

Workshop-3



Santiago Gomez Castro (2226287)

Docente: Javier Alejandro Vergara Zorrilla

Universidad Autónoma de Occidente

Facultad de Ingeniería

Santiago de Cali

2024

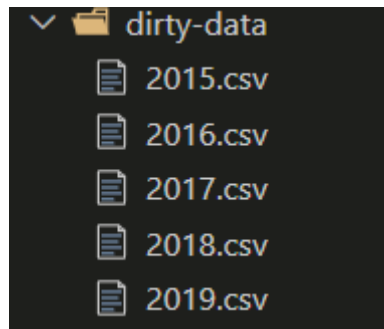
Introducción

Este proyecto busca la creación de un modelo de machine learning con las capacidades de predecir la felicidad de un país de acuerdo a los datos recibidos, usando como datos de prueba y entrenamiento 5 archivos CSVs con la información de esperanza de vida, PIB, confianza con el gobierno, ETC, de un país. Antes de iniciar con el entrenamiento del modelo primero hay que hacer revisión y limpieza de los CSVs para asegurarnos que los datos estén listos sin valores nulos o datos erróneos.

Datos sucios

Empezaremos explicando las columnas que poseen los diferentes dataset:

CSV:



- **Country:**
 - El nombre del país al que corresponden los datos en esa fila.
- **Region:**
 - La región geográfica a la que pertenece el país.
- **Happiness Rank:**
 - El rango o clasificación del país en el índice de felicidad. Un rango más bajo indica una mayor felicidad general en el país en comparación con otros países.
- **Happiness Score:**
 - El puntaje de felicidad general de un país. Este puntaje es el que se utiliza para determinar la clasificación de cada país en el índice de felicidad.
- **Standard Error:**
 - El error estándar asociado con el puntaje de felicidad, que indica la variabilidad de la estimación del puntaje de felicidad. Un error estándar más bajo significa una mayor precisión en la estimación.
- **Economy (GDP per Capita):**

- La contribución de la economía (medida por el PIB per cápita) al puntaje de felicidad de un país. Los valores más altos indican que el país tiene una economía más fuerte y próspera.
- **Family:**
 - Este indicador refleja la calidad de las relaciones familiares y el apoyo social que reciben los ciudadanos en ese país. Valores más altos indican un fuerte soporte social y familiar.
- **Health (Life Expectancy):**
 - Este indicador mide la esperanza de vida en el país, lo que tiene un impacto directo en el bienestar y la felicidad. Un valor más alto indica una mayor esperanza de vida.
- **Freedom:**
 - Refleja el nivel de libertad que perciben los ciudadanos para tomar decisiones importantes en sus vidas, como elegir un trabajo, mudarse o hacer elecciones personales. Un valor más alto indica mayor libertad.
- **Trust (Government Corruption):**
 - Indica la confianza de los ciudadanos en su gobierno y la percepción de corrupción. Un valor más alto significa que los ciudadanos confían más en su gobierno y perciben menos corrupción.
- **Generosity:**
 - Mide el nivel de generosidad y disposición de los ciudadanos a ayudar a los demás, como donar dinero o tiempo a causas benéficas. Un valor más alto indica una mayor generosidad.
- **Dystopia Residual:**
 - Este valor es un ajuste hipotético que se utiliza para comparar el país con una "distopía" (una sociedad extremadamente infeliz). Este indicador ayuda a mostrar cuán lejos está un país de una distopía en términos de felicidad.
- **Lower Confidence Interval (Intervalo de Confianza Inferior):**
 - Representa el límite inferior de un intervalo de confianza, es el valor más bajo dentro del rango en el que se espera que se encuentre la verdadera media o valor estimado, con un nivel de confianza determinado (por ejemplo, 95%). Indica que, con este nivel de confianza, el verdadero puntaje de felicidad se espera que sea mayor o igual a este valor.
- **Upper Confidence Interval (Intervalo de Confianza Superior):**
 - Representa el límite superior de un intervalo de confianza es el valor más alto dentro del rango en el que se espera que se encuentre la verdadera media o valor estimado, con un nivel de confianza dado

