Checkpoint 1 - Grupo 01

Análisis Exploratorio

El dataset en crudo que se nos otorga cuenta con más de 450 mil registros y 20 columnas.
Las características más destacables del mismo son:

- **property_price:** El precio de publicación es el dato que buscamos que nuestro modelo pueda predecir por lo que es necesario entrenarlo conociendo este dato. Es de tipo numérico.
- **property_currency:** Sirve para realizar el filtrado inicial. Es de tipo string. Una vez filtrado por este dato, la columna es descartable ya que tendría el mismo valor en todas las filas (“USD”).
- **operation:** Sirve para realizar el filtrado inicial. Es de tipo string. Una vez filtrado por este dato, la columna es descartable ya que tendría el mismo valor en todas las filas (“Venta”).
- **property_type:** Sirve para realizar el filtrado inicial. Determina qué es lo que está siendo publicado (Casa, local comercial, etc). Es de tipo string. Una vez filtrado nos quedan 3 tipos (Casa, PH, Departamento) los cuales pueden llegar a tener correlación con el precio.
- **Property_rooms** y **property_bedrooms:** Indican la cantidad de habitaciones y ambientes (respectivamente). Son de tipo numérico. Puede existir una relación entre estas cantidades y el precio de la publicación.
- **property_surface_total** y **property_surface_covered:** Indican los metros cuadrados del total de la propiedad y los cubiertos (respectivamente). Son de tipo numérico. Puede existir una relación entre estas cantidades y el precio de la publicación.

- **place_l2, place_l3, place_l4...:** Indican la ubicación de la propiedad. Sea provincia, localidad, barrio. Son de tipo string. Estos datos son de alto interés ya que los precios de las propiedades suelen estar altamente influenciados por la ubicación. Existe una grán proporción de datos nulos cuanto más específica es la ubicación.
- **property_title**: Indica el título con el cual se realizó la publicación. Este puede llegar a aportar información útil al momento de tomar decisiones sobre las otras columnas, ya que generalmente en cada registro aporta información sobre la cantidad de ambientes y/o locación. Es de tipo string.

Analizamos las columnas para filtrar según los criterios indicados buscando irregularidades que causen un filtrado menos preciso. Por ejemplo, que la columna **property_currency** (la cual tenemos que filtrar por el valor “USD”) no tenga algunos registros en mayúscula y otros en minúsculas, o no diga “Dólares” en lugar de “USD‘ y por ello no sea captada por el filtro. Ninguna de las columnas que son criterio de filtrado presentó problemas de este tipo.

Al filtrar obtenemos un dataframe de más de 94 mil registros. Este se divide en conjunto de entrenamiento y prueba en una proporción 80/20, teniendo como resultado 75 mil registros de entrenamiento y casi 19 mil de prueba.

Todo el análisis se realiza sobre el conjunto de entrenamiento.

Hipótesis:

- Las casas tienden a tener 1 ambiente más que el total de habitaciones (el living).
- Las coordenadas (latitud y longitud) son mucho más propensas a presentar un grave error de precisión a comparación del barrio.
- La superficie cubierta es menor o igual a la superficie total.
- Las propiedades con un exagerado número de habitaciones (mansiones por ejemplo) son un mercado paralelo y no rigen los mismos criterios que para la propiedad promedio.

Preprocesamiento de Datos

Tras filtrar el dataset según los criterios indicados, nuestro dataset presenta ciertas características:

1. **Columnas eliminadas:**
   a. place_l4: +96% nulls
   b. place_l5: 100% nulls
   c. place_l6: 100% nulls
   d. start_date: No aporta información
   e. end_date: No aporta información
   f. created_on: No aporta información
   g. property_currency: Todos los registros tienen el mismo valor
   h. operation: Todos los registros tienen el mismo valor
2. **Correlaciones:**
   a. Cantidad de habitaciones y cantidad de ambientes: 0.87. Esto confirma nuestra hipótesis sobre la tendencia de las propiedades de tener 1 ambiente más que la cantidad de habitaciones.
   b. Superficie cubierta y superficie total por tipo de propiedad:
      En PH's y casas la correlación es mayor a 0.9. En departamentos ronda el 0.55.
      En casi todos los barrios los departamentos tienen correlación +0.8, lo cual tiene mucho sentido ya que generalmente un departamento es en su mayoría superficie cubierta.
      Este dato nos es útil para imputar las filas donde uno de los datos es nulo y el otro no.

