Inteligencia de Negocios

“Preprocesamiento y Transformación de Datos”

Elaborado por: Diego García

Docente: Arlen Jeanette Lopez

Fecha de entrega: 17/09/2024

1. Cargar Datos

```python
df = pd.read_csv('dane/train.csv')
print(df.head())
```

```
PS C:\Users\ellie\Downloads\PyTD\PyTD> & C:/Users/ellie/AppData/Local/Programs/Python/Python312/python.exe c:/Users/ellie/Downloads/PyTD/PyTD/transform.py
   Id  MSSubClass MSZoning  LotFrontage  LotArea Street Alley LotShape LandContour Utilities  ... PoolArea PoolQC Fence MiscFeature MiscVal MoSold YrSold  SaleType  SaleCondition  SalePrice
0   1          60       RL         65.0     8450   Pave   NaN      Reg         Lvl    AllPub  ...        0    NaN   NaN         NaN       0      2   2008        WD         Normal     208500
1   2          20       RL         80.0     9600   Pave   NaN      Reg         Lvl    AllPub  ...        0    NaN   NaN         NaN       0      5   2007        WD         Normal     181500
2   3          60       RL         68.0    11250   Pave   NaN      IR1         Lvl    AllPub  ...        0    NaN   NaN         NaN       0      9   2008        WD         Normal     223500
3   4          70       RL         60.0     9550   Pave   NaN      IR1         Lvl    AllPub  ...        0    NaN   NaN         NaN       0      2   2006        WD        Abnorml     140000
4   5          60       RL         84.0    14260   Pave   NaN      IR1         Lvl    AllPub  ...        0    NaN   NaN         NaN       0     12   2008        WD         Normal     250000
```

Comenzamos esta Recopilación y transformación de datos por La carga de estos desde el CSV, Utilizando el Data Frame de Pandas, cargamos nuestros datos raw (es decir, sin ningún cambio)

2. Exploración Inicial

```python
#Impresión de las columnas y sus tipos de Datos
print(df.info())
```

```
#   Column         Non-Null Count  Dtype
---  ------         --------------  -----
0   Id             1460 non-null   int64
1   MSSubClass     1460 non-null   int64
2   MSZoning       1460 non-null   object
3   LotFrontage    1201 non-null   float64
4   LotArea        1460 non-null   int64
5   Street         1460 non-null   object
6   Alley          91 non-null     object
7   LotShape       1460 non-null   object
8   LandContour    1460 non-null   object
9   Utilities      1460 non-null   object
10  LotConfig      1460 non-null   object
11  LandSlope      1460 non-null   object
12  Neighborhood   1460 non-null   object
13  Condition1     1460 non-null   object
14  Condition2     1460 non-null   object
```

Después pasariamos a investigar nuestro data frame y encontrarnos con los datos más importante de la base.

Los Datos mas importantes serian SalePrice, YrSold, OverallQual, OverallCond, LotArea, Neighborhood debido a que nuestro enfoque sería saber cuántas casas se han vendido, los años mas populares, el precio mas común, Condición de la casa, Area del lote y Vecindario que se encuentra.

3. Limpieza de Datos

```
#Cantidad de variables nulas y las columnas
percent = df.isnull().sum() * 100 / len(df)
missingtable = pd.DataFrame({'percent': percent})
print(missingtable, "\n")
print(df.columns[df.isnull().any()].tolist())
```

```
                percent
Id             0.000000
MSSubClass     0.000000
MSZoning       0.000000
LotFrontage   17.739726
LotArea        0.000000
...                 ...
MoSold         0.000000
YrSold         0.000000
SaleType       0.000000
SaleCondition  0.000000
SalePrice      0.000000
```

```
#Imputa de valores
#Utilizare la mediana debido a que es una forma de imputación mas consistente que la media y la moda cuando los datos
#estan sesgados o tiene valores extremos como en este Dataframe, al mismo tiempo cuando los datos faltantes son aleatorios.
df['LotFrontage'] = df['LotFrontage'].fillna(df['LotFrontage'].median())
df['MasVnrArea'] = df['MasVnrArea'].fillna(df['MasVnrArea'].median())
df['GarageYrBlt'] = df['GarageYrBlt'].fillna(df['GarageYrBlt'].median())
print(df.head(8))
```

```
   Id  MSSubClass MSZoning  LotFrontage  LotArea Street Alley LotShape LandContour Utilities  ... PoolArea PoolQC  Fence MiscFeature MiscVal  MoSold  YrSold  SaleType  SaleCondition  SalePrice
0   1          60       RL         65.0     8450   Pave   NaN      Reg         Lvl    AllPub  ...        0    NaN    NaN         NaN       0       2    2008        WD         Normal     208500
1   2          20       RL         80.0     9600   Pave   NaN      Reg         Lvl    AllPub  ...        0    NaN    NaN         NaN       0       5    2007        WD         Normal     181500
2   3          60       RL         68.0    11250   Pave   NaN      IR1         Lvl    AllPub  ...        0    NaN    NaN         NaN       0       9    2008        WD         Normal     223500
3   4          70       RL         60.0     9550   Pave   NaN      IR1         Lvl    AllPub  ...        0    NaN    NaN         NaN       0       2    2006        WD        Abnorml     140000
4   5          60       RL         84.0    14260   Pave   NaN      IR1         Lvl    AllPub  ...        0    NaN    NaN         NaN       0      12    2008        WD         Normal     250000
5   6          50       RL         85.0    14115   Pave   NaN      IR1         Lvl    AllPub  ...        0    NaN  MnPrv        Shed     700      10    2009        WD         Normal     143000
6   7          20       RL         75.0    10084   Pave   NaN      Reg         Lvl    AllPub  ...        0    NaN    NaN         NaN       0       8    2007        WD         Normal     307000
7   8          60       RL         69.0    10382   Pave   NaN      IR1         Lvl    AllPub  ...        0    NaN    NaN        Shed     350      11    2009        WD         Normal     200000
```

