Week3: ImmoEliza Dataset Analysis Make beautiful visualisation

Using pandas and data visualisation libraries (Matplotlib, Seaborn), let's establish conclusions about a dataset.

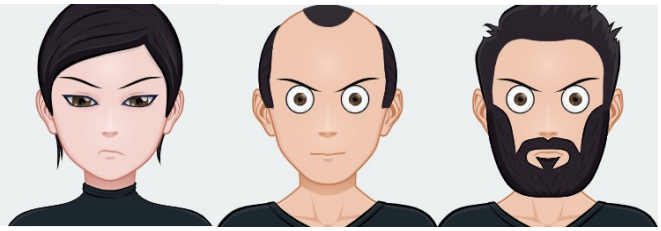
This week's summary

The week is: Cleaning and doing a complete charts contains interesting information

Description

- This project focuses on cleaning and preparing a real estate dataset for analysis by addressing missing values and detecting outliers.
- The dataset contains various property attributes such as construction year, room count, and more.
- To ensure the data's accuracy and reliability for subsequent analysis, we use K-Nearest Neighbors (KNN) imputation to fill missing values and apply statistical methods to identify and handle outliers.

Team member:



Ezgi → https://github.com/ezgitandogan

Mehmet → https://github.com/mehmetbatar35

Ziadoon → https://github.com/ziadoonAlobaidi/

Timeline

- **Day 1:** Data exploration and initial cleaning
- **Day 2-3:** Implement KNN imputation for missing values
- **Day 4:** Outlier detection and handling
- **Day 5:** Data Visualization and Analysis

Objectives:

File Information

- CSV File: `final_dataset.csv`
- Original JSON File: `final_dataset.json`

Installation

- To work with this project, you need to have Python installed along with the necessary libraries. Follow these steps to set up the environment:

1. **Clone the Repository:**

```
git clone https://github.com/mehmetbatar35/ImmoEliza_EDA.git
```

2. **Navigate to the Project Directory:**

```
cd ImmoEliza_EDA
```

3. **Install Required Libraries:**

   It is recommended to use a virtual environment. Install the required libraries

Usage

Cleaning dataset

> **Import the necessary libraries**

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

> **load the dataset**
> Ensure you have the dataset file (`final_dataset.csv`) in the project directory.

```
df = pd.read_csv('cleaned_dataset.csv')
#Transform column names to lowerCase to avoid name missmatching
df.columns = df.columns.str.lower()
```

> **Drop columns**

```
df.drop('url', axis = 1, inplace = True)
df.drop('gardenarea', axis = 1, inplace = True)
```

Fill null values with mean

```
df['bathroomcount'].fillna(mean_value, inplace = True)
```

4. **KNN Imputation:**

```
from sklearn.impute import KNNImputer

# Columns to be imputed
columns_for_imputation = ['roomcount', 'surfaceofplot',
'stateofbuilding', 'numberoffacades', 'livingarea', 'bedroomcount',
'bathroomcount', 'constructionyear', 'price', 'kitchen']

# Select only the columns to impute
df_impute = df[columns_to_impute].copy()6

# Convert categorical columns to numerical if necessary
df_impute['stateofbuilding'] =
df_impute['stateofbuilding'].astype('category').cat.codes
df_impute['kitchen'] = df_impute['kitchen'].astype('category').cat.codes

# Initialize the KNN Imputer
knn_imputer = KNNImputer(n_neighbors=5)

# Impute missing values
 df_imputed =
pd.DataFrame(knn_imputer.fit_transform(df[columns_for_imputation]),
 columns=columns_for_imputation)


# Round and map imputed categorical values
 df_imputed['stateofbuilding'] =
np.round(df_imputed['stateofbuilding']).astype(int)
 df_imputed['stateofbuilding'] =
df_imputed['stateofbuilding'].apply(lambda x: x if x in
 state_mapping.values() else np.nan)
 df['stateofbuilding'] = df_imputed['stateofbuilding'].values

 df_imputed['kitchen'] = np.round(df_imputed['kitchen']).astype(int)
 df_imputed['kitchen'] = df_imputed['kitchen'].apply(lambda x: x if x in
 kitchen_new_mapping.values() else np.nan)
 df['kitchen'] = df_imputed['kitchen'].values

# Update original DataFrame with imputed values
 df['constructionyear'] = df_imputed['constructionyear'].values
 df['roomcount'] = df_imputed['roomcount'].values
 df['surfaceofplot'] = df_imputed['surfaceofplot'].values
 df['numberoffacades'] = df_imputed['numberoffacades'].values
 df['livingarea'] = df_imputed['livingarea'].values
```

5. **Outlier Detection and Handling:**

- Outliers can significantly skew analysis and models. The following steps demonstrate how to detect and handle outliers, focusing on the `price` column for `residential_sale` properties.

