

DengAI predicción de la propagación de enfermedades



Adrián Yared Armas de la Nuez

Contenido

1. Objetivo.....	2
2. Actividad.....	3
2.1 Imports.....	3
2.4.5 Comando.....	3
2.1 Dataset.....	4
2.2.2 Descarga del dataset desde github.....	4
2.2.3 Ejecución.....	4
2.2.2 Descomprimir.....	4
2.2.3 Verificar la descarga y descompresión.....	5
2.2.4 Ejecución.....	5
2.3 Limpieza de datos.....	5
2.3.1 Carga de dengue_features_test.csv.....	5
2.3.2 Fusionar funciones y etiquetas.....	5
2.4 Preprocesamiento.....	5
2.4.1 Codificar ciudad.....	5
2.4.2 Extraer mes del conjunto week_start_date.....	6
2.4.3 características cíclicas para el mes y la semana del año.....	6
2.4.4 Eliminación de columnas innecesarias.....	6
2.4.5 Rellenado de valores (mediana).....	6
2.4.6 División de train x e y.....	7
2.4.7 Set de validación.....	7
2.4.8 Selección de funciones.....	7
2.4.8.1 Usando un metodo no gráfico.....	7
2.4.8.1.1 Código.....	7
2.4.8.1.2 Ejecución.....	7
2.4.8.2 Usando un metodo gráfico.....	8
2.4.8.2.1 Código.....	8
2.4.8.2.2 Ejecución.....	8
2.4.8.3 Matriz de correlación.....	8
2.4.8.3.1 Código.....	8
2.4.8.3.2 Ejecución.....	9
2.5 Elegir y entrenar el modelo.....	9
2.5.1 Comparación de modelos.....	9
2.5.1.1 Código.....	9
2.5.2 Validación cruzada y evaluación del modelo.....	10
2.5.2.1 Código.....	10

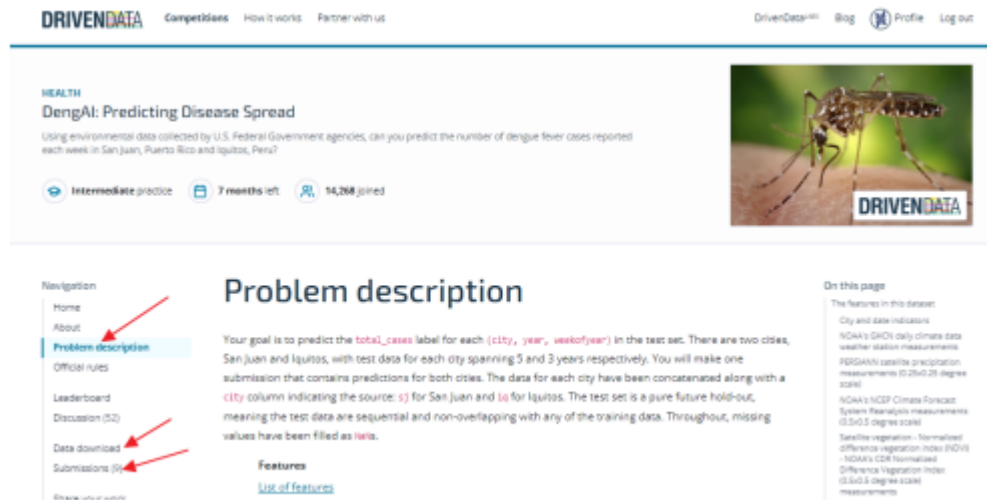
2.5.2.2 Ejecución.....	10
2.5.3 Comparación gráfica.....	10
2.5.3.1 Código.....	10
2.5.3.2 Ejecución.....	11
2.5.4 Entrenamiento del modelo.....	11
2.5.4.1 Ejecución.....	11
2.5.4.2 Ejecución.....	11
2.6 Evaluar el modelo (MAE).....	12
2.6.1 No gráficamente.....	12
2.6.1.1 Código.....	12
2.6.1.2 Ejecución.....	12
2.6.1 Gráficamente.....	12
2.6.1.1 Código.....	12
2.6.1.2 Ejecución.....	13
2.7 Entrenamiento final.....	13
2.7.1 Fit.....	13
2.7.1.1 Código.....	13
2.7.1.2 Ejecución.....	13
2.7.2 Preparación de los datos de test.....	14
2.7.2.1 Código.....	14
2.8 Descarga de csv.....	14
2.7.1 Formato correcto.....	14
2.7.1.1 Código.....	14
2.7.2 Añado la ciudad.....	14
2.7.2.1 Código.....	14
2.7.3 Orden.....	15
2.7.3.1 Código.....	15
2.7.4 Guardar el csv.....	15
2.7.4.1 Código.....	15
2.7.5 Descarga del csv.....	15
2.7.5.1 Código.....	15
3. Resultado de la competición.....	15
4. Bibliografía.....	16
5. Github y Colab.....	16

