

INSTRUCTIVO: Modelos y Métricas

BASE TEORICAS: Revisemos un contexto de qué criterios medir y su valor.

Métricas

¿Que métricas podemos utilizar?



Exactitud
(Accuracy)



Sensibilidad
(Recall)



Precisión



Puntuación F1
(F1 Score)

Matriz de confusión

		Predicción		
		Positivo	Negativo	
Actual	Positivo	48	2	Error tipo II
	Negativo	5	45	
		Error tipo I		

Exactitud (Accuracy)



- Predicciones correctas.
- Dividido (/)
- Número total de predicciones

CONCLUSION: SE USA CUANDO EL NUMERO PARECIDO DE LAS MUESTRAS SELECCIONADAS

Sensibilidad (*Recall*)



Permite encontrar todos los casos relevantes:

- Numero de verdadero positivos
- Dividido (/)
- Numero de verdadero positivos + falso negativos

Sensibilidad (*Recall*)

Permite encontrar todos los casos relevantes:

- Numero de verdadero positivos
- Dividido (/)
- Numero de verdadero positivos + falso negativos

$$48 / 48 + 2 = 0.96$$

Precisión



Permite encontrar solo los casos relevantes:

- Numero de verdadero positivos
- Dividido (/)
- Numero de verdadero positivos + falso positivos

Precisión

Permite encontrar solo los casos relevantes:

- Numero de verdadero positivos
- Dividido (/)
- Numero de verdadero positivos + falso positivos

$$48 / 48 + 5 = 0.905$$

Puntuación F1 (*F1 Score*)



Es una combinación entre **Sensibilidad (*Recall*)** y **Precisión**

Utilizando la media armonica:

$$F1 = 2 * (\text{sensibilidad} * \text{precision}) / (\text{sensibilidad} + \text{precision})$$
$$2 * (0.96 * 0.905) / (0.96 + 0.905) = 0.93$$

Métricas

¿Que métricas podemos utilizar?



Error absoluto
medio



Error cuadrado
medio



Error cuadrático
medio

Error absoluto medio (MAE)

Es básicamente la media del error absoluto de cada predicción. Muy fácil de entender.

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

Error cuadrado medio (MSE)

Es básicamente la media del error absoluto de cada predicción pero cuadrado.

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

Error cuadrático medio (RMSE)

Es básicamente la media del error absoluto de cada predicción pero cuadrado y después cogiendo la raíz cuadrada.

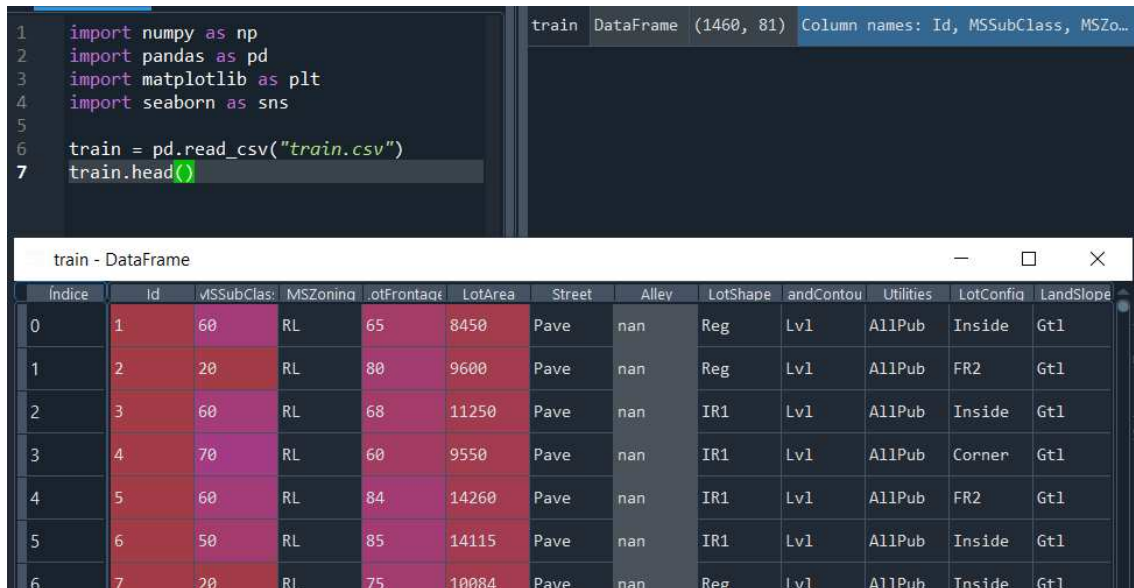
$$\sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2}$$

