Asignatura	Datos del alumno	Fecha
Aprendizaje Automático	Apellidos: Cortes Orozco	23/08/2021
	Nombre: Wilfrido	

Actividad: Árboles y *random forest* para regresión y clasificación

Análisis descriptivo de los datos.

El set de datos llamado “housing_train” cuenta con una cantidad bastante alta de datos, de los cuales unos cuantos son de variables numéricas y otros tantos de variables categóricas. Primero hay que cargar el set de datos y ponerlo en una variable, para eso se utilizará la librería de Pandas y la librería de numpy.

```
1 import pandas as pd
2 import numpy as np
```

El set de datos cuenta con 81 variables (incluyendo el identificador), de las cuales 35 son enteras, 3 son flotantes y 43 son objetos. Para poder obtener este dato, se utilizó el comando “info” de Pandas.

```
1 df_Entrenamiento = pd.read_csv("housing_train.csv")
```

```
1 df_Entrenamiento.info()
```

El resultado que arrojó es el siguiente:

```
0   Id             1460 non-null   int64     21  RoofStyle      1460 non-null   object
1   MSSubClass     1460 non-null   int64     22  RoofMatl       1460 non-null   object
2   MSZoning       1460 non-null   object    23  Exterior1st    1460 non-null   object
3   LotFrontage    1201 non-null   float64   24  Exterior2nd    1460 non-null   object
4   LotArea        1460 non-null   int64     25  MasVnrType     1452 non-null   object
5   Street         1460 non-null   object    26  MasVnrArea     1452 non-null   float64
6   Alley          91 non-null     object    27  ExterQual      1460 non-null   object
7   LotShape       1460 non-null   object    28  ExterCond      1460 non-null   object
8   LandContour    1460 non-null   object    29  Foundation     1460 non-null   object
9   Utilities      1460 non-null   object    30  BsmtQual       1423 non-null   object
10  LotConfig      1460 non-null   object    31  BsmtCond       1423 non-null   object
11  LandSlope      1460 non-null   object    32  BsmtExposure   1422 non-null   object
12  Neighborhood   1460 non-null   object    33  BsmtFinType1   1423 non-null   object
13  Condition1     1460 non-null   object    34  BsmtFinSF1     1460 non-null   int64
14  Condition2     1460 non-null   object    35  BsmtFinType2   1422 non-null   object
15  BldgType       1460 non-null   object    36  BsmtFinSF2     1460 non-null   int64
16  HouseStyle     1460 non-null   object    37  BsmtUnfSF      1460 non-null   int64
17  OverallQual    1460 non-null   int64     38  TotalBsmtSF    1460 non-null   int64
18  OverallCond    1460 non-null   int64     39  Heating        1460 non-null   object
19  YearBuilt      1460 non-null   int64     40  HeatingQC      1460 non-null   object
20  YearRemodAdd   1460 non-null   int64     41  CentralAir     1460 non-null   object
```

42	Electrical	1459 non-null	object	63	GarageQual	1379 non-null	object
43	1stFlrSF	1460 non-null	int64	64	GarageCond	1379 non-null	object
44	2ndFlrSF	1460 non-null	int64	65	PavedDrive	1460 non-null	object
45	LowQualFinSF	1460 non-null	int64	66	WoodDeckSF	1460 non-null	int64
46	GrLivArea	1460 non-null	int64	67	OpenPorchSF	1460 non-null	int64
47	BsmtFullBath	1460 non-null	int64	68	EnclosedPorch	1460 non-null	int64
48	BsmtHalfBath	1460 non-null	int64	69	3SsnPorch	1460 non-null	int64
49	FullBath	1460 non-null	int64	70	ScreenPorch	1460 non-null	int64
50	HalfBath	1460 non-null	int64	71	PoolArea	1460 non-null	int64
51	BedroomAbvGr	1460 non-null	int64	72	PoolQC	7 non-null	object
52	KitchenAbvGr	1460 non-null	int64	73	Fence	281 non-null	object
53	KitchenQual	1460 non-null	object	74	MiscFeature	54 non-null	object
54	TotRmsAbvGrd	1460 non-null	int64	75	MiscVal	1460 non-null	int64
55	Functional	1460 non-null	object	76	MoSold	1460 non-null	int64
56	Fireplaces	1460 non-null	int64	77	YrSold	1460 non-null	int64
57	FireplaceQu	770 non-null	object	78	SaleType	1460 non-null	object
58	GarageType	1379 non-null	object	79	SaleCondition	1460 non-null	object
59	GarageYrBlt	1379 non-null	float64	80	SalePrice	1460 non-null	int64
60	GarageFinish	1379 non-null	object				
61	GarageCars	1460 non-null	int64				
62	GarageArea	1460 non-null	int64				

Dentro de todos esos datos, el comando permite saber cuántos datos nulos tiene cada variable. Como se puede observar, por ejemplo, el Id cuenta con 1460 datos que no son nulos (la cantidad de datos totales que hay en esa columna), mientras que Alley cuenta con solo 91 datos que no son nulos.

De momento se tomarán únicamente los datos numéricos para poder obtener valores estadísticos que brinden utilidad. Después se tomarán los valores del tipo objeto, para poder obtener información sobre esas variables categóricas.

- VARIABLES NUMÉRICAS:

Para poder obtener datos estadísticos de las variables numéricas, se puede utilizar el comando “describe” de Pandas. Ello regresará los datos más “útiles” estadísticamente hablando, como es la media, mediana, los cuartiles, el mínimo y el máximo, además de un conteo. Lo acomoda de acuerdo al nombre de la columna.

