Checkpoint 2 - Grupo 11

Análisis Exploratorio

El dataset original consta de unas 460154 registros y 20 columnas. Todos estos registros resultan ser anuncios de propiedades, ya sea en venta, alquiler o alquiler temporal, no solo en la Argentina, sino incluso en Uruguay y otros países. Además, hay anuncios no solo para casas, PHs y departamentos, sino también para lotes, locales comerciales, depósitos, oficinas, casas de campo, cocheras y otros.

Dentro de los features más destacables del dataset hallamos:

- `operation`: Detalla el tipo de operación bajo la cual está listada cada propiedad Es una variable categórica cuyo valores incluyen: venta, alquiler, alquiler temporal. En este caso particular, sólo nos interesaban las propiedades en venta.
- `property_price`: Indica el precio de una propiedad. Es el valor que deseamos predecir. Es una cuantitativa variable continua.
- `property_currency`: Detalla el tipo de moneda del precio de la publicación Es una variable categórica cuyos posibles valores incluyen: ARS, USD. Solo nos interesan las propiedades listadas en USD.
- `property_surface_total`: Indica la superficie total de la propiedad. Es una variable cuantitativa continua.
- `property_rooms`: Detalla la cantidad de habitaciones en la propiedad. Es una variable cuantitativa discreta.
- `place_l3`: Renombrada a “neighbourhood”, describe el barrio donde se encuentra la propiedad. Es la variable categórica que más llama la atención, ya que la información acerca de la ubicación suele ser de suma importancia cuando se busca comprar una propiedad.

Supuestos e hipótesis tomados:

- Los datos de superficie están en metros cuadrados.
- Los precios indicados en dólares efectivamente estaban en dólares.

Preprocesamiento de Datos

En el preprocesamiento de datos se decidió eliminar las columnas `place_l4`, `place_l5`, `place_l6` debido a que muy pocas de las propiedades de interés contaban con información alguna en estos campos. De igual forma, se decidió eliminar las columnas `place_l2`, `operation` y `property_currency` debido a que todas estas columnas quedaban solamente con registros de un mismo valor después de hacer el filtrado del dataset (Capital Federal, Venta y USD respectivamente).

Se detectó que las variables `property_rooms` y `property_bedrooms` tenían una correlación muy elevada entre ellas, de 0,87. Previo al análisis de outliers e imputación de datos faltantes, las otras variables que resultaron tener una correlación algo elevada son `property_surface_total` y `property_surface_covered`, con un valor de 0,6. Si bien luego de hacer el manejo de datos nulos y el análisis de outliers en estas dos últimas columnas, su correlación incrementó hasta 0,85, optamos por conservar ambas columnas porque a nivel de concepto, nos parece significativo hacer una distinción entre el área cubierta y total de una propiedad, por ejemplo, esto sería muy significativo cuando se compra una casa con patio.

Durante el tratamiento de los análisis nulos de las variables `property_surface_total` y `property_surface_covered` se experimentó utilizando un método de imputación basado en un modelo de regresión lineal que predijera el valor de esta última a partir de la superficie total y el número de cuartos. Para mejorar la correlación entre las variables se probó escalando las variables de superficie para minimizar la importancia de las diferencias entre las unidades de medida de las cantidad de cuartos y las superficies. Se terminó hallando que la mejor opción era aplicar el logaritmo a las superficies. Sin embargo, finalmente se terminó optando el método MICE por practicidad, ya que el modelo de regresión también requería de un análisis de nulos adicional para evitar logaritmos de valores nulos o ceros.

