Proyecto Final, Avance Módulo 3

David Rincon Morales

Noviembre 2022

Contents

1 Introducción

El propósito de este documento es mostrar avance sobre el proyecto final para el Diplomado en Ciencia de Datos. Para dicho proyecto, se eligió un dataset de la página de Kaggle, el cual contiene información de la venta de casas en la ciudad de Portland, Estados Unidos entre julio 2020 y julio 2021. Los datos se pueden obtener a través de la siguiente liga:

https://www.kaggle.com/datasets/threnjen/portland-housing-prices-sales-jul-2020-jul-2021

Se decidió usar este dataset ya que contiene toda la información relacionada a cada una de las casas vendidas. Desde ciudad y código postal, hasta información relacionada a ventas anteriores (si aplica) y escuelas cercanas a cada vivienda. Es por esto que nos puede servir a lo largo del diplomado; para modelos supervisados y modelos no supervisados.

2 Datos

2.1 Pretratamiento

El autor de la página de Kaggle originalmente extrajo los datos de la API pública de Zillow, el sitio más grande de venta de bienes de raíces a nivel mundial; por lo que, están relativamente limpios los datos. Lo principal al hacer la lectura, fue identificar cuáles serían las principales columnas a preservar; ya que, dependiendo del historial de la vivienda, puede tener mucha información relacionada a precios históricos. En nuestro caso, de un total de 348 columnas, filtramos algunas variables que pór el momento podemos evitar y terminamos con un dataset inicial de 27 columnas.

	address_city	address_zipcode	bathrooms	bedrooms	brokerageName	daysOnZillow	homeStatus	homeType
0	Fairview	97024	3.0	3.0	NaN	25.0	RECENTLY_SOLD	TOWNHOUSE
1	Fairview	97024	3.0	3.0	Harcourts Real Estate Network Group	53.0	RECENTLY_SOLD	SINGLE_FAMILY
2	Gresham	97080	3.0	4.0	ERA Freeman & Associates	11.0	RECENTLY_SOLD	SINGLE_FAMILY
3	Portland	97230	1.0	3.0	Premiere Property Group, LLC	11.0	RECENTLY_SOLD	SINGLE_FAMILY
4	Gresham	97030	3.0	6.0	NaN	14.0	RECENTLY_SOLD	APARTMENT

2.2 Creación de Variables

Para la creación de variables, tuvimos 3 casos:

1. ***p_bathrooms***: Esta variable es la relación entre el número de baños y el número de recámaras en la vivienda (e.g. 1 significa que hay el mismo número de baños que de recámaras, mientras que 1.5 significa que hay más baños).

2. ***p_lastPrice***: Es la relación entre el último precio de venta vs el tamaño del terreno (lotSize)

3. ***p_livingArea***: Es la relación entre el "área de vivienda" (tamaño de construcción en México) y el tamaño del terreno

2.3 Previa Variable Target (Casa Premium)

El objetivo de poder clusterizar es saber cuál casa es premium o no, dependiendo de sus características; para ello, vamos a revisar nuestros datos para tener una base de antescedente. En nuestro caso, creamos un nuevo dataframe llamado ***"prev"***, el cual nos dice si una casa es premium o no en base a 3 variables:

1. ***bedrooms***

2. ***bathrooms***

3. ***lastSoldPrice***

Luego, empezamos a analizar los diferentes percentiles en las variables, llegando a la conclusión que viviendas con 5 recámaras o más pueden ser consideradas premium dadas sus características; por ejemplo:

- **Número de recámaras** Sólo cerca del 11% de las casas del conjunto de datos cuenta con al menos 5 recámaras.

- **Número de baños** Para el número de baños, el 0.25 percentil de las casas premium son 3 baños, el cual representaría el 0.75 percentil en todo el conjunto de datos.

- **Último precio de venta** En el 0.25 percentil de este subconjunto de casas premium, el promedio del último precio de venta era substancialmente mayor al promedio en el conjuntos completo de datos (866K USD vs 585K USD).

