Autor(es):

- Daniela Chacón – 201910858
- Esteban Ortiz - 201913613
- Manuel Porras - 201913911
- Angie Rincón - 201114323

Fecha: Noviembre 6 de 2023

# PROYECTO FINAL – SEGUNDA ENTREGA

- **OBJETIVOS**
  - Finalizar la actividad de entendimiento de los datos aplicando diferentes técnicas pertinentes para la solución planteada.
  - Realizar la preparación y limpieza de datos requerida para la construcción del modelo de machine learning y/o dashboard planteado.
  - Construir una primera versión del producto de datos y realizar una primera evaluación de resultados.
- **ACTIVIDADES DEL SPRINT Y ENTREGABLES:**

## 1 [35%] PREPARACIÓN DE DATOS

### 1.1 Descripción del proceso

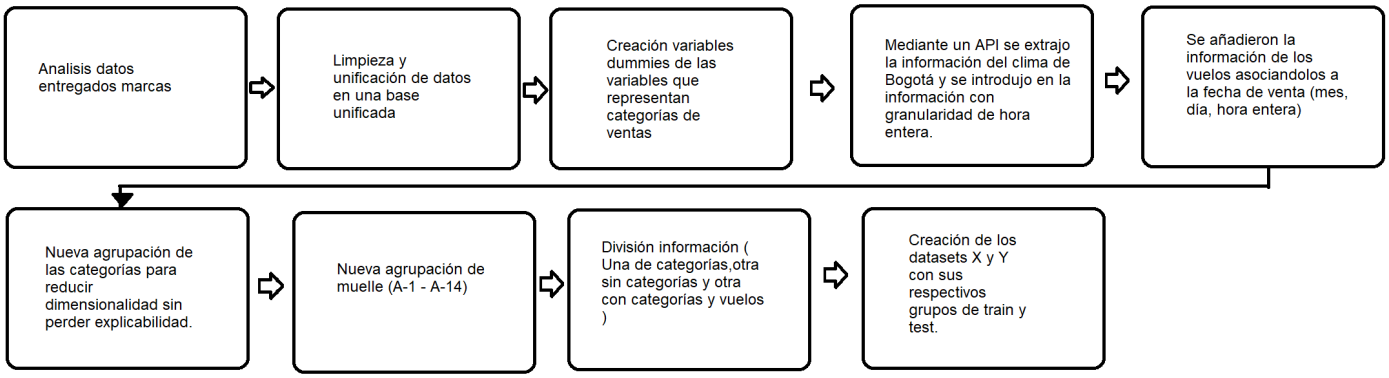



*Figura 1. Diagrama de bloques procesamiento de datos.*

**Análisis datos entregados de las marcas:** Se miró la estructura que tenían los datos de las 10 marcas que entregaron información y se identificó que había columnas que tenían nombres distintos pero sus datos significaban lo mismo por ejemplo Ubicación, muelle y lugar de venta. Se identificaron que había marcas que no tenían la misma granularidad y agrupación de sus ventas, es decir, había marcas que tenían registrada todas las ventas sin agrupaciones, otras que tenían las ventas agrupadas por minuto y la mayoría la tenían agrupada por hora entera. Además, había unas 4 marcas que tenían información de categorías solamente, las demás no tenían información al respecto. También, había marcas que tenían ciertas columnas que no tenían otras pero que se podían generar como, por ejemplo: hora_entera, mes_string, etc. Finalmente, al ver tanta disparidad de nombres de columnas se decidió cambiar todos los nombres de las columnas quitando las tildes, espacios y mayúsculas de las mismas.

**Limpieza y unificación de datos en una base unificada:** Se crea una base de datos con todos los datos de las 10 marcas procesados y organizados según lo dicho en el inciso de análisis de los datos entregados esto para tener un mayor control y una

facilidad de cara al procesamiento que se describirá en los siguientes pasos. Además, se eliminaron los datos con ventas negativas y escogimos una temporalidad de ventas de solo los años 2022 y 2023.

**Creaciones variables dummies:** Se identificó la necesidad de crear columnas binarias que representarán clases para las columnas de mes_string, categoría, tienda y marca. Esto con la finalidad de tener una explicabilidad de estas variables más clara cuando se cree la regresión respectiva.

