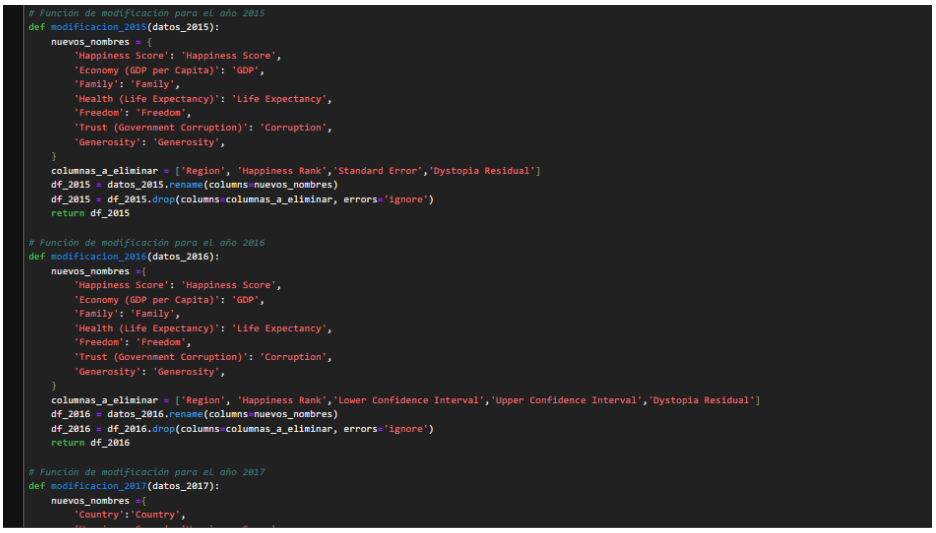
JUAN FELIPE ZAMBRANO BAEZA

WORKSHOP FINAL

1.Análisis de los datasets

El primer paso fue analizar el dataset para así identificar dentro de estos que columnas compartían entre ellos para después de esto realizarla transformaciones a cada uno de los dataset de acuerdo a los nombre que se escogieron para que cada uno tuviera el mismo nombre y las columnas y de esta forma realizar la unión de los 5 dataset

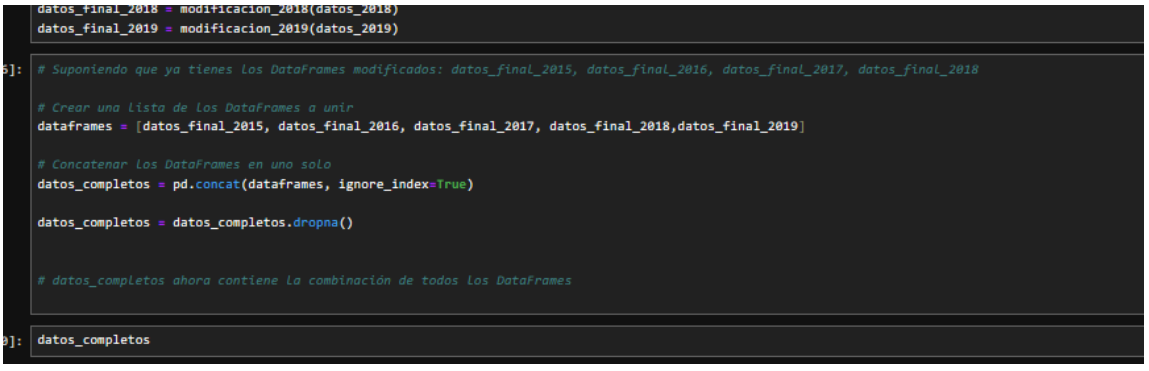
evidencia:



```
# Función de modificación para el año 2015
def modificacion_2015(datos_2015):
    nuevos_nombres = {
        'Happiness Score': 'Happiness Score',
        'Economy (GDP per Capita)': 'GDP',
        'Family': 'Family',
        'Health (Life Expectancy)': 'Life Expectancy',
        'Freedom': 'Freedom',
        'Trust (Government Corruption)': 'Corruption',
        'Generosity': 'Generosity',
    }
    columnas_a_eliminar = ['Region', 'Happiness Rank','Standard Error','Dystopia Residual']
    df_2015 = datos_2015.rename(columns=nuevos_nombres)
    df_2015 = df_2015.drop(columns=columnas_a_eliminar, errors='ignore')
    return df_2015

# Función de modificación para el año 2016
def modificacion_2016(datos_2016):
    nuevos_nombres ={
        'Happiness Score': 'Happiness Score',
        'Economy (GDP per Capita)': 'GDP',
        'Family': 'Family',
        'Health (Life Expectancy)': 'Life Expectancy',
        'Freedom': 'Freedom',
        'Trust (Government Corruption)': 'Corruption',
        'Generosity': 'Generosity',
    }
    columnas_a_eliminar = ['Region', 'Happiness Rank','Lower Confidence Interval','Upper Confidence Interval','Dystopia Residual']
    df_2016 = datos_2016.rename(columns=nuevos_nombres)
    df_2016 = df_2016.drop(columns=columnas_a_eliminar, errors='ignore')
    return df_2016

# Función de modificación para el año 2017
def modificacion_2017(datos_2017):
    nuevos_nombres ={
        'Country':'Country',
```