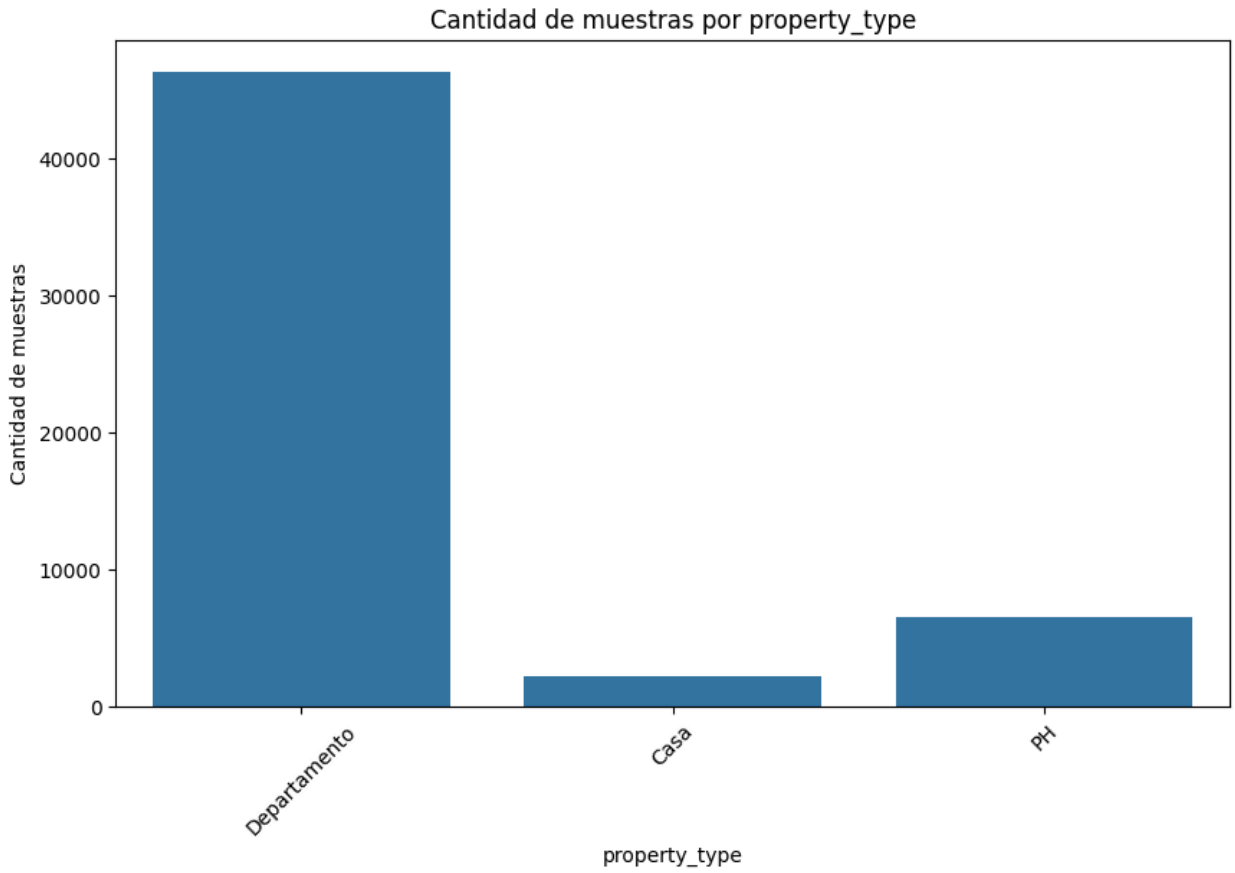
3. **Nuevas features:**
   a. total_m2_price: Precio por metro cuadrado de superficie total.
   b. covered_m2_price: Precio por metro cuadrado de superficie cubierta.
   c. Casa, departament y PH: Aplicando One Hot Encoding reemplazamos la columna "property_type" por estas 3 nuevas columnas dummies

   d. Palermo, Belgrano, Caballito, Otros: Aplicando One Hot Encoding reemplazar la columna "place_l3" por dummies de las ubicaciones con mayor presencia. Útil para el análisis de grupos.

