Data Wrangling on Real Estate Market

July 24, 2024

[4]:
```
import pandas as pd
import numpy as np
from sklearn.preprocessing import OneHotEncoder, LabelEncoder

# 1. Import the dataset and clean column names
file_path = "C:\\Users\\MUBASHIR KHAN\\Desktop\\jupyter\\DMV\\Real estate.csv"
df = pd.read_csv(file_path)

# Clean column names by removing spaces and special characters, and renaming␣
 ↪for clarity
df.columns = df.columns.str.replace(' ', '_').str.replace(r'[^\w]', '',␣
 ↪regex=True)
print("Cleaned Column Names:\n", df.columns)

# 2. Handle missing values
# Checking for missing values
print("\nMissing Values:\n", df.isnull().sum())

# Filling missing values or dropping
# Example: If missing values are found in a column, fill with median or drop␣
 ↪rows/columns
numeric_columns = df.select_dtypes(include=[np.number]).columns
df[numeric_columns] = df[numeric_columns].fillna(df[numeric_columns].median())

# For categorical columns, we can fill missing values with the mode
categorical_columns = df.select_dtypes(include=[object]).columns
for col in categorical_columns:
    df[col].fillna(df[col].mode()[0], inplace=True)

# 3. Perform data merging (assuming additional datasets are available,␣
 ↪otherwise skipping this step)
# Example: Merging with another dataset containing neighborhood demographics
# additional_df = pd.read_csv("path_to_additional_dataset.csv")
# df = pd.merge(df, additional_df, on="common_column", how="left")

# 4. Filter and subset the data
```

```
# Example: Filtering data for properties sold after 2010 if 'SaleDate' column␣
 ↪exists
if 'SaleDate' in df.columns:
    df['SaleDate'] = pd.to_datetime(df['SaleDate'])
    filtered_df = df[df['SaleDate'] > '2010-01-01']
else:
    filtered_df = df

# Example: Filtering for specific property types if 'PropertyType' column exists
if 'PropertyType' in filtered_df.columns:
    property_types = ['Single Family', 'Condo']
    filtered_df = filtered_df[filtered_df['PropertyType'].isin(property_types)]

# 5. Handle categorical variables by encoding
# Example: One-hot encoding for 'Neighborhood' column if it exists
if 'Neighborhood' in filtered_df.columns:
    encoded_df = pd.get_dummies(filtered_df, columns=['Neighborhood'],␣
 ↪drop_first=True)
else:
    encoded_df = filtered_df

# Example: Label encoding for 'PropertyType' if it exists
if 'PropertyType' in encoded_df.columns:
    label_encoder = LabelEncoder()
    encoded_df['PropertyType'] = label_encoder.
 ↪fit_transform(encoded_df['PropertyType'])

# 6. Aggregate the data to calculate summary statistics
# Example: Average sale price by neighborhood if 'Neighborhood' and 'SalePrice'␣
 ↪columns exist
if 'Neighborhood' in encoded_df.columns and 'SalePrice' in encoded_df.columns:
    average_price_by_neighborhood = encoded_df.
 ↪groupby('Neighborhood')['SalePrice'].mean().reset_index()
    average_price_by_neighborhood.columns = ['Neighborhood', 'AverageSalePrice']
    print("\nAverage Sale Price by Neighborhood:\n",␣
 ↪average_price_by_neighborhood)

# 7. Identify and handle outliers
# Example: Using IQR to handle outliers in 'SalePrice' column if it exists
if 'SalePrice' in encoded_df.columns:
    Q1 = encoded_df['SalePrice'].quantile(0.25)
    Q3 = encoded_df['SalePrice'].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
```

```
    filtered_df = encoded_df[(encoded_df['SalePrice'] >= lower_bound) &␣
 ↪(encoded_df['SalePrice'] <= upper_bound)]

# Export the cleaned and wrangled dataset for further analysis or modeling
cleaned_file_path = "C:\\Users\\MUBASHIR␣
 ↪KHAN\\Desktop\\jupyter\\DMV\\Cleaned_RealEstate_Prices.csv"
filtered_df.to_csv(cleaned_file_path, index=False)

print(f"Cleaned and wrangled dataset saved to {cleaned_file_path}")
```

```
Cleaned Column Names:
 Index(['No', 'X1_transaction_date', 'X2_house_age',
       'X3_distance_to_the_nearest_MRT_station',
       'X4_number_of_convenience_stores', 'X5_latitude', 'X6_longitude',
       'Y_house_price_of_unit_area'],
      dtype='object')

Missing Values:
 No                                      0
X1_transaction_date                     0
X2_house_age                            0
X3_distance_to_the_nearest_MRT_station  0
X4_number_of_convenience_stores         0
X5_latitude                             0
X6_longitude                            0
Y_house_price_of_unit_area              0
dtype: int64
Cleaned and wrangled dataset saved to C:\Users\MUBASHIR
KHAN\Desktop\jupyter\DMV\Cleaned_RealEstate_Prices.csv
```

[ ]: