Universidad de Buenos Aires
Facultad De Ingeniería
Año 2019 - Segundo Cuatrimestre

75.06 - Organización de Datos
Trabajo Práctico

Análisis Exploratorio de Datos

Integrantes

Nombre	Padrón
Josué Giovanni Valdivia	93075

Repositorio de github: https://github.com/Giova262/OrgDatTP

- Introducción
- Objetivo
- Feature Engineering
- Preprocesamiento de los datos
- Algoritmos Utilizados
- Algoritmo Final
- Conclusiones

Introducción

El presente trabajo se enfoca en el set datos de registros históricos de publicaciones en www.zonaprop.com.ar. realizadas en México.

Dicho set datos consta de propiedades en venta en México entre los años 2012 y 2016 , valuadas en pesos mexicanos, cada fila representa una propiedad publicada en zonaprop y cada columna representa una característica o atributo distinto de cada propiedad.

Los features con los que contamos son :

Atributos de publicaciones		
Nombre	**Tipo**	**Descripción**
título	String	Título de la propiedad publicada
descripción	String	Descripción de la propiedad publicada
fecha	Date	Fecha de publicación
precio	Float	Valor de la publicación de la propiedad en pesos mexicanos

Atributos de localización		
Nombre	**Tipo**	**Descripción**
dirección	String	Dirección de la propiedad
ciudad	String	Ciudad de la propiedad
provincia	String	Provincia de la propiedad
idzona	Integer	La zona es un valor numérico correspondiente a una parte de la ciudad
lat	Float	Latitud geográfica de la propiedad
lng	Float	Longitud geográfica de la propiedad

Atributos de características básicas		
Nombre	**Tipo**	**Descripcipción**
tipopropiedad	String	Tipo de propiedad (casa,apartamento,terreno,etc)
metrostotales	Integer	Metros totales de la propiedad
metroscubiertos	Integer	Metros cubiertos de la propiedad
antiguedad	Integer	Antigüedad de la propiedad
habitaciones	Integer	Cantidad de habitaciones
garages	Integer	Cantidad de garages
banos	Integer	Cantidad de baños

Atributos de características adicionales

Nombre	Tipo	Descripcipción
gimnasio	Boolean	Indica si la propiedad tiene un gimnasio
usosmultiples	Boolean	Indica si la propiedad tiene un SUM
piscina	Boolean	Indica si la propiedad tiene una piscina
escuelascercanas	Boolean	Indica si la propiedad tiene escuelas cerca
centroscomercialescercanos	Boolean	Indica si la propiedad tiene centros comerciales cerca

Objetivo

Luego de haber realizado el Análisis Exploratorio correspondiente en el trabajo práctico anterior, el objetivo ahora es poder predecir el precio en el mercado de una o muchas propiedades dadas con sus características en el mismo formato que poseían las anteriores propiedades publicadas. Para cumplir con lo pedido utilice diversos algoritmos, encoding de variables categóricas, transformaciones de features, interacción entre features y mezclas de algoritmos

Este es un tipo de problema Supervisado de Regresión ya que la variable a predecir es continua y tenemos los labels en el set de entrenamiento

Feature Engineering

Transformaciones Realizadas

Las transformaciones que se realizaron al momento de abrir los archivos csv de test y de train fueron :

El feature fecha se lo transforma al tipo datatime para luego poder transformar el feature en subfeatures tales como año, mes y dia. Además se utilizó el feature año para hacer predicciones por año, para poder tener más presente los precios de un año dado ya que al pasar los años la inflación sube y puedo tener errores graves debido a eso ya que si intento predecir el precio de una propiedad del 2016 con datos que provienen de distintos años mezclados o en el peor caso donde pudiera tener solo los del 2012 seguro tendré un error debido a la inflación es por eso que decidí agrupar publicaciones por año y de acuerdo al año calcular el precio, finalmente concatenar los resultados de todos los años para armar el archivo de predicciones que se sube a Kaggle

Los features titulo , descripcion , direccion , ciudad, provincia se transformaron a la cantidad de caracteres que posee cada uno

Los metrostotales, metroscubiertos a sus valores les tome el logaritmo para tener más uniformidad en los valores

Encoding variables categóricas

*El feature ‘ tipodepropiedad ‘ pasó por varias transformaciones, la primera fue simplemente asignar un número a cada tipo de propiedad (**Label Encoding / Simple Encoding**)*

- *"Casa": 1,*
- *"Apartamento": 2,*
- *"Casa en condominio": 3,*
- *"Terreno": 4,*
- *"Local Comercial": 5,*
- *"Oficina comercial": 6,*
- *"Bodega comercial": 7,*
- *"Edificio": 8,*
- *"Terreno comercial": 9,*
- *"Casa uso de suelo": 10,*
- *"Quinta Vacacional": 11,*
- *"Duplex": 12,*
- *"Villa": 13,*
- *"Inmuebles productivos urbanos": 14,*
- *"Rancho": 15,*
- *"Local en centro comercial": 16,*
- *"Departamento Compartido": 17,*
- *"Otros": 18,*
- *"Nave industrial": 19,*
- *"Terreno industrial": 20,*
- *"Huerta": 21,*
- *"Lote": 22,*
- *"Hospedaje": 23,*
- *"Garage": 24*

*Pero este tipo de transformación no es muy buena ya que estoy ordenando las propiedades a dedo y asignando más valor a una que a otra simplemente por el orden y para algunos algoritmos este tipo de codificación no es nada buena, por lo que luego transforme el feature usando **“ One Hot Encoding “** donde marca cada categoría como un nuevo feature booleano y tiene más sentido para los modelos de machine learning*

