Workshop 3

Presented by:

Juan Camilo Buitrago Gonzalez

ETL

Teacher:

Javier Alejandro Vergara Zorrilla

Universidad Autónoma de Occidente

08/03/2024

Introduction:

In this workshop, I embarked on a journey to develop a predictive model capable of estimating happiness scores based on various socio-economic and health indicators. The challenge was amplified by the diverse structures of datasets spanning several years, each presenting unique preprocessing demands.

The process began with data preprocessing, where I tackled the complexities of merging multiple datasets with varying formats. My goal was to ensure the integrity and consistency of the data, a foundational step crucial for accurate modeling. This involved cleaning the data, handling missing values, and standardizing features across datasets to create a unified dataset.

Following data preparation, I engaged in an extensive exploratory data analysis (EDA) phase. This phase was instrumental in understanding the key variables that influence happiness scores. Through techniques like correlation analysis and feature selection, I identified significant predictors and gained insights into the data's underlying patterns.

The core of my work centered on building the predictive model. I explored various modeling techniques, focusing on regression analysis and Random Forests due to their effectiveness in handling complex, non-linear relationships among variables. The modeling phase was iterative, involving tuning parameters and validating the model to optimize performance and ensure robustness.

An exciting aspect of this workshop was the real-time application of the model through data streaming. I employed Kafka to stream the data into a PostgreSQL database, setting up a dynamic pipeline that simulated a real-world application of predictive analytics. This not only tested the model's performance in real-time but also enhanced my understanding of deploying machine learning models in production environments.

Technologies used:

The technologies used for this workshop were:

Python: The language used for the workshop

Jupyter notebook: The notebook platform used to make the EDA and other transformations.

Visual Studio Code: The chosen code editor for the workshop management and development

PostgreSQL: The database management system used for storing the prediction of the happiness score and the columns used for it.

Docker: Used to use zookeeper - kafka for the producer and consumer.

Architecture:

My workshop architecture is organized in a simple structure:

- .gitignore: Specifies intentionally untracked files to ignore (e.g., sensitive credentials, local environment files).
- config.json: Contains configuration settings, such as database connection details.
- consumer.py: Consumes data from Kafka, applies the machine learning model, and inserts predictions into the database.
- db.py: Manages database connections and operations, such as creating tables and inserting data.
- docker-compose.yml: Defines and runs multi-container Docker applications; used here to set up services like Kafka, Zookeeper, and PostgreSQL.
- functions.py: Contains reusable code functions used across the project, such as data transformations or utility functions.
- Metrics.ipynb: Jupyter notebook used for calculating and reviewing performance metrics of the machine learning model.
- producer.py: Produces and sends data to Kafka, typically after fetching or generating predictions.
- README.md: Provides an overview of the project, setup instructions, and other essential information.
- data Folder
  - 2015.csv to 2019.csv: Annual datasets containing happiness scores and various predictors from different countries.
  - X_test.csv & y_test.csv: Test datasets used for validating the machine learning model.
- docs Folder
  - documentation.pdf: Detailed documentation of the project including methodology, code references, and explanations of the processes.
- models Folder
  - randomForest.pkl: Serialized version of the trained RandomForest model, ready for use in predictions.

- notebooks Folder
  - EDA_001.ipynb: Jupyter notebook used for exploratory data analysis, including visualizations and statistical tests to understand the data better. Also used to select the features and the model.

Implementation:

If you already have all the requirements, you only need to run Docker Desktop and then:

1. Run all the notebook called EDA_001

2. Go to the root of the repository

3. Run `docker-compose up`

4. Run producer.py

5. Run consumer.py

6. Go to the database and refresh to see how the data is streaming

7. (Optional) Run the Metrics notebook.

Summary of the Exploratory Data Analysis (EDA) Process:

Importing Libraries:

Imported necessary libraries including pandas, numpy, seaborn, matplotlib, joblib, and several from sklearn for model training and evaluation.

Loading Data:

Loaded happiness datasets from the years 2015 to 2019 using pd.read_csv().

Initial Data Inspection:

Displayed the first few rows of each dataset to understand their structure and identify discrepancies in column names.

Checked the data types and non-null counts of each column using .info() to assess data quality.

Data Cleaning and Normalization:

Renamed columns to standardize the naming convention across different datasets.

Added a 'Year' column to each dataset for better tracking and merging.

Identified and addressed missing values by either filling or dropping them.

Merged datasets by concatenating them into a single DataFrame.

Handling Inconsistent Data:

Addressed differences in country names and ensured consistent representation.