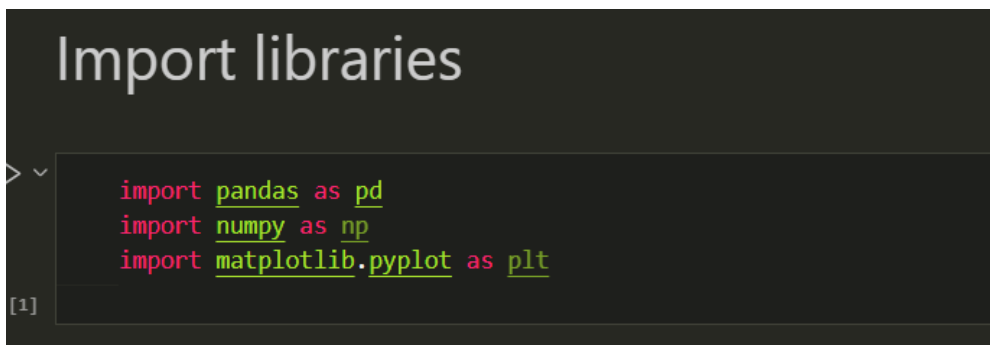
(por ejemplo, 95%). Indica que, con este nivel de confianza, se espera que el verdadero puntaje de felicidad sea menor o igual a este valor.

Herramientas usadas

- **Python:** Se usó python para la creación de scripts de subida de datos a la base de dato y Dags para el funcionamiento de Airflow
- **Jupyter:** Se emplean Notebooks de jupyter para el EDA de ambos dataset, donde se limpia, transforma la información y a su vez se crean gráficas para entender más sencillamente la información.
- **Poetry:** Ambiente de desarrollo en Python para la gestión de las librerías requeridas.
- **Git y Github:** Gestores de versión de código para guardar y compartir el proyecto.
- **SQLAlchemy:** Esta librería nos permite la conexión a la base de datos para la obtención de los datos y luego actualizarlos después de haber pasado por el EDA.
- **Pandas:** Librería para el análisis de los datos y su manipulación.
- **Dontev:** Libreria para acceder a las credenciales de la base de datos y no exponerlas en el código.
- **PostgreSQL:** Base de datos relacional que nos facilitara el guardado y gestión de los datos.
- **Apache-kafka:** Nos permite hacer streaming de datos para el envío de los datos entre el punto A al B.
- **Docker:** Contenedores Docker para Kafka y ZooKeeper

Limpieza y transformación de los datos sucios

Importaciones de las librerías



```
Import libraries

> 
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

[1]
```

Empezamos imprimiendo los datos de forma general para visualizar el contenido que posee.

	Country	Region	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedom	Trust (Government Corruption)	Generosity	Dystopia Residual
0	Switzerland	Western Europe	1	7.587	0.03411	1.39651	1.34951	0.94143	0.66557	0.41978	0.29678	2.51738
1	Iceland	Western Europe	2	7.561	0.04884	1.30232	1.40223	0.94784	0.62877	0.14145	0.43630	2.70201
2	Denmark	Western Europe	3	7.527	0.03328	1.32548	1.36058	0.87464	0.64938	0.48357	0.34139	2.49204
3	Norway	Western Europe	4	7.522	0.03880	1.45900	1.33095	0.88521	0.66973	0.36503	0.34699	2.46531
4	Canada	North America	5	7.427	0.03553	1.32629	1.32261	0.90563	0.63297	0.32957	0.45811	2.45176
...
153	Rwanda	Sub-Saharan Africa	154	3.465	0.03464	0.22208	0.77370	0.42864	0.59201	0.55191	0.22628	0.67042
154	Benin	Sub-Saharan Africa	155	3.340	0.03656	0.28665	0.35386	0.31910	0.48450	0.08010	0.18260	1.63328
155	Syria	Middle East and Northern Africa	156	3.006	0.05015	0.66320	0.47489	0.72193	0.15684	0.18906	0.47179	0.32858
156	Burundi	Sub-Saharan Africa	157	2.905	0.08658	0.01530	0.41587	0.22396	0.11850	0.10062	0.19727	1.83302
157	Togo	Sub-Saharan Africa	158	2.839	0.06727	0.20868	0.13995	0.28443	0.36453	0.10731	0.16681	1.56726