PASOS:

1. Descargar el recurso publicado en aula virtual. El recurso trata acerca de precios de casas.
2. Aplicar el siguiente comando:

```
pip install seaborn
```

3. Importamos y revisamos el contenido:



```

1 import numpy as np
2 import pandas as pd
3 import matplotlib as plt
4 import seaborn as sns
5
6 train = pd.read_csv("train.csv")
7 train.head()

```

train DataFrame (1460, 81) Column names: Id, MSSubClass, MSZo...

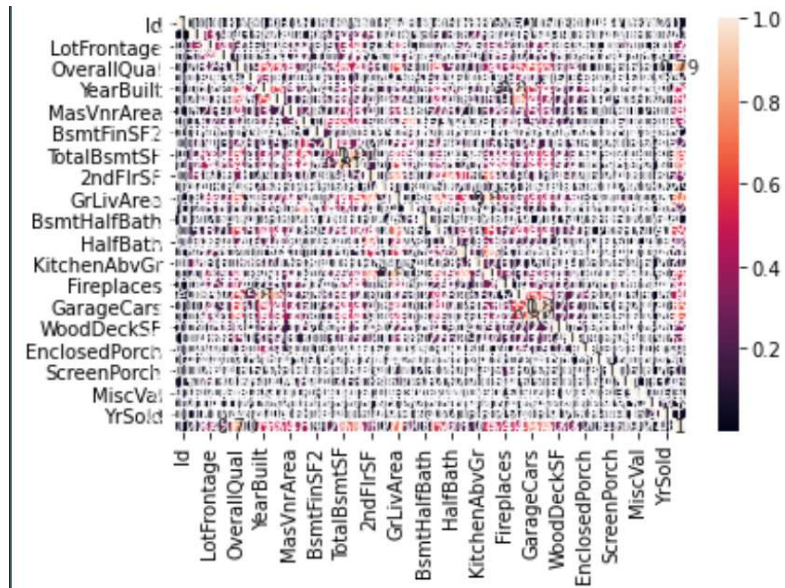
Indice	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	LotConfig	LandSlope
0	1	60	RL	65	8450	Pave	nan	Reg	Lv1	AllPub	Inside	Gtl
1	2	20	RL	80	9600	Pave	nan	Reg	Lv1	AllPub	FR2	Gtl
2	3	60	RL	68	11250	Pave	nan	IR1	Lv1	AllPub	Inside	Gtl
3	4	70	RL	60	9550	Pave	nan	IR1	Lv1	AllPub	Corner	Gtl
4	5	60	RL	84	14260	Pave	nan	IR1	Lv1	AllPub	FR2	Gtl
5	6	50	RL	85	14115	Pave	nan	IR1	Lv1	AllPub	Inside	Gtl
6	7	20	RL	75	10084	Pave	nan	Reg	Lv1	AllPub	Inside	Gtl