2.unión datasets

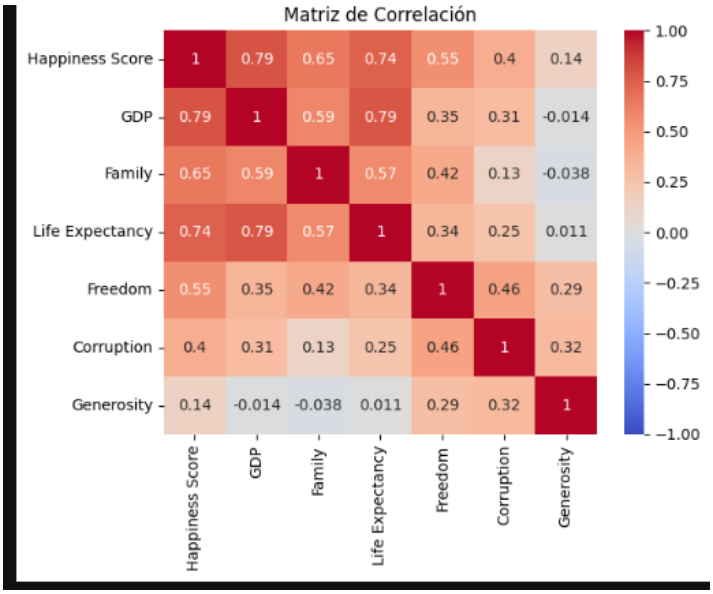
aqui se realizó la unión de los 5 datasets para crear uno solo



```
datos_final_2018 = modificacion_2018(datos_2018)
datos_final_2019 = modificacion_2019(datos_2019)

5]: # Suponiendo que ya tienes los DataFrames modificados: datos_final_2015, datos_final_2016, datos_final_2017, datos_final_2018

# Crear una lista de los DataFrames a unir
dataframes = [datos_final_2015, datos_final_2016, datos_final_2017, datos_final_2018,datos_final_2019]

# Concatenar los DataFrames en uno solo
datos_completos = pd.concat(dataframes, ignore_index=True)

datos_completos = datos_completos.dropna()


# datos_completos ahora contiene la combinación de todos los DataFrames

9]: datos_completos
```