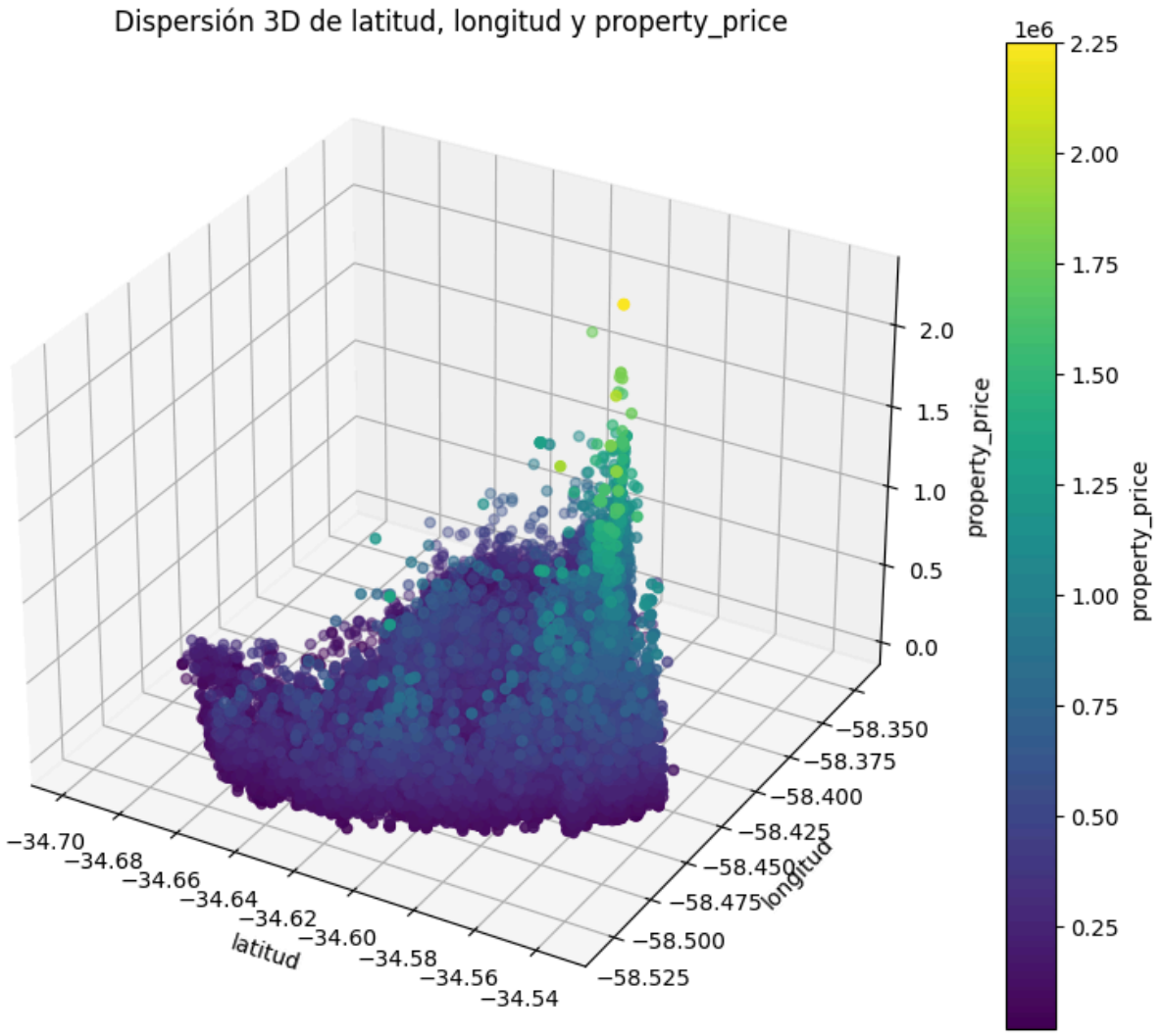
4. **Valores atípicos:**
    a. Superficie total menor a superficie cubierta: Este es un absurdo en el cual optamos por invertir los valores, asignando la superficie cubierta como superficie total y viceversa. Esto incrementó en grán medida la correlación entre estos datos para propiedades del tipo Casa.
    b. Superficie total exageradamente grande: Haciendo un gráfico de dispersión entre la superficie total y la cubierta se ve un claro cúmulo en una esquina del gráfico. Eliminamos las publicaciones cuya superficie total es desde 7 veces mayor a la mediana.
    c. Precio por metro cuadrado:
    En algunas propiedades obtuvimos valores por metro cuadrado irrisorios, de más de 5 veces la mediana inclusive. Las eliminamos.