Preprocesamiento del set de datos

Cuento con dos set de datos, el de entrenamiento ('train') y el de testing ('test') , este último son las publicaciones a las cuales les quiero predecir su precio en el mercado mexicano

Para poder tener una idea del error que estoy cometiendo con mis predicciones y no solo darme cuenta al momento de subir mis predicciones a Kaggle, separe el set de train en :

Entrenamiento : X_train Y_train
Testeo : X_test Y_test

Con esta separación lo que hago es entrenar mi algoritmo con el set de X_train Y_train para luego de manera local predecir los valores de X_test , una vez que tenga los resultados los comparó con el Y_test y obtengo una medida del error que estoy cometiendo sin tener que subir a Kaggle y predecir a ciegas

Para el cálculo del error utilizó la métrica Root Mean Squared Logarithmic Error , que posee la siguiente fórmula

$$RMSLE = \sqrt{\frac{\sum((log(\text{actual} + 1) - log(\text{pred} + 1))^2}{n}}$$

La razón por la cual se utiliza esta métrica es porque relativiza el error absoluto considerado.

Además Utilizó a modo de re chequeo la métrica que utiliza Kaggle para calcular el score

Para el Manejo de Nulls fui variando bastante, primero para los features numéricos simplemente utilice el valor promedio mean(), y para los categóricos por

ejemplo los tipos de propiedad lo que hice fue utilizar la moda, que sería poner por ejemplo en tipodepropiedad al tipo más frecuente que es una 'casa', luego fui variando de manera aleatoria cada uno de ellos como por ejemplo poner en algunos el valor max(), min() , cambiar el tipo de propiedad a la más cara , más barata, menos frecuente, etc

Algoritmos Utilizados

- ***Constante***

Como primer algoritmo y para poder introducirme en el mundo de matching learning, entender cómo es la secuencia que se debe seguir como definir el modelo, entrenar, predecir, calcular los errores empecé con el DummyRegressor definiendo la estrategia constante y el valor de la constante que quería :

strategy= constant
constant=500

Luego de hacer el Split de mi set de entrenamiento en uno de train y de test

```
Train shapes: X=(180000, 14) y=(180000,)
Test  shapes: X=(60000, 14)  y=(60000,)
```

Entrenó el modelo con el set de train utilizando ' fit ' para luego obtener mis predicciones con ' predict ', a la hora de entrenar no seleccione ningún feature en particular por que sea lo que sea pondria una constante como predicción.
.
Cálculo de Errores :

Una vez que tengo el modelo entrenado realice el cálculo de errores contra el mismo set de entrenamiento y contra el de test utilizando las dos métricas :

Root Mean Squared Logarithmic Error(train) : 8.23876
Root Mean Squared Logarithmic Error(test) : 8.23005

MAE Error (train) : 2536413.14206
MAE Error (test) : 2512113.95740

Análisis de los resultados :

Underfitting : El modelo no es suficientemente complejo para los datos , no es suficientemente expresivo. Solución es un modelo más complejo o cambiar el modelo. Sucede cuando tengo malos resultados con el propio set de entrenamiento

Score en Kaggle : 1608422.62617

- ***Promedio***

Siguiendo con el DummyRegressor pero cambiando la estrategia a la del valor promedio de los precios de las propiedades con las que entreno

strategy= mean

Luego de hacer el mismo Split de train y de test

```
Train shapes: X=(180000, 14) y=(180000,)
Test  shapes: X=(60000, 14)  y=(60000,)
```

Cálculo de Errores :

Root Mean Squared Logarithmic Error(train) : 0.90228
Root Mean Squared Logarithmic Error(test) : 0.90318

MAE Error (train) : 1612604.48000
MAE Error (test) : 1602549.96274

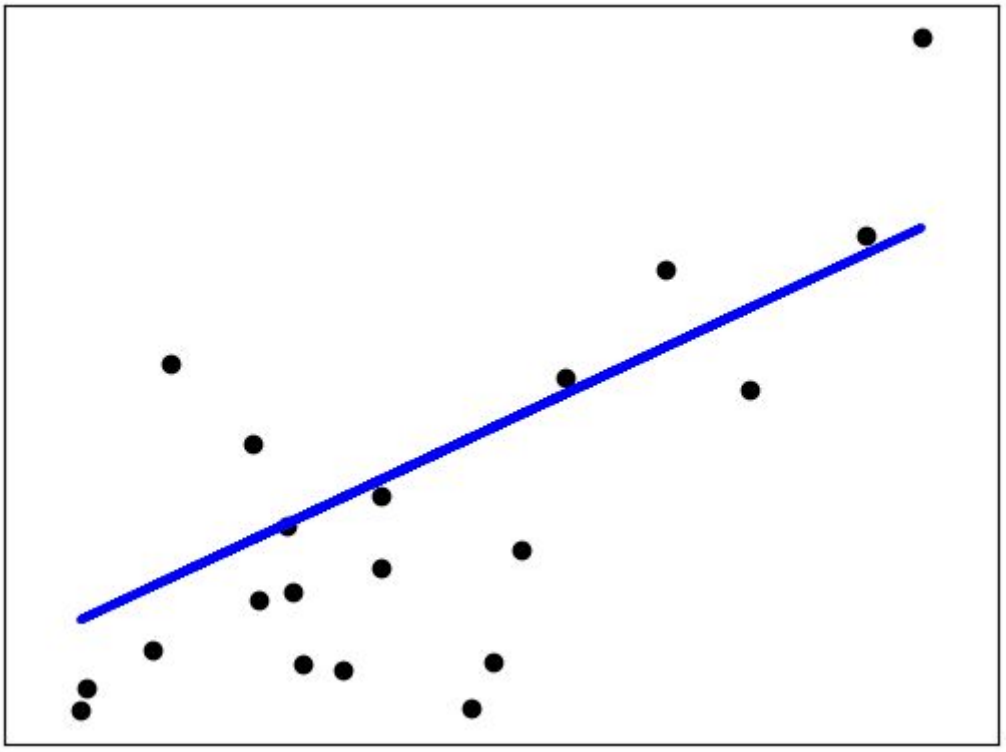
Análisis de los resultados :