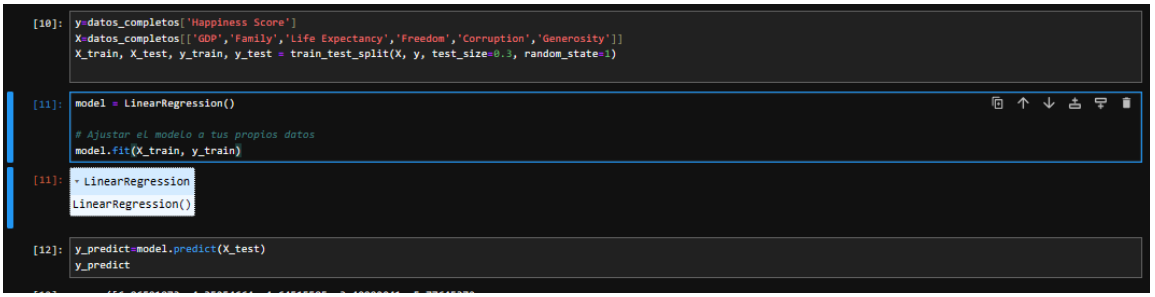
3. Creación del modelo de regresión lineal

Despues de unir los Datasets, se procede analizar el proceso de análisis de las columnas el cual consistió en analizar la correlación entre las columnas y esta se hizo apartir de una matriz de correlación



Después se utilizó un modelo de regresión lineal para predecir la variable objetivo 'Happiness Score' basándose en las características proporcionadas en el conjunto de datos. Se dividen los datos en conjuntos de entrenamiento y prueba, y se ajusta el modelo a los datos de entrenamiento

en este modelo la variable dependiente es “Hapiness score” y las variables independientes son gdp,family,Life expectancy,freedom corruption y generosity

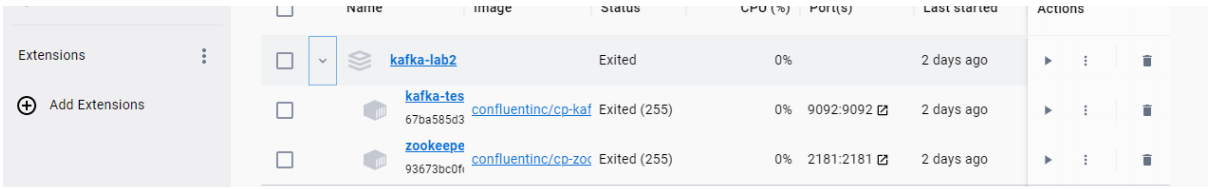


Después de realizado dicho proceso se exportó el modelo de regresión para poderlo utilizar más adelante

```
[13]: 0.7675496735951374

[14]: modelo_exportado = "model.pkl"

      # Carga el modelo desde el archivo
      model = joblib.dump(model,modelo_exportado)

[15]: modelo_file = "model.pkl"
      model = joblib.load(modelo_file)

      y_predict=model.predict(X_test)
      y_predict
```

4.creación de el contenedor de kafka y el topic

para esto se realizó la creacion de el archivo yml para así de esta manera poder crear el contenedor de kafka y así de esta manera poder crear el topic de kafka en el cual en este caso tiene de nombre kafka_prueba




```
def kafka_consumer():
    consumer = KafkaConsumer(
        'kafka_prueba',
        enable_auto_commit=True,
        group_id='my-group-1',
        value_deserializer=lambda m: loads(m.decode('utf-8')),
```

5. creación del script y envio de los datos

En este script se carga información de cinco conjuntos de datos correspondientes a cada año, aplicando funciones específicas para normalizar y estandarizar las columnas de interés. Posteriormente, los conjuntos de datos modificados se concatenan en uno solo, se eliminan los valores nulos y se dividen en conjuntos de entrenamiento y prueba para un modelo de regresión lineal. Se identifican los índices correspondientes al conjunto de prueba en el conjunto original, y se crea un nuevo DataFrame con esas filas. Finalmente, se envía este DataFrame de prueba a un sistema Kafka para su procesamiento adicional.

