The Impact of Short-Term Rentals on Long-Term Housing Market in New York City

Ruiyang Wang

University of Toronto, St. George Campus

March 2025

Project 1

1.1 Introduction

Airbnb is a peer-to-peer platform allowing individuals to rent out properties or rooms for short-term stays. Consumers favor short-term rental (STR) platforms like Airbnb for their flexibility and cost-effectiveness compared to hotels. While STRs reduce market friction and improve housing resource utilization, they also cause local market fluctuations and negative externalities (Filippas and Horton 2023).

Economic theories suggest STRs affect residential housing markets. Data from FRED shows the median U.S. house price increased from 165,300 dollars in Q1 2000 to 419,200 dollars in Q4 2024 (FRED 2025). Barron, Kung, and Proserpio (2019) found that a 10% rise in Airbnb listings leads to a 0.76% increase in house prices. In Virginia, STRs like Airbnb removed between 7,000 and 13,500 units from New York City's long-term housing market, impacting high-priced housing (Wachsmuth et al. 2018). Filippas and Horton (2023) also note that high Airbnb density causes noise pollution, reducing housing prices (Trojanek 2023). In London, a 10% increase in Airbnb listings is linked to a 3% rise in burglaries and five additional robberies per 100 properties (Lanfear 2024). These externalities harm local housing markets (Filippas and Horton 2023).

This study initially introduces the Airbnb-Housing Ratio (AHR) as a preliminary metric to explore potential relationships between short-term rental activity and housing market trends. However, AHR is not used in the core statistical models due to its sensitivity to local sample sizes and aggregation levels. It primarily serves an exploratory role to identify borough-level patterns. For more rigorous analysis, the study relies on short-term rental (STR) density—defined as the number of Airbnb

listings per square kilometer—which emerges as the most significant and consistent metric throughout both regression and machine learning sections.

New York City serves as a case study due to its strong tourism industry and high short-term rental demand (Wachsmuth et al. 2018). The analysis integrates the USA Real Estate Dataset (Sakib n.d.) and the New York City Airbnb Open Data (Gomonov n.d.), containing housing prices and 2019 Airbnb listings, respectively. The datasets were cleaned and merged using Python and Jupyter Notebook.

Key variables are summarized through tables and visualizations. Violin plots compare pre- and post-2019 New York housing counts, while the Airbnb-Housing Ratio examines housing price trends. Although AHR-based plots provide preliminary observations, they do not reveal a stable or statistically supported link between Airbnb concentration and price growth. In summary, plots do not reveal a clear relationship between the Airbnb-Housing Ratio and housing price growth. Additionally, some graphs suggest that higher Airbnb-Housing Ratios are associated with greater long-term housing price variability in certain boroughs. Geographical maps show that Manhattan generally has higher STR density, median prices over the years, and price volatility, followed by Queens and Brooklyn. To explore these relationships more rigorously, the paper introduced multiple OLS regression models incorporating STR density, price volatility, and structural housing features like bed and bath counts. The results confirmed a statistically significant and positive association between Airbnb density and both housing price and price variability, particularly after including fixed effects by borough. Additionally, geographical visualizations reinforced this result by highlighting spatial clustering of STR intensity and housing volatility.

To supplement the OLS analysis, machine learning models such as regression trees and random forests were applied to uncover non-linear relationships and variable interactions. These models highlighted pattern thresholds and ranked STR density as the most influential factor in predicting price, providing a more flexible view of housing dynamics. Compared to OLS, which offers interpretable coefficients and statistical inference, machine learning emphasizes predictive accuracy and structural complexity, thus uncovering effects that may not be apparent in OLS.

This combined strategy allows the paper to balance statistical rigor with pattern discovery, strengthening the case for STR density as a meaningful factor in the housing market.

