

WORKSHOP 3

cut 3

Dayanna Vanessa Suarez Mazuera - 2221224

ETL

Alejandro Vergara Zorrilla

Universidad Autonoma de Occidente

Santiago de Cali, Valle del Cauca

2024 – 01

Project Report: Machine Learning and Data Streaming for Predicting Happiness Score

Introduction

In this project, we aimed to develop a regression model to predict the happiness score of different countries using historical data from five CSV files. The project involved a comprehensive process of Exploratory Data Analysis (EDA), data cleaning and transformation, feature selection, model training and evaluation, and data streaming using Kafka. Finally, the predicted data was stored in a database for further analysis.

Data and Tools

Data: Five CSV files containing happiness data from different years.

Tools: Python, Jupyter Notebook, Kafka, Scikit-learn, SQLAlchemy.

Database: A PostgreSQL database for storing prediction results.

Libraries: Pandas, NumPy, Seaborn, Matplotlib, Joblib, and more.

Exploratory Data Analysis (EDA)

Data Loading and Initial Exploration:

The data from 2015 to 2019 was loaded into Pandas DataFrames.

Basic information such as data types, non-null counts, and initial rows were inspected using `.info()` and `.head()` methods.

Descriptive statistics for each year's data were generated using `.describe()`.

Handling Missing Values:

We identified missing values and null entries, particularly in the 2018 dataset. These null entries were removed to ensure data integrity.

Fixing Column Discrepancies:

Some datasets had columns with different names or additional columns. We standardized the column names and dropped unnecessary columns to make the datasets uniform across years.

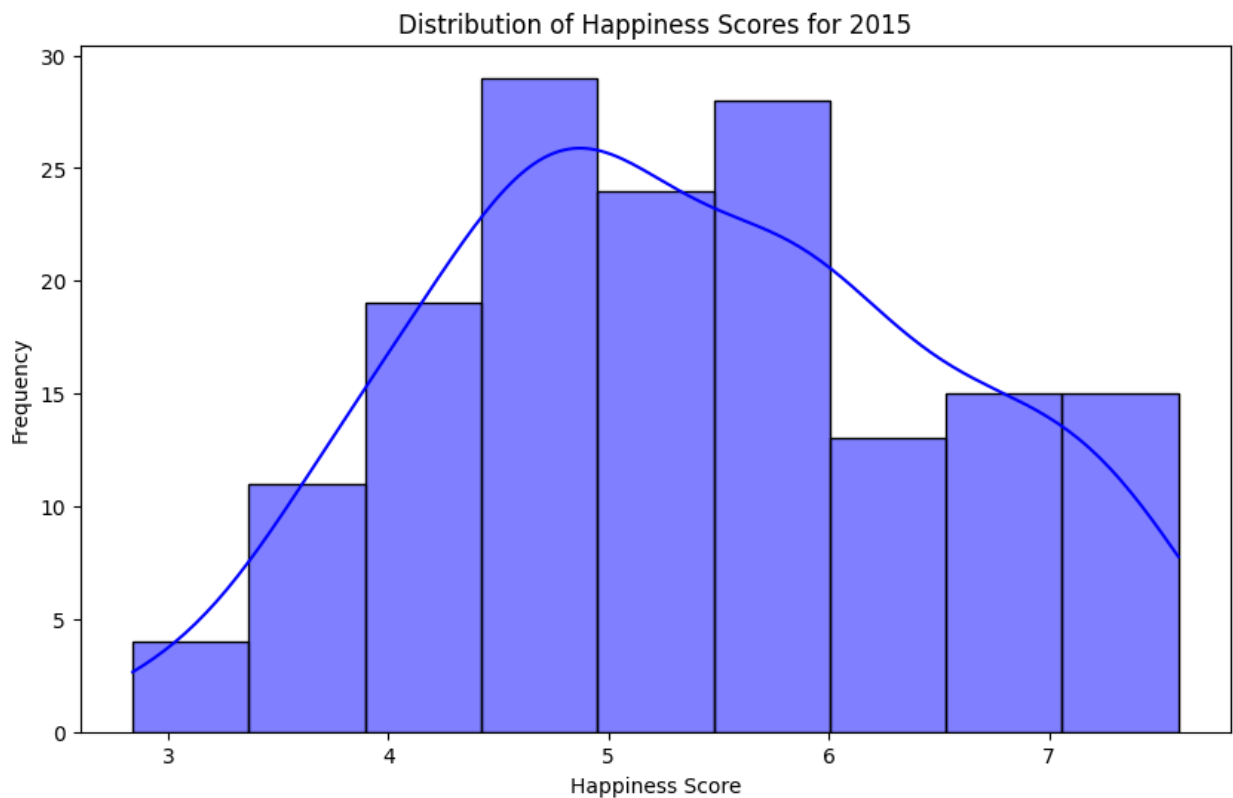
Combining Data:

The cleaned datasets from each year were combined into a single DataFrame. A new column, Year, was added to distinguish the data from different years.

Duplicates and irrelevant columns were removed.