4. Realizamos correlaciones entre el precio del inmueble y sus variables. Para el contamos con la creación de un mapa de "calor".

```

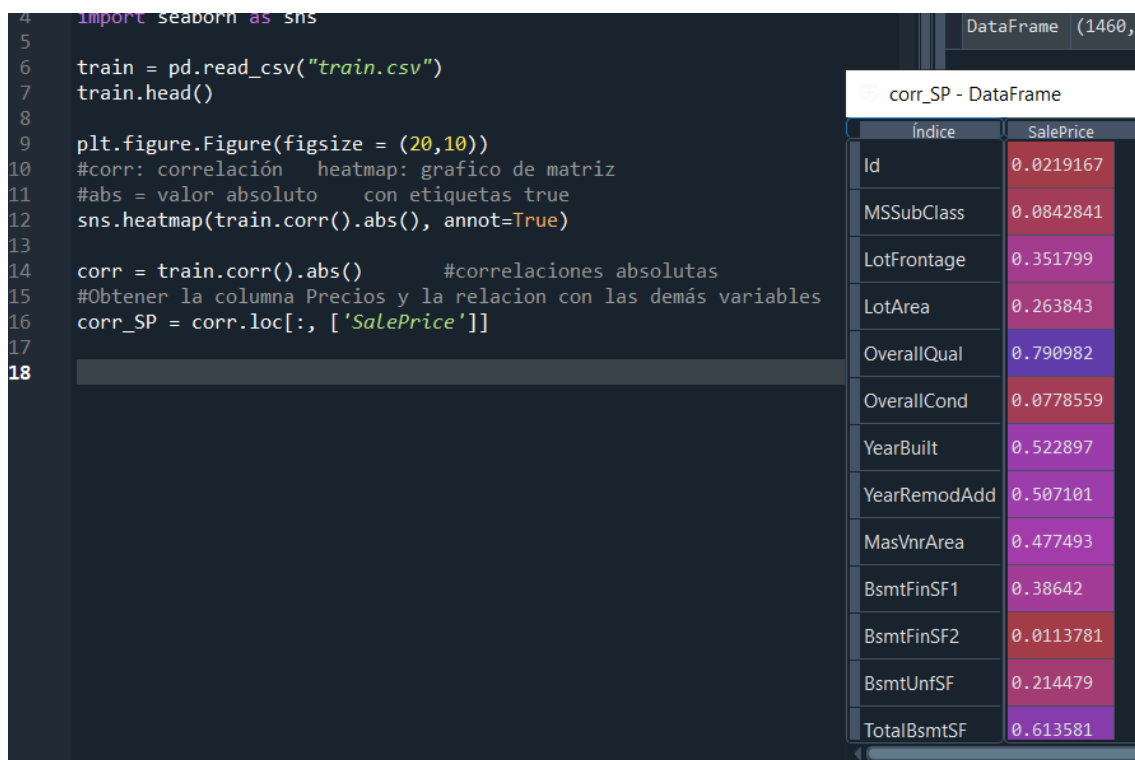
6 train = pd.read_csv("train.csv")
7 train.head()
8
9 plt.figure(figsize = (20,10))
10 #corr: correlación heatmap: grafico de matriz
11 #abs = valor absoluto con etiquetas true
12 sns.heatmap(train.corr().abs(), annot=True)
13

