Housing Prices Competition for Kaggle Learn Users (pt. 2)

Pablo de la Asunción Cumbrera Conde

En el siguiente proyecto, continuación de 'Housing Prices Competition for Kaggle Learn Users (pt.1)', profundizaremos en la mejora de modelos desarrollados para la competición Housing Prices Competition for Kaggle Learn Users sobre la aplicación de Machine Learning para la predicción de precios de viviendas en Ames, Iowa.

La descripción e introducción de la competición reza así:

- Pídale a un comprador de vivienda que describa la casa de sus sueños, y probablemente no comenzará con la altura del techo del sótano o la proximidad a un ferrocarril que cruce todo el país. No obstante, este conjunto de datos para esta distendida competición demuestra que influye mucho más sobre el precio las negociaciones que el número de dormitorios o una valla blanca.

Objetivos para la comeptición:

- Con 79 variables explicativas que describen (casi) todos los aspectos de las viviendas residenciales en Ames, Iowa, esta competención desafía al usuario a predecir el precio final de cada casa.

Habilidades a poner en práctica:

- Ingeniería creativa de funciones
- Técnicas de regresión avanzadas como 'Random Forest' y 'Gradient Boosting'.
- Limpieza de datos y optimización de modelos

Contenidos

- Update y set del conjunto de datos
- Métodos y valoración de sustitución de valores ausentes
- Construcción y validación de modelo 'Random Forest' con mejoras introducidas
- Orientación para continuar el proceso de optimización: Variables categóricas

Update y set del conjunto de datos

In [1]:
```
import pandas as pd
from sklearn.model_selection import train_test_split

# Leemos Datos
X_full = pd.read_csv('file:///C:/Users/pablo/OneDrive/Escritorio/Data/Data%20sets/housingprice_train.csv', index_col='Id')
X_test_full = pd.read_csv('file:///C:/Users/pablo/OneDrive/Escritorio/Data/Data%20sets/housingprice_test.csv', index_col='Id')

# Obtención de objetivos y predictores
y = X_full.SalePrice
features = ['LotArea', 'YearBuilt', '1stFlrSF', '2ndFlrSF', 'FullBath', 'BedroomAbvGr', 'TotRmsAbvGrd']
X = X_full[features].copy()
X_test = X_test_full[features].copy()

# Para ello primero separamos las filas con valores ausentes, haciendo a un lado objetivos y predictores.
X_full.dropna(axis=0, subset=['SalePrice'], inplace=True)
y = X_full.SalePrice
X_full.drop(['SalePrice'], axis=1, inplace=True)


# Para mantener la simpleza, solo utilizaremos predictores numéricos.
X = X_full.select_dtypes(exclude=['object'])
X_test = X_test_full.select_dtypes(exclude=['object'])


# Separación de valores del set para entrenamiento
X_train, X_valid, y_train, y_valid = train_test_split(X, y, train_size=0.8, test_size=0.2,
                                                      random_state=0)

X_full.head()
```

Out[1]:

Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	LotConfig	...	ScreenPorch	PoolArea	PoolQC	Fence	MiscFeature	MiscVal	MoSold	YrSold	SaleType	SaleCondition
1	60	RL	65.0	8450	Pave	NaN	Reg	Lvl	AllPub	Inside	...	0	0	NaN	NaN	NaN	0	2	2008	WD	Normal
2	20	RL	80.0	9600	Pave	NaN	Reg	Lvl	AllPub	FR2	...	0	0	NaN	NaN	NaN	0	5	2007	WD	Normal
3	60	RL	68.0	11250	Pave	NaN	IR1	Lvl	AllPub	Inside	...	0	0	NaN	NaN	NaN	0	9	2008	WD	Normal
4	70	RL	60.0	9550	Pave	NaN	IR1	Lvl	AllPub	Corner	...	0	0	NaN	NaN	NaN	0	2	2006	WD	Abnorml
5	60	RL	84.0	14260	Pave	NaN	IR1	Lvl	AllPub	FR2	...	0	0	NaN	NaN	NaN	0	12	2008	WD	Normal