1.2 Data Cleaning/Loading

In [107…

```
### Clean & filtering for real estate data(1)
import pandas as pd
import os

# Check if the file exists before loading
file_path = r"C:\Users\User\Desktop\ECO225\Datasets\realtor-data.zip.csv"
if os.path.exists(file_path):
    # Load the dataset
    df = pd.read_csv(file_path)

    # Remove all rows with missing values
    df_cleaned = df.dropna()

    # Save the cleaned dataset
    cleaned_file_path = r"C:\Users\User\Desktop\ECO225\Datasets\realtor-data-cle
    df_cleaned.to_csv(cleaned_file_path, index=False)

else:
    print("Error: File not found. Please check the file path.")

# Define the cleaned file path
cleaned_file_path = r"C:\Users\User\Desktop\ECO225\Datasets\realtor-data-cleaned

# Cleaning for US real estate data (2)
# Load the cleaned dataset
try:
    df = pd.read_csv(cleaned_file_path, parse_dates=['prev_sold_date'], dayfirst
except FileNotFoundError:
    print("Error: Cleaned file not found. Please check the file path.")
    exit()

# Step 1: Remove rows where price is less than a threshold (e.g., remove prices
df_cleaned = df[df['price'] > 2]

# Step 2: Filter for only the five NYC boroughs
nyc_boroughs = ["Bronx", "Brooklyn", "Manhattan", "Queens", "Staten Island"]
ny_housing_data = df_cleaned[df_cleaned['city'].isin(nyc_boroughs)]

# Step 3: Filter for the time range in 'prev_sold_date'
ny_housing_filtered = ny_housing_data[
    (ny_housing_data['prev_sold_date'] >= '2014-01-01') &
    (ny_housing_data['prev_sold_date'] <= '2024-12-31')
]

# Step 4: Save the filtered dataset to a new file
nyhousing_file_path = r"C:\Users\User\Desktop\ECO225\Datasets\NYChousing.csv"
ny_housing_filtered.to_csv(nyhousing_file_path, index=False)
```

In [207…

```
### Data cleaning/filtering for airbnb dataset
import pandas as pd
import warnings
warnings.filterwarnings("ignore")  # Suppress all warnings
# File path
file_path = r"C:\Users\User\Desktop\ECO225\Datasets\AB_NYC_2019.csv"

# Load the dataset with appropriate encoding
airbnb_data = pd.read_csv(file_path, encoding='latin1')

# Select only the specified columns
```

```
selected_columns = ['id', 'name', 'host_id', 'neighbourhood_group', 'neighbourho
filtered_data = airbnb_data[selected_columns]

# Remove rows with NA values in the selected columns
cleaned_data = filtered_data.dropna()

# Save the cleaned dataset to a new CSV file
output_path = r"C:\Users\User\Desktop\ECO225\Datasets\AB_NYC_2019_Cleaned.csv"
cleaned_data.to_csv(output_path, index=False)
```

In [109…

```
### Merging datasets
import pandas as pd

# Load datasets
housing_data = pd.read_csv(r"C:\Users\User\Desktop\ECO225\Datasets\new_york_data
rental_data = pd.read_csv(r"C:\Users\User\Desktop\ECO225\Datasets\AB_NYC_2019_Cl

# Rename 'neighbourhood_group' to 'city' in the rental dataset
rental_data.rename(columns={"neighbourhood_group": "city"}, inplace=True)

# Merge the datasets using an outer join on 'city'
merged_data = pd.merge(housing_data, rental_data, how="outer", on="city")

# Save the merged dataset to a new CSV file
merged_data.to_csv(r"C:\Users\User\Desktop\ECO225\Datasets\merged_data.csv", ind
```