Al final del pretratamiento y creación de variables, definimos nuestras variables de la siguiente manera:

- **Unidad Muestral**: *zpid* (identificador del listado)
- **Variables Discretas**: *address_city, address_zipcode, brokerageName, homeStatus, homeType, rentalApplicationsAcceptedType*
- **Variables Continuas**: *bathrooms, bedrooms, daysOnZillow, lastSoldPrice, livingArea, lotSize, propertyTaxRate, rentZestimate, restimateHighPercent, restimateLowPercent, solarPotential_buildFactor, solarPotential_climateFactor, solarPotential_electricityFactor, solarPotential_solarFactor, solarPotential_sunScore, taxAssessedValue, taxAssessedYear, yearBuilt, zestimateHighPercent, zestimateLowPercent, p_bathrooms, p_lastPrice, p_livingArea*

3 Exploración de Datos

3.1 Variables Discretas

Para nuestras variables discreta, completamos las variables faltantes como "WITHOUT CATEGORY".

3.1.1 Normalización

Siguiendo con nuestro tratamiento de variables discretas, realizamos la normalización de variables; con esto, podemos seguir identificando cuáles variables pueden ser unarias. Al termino de este proceso, concluímos que las variables: ***v_homeStatus, v_rentalApplicationsAcceptedType*** eran unarias, ya que todos sus datos se centraban en una sola categoría.

```
Frequencies table for variable v_homeStatus

                   FA   FR    FAA  FRA
RECENTLY_SOLD  25692  1.0  25692  1.0

Frequencies table for variable v_homeType

                   FA        FR    FAA       FRA
SINGLE_FAMILY  22582  0.878951  22582  0.878951
TOWNHOUSE       1560  0.060719  24142  0.939670
CONDO           1550  0.060330  25692  1.000000

Frequencies table for variable v_rentalApplicationsAcceptedType

                     FA   FR    FAA  FRA
REQUEST_TO_APPLY  25692  1.0  25692  1.0
```

3.2 Variables Continuas

3.2.1 Análisis Univariado

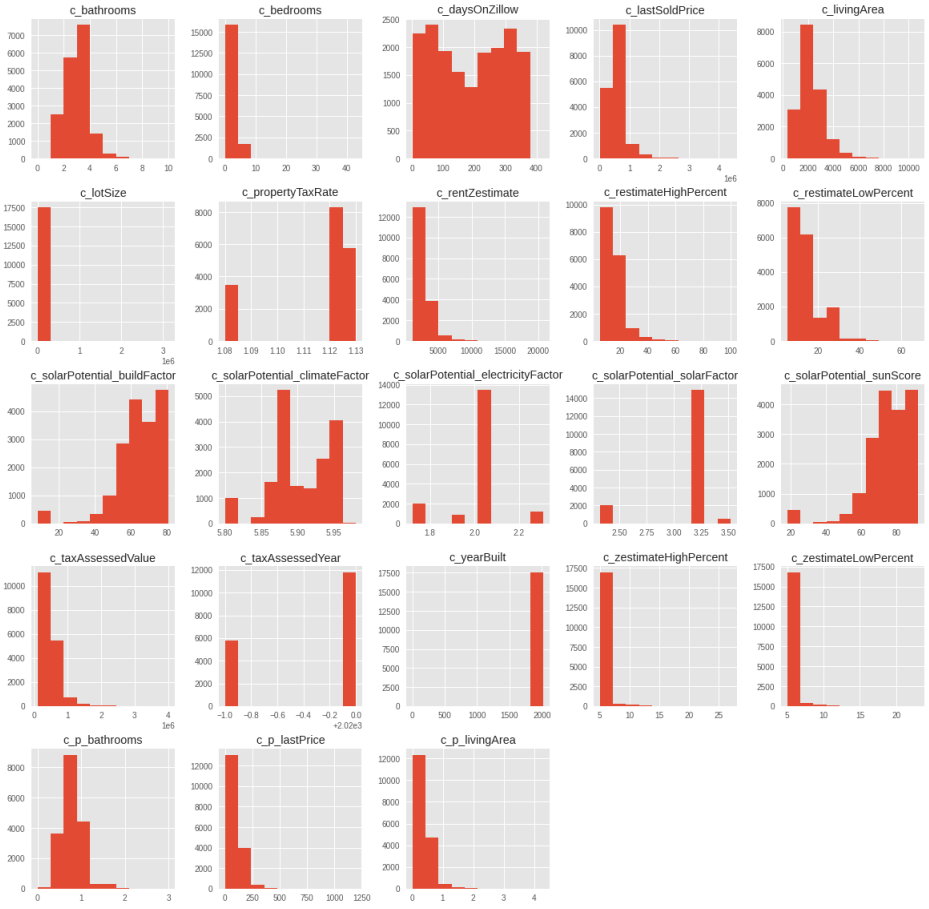
Para poder hacer el primer análisis univariado en las continuas, aplicamos la funcion "describe" que ofrece pandas. Con esta, pudimos ver información

