Entrega Final

Integrantes

Jhon Alexander Botero Gómez
C.C. 1038418435
Giovani Steven Cardona Marín
C.C. 1035913434

Proyecto

Allstate Claims Severity
Competición de Kaggle

Semestre 2023-2

Introducción

En el marco de este proyecto de inteligencia artificial, nos embarcamos en la tarea desafiante y apasionante de desarrollar un modelo predictivo destinado a prever la severidad de los reclamos presentados a la aseguradora Allstate. Enfocándonos en un conjunto de datos robusto proporcionado por la compañía, compuesto por 188,318 filas y 131 columnas, abordamos el análisis de información tanto numérica como categórica.

En este proyecto, exploraremos las estrategias para abordar la ausencia de datos, destacando la simulación de eliminación y la posterior sustitución mediante sustitución de datos. Además, profundizaremos en las decisiones tomadas durante la experimentación y la investigación, explorando el proceso de análisis de datos, la identificación de patrones, y la construcción de correlaciones esenciales para la comprensión de las interacciones entre variables. Este proyecto no solo se centra en la creación de un modelo predictivo eficiente, sino también en el desarrollo de una base analítica sólida que contribuya a mejorar el aprendizaje de nuestro equipo en procesos de inteligencia artificial y el potencial de aplicaciones que puede llegar a tener.

Exploración descriptiva del dataset

Se procede a cargar los datos del kaggle en el colab con los siguientes comandos

```
!pip install -U -q kaggle
!mkdir -p ~/.kaggle
from google.colab import files
files.upload() # Please follow the repository instructions to join the competition and load your account kaggle.json
```

```
Elegir archivos  kaggle.json
• kaggle.json(application/json) - 70 bytes, last modified: 25/11/2023 - 100% done
Saving kaggle.json to kaggle.json
{'kaggle.json': b'{"username":"giovanicardona","key":"69381ff6804d8033c6dcc4bccda94484"}'}
```

Se instala Kaggle en Google Colab para cargar competencias, requiriendo el archivo Kaggle.json generado por la plataforma de Kaggle para su selección en el entorno de Colab.

```
!cp kaggle.json ~/.kaggle/
!kaggle competitions download -c allstate-claims-severity
```

```
Warning: Your Kaggle API key is readable by other users on this system! To fix this, you can run 'chmod 600 /root/.kaggle/kaggle.json'
Downloading allstate-claims-severity.zip to /content
 68% 33.0M/48.8M [00:00<00:00, 48.4MB/s]
100% 48.8M/48.8M [00:00<00:00, 70.3MB/s]
```

con esto cargamos los datos de la competencia al colab

```
loss_amount_train = pd.read_csv('kaggle/train.csv', low_memory=True)
loss_amount_train.info()
loss_amount_train.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 188318 entries, 0 to 188317
Columns: 132 entries, id to loss
dtypes: float64(15), int64(1), object(116)
memory usage: 189.7+ MB
```

	id	cat1	cat2	cat3	cat4	cat5	cat6	cat7	cat8	cat9	...	cont6	cont7	cont8	cont9	cont10	cont11	cont12	cont13	cont14	loss
0	1	A	B	A	B	A	A	A	A	B	...	0.718367	0.335060	0.30260	0.67135	0.83510	0.569745	0.594646	0.822493	0.714843	2213.18
1	2	A	B	A	A	A	A	A	A	B	...	0.438917	0.436585	0.60087	0.35127	0.43919	0.338312	0.366307	0.611431	0.304496	1283.60
2	5	A	B	A	A	B	A	A	A	B	...	0.289648	0.315545	0.27320	0.26076	0.32446	0.381398	0.373424	0.195709	0.774425	3005.09
3	10	B	B	A	B	A	A	A	A	B	...	0.440945	0.391128	0.31796	0.32128	0.44467	0.327915	0.321570	0.605077	0.602642	939.85
4	11	A	B	A	B	A	A	A	A	B	...	0.178193	0.247408	0.24564	0.22089	0.21230	0.204687	0.202213	0.246011	0.432606	2763.85