    - **Filter Data by Sale Type:**

```
df_sales = df[df['typeofsale'] == 'residential_sale']
df_rent = df[df['typeofsale'] == 'residential_monthly_rent']
```

    - **Calculate Outlier Bounds for `price`:**

```
Q1 = df_sales['price'].quantile(0.15)
Q3 = df_sales['price'].quantile(0.9)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
```

    - **Filter Out Outliers:**

```
df_sales = df_sales[(df_sales['price'] > lower_bound) &
(df_sales['price'] < upper_bound)]
```

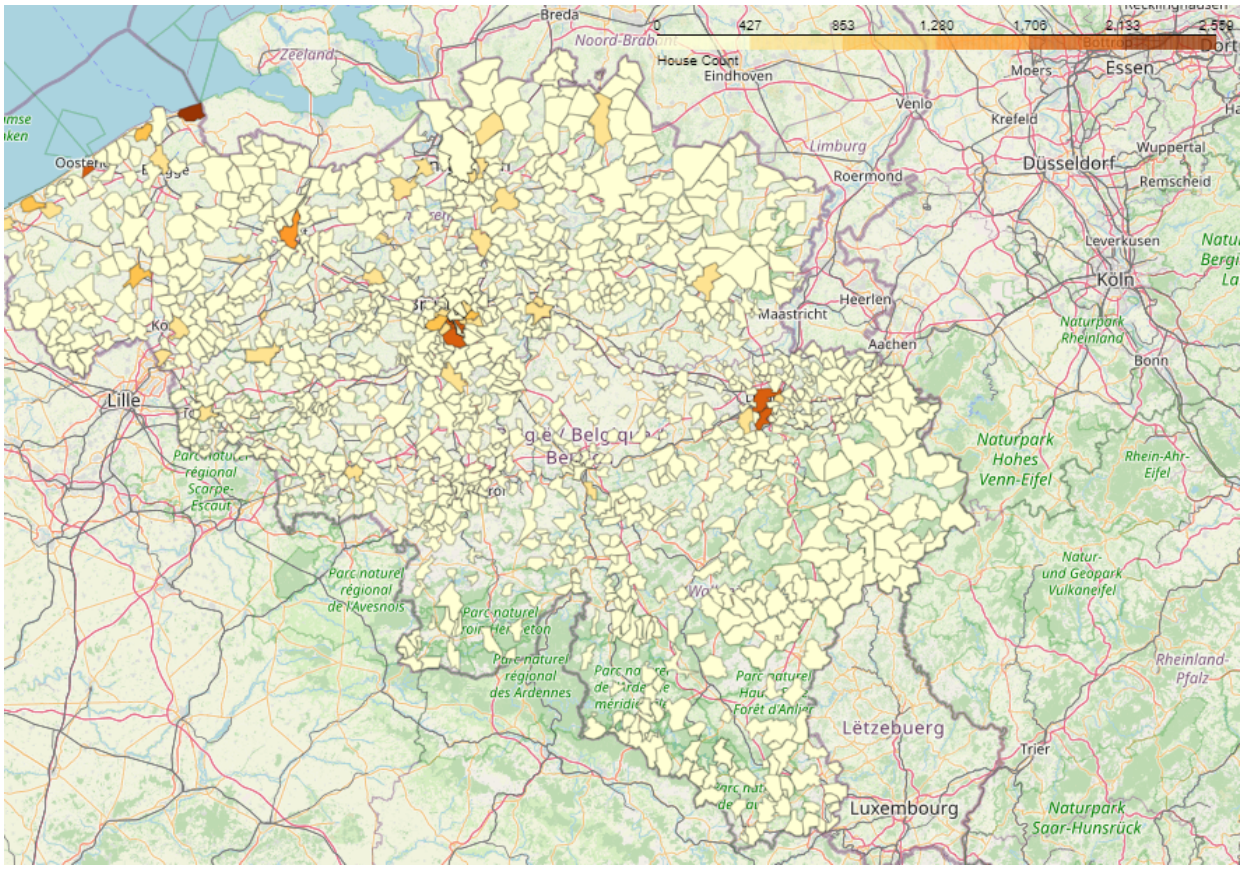
5. **Correlation**
   To find the most correlated column with the price :

```
correlation = df.select_dtypes(include = ['float64', 'int64',
'int32']).corr()

price_correlation  = correlation.price.drop('price')

most_correlated_column = price_correlation.idxmax()
```