Visualization:

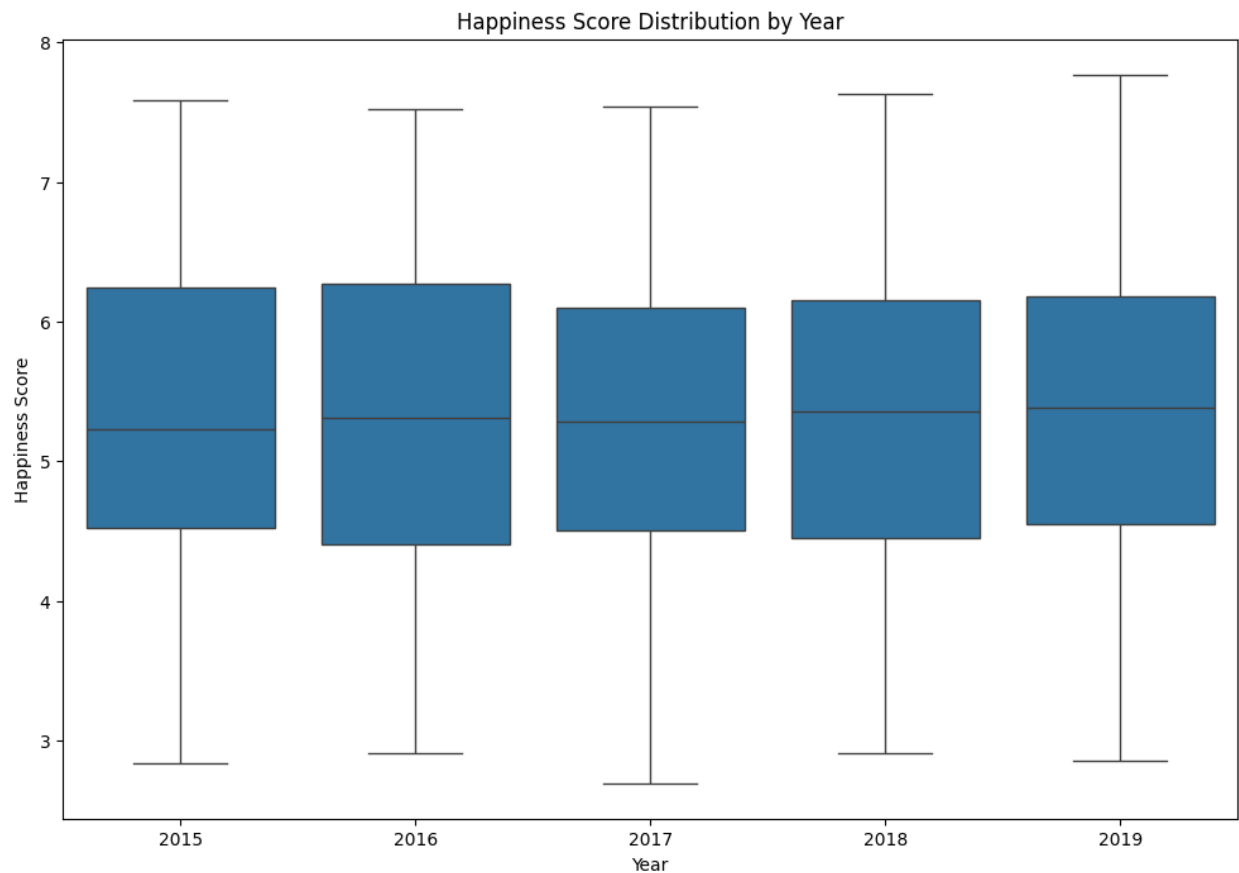
We visualized the distribution of happiness scores for each year using histograms and box plots.



The distribution of happiness scores is roughly bell-shaped, indicating a normal distribution with slight skewness. The scores cover a wide range, reflecting diversity in happiness levels across different countries. Most often, happiness scores fall between 4 and 5, suggesting that most countries have scores within this range.

The distribution is slightly skewed to the left, with a longer tail on the left side, indicating fewer countries with very low happiness scores. The scores range from approximately 3 to 7, showing a moderate spread. This implies that there are countries with both relatively low and high happiness scores, although extreme values are less common.

In summary, the graphic illustrates that most countries have happiness scores in the mid-range, with fewer countries experiencing very low or very high scores in 2015.



