

WORKSHOP 3

MARÍA FERNANDA TELLO VERGARA

2225338

JAVIER ALEJANDRO VERGARA

ETL

UNIVERSIDAD AUTÓNOMA DE OCCIDENTE

NOVIEMBRE 14 2024

CONTEXT

We train a regression model with all the EDA and feature selection, stream the data using kafka, use the trained model to predict values of the test dataset and extract a performance metric.

DESCRIPTION

We are going to train a regression machine learning model to predict happiness score.

I perform ETL to extract features from the files, train the model using a 70-30 data split (70% for training - 30% for testing), stream the transformed data and then at the consumer get the data and use the trained model to predict happiness score and finally store the predictions with the respective inputs in a database.

TOOLS

- Python
- Jupiter Notebook
- Database – postgres
- Kafka
- CSV files
- Scikit-learn library

STEP BY STEP

EDA

In the following images you can see that we read each of the datasets.

```
data_2015.head(5)
```

	Country	Region	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedom	Trust (Government Corruption)	Generosity	Dysto Resid
0	Switzerland	Western Europe	1	7.587	0.03411	1.39651	1.34951	0.94143	0.66557	0.41978	0.29678	2.517
1	Iceland	Western Europe	2	7.561	0.04884	1.30232	1.40223	0.94784	0.62877	0.14145	0.43630	2.702
2	Denmark	Western Europe	3	7.527	0.03328	1.32548	1.36058	0.87464	0.64938	0.48357	0.34139	2.492
3	Norway	Western Europe	4	7.522	0.03880	1.45900	1.33095	0.88521	0.66973	0.36503	0.34699	2.465
4	Canada	North America	5	7.427	0.03553	1.32629	1.32261	0.90563	0.63297	0.32957	0.45811	2.451

```
data_2016.head(5)
```

	Country	Region	Happiness Rank	Happiness Score	Lower Confidence Interval	Upper Confidence Interval	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedom	Trust (Government Corruption)	Ger
0	Denmark	Western Europe	1	7.526	7.460	7.592	1.44178	1.16374	0.79504	0.57941	0.44453	
1	Switzerland	Western Europe	2	7.509	7.428	7.590	1.52733	1.14524	0.86303	0.58557	0.41203	
2	Iceland	Western Europe	3	7.501	7.333	7.669	1.42666	1.18326	0.86733	0.56624	0.14975	
3	Norway	Western Europe	4	7.498	7.421	7.575	1.57744	1.12690	0.79579	0.59609	0.35776	
4	Finland	Western Europe	5	7.413	7.351	7.475	1.40598	1.13464	0.81091	0.57104	0.41004	

```
data_2017.head(5)
```

	Country	Happiness.Rank	Happiness.Score	Whisker.high	Whisker.low	Economy..GDP.per.Capita.	Family	Health..Life.Expectanc
0	Norway	1	7.537	7.594445	7.479556	1.616463	1.533524	0.79666
1	Denmark	2	7.522	7.581728	7.462272	1.482383	1.551122	0.79256
2	Iceland	3	7.504	7.622030	7.385970	1.480633	1.610574	0.83359
3	Switzerland	4	7.494	7.561772	7.426227	1.564980	1.516912	0.85813
4	Finland	5	7.469	7.527542	7.410458	1.443572	1.540247	0.80915

Se muestra la información de los primeros cinco países. Las variables clave, como Happiness Score, Economy (GDP per Capita), y Dystopia Residual, siguen presentes.

```
data_2018.head(5)
```

	Overall rank	Country or region	Score	GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices	Generosity	Perceptions of corruption
0	1	Finland	7.632	1.305	1.592	0.874	0.681	0.202	0.393
1	2	Norway	7.594	1.456	1.582	0.861	0.686	0.286	0.340
2	3	Denmark	7.555	1.351	1.590	0.868	0.683	0.284	0.408
3	4	Iceland	7.495	1.343	1.644	0.914	0.677	0.353	0.138
4	5	Switzerland	7.487	1.420	1.549	0.927	0.660	0.256	0.357

```
data_2019.head(5)
```

	Overall rank	Country or region	Score	GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices	Generosity	Perceptions of corruption
0	1	Finland	7.769	1.340	1.587	0.986	0.596	0.153	0.393
1	2	Denmark	7.600	1.383	1.573	0.996	0.592	0.252	0.410
2	3	Norway	7.554	1.488	1.582	1.028	0.603	0.271	0.341
3	4	Iceland	7.494	1.380	1.624	1.026	0.591	0.354	0.118
4	5	Netherlands	7.488	1.396	1.522	0.999	0.557	0.322	0.298

We will remove the columns that will not be used in the model.

```
# Concatenar los DataFrames
df_concatenado = pd.concat(dataframes, axis=0, ignore_index=True)

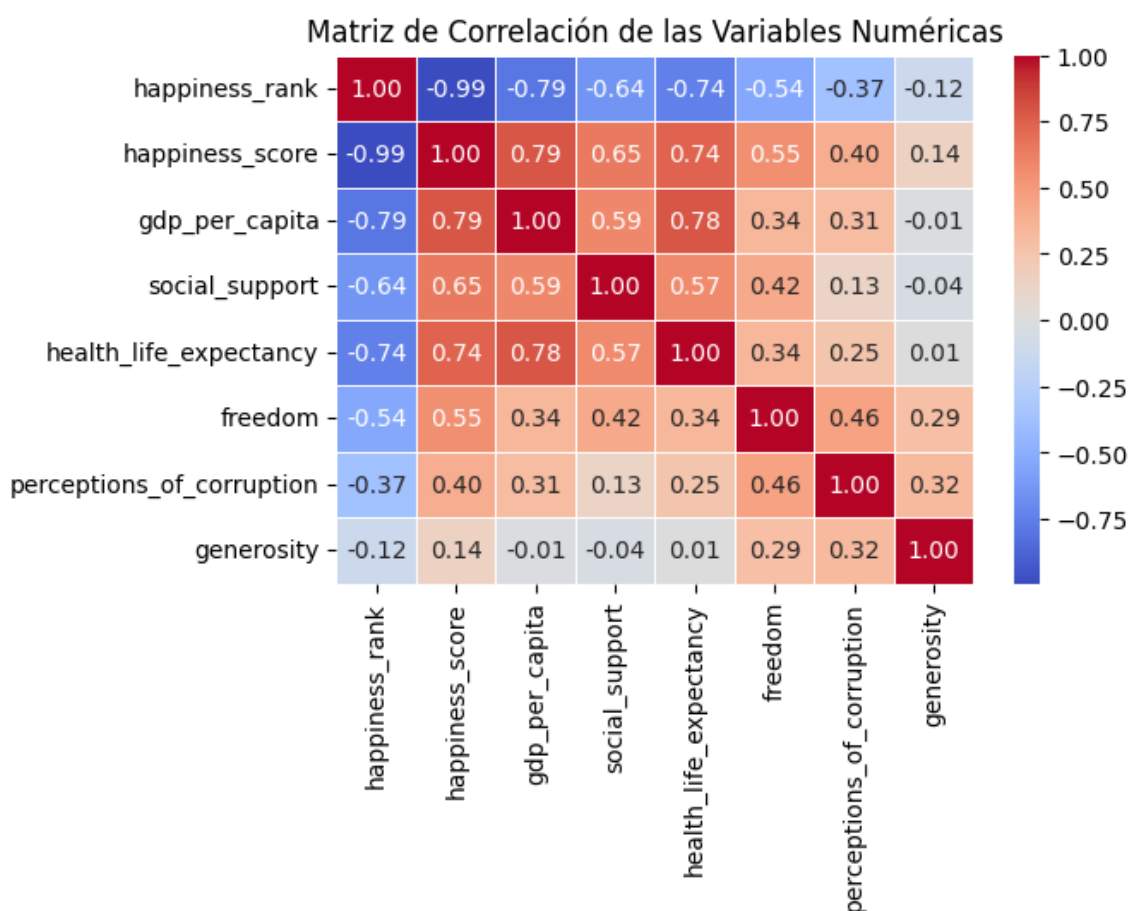
# Mostrar la forma del DataFrame final para verificar
print(f"El DataFrame concatenado tiene {df_concatenado.shape[0]} filas y {df_concatenado.shape[1]} columnas.")
```

El DataFrame concatenado tiene 782 filas y 9 columnas.

Concatenation of DataFrames.

```
# Eliminación de columnas no deseadas, ignorando errores si las columnas no existen
data_2015.drop(columns=['region', 'standard_error', 'dystopia_residual'], inplace=True, errors='ignore')
data_2016.drop(columns=['region', 'lower_confidence_interval', 'upper_confidence_interval', 'dystopia_residual'], inplace=True, errors='ignore')
data_2017.drop(columns=['whisker_high', 'whisker_low', 'dystopia_residual'], inplace=True, errors='ignore')
```

Correlation matrix.



- Happiness Rank and Happiness Score: There is a strong negative correlation (-0.99). This makes sense because a lower Happiness Rank (lower number in the ranking) indicates a higher Happiness Score.

