En este avance de proyecto, se analiza el dataset a profundidad, se hace una limpieza general de los datos y se estudian variables que tienen una influencia en el precio final de la vivienda. Se mostrará el proceso de los resultados y gráficos representativos de los mismos con su respectivo análisis.

1. Librerías

A continuación, se muestran las librerías que se usaron para el procedimiento mencionado anteriormente:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
from scipy.stats import norm
from sklearn.model_selection import train_test_split
```

Figura 1. Librerías

2. Carga de archivos a utilizar

Se procede a realizar la carga de los datos de entrenamiento y prueba respectivamente para el modelo de predicción, en este caso se muestra en la *Figura 2* las primeras 5 filas de los datos de entrenamiento:

```
entrenamiento.head()
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	PoolArea	PoolQC	Fence	MiscFeature	MiscVal	MoSold
0	1	60	RL	65.0	8450	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	0	2
1	2	20	RL	80.0	9600	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	0	5
2	3	60	RL	68.0	11250	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0	9
3	4	70	RL	60.0	9550	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0	2
4	5	60	RL	84.0	14260	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0	12

5 rows × 81 columns

Figura 2. Datos entrenamiento.

Se informan de las dimensiones de nuestros DataFrames:

```
Dimensión train.csv: (1460, 81)
Dimensión test.csv: (1459, 80)
```

Figura 3. Dimensiones.

3. Análisis de datos

3.1. Reduciendo variables:

Ya que la variable que se intentará predecir es "SalePrice", Se informa sobre ella usando el método describe (), el cual, me entrega datos relevantes, como lo son: Cantidad de datos, promedio, desviación estándar, etc.

```
entrenamiento["SalePrice"].describe()

count      1460.000000
mean     180921.195890
std       79442.502883
min       34900.000000
25%      129975.000000
50%      163000.000000
75%      214000.000000
max      755000.000000
Name: SalePrice, dtype: float64
```

Figura 4. Descripción SalePrice.

Ya que la variable que interesa analizar es el precio de venta "SalePrice", se verifica que no tenga valores vacíos, en caso tal de que los haya, se remueven.

Para ver la forma de los datos del precio de venta de las casas, se procede a graficar un histograma:

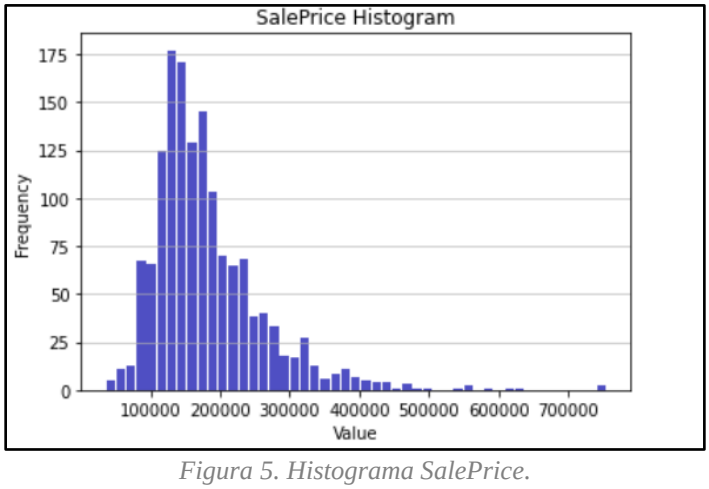

Figura 5. Histograma SalePrice.

Ya que hay muchas variables en este ejercicio, se procede a ver las variables que tengan una mayor correlación con SalePrice, para así, tener a estas en cuenta a la hora de proceder al entrenamiento para predecir el precio. Para ver la correlación de las variables, crearemos una matriz de correlación.