    d. Cantidad de habitaciones y de ambientes:
    En aquellos casos donde existen más habitaciones que ambientes asignamos el valor de ambientes al de habitación.
    Eliminamos aquellos registros donde la cantidad de habitaciones es más de 7 veces la mediana.

Visualizaciones



En el gráfico visualizamos la cantidad de muestras por “propety_type” la cual nos permite apreciar que en nuestro dataset poseemos una mayor distribución de Departamentos, lo cual es lo que uno esperaría de Capital Federal. Se eligió mostrar este gráfico ya que nos permite observar si un dato predomina sobre los otros y tenerlo más en cuenta por su peso



El gráfico permite visualizar la dispersión dimensional de las publicaciones gracias a los datos de latitud y longitud, así como también. Vemos un cúmulo importante de de datos con un amplio rango de precios en el sector de color más claro del gráfico. Mientras ciertas zonas mantienen un rango relativamente reducido en sus precios (eje vertical), este cúmulo más claro indica que esa ubicación (Palermo, seguido por Belgrano y Caballito) no tiene una fuerte tendencia del precio basado únicamente en su ubicación ya que el abanico de valores es más amplio.

Clustering

Buscamos con el SSE que cantidad de clusters sería apropiado para el dataframe dado, y hemos llegado a la respuesta de que el mayor silhouette score es para dos grupos. También con los gráficos observamos que el criterio que utilizó el algoritmo de KMeans para separar dichos clusters está relacionado al valor de la vivienda.

Estado de Avance

1. Análisis Exploratorio y Preprocesamiento de Datos

Porcentaje de Avance: 100%/100%

Hemos terminado con esta etapa, lo único que dejamos con cierta duda, es respecto a aquellas filas que tenían un *place_l3* no nulo, pero tanto latitud como longitud sí lo eran. Probaremos cómo funcionan los modelos, y en caso de necesitarlo, buscaremos ubicar a estas propiedades en los centros de los polígonos del barrio correspondiente al *place_l3*.

2. Agrupamiento

Porcentaje de Avance: 100%/100%

Tiempo dedicado

Integrante	Tarea	Prom. Hs Semana
Taiel Molina	Exploración De Datos Visualización de Datos Imputación de Datos Agrupamiento	7
Ignacio Fernández	Análisis de Correlaciones Armado de Reporte	7

	Análisis de outliers Imputación de Datos	
Sebastián Vera	Visualización de Datos Armado de Reporte	4

Checkpoint 2 - Grupo 01

Clasificación

Mencionar cuál es la alternativa seleccionada para construir la variable “tipo_precio” justificando su elección. Mostrar en un mapa de CABA los avisos coloreados por tipo_precio ¿Qué diferencias o similitudes encuentran con el agrupamiento realizado por K-Means con 3 grupos?
Si llegaron a entrenar alguno de los modelos, mencionar cuáles y qué métricas obtuvieron en test y si realizaron nuevas transformaciones sobre los datos (encoding, normalización, etc) completando los ítems a y b:

La variable *tipo_precio* fue construida con la alternativa de los cuartiles (sin diferenciación entre tipo de propiedad), y fue elegida así ya que probamos los modelos con las otras dos también, y respecto a la primera (separando en tercios) la diferencia era poca, pero era superior la elegida, y respecto a la de cuartiles pero separando por tipo de propiedad, daban mucho más bajo los scores.