In [211…

```
### calculating the  Airbnb-to-Housing Ratio
import pandas as pd

# Load the Airbnb dataset
airbnb_path = r"C:\Users\User\Desktop\ECO225\Datasets\AB_NYC_2019_Cleaned.csv"
airbnb_data = pd.read_csv(airbnb_path)

# Load the NY housing dataset
housing_path = r"C:\Users\User\Desktop\ECO225\Datasets\NYChousing.csv"
housing_data = pd.read_csv(housing_path)

# Step 1: Count the number of Airbnb listings per city
airbnb_counts = airbnb_data.groupby('neighbourhood_group').size().reset_index(na

# Step 2: Count the number of housing units per city
housing_counts = housing_data.groupby('city').size().reset_index(name='housing_c

# Step 3: Merge the counts based on the city
city_data = pd.merge(airbnb_counts, housing_counts, left_on='neighbourhood_group

# Step 4: Calculate the Airbnb-to-Housing Ratio
city_data['airbnb_to_housing_ratio'] = city_data['airbnb_count'] / city_data['ho

# Step 5: Save the result to a CSV file
output_path = r"C:\Users\User\Desktop\ECO225\Datasets\airbnb_to_housing_ratio.cs
city_data.to_csv(output_path, index=False)

### Summary table for the ratio
import pandas as pd

# Load the merged dataset (with the ratio)
merged_data_path = r"C:\Users\User\Desktop\ECO225\Datasets\airbnb_to_housing_rat
merged_data = pd.read_csv(merged_data_path)
```

```
# Calculate summary statistics
summary_stats = merged_data[['airbnb_to_housing_ratio']].describe().T

# Add additional metrics like standard deviation (if not included in describe())
summary_stats['std'] = merged_data['airbnb_to_housing_ratio'].std()

# Rename columns for better readability
summary_stats = summary_stats.rename(columns={
    'mean': 'Mean',
    'std': 'Standard Deviation',
    'min': 'Minimum',
    '25%': '25th Percentile',
    '50%': 'Median',
    '75%': '75th Percentile',
    'max': 'Maximum'
})
```

In [ ]:

```
### Create 3 new variables(Airbnb density, median price(2014-2022), and price vo
import pandas as pd
import numpy as np
import geopandas as gpd

# Load the NYChousing dataset
housing_file = r"C:\Users\User\Desktop\ECO225\Datasets\NYChousing.csv"
housing_df = pd.read_csv(housing_file)

# Load the Airbnb listings dataset
airbnb_file = r"C:\Users\User\Desktop\ECO225\Datasets\AB_NYC_2019_Cleaned.csv"
airbnb_df = pd.read_csv(airbnb_file)

# Load the NYC borough shapefile
shapefile_path = r"C:\Users\User\Desktop\ECO225\Project 2\NYC boundaries\nybb.sh
boroughs = gpd.read_file(shapefile_path)
# Convert 'prev_sold_date' to datetime format
housing_df["prev_sold_date"] = pd.to_datetime(housing_df["prev_sold_date"], erro

# Extract year from 'prev_sold_date'
housing_df["year"] = housing_df["prev_sold_date"].dt.year

# Keep only data from 2014 to 2022
years_to_keep = list(range(2014, 2023))
housing_df_filtered = housing_df[housing_df["year"].isin(years_to_keep)]

# Compute median home price per borough for each year
median_price_by_year = housing_df_filtered.groupby(["year", "city"])["price"].me

# Pivot the dataframe to have boroughs as rows and years as columns
median_price_pivot = median_price_by_year.pivot(index="city", columns="year", va

# Rename columns to match "median_price_YYYY" format
median_price_pivot.columns = [f"median_price_{year}" for year in median_price_pi

# Function to calculate the standard deviation of annual price changes
def calculate_price_change_std(prices):
    yearly_changes = np.diff(prices) / prices[:-1]  # Compute annual percentage
    return np.std(yearly_changes)  # Compute standard deviation

# Compute price volatility for each borough
price_volatility = {}
```

```
for city in median_price_pivot.index:
    prices = median_price_pivot.loc[city].dropna().values  # Get non-null price
    if len(prices) > 1:  # Ensure at least two years of data
        price_volatility[city] = calculate_price_change_std(prices)
    else:
        price_volatility[city] = np.nan  # Not enough data to compute volatility

# Convert volatility dictionary to DataFrame
volatility_df = pd.DataFrame.from_dict(price_volatility, orient="index", columns

# Merge volatility data with the median price DataFrame
final_df = median_price_pivot.merge(volatility_df, left_index=True, right_index=


# Convert CRS to latitude/longitude if necessary
if boroughs.crs != "EPSG:4326":
    boroughs = boroughs.to_crs(epsg=4326)

# Keep only necessary columns from the shapefile
boroughs = boroughs[['BoroName', 'geometry']]

# Calculate borough area in square kilometers
boroughs['area_km2'] = boroughs.geometry.to_crs(epsg=6933).area / 1e6  # Convert

# Count Airbnb listings per borough
valid_boroughs = ["Manhattan", "Brooklyn", "Queens", "Bronx", "Staten Island"]
airbnb_df = airbnb_df[airbnb_df['neighbourhood_group'].isin(valid_boroughs)]

# Count Airbnb listings per borough
airbnb_counts = airbnb_df['neighbourhood_group'].value_counts().reset_index()
airbnb_counts.columns = ['city', 'num_airbnb_listings']

# Merge Airbnb counts with the borough shapefile to get area
airbnb_density_df = boroughs.merge(airbnb_counts, left_on="BoroName", right_on="

# Compute Airbnb density: num_airbnb_listings per km²
airbnb_density_df["airbnb_density"] = airbnb_density_df["num_airbnb_listings"] /

# Merge Airbnb density into the final dataset
final_df = final_df.merge(airbnb_density_df[['city', 'airbnb_density']], on="cit

# Reset index to keep borough names as a separate column
final_df.reset_index(inplace=True)

# Save the processed data for mapping
processed_data_path = r"C:\Users\User\Desktop\ECO225\Datasets\processed_df.csv"
final_df.to_csv(processed_data_path, index=False)
```