Asignatura	Datos del alumno	Fecha
Aprendizaje Automático	Apellidos: Cortes Orozco Nombre: Wilfrido	23/08/2021

```
1 df_Entrenamiento['Id'].describe()
```

```
count    1460.000000
mean      730.500000
std       421.610009
min         1.000000
25%       365.750000
50%       730.500000
75%      1095.250000
max      1460.000000
Name: Id, dtype: float64
```

```
1 df_Entrenamiento['LotArea'].describe()
```

```
count      1460.000000
mean      10516.828082
std        9981.264932
min        1300.000000
25%        7553.500000
50%        9478.500000
75%       11601.500000
max      215245.000000
Name: LotArea, dtype: float64
```

```
1 df_Entrenamiento['MSSubClass'].describe()
```

```
count    1460.000000
mean       56.897260
std        42.300571
min        20.000000
25%        20.000000
50%        50.000000
75%        70.000000
max       190.000000
Name: MSSubClass, dtype: float64
```

```
1 df_Entrenamiento['OverallQual'].describe()
```

```
count    1460.000000
mean        6.099315
std         1.382997
min         1.000000
25%         5.000000
50%         6.000000
75%         7.000000
max        10.000000
Name: OverallQual, dtype: float64
```

```
1 df_Entrenamiento['LotFrontage'].describe()
```

```
count    1201.000000
mean       70.049958
std        24.284752
min        21.000000
25%        59.000000
50%        69.000000
75%        80.000000
max       313.000000
Name: LotFrontage, dtype: float64
```

```
1 df_Entrenamiento['OverallCond'].describe()
```

```
count    1460.000000
mean        5.575342
std         1.112799
min         1.000000
25%         5.000000
50%         5.000000
75%         6.000000
max         9.000000
Name: OverallCond, dtype: float64
```

```
1 df_Entrenamiento['YearBuilt'].describe()
```

```
count    1460.000000
mean     1971.267808
std        30.202904
min      1872.000000
25%      1954.000000
50%      1973.000000
75%      2000.000000
max      2010.000000
Name: YearBuilt, dtype: float64
```

```
1 df_Entrenamiento['YearRemodAdd'].describe()
```

```
count    1460.000000
mean     1984.865753
std        20.645407
min      1950.000000
25%      1967.000000
50%      1994.000000
75%      2004.000000
max      2010.000000
Name: YearRemodAdd, dtype: float64
```

```
1 df_Entrenamiento['BsmtUnfSF'].describe()
```

```
count    1460.000000
mean      567.240411
std       441.866955
min         0.000000
25%       223.000000
50%       477.500000
75%       808.000000
max      2336.000000
Name: BsmtUnfSF, dtype: float64
```

```
1 df_Entrenamiento['MasVnrArea'].describe()
```

```
count    1452.000000
mean      103.685262
std       181.066207
min         0.000000
25%         0.000000
50%         0.000000
75%       166.000000
max      1600.000000
Name: MasVnrArea, dtype: float64
```

```
1 df_Entrenamiento['TotalBsmtSF'].describe()
```

```
count    1460.000000
mean     1057.429452
std       438.705324
min         0.000000
25%       795.750000
50%       991.500000
75%      1298.250000
max      6110.000000
Name: TotalBsmtSF, dtype: float64
```

```
1 df_Entrenamiento['BsmtFinSF1'].describe()
```

```
count    1460.000000
mean      443.639726
std       456.098091
min         0.000000
25%         0.000000
50%       383.500000
75%       712.250000
max      5644.000000
Name: BsmtFinSF1, dtype: float64
```

```
1 df_Entrenamiento['1stFlrSF'].describe()
```

```
count    1460.000000
mean     1162.626712
std       386.587738
min       334.000000
25%       882.000000
50%      1087.000000
75%      1391.250000
max      4692.000000
Name: 1stFlrSF, dtype: float64
```

```
1 df_Entrenamiento['BsmtFinSF2'].describe()
```

```
count    1460.000000
mean       46.549315
std       161.319273
min         0.000000
25%         0.000000
50%         0.000000
75%         0.000000
max      1474.000000
Name: BsmtFinSF2, dtype: float64
```

```
1 df_Entrenamiento['2ndFlrSF'].describe()
```

```
count    1460.000000
mean      346.992466
std       436.528436
min         0.000000
25%         0.000000
50%         0.000000
75%       728.000000
max      2065.000000
Name: 2ndFlrSF, dtype: float64
```

Asignatura	Datos del alumno	Fecha
Aprendizaje Automático	Apellidos: Cortes Orozco Nombre: Wilfrido	23/08/2021

```
1 df_Entrenamiento['LowQualFinSF'].describe()
```

```
count    1460.000000
mean        5.844521
std        48.623081
min         0.000000
25%         0.000000
50%         0.000000
75%         0.000000
max       572.000000
Name: LowQualFinSF, dtype: float64
```

```
1 df_Entrenamiento['FullBath'].describe()
```

```
count    1460.000000
mean        1.565068
std         0.550916
min         0.000000
25%         1.000000
50%         2.000000
75%         2.000000
max         3.000000
Name: FullBath, dtype: float64
```

```
1 df_Entrenamiento['GrLivArea'].describe()
```

```
count    1460.000000
mean     1515.463699
std       525.480383
min       334.000000
25%      1129.500000
50%      1464.000000
75%      1776.750000
max      5642.000000
Name: GrLivArea, dtype: float64
```

```
1 df_Entrenamiento['HalfBath'].describe()
```

```
count    1460.000000
mean        0.382877
std         0.502885
min         0.000000
25%         0.000000
50%         0.000000
75%         1.000000
max         2.000000
Name: HalfBath, dtype: float64
```

```
1 df_Entrenamiento['BsmtFullBath'].describe()
```

```
count    1460.000000
mean        0.425342
std         0.518911
min         0.000000
25%         0.000000
50%         0.000000
75%         1.000000
max         3.000000
Name: BsmtFullBath, dtype: float64
```

```
1 df_Entrenamiento['BedroomAbvGr'].describe()
```

```
count    1460.000000
mean        2.866438
std         0.815778
min         0.000000
25%         2.000000
50%         3.000000
75%         3.000000
max         8.000000
Name: BedroomAbvGr, dtype: float64
```

```
1 df_Entrenamiento['BsmtHalfBath'].describe()
```

```
count    1460.000000
mean        0.057534
std         0.238753
min         0.000000
25%         0.000000
50%         0.000000
75%         0.000000
max         2.000000
Name: BsmtHalfBath, dtype: float64
```

```
1 df_Entrenamiento['KitchenAbvGr'].describe()
```

```
count    1460.000000
mean        1.046575
std         0.220338
min         0.000000
25%         1.000000
50%         1.000000
75%         1.000000
max         3.000000
Name: KitchenAbvGr, dtype: float64
```

```
1 df_Entrenamiento['TotRmsAbvGrd'].describe()
```

```
count    1460.000000
mean        6.517808
std         1.625393
min         2.000000
25%         5.000000
50%         6.000000
75%         7.000000
max        14.000000
Name: TotRmsAbvGrd, dtype: float64
```

```
1 df_Entrenamiento['GarageArea'].describe()
```

```
count    1460.000000
mean      472.980137
std       213.804841
min         0.000000
25%       334.500000
50%       480.000000
75%       576.000000
max      1418.000000
Name: GarageArea, dtype: float64
```

```
1 df_Entrenamiento['Fireplaces'].describe()
```

```
count    1460.000000
mean        0.613014
std         0.644666
min         0.000000
25%         0.000000
50%         1.000000
75%         1.000000
max         3.000000
Name: Fireplaces, dtype: float64
```

```
1 df_Entrenamiento['WoodDeckSF'].describe()
```

```
count    1460.000000
mean       94.244521
std       125.338794
min         0.000000
25%         0.000000
50%         0.000000
75%       168.000000
max       857.000000
Name: WoodDeckSF, dtype: float64
```

```
1 df_Entrenamiento['GarageYrBlt'].describe()
```

```
count    1379.000000
mean     1978.506164
std        24.689725
min      1900.000000
25%      1961.000000
50%      1980.000000
75%      2002.000000
max      2010.000000
Name: GarageYrBlt, dtype: float64
```

```
1 df_Entrenamiento['OpenPorchSF'].describe()
```

```
count    1460.000000
mean       46.660274
std        66.256028
min         0.000000
25%         0.000000
50%        25.000000
75%        68.000000
max       547.000000
Name: OpenPorchSF, dtype: float64
```

```
1 df_Entrenamiento['GarageCars'].describe()
```

```
count    1460.000000
mean        1.767123
std         0.747315
min         0.000000
25%         1.000000
50%         2.000000
75%         2.000000
max         4.000000
Name: GarageCars, dtype: float64
```

```
1 df_Entrenamiento['EnclosedPorch'].describe()
```

```
count    1460.000000
mean       21.954110
std        61.119149
min         0.000000
25%         0.000000
50%         0.000000
75%         0.000000
max       552.000000
Name: EnclosedPorch, dtype: float64
```