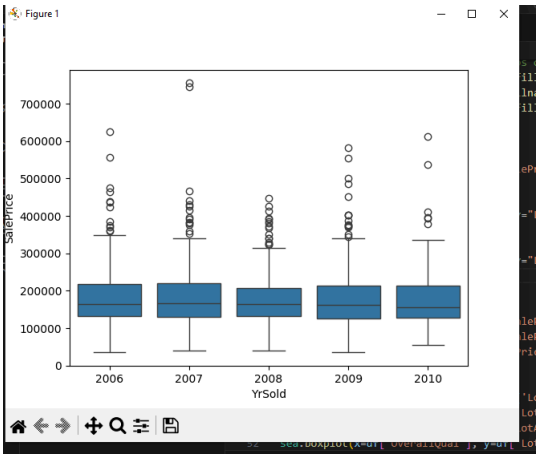
Ahora pasaremos a revisar cuáles datos son nulos en nuestro data frame, una vez identificados utilizaremos la mediana debido a que es una forma de imputación más consistente que la media y la moda cuando los datos están sesgados o tiene valores extremos como en este Data Frame, al mismo tiempo cuando los datos faltantes son aleatorios.

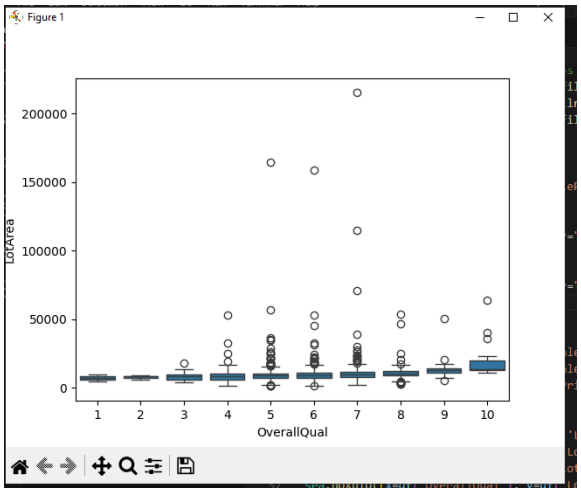
4. Manejo de Outliers

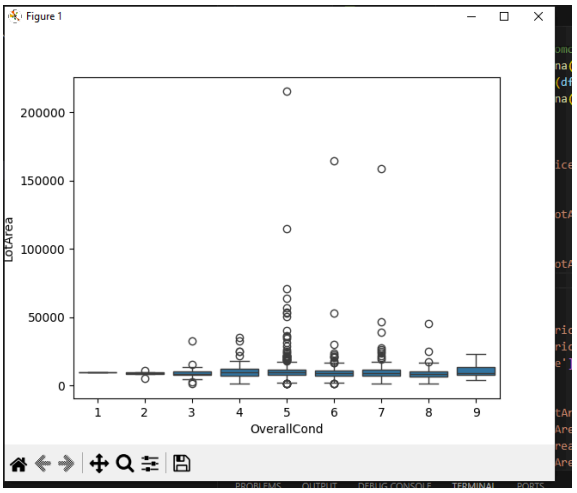
```python
# Detección de outliers
sea.boxplot(data=df, x="YrSold", y="SalePrice")
plt.show()

sea.boxplot(data=df, x="OverallQual", y="LotArea")
plt.show()

sea.boxplot(data=df, x="OverallCond", y="LotArea")
plt.show()
```