158 rows x 12 columns

Revisamos ahora el tipo de dato que cada columna posee.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 158 entries, 0 to 157
Data columns (total 12 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   Country                              158 non-null    object
1   Region                              158 non-null    object
2   Happiness Rank                      158 non-null    int64
3   Happiness Score                    158 non-null    float64
4   Standard Error                    158 non-null    float64
5   Economy (GDP per Capita)          158 non-null    float64
6   Family                            158 non-null    float64
7   Health (Life Expectancy)          158 non-null    float64
8   Freedom                          158 non-null    float64
9   Trust (Government Corruption)      158 non-null    float64
10  Generosity                        158 non-null    float64
11  Dystopia Residual                  158 non-null    float64
dtypes: float64(9), int64(1), object(2)
memory usage: 14.9+ KB
```

Observamos que los nombres de las columnas poseen mayúscula, las pasamos a minúsculas para no tener conflicto con las mayúsculas cuando se requiera de una columna en específico o si en las demás columnas de los CSVs tienen mayúsculas diferentes.

```
df2015.columns = df2015.columns.str.lower()
```

```
print(df2015.columns)
```

```
Index(['country', 'region', 'happiness rank', 'happiness score',  
      'standard error', 'economy (gdp per capita)', 'family',  
      'health (life expectancy)', 'freedom', 'trust (government corruption)',  
      'generosity', 'dystopia residual'],  
      dtype='object')
```

Pasamos a normalizar los nombres de las columnas porque en los demás CSVs tendrán nombres diferentes, entonces de esta forma será más fácil trabajar y manipularlos.

```
nuevos_nombres = {  
    'economy (gdp per capita)': 'gdp',  
    'health (life expectancy)': 'life_expectancy',  
    'trust (government corruption)': 'government_trust',  
    'dystopia residual': 'dystopia_residual',  
    'standard error': 'standard_error',  
    'happiness rank': 'happiness_rank',  
    'happiness score': 'happiness_score',  
}
```

```
df2015 = df2015.rename(columns=nuevos_nombres)
```

```
print(df2015.columns)
```

```
Index(['country', 'region', 'happiness_rank', 'happiness_score',  
      'standard_error', 'gdp', 'family', 'life_expectancy', 'freedom',  
      'government_trust', 'generosity', 'dystopia_residual'],  
      dtype='object')
```

Ahora revisamos que no hayan valores duplicados en el dataset.

```
duplicados = df2015['country'].duplicated().sum()
```

```
print("Duplicate values in column 'country':", duplicados)
```

Por último se tiene el dataset ya limpio y normalizado.

```
df2015.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 158 entries, 0 to 157  
Data columns (total 12 columns):  
#   Column                Non-Null Count  Dtype  
---  -  
0   country               158 non-null   object  
1   region                158 non-null   object  
2   happiness_rank        158 non-null   int64  
3   happiness_score       158 non-null   float64  
4   standard_error        158 non-null   float64  
5   gdp                   158 non-null   float64  
6   family                158 non-null   float64  
7   life_expectancy       158 non-null   float64  
8   freedom               158 non-null   float64  
9   government_trust      158 non-null   float64  
10  generosity             158 non-null   float64  
11  dystopia_residual     158 non-null   float64  
dtypes: float64(9), int64(1), object(2)  
memory usage: 14.9+ KB
```

Todos estos pasos se aplican en los demás CSVs hasta realizar la unión de todos y obtener el Dataset final.

```
finalDataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 782 entries, 0 to 781  
Data columns (total 15 columns):  
#   Column                Non-Null Count  Dtype  
---  ---  
0   country                782 non-null    object  
1   region                 315 non-null    object  
2   happiness_rank         782 non-null    int64  
3   happiness_score        782 non-null    float64  
4   standard_error         470 non-null    float64  
5   gdp                    782 non-null    float64  
6   family                 782 non-null    float64  
7   life_expectancy        782 non-null    float64  
8   freedom                782 non-null    float64  
9   government_trust       781 non-null    float64  
10  generosity              782 non-null    float64  
11  dystopia_residual       470 non-null    float64  
12  year                    782 non-null    int64  
13  lower_confidence_interval 312 non-null    float64  
14  upper_confidence_interval 312 non-null    float64  
dtypes: float64(11), int64(2), object(2)  
memory usage: 91.8+ KB
```

Por último guardamos el dataset en formato csv.

