Proyecto Informe Final

Por:
Edwin David Duque Grajales cc. 1017255650 Ing Ambiental
Peter Daniel Garrido Rodriguez cc. 1085952515 Ing. Industrial
Jheison Andres Benavides Rincon cc.1015466242 Ing. Civil

Materia:
Introducción a la Inteligencia Artificial para las Ingenierías

Profesor:
Raúl Ramos Pollan

Universidad de Antioquia
Facultad de Ingeniería
Medellín 2023

INTRODUCCIÓN

La inteligencia artificial es una herramienta que nos permite modelar y procesar información que se puede utilizar para resolver problemáticas y desafíos que se presentan diariamente, en este caso los algoritmos de Machine Learning nos permiten procesar información para crear modelos de aprendizaje automático que utilicen datos de emisiones de fuente abierta (de observaciones del satélite Sentinel-5P) para predecir las emisiones de carbono. La aplicación de esta herramienta a buen término plantea una opción para interpretar las observaciones de otros satélites para las mismas variables o con otros términos de referencia , donde se hagan estimaciones , cálculos, proyecciones y análisis de los fenómenos de las emisiones de carbono y los datos observados.

Planteamiento del problema

Analizar los fenómenos de la naturaleza en los últimos años se ha vuelto una prioridad para aprender del pasado y en base a esto realizar estimaciones y proyecciones del posible futuro. Las emisiones de carbono en la actualidad son de interés científico, económico y social, ya que identificar las emisiones presentes en el entorno y las variables que las ocasionan facilita plantear y mejorar soluciones para reducirlas hasta el punto que su reducción ayude con la problemática del calentamiento global y al mismo tiempo implementar medidas puede mejorar la economía de varios sectores por su valor monetario en bolsa o programas que se encargan de reducir huellas de carbono a cambio de recursos económicos, ambientales y sociales.

Exploración del Dataset

El dataset a utilizar proviene de una competencia de Kaggle en la cual se proporcionan datos que provienen de la selección de aproximadamente 497 ubicaciones únicas de múltiples áreas de Ruanda, distribuidas entre tierras agrícolas, ciudades y plantas de energía.

Se extrajeron siete características principales semanalmente de Sentinel-5P desde enero de 2019 hasta noviembre de 2022. Cada característica (dióxido de azufre, monóxido de carbono, etc.) contiene subcaracterísticas cómo column number density, qué es la densidad de la columna vertical a nivel del suelo, calculada utilizando la técnica DOAS. Puede leer más sobre cada característica en los enlaces a continuación, incluido cómo se miden y las definiciones de las variables. Se le proporcionan los valores de estas características en el conjunto de prueba y su objetivo es predecir las emisiones de CO2 utilizando información de tiempo además de estas características.

- Dióxido de azufre - COPERNICUS/S5P/NRTI/L3_SO2
- Monóxido de carbono - COPERNICUS/S5P/NRTI/L3_CO
- Dióxido de nitrógeno - COPERNICUS/S5P/NRTI/L3_NO2

- Formaldehído - COPERNICUS/S5P/NRTI/L3_HCHO
- Índice de aerosoles UV - COPERNICUS/S5P/NRTI/L3_AER_AI
- Ozono - COPERNICUS/S5P/NRTI/L3_O3
- Nube - COPERNICUS/S5P/OFFL/L3_CLOUD

Inicialmente se hizo énfasis en la exploración y limpieza de los datos suministrados. En la exploración identificamos que el tamaño del dataset presenta 79023 observaciones y 76 características para cada observación.

```
[ ]  # Tamaño del dataset
     print (d.shape)
```

```
(79023, 76)
```

secuencialmente identificamos los valores estadísticos de los datos del estudio

```
#Valores estadisticos de las columnas
d._get_numeric_data().describe().T
```

