Workshop #3 - Documentation

Martín García Chagüezá

About the workshop

In this workshop, the World Happiness Report dataset will be used, comprising four CSV files with data from 2015 to 2019. A streaming data pipeline will be implemented using Apache Kafka. Once processed, the data will be fed into a Random Forest regression model to estimate the Happiness Score based on other scores in the dataset. The results will then be uploaded to a database, where the information will be analyzed to assess the accuracy and insights of the predictions.

The tools used are:

- Python 3.10 ➜ Download site
- Jupyter Notebook ➜ VS Code tool for using notebooks
- Docker ➜ Download site for Docker Desktop
- PostgreSQL ➜ Download site
- Power BI (Desktop version) ➜ Download site

The dependencies needed for Python are:

- python-dotenv
- kafka-python-ng
- country-converter
- pandas
- matplotlib
- seaborn
- plotly
- nbformat
- scikit-learn
- sqlalchemy
- psycopg2-binary

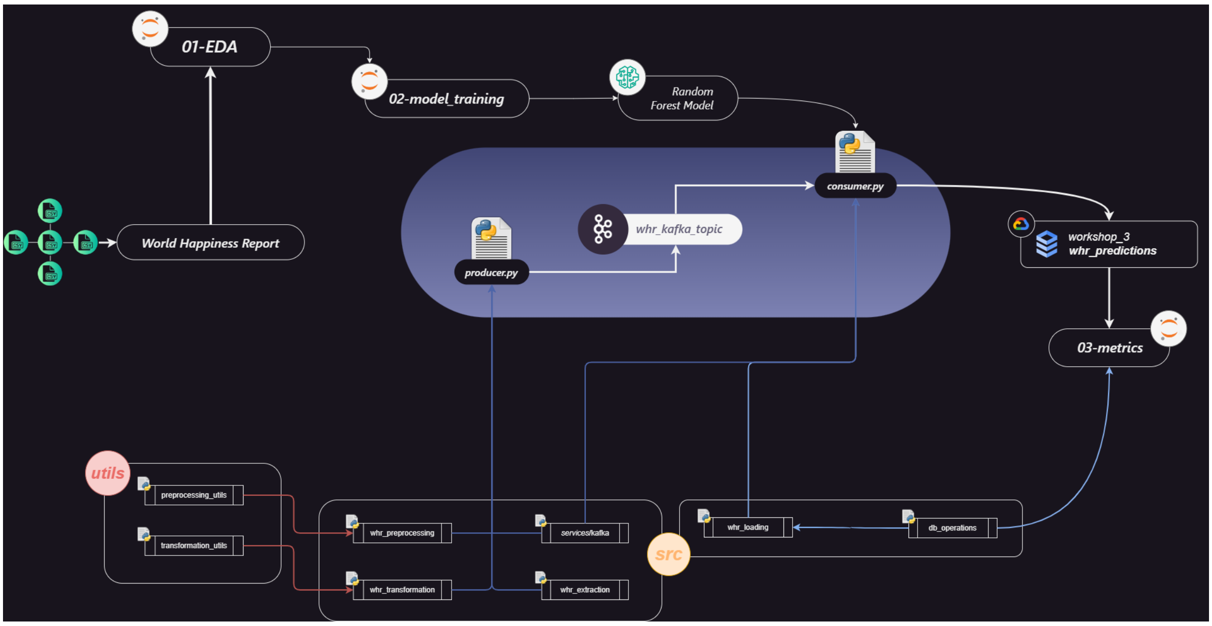
These and more dependencies are included in the Poetry project config file (`pyproject.toml`).

The images used in Docker are:

- confluentinc/cp-zookeeper
- confluentinc/cp-kafka

The configuration and installation of these images are facilitated by the Docker Compose config file (`docker-compose.yml`).

Data flow



Process

Creating virtualenv and installing the dependencies with *Poetry*

> *For a tutorial on how to install and configure Poetry, follow the next page → Instalar Poetry*

Poetry works as a dependency manager primarily, but it also let us create a virtual enviroment to add or install the dependencies we need for our project. Through `poetry add <dependency>` we can add the libraries to our project: those libraries are going to be registered in the Poetry config file, named ***pyproject.toml***.