Asignatura	Datos del alumno	Fecha
Aprendizaje Automático	Apellidos: Cortes Orozco	23/08/2021
	Nombre: Wilfrido	

```
df_Entrenamiento['3SsnPorch'].describe()
```

```
count    1460.000000
mean        3.409589
std        29.317331
min         0.000000
25%         0.000000
50%         0.000000
75%         0.000000
max       508.000000
Name: 3SsnPorch, dtype: float64
```

```
df_Entrenamiento['MoSold'].describe()
```

```
count    1460.000000
mean        6.321918
std         2.703626
min         1.000000
25%         5.000000
50%         6.000000
75%         8.000000
max        12.000000
Name: MoSold, dtype: float64
```

```
df_Entrenamiento['ScreenPorch'].describe()
```

```
count    1460.000000
mean       15.060959
std        55.757415
min         0.000000
25%         0.000000
50%         0.000000
75%         0.000000
max       480.000000
Name: ScreenPorch, dtype: float64
```

```
df_Entrenamiento['YrSold'].describe()
```

```
count    1460.000000
mean     2007.815753
std         1.328095
min      2006.000000
25%      2007.000000
50%      2008.000000
75%      2009.000000
max      2010.000000
Name: YrSold, dtype: float64
```

```
df_Entrenamiento['PoolArea'].describe()
```

```
count    1460.000000
mean        2.758904
std        40.177307
min         0.000000
25%         0.000000
50%         0.000000
75%         0.000000
max       738.000000
Name: PoolArea, dtype: float64
```

```
df_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
```

```
df_Entrenamiento['MiscVal'].describe()
```

```
count     1460.000000
mean        43.489041
std        496.123024
min          0.000000
25%          0.000000
50%          0.000000
75%          0.000000
max      15500.000000
Name: MiscVal, dtype: float64
```

- VARIABLES CATEGÓRICAS:

Ahora, para poder obtener las diferentes categorías que hay por cada variable, se usa el comando “unique” de pandas, pero también se puede usar el comando “value_counts()”; este último brindará un listado de incidencias por cada posible valor de la variable, el cuál se podrá representar con la librería seaborn.

Hay 43 “objetos”, es decir, 43 variables categóricas, los resultados obtenidos son los siguientes:

Asignatura	Datos del alumno	Fecha
Aprendizaje Automático	Apellidos: Cortes Orozco	23/08/2021
	Nombre: Wilfrido	

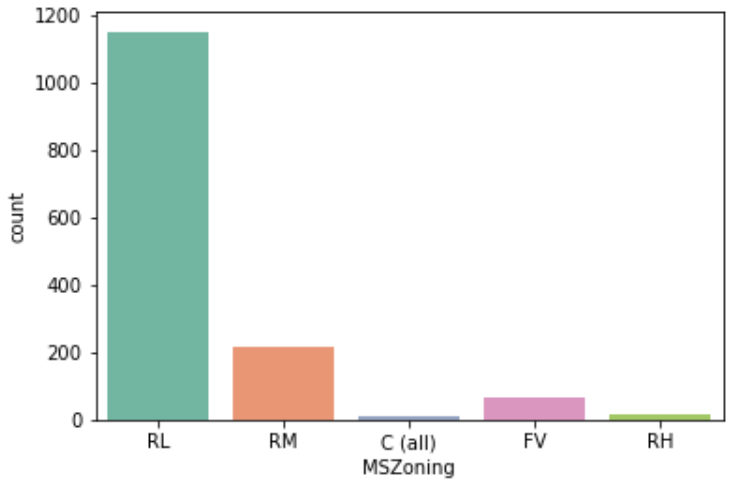
```
1 import seaborn as sns
2 import matplotlib.pyplot as plt
3 print(pd.unique(df_Entrenamiento["MSZoning"]))
```

```
['RL' 'RM' 'C (all)' 'FV' 'RH']
```

```
1 df_Entrenamiento["MSZoning"].value_counts()
```

```
RL         1151
RM          218
FV           65
RH           16
C (all)      10
Name: MSZoning, dtype: int64
```

```
1 sns.countplot(x="MSZoning", data=df_Entrenamiento, palette='Set2')
2 plt.show()
```



Ahora un listado de todos los posibles valores por variable.