	count	mean	std	min	25%	50%	75%	max
latitude	79023.0	-1.891072	0.694522	-3.299000	-2.451000	-1.882000	-1.303000	-0.510000
longitude	79023.0	29.880155	0.810375	28.228000	29.262000	29.883000	30.471000	31.532000
year	79023.0	2020.000000	0.816502	2019.000000	2019.000000	2020.000000	2021.000000	2021.000000
week_no	79023.0	26.000000	15.297155	0.000000	13.000000	26.000000	39.000000	52.000000
SulphurDioxide_SO2_column_number_density	64414.0	0.000048	0.000272	-0.000996	-0.000096	0.000024	0.000153	0.004191
...	...	...	...	...	...	...	...	...
Cloud_sensor_azimuth_angle	78539.0	-10.784832	30.374462	-102.739731	-30.309170	-12.673914	9.402202	78.223037
Cloud_sensor_zenith_angle	78539.0	40.436976	6.428216	2.998873	35.829907	41.119630	44.446272	65.951248
Cloud_solar_azimuth_angle	78539.0	-86.800583	37.837269	-153.464211	-125.991158	-84.644352	-48.132701	-22.653170
Cloud_solar_zenith_angle	78539.0	27.925981	4.403835	10.818288	24.686763	28.333630	31.499883	42.060436
emission	79023.0	81.940552	144.299648	0.000000	9.797995	45.593445	109.549595	3167.768000

75 rows × 8 columns

Esta operación nos permite identificar de manera clara los datos y nos obliga a tomar decisiones para la limpieza de datos. El paso que sigue es identificar los datos faltantes que presenta nuestro dataset y presentarlo en porcentaje con el fin de identificar la cantidad de datos faltantes que tiene nuestro dataset. Esto es un punto que nos va permitir filtrar datos cuando definamos las columnas categóricas del proyecto.

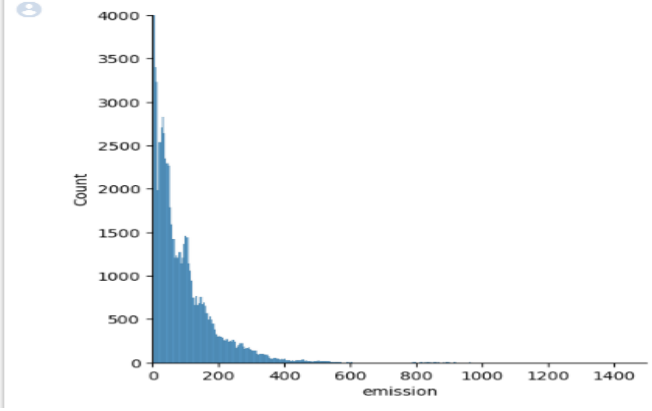
```
#Porcentaje de datos faltantes para cada variable del dataset
total          = d.isnull().sum().sort_values(ascending=False)
percent        = (d.isnull().sum()/d.isnull().count()*100).sort_values(ascending=False)
missing_train1 = pd.concat([total,percent],axis=1,keys=["Total","Percent"])
missing_train1
```

	Total	Percent
UvAerosolLayerHeight_aerosol_height	78584	99.444466
UvAerosolLayerHeight_solar_zenith_angle	78584	99.444466
UvAerosolLayerHeight_solar_azimuth_angle	78584	99.444466
UvAerosolLayerHeight_sensor_azimuth_angle	78584	99.444466
UvAerosolLayerHeight_aerosol_pressure	78584	99.444466
...	...	...

Ya con la información de las variables y la clasificación de los datos de nuestro dataset se realiza una representación de la variable principal y de las variables de las diferentes

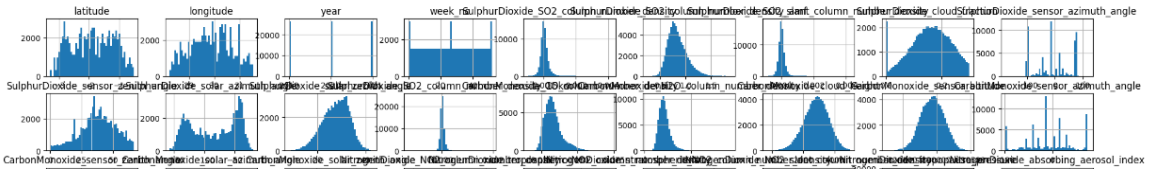
columnas para identificar el comportamiento de los datos y más adelante identificar los datos faltantes que son significativos para el comportamiento de las variables.

```
#Distribucion de la variable objetivo
sns.displot(d['emission'])
plt.xlim(0, 1500)
plt.ylim(0, 4000)
plt.show()
```




```
#Histogramas con los comportamientos de las diferentes columnas
d.hist(bins=50, figsize=(24,15))
plt.suptitle('Histograms For the Different Features', fontsize=30)
plt.show()
```