- GDP per Capita and Happiness Score: High positive correlation (0.79), suggesting that as GDP per capita increases, happiness scores also tend to increase.

- Health (Life Expectancy) and Happiness Score: High positive correlation (0.74), indicating that a higher healthy life expectancy is related to a higher level of happiness.
- Social Support and Happiness Score: Significant positive correlation (0.65), showing that higher social support is associated with higher happiness.
- Perceptions of Corruption: Negative correlation with the Happiness Score (-0.40), indicating that as the perception of corruption decreases, the level of happiness increases.
- Freedom and Happiness Score: Positive correlation (0.55), suggesting that countries with more freedom to make personal decisions tend to be happier.

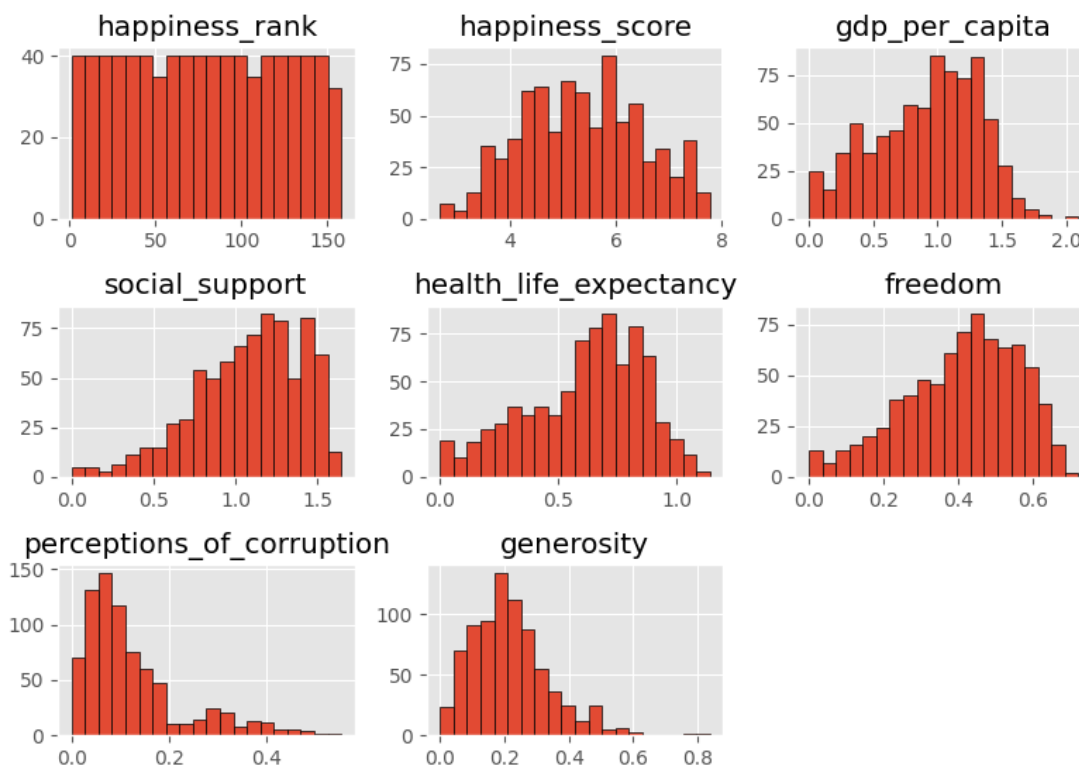
The variables that most positively influence happiness are:

- GDP per Capita.
- Health (Life Expectancy).
- Social Support.

On the other hand, the perception of corruption has a negative impact on happiness, meaning that countries with a lower perception of corruption tend to be happier.

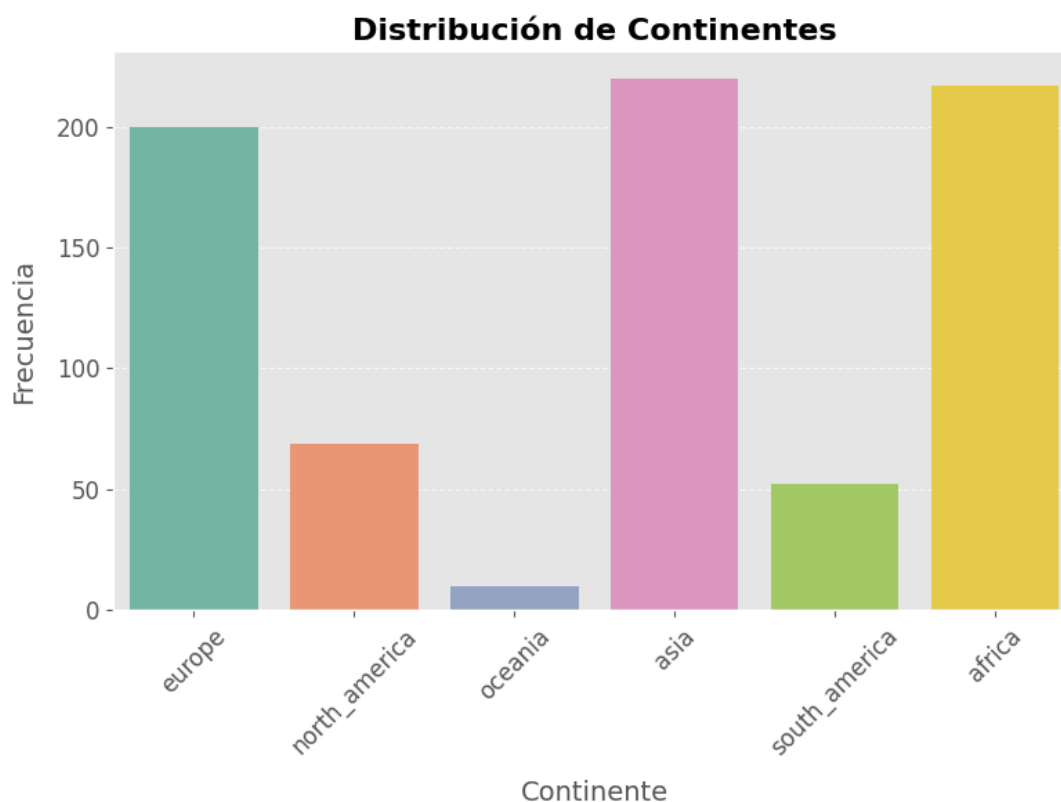
Histograms of numerical variables to detect distribution and outliers.

Distribución de las Variables Numéricas



- Happiness Rank: Even distribution, countries are evenly distributed in the happiness ranking.
- Happiness Score: Most countries have happiness scores between 4 and 7.
- GDP per Capita: Most have low GDP, with few countries at the high end.
- Social Support: Most countries have high social support (around 1.5).
- Health (Life Expectancy): Life expectancy is concentrated between 0.5 and 1.
- Freedom: Perceived freedom is between 0.3 and 0.6.
- Perceptions of Corruption: Most countries have low perceptions of corruption (< 0.2).
- Generosity: Low generosity predominates, with few countries at high values.

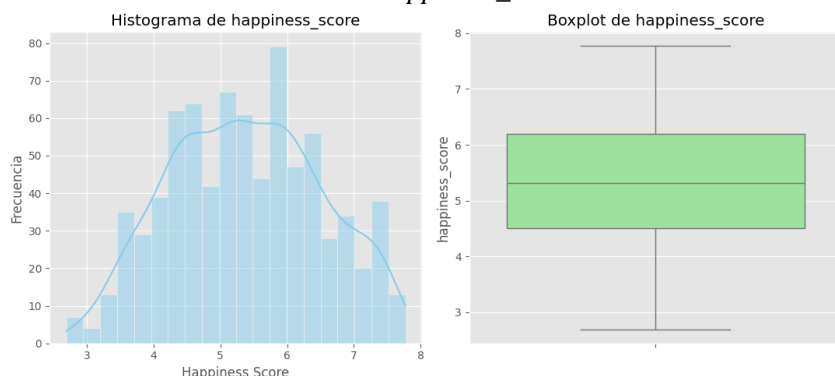
Bar chart to show the distribution of continents.



Asia, Africa, and Europe are the continents with the highest frequency, representing a large number of countries in the data set.

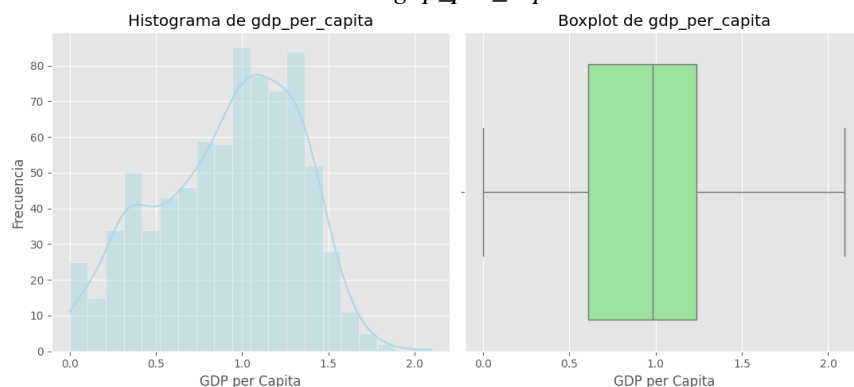
- Oceania and South America have the lowest frequencies.
- North America occupies an intermediate position in terms of frequency.