Respecto a las similitudes con K-Means, observamos que los agrupamientos son similares, y se puede observar claramente en el mapa ploteado de CABA. Acerca de K-Means y en base a qué se habían formado los grupos, ya lo hemos comentado más arriba (en el CHP1), pero recordándolo habían sido conformados en base al precio de la propiedad; por lo cual tiene todo el sentido que, siendo que *tipo_precio* depende del precio por metro cuadrado (lo cual tiene una relación directa con el precio de la propiedad), ambos mapas sean similares

a. Construcción del modelo

Arbol de Decision

- ¿Optimizaron hiperparámetros? ¿Cuáles?

  Se buscaron optimizar los hiperparámetros de: Criterion, min_samples_split, ccp_alpha y el max_depth

- ¿Utilizaron K-fold Cross Validation?¿Cuántos folds utilizaron?

  Se uso RandomizedSearchCV con 5 folds

- ¿Qué métrica utilizaron para buscar los hiperparámetros?

La métrica que se usó para buscar los hiperparámetros es accuracy

- Añadir imagen del árbol generado e incluir descripciones que consideren adecuadas para entender el mismo. Si es muy extenso mostrar una porción representativa.

Random Forest

- ¿Optimizaron hiperparámetros? ¿Cuáles?

  Se buscaron optimizar los parámetros de: Criterion, min_samples_leaf, min_samples_split y n_estimators

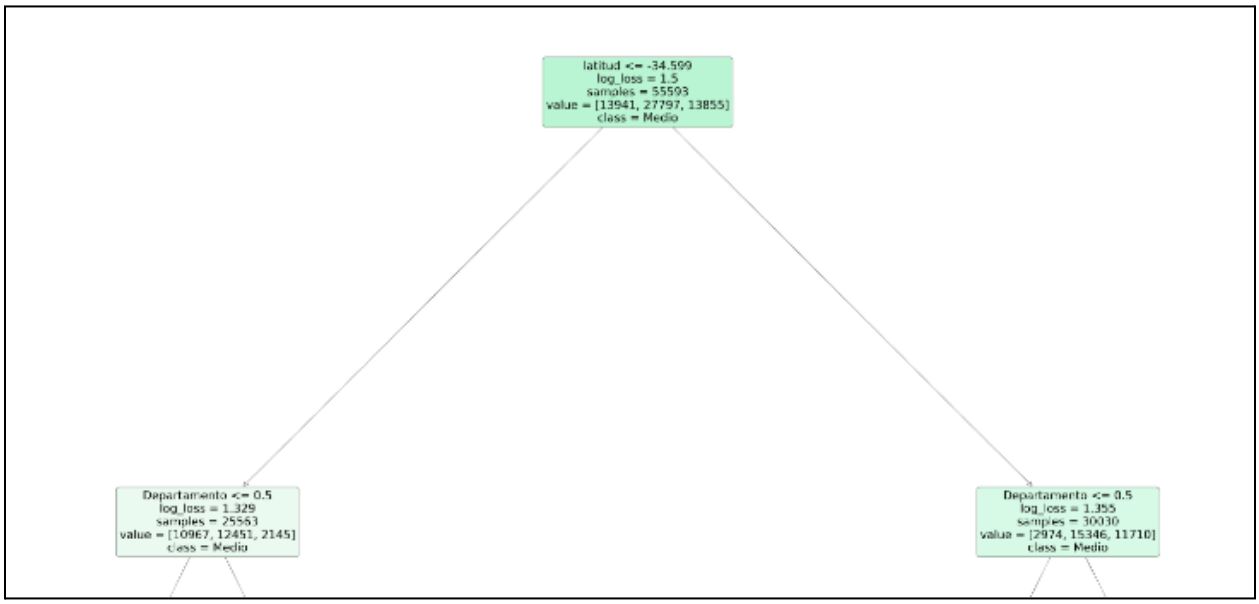
- ¿Utilizaron K-fold Cross Validation?¿Cuántos folds utilizaron?

  Se usó RandomizedSearchCV con 5 folds

- ¿Qué métrica utilizaron para buscar los hiperparámetros?

  La métrica que se usó para buscar hiperparámetros es accuracy

Mostrar la conformación final de uno de los árboles generados. Si es muy extenso mostrar una porción representativa y explicar las primeras reglas.