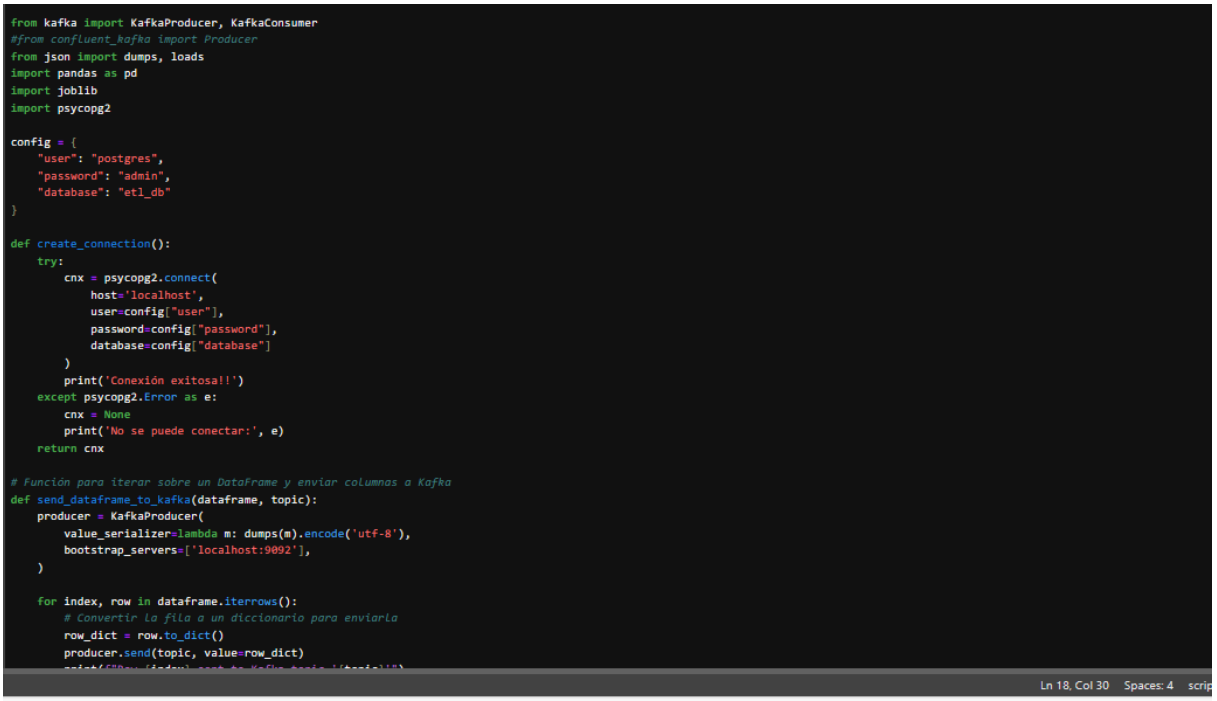
```python
import pandas as pd
from script import send_dataframe_to_kafka
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
datos_2015=pd.read_csv('2015.csv')
datos_2016=pd.read_csv('file:///C:/Users/Admin/Downloads/2016.csv')
datos_2017=pd.read_csv('file:///C:/Users/Admin/Downloads/2017.csv')
datos_2018=pd.read_csv('file:///C:/Users/Admin/Downloads/2018.csv')
datos_2019=pd.read_csv('file:///C:/Users/Admin/Downloads/2019.csv')

# Función de modificación para el año 2015
def modificacion_2015(datos_2015):
    nuevos_nombres = {
        'Happiness Score': 'Happiness Score',
        'Economy (GDP per Capita)': 'GDP',
        'Family': 'Family',
        'Health (Life Expectancy)': 'Life Expectancy',
        'Freedom': 'Freedom',
        'Trust (Government Corruption)': 'Corruption',
        'Generosity': 'Generosity',
    }
    columnas_a_eliminar = ['Region', 'Happiness Rank','Standard Error','Dystopia Residual']
    df_2015 = datos_2015.rename(columns=nuevos_nombres)
    df_2015 = df_2015.drop(columns=columnas_a_eliminar, errors='ignore')
    return df_2015

# Función de modificación para el año 2016
def modificacion_2016(datos_2016):
    nuevos_nombres ={
        'Happiness Score': 'Happiness Score',
        'Economy (GDP per Capita)': 'GDP',
        'Family': 'Family',
        'Health (Life Expectancy)': 'Life Expectancy',
        'Freedom': 'Freedom',
        'Trust (Government Corruption)': 'Corruption',
        'Generosity': 'Generosity',
    }
    columnas_a_eliminar = ['Region', 'Happiness Rank','Lower Confidence Interval','Upper Confidence Interval','Dystopia Residual']
    df_2016 = datos_2016.rename(columns=nuevos_nombres)
```

```python
olumnas_a_eliminar = ['Overall rank']
f_2019 = datos_2019.rename(columns=nuevos_nombres)
f_2019 = df_2019.drop(columns=columnas_a_eliminar, errors='ignore')

eturn df_2019

oniendo que tienes DataFrames correspondientes a cada año (datos_2015, datos_2016, datos_2017, datos_2018)
icar las modificaciones a los DataFrames de cada año

_final_2015 = modificacion_2015(datos_2015)
_final_2016 = modificacion_2016(datos_2016)
_final_2017 = modificacion_2017(datos_2017)
_final_2018 = modificacion_2018(datos_2018)
_final_2019 = modificacion_2019(datos_2019)


rames = [datos_final_2015, datos_final_2016, datos_final_2017, datos_final_2018,datos_final_2019]

catenar los DataFrames en uno solo
_completos = pd.concat(dataframes, ignore_index=True)

_completos = datos_completos.dropna()
os_completos['Happiness Score']
os_completos[['GDP','Family','Life Expectancy','Freedom','Corruption','Generosity']]
in, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1)


es_X_test = X.index[X.isin(X_test.to_dict(orient='list')).all(axis=1)]

ar un nuevo DataFrame con las filas de X_test en datos completos
_completos_test = datos_completos.loc[indices_X_test]


dataframe_to_kafka(datos_completos_test,'kafka_prueba')
```