Column happiness_score



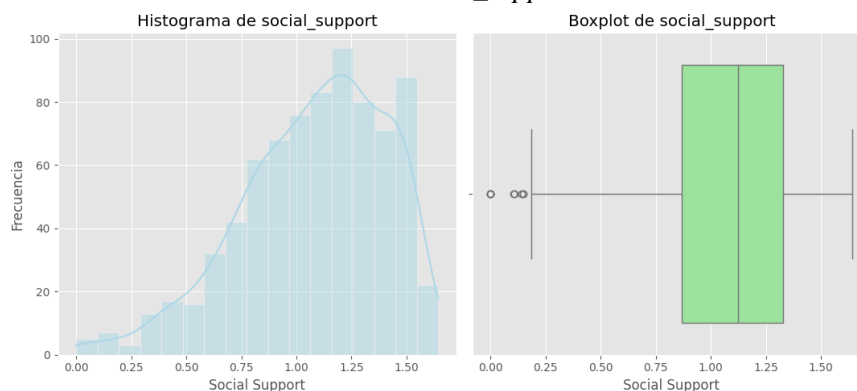
Analysis of the distribution of happiness scores shows that most countries have a score between 4.5 and 6.5, with an average of 5.41. The median is 5.38, indicating that half of the countries have a score at or below this value. No significant outliers are observed, and the variability of the scores is moderate, with a standard deviation of 1.11. Overall, happiness scores are relatively evenly distributed, with a slight tendency towards higher values.

Column gdp_per_capita



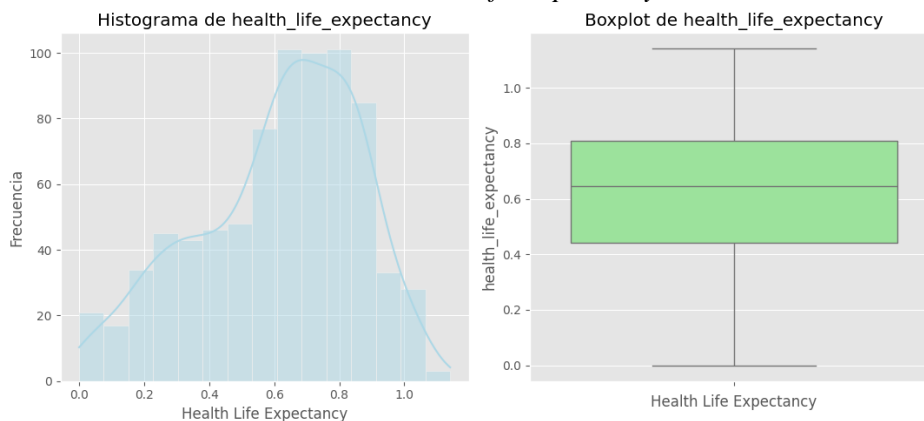
Analysis of GDP per capita shows that most countries have an average value close to 0.91, with few extreme countries. The distribution has an accumulation around the value of 1, while the maximum value reaches 1.87. The histogram indicates that most economies are concentrated near the average.

Column social_support



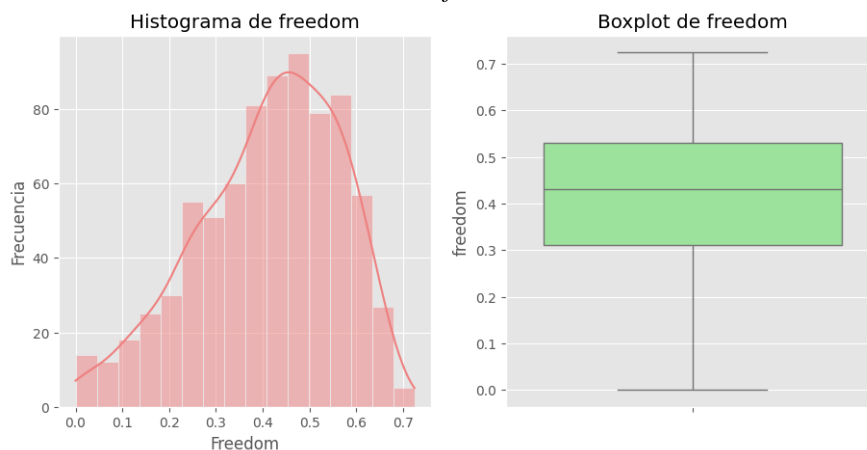
Analysis of the `social_support` column shows that the average social support is approximately 1.078. The histogram presents a distribution that is skewed to the left, with most countries reporting high levels of social support, close to 1.5. The boxplot indicates that most of the data are clustered between 0.87 and 1.32, with some outliers below 0.25, reflecting countries with significantly low levels of social support.

Column health_life_expectancy



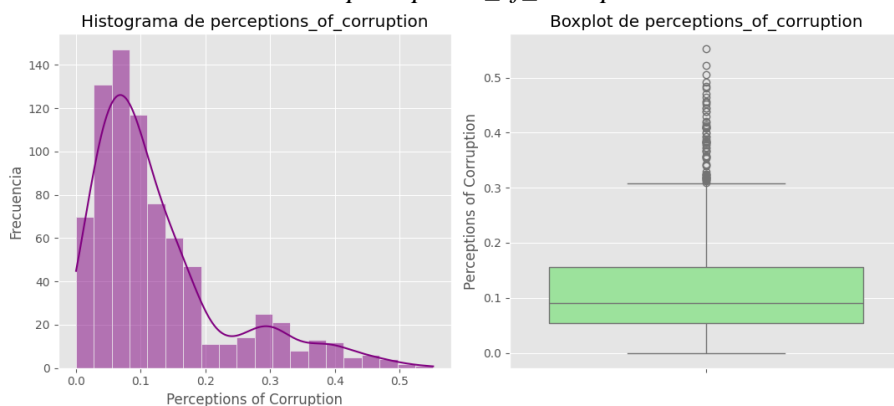
Analysis of the `health_life_expectancy` column shows that the mean is 0.61, indicating that, on average, healthy life expectancy is relatively high among the countries analyzed. The histogram reveals that most countries have values close to 0.6 or higher, while the boxplot shows that there are some values close to 0, which could be considered as outliers, and reflect countries with a lower healthy life expectancy.

Column freedom



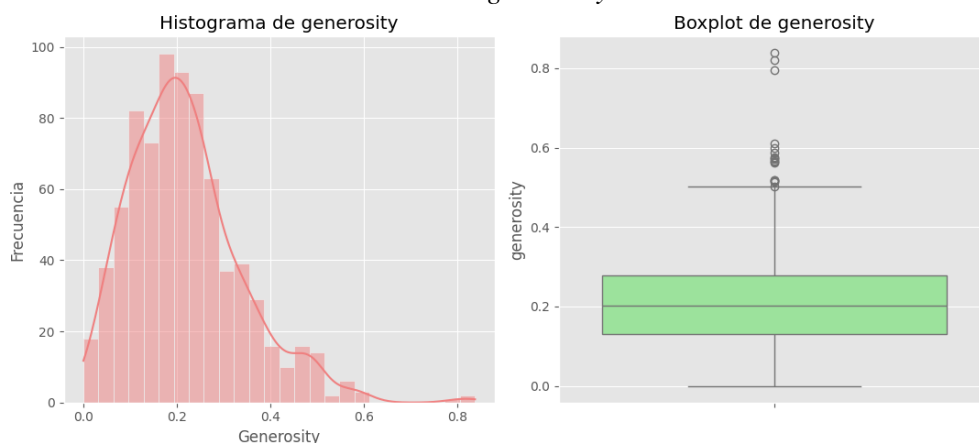
Analysis of the `freedom` column reveals that most countries have freedom scores in the range of 0.3 to 0.6, with a median of 0.41. The histogram shows a right-skewed distribution, indicating that there are countries with relatively low levels of freedom. The boxplot shows that the median is around 0.43, and although most countries are clustered in the interquartile range between 0.31 and 0.53, there are some minimum values close to 0, indicating possible countries with very little freedom.

Column perceptions_of_corruption



Most countries have low perceptions of corruption, with values concentrated between 0.05 and 0.15. However, some countries have higher values, indicating disparities in the perception of corruption.

Column generosity



The generosity column shows a right-skewed distribution, with most values concentrated between 0 and 0.3. The outliers in the boxplot indicate that some countries have higher levels of generosity. The mean is 0.218, reflecting that generosity tends to be low in most countries.

