Datos Inmobiliarios de la Empresa Gilmar

Modelo predictivo del precio de los inmuebles

Contenido

Carolina Cordo Nievas

08/03/2023

El presente estudio tiene por objeto obtener, a partir de los datos provistos, un modelo que prediga el precio de los inmuebles.

Estudio previo de los datos (informe EDA)

Los datos cuentan con 16 variables y un total de 1728 registros.

```
df.shape #  registros 1728 / 16 variables
```

```
(1728, 16)
```

En principio no se observan nulos. Los datos cuantitativos son enteros.

```
RangeIndex: 1728 entries, 0 to 1727
Data columns (total 16 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   precio         1728 non-null   int64 
 1   m2Brutos       1728 non-null   int64 
 2   edad           1728 non-null   int64 
 3   valorTerreno   1728 non-null   int64 
 4   m2Util         1728 non-null   int64 
 5   perUni         1728 non-null   int64 
 6   numDormi       1728 non-null   int64 
 7   numChime       1728 non-null   int64 
 8   numServi       1728 non-null   int64 
 9   numHabita      1728 non-null   int64 
 10  calefaccion    1728 non-null   object
 11  alimentacion   1728 non-null   object
 12  tipoDesague    1728 non-null   object
 13  conVistas      1728 non-null   object
 14  construccion   1728 non-null   object
 15  aire           1728 non-null   object
dtypes: int64(10), object(6)
memory usage: 216.1+ KB
```

Se analizan las variables una a una, en cuanto a su significado, valores, rangos, impacto en la variable objetivo, etc.

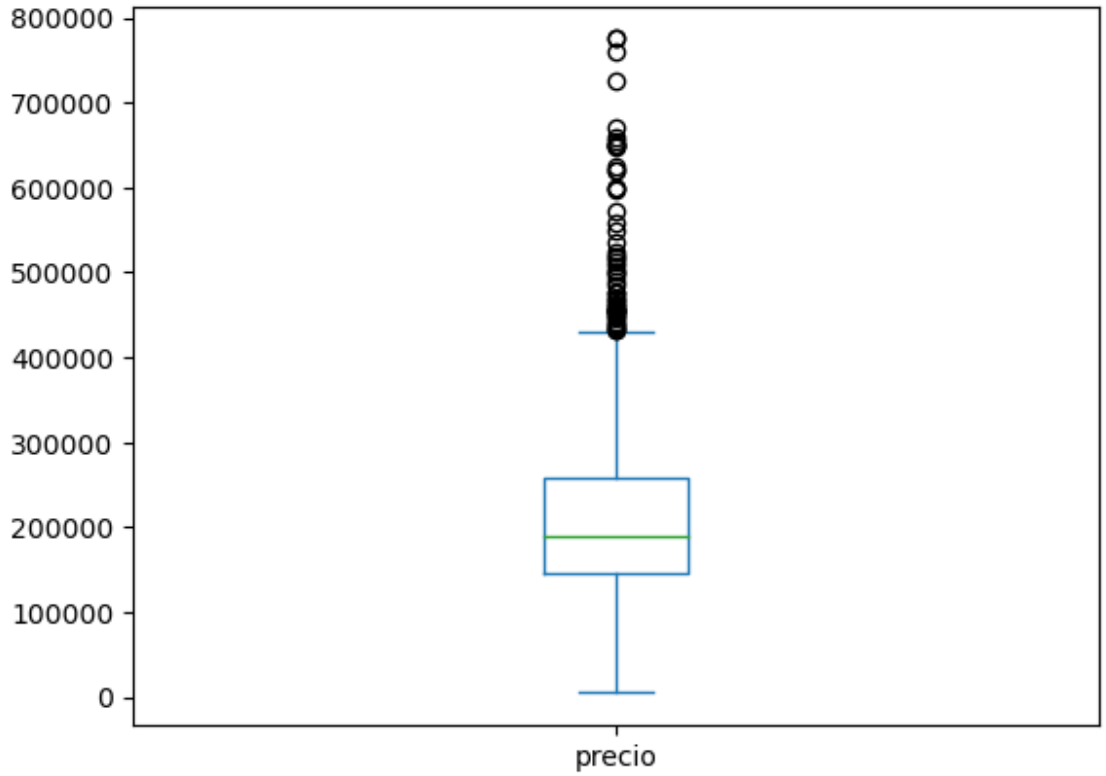
Asimismo, se detalla el tratamiento que se les dará y si se incluirán o no en el modelado.

• **precio** (precio de la vivienda). Variable cuantitativa.

Es nuestra variable objetivo. Presenta outliers que deben estudiarse y tratarse.

```
df[['precio']].plot.box()
```

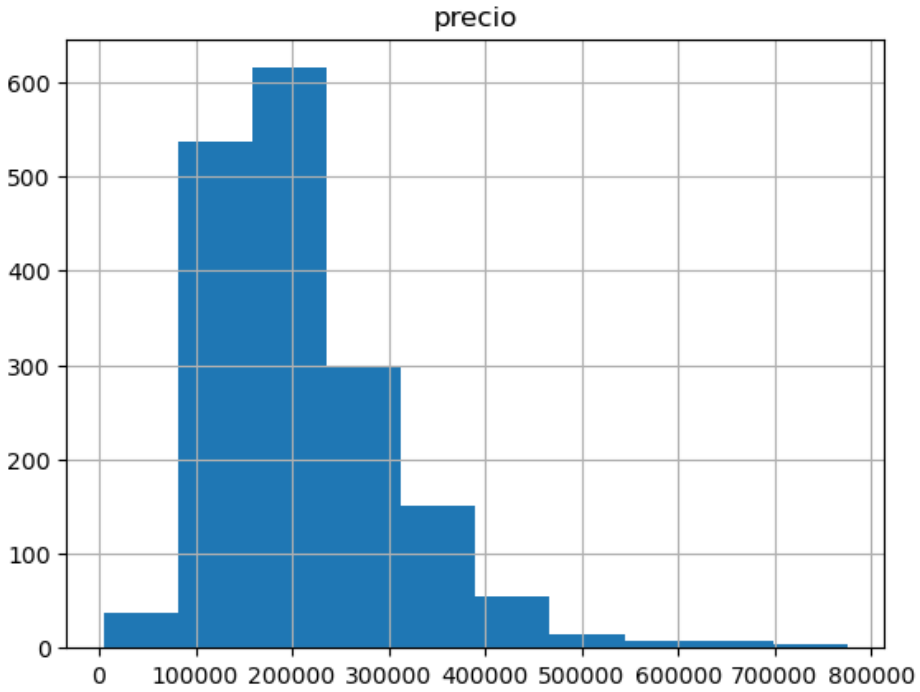
```
<AxesSubplot:>
```



Su distribución por histograma es la siguiente.

```
df[['precio']].hist()
```

```
array([[<AxesSubplot:title={'center':'precio'}>]], dtype=object)
```



Se observa que el mayor porcentaje se encuentra entre lo 80k y 460k aproximadamente.

Numéricamente los outliers que presenta resultan en un 3% de los casos, considerando un coeficiente de 1.5 aplicado al rango intercuartílico. *Estos 53 registros serán eliminados del modelado.*

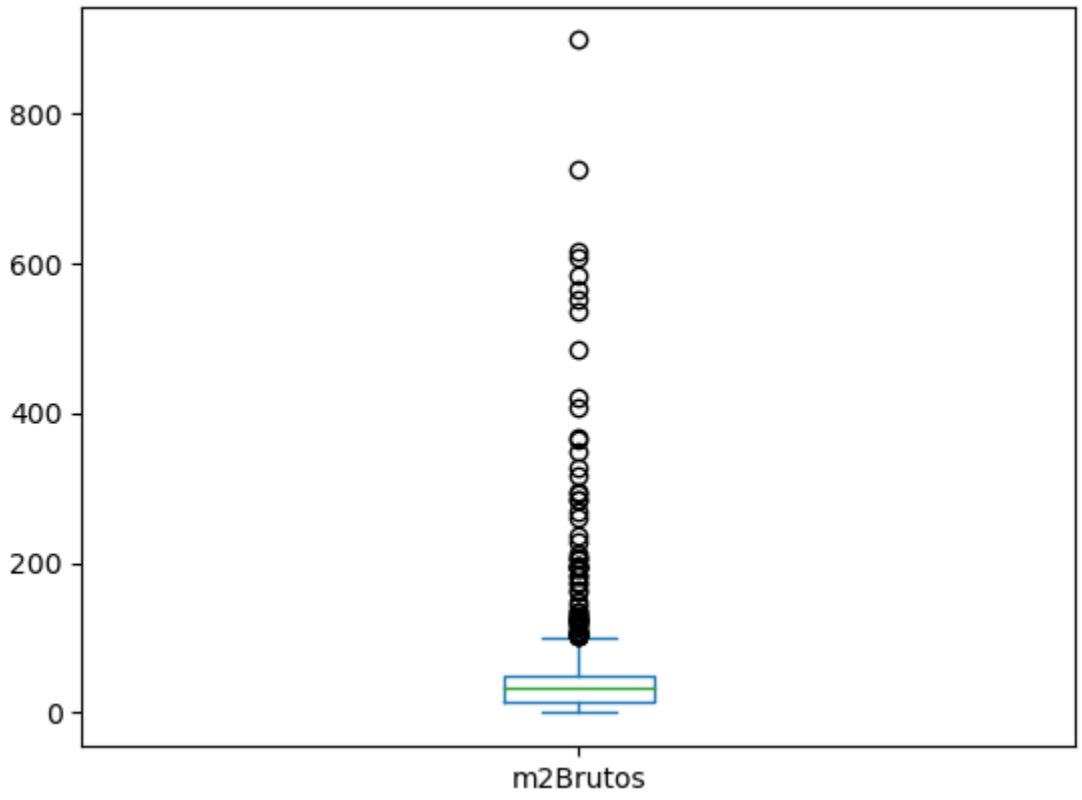
Las variables que presentan una correlación más alta con la variable son:

```
df[['precio','valorTerreno','m2Util','numServi', 'numHabita', 'numDormi','numChime']].corr()
```