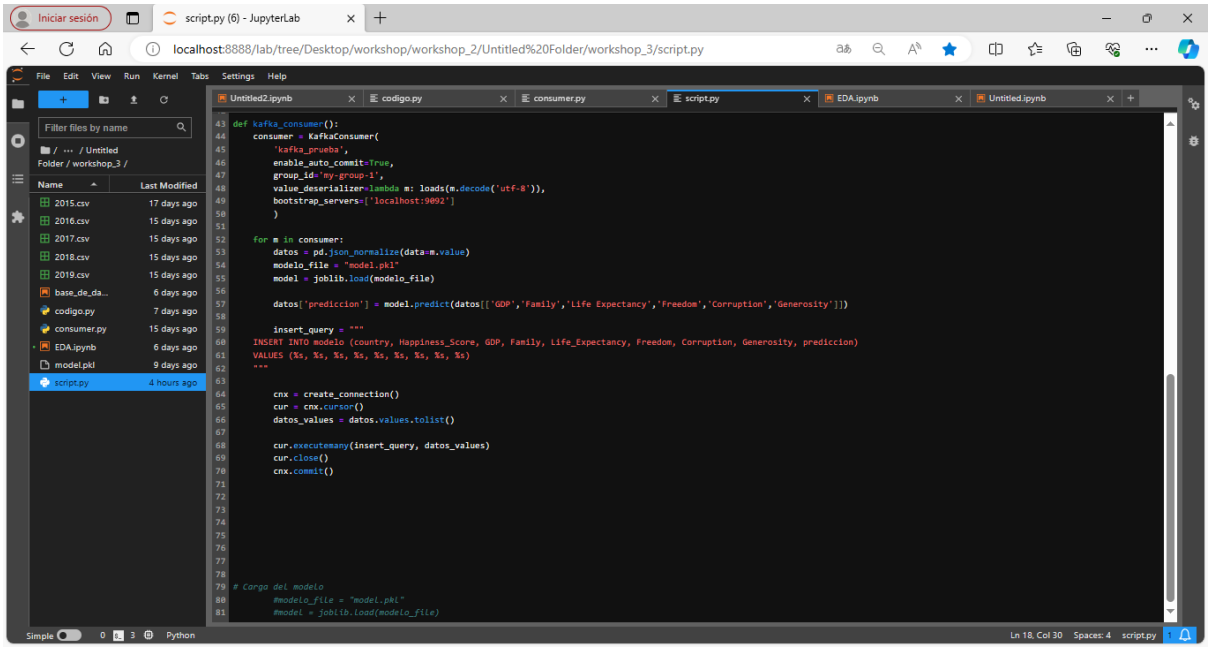
6.recepcion de los datos y carga a la base de datos