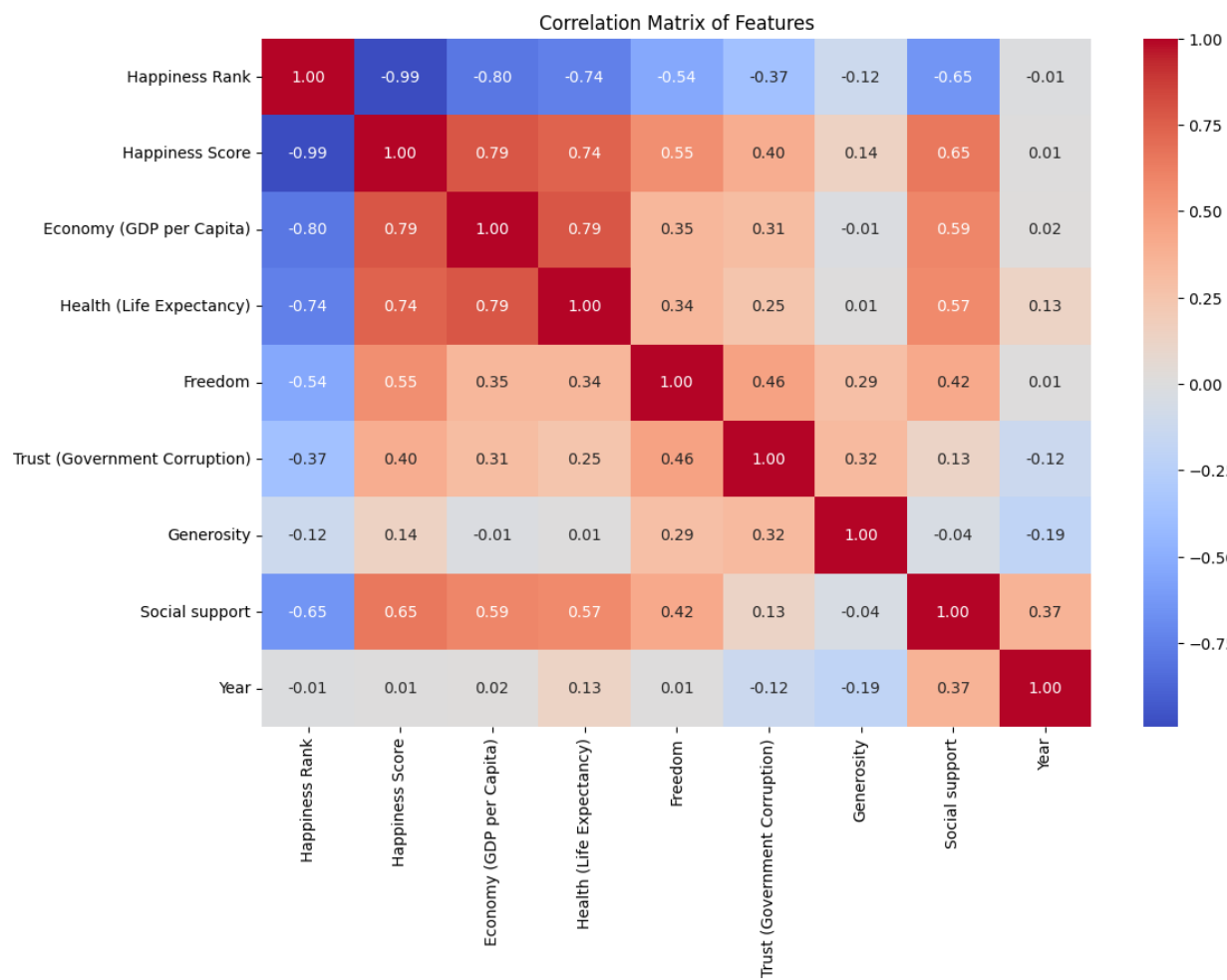
The median happiness scores remained relatively steady from 2015 to 2019, showing no significant fluctuations. This suggests that overall happiness levels were stable during this period. Each year, the happiness scores exhibited a similar range, with the interquartile range (IQR) spanning approximately from 4 to 6.5. This indicates that the central 50% of countries' happiness scores were consistent year over year.

The distribution of happiness scores each year appears fairly symmetrical, as evidenced by the similar lengths of the whiskers and the medians' positions within the boxes.

No notable outliers are present in any year, suggesting that the data does not contain extreme values outside the expected range. The whiskers extend to approximately 3 and 8 across all years, indicating that the overall range of happiness scores remained quite similar from year to year.

In summary, the box plot indicates a stable and consistent distribution of happiness scores from 2015 to 2019, with no significant changes in the overall levels of happiness among the countries in the dataset.

And a correlation heatmap was generated to understand the relationships between different features.