In [ ]:

```
### Merge the housing dataset and all processed variables together for future LR
import pandas as pd

# Load the datasets
housing_file = r"C:\Users\User\Desktop\ECO225\Datasets\NYChousing.csv"
median_price_file = r"C:\Users\User\Desktop\ECO225\Datasets\processed_df.csv"

housing_df = pd.read_csv(housing_file)
median_price_df = pd.read_csv(median_price_file)

# Merge the datasets on "city"
merged_df = housing_df.merge(median_price_df, on="city", how="left")
```

```
# Save the merged dataset
merged_file_path = r"C:\Users\User\Desktop\ECO225\Datasets\merged_housing.csv"
merged_df.to_csv(merged_file_path, index=False)
```

1.3 Summary Statistic Tables

We selected multiple independent variables and one response variable including controlled variables like 'bed', 'bath','room_type' and 'acre_lot' (house area) affect prices and help group data for analysis. Table 1 and 2 shows those controlled variables along with response. Table 3 displays a new variable that direcly help further analysis on price growth rate. Table 4 introduces 3 new variables, median price, price volatility, and STR density.

In [47]:
```
import pandas as pd
from IPython.display import display

# File path
ny_housing_path = r"C:\Users\User\Desktop\ECO225\Datasets\NYChousing.csv"

# Load dataset
ny_housing = pd.read_csv(ny_housing_path)

# Ensure borough names are consistent
boroughs = ['Bronx', 'Brooklyn', 'Manhattan', 'Queens', 'Staten Island']

# Filter dataset for the five boroughs
ny_housing_filtered = ny_housing[ny_housing['city'].isin(boroughs)]

# Compute summary statistics
summary_table = ny_housing_filtered.groupby('city').agg(
    housing_count=('city', 'count'),
    median_price=('price', 'median'),
    std_price=('price', 'std'),
    median_house_size=('house_size', 'median'),
    avg_