Universidad Iberoamericana de Puebla

Ingeniería en Sistemas Computacionales

Tecnologias Emergentes en Computación

Analisis y Predicción de Precios de Viviendas

Diego Pacheco Valdez

Otoño 2024

Índice

1. Selección y Análisis del Modelo de Machine Learning

1.1 Analisis de la Problemática

Para poder decidir el tipo de modelo de machine learning es necesario primero realizar un analisis general de nuestra base de datos de precios de viviendas y en si, la problemática a resolver.

Debido a que estamos trabajando con datos indentados sabemos que se trata de un problema de Aprendizaje Supervisado, y gracias a que sabemos que el resultado final debe ser una prediccion del precio de una casa (es decir, un valor exclusivo de tipo numerico) sabemos que al final se trata de un problema de Regresión.

Con esto en mente, tenemos algunas opciones en cuanto a modelos con los cuales analizar la base de datos.

- Regresión Lineal
    - El modelo de regresión lineal busca aquellas relaciones entre la variable dependiente (en nuestro caso se trataria del precio de la casa) y las variables independientes. La regresión lineal es util cuando se tiene una baja cantidad de categorias y que estas tengan una relación lineal con la variable dependiente.
- Regresión Polinominal
    - El modelo de regresión polinominal funciona sobre todo cuando se tienen relaciones no lineales entre la variable dependiente y la independiente
- Regresión de Arboles de Decisión
    - Este algoritmo de regresión construye un arbol de decisión con la intención de predecir la variable dependiente, de esta forma se van creando nodos hasta conseguir una predicción continua.
- Regresión de Bosque Aleatorio
    - En este modelo se combinan varios arboles de decisión y se promedian sus predicciones, este es especialmente útil cuando se tienen grandes volumenes de datos

1.2 Analisis General de los Datos Almacenados

Para poder definir esto, primero haremos un analisis mas profundo de nuestros datos con el uso de la librería pandas.

```
import pandas as pd

# Cargamos la base de datos
df = pd.read_csv('data.csv')

# Se nos muestran las columnas, el tipo de dato que tiene y
```

```python
# la cantidad de valores vacios en cada una
print(df.info())
# Se nos muestra cuantos ceros tiene cada columna
print((df == 0).sum())
# Se nos muestran valores estadisticos de los valores de
# aquellas columnas con valores numericos
print(df.describe())
```

Se nos muestra entonces…

No se tienen valores del tipo null. Se tienen 13 columnas de tipo numerico y cinco de tipo categorico.

```
#   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   date           4600 non-null   object
 1   price          4600 non-null   float64
 2   bedrooms       4600 non-null   float64
 3   bathrooms      4600 non-null   float64
 4   sqft_living    4600 non-null   int64
 5   sqft_lot       4600 non-null   int64
 6   floors         4600 non-null   float64
 7   waterfront     4600 non-null   int64
 8   view           4600 non-null   int64
 9   condition      4600 non-null   int64
 10  sqft_above     4600 non-null   int64
 11  sqft_basement  4600 non-null   int64
 12  yr_built       4600 non-null   int64
 13  yr_renovated   4600 non-null   int64
 14  street         4600 non-null   object
 15  city           4600 non-null   object
 16  statezip       4600 non-null   object
 17  country        4600 non-null   object
```

De todas las columnas, solo se tienen 7 columnas donde se encuentren valores que sean igual a cero, en algunos casos este 0 representa una calificación mientras que en otros implica la ausencia de la caracteristicas de su propia columna.

```
date                0
price               49
bedrooms            2
bathrooms           2
sqft_living         0
sqft_lot            0
floors              0
waterfront          4567
view                4140
condition           0
sqft_above          0
sqft_basement       2745
yr_built            0
yr_renovated        2735
street              0
```

```
city                                    0
statezip                                0
country                                 0
```

1.3 Análisis Descriptivo con Pandas

Por último, tenemos también los datos estadísticos de los valores numéricos dados gracias al uso de la librería de pandas.