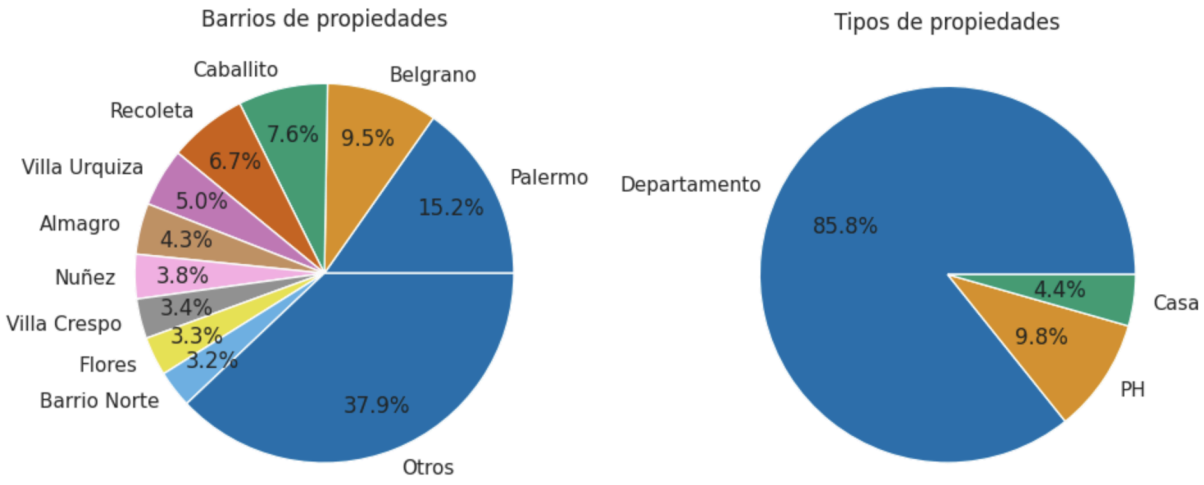
Se encontraron muchos valores atípicos sobre todo en los valores de las superficies, teniendo algunas superficies totales que eran menores que las superficies cubiertas, o valores que resultaban muy grandes y con poco sentido para la cantidad de habitaciones de algunas propiedades (ej. monoambientes con 51100m^2 de superficie

cubierta). Algunos se analizaron y se pudieron salvar corroborando con la información que nos proveían otras columnas como `property_title`, que mostraba una descripción de la propiedad.

Se utilizaron las técnicas de boxplots e IQRs para el análisis de los outliers univariados y scatterplots para el análisis multivariado de outliers particulares. Para la mayoría de variables correlacionadas, notamos que al tratar los outliers univariados de alguna de los variables, muchos de los outliers multivariados entre ambas quedaban resueltos automáticamente.

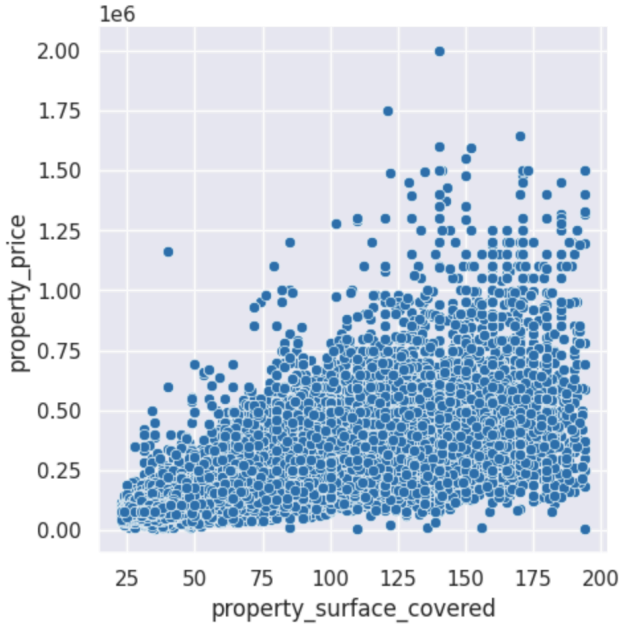
Visualizaciones

Los gráficos de torta nos resultaron una forma útil de analizar las distribuciones de las variables categóricas del tipo de barrio y el tipo de propiedad.



Los datos más importantes que obtenemos de estas visualizaciones son que hay muestras de propiedades de una gran cantidad de barrios de la ciudad, por lo que no nos estamos concentrando únicamente en un grupo selecto de barrios. Y, principalmente, que la gran mayoría de propiedades totales de dataset son departamentos.

Otra visualización interesante para comprender el problema es el gráfico de dispersión entre el precio de las propiedades y la cantidad de superficie cubierta.