If the ***pyproject.toml*** is already in our directory, but we're working in a different *virtualenv*, we can use the `poetry install` command, as showed in the image, to create the new virtual enviroment and install the registered dependencies in the Poetry configuration file.

```
PS C:\Users\marti\OneDrive\Escritorio - PC\Ingenieria de Datos e IA - UAO\Semestre 4\ETL\Semana #1 - #6\Workshop #1> poetry install
Creating virtualenv workshop-1-Ih9GMGpq-py3.12 in C:\Users\marti\AppData\Local\pypoetry\Cache\virtualenvs
Installing dependencies from lock file

Package operations: 45 installs, 0 updates, 0 removals

  - Installing six (1.16.0)
  - Installing asttokens (2.4.1)
  - Installing executing (2.0.1)
  - Installing numpy (2.1.0)
  - Installing parso (0.8.4)
  - Installing platformdirs (4.2.2)
  - Installing pure-eval (0.2.3)
  - Installing pywin32 (306)
  - Installing traitlets (5.14.3)
  - Installing wcwidth (0.2.13)
  - Installing colorama (0.4.6): Installing...
  - Installing contourpy (1.2.1)
  - Installing cycler (0.12.1)
  - Installing decorator (5.1.1)
  - Installing contourpy (1.2.1)
  - Installing cycler (0.12.1)
  - Installing decorator (5.1.1)
```

Some dependencies deserve special attention as they play crucial roles in this workshop:

- ***kafka-python-ng*** → A modern, actively maintained fork of kafka-python that provides Python bindings for Apache Kafka. I use the "ng" (*next generation*) version since it offers better performance, async support, and fixes several bugs present in the original *kafka-python* library. *kafka-python* intended to be the library that I was going to use to execute the Python scripts, but due to failures it was replaced with this one.
- ***country-converter*** → A Python package that standardizes and converts country names, codes, and other attributes. Essential in this workshop for mapping countries to their respective continents, ensuring consistent geographical analysis of the World Happiness Report data.

- ***nbformat*** → A base implementation for working with Jupyter notebook files. Required as a dependency by Plotly to properly render interactive visualizations within Jupyter notebooks, especially for our geographical and statistical plots.
- ***psycopg2-binary*** → The binary distribution of the PostgreSQL adapter for Python. I used the binary version (*-binary*) instead of the source package because Poetry often has difficulties building psycopg2 from source due to its C extension dependencies. The binary version comes pre-compiled and avoids these installation issues.

Configuring Docker Compose to run the Kafka environment

The `docker-compose.yml` file sets up a local Apache Kafka environment with two main services:

Zookeeper Service

```
zookeeper:
  image: confluentinc/cp-zookeeper:latest
  container_name: zookeeper_docker
```

- Uses the latest Confluent Platform Zookeeper image
- Exposes port 2181 for client connections
- Configures basic Zookeeper parameters like client port and tick time
- Required for Kafka cluster coordination and management

Kafka Broker Service

```
kafka:
  image: confluentinc/c
```

- Uses the latest Confluent Platform Kafka image

```
p-kafka:latest
  container_name: kafka
_docker
```

- Depends on Zookeeper service
- Exposes port 9092 for client connections
- Includes essential configurations:
  - Unique broker ID
  - Zookeeper connection string
  - Listener configurations for internal and external communications
  - Security protocol mappings
  - Replication factors for system topics

Implications for Project Execution

1. **Local Development**: This configuration creates a single-node Kafka cluster suitable for local development and testing.
2. **Connectivity**: Applications can connect to Kafka at `localhost:9092` for producing and consuming messages.
3. **State Management**: Data persistence is handled by Zookeeper running on port 2181.
4. **Security**: Uses plain text protocol without authentication (*suitable for development only*).