Feature	**Count**	**Mean**	**Std**	**Min**	**25%**	**50%**	**75%**	**Max**
price	4600	5.51963E+05	5.63835E+05	0	3.22875E+05	4.60944E+05	6.54963E+05	2.659E+07
bedrooms	4600	3.40087	0.908848	0	3	3	4	9
bathrooms	4600	2.16082	0.783781	0	1.75	2.25	2.5	8
sqft_living	4600	2139.35	963.207	370	1460	1980	2620	13540
sqft_lot	4600	1.48525E+04	3.58844E+04	638	5000.75	7683	1.10013E+04	1.07422E+06
floors	4600	1.51206	0.538288	1	1	1.5	2	3.5
waterfront	4600	0.007174	0.084404	0	0	0	0	1
view	4600	0.240652	0.778405	0	0	0	0	4
condition	4600	3.45174	0.67723	1	3	3	4	5
sqft_above	4600	1827.27	862.169	370	1190	1590	2300	9410
sqft_ basement	4600	312.082	464.137	0	0	0	610	4820
yr_built	4600	1970.79	29.7318	1900	1951	1976	1997	2014
yr_ renovated	4600	808.608	979.415	0	0	0	1999	2014

El analisis de esto nos lleva a las siguientes conclusiones

- El precio promedio suele tratatarse de $551,963 con un alto sesgo hacia los inmuebles de precio alto, ademas de la exstencia de valores 0 los cuales deben ser elliminados

- En promedio cada casa tiene 3 Cuartos y 2 baños
- Se tienen en promedio salas de 2140 pies, lotes de 14,853 pies.
- La mitad de las casas no tienen sotanos.
- La gran mayoria de propiedasd tienen entre uno y dos pisos.
- En general casi ninguna casa tiene una vista que se considere buena y la gran mayoria de casas no tienen vista a cuerpo maritimos.
- En promedio las casas tienen una condición de 3.5 con la mayoria estando en el rango de 3 a 4
- En promedio las casas se construyeron en 1971, la mas reciente habiendose construido en 2014. Y en si alrededor de la mitad no han sido renovadas.

Esta información por si sola aun no nos indica el modelo adcecuado que nos pueda ayudar para este caso especifico, pero nos va indicando que debido a la complejidad de datos es muy probable que se traten de relaciones no lineales.

2 Limpieza Superficial y Transformación de datos

El siguiente paso seria la creación de una matriz de relación, para la cual se necesitara pasar todo dato de tipo objeto a una versión númerica. A continuación se mostrara los procesos realizados para llevar esto acabo. Asi tambien se aprovecha la oportunidad para eliminar cualquier dato repetido asi como los datos donde el precio es igual a cero, la categoria pais al tratarse del mismo, la categoria calle debido a su extremadamente alta variación y la "fecha" pues se refiere a la fecha en que se agrego el valor a la base de datos y esto no tiene relación con los precios

Para la transformación de datos se usa la tecnica de label encoding el cual consiste en convertir un valor en su propia categoria (Es decir, si tuviesemos el valor "seattle" este seria asignado un numero, por ejemplo el 1, y toda instancia con el valor "seattle" seria remplazada con el numero asignado). En el caso especifico del codigo ZIP, simplemente se eliminaran los caracteres ya que indican que todas las casas se encuentran en el mismo estado a cambio de dejar los valores numericos del código.