```
finalDataset.to_csv("../clean-data/finalDataset.csv", index=False)  
print("Dataset saved as 'finalDataset.csv'")
```

Ahora vamos a visualizar los 5 países más felices de cada año desde el 2015 hasta el 2019.

Top 5 happiest countries in the year 2015:

	country	happiness_score
1	Switzerland	7.587
2	Iceland	7.561
3	Denmark	7.527
4	Norway	7.522
5	Canada	7.427

Top 5 happiest countries in the year 2016:

	country	happiness_score
1	Denmark	7.526
2	Switzerland	7.509
3	Iceland	7.501
4	Norway	7.498
5	Finland	7.413

Top 5 happiest countries in the year 2017:

	country	happiness_score
1	Norway	7.537
2	Denmark	7.522
3	Iceland	7.504
4	Switzerland	7.494
5	Finland	7.469

Top 5 happiest countries in the year 2018:

	country	happiness_score
1	Finland	7.632
2	Norway	7.594
3	Denmark	7.555
4	Iceland	7.495
5	Switzerland	7.487

Top 5 happiest countries in the year 2019:

	country	happiness_score
1	Finland	7.769
2	Denmark	7.600
3	Norway	7.554
4	Iceland	7.494
5	Netherlands	7.488

Observamos que los países ocupan los 5 primeros lugares son países del norte de Europa tales como Noruega, Finlandia y Dinamarca, también se ve como Suiza pasa del ser al primer país más feliz en 2015 a en 2019 salir de los 5 primeros y es reemplazado por Finlandia.

Top los 5 países mas infelices de cada año.

Top 5 least happy countries in the year 2015:

	country	happiness_score
1	Togo	2.839
2	Burundi	2.905
3	Syria	3.006
4	Benin	3.340
5	Rwanda	3.465

Top 5 least happy countries in the year 2016:

	country	happiness_score
1	Burundi	2.905
2	Syria	3.069
3	Togo	3.303
4	Afghanistan	3.360
5	Benin	3.484

Top 5 least happy countries in the year 2017:

	country	happiness_score
1	Central African Republic	2.693
2	Burundi	2.905
3	Tanzania	3.349
4	Syria	3.462
5	Rwanda	3.471

Top 5 least happy countries in the year 2018:

	country	happiness_score
1	Burundi	2.905
2	Central African Republic	3.083
3	South Sudan	3.254
4	Tanzania	3.303
5	Yemen	3.355

Top 5 least happy countries in the year 2019:

	country	happiness_score
1	South Sudan	2.853
2	Central African Republic	3.083
3	Afghanistan	3.203
4	Tanzania	3.231
5	Rwanda	3.334

En este caso observamos que todos los países a excepción de Afganistán son africanos, mostrando lo infelices, esto refleja la esperanza de vida, PIB y la confianza en el gobierno que poseen estos países.

Elección de filas

Después de terminar con la limpieza y unión de los datos, empezaremos con la creación del modelo predictivo de felicidad.

Elegimos las columnas que consideramos necesarias para la creación de este modelo.

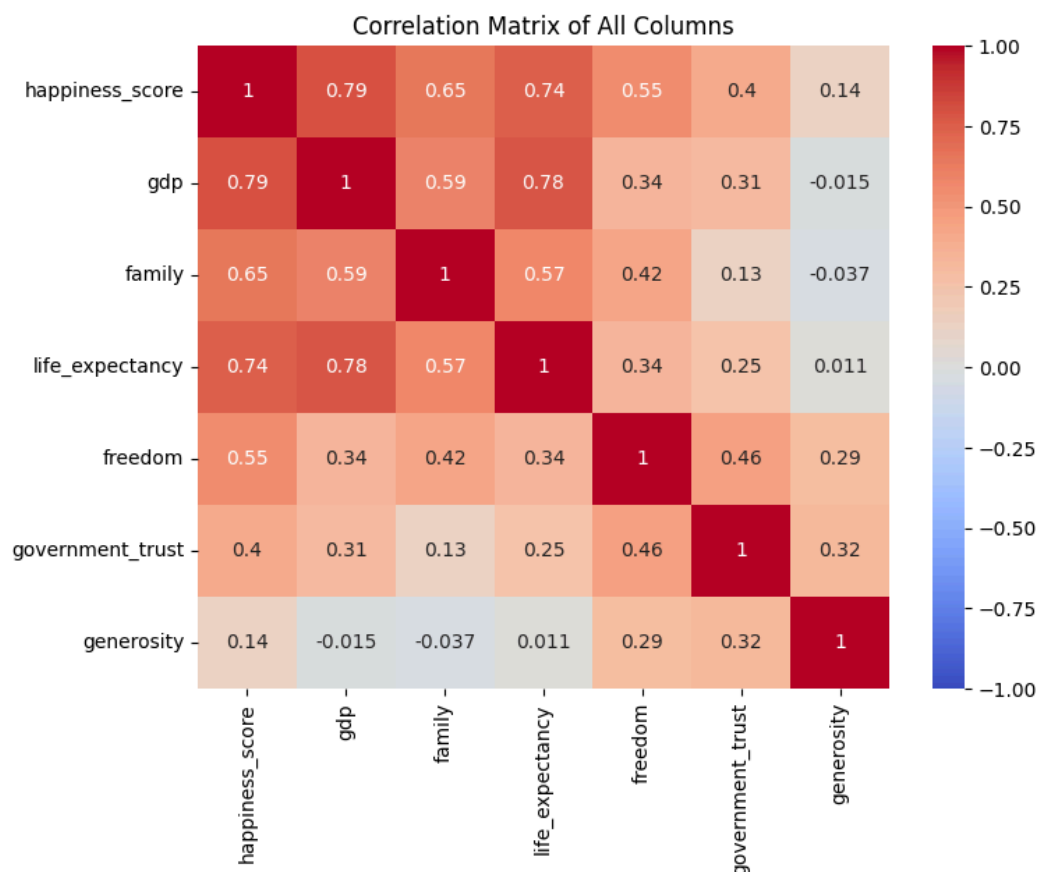
```
df = df[['happiness_score', 'gdp', 'family', 'life_expectancy', 'freedom', 'government_trust', 'generosity']]