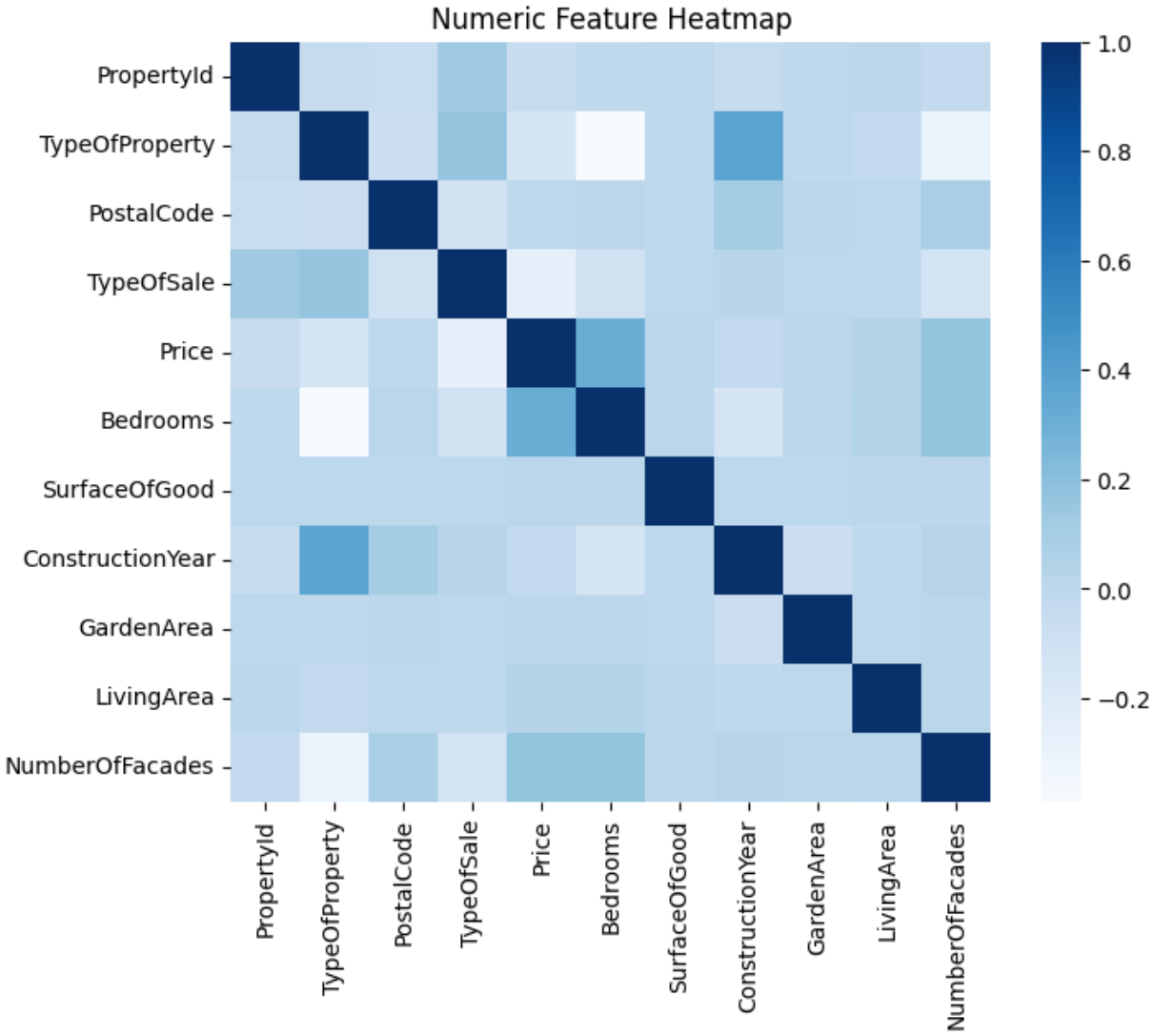
   By executing this code, the most correlated column is living area and then the bedrooms count