docker-compose.yml

docker-compose.yml

You, hace 4 días | 1 author (You)

```yaml
services:
  zookeeper:
    image: confluentinc/cp-zookeeper:latest
    container_name: zookeeper_docker
    environment:
      ZOOKEEPER_CLIENT_PORT: 2181
      ZOOKEEPER_TICK_TIME: 2000
    ports:
      - 2181:2181

  kafka:
    image: confluentinc/cp-kafka:latest
    container_name: kafka_docker      You, hace 4 días via  PR #2 • Updating the modules and using the Random Fore
    depends_on:
      - zookeeper
    ports:
      - 9092:9092
    environment:
      KAFKA_BROKER_ID: 1
      KAFKA_ZOOKEEPER_CONNECT: zookeeper:2181
      KAFKA_LISTENER_SECURITY_PROTOCOL_MAP: PLAINTEXT:PLAINTEXT,PLAINTEXT_INTERNAL:PLAINTEXT
      KAFKA_ADVERTISED_LISTENERS: PLAINTEXT://localhost:9092,PLAINTEXT_INTERNAL://broker:29092
      KAFKA_INTER_BROKER_LISTENER_NAME: PLAINTEXT
      KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR: 1
      KAFKA_TRANSACTION_STATE_LOG_MIN_ISR: 1
      KAFKA_TRANSACTION_STATE_LOG_REPLICATION_FACTOR: 1
```

Exploratory Data Analysis

During my analysis of happiness metrics from 2015 to 2019, I found several interesting patterns that help understand what contributes to national happiness levels.

Initial Data Challenges

1. I had to standardize different yearly datasets since they weren't consistently formatted

```python
column_mapping = {
    # Country
    'Country': 'country',
    'Country or region': 'country',

    # Happiness Score
    'Happiness Score': 'happiness_score',
    'Happiness.Score': 'happiness_score',
    'Score': 'happiness_score',

    # Happiness Rank
    'Happiness Rank': 'happiness_rank',
    'Happiness.Rank': 'happiness_rank',
    'Overall rank': 'happiness_rank',

    # Economy (GDP per Capita)
    'Economy (GDP per Capita)': 'economy',
    'Economy..GDP.per.Capita.': 'economy',
    'GDP per capita': 'economy',

    # Health (Life Expectancy)
    'Health (Life Expectancy)': 'health',
    'Health..Life.Expectancy.': 'health',
    'Healthy life expectancy': 'health',

    # Social Support
    'Family': 'social_support',
    'Social support': 'social_support',

    # Freedom
    'Freedom': 'freedom',
    'Freedom to make life choices': 'freedom',

    # Perceptions of Corruption
    'Trust (Government Corruption)': 'corruption_perception',
    'Trust..Government.Corruption.': 'corruption_perception',
    'Perceptions of corruption': 'corruption_perception',

    # Generosity
    'Generosity': 'generosity',

    # Dystopia Residual
    'Dystopia Residual': 'dystopia_residual',
    'Dystopia.Residual': 'dystopia_residual',
}
```

2. Found and fixed one missing corruption value for UAE in 2018

```python
df["corruption_perception"] = (
                                df["corruption_perception"]
                                .fillna(df["corruption_perception"].mean())
                              )
```

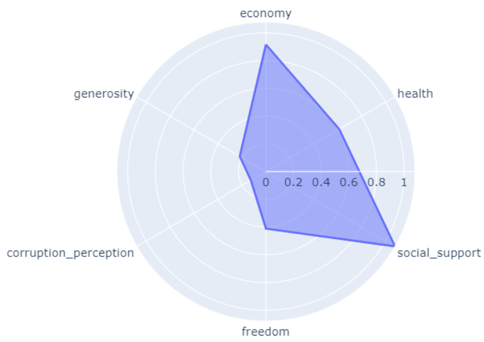
```python
df.query("country == 'United Arab Emirates' & year == '2018'")
```

	corruption_perception	year	health	generosity	happiness_rank	freedom	happiness_score	social_support	country	economy
489	0.125436	2018	0.67	0.186	20	0.284	6.774	0.776	United Arab Emirates	2.096

Main Happiness Factors

Looking at the average scores:

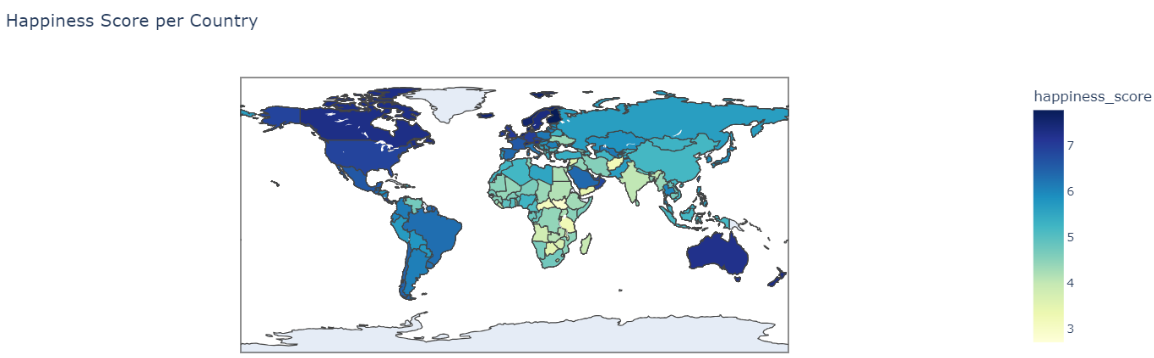
- Economic conditions show strong positive influence
- People's trust in their governments (corruption perception) is concerningly low
- Social bonds and community support remain highly valued
- Health and personal freedom sit in the middle range



Global Happiness Map

The world picture reveals:

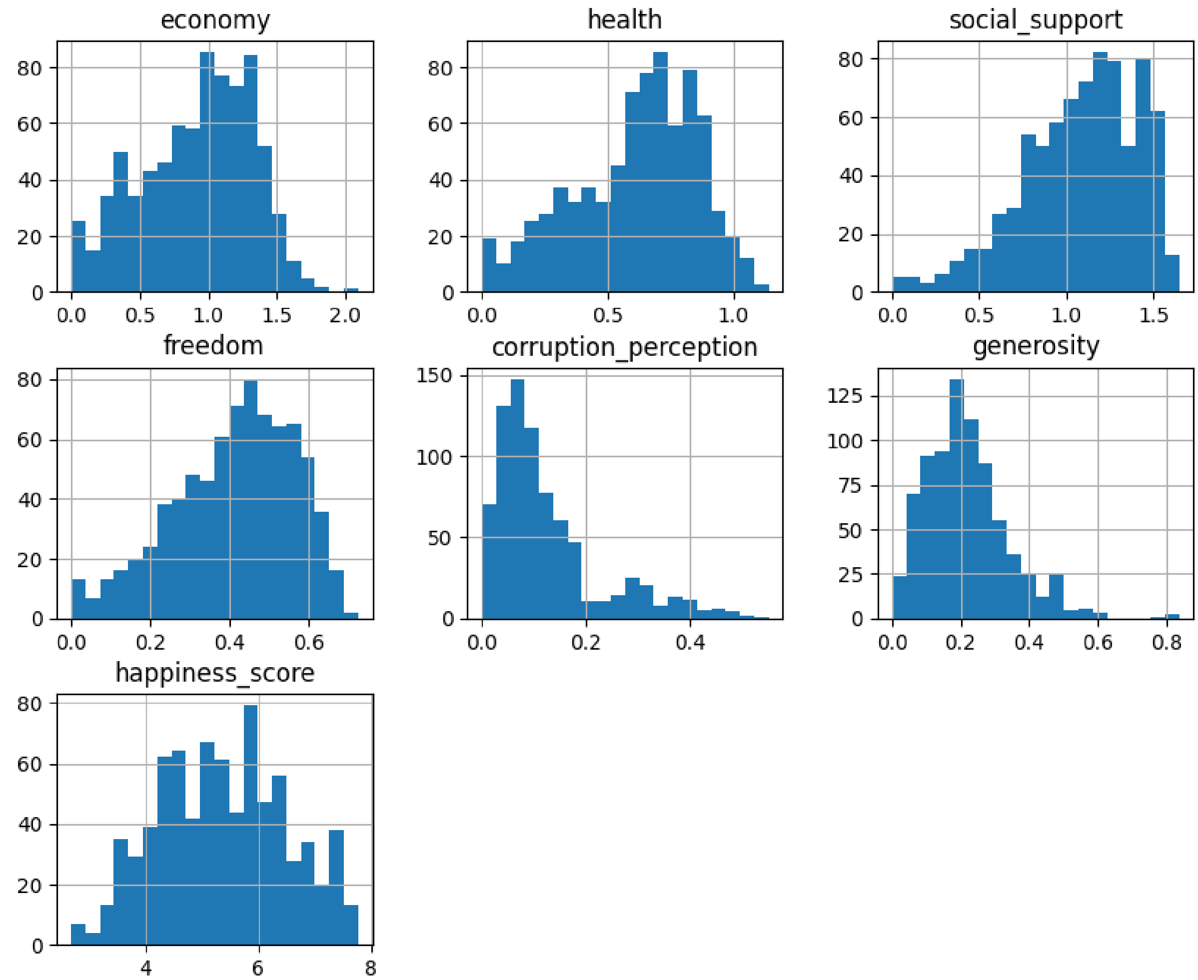
- Western Europe, especially Nordic countries, consistently ranks happiest
- Most African nations show lower happiness levels
- Clear happiness divides exist between neighboring regions
- Location seems to play a significant role in happiness levels