Histograms For the Different Features



Para darle poder de significancia a los datos del dataset hacemos la matriz de correlaciones de las variables involucradas con la variable objetivo en este caso la emisión de carbono.

```
#Correlaciones de las variables con la variable objetivo
target_corr_df = pd.DataFrame(corr_matrix["emission"].sort_values(ascending=False))
target_corr_df
```

	emission
emission	1.000000
longitude	0.102746
UvAerosolLayerHeight_aerosol_height	0.069008
Cloud_surface_albedo	0.046587
Formaldehyde_tropospheric_HCHO_column_number_density_amf	0.040263
...	...
Formaldehyde_tropospheric_HCHO_column_number_density	-0.033333
NitrogenDioxide_solar_azimuth_angle	-0.033417
CarbonMonoxide_CO_column_number_density	-0.041328
CarbonMonoxide_H2O_column_number_density	-0.043217
UvAerosolLayerHeight_aerosol_pressure	-0.068138

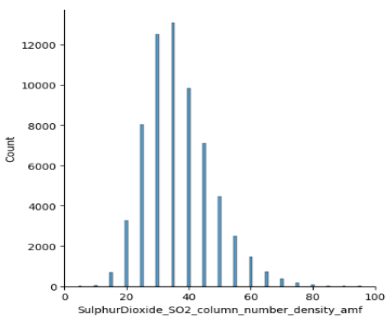
75 rows × 1 columns

Después de esta exploración de datos, procedemos a limpiar los datos que tengan faltantes, sin embargo se definen las columnas categóricas y observamos su comportamiento gráfico, este procedimiento se realiza para las columnas categóricas que están establecidas.

Generamos algunas columnas categoricas

```
[ ] list_catgr = ["SulphurDioxide_SO2_column_number_density_amf", "CarbonMonoxide_CO_column_number_density",
    d_01 = d.copy()

    for i in list_catgr:
      intervalo = np.linspace(d_01[i].min(), d_01[i].max(), 20).tolist()
      d_01[i] = pd.cut(d_01[i], bins=intervalo, labels=list(range(5,100,5)))
```

```
for i in list_catgr:
  sns.displot(d_01[i])
  plt.show()
```



Continuando se procede a segmentar esos datos de tal manera que escojamos aquellos que tengan un porcentaje de datos faltantes menor al 10%, esto con el fin de cumplir los criterios establecidos del proyecto y utilizar los datos que más nos convengan para evaluar el modelo con respecto a las columnas categóricas.

Hora debemos rellenar los datos que siguen estando faltantes

```
[ ] #Porcentaje de datos faltantes para cada variable del dataset
    total          = d_02.isnull().sum().sort_values(ascending=False)
    percent        = (d_02.isnull().sum()/d_02.isnull().count()*100).sort_values(ascending=False)
    missing_train1 = pd.concat([total,percent],axis=1,keys=["Total","Percent"])
    missing_train1
```

	Total	Percent
Formaldehyde_tropospheric_HCHO_column_number_density	7278	9.209977
Formaldehyde_tropospheric_HCHO_column_number_density_amf	7277	9.208711
Formaldehyde_sensor_zenith_angle	7277	9.208711
Formaldehyde_solar_azimuth_angle	7277	9.208711
Formaldehyde_solar_zenith_angle	7277	9.208711
Formaldehyde_cloud_fraction	7277	9.208711
Formaldehyde_HCHO_slant_column_number_density	7277	9.208711

Para finalizar escogemos el parámetro para resolver los datos faltantes, lo cual se lleva a cabo reemplazando el dato faltante por la moda del grupo de datos, con el fin de representar los datos de tal manera que no cambien de manera significativa en estos casos y no representan cambios drásticos de acuerdo a la ubicación que se encuentran el dataset.

```
[ ] # Completar los valores faltantes en las columnas restantes con la moda
    for feature in d_02.columns:
        if d_02[feature].isnull().sum() > 0:
            mode_value = d_02[feature].mode()[0] # Calcular la moda de la columna
            d_02[feature].fillna(mode_value, inplace=True)

    print(d_02.head)
```

```
<bound method NDFrame.head of            ID_LAT_LON_YEAR_WEEK  latitude  longitude  year  week_no  \
0      ID_-0.510_29.290_2019_00    -0.510     29.290  2019        0
1      ID_-0.510_29.290_2019_01    -0.510     29.290  2019        1
2      ID_-0.510_29.290_2019_02    -0.510     29.290  2019        2
3      ID_-0.510_29.290_2019_03    -0.510     29.290  2019        3
4      ID_-0.510_29.290_2019_04    -0.510     29.290  2019        4
...                         ...       ...        ...   ...      ...
79018  ID_-3.299_30.301_2021_48    -3.299     30.301  2021       48
79019  ID_-3.299_30.301_2021_49    -3.299     30.301  2021       49
79020  ID_-3.299_30.301_2021_50    -3.299     30.301  2021       50
79021  ID_-3.299_30.301_2021_51    -3.299     30.301  2021       51
79022  ID_-3.299_30.301_2021_52    -3.299     30.301  2021       52
```