Finally, we save the dataset to continue with the model training.

```
# Guardar el DataFrame limpio en la carpeta 'data'
df_cleaned.to_csv('../data/datos_limpios.csv', index=False)
```

CLEAN DATA

In the following image you can see the clean dataset with the columns that we have selected for training the model.

	happiness_score	gdp_per_capita	social_support	health_life_expectancy	freedom	perceptions_of_corruption	generosity	year	continent_africa	continent_asia	continent_europe	continent_north_america	continent_oceania	continent_south_america
2	7.561	1.30232	1.40223	0.94784	0.62877	0.14145	0.4363	2015	0	0	1	0	0	0
3	7.527	1.32548	1.36058	0.87464	0.64938	0.48357	0.34139	2015	0	0	1	0	0	0
4	7.522	1.459	1.33095	0.88521	0.66973	0.36503	0.34699	2015	0	0	1	0	0	0
5	7.427	1.32629	1.32261	0.90563	0.63297	0.32957	0.45811	2015	0	0	0	1	0	0
6	7.406	1.29025	1.31826	0.88911	0.64169	0.41372	0.23351	2015	0	0	1	0	0	0
7	7.378	1.32944	1.28017	0.89284	0.61576	0.31814	0.4761	2015	0	0	1	0	0	0
8	7.364	1.33171	1.28907	0.91087	0.6598	0.43844	0.36262	2015	0	0	1	0	0	0
9	7.286	1.25018	1.31967	0.90837	0.63938	0.42922	0.47501	2015	0	0	0	0	1	0
10	7.284	1.33358	1.30923	0.93156	0.65124	0.35637	0.43562	2015	0	0	0	0	1	0
11	7.278	1.22857	1.22393	0.91387	0.41319	0.07785	0.33172	2015	0	1	0	0	0	0
12	7.226	0.95578	1.23788	0.86027	0.63376	0.10583	0.25497	2015	0	0	0	1	0	0
13	7.2	1.33723	1.29704	0.89042	0.62433	0.18676	0.33088	2015	0	0	1	0	0	0
14	7.187	1.02054	0.91451	0.81444	0.48181	0.21312	0.14074	2015	0	0	0	1	0	0
15	7.119	1.39451	1.24711	0.86179	0.54604	0.1589	0.40105	2015	0	0	0	1	0	0

TRAINING MODEL

Define x (all columns except the target column) and y (the target column).

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

# Dividir el dataset en entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

MODELS

- Linear Regression: Assumes a linear relationship between the independent and dependent variables, seeking to minimize errors between predictions and actual values.
- Random Forest: Set of decision trees that average their predictions, which reduces variance and improves accuracy.
- Ridge Regression: Linear regression with regularization, penalizes large coefficients to avoid overfitting.
- Gradient Boosting: Algorithm that trains sequential models, where each model corrects the errors of the previous one, achieving a more accurate model.
- KNeighbors Regressor (KNN): Predicts values based on the nearest neighbors, making decisions based on the distance between observations.
- XGBoost: Optimized version of Gradient Boosting, fast and accurate, with internal optimizations that make it effective in machine learning.

METRICS

- R^2 (Coefficient of determination): Indicates how well the predictions explain the variability of the data. A value close to 1 indicates a better fit.
- MSE (Mean Squared Error): Calculates the average squared error between the predictions and the actual values, penalizing large errors more.
- RMSE (Root Mean Squared Error): It is the square root of the MSE, making interpretation easier as it is on the same scale as the actual values. It penalizes large errors.
- MAE (Mean Absolute Error): It measures the average absolute error without penalizing large errors as heavily, since it uses absolute values.

Linear regression.

```
model = LinearRegression()
model.fit(X_train, y_train)
y_pred_linear = model.predict(X_test)
mse, rmse, mae, r2_linear = calcular_metricas(y_test, y_pred_linear)

print(f"Mean squared error on the test set (mse): {mse}")
print(f"Root Mean Squared Error (RMSE): {rmse}")
print(f"Mean Absolute Error (MAE): {mae}")
print(f"Coeficiente de determinación (R²): {r2_linear}")
```

Mean squared error on the test set (mse): 0.23571260881993988
Root Mean Squared Error (RMSE): 0.4855024292626556
Mean Absolute Error (MAE): 0.3601615601675249
Coeficiente de determinación (R²): 0.8031904397594047

Random Forest.

```
model_rfr = RandomForestRegressor(n_estimators=100, random_state=50)
model_rfr.fit(X_train, y_train)
y_pred = model_rfr.predict(X_test)
mse, rmse, mae, r2 = calcular_metricas(y_test, y_pred)

print(f"Mean squared error on the test set (mse): {mse}")
print(f"Root Mean Squared Error (RMSE): {rmse}")
print(f"Mean Absolute Error (MAE): {mae}")
print(f"Coeficiente de determinación (R²): {r2}")
```

Mean squared error on the test set (mse): 0.18690181026447028
Root Mean Squared Error (RMSE): 0.43232142008518415
Mean Absolute Error (MAE): 0.3369188661251941
Coeficiente de determinación (R²): 0.8439452888393391

Ridge Regression.

```
model_ridge = Ridge()
model_ridge.fit(X_train, y_train)
y_pred_ridge = model_ridge.predict(X_test)
mse, rmse, mae, r2_ridge = calcular_metricas(y_test, y_pred_ridge)

print(f"Mean squared error on the test set (mse): {mse}")
print(f"Root Mean Squared Error (RMSE): {rmse}")
print(f"Mean Absolute Error (MAE): {mae}")
print(f"Coeficiente de determinación (R²): {r2_ridge}")
```

Mean squared error on the test set (mse): 0.23486251971561767
Root Mean Squared Error (RMSE): 0.48462616491025085
Mean Absolute Error (MAE): 0.3608132399178679
Coeficiente de determinación (R²): 0.8039002264085983

Gradient Boosting.

```
model_gb = GradientBoostingRegressor(n_estimators=100, random_state=42)
model_gb.fit(X_train, y_train)
y_pred_gb = model_gb.predict(X_test)
mse, rmse, mae, r2_gb = calcular_metricas(y_test, y_pred_gb)

print(f"Mean squared error on the test set (mse): {mse}")
print(f"Root Mean Squared Error (RMSE): {rmse}")
print(f"Mean Absolute Error (MAE): {mae}")
print(f"Coeficiente de determinación (R²): {r2_gb}")
```

Mean squared error on the test set (mse): 0.17877536882798506
Root Mean Squared Error (RMSE): 0.4228183638726978
Mean Absolute Error (MAE): 0.32804981618176143
Coeficiente de determinación (R²): 0.8507305065391582

KNeighbors Regressor (KNN).

```
model_knn = KNeighborsRegressor()
model_knn.fit(X_train, y_train)
y_pred_knn = model_knn.predict(X_test)
mse, rmse, mae, r2_knn = calcular_metricas(y_test, y_pred_knn)

print(f"Mean squared error on the test set (mse): {mse}")
print(f"Root Mean Squared Error (RMSE): {rmse}")
print(f"Mean Absolute Error (MAE): {mae}")
print(f"Coeficiente de determinación (R²): {r2_knn}")
```

Mean squared error on the test set (mse): 0.25778268976021435
Root Mean Squared Error (RMSE): 0.5077230443462404
Mean Absolute Error (MAE): 0.3884391269269197
Coeficiente de determinación (R²): 0.7847629023184703

XGBoost

```
model_xgb = XGBRegressor(n_estimators=100, random_state=42)
model_xgb.fit(X_train, y_train)
y_pred_xgb = model_xgb.predict(X_test)
mse, rmse, mae, r2_xgb = calcular_metricas(y_test, y_pred_xgb)

print(f"Mean squared error on the test set (mse): {mse}")
print(f"Root Mean Squared Error (RMSE): {rmse}")
print(f"Mean Absolute Error (MAE): {mae}")
print(f"Coeficiente de determinación (R²): {r2_xgb}")
```

Mean squared error on the test set (mse): 0.19644346786650072
Root Mean Squared Error (RMSE): 0.44321943534382685
Mean Absolute Error (MAE): 0.355347638362387
Coeficiente de determinación (R²): 0.8359784284918026

Gradient Boosting is the best performing model, as:

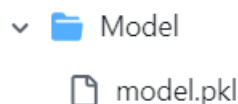
- It has the highest R^2 value (0.8507), indicating that it explains most of the variability in the data.
- It has the lowest MSE, RMSE, and MAE values compared to the other models, indicating that the errors in the predictions are lower.

We keep the best model.