```

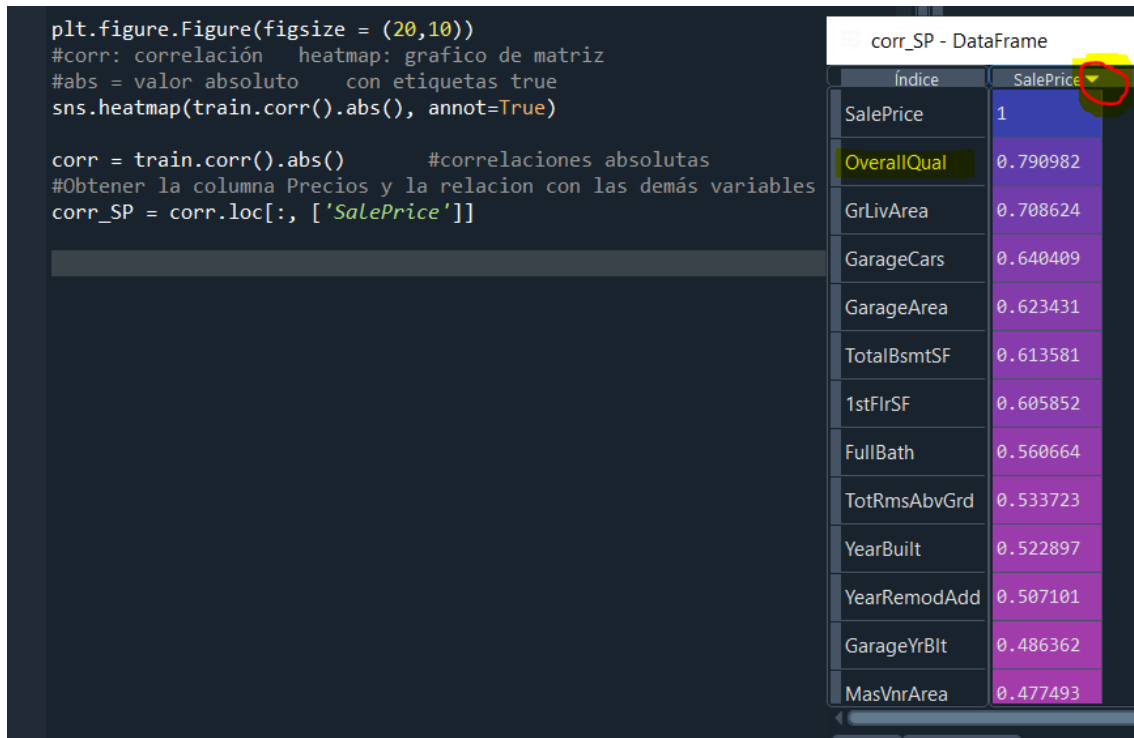



Conclusión no existe variables que relacionen lo suficiente con el precio.

5. Entonces empezamos a enfocarnos en precio y el valor que tiene con los demás.

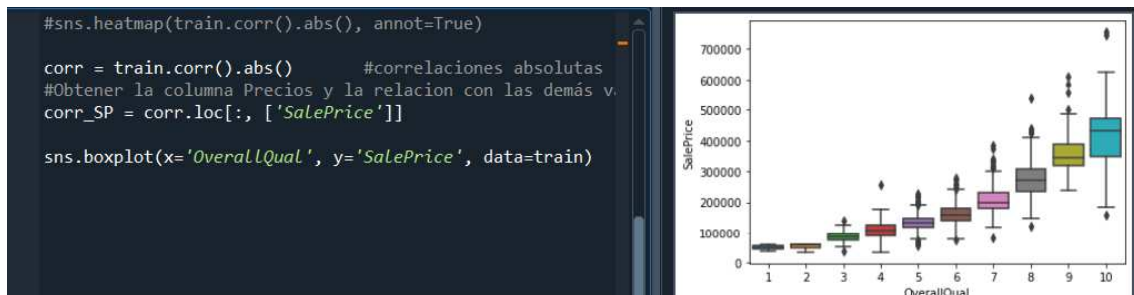


Ordenamos de forma descendente.

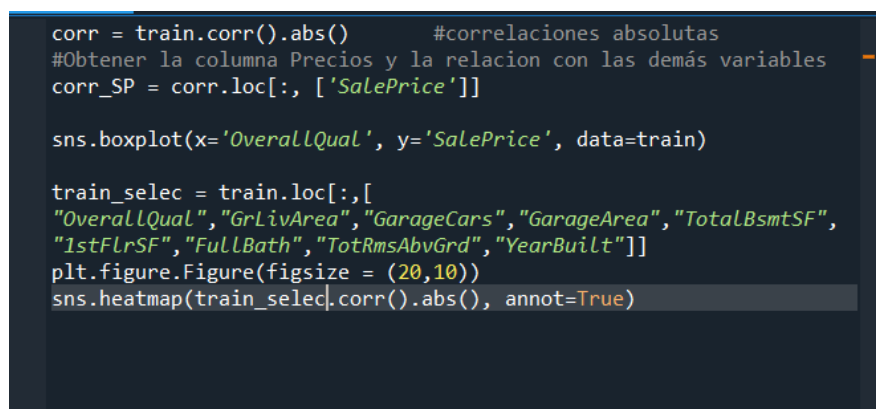


Como observamos el mayor valor se encuentra con "OverallQual".