5 rows × 132 columns

```
loss_amount_test = pd.read_csv('kaggle/test.csv', low_memory=True)
loss_amount_test.info()
loss_amount_test.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 125546 entries, 0 to 125545
Columns: 131 entries, id to cont14
dtypes: float64(14), int64(1), object(116)
memory usage: 125.5+ MB
```

	id	cat1	cat2	cat3	cat4	cat5	cat6	cat7	cat8	cat9	...	cont5	cont6	cont7	cont8	cont9	cont10	cont11	cont12	cont13	cont14
0	4	A	B	A	A	A	A	A	A	B	...	0.281143	0.466591	0.317681	0.61229	0.34365	0.38016	0.377724	0.369858	0.704052	0.392562
1	6	A	B	A	B	A	A	A	A	B	...	0.836443	0.482425	0.443760	0.71330	0.51890	0.60401	0.689039	0.675759	0.453468	0.208045
2	9	A	B	A	B	B	A	B	A	B	...	0.718531	0.212308	0.325779	0.29758	0.34365	0.30529	0.245410	0.241676	0.258586	0.297232
3	12	A	A	A	A	B	A	A	A	A	...	0.397069	0.369930	0.342355	0.40028	0.33237	0.31480	0.348867	0.341872	0.592264	0.555955
4	15	B	A	A	A	A	B	A	A	A	...	0.302678	0.398862	0.391833	0.23688	0.43731	0.50556	0.359572	0.352251	0.301535	0.825823