	precio	valorTerreno	m2Util	numServi	numHabita	numDormi	numChime
precio	1.000000	0.581266	0.712390	0.597250	0.531170	0.400349	0.376786
valorTerreno	0.581266	1.000000	0.423441	0.297498	0.298865	0.202449	0.211727
m2Util	0.712390	0.423441	1.000000	0.718564	0.733666	0.656196	0.473788
numServi	0.597250	0.297498	0.718564	1.000000	0.517585	0.458033	0.436234
numHabita	0.531170	0.298865	0.733666	0.517585	1.000000	0.671863	0.319894
numDormi	0.400349	0.202449	0.656196	0.458033	0.671863	1.000000	0.284475
numChime	0.376786	0.211727	0.473788	0.436234	0.319894	0.284475	1.000000

• **m2Brutos** (metros cuadrados de la vivienda). Variable cuantitativa.

Se observan un rango y una distribución incoherentes en esta variable. El diagrama de box plot se muestra así.



Los valores son demasiado bajos.

```
df[df['m2Brutos']<50]['m2Brutos'].count()
```

```
1343
```

Un 78% de los datos presenta valores por debajo de los 50m2 brutos de superficie.

```
round(df[df['m2Brutos']<50]['m2Brutos'].count()*100/df.shape[0],2)
```

```
77.72
```

Y no guarda relación con m2Utiles.

```
df[['m2Brutos','m2Util']].corr()
```

	m2Brutos	m2Util
m2Brutos	1.000000	0.136701
m2Util	0.136701	1.000000

```
df[['m2Brutos','m2Util']].describe()
```

	m2Brutos	m2Util
count	1728.000000	1728.000000
mean	40.380787	1754.975694
std	58.035467	619.935553
min	0.000000	616.000000
25%	14.000000	1300.000000
50%	32.000000	1634.500000
75%	48.000000	2137.750000
max	897.000000	5228.000000

Tampoco tiene correlación con la variable objetivo.

```
df[['m2Brutos','precio']].corr()
```

	m2Brutos	precio
m2Brutos	1.000000	0.116191
precio	0.116191	1.000000

Ni aun tomando solo los datos que tienen un valor razonable, guarda relación con la variable objetivo.

```
df[df['m2Brutos']>30][['m2Brutos','precio']].corr()
```

	m2Brutos	precio
m2Brutos	1.000000	0.057729
precio	0.057729	1.000000

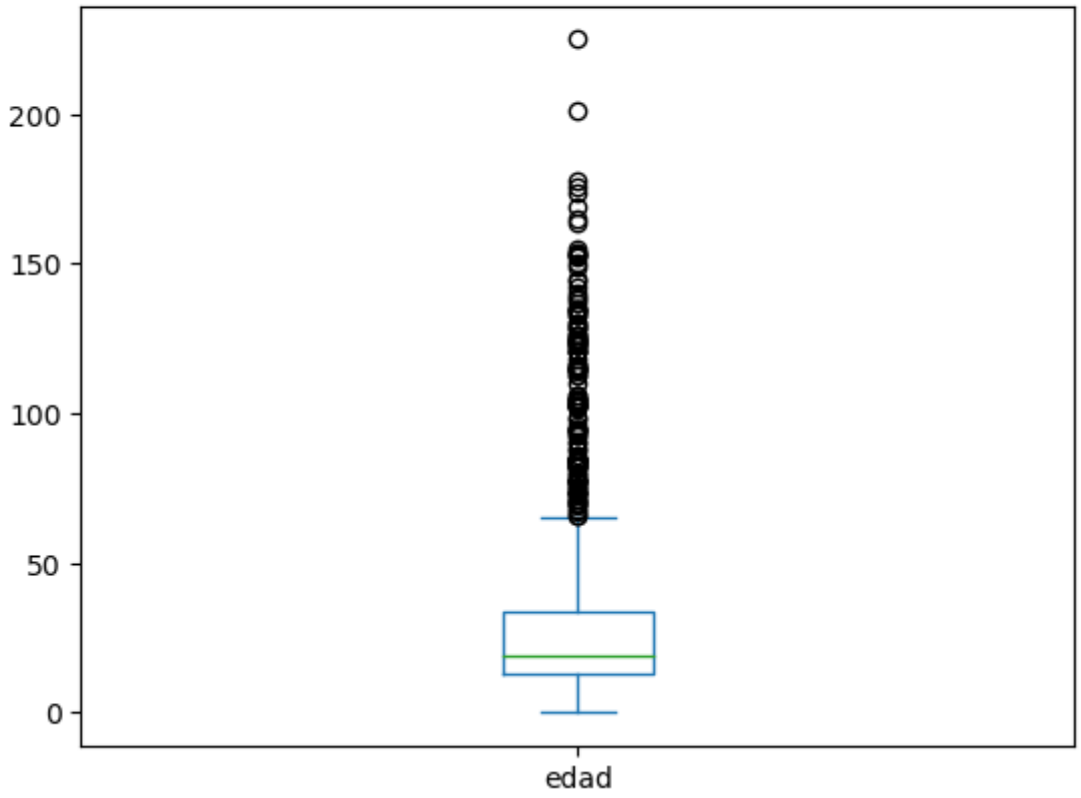
Por lo tanto, esta variable *queda excluida del modelo.*

• **edad** (antigüedad de la vivienda). Variable cuantitativa.

También presenta datos incorrectos y tiene baja correlación con precio (variable objetivo).

```
df[['edad']].plot.box()
```

```
<AxesSubplot:>
```



```
df[['edad','precio']].corr()
```

	edad	precio
edad	1.000000	-0.188793
precio	-0.188793	1.000000

Aun analizando la correlación en un rango de datos que podría considerarse normal para esta variable, tampoco se encuentra relación con la variable objetivo.

```
df[df['edad']<100][['edad','precio']].corr()
```

	edad	precio
edad	1.000000	-0.276479
precio	-0.276479	1.000000

Por lo tanto, también *se excluye esta variable del modelado*.

• **valorTerreno** (valor del terreno). Variable cuantitativa.

Esta es una variable importante puesto que presenta cierto grado de correlación con la variable objetivo.

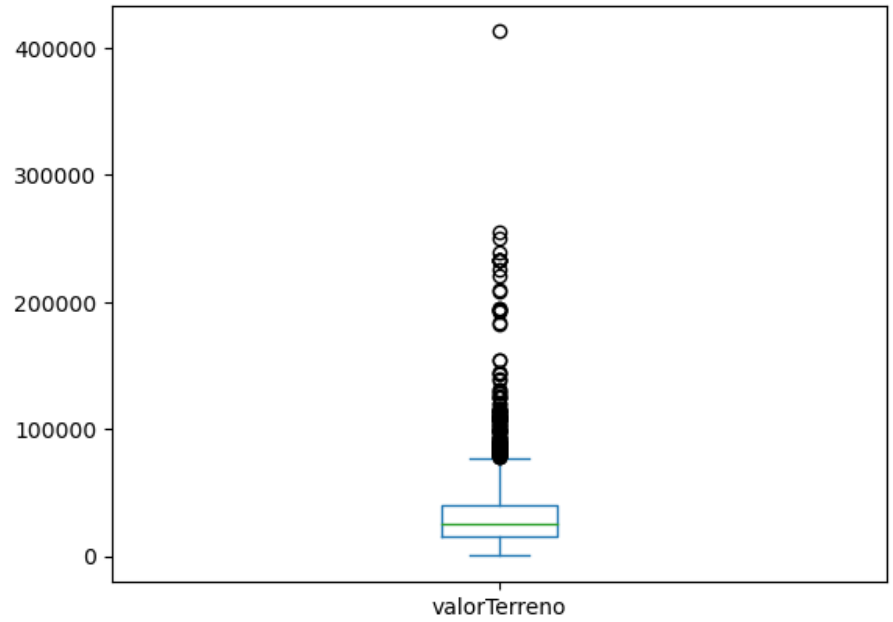
```
df[['valorTerreno','precio']].corr()
```

	valorTerreno	precio
valorTerreno	1.000000	0.581266
precio	0.581266	1.000000

También presenta gran cantidad de outliers en los valores altos.

```
df[['valorTerreno']].plot.box()
```

```
<AxesSubplot:>
```

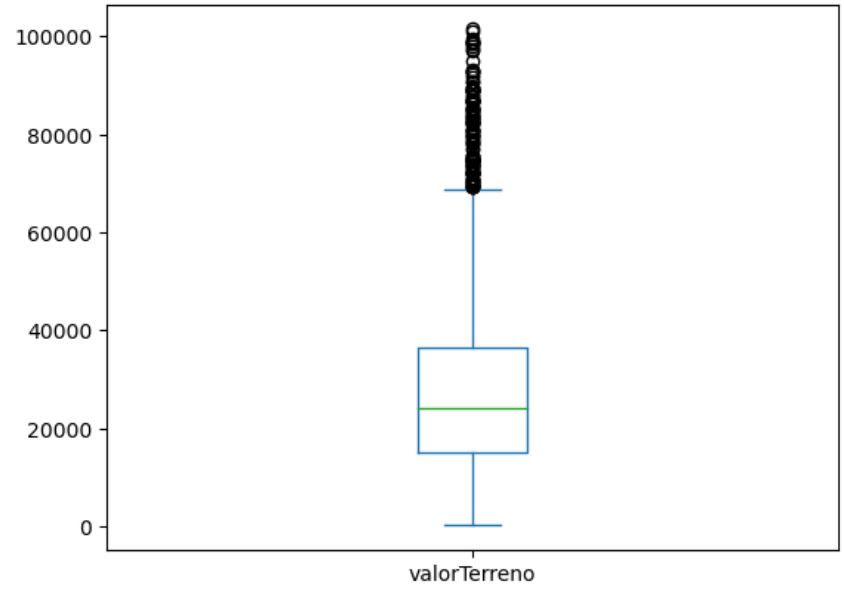


Analizando los ouliers, se decide aplicar un coeficiente de 2.5 al rango intercuartílico para fijar los valores fuera de rango, resultando así un 5% de outliers, lo que resulta en *92 registros que se eliminan del modelado*.

Analizando los casos que quedan después de eliminar estos casos (1636 reg), la gráfica se observa mejor.

```
df.iloc[list(set(df.index.values).difference(set(ind_Atipicos_ValorT)))]['valorTerreno'].plot.box()
```

```
<AxesSubplot:>
```



Sin embargo, la correlación con la variable objetivo baja.

```
df.iloc[list(set(df.index.values).difference(set(ind_Atipicos_ValorT)))][['valorTerreno','precio']].corr()
```

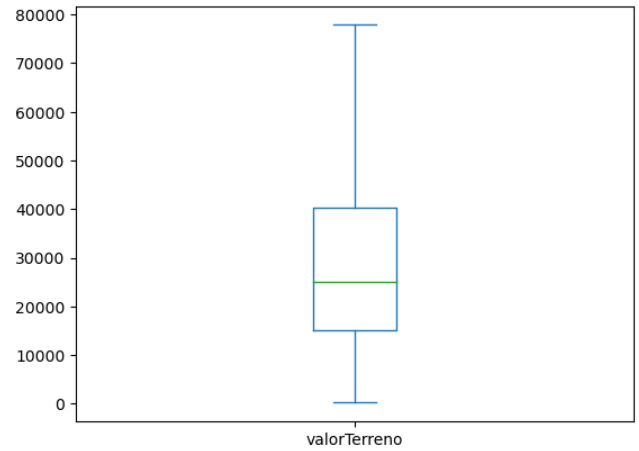
	valorTerreno	precio
valorTerreno	1.000000	0.497501
precio	0.497501	1.000000

Por lo tanto, se intenta otra estrategia y se recuperan/reemplazan los 166 outliers (9.61%) por el valor más alto del rango establecido al aplicar un coeficiente de 1.5 al rango intercuartílico y se vuelve a analizar.

La variable se observa más normalizada en la gráfica de box&whiskers.

```
df_aux[['valorTerreno']].plot.box()
```

```
<AxesSubplot:>
```



La correlación con la variable objetivo se ha corregido.

```
df_aux[['valorTerreno','precio']].corr()
```

	valorTerreno	precio
valorTerreno	1.000000	0.566488
precio	0.566488	1.000000

Por tanto, *la variable recuperada de este modo, será considerada para el modelado*.

• **m2Utiles** (metros cuadrados habitables). Variable cuantitativa.

Esta variable tiene una correlación alta (0.71) con la variable objetivo por lo que resulta importante para el modelado.

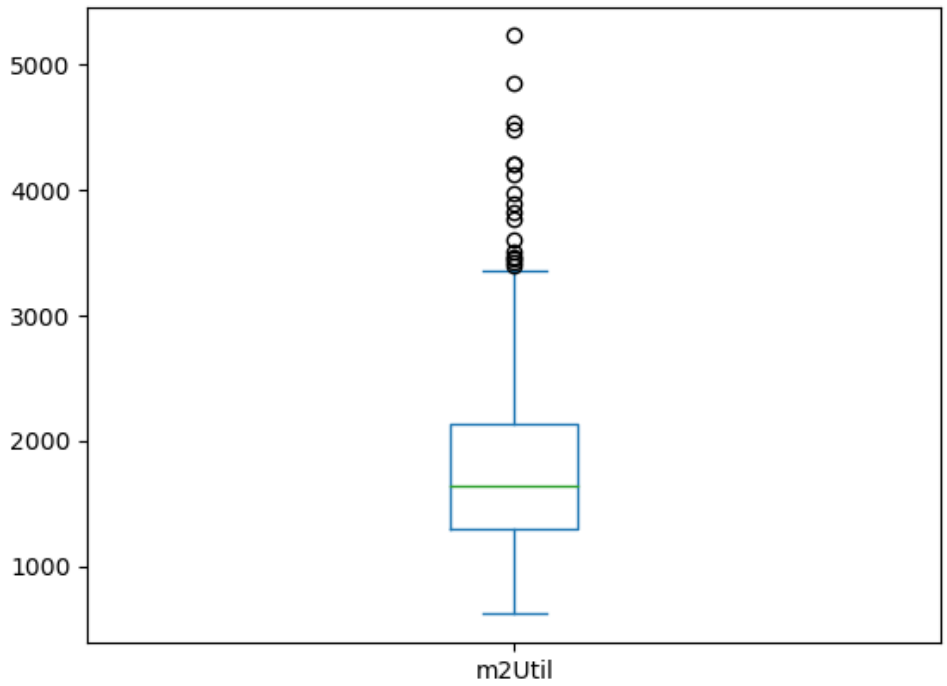
```
df_aux[['m2Util','precio']].corr()
```

	m2Util	precio
m2Util	1.00000	0.71239
precio	0.71239	1.00000

También presenta algunos outliers en los valores altos.

```
df_aux[['m2Util']].plot.box()
```

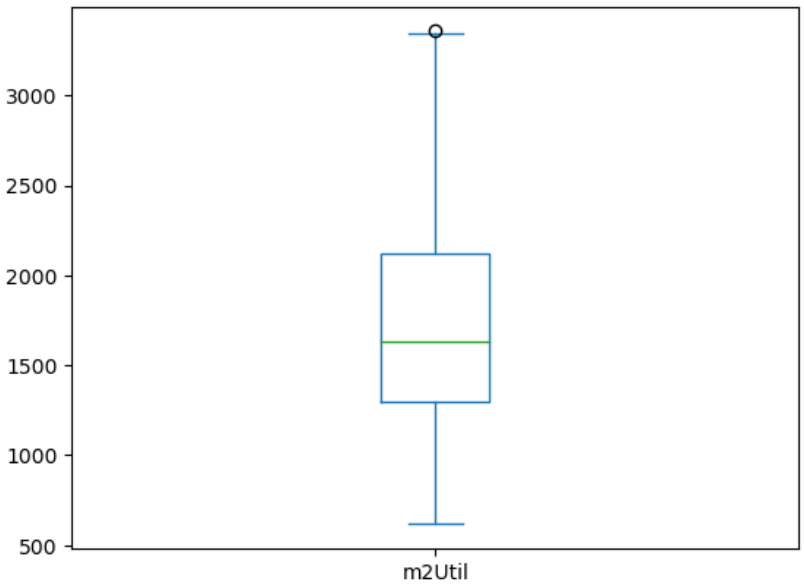
```
<AxesSubplot:>
```



Son 17 registros (1%). Quitados estos la gráfica box se normaliza.

```
df_aux.iloc[list(set(df_aux.index.values).difference(set(ind_Atipicos_ValorU)))]['m2Util'].plot.box()
```

```
<AxesSubplot:>
```



La correlación con la variable objetivo baja un poco, pero sigue siendo importante.

```
df_aux.iloc[list(set(df_aux.index.values).\
difference(set(ind_Atipicos_ValorU)))][['m2Util','precio']].corr()
```

	m2Util	precio
m2Util	1.000000	0.684993
precio	0.684993	1.000000

Se decide prescindir de los outliers (17 casos) para que no tengan influencia en el modelado y se incluye la variable en el modelado.

• **perUni** (porcentaje del vecindario con título universitario). Variable cuantitativa.

La variable presenta una baja correlación con la variable objetivo (0.20).

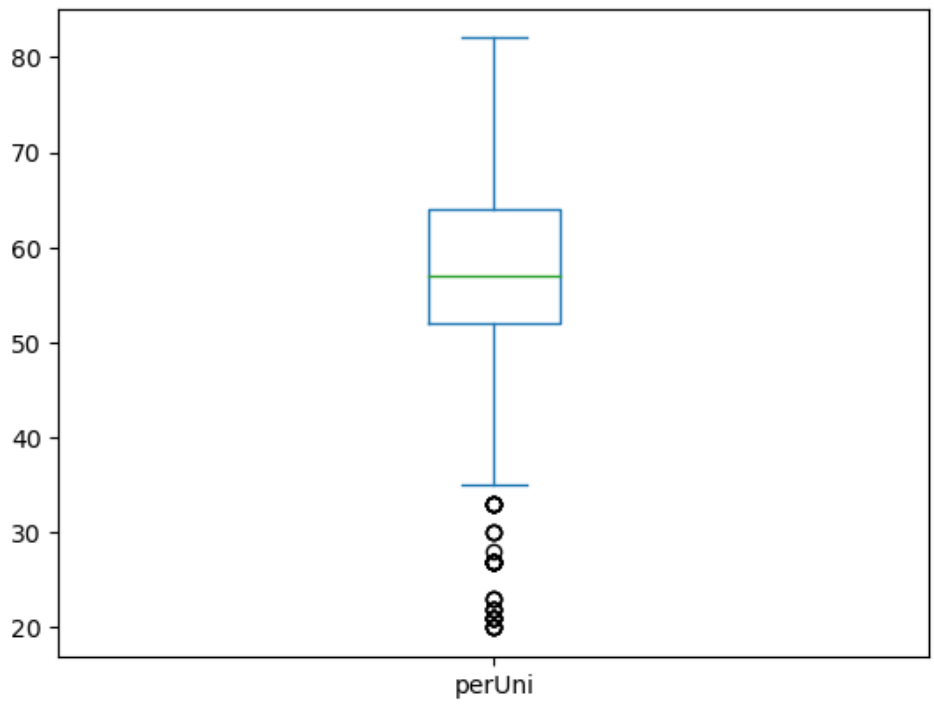
```
df_aux[['perUni','precio']].corr()
```

	perUni	precio
perUni	1.000000	0.199075
precio	0.199075	1.000000

Se analiza la distribución.

```
df_aux[['perUni']].plot.box()
```

```
<AxesSubplot:>
```

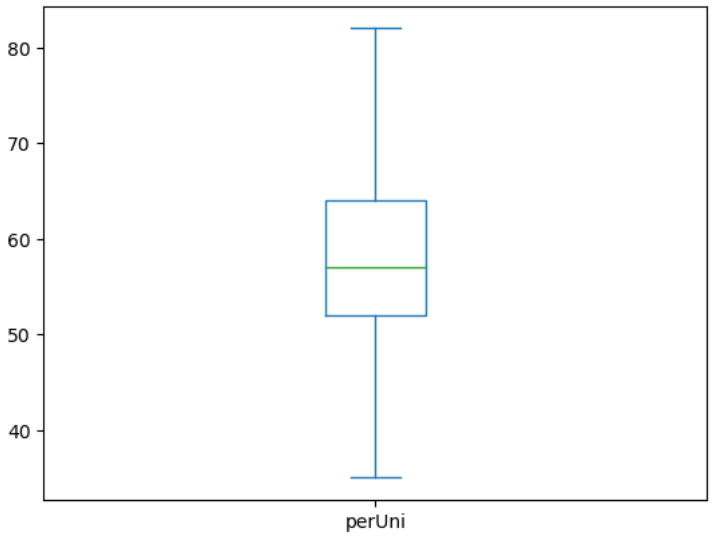


Se analiza la correlación sin los outliers que se observan para determinar si la correlación mejora.

Gráfica de box quitados los outliers.

```
df_aux.loc[list(set(df_aux.index.values).difference(set(ind_Atipicos_ValorPU)))]['perUni'].plot.box()
```

```
<AxesSubplot:>
```




```
df_aux.loc[list(set(df_aux.index.values).\
difference(set(ind_Atipicos_ValorU)))][['perUni','precio']].corr()
```

	perUni	precio
perUni	1.000000	0.199075
precio	0.199075	1.000000

La eliminación de atípicos no mejora la correlación con la variable objetivo por lo que *se elimina esta variable del modelado.*

• **numDormi** (número de dormitorios). Variable cuantitativa discreta.

La correlación con precio es débil.

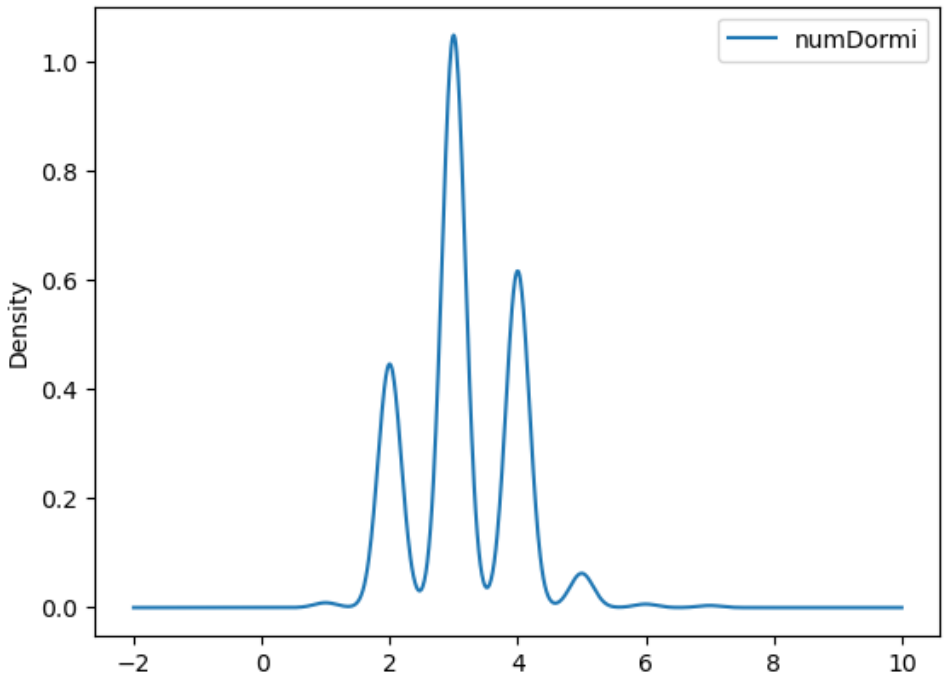
```
df_aux[['numDormi','precio']].corr()
```

	numDormi	precio
numDormi	1.000000	0.388102
precio	0.388102	1.000000

Se analizan la distribución y frecuencia.

```
df_aux[['numDormi']].plot.kde()
```

```
<AxesSubplot:ylabel='Density'>
```



Tiene tres modas. Se analiza la frecuencia:

```
df_aux[['numDormi', 'precio']].groupby(by='numDormi', sort='numDormi').count()
```

	precio
numDormi	
1	7
2	348
3	818
4	481
5	49
6	5
7	3

Se consideran los registros con numDormi entre 2 y 5 dormitorios, tomando como outliers las de 1,6 y 7.

Son 15 reg representan 0.9%. Se analiza si quitando estos casos extremos mejora la correlación con la variable objetivo.

```
df_aux[(df_aux['numDormi']>=2) & (df_aux['numDormi']<=5) ][['numDormi', 'precio']].corr()
```

	numDormi	precio
numDormi	1.000000	0.406777
precio	0.406777	1.000000

Algo mejora, pero no es suficiente. La correlación sigue siendo débil.

De todos modos, se convertirá la variable en categórica, y se incluirá como dummy por si pudiera aportar al modelo.