6. Ahora vamos a mostrar una gráfica alusiva entre OverallQual y SalePrice.



7. Ahora escogemos las correlaciones de 0.5 hacia arriba de acuerdo a la lista y hacemos un mapa.



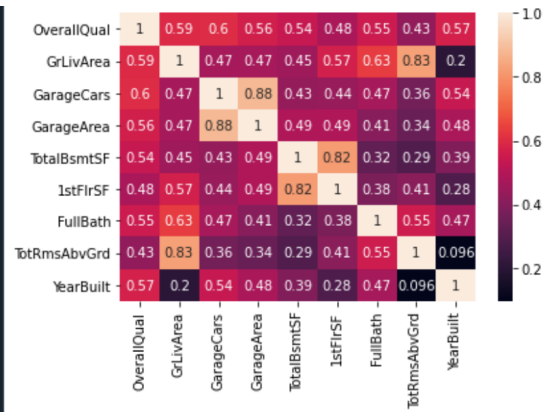
```

corr = train.corr().abs() #correlaciones absolutas
#Obtener la columna Precios y la relacion con las demás variables
corr_SP = corr.loc[:, ['SalePrice']]

sns.boxplot(x='OverallQual', y='SalePrice', data=train)

train_selec = train.loc[:, [
    "OverallQual", "GrLivArea", "GarageCars", "GarageArea", "TotalBsmtSF",
    "1stFlrSF", "FullBath", "TotRmsAbvGrd", "YearBuilt"]]
plt.figure(figsize = (20,10))
sns.heatmap(train_selec.corr().abs(), annot=True)

```



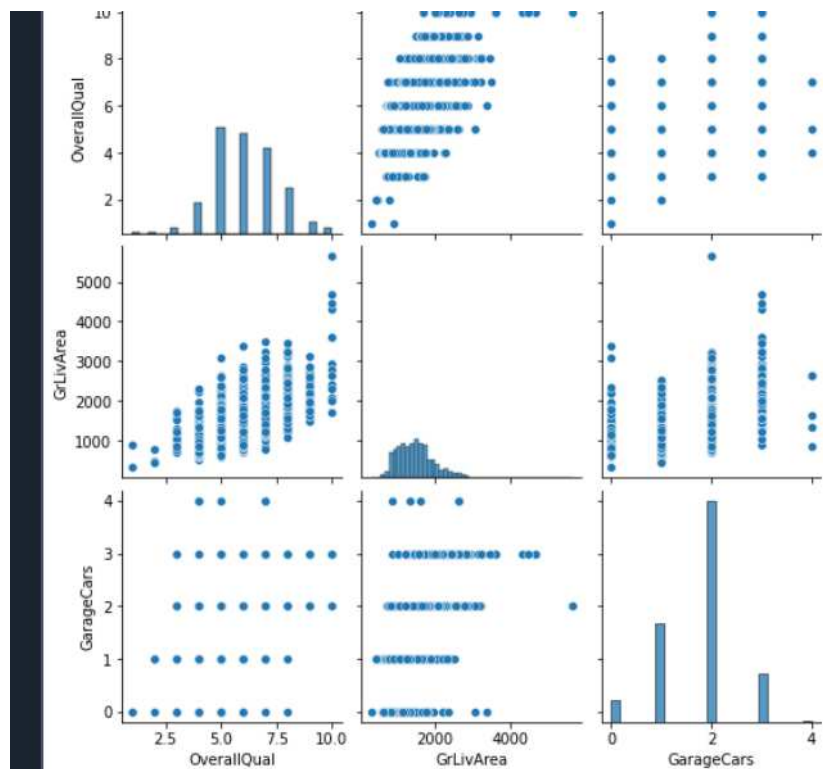
8. Ahora creamos una gráficas comparativas con los valores más altos de correlación.

```

"OverallQual", "GrLivArea", "GarageCars", "GarageArea", "TotalBsmtSF",
"1stFlrSF", "FullBath", "TotRmsAbvGrd", "YearBuilt"]]]
plt.figure(figsize = (20,10))
sns.heatmap(train_selec.corr().abs(), annot=True)

train_selec2 = train.loc[:, ["OverallQual", "GrLivArea", "GarageCars"]]
sns.pairplot(train_selec2)

```



Para empezar a dividir datos y generar nuestro modelo.

9. Importamos las librerías, y los componentes X, Y


```
from sklearn.model_selection import train_test_split

X = train.loc[:,["OverallQual","GrLivArea","GarageCars"]]
Y = train.loc[:,["SalePrice"]]

X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.30, random_state=33)
```

Se escogido solamente las 3 primeras columnas. Y selección de precios.

La cantidad de elementos implicados para el tamaño del test es 30% de los elementos y aleatorizado. Lo restante pasa al entrenamiento.

10. Creamos el modelo lineal, usamos los elementos seleccionados para el entrenamiento.

```
from sklearn.linear_model import LinearRegression

lm = LinearRegression()
lm.fit(X_train, y_train)
```

Ejecutamos para empeece el ajuste al modelo lineal.

```
In [57]: from sklearn.linear_model import LinearRegression
...:
...: lm = LinearRegression()
...: lm.fit(X_train, y_train)
Out[57]: LinearRegression()
```

11. Ahora debemos observar si obtenemos interpolado y luego mostrar coeficientes.

```
print(lm.intercept_)

lm.coef_
print(str(lm.coef_))
```

Resultado:

```
...: lm.coef_
...: print(str(lm.coef_))
[-98211.22722987]
[[26073.55023041    55.45399707 20534.91339951]]
```

Con ello tenemos los coeficientes de la regresión con los valores de entrenamiento.

12. Ahora aplicamos con los valores de test empezamos a realizar predicciones y tendremos un conjunto de datos.

```
predicciones = lm.predict(X_test)
print(predicciones)
```

Resultado: Y_test

```
...: print(predicciones)
[[288324.40067172]
 [192961.70200842]
 [207556.27484465]
 [235338.72737838]
 [243944.26853069]
 [356774.97627209]
 [191187.17410207]
 [111639.03621061]
 [107313.62443889]
 [268139.14573704]
```

13. Vamos a convertir el resultado como tabla en el estilo de dataframe.

```
DataFramePredicciones = pd.DataFrame(predicciones)
DataFramePredicciones.reset_index(drop = True, inplace = True)
y_test.reset_index(drop = True, inplace = True)
df_unido = y_test.join(DataFramePredicciones)
print(df_unido)
```

Resultado

```

SalePrice      0
0      275500  288324.400672
1      127500  192961.702008
2      143000  207556.274845
3      225000  235338.727378
4      232000  243944.268531
..      ...      ...
433     150500  130898.507677
434     107000  100260.795205
435     186500  198673.463707
436     208900  200347.255225
437     130500  119236.233810

[438 rows x 2 columns]
```

El reset_index aplica el reseteo de índice "0"

14. Ahora aplicamos métricas para observar si lo obtenido presenta muchos errores.

```
#METRICAS
from sklearn import metrics
print('MAE', metrics.mean_absolute_error(y_test, predicciones))
print('MSE', metrics.mean_squared_error(y_test, predicciones))
print('RMSE', np.sqrt(metrics.mean_squared_error(y_test, predicciones)))

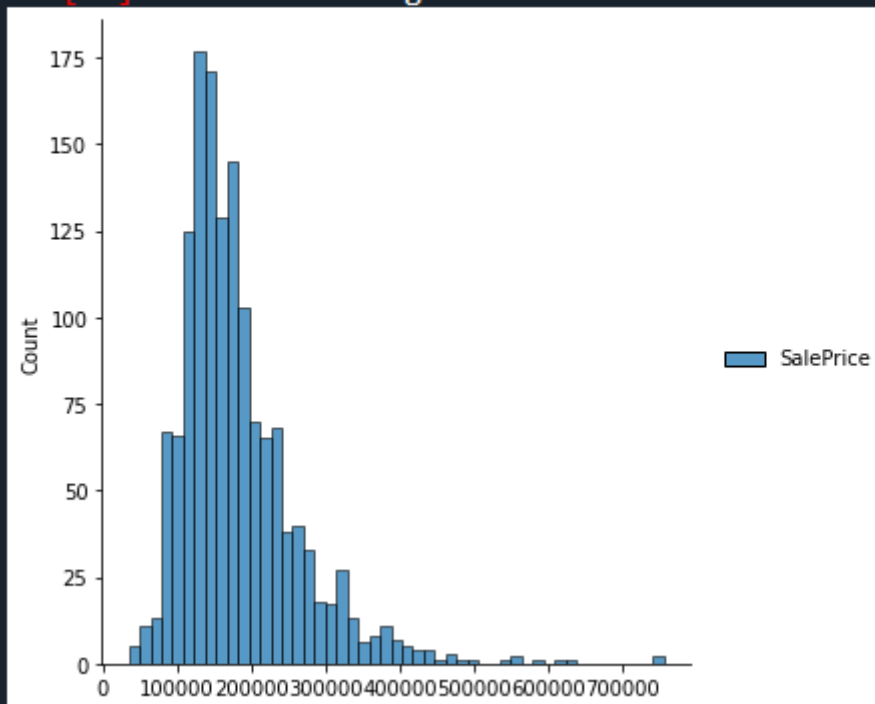
sns.displot(train.loc[:, ['SalePrice']])
```