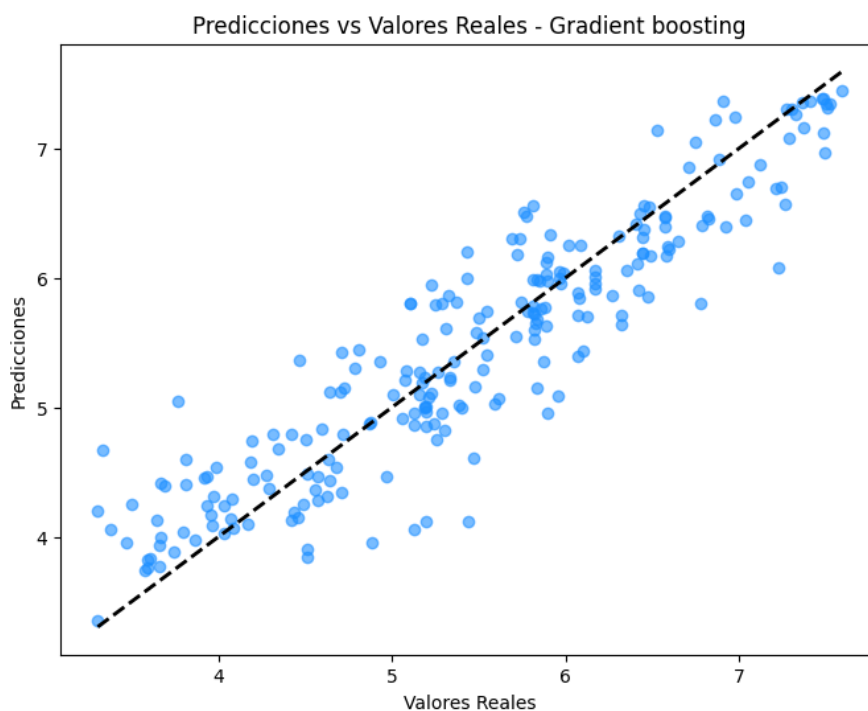
```
: joblib.dump(model_gb, '../Model/model.pkl')
```

```
: ['../Model/model.pkl']
```

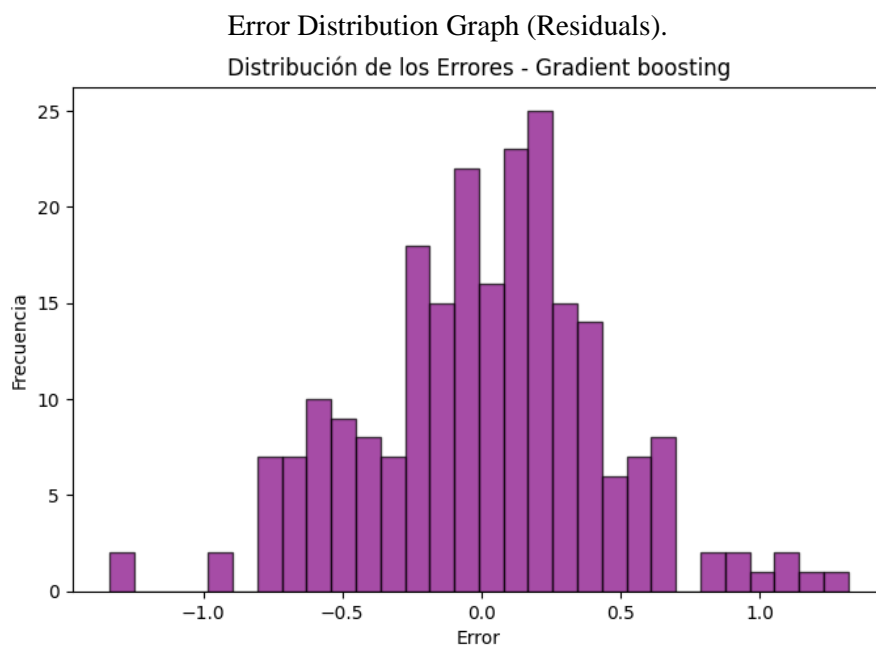
As we can see in the following image, the model has already been saved.



Plot of Predictions vs. Actual Values for the Best Model (Gradient Boosting).



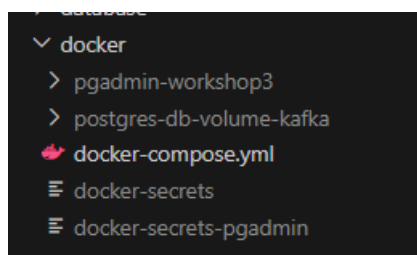
In this Predictions vs. Actual Values graph for the Gradient boosting model, we can see that most of the points are close to the dotted diagonal line, indicating that the model predictions are fairly aligned with the actual values. This suggests that the model is performing well and that the predictions are reasonably accurate. However, there is still some scatter, especially at low values, showing that some predictions may be somewhat off from the actual values.



In this histogram of the Error Distribution for the Gradient boosting model, we observe that most errors are close to zero, which is a positive sign that the model is making accurate predictions overall. The symmetrical shape centered around zero indicates that the errors are distributed in a relatively balanced manner, with no clear bias towards overprediction.

DOCKER PROCESS

Inside the docker folder we have several files added in the .gitignore, since they are some credentials for the database.



On the other hand, we have the docker-compose.yml, which is the one we configure to run four containers in the same environment:

1. A Zookeeper container to manage the Kafka cluster.
2. A Kafka container that depends on Zookeeper and acts as the message broker.
3. A PostgreSQL container to manage a persistent database.
4. A pgAdmin container to manage PostgreSQL from a graphical interface.

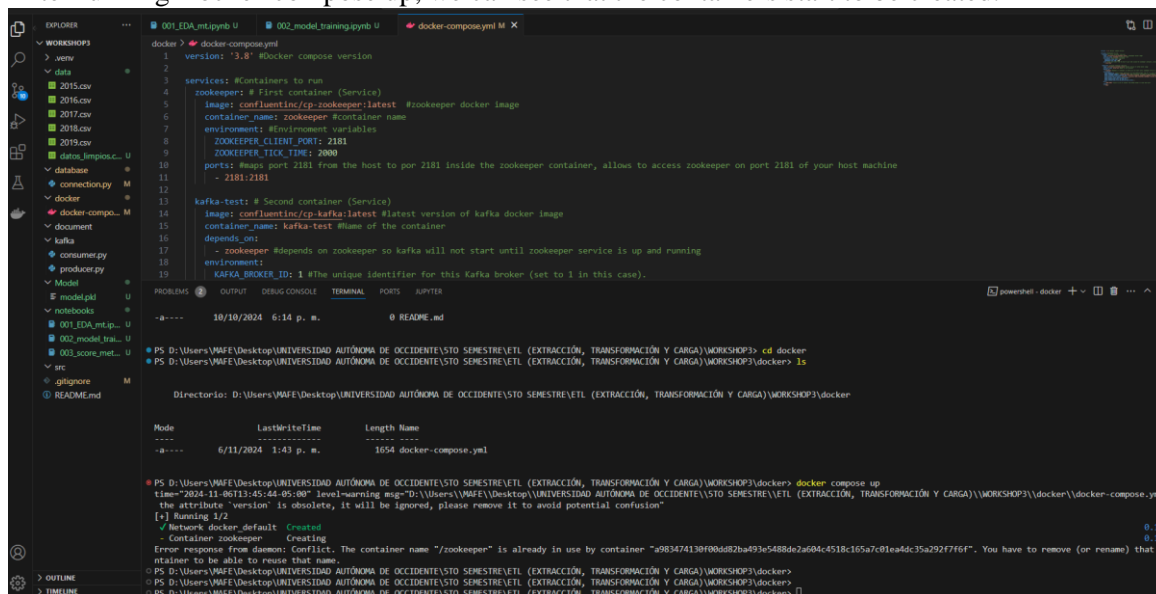
```

services: # Containers to run
  zookeeper: # First container (Service)
    image: confluentinc/cp-zookeeper:latest # zookeeper docker image
    container_name: zookeeper # container name
    environment: # Environment variables
      ZOOKEEPER_CLIENT_PORT: 2181
      ZOOKEEPER_TICK_TIME: 2000
    ports: # Maps port 2181 from the host to port 2181 inside the zookeeper container, allows access to zookeeper on port 2181 of your host machine
      - 2181:2181

  kafka-test: # Second container (Service)
    image: confluentinc/cp-kafka:latest # latest version of kafka docker image
    container_name: kafka-test # Name of the container
    depends_on:
      - zookeeper # Depends on zookeeper so kafka will not start until zookeeper service is up and running
    environment:
      KAFKA_BROKER_ID: 1 # The unique identifier for this Kafka broker (set to 1 in this case)
      KAFKA_ZOOKEEPER_CONNECT: 'zookeeper:2181' # The connection information for Zookeeper, accessible at zookeeper:2181
      KAFKA_LISTENER_SECURITY_PROTOCOL_MAP: PLAINTEXT:PLAINTEXT,PLAINTEXT_INTERNAL:PLAINTEXT # Security protocols for Kafka listeners
      KAFKA_ADVERTISED_LISTENERS: PLAINTEXT://localhost:9092,PLAINTEXT_INTERNAL://broker:29092 # Advertised endpoints for clients to connect to the Kafka broker
      KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR: 1
      KAFKA_TRANSACTION_STATE_LOG_MIN_ISR: 1
      KAFKA_TRANSACTION_STATE_LOG_REPLICATION_FACTOR: 1
    ports:
      - "9092:9092" # Port to use to connect from kafka broker to local host port

```

After running Docker-compose up, we can see that the containers start to be created.



