Documentación - Entregable 2

Grupo 19 - Análisis y Curación de Datos

2 de julio de 2021

1. Introducción

Importamos el conjunto de datos obtenido al final de la primer parte de este entregable. Se lo guardo en la variable **melb_df**. Nuestro conjunto de datos contiene la siguiente información sobre las propiedades:

1.1. Características categóricas

- **Suburb** : El suburbio en Melbourne y sus alrededores.
- **Type** :
  - h - house,cottage,villa, semi,terrace;
  - u - unit, duplex;
  - t - townhouse;
- **Method** : Método de venta
  - **S** property sold;
  - **SP** property sold prior;
  - **PI** property passed in;
  - **VB** vendor bid;
  - **SA** sold after auction;
- **Postcode** : Código Postal.
- **CouncilArea** : Área administrativa
- **Regionname** : Región general

1.2. Características numéricas

- **Rooms** : La cantidad de habitaciones.
- **Price** : Precio en dólares.
- **Date** : Fecha de venta (en formato **string**)
- **Distance** : distancia al CBD (central business district).
- **Bathroom** : Cantidad de baños.
- **Car** : Cantidad de cocheras.
- **BuildingArea** : Superficie construida.
- **YearBuilt** : Año de construcción.
- **Lattitude** : Latitud

- **Longtitude** : Longitud
- **date** : Fecha de venta (en formato **datetime**)
- Las últimas 5 columnas se obtuvieron de un conjunto de datos de scrapings del sitio Airbnb realizado por Tyler Xie para una competencia de Kaggle. A partir del precio para alquilar por una noche una propiedad en Airbnb, y de su código postal, se obtuvieron estadísticos de esos precios de venta de las propiedades, previa agrupación según código postal.
  - **price count** : Cantidad de casas alquiladas al mismo precio
  - **price mean** : Media de los precios de las casas con el mismo código postal.
  - **price min** : Precio mínimo de las casas con el mismo código postal.
  - **price 50 %** : Mediana del conjunto de precios de alquiler de las casas con el mismo código postal.
  - **price max** : Precio mínimo de las casas con el mismo código postal.

2. Criterios de exclusión de columnas

Quitamos del data set a emplear las columnas: 'Date', 'date', 'Lattitude', 'Longtitude' debido a que las mismas no las empleamos luego. Todas las columnas categóricas se mantuvieron.

3. Criterios de exclusión de filas

Se mantuvieron todas las filas; debido a que el conjunto de datos del que partimos es resultado de un proceso previo en el Entregable 1, en el cual sí se aplicaron filtros sobre el conjunto de filas.

4. Tansformaciones

4.1. Características categóricas.

Reducimos las categorías de las columnas **Suburb** y **CouncilArea**, dejando aquellas con una frecuencia de al menos 250 en el primer caso y de 125 en el segundo. En ambos columnas; las categorías con frecuencia menor fueron agrupadas en la nueva categoría 'Other'. Así, las columnas categóricas quedaron con un máximo de 32 categorías distintas (para Suburb) y un mínimo de 3 categorías distintas (el caso de la columna Type).

4.2. Encoding

Se guardaron las columnas categóricas en la variable **melb_cat**, sobre la cual se aplicó una codificación One-hot encoding a cada fila, vía *OneHotEncoder* de la librería **sklearn.preprocessing**.

En la matriz esparsa **melb_matrix** se guardaron las columnas categóricas codificadas, y las columnas numéricas(excepto *YearBuilt* y *BuildingArea*).

4.3. Imputación por KNN.

Se realizarán imputaciones vía ***impute.IterativeImputer*** y ***neighbors.KNeighborsRegressor*** de sklearn. Se hará escalado vía ***preprocessing.MinMaxScaler*** de sklearn.

4.3.1. Matriz densa.

En **melb_df_numeric** se guardaron las columnas numéricas (incluyendo *YearBuilt* y *BuildingArea*) de **melb_df**.

A **melb_df_numeric** se lo escaló, se lo imputó y se lo desescaló. Se guardó el resultado en el DataFrame **melb_df_numeric_scaled**.

A **melb_df_numeric** se lo imputó (sin escalar/desescalar). Se guardó el resultado en el DataFrame **melb_df_numeric_scaled**.

4.3.2. Matriz esparsa.

A **melb_matrix** le agregamos *YearBuilt* y *BuildingArea*.

A **melb_matrix** se lo escaló, se lo imputó y se lo desescaló. Se guardó el resultado en el DataFrame **melb_matrix_scaled**.

A **melb_matrix** se lo imputó (sin escalar/desescalar). Se guardó el resultado en el DataFrame **melb_matrix_noscaled**.

En ambos casos, se utilizó el atributo *get_feature_names()* de OneHotEncoder para nombrar las columnas "categóricas".

Finalmente se hizo textbfkdeplot de seaborn para las columnas *YearBuilt* y *BuildingArea* comparando las 4 imputaciones.

5. Reducción de dimensionalidad.

Escalamos a melb_matrix y le aplicamos sklearn.decomposition.PCA (n_components=20).

Generamos dos gráficos de barra en donde en ambos el eje X son las 20 componentes principales. En el primero el eje Y es la **pca.explained_variance_ratio_** de cada componente, y en el segundo la **pca.explained_variance_ratio_** acumulada.

6. Composición del resultado.

Ya habíamos transformado **melb_matrix_scaled** en un DataFrame, ahora lo guardamos como el archivo csv textbfdf_melbourne_entregable2.csv.