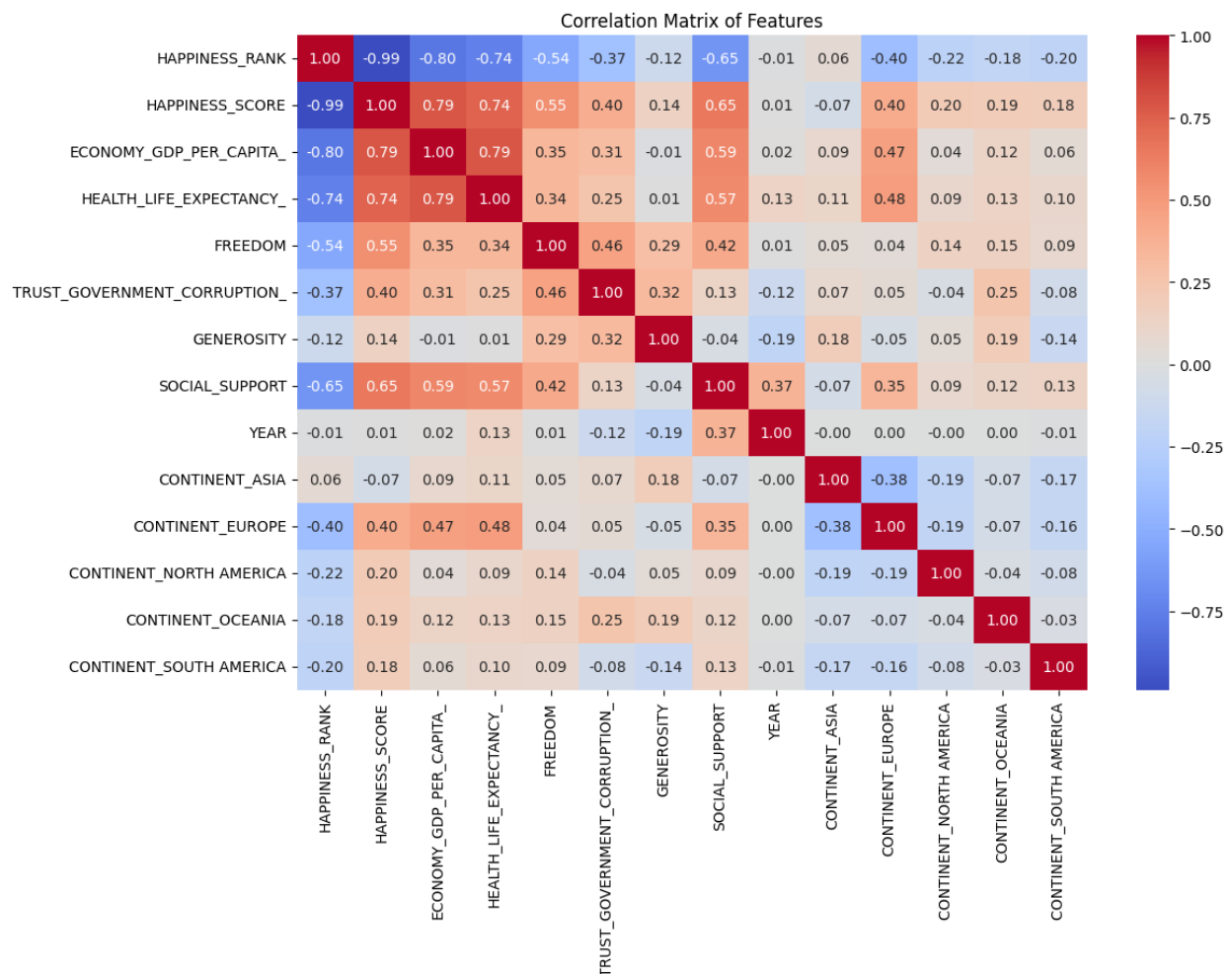
Data Transformation and Feature Engineering

Creating Dummy Variables:

Countries were mapped to their respective continents to create a new categorical feature. This feature was then converted into dummy variables.

Standardizing Column Names:

Column names were standardized to ensure consistency and avoid issues with special characters.



Feature Selection:

Irrelevant features such as Happiness Rank and Year were dropped. Only the features relevant for predicting happiness scores were retained.

Model Training and Evaluation

Data Splitting:

The combined dataset was split into training and testing sets using a 70-30 split.

The training data was used to train multiple regression models, while the testing data was used to evaluate their performance.

Model Training:

Three regression models were trained: Linear Regression, Random Forest Regressor, and Gradient Boosting Regressor.

Linear Regression Model Results:

- Mean Squared Error (MSE): 0.2750301963936654
- Coefficient of determination (R^2): 0.7797452278701514

Random Forest Regression Model Results:

- Mean Squared Error (MSE): 0.204113145423792
- Coefficient of determination (R^2): 0.8365383331593351

Gradient Boosting Regression Model Results:

- Mean Squared Error (MSE): 0.20348507971737131
- Coefficient of determination (R^2): 0.8370413123625796

The models were evaluated using Mean Squared Error (MSE) and the coefficient of determination (R^2).

Model Selection:

The Random Forest Regressor showed the best performance with the lowest MSE and highest R^2 score. This model was saved using Joblib for future predictions.

Data Streaming

After completing the EDA and machine learning model training, the next step was to set up a data streaming pipeline using Kafka. This involved creating both a producer and a consumer to handle the streaming data.

Producer

Kafka Producer Setup:

A Kafka producer was set up to stream the test data. The producer reads the test data from the CSV files and sends it to a Kafka topic named "test-data".

The `data_test` function loads the test data, applies feature selection, continent transformation, dummy variable creation, and column standardization.

The producer function `kafka_producer` sends each row of the transformed test data to the Kafka topic with a delay to simulate streaming.

Consumer

Kafka Consumer Setup:

A Kafka consumer was set up to read the streaming data from the "test-data" topic.

The consumer function `kafka_consumer` listens to the Kafka topic and processes incoming messages.

Each received message is passed to the `predict` function, which uses the pre-trained Random Forest model to make predictions on the happiness score.

The predicted scores are added to the data, and the resulting DataFrame is stored in a database.

Feature Engineering and Transformations

Several feature engineering steps and transformations were implemented to ensure the data was in the right format for both the machine learning model and the streaming process:

Feature Selection:

Certain features that were not relevant to the model, such as "Happiness Rank" and "Year", were dropped.

Continent Mapping:

Countries were mapped to their respective continents to create a new categorical feature, which was then converted into dummy variables.

Column Standardization:

Column names were standardized to ensure consistency and avoid issues with special characters.

Database Integration

Database Configuration:

A database configuration was set up using SQLAlchemy to handle the connection and data storage.