1. Objetivo

El objeto de esta actividad es participar en la competición de ofrecida de la web de DrivenData denominada: DengAI: Predicting Disease Spread. Para ello accederemos a la siguiente web y nos crearemos un usuario:

Título: DengAI: Predicting Disease Spread

Url: <https://www.drivendata.org/competitions/44/dengai-predicting-disease-spread/>



The screenshot shows the DrivenData website for the 'DengAI: Predicting Disease Spread' competition. The header includes the DrivenData logo and navigation links like 'Competitions', 'How it works', and 'Partner with us'. The main content area features a large image of a mosquito and the competition title. Below the title, it states: 'Using environmental data collected by U.S. Federal Government agencies, can you predict the number of dengue fever cases reported each week in San Juan, Puerto Rico and Iquitos, Peru?'. It also indicates 'Intermediate problem', '7 months left', and '14,268 joined'. A sidebar on the left contains navigation links: 'Home', 'About', 'Problem description', 'Official rules', 'Leaderboard', 'Discussion (52)', 'Data download', and 'Submissions (0)'. The 'Problem description' section is highlighted, showing the goal: 'Your goal is to predict the total_cases label for each (city, year, weekofyear) in the test set. There are two cities, San Juan and Iquitos, with test data for each city spanning 5 and 3 years respectively. You will make one submission that contains predictions for both cities. The data for each city have been concatenated along with a city column indicating the source: s for San Juan and i for Iquitos. The test set is a pure future hold-out, meaning the test data are sequential and non-overlapping with any of the training data. Throughout, missing values have been filled as na.' A 'Features' section lists various data sources like NOAA's GPCP, PDSATs, and NDVI. A 'On this page' section lists the features in the dataset.

La actividad consiste en subir a dicha web un fichero csv con la estimación que hayamos obtenido al aplicar los modelos que consideres oportunos. Podrás observar que se pueden realizar hasta un máximo de 3 subidas diarias y la propia web realizará una valoración de tu solución. (La valoración se realiza utilizando el MAE como criterio de valoración de calidad)

Hay que realizar todo el proceso completo de importación del dataset, ajuste de características, selección de características, entrenamiento y selección de un modelo para entrenarlo, predicción y subida del fichero para que la web lo valore y clasifique en la competición.

Consideraciones a tener en cuenta:

- Es necesario realizar pruebas con al menos tres modelos diferentes considerando que han de utilizarse como mínimo dos modelos de árboles y otro modelo que consideres oportuno
- Realizar pruebas de hiperparametrización con las dos técnicas explicadas: GridSearch y Random Search.

Una vez hayas realizado las diferentes pruebas, has de generar un documento pdf donde: Expliques con detalle tu solución. Ha de contener capturas de tu posicionamiento en la competición con la clasificación que has obtenido en la competición con los diferentes SUBMITs realizados. Sería muy interesante que expliques las mejoras o no, que has obtenido con las diferentes pruebas.

PYTHON VS SQL

Operation	Python	SQL
Create/Load Data	<code>df = pd.read_csv('file.csv')</code>	<code>CREATE TABLE table_name (column1 datatype, column2 datatype, ...);</code>
View Data	<code>df.head()</code>	<code>SELECT * FROM table_name LIMIT 5;</code>
Get Dimensions	<code>df.shape</code>	<code>SELECT COUNT(*) FROM table_name;</code>
Show Info	<code>df.info()</code>	<code>DESCRIBE table_name; or SHOW COLUMNS FROM table_name;</code>
Summary Statistics	<code>df.describe()</code>	<code>SELECT AVG(column), MIN(column), MAX(column), COUNT(*) FROM table_name;</code>
Select Columns	<code>df[['col1', 'col2']]</code>	<code>SELECT col1, col2 FROM table_name;</code>
Filter Rows	<code>df[df['column'] > value] or df.query('column > value')</code>	<code>SELECT * FROM table_name WHERE column > value;</code>
Select Rows by Index	<code>df.iloc[row_index]</code>	<code>SELECT * FROM table_name WHERE id = row_index;</code>
Sort Data	<code>df.sort_values('column')</code>	<code>SELECT * FROM table_name ORDER BY column ASC/DESC;</code>
Group and Aggregate	<code>df.groupby('column').agg()</code>	<code>SELECT column, COUNT(*) FROM table_name GROUP BY column;</code>
Join Tables	<code>pd.merge(df1, df2, on='key')</code>	<code>SELECT a.col1, b.col2 FROM table1 a JOIN table2 b ON a.key = b.key;</code>