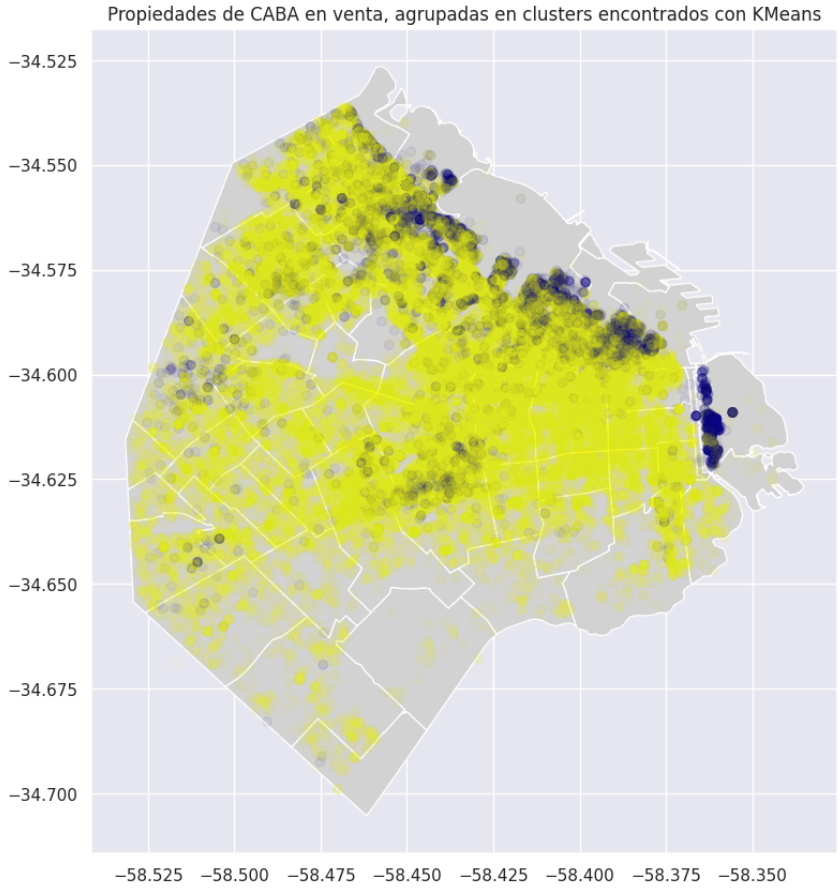
La variable `property_surface_covered` resultó ser la que más correlación tenía con el precio, existiendo un coeficiente de correlación de valor 0,75, positivo entre ambas variables, tal y como se aprecia en el gráfico.

Clustering

Pudimos descubrir mediante la estadística de Hopkins que el dataset presenta una fuerte tendencia al clustering, con un valor de alrededor de 0,000276 (para la implementación de esta métrica en librería `pyclustertend`, cuanto más cercano a 0 es más fuerte es la tendencia al clustering).

La cantidad apropiada de grupos que se deben formar resultó ser 2, llegamos a esta cantidad de grupos analizando el score de Silhouette para valores desde 2 clusters hasta 9 clusters y analizando y comparando el score de cada grupo. Este score cae

cuánto más se incrementa la cantidad de grupos, siendo como se mencionó antes el más elevado de 0,75 aproximadamente para 2 grupos.



La diferencia más llamativa entre ambos clusters son los rangos de precios que abarca cada uno, siendo los precios del segundo cluster más altos que los del primero. Además estos rangos de precios son disjuntos, es decir, los precios de las propiedades de un clúster no se superponen con los precios de las del otro, generando así una separación total. Finalmente, notamos que las estadísticas de las demás columnas son similares entre las propiedades de ambos grupos, salvo por el precio y ligeramente por las medias de los datos de las áreas de las propiedades.

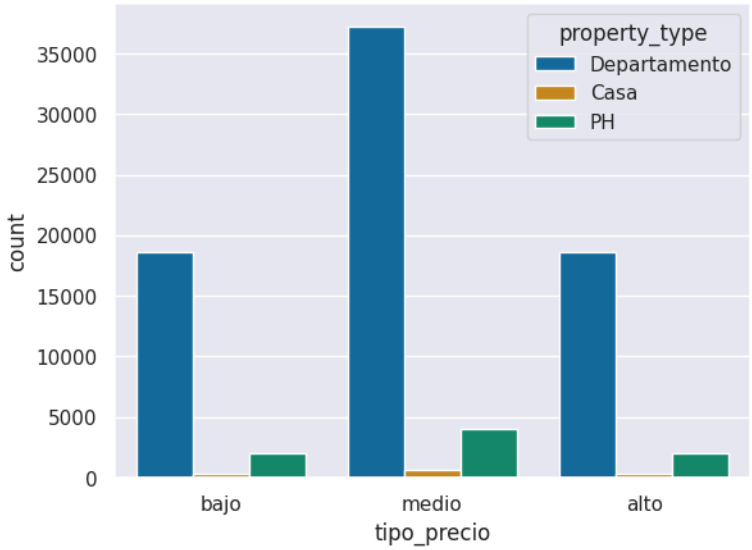
Con toda esta información, concluimos que el primer cluster está conformado por las propiedades más costosas y generalmente más espaciosas, que resultan también ser las menos numerosas; mientras que el segundo cluster está formado por las

propiedades menos costosas y más pequeñas, que resultan conformar la mayor parte del dataset.