Asignatura	Datos del alumno	Fecha
Aprendizaje Automático	Apellidos: Cortes Orozco Nombre: Wilfrido	23/08/2021

```
df_Entrenamiento["Street"].value_counts()
```

```
Pave    1454
Grvl       6
Name: Street, dtype: int64
```

```
df_Entrenamiento["Alley"].value_counts()
```

```
Grvl    50
Pave    41
Name: Alley, dtype: int64
```

```
df_Entrenamiento['LotShape'].value_counts()
```

```
Reg    925
IR1    484
IR2     41
IR3     10
Name: LotShape, dtype: int64
```

```
df_Entrenamiento['LandContour'].value_counts()
```

```
Lvl    1311
Bnk      63
HLS      50
Low      36
Name: LandContour, dtype: int64
```

```
df_Entrenamiento['Utilities'].value_counts()
```

```
AllPub    1459
NoSeWa       1
Name: Utilities, dtype: int64
```

```
df_Entrenamiento['LotConfig'].value_counts()
```

```
Inside     1052
Corner      263
CulDSac      94
FR2          47
FR3           4
Name: LotConfig, dtype: int64
```

```
df_Entrenamiento['LandSlope'].value_counts()
```

```
Gtl    1382
Mod      65
Sev      13
Name: LandSlope, dtype: int64
```

```
df_Entrenamiento['Neighborhood'].value_counts()
```

```
NAmes      225
CollgCr    150
OldTown    113
Edwards    100
Somerst     86
Gilbert     79
NridgHt     77
Sawyer      74
NWAmes      73
SawyerW     59
BrkSide     58
Crawfor     51
Mitchel     49
NoRidge     41
Timber      38
IDOTRR      37
ClearCr     28
StoneBr     25
SWISU       25
MeadowV     17
Blmngtn     17
BrDale      16
Veenker     11
NPkVill      9
Blueste      2
Name: Neighborhood, dtype: int64
```

```
df_Entrenamiento['Condition1'].value_counts()
```

```
Norm      1260
Feedr       81
Artery      48
RRAn        26
PosN        19
RRAe        11
PosA         8
RRNn         5
RRNe         2
Name: Condition1, dtype: int64
```

```
df_Entrenamiento['Condition2'].value_counts()
```

```
Norm      1445
Feedr        6
Artery       2
PosN         2
RRNn         2
RRAe         1
PosA         1
RRAn         1
Name: Condition2, dtype: int64
```

```
df_Entrenamiento['BldgType'].value_counts()
```

```
1Fam      1220
TwnhsE     114
Duplex      52
Twnhs       43
2fmCon      31
Name: BldgType, dtype: int64
```

```
df_Entrenamiento['HouseStyle'].value_counts()
```

```
1Story    726
2Story    445
1.5Fin    154
SLvl       65
SFoyer     37
1.5Unf     14
2.5Unf     11
2.5Fin      8
Name: HouseStyle, dtype: int64
```

```
df_Entrenamiento['RoofStyle'].value_counts()
```

```
Gable      1141
Hip         286
Flat         13
Gambrel      11
Mansard       7
Shed          2
Name: RoofStyle, dtype: int64
```

```
df_Entrenamiento['RoofMatl'].value_counts()
```

```
CompShg    1434
Tar&Grv      11
WdShngl       6
WdShake       5
Membran       1
Roll          1
ClyTile       1
Metal         1
Name: RoofMatl, dtype: int64
```

```
df_Entrenamiento['Exterior1st'].value_counts()
```

```
VinylSd    515
HdBoard    222
MetalSd    220
Wd Sdng    206
Plywood    108
CemntBd     61
BrkFace     50
WdShing     26
Stucco      25
AsbShng     20
BrkComm      2
Stone        2
CBlock       1
ImStucc      1
AsphShn      1
Name: Exterior1st, dtype: int64
```

```
df_Entrenamiento['Exterior2nd'].value_counts()
```

```
VinylSd    504
MetalSd    214
HdBoard    207
Wd Sdng    197
Plywood    142
CmentBd     60
Wd Shng     38
Stucco      26
BrkFace     25
AsbShng     20
ImStucc     10
Brk Cmn      7
Stone        5
AsphShn      3
CBlock       1
Other        1
```

```
df_Entrenamiento['MasVnrType'].value_counts()
```

```
None       864
BrkFace    445
Stone      128
BrkCmn      15
Name: MasVnrType, dtype: int64
```

```
df_Entrenamiento['ExterQual'].value_counts()
```

```
TA    906
Gd    488
Ex     52
Fa     14
Name: ExterQual, dtype: int64
```

```
df_Entrenamiento['ExterCond'].value_counts()
```

```
TA    1282
Gd     146
Fa      28
Ex       3
Po       1
Name: ExterCond, dtype: int64
```

```
df_Entrenamiento['Foundation'].value_counts()
```

```
PConc     647
CBlock    634
BrkTil    146
Slab       24
Stone       6
Wood        3
Name: Foundation, dtype: int64
```

```
df_Entrenamiento['BsmtQual'].value_counts()
```

```
TA    649
Gd    618
Ex    121
Fa     35
Name: BsmtQual, dtype: int64
```

```
df_Entrenamiento['BsmtCond'].value_counts()
```

```
TA    1311
Gd      65
Fa      45
Po       2
Name: BsmtCond, dtype: int64
```

```
df_Entrenamiento['BsmtExposure'].value_counts()
```

```
No    953
Av    221
Gd    134
Mn    114
Name: BsmtExposure, dtype: int64
```

```
df_Entrenamiento['BsmtFinType1'].value_counts()
```

```
Unf    430
GLQ    418
ALQ    220
BLQ    148
Rec    133
LwQ     74
Name: BsmtFinType1, dtype: int64
```

```
df_Entrenamiento['BsmtFinType2'].value_counts()
```

```
Unf    1256
Rec      54
LwQ      46
BLQ      33
ALQ      19
GLQ      14
Name: BsmtFinType2, dtype: int64
```

Asignatura	Datos del alumno	Fecha
Aprendizaje Automático	Apellidos: Cortes Orozco	23/08/2021
	Nombre: Wilfrido	

```
df_Entrenamiento['Heating'].value_counts()
```

```
GasA     1428
GasW       18
Grav        7
Wall        4
OthW        2
Floor       1
Name: Heating, dtype: int64
```

```
df_Entrenamiento['HeatingQC'].value_counts()
```

```
Ex    741
TA    428
Gd    241
Fa     49
Po      1
Name: HeatingQC, dtype: int64
```

```
df_Entrenamiento['CentralAir'].value_counts()
```

```
Y    1365
N      95
Name: CentralAir, dtype: int64
```

```
df_Entrenamiento['Electrical'].value_counts()
```

```
SBrkr    1334
FuseA      94
FuseF      27
FuseP       3
Mix         1
Name: Electrical, dtype: int64
```

```
df_Entrenamiento['KitchenQual'].value_counts()
```

```
TA    735
Gd    586
Ex    100
Fa     39
Name: KitchenQual, dtype: int64
```

```
df_Entrenamiento['Functional'].value_counts()
```

```
Typ     1360
Min2      34
Min1      31
Mod       15
Maj1      14
Maj2       5
Sev        1
Name: Functional, dtype: int64
```

```
df_Entrenamiento['FireplaceQu'].value_counts()
```

```
Gd    380
TA    313
Fa     33
Ex     24
Po     20
Name: FireplaceQu, dtype: int64
```

```
df_Entrenamiento['GarageType'].value_counts()
```

```
Attchd     870
Detchd     387
BuiltIn     88
Basment     19
CarPort      9
2Types       6
Name: GarageType, dtype: int64
```

```
df_Entrenamiento['GarageFinish'].value_counts()
```

```
Unf    605
RFn    422
Fin    352
Name: GarageFinish, dtype: int64
```

```
df_Entrenamiento['GarageQual'].value_counts()
```

```
TA    1311
Fa      48
Gd      14
Ex       3
Po       3
Name: GarageQual, dtype: int64
```

```
df_Entrenamiento['GarageCond'].value_counts()
```

```
TA    1326
Fa      35
Gd       9
Po       7
Ex       2
Name: GarageCond, dtype: int64
```

```
df_Entrenamiento['PavedDrive'].value_counts()
```

```
Y    1340
N      90
P      30
Name: PavedDrive, dtype: int64
```

```
df_Entrenamiento['PoolQC'].value_counts()
```

```
Gd    3
Fa    2
Ex    2
Name: PoolQC, dtype: int64
```

```
df_Entrenamiento['Fence'].value_counts()
```

```
MnPrv    157
GdPrv     59
GdWo      54
MnWw      11
Name: Fence, dtype: int64
```

```
df_Entrenamiento['MiscFeature'].value_counts()
```

```
Shed    49
Gar2     2
Othr     2
TenC     1
Name: MiscFeature, dtype: int64
```

```
df_Entrenamiento['SaleType'].value_counts()
```

```
WD       1267
New       122
COD        43
ConLD       9
ConLw       5
ConLI       5
CWD         4
Oth         3
Con         2
Name: SaleType, dtype: int64
```

```
df_Entrenamiento['SaleCondition'].value_counts()
```

```
Normal     1198
Partial     125
Abnorml     101
Family       20
Alloca       12
AdjLand       4
Name: SaleCondition, dtype: int64
```