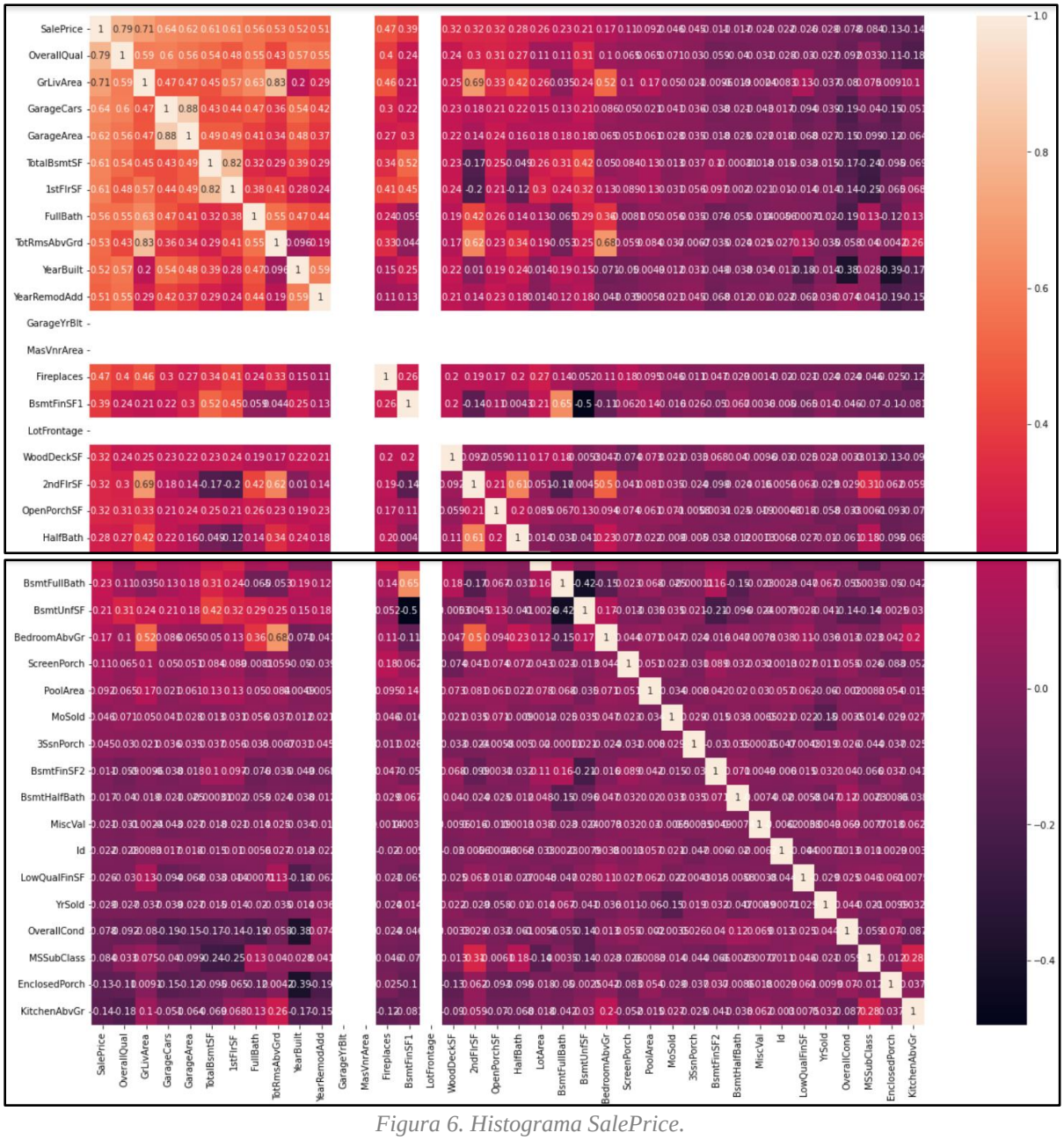

Figura 6. Histograma SalePrice.

Se puede observar en la *Figura 6.* los valores de los coeficientes correlación de Pearson correspondientes a la columna "SalePrice". Es una matriz simétrica respecto a la diagonal unitaria.