A function `create_table` was implemented to create a table for storing the prediction results if it did not already exist.

The `load_data` function was used to insert the prediction results into the database.

EDA PREDICTION

Loading Prediction Data

To analyze the performance of our machine learning model, we conducted an exploratory data analysis (EDA) on the predictions stored in the database. The database configuration details were loaded from a configuration file, and the connection to the PostgreSQL database was established using SQLAlchemy.

Load Configuration:

The configuration file database.json was read to retrieve the database credentials and connection details.

Establish Database Connection:

Using SQLAlchemy, a connection to the PostgreSQL database was established.

Load Prediction Data:

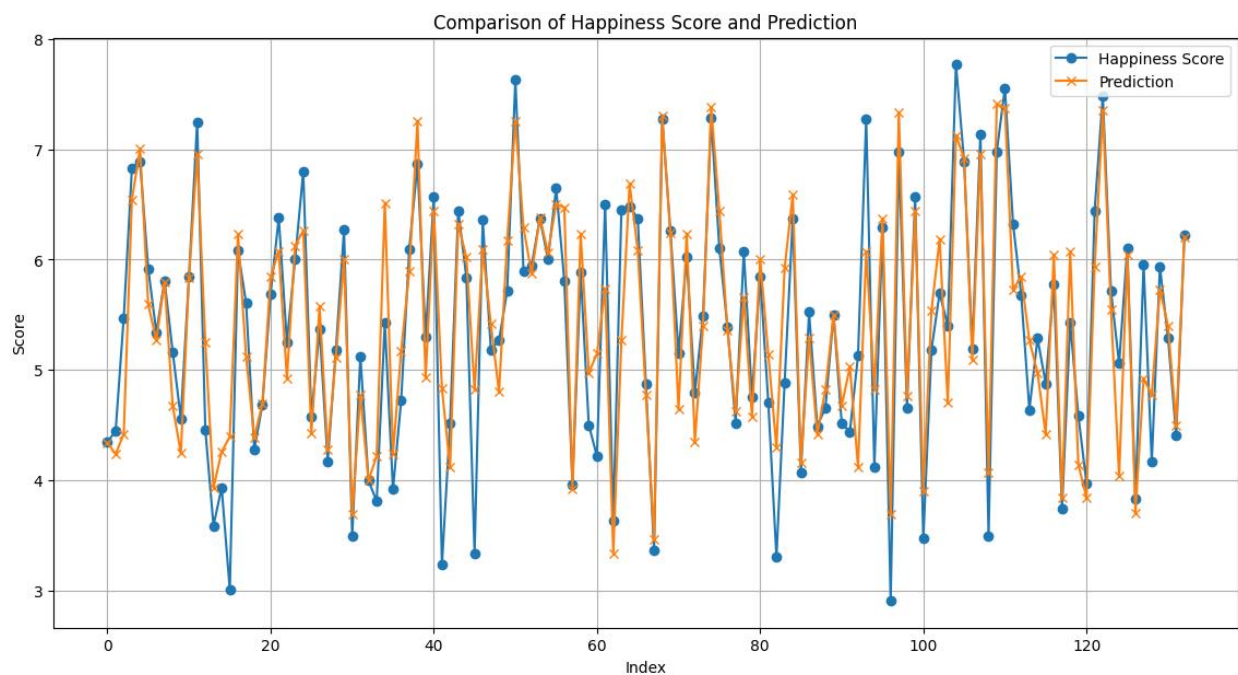
The prediction results, stored in the table prediction, were loaded into a Pandas DataFrame for analysis.

Visualization of Prediction Results

To understand how well the model performed, various visualizations were created to compare the actual happiness scores with the predicted scores.

Line Plot:

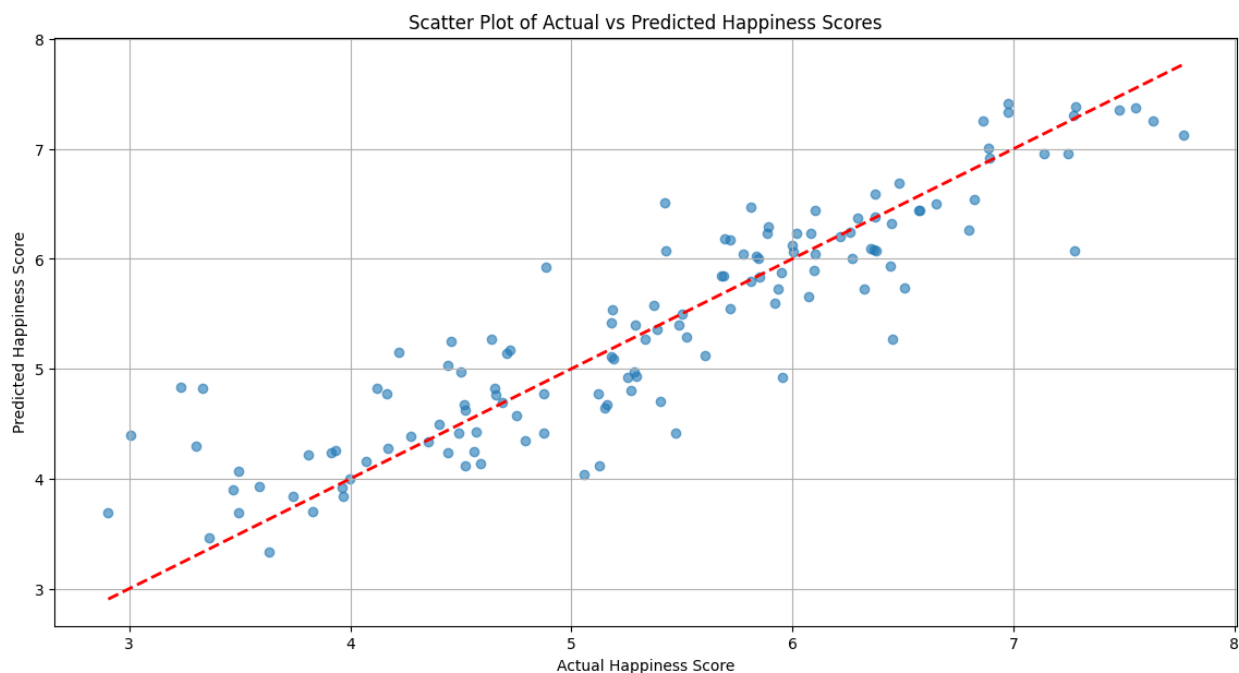
A line plot was created to compare the actual happiness scores and the predicted scores.