Fuente: https://www.linkedin.com/feed/update/urn:li:activity:7290306350243274752/?utm_source=share&utm_medium=member_desktop

Título: Modelos de machine learning: Guía básica para principiantes

Url: <https://planetachatbot.com/modelos-de-machine-learning-guia-basica-paraprincipiantes/>

2. Actividad

2.1 Imports

2.4.5 Comando

```
import pandas as pd
import zipfile
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split, cross_val_score, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import Ridge
from sklearn.metrics import mean_absolute_error
```



DengAI predicción de la propagación de enfermedades

```
from google.colab import files
from sklearn.feature_selection import SelectKBest, f_regression
from sklearn.ensemble import RandomForestRegressor,
GradientBoostingRegressor
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
```

2.1 Dataset

2.2.2 Descarga del dataset desde github

```
# Download dataset from github
!wget -O DengAI_dataset.zip
"https://github.com/AdrianYArmas/IaBigData/raw/main/SNS/3%20%20-%20Algo
ritmos%20y%20herramientas%20para%20el%20aprendizaje%20supervisado%20/3.
6%20DengAI%20predicci%C3%B3n%20de%20la%20propagaci%C3%B3n%20de%20enferm
edades/Resources/DengAI%20dataset.zip"
```

2.2.3 Ejecución

```
--2025-02-15 16:14:35-- https://github.com/AdrianYArmas/IaBigData/raw/main/SNS
Resolving github.com (github.com)... 140.82.112.3
Connecting to github.com (github.com)|140.82.112.3|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://raw.githubusercontent.com/AdrianYArmas/IaBigData/main/SNS/3%2
--2025-02-15 16:14:36-- https://raw.githubusercontent.com/AdrianYArmas/IaBigDa
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.108.
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.108
HTTP request sent, awaiting response... 200 OK
Length: 119655 (117K) [application/zip]
Saving to: 'DengAI_dataset.zip'

DengAI_dataset.zip 100%[=====>] 116.85K  --.-KB/s   in 0.02s

2025-02-15 16:14:36 (7.35 MB/s) - 'DengAI_dataset.zip' saved [119655/119655]
```

2.2.2 Descomprimir

```
with zipfile.ZipFile("DengAI_dataset.zip", 'r') as zip_ref:
    zip_ref.extractall("DengAI_dataset")
```



2.2.3 Verificar la descarga y descompresión

```
os.listdir("DengAI_dataset")
```

2.2.4 Ejecución

```
['DengAI dataset']
```

2.3 Limpieza de datos

2.3.1 Carga de dengue_features_test.csv

dengue_features_test.csv tiene como objetivo de este conjunto de datos es mostrar todos los datos recopilados, por lo que elegiré datos relevantes para hacer la predicción. (En este caso son columnas numéricas, no fechas ni cadenas)

```
# Data load
path = '/content/DengAI_dataset/DengAI_dataset/'
train_features = pd.read_csv(path+'dengue_features_train.csv')
train_labels = pd.read_csv(path+'dengue_labels_train.csv')
test_features = pd.read_csv(path+'dengue_features_test.csv')
```

2.3.2 Fusionar funciones y etiquetas

Fusioné etiquetas para garantizar que las funciones y las etiquetas correspondientes estén alineadas correctamente para el entrenamiento. Combina los datos en un único DataFrame para la entrada del modelo.

```
train = train_features.merge(train_labels, on=['city', 'year',
'weekofyear'])
```