How Happiness Elements Are Distributed

My analysis of the distributions shows:

- Most countries cluster around a middle happiness score
- The perception of the economy varies significantly between countries, being one of the factors with the greatest impact on the overall happiness score
- Social support is generally strong across most countries
- Freedom and corruption scores vary widely between nations
- Health measures show two distinct groups of countries

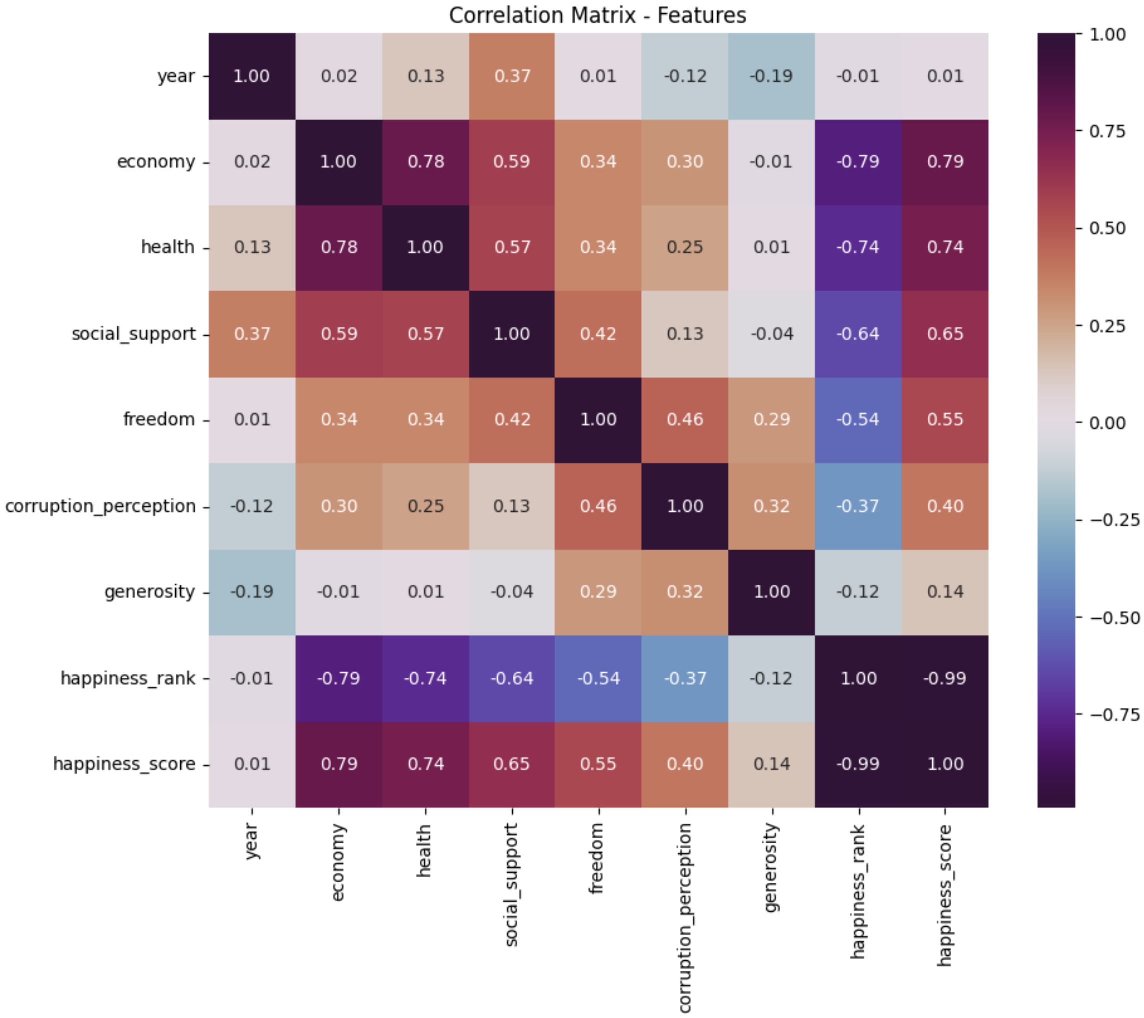


How Factors Interact

The correlation analysis reveals interesting relationships between people's perceptions:

- Countries where citizens perceive better economic and health conditions tend to also report stronger social support systems

- Places where people feel more personal freedom show higher overall happiness scores
- The perception of generosity in a society appears independent from other happiness factors
- In countries where people perceive higher levels of corruption, they also tend to report lower happiness levels



Model Training Process

Feature Selection

Based on the EDA findings, I made strategic decisions for feature selection:

1. **Key Predictors Included**:

- Economy (0.79 correlation with happiness_score)
- Social Support (0.65 correlation)
- Health (0.74 correlation)
- Freedom (0.54 correlation)
- Corruption Perception (0.40 correlation)

2. **Excluded Features**:
    - `happiness_rank` (removed due to -0.99 correlation with target variable)
    - `country` (removed to avoid high cardinality)
    - Generosity (kept despite low 0.14 correlation to test its influence)
3. **Encoded Features**:
    - Continent converted to dummy variables to capture geographical patterns shown in choropleth analysis

Model Training Results

Linear Regression

- MSE: 0.2109
- R^2: 0.8333

Established a baseline but showed limitations in capturing non-linear relationships identified in the EDA.

```
lr_model = LinearRegression()
lr_model.fit(X_train, y_train)