Los resultados son mejores al anterior pero aun así el error es muy grande y sufre de Underfitting

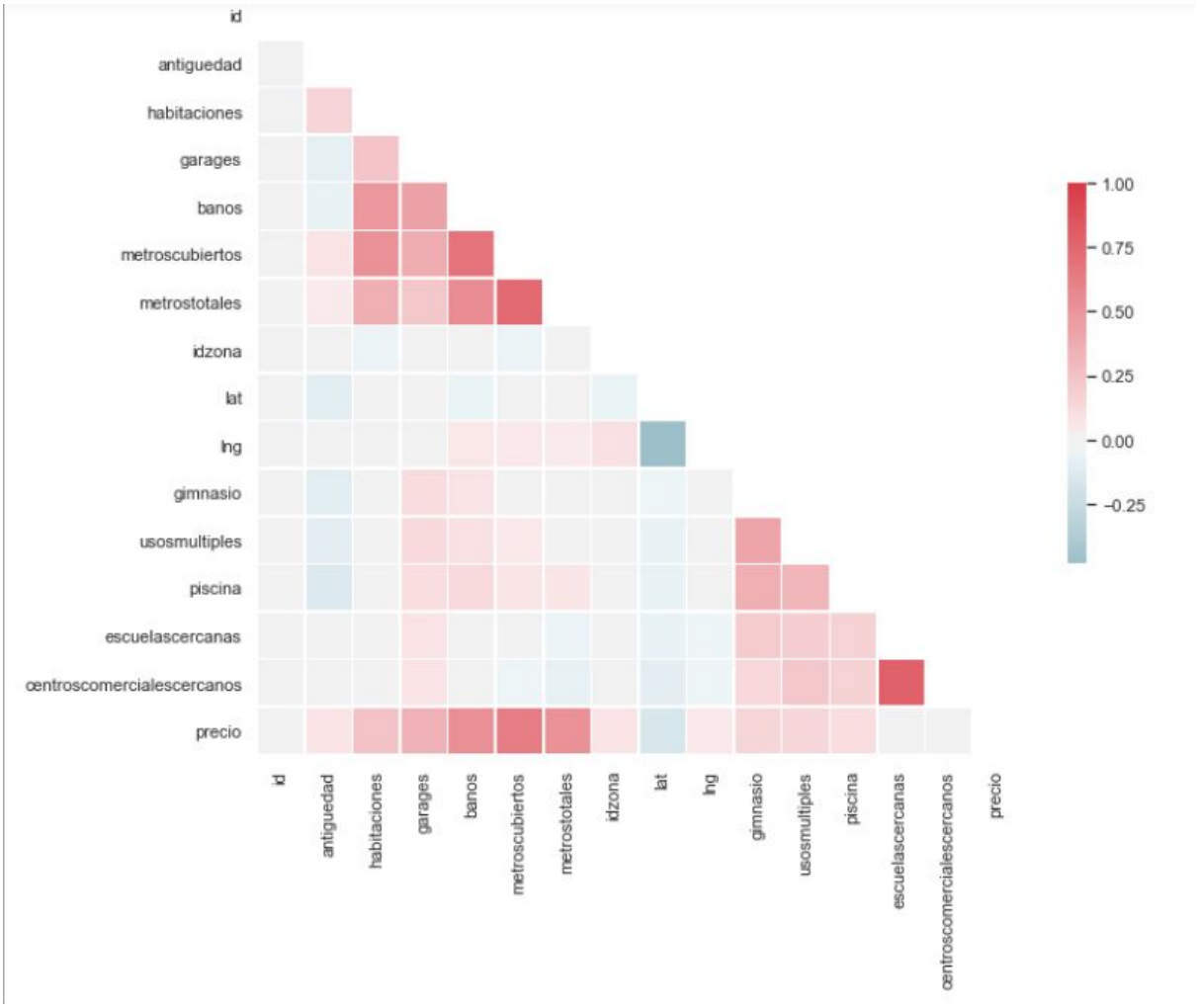
Score en Kaggle : 1192523.74533

- ***Regresión Lineal***

Para este modelo utilizo unicamente 1 feature para que el modelo ajuste una recta y pueda predecir los precios



El tema ahora es elegir qué feature voy a utilizar, saber cual o cuales son los más importantes, me apoye en los análisis exploratorios del trabajo practico pasado, como el plot de correlaciones donde cuando un feature aumenta el otro tiende a aumentar también



Enfocandome principalmente en la fila de precio, veo los features que se correlacionan mejor con él y llegó a la conclusión que son : Antiguedad, habitaciones, garages, baños, metroscubiertos, metrostotales, idzona, lng, gimnasio , usosmultiples y piscina; A priori esos serian los mas importantes para determinar el precio de una propiedad publicada

A continuación mostraré para cada feature los errores cometidos y una breve conclusión :

1- metros cubiertos :

Nota : Para este llené los Nulls con el promedio de los valores

Root Mean Squared Logarithmic Error(train) : 0.65673
Root Mean Squared Logarithmic Error(test) : 0.65648

MAE Error (train) : 1202603.98514
MAE Error (test) : 1191062.51364

Score en Kaggle :1120935.99699

Conclusión : Aun tengo underfitting pero en su momento me gusto el resultado

2- metros cubiertos :