Clasificación

Optamos por utilizar la tercera estrategia de división propuesta, que consiste en trabajar la variable precio por metro cuadrado relativa a cada tipo de propiedad y luego dividirla en tres intervalos, el primero con el 25% de las observaciones, el segundo con el 50% y el último con el 25% restante. El motivo de esta elección fue principalmente la proporción entre la cantidad de departamentos comparada con los otros dos tipos de

propiedad, por lo que si hacíamos la división de grupos en conjunto, los percentiles iban a estar altamente influenciados por los precios de los departamentos, y los precios de las casas y PHs iban a tener poca relevancia. En el gráfico siguiente, se ilustran las distribuciones de los tipos de precio por propiedad con la estrategia elegida.



Notamos que al igual que cuando usamos KMeans con 3 grupos, la forma elegida para hacer la división de los tipos de precio hace que el grupo que contiene los precios

intermedios sea el más numeroso. Sin embargo, cuando usamos KMeans, los otros dos clusters eran mucho más pequeños, y no llegaban a contener el 25% de los datos.

Otra diferencia muy clara es que cuando hicimos KMeans, la distribución de los tipos de propiedades era distinta en cada cluster. Usando este otro método, nos aseguramos que la distribución de tipos de propiedades de cada grupo sea exactamente la misma.

Árbol de Decisión

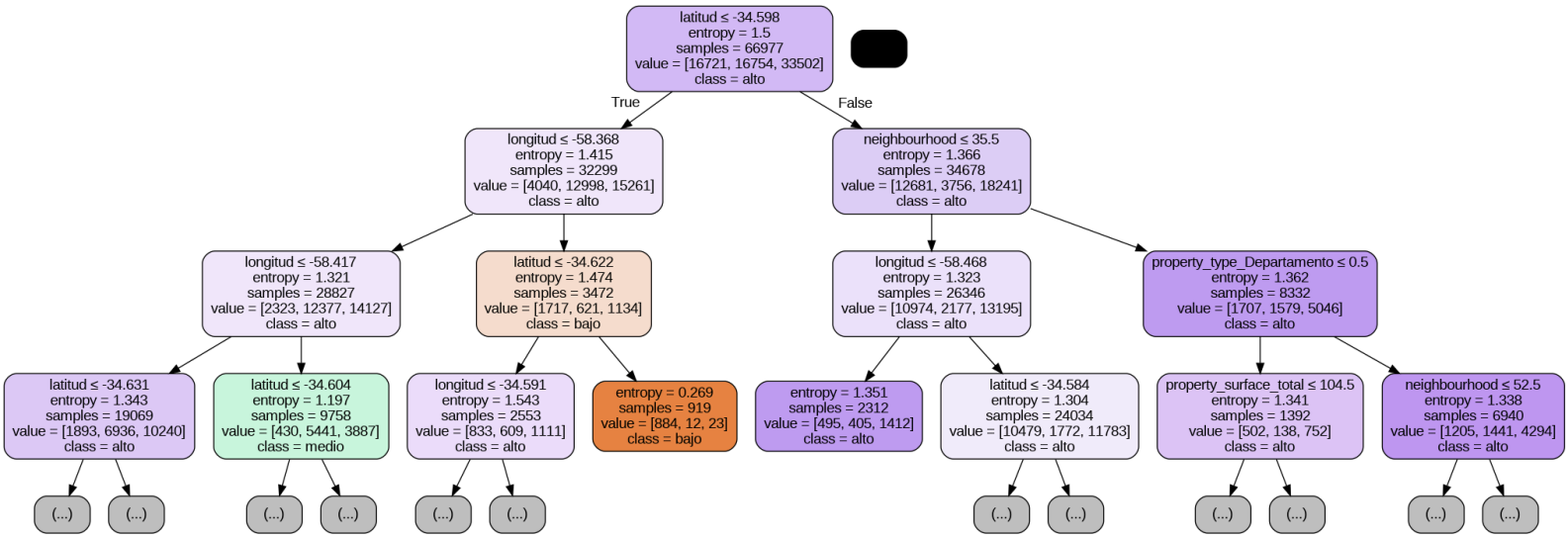
Optimizamos los hiperparámetros mediante la utilidad de sklearn `RandomizedSearch` y optamos por no utilizar `GridSearch`, ya que resultó muy costosa en cuanto a tiempos de ejecución, llegando a tardar 1 hora o más para una grilla que no contemplaba todos los hiperparámetros posteriormente descriptos. Los parámetros que buscamos optimizar son `criterion` (si se utiliza Gini o Entropía), `ccp_alpha` (parámetro de poda), `min_samples_split` (número mínimo de muestras que se requieren en un nodo antes de que se pueda dividir en nodos hijos adicionales), `min_samples_leaf`(número mínimo de muestras que se requieren en un nodo hoja), y `max_depth` (la profundidad máxima del árbol).