valiosa con los percentiles; por ejemplo, hay varias viviendas que aparecen con 0 recámaras y 0 baños, esto provoca que tengamos valores indefinidos en las variables que creamos; es por eso, que reemplazamos los infinitos con "0". Además, tenemos valores extremos que se alejan considerablemente del 0.99 percentil. Es por esto que, a comparación de otros casos, seguimos con la eliminación de extremos.

	c_bathrooms	c_bedrooms	c_daysOnZillow	c_lastSoldPrice	c_livingArea	c_lotSize
count	25208.000000	24922.000000	25686.000000	2.569200e+04	25227.000000	2.280800e+04
mean	2.561209	3.356593	184.673013	5.850185e+05	2179.004321	2.742676e+04
std	1.009345	1.001984	113.423035	4.510903e+05	1129.526349	1.392288e+06
min	0.000000	0.000000	1.000000	3.000000e+02	0.000000	0.000000e+00
1%	1.000000	1.000000	8.000000	1.499819e+05	640.520000	1.306000e+03
10%	1.000000	2.000000	33.000000	3.150000e+05	1078.000000	3.049000e+03
50%	3.000000	3.000000	185.000000	5.220000e+05	1996.000000	6.969000e+03
90%	4.000000	5.000000	339.000000	8.799962e+05	3425.000000	1.698800e+04
99%	5.000000	6.000000	361.000000	1.900000e+06	5642.220000	2.182350e+05
max	23.000000	43.000000	422.000000	4.100000e+07	51290.000000	2.087221e+08

3.2.2 Extremos

Para el caso de los extremos, en vez de usar un método como la cerca percentil, usamos una método dentro de Scikit Learn llamado "LocalOutlierFactor"; el cual, nos predecirá si un registro es extremo o no. Este proceso nos va a quitar cerca del 10% de nuestros datos