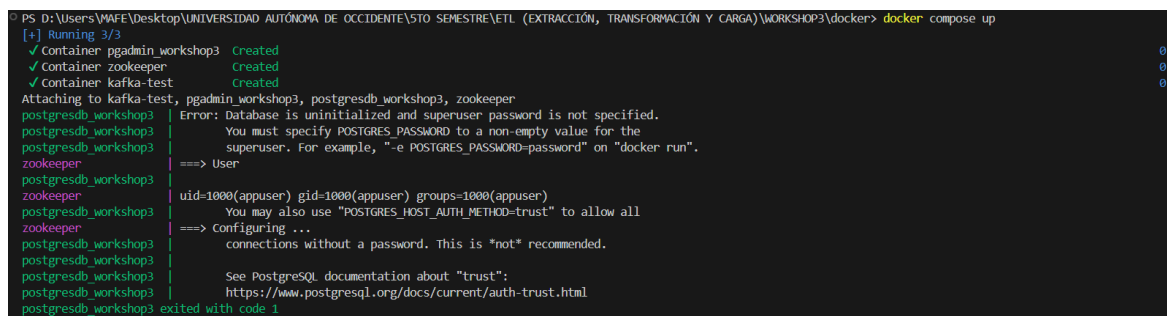
```

PS D:\Users\VAFE\Desktop\UNIVERSIDAD AUTÓNOMA DE OCCIDENTE\5TO SEMESTRE\ETL (EXTRACCIÓN, TRANSFORMACIÓN Y CARGA)\WORKSHOP3> cd docker
PS D:\Users\VAFE\Desktop\UNIVERSIDAD AUTÓNOMA DE OCCIDENTE\5TO SEMESTRE\ETL (EXTRACCIÓN, TRANSFORMACIÓN Y CARGA)\WORKSHOP3\docker> ls
Directory: D:\Users\VAFE\Desktop\UNIVERSIDAD AUTÓNOMA DE OCCIDENTE\5TO SEMESTRE\ETL (EXTRACCIÓN, TRANSFORMACIÓN Y CARGA)\WORKSHOP3\docker

Mode                LastWriteTime         Length Name
----                -
-a----             6/11/2024  1:43 p. m.           1054 docker-compose.yml

PS D:\Users\VAFE\Desktop\UNIVERSIDAD AUTÓNOMA DE OCCIDENTE\5TO SEMESTRE\ETL (EXTRACCIÓN, TRANSFORMACIÓN Y CARGA)\WORKSHOP3\docker> docker compose up
time="2024-11-06T13:45:44-05:00" level=warning msg="D:\Users\VAFE\Desktop\UNIVERSIDAD AUTÓNOMA DE OCCIDENTE\5TO SEMESTRE\ETL (EXTRACCIÓN, TRANSFORMACIÓN Y CARGA)\WORKSHOP3\docker\docker-compose.yml:
the attribute 'version' is obsolete, it will be ignored, please remove it to avoid potential confusion"
[+] Running 1/2
 ✓ Network docker_default Created                                0.1s
   Container zookeeper Creating                                  0.1s
   Error response from daemon: Conflict. The container name "/zookeeper" is already in use by container "a083474130f06a82ba493e5488de2a604c4518c165a7c81ea4dc35a292f7f6f". You have to remove (or rename) that co
ntainer to be able to reuse that name.
PS D:\Users\VAFE\Desktop\UNIVERSIDAD AUTÓNOMA DE OCCIDENTE\5TO SEMESTRE\ETL (EXTRACCIÓN, TRANSFORMACIÓN Y CARGA)\WORKSHOP3\docker>
PS D:\Users\VAFE\Desktop\UNIVERSIDAD AUTÓNOMA DE OCCIDENTE\5TO SEMESTRE\ETL (EXTRACCIÓN, TRANSFORMACIÓN Y CARGA)\WORKSHOP3\docker>
PS D:\Users\VAFE\Desktop\UNIVERSIDAD AUTÓNOMA DE OCCIDENTE\5TO SEMESTRE\ETL (EXTRACCIÓN, TRANSFORMACIÓN Y CARGA)\WORKSHOP3\docker>

```



```

PS D:\Users\VAFE\Desktop\UNIVERSIDAD AUTÓNOMA DE OCCIDENTE\5TO SEMESTRE\ETL (EXTRACCIÓN, TRANSFORMACIÓN Y CARGA)\WORKSHOP3\docker> docker compose up
[+] Running 3/3
 ✓ Container pgadmin_workshop3 Created                                0.3s
 ✓ Container zookeeper Created                                       0.3s
 ✓ Container kafka-test Created                                     0.3s
Attaching to kafka-test, pgadmin_workshop3, postgresdb_workshop3, zookeeper
postgresdb_workshop3 Error: Database is uninitialized and superuser password is not specified.
postgresdb_workshop3 You must specify POSTGRES_PASSWORD to a non-empty value for the
postgresdb_workshop3 superuser. For example, "-e POSTGRES_PASSWORD=password" on "docker run".
zookeeper ==> User
postgresdb_workshop3 uid=1000(appuser) gid=1000(appuser) groups=1000(appuser)
postgresdb_workshop3 You may also use "POSTGRES_HOST_AUTH_METHOD=trust" to allow all
postgresdb_workshop3 ==> Configuring ...
postgresdb_workshop3 connections without a password. This is *not* recommended.
postgresdb_workshop3 See PostgreSQL documentation about "trust":
postgresdb_workshop3 https://www.postgresql.org/docs/current/auth-trust.html
postgresdb_workshop3 exited with code 1

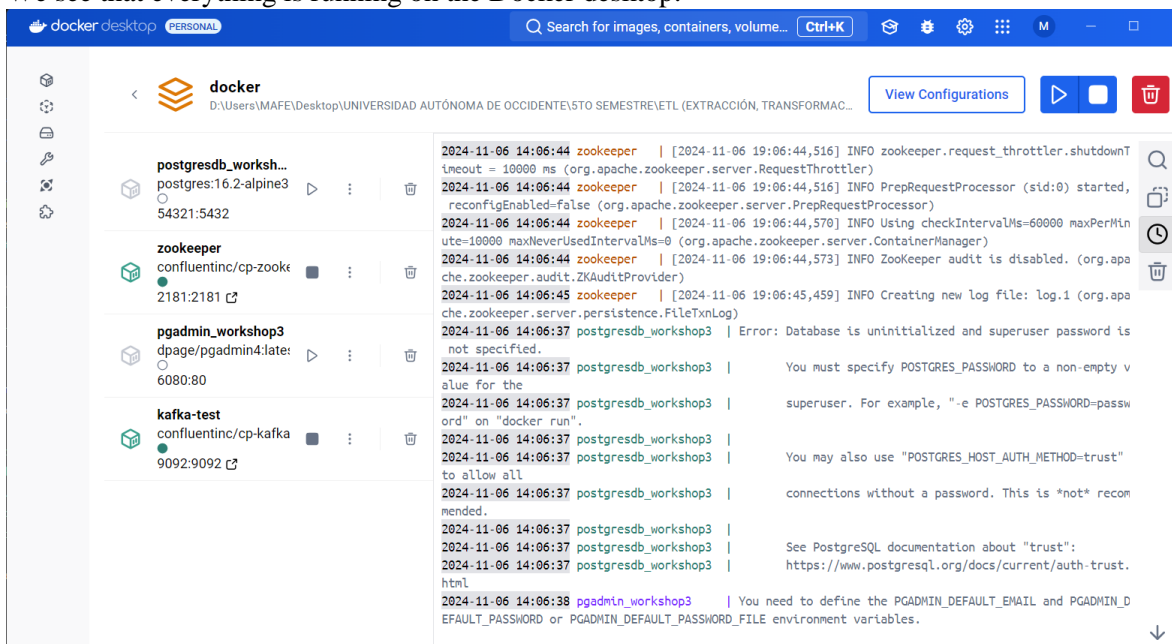
```



```
kafka-test      | ==> User
kafka-test      | uid=1000(appuser) gid=1000(appuser) groups=1000(appuser)
kafka-test      | ==> Configuring ...
pgadmin_workshop3 | You need to define the PGADMIN_DEFAULT_EMAIL and PGADMIN_DEFAULT_PASSWORD or PGADMIN_DEFAULT_PASSWORD_FILE environment variables.
kafka-test      | Running in Zookeeper mode...
pgadmin_workshop3 | exited with code 1
zookeeper        | ==> Running preflight checks ...
zookeeper        | ==> Check if /var/lib/zookeeper/data is writable ...
zookeeper        | ==> Check if /var/lib/zookeeper/log is writable ...
zookeeper        | ==> Launching ...
zookeeper        | ==> Launching zookeeper ...
kafka-test      | ==> Running preflight checks ...
kafka-test      | ==> Check if /var/lib/kafka/data is writable ...
kafka-test      | ==> Check if Zookeeper is healthy ...
zookeeper        | [2024-11-06 19:06:43,397] INFO Reading configuration from: /etc/kafka/zookeeper.properties (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
zookeeper        | [2024-11-06 19:06:43,404] INFO clientPortAddress is 0.0.0.0:2181 (org.apache.zookeeper.server.quorum.QuorumPeerConfig)

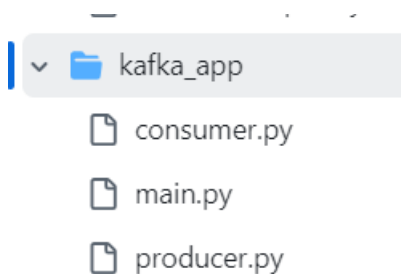
zookeeper        | [2024-11-06 19:06:43,475] INFO ServerMetrics initialized with provider org.apache.zookeeper.metrics.impl.DefaultMetricsProvider@d68ef40 (org.apache.zooke
per.server.ServerMetrics)
zookeeper        | [2024-11-06 19:06:43,488] INFO ACL digest algorithm is: SHA1 (org.apache.zookeeper.server.auth.DigestAuthenticationProvider)
zookeeper        | [2024-11-06 19:06:43,489] INFO zookeeper.DigestAuthenticationProvider.enabled = true (org.apache.zookeeper.server.auth.DigestAuthenticationProvider)
zookeeper        | [2024-11-06 19:06:43,505] INFO zookeeper.snapshot.trust.empty : false (org.apache.zookeeper.server.persistence.FileTxnSnapLog)
zookeeper        | [2024-11-06 19:06:43,548] INFO (org.apache.zookeeper.server.ZooKeeperServer)
zookeeper        | [2024-11-06 19:06:43,548] INFO (org.apache.zookeeper.server.ZooKeeperServer)
zookeeper        | [2024-11-06 19:06:43,548] INFO (org.apache.zookeeper.server.ZooKeeperServer)
zookeeper        | [2024-11-06 19:06:43,549] INFO (org.apache.zookeeper.server.ZooKeeperServer)
zookeeper        | [2024-11-06 19:06:43,549] INFO (org.apache.zookeeper.server.ZooKeeperServer)
zookeeper        | [2024-11-06 19:06:43,549] INFO (org.apache.zookeeper.server.ZooKeeperServer)
zookeeper        | [2024-11-06 19:06:43,550] INFO (org.apache.zookeeper.server.ZooKeeperServer)
zookeeper        | [2024-11-06 19:06:43,550] INFO (org.apache.zookeeper.server.ZooKeeperServer)
zookeeper        | [2024-11-06 19:06:43,550] INFO (org.apache.zookeeper.server.ZooKeeperServer)
zookeeper        | [2024-11-06 19:06:43,560] INFO Server environment:zookeeper.version=3.8.4-9316c2a7a97e1666d8f4593f34dd6fc36ecc436c, built on 2024-02-12 22:16 UTC (org.apa
che.zookeeper.server.ZooKeeperServer)
zookeeper        | [2024-11-06 19:06:43,560] INFO Server environment:host.name=8943ad836e64 (org.apache.zookeeper.server.ZooKeeperServer)
```