df
```

Explicación de por qué se eligieron estas columnas:

- **Happiness Score:** Esta es la variable objetivo que se busca predecir, representando el nivel de felicidad de cada país.
- **GDP (Producto Interno Bruto):** El PIB per cápita es un indicador clave de la riqueza y el nivel de vida de los habitantes de un país. Generalmente, un mayor PIB está correlacionado con una mejor calidad de vida y, por ende, mayor felicidad.
- **Family (Apoyo Familiar):** Este factor mide el apoyo social que los individuos perciben de sus familias. La familia es una fuente importante de bienestar emocional, y se ha demostrado que un fuerte apoyo social está relacionado con niveles más altos de felicidad.
- **Life Expectancy (Esperanza de Vida):** La esperanza de vida refleja la salud general de la población. Una mayor esperanza de vida a menudo está asociada con mejores condiciones de vida y atención médica, lo que contribuye a una mayor felicidad.

- **Freedom (Libertad para Tomar Decisiones):** La libertad individual es fundamental para la satisfacción personal. Las personas que sienten que tienen la libertad de tomar decisiones sobre sus propias vidas tienden a ser más felices.
- **Government Trust (Confianza en el Gobierno):** La confianza en las instituciones gubernamentales es un factor crucial para la estabilidad social. Las sociedades donde la población confía en su gobierno suelen experimentar menos conflictos y mayores niveles de satisfacción y felicidad.
- **Generosity (Generosidad):** Este indicador refleja el nivel de donaciones y ayuda entre la población. La generosidad puede estar relacionada con el bienestar social y la felicidad, ya que contribuir al bienestar de los demás puede generar satisfacción personal.

Matriz de correlación



Algunas conclusiones que podemos sacar de esta grafica serian:

happiness_score: Tiene una fuerte correlación positiva con gdp (0.79) y life_expectancy (0.74), lo que sugiere que a medida que aumenta el PIB per cápita y la esperanza de vida, también tiende a aumentar el puntaje de felicidad.

gdp: Presenta una correlación positiva con todas las demás variables, especialmente con happiness_score (0.79) y life_expectancy (0.78), indicando que el crecimiento económico es un factor importante para la felicidad y la salud.

life_expectancy: Tiene una alta correlación con happiness_score (0.74) y gdp (0.78). Esto sugiere que vivir más tiempo y de manera saludable se asocia con mayores niveles de felicidad.

Luego de elegir los datos verificamos nuevamente que no tenga valores nulos y si los posee borrarlos directamente.