Utilizamos stratified k-fold cross validation para que los folds que se generan estén balanceados respecto a las clases, contando con 5 folds.

Para buscar los hiperparámetros optimizados utilizamos la métrica de f1 score ya que, basándonos en la teoría, es la que nos permite minimizar los falsos positivos y negativos, así como maximizar los verdaderos positivos.

Aclaración importante: los barrios fueron encodeados utilizando LabelEncoder, utilidad provista por sklearn, ya que resultaba más conveniente que OneHot encoding, el cual generaba excesivas columnas al dataframe dada la cantidad de barrios.

A continuación podemos ver el árbol que seleccionamos de todos los que se entrenaron:



En el mismo, la latitud del aviso toma mucha importancia a la hora de clasificar y junto con ella también la longitud, de modo que la ubicación de un aviso resulta de amplia importancia para predecir el tipo de precio que tendrá.

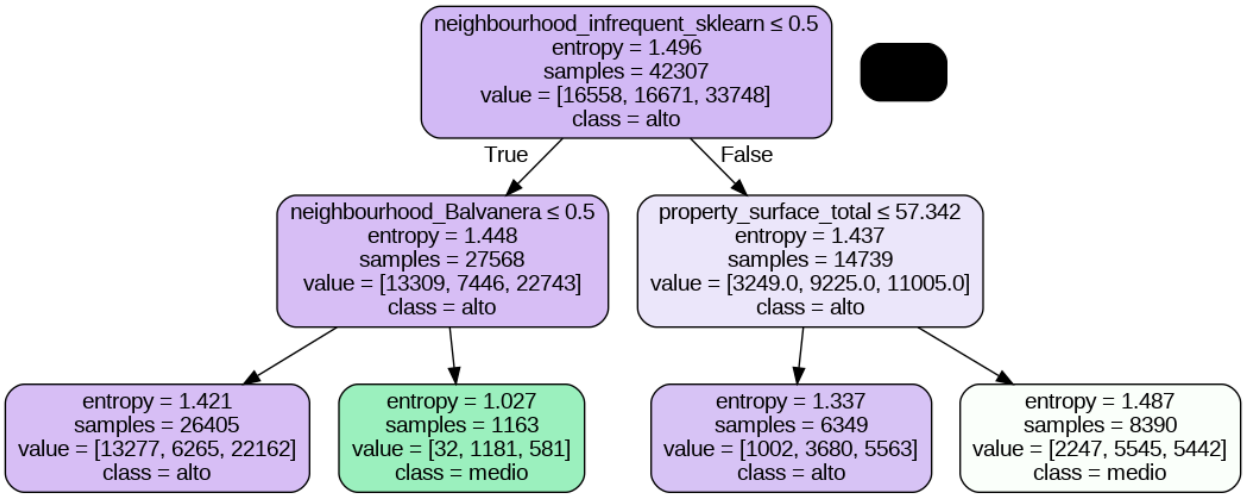
Random Forest

Para entrenar el modelo de Random Forest, usamos K-fold Cross Validation para la búsqueda de hiperparámetros, con 5 folds. Usamos Stratified KFold en lugar de solo KFold porque hay una distribución dispareja en las categorías del target. Por cómo lo armamos, hay más propiedades con la categoría medio que de las otras dos. Con lo cual, queremos tratar de mantener esa proporción en cada fold.
Los hiperparámetros a optimizar fueron:

- Número de árboles.
- El criterio a utilizar para medir la calidad de cada *split*.
- Profundidad máxima.
- Número mínimo de samples para un *split*.
- Número máximo de features a considerar para realizar un *split*.

Consideramos que estos hiperparámetros eran los más relevantes para el entrenamiento del clasificador. Para buscarlos, consideramos que la métrica más adecuada era la de F1 score, que considera tanto la precisión del modelo, como su recall.

Se muestra a continuación la conformación de uno de los árboles generados por el modelo:



Pareciera que la misma se vio demasiado afectada por las columnas relacionadas a los barrios, lo cual no es óptimo, puesto que ignora gran parte de los datos de los que disponemos.