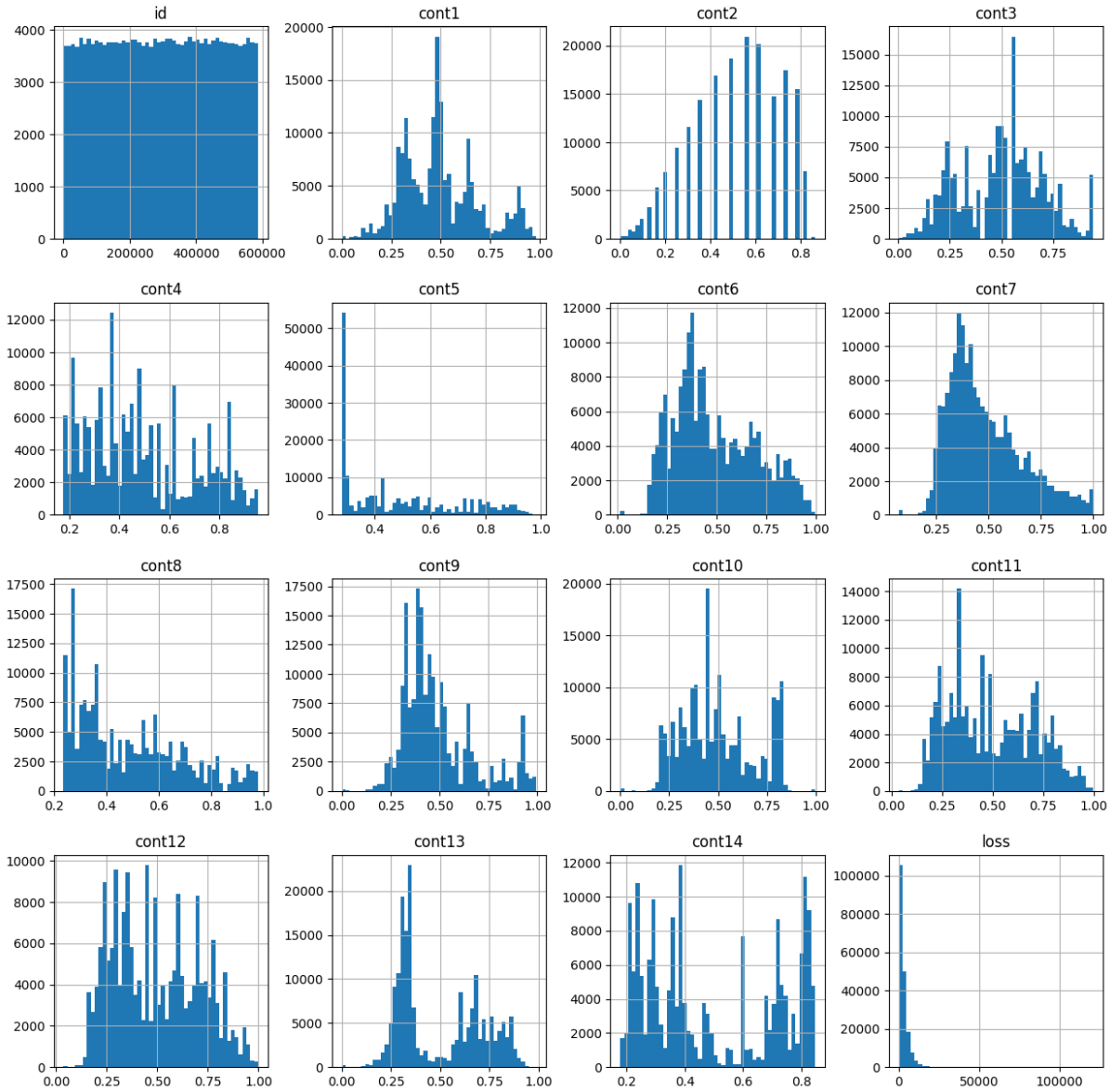
5 rows × 131 columns

se empieza a mirar los datos a trabajar obteniendo más información sobre de ellos

```
df_tr = loss_amount_train.copy()
df_tr.isnull().values.any()
```

```
False
```

miramos hay nulo en la base de datos



hacemos un histograma para tener más información sobre los datos

Preprocesado de Datos

El código selecciona aleatoriamente tres columnas del conjunto de datos y establece el 5% de sus valores en NaN (valores nulos) para simular información faltante. Esto se hace asumiendo que los datos originales están completos y se necesita emular la presencia de información ausente con fines de análisis o pruebas relacionadas con datos faltantes.
El objetivo de esta acción es generar un escenario donde se simula la falta de información en ciertas columnas de manera aleatoria. Esta simulación es común en tareas de análisis de datos para comprender cómo los modelos o algoritmos manejan y se comportan ante la presencia de datos faltantes.

```
empties_qty = int(np.ceil(len(df_tr) * 0.05))

# Obtener 3 columnas numéricas para simular datos faltantes
cols_nan = df_tr.iloc[:,-15:-1].sample(n=3,axis='columns').columns

for column in cols_nan:
    print(column)
    print('Max: ' + str(max(df_tr.loc[:,column])))
    print('Min: ' + str(min(df_tr.loc[:,column])))
    print('Mean: ' + str(np.mean(df_tr.loc[:,column])))
    print('')

    nan_rows = df_tr.sample(n=empties_qty)
    nan_rows[column] = np.nan
    df_tr.loc[df_tr.index.isin(nan_rows.index),column] = nan_rows[column]
```

```
cont2
Max: 0.862654
Min: 0.001149
Mean: 0.507188356179441

cont4
Max: 0.954297
Min: 0.176921
Mean: 0.49181230258923736

cont6
Max: 0.997162
Min: 0.012683
Mean: 0.49094453373548996
```

Se remplazan los valores nulos con la media de la columna

```
for col in cols_nan:
    mean = np.mean(df_tr[col])
    df_tr[col].fillna(mean, inplace=True)
```

describe un proceso donde se convierten variables categóricas en variables continuas. Durante este proceso, se observa que la correlación

más alta entre las variables transformadas es aproximadamente de 0.9557. Ninguna pareja de variables tiene una correlación igual a 1, lo que sugiere una alta pero no perfecta relación lineal entre ellas. A continuación, se menciona que se va a proceder a visualizar esta correlación.

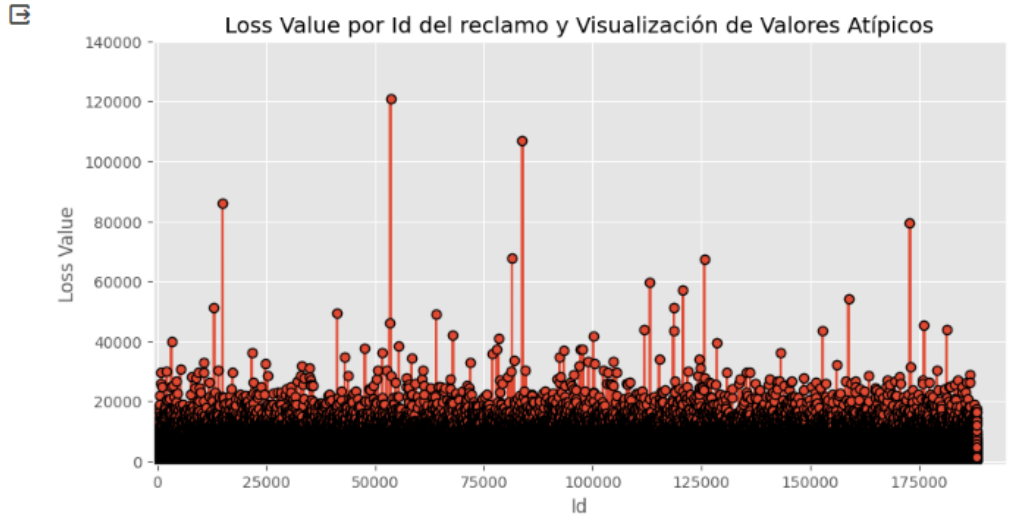
Se destaca la importancia de las características en relación con la variable "loss". De un total de 130 características, se analizarán las 30 que tienen una contribución más significativa hacia la variable de pérdida.