```
null_counts = df.isnull().sum()

print(null_counts)
```

happiness_score	0
gdp	0
family	0
life_expectancy	0
freedom	0
government_trust	1
generosity	0
dtype: int64	

```
df = df.dropna()
```

Ya con los datos listos completamente, importamos la librería de Sklearn donde están los modelos y métricas que vamos a usar para la creación del modelo y elección del mismo.

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, Lasso, Ridge
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error
```

- **Regresión Lineal (Linear Regression):**

La regresión lineal asume que hay una relación lineal entre las variables independientes (features) y la variable dependiente (target). El modelo ajusta una línea que minimiza la suma de los cuadrados de las diferencias entre los valores observados y los predichos.

- **Regresión Lasso (Lasso Regression):**

Lasso (Least Absolute Shrinkage and Selection Operator) es una forma de regresión lineal que incluye un término de penalización

(regularización) que es proporcional a la suma de los valores absolutos de los coeficientes. Esto ayuda a reducir el sobreajuste y también puede llevar a que algunos coeficientes sean exactamente cero, lo que puede ser útil para la selección de características.

- **Regresión Ridge (Ridge Regression):**

Al igual que Lasso, Ridge también es una técnica de regresión lineal que incluye un término de penalización, pero en este caso es proporcional a la suma de los cuadrados de los coeficientes. Esto ayuda a prevenir el sobreajuste, pero a diferencia de Lasso, no reduce los coeficientes a cero, lo que significa que no realiza selección de características.

- **Bosque Aleatorio (Random Forest):**

Random Forest es un modelo de aprendizaje automático basado en árboles de decisión. Consiste en crear múltiples árboles de decisión y promediar sus resultados para obtener una predicción más robusta. Este enfoque ayuda a reducir la varianza y mejora la precisión de las predicciones en comparación con un solo árbol de decisión.

- **Aumento de Gradiente (Gradient Boosting):**

Este es otro modelo basado en árboles de decisión que construye modelos de forma secuencial, donde cada nuevo modelo intenta corregir los errores del anterior. Utiliza un enfoque de optimización que se basa en el gradiente para ajustar el modelo a los errores de los datos anteriores, lo que permite mejorar continuamente la precisión de las predicciones.

El código devolverá el resultado de estos cálculos con las siguientes métricas:

R^2 : R^2 varía entre 0 y 1. Un R^2 de 1 indica que el modelo explica toda la variación de los datos, mientras que un R^2 de 0 indica que no explica ninguna variación.

Mean Squared Error (MSE): Un MSE más bajo indica que el modelo tiene un mejor ajuste, es decir, las predicciones están más cerca de los valores reales