Root Mean Squared Logarithmic Error(train) : 0.77069
Root Mean Squared Logarithmic Error(test) : 0.76986

MAE Error (train) : 1384485.13818
MAE Error (test) : 1375006.68201

Conclusión : Al ver los resultados de los errores simplemente no lo subi a kaggle y decidí seguir probando con otros features

3- banos:

Root Mean Squared Logarithmic Error(train) : 0.71305
Root Mean Squared Logarithmic Error(test) : 0.71311

MAE Error (train) : 1326576.26867
MAE Error (test) : 1316655.85210

Conclusión : Al ver los resultados de los errores simplemente no lo subi a kaggle y decidí seguir probando con otros features

4- garages:

Root Mean Squared Logarithmic Error(train) : 0.84448
Root Mean Squared Logarithmic Error(test) : 0.84314

MAE Error (train) : 1505488.62183
MAE Error (test) : 1492386.44838

Conclusión : Al ver los resultados de los errores simplemente no lo subi a kaggle y decidí seguir probando con otros features

5- habitaciones:

Root Mean Squared Logarithmic Error(train) : 0.85682
Root Mean Squared Logarithmic Error(test) : 0.85757

MAE Error (train) : 1549433.63003
MAE Error (test) : 1541163.97837

Conclusión : Al ver los resultados de los errores simplemente no lo subi a kaggle y decidí seguir probando con otros features

6- gimnasio:

Root Mean Squared Logarithmic Error(train) : 0.89044
Root Mean Squared Logarithmic Error(test) : 0.89150

MAE Error (train) : 1587871.29864
MAE Error (test) : 1579081.68876

Conclusión : Al ver los resultados de los errores simplemente no lo subi a kaggle y decidí seguir probando con otros features

7- usosmultiples:

Root Mean Squared Logarithmic Error(train) : 0.89135
Root Mean Squared Logarithmic Error(test) : 0.89216

MAE Error (train) : 1590987.90271
MAE Error (test) : 1580698.12959

Conclusión : Al ver los resultados de los errores simplemente no lo subi a kaggle y decidí seguir probando con otros features

8- piscina:

Root Mean Squared Logarithmic Error(train) : 0.89413
Root Mean Squared Logarithmic Error(test) : 0.89515

MAE Error (train) : 1598374.73603
MAE Error (test) : 1588343.97441

Conclusión : Al ver los resultados de los errores simplemente no lo subi a kaggle y decidí seguir probando con otros features

9- titulo:

Nota: Acá al feature le hice una transformación reemplace el título con la cantidad de caracteres que tiene el mismo por que note que las propiedades más caras tenían títulos más cortos y decidí probar. Realice el mismo procedimiento con los nulls que en el anterior caso

Root Mean Squared Logarithmic Error(train) : 0.90225
Root Mean Squared Logarithmic Error(test) : 0.90318

MAE Error (train) : 1612614.37721
MAE Error (test) : 1602586.869903

Conclusión : Al ver los resultados de los errores simplemente no lo subi a kaggle y decidí seguir probando con otros features

10- tipodepropiedad:

Nota: Acá al feature le hice un Label encoding y sobre eso realice la predicción , luego me daría cuenta que no fue la mejor opción ya que podía haber usado one hot encoding. Con los nulls lo que hice fue ponerle el valor promedio de los labels que había puesto (luego me di cuenta que hice muy mal , ni siquiera puse la moda)

Root Mean Squared Logarithmic Error(train) : 0.90122
Root Mean Squared Logarithmic Error(test) : 0.90258

MAE Error (train) : 1611042.38531
MAE Error (test) : 1602328.47233

Conclusión : Al ver los resultados de los errores simplemente no lo subi a kaggle y decidí seguir probando con otros features

11- fecha:

Nota: Transforme la fecha a unicamente al año y realice el mismo procedimiento con los nulls que en el anterior caso

Root Mean Squared Logarithmic Error(train) : 0.89211
Root Mean Squared Logarithmic Error(test) : 0.89394

MAE Error (train) : 1597245.39542
MAE Error (test) : 1588343.37661

Conclusión : Al ver los resultados de los errores simplemente no lo subi a kaggle y decidí seguir probando con otros features

- ***Regresion Lineal Multiple***

Este es el mismo modelo que el anterior pero ahora empece a agregar más features al mismo tiempo , a continuación están los features que utiliza y sus resultados

1-

Features : metrostotales,metroscubiertos ,banos

Root Mean Squared Logarithmic Error(train) : 0.62025
Root Mean Squared Logarithmic Error(test) : 0.61993

MAE Error (train) : 1161884.23342
MAE Error (test) : 1152018.48875

2-

Features : 'metrostotales','metroscubiertos','banos','habitaciones','gimnasio', 'usosmultiples','piscina'

Root Mean Squared Logarithmic Error(train) : 0.60806
Root Mean Squared Logarithmic Error(test) : 0.60794

MAE Error (train) : 1129269.61449
MAE Error (test) : 1119327.35026

3-

Features : "metrostotales','metroscubiertos','banos','gimnasio','habitaciones', 'antiguedad'

Root Mean Squared Logarithmic Error(train) : 0.60840
Root Mean Squared Logarithmic Error(test) : 0.60735

MAE Error (train) : 1124385.30383
MAE Error (test) : 1114935.52215

4-

Features : 'anio','mes','metrostotales','centroscomercialescercanos', 'metroscubiertos','banos','habitaciones','idzona','gimnasio','usosmultiples'

Root Mean Squared Logarithmic Error(train) : 0.61060
Root Mean Squared Logarithmic Error(test) : 0.60723