2.4 Preprocesamiento

2.4.1 Codificar ciudad

Esta línea codifica la columna de la ciudad asignando los valores de cadena ('sj' y 'iq') a valores numéricos (0 y 1) para facilitar el procesamiento del modelo. Se aplica la misma codificación a los DataFrames de entrenamiento y de prueba.

```
# Encode city (sj -> 0, iq -> 1)
```

```
train['city'] = train['city'].map({'sj': 0, 'iq': 1})
test = test_features.copy()
test['city'] = test['city'].map({'sj': 0, 'iq': 1})
```

2.4.2 Extraer mes del conjunto week_start_date

Agrega la información del mes extraída como una nueva columna en los DataFrames de tren y de prueba.

```
train['month'] = pd.to_datetime(train['week_start_date']).dt.month
test['month'] = pd.to_datetime(test['week_start_date']).dt.month
```

2.4.3 características cíclicas para el mes y la semana del año

Características cíclicas al convertir el mes y la semana del año en transformaciones de seno y coseno.

Estas nuevas columnas (mes_sin, mes_cos, semanadel año_sin, semanadel año_cos) se agregan a los DataFrames de entrenamiento y de prueba para capturar patrones periódicos (Se usan transformaciones seno y coseno para representar meses y semanas de forma cíclica, añadiendo estas columnas a los DataFrames para capturar patrones temporales.)

```
# Create cyclical features for month and weekofyear
for df in [train, test]:
    df['month_sin'] = np.sin(2 * np.pi * df['month'] / 12)
    df['month_cos'] = np.cos(2 * np.pi * df['month'] / 12)
    df['weekofyear_sin'] = np.sin(2 * np.pi * df['weekofyear'] / 52)
    df['weekofyear_cos'] = np.cos(2 * np.pi * df['weekofyear'] / 52)
```

2.4.4 Eliminación de columnas innecesarias

```
# Drop unnecessary columns
columns_to_drop = ['week_start_date', 'year', 'month', 'weekofyear']
train = train.drop(columns=columns_to_drop)
test = test.drop(columns=columns_to_drop)
```

2.4.5 Rellenado de valores (mediana)

Rellena valores nulos con la mediana para no tener valores tan extremos.

```
# Fill missing values
for col in train.columns:
    if train[col].isnull().any():
        # Fill with median grouped by city
```



```
train[col] = train.groupby('city')[col].transform(lambda x:
x.fillna(x.median()))

for col in test.columns:
    if test[col].isnull().any():
        test[col] = test.groupby('city')[col].transform(lambda x:
x.fillna(x.median()))
```

2.4.6 División de train x e y

```
# Split into features and target
X = train.drop(columns=['total_cases'])
y = train['total_cases']
```

2.4.7 Set de validación

```
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2,
random_state=42)
```

2.4.8 Selección de funciones

2.4.8.1 Usando un metodo no gráfico

2.4.8.1.1 Código

```
selector = SelectFromModel(XGBRegressor(objective='reg:absoluteerror',
n_estimators=1000, learning_rate=0.05, random_state=42))
selector.fit(X_train, y_train)
selected_features = X_train.columns[(selector.get_support())]
print(f"Selected features: {selected_features}")
```

