

Correlation Analysis Between Airbnb Prices & Subway Distance

Gangyi Li
12/04/2025

Issues and Objectives

- **Core issue:** Are Airbnb prices really higher when you are closer to the subway?
- **Code:** Multi-library collaboration (pandas/geopandas/sklearn/folium)
- **Process:** data → modeling → visualization
- **Outcome:** Python implementation for spatial data processing and regression analysis.

Data Source

Airbnb Listing Data:

Content: 20,000+ NYC records in CSV/GZ format

Python Implementation: `pd.read_csv()` with native compression support

Subway Station Data:

Content: Latitude, Longitude, Station Name

Python Implementation: Custom `read_subway()` function for automatic column name detection

Neighborhood Income Data:

Content: Sourced from Census API (Population, Median Income)

Python Implementation: Data fetched via `requests` and parsed into a `pd.DataFrame()`

Analysis and Models

Data Cleaning

Remove Missing Values:

```
df_airbnb = df_airbnb.dropna(subset=['latitude', 'longitude', 'price'])
```

Standardize Parameters:

```
df_airbnb['price'] = df_airbnb['price'].apply(parse_price)
```

Spatial Format Conversion:

```
gdf_airbnb = gpd.GeoDataFrame(  
    df_airbnb,  
    geometry=gpd.points_from_xy(df_airbnb.longitude, df_airbnb.latitude),  
    crs="EPSG:4326"  
)
```

Analysis and Models

Spatial Analysis

Use KMeans to cluster properties with similar geographic locations into 5 categories:

```
coords = pd.DataFrame({'x': gdf_airbnb_m.geometry.x, 'y': gdf_airbnb_m.geometry.y})  
gdf_airbnb_with_tract['cluster'] = KMeans(n_clusters=5, random_state=42).fit_predict(coords)
```

Analysis and Models

Regression Modeling

```
X = df_model[['dist_km', 'median_household_income']]  
reg = LinearRegression().fit(X, np.log1p(df_model['price']))
```

Why `np.log1p(y)`? Reduce impact of \$1000+ luxury listings outliers

Outputs: Coefficients (impact strength) + R^2 (model fit)

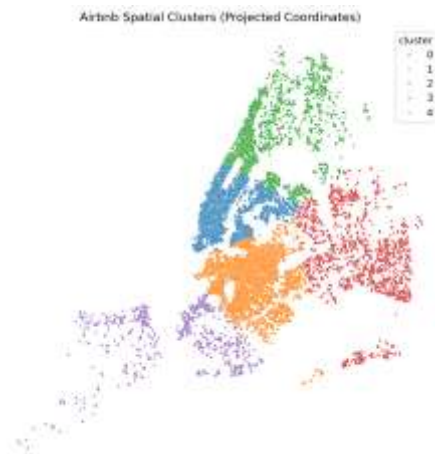
Content: Cleaned data (including distance, clustering, and community attributes)