Datos originales en la variable:

```
df_aux[['numDormi', 'precio']].groupby(by='numDormi', sort='numDormi').count()
```

	precio
numDormi	
1	7
2	348
3	817
4	478
5	49
6	5
7	3

Variable convertida en dummies. Dos de ellas se incluirán en el modelo (la tercera es redundante).

```
df_aux[['numDormi','DormiHasta2','Dormi3','DormiDesde4']].sample(5)
```

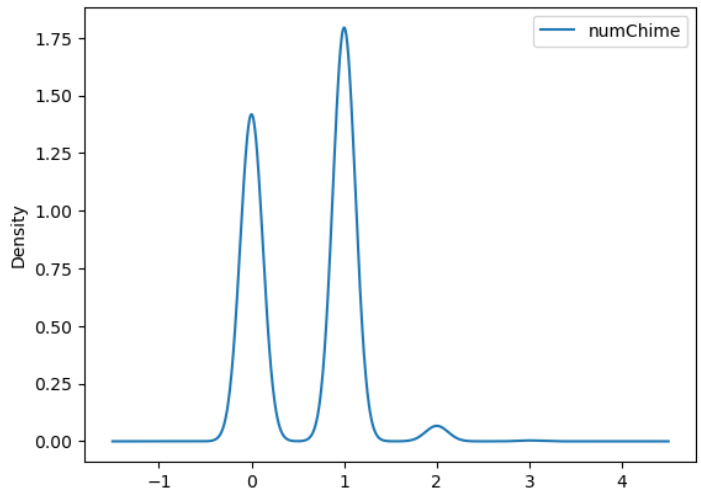
	numDormi	DormiHasta2	Dormi3	DormiDesde4
618	3	0	1	0
993	3	0	1	0
644	3	0	1	0
1550	2	1	0	0
161	4	0	0	1

Las variables dummy Dormi3 y DormiDesde4 se incorporarán al modelo.

• **numChime** (número de chimeneas). Variable cuantitativa discreta.

```
df_aux[['numChime']].plot.kde()
```

```
<AxesSubplot:ylabel='Density'>
```



Tiene dos modas que representan si el inmueble tiene o no tiene chimenea.

```
df_aux[['numChime', 'precio']].groupby(by='numChime', sort='numChime').count()
```

	precio
numChime	
0	739
1	935
2	35
3	2

Se convierte la variable en booleana.

```
df_aux[['numChime', 'precio']].groupby(by='numChime', sort='numChime').count()
```

	precio
numChime	
0	739
1	972

Convertida la variable en una nominal binomial, se evalúa la correlación mediante t-test o t-student.

```
ttest_ind(df_aux[df_aux['numChime']==0]['precio'], df_aux[df_aux['numChime']==1]['precio'])
```

```
Ttest_indResult(statistic=-14.19503869180475, pvalue=2.6013330109202634e-43)
```

El p-valor es bajo por lo que se acepta la hipótesis de que las variables están relacionadas y *se incluye la variable en el modelado*.

• **numServi** (número de cuartos de baño, el valor 0.5 hace referencia a cuartos de baño sin ducha). Variable cuantitativa discreta.

La variable guarda correlación con precio, la variable objetivo.

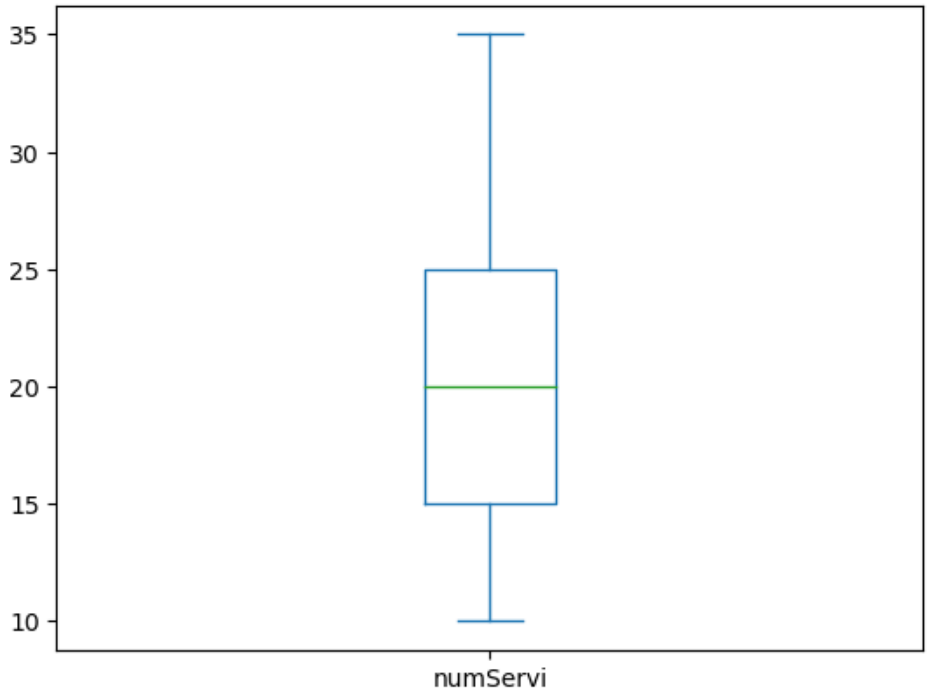
```
df_aux[['numServi','precio']].corr()
```

	numServi	precio
numServi	1.000000	0.579038
precio	0.579038	1.000000

Se analiza su distribución y frecuencia.

```
df_aux[['numServi']].plot.box()
```

```
<AxesSubplot:>
```



```
df_aux[['numServi', 'precio']].groupby(by='numServi', sort='numServi').count()
```

numServi	precio
0	1
10	325
15	497
20	255
25	544
30	55
35	31
40	3

Si bien no cuenta con outliers, se quitan los dos extremos para mayor limpieza de la variable. La correlación queda así:

```
df_aux[['numServi','precio']].corr()
```

	numServi	precio
numServi	1.000000	0.577936
precio	0.577936	1.000000

La variable numServi será considerada en el modelado.

• **numHabita** (número de habitaciones). Variable cuantitativa discreta.

La variable guarda correlación media con la variable objetivo (0.5).

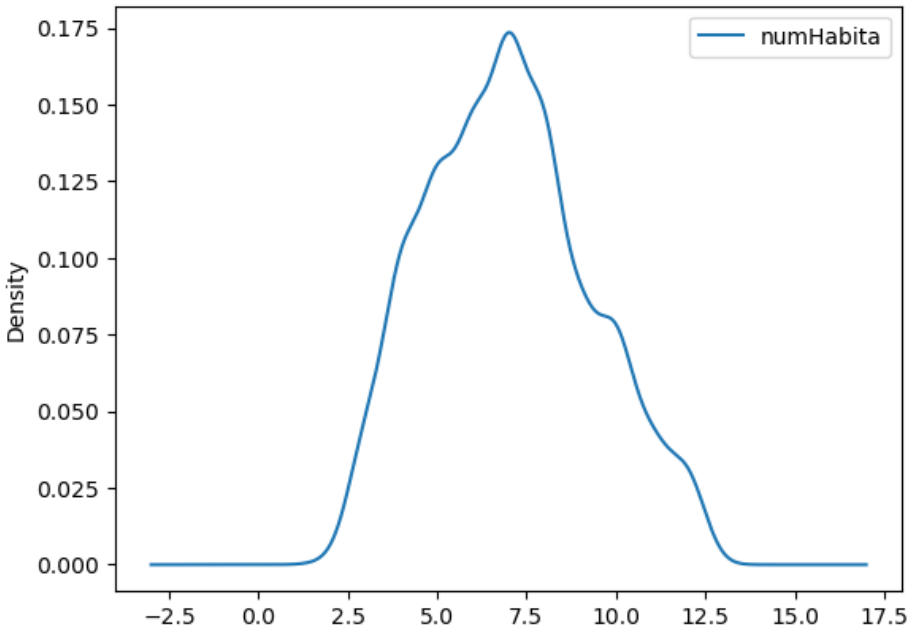
```
df_aux[['numHabita','precio']].corr()
```

	numHabita	precio
numHabita	1.000000	0.503682
precio	0.503682	1.000000

Se analiza su distribución.

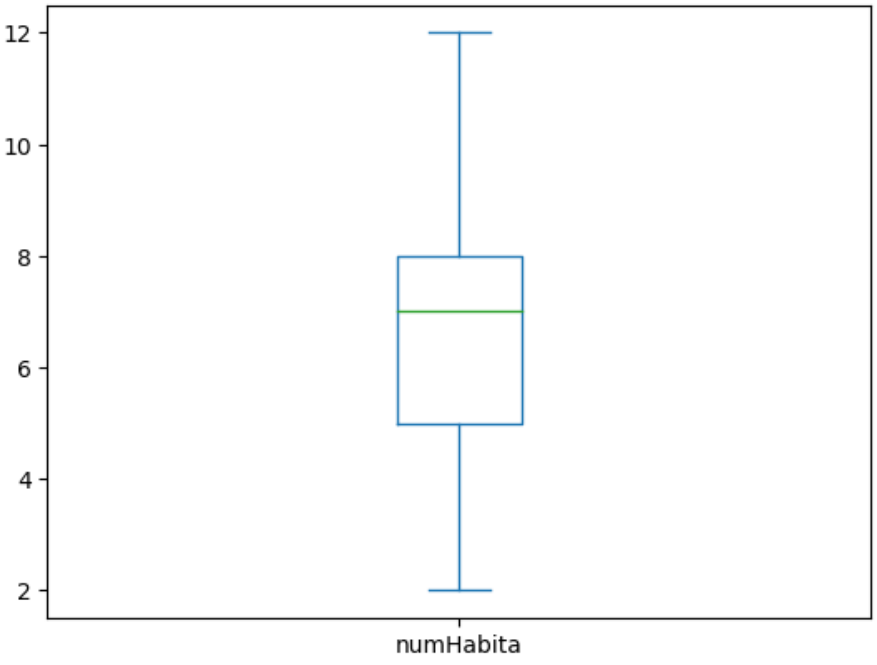
```
df_aux[['numHabita']].plot.kde()
```

```
<AxesSubplot:ylabel='Density'>
```




```
df_aux[['numHabita']].plot.box()
```

```
<AxesSubplot:>
```



No presenta outliers.