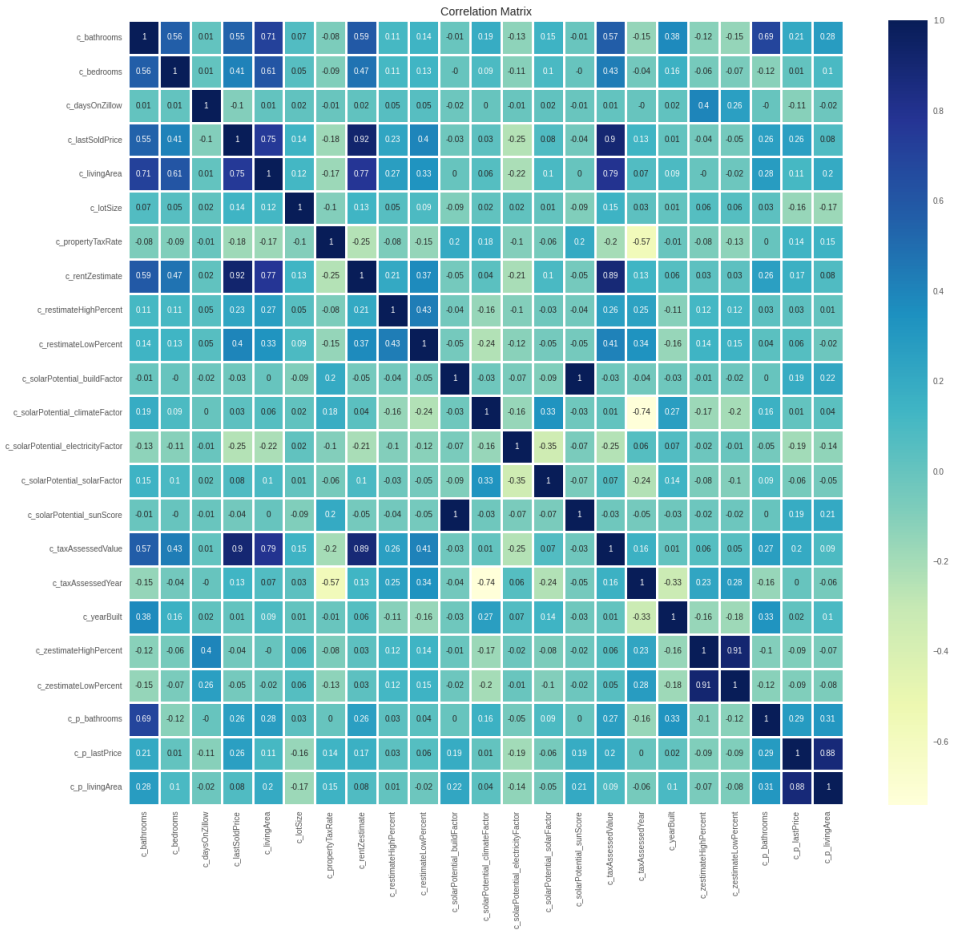


3.2.3 Valores Ausentes

Gracias a nuestro proceso de pretratamiento y selección de Outliers, nos damos cuenta que ya no tenemos valores ausentes

3.2.4 Análisis Bivariado (Correlación)

Para el análisis de correlación, revisamos que no hubieran patrones dentro de los datos usando una "heatmap".



Nos damos cuenta que hay algunas variables con correlación, pasamos a un análisis de multicolinealidad para poder seleccionar nuestras mejores variables.

3.2.5 Multicolinealidad

Para poder identificar la multicolinealidad de las variables, utilizamos varclushi, el cual nos ayuda a realizar clusters de variables, y de esta manera identificar las mejores a usar. En nuestro caso, se realizaron 7 clusters diferentes:

	Cluster	Variable	RS_Own	RS_NC	RS_Ratio	id
0	0	c_rentZestimate	0.882475	0.215008	0.149715	1
1	0	c_taxAssessedValue	0.876411	0.205261	0.155509	2
2	0	c_lastSoldPrice	0.863820	0.192684	0.168683	3
3	0	c_livingArea	0.802067	0.286445	0.277390	4
4	0	c_bedrooms	0.385511	0.056698	0.651423	5
5	0	c_solarPotential_electricityFactor	0.101292	0.027805	0.924412	6
6	1	c_solarPotential_climateFactor	0.769487	0.057981	0.244701	1
7	1	c_taxAssessedYear	0.755468	0.319750	0.359474	2
8	1	c_solarPotential_solarFactor	0.279689	0.016037	0.732050	3
9	1	c_yearBuilt	0.295258	0.147899	0.827063	4
10	2	c_solarPotential_sunScore	0.999879	0.046244	0.000127	1
11	2	c_solarPotential_buildFactor	0.999879	0.047225	0.000127	2
12	3	c_zestimateHighPercent	0.933423	0.051707	0.070208	1
13	3	c_zestimateLowPercent	0.858485	0.073599	0.152758	2
14	3	c_daysOnZillow	0.314711	0.004507	0.688392	3
15	4	c_p_lastPrice	0.913636	0.076603	0.093529	1
16	4	c_p_livingArea	0.914308	0.104767	0.095720	2
17	4	c_lotSize	0.111779	0.018120	0.904613	3
18	5	c_p_bathrooms	0.847100	0.086612	0.167399	1
19	5	c_bathrooms	0.847100	0.447450	0.276717	2
20	6	c_restimateHighPercent	0.713933	0.064033	0.305637	1
21	6	c_restimateLowPercent	0.713933	0.150709	0.336830	2
22	7	c_propertyTaxRate	1.000000	0.082998	0.000000	1