The predicted scores (orange crosses) generally follow the pattern of the actual scores (blue circles), indicating that the model captures the overall trend well. There is some variability between the actual and predicted scores, with some predictions being slightly higher or lower than the actual scores. This is expected and indicates areas where the model could be improved. The consistency in the trends suggests that the model is performing reliably across different data points.

Scatter Plot:

A scatter plot of actual vs. predicted happiness scores was created. A reference line with a slope of 1 was added to indicate the ideal scenario where predictions perfectly match the actual scores.

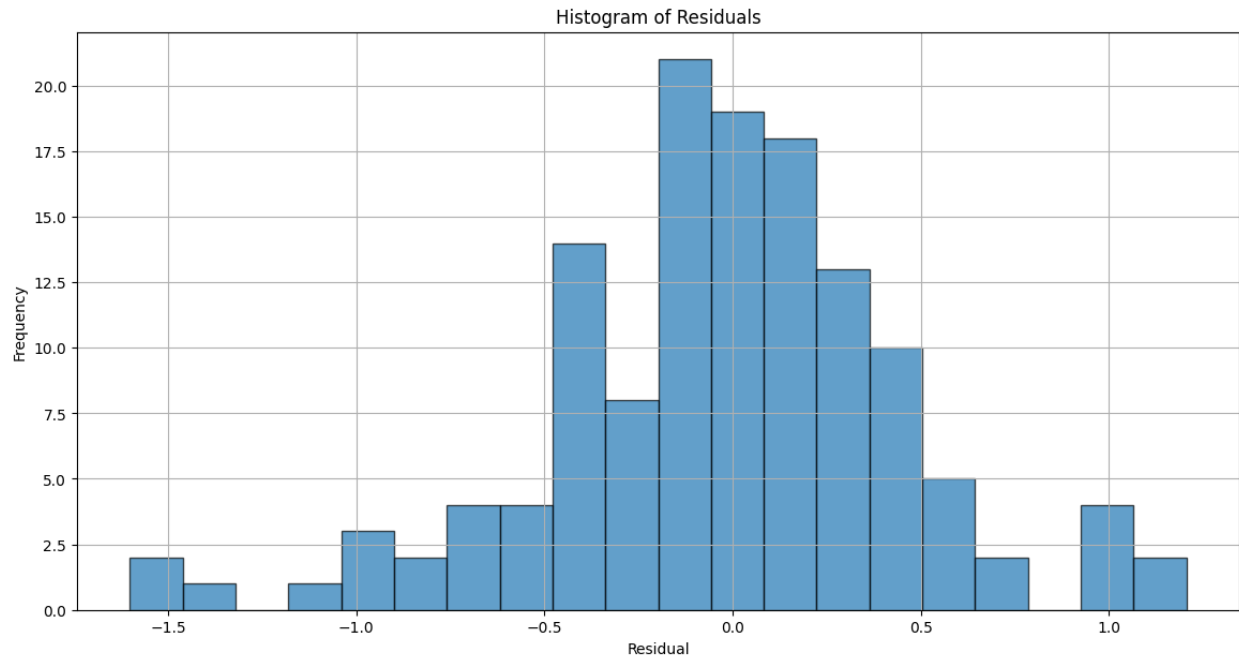


The points are generally clustered around the red dashed line, indicating a strong fit between the actual and predicted values. This suggests that the model is effective in predicting happiness scores.

However, some points deviate significantly from the red dashed line, particularly at the lower and higher ends of the score range. This indicates that the model's predictions are less accurate for these instances, resulting in overestimations or underestimations in certain cases. The spread of points around the line demonstrates that while the model captures the overall trend, there is some variability. The model tends to slightly overestimate or underestimate in different areas.

Residual Histogram:

A histogram of the residuals (the difference between actual and predicted scores) was plotted.