Se considerará la variable para el modelado.

• **calefacción** (tipo de calefacción). Variable cualitativa nominal.

```
df_aux['calefaccion'].value_counts()
```

```
aerotermia        1106
Electrica          304
suelo radiante     297
Name: calefaccion, dtype: int64
```

La variable no presenta valores nulos.

Se trata de una variable trinomial.

Analizada la correlatividad de las variables cuantitativas, para cada una de las categorías de esta variable, se observan variaciones, por lo que *se decide convertirla a dummies y se incorporan al modelo.*

• **alimentación**: tipo de alimentación de la calefacción (gas, electricidad o diésel).

Variable cualitativa nominal.

```
df_aux['alimentacion'].value_counts()
```

```
gas          1179
Electrica     314
Gasoil        214
Name: alimentacion, dtype: int64
```

No presenta valores nulos.

Se trata de una variable trinomial.

Las distintas categorías de esta variable producen cambios en la correlación de las cuantitativas, *por lo que se decide dummificar e incorporar al modelo.*

• **tipoDesague**: tipo de desagüe.

Variable cualitativa nominal.

```
df_aux['tipoDesague'].value_counts()
```

```
comunitario     1200
fosa septica     495
none              12
Name: tipoDesague, dtype: int64
```

Presenta 12 registros nulos.

Se analiza mediante un test de student la correlación con la variable objetivo.

```
ttest_ind(df[df['tipoDesague']=='comunitario']['precio'], df[df['tipoDesague']=='fosa septica']['precio'])
```

```
Ttest_indResult(statistic=3.1057310474861106, pvalue=0.0019291857207055152)
```

El resultado da un p-valor de 0.0019. Es muy bajo por tanto concluimos que las variables tienen dependencia.

Los valores faltantes representan menos del 1% (0.7) del total por lo que se eliminarán del modelado.

Se incorpora la variable convertida a dummy al modelo.

• **conVistas**: si la vivienda tiene vistas o no.

Variable cualitativa binomial.

```
df_aux['conVistas'].value_counts()
```

```
No    1681
Sí      14
Name: conVistas, dtype: int64
```

La incidencia de esta variable no será determinante puesto su poca variabilidad. De todos modos, se evalúa la correlación con la variable objetivo mediante un t-test.

El resultado de p-valor es muy bajo con lo que se asume que las variables son dependientes.

```
# t-test
ttest_ind(df_aux[df_aux['conVistas']=='No']['precio'], df_aux[df_aux['conVistas']=='Sí']['precio'])
```

```
Ttest_indResult(statistic=-6.396278144912916, pvalue=2.0546916028774506e-10)
```

Se convierte a booleana la variable y *se incorpora al modelo.* Las sucesivas iteraciones del modelo eliminarán la variable si no aporta a la predicción.

• **construcción**: si la vivienda es de nueva construcción.

Variable cualitativa binomial.

```
df_aux['construccion'].value_counts()
```

```
No    1616
Sí      79
Name: construccion, dtype: int64
```

Se evalúa la correlación con la variable objetivo mediante un t-test. El resultado de p-valor es muy bajo con lo que se asume que las variables son dependientes.

```
ttest_ind(df_aux[df_aux['construccion']=='No']['precio'], df_aux[df_aux['construccion']=='Sí']['precio'])
```

```
Ttest_indResult(statistic=-7.127872688685746, pvalue=1.5030667202629033e-12)
```

Se convierte a booleana la variable y *se incorpora al modelo.*

• **aire**: si la vivienda tiene aire acondicionado.

Variable cualitativa binomial.

```
df_aux['aire'].value_counts()
```

```
No    1081
Sí     614
Name: aire, dtype: int64
```

Se evalúa la correlación con la variable objetivo mediante un t-test. El resultado de p-valor es muy bajo con lo que se asume que las variables son dependientes.

```
ttest_ind(df_aux[df_aux['aire']=='No']['precio'], df_aux[df_aux['aire']=='Sí']['precio'])
```

```
Ttest_indResult(statistic=-13.818226556483944, pvalue=3.130172621978504e-41)
```

Se convierte a booleana la variable y *se incorpora al modelo.*

Recuento de las variables candidatas como predictoras

Resultante del EDA, se seleccionan las siguientes variables como predictoras:

- Cuantitativas: valorTerreno, m2Util, numServi, numHabita.

  Estas variables tienen algún grado de correlación (medio, medio/alto) con la variable objetivo y baja correlación entre sí.

- Categóricas/dummies (cualitativas nominales y booleanas): Dormi3, DormiDesde4, numChime , calef_elec , calef_aero, alim_gas, alim_elec, des_comuni, conVistas, construcción, aire.

  Las variables binomiales numChime, tipoDesague (dummy des_comuni), conVistas, construcción y aire mostraron tener correlación con la variable objetivo mediante t-test.

  También se incluyen las variables multinomiales numDormi, calefacción y alimentación que se dumificaron para el modelo.

Tratamiento de los datos

Entre el tratamiento realizado a los datos están la dummificación de las variables multinomiales citadas, de las binomiales y la eliminación de NaNs en tipo_Desague.

También el tratamiento de outliers de las variables cuantitativas analizadas en cada caso.

Por último, se deben tipificar a numéricas (int64) todas las variables que tengan otra tipología.

Modelado 1 (RLM con datos outliers tratados)

Dada la naturaleza de la variable objetivo (cuantitativa continua) y el número de variables predictoras, se opta por un modelo de RLM (regresión lineal múltiple) que incorporará en el primer modelado todas las variables seleccionadas y progresivamente se irán eliminando las que no aporten a la predicción.

Se fija el nivel de significancia en 0.05 (p-valor).

El set de datos cuenta con casi 1700 registros por lo que la división de datos para training y test se realiza al 70/30% (1186 y 509 respectivamente).

Mediante la evaluación OLS (mínimos cuadrados ordinarios) se eliminan progresivamente las variables que resulten con el p-valor más alto, y volvemos a modelar con la variable eliminada en cada caso, hasta que ninguna resulte con p-valor>0.05.

Las eliminaciones sucesivas fueron:

- 1. (x13 con p-valor=0,964) alim_elec
- 2. (x3 con p-valor=0,669) numChime
- 3. (x12 con p-valor=0,256) alim_gas
- 4. (x10 con p-valor=0,407) calef_elec
- 5. (x8 con p-valor=0,174) Dormi3

Se eliminan 5 de las variables originales. Quedan 10 variables predictoras, a saber,

'valorTerreno', 'm2Util', 'numServi', 'numHabita', 'conVistas', 'construccion', 'aire', 'DormiDesde4', 'calef_aero', 'des_comuni'.

Evaluación del Modelo 1

El R2 (r-cuadrado) del modelo aplicado sobre el lote de entrenamiento es 0.62.

El R2 (r-cuadrado) del modelo aplicado sobre el lote de testing es 0.59.

El porcentaje de efectividad del modelo es bajo.

Modelado 2 (RLM con datos sin tratar)

El tratamiento/eliminación de los outliers presentes en los datos, no ha redundado en una mejora en el modelado. Por lo que se intenta un nuevo modelado de RLM con las variables que presentan mejor correlación, sin tratar los outliers.

```
df[['precio','valorTerreno','m2Util','numServi', 'numHabita', 'numDormi','numChime']].corr()
```

	precio	valorTerreno	m2Util	numServi	numHabita	numDormi	numChime
precio	1.000000	0.581266	0.712390	0.597250	0.531170	0.400349	0.376786
valorTerreno	0.581266	1.000000	0.423441	0.297498	0.298865	0.202449	0.211727
m2Util	0.712390	0.423441	1.000000	0.718564	0.733666	0.656196	0.473788
numServi	0.597250	0.297498	0.718564	1.000000	0.517585	0.458033	0.436234
numHabita	0.531170	0.298865	0.733666	0.517585	1.000000	0.671863	0.319894
numDormi	0.400349	0.202449	0.656196	0.458033	0.671863	1.000000	0.284475
numChime	0.376786	0.211727	0.473788	0.436234	0.319894	0.284475	1.000000

Se consideran las variables que correlacionan mejor con la variable objetivo:

'valorTerreno', 'm2Util', 'numServi', 'numHabita', 'numDormi', 'numChime'.

