Análisis de regresión para la predicción de precios de venta de casas en Boston usando MAE

Fernando Rodrigo Aguilar Javier

I. Introducción

En este reporte se trata de predecir cuanto cuesta una casa que esta a la venta en la ciudad de Boston, Massachusets, en este contexto tratare de explorar los datos y visualizarlos para ver si con un modelo de regresión logro aproximar su precio y calcular el error usando Mean Absolute Error MAE. Para esto se llevo a cabo una limpieza y preparacin de los datos usando las librerias estandar de python SckitLearn, Pandas, Sborn, numpy y scipy.

II. Descripción de los datos

Los conjuntos de datos se tomaron del sitio https://www.kaggle.com/c/house-prices-advanced-regression-techniques/data
Este sitio contenía los datos train y test en formato Comma Separate Values(CSV) pero les hacia falta un preprocesamiento general, el archivo train.csv y test.csv contienen 81 campos(atributos), de estos 3 son de tipo flotante, 34 de tipo entero y 43 de tipo categórico y 1460 instancias:
Para la descripcción completa del conjunto de datos Houses Prices: Advanced Regression Techniques viste https://www.kaggle.com/c/house-prices-advanced-regression-techniques/data

A. Exploración de los datos

A primera vista podemos observar que existen muchos datos vacíos o no disponibles(NaN), eso puede llegar a ser un problema y pues una opción muy simple es quitarlos.(ver Fig. 1)

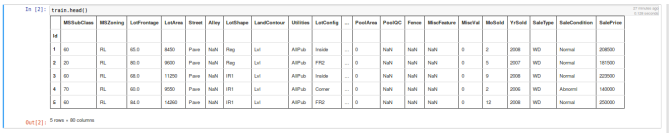

Fig. 1. Primeros 5 campos (instancias)de los datos

Usando el metodo **describe** de la libreria Pandas de Python se logra obtener un analisis rapido de los datos para cada variable (ver Fig. 2) nos muestra su media, su desviación estandar, los valores max y min, asi como que porcentaje de los datos representan. Esto nos permite realizar una exploración a los datos de forma rápida y facíl de ver.

Advisor: Dr. Raggi Perez, Miguel, ENES Unidad Morelia, UNAM

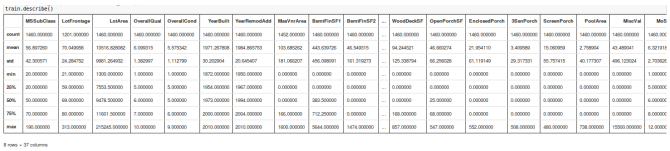

Fig. 2. Matriz descriptiva de los datos

III. Visualización de los datos

Se plotteo la matriz de correlaciones de atributos de todos contra todos como primera instancia (ver Fig. 3)

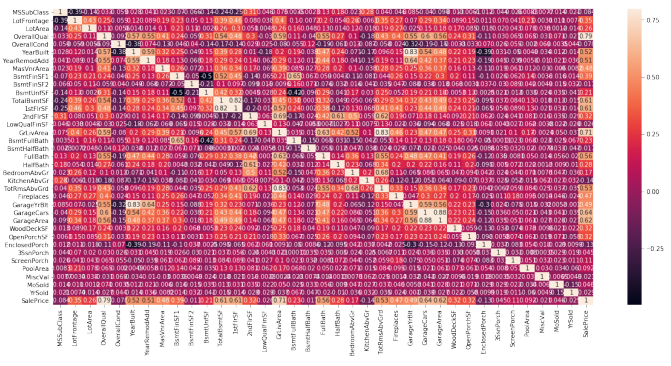

Fig. 3. Matriz descriptiva de los datos

Para apreciar mejor la correlación que existe entre las variables eh decido tomar aquellos atributos que tengan una correlacin mayor a 0.5 con respecto únicamente a SalePrice (Precio a la venta) ya que es nuestro campo de interés (ver Fig. 4). En la figura se muestra que los atributos estan correlacionados con otros atributos, pues están correlacionados no solamente con SalePrice sino entre ellos mismos tambien, como Garage Cars y Garage Área que es de .88
OverallQual (calidad en general de la casa) está altamente correlacionado con la característica de interés SalePrice 0.79 como se observa en la Fig. 4. Ahora se presenta cómo se efectuó el precio de venta en la gráfica de la Fig. 5.

IV. Preprocesamiento

Ahora bien lo que queremos es crear un modelo que prediga el precio a la venta de las casas(Saleprice), pero es necesario realizar un estudio mas detallado sobre el comportamiento de esta variable.
Para eso are un análisis estadístico usando scipy que contiene el modulo stats y sus métodos norm y skew.

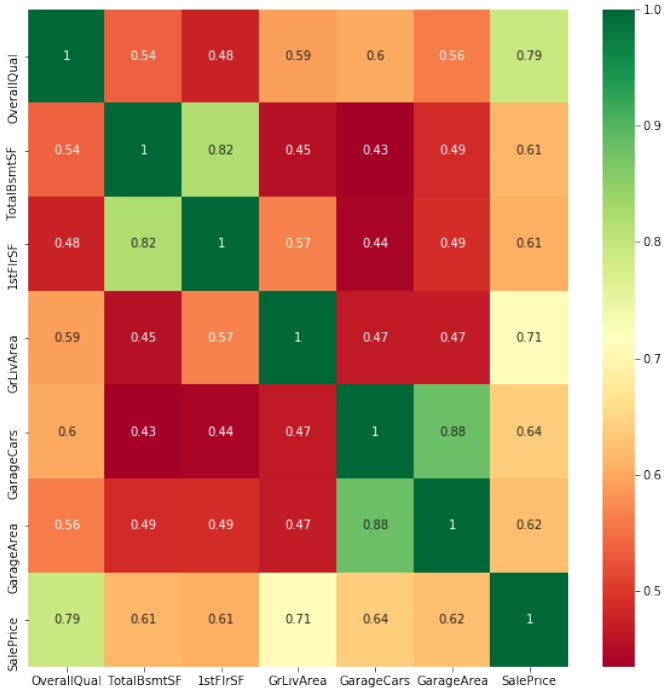

Fig. 4. Matriz descriptiva de los datos

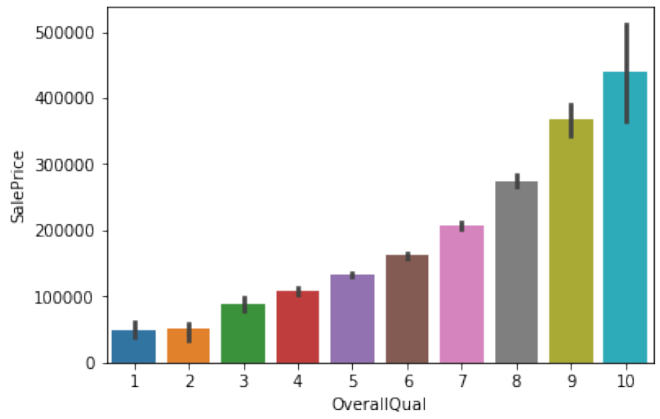

Fig. 5. Matriz descriptiva de los datos

stats Es un modulo de scipy que contiene muchas distribuciones de probabilidad.
Skew Calcula la asimetría de un conjunto de datos.
norm Crea una variable normal continua de forma aleatoria.

Se ploteo la variable SalePrice(precio a la venta de las casas) contra GrLivArea(pies cuadrados de la casa) pues es la variable que tiene mayor correlación de .71 (ver Fig. 7)
En la Fig. 8 se ven algunos outlayers de las variables, que podrían afectar en cierta medida al modelo dado que es la variable mas importante pues existe una correlación muy alta.