A continuación, se crea una nueva variable que almacena el nuevo nivel de correlación positivo mínimo para ser considerado en el modelo. En este caso se deciden variables con un valor de correlación de al menos el 0.3

Se grafica la variable SalePrice junto con las variables que más se correlacionan a ella, tal como se podrá observar, todas estas presentan una correlación positiva, es decir, si una de las variables aumenta, la otra también lo hace:

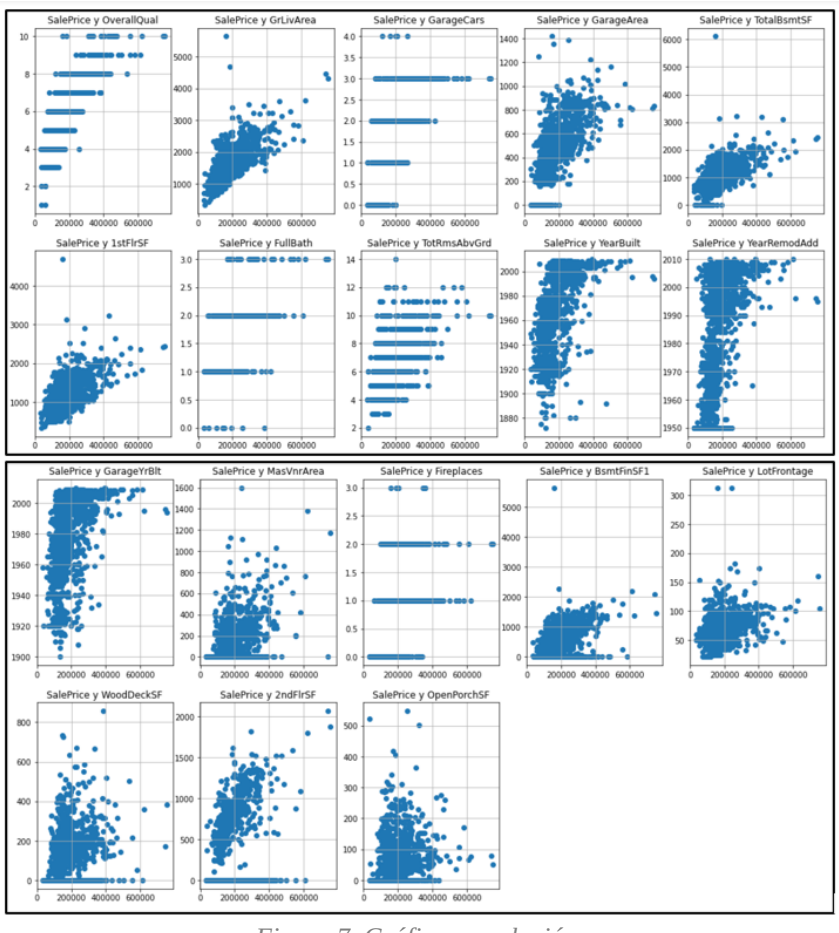

Figura 7. Gráfica correlación.

Se puede observar la variable “SalePrice” en el eje de las x y la otra variable relacionada en el eje y. De izquierda a derecha, disminuye el nivel de correlación de Pearson, siendo la variable independiente “OverallQual” la que mas incide en el precio final de la vivienda.

3.2. Normalización

Tal como se verá a continuación en la *Figura 8a y 8b.* los datos de la columna SalePrice del DataFrame, no se pueden representar mediante una distribución de probabilidad normal:

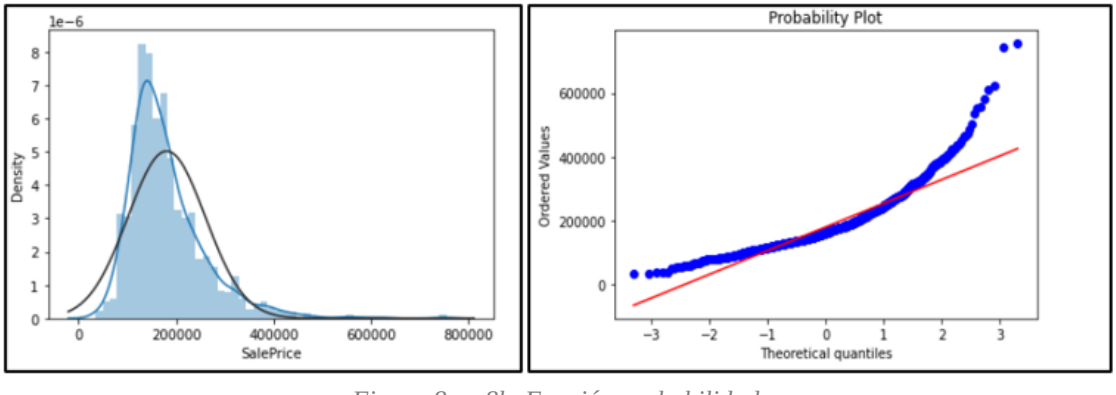

Figura 8a y 8b. Función probabilidad.

