

**PREDICCIÓN DE PRECIOS DE VIVIENDAS  
TÉCNICAS AVANZADAS DE REGRESIÓN**

**POR:**

César Augusto López Castillo

Mateo Yepes Sierra

Brahian Monsalve Mejía

**MATERIA:**

Introducción a la Inteligencia Artificial

**PROFESOR:**

Raúl Ramos Pollan



UNIVERSIDAD DE ANTIOQUIA

FACULTAD DE INGENIERÍA

MEDELLÍN 2022

## Contenido

<b>Planteamiento del problema .....</b>	<b>4</b>
<b>Data set .....</b>	<b>4</b>
<b>Descripción de los datos: .....</b>	<b>4</b>
<b>Métrica de Desempeño .....</b>	<b>7</b>
<b>Variable Objetivo .....</b>	<b>7</b>
<b>Exploración de variables .....</b>	<b>7</b>
<b>Análisis de la variable objetivo .....</b>	<b>10</b>
<b>Tratamiento de datos.....</b>	<b>12</b>
<b>Remoción de lecturas cero en la variable objetivo:.....</b>	<b>12</b>
<b>Combinación de data set:.....</b>	<b>12</b>
<b>Relleno de datos faltantes: .....</b>	<b>12</b>
<b>Métodos supervisados .....</b>	<b>14</b>
<b>Selección de modelos.....</b>	<b>14</b>
<b>Ajuste de la métrica:.....</b>	<b>14</b>
<b>Partición de los datos.....</b>	<b>15</b>
<b>Código de selección de modelos .....</b>	<b>15</b>
<b>Mejores hiperparámetros de los modelos.....</b>	<b>15</b>
<b>Combinación de las predicciones.....</b>	<b>16</b>
<b>Retos y condiciones de despliegue del modelo .....</b>	<b>17</b>
<b>Conclusiones .....</b>	<b>18</b>
<b>Bibliografía.....</b>	<b>18</b>

## INTRODUCCIÓN

A través de la inteligencia artificial se pueden desarrollar infinidad de proyectos. En este caso emplearemos algoritmos de *regresión* para predecir los precios de venta y algunos costos relacionados con la negociación de edificaciones de vivienda, partiendo de modelos que analicen los datos, requerimientos vigentes, requisitos por parte del comprador y demás.

Claramente este proyecto resulta muy útil en diferentes áreas de la economía, venta y urbanización, ya que se podrán tener en cuenta gran variedad de aspectos importantes al momento de satisfacer todas las necesidades y requerimientos en la negociación de la vivienda, con gran precisión en poco tiempo

## Planteamiento del problema

Si se le pide a un comprador que describa la casa de sus sueños, probablemente no empezará por la altura del techo del sótano o la proximidad a una vía férrea este-oeste. Pero el conjunto de datos de este concurso demuestra que hay muchas más cosas que influyen en las negociaciones sobre el precio que el número de dormitorios o una valla de malla blanca.

Con 79 variables explicativas que describen (casi) todos los aspectos de las viviendas residenciales en Ames, Iowa, esta competición le reta a predecir el precio final de cada vivienda.

El problema consiste en predecir el precio de venta de cada casa. Para cada ID del conjunto de pruebas, se debe predecir el valor de la variable Sale Prices.

### Data set

Vamos a usar el data set de Kaggle perteneciente a la competencia “House Prices - Advanced Regression Techniques”:

<https://www.kaggle.com/competitions/house-prices-advanced-regression-techniques/data>

Descripción de los archivos:

- train.csv - Set de entrenamiento
- test.csv - Set de testeo
- data\_description.txt - Descripción completa de cada columna, preparada originalmente por Dean De Cock, pero ligeramente editada para que coincida con los nombres de las columnas utilizados aquí
- sample\_submission.csv - Una presentación de referencia a partir de una regresión lineal sobre el año y el mes de la venta, los metros cuadrados del lote y el número de habitaciones

### Descripción de los datos:

En los archivos de datos encontraremos las siguientes variables:

- SalePrice - El precio de la propiedad en dólares. Esta es la variable objetivo que vamos a predecir
- MSSubClass: La clase de edificación
- MSZoning: la clasificación de la zona general
- LotFrontage: distancia lineal en pies que está conectada a la carretera
- LotArea: área del lote en pies al cuadrado