**Extracción información clima por día y hora mediante API:** Mediante la API de la página: https://www.worldweatheronline.com/bogota-weather-history/cundinamarca/co.aspx se extrajo la información del clima de Bogotá sobre toda la línea de tiempo que nos interesa sobre los datos que es todo el año 2022 y los datos que tenemos del 2023. Este proceso se realizó con una granularidad igual que los datos de ventas (sobre hora_entera).

**Información de vuelos internacionales:** Teniendo en cuenta que los datos de ventas están sobre el muelle internacional se decidió relacionar esta información sobre los datos de vuelos internacionales que ocurrieron en la misma temporalidad de las ventas agrupándolos por categorías dummies en una granularidad mínima de hora_entera. Se tomaron en cuenta las variables del destino, el tipo de aerolínea, el día de la semana, y la aerolínea; variables que fueron convertidas a dummy para el procesamiento de los modelos.

**Nueva agrupación de categorías:** Se crearon nuevas categorías agrupando a las categorías mucho más específicos de la iteración anterior :

```
dataframe_unificado_final['viaje_ejecutivo_negocio'] =
dataframe_unificado_final['tabaco_alcohol'] = dataframe_
dataframe_unificado_final= dataframe_unificado_final.dr
dataframe_unificado_final['accesorios'] = dataframe_uni
dataframe_unificado_final['regalos_obsequios'] = datafr
dataframe_unificado_final['artesanias']= dataframe_unif
dataframe_unificado_final['calzado'] = dataframe_unific
dataframe_unificado_final['accesorios_de_mano']= datafr
dataframe_unificado_final['accesorios_belleza_cuidado']
dataframe_unificado_final['prendas_vestir'] =  datafram
dataframe_unificado_final['prendas_vestir_cabeza'] = da
dataframe_unificado_final['lujo'] = dataframe_unificado
dataframe_unificado_final['marroquineria'] = dataframe_
dataframe_unificado_final['cafe'] = dataframe_unificado
dataframe_unificado_final['tecnologia'] = dataframe_uni
dataframe_unificado_final['libros_revistas_papel']= dat
dataframe_unificado_final['cosas_hogar']= dataframe_uni
dataframe_unificado_final['comida_galguerias']= datafra
```

*Figura 2. Nuevas categorías.*

**Nueva agrupación de muelle:** Se crearon las siguientes agrupaciones de acuerdo al lugar espacial de las tiendas dentro del puente internacional con las siguientes denominaciones: A10-A12, A6-A8, A2-A4, A4, A5-A6 y A8-A10.

**División información:** Se divide la información entre dos datasets: Uno que contiene solo la información de ventas que tiene categorías y otro que tiene que solo la información de las ventas.

**Creación de los datasets X y Y:** Se divide la información entre dos datasets X y Y:

```
X=dataframe_unificado_final_solo_marca_categorias.drop(columns=['unidades','transacciones','ventas','muelle
','fecha_de_venta','mes_string','tienda','marca','descripcion','objeto1','objeto2','objeto3','objeto4','obj
eto5','tipo_de_tienda','ubicacion_esp','ventas_sin_iva','precio_bruto_total','impuesto_total','descuento_to
tal','tienda a','tienda b','personas_que_ingresan'])
```

```
y = dataframe_unificado_final_solo_marca_categorias['ventas']
```

Después se dejan los datos del año 2022 para los datasets del training y los del año 2023 para los datasets de prueba:

```
X_train = X[X['anio_entero']==2022]
X_test = X[X['anio_entero']==2023]
y_train = y[X['anio_entero']==2022]
y_test = y[X['anio_entero']==2023]
```

## 1.2 Antes y después de los datos

Datos unificados primera versión:

```
1  dataframe_unificado.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 246221 entries, 0 to 2032
Columns: 244 entries, fecha_de_venta to categoria 99
dtypes: datetime64[ns](1), float64(232), object(11)
memory usage: 460.2+ MB
```

*Figura 3. Primera versión datos unificados.*

Se tienen un total de 246221 filas con un total de 244 columnas. Con muchos problemas de valores nulos entre las columnas.

Datos unificados última versión:

```
1  dataframe_unificado_final.shape
```

```
(240436, 196)
```

*Figura 4. Última versión de datos unificados.*

Finalmente, se tienen un total de 240436 filas con 196 columnas.