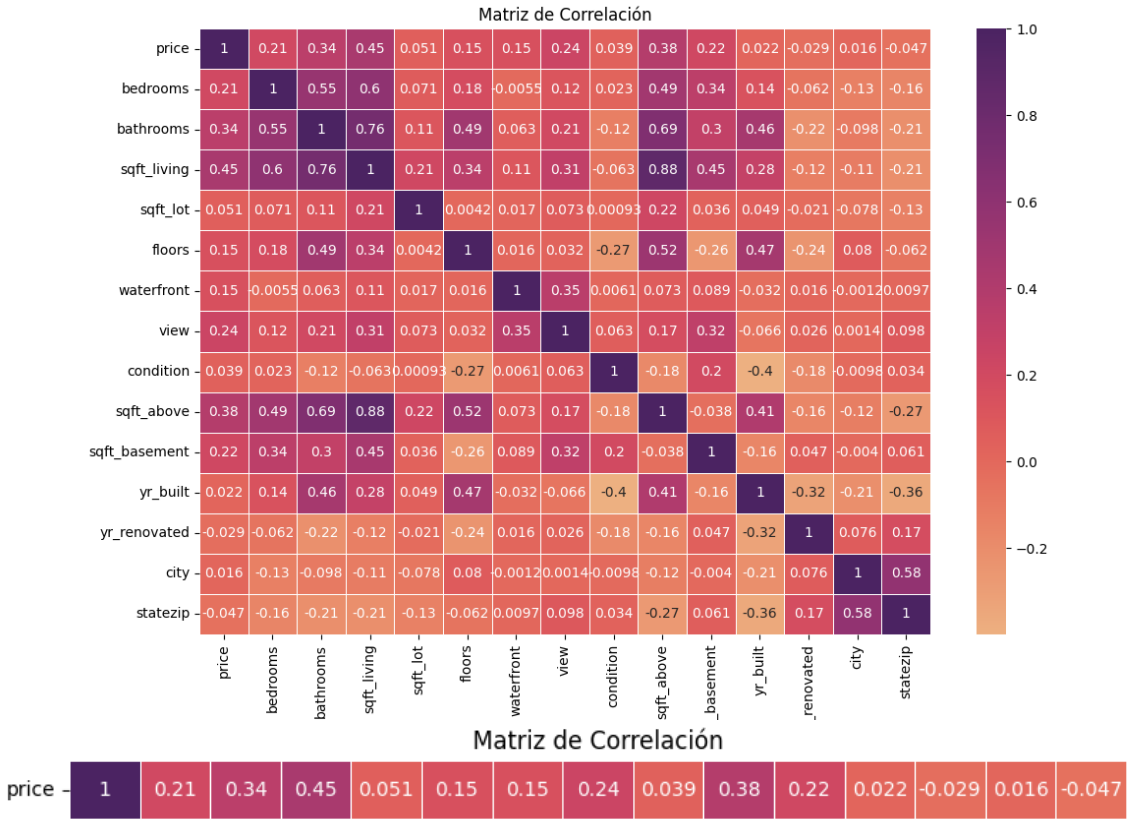
```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
df = df.drop_duplicates()
df = df.drop('date', axis = 1)
df = df[df['price'] != 0]
df = df.drop('country', axis = 1)
df = df.drop('street', axis=1)
df['city'] = LabelEncoder().fit_transform(df['city'])
df['statezip'] = pd.to_numeric(df['statezip'].str.replace('WA ', ''))
print(df.info())
```

Estos cambios nos dan como resultado que todas las columnas restantes sean de numericas y permite la creación de la matriz de relación

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# La matriz de correlacion nos muestra la importancia de
# ciertos valores en el resultado de otras columnas
correlation_matrix = df.corr()
print(correlation_matrix)

plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='flare', linewidths=0.5)
plt.title('Matriz de Correlación')
plt.show()
```

3. Matriz de Relación y sus Implicaciones.

Como se puede apreciar en la siguiente grafica, cada valor tiene un impacto distinto en su relación con el precio, en si todos teniendo un impacto menor al 0.60.



Por el momento, los valores con mayor relación al precio son:

- Pies^2 de sala de estar: 0.45
- Pies^2 de Espacio Superior: 0.38
- Numero de baños: 0.34

Algunos datos que uno esperaria tuvieran un mayor impacto son:

- Pies^2 del Sotano: 0.22
- Pies^2 del Lote: 0.051
- Condición: 0.039

4 Primer Modelo de Aprendizaje y sus Resultados

Con esto en mente, podemos definir el modelo de aprendizaje a utilizar; se tratara del algoritmo de regresión de bosque aleatorio debido a la alta cantidad de datos con la que estamos trabajando.

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestRegressor

df = pd.read_csv('data.csv')

print(df.head)
print(df.info())
print((df == 0).sum())
print(df.describe())

df = df.drop_duplicates()
df = df.drop('date', axis = 1)
df = df[df['price'] != 0]
df = df.drop('country', axis = 1)
df = df.drop('street', axis=1)
df['city'] = LabelEncoder().fit_transform(df['city'])
df['statezip'] = pd.to_numeric(df['statezip'].str.replace('WA ', ''))

correlation_matrix = df.corr()
print(correlation_matrix)

plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='flare', linewidths=0.5)
plt.title('Matriz de Correlación')
plt.show()
```

```
# En la variable X almacenamos las variables independientes
# que se relacionan con el resultado
X = df.drop('price', axis=1)

# Y aqui se guarda la variable dependiente,
# es decir, el resultado
y = df['price']

# Dividimos los datos en datos de entrenamientos y datos de
# prueba, el primero le dara al modelo la oportunidad de
# identificar relaciones entre las variables mientras que
# el segundo le permitira poner a prueba las relaciones
# encontradas

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Entrenamos al modelo
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Realizamos predicciones con el modelo
y_pred = model.predict(X_test)

# Calculo de Metricas Medición
print(f'MAE: {mean_absolute_error(y_test, y_pred)}')
print(f'MSE: {mean_squared_error(y_test, y_pred)}')
print(f'R2: {r2_score(y_test, y_pred)}')

# Grafica Comparativa
df_comparacion = pd.DataFrame({
    'Index': range(len(y_test)),
    'Valores Reales': y_test.values,
    'Valores Predichos': y_pred
})

plt.figure(figsize=(12, 6))
sns.lineplot(x='Index', y='Valores Reales', data=df_comparacion, label='Valores
Reales', color='orange')
sns.lineplot(x='Index', y='Valores Predichos', data=df_comparacion,
label='Valores Predichos', color='purple')
plt.xlabel('Índice de Muestras')
plt.ylabel('Valor')
plt.title('Comparación de y_test y y_pred')
plt.legend()
plt.grid(True)
plt.show()
```

```
MAE:            122266.96580864681
MSE:            59306376170.77017
R2:             0.6013563189815866
```

Las metricas mostradas al final nos muestran la efectividad de nuestro modelo, a continuación se explicara lo que estas significan.

4.1 Metricas de Medición y sus Resultados

MAE Error Absoluto Medio

El MAE es un promedio de cuanto se llega a equivocar nuestro modelo, se calcula por medio de dividir el promedio de la diferencia absoluta entre los valores reales y los valores predichos por el modelo.

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$$

En el caso de nuestro modelo, podemos interpretar nuestro resultado como lo alejado que solemos estar del precio real, en este caso. $122,267 dolares de diferencia. Idealmente este numero debera acercarse lo mas posible a cero.

MSE Error Medio Cuadrado

El MSE es un el cuadrado del promedio de cuanto se llega a equivocar nuestro modelo, al ser basicamente la metrica anterior al cuadrado, funciona como una penalización mas dura en aquellos errores grandes.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2$$