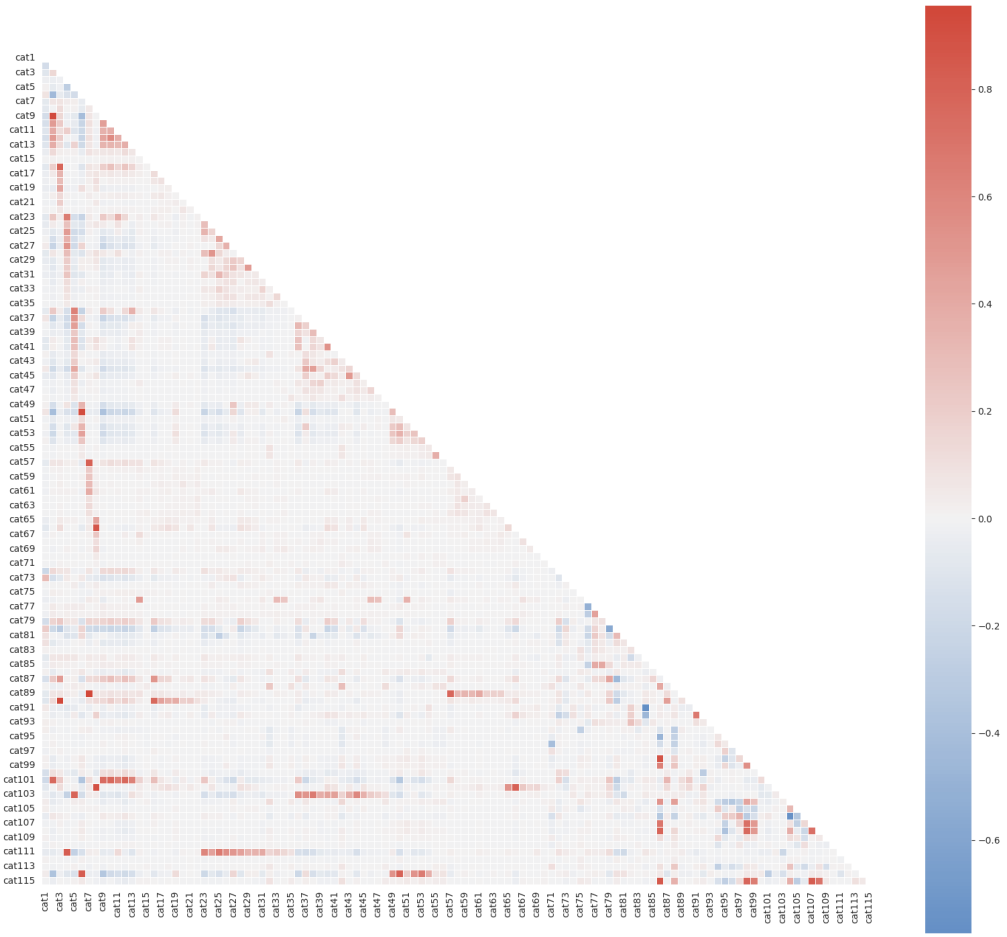
El texto también hace referencia a un gráfico anterior que muestra el monto de pérdida por identificación de reclamo. Este gráfico resalta valores atípicos en los montos de reclamo, específicamente señalando un valor atípico notablemente alto con una pérdida cercana a 120,000 USD.