5 rows × 79 columns

In [2]:
```
X_test_full.head()
```

Out[2]:

Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	LotConfig	...	ScreenPorch	PoolArea	PoolQC	Fence	MiscFeature	MiscVal	MoSold	YrSold	SaleType	SaleCondition
1461	20	RH	80.0	11622	Pave	NaN	Reg	Lvl	AllPub	Inside	...	120	0	NaN	MnPrv	NaN	0	6	2010	WD	Normal
1462	20	RL	81.0	14267	Pave	NaN	IR1	Lvl	AllPub	Corner	...	0	0	NaN	NaN	Gar2	12500	6	2010	WD	Normal
1463	60	RL	74.0	13830	Pave	NaN	IR1	Lvl	AllPub	Inside	...	0	0	NaN	MnPrv	NaN	0	3	2010	WD	Normal
1464	60	RL	78.0	9978	Pave	NaN	IR1	Lvl	AllPub	Inside	...	0	0	NaN	NaN	NaN	0	6	2010	WD	Normal
1465	120	RL	43.0	5005	Pave	NaN	IR1	HLS	AllPub	Inside	...	144	0	NaN	NaN	NaN	0	1	2010	WD	Normal

5 rows × 79 columns

In [3]:
```
X_train.head()
```

Out[3]:

Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	BsmtFinSF1	BsmtFinSF2	...	GarageArea	WoodDeckSF	OpenPorchSF	EnclosedPorch	3SsnPorch	ScreenPorch	Poo
619	20	90.0	11694	9	5	2007	2007	452.0	48	0	...	774	0	108	0	0	260	
871	20	60.0	6600	5	5	1962	1962	0.0	0	0	...	308	0	0	0	0	0	
93	30	80.0	13360	5	7	1921	2006	0.0	713	0	...	432	0	0	44	0	0	
818	20	NaN	13265	8	5	2002	2002	148.0	1218	0	...	857	150	59	0	0	0	
303	20	118.0	13704	7	5	2001	2002	150.0	0	0	...	843	468	81	0	0	0	

5 rows × 36 columns

Realizamos una inserción en los valores que faltan (NaN- Not A Number) a través de la prueba de diversos métodos para probar cuál es el óptimo. Para evaluar las diferentes formas de aproximación utilizaremos el 'MAE'.

In [4]:
```
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error

# Función para comprobar la exactitud de las aproximaciones
def score_dataset(X_train, X_valid, y_train, y_valid):
    model = RandomForestRegressor(n_estimators=100, random_state=0)
    model.fit(X_train, y_train)
    preds = model.predict(X_valid)
    return mean_absolute_error(y_valid, preds)
```

Utilizaremos el set de entrenamiento de la X

In [36]:
```
print(X_train.shape)

# Number of missing values in each column of training data# Fill in the line below: get names of columns with missing values
cols_with_missing = [col for col in X_train.columns
                     if X_train[col].isnull().any()]

# Drop columns in training and validation data
reduced_X_train = X_train.drop(cols_with_missing, axis=1)
reduced_X_valid = X_valid.drop(cols_with_missing, axis=1) # Your code here

# Check your answers
missing_val_count_by_column = (X_train.isnull().sum())
print(missing_val_count_by_column[missing_val_count_by_column > 0])
```

```
(1168, 36)
LotFrontage    212
MasVnrArea       6
GarageYrBlt     58
dtype: int64
```

Tenemos 1168 entradas conocidas, divididas en 37 columnas. En adición, faltan 212 de la sección 'LotFrontage',6 de 'MasVnrArea' y 58 de 'GarageYrBlt'. ¿Qué porcentaje de nuestros datos suponen estos faltantes?

In [6]:
```
#Sumamos los datos y los dividimos entre en total
a=(212 + 6 + 58) / (212 + 6 +58 + 1168)
a
```

Out[6]:
```
0.1911357340720221
```

Métodos de sustitución de valores ausentes

1.Dado que los datos restantes no suponen una gran cantidad del total (por debajo del 20%) probaremos si la mejor forma de lidiar con ellos sería eliminar estas entradas.