Leyenda:

- Error Absoluto Medio (MAE)
- Error Cuadrado Medio (MSE)
- Error Cuadrático Medio (RMSE)

Resultado:

```
MAE 28265.464766621095
MSE 1815129145.9407864
RMSE 168.12336175148621
Out[63]: <seaborn.axisgrid.FacetGrid at 0x215e4911848>
```



15. Para tratamiento homogéneo de casas con alto precio como las de bajo precio y otorgar las mismas probabilidades utilizamos el recurso de la media cuadrática de errores.

```
from sklearn.metrics import mean_squared_log_error
print('Log RMSE', np.sqrt(metrics.mean_squared_log_error(y_test, predicciones)))
```

Resultado:

Log RMSE 0.22639145935348678

16. Procedemos ahora el proceso de test.

```
test = pd.read_csv('test.csv')
X = test.loc[:,["OverallQual","GrLivArea","GarageCars"]]

predicciones = lm.predict(X)
# => Lanza error

#Revisamos si cuenta con valores nulos
X.isna().sum()
#Corregimos colocando el valor de 0 en vez de null
X['GarageCars'].fillna(0,inplace = True)
```

Resultado obtenido del error: La ejecución es por grupo de líneas de código presentados.

```
OverallQual    0
GrLivArea      0
GarageCars     1
dtype: int64
```

```
predicciones = lm.predict(X)
DataFramePredicciones = pd.DataFrame(predicciones)
DataFramePredicciones.reset_index(drop = True, inplace = True)
X_test.reset_index(drop = True, inplace = True)
df_entrega = X.join(DataFramePredicciones)
print(df_entrega)
#Se obtiene las columnas y la columna resultante "0"
```

Obtenemos el resultado matricial:

```
OverallQual  GrLivArea  GarageCars    0
0           5         896          1.0  102378.218699
1           6        1329          1.0  152463.349663
2           5        1629          2.0  163560.911954
3           6        1604          2.0  188248.112257
4           8        1280          2.0  222428.117666
...         ...         ...         ...         ...
1454         4        1092          0.0   66638.738496
1455         4        1092          1.0   87173.651895
1456         5        1224          2.0  141102.043139
1457         5         970          0.0   85946.901083
1458         7        2000          3.0  256816.358728

[1459 rows x 4 columns]
```