MAE Error (train) : 1105706.21188
MAE Error (test) : 1097307.19302

- ***Random Forest***

Modelo basado en árboles de decisión donde cada árbol depende de los valores de un vector aleatorio probado independientemente y con la misma distribución para cada árbol, a continuación muestros los distintos hiperparametros que fui usando y los features

1-

***Features** : 'metrostotales','centroscomercialescercanos', 'metroscubiertos','banos','habitaciones','idzona','gimnasio','usosmultiples'*

***Hiperparametros** : random_state=0, n_jobs=-1,max_depth = 20,min_samples_split =20 ,n_estimators =50*

Root Mean Squared Logarithmic Error(train) : 0.33201
Root Mean Squared Logarithmic Error(test) : 0.41936

MAE Error (train) : 559334.94425
MAE Error (test) : 718305.31512

Nota: Me pareció un muy buen resultado y lo subi a Kaggle, además se percibía un poco como da mejor resultado con el propio set de entrenamiento que con el de test lo que me muestra un poco de Overfitting (está memorizando el set de

entrenamiento). Una solución es conseguir más datos , aplicar una regularización o buscar otro modelo menos complejo

Score en Kaggle : 716812.39431

2-

***Features** : 'metrostotales','centroscomercialescercanos', 'metroscubiertos','banos','habitaciones','idzona','gimnasio','usosmultiples'*

***Hiperparametros** : random_state=0, n_jobs=-1,max_depth = 100,max_features =6,min_samples_split = 4 ,n_estimators =200*

Root Mean Squared Logarithmic Error(train) : 0.22532
Root Mean Squared Logarithmic Error(test) : 0.40942

MAE Error (train) : 337653.34109
MAE Error (test) : 704304.79830

Nota : Esta Overfitteando mucho

Score en Kaggle : 702024.74347

3-

***Features** :*
'anio','mes','metrostotales','centroscomercialescercanos','piscina','escuelascercanas', 'metroscubiertos','banos','habitaciones','lat','lng','idzona','gimnasio','usosmultiples'

***Hiperparametros** : random_state=0,bootstrap =False,min_samples_leaf=2 ,n_jobs=-1,max_depth = 100,max_features = 8 ,min_samples_split = 2 ,n_estimators =99*

Root Mean Squared Logarithmic Error(train) : 0.12171
Root Mean Squared Logarithmic Error(test) : 0.39234

MAE Error (train) : 150703.83197
MAE Error (test) : 659485.92929

Nota : Esta Overfitteando mucho

Score en Kaggle : 669492.31492

4-

Nota : Acá utilice tipodepropiedad haciendo el label encoding luego usaría el one hot encoding , la fecha la descompuse en año y mes , y los nulls los llene con el valor promedio

***Features** :*
'tipodepropiedad','anio','metrostotales','antiguedad','centroscomercialescercanos','garages','piscina','metroscubiertos','banos','habitaciones','lat','lng','idzona','gimnasio','usosmultiples'

***Hyperparameters** : random_state=0,bootstrap =False,min_samples_leaf=2 ,n_jobs=-1,max_depth = 100,max_features = 5 ,min_samples_split = 2 ,n_estimators = 400*

Root Mean Squared Logarithmic Error(train) : 0.13892
Root Mean Squared Logarithmic Error(test) : 0.35932

MAE Error (train) : 183518.00384
MAE Error (test) : 599651.31538

Nota : Esta Overfitteando mucho

Score en Kaggle : 583658.64804 (Mejor Resultado 29/11/19 pero al generarlo llene los nulls con el promedio de X_train es decir llene con un set de train anterior o posterior la verdad no se cual fue, y no pude encontrar los valores de nuevo..)

5-

Nota : Acá utilice tipodepropiedad haciendo el label encoding luego usaría el onehotencoding , la fecha la descompuse en año y mes , y los nulls los llene con el valor promedio

Features *:*
"tipodepropiedad','anio','mes','metrostotales','antiguedad','garages','piscina','metroscubiertos','banos','habitaciones','lat','lng','idzona','gimnasio','ususosmultiples'

Hyperparameters *: random_state=0,bootstrap =False,min_samples_leaf=2 ,n_jobs=-1,max_depth = 80,max_features = 10 ,min_samples_split = 2 ,n_estimators = 500*

Root Mean Squared Logarithmic Error(train) : 0.09347
Root Mean Squared Logarithmic Error(test) : 0.35775

MAE Error (train) : 112026.05990
MAE Error (test) : 591196.32856
.
Nota : Acá el tiempo y consumo de recursos es mucho ya se nota . Decidí parar un rato y probar mas modelos para luego seleccionar el que me parecía mas conveniente

- ***KNN***

Features *:*
'tipodepropiedad','anio','mes','metrostotales','antiguedad','garages','piscina','metroscubiertos','banos','habitaciones','lat','lng','idzona','gimnasio','ususosmultiples'

Hiperparametros *: n_neighbors = 20*

Root Mean Squared Logarithmic Error(train) : 0.42322
Root Mean Squared Logarithmic Error(test) : 0.44397

MAE Error (train) : 729996.35736
MAE Error (test) : 762946.37406

Nota: Lo que me gusto es que ya no overfiteaba y era más regular con las predicciones tanto en el set de entrenamiento y el de test. Estuve probando más variantes pero ninguna daba mejores resultados que el random forest

Score en Kaggle : 745536.27245

- ***Ridge Regression***

Features *:*
'tipodepropiedad','anio','mes','metrostotales','antiguedad','garages','piscina','metroscubiertos','banos','habitaciones','lat','lng','idzona','gimnasio','usosmultiples'

