government_corruption', 'generosity', 'continent_Africa', 'continent_America', 'continent_Asia', 'continent_Central_America', 'continent_Europe', 'continent_North_America', 'continent_Oceania', 'continent_South_America'], and these were successfully aligned with the database columns.

Data Streaming and Storage

- The pipeline streamed the 30% test set (235 rows) from a total dataset of 782 rows, aligning with the 70/30 train-test split. The Kafka producer and consumer successfully processed and stored these predictions in the `happiness` database table, with each row including input features, actual happiness scores, and predicted happiness scores. The dummy variable values in the database were: `continent_Central_America`: 14, `continent_North_America`: 3, and `continent_South_America`: 18, with a continent distribution of `Other`: 200, `South America`: 18, `Central America`: 14, and `North America`: 3.

Visual and Analytical Insights

- **Actual vs Predicted Happiness Scores**: A scatter plot of actual versus predicted happiness scores closely follows the ideal line (y=x), indicating high predictive accuracy. Most predictions deviate by less than 0.5 points from the actual scores, consistent with the RMSE of 0.4149, demonstrating the model's reliability.

- **Average Predicted Happiness Score by Continent**: Analysis by continent revealed distinct regional patterns. North America exhibited the highest average predicted happiness score at 6.448847, followed by South America at 6.038931, Central America at 5.544421, and the "Other" category (encompassing Africa, Asia, Europe, Oceania, and America) at 5.337529. These values align with socio-economic conditions and highlight regional happiness variations.
- **Original vs Predicted Happiness Scores by Continent**: A bar chart comparing original and predicted average happiness scores showed strong alignment. For instance, Central America's original average of 5.804705 was predicted as 5.544421, North America's 7.024867 as 6.448847, and South America's 6.126091 as 6.038931. The "Other" category averaged 5.6 originally and 5.337529 predicted, with minor differences (less than 0.5 on average) confirming the model's robustness across diverse regions.
- **Feature Importance**: The model identified `gdp_per_capita` and `health_life_expectancy` as the most influential predictors, followed by `freedom`, `social_support`, `generosity`, and `government_corruption`.

Author

Created by **Sebastian Belalcazar Mosquera**.

Connect with me on LinkedIn for feedback, suggestions, or collaboration opportunities!