y_pred_lf = lr_model.predict(X_test)

mse_lr = mean_squared_error(y_test, y_pred_lf)
r2_lr = r2_score(y_test, y_pred_lf)

print("Mean Squared Error for Linear Regression: ", mse_lr)
print("R2 Score for Linear Regression: ", r2_lr)
```

[11]

```
Mean Squared Error for Linear Regression:  0.21087396980793913
R2 Score for Linear Regression:  0.8332893378421595
```

Random Forest Regressor

- MSE: 0.1700
- R^2: 0.8656

Best performer, likely due to its ability to capture the complex interactions between features observed in correlation analysis.

```
rf_model = RandomForestRegressor(n_estimators=50, random_state=200)
rf_model.fit(X_train, y_train)

y_pred_rf = rf_model.predict(X_test)

mse_rf = mean_squared_error(y_test, y_pred_rf)
r2_rf = r2_score(y_test, y_pred_rf)

print("Mean Squared Error for Random Forest: ", mse_rf)
print("R2 Score for Random Forest: ", r2_rf)
```

[12]

```
Mean Squared Error for Random Forest:  0.17005002768093916
R2 Score for Random Forest:  0.865563527160472
```

Gradient Boosting Regressor

- MSE: 0.1714
- R^2: 0.8645

Similar performance to Random Forest, validating the robustness of ensemble methods for this dataset.

```
gb_model = GradientBoostingRegressor()
gb_model.fit(X_train, y_train)

y_pred_gb = gb_model.predict(X_test)

mse_gb = mean_squared_error(y_test, y_pred_gb)
r2_gb = r2_score(y_test, y_pred_gb)

print("Mean Squared Error for Gradient Boosting: ", mse_gb)
print("R2 Score for Gradient Boosting: ", r2_gb)
```

[13]

```
Mean Squared Error for Gradient Boosting:  0.17144932369572535
R2 Score for Gradient Boosting:  0.8644572855252797
```

Selection of Final Model

I chose the **Random Forest Regressor** because:

1. Best performance metrics
2. Better handling of the non-linear relationships discovered in EDA
3. Good ability to manage the diverse feature set including both continuous variables and dummy-encoded continents
4. The model was saved using joblib for later deployment in the streaming pipeline.