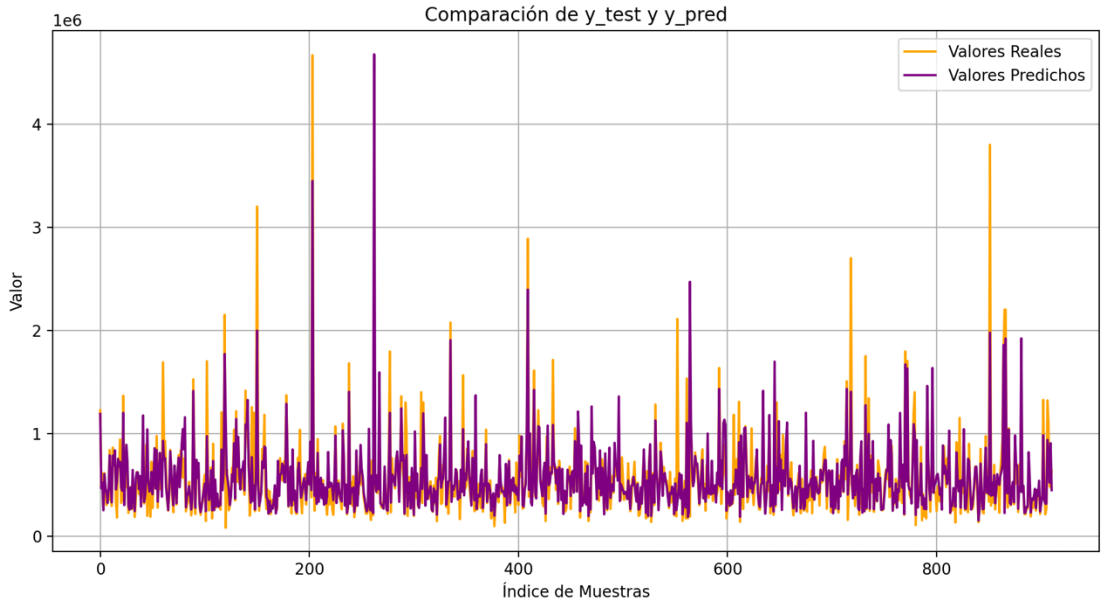
R^2 Coeficiente de Determinación

El coeficiente de determinación es la proporcion de variación en la variable dependiente que es predecible desde las variables independientes. Mientras mas se acerca a cero, implica un mejor modelo.

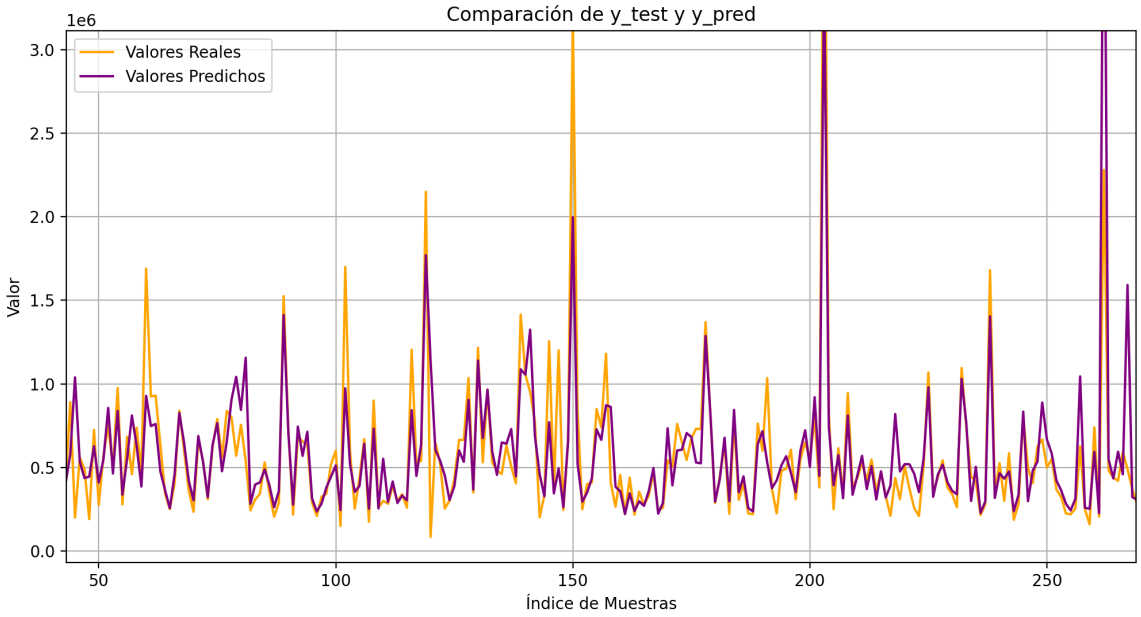
$$R^2 = 1 - \frac{\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}{\sum_{i=1}^{n}(Y_i - \overline{Y}_i)^2}$$

4.2 Grafica Comparativa

Para visualizar de mejor manera la efectividad del modelo en acercarse sus predicciones al valor real.



Al acercarnos en la grafica podemos apreciar las discrepancias entre los valores reales y los predecidos al ver la distancia entre cada pico.



El coeficiente de nuestro modelo (0.6014) nos indica que nuestro modelo va en buen camino, sin embargo las medidas MSE y MAE nos indican que los datos aun tienen varios detalles que previenen un funcionamiento optimo del modelo, esto siendo corroborado gracias a la grafica comparativa, por ende deberemos realizar nuevas tecnicas de transformación y normalización.

5 Normalización de datos y eliminacion de distintas columnas

Datos como el tamaño de los sotános tienen un detalle que puede causar confusión, cuando se carece de un sotáno se indica que el tamaño del mismo es cero, solo para que el siguiente tenga valores de inclusive 4000 pies. Es debido a esto que se optara para este y otros valores realizar el proceso de “binning”.