Hiperparametros *: alpha=1.0 , fit_intercept = False , max_iter =1000*

Root Mean Squared Logarithmic Error(train) : 0.61568
Root Mean Squared Logarithmic Error(test) : 0.61833

MAE Error (train) : 1084562.35267
MAE Error (test) : 1075626.77831

Nota: Fue muy rápido a la hora de entrenar y de predecir pero por los resultados decidí seguir probando más algoritmos

- ***Support Vector Machines***

Features *:*
'tipodepropiedad','anio','mes','metrostotales','antiguedad','garages','piscina','metroscubiertos','banos','habitaciones','lat','lng','idzona','gimnasio','usosmultiples'

Nota : Este tarda horas en obtener un resultado y cuando me lo dio fue malo , y me canse de esperar y no volví a probarlo (El resultado no me lo acuerdo)

- ***PLSRegression***

Nota : Intente correr con este modelo pero nunca pude por que me tiraba Memory Error y no pude solucionarlo

- ***Naive Bayes***

Nota : Intente correr con este modelo también pero esta vez murió mi computadora ademas de tardar mucho lageo la pc y tuve que reiniciarla

- ***Arbol de decision***

Para el encoding de variables categóricas como tipodepropiedad ahora sí utilice OnehotEncoding.

El manejo de los nulls :

'tipodepropiedad' con 'Oficina comercial'
'metroscubiertos' con min()
'antiguedad' con max()
'habitaciones' con mode()
'banos' con max()
'idzona' con mode()
'garages' con min()
'metrostotales' con mean()
'lng' con mode()
'lat' con mode()

***Features** :*
'tipodepropiedad','anio','metrostotales','antiguedad','garages',
'piscina','metroscubiertos','banos','habitaciones','lat','lng','idzona','gimnasio',
'ususosmultiples'

***Hiperparametros** : random_state=10 ,*
max_depth = 15 ,
min_samples_split = 3,
min_samples_leaf =4 ,
min_weight_fraction_leaf = 0.000000000000001,
max_features = 13,
max_leaf_nodes = 50000,
min_impurity_decrease =2,
presort=False

Root Mean Squared Logarithmic Error(train) : 0.33982
Root Mean Squared Logarithmic Error(test) : 0.41956

MAE Error (train) : 586238.23402
MAE Error (test) : 733616.94800

Nota: Como me pareció mejor los resultados del random forest decidí probar con mas modelos

- ***<u>ExtraTreesRegressor</u>***

Para este utilice igual que en el anterior OnehotEncoding y el mismo relleno de los nulls

Features *:*
'anio','metrostotales','antiguedad','centros comerciales cercanos','garages','piscina','metroscubiertos','banos','habitaciones','lat','lng','idzona','gimnasio','usosmultiples','tipodepropiedad_1_oh','tipodepropiedad_2_oh','tipodepropiedad_3_oh','tipodepropiedad_4_oh','tipodepropiedad_5_oh','tipodepropiedad_6_oh','tipodepropiedad_7_oh','tipodepropiedad_8_oh','tipodepropiedad_9_oh','tipodepropiedad_10_oh','tipodepropiedad_11_oh','tipodepropiedad_12_oh','tipodepropiedad_13_oh','tipodepropiedad_14_oh','tipodepropiedad_15_oh','tipodepropiedad_16_oh','tipodepropiedad_17_oh','tipodepropiedad_18_oh','tipodepropiedad_19_oh','tipodepropiedad_20_oh','tipodepropiedad_21_oh','tipodepropiedad_22_oh','tipodepropiedad_23_oh', 'tipodepropiedad_24_oh'

Hiperparametros *: random_state=10 ,*
max_depth = 15 ,
min_samples_split = 3,
min_samples_leaf =4 ,
min_weight_fraction_leaf = 0.000000000000001,
max_features = 13,
max_leaf_nodes = 50000,
min_impurity_decrease =2

Root Mean Squared Logarithmic Error(train) : 0.42989
Root Mean Squared Logarithmic Error(test) : 0.45495

MAE Error (train) : 766228.01422
MAE Error (test) : 819916.91040

Nota: Como me pareció mejor los resultados del random forest decidí probar con mas modelos

- ***<u>Gradient BoostingRegressor</u>***

Nota : Este tarda horas en obtener un resultado y cuando me lo dio fue malo , y me canse de esperar y no volví a probarlo como en un modelo anterior me paso (El resultado no me lo acuerdo)

- ***Voting Regressor***

Este me gusto muchisimo ya que podía usar 2 modelos diferentes y entre las predicciones hacer algún tipo de promedio

Features :

'anio','metrostotales','antiguedad','centroscomercialescercanos',
'garages','piscina','metroscubiertos','banos','habitaciones','lat','lng','idzona',
'gimnasio','usosmultiples','tipodepropiedad_1_oh','tipodepropiedad_2_oh',
'tipodepropiedad_3_oh','tipodepropiedad_4_oh','tipodepropiedad_5_oh','tipodepropiedad_6_oh','tipodepropiedad_7_oh','tipodepropiedad_8_oh','tipodepropiedad_9_oh','tipodepropiedad_10_oh','tipodepropiedad_11_oh','tipodepropiedad_12_oh','tipodepropiedad_13_oh','tipodepropiedad_14_oh','tipodepropiedad_15_oh',
'tipodepropiedad_16_oh','tipodepropiedad_17_oh','tipodepropiedad_18_oh','tipodepropiedad_19_oh','tipodepropiedad_20_oh','tipodepropiedad_21_oh','tipodepropiedad_22_oh','tipodepropiedad_23_oh', 'tipodepropiedad_24_oh'

Modelos :