# 2 [10%] ESTRATEGIA DE VALIDACIÓN Y SELECCIÓN DE MODELO

Para este apartado nos apoyamos en la librería de lazypredict que permite probar 42 modelos de regresión para ver cuál o cuáles son los que más se ajustan a nuestros datos e hicimos dos pruebas uno con el conjunto de datos que tiene la información de las categorías y otro con el que no.

```python
# Lista de modelos a excluir
modelos_a_excluir = ['SVR', 'NuSVR', 'GaussianProcessRegressor']
reg = LazyRegressor(verbose=0, ignore_warnings=False, custom_metric=None)
models, predictions = reg.fit(X_train, X_test, y_train, y_test)

print(models)
```

*Figura 5. Uso de la librería lazypredict y la clase LazyRegressor.*

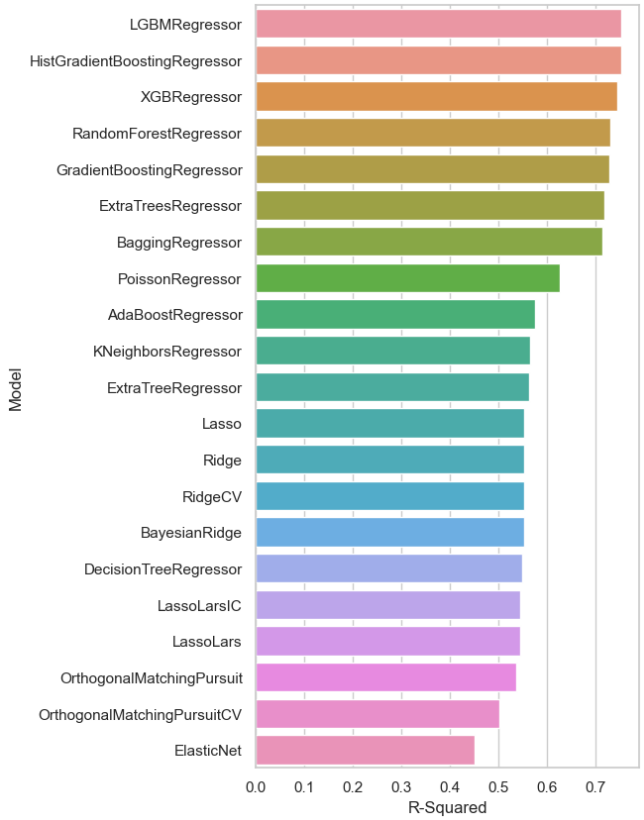



*Figura 6. Puntaje R2 modelos de regresión. (Con categorías).*

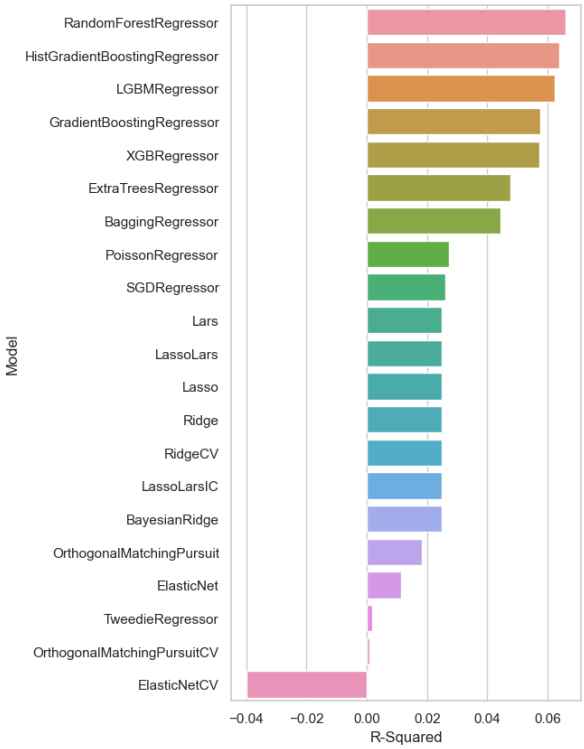



*Figura 7. Puntaje R2 de los modelos de regresión. (Sin categorías).*

De acuerdo a las figuras 6 y 7, se escogen los mejores 3 modelos para cada sub-dataset para desarrollar el siguiente punto de construcción de modelo.

# 3 [20%] CONSTRUCCIÓN DEL MODELO

Para cada modelo, se han ajustado los hiperparámetros para maximizar la métrica R ^ 2 a través de una validación cruzada con 5 folds para los modelos (excepto en Random Forest Regressor para reducir la complejidad).