- Street: Tipo de carretera de acceso
- Alley: tipo de callejón para acceder
- LotShape: forma general de la propiedad
- LandContour: llanura de la propiedad
- Utilities: tipo de utilidades disponibles
- LotConfig: configuración del lote
- LandSlope: pendiente de la propiedad
- Neighborhood: ubicaciones físicas dentro de los límites de la ciudad de Ames
- Condition1: Proximidad de la carretera principal o vía férrea
- Condition2: Proximity to main road or railroad (if a second is present)
- BldgType: tipo de vivienda
- HouseStyle: estilo de la vivienda
- OverallQual: Material general y calidad del acabado
- OverallCond: calificación de la condición general
- YearBuilt: fecha de la construcción original
- YearRemodAdd: fecha de remodelación
- RoofStyle: Tipo de tejado
- RoofMatl: material del tejado
- Exterior1st: Revestimiento general de la casa
- Exterior2nd: Revestimiento general de la casa si hay más de un material
- MasVnrType: tipo de chapa de mampostería
- MasVnrArea: area de revestimiento de mampostería en pies cuadrados
- ExterQual: calidad del material exterior
- ExterCond: condición actual del material en el exterior
- Foundation: tipo de cimiento
- BsmtQual: altura del sótano
- BsmtCond: condición general del sótano
- BsmtExposure: cantidad de muros en el sótano
- BsmtFinType1: calidad del área terminada del sótano
- BsmtFinSF1: Pies cuadrados terminados de tipo 1
- BsmtFinType2: Calidad de la segunda área terminada (si existe)
- BsmtFinSF2: Pies cuadrados terminados de tipo 2
- BsmtUnfSF: Pies cuadrados de superficie de sótano sin terminar
- TotalBsmtSF: Total de pies cuadrados de superficie del sótano
- Heating: Tipo de calentamiento
- HeatingQC: Calidad y estado de la calefacción
- CentralAir: Aire acondicionado central
- Electrical: Sistema eléctrico
- 1stFlrSF: Pies cuadrados del primer piso
- 2ndFlrSF: Pies cuadrados del segundo piso
- LowQualFinSF: Pies cuadrados de calidad inferior (todas las plantas)

- GrLivArea: Superficie habitable por encima del nivel del suelo (pies cuadrados)
- BsmtFullBath: Baños completos en el sótano
- BsmtHalfBath: Medios baños en el sótano
- FullBath: Baños completos sobre el nivel del suelo
- HalfBath: Medios baños sobre el nivel del suelo
- Bedroom: Número de habitaciones por encima del nivel del sótano
- Kitchen: Número de cocinas
- KitchenQual: Calidad de la cocina
- TotRmsAbvGrd: Total de habitaciones sobre el nivel del suelo (no incluye los baños)
- Functional: Valoración de la funcionalidad del hogar
- Fireplaces: número de chimeneas
- FireplaceQu: Calidad de la chimenea
- GarageType: Localización del garaje
- GarageYrBlt: Año de construcción del garaje
- GarageFinish: Acabado interior del garaje
- GarageCars: Tamaño del garaje en capacidad de carros
- GarageArea: Tamaño del garaje en pies cuadrados
- GarageQual: Calidad del garaje
- GarageCond: Condición del garaje
- PavedDrive: Calzada pavimentada
- WoodDeckSF: Superficie de la cubierta de madera en pies cuadrados
- OpenPorchSF: Área del pórtico abierto en pies cuadrados
- EnclosedPorch: Superficie del pórtico cerrado en pies cuadrados
- 3SsnPorch: Superficie del pórtico de tres estaciones en pies cuadrados
- ScreenPorch: Superficie del pórtico en pies cuadrados
- PoolArea: Área de la piscina en pies cuadrados
- PoolQC: Calidad de la piscina
- Fence: Calidad de valla
- MiscFeature: Características diversas no incluidas en otras categorías
- MiscVal: \$Valor de la función miscelánea
- MoSold: Mes de vendido
- YrSold: Año de vendido
- SaleType: Tipo de venta
- SaleCondition: Condición de venta

## Métrica de Desempeño

La métrica para medir el desempeño de la predicción será con el error cuadrático medio (RMSE) entre el logaritmo del valor predicho y el logaritmo del precio de venta observado.

$$RMSE = \sqrt{\frac{\sum_{i=1}^N (Predicted_i - Actual_i)^2}{N}}$$

Donde:

- N: es el número de datos totales
- Predicted: los datos que predice el modelo actual
- Actual: el valor real del dato

En cuanto a la métrica de negocio, se busca predecir el valor de la casa a partir de ciertas características que la describen. Se espera que esto ayude a ahorrar tiempo y dinero en procesos de selección por parte de los clientes.

## Variable Objetivo

Como ya se mencionó anteriormente, la variable objetivo que se desea predecir en este caso es "Sale Price", la cual nos dará el valor de cada casa según sus características, se analizarán y, posteriormente, se decidirá cuáles variables serán las entradas para el entrenamiento de los algoritmos.

## Exploración de variables

Para iniciar con la exploración de variables lo primero que hace es empezar a observar y analizar las diferentes columnas en los datos de "train" (descargados desde kaggle y adjuntados al repositorio del proyecto en GitHub). De forma que se obtiene un data set con todas las variables disponibles, y el análisis inicial de la variable objetivo y su relación con el data set. También incluimos el importe de todas las librerías y módulos necesarios para el adecuado desarrollo del proyecto. El objetivo de hacer una exploración inicial de datos es saber que variables tienen una correlación más influyente con la variable objetivo que en este caso es "sales price" y también arreglar el

data set de manera que al final solo tengamos una matriz numérica y ordenada.

Primero empezamos mirando que tamaño tenía la matriz y nos arrojó (1460,81) es decir, 1460 casas y 79 descriptores de cada casa quitando el ID y el Sales Price.

Luego miramos con comando de Pandas qué columnas tienen valores faltantes.

Después de eso hicimos una distribución de la variable objetivo y nos dimos que no se comporta como una distribución normal.

Más adelante inspeccionamos solo las columnas numéricas mediante una tabla que muestra la media, la cantidad de datos, el valor máximo y mínimo, etc. y vimos que hay unas pocas que tienen valores faltantes.

Luego construimos una matriz de gráficas para encontrar o analizar posibles correlaciones

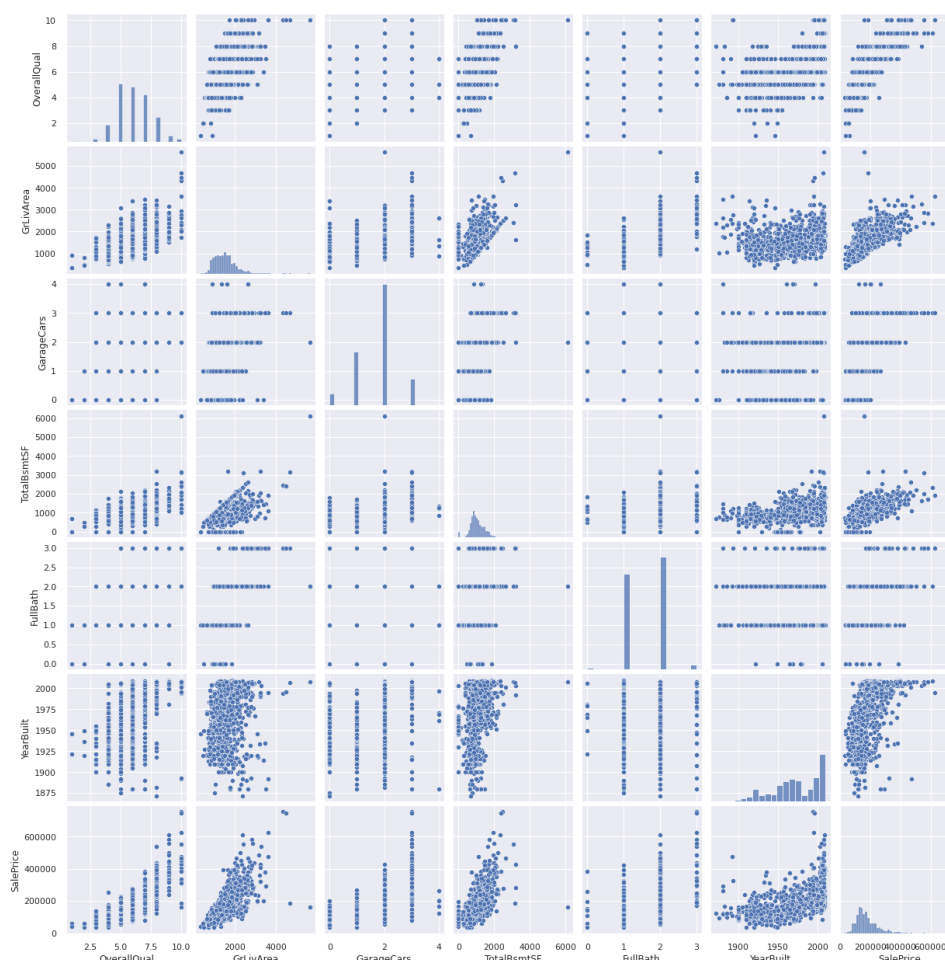


Figura 1. Matriz de correlaciones entre diferentes variables numéricas

Por ejemplo, que el tamaño de la sala es directamente proporcional al precio



de venta, pero desde cierto umbral hay mucha variabilidad en dicho precio, es una variable muy informativa y que guarda mucha correlación con el objetivo, mientras que por ejemplo el año de construcción de la casa si bien guarda una estrecha relación con el objetivo, no tiene una tendencia tan marcada como el area de la sala.

También vimos que algunas variables se comportan de manera totalmente discreta.

Este tipo de análisis nos sirve para intuir un poco o para localizar errores en las predicciones.

Luego construimos la siguiente grafica que nos ayuda a identificar mejor esas correlaciones.

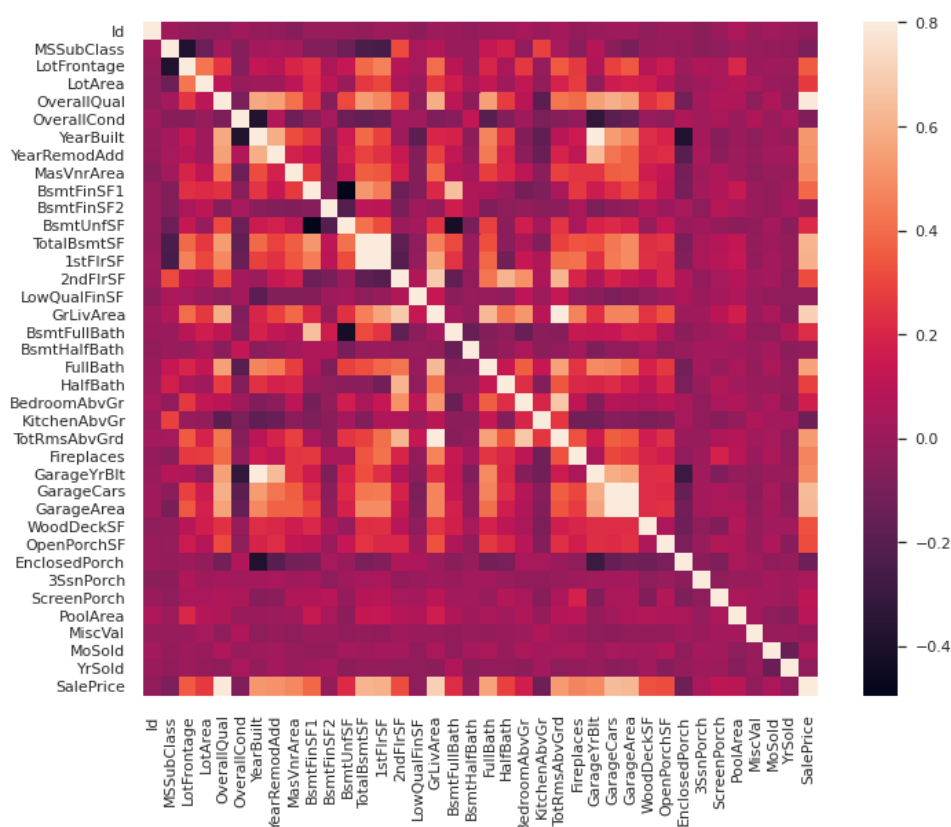


Figura 2. Tabla de correlaciones

Los puntos blancos son las variables más correlacionadas.

Y por último empezamos a inspeccionar las variables categóricas

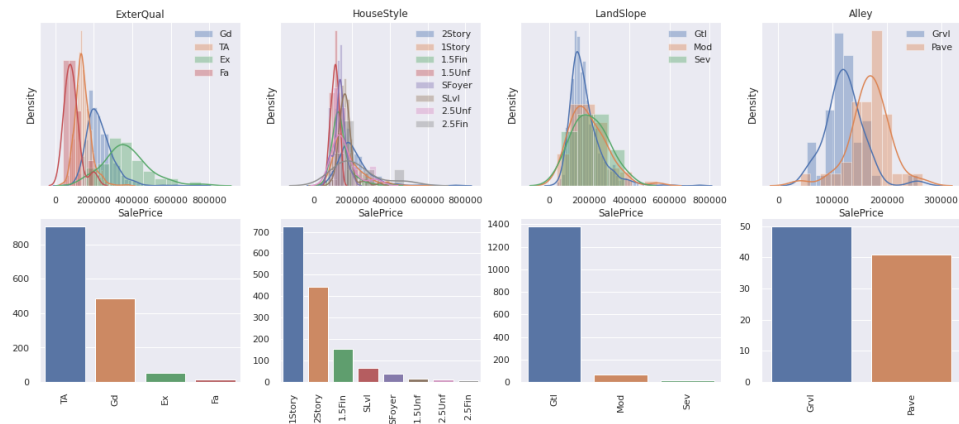


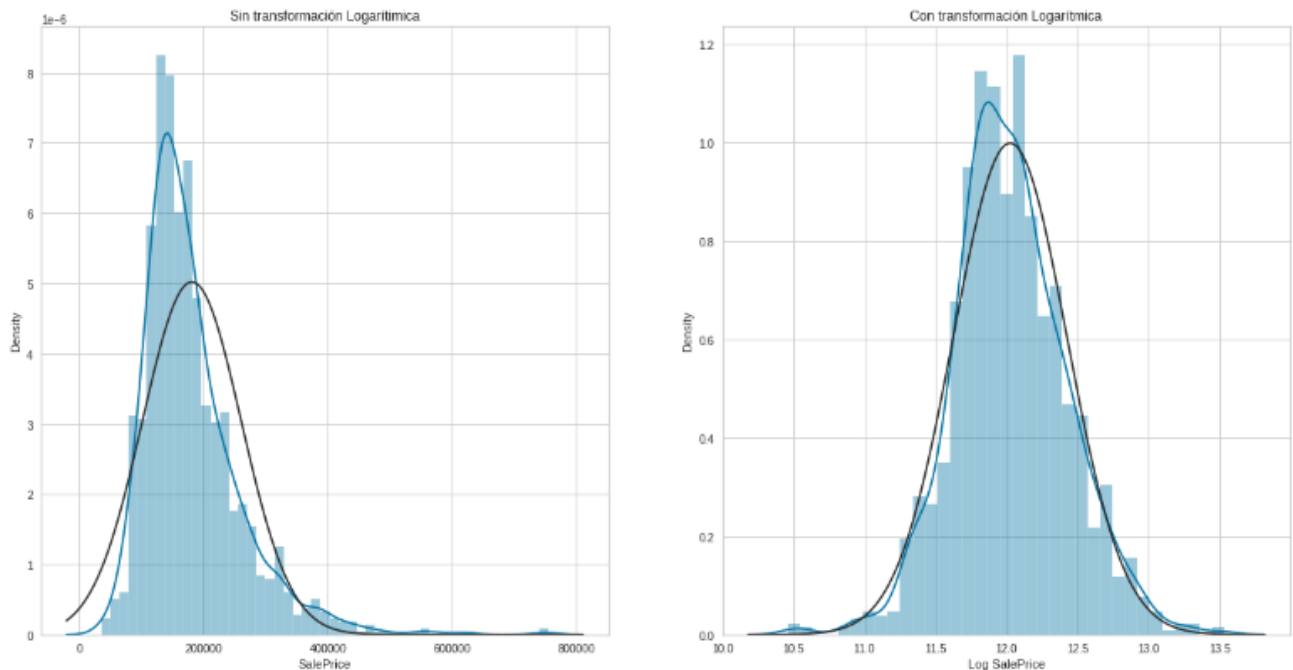
Figura 3. Matriz de correlaciones de variables categóricas

Vemos pues a modo de ejemplo que la variable Alley guarda más información con respecto al precio de venta que la variable HouseStyle, ya que House Style tiene los datos demasiado mezclados.

Mediante este tipo de gráficas y siguiendo este tipo de razonamientos terminamos la exploración de los datos.

## Análisis de la variable objetivo

Para empezar con la exploración de variables, se analiza el comportamiento de la distribución que tiene la variable objetivo.



*Figura 4. Distribución de la variable objetivo*

En la gráfica se puede observar que la variable objetivo tiene una asimetría hacia la izquierda. Este problema puede ser arreglado mediante una transformación logarítmica que aplicaremos después de tener un data set más ordenado con la limpieza de datos necesaria.

Buscamos transformar nuestros datos para que tengan una distribución normal, de esta manera esperamos que el modelo se desenvuelva mucho mejor. Lo que buscamos es volver más normal (simétrica) la distribución de los datos que puede estar inclinada a la derecha o a la izquierda.

observemos la inclinación (skew) de las columnas. Si este valor es negativo, significa que la mayoría de los datos se encuentran

#a la izquierda, si es cero significa que es perfectamente simétrico y si es positivo, significa que está inclinada hacia la derecha.

Podemos ver cómo está la inclinación de las columnas que contienen valores numéricos

Creamos un dataframe para almacenar las inclinaciones

sacamos la inclinación de cada una de las columnas y la almacenamos

Como tenemos algunos valores negativos aplicamos valor absoluto y almacenamos en otra columna

Ahora creamos otra columna para verificar si está inclinado basado en el criterio de +0.5 y almacenamos

Tomamos del dataframe los que están "inclinados", es decir los que en la línea anterior dieron True

Y a estos les aplicamos la transformación logarítmica para que tengan una distribución más normal y no inclinada,

: cómo podemos tener valores cero y log no está definido en cero, usamos `np.log1p` para no tener este error

Esta columna tiene un comportamiento cíclico (el mes en que se vendió una casa), aplicaremos en esta una

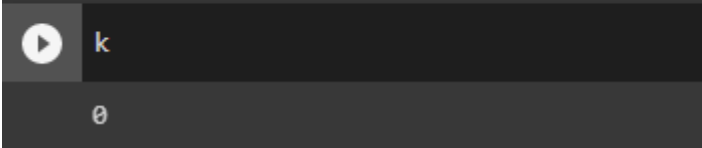
#Transformación Coseno que nos permitirá deducir el clima durante el año, donde dependiendo del mes obtendremos un clima frío (negativo) o caliente (positivo)  
#Esto nos servirá más adelante

## Tratamiento de datos

### Remoción de lecturas cero en la variable objetivo:

Nuestra variable objetivo no tiene datos vacíos.

```
[42] k = train0["SalePrice"].isna().sum()
      #k[k!=0]
```



A screenshot of a Jupyter Notebook cell. The code defines a variable 'k' as the sum of missing values in the 'SalePrice' column of the 'train0' dataset. The output shows 'k' with a value of 0.

Figura 5. Ceros de la data set “train”

### Combinación de data set:

Como se encontró en el análisis exploratorio, existen algunas variables que tienen datos faltantes. La figura muestra el código implementado para eliminar dichas variables.

Para esto extrajimos de los dos data set, Entrenamiento y Testeo, las variables objetivo “SalePrice” e “ID”, para luego combinarlas y tener un solo data set.

```
#Con el fin de no mezclar la información relevante, vamos a aislar nuestras variables objetivo y a sacarlas del dataset que vamos a copiar:
target = train0["SalePrice"]
test_ids = test0["Id"]
#Ahora sacamos de los dataset los objetivos y creamos una copia de estos:
train1 = train0.drop(["Id", "SalePrice"], axis=1)
test1 = test0.drop("Id", axis=1)
#Concatenamos
data1 = pd.concat([train1, test1], axis=0).reset_index(drop=True)
```

Figura 6. Código para combinar los data set.

### Relleno de datos faltantes:

El relleno de los datos faltantes se hizo en dos partes. Dado que tenemos datos categóricos y numéricos, decidimos identificar cómo eran los categóricos, si estos tenían vacíos con significado, los rellenamos con “None” y si no tienen significado, se llenan con NA.

```
#Reemplazamos los vacíos, en las columnas cuyo vacío tiene significado:
for column in ['Alley', 'BsmtQual', 'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinType2', 'FireplaceQu', 'GarageType', 'GarageFinish',
               'GarageQual', 'GarageCond', 'PoolQC', 'Fence', 'MiscFeature']:
    data2[column] = data2[column].fillna("None")
#Reemplazamos los vacíos, en las columnas cuyo vacío no tiene significado:
for column in ['MSZoning', 'Utilities', 'Exterior1st', 'Exterior2nd', 'MasVnrType', 'Electrical', 'KitchenQual', 'Functional', 'SaleType']:
    data2[column] = data2[column].fillna(data2[column].mode()[0])
```

Figura 7. Código implementado para rellenar los datos categóricos.

Luego creamos una función para llenar los datos vacíos numéricos mediante predicción usando el regressor KNeighbors, así:

```
def value_knn(df, na_target):
    df = df.copy()
    #Seleccionamos únicamente los datos numéricos del df
    numeric_df = df.select_dtypes(np.number)
    #Estas son los valores de las columnas que contienen datos numéricos que no tienen datos vacíos
    non_na_columns = numeric_df.loc[:, numeric_df.isna().sum() == 0].columns
    #básicamente vamos a usar esta información para rellenar los vacíos
    #y_train tiene todos los valores de las columnas dentro de na_target que no son vacíos
    y_train = numeric_df.loc[numeric_df[na_target].isna() == False, na_target]
    #x_train contiene todos los valores de las columnas de non_na_columns (el resto de los datos) que no son vacíos
    x_train = numeric_df.loc[numeric_df[na_target].isna() == False, non_na_columns]
    #x_test contienen todos los valores de las columnas que contienen datos vacíos
    x_test = numeric_df.loc[numeric_df[na_target].isna() == True, non_na_columns]
    #Creamos el regressor
    knn = KNeighborsRegressor()
    #Aplicamos el fit
    knn.fit(x_train, y_train)
    #Hacemos la predicción
    y_pred = knn.predict(x_test)
    #finalmente y_pred contiene el valor con el que vamos a llenar los vacíos
    df.loc[df[na_target].isna() == True, na_target] = y_pred

    return df
```

Figura 8. Función para llenar datos numéricos vacíos

Finalmente llenamos con estos datos las columnas de la data set que presentaban vacíos:

```
#Usamos la función anterior para rellenar los datos vacíos en todas las columnas:
for column in ['LotFrontage', 'MasVnrArea', 'BsmntFinSF1', 'BsmntFinSF2', 'BsmntUnfSF', 'TotalBsmntSF', 'BsmntFullBath', 'BsmntHalfBath', 'GarageYrBlt',
               'GarageCars', 'GarageArea']:
    data3 = value_knn(data3, column) #Definimos data3
data4 = data3.copy() #hacemos una copia por si las moscas
```

Figura 9. Llenado de datos numéricos vacíos

**Adición de la variable estación y variable para saber si es de día o noche:** Para este análisis se encuentra interesante el hecho de saber de qué estación del año es el cada dato, ya que normalmente en sistemas que ver con calentamiento de agua, o enfriadores de agua para sistemas de aire acondicionado, o sistemas de vapor, la estación del año puede influir en la carga que se impone en los sistemas aumentando o disminuyendo su consumo. En la Figura 15 se muestra el código empleado para añadir estas variables a la data set.

```
data4['MoSold'] = (-np.cos(0.5236 * data5['MoSold'])) #Esta columna tiene un comportamiento cíclico (el mes en que se vendió una casa), aplicaremos en esta una
#Transformación Coseno que nos permitirá deducir el clima durante el año, donde dependiendo del mes obtendremos un clima frío (negativo) o caliente (positivo)
#Esto nos servirá más adelante
data6 = data5.copy() #Hacemos otra copia
```

Figura 10. Código para añadir variables de interés.

- *Identificamos la “inclinación de los datos:*  
Para esto observamos el número del Skew que presenta el data set, luego vemos cuáles están por encima de 0.5 y a estos les aplicamos una transformación logarítmica con la función `np.log1p` que nos permite aplicarla incluso si tenemos valores numéricos cero:

```
pd.DataFrame(scp.stats.skew(data4.select_dtypes(np.number)))

skew_df = pd.DataFrame(data5.select_dtypes(np.number).columns, columns=['feature']) #Creamos un dataframe para almacenar las inclinaciones
skew_df['skew'] = skew_df['feature'].apply(lambda feature: scp.stats.skew(data5[feature])) #Sacamos la inclinación de cada una de las columnas y la almacenamos
skew_df['Absolute Skew'] = skew_df['skew'].apply(abs) #Como tenemos algunos valores negativos aplicamos valor absoluto y almacenamos en otra columna
skew_df['skewed'] = skew_df['Absolute Skew'].apply(lambda x: True if x >= 0.5 else False) #Ahora creamos otra columna para verificar si está inclinado basado en el criterio de >=0.5 y almacenamos
for column in skew_df.query('skewed == True')['feature'].values: #Tomamos del dataframe los que están "inclinados", es decir los que en la línea anterior dijeron True
    data5[column] = np.log1p(data5[column]) #Y a estos les aplicamos la transformación logarítmica para que tengan una distribución más normal y no inclinada,
#Nota: como podemos tener valores cero y log no está definido en cero, usamos np.log1p para no tener este error
```

*Figura 11. Inclinación de los datos*

- **Escalamiento:**  
Para trabajar más fácilmente con los datos, se realizó un escalamiento de los datos.

```
scaler = StandardScaler() #Le daremos a cada columna el mismo rango de valores
scaler.fit(data7) #Aplicamos el fit en el dataset codificado anteriormente

data7 = pd.DataFrame(scaler.transform(data7), index=data7.index, columns=data7.columns) #Aplicamos la transformación manteniendo las columnas anteriores
data8 = data7.copy()
```

*Figura 12. Escalamiento de los datos*

## Métodos supervisados

### Selección de modelos

#### Ajuste de la métrica:

Como se mencionó anteriormente, la métrica para medir el desempeño de los modelos será el RMSLE (Root Mean Squared Logarithmic Error), sin embargo, como realizamos una transformación logarítmica a la variable objetivo, la métrica que se puede implementar es el RMSE (Root Mean Squared Error), y a este valor sacarle la raíz cuadrada, y de esta forma ya tendríamos el RMSLE. En la *Figura 13* se muestra la función mediante la cual se calcula el RMSLE. Y además se realiza validación cruzada en 10 iteraciones.

```
results = {}

kf = KFold(n_splits=10)

for name, model in models.items():
    result = np.exp(np.sqrt(-cross_val_score(model, train_final, log_target, scoring='neg_mean_squared_error', cv=kf)))
    results[name] = result
```

*Figura 13. Función de cálculo del RMSLE.*

## Partición de los datos

Para entrenar los modelos se usará el data set que se generó, llamado *train*. De todo el conjunto de estos datos es necesario hacer una partición para entrenar y otra para prueba.

```
train_final = data8.loc[:train0.index.max(), :].copy()
test_final = data8.loc[train0.index.max() + 1:, :].reset_index(drop=True).copy()
```

Figura 14. Código para realizar la partición de datos.

## Código de selección de modelos

Para desarrollar el análisis se plantean inicialmente tres modelos: usaremos los modelos **Bayesian Ridge**, **Ridge Regressor** y **LGBM Regressor** para realizar las predicciones. Establecemos para cada modelo los parámetros que necesitar

```
for name, model in models.items():
    model.fit(train_final, log_target)
    print(name + " entrenado.")
```

```
Bayesian Ridge entrenado.
lightgbm Regressor entrenado.
Ridge Regressor entrenado.
```

Figura 15. Entrenamiento de modelos.