We see that everything is running on the Docker desktop.

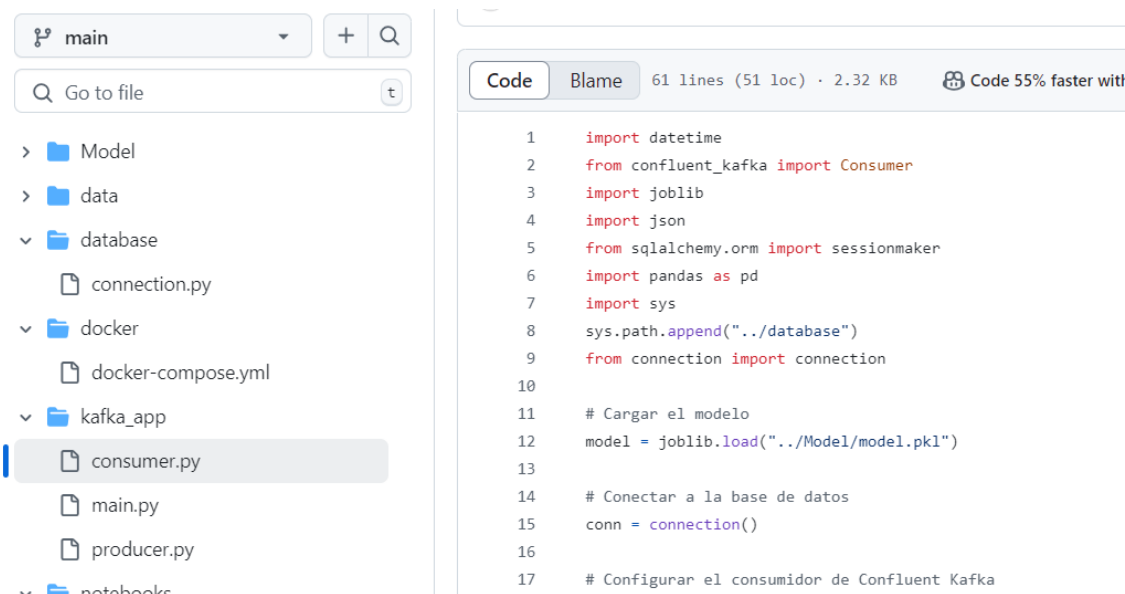


KAFKA CONSUMER

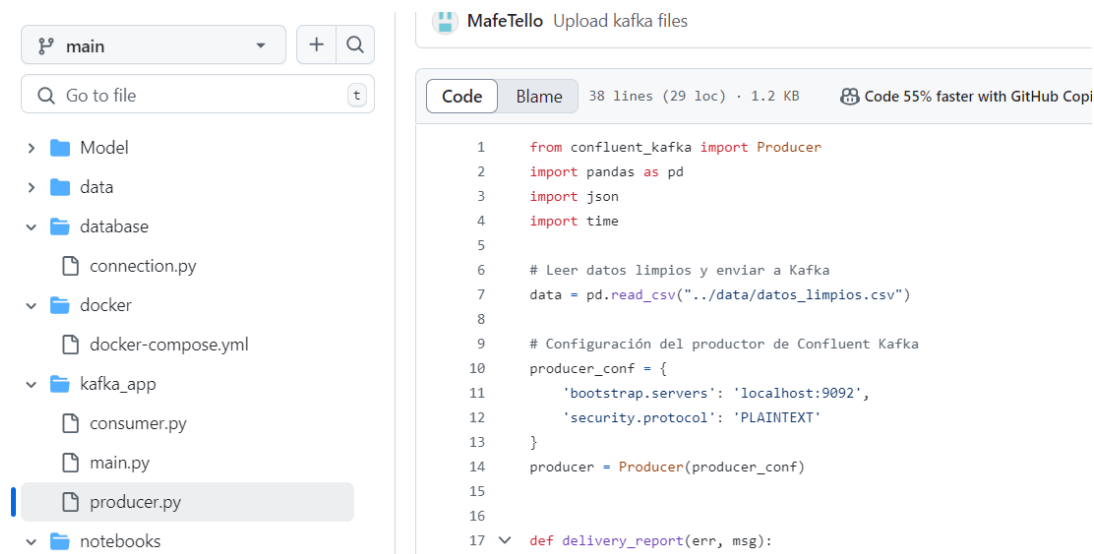
In this part I have a folder called Kafka_app, where the files “consumer.py”, “producer.py” and “main.py” are.



In consumer.py, we consume data from Kafka, use a prediction model to calculate a happiness value, and then save the results to a PostgreSQL table. Finally, the data is stored in the database.



In producer.py we read data from a CSV file, send it to a Kafka topic called `happiness` using a Kafka producer, and check the delivery of each message. For each row in the CSV, we create a JSON message with the relevant characteristics and send it to the topic.



In main.py, the modules `producer` and `consumer` are executed, which contain functions for sending data to the Kafka topic and for consuming it. When executed, it starts sending data using `producer()` and then starts receiving and processing it with `consumer()`.

Here we can see that the topic called “happiness” has already been created.

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\Users\MAFE\Desktop\UNIVERSIDAD AUTÓNOMA DE OCCIDENTE\5TO SEMESTRE\ETL (EXTRACCIÓN, TRANSFORMACIÓN Y CARGA)\WORKSHOP3> docker exec -it kafka-test kafka-topics --create --topic happiness --bootstrap-server localhost:9092
Created topic happiness.

What's next:
Try Docker Debug for seamless, persistent debugging tools in any container or image → docker debug kafka-test
Learn more at https://docs.docker.com/go/debug-cli/
PS D:\Users\MAFE\Desktop\UNIVERSIDAD AUTÓNOMA DE OCCIDENTE\5TO SEMESTRE\ETL (EXTRACCIÓN, TRANSFORMACIÓN Y CARGA)\WORKSHOP3>

```

In the following image we can see that the data is already being sent to the table created previously, this is done after executing consumer.py.