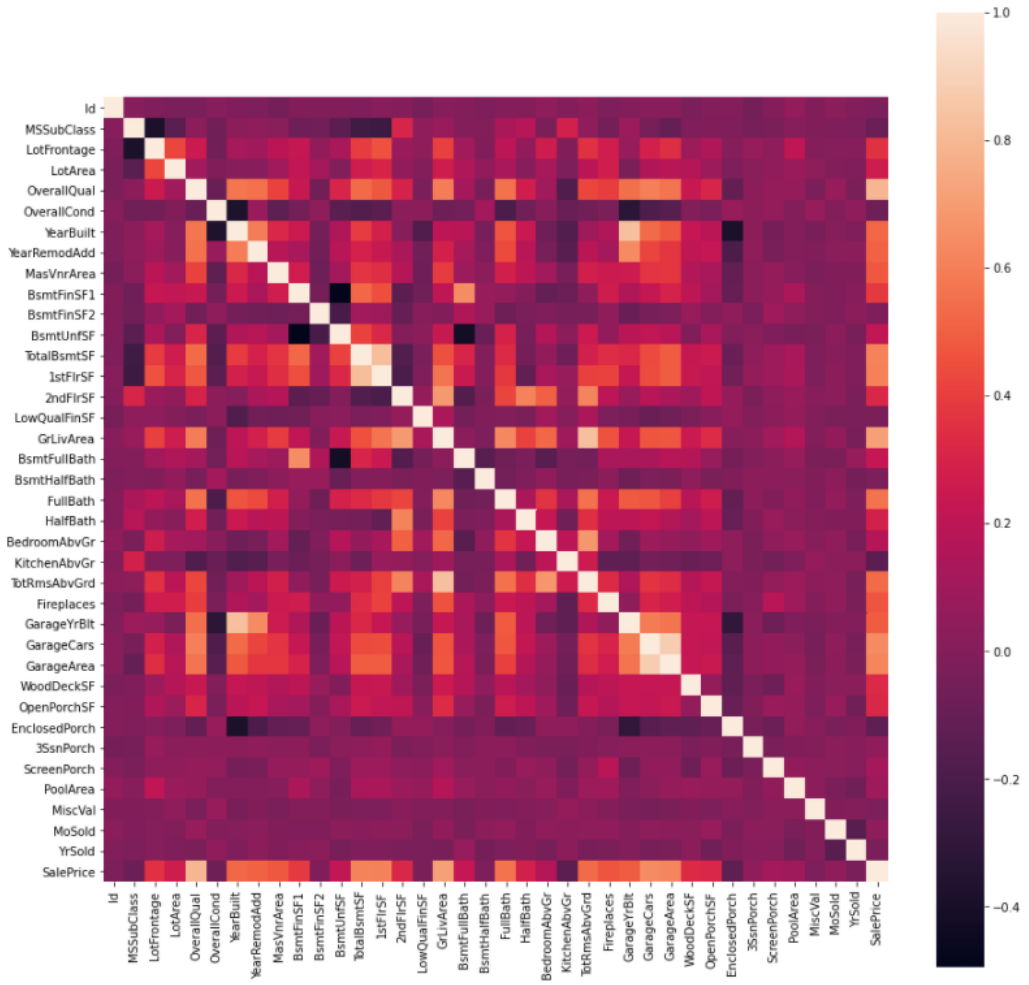
Asignatura	Datos del alumno	Fecha
Aprendizaje Automático	Apellidos: Cortes Orozco	23/08/2021
	Nombre: Wilfrido	

- MATRIZ DE CORRELACIONES

Para poder crear la matriz de correlaciones, se hizo uso de la función “corr()” del data frame.

```
correlacion = df_Entrenamiento.corr()
```

```
figura, axis = plt.subplots(figsize=(15,15))
sns.heatmap(correlacion,square=True)
plt.show
```



Los valores más claros son aquellos que se ven correlacionados más directamente. Las relaciones más directas son:

- 1stFlrSF con TotalBsmtSF.
- OverallQual con SalePrice.
- GarageYrBlt con YearBuilt.
- TotRmsAbvGrd con GrLivArea.
- GarageArea con GarageCars.

Asignatura	Datos del alumno	Fecha
Aprendizaje Automático	Apellidos: Cortes Orozco	23/08/2021
	Nombre: Wilfrido	

Considerando la variable principal que buscamos (SalePrice), la variable más importante directamente relacionada es OverAllQual y GrLivArea. Pero se puede ser más preciso con el siguiente comando:

```
1 correlacion2=df_Entrenamiento.corr()
2 print(correlacion2['SalePrice'].sort_values(ascending=False)[:10],'\n')
```

```
SalePrice       1.000000
OverallQual     0.790982
GrLivArea       0.708624
GarageCars      0.640409
GarageArea      0.623431
TotalBsmtSF     0.613581
1stFlrSF        0.605852
FullBath        0.560664
TotRmsAbvGrd    0.533723
YearBuilt       0.522897
Name: SalePrice, dtype: float64
```

Tratamiento de missing.

Lo primero es localizar los valores faltantes. Se pueden obtener con el comando previo “info()”. Las variables que más datos nulos tienen son: Alley (1369), FireplaceQu (690), PoolQC (1453), Fence (1179) y MiscFeature (1406).

En el caso de las que más datos faltantes tienen, de tipo categórica, se decidió la eliminación de la variable, esto es para: “Alley”, “PoolQC”, “Fence” y “MiscFeature”, ya que no tienen tanto peso en la definición del precio de la venta de la casa. Para la eliminación se utiliza el siguiente comando:

In [99]:
```
1 df_Temporal = df_Entrenamiento
2 df_Temporal = df_Temporal.drop(['Alley', 'PoolQC', 'Fence', 'MiscFeature'], axis=1)
3 df_Temporal.head()
```

Out[99]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContour	Utilities	LotConfig	...	EnclosedPorch	3SsnPorch	ScreenPorch	PoolAre
0	1	60	RL	65.0	8450	Pave	Reg	Lvl	AllPub	Inside	...	0	0	0	
1	2	20	RL	80.0	9600	Pave	Reg	Lvl	AllPub	FR2	...	0	0	0	
2	3	60	RL	68.0	11250	Pave	IR1	Lvl	AllPub	Inside	...	0	0	0	
3	4	70	RL	60.0	9550	Pave	IR1	Lvl	AllPub	Corner	...	272	0	0	
4	5	60	RL	84.0	14260	Pave	IR1	Lvl	AllPub	FR2	...	0	0	0	

5 rows × 77 columns

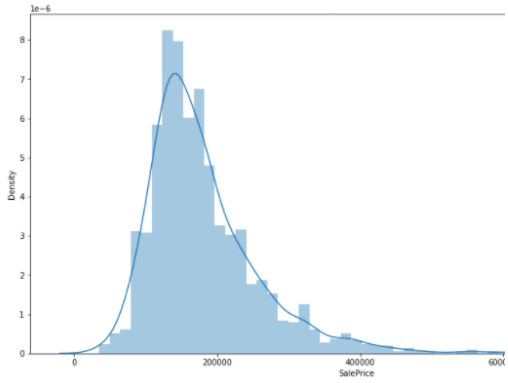
Asignatura	Datos del alumno	Fecha
Aprendizaje Automático	Apellidos: Cortes Orozco	23/08/2021
	Nombre: Wilfrido	

Ahora que quedan menos datos, tomaré la mediana para poder rellenar los datos faltantes, la mediana afecta menos la curva que tiene la grafica

```
plt.figure(figsize=(15,8))
sns.distplot(df_Entrenamiento['SalePrice'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

```
<AxesSubplot:xlabel='SalePrice', ylabel='Density'>
```




```
df_Entrenamiento['SalePrice'].mean()
```

```
180921.19589041095
```

```
df_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
```

El promedio es 180921, si lo incrementamos podríamos generar un cambio severo en la gráfica, en cambio si incrementamos el valor de la mediana (al meter más datos a ese valor) se seguirá respetando la curva de la gráfica, es decir, se verá menos afectada.

Por lo que se decidió rellenar en base a la mediana los datos faltantes. Para eso se utiliza el comando “fillna()” con parámetro la mediana del frame set.

Una vez hecho eso, ahora falta rellenar los datos de las demás variables categóricas, para ello se utilizará el valor siguiente y el valor anterior.