Por lo que tenemos que proceder a hacer una reducción de dimensiones

4 Ingeniería de Variables

En nuestro caso, hicimos una ingeniería de variables sencilla. Obtuvimos los dummies de nuestras variables discretas; así cómo, realizamos el escalado de nuestras variables continuas de dos diferentes maneras:

1. ***Standard Scaler***
2. ***MinMax Scaler***

Esto para poder comparar resultados y verificar cuál se nos acomoda de mejor forma.

5 Escalado Dimensional (PCA)

En la parte de escalamiento, analizamos cuántos componentes necesitaríamos para representar el 90% de los datos. Para ello, obtuvimos las siguientes gráficas.

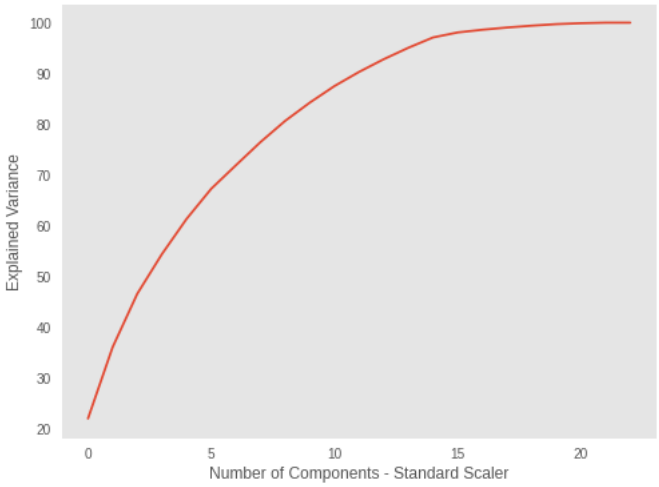


Figure 1: Reducción con Standard Scaler

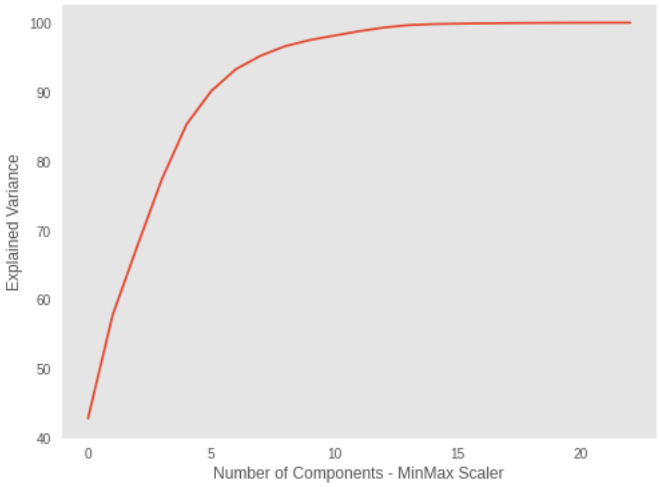


Figure 2: Reducción con MinMax Scaler

Si nos ponemos a comparar la varianza explicada en estods dos casos tenemos lo siguiente:

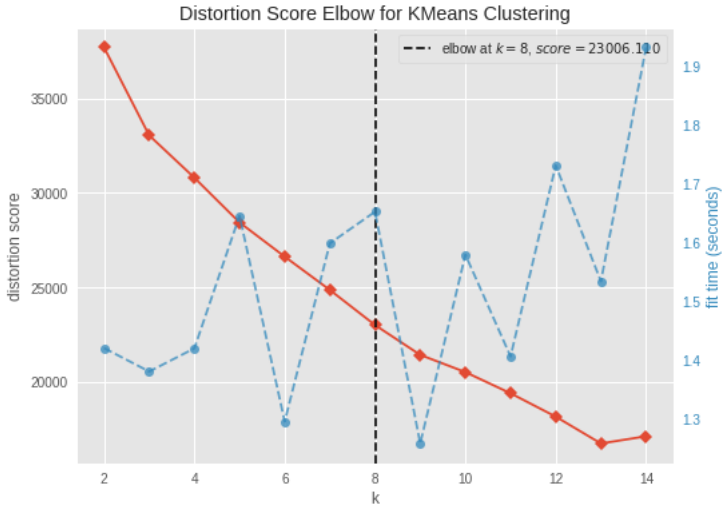
- **Standard Scaler**: Con Standard Scaler necesitamos 12 componentes para apenas poder explicar el 90% de los datos
- **MinMax Scaler**: Con MinMax, con tan solo 6 componentes podemos explicar el 90% de los datos. Esto por esto que elegimos esta ruta.