## 3.1 Modelos con categorías:

- **LightGBM Regressor:**
  - **Hiperparámetros del Grid Search:**

```
param_grid = {
    'num_leaves': [31, 50, 70],
    'reg_alpha': [0.1, 0.5],
    'min_data_in_leaf': [30, 50, 100],
    'learning_rate': [0.01, 0.05, 0.1],
    'max_depth': [-1, 5, 10]
}

lgbm = lgb.LGBMRegressor()

grid_search = GridSearchCV(estimator=lgbm, param_grid=param_grid, cv=5, scoring='r2', verbose=1, n_jobs=-1)
```

- **Mejores hiperparámetros encontrados:**
  - learning_rate: 0.05
  - max_depth: -1 (sin límite)
  - min_data_in_leaf: 100
  - num_leaves: 70
  - reg_alpha: 0.5

- **Histogram-based Gradient Boosting Regressor**

  - **Hiperparámetros del Grid Search:**

```
param_grid = {
    'max_iter': [100, 200],
    'max_depth': [3, 5, 10],
    'min_samples_leaf': [20, 40],
    'learning_rate': [0.01, 0.1, 0.2],
    'l2_regularization': [0.0, 0.1]
}

hist_gbr = HistGradientBoostingRegressor()

grid_search = GridSearchCV(estimator=hist_gbr, param_grid=param_grid, cv=5, scoring='r2', verbose=1, n_jobs=-1)
```

  - **Mejores hiperparámetros encontrados:**
    - learning_rate: 0.05
    - max_depth: -1 (sin límite)
    - min_data_in_leaf: 100
    - num_leaves: 70
    - reg_alpha: 0.5

- **XGBoost Regressor**

  - **Hiperparámetros del Grid Search:**

```
param_grid = {
    'n_estimators': [100, 200, 300],
    'learning_rate': [0.01, 0.1, 0.2],
    'max_depth': [3, 6, 9],
    'colsample_bytree': [0.5, 0.7],
    'subsample': [0.6, 0.8, 1.0],
    'gamma': [0, 0.1, 0.2]
}

# Inicializar el modelo XGBRegressor
xgb_regressor = XGBRegressor(objective='reg:squarederror')

# Configurar GridSearchCV
grid_search = GridSearchCV(estimator=xgb_regressor, param_grid=param_grid,
                           scoring='r2', cv=5, verbose=1, n_jobs=-1)
```

- **Mejores hiperparámetros encontrados:**
  - **colsample_bytree: 0.7**
  - **gamma: 0**
  - **learning_rate: 0.01**
  - **max_depth: 9**
  - **n_estimators: 100**
  - **subsample: 0.8**

## 3.2 Modelos sin categorías:

- **RandomForest Regressor**

  - **Hiperparámetros del Grid Search:**

```
param_grid = {
    'n_estimators': [100, 200],  # Menos árboles
    'max_depth': [10, 20, None],  # Profundidad limitada y una opción sin límite
    'min_samples_split': [2, 5],  # Menos opciones para la cantidad mínima de muestras para dividir
    'min_samples_leaf': [1, 2],   # Menos opciones para la cantidad mínima de muestras en una hoja
    'max_features': ['auto', 'sqrt'],  # Opciones para el número de características a considerar al buscar la mejor división
}

# Inicializar el modelo RandomForestRegressor
rf_regressor = RandomForestRegressor(random_state=0)

# Configurar GridSearchCV
grid_search = GridSearchCV(estimator=rf_regressor, param_grid=param_grid,
                           scoring='r2', cv=3, verbose=1, n_jobs=-1)
```

  - **Mejores hiperparámetros encontrados:**
    - max_depth: 5
    - min_samples_leaf: 1
    - n_estimators: 200
    - min_samples_split: 2
    - max_features: 'auto'

- **Histogram-based Gradient Boosting Regressor**

  - **Hiperparámetros del Grid Search:**

```
# Definir una cuadrícula de parámetros más pequeña y menos exhaustiva
param_grid = {
    'max_iter': [100, 200],
    'max_depth': [3, 5, 10],
    'min_samples_leaf': [20, 40],
    'learning_rate': [0.01, 0.1, 0.2],
    'l2_regularization': [0.0, 0.1]
}

hist_gbr = HistGradientBoostingRegressor()

grid_search = GridSearchCV(estimator=hist_gbr, param_grid=param_grid, cv=5, scoring='r2', verbose=1, n_jobs=-1)
```