Se elimina la variable 'numChime', con p-valor = 0.167.

En resto de variables presenta un p-valor < 0.05 (significancia fijada).

Evaluación del Modelo 2

El R2 (r-cuadrado) del modelo 2 aplicado sobre el lote de entrenamiento es 0.64.

El R2 (r-cuadrado) del modelo 2 aplicado sobre el lote de testing es 0.59.

Misma efectividad que el modelo 1.

Modelado 3 (SVR)

Para el modelado se utiliza el lote de datos con las transformaciones de outliers/missings y se consideran dos casos:

CASO 1: Se escogen para el modelado las cuatro variables que mejor correlacionan con la variable objetivo: 'm2Util','valorTerreno','numServi','numHabita'.

CASO 2: Se escogen para el modelado las diez variables resultantes de la iteración OLS (mínimos cuadrados ordinarios): 'valorTerreno', 'm2Util', 'numServi', 'numHabita', 'conVistas', 'construccion', 'aire', 'DormiDesde4', 'calef_aero', 'des_comuni'.

Previa normalización de los datos (requerido para el modelo SVR), se prueban los distintos kernel del modelo SVR.

- Linear CASO 1

```
reg_svr_klineal=SVR(kernel='linear')
reg_svr_klineal.fit(x4_train,y4_train.ravel())
```

```
SVR(kernel='linear')
```

```
y_svr_klineal=reg_svr_klineal.predict(sc_x.fit_transform( x4_test ) )
sc_y.inverse_transform(  [y_svr_klineal]  )
```

```
array([[128606.3413085 ,  97200.8163758 , 140586.53429094,
        320711.13506757, 194126.27238268, 234824.09849815,
        131213.28812143, 189764.74578377, 160434.51240928,
        157097.29221096, 171815.86097398, 137370.74853727,
        138435.65083472, 251657.64419174, 126613.45297236,
        118339.56088105, 165427.91892097, 380945.48226269,
        163485.61975346, 312380.82202359, 211304.22975415,
```

```
r2_klineal=r2_score(y4_test, sc_y.inverse_transform(  [y_svr_klineal]  ).reshape(509,1) )
mae_klineal=mean_absolute_error(y4_test, sc_y.inverse_transform(  [y_svr_klineal]  ).reshape(509,1))
mse_klineal=mean_squared_error(y4_test, sc_y.inverse_transform(  [y_svr_klineal]  ).reshape(509,1))
print(f'Estadisticos de la SVR con kernel lineal\n r2: {r2_klineal:.2f} \n mae: {mae_klineal:.2f} \n mse: {mse_klineal:.2f}')
```

```
Estadisticos de la SVR con kernel lineal
 r2: 0.54
 mae: 43541.42
 mse: 4056493690.01
```

Resultado r2: 0.54

- Linear CASO 2

```
r2_klineal=r2_score(y4_test, sc_y.inverse_transform(  [y_svr_klineal]  ).reshape(509,1) )
mae_klineal=mean_absolute_error(y4_test, sc_y.inverse_transform(  [y_svr_klineal]  ).reshape(509,1))
mse_klineal=mean_squared_error(y4_test, sc_y.inverse_transform(  [y_svr_klineal]  ).reshape(509,1))
print(f'Estadisticos de la SVR con kernel lineal\n r2: {r2_klineal:.2f} \n mae: {mae_klineal:.2f} \n mse: {mse_klineal:.2f}')
```

```
Estadisticos de la SVR con kernel lineal
 r2: 0.57
 mae: 42524.57
 mse: 3736309424.54
```

Resultado r2: 0.57

- Polinómico CASO 1

```
r2_kpoly=r2_score(y4_test, sc_y.inverse_transform(  [y_svr_kpoly]  ).reshape(509,1) )
mae_kpoly=mean_absolute_error(y4_test, sc_y.inverse_transform(  [y_svr_kpoly]  ).reshape(509,1)  )
mse_kpoly=mean_squared_error(y4_test, sc_y.inverse_transform(  [y_svr_kpoly]  ).reshape(509,1)   )
print(f'Estadisticos de la SVR con kernel polinomico\n r2: {r2_kpoly:.2f} \n mae: {mae_kpoly:.2f} \n mse: {mse_kpoly:.2f}')
```

```
Estadisticos de la SVR con kernel polinomico
 r2: 0.46
 mae: 48327.49
 mse: 4687780814.87
```

Resultado r2: 0.46

- Polinómico CASO 2

```
r2_kpoly=r2_score(y4_test, sc_y.inverse_transform(  [y_svr_kpoly]  ).reshape(509,1) )
mae_kpoly=mean_absolute_error(y4_test, sc_y.inverse_transform(  [y_svr_kpoly]  ).reshape(509,1)  )
mse_kpoly=mean_squared_error(y4_test, sc_y.inverse_transform(  [y_svr_kpoly]  ).reshape(509,1)   )
print(f'Estadisticos de la SVR con kernel polinomico\n r2: {r2_kpoly:.2f} \n mae: {mae_kpoly:.2f} \n mse: {mse_kpoly:.2f}')
```

```
Estadisticos de la SVR con kernel polinomico
 r2: 0.57
 mae: 42748.10
 mse: 3729324352.33
```

Resultado r2: 0.57

- Radial CASO 1

```
reg_svr_kgauss=SVR(kernel='rbf')  #
reg_svr_kgauss.fit(x4_train,y4_train.ravel())
```

```
▾ SVR
SVR()
```

```
y_svr_kgauss=reg_svr_kgauss.predict(sc_x.fit_transform(x4_test))
sc_y.inverse_transform( [y_svr_kgauss] )
```

```
        140819.96579763, 193046.37162153, 143925.05701323,
        139631.47256287, 209337.937061  , 132147.51168032,
        114023.95117215, 150785.97460806, 207772.54983631,
        162665.8810918 , 335730.29429643, 284341.44959607,
        170060.46417174, 174346.69152146, 119942.00658358,
        124567.99253186, 110985.41833504, 274226.67479798,
```

```
r2_kgauss=r2_score(y4_test, sc_y.inverse_transform(  [y_svr_kgauss]  ).reshape(509,1)  )
mae_kgauss=mean_absolute_error(y4_test, sc_y.inverse_transform(  [y_svr_kgauss]  ).reshape(509,1))
mse_kgauss=mean_squared_error(y4_test, sc_y.inverse_transform(  [y_svr_kgauss]  ).reshape(509,1))
print(f'Estadisticos de la SVR con kernel radial\n r2: {r2_kgauss:.2f} \n mae: {mae_kgauss:.2f} \n mse: {mse_kgauss:.2f}')
```

```
Estadisticos de la SVR con kernel radial
 r2: 0.54
 mae: 43672.20
 mse: 4022182778.11
```

Resultado r2: 0.54

- Radial CASO 2

```
r2_kgauss=r2_score(y4_test, sc_y.inverse_transform(  [y_svr_kgauss]  ).reshape(509,1)  )
mae_kgauss=mean_absolute_error(y4_test, sc_y.inverse_transform(  [y_svr_kgauss]  ).reshape(509,1))
mse_kgauss=mean_squared_error(y4_test, sc_y.inverse_transform(  [y_svr_kgauss]  ).reshape(509,1))
print(f'Estadisticos de la SVR con kernel radial\n r2: {r2_kgauss:.2f} \n mae: {mae_kgauss:.2f} \n mse: {mse_kgauss:.2f}')
```

```
Estadisticos de la SVR con kernel radial
 r2: 0.59
 mae: 41318.00
 mse: 3603536127.10
```

Resultado r2: 0.59

Con SVR no se obtiene una mejor predicción. Se iguala a RLM.

Modelado 4 (Árbol de decisión)

Con los mismos datos (caso 1 con cuatro variables que mejor correlacionan y caso 2 con las diez variables de la selección OLS) se prueban diferentes variantes del modelo regresor por árbol de decisión, previa normalización de los datos (requerido para este modelo).

El mejor resultado se obtiene con criterion squared_error o friedman_mse.