6 Clustering

6.1 Número de Clusters

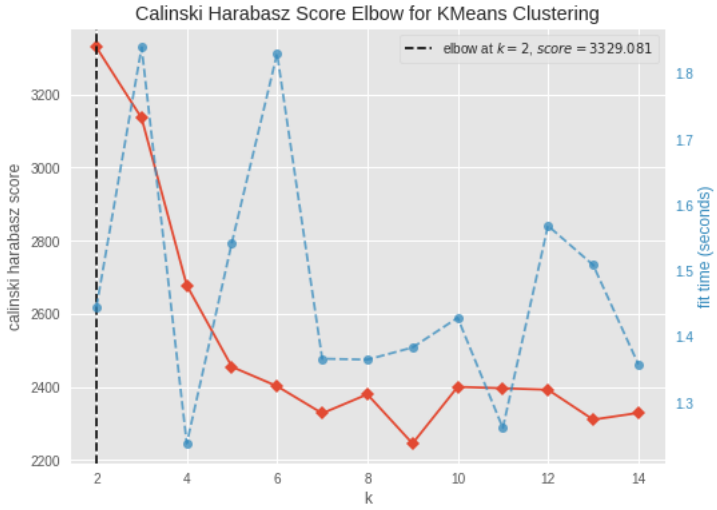
Para la selección del número de Clusters hicimos los 3 siguientes análisis.

6.1.1 Codo



6.1.2 Silueta

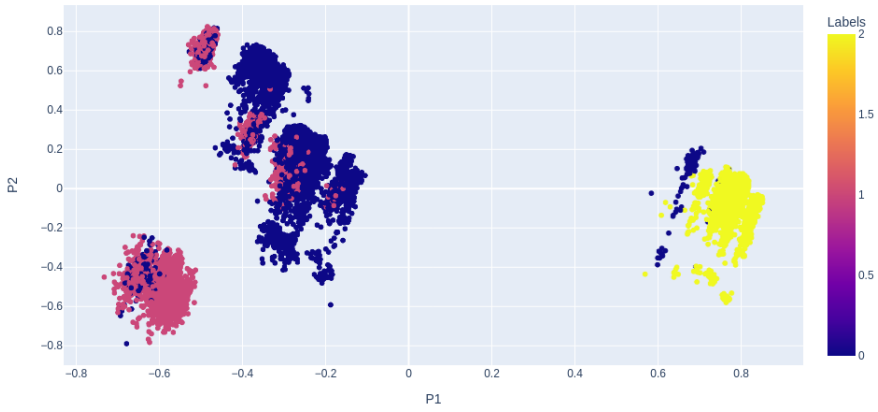
6.1.3 Calinski



6.2 Visualización de Clusters

A partir de los análisis anteriores, decidimos relizar 3 y 4 clusters y así ver qué tipo de perfilamiento nos es más conveniente

6.3 3 Clusters



6.4 4 Clusters

7 Perfilamiento

El mejor perfilamiento obtenido se realizcó a partir de 3 clusters; ya que, como vimos en las visualizaciones, segmenta de mejor manera los tipos de casas. Las diferentes características son las siguientes.

- Cluster 0: Casas Promedio

  Son las casas más viejas, fueron construidas en los 1950's.

  Tienen el sqft más caro con 109 USD.