- **Mejores hiperparámetros encontrados:**
  - l2_regularization: 0.1
  - learning_rate: 0.01
  - max_depth: 3
  - max_iter: 100
  - min_samples_leaf: 40

- **LightGBM Regressor**

  - **Hiperparámetros del Grid Search:**

```
param_grid = {
    'num_leaves': [31, 50, 70],
    'reg_alpha': [0.1, 0.5],
    'min_data_in_leaf': [30, 50, 100],
    'learning_rate': [0.01, 0.05, 0.1],
    'max_depth': [-1, 5, 10]
}

lgbm = lgb.LGBMRegressor()

grid_search = GridSearchCV(estimator=lgbm, param_grid=param_grid, cv=5, scoring='r2', verbose=1, n_jobs=-1)
```

  - **Mejores hiperparámetros encontrados:**
    - learning_rate: 0.01
    - max_depth: 5
    - min_data_in_leaf: 30
    - num_leaves: 50
    - reg_alpha: 0.5

## 3.3 Modelo categoría con información de vuelos

```
regression = LinearRegression()
regression.fit(X_train, y_train)
```

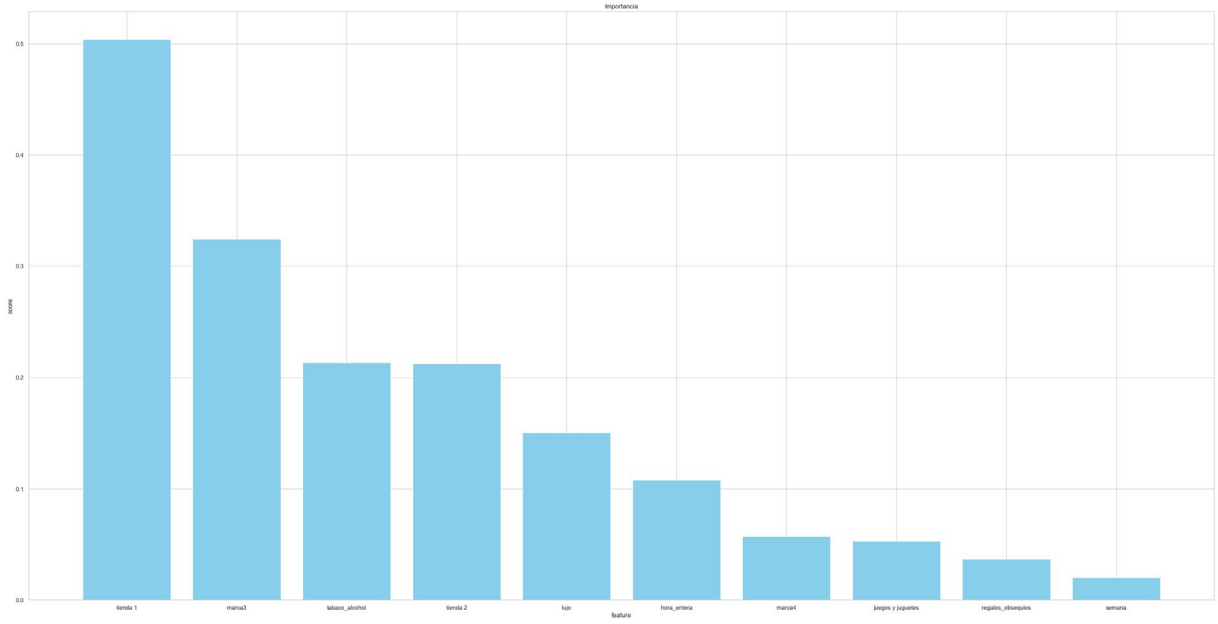
# 4 [20%] EVALUACIÓN DEL MODELO

## 4.1 Modelos con categorias

| Modelo | MAE Train | MAE Test | RMSE Train | RMSE Test | R2 Train | R2 Test |
|---|---|---|---|---|---|---|
| LGBM | 7,862,636.43 | 11,218,845.19 | 15,646,572.29 | 20,700,288.21 | 0.84 | 0.75 |
| HISTGBR | 8,111,847.00 | 11,220,512.78 | 16,018,202.92 | 20,371,537.14 | 0.83 | **0.76** |

| | | | | | | |
|---|---|---|---|---|---|---|
| XGBR | 10,261,177.53 | 15,207,824.47 | 23,283,554.16 | 29,066,946.27 | 0.65 | 0.51 |