En la *Figura 9.* los datos de la columna SalePrice del DataFrame (gráfico azul), no sigue la distribución de probabilidad normal (gráfico rojo), por tanto, para tener un mejor entrenamiento de la variable, se debe encontrar una distribución que represente bien el comportamiento de los datos. En la siguiente página se encuentra que para transformar distribuciones con sesgo positivo (como es nuestro caso), la transformación logarítmica es la más usada, puesto que en la escala logarítmica, la distancia es exactamente la misma entre 1 y 10 que entre 10 y 100 o 100 y 1000, etc. Lo que resulta en que la parte izquierda se expandirá, mientras que la parte derecha se comprimirá, lo que favorecerá a la curva resultante para que se ajuste mejor a una normal. En la *Figura 9a y 9b.* se observa la normalización llevada a cabo.

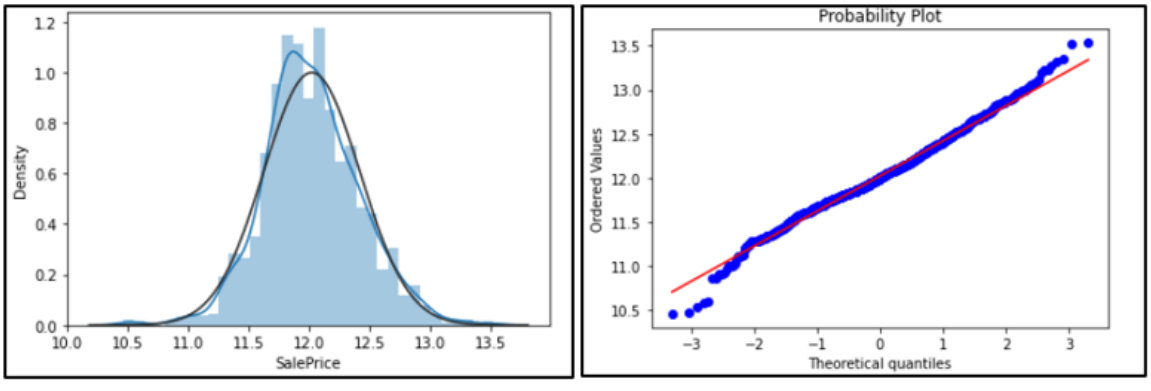

Figura 9a y 9b. Función probabilidad normalizada.

3.3. Variables Dummies

En la siguiente página se habla acerca de lo que son las variables Dummies o indicadoras, las cuales sirven para identificar las categorías a las cuales pertenecen las observaciones, estas variables pueden tomar valores de 0 o de 1. Por tanto, transformaremos el DataFrame que se tiene en el momento (sin datos faltantes) a variables Dummies.

```
df_entrenamiento_3 = pd.get_dummies(df_entrenamiento_3)
df_prueba_3 = pd.get_dummies(df_prueba_3)
```

Figura 10. Dummies.

Como ya no se tienen datos faltantes en estas últimas variables, están listas para para el entrenamiento y la prueba.

```
y_train = df_entrenamiento_3.SalePrice.values
df_entrenamiento_3.drop("SalePrice", axis = 1, inplace = True)
test_pct = 0.2
# Con la siguiente línea, de los datos df_entrenamiento_3 y de y_train, serán divididos aleatoriamente de
# tal forma que el 20% correspondrá a test y el 80% a train
X_train, X_test, y_train, y_test = train_test_split(df_entrenamiento_3, y_train, test_size=test_pct)

print(X_train.shape, X_test.shape)
print(y_train.shape, y_test.shape)

(1168, 17) (292, 17)
(1168,) (292,)
```

Figura 11. Entrenamiento y prueba.