  Tienen el mejor potencial de adaptación a energía eléctrica

- Cluster 1: Casas Premium

  Son las casas más costosas. Su precio promedio se encuentra cerca de 625K USD.

  Su renta es la más cara, con un precio de 3169 USD mensuales (10% extra que las casas promedio)

  Su tamaño de terreno es más del doble que las demás.

- Cluster 2: Casas "Entry Level"

  Son las casas más baratas, tienen un precio promedio de 525K USD (comparado con 600k+ de las casas promedio).

  Son las más nuevas, fueron construídas en los 1990's.

  Su cantidad de baños y recámaras tiende a ser el mismo

A Anexo Modulo 2

Para poder realizar los modelos de clasificación en el módulo 2, se realizó una transformación WoE a los datos; por lo que, este anexo contendrá información a partir de dicho procedimiento. **Este procedimiento NO toma en cuenta el módulo 3** ya que se hizo un tratamiento de datos distinto.

A.1 Discretización

Por último, discretizamos nuestras variables obtenidas del análisis de multicolinealidad; de tal manera que, terminamos con 16 variables.

```
16

['d_rentZestimate_2',
 'd_rentZestimate_3',
 'd_rentZestimate_4',
 'd_rentZestimate_5',
 'd_solarPotential_buildFactor_2',
 'd_solarPotential_buildFactor_3',
 'd_solarPotential_buildFactor_4',
 'd_solarPotential_buildFactor_5',
 'd_daysOnZillow_2',
 'd_daysOnZillow_3',
 'd_daysOnZillow_4',
 'd_daysOnZillow_5',
 'd_restimateLowPercent_2',
 'd_restimateLowPercent_3',
 'd_restimateLowPercent_4',
 'd_restimateLowPercent_5']
```

A.2 Poder predictivo

Para poder obtener el poder predictivo de nuestras variables continuas y discretas, utilizamos el valor de *Information Value*. Luego, seleccionamos las variables con un valor iv mayor a 0.08 y menor a 0.8. A lo que nos lleva a las 3 mejores variables finales: ***d_restimateLowPercent_5, n_address_zipcode*** y ***n_homeType***

A.2.1 Transformación WoE y TAD

Por último, para crear la TAD, pasamos nuestros datos y mejores variables a través de un mapa WoE

	zpid	w_d_restimateLowPercent_5	w_n_address_zipcode	w_n_homeType	Premium_House
0	48211221	-0.820979	0.080927	-0.129150	No
1	80926247	0.423420	0.080927	-0.129150	No
2	48576711	0.423420	-1.136887	-0.129150	No
3	248439455	-0.820979	0.080927	2.888465	No
4	53933245	-0.130543	0.080927	-0.129150	No

A.3 Dashboard

Para poder analizar nuestra información, exportamos nuestra información antes de hacer la discretización de variables continuas. En el dashboard, se encuentra información general del mercado de vivienda en Portland; así como, una sección espefícia de viviendas "Premium". Para poder acceder al dashboard, favor de dirigirse al siguiente link:

https://datastudio.google.com/reporting/42cc0d0c-59d2-4048-90c1-4c88a925fa68

A.4 Modelado

Nuestra problemática es de tipo clasificación, por lo que tenemos diversas opciones para utilizar, desde regresiones logísticas hasta técnicas más robustas como redes neuronales. En nuestro caso, vamos a utilizar dos técnicas diferentes para poder ver cuál nos ayuda a predecir de mejor manera.

- **Red Neuronal**
- **Árbol de Decisión**

En ambos casos ya no tenemos que escalar nuestro datos dado que utilizamos una transformación WoE.

A.4.1 Árbol de Decisión

Para el Árbol de Decisión a implementar, vamos a utilizar la clase "DecisionTreeClassifier" dentro de la librería de Scikit-learn. Una vez que le pasamos nuestros datos de entrenamiento y sacamos la matriz de confusión, nos damos cuenta que tenemos un problema, tenemos muchas predicciones "falso-positivo".

```
tree_metrics(Tree_Classifier,Xt_t, Yt_t)
```
executed in 68ms, finished 21:53:25 2022-09-30

```
Roc Validate: 0.695
Acc Validate: 0.908
Matrix Conf Validate:
[[10413     0]
[ 1059     0]]
```

Las predicciones falso positivas, son valores que nuestro modelo predice como "negativas" cuando en realidad son positivas. Esto lo podemos solucionar al hiper parametrizar nuestro modelo.