Modelo a Elección

- ¿Optimizaron hiperparámetros? ¿Cuáles?

Se buscaron optimizar los parámetros de: weights, algorithm y p

- ¿Utilizaron K-fold Cross Validation?¿Cuántos folds utilizaron?

  Se usó RandomizedSearchCV con 10 folds

- ¿Qué métrica utilizaron para buscar los hiperparámetros?

  La metrica que se usó para buscar hiperparámetros es accuracy

- Mostrar la conformación final de uno de los árboles generados. Si es muy extenso mostrar una porción representativa y explicar las primeras reglas.

b. Cuadro de Resultados

Realizar un cuadro de resultados comparando los modelos que entrenaron (entre ellos debe figurar cuál es el que seleccionaron como mejor predictor).

Medidas de rendimiento en el conjunto de TEST:

- F1
- Precision
- Recall
- Accuracy
- XXX: si seleccionaron alguna métrica adicional...
-

Confeccionar el siguiente cuadro con esta información:

Modelo	F1-Test	Precision Test	Recall Test	Accuracy Test
Arbol de Decision	0.7	0.7	0.7	0.7
Random Forest	0.76	0.76	0.76	0.75
KNN	0.69	0.69	0.70	0.69

En cada caso ¿Cómo resultó la performance respecto al set de entrenamiento?

Modelo	F1-Test	Precision Test	Recall Test	Accuracy Test
Arbol de Decision	0.88	0.89	0.88	0.88

Random Forest	0.98	0.98	0.98	0.98
KNN	0.98	0.98	0.98	0.98

Nota: las medidas indicadas en los cuadros previos son los weighted avg de cada métrica.

Nota: indicar brevemente en qué consiste cada modelo de la tabla, por ejemplo

Arbol de decision: `{'min_samples_split': 11, 'max_depth': 27, 'criterion': 'entropy', 'ccp_alpha': 0.0}`

Random Forest: {'n_estimators': 10, 'min_samples_split': 2, 'min_samples_leaf': 1, 'criterion': 'gini'}

KNN: {'weights': 'distance', 'p': 1, 'algorithm': 'brute'}

Regresión

a. Construcción del modelo

Se realizaron ejecuciones de los modelos KNN, XGBoost y LightGBM utilizando *RandomizedSearchCV* para optimizar la búsqueda de parámetros dividiendo el dataframe de entrenamiento en 5 folds.

Los dataframes puestos a prueba y evaluados fueron previamente normalizados mediante *StandardScaler*. Esta normalización mejoró hasta en un 8% los resultados de la validación cruzada.

Esta evaluación se repitió con 3 métodos de scoring diferentes: MSE, RMSE y R2. Los mejores resultados de cada prueba por la validación cruzada utilizando los mejores parámetros se muestran en el apartado **Cuadro de resultados**.

KNN: El algoritmo, aplicado a un problema de regresión, busca predecir el valor de la variable objetivo en base a las propiedades de sus vecinos más cercanos en un espacio de características.

Algunos de los parámetros con los que se evaluó el modelo son los siguientes:

- **N_neighbors:** Cantidad de vecinos por cada “centroide”. Disminuir mucho el valor puede llevar al overfitting, mientras que un aumento desmedido tiene el efecto contrario.
- **Weights:** Determina la forma en la que influyen las distancias entre vecinos en el cálculo.
- **Algorithm:** Determina qué algoritmo se utiliza para identificar los nodos vecinos.
- **Metric:** Determina cómo se calcula la distancia entre vecinos. Algunos ejemplos son la distancia euclidia o la manhattan.

XGBoost: El algoritmo utiliza una jerarquía de árboles de decisión donde cada árbol hijo aprende del error de su árbol padre para reforzar ese punto. Es un modelo muy eficiente para el trabajo con grandes cargas de datos.

Algunos de los parámetros con los que se evaluó el modelo son los siguientes:

- **N_estimators:** Número de árboles utilizados. Similar comportamiento al n_neighbors de KNN.
- **Max_depth:** Profundidad máxima de cada árbol.
- **Learning_rate:** Indica la taza con la que el modelo aprende los datos. Disminuirla implica un rendimiento más lento pero disminuye el riesgo de overfitting.
- **Reg_alpha:** Regula la complejidad del modelo. Aumentar el reg_alpha implica una menor influencia de las variables menos importantes, lo cual puede llevar a un underfitting.