The residuals are centered around zero, indicating that the model's predictions are generally unbiased, with no systematic overestimation or underestimation. The distribution is fairly symmetrical, suggesting that the errors are evenly distributed around zero. This is a positive indicator, implying that the model's errors are normally distributed.

Most residuals fall within the range of -1 to 1, with a higher concentration of residuals near zero. This indicates that the majority of the model's predictions are close to the actual values. There are a few residuals outside the -1 to 1 range, suggesting the presence of outliers where the model's predictions significantly deviate from the actual values.

Box Plot:

Box plots for actual and predicted happiness scores were created to compare the spread and central tendency of both sets of scores. This helps in visualizing the range, median, and any outliers in the predictions.



- Median Comparison: The medians (indicated by the orange lines) of both the actual and predicted scores are very close, showing that the model's predictions align well with the central tendency of the actual data.
- Interquartile Range (IQR): The IQR (the range between the first and third quartiles) of both actual and predicted scores are similar, indicating that the model accurately captures the spread of the central 50% of the data.
- Outliers and Whiskers: The whiskers (lines extending from the top and bottom of the boxes) and the presence of outliers are comparable between the actual and predicted scores, suggesting that the model effectively captures the data range.
- Overall Distribution: The overall shapes of the distributions for the actual and predicted scores are similar, implying that the model not only predicts central values accurately but also captures the variability in the data.

Evaluation Metrics

To quantitatively evaluate the model's performance, several metrics were calculated:

R-squared (R^2):

The R-squared value indicates the proportion of variance in the dependent variable predictable from the independent variables. An R^2 value of 0.84 means that 84% of the variance in happiness scores is explained by the model.

Mean Absolute Error (MAE):

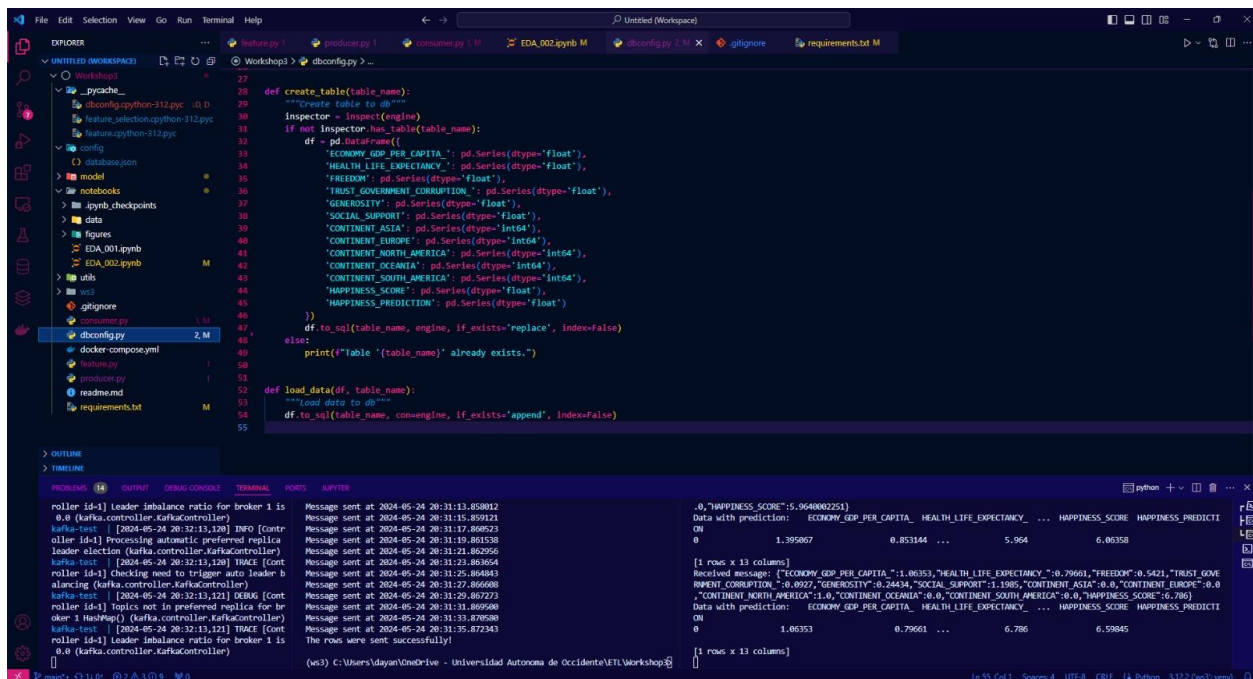
The MAE measures the average magnitude of errors in a set of predictions, without considering their direction. An MAE of 0.34 indicates that, on average, the predictions are off by 0.34 units of happiness score.

Mean Squared Error (MSE):

The MSE measures the average of the squares of the errors, giving more weight to larger errors. An MSE of 0.20 indicates the average squared difference between the actual and predicted happiness scores.

