Predicción del precio de una vivienda

Restb.ai challenge

Jordi Muñoz, Ismael El Basli y Oscar Garries

Datathon FME 2024

17 de noviembre de 2024

Contenido

restb.ai

Contenido

restb.ai

Introducción

Objetivo final: obtener un modelo capaz de predecir con precisión razonable los precios de una vivienda en base a ciertas características:

- **Image Data**: Data obtenida por Restb.ai a partir de únicamente imágenes.
- **Listing Data**: Data de parte de Midwest Real Estate Data, ofrece detalles específicos sobre la vivienda.
- **Census Data**: Geolocalización y censo ofrecidos por la API de US Census, si es que existen.

Contenido

restb.ai

Análisis de los datos

Al analizar nuestros datos hemos visto:

- Tenemos una **mezcla de variables categóricas y numéricas**, y algunas categóricas tienen un número de categorías muy elevado que dificultan su codificación.
- Hay bastantes variables con **missing values**, de los cuales hemos observado que son aleatorios ya que no siguen ninguna distribución reconocible.
- Hay **variables muy relacionadas**, que pueden representar la misma información y ser redundantes.

Missing values

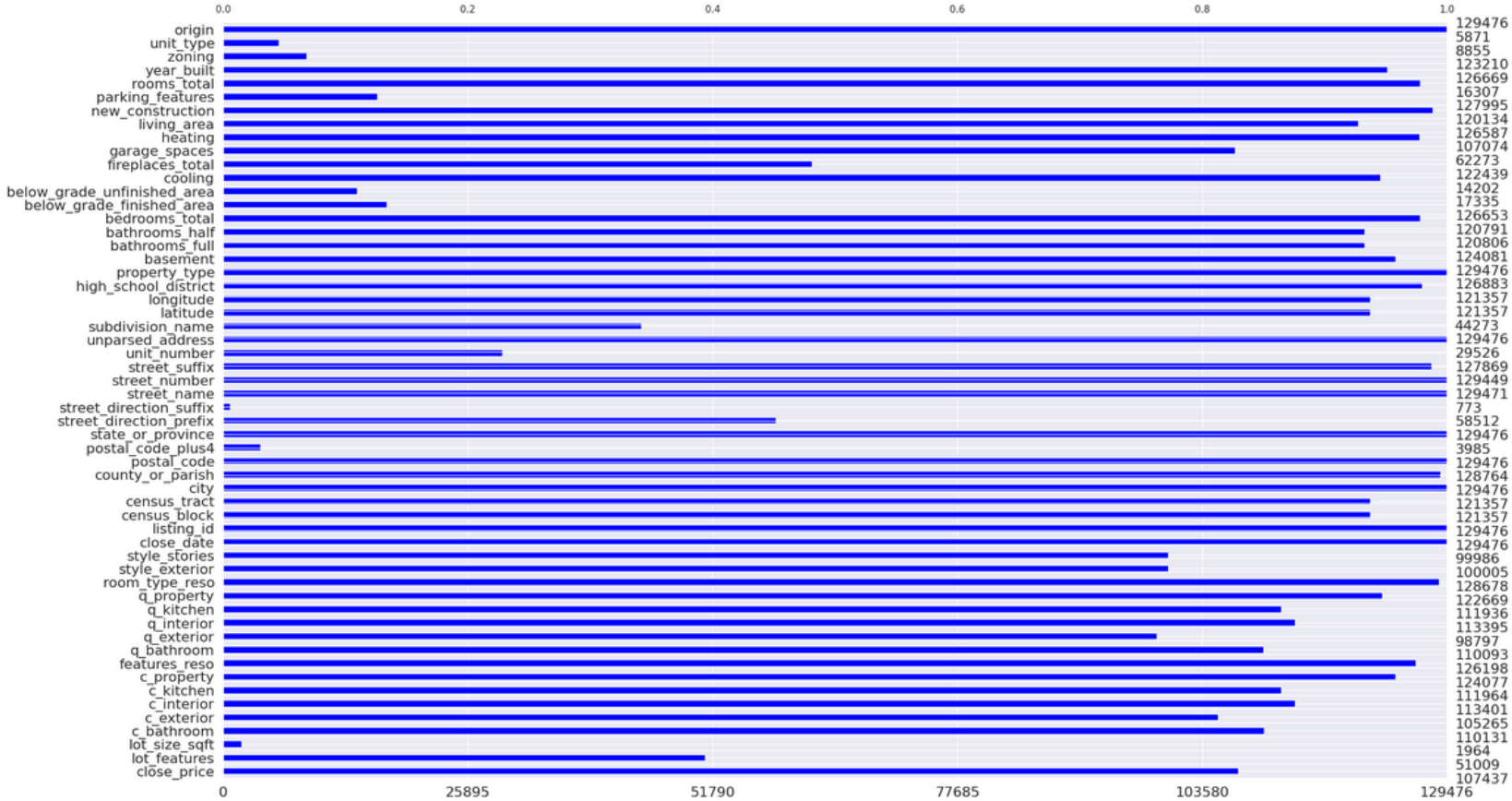


Figura: Missing values

Contenido

restb.ai

Pipeline de preprocesado

De cara a preprocesar los datos hemos seguido un pipeline específico:

1. Eliminación de variables irrelevantes
2. Eliminación de variables con demasiados missing values
3. Codificación de variables categóricas
4. Tratamiento de missing values

Eliminación de variables irrelevantes

Hemos considerado como variables irrelevantes las siguientes:

- **listing_id**: porque un identificador único por fila no aporta nada en la toma de decisiones.
- **unparsed_address, street_number y street_suffix**: porque son identificadores muy concretos que llevan a posibles sobreajustes, redundancia y no aportan información que consideremos razonable.
- **state_or_province**: porque siempre es Illinois.

Eliminación de variables irrelevantes