Iteraciones de Desarrollo

Con el enfoque de los datos ordenados y teniendo una limpieza de los datos del Datased, se trabaja con la variable principal que son las emisiones de carbono que están presentes en los datos que están presentes hasta este punto. Para ello, se calibran las particiones del periodo y la modelación correspondiente en este caso. Al realizar la calibración tenemos en cuenta que los datos del proceso y modelamiento son de 79023 datos de emisiones de carbono que van a ser estudiados por lo cual puede presentarse un procesamiento demorado debido a la cantidad de datos de análisis.

Hacemos las particiones para el periodo de calibraciòn y modelaciòn

```
d_02 = d_02.set_index('ID_LAT_LON_YEAR_WEEK')  # Indico la columna que quiero que me tome como indice
```

```
[ ] X = d_02.values[:,:-1]          # Separo los valores que quiero que sean predichos del dataset inicial
    y = d_02["emission"].values
    print (X.shape, y.shape)

    (79023, 46) (79023,)
```

Una vez segmentados los datos de emisión de gases de carbono del dataset, se procede a correr un diagnóstico de modelos de regresión lineal para determinar el valor de error absoluto medio (MAE), y definir qué metodología de regresión lineal muestra mejores resultados. con ello una calibración y validación de los datos es necesaria para continuar con el proceso de diagnóstico.

```
[ ] Xtr, Xts, ytr, yts = train_test_split(X,y, test_size=0.3)        # Hago la particion de calibracion (trent) y validacion (test)
    print (Xtr.shape, ytr.shape, Xts.shape, yts.shape)

    (55316, 46) (55316,) (23707, 46) (23707,)
```

El Error Absoluto Medio (MAE) es una métrica de evaluación comúnmente utilizada en problemas de regresión. Indica el promedio de las diferencias absolutas entre las predicciones del modelo y los valores reales en el conjunto de datos de prueba. El MAE se expresa en las mismas unidades que la variable objetivo y proporciona una medida absoluta del error promedio. El rango del MAE depende de la escala de la variable objetivo. Dado que es una medida absoluta, no tiene una escala específica y puede variar según el rango de los valores de la variable objetivo. Cuanto más pequeño sea el valor del MAE, mejor será el rendimiento del modelo.

Debido a la aclaración del MAE, el primer diagnóstico se realizó con el método de regresión lineal sencillo donde se realiza una regresión lineal de los datos que fueron calibrados y validados. se determina el valor del error de la métrica definida, para este caso el valor del

error absoluto medio de esta regresión es de MAE: 68.89609001910372. lo cual nos determina que es un factor considerable de error de los datos predecibles del modelo con respecto a los valores reales.

```
[ ]  lr = LinearRegression()                #implemento un primer modelo muy sencillo (regresion lineal)
     lr.fit(Xtr, ytr)
     y_pred_lr = lr.predict(Xts)
     mae = mean_absolute_error(yts, y_pred_lr)
     print(f'MAE: {mae}')

     MAE: 68.89609001910372
```

Para el segundo diagnóstico se utilizó el Modelo support vector machine, donde se tiene en cuenta los mismos parámetros del primero modelo respecto a la validación y la calibración de los datos , donde el valor fue de MAE: 66.14332683234684, lo cual determina que es un valor muy semejante al anterior de regresión lineal.

```
svr_model = SVR(kernel='poly')          #Modelo support vector machine
svr_model.fit(Xtr, ytr)
y_pred_svr = svr_model.predict(Xts)
mae = mean_absolute_error(yts, y_pred_svr)
print(f'MAE: {mae}')

MAE: 66.14332683234684
```

El siguiente modelo de predicciones es el llamado “Modelo Random Forest Regressor” y siguiendo los lineamientos de validación y calibración de las emisiones ya segmentadas se obtuvo el valor de MAE: 8.105361054326764 , lo cual resultó en el valor menor de los 3 modelos propuestos de análisis, es decir es el de mayor valor de predicción respecto a los datos reales de base.

```
rf_model = RandomForestRegressor(random_state=42)      # Modelo Random Forest Regressor
rf_model.fit(Xtr, ytr)
y_pred_rf = rf_model.predict(Xts)
mae = mean_absolute_error(yts, y_pred_rf)
print(f'MAE: {mae}')

MAE: 8.105361054326764
```

Adicionalmente un modelo a tener en cuenta y que está implementado en las soluciones de Kaggle, fue dispuesto en el colbab de referencia del trabajo para tenerlo en cuenta. Una nota de este modelo es el alto costo económico. el valor del MAE : 8.019250045586999.

```
1 rf_2 = RandomForestRegressor(n_estimators=300, random_state=200, n_jobs=-1)
2 rf_2.fit(Xtr, ytr)
3 y_preds_rf_2 = rf_2.predict(Xts)
4 mae = mean_absolute_error(yts, y_preds_rf_2)
5 print(f'MAE: {mae}')

MAE: 8.019250045586999
```

También hacemos una visualización de las predicciones que se tienen hasta el momento de acuerdo a la modelación y los procesos de análisis realizados.

```
print(predictions)
print(yts)
```

```
[189.2114    78.65956  113.804085 ...  11.874097  84.92711  102.827644]
[239.34853   226.12766    84.97233   ...   0.9152649  57.74897
  98.36536  ]
```

En términos generales, el MAE se interpreta como :

- MAE = 0:* Significa que el modelo predice exactamente los valores reales. Esto es poco común en la práctica y podría indicar un sobreajuste.
- MAE cercano a 0:* Indica un buen rendimiento del modelo, donde las predicciones son muy cercanas a los valores reales.
- MAE más grande:* A medida que el MAE aumenta, indica que las predicciones del modelo tienen mayores diferencias absolutas con los valores reales. Un MAE más grande sugiere un rendimiento menos preciso del modelo.

El MAE proporciona una medida de la precisión promedio del modelo en términos absolutos. Un MAE más bajo indica un mejor rendimiento, pero la interpretación específica puede depender del contexto y de la escala de la variable objetivo.

Una vez observados los modelos de predicción y los valores del MAE , se continúa con la validación cruzada que es una técnica utilizada en machine learning para evaluar el rendimiento de un modelo de manera más robusta y fiable. La idea principal detrás de la validación cruzada es dividir el conjunto de datos en múltiples subconjuntos, entrenar y evaluar el modelo varias veces, y luego promediar los resultados para obtener una estimación más precisa del rendimiento del modelo.

validación Cruzada

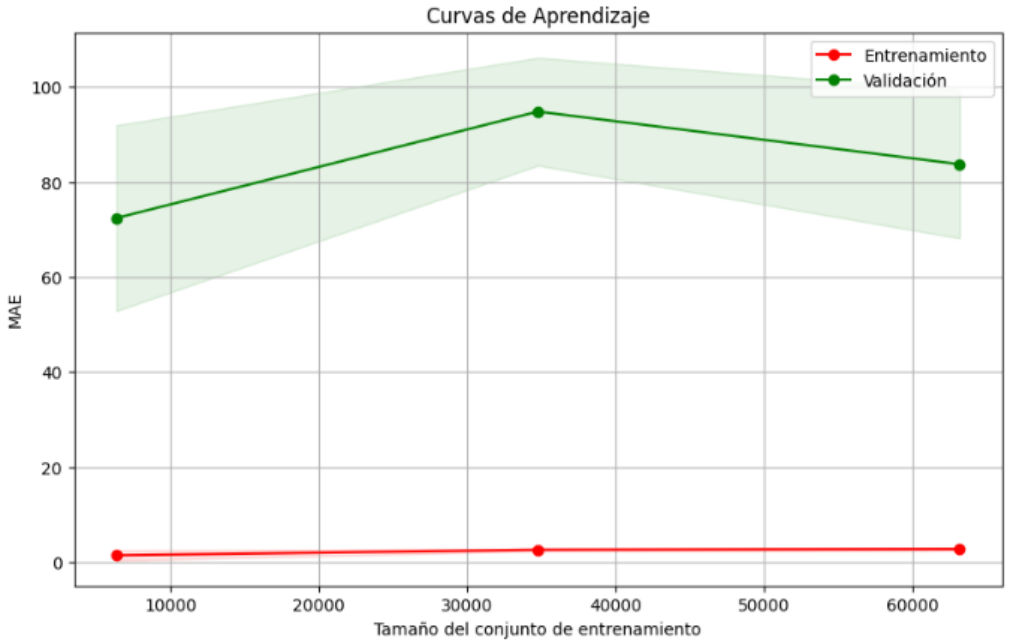
```
[ ] from sklearn.model_selection import cross_val_score      # Validacion cruzada

    # Realizar validación cruzada con 5 folds
    scores = cross_val_score(rf_model, X, y, scoring='neg_mean_absolute_error', cv=5) # se ingresa el modelo de random forest creado anteriormente

    # Calcular el MAE promedio a partir de los scores
    mae_cv = -scores.mean()

    print("MAE - Validación Cruzada:", mae_cv)

    MAE - Validación Cruzada: 83.70171352383599
```