17. Convertimos las predicciones para luego exportarlo y finalmente comparar gráficas.

```
predicciones = lm.predict(X)
DataFramePredicciones = pd.DataFrame(predicciones)
DataFramePredicciones.reset_index(drop = True, inplace = True)
id = test.loc[:,['Id']]
id.reset_index(drop = True, inplace = True)
df_entrega = id.join(DataFramePredicciones)
print(df_entrega)

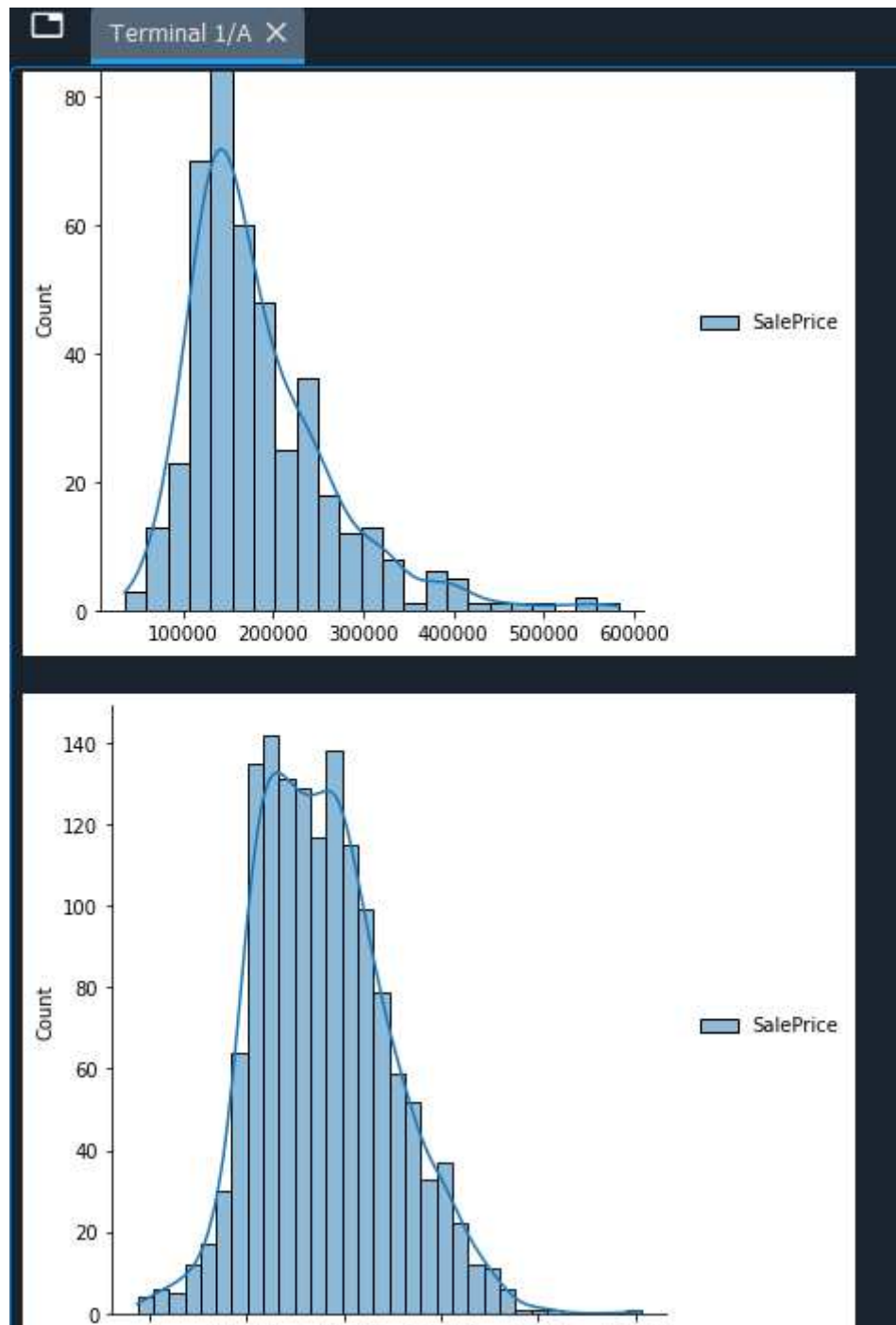
df_entrega.columns = ['Id', 'SalePrice']
df_entrega.to_csv('entrega.csv', index = False)

sns.displot(df_unido.loc[:,['SalePrice']], color="skyblue", label="X", kde=True)
sns.displot(df_entrega.loc[:,['SalePrice']], color="red", label="X", kde=True)
```

Resultado:

```
   Id  SalePrice
0  1461  102378.218699
1  1462  152463.349663
2  1463  163560.911954
3  1464  188248.112257
4  1465  222428.117666
...
1454  2915   66638.738496
1455  2916   87173.651895
1456  2917  141102.043139
1457  2918   85946.901083
1458  2919  256816.358728

[1459 rows x 2 columns]
```

Conclusión:

La comparativa entre ambas gráficas promueve una cercanía, con medias de errores tratados.