El modelo que usare como primera instancia es regresión con un árbol de decisión para ello tomare solo variables numericas (ver Fig. 10), puesto que los arboles de decisión

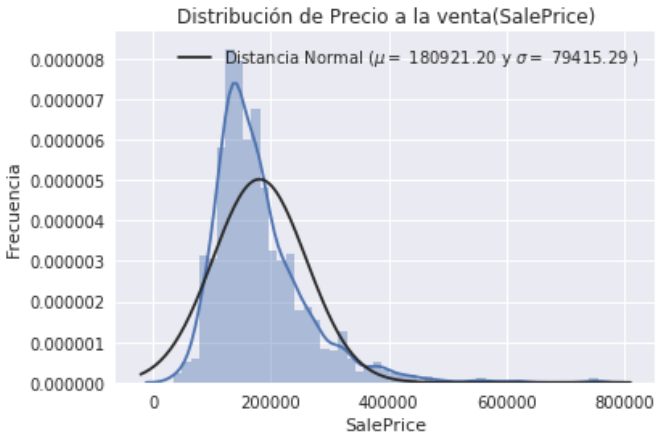

Fig. 6. Distribución normal con sesgo de la variable SalePrice

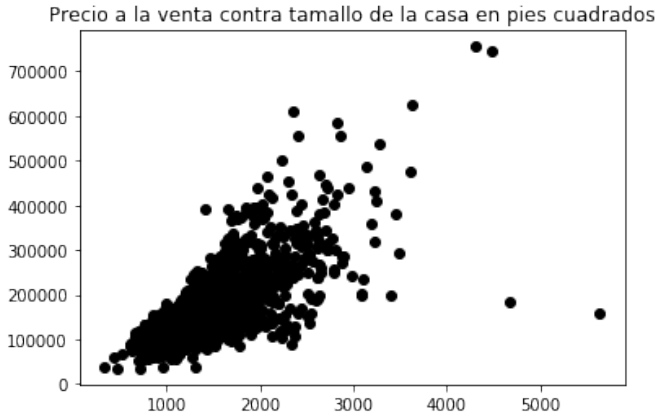

Fig. 7. Scatter plot con los valores normales de SalePrice

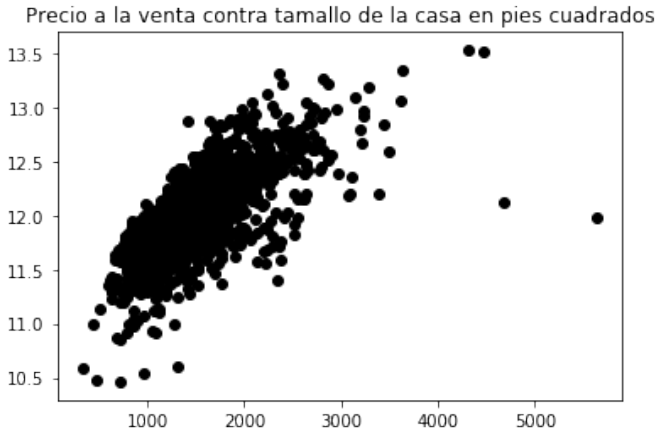

Fig. 8. Scatter plot con los valores de SalePrice mapeados con el logaritmo natural

solo pueden trabajar con este tipo y ademas tomare de estas variables las que tienen mayor grado de correlación (ver Fig. 11).

Se creo el modelo de regresión (ver Fig. 12) con un árbol de decisión usando la líbreria de **Sklearn** de Python Vemos

```
In [39]: corr = train.corr()
         corr.sort_values(["SalePrice"], ascending = False, inplace = True)
         corr.SalePrice

Out[39]: SalePrice        1.000000
         OverallQual      0.817185
         GrLivArea        0.700927
         GarageCars       0.680625
         GarageArea       0.650888
         TotalBsmtSF      0.612134
         1stFlrSF         0.596981
         FullBath         0.594771
         YearBuilt        0.586570
         YearRemodAdd     0.565608
         GarageYrBlt      0.541073
         TotRmsAbvGrd     0.534422
         Fireplaces       0.489450
         MasVnrArea       0.430809
         BsmtFinSF1       0.372023
         LotFrontage      0.355879
         WoodDeckSF       0.334135
         OpenPorchSF      0.321053
         2ndFlrSF         0.319300
         HalfBath         0.313982
         LotArea          0.257320
         BsmtFullBath     0.236224
         BsmtUnfSF        0.221985
         BedroomAbvGr     0.209043
         ScreenPorch      0.121208
         PoolArea         0.069798
         MoSold           0.057330
         3SsnPorch        0.054900
         BsmtFinSF2       0.004832
         BsmtHalfBath    -0.005149
         MiscVal         -0.020021
         OverallCond     -0.036868
         YrSold          -0.037263
         LowQualFinSF    -0.037963
         MSSubClass      -0.073959
         KitchenAbvGr    -0.147548
         EnclosedPorch   -0.149050
         Name: SalePrice, dtype: float64
```