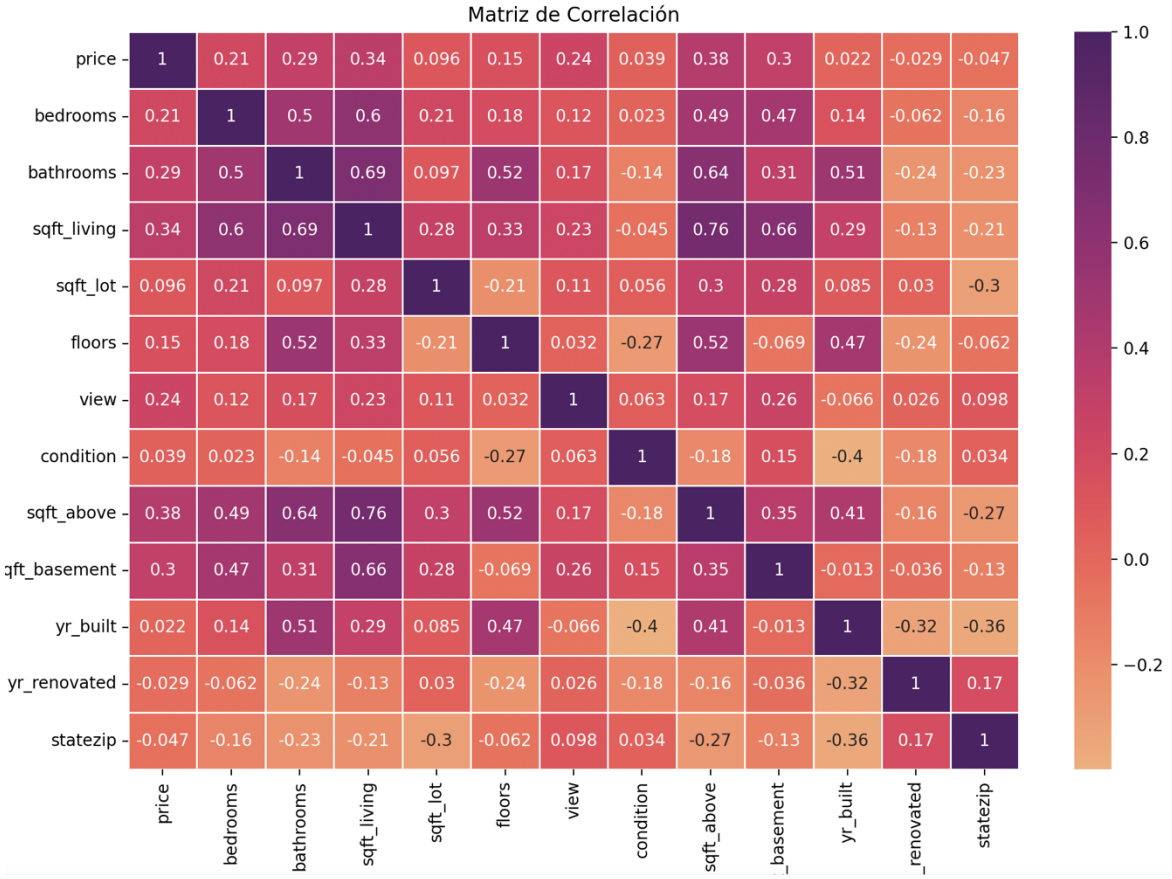
5.1 Binning

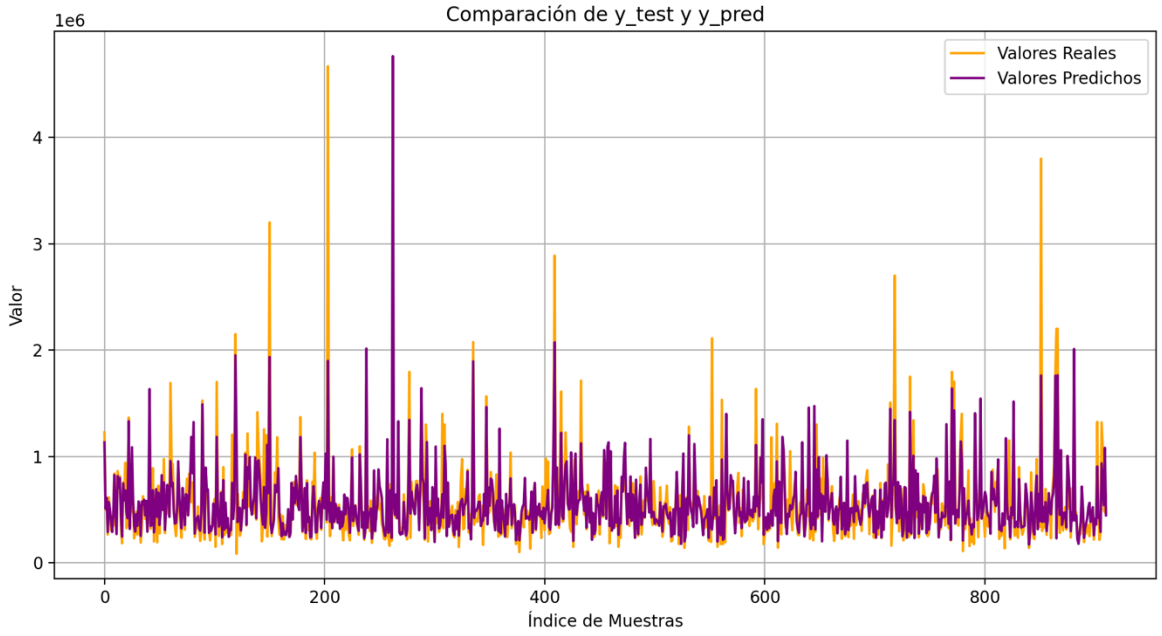
Binning es un proceso de normalización donde se convierten distintos datos de variables cantidades a una cantidad mucho mas facil de analizar, en el caso del soáno se puede tratar de cambiar el tamaño exacto del sotano por un numero que represente este, es decir, un 0 para su ausencia y el 4 para indicar un gran sotáno.

```
df = df.drop('city', axis = 1)
# Binning para el sotano
bins = [0, 600, 1000, 2000, df['sqft_basement'].max()]
df['sqft_basement_binning'] = pd.cut(df['sqft_basement'], bins = bins,
labels=False)
print(df['sqft_basement_binning'].value_counts())


# Binning para sala de estar
# bins= [500, 750, 2500, 3000, df['sqft_living'].max()]
df['sqft_living_binning'] = pd.qcut(df['sqft_living'], q=4, labels=False)
print(df['sqft_living_binning'].value_counts())

df['sqft_lot'] = pd.qcut(df['sqft_lot'], q=4, labels=False)
print(df['sqft_lot'].value_counts())

bins = [0, 1.5, 2, 3, df['bathrooms'].max()]
df['bathrooms_binning'] = pd.cut(df['bathrooms'], bins=bins, labels=False)
print(df['bathrooms_binning'].value_counts())

df = df.drop('waterfront', axis = 1)
```

Esto nos permite que sea mucho mas facil para el modelo el poder analizar todos los valores que involucran todas las areas en pies de forma mas sencilla, como mostraremos a continuación

5.2 Resultados despues de los cambios





R2: 0.5517665644159838

6. Conclusión

Por razones que sigo sin entender del todo el hecho de cambiar los datos existentes por datos de mayor facilidad a la hora de procesar no llevo a mejores resultados, los tres dias pasados me han hecho darme cuenta de que muy seguramente hay en varios metodos que aun no entiendo y que el machine learning es una rama que, con todo y dificultades, disfruto.
Sin embargo. Lamentablemente por tiempo no tendre un resultado satisfactorio que cruce la barrera del r^2 a 72%. En estos momentos son las 10:35 pm del 20 de Septiembre, y con miedo a que el internet se vuelva a ir he tomado la deicsión de cortar mi avance y enviar el reporte como esta.

Agradezco esta actividad y buscare tratar de mejorarle lo mas posible este fin de semana por orgullo.

Agradezco la oportunidad y lamento unicamente no haber hecho mejor, me quedo con 3 versiones incompletas de un codigo que espero poder convertir en un solo modelo que funcione perfectamente.

Atentamente.
Diego Pacheco Valdez