3_SolarGeneration_Pipeline

November 30, 2024

En este notebook se crea el pipeline necesario para poder guardar el modelo y utilizarlo posteriormente con la API que despliegue un endpoint para realizar las predicciones del modelo.

1 Carga de Datos

[1]:
```
import pandas as pd
import warnings
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.model_selection import RandomizedSearchCV

from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.metrics import mean_absolute_error, median_absolute_error

import joblib
import lightgbm as lgb

warnings.filterwarnings("ignore")
```

Cargamos los datos y los visualizamos

[2]:
```
df = pd.read_csv("Generation_data.csv")
df.head()
```

[2]:
```
   MODULE_TEMP  Amb_Temp  WIND_Speed  IRR (W/m2)  DC Current in Amps  \
0      18.7675  17.85190    47.60506    6.388252                0.60
1      18.6150  18.59573    64.26684   12.776500                0.66
2      18.9200  18.59573    85.68912   17.035340                4.74
3      18.9200  18.59573    83.30886   25.553010                8.18
4      19.0725  18.59573    57.12608   36.200090               26.66

   AC Ir in Amps  AC Iy in Amps  AC Ib in Amps  AC Power in Watts
0            8.6            8.6            8.7               3233
1            9.6            9.7           10.0               4504
2           11.9           12.0           12.4               6614
3           14.8           14.7           14.7               8971
4           18.6           18.4           18.5              12071
```

[3]:
```python
# Mantener solo las columnas relevantes
df = df[['MODULE_TEMP', 'Amb_Temp', 'IRR (W/m2)', 'AC Power in Watts']]
df.head()
```

[3]:

	MODULE_TEMP	Amb_Temp	IRR (W/m2)	AC Power in Watts
0	18.7675	17.85190	6.388252	3233
1	18.6150	18.59573	12.776500	4504
2	18.9200	18.59573	17.035340	6614
3	18.9200	18.59573	25.553010	8971
4	19.0725	18.59573	36.200090	12071

2 Separar las variables en X y Y

[4]:
```python
# Separar las características (X) y la variable objetivo (y)
X = df[["MODULE_TEMP", "Amb_Temp", "IRR (W/m2)"]]
y = df["AC Power in Watts"]
```

3 Dividir el dataset en entrenamiento y test

[5]:
```python
# Dividir los datos en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)
```

[6]:
```python
# Detectar si hay GPU disponible
import torch

if torch.cuda.is_available():
  device = torch.device("cuda")
  print("GPU está disponible. Usando GPU:", torch.cuda.get_device_name(0))
else:
  device = torch.device("cpu")
  print("GPU no está disponible. Usando CPU.")
```

```
GPU está disponible. Usando GPU: NVIDIA GeForce RTX 3060
```

4 Guardar el Pipeline con el mejor modelo

El mejor modelo en nuestro caso fue el modelo de LightGBM. Aquí están las métricas de los modelos entrenados actualizadas:

- **Linear Regression**:
  - Mean Squared Error: 159747971.16967878
  - R^2 Score: 0.9808629271950605
  - Mean Absolute Error: 7473.24538237479
  - Median Absolute Error: 4958.310957427253

- **Polynomial Regression**:
  - Mean Squared Error: 140500997.19574532
  - R^2 Score: 0.9831686262253331
  - Mean Absolute Error: 6821.709353959054
  - Median Absolute Error: 4211.522203715169
- **Random Forest**:
  - Mean Squared Error: 163281125.82455277
  - R^2 Score: 0.9804396715044476
  - Mean Absolute Error: 7449.578253928051
  - Median Absolute Error: 4566.7236507936905
- **Best Random Forest**:
  - Mean Squared Error: 135551253.0213555
  - R^2 Score: 0.9837615828302749
  - Mean Absolute Error: 6606.933817275343
  - Median Absolute Error: 4125.318189967948
- **XGBoost**:
  - Mean Squared Error: 137778087.35706922
  - R^2 Score: 0.9834948182106018
  - Mean Absolute Error: 6579.549506271723
  - Median Absolute Error: 4019.5625
- **LightGBM**:
  - Mean Squared Error: **133961829.8498358**
  - R^2 Score: **0.983951988421836**
  - Mean Absolute Error: **6521.984914138249**
  - Median Absolute Error: **4032.6299075853167**
- **SVM**:
  - Mean Squared Error: **6096543796.479938**
  - R^2 Score: **0.2696620705885764**
  - Mean Absolute Error: **64262.76685718178**
  - Median Absolute Error: **58646.782756845016**

El modelo de LightGBM tiene el menor Mean Squared Error (**133961829.8498358**) y un alto R^2 Score (**0.983951988421836**), lo que indica que es el mejor modelo para nuestro caso.

4.1 Crear el Pipeline y Entrenar el Modelo

En este bloque de código, vamos a crear un pipeline y entrenar un modelo utilizando LightGBM, ya que previamente hemos determinado que este algoritmo ofrece el mejor rendimiento. Además, vamos a experimentar con diferentes técnicas de escalamiento de variables y realizar una búsqueda de hiperparámetros para optimizar el modelo.

[7]:
```python
from sklearn.preprocessing import MinMaxScaler, StandardScaler, RobustScaler

# Definir el pipeline con diferentes scalers
scalers = [MinMaxScaler(), StandardScaler(), RobustScaler()]
best_score = float('inf')
best_pipeline = None
best_scaler = None
```

```python
# Definir un rango más amplio y detallado de hiperparámetros
param_grid = {
  'model__num_leaves': [31, 50, 70, 100, 150],
  'model__learning_rate': [0.01, 0.03, 0.05, 0.07, 0.1],
  'model__n_estimators': [100, 200, 300, 400, 500],
  'model__max_depth': [-1, 10, 20, 30, 40],
  'model__min_child_samples': [10, 20, 30, 40, 50]
}

for scaler in scalers:
  pipeline = Pipeline(
    [
      ("scaler", scaler),
      ("model", lgb.LGBMRegressor(device='gpu', random_state=42))
    ]
  )

  # Configurar RandomizedSearchCV
  random_search = RandomizedSearchCV(pipeline, param_distributions=param_grid, cv=3, scoring='neg_mean_squared_error', verbose=1, n_jobs=-1, n_iter=50)

  # Entrenar el pipeline con búsqueda de hiperparámetros
  random_search.fit(X_train, y_train)
  if random_search.best_score_ < best_score:
    best_score = random_search.best_score_
    best_pipeline = random_search.best_estimator_
    best_scaler = scaler
    joblib.dump(best_pipeline, "best_model_pipeline.pkl")
    print(f"New best model saved with score: {best_score} and scaler: {scaler}")

# Obtener el mejor modelo
best_pipeline = joblib.load("best_model_pipeline.pkl")

# Hacer predicciones con el mejor modelo
y_pred_pipeline = best_pipeline.predict(X_test)

# Evaluar el mejor modelo
mse_pipeline = mean_squared_error(y_test, y_pred_pipeline)
r2_pipeline = r2_score(y_test, y_pred_pipeline)
mae_pipeline = mean_absolute_error(y_test, y_pred_pipeline)
medae_pipeline = median_absolute_error(y_test, y_pred_pipeline)

print(f"Best Scaler: {best_scaler}")
print(f"Best Parameters: {random_search.best_params_}")
print(f"Mean Squared Error (Pipeline): {mse_pipeline}")
print(f"R^2 Score (Pipeline): {r2_pipeline}")
```

```python
print(f"Mean Absolute Error (Pipeline): {mae_pipeline}")
print(f"Median Absolute Error (Pipeline): {medae_pipeline}")
```

```
Fitting 3 folds for each of 50 candidates, totalling 150 fits
[LightGBM] [Info] This is the GPU trainer!!
[LightGBM] [Info] Total Bins 544
[LightGBM] [Info] Number of data points in the train set: 95092, number of used
features: 3
[LightGBM] [Info] Using GPU Device: NVIDIA GeForce RTX 3060, Vendor: NVIDIA
Corporation
[LightGBM] [Info] Compiling OpenCL Kernel with 256 bins…
[LightGBM] [Info] GPU programs have been built
[LightGBM] [Info] Size of histogram bin entry: 8
[LightGBM] [Info] 3 dense feature groups (0.36 MB) transferred to GPU in
0.001518 secs. 0 sparse feature groups
[LightGBM] [Info] Start training from score 127979.374890
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
New best model saved with score: -146263011.82586992 and scaler: MinMaxScaler()
Fitting 3 folds for each of 50 candidates, totalling 150 fits
[LightGBM] [Info] This is the GPU trainer!!
[LightGBM] [Info] Total Bins 545
[LightGBM] [Info] Number of data points in the train set: 95092, number of used
features: 3
[LightGBM] [Info] Using GPU Device: NVIDIA GeForce RTX 3060, Vendor: NVIDIA
Corporation
[LightGBM] [Info] Compiling OpenCL Kernel with 256 bins…
[LightGBM] [Info] GPU programs have been built
[LightGBM] [Info] Size of histogram bin entry: 8
[LightGBM] [Info] 3 dense feature groups (0.36 MB) transferred to GPU in
0.000862 secs. 0 sparse feature groups
[LightGBM] [Info] Start training from score 127979.374890
Fitting 3 folds for each of 50 candidates, totalling 150 fits
[LightGBM] [Info] This is the GPU trainer!!
[LightGBM] [Info] Total Bins 544
[LightGBM] [Info] Number of data points in the train set: 95092, number of used
features: 3
[LightGBM] [Info] Using GPU Device: NVIDIA GeForce RTX 3060, Vendor: NVIDIA
Corporation
[LightGBM] [Info] Compiling OpenCL Kernel with 256 bins…
[LightGBM] [Info] GPU programs have been built
[LightGBM] [Info] Size of histogram bin entry: 8
```

```
[LightGBM] [Info] 3 dense feature groups (0.36 MB) transferred to GPU in
0.001103 secs. 0 sparse feature groups
[LightGBM] [Info] Start training from score 127979.374890
Best Scaler: MinMaxScaler()
Best Parameters: {'model__num_leaves': 31, 'model__n_estimators': 200,
'model__min_child_samples': 50, 'model__max_depth': 40, 'model__learning_rate':
0.05}
Mean Squared Error (Pipeline): 133954591.3241406
R^2 Score (Pipeline): 0.9839528555639486
Mean Absolute Error (Pipeline): 6512.437414794143
Median Absolute Error (Pipeline): 4030.7971097785685
```

5 Guardar el Modelo

[8]:
```python
# Guardar el pipeline con el mejor modelo
joblib.dump(best_pipeline, "best_model_pipeline.pkl")

print("Modelo guardado exitosamente.")
```

```
Modelo guardado exitosamente.
```

6 Conclusiones

En este notebook, hemos desarrollado un pipeline para entrenar y guardar un modelo de predicción utilizando LightGBM. A continuación, se resumen los resultados obtenidos:

- **Mejor modelo**: LightGBM
- **Mejor escalador**: MinMaxScaler
- **Mejores hiperparámetros**:
  - `model__num_leaves`: 31
  - `model__n_estimators`: 200
  - `model__min_child_samples`: 50
  - `model__max_depth`: 40
  - `model__learning_rate`: 0.05

6.1 Métricas del mejor modelo:

- **Mean Squared Error (MSE)**: 133,954,591.3241406
- **R^2 Score**: 0.9839528555639486
- **Mean Absolute Error (MAE)**: 6,512.437414794143
- **Median Absolute Error (MedAE)**: 4,030.7971097785685

El modelo de LightGBM entrenado con los hiperparámetros óptimos y utilizando MinMaxScaler como técnica de escalamiento ha demostrado ser el más efectivo, logrando un alto R^2 Score y bajos errores absolutos y cuadrados medios. Además, el uso de GPU (NVIDIA GeForce RTX 3060) ha permitido acelerar el proceso de entrenamiento.