In [7]:
```
# Llamamos a los datos que faltan
cols_with_missing = [col for col in X_train.columns
                     if X_train[col].isnull().any()]


# Eliminamos estas columnas en los DOS SET DE DATOS (No solo en uno) entrenamiento y validación.
reduced_X_train = X_train.drop(cols_with_missing, axis=1)
reduced_X_valid = X_valid.drop(cols_with_missing, axis=1)
```

In [38]:
```
print("MAE (Sin columnas de valores vacios):")
print(score_dataset(reduced_X_train, reduced_X_valid, y_train, y_valid))
```

```
MAE (Sin columnas de valores vacios):
17837.82570776256
```

2. Probamos el método de la sustitución de los valores restantes por valores medios de cada columna (Imputation)

In [9]:
```
from sklearn.impute import SimpleImputer

# Imputation
my_imputer = SimpleImputer()
imputed_X_train = pd.DataFrame(my_imputer.fit_transform(X_train))
imputed_X_valid = pd.DataFrame(my_imputer.transform(X_valid))

# Imputation: eliminación de valores ausentes; sustitución.
imputed_X_train.columns = X_train.columns
imputed_X_valid.columns = X_valid.columns
```

In [10]:
```
print("MAE (Imputation):")
print(score_dataset(imputed_X_train, imputed_X_valid, y_train, y_valid))
```

```
MAE (Imputation):
18062.894611872147
```

Los resultados arrojan la siguiente información: El segundo método (Imputation) funciona mejor que el primero (Dropping)

Construcción de modelo 'Random Forest' con mejoras introducidas

In [11]:
```
# Imputation
final_imputer = SimpleImputer(strategy='median')
final_X_train = pd.DataFrame(final_imputer.fit_transform(X_train))
final_X_valid = pd.DataFrame(final_imputer.transform(X_valid))

# Imputation: eliminación de valores ausentes; sustitución.
final_X_train.columns = X_train.columns
final_X_valid.columns = X_valid.columns
```

In [12]:
```
# Definimos y encajamos el modelo
model = RandomForestRegressor(n_estimators=100, random_state=0)
model.fit(final_X_train, y_train)

# Get validation predictions and MAE
preds_valid = model.predict(final_X_valid)
print("MAE (Acercamiento):")
print(mean_absolute_error(y_valid, preds_valid))
```

```
MAE (Acercamiento):
17791.59899543379
```

In [13]:
```
# Preprocesamos los datos test
final_X_test = pd.DataFrame (final_imputer.transform(X_test))

# OBtenemos predicciones del test
preds_test = model.predict(final_X_test)
preds_test
```

Out[13]:
```
array([125985.5 , 154599.5 , 180070.24, ..., 154751.44, 107387.  ,
       229281.55])
```

Guardamos los resultados y los enviamos a la competición.

In [39]:
```
# Save predictions in format used for competition scoring
output = pd.DataFrame({'Id': X_test.index,
                       'SalePrice': preds_test})
output.to_csv('submission.csv', index=False)
```

Orientación para continuar proceso de optimización: Variables categóricas

Para continuar con el proceso de optimización del modelo proponemos lidiar con las variables categóricas son aquellas variables que tan solo toman un número limitado de valores. (Como por ejemplo la frecuencia - 'nunca','a veces'... o marcas de algún producto - 'MSI', 'HP', 'ASUS'...)

Para lidiar con las variables categóricas de nuestro modelo, procederemos similarmente a como lidiamos anteriormente con los 'valores ausentes'. Para ello tenemos 3 maneras diferentes:

1. Eliminar las categorías variables (Drop categorical variables).
2. Label encoding (Programar etiquietas) lo cual consiste en asignar un número a cada valor de la variable categorial.
3. Hot encoding (Codificación en caliente) donde representamos con 1 y 0 la presencia/ausencia de estos valores.