Mean Absolute Error (MAE): Al igual que el MSE, un MAE más bajo indica un mejor ajuste del modelo. MAE es más robusto frente a valores atípicos en comparación con el MSE, ya que no eleva al cuadrado las diferencias.

```

===== Linear Regression =====
Mean Squared Error (MSE): 0.309824650958382
R^2: 0.7518804887905026
Mean Absolute Error (MAE): 0.43214584081745866

===== Lasso Regression =====
Mean Squared Error (MSE): 0.4879262664560984
R^2: 0.6092498567661606
Mean Absolute Error (MAE): 0.5463957355538125

===== Ridge Regression =====
Mean Squared Error (MSE): 0.30719565413260436
R^2: 0.7539858906859473
Mean Absolute Error (MAE): 0.43147215094277

===== Random Forest =====
Mean Squared Error (MSE): 0.2385961209682197
R^2: 0.8089230384735623
Mean Absolute Error (MAE): 0.3813074026961888

===== Gradient Boosting =====
Mean Squared Error (MSE): 0.268155112075604
R^2: 0.7852510601376769
Mean Absolute Error (MAE): 0.4043956541043871

```

El modelo con el mejor rendimiento sería Random Forest por tener mayor R^2 y también tener los MSE y MAE más bajos, mostrando ser el más óptimo entre los 5 modelos comparados para la predicción de la felicidad.

```

===== Linear Regression =====
Mean Squared Error (MSE): 0.29568735162348175
R^2: 0.7632021824967565
Mean Absolute Error (MAE): 0.4201016400225439

===== Lasso Regression =====
Mean Squared Error (MSE): 0.4879262664560984
R^2: 0.6092498567661606
Mean Absolute Error (MAE): 0.5463957355538125

===== Ridge Regression =====
Mean Squared Error (MSE): 0.29352374946315857
R^2: 0.7649348784226959
Mean Absolute Error (MAE): 0.4195662775388524

===== Random Forest =====
Mean Squared Error (MSE): 0.228710359940642
R^2: 0.8168399365851524
Mean Absolute Error (MAE): 0.37363140334414047

===== Gradient Boosting =====
Mean Squared Error (MSE): 0.24726001763653446
R^2: 0.8019846563924005
Mean Absolute Error (MAE): 0.392325622374462

```

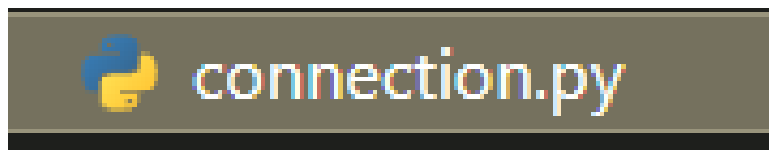
Esta es una prueba extra donde se añade la columna “year” al dataset que se usó para entrenar los modelos, aquí se visualiza una mejora de 80% pasa a un 81% con Random Forest, siendo aún el mejor modelo. No solo eso, se visualiza que Linear Regression tiene mejora del 1%, Lasso Regression no tiene ningún cambio, Ridge Regression decae un 1% y Gradient Boosting tiene una mejora significativa pasando del 78% al 80%.

Entonces guardamos el modelo en formato PKL y el dataset que se usó en CSV.

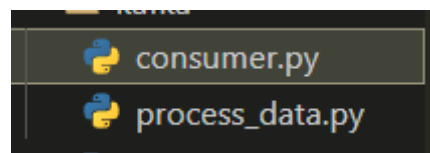
```
Dataset saved as 'modelDataset.csv'  
Random Forest model saved as 'random_forest.pkl'
```

Conexión a la base de datos:

En este script de python subimos los CSVs modelDataset y finalDataset a la base de datos para tenerlos guardado.



Kafka:



En la carpeta kafka están los siguientes scripts, estos script tienen toda la lógica consumidor y productor para la transmisión de los datos, y también en el consumer se realiza la predicción de la felicidad para luego subirla a la base de datos. Con esto se guardaría la predicción y las columnas usadas, a continuación una breve descripción de lo que hace cada script:

process_data.py: Este script coge los csv y los pasa por todo el proceso de transformación y unión para empezar con la transmisión de los datos hacia el consumer, esto se realiza en el topic llamado "happinessPredictions".

consumer.py: En este script recibe los datos transmitidos en el topic "happinessPredictions", los datos se reciben en filas y usando el archivo PKL realiza la predicción agregando la columna "predicted_happiness_score", estas filas son acumuladas en un batch de máximo 20 filas para realizar la subida de los datos a la base de datos.