Evidence

Consumer and producer in operation



The screenshot shows a VS Code editor with a Python script named `dbconfig.py` and its execution output in the terminal. The script defines a function `create_table` that creates a table in a database with columns for various happiness indicators. The terminal output shows the script running successfully, creating the table and inserting data.

```
def create_table(table_name):
    """Create table to db"""
    inspector = inspect(engine)
    if not inspector.has_table(table_name):
        df = pd.DataFrame({
            'ECONOMY_GDP_PER_CAPITA': pd.Series(dtype='float'),
            'HEALTH_LIFE_EXPECTANCY': pd.Series(dtype='float'),
            'FREEDOM': pd.Series(dtype='float'),
            'TRUST_GOVERNMENT_CORRUPTION': pd.Series(dtype='float'),
            'GENEROSITY': pd.Series(dtype='float'),
            'SOCIAL_SUPPORT': pd.Series(dtype='float'),
            'CONTINENT_ASIA': pd.Series(dtype='int64'),
            'CONTINENT_EUROPE': pd.Series(dtype='int64'),
            'CONTINENT_NORTH_AMERICA': pd.Series(dtype='int64'),
            'CONTINENT_OCEANIA': pd.Series(dtype='int64'),
            'CONTINENT_SOUTH_AMERICA': pd.Series(dtype='int64'),
            'HAPPINESS_SCORE': pd.Series(dtype='float'),
            'HAPPINESS_PREDICTION': pd.Series(dtype='float')
        })
        df.to_sql(table_name, engine, if_exists='replace', index=False)
    else:
        print(f"Table '{table_name}' already exists.")

def load_data(df, table_name):
    """Load data to db"""
    df.to_sql(table_name, con=engine, if_exists='append', index=False)
```

The terminal output shows the script running successfully, creating the table and inserting data. The output includes the following data:

	ECONOMY_GDP_PER_CAPITA	HEALTH_LIFE_EXPECTANCY	FREEDOM	TRUST_GOVERNMENT_CORRUPTION	GENEROSITY	SOCIAL_SUPPORT	CONTINENT_ASIA	CONTINENT_EUROPE	CONTINENT_NORTH_AMERICA	CONTINENT_OCEANIA	CONTINENT_SOUTH_AMERICA	HAPPINESS_SCORE	HAPPINESS_PREDICTION
0	1.395067	0.85344	...	5.964	6.80358
1	1.86353	0.79661	...	6.786	6.59845

Data saved successfully into posgreSQL

pgAdmin 4

File Object Tools Help

Object Explorer

- Servers (1)
 - PostgreSQL 16
 - Databases (5)
 - candidates
 - games
 - postgres
 - ws2
 - ws3
 - Casto
 - Catalogs
 - Event Triggers
 - Extensions
 - Foreign Data Wrappers
 - Languages
 - Publications
 - Schemas (1)
 - public
 - Aggregates
 - Collations
 - Domains
 - FTS Configurations
 - FTS Dictionaries
 - FTS Parsers
 - FTS Templates
 - Foreign Tables
 - Functions
 - Materialized Views
 - Operators
 - Procedures
 - Sequences
 - Tables (2)
 - Prediction
 - prediction
 - Trigger Functions
 - Types
 - Views
 - Subscriptions
 - Login/Group Roles
 - Tablespaces

ws3/postgres@PostgreSQL 16

Query: `SELECT * FROM prediction`

Data Output Messages Notifications

	ECONOMY_GDP_PER_CAPITA	HEALTH_LIFE_EXPECTANCY	FREEDOM	TRUST_GOVERNMENT_CORRUPTION	GENEROSITY	SOCIAL_SUPPORT	CONTINENT_ASIA	CONTINENT_EUROPE
	double precision	double precision	double precision	double precision	double precision	double precision	bigint	bigint
1	1.503	0.825	0.598	0.182	0.262	1.31	1	0
2	1.398	0.819	0.547	0.133	0.291	1.471	0	0
3	1.22943	0.57386	0.4052	0.11132	0.15011	0.95544	1	0
4	0.9910123944	0.6045900583	0.4184211493	0.1198032722	0.1721704602	1.239088929	1	0
5	1.151	0.599	0.399	0.025	0.065	1.479	0	1
6	0.93383	0.70766	0.09511	0	0.29889	0.64367	0	1
7	0.682	0.343	0.514	0.077	0.091	0.811	0	0
8	1.2607486248	0.6385669708	0.3257079124	0.0738427266	0.1530747861	1.404714942	0	1
9	1.376	1.016	0.532	0.226	0.244	1.475	0	1
10	0.562	0.723	0.527	0.143	0.166	0.928	1	0
11	0.186	0.306	0.531	0.08	0.21	0.541	0	0
12	0.274	0.555	0.148	0.041	0.169	0.916	0	0
13	0.6632	0.72193	0.15684	0.18906	0.47179	0.47489	1	0
14	1.474	0.675	0.554	0.106	0.167	1.301	1	0
15	0.93929	0.61766	0.28579	0.17383	0.07822	1.07772	0	0
16	0.63107	0.29681	0.40973	0.0326	0.21203	0.49353	0	0
17	1.0088	0.69805	0.30033	0.05863	0.38086	0.54447	1	0
18	1.20813	0.92356	0.40472	0.06146	0.30638	0.89318	0	1
19	1.39729	0.79565	0.32377	0.0663	0.25495	0.92624	1	0
20	0.77042	0.57407	0.53206	0.15445				

Total rows: 232 of 232 Query complete 00:00:00.088

Successfully run. Total query runtime: 88 msec. 232 rows affected.

Ln 1, Col 25