Comparison between Original and Predicted Happiness Scores

Kafka Data Streaming Architecture

1. Data Producer (`kafka/producer.py`)

The producer script handles:

- **Data Loading & Preprocessing**

  ```
  # Load and transform data
  happiness_dataframes = extracting_data()
  df = transforming_data(happiness_dataframes)
  df = preprocessing_data(df)
  ```

  - Loads World Happiness Report data
  - Transforms and preprocesses it for analysis
  - Prepares data for streaming
- **Message Publishing**

```
# Send data to Kafka topic
get_kafka_producer(df, "whr_kafka_topic")
```

- Streams processed data to Kafka topic
- Each row becomes a message

2. Data Consumer (`kafka/consumer.py`)

The consumer script performs:

- **Message Reception**

```
# Consume messages from topic
consumer = get_kafka_consumer("whr_kafka_topic")
consumer_messages = [json.loads(message) for messa
ge in consumer]
```

  - Subscribes to Kafka topic
  - Deserializes incoming JSON messages

- **ML Prediction & Storage**

```
# Load model and make predictions
rf_model = joblib.load("./model/rf_model.pkl")
predictions = rf_model.predict(df_test)

# Store results in database
df["predicted_happiness_score"] = predictions
loading_data(df, "whr_predictions")
```

  - Loads Random Forest model
  - Makes happiness score predictions
  - Stores results in database

3. Kafka Service Layer (`src/services/kafka.py`)

Contains core Kafka functionality:

- **Producer Service**

```
def get_kafka_producer(df: pd.DataFrame, topic: st
r):
    producer = KafkaProducer(
        bootstrap_servers="localhost:9092",
        value_serializer=lambda v: json.dumps(v).e
ncode('utf-8')
    )

    for index, row in df.iterrows():
        dict_row = dict(row)
        producer.send(topic, value=dict_row)
        time.sleep(1)
```

  - Serializes DataFrame rows to JSON
  - Implements flow control (1s delay)
  - Handles error logging

- **Consumer Service**

```
def get_kafka_consumer(topic: str) -> list:
    consumer = KafkaConsumer(
        topic,
        bootstrap_servers="localhost:9092",
        value_deserializer=lambda v: json.loads(v.
decode('utf-8')),
        consumer_timeout_ms=3000
    )
    return [message for message in consumer]
```

  - Configures consumer settings

- Deserializes messages
- Collects messages into list format
- Implements error handling

Conclusions

- **Data Pipeline Setup**: The workshop uses Apache Kafka to create a real-time data pipeline, streaming the World Happiness Report data from 2015–2019. A producer-consumer setup handles data ingestion and model prediction, forming an efficient workflow for handling large datasets.
- **Dependency and Environment Management**: Poetry is used to streamline package management and environment setup, with essential dependencies like `kafka-python-ng` for Kafka streaming, `country-converter` for geographic data, and `psycopg2-binary` for PostgreSQL integration.
- **Exploratory Data Insights**: The EDA phase reveals that economic and social support factors significantly influence happiness, with notable happiness divides by region. Western Europe ranks highest, while many African countries show lower scores.
- **Model Selection**: A Random Forest Regressor was selected for its superior performance (R^2 = 0.8656) and ability to model non-linear relationships within the data. Key predictors include economic strength, social support, health, and freedom scores.
- **Kafka Streaming Design**: Kafka producer and consumer scripts manage data flow and model predictions effectively. JSON serialization and deserialization ensure smooth data transfer, while robust error handling improves reliability.
- **Integrated Tooling**: The setup includes Docker for containerization and PostgreSQL for storage, creating a replicable, testable environment that supports both model training and deployment.