Primero, establece una conexión a la base de datos utilizando los parámetros de configuración proporcionados. Luego, define una función send_dataframe_to_kafka que toma un DataFrame de pandas y envía cada fila como un diccionario serializado a un tópico de Kafka especificado. Además, hay una función kafka_consumer que

consume mensajes desde un tópico de Kafka llamado 'kafka_prueba', deserializa los datos, normaliza el DataFrame utilizando pandas y realiza una predicción utilizando un modelo previamente entrenado. Posteriormente, inserta los resultados, incluida la predicción, en una tabla de la base de datos PostgreSQL llamada 'modelo'.



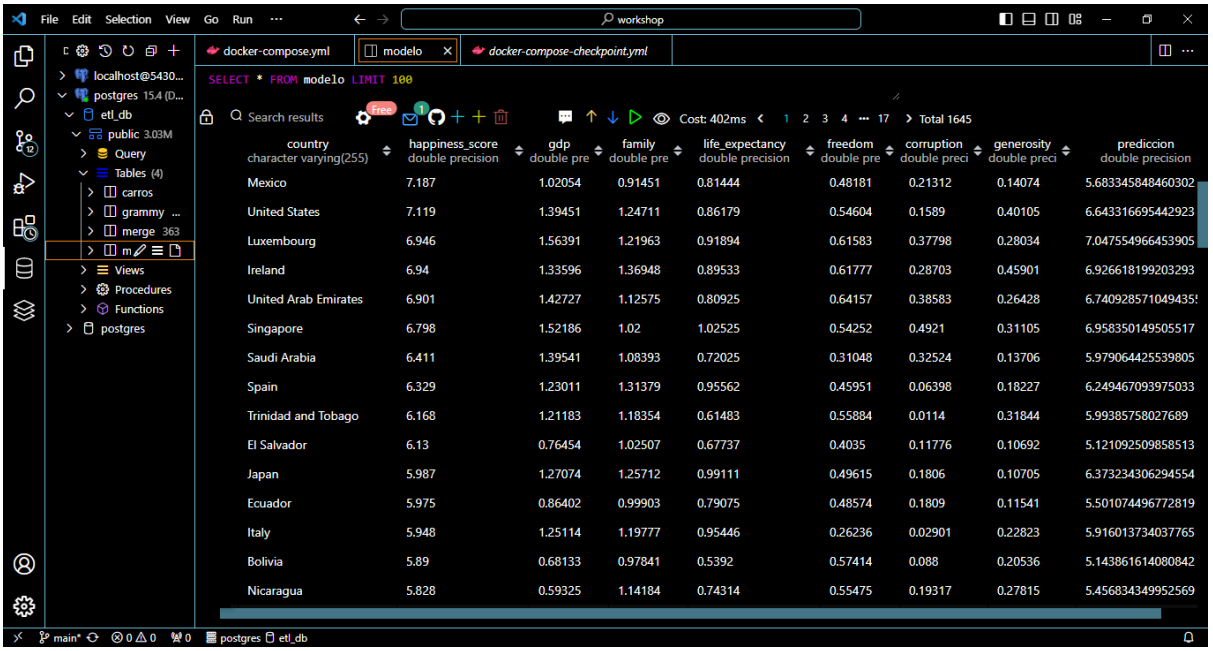
```python
from kafka import KafkaProducer, KafkaConsumer
#from confluent_kafka import Producer
from json import dumps, loads
import pandas as pd
import joblib
import psycopg2

config = {
    "user": "postgres",
    "password": "admin",
    "database": "etl_db"
}

def create_connection():
    try:
        cnx = psycopg2.connect(
            host='localhost',
            user=config["user"],
            password=config["password"],
            database=config["database"]
        )
        print('Conexión exitosa!!')
    except psycopg2.Error as e:
        cnx = None
        print('No se puede conectar:', e)
    return cnx

# Función para iterar sobre un DataFrame y enviar columnas a Kafka
def send_dataframe_to_kafka(dataframe, topic):
    producer = KafkaProducer(
        value_serializer=lambda m: dumps(m).encode('utf-8'),
        bootstrap_servers=['localhost:9092'],
    )

    for index, row in dataframe.iterrows():
        # Convertir la fila a un diccionario para enviarla
        row_dict = row.to_dict()
        producer.send(topic, value=row_dict)
```



```python
def kafka_consumer():
    consumer = KafkaConsumer(
        'kafka_prueba',
        enable_auto_commit=True,
        group_id='my-group-1',
        value_deserializer=lambda m: loads(m.decode('utf-8')),
        bootstrap_servers=['localhost:9092']
        )

    for m in consumer:
        datos = pd.json_normalize(data=m.value)
        modelo_file = "model.pkl"
        model = joblib.load(modelo_file)

        datos['prediccion'] = model.predict(datos[['GDP','Family','Life Expectancy','Freedom','Corruption','Generosity']])

        insert_query = """
    INSERT INTO modelo (country, Happiness_Score, GDP, Family, Life_Expectancy, Freedom, Corruption, Generosity, prediccion)
    VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s)
    """

        cnx = create_connection()
        cur = cnx.cursor()
        datos_values = datos.values.tolist()

        cur.executemany(insert_query, datos_values)
        cur.close()
        cnx.commit()

# Carga del modelo
        #modelo_file = "model.pkl"
        #model = joblib.load(modelo_file)
```