2.4.8.1.2 Ejecución

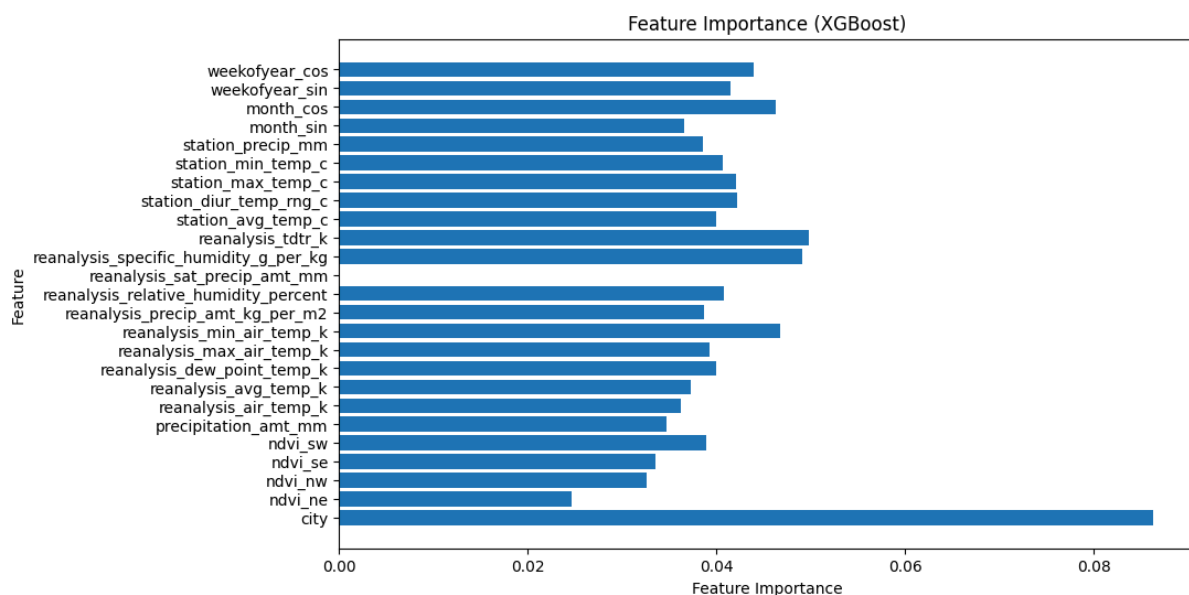
```
Selected features: Index(['city', 'reanalysis_min_air_temp_k',
'reanalysis_relative_humidity_percent',
'reanalysis_specific_humidity_g_per_kg', 'reanalysis_tdtr_k',
'station_diur_temp_rng_c', 'station_max_temp_c', 'station_min_temp_c',
'month_cos', 'weekofyear_sin', 'weekofyear_cos'],
dtype='object')
```

2.4.8.2 Usando un metodo gráfico

2.4.8.2.1 Código

```
model = XGBRegressor(objective='reg:absoluteerror', n_estimators=1000,
learning_rate=0.05, random_state=42)
model.fit(X_train, y_train)
plt.figure(figsize=(10, 6))
plt.barh(X_train.columns, model.feature_importances_)
plt.xlabel('Feature Importance')
plt.ylabel('Feature')
plt.title('Feature Importance (XGBoost)')
plt.show()
```

2.4.8.2.2 Ejecución



2.4.8.3 Matriz de correlación

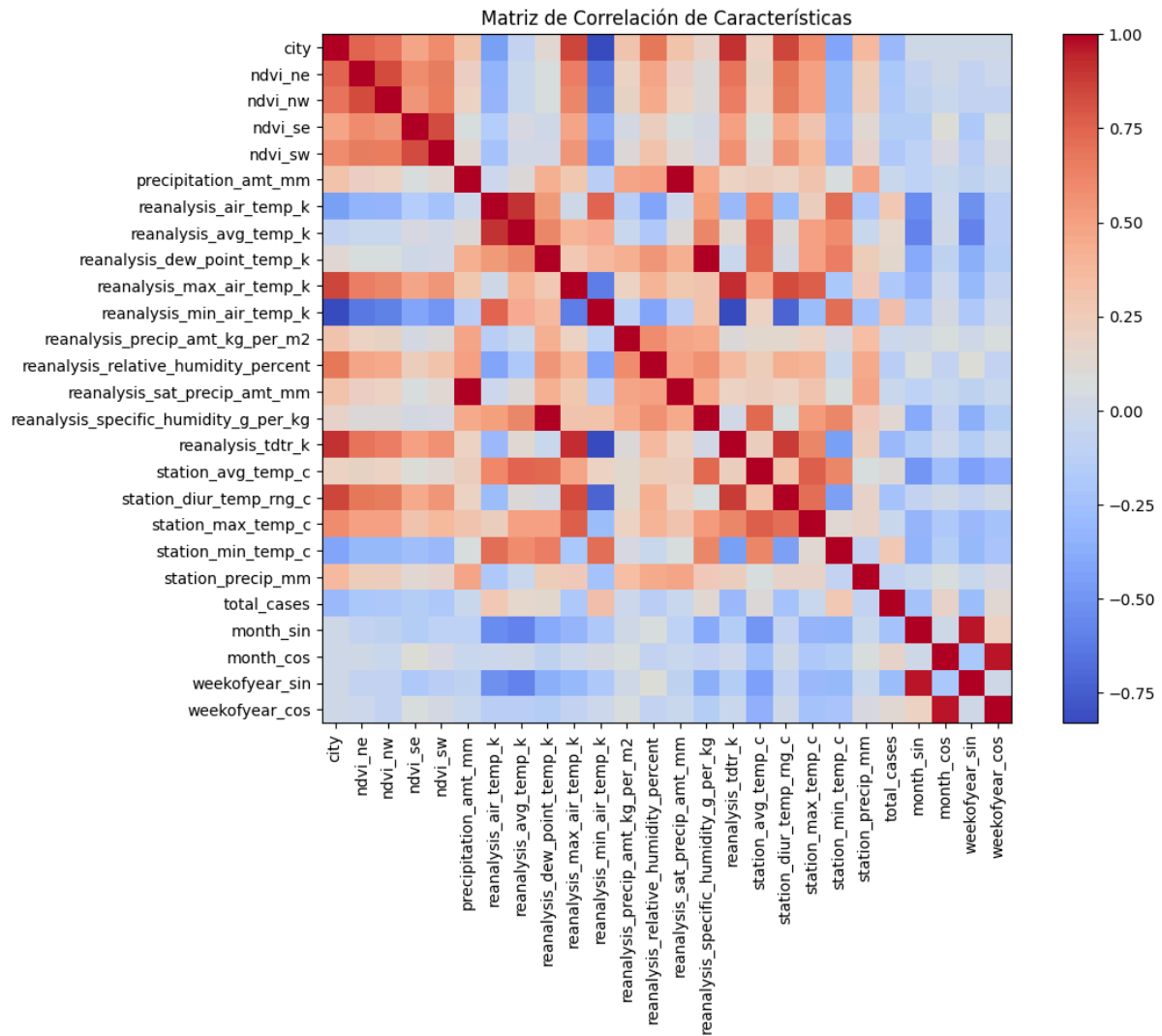
2.4.8.3.1 Código

```
correlation_matrix = train.corr()

# Heat map correlation matrix
plt.figure(figsize=(12, 8))
plt.imshow(correlation_matrix, cmap='coolwarm', interpolation='none')
plt.colorbar()
plt.xticks(np.arange(len(correlation_matrix.columns)),
correlation_matrix.columns, rotation=90)
```

```
plt.yticks(np.arange(len(correlation_matrix.columns)),
correlation_matrix.columns)
plt.title('Matriz de Correlación de Características')
plt.show()
```