- *Random Forest :*

random_state=0,bootstrap =False,min_samples_leaf=3 ,n_jobs=-1, max_depth = 40,max_features = 21 ,min_samples_split = 2 , n_estimators = 400

- *KNN :*

n_neighbors = 100

Hiperparametros : *[('rf', r1),('rf2', r3)],weights =[4,1]*

Root Mean Squared Logarithmic Error(train) : 0.22130
Root Mean Squared Logarithmic Error(test) : 0.36138

MAE Error (train) : 324046.11078
MAE Error (test) : 600385.19296

Nota: Me gusto poder mezclar modelos pero quise seguir probando más algoritmos antes de decidirme por cual ir

- ***XGBoost***

Features :
'anio','mes','metrostotales','antiguedad','centroscomercialescercanos','garages','piscina','metroscubiertos','banos','habitaciones','lat','lng','idzona','gimnasio','usosmultiples','tipodepropiedad_1_oh','tipodepropiedad_2_oh','tipodepropiedad_3_oh','tipodepropiedad_4_oh','tipodepropiedad_5_oh','tipodepropiedad_6_oh','tipodepropiedad_7_oh','tipodepropiedad_8_oh','tipodepropiedad_9_oh','tipodepropiedad_10_oh','tipodepropiedad_11_oh','tipodepropiedad_12_oh','tipodepropiedad_13_oh','tipodepropiedad_14_oh','tipodepropiedad_15_oh','tipodepropiedad_16_oh','tipodepropiedad_17_oh','tipodepropiedad_18_oh','tipodepropiedad_19_oh','tipodepropiedad_20_oh','tipodepropiedad_21_oh','tipodepropiedad_22_oh','tipodepropiedad_23_oh',
'tipodepropiedad_24_oh'

Hiperparametros :

objective ='reg:linear',
colsample_bytree = 0.3,
learning_rate = 0.1,
max_depth = 5, alpha = 10,
n_estimators = 10

Root Mean Squared Logarithmic Error(train) : 0.53711
Root Mean Squared Logarithmic Error(test) : 0.53297

MAE Error (train) : 1089806.14092
MAE Error (test) : 1073217.64467

Cross-Validation Hyperparameter : dtrain=data_dmatrix, params=params, nfold=10,
num_boost_round=10, early_stopping_rounds=10,
metrics="rmse", as_pandas=True, seed=123

	train-rmse-mean	train-rmse-std	test-rmse-mean	test-rmse-std
0	3071101.20	2651.51	3071068.75	24508.47
1	2861987.92	13132.33	2862488.55	25327.69
2	2676849.85	21276.95	2677351.85	30523.32
3	2522175.10	23542.44	2522705.45	29816.63
4	2384714.12	21129.21	2385403.75	27770.21
5	2263496.92	13136.16	2264467.33	22975.82
6	2148835.65	16110.80	2149721.55	18899.04
7	2054206.27	13379.05	2055328.62	17883.03
8	1969260.71	19910.36	1970516.36	17527.18
9	1898279.93	16268.10	1899784.68	16577.63

Nota : Aca utilice cross Validation que evalúa los que los resultados sean independientes de la partición entre datos de entrenamiento y prueba

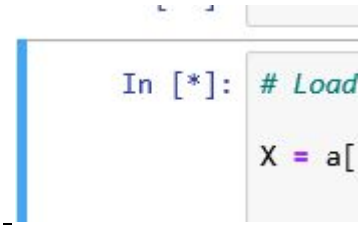
- ***Red Neuronal Clásica - Perceptrón***

Es simpático y tiene convergencia asegurada

Features :
'anio','mes','metrostotales','antiguedad','centroscomercialescercanos','garages','piscina','metroscubiertos','banos','habitaciones','lat','lng','idzona','gimnasio','usosmultiples','tipodepropiedad_1_oh','tipodepropiedad_2_oh','tipodepropiedad_3_oh','tipodepropiedad_4_oh','tipodepropiedad_5_oh','tipodepropiedad_6_oh','tipodepropiedad_7_oh','tipodepropiedad_8_oh','tipodepropiedad_9_oh','tipodepropiedad_10_oh','tipodepropiedad_11_oh','tipodepropiedad_12_oh','tipodepropiedad_13_oh','tipodepropiedad_14_oh','tipodepropiedad_15_oh','tipodepropiedad_16_oh','tipodepropiedad_17_oh','tipodepropiedad_18_oh','tipodepropiedad_19_oh','tipodepropiedad_20_oh','tipodepropiedad_21_oh','tipodepropiedad_22_oh','tipodepropiedad_23_oh', 'tipodepropiedad_24_oh'

Hiperparametros : *tol=1e-3, random_state=0*

Nota: Me canse de esperar..2 hrs.. sin computadora y seguía ..



Algoritmo final

Después de haber probado todos los algoritmos anteriormente mencionados decidí utilizar el de Random Forest y enfocarme en él, como mejorar los features, los hiperparametros , manejo de nulls , encoding de variables categóricas, generación de nuevos features, relaciones entre features

Preprocesamiento de datos :

Al estudiar el set de test que tenemos note que en feature tipodepropiedad poseían 22 tipos y en el de train 24, entonces los dos tipos que faltan son 'Hospedaje' y 'Garaje' como medida a esto lo que hice fue filtrar el set de train para eliminarlos y no entrenar con datos que tengan ese tipo de propiedad para que el set de train se parezca más al de test

Manejo de los nullls

Los features que poseen nulls son :

	Nulls	%porcentaje
lng	123455	51.45
lat	123455	51.45
direccion	53028	22.10
metrostotales	51456	21.44
antiguedad	43517	18.14
garages	37734	15.73
idzona	28579	11.91
banos	26189	10.91
habitaciones	22452	9.36
metroscubiertos	17386	7.25
titulo	5386	2.24
descripcion	1615	0.67
ciudad	354	0.15
provincia	153	0.06