Visuals

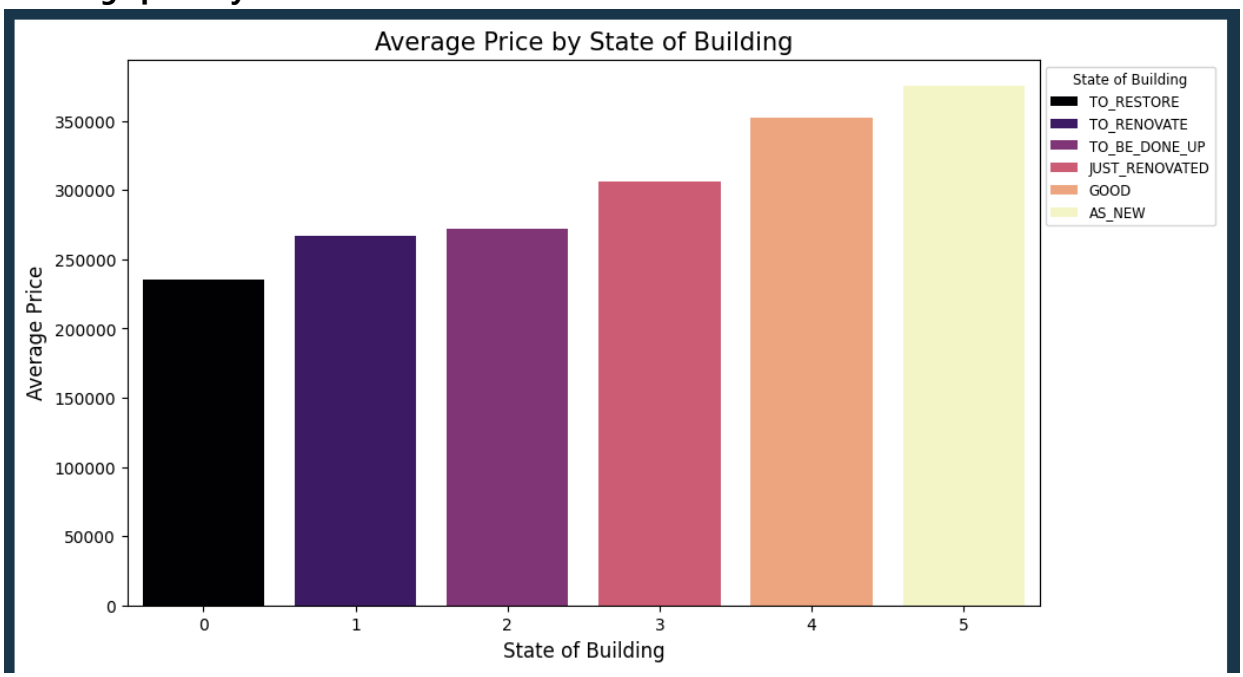
- Here are some example visualizations that can help in understanding the dataset and the effects of imputation and outlier removal:
  1.**Number of houses per Area**



2.**Heatmap showing the correlation between all data**

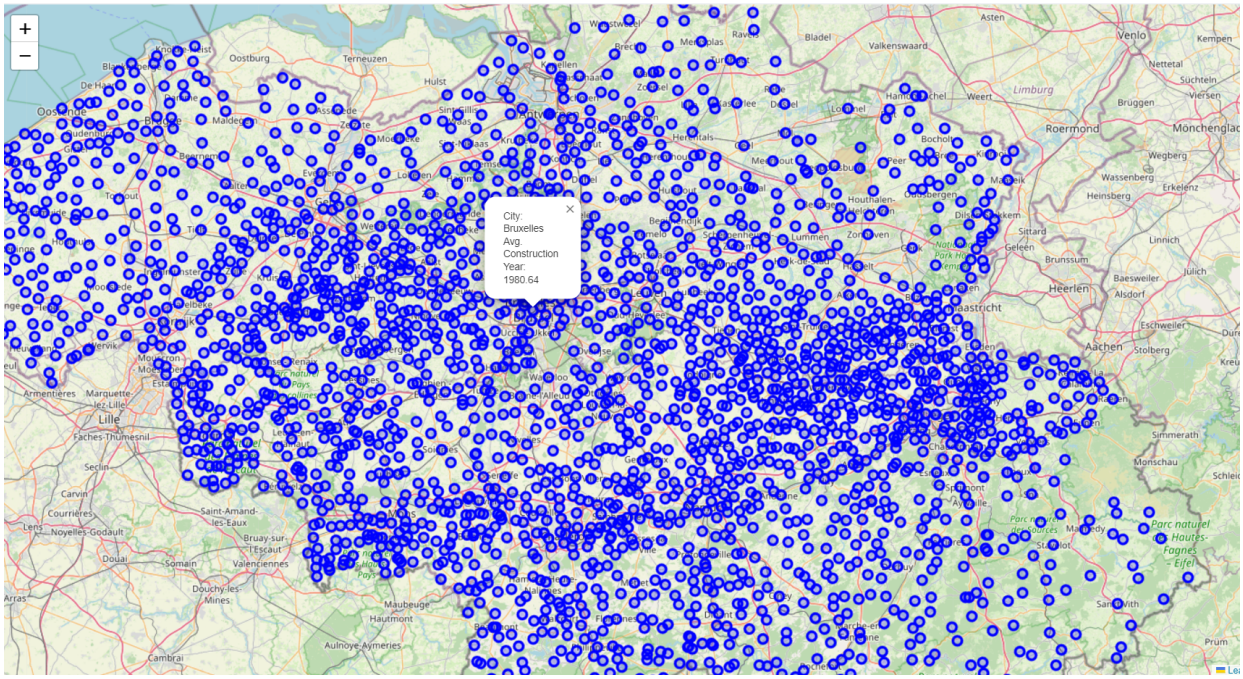


3.**Average price by state**



4.**Average construction year per zip code**

Here is a quick introduction to our work: A map of all belgian's municipalities with their average construction year.



5.**Average price by construction year**