2.4.8.3.2 Ejecución



2.5 Elegir y entrenar el modelo

2.5.1 Comparación de modelos

2.5.1.1 Código

He añadido XGBRegressor porque en pruebas anteriores daba mejores resultados que otros modelos tales como randomForest u otros de regresión lineal.

```
models = {  
    'Naive Bayes': GaussianNB(),  
    'KNN': KNeighborsRegressor(),  
    'XGBRegressor': XGBRegressor(objective='reg:absoluteerror',  
n_estimators=1000, learning_rate=0.05, random_state=42)  
}
```

2.5.2 Validación cruzada y evaluación del modelo

2.5.2.1 Código

```
mae_scores = {}  
for model_name, model in models.items():  
    cv_scores = cross_val_score(model, X_train[selected_features],  
y_train, cv=5, scoring='neg_mean_absolute_error')  
    mae_scores[model_name] = -cv_scores.mean()  
    print(f"{model_name} - Cross-validated MAE: {-cv_scores.mean()}")
```

2.5.2.2 Ejecución

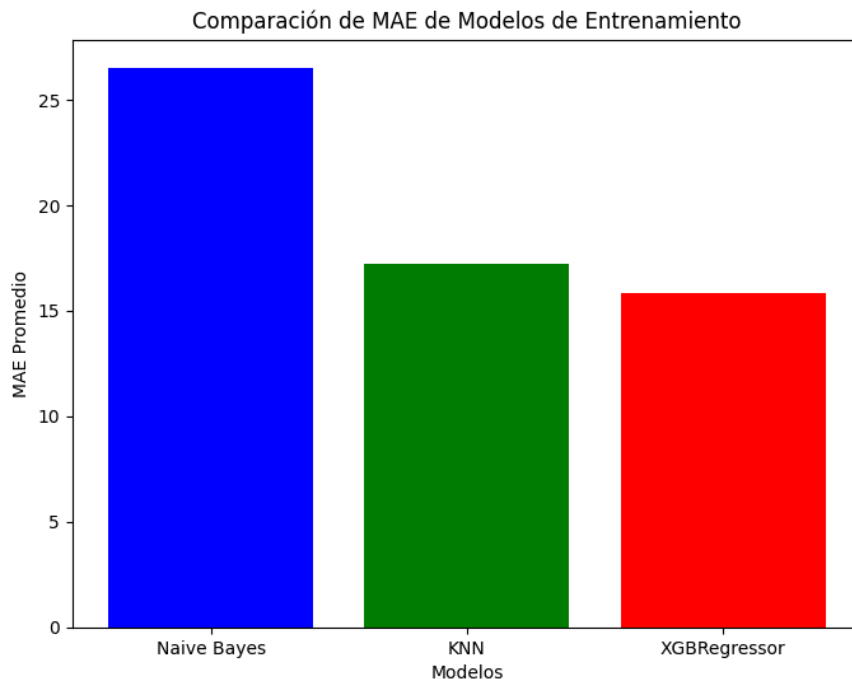
```
Naive Bayes - Cross-validated MAE: 26.523394257806718  
KNN - Cross-validated MAE: 17.212598786443685  
XGBRegressor - Cross-validated MAE: 15.816251182556153
```

2.5.3 Comparación gráfica

2.5.3.1 Código

```
# Graficar comparación de MAE de los modelos  
plt.figure(figsize=(8, 6))  
plt.bar(mae_scores.keys(), mae_scores.values(), color=['blue', 'green',  
'red'])  
plt.xlabel('Modelos')  
plt.ylabel('MAE Promedio')  
plt.title('Comparación de MAE de Modelos de Entrenamiento')  
plt.show()
```

2.5.3.2 Ejecución



2.5.4 Entrenamiento del modelo

Entrena en mejor resultado de los modelos.

2.5.4.1 Ejecución

```
best_model = XGBRegressor(objective='reg:absoluteerror',  
n_estimators=1000, learning_rate=0.05, random_state=42)  
  
best_model.fit(X_train, y_train)
```

2.5.4.2 Ejecución

XGBRegressor

```
XGBRegressor(base_score=None, booster=None, callbacks=None,  
             colsample_bylevel=None, colsample_bynode=None,  
             colsample_bytree=None, device=None, early_stopping_rounds=None,  
             enable_categorical=False, eval_metric=None, feature_types=None,  
             gamma=None, grow_policy=None, importance_type=None,  
             interaction_constraints=None, learning_rate=0.05, max_bin=None,  
             max_cat_threshold=None, max_cat_to_onehot=None,  
             max_delta_step=None, max_depth=None, max_leaves=None,  
             min_child_weight=None, missing=nan, monotone_constraints=None,  
             multi_strategy=None, n_estimators=1000, n_jobs=None,  
             num_parallel_tree=None, objective='reg:absoluteerror', ...)
```

2.6 Evaluar el modelo (MAE)

2.6.1 No gráficamente

2.6.1.1 Código

```
# Evaluate on validation set
y_pred = best_model.predict(X_val)
mae = mean_absolute_error(y_val, y_pred)
print(f'MAE on Validation Set: {mae}')
```

2.6.1.2 Ejecución

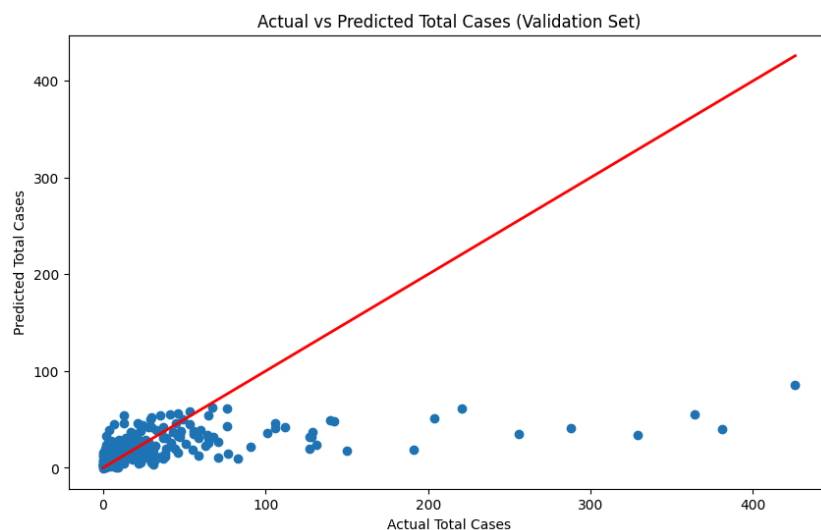
```
MAE on Validation Set: 21.01974868774414
```