Importancia de los atributos del mejor modelo:




## 4.2 Modelos sin categorias

| Modelo | MAE Train | MAE Test | RMSE Train | RMSE Test | R2 Train | R2 Test |
|---|---|---|---|---|---|---|
| RFRegressor | 14,924,398.46 | 16,056,810.96 | 28,082,341.46 | 31,236,064.37 | 0.093 | **0.041** |
| HISTGBR | 15,497,472.35 | 16,409,970.83 | 28,641,938.71 | 31,416,677.50 | 0.057 | 0.030 |
| LGBM | 15,330,679.43 | 16,202,557.48 | 28,468,636.89 | 31,341,046.67 | 0.068 | 0.034 |

## 4.3 Modelo con categorías e información de vuelos

| Modelo | MAE Train | MAE Test | RMSE Train | RMSE Test | R2 Train | R2 Test |
|---|---|---|---|---|---|---|
| Regresión lineal | 17227486.6 | 16,056,810.96 | 28,082,341.46 | 31,236,064.37 | 0.093 | **0.041** |

## 4.4 Evaluación de los resultados

En los modelos con categorías:

- El Histogram-based Gradient Boosting Regressor (HISTGBR) obtiene el mejor rendimiento en el conjunto de prueba con un R2 de 0.76, lo cual indica que el modelo puede explicar aproximadamente el 75.7% de la varianza en los datos de prueba. Aunque tiene un error absoluto medio (MAE) y un error cuadrático medio (RMSE) ligeramente más altos en la fase de prueba en comparación con el LGBM, su mayor R2 en la fase de prueba sugiere una mejor generalización. De la misma manera sus atributos más importantes son: tienda_1, marca3, tabaco_alcohol, tienda_2 y lujo.
- El LightGBM (LGBM) tiene un buen rendimiento en el conjunto de entrenamiento, pero muestra una caída en la puntuación R2 cuando se evalúa en el conjunto de prueba. Esto puede ser indicativo de un ligero sobreajuste al conjunto de entrenamiento.
- El XGBoost Regressor (XGBR) tiene la puntuación más baja de R2 en el conjunto de prueba, lo que sugiere que tiene un poder predictivo inferior en comparación con los otros modelos con categorías.

En los modelos sin categorías:

- Todos los modelos tienen un rendimiento muy bajo, con puntuaciones R2 que indican que los modelos no están capturando bien la variabilidad de los datos. El Random Forest Regressor tiene el R2 más alto en el conjunto de prueba de los tres, pero sigue siendo muy bajo (0.041), lo que indica que prácticamente no tiene capacidad predictiva.

# 5 [15%] CONCLUSIONES

## 5.1 Dificultades Encaradas:

Durante el desarrollo del proyecto, nos enfrentamos a varias dificultades, entre las cuales las más destacadas fueron:

- Overfitting en Modelos con Categorías: Se observó que algunos modelos, como el LightGBM, mostraban un alto rendimiento en los datos de entrenamiento, pero una disminución significativa en los datos de prueba, lo que sugiere una especialización excesiva en los datos de entrenamiento y una generalización insuficiente.
- Bajo Rendimiento en Modelos sin Categorías: Los modelos sin categorías mostraron una capacidad predictiva extremadamente baja, indicada por puntuaciones R2 cercanas a cero, lo que implica que no fueron capaces de capturar la variabilidad de los datos y realizar predicciones fiables.

## 5.2 Estrategias de Mitigación:

Para abordar estas dificultades, proponemos las siguientes estrategias:

- Regularización y Ajuste de Hiperparámetros: Implementaremos técnicas de regularización y optimización de hiperparámetros para mejorar la generalización de los modelos.
- Análisis de Sesgo: Realizaremos un análisis para detectar y mitigar cualquier sesgo presente en los datos que pueda estar afectando el rendimiento del modelo.

## 5.3 Condiciones de los Datos para Mejorar Resultados:

Se considera que los datos podrían mejorar en los siguientes aspectos:

- Reducción de Sesgo: Una selección y procesamiento de datos que busque activamente reducir sesgos puede resultar en modelos más justos y precisos.

## 5.4 Evaluación del Mejor Modelo:

El modelo más prometedor identificado hasta la fecha es el Histogram-based Gradient Boosting Regressor (HISTGBR) con categorías, el cual ha mostrado una capacidad considerable para explicar la varianza en los datos de prueba (R2 de 0.757).