Fig. 9. Listado en forma ascendente de la correlación de las variables con respecto a SalePrice

```
In [65]: numerical_features = train.select_dtypes(exclude=['object']).columns
         column_interest = numerical_features[:-1]
         column_interest

Out[65]: Index(['MSSubClass', 'LotFrontage', 'LotArea', 'OverallQual', 'OverallCond',
                'YearBuilt', 'YearRemodAdd', 'MasVnrArea', 'BsmtFinSF1', 'BsmtFinSF2',
                'BsmtUnfSF', 'TotalBsmtSF', '1stFlrSF', '2ndFlrSF', 'LowQualFinSF',
                'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath', 'FullBath', 'HalfBath',
                'BedroomAbvGr', 'KitchenAbvGr', 'TotRmsAbvGrd', 'Fireplaces',
                'GarageYrBlt', 'GarageCars', 'GarageArea', 'WoodDeckSF', 'OpenPorchSF',
                'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea', 'MiscVal',
                'MoSold', 'YrSold'],
               dtype='object')
```

Fig. 10. Lista de variables númericas

```
In [71]: column_interest = ['OverallQual', 'TotalBsmtSF', '1stFlrSF', 'GrLivArea', 'FullBath', 'Fireplaces',
         'GarageCars', 'GarageArea']
         X_train = train[column_interest]
         X_train = X_train.dropna(axis=0)[:1457]
         Y_train  =  train.SalePrice[:1457]
         X_test = test[column_interest]
         X_test = X_test.dropna(axis=0)
         print(X_train.shape, Y_train.shape, X_test.shape)
         column_interest

         (1457, 8) (1457,) (1457, 8)

Out[71]: ['OverallQual',
          'TotalBsmtSF',
          '1stFlrSF',
          'GrLivArea',
          'FullBath',
          'Fireplaces',
          'GarageCars',
          'GarageArea']
```

Fig. 11. Lista de variables númericas que tienen mayor correlación

```
In [72]: #Creamos el modelo
         House_saleprice_model = DecisionTreeRegressor()
         #Entrenamos el modelo
         House_saleprice_model.fit(X_train, Y_train)

Out[72]: DecisionTreeRegressor(criterion='mse', max_depth=None, max_features=None,
                               max_leaf_nodes=None, min_impurity_decrease=0.0,
                               min_impurity_split=None, min_samples_leaf=1,
                               min_samples_split=2, min_weight_fraction_leaf=0.0,
                               presort=False, random_state=None, splitter='best')
```

Fig. 12. Valores que tomara el modelo

que el modelo tiene un error muy alto de mas menos $81299 dolares (ver Fig. 13).

De esta forma observamos que el modelo tiene un bajo ajuste (underfitting), para controlar esto en un modelo de arboles de decisión hay que aumentar la profundidad del árbol (ver Fig. 14)

Al momento de graficar la curva de validación (ver Fig. 15) se observa en los datos de prueba comienza a bajar su MAE pero en una profundidad de mas de 500 hojas existe ya un sobreajuste (Overfitting).

Por lo anterior la mejor configuración del modelo es con un maximo numero de hojas de 500 ya que el error no supero los 8000 dolares que comparado con el modelo inicial es muy bueno. Logro bajar casi .10 el error.

```
In [73]: from sklearn.metrics import mean_absolute_error

         predicted_home_saleprices = House_saleprice_model.predict(X_test)
         mean_absolute_error(Y_train, predicted_home_saleprices)

Out[73]: 81299.52184854724
```