Support Vector Machine (modelo a elección)

Se utilizó Grid Search con 5-fold cross-validation para encontrar los valores óptimos de los hiperparámetros dentro de los valores propuestos para probar. En este caso se se probó con diferentes valores de la constante `C` para cada uno de los diferentes kernels probados, entre los cuáles están el kernel lineal, polinómico, de grados 2 a 4, y finalmente el kernel RBF, para el que también se probaron diferentes valores del parámetro Gamma.

Los valores de la constante fueron propuestos de tal forma que se abarcara un amplio rango de valores. Esta constante sirve como parámetro de regularización para evitar el overfitting del algoritmo, algo especialmente importante para el kernel polinómico, ya que a mayores grados del polinomio más se tiende al overfitting, razón por la cuál se limitaron las pruebas hasta un polinomio de grado 4.

Los valores del parámetro Gamma para el kernel RBF fueron seleccionados según las sugerencias de Scikit Learn.

Se optó por utilizar la métrica accuracy o precisión debido a su facilidad de interpretación.

Lamentablemente, debido a la alta dimensionalidad del dataset, no se pueden ilustrar fácilmente las barreras de clasificación de la SVM.

Cuadro de Resultados

Modelo	F1-Test	Precision Test	Recall Test	Accuracy Test
Arbol de Decisión	0,57	0,58	0,58	0,58
Random Forest	0,27	0,40	0,35	0.50
SVM	-	-	-	-

=== Explicar en cada caso Cómo resultó la performance respecto al set de entrenamiento? ===
Para el árbol de decisión, la performance respecto al set de entrenamiento fue similar en testing, dando valores en torno a los registrados en el cuadro comparativo. Los hiperparámetros obtenidos fueron:

min_samples_split	15
min_samples_leaf	5
max_depth	17
criterion	entropy
ccp_alpha	0.0014444444444444444

Para el Random Forest, su performance en comparación con su performance durante el entrenamiento fue muy similar, dando el mismo resultado de F1 Score.

Los hiperparámetros obtenidos fueron los siguientes:

n_estimators	35
min_samples_split	0.2
max_features	15
max_depth	2
criterion	entropy

Estado de Avance

Análisis Exploratorio y Preprocesamiento de Datos

- Porcentaje de Avance: 100%/100%
- Tareas en curso: -.
- Tareas planificadas: -.
- Impedimentos: -.

Agrupamiento

- Porcentaje de Avance: 100%/100%
- Tareas en curso: -.
- Tareas planificadas: -.
- Impedimentos: -.

Clasificación

- Porcentaje de Avance: 30%/100%
- Tareas en curso:
    - Debido a un error en la selección de características con la que se había entrenado el modelo inicialmente (se habían seleccionado un grupo reducido de las mismas, solamente las que habían sido escaladas, es

decir 4 de 62), se tuvo que repetir el entrenamiento del mismo. Nos dimos cuenta de esto a último momento y no se llegó a finalizar el entrenamiento del modelo a tiempo. Durante los próximos días repetiremos el entrenamiento y posterior análisis del que resulte como el mejor clasificador.
    - Realizar Grid Search Cross Validation con el modelo de Random Forest.
- Tareas planificadas:
    - Terminar de entrenar los modelos pendientes.
    - Ver performance antes y después de entrenar los modelos pendientes.
    - Obtener el mejor modelo para los modelos aún pendientes y seleccionar el mejor predictor.
    - Ver por qué el árbol de decisión resulta tan pobre más en profundidad.
- Impedimentos:
    - Realmente es una incógnita por qué el árbol de decisión performa de manera tan pobre. Lo mismo nos sucedió con el modelo de Random Forest, aunque la conformación de sus árboles pareciera indicar que está relacionado con la presencia de las columnas que se desprendieron de la columna barrio por el One Hot Encoding.

Regresión

- Porcentaje de Avance: 0%/100%
- Tareas en curso: -.
- Tareas planificadas: -.
- Impedimentos: -.

Tiempo dedicado para Checkpoint 2

Integrante	Tarea	Prom. Hs Semana
Carlos Castillo	Construcción del target (en conjunto) Entrenamiento SVM (pendiente)	5

Juan Pablo Destefanis	Construcción del target (en conjunto) Entrenamiento RandomForest.	8
Celeste Gómez	Construcción del target (en conjunto) Entrenamiento Árbol de Decisión.	8