| Year | Age | Sex | Weight (kg) | Height (cm) | Body mass index (kg/m ²) | Heart rate (b/min) | Stroke volume (L) | Cardiac output (L/min) | Stroke volume index (L/m ²) | Cardiac output index (L/min/m ²) |
|------|-----|-----|-------------|-------------|--------------------------------------|--------------------|-------------------|------------------------|---|--|
| 1990 | 20 | M | 70 | 175 | 22.6 | 70 | 1.0 | 7.0 | 0.06 | 0.50 |
| 1991 | 21 | F | 55 | 160 | 21.5 | 65 | 0.9 | 5.8 | 0.05 | 0.45 |
| 1992 | 22 | M | 80 | 180 | 24.7 | 75 | 1.1 | 8.2 | 0.07 | 0.55 |
| 1993 | 23 | F | 60 | 165 | 22.0 | 70 | 1.0 | 7.0 | 0.06 | 0.50 |
| 1994 | 24 | M | 90 | 190 | 25.3 | 80 | 1.2 | 9.6 | 0.08 | 0.60 |
| 1995 | 25 | F | 70 | 170 | 24.1 | 75 | 1.1 | 8.2 | 0.07 | 0.55 |
| 1996 | 26 | M | 100 | 200 | 25.0 | 85 | 1.3 | 10.8 | 0.09 | 0.65 |
| 1997 | 27 | F | 80 | 180 | 24.7 | 80 | 1.2 | 9.6 | 0.08 | 0.60 |
| 1998 | 28 | M | 110 | 210 | 24.8 | 90 | 1.4 | 12.6 | 0.10 | 0.70 |
| 1999 | 29 | F | 90 | 190 | 25.3 | 85 | 1.3 | 10.8 | 0.09 | 0.65 |
| 2000 | 30 | M | 120 | 220 | 24.5 | 95 | 1.5 | 14.2 | 0.11 | 0.75 |
| 2001 | 31 | F | 100 | 200 | 25.0 | 90 | 1.4 | 12.6 | 0.10 | 0.70 |
| 2002 | 32 | M | 130 | 230 | 24.8 | 100 | 1.6 | 16.0 | 0.12 | 0.80 |
| 2003 | 33 | F | 110 | 210 | 24.8 | 95 | 1.5 | 14.2 | 0.11 | 0.75 |
| 2004 | 34 | M | 140 | 240 | 25.0 | 105 | 1.7 | 18.2 | 0.13 | 0.85 |
| 2005 | 35 | F | 120 | 220 | 24.5 | 100 | 1.6 | 16.0 | 0.12 | 0.80 |
| 2006 | 36 | M | 150 | 250 | 24.0 | 110 | 1.8 | 20.4 | 0.14 | 0.90 |
| 2007 | 37 | F | 130 | 230 | 24.8 | 105 | 1.7 | 18.2 | 0.13 | 0.85 |
| 2008 | 38 | M | 160 | 260 | 23.9 | 115 | 1.9 | 22.8 | 0.15 | 0.95 |
| 2009 | 39 | F | 140 | 240 | 25.0 | 110 | 1.8 | 20.4 | 0.14 | 0.90 |
| 2010 | 40 | M | 170 | 270 | 23.0 | 120 | 2.0 | 25.0 | 0.16 | 1.00 |
| 2011 | 41 | F | 150 | 250 | 24.0 | 115 | 1.9 | 22.8 | 0.15 | 0.95 |
| 2012 | 42 | M | 180 | 280 | 23.0 | 125 | 2.1 | 28.2 | 0.17 | 1.05 |
| 2013 | 43 | F | 160 | 260 | 23.9 | 120 | 2.0 | 25.0 | 0.16 | 1.00 |
| 2014 | 44 | M | 190 | 290 | 23.0 | 130 | 2.2 | 30.6 | 0.18 | 1.10 |
| 2015 | 45 | F | 170 | 270 | 23.0 | 125 | 2.1 | 28.2 | 0.17 | 1.05 |
| 2016 | 46 | M | 200 | 300 | 22.2 | 135 | 2.3 | 33.0 | 0.19 | 1.15 |
| 2017 | 47 | F | 180 | 280 | 23.0 | 130 | 2.2 | 30.6 | 0.18 | 1.10 |
| 2018 | 48 | M | 210 | 310 | 21.9 | 140 | 2.4 | 36.0 | 0.20 | 1.20 |
| 2019 | 49 | F | 190 | 290 | 23.0 | 135 | 2.3 | 33.0 | 0.19 | 1.15 |
| 2020 | 50 | M | 220 | 320 | 21.5 | 145 | 2.5 | 40.0 | 0.21 | 1.25 |
| 2021 | 51 | F | 200 | 300 | 22.2 | 140 | 2.4 | 36.0 | 0.20 | 1.20 |
| 2022 | 52 | M | 230 | 330 | 21.0 | 150 | 2.6 | 45.0 | 0.22 | 1.30 |
| 2023 | 53 | F | 210 | 310 | 21.9 | 145 | 2.5 | 40.0 | 0.21 | 1.25 |
| 2024 | 54 | M | 240 | 340 | 20.6 | 155 | 2.7 | 50.4 | 0.23 | 1.35 |
| 2025 | 55 | F | 220 | 320 | 21.5 | 150 | 2.6 | 45.0 | 0.22 | 1.30 |
| 2026 | 56 | M | 250 | 350 | 20.0 | 160 | 2.8 | 56.0 | 0.24 | 1.40 |
| 2027 | 57 | F | 230 | 330 | 21.0 | 155 | 2.7 | 50.4 | 0.23 | 1.35 |
| 2028 | 58 | M | 260 | 360 | 19.4 | 165 | 2.9 | 63.0 | 0.25 | 1.45 |
| 2029 | 59 | F | 240 | 340 | 20.6 | 160 | 2.8 | 56.0 | 0.24 | 1.40 |
| 2030 | 60 | M | 270 | 370 | 19.0 | 170 | 3.0 | 70.2 | 0.26 | 1.50 |
| 2031 | 61 | F | 250 | 350 | 20.0 | 165 | 2.9 | 63.0 | 0.25 | 1. |

Content: Interactive Maps
(Property Listings, Subway Lines,
Income Heat Maps)

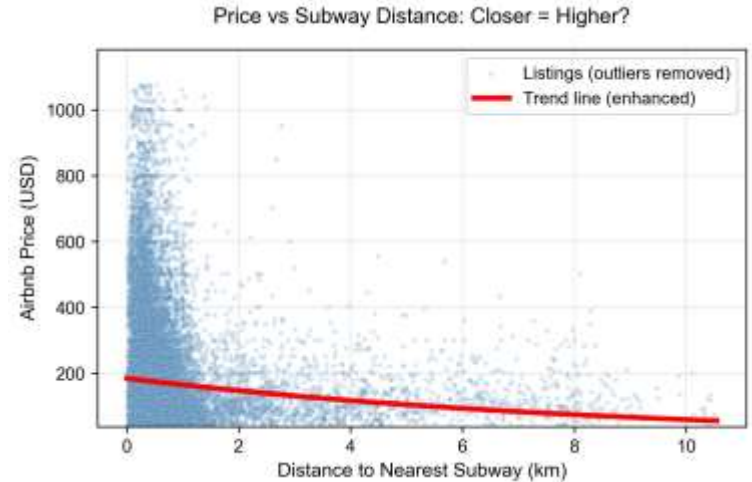
Content: Spatial cluster results (5 groups)

Python Implementation: Clustered via **sklearn.KMeans**, visualized with seaborn.



Result

- The spatial autocorrelation index Moran's I is 0.035 ($p=0.026$) : Weak but significant price clustering.
- In the regression model, the coefficient for subway distance is -0.05: 5% price drop per 1km.
- Model fit $R^2 = 0.45$: 45% price variation explained (distance & income)



- The red trendline illustrates the pattern that 'the farther from the subway, the lower the housing prices,' controlling for confounding factors such as community income.

Challenge

Data Format Inconsistency:

- The data contains numerous mixed string formats (e.g., price information: \$100, \$200, “350”, etc.).
- `def parse_price(x): s = str(x).replace('$', '').replace(',', '').strip()`

Outlier Impact on Regression:

- High-priced properties distort the results.
- `y = np.log1p(df_model['price'])`