Fig. 13. MAE del modelo

```
In [18]: from sklearn.metrics import mean_absolute_error
         from sklearn.tree import DecisionTreeRegressor

         def get_mae(max_leaf_nodes, predictors_train, predictors_val, targ_train, targ_val):
             model = DecisionTreeRegressor(max_leaf_nodes=max_leaf_nodes, random_state=0)
             model.fit(predictors_train, targ_train)
             preds_val = model.predict(predictors_val)
             mae = mean_absolute_error(targ_val, preds_val)
             return(mae)
         # compare MAE with differing values of max_leaf_nodes
         for max_leaf_nodes in [2, 5, 50, 500,1000, 3000]:
             my_mae = get_mae(max_leaf_nodes, X_train, X_test, Y_train, predicted_home_saleprices)
             print("Maximo numero de hojas: %d  \t\t Valor absoluto medio del error:  %d" %(max_leaf_nodes, my_mae))

         Maximo numero de hojas: 2               Valor absoluto medio del error:  41868
         Maximo numero de hojas: 5               Valor absoluto medio del error:  31805
         Maximo numero de hojas: 50              Valor absoluto medio del error:  19800
         Maximo numero de hojas: 500             Valor absoluto medio del error:  7203
         Maximo numero de hojas: 1000            Valor absoluto medio del error:  4591
         Maximo numero de hojas: 3000            Valor absoluto medio del error:  4338
```

Fig. 14. MAE con distintas profundidades del árbol

```
In [72]: plt.plot(arr_n_nodes, arr_mae)
         plt.title('Curva de validación')
         plt.xlabel('Profundidad del arbol')
         plt.ylabel('Valor absoluto medio del error')

         Text(0,0.5,'Valor absoluto medio del error')

Out[72]:
```

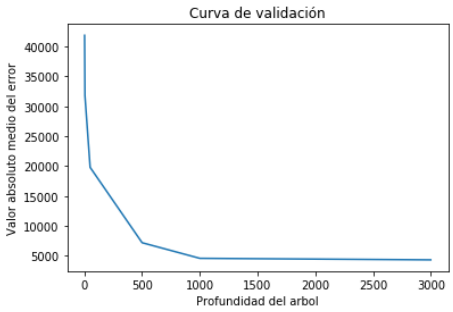

Fig. 15. Curva de validación

V. Conclusión

Es claro que la tendecía de uso y de atención al aprendizaje automático va creciendo cada vez mas y con forme uno se adentra mas al tema, se va dando uno cuenta de que existen un sin numero de algoritmos que son mas robustos, mas poderosos o con mayor rendimiento(performance), ya que permiten crear un modelo con mayor aproximación a la realidad de nuestro tema de estudio, ademas existen tecnicas mas avazadas y con forme uno practica y se adentra a este campo va adquiriendo ese filling que te caracteriza de los novatos para determinar que modelo ajustaría mejor a los datos

de manera cada vez mas rápida, a parte de que en varios lenguajes de programación ya permiten el uso de algoritmos que en otra ocación seria tardado e incluso muy lento por el tema de la impelmentación si lo programaras tu solo, por ello Python, R, Octave y varios mas están en su auge ya que permiten el uso de librerías, métodos standard con algoritmos ya implementados y que ademas son de uso libre (Free Software).

En este reporte vimos que aun el algoritmo mas sencillo como lo es arboles de decisión pueden tener un error considerable, en este contexto no basta con quedarse con un solo modelo hay que probar con otros tales como random forest que creo dará mejores resultados sin duda, o con aquellos modelos que puedan trabajar con datos híbridos, es decir, categóricos y númericos. De este modo se tomara mas información útil que mejore el modelo en base a los datos de entrenamiento, ademas del uso de cross validation, K-folds y mas métodos para mejorar la certeza del modelo.

REFERENCES

[1] Aleksey, Bilogur: Pandas, Short hands-on challenges to perfect your data manipulation skills https://www.kaggle.com/learn/pandas.

[2] Marcelino, Pedro: PComprehensive data exploration with Python https://www.kaggle.com/pmarcelino/comprehensive-data-exploration-with-python.

[3] Becker, Dans: Machine learning is the hottest field in data science, and this track will get you started quickly.. https://www.kaggle.com/learn/machine-learning.

[4] Underfitting, Overfitting and Model Optimization, Machine Learning Learn Tutorial
Author: dansbecker
https://www.kaggle.com/dansbecker/underfitting-overfitting-and-model-optimization