Una guia para realizar la validación cruzada en el caso particular del proyecto es :

- Crear un modelo:* En este ejemplo, se utiliza un modelo de bosque aleatorio (`RandomForestRegressor`).
- Crear un objeto de validación cruzada (`KFold`):* `KFold` divide el conjunto de datos en k particiones (folds) para realizar iteraciones de entrenamiento y evaluación. `n_splits` indica el número de particiones, `shuffle=True` mezcla los datos antes de dividirlos y `random_state` proporciona reproducibilidad.
- Realizar validación cruzada:* La función `cross_val_score` toma el modelo, el conjunto de datos (`X` e `y`), el objeto de validación cruzada (`kf`), y la métrica de evaluación. Devuelve una matriz de puntuaciones de rendimiento para cada iteración.
- Imprimir los resultados.

Continuando con el proceso del desarrollo del proyecto identificamos las curvas de aprendizaje donde identificamos los valores promedio y las desviaciones estándar de las emisiones de carbono, donde los valores del MAE serán observados respecto al número de datos para las predicciones. y se observa que el comportamiento de los datos del test con respecto a los datos que estudia el modelo (entrenamiento) es de tipo overfitting o de sobreajuste, lo es un fenómeno común en el aprendizaje automático (machine learning).

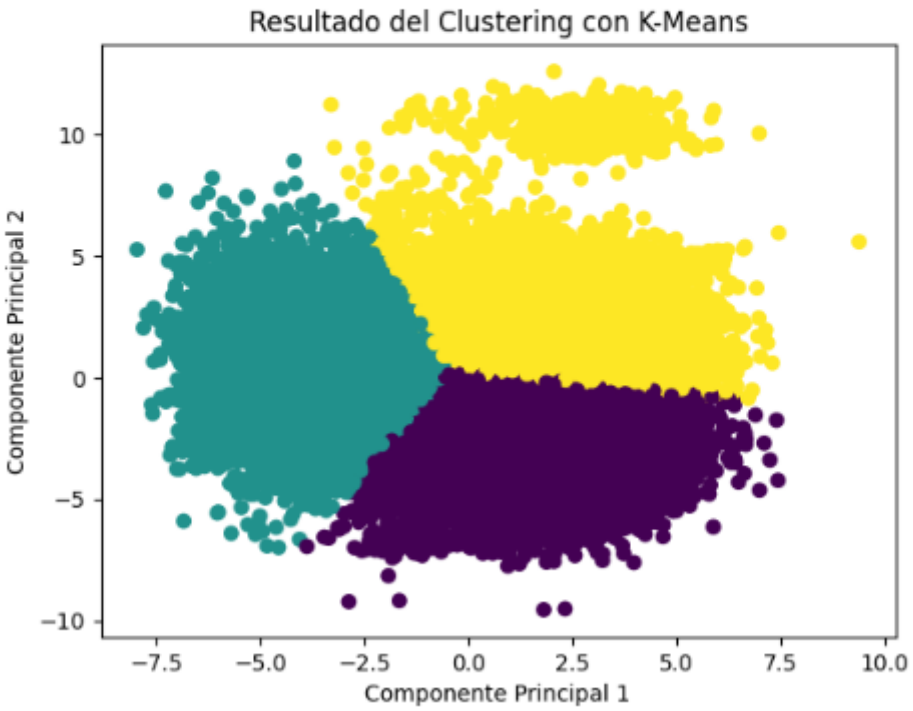
```python
from sklearn.model_selection import learning_curve                #Curva de aprendizaje
train_sizes, train_scores, val_scores = learning_curve(
    rf_2, X, y, train_sizes=np.linspace(0.1, 1.0, 3),
    scoring='neg_mean_absolute_error', cv=5
)

# Calcular los valores promedio y desviaciones estándar
train_scores_mean = -np.mean(train_scores, axis=1)
train_scores_std = np.std(train_scores, axis=1)
val_scores_mean = -np.mean(val_scores, axis=1)
val_scores_std = np.std(val_scores, axis=1)

# Plotear las curvas de aprendizaje
plt.figure(figsize=(10, 6))
plt.title('Curvas de Aprendizaje')
plt.xlabel('Tamaño del conjunto de entrenamiento')
plt.ylabel('MAE')
plt.grid(True)
plt.fill_between(train_sizes, train_scores_mean - train_scores_std,
                 train_scores_mean + train_scores_std, alpha=0.1, color='r')
plt.fill_between(train_sizes, val_scores_mean - val_scores_std,
                 val_scores_mean + val_scores_std, alpha=0.1, color='g')
plt.plot(train_sizes, train_scores_mean, 'o-', color='r', label='Entrenamiento')
plt.plot(train_sizes, val_scores_mean, 'o-', color='g', label='Validación')
plt.legend(loc='best')
plt.show()
```