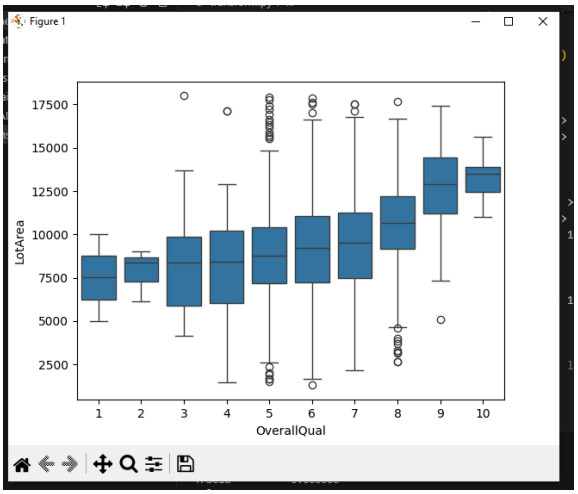


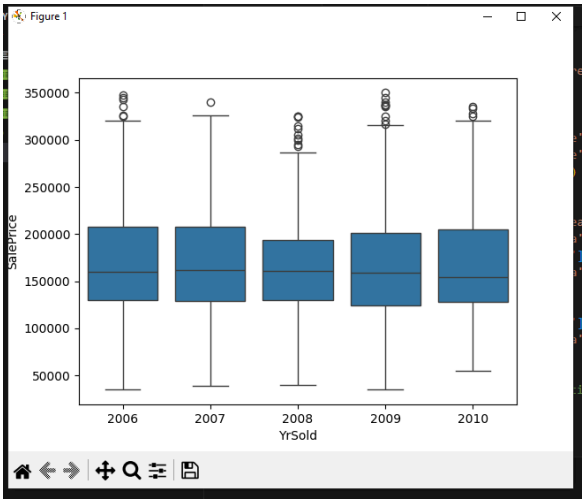
Mediante el uso de Boxplots nos podemos dar cuenta que si nuestros datos cuentan con outliers los cuales comprometen la precisión y uso de nuestros datos.

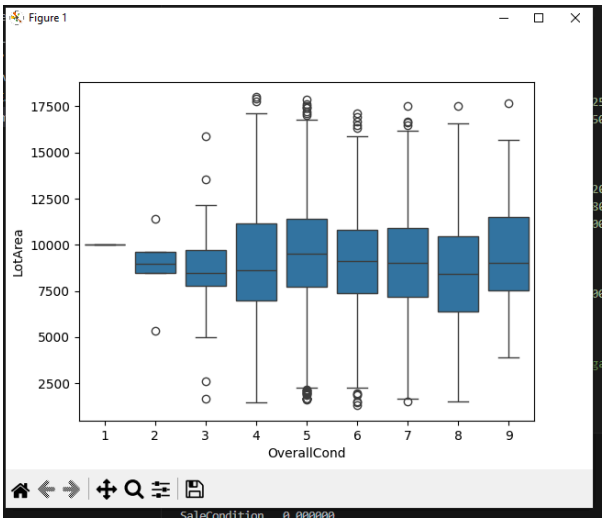
```python
#Tratamiento de outliers
df.loc[(df['YrSold'] == 2008) & (df['SalePrice'] > 325000), 'SalePrice'] = np.nan
df.loc[(df['YrSold'] != 2008) & (df['SalePrice'] > 350000), 'SalePrice'] = np.nan
sea.boxplot(x=df['YrSold'], y=df['SalePrice'])
plt.show()

df.loc[(df['OverallQual'] == 10) & (df['LotArea'] > 20000), 'LotArea'] = np.nan
df.loc[(df['OverallQual'] == 9) & (df['LotArea'] > 18000), 'LotArea'] = np.nan
df.loc[(df['OverallQual'] < 9) & (df['LotArea'] > 18000), 'LotArea'] = np.nan
sea.boxplot(x=df['OverallQual'], y=df['LotArea'])
plt.show()

df.loc[(df['OverallCond'] < 9) & (df['LotArea'] > 18000), 'LotArea'] = np.nan
sea.boxplot(x=df['OverallCond'], y=df['LotArea'])
plt.show()
```

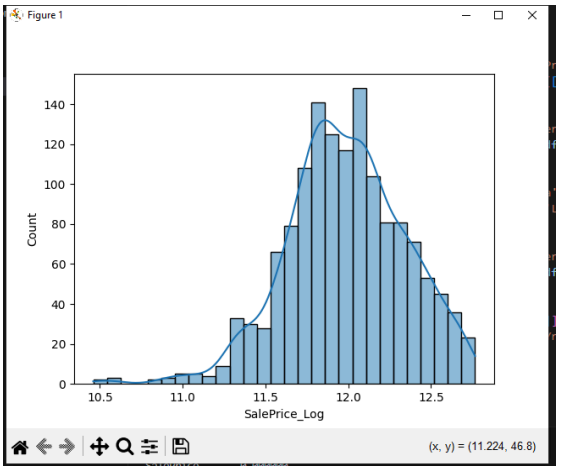






Para deshacernos de estos Outliers utilice método de Transformación del Boxplot el cual le puse un límite de los datos representados, cualquier dato mayor que el rango del boxplot no va a ser mostrado.