## Mejores hiperparámetros de los modelos

Una vez obtenidos los modelos a trabajar, se hace un estudio para obtener los mejores hiperparámetros de las variables. Esto se realiza mediante un módulo de la librería de *sk.learn.model\_selection*, llamado *GridSearchCV*. Este módulo permite hacer un análisis variando los hiperparámetros del algoritmo de interés implementando una metodología de *cross-validation*.

```
kf = KFold(n_splits=10)

for name, model in models.items():
    result = np.exp(np.sqrt(-cross_val_score(model, train_final, log_target, scoring='neg_mean_squared_error', cv=kf)))
    results[name] = result
```

Figura 16. Valores de los hiperparámetros para los modelos.

Para validar que los modelos tomados son buenos, sacamos el promedio de los valores y la desviación estándar. Vemos que la desviación estándar es baja y que el promedio es constante en los tres:

```
-----  
Bayesian Ridge  
Promedio: 1.1351015144106285  
Desviación Estandar: 0.025656641991269765  
-----  
lightgbm Regressor  
Promedio: 1.1323401113019387  
Desviación Estandar: 0.022377581009135113  
-----  
Ridge Regressor  
Promedio: 1.1345319533871139  
Desviación Estandar: 0.024898613347420384
```

*Figura 17. Promedio resultados*

## **Combinación de las predicciones**

Como encontramos que la variación del comportamiento de las predicciones con los tres modelos es bastante similar, decidimos combinar los tres resultados en partes iguales, de tal manera que la predicción se hiciera en base a los tres modelos:

```
final_predictions = ((1/3) * np.exp(models['br'].predict(test_final)) +  
    (1/3) * np.exp(models['lightgbm'].predict(test_final)) +  
    (1/3) * np.exp(models['ridge'].predict(test_final)))  
final_predictions
```

*Figura 18. Combinación de las predicciones*



No olvidemos que los datos estaban transformados logarítmicamente, por esto debemos aplicar la exponencial para volver a las unidades del inicio.

Finalmente obtenemos la predicción de “SalePrice”, así:

```
submission = pd.concat([test_ids, pd.Series(final_predictions, name='SalePrice')], axis=1)
submission
```

	Id	SalePrice
0	1461	121402.342757
1	1462	159052.561868
2	1463	181559.153288
3	1464	194069.231670
4	1465	193517.474485
...	...	...
1454	2915	85179.976389
1455	2916	81412.541374
1456	2917	164313.681385
1457	2918	116693.932071
1458	2919	219754.793583

*Figura 19. Predicción de la variable objetivo*

## Retos y condiciones de despliegue del modelo

Para evaluar el desempeño mínimo con el cual se consideraría que el modelo genera un beneficio, se debe comparar el gasto energético generado por las predicciones del modelo con el posible ahorro monetario que se podría lograr al implementar estrategias que reduzcan ese gasto. Si el modelo permite obtener un nivel de ahorro objetivo mínimo, el cual puede ser establecido por los inversores. Administradores o gerentes de las edificaciones, se puede considerar que el modelo es apto para desplegarse en producción.

Para desplegar el modelo en producción, es necesario conectar todos los sensores que brinden la información necesaria, a un computador donde puedan ser usados los datos para generar las predicciones. Uno de los retos que se tiene para implementar este modelo, es que las edificaciones deben ser de alguna manera “inteligentes” ya que deben contar con la instrumentación necesaria para tomar los datos e implementar una tecnología del internet de las cosas (IOT), ya que es necesario que estos dispositivos se estén comunicando para que el modelo puede ser usado. Así mismo también es necesario implementar herramientas de la nube que permitan guardar información y medir constantemente el desempeño del modelo, y en el caso en el que dicho desempeño caiga por debajo de los niveles establecidos, brindar información para reentrenar el modelo.

## Conclusiones

- Es necesario hacer un análisis detallado de que variables tienen más peso en los modelos entrenados, ya que de esta manera se puede reducir el error.
- Es necesario aumentar la complejidad de algunos modelos ya que se presentan problemas de overfitting.
- Los métodos no supervisados pueden ser implementados como una forma de preprocesado de datos, que nos permita generar un data set más compacto guardando las variables más significativas.
- Se puede implementar un modelo de *Clustering* para analizar los patrones y comportamientos de cada tipo de edificación.

## Bibliografía

- ASHRAE - Great Energy Predictor III | Kaggle. (2021). Retrieved 16 December 2021, from <https://www.kaggle.com/c/ashrae-energy-prediction/overview/description>
- House Prices - Advanced Regression Techniques | Kaggle. (2016). Kaggle. <https://www.kaggle.com/competitions/house-prices-advanced-regression-techniques/overview/description>