2.6.1 Gráficamente

2.6.1.1 Código

```
# Visualization: Predicted vs Actual (Validation Set)
plt.figure(figsize=(10, 6))
plt.scatter(y_val, y_pred)
plt.plot([min(y_val), max(y_val)], [min(y_val), max(y_val)],
color='red', lw=2)
plt.xlabel('Actual Total Cases')
plt.ylabel('Predicted Total Cases')
plt.title('Actual vs Predicted Total Cases (Validation Set)')
plt.show()
```

2.6.1.2 Ejecución



Como podemos observar en la gráfica, se tiene una tendencia correcta pese a tener algunos outliers que según he podido observar son valores que no cumplen todas las características correspondientes al tipo de dato o guardados en algunas columnas.

2.7 Entrenamiento final

Entrenamiento usando todos los datos

2.7.1 Fit

2.7.1.1 Código

```
# Train final model on all data
best_model.fit(X, y)
```

2.7.1.2 Ejecución

```
XGBRegressor
XGBRegressor(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=0.05, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              multi_strategy=None, n_estimators=1000, n_jobs=None,
              num_parallel_tree=None, objective='reg:absoluteerror', ...)
```

2.7.2 Preparación de los datos de test

2.7.1.1 Código

El código usa un modelo entrenado (best_model) para predecir los casos en los datos de prueba (test_X). Luego, redondea las predicciones al entero más cercano para obtener valores realistas.

```
# Prepare test data (no total_cases in test_features)
test_X = test.copy() # No need to drop 'total_cases' since it doesn't
                     # exist in test data it was a temporal data

# Predict cases and round to nearest integer
predicted_cases = best_model.predict(test_X)
predicted_cases = np.round(predicted_cases).astype(int)
```

2.8 Descarga de csv

2.7.1 Formato correcto

2.7.1.1 Código

```
submission = test_features[['year', 'weekofyear']].copy()  
submission['total_cases'] = predicted_cases
```

2.7.2 Añado la ciudad

Aquí, asumimos que la predicción debe utilizar la ciudad codificada (0 o 1) según lo predicho por el modelo.

Usaremos el mapa inverso (0 -> 'sj', 1 -> 'iq') para la ciudad.

Puede utilizar una lógica para predecir la ciudad basándose en alguna regla, por ejemplo, a partir de otros datos o según los resultados del modelo.

Para simplificar, usaremos la misma ciudad para todas las muestras de prueba, ya que el modelo predice el total de casos para una ciudad específica.

2.7.2.1 Código

```
# Add the predicted city to the submission  
submission.insert(0, 'city', predicted_cities.map({0: 'sj', 1: 'iq'}))  
# Mapping city codes back to 'sj' and 'iq'
```

2.7.3 Orden

Añade el orden de las columnas

2.7.3.1 Código

```
submission = submission[['city', 'year', 'weekofyear', 'total_cases']]
```

2.7.4 Guardar el csv

Guarda los datos actualizados en el csv



2.7.4.1 Código

```
# Save to CSV with the correct format

submission.to_csv('submission.csv', index=False)
```

2.7.5 Descarga del csv

2.7.5.1 Código

```
# Download the file

files.download('submission.csv')
```

3. Resultado de la competición

Tras subir el archivo csv, estos son los resultados obtenidos.

New submission

Woohoo, your submission was successful! Your submission score is

25.6202

Post

Done

Best score	Current rank	Submissions used
25.6202	<u>#1677</u>	3 of 3

3. Conclusiones

Mi conclusión acerca de esta actividad y mi procedimiento de evolución en la misma es la siguiente, pese a la importancia de elección de modelo, entrenamiento y el conjunto de herramientas entorno al modelo descrito, la mayor carga de importancia y avance fue durante la búsqueda de mejoras a la hora de realizar la limpieza de datos, por lo que a partir



DengAI predicción de la propagación de enfermedades

de ahora comenzaré a preocuparme algo más en la limpieza y preprocesamiento de datos antes que en solo el uso del modelo o la mejor técnica.

4. Bibliografía

Para el uso de la gráfica de importancia de las características use:

https://www.w3schools.com/python/matplotlib_pyplot.asp

Y además utilicé la siguiente guía de XGBoost:

<https://machinelearningmastery.com/xgboost-for-regression/>

5. Github y Colab