Mapped countries to their respective continents and added a 'Continent' column for better feature extraction.

Feature Engineering:

Created dummy variables for categorical columns like 'Continent' using pd.get_dummies().

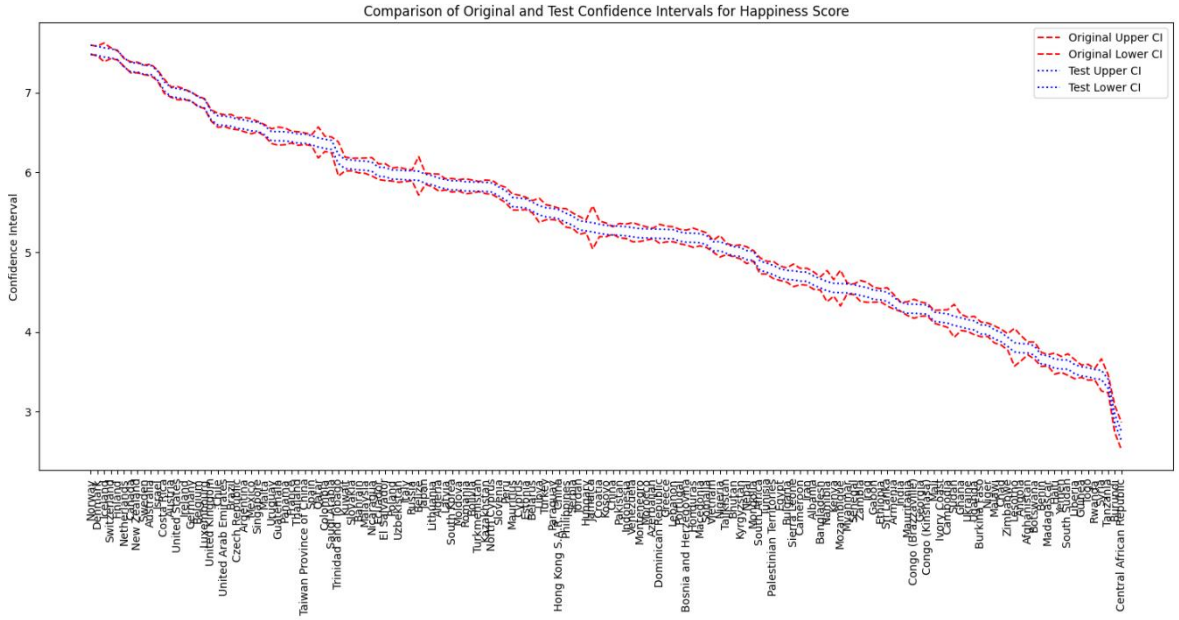
continent_africa	continent_asia	continent_europe	continent_north_america	continent_oceania	continent_south_america
0	0	0	1	0	0
1	0	0	0	0	0
0	0	0	0	0	1
0	0	1	0	0	0
0	0	1	0	0	0

Standardized column names to avoid errors in further processing.

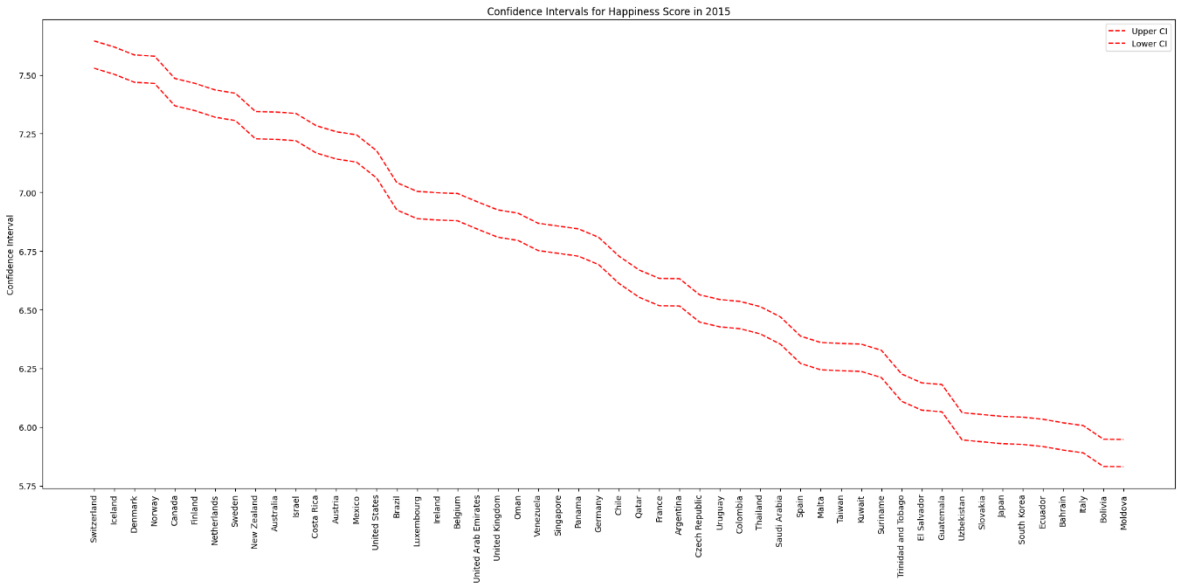
Calculating Confidence Intervals:

Calculated upper and lower confidence intervals for happiness scores to understand the variability in the data.

This was the first test in the 2017 dataset.



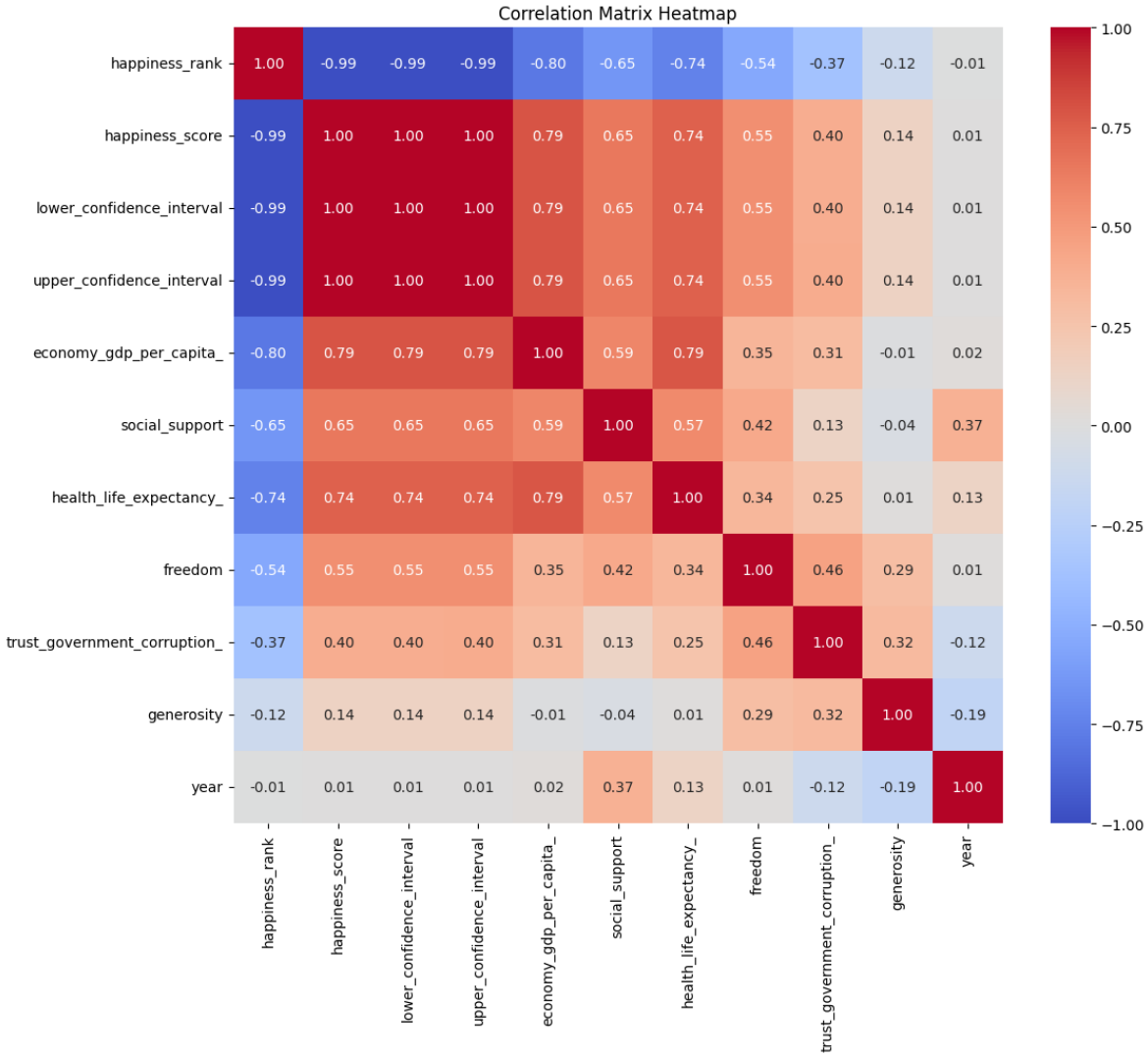
This was a sample of the application to the 2015 dataset.