- **street_name, census_block, census_tract y high_school_district**: porque son variables categóricas que tienen muchas categorías posibles y, a parte de estar estrechamente relacionadas con otras, no aportan lo suficiente.
- **basement**: después de ejecutar un PCA hemos visto que no tiene un peso a considerar en cuanto a la explicación de la varianza de los datos, no es importante para la toma de decisiones.
- **city**: porque está estrechamente relacionada con otras variables geográficas, es redundante.

restb.ai

Eliminación de variables con demasiados missing values

En este punto hemos decidido eliminar todas aquellas variables que superen el treshold del 50 % de valores perdidos, ya que no tendría sentido imputar valores dada una pérdida de información tan elevada.

```
columns_with_many_missing = []

threshold = len(df_without_irrelevant_vars) * 0.5

for column in df_without_irrelevant_vars.columns:
    if df_without_irrelevant_vars[column].isna().sum() > threshold:
        columns_with_many_missing.append(column)

df_clean = df_without_irrelevant_vars.drop(columns=columns_with_many_missing, inplace=False)
```

Figura: Eliminación de variables con demasiados missings

Codificación de variables categóricas

Debemos decidir como codificar las variables categóricas. No siempre es trivial, usar de forma sistemática One-Hot encoding sin tener en cuenta el total de categorias posibles seria un error. En este punto tenemos las siguientes variables categóricas:

```
Tenemos 11 variables categoricas:

Column: features_reso: 112557
Column: room_type_reso: 109106
Column: style_exterior: 41
Column: style_stories: 5
Column: close_date: 412
Column: county_or_parish: 83
Column: postal_code: 1544
Column: property_type: 7
Column: cooling: 196
Column: heating: 769
Column: new_construction: 2
```

Figura: Variables categóricas: categorías posibles

restb.ai

Codificación de variables categóricas

Vemos como **features_reso y room_type_reso** tienen muchas categorías posibles $\rightarrow$ son características específicas de la vivienda, que aportan un valor añadido.