En resumen, se detalla un proceso de transformación de variables, se destaca la correlación entre ellas, se analiza la importancia de las características en relación con la variable "loss" y se menciona la presencia de valores atípicos en el monto de pérdida por identificación de reclamo.

Si necesitas más detalles sobre alguna parte específica o alguna aclaración adicional, no dudes en preguntar.

```
plt.figure(figsize=(10,5))
plt.xlabel('Id')
plt.ylabel('Loss Value')
plt.title('Loss Value por Id del reclamo y Visualización de Valores Atípicos')
plt.xlim([-1000, 195000])
plt.ylim([-1000, 140000])
plt.plot(df_tr.index, df_tr["loss"], marker='o', markeredgecolor='k')
plt.show()
```




cat1
cat3
cat5
cat7
cat9
cat11
cat13
cat15
cat17
cat19
cat21
cat23
cat25
cat27
cat29
cat31
cat33
cat35
cat37
cat39
cat41
cat43
cat45
cat47
cat49
cat51
cat53
cat55
cat57
cat59
cat61
cat63
cat65
cat67
cat69
cat71
cat73
cat75
cat77
cat79
cat81
cat83
cat85
cat87
cat89
cat91
cat93
cat95
cat97
cat99
cat101
cat103
cat105
cat107
cat109
cat111
cat113
cat115
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6

Modelos Supervisados

En el desarrollo de este proyecto, se llevó a cabo una exhaustiva evaluación de diferentes modelos supervisados con el objetivo de determinar cuál proporcionaría la representación más significativa de los datos. Este proceso iterativo implicó la prueba y comparación de varios modelos, entre los cuales se incluyeron la Regresión Lineal, Regresión Ridge, Regresión Lasso, Regresión Elastic Net y Regresión Random Forest.

Cada modelo fue sometido a una evaluación rigurosa mediante indicadores y análisis gráficos con el fin de validar su desempeño y su capacidad para capturar la complejidad inherente de los datos. A lo largo de este proceso, se realizaron comparaciones detalladas, considerando múltiples criterios para tomar decisiones informadas sobre cuál sería el modelo más adecuado para los objetivos del proyecto.

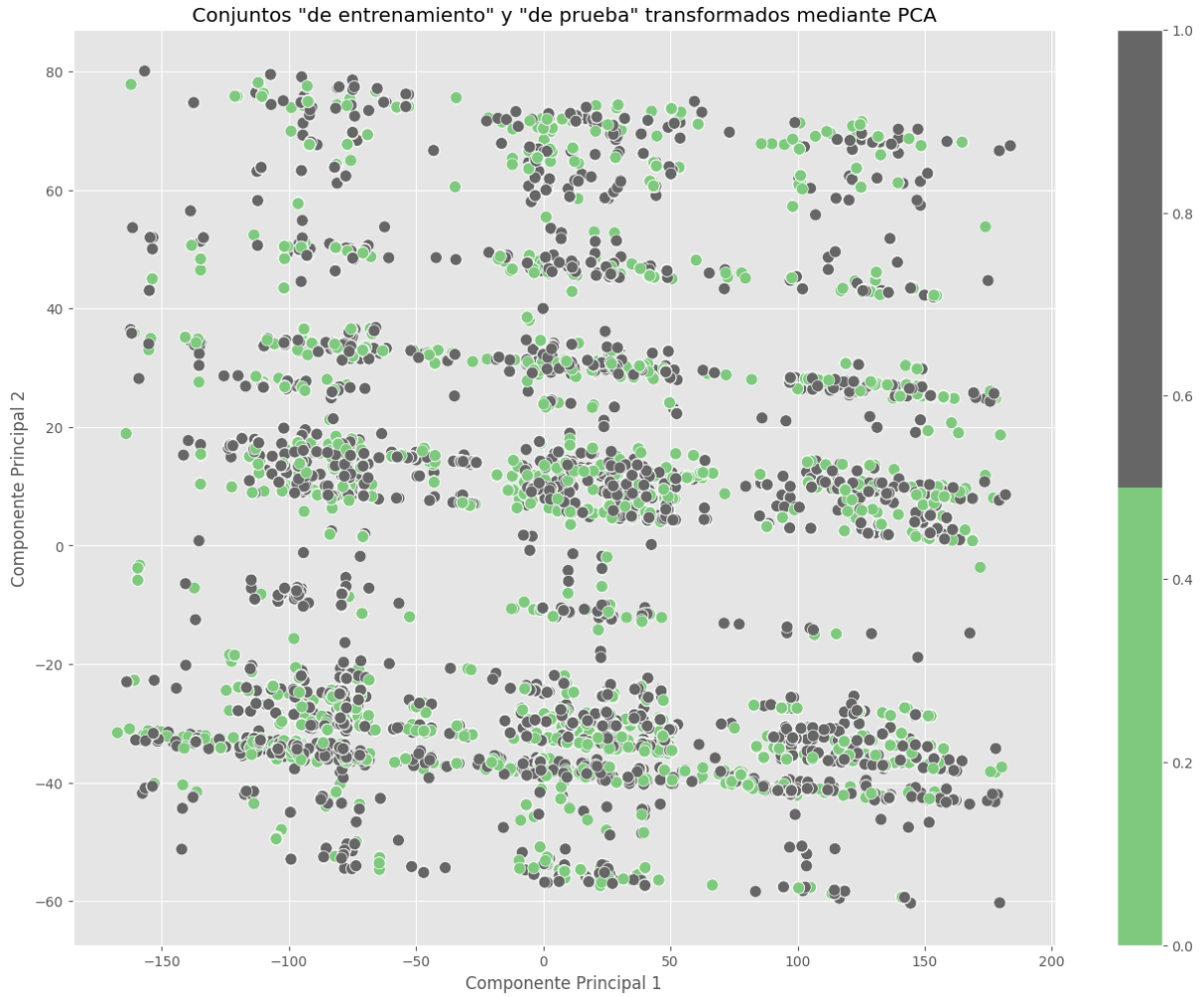
Tras un análisis exhaustivo y cuidadoso, documentado en el Notebook del Colaboratory, se llegó a la conclusión de que la Regresión Random Forest emergió como el modelo final y más representativo. Este modelo demostró ser eficaz al abordar la complejidad de los datos, ofreciendo resultados sólidos y consistentes, lo que respalda su elección como la solución óptima para la tarea de predicción de la severidad de los reclamos en este contexto específico.