```
df_Entrenamiento_limpio = df_Temporal.fillna(df_Temporal.median())
```

```
dataNull = df_Entrenamiento_limpio.isnull().sum()
print([dataNull[dataNull>0]])
```

```
[MasVnrType      8
BsmtQual        37
BsmtCond        37
BsmtExposure    38
BsmtFinType1    37
BsmtFinType2    38
Electrical       1
FireplaceQu    690
GarageType      81
GarageFinish    81
GarageQual      81
GarageCond      81
dtype: int64]
```

Para asignar el valor siguiente y el valor anterior se utiliza el comando “fillna” con “ffill” y “bfill”.

```
df_Entrenamiento_limpio = df_Entrenamiento_limpio.fillna(method='ffill')
```

```
dataNull = df_Entrenamiento_limpio.isnull().sum()
print([dataNull[dataNull>0]])
```

```
[FireplaceQu    1
dtype: int64]
```

```
df_Entrenamiento_limpio = df_Entrenamiento_limpio.fillna(method='bfill')
```

```
dataNull = df_Entrenamiento_limpio.isnull().sum()
print([dataNull[dataNull>0]])
```

```
[Series([], dtype: int64)]
```

Al final se puede observar que ya no hay valores nulos en el set. Ahora se consolidan los datos para dejar todo como numérico.

Asignatura	Datos del alumno	Fecha
Aprendizaje Automático	Apellidos: Cortes Orozco	23/08/2021
	Nombre: Wilfrido	

Para ello se dejan capturas del método de consolidación:

Consolidación de datos

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.pipeline import Pipeline
```

```
class MultiColumnLabelEncoder:
    def __init__(self,columns = None):
        self.columns = columns # array of column names to encode

    def fit(self,X,y=None):
        return self # not relevant here

    def transform(self,X):
        '''
        Transforms columns of X specified in self.columns using
        LabelEncoder(). If no columns specified, transforms all
        columns in X.
        '''
        output = X.copy()
        if self.columns is not None:
            for col in self.columns:
                output[col] = LabelEncoder().fit_transform(output[col])
        else:
            for colname,col in output.iteritems():
                output[colname] = LabelEncoder().fit_transform(col)
        return output

    def fit_transform(self,X,y=None):
        return self.fit(X,y).transform(X)
```

```
from sklearn.preprocessing import LabelEncoder

df_categorias = df_Entrenamiento_limpio.select_dtypes(exclude=['int64','float64'])
df_nums = df_Entrenamiento_limpio.select_dtypes(exclude=['object'])
#df_nums = MultiColumnLabelEncoder(columns = list(df_nums.columns.values)).fit_transform(df_nums)
```

```
df_consolidados = pd.concat([df_nums,df_categorias], axis=1)
df_consolidados.head()
df_consolidados.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1460 entries, 0 to 1459
Data columns (total 77 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             1460 non-null   int64
 1   MSSubClass     1460 non-null   int64
 2   LotFrontage    1460 non-null   float64
 3   LotArea        1460 non-null   int64
 4   OverallQual    1460 non-null   int64
 5   OverallCond    1460 non-null   int64
 6   YearBuilt      1460 non-null   int64
 7   YearRemodAdd   1460 non-null   int64
 8   MasVnrArea     1460 non-null   float64
 9   BsmtFinSF1     1460 non-null   int64
 10  BsmtFinSF2     1460 non-null   int64
 11  BsmtUnfSF      1460 non-null   int64
 12  TotalBsmtSF    1460 non-null   int64
 13  1stFlrSF       1460 non-null   int64
 14  2ndFlrSF       1460 non-null   int64
 15  LowQualFinSF   1460 non-null   int64
 16  GrLivArea      1460 non-null   int64
 17  BsmtFullBath   1460 non-null   int64
 18  BsmtHalfBath   1460 non-null   int64
 19  FullBath       1460 non-null   int64
 20  HalfBath       1460 non-null   int64
 21  BedroomAbvGr   1460 non-null   int64
 22  KitchenAbvGr   1460 non-null   int64
 23  TotRmsAbvGrd   1460 non-null   int64
 24  Fireplaces     1460 non-null   int64
 25  GarageYrBlt    1460 non-null   float64
 26  GarageCars     1460 non-null   int64
 27  GarageArea     1460 non-null   int64
 28  WoodDeckSF     1460 non-null   int64
 29  OpenPorchSF    1460 non-null   int64
 30  EnclosedPorch  1460 non-null   int64
 31  3SsnPorch      1460 non-null   int64
 32  ScreenPorch    1460 non-null   int64
 33  PoolArea       1460 non-null   int64
 34  MiscVal        1460 non-null   int64
 35  MoSold         1460 non-null   int64
 36  YrSold         1460 non-null   int64
 37  SalePrice      1460 non-null   int64
 38  MSZoning       1460 non-null   object
 39  Street         1460 non-null   object
 40  LotShape       1460 non-null   object
 41  LandContour    1460 non-null   object
 42  Utilities      1460 non-null   object
 43  LotConfig      1460 non-null   object
 44  LandSlope      1460 non-null   object
 45  Neighborhood   1460 non-null   object
 46  Condition1     1460 non-null   object
 47  Condition2     1460 non-null   object
 48  BldgType       1460 non-null   object
 49  HouseStyle     1460 non-null   object
 50  RoofStyle      1460 non-null   object
 51  RoofMatl       1460 non-null   object
 52  Exterior1st    1460 non-null   object
 53  Exterior2nd    1460 non-null   object
```

```
df_consolidados = df_consolidados[list(df_Entrenamiento_limpio.columns.values)]
df_consolidados.head()
df_consolidados.info
```

```
df_consolidados = MultiColumnLabelEncoder(columns = ['MSZoning',
 'Street',
 'LotShape',
 'LandContour',
 'Utilities',
 'LotConfig',
 'LandSlope',
 'Neighborhood',
 'Condition1',
 'Condition2',
 'BldgType',
 'HouseStyle',
 'RoofStyle',
 'RoofMatl',
 'Exterior1st',
 'Exterior2nd',
 'MasVnrType',
 'ExterQual',
 'ExterCond',
 'Foundation',
 'BsmtQual',
 'BsmtCond',
 'BsmtExposure',
 'BsmtFinType1',
 'BsmtFinType2',
 'Heating',
 'HeatingQC',
 'CentralAir',
 'Electrical',
 'KitchenQual',
 'Functional',
 'FireplaceQu',
 'GarageType',
 'GarageFinish',
 'GarageQual',
 'GarageCond',
 'PavedDrive',
 'SaleType',
 'SaleCondition']).fit_transform(df_consolidados)
df_consolidados.head()
df_consolidados.info()
print(df_consolidados['SaleType'])
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1460 entries, 0 to 1459
Data columns (total 77 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             1460 non-null   int64
 1   MSSubClass     1460 non-null   int64
 2   MSZoning       1460 non-null   int32
```