5. Transformación de Variables




```python
#Normalización, Estandarización y  Transformación logarítmica

scaler = StandardScaler()
MinMax_scaler = MinMaxScaler()

df['SalePrice_Scaled'] = scaler.fit_transform(df[['SalePrice']])
df['SalePrice_MinMax'] = MinMax_scaler.fit_transform(df[['SalePrice']])
df['SalePrice_Log'] = np.log1p(df['SalePrice'])

df['OverallQual_Scaled'] = scaler.fit_transform(df[['OverallQual']])
df['OverallQual_MinMax'] = MinMax_scaler.fit_transform(df[['OverallQual']])
df['OverallQual_Log'] = np.log1p(df['OverallQual'])

df['LotArea_Scaled'] = scaler.fit_transform(df[['LotArea']])
df['LotArea_MinMax'] = MinMax_scaler.fit_transform(df[['LotArea']])
df['LotArea_Log'] = np.log1p(df['LotArea'])

df['OverallCond_Scaled'] = scaler.fit_transform(df[['OverallCond']])
df['OverallCond_MinMax'] = MinMax_scaler.fit_transform(df[['OverallCond']])
df['OverallCond_Log'] = np.log1p(df['OverallCond'])

df['YrSold_Scaled'] = scaler.fit_transform(df[['YrSold']])
df['YrSold_MinMax'] = MinMax_scaler.fit_transform(df[['YrSold']])

sea.histplot(df['SalePrice_Log'], kde=True)
plt.show()
```

```python
print(df[['SalePrice', 'SalePrice_Scaled', 'SalePrice_MinMax']].head())
print("\n")
print(df[['LotArea', 'LotArea_Scaled', 'LotArea_MinMax']].head())
print("\n")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
[8 rows x 81 columns]
   SalePrice  SalePrice_Scaled  SalePrice_MinMax
0   208500.0          0.633812          0.550936
1   181500.0          0.181922          0.465249
2   223500.0          0.884862          0.598540
3   140000.0         -0.512650          0.333545
4   250000.0          1.328383          0.682640


   LotArea  LotArea_Scaled  LotArea_MinMax
0   8450.0       -0.258348        0.428144
1   9600.0        0.097072        0.497006
2  11250.0        0.607021        0.595808
3   9550.0        0.081619        0.494012
4  14260.0        1.537293        0.776048
```

Ahora aplicamos normalización y estandarización con usos de métodos Z-score y Min-Max Scaling a nuestros datos relevantes y los imprimimos. Se utiliza la normalización cuando la distribución de los datos no se conoce o cuando se desea preservar la forma original de distribución, en cambio, la estandarización cuando se asume una distribución gaussiana, o cuando se desea comparar variables que tienen diferentes unidades o escalas.

6. Ingeniería de Características

```
#Creación de nuevas variables y Label Encoding

df['LotAreaIndividualValue'] = df['SalePrice'] / df['LotArea']
YrSold_cut = pd.qcut(df['YrSold'], q=3, labels=['Old Sale', 'Recent Sale', 'Brand New Sale'])
df['YrSold_cat'] = YrSold_cut

category_mapping = {
    'Old Sale': 0,
    'Recent Sale': 1,
    'Brand New Sale': 2
}

print(df[['YrSold', 'YrSold_cat']].sample(8))
```

```
[8 rows x 81 columns]
     YrSold      YrSold_cat
66     2010  Brand New Sale
591    2009     Recent Sale
312    2006        Old Sale
865    2009     Recent Sale
427    2008     Recent Sale
214    2010  Brand New Sale
891    2009     Recent Sale
436    2006        Old Sale
```

Finalmente, Le añadimos dos nuevas tablas, una que guarde el valor del lote individual dividiendo el lote por la cantidad que se vendió la casa, y para la segunda columna se utilizó el método Label Encoding para meter categorías acerca de qué tan antiguo o reciente fue la venta de la casa, en este caso, Label Encoding es el más indicado, organizado y Eficiente debido a que solo utiliza una sola variable para cada categoría, en vez de One Hot Encoding que utiliza 0 y 1 y se crean varias tablas jerárquicas para tomar en cuenta la selección que se está haciendo.

7. Análisis Comparativo

A diferencia de cómo comenzó el proyecto hasta ahora en su final, se puede notar bastantes correcciones a los Datos y sus Gráficos para poder sacarle su máximo potencial y utilizar estos datos a nuestro favor. Descubrimos cómo limpiar, imputar y transformar nuestros datos.