Hiper Parametrización La Hiper Parametrización es una forma en la que podemos obtener las mejores "opciones" para el modelo que estemos trabajando. Esto lo hace al iterar una cierta cantidad de veces sobre todos nuestros datos de entrenamiento.

En este caso, vamos a realizar 10K iteraciones sobre nuestros datos de entrenamiento. A lo cual obtenemos mucho mejor resultados.

```
tree_metrics(best_tree,Xt_t,Yt_t)
```
executed in 68ms, finished 21:54:41 2022-09-30

```
Roc Validate: 0.668
Acc Validate: 0.794
Matrix Conf Validate:
[[8671 1742]
[ 622  437]]
```

```
tree_metrics(best_tree,Xt_v,Yt_v)
```
executed in 38ms, finished 21:54:50 2022-09-30

```
Roc Validate: 0.653
Acc Validate: 0.791
Matrix Conf Validate:
[[3716  768]
[ 262  171]]
```

En este caso, aunque ya nuestros valores de accuracy, no estén cerca de los 0.9, predecimos de mejor manera cuáles casas son premium o no ya que al menos no deja en 0 casas premium.

A.4.2 Redes Neuronales

Para el modelo de Redes Neuronales, usamos la clase llamada "MLPClassifier" dentro de scikit-learn. Una vez que mandamos a llamar a la clase para poder modelar, y le pasamos nuestros datos de entrenamiento, nos damos cuenta de tener unos primeros resultados no tan buenos.

```
nn_metrics(mlp,Xnn_t,Ynn_t)
```
executed in 109ms, finished 21:55:03 2022-09-30

```
Acc Validate: 0.908
Matrix Conf Validate:
[[10413     0]
[ 1059     0]]
```

```
nn_metrics(mlp,Xnn_v,Ynn_v)
```
executed in 69ms, finished 21:55:04 2022-09-30

```
Acc Validate: 0.912
Matrix Conf Validate:
[[4484    0]
[ 433    0]]
```

Nos pasa similar que las métricas iniciales que en el árbol de decisión, nuestra red no está prediciendo de forma correcta sin importar de tener "buen" accuracy; esto lo notamos dada la cantidad de falsos negativos en nuestras predicciones.

Hiper Parametrización Esto lo vamos a volver a intentar solucionar mediante la hiper parametrización. En este caso, nos damos cuenta que nuestra neurona sigue sin mejorar ya que sigue arrojando muchos falsos negativos, a lo que podemos concluír a que, por ahora, nuestro árbol es mejor opción.

```
nn_metrics(model_best,Xnn_t,Ynn_t)
```
executed in 155ms, finished 21:55:29 2022-09-30

```
Acc Validate: 0.908
Matrix Conf Validate:
[[10413     0]
[ 1059     0]]
```

```
nn_metrics(model_best,Xnn_v,Ynn_v)
```
executed in 90ms, finished 21:55:30 2022-09-30

```
Acc Validate: 0.912
Matrix Conf Validate:
[[4484    0]
[ 433    0]]
```

A.4.3 Conclusiones y posibles mejoras para el modelo

1. La mejor opción de predicción actualmente es un árbol de decisión dada su capacidad de predecir de mejor manera.

2. Posiblemente las 3 variables que creamos no fueron lo suficientemente buenas para ayudar con la predicción del modelo. Podemos crear nuevas variables y ver el comportamiento.

3. Para nuestro modelo, no tomamos en cuenta otro tipo de datos sobre cada propiedad (texto), podemos analizar si hay posibilidad de incluír estos datos para futuras iteraciones.

4. Finalmente, podemos intentar con otro tipo de técnicas de modelado para revisar si podemos predecir aún mejor sobre nuestros datos.