Después de consolidar:

```
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             1460 non-null   int64
 1   MSSubClass     1460 non-null   int64
 2   MSZoning       1460 non-null   int32
 3   LotFrontage    1460 non-null   float64
 4   LotArea        1460 non-null   int64
 5   Street         1460 non-null   int32
 6   LotShape       1460 non-null   int32
 7   LandContour    1460 non-null   int32
 8   Utilities      1460 non-null   int32
 9   LotConfig      1460 non-null   int32
 10  LandSlope      1460 non-null   int32
 11  Neighborhood   1460 non-null   int32
 12  Condition1     1460 non-null   int32
 13  Condition2     1460 non-null   int32
 14  BldgType       1460 non-null   int32
 15  HouseStyle     1460 non-null   int32
 16  OverallQual    1460 non-null   int64
 17  OverallCond    1460 non-null   int64
 18  YearBuilt      1460 non-null   int64
 19  YearRemodAdd   1460 non-null   int64
 20  RoofStyle      1460 non-null   int32
 21  RoofMatl       1460 non-null   int32
 22  Exterior1st    1460 non-null   int32
 23  Exterior2nd    1460 non-null   int32
 24  MasVnrType     1460 non-null   int32
 25  MasVnrArea     1460 non-null   float64
 26  ExterQual      1460 non-null   int32
 27  ExterCond      1460 non-null   int32
 28  Foundation     1460 non-null   int32
 29  BsmtQual       1460 non-null   int32
 30  BsmtCond       1460 non-null   int32
 31  BsmtExposure   1460 non-null   int32
 32  BsmtFinType1   1460 non-null   int32
 33  BsmtFinSF1     1460 non-null   int64
 34  BsmtFinType2   1460 non-null   int32
 35  BsmtFinSF2     1460 non-null   int64
 36  BsmtUnfSF      1460 non-null   int64
 37  TotalBsmtSF    1460 non-null   int64
 38  Heating        1460 non-null   int32
 39  HeatingQC      1460 non-null   int32
 40  CentralAir     1460 non-null   int32
 41  Electrical     1460 non-null   int32
 42  1stFlrSF       1460 non-null   int64
 43  2ndFlrSF       1460 non-null   int64
 44  LowQualFinSF   1460 non-null   int64
 45  GrLivArea      1460 non-null   int64
 46  BsmtFullBath   1460 non-null   int64
 47  BsmtHalfBath   1460 non-null   int64
 48  FullBath       1460 non-null   int64
 49  HalfBath       1460 non-null   int64
 50  BedroomAbvGr   1460 non-null   int64
 51  KitchenAbvGr   1460 non-null   int64
 52  KitchenQual    1460 non-null   int32
 53  TotRmsAbvGrd   1460 non-null   int64
 54  Functional     1460 non-null   int32
 55  Fireplaces     1460 non-null   int64
 56  FireplaceQu    1460 non-null   int32
 57  GarageType     1460 non-null   int32
 58  GarageYrBlt    1460 non-null   float64
 59  GarageFinish   1460 non-null   int32
 60  GarageCars     1460 non-null   int64
 61  GarageArea     1460 non-null   int64
 62  GarageQual     1460 non-null   int32
 63  GarageCond     1460 non-null   int32
 64  PavedDrive     1460 non-null   int32
 65  WoodDeckSF     1460 non-null   int64
 66  OpenPorchSF    1460 non-null   int64
 67  EnclosedPorch  1460 non-null   int64
 68  3SsnPorch      1460 non-null   int64
 69  ScreenPorch    1460 non-null   int64
 70  PoolArea       1460 non-null   int64
 71  MiscVal        1460 non-null   int64
 72  MoSold         1460 non-null   int64
 73  YrSold         1460 non-null   int64
 74  SaleType       1460 non-null   int32
 75  SaleCondition  1460 non-null   int32
 76  SalePrice      1460 non-null   int64
dtypes: float64(3), int32(39), int64(35)
memory usage: 656.0 KB
0       8
1       8
2       8
3       8
4       8
      ..
1455    8
1456    8
1457    8
1458    8
1459    8
Name: SaleType, Length: 1460, dtype: int32
```

Asignatura	Datos del alumno	Fecha
Aprendizaje Automático	Apellidos: Cortes Orozco	23/08/2021
	Nombre: Wilfrido	

Problema de regresión, árboles y random forest.

ÁRBOL DE REGRESIÓN.

OTRO TEST DE ÁRBOL

In [151]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import r2_score

Xtrain, Xtest, ytrain, ytest = train_test_split(df_consolidados, y, test_size=0.2, random_state=12
my_dt = GridSearchCV(DecisionTreeRegressor(random_state=12345),
                     {'min_samples_split': list(range(20,50,2)),
                     'max_features':[0.6, 0.7, 0.8, 0.9, 1.],
                     'criterion': ['mse','mae']},
                     scoring='r2',n_jobs=-1)
Xtrain.shape
ytrain.shape
my_dt.fit(Xtrain,ytrain)
```

Out[151]:
```
GridSearchCV(estimator=DecisionTreeRegressor(random_state=12345), n_jobs=-1,
             param_grid={'criterion': ['mse', 'mae'],
                         'max_features': [0.6, 0.7, 0.8, 0.9, 1.0],
                         'min_samples_split': [20, 22, 24, 26, 28, 30, 32, 34,
                                               36, 38, 40, 42, 44, 46, 48]},
             scoring='r2')
```

In [152]:
```
my_dt.best_params_
```

Out[152]:
```
{'criterion': 'mse', 'max_features': 1.0, 'min_samples_split': 20}
```

In [156]:
```
print(r2_score(list(ytest), list(my_dt.predict(Xtest))))

```

```
0.9844407674092782
```

In [163]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.neighbors import KNeighborsRegressor
my_knn = GridSearchCV(KNeighborsRegressor(),
                      {'n_neighbors':[2, 4, 6, 8, 10, 12]},
                      scoring = 'r2', n_jobs = -1)
my_knn.fit(Xtrain, ytrain)
print(r2_score(list(ytest), list(my_knn.predict(Xtest))))

```

```
0.9975104714348556
```

In [164]:
```
print(my_knn.best_estimator_)
```

```
KNeighborsRegressor(n_neighbors=2)
```

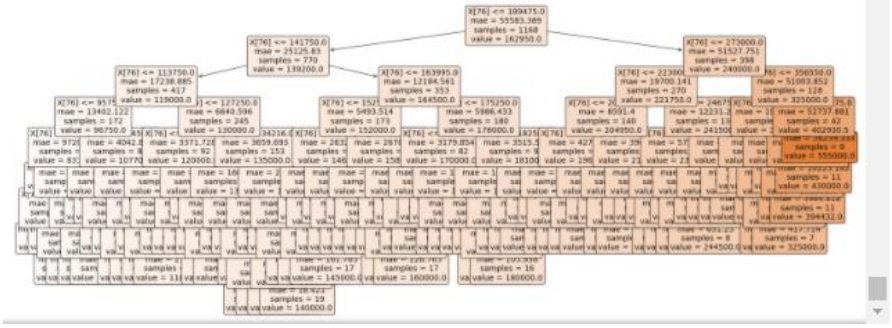
Asignatura	Datos del alumno	Fecha
Aprendizaje Automático	Apellidos: Cortes Orozco	23/08/2021
	Nombre: Wilfrido	

```
testFinal = DecisionTreeRegressor(min_samples_split = 20, max_features=1.0, criterion = 'mae')
testFinal.fit(Xtrain,ytrain)
ordenados=testFinal.feature_importances_
print(ordenados)
print(r2_score(list(ytest), list(testFinal.predict(Xtest))))
```

```
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 1.]
0.9702584182839398
```

```
from sklearn.tree import plot_tree
plt.figure(figsize=(25,10))
plot_tree(testFinal,filled=True,rounded=True,fontsize=14)
```

```
Text(1373.5384615384614, 244.62, 'mae = 10225.182\nsamples = 11\nvalue = 430000.0'),
Text(1384.2692307692307, 298.98, 'mae = 56259.333\nsamples = 9\nvalue = 555000.0')]
```



El árbol logró predecir con una cantidad muy aceptable, obtuvo un resultado de 97% de acierto con los mejores parámetros. Sin embargo, con el KNN obtuvo un 99% de acierto.

RANDOM FOREST

```
# Create Train test split
from sklearn.model_selection import train_test_split
Xtrain, Xtest, ytrain, ytest = train_test_split(df_consolidados, y, test_size=0.2, random_state=12

from sklearn.ensemble import RandomForestRegressor
randomForest = RandomForestRegressor()
randomForest.fit(Xtrain, ytrain)
prediccion = randomForest.predict(Xtest)

from sklearn.metrics import r2_score
r2_score(list(ytest), list(prediccion))
```

```
0.9916902066525514
```

El random ofrece un resultado superior al obtenido por el árbol de regresión, pero inferior al obtenido por los KNN, sin embargo, cuenta con un 99.16% de acierto en su predicción.

Algo que noté, el random forest cargó con mucha mayor velocidad, por encima de lo que tardó el árbol de regresión, desconozco si sea un caso aislado, pero me parece que es más veloz.

Considero el random como un método superior y hasta cierto punto más versátil que el árbol de regresión líneal.

Asignatura	Datos del alumno	Fecha
Aprendizaje Automático	Apellidos: Cortes Orozco	23/08/2021
	Nombre: Wilfrido	

Problema de clasificación.

CONDICIONALES

```
df_finales = df_consolidados
#df_finales.head()
df_finales = df_finales.drop('SalePrice', axis=1)
df_finales.head()
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContour	Utilities	LotConfig	...	OpenPorchSF	En
0	1	60	3	65.0	8450	1	3	3	0	4	...	61	
1	2	20	3	80.0	9600	1	3	3	0	2	...	0	
2	3	60	3	68.0	11250	1	0	3	0	4	...	42	
3	4	70	3	60.0	9550	1	0	3	0	0	...	35	
4	5	60	3	84.0	14260	1	0	3	0	2	...	84	

5 rows × 76 columns

```
df_finales2 = df_finales
df_finales.shape
df_finales2.shape
df_finales.info()
conditions = [
    (df_consolidados['SalePrice'] <= 100000),
    (df_consolidados['SalePrice'] > 100000) & (df_consolidados['SalePrice'] <= 500000),
    (df_consolidados['SalePrice'] > 500000)
    ]
values = ['Plow','PMedium','PHigh']

df_finales['CatPrice'] = np.select(conditions,values)
#yclass = df_finales['CatPrice']

```

Se definen las clases, se utilizarán en un clasificador:

```
df_finales['CatPrice'].value_counts()
```

```
PMedium    1328
Plow        123
PHigh         9
Name: CatPrice, dtype: int64
```

```
yclass = np.select(conditions, values)
df_finales = df_finales.drop('CatPrice',axis=1)
Xclasstrain, Xclasstest, yclasstrain, yclasstest = train_test_split(df_finales,yclass,test_size=0.
```

```
max_depth = []
acc_gini = []
acc_entropy = []

for i in range (1,8):
    dtree = DecisionTreeClassifier(criterion='gini', max_depth=i, random_state=12345)
    dtree.fit(Xclasstrain, yclasstrain)
    yclasspred = dtree.predict(Xclasstest)
    acc_gini.append(metrics.accuracy_score(yclasstest, yclasspred))
    print("i=",i,"gini     :", metrics.accuracy_score(yclasstest, yclasspred) )
    ##
    dtree = DecisionTreeClassifier(criterion='entropy', max_depth=i, random_state=12345)
    dtree.fit(Xclasstrain, yclasstrain)
    yclasspred = dtree.predict(Xclasstest)
    acc_entropy.append(metrics.accuracy_score(yclasstest, yclasspred))
    ##
    max_depth.append(i)
    print("i=",i,"entropy : ", metrics.accuracy_score(yclasstest, yclasspred) )

```

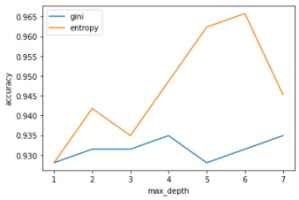
```
i= 1 gini     : 0.928082191780822
i= 1 entropy :  0.928082191780822
i= 2 gini     : 0.9315068493150684
i= 2 entropy :  0.9417808219178082
i= 3 gini     : 0.9315068493150684
i= 3 entropy :  0.934931506849315
i= 4 gini     : 0.934931506849315
i= 4 entropy :  0.9486301369863014
i= 5 gini     : 0.928082191780822
i= 5 entropy :  0.9623287671232876
i= 6 gini     : 0.9315068493150684
i= 6 entropy :  0.9657534246575342
i= 7 gini     : 0.934931506849315
i= 7 entropy :  0.9452054794520548
```

```
d = pd.DataFrame({'acc_gini':pd.Series(acc_gini),
                  'acc_entropy':pd.Series(acc_entropy),
                  'max_depth':pd.Series(max_depth)})
plt.plot('max_depth','acc_gini',data=d, label='gini')
plt.plot('max_depth','acc_entropy',data=d, label='entropy')
plt.xlabel('max_depth')
plt.ylabel('accuracy')
plt.legend()

dtreeg = DecisionTreeClassifier(criterion='gini',max_depth=6,random_state=12345)
dtreeg = dtreeg.fit(Xclasstrain, yclasstrain)
yclasspred = dtreeg.predict(Xclasstest)
print("Desempeño gini=",metrics.accuracy_score(yclasstest,yclasspred))

clf = DecisionTreeClassifier(criterion='entropy',max_depth=6,random_state=12345)
clf = clf.fit(Xclasstrain, yclasstrain)
yclasspred = clf.predict(Xclasstest)
print("Desempeño entropy=",metrics.accuracy_score(yclasstest,yclasspred))
```

```
Desempeño gini= 0.9315068493150684
Desempeño entropy= 0.9657534246575342
```



Gini resultó un poco menos eficiente en las pruebas que se hicieron, al menos en este caso, entropy demostró una precisión mucho más elevada, aunque se ve que a cierto punto, con la profundidad incrementando, el valor de la precisión de entropy empezó a decaer, sin embargo, el rendimiento general sigue siendo superior.