Modelos No Supervisados

La proyección en PCA bidimensional es una técnica que nos permite visualizar datos de alta dimensionalidad en un plano de dos dimensiones. En este proceso, descomponemos las características originales en dos componentes principales, que son combinaciones lineales de las características originales. Estas componentes capturan la mayor varianza en los datos, permitiéndonos reducir la complejidad y representar la información de manera más manejable.

Al realizar esta proyección, obtenemos un gráfico de dispersión en 2D que nos brinda una visión simplificada pero informativa de la distribución de los puntos de datos en el espacio. Este enfoque es valioso para identificar patrones, tendencias o agrupamientos en los datos, lo que puede facilitar la interpretación y el análisis.

En resumen, al aplicar PCA para la proyección bidimensional, estamos simplificando la representación de los datos mientras intentamos retener la mayor cantidad posible de información relevante, lo que nos permite visualizar de manera efectiva la estructura subyacente de los datos en un espacio de menor dimensión.


Conjuntos "de entrenamiento" y "de prueba" transformados mediante PCA
Componente Principal 2
Componente Principal 1
80
60
40
20
0
−20
−40
−60
−150
−100
−50
0
50
100
150
200
1.0
0.8
0.6
0.4
0.2
0.0

Resultados

En resumen, al analizar el impacto de la variación en la profundidad y el parámetro min_samples_leaf en el rendimiento del modelo Random Forest, hemos observado que estos ajustes no siempre conducen a un incremento uniforme en la efectividad del modelo. De manera interesante, en algunos casos, el aumento en la profundidad no se traduce en una mejora constante, y la curva puede incluso disminuir.

Este fenómeno es comprensible al considerar que el parámetro min_samples_leaf establece la cantidad mínima de hojas que debe haber en cada nodo hoja, lo que puede afectar la complejidad y la generalización del modelo. Después de una evaluación minuciosa, se ha determinado que mantener el valor predeterminado para este parámetro es la elección más acertada.

Como conclusión final, el modelo seleccionado para su implementación es "model_1n_md" del Random Forest, ya que ha demostrado ser el más efectivo y ha producido resultados destacados en términos de rendimiento, según lo evidenciado en los análisis gráficos realizados. Este modelo específico se considera la opción óptima para abordar la tarea de predicción en este contexto.