Para terminar con la parte de desarrollo, identificamos los métodos no supervisados de los datos de estudio en este caso, los valores de las emisiones de carbono.

```
# Normalizar los datos
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Especificar el número de componentes principales deseados para PCA
n_components = 2

# Crear el modelo PCA
pca = PCA(n_components=n_components)

# Aplicar PCA a los datos normalizados
X_pca = pca.fit_transform(X_scaled)

# Crear un nuevo DataFrame con los componentes principales
columns_pca = [f'PC{i+1}' for i in range(n_components)]
df_pca = pd.DataFrame(data=X_pca, columns=columns_pca)

# Seleccionar el número de clusters para K-Means
num_clusters = 3

# Crear el modelo K-Means
kmeans = KMeans(n_clusters=num_clusters, random_state=1)

# Aplicar K-Means a los componentes principales
kmeans.fit(X_pca)

# Agregar las etiquetas de cluster al DataFrame original
d_02['cluster'] = kmeans.labels_

# Crear un nuevo DataFrame con los componentes principales y las etiquetas de cluster
df_pca_cluster = pd.concat([df_pca, d_02[['emission', 'cluster']]], axis=1)
```

```
[ ]  # Visualizar los resultados del clustering
     plt.scatter(X_pca[:, 0], X_pca[:, 1], c=d_02['cluster'], cmap='viridis')
     plt.title('Resultado del Clustering con K-Means')
     plt.xlabel('Componente Principal 1')
     plt.ylabel('Componente Principal 2')
     plt.show()
```



Ya con estos resultados obtenidos se identifican los puntos más importantes y consideraciones a tener en cuenta de las modelaciones.

Consideraciones

Es importante comparar el MAE con el rango de valores de la variable objetivo para contextualizar su significado. Por ejemplo, si los con datos que tienen una gran variabilidad, un MAE relativamente pequeño podría ser aceptable. Sin embargo, si los valores de la variable objetivo son todos pequeños, el mismo MAE podría indicar un rendimiento deficiente. En resumen, el MAE proporciona una medida de la precisión promedio del modelo en términos absolutos. Un MAE más bajo indica un mejor rendimiento, pero la interpretación específica puede depender del contexto y de la escala de la variable objetivo.

En cuanto a la validación cruzada proporciona una evaluación más robusta del rendimiento del modelo al utilizar múltiples divisiones del conjunto de datos. Esto es especialmente útil

cuando el tamaño del conjunto de datos es limitado y se desea una estimación más precisa del rendimiento del modelo.

Por último, el sobreajuste, o overfitting, es un fenómeno común en el aprendizaje automático (machine learning) y la inteligencia artificial. Ocurre cuando un modelo se ajusta demasiado a los datos de entrenamiento, capturando no solo los patrones inherentes a los datos sino también el ruido o la variabilidad aleatoria. Como resultado, el modelo puede tener un rendimiento excelente en los datos de entrenamiento, pero generaliza mal a nuevos datos, ya que ha memorizado peculiaridades en lugar de aprender patrones más generales.