```

[2024-11-09 20:52:44.014552] - Datos cargados exitosamente en la tabla 'happiness'
[2024-11-09 20:52:45.036186] - Datos cargados exitosamente en la tabla 'happiness'
[2024-11-09 20:52:46.024302] - Datos cargados exitosamente en la tabla 'happiness'
[2024-11-09 20:52:47.034089] - Datos cargados exitosamente en la tabla 'happiness'
[2024-11-09 20:52:48.043422] - Datos cargados exitosamente en la tabla 'happiness'
[2024-11-09 20:52:49.054792] - Datos cargados exitosamente en la tabla 'happiness'
[2024-11-09 20:52:50.035647] - Datos cargados exitosamente en la tabla 'happiness'
[2024-11-09 20:52:51.025687] - Datos cargados exitosamente en la tabla 'happiness'
[2024-11-09 20:52:52.055313] - Datos cargados exitosamente en la tabla 'happiness'
[2024-11-09 20:52:53.042795] - Datos cargados exitosamente en la tabla 'happiness'
[2024-11-09 20:52:55.986200] - Datos cargados exitosamente en la tabla 'happiness'
[2024-11-09 20:52:57.002500] - Datos cargados exitosamente en la tabla 'happiness'
[2024-11-09 20:52:57.994910] - Datos cargados exitosamente en la tabla 'happiness'
[2024-11-09 20:52:58.995404] - Datos cargados exitosamente en la tabla 'happiness'
[2024-11-09 20:52:59.985991] - Datos cargados exitosamente en la tabla 'happiness'
[2024-11-09 20:53:00.994707] - Datos cargados exitosamente en la tabla 'happiness'
[2024-11-09 20:53:02.005097] - Datos cargados exitosamente en la tabla 'happiness'
[2024-11-09 20:53:03.001440] - Datos cargados exitosamente en la tabla 'happiness'
[2024-11-09 20:53:04.011810] - Datos cargados exitosamente en la tabla 'happiness'
[2024-11-09 20:53:05.004782] - Datos cargados exitosamente en la tabla 'happiness'
[2024-11-09 20:53:05.991374] - Datos cargados exitosamente en la tabla 'happiness'
[2024-11-09 20:53:06.993380] - Datos cargados exitosamente en la tabla 'happiness'
[2024-11-09 20:53:08.005014] - Datos cargados exitosamente en la tabla 'happiness'

```

In the following image we can see that the data is already being sent to the table created previously, this is done after executing consumer.py.

```

(.venv) PS D:\Users\MAFE\Desktop\UNIVERSIDAD AUTÓNOMA DE OCCIDENTE\5TO SEMESTRE\ETL (EXTRACCIÓN, TRANSFORMACIÓN Y CARGA)\WORKSHOP3\kafka_app> python .\producer.py
enviando datos

```

DATABASE

Inside the database folder I have a file called connection.py which is where a connection is established to PostgreSQL using SQLAlchemy and credentials from the `.env` file.

```

main
Go to file
Model
data
database
  connection.py
docker
  docker-compose.yml
kafka_app

MafeTello Change
Code Blame 11 lines (9 loc) · 313 Bytes Code 55% faster with GitHub Copilot
1 from sqlalchemy import create_engine
2 from decouple import config
3
4 # Specify the driver in the connection string
5 engine = create_engine(
6     f"postgresql+psycopg2://{config('DB_USER')}:{config('DB_PASSWORD')}@"
7     f"{config('DB_HOST')}:{config('DB_PORT')}/{config('DB_DB')}"
8 )
9
10 def connection():

```

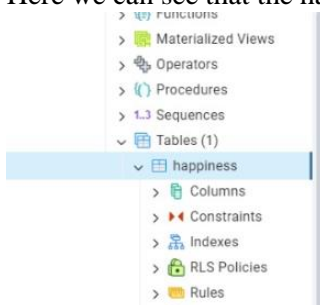
In this image we can see that pgadmin can now be opened in the browser. And we put the credentials that are in one of the Docker files.



And we can see that we actually got in.



Here we can see that the happiness table has already been successfully created.



We can see the following results when performing the query in the database.

Admin File Object Tools Help workshop3@gmail.com (internal)

Object Explorer

- public
 - Aggregates
 - Collations
 - Domains
 - FTS Configurations
 - FTS Dictionaries
 - FTS Parsers
 - FTS Templates
 - Foreign Tables
 - Functions
 - Materialized Views
 - Operators
 - Procedures
 - Sequences
 - Tables (1)
 - happiness
 - Columns
 - Constraints
 - Indexes
 - RLS Policies
 - Rules
 - Triggers
 - Trigger Functions
 - Types
 - Views
 - Subscriptions
 - Login/Group Roles

Dashboard Properties SQL Statistics Dependencies Dependents Processes workshop3/postgres@workshop3*

workshop3/postgres@workshop3

Query Query History

1 select * from happiness

Scratch Pad

Data Output Messages Notifications

	happiness_score double precision	gdp_per_capita double precision	social_support double precision	health_life_expectancy double precision	freedom double precision	perceptions_of_corruption double precision	generosity double precision	year double precision	continent_lat double precision
1	7.527	1.32548	1.36058	0.87464	0.64938	0.48357	0.34139	2015	
2	7.522	1.459	1.33095	0.88521	0.66973	0.36503	0.34699	2015	
3	7.427	1.32629	1.32261	0.90563	0.63297	0.32957	0.45811	2015	
4	7.406	1.29025	1.31826	0.88911	0.64169	0.41372	0.23351	2015	
5	7.378	1.32944	1.28017	0.89284	0.61576	0.31814	0.4761	2015	
6	7.364	1.33171	1.28907	0.91087	0.6598	0.43844	0.36262	2015	
7	7.286	1.25018	1.31967	0.90837	0.63938	0.42922	0.47501	2015	

While in the kafka files you can see that the data is being loaded one by one.

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

8, 'social_support': 1.01528, 'health_life_expectancy': 0.61826, 'freedom': 0.32818, 'perceptions_of_corruption': 0.01615, 'generosity': 0.20951, 'year': 2015.0, 'continent_africa': 0.0, 'continent_asia': 0.0, 'continent_europe': 1.0, 'continent_north_america': 0.0, 'continent_oceania': 0.0, 'continent_south_america': 0.0}	[2024-11-09 21:01:29.302553] - Datos cargados exitosamente en la tabla 'happiness'
Mensaje enviado al tópic happiness {'happiness_score': 5.878, 'gdp_per_capita': 0.75985, 'social_support': 1.30477, 'health_life_expectancy': 0.66098, 'freedom': 0.53899, 'perceptions_of_corruption': 0.08242, 'generosity': 0.3424, 'year': 2015.0, 'continent_africa': 0.0, 'continent_asia': 0.0, 'continent_europe': 0.0, 'continent_north_america': 0.0, 'continent_oceania': 0.0, 'continent_south_america': 1.0}	[2024-11-09 21:01:30.303526] - Datos cargados exitosamente en la tabla 'happiness'
Mensaje enviado al tópic happiness {'happiness_score': 5.855, 'gdp_per_capita': 1.12254, 'social_support': 1.12241, 'health_life_expectancy': 0.64368, 'freedom': 0.51649, 'perceptions_of_corruption': 0.08454, 'generosity': 0.11827, 'year': 2015.0, 'continent_africa': 0.0, 'continent_asia': 1.0, 'continent_europe': 0.0, 'continent_north_america': 0.0, 'continent_oceania': 0.0, 'continent_south_america': 0.0}	[2024-11-09 21:01:32.109092] - Datos cargados exitosamente en la tabla 'happiness'
Mensaje enviado al tópic happiness {'happiness_score': 5.848, 'gdp_per_capita': 1.18498, 'social_support': 1.27385, 'health_life_expectancy': 0.87337, 'freedom': 0.60855, 'perceptions_of_corruption': 0.03787, 'generosity': 0.25328, 'year': 2015.0, 'continent_africa': 0.0, 'continent_asia': 0.0, 'continent_europe': 1.0, 'continent_north_america': 0.0, 'continent_oceania': 0.0, 'continent_south_america': 0.0}	[2024-11-09 21:01:32.292193] - Datos cargados exitosamente en la tabla 'happiness'
	[2024-11-09 21:01:33.287918] - Datos cargados exitosamente en la tabla 'happiness'
	[2024-11-09 21:01:34.299955] - Datos cargados exitosamente en la tabla 'happiness'
	[2024-11-09 21:01:35.297621] - Datos cargados exitosamente en la tabla 'happiness'
	[2024-11-09 21:01:36.302201] - Datos cargados exitosamente en la tabla 'happiness'
	[2024-11-09 21:01:37.311760] - Datos cargados exitosamente en la tabla 'happiness'
	[2024-11-09 21:01:38.301694] - Datos cargados exitosamente en la tabla 'happiness'
	[2024-11-09 21:01:39.342629] - Datos cargados exitosamente en la tabla 'happiness'
	[2024-11-09 21:01:40.308439] - Datos cargados exitosamente en la tabla 'happiness'
	[2024-11-09 21:01:41.318683] - Datos cargados exitosamente en la tabla 'happiness'
	[2024-11-09 21:01:42.294847] - Datos cargados exitosamente en la tabla 'happiness'
	[2024-11-09 21:01:43.303336] - Datos cargados exitosamente en la tabla 'happiness'
	[2024-11-09 21:01:44.311597] - Datos cargados exitosamente en la tabla 'happiness'
	[2024-11-09 21:01:45.338420] - Datos cargados exitosamente en la tabla 'happiness'
	[2024-11-09 21:01:46.305995] - Datos cargados exitosamente en la tabla 'happiness'

We can see this new column that is added in the database table where we can see that the prediction is successfully made.

Predicted_Score double precision
7.107844266066104
7.298675572041195
7.275769072901931
7.370204379191702
7.241023639726708
7.198632143337921

Finally, we uploaded all the folders to the GitHub repository!