Para llenar los nulls lo que hice fue poner en los casos de valores numéricos el promedio y en los casos de texto los llene con la moda

Encoding variables categóricas

Para el feature tipo de propiedad utilizó one hot encoding que convierte cada categoría a un nuevo booleano feature

Transformaciones de feature

El feature fecha lo convierto a datetime y sub dividido el feature en año,mes y dia

Los feature título,descripción,dirección,ciudad,provincia lo convierto a un número que representa la cantidad de caracteres que usaron para escribir el título

En algunas pruebas cambie los features de metrostotales , metroscubiertos a log del valor para uniformizar más los valores

Después de esto me quedan 239952 filas con 50 features

Divido set de Datos

Los feature que utilizo son :

'anio'
'mes',
'dia',
'título',
'descripcion',
'direccion',
'ciudad',
'provincia',
'metrostotales',
'escuelascercanas',
'antiguedad',
'centroscomercialescercanos',
'garages',
'piscina',
'metroscubiertos',
'banos',
'habitaciones',
'lat','lng','idzona',
'gimnasio',
'usosmultiples',
'tipodepropiedad_1_oh',
'tipodepropiedad_2_oh',
'tipodepropiedad_3_oh',
'tipodepropiedad_4_oh',
'tipodepropiedad_5_oh',
'tipodepropiedad_6_oh',
'tipodepropiedad_7_oh',
'tipodepropiedad_8_oh',
'tipodepropiedad_9_oh',
'tipodepropiedad_10_oh',
'tipodepropiedad_11_oh',
'tipodepropiedad_12_oh',
'tipodepropiedad_13_oh',
'tipodepropiedad_14_oh',
'tipodepropiedad_15_oh',
'tipodepropiedad_16_oh',
'tipodepropiedad_17_oh',
'tipodepropiedad_18_oh',
'tipodepropiedad_19_oh',
'tipodepropiedad_20_oh',
'tipodepropiedad_21_oh',
'tipodepropiedad_22_oh'

Ahora que tengo los set de X_train, X_test, y_train, y_test , defino el modelo RandomForestRegressor y para buscar los hiper parámetros utilizó Random Search con Cross Validation

```
n_estimators = [int(x) for x in np.linspace(start = 50, stop = 400, num = 15)]
max_features = [int(x) for x in np.linspace(start = 4, stop = 44, num = 15)]
max_depth = [int(x) for x in np.linspace(50, 100, num = 10)]
min_samples_split = [2, 5, 10]
min_samples_leaf = [1, 2, 4]
bootstrap = [ False]


# Create the random grid
random_grid = {'n_estimators': n_estimators,
               'max_features': max_features,
               'max_depth': max_depth,
               'min_samples_split': min_samples_split,
               'min_samples_leaf': min_samples_leaf,
               'bootstrap': bootstrap}
```

El RandomizedSearchCV implementa los metodos fit , score , predict , predict_proba, decision_function , otros y los parámetros del estimador usados son optimizados por cross-validated

Una vez que tengo los mejores hiperparametros encontrados realizó la predicción

También una vez que tengo los mejores hiperparametros utilizó VottingRegressor para combinar métodos y convino el random forest con estos hiperparametros con algún otro modelo como por ejemplo KNN u otro árbol con distintos hiperparametros

Al final generó el csv respuesta .

Nota : Al hacer todo esto y probar cada vez con más features y transformaciones distintas no he podido conseguir buenos resultados en kaggle 626993.53460

Otra forma en la que encaré el problema fue creando un modelo para cada año es decir para las publicaciones del 2012 entreno un modelo que solo se preocupe de predecir precios de publicaciones que fueron del 2012 , y así para los demás años por lo tanto tendria 5 modelos distintos para cada año :

- *modelo2012*
- *modelo2013*
- *modelo2014*
- *modelo2015*
- *modelo2016*

Para cada modelo calculo sus errores e intentar mejorarlos individualmente, entonces agarro el set de test lo separo por años y para cada uno de los set hago las predicciones con el modelo correspondiente, al final vuelvo a juntarlos todos y género el csv para la competencia de kaggle

Nota : Pensé que hacer esto tenía más lógica ya que por año los precios van subiendo y es mejor predecir cosas que se entrenaron en el mismo espacio de tiempo pero lamentablemente para mi , los resultados en kaggle no fueron buenos y no pude mejorar mi score 647152.47503

Conclusiones Finales *: Después de probar todo lo antes mencionado no quede satisfecho con los resultados pienso que no inverti bien el tiempo en los features y en el manejo adecuado de los nulls , pienso que el modelo que elegí no es malo pero no logré sacarle jugo , cosas para mejorar sería trabajar más en los features y seguir con la idea de hacer predicciones por años y no un modelo que englobe a todos los años por igual quizas deberia tambien agregar ruido ya que al splitear por años obtengo muy pocos datos para entrenar también es algo que tenía pensado, el ruido serían publicaciones en todos los años y agrega publicaciones de años cómo 2017 con precios más altos en promedio q el 2016. También me hubiera gustado solucionar los casos de overfitting que estaba obteniendo en las últimas predicciones pero no pude hacerlo correctamente intente mezclando 2 modelos con voting regressor mezclar random forest con KNN y como resultado me dan predicciones con más error pero no overfitea tanto pero no conseguí buenos resultados tampoco . Pienso muy fuertemente que la clave es cómo llenar los nulls hace mucha diferencia eso y me hubiera gustado dedicarme a estudiar cómo llenarlos apropiadamente .*