This was maked to try to get more information in the datasets and get better R^2, but unfortunately this overfit the model, so it doesn't work for the model.

However, this may be useful in case someone else tries to do it later and sees this and knows that this is a step they can skip.

Correlation Analysis:



Dropped columns that were highly correlated or not contributing significantly to the model, such as confidence intervals and happiness rank.

Plotted a correlation matrix heatmap to visualize the relationships between features.

Model Training and Evaluation:

Split the data into training and test sets using train_test_split.

```
y = df['happiness_score']
X = df.drop(columns= ['happiness_score'], axis=1)
```

```
X_train, X_test, y_train, y_test = train_test_split(df.drop('happiness_score', axis=1), df['happiness_score'], test_si
```

Trained multiple models (Random Forest, Linear Regression, Gradient Boosting) to predict happiness scores.

Evaluated model performance using metrics like Mean Squared Error (MSE) and R-squared (R^2).

Selected Random Forest as the best-performing model with an R^2 of 0.84 and an MSE of 0.2.

```
randomForest = RandomForestRegressor(n_estimators=100, random_state=42)
randomForest.fit(X_train, y_train)
y_pred_rf = randomForest.predict(X_test)
mse_rf = mean_squared_error(y_test, y_pred_rf)
r2_rf = r2_score(y_test, y_pred_rf)

print("Random Forest Regression Model Results:")
print("Mean Squared Error (MSE):", mse_rf)
print("Coefficient of determination (R2):", r2_rf)
```

```
Random Forest Regression Model Results:
Mean Squared Error (MSE): 0.1991724397213645
Coefficient of determination (R2): 0.8404950405424444
```

Saved the trained model using joblib.

```
joblib.dump(randomForest, '../models/randomForest.pkl')
```

```
['../models/randomForest.pkl']
```

Final Features for the Model:

Determined the final set of features used for predicting happiness scores:

economy_gdp_per_capita_

social_support

health_life_expectancy_

freedom

trust_government_corruption_

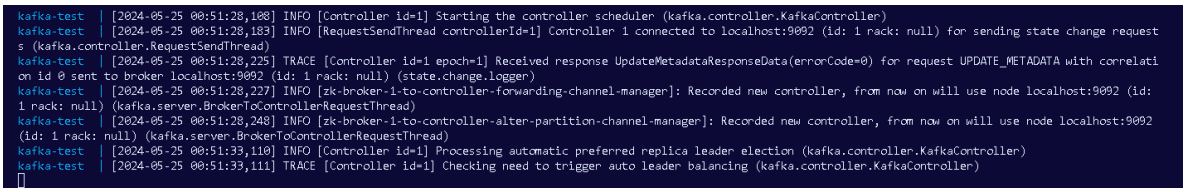
generosity

year

Dummy variables for continents (continent_africa, continent_asia, etc.)

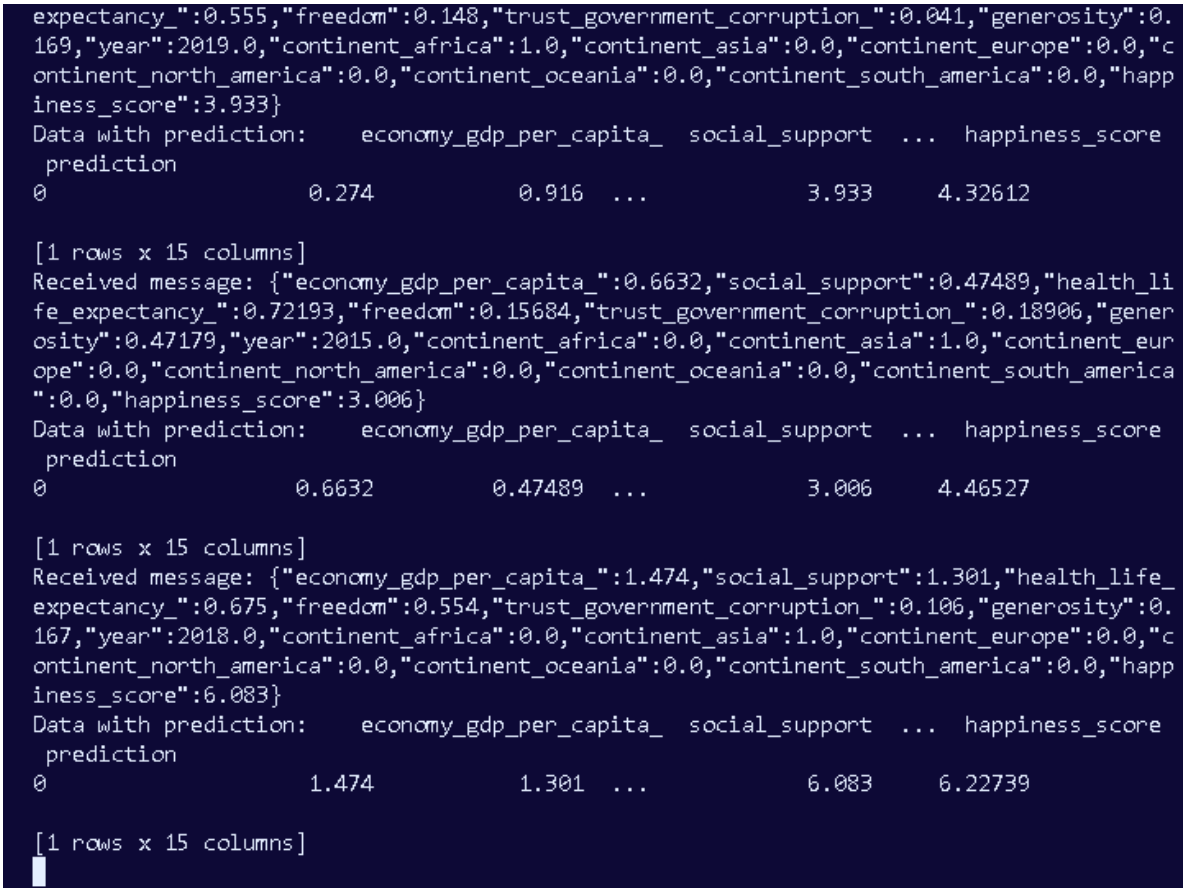
Kafka evidence

Docker zookeeper running:



Producer.py running:

Consumer.py running:



Database receiving the data:

```
1  select * from happiness_predictions
```

Data Output | Messages | Notifications

	economy_gdp_per_capita_ double precision	social_support double precision	health_life_expectancy_ double precision	freedom double precision	trust_government_corruption_ double precision	generosity double precision	year double
1	0.308	0.95	0.391	0.452	0.146	0.22	
2	0.874	1.281	0.365	0.519	0.064	0.051	
3	0.308	0.95	0.391	0.452	0.146	0.22	
4	0.874	1.281	0.365	0.519	0.064	0.051	
5	0.652	0.81	0.424	0.334	0.113	0.216	
6	1.503	1.31	0.825	0.598	0.182	0.262	
7	1.398	1.471	0.819				
8	1.22943	0.95544	0.57386				

✓ Successfully run. Total query runtime: 72 msec. 147 rows affected. ×

✓ Successfully run. Total query runtime: 72 msec. 146 rows affected. ×

Total rows: 147 of 147 | Query complete 00:00:00.072 | Ln 1, Col 37

Conclusions:

Importance of Data Cleaning and Normalization:

Different datasets often have variations in column names and structures. Standardizing these columns is crucial for effective data analysis and modeling.

Handling missing values appropriately, whether by imputation or removal, ensures data quality and integrity.

Effective Feature Engineering:

Adding meaningful columns such as 'Year' and 'Continent' provided additional context that improved the model's predictive power.

Creating dummy variables for categorical data (e.g., continents) helped the model interpret these features better.

Correlation and Feature Selection:

Visualizing correlations between variables through heatmaps can reveal insights into which features are most impactful.

Dropping highly correlated features (like confidence intervals) helps prevent overfitting and simplifies the model without losing predictive power.

Skill Development:

The workshop provided hands-on experience with data preprocessing, exploratory data analysis (EDA), feature engineering, model training, and evaluation.

Participants like me enhanced their skills in using tools like pandas, seaborn, sklearn, joblib, Kafka, and PostgreSQL, making us better equipped for similar projects in the future.

About the class:

About the course, thank you for making such an effort to help us understand the subject, it is very much appreciated to have had a teacher like you so dedicated to the subject and to give a little more to make everything clear and recorded to be able to use it in the future in a more professional field.