LightGBM: Similar a XGBoost, este modelo se distingue por la forma en la que distribuye los datos de mayor peso en los árboles hijos. El modelo busca distribuir el peso de todas las variables evitando la concentración de XGBoost en las más relevantes.

Algunos de los parámetros con los que se evaluó el modelo son los siguientes:

- **border_count:** Determina el número de cortes máximo por cada característica del dataset. Aumentar este valor aumenta el tiempo de cómputo pero también aumenta la precisión.
- **begging_temperature:** .Regula la tasa de variabilidad en la selección de los datos que entrena a cada árbol del modelo. Es una propiedad de LightGBM e implementaciones del mismo, como CatBoost.
- **random_strenght:** Controla el peso de la aleatoriedad en el entrenamiento de los árboles. Aumentar el valor ayuda a ganar variabilidad y lograr un mejor rendimiento general.

b. Cuadro de Resultados

Modelo	**MSE**	**RMSE**	**R2**
KNN	-5761099671	-74518.06	0.8382
XGBoost	-6153214271	-77324.06	0.8319
LightGBM	-6082288721	-76924.30	0.8336

Para los métodos de scoring MSE y RMSE el valor más bajo es el mejor, mientras que en R2 los es el más próximo a 1.

Luego se evalúa cada uno de los modelos utilizando los mejores parámetros obtenidos en el paso anterior. Se evalúan los resultados con el conjunto de prueba y de entrenamiento para descartar casos de overfitting.
La métrica para puntuar este paso es R2.

Modelo	Train	Test	Observaciones
KNN	0.998	0.885	Overfit en entrenamiento
XGBoost	0.903	0.823	Menor puntaje en test
LightGBM	0.888	0.869	Valores similares en entrenamiento y test

Esta prueba muestra una muy buena respuesta de los modelos al conjunto de test, incluso cuando parece existir un overfit con el de entrenamiento, como es el caso de KNN.
Por la cantidad e intensidad de las pruebas hechas, y por el funcionamiento de KNN, los tiempos de espera en su ejecución son demasiado largos. Por ello en la notebook dejamos comentada su ejecución, pero con los resultados en pantalla.
Dado que el modelo LightGBM obtiene un puntaje de test bastante similar al más alto (KNN), pero alejándose del overfit a los datos de prueba, optamos por avanzar con el mismo como modelo definitivo.
Además, vemos que la diferencia de score entre train y test de este modelo es muy pequeña, lo cual es un indicio de que el modelo es poco susceptible a los cambios en el dataframe a procesar.

Habiendo elegido el modelo, vamos a hacer una última optimización volviendo a evaluarlo pero con mayor cantidad de parámetros y más valores por cada uno.

Intento	Train	Test
Antes	0.888	0.869
Despues	0.908	0.866

Tras esto se puede apreciar un aumento en el score con el set de entrenamiento, y una ligera disminución en el score del set de evaluación. Claramente esto indica una tendencia al overfitting, por lo que vamos a conservar los parámetros de la evaluación original.

Mejores parámetros utilizados en LightGBM:

- random_strengt: 0.5
- learning_rate: 0.1
- l2_leaf_reg: 7
- depth: 6

- border_count: 32
- bagging_temperature: 1.5

Estado de Avance

1. Análisis Exploratorio y Preprocesamiento de Datos

Porcentaje de Avance: 100%/100%

2. Agrupamiento

Porcentaje de Avance: 100%/100%

3. Clasificación

Porcentaje de Avance: 100%/100%

4. Regresión

Porcentaje de Avance: 100%/100%

Tiempo dedicado

Integrante	Tarea	Prom. Hs Semana
Taiel Molina	Clasificación (Random Forest) Regresión (KNN) Armado de Reporte	6
Ignacio Fernandez	Clasificación (KNN) Regresión (XGBoost y LightGBM) Armado de Reporte	5
Sebastián Vera	Clasificación (Árbol de Decisión) Clasificación (KNN) Armado de Reporte	5