Después de habernos informado sobre estudios previos en este tipo de predicciones, hemos llegado a la conclusión de que lo ideal es, dado el array de features pertinente, codificar la variable como el size de dicho array.

Es decir, como mas "extras" hayan, mas grande será el número. Algo similar sucede con **cooling** y **heating**.

restb.ai

Codificación de variables categóricas

En este punto tenemos las siguientes variables categóricas:

```
Tenemos 7 variables categoricas:

Column: style_exterior: 41
Column: style_stories: 5
Column: close_date: 412
Column: county_or_parish: 83
Column: postal_code: 1544
Column: property_type: 7
Column: new_construction: 2
```

Figura: Variables categóricas: categorías posibles

Codificación de variables categóricas

Hemos decidido lo siguiente:

- **style_exterior**: vamos a hacer un procesado que agrupe los distintos tipos de vivienda, para posteriormente hacer un One-Hot encoding mas viable.
- **style_stories**: haremos un One-Hot encoding.
- **county_or_parish**: mapeo a una variable numérica relacionada con la renta media de la zona.
- **postal_code**: mapeo a una variable numérica relacionada con la renta media de la zona.

Codificación de variables categóricas

- **property_type**: haremos un One-Hot encoding.
- **new_construction**: es una categórica booleana, es trivial pasarla a numérica.
- **close_date**: agruparemos por estaciones, aportan información muy importante

Mapeo de postal_code y county_or_parish a renta media

Hemos considerado que podía ser buena idea realizar el mapeo, ya que es algo que puede tener un peso considerable en la toma de decisiones.

Para ello hemos usado *WebScrapping* para obtener los datos sobre renta por capita de wikipedia. Además, para poder relacionar los códigos postales con los condados y sus respectivos nombres, hemos utilizado la información de la página `zipdatamaps.com`.

Tratamiento de missing values

En resumen hemos decidido:

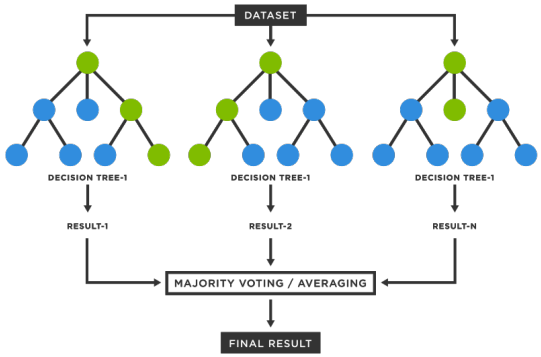
- Es una variable que tiene sentido y es posible imputar $\rightarrow$ imputar con KNNImputer de la libreria scikit-learn. Es importante destacar que para las imputaciones no hemos tenido en cuenta la variable objetivo, ya que estaríamos sesgando el resultado.
- En caso contrario $\rightarrow$ la eliminamos.

Contenido

restb.ai

Entrenamiento de modelos

Para entrenar modelos, hemos usado siempre validación cruzada de 5 folds sobre el conjunto de entrenamiento. Entrenar sin validación cruzada nos podría llevar a un sobreajuste y resultados no realistas. En base a estudios previos en el ámbito, hemos decido entrenar: **Random Forest** y **XGBoost**. Hemos probado con otros modelos pero no han mejorado.



dmlc XGBoost

restb.ai

Contenido

Resultados

Métrica	**Random Forest**	**XGBoost**
Training MSE	11,565,843,206.5	21,223,137,755.83
Training MAE	49,353.65	70,925.30
Training R^2	0.923	0.858

Cuadro: Resultados

Contenido

restb.ai

Conclusiones

- Para el tiempo que hemos tenido, consideramos que hemos tomado buenas decisiones en el preprocesado de los datos.
- Si tuviéramos mas tiempo haríamos mas *feature engineering*.
- Los modelos nos han dado resultados que consideramos bastante decentes, aunque obviamente hay margen de mejora.
- Hemos aprendido mucho, ya que ha sido la primera vez que nos hemos enfrentado a un dataset real.