CASO 1:

```
reg_arbol=DecisionTreeRegressor(criterion='squared_error', min_samples_split=15, max_leaf_nodes=200, random_state=1987)
reg_arbol.fit(x5_train,y5_train)
```

```
DecisionTreeRegressor(max_leaf_nodes=200, min_samples_split=15,
                      random_state=1987)
```

```
y_arbol=reg_arbol.predict(sc_x.fit_transform(x5_test))
sc_y.inverse_transform([y_arbol])
```

```
array([[ 97286.66666667, 141709.71428571, 174277.77777778,
        442436.27272727, 131000.        , 206950.        ,
        165483.33333333, 202246.78571429, 134243.5       ,
        165483.33333333, 134066.66666667, 176753.        ,
        192666.66666667, 320447.72727273, 174277.77777778,
        146466.66666667, 180755.55555556, 491861.5       ,
```

```
r2_arbol=r2_score(y5_test, sc_y.inverse_transform([y_arbol]).reshape(509,1) )
mae_arbol=mean_absolute_error(y5_test, sc_y.inverse_transform([y_arbol]).reshape(509,1))
mse_arbol=mean_squared_error(y5_test, sc_y.inverse_transform([y_arbol]).reshape(509,1))
print(f'Estadisticos de la regresion por Arbol de Desicion\n r2: {r2_arbol:.2f} \n mae: {
```

```
Estadisticos de la regresion por Arbol de Desicion
 r2: 0.41
 mae: 50776.03
 mse: 5196328790.50
```

Resultado R2: 0.41

CASO 2:

```
r2_arbol=r2_score(y5_test, sc_y.inverse_transform([y_arbol]).reshape(509,1) )
mae_arbol=mean_absolute_error(y5_test, sc_y.inverse_transform([y_arbol]).reshape(509,1))
mse_arbol=mean_squared_error(y5_test, sc_y.inverse_transform([y_arbol]).reshape(509,1))
print(f'Estadisticos de la regresion por Arbol de Desicion\n r2: {r2_arbol:.2f} \n mae: {mae_arbol:.2f} \n mse: {mse_arbol:.2f}')
```

```
Estadisticos de la regresion por Arbol de Desicion
 r2: 0.42
 mae: 49333.80
 mse: 5080109298.68
```

Resultado R2: 0.42

Tampoco resulta una mejora en la predicción.

Modelado 5 (Bosques Aleatorios)

Se evalúan los mismos dos casos con el modelo un random forest, con distintos parámetros (criterion, n_estimators), resultando el de mejor R2, el criterion=absolute_error con n_estimators=500.

CASO 1:

```
reg_randomForest=RandomForestRegressor(n_estimators=500, criterion='absolute_error', max_features='auto', random_state=1987)
reg_randomForest.fit(x6_train,y6_train) # ya estan normalizadas
```

```
y_randomForest=reg_randomForest.predict(sc_x.fit_transform(x6_test))
sc_y.inverse_transform([y_randomForest])
```

```
array([[123537.202, 129560.52 , 150791.5  , 356930.13 , 202630.36 ,
        226901.676, 166666.868, 158660.876, 152368.499, 187762.58 ,
```

```
r2_arbol=r2_score(y6_test, sc_y.inverse_transform([y_randomForest]).reshape(509,1) )
mae_arbol=mean_absolute_error(y5_test, sc_y.inverse_transform([y_randomForest]).reshape(509,1))
mse_arbol=mean_squared_error(y5_test, sc_y.inverse_transform([y_randomForest]).reshape(509,1))
print(f'Estadisticos de la regresion por Bosques Aleatorios\n r2: {r2_arbol:.2f} \n mae: {mae_arbol:.2f} \n mse: {mse_arbol:.2f}'
```

```
Estadisticos de la regresion por Bosques Aleatorios
 r2: 0.55
 mae: 43787.90
 mse: 3923663371.09
```

Resultado R2: 0.55

CASO 2:

```
r2_arbol=r2_score(y6_test, sc_y.inverse_transform([y_randomForest]).reshape(509,1) )
mae_arbol=mean_absolute_error(y5_test, sc_y.inverse_transform([y_randomForest]).reshape(509,1))
mse_arbol=mean_squared_error(y5_test, sc_y.inverse_transform([y_randomForest]).reshape(509,1))
print(f'Estadisticos de la regresion por Bosques Aleatorios\n r2: {r2_arbol:.2f} \n mae: {mae_arbol:.2f} \n mse: {ms
```

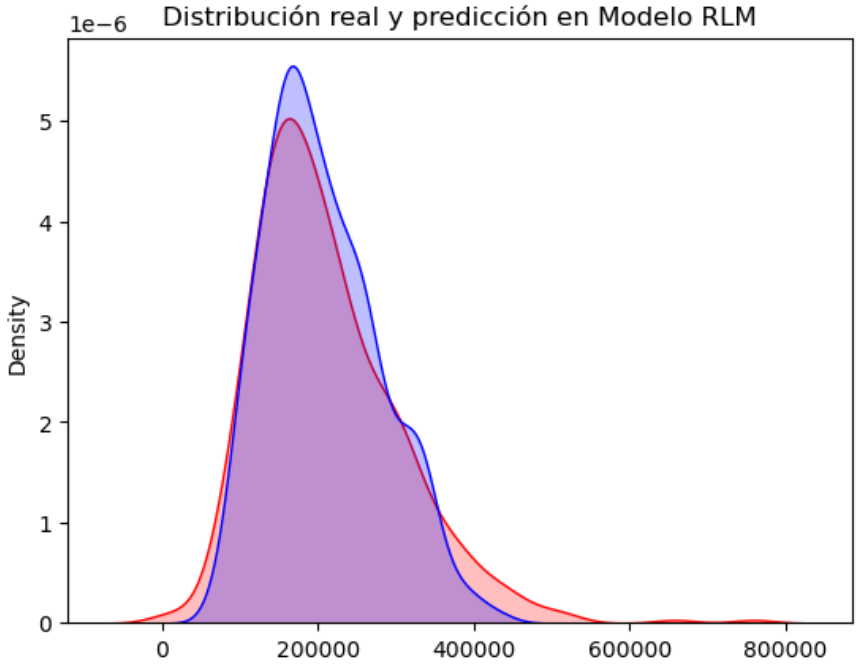
```
Estadisticos de la regresion por Bosques Aleatorios
 r2: 0.59
 mae: 41970.68
 mse: 3555614933.66
```

Resultado R2: 0.59

Iguala el resultado del modelado por RLM (regresión lineal múltiple).

Visualización test Vs predicción en la RLM

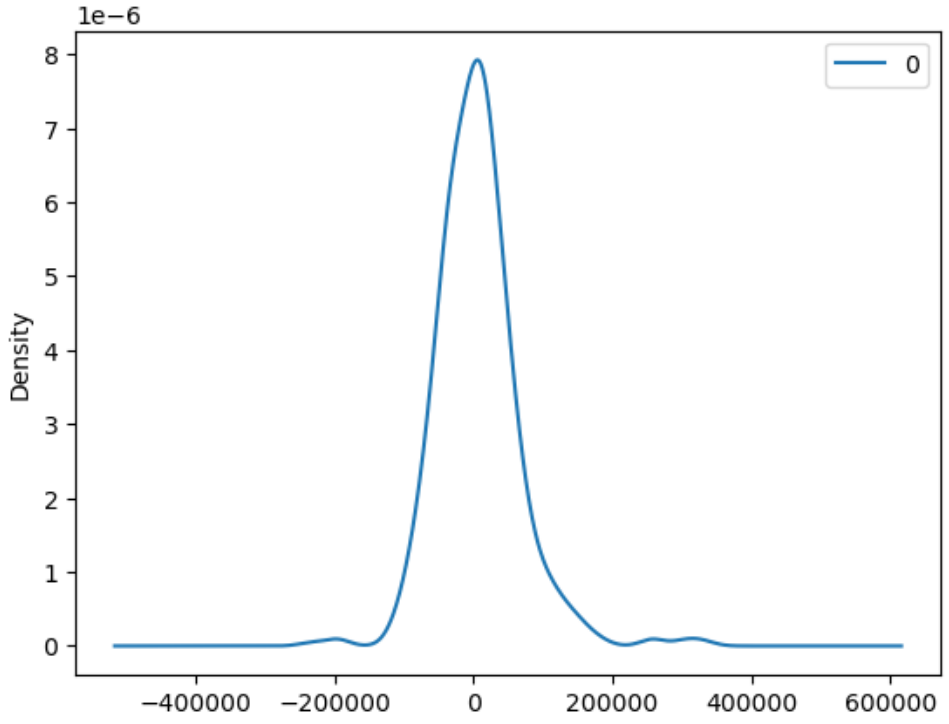
```
plt.title('Distribución real y predicción en Modelo RLM')
res = sn.kdeplot(y_test, color='red', shade='True', legend='Distribución real')
res2 = sn.kdeplot(y_pred_test, color='blue', shade='False', legend='Distribución predicción')
plt.show()
```



Distribución de las diferencias

```
pd.DataFrame(np.round(y_test-y_pred_test)).plot.kde()
```

```
<AxesSubplot:ylabel='Density'>
```



CONCLUSIONES

Con los datos aportados se consiguieron tres modelos (RLM, SVR Radial y Random Forest) con los que se obtuvo una bondad de ajuste R2 de 0.59 al predecir la variable objetivo ‘precio’. El r2 más alto de todos los modelos y casos aplicados.

Los datos presentan en general bastante dispersión y no alcanzan a explicar la variable objetivo con más precisión.