7.datos insertados en la base de datos

aquí se evidencia los datos ingresados dentro de la base de datos con su columna predicción



R CUADRADO:

La función traer_tabla establece una conexión con la base de datos, recupera todos los datos de la tabla 'modelo', los almacena en un DataFrame de pandas y retorna el resultado. El código posterior elimina duplicados en el DataFrame resultante y realiza la impresión del DataFrame. Además, calcula y muestra el coeficiente de determinación (R cuadrado) entre las columnas 'happiness_score' y 'prediccion', proporcionando una evaluación de la precisión de un modelo de regresión. Cabe destacar que este código presupone la existencia de una tabla 'modelo' en la base de datos con las columnas mencionadas.

```
def traer_tabla():
    try:
        conexion = create_connection()
        cursor = conexion.cursor()


        # Realiza una consulta SQL para obtener todos los datos de la tabla
        query = "SELECT * FROM modelo"
        cursor.execute(query)

        # Obtén todos los resultados como una lista de tuplas
        data = cursor.fetchall()
        #print(data)

        column_names = [desc[0] for desc in cursor.description]

        # Crea un DataFrame de pandas con los datos y nombres de columnas
        df = pd.DataFrame(data, columns=column_names)
        return df
    except psycopg2.Error as e:
        print("Error al crear la tabla:", e)

    finally:
        if conexion:
            conexion.close()



# Llama a la función para obtener los datos desde la base de datos
base_datos= traer_tabla()
base_datos=base_datos.drop_duplicates()

print(base_datos)
# Supongamos que 'y_true' son los valores reales y 'y_pred' son las predicciones
real = base_datos['happiness_score']
prediccion = base_datos['prediccion']

r_squared = r2_score(real, prediccion)
```

el resultado del r cuadrado fue de 0.7675496735951374