Retos

- Determinar un rango de datos de la variable objetivo de emisiones de gases de carbono con variabilidad pequeña para determinar un modelo aceptable en las predicciones.
- Identificar los tipos de datos y la desviación estándar de los mismos, como el valor significativo ya que valores pequeños podrían ocasionar que el MAE tenga un valor que se considere deficiente debido a esta característica de los datos.
- Obtener un valor del MAE menor del 10% ya que puede determinar que es una predicción aceptable para análisis.
- Identificar la manera para que la cantidad de datos de estudio puedan ser analizados de manera sencilla y eficiente con respecto al tiempo de procesamiento de la información.
- Limitar los datos del procesamiento que permitan resultados que puedan ser analizados dentro del margen de error permitido
- Desarrollar el modelo para que identifique los patrones de la información para que la variabilidad de los datos predecidos no sea procesada de manera errónea (overfitting)

Conclusiones

Al analizar los valores del Error Absoluto Medio (MAE) de los distintos métodos de regresión, se observa que la regresión lineal sencilla tiene un MAE más alto (68.90), indicando una mayor discrepancia entre las predicciones y los valores reales. Por otro lado, tanto el modelo Support Vector Machine (66.14) como el Random Forest Regressor (8.11) muestran MAE más bajos, lo que sugiere una mejor capacidad de predicción y ajuste a los datos.

Es destacable que un modelo implementado en una solución en Kaggle, a pesar de tener un alto costo computacional, presenta un MAE (8.02) similar al del Random Forest Regressor. Esto sugiere que la solución implementada en Kaggle, a pesar de su mayor complejidad computacional, logra resultados comparables en términos de precisión predictiva al modelo Random Forest Regressor, que es conocido por su capacidad para manejar conjuntos de datos complejos. En consecuencia, la elección del modelo para el proyecto puede ser el modelo implementado en una solución en Kaggle y el Random Forest Regressor debido a su menor discrepancia.

Además, los valores del Error Absoluto Medio (MAE) proporcionan una métrica efectiva para evaluar y comparar el rendimiento de diferentes modelos de regresión en este conjunto de datos. El MAE, al medir la magnitud promedio de los errores absolutos entre las predicciones y los valores reales, ofrece una indicación clara de la precisión de cada modelo.En este contexto, observamos que la regresión lineal sencilla tiene un MAE más alto, sugiriendo una menor precisión en sus predicciones. Por otro lado, tanto el modelo Support Vector Machine como el Random Forest Regressor presentan MAE más bajos, indicando un mejor ajuste a los datos y una mayor capacidad predictiva.

Es interesante notar que, a pesar del alto costo computacional, un modelo implementado en una solución en Kaggle muestra un MAE comparable al del Random Forest Regressor. Esto destaca la importancia de considerar no sólo la precisión, sino también los recursos computacionales necesarios al seleccionar un modelo para una aplicación práctica. En resumen, el MAE emerge como una métrica valiosa en la evaluación de modelos de regresión, proporcionando información clave sobre su rendimiento y permitiendo la toma de decisiones informadas en la selección del modelo más adecuado para el análisis de estos datos.

Teniendo en cuenta la curva de aprendizaje se puede observar que el comportamiento del MAE en los tamaños de entrenamiento y validación tiene un comportamiento de sobreajuste, o overfitting. lo cual sugiere hacer técnicas de regularización, la validación cruzada, o utilizar más datos de entrenamiento. además de verificar los valores de los datos si es que el MAE está aplicando correctamente. Cabe destacar que tener un modelo muy complejo también es un factor a tener en cuenta en estos resultados.

Bibliografía

Communications. (2021, marzo 9). ¿Qué es el dióxido de carbono (CO2) y cómo impacta en el planeta? BBVA. https://www.bbva.com/es/sostenibilidad/que-es-el-dioxido-de-carbono-co2-y-como-impacta-en-el-planeta/

Datacleaning Limpieza de datos: definición, importancia. (2022, abril 7). Formation Data Science | Datascientest.com. https://datascientest.com/es/datacleaning-limpieza-de-datos-definicion-tecnicas-importancia-en-data-science

Differential Optical Absorption Spectroscopy, 2011. Recuperado el 22 de septiembre de 2023, http://repositorio.gestiondelriesgo.gov.co/bitstream/20.500.11762/20564/3/Fenomeno_nino-2016.pdf

Predict CO2 emissions in Rwanda. (s/f). Kaggle.com. Recuperado el 22 de octubre de 2023, de https://www.kaggle.com/competitions/playground-series-s3e20/data

Wikipedia contributors. (s/f). *Error absoluto medio*. Wikipedia, The Free Encyclopedia. https://es.wikipedia.org/w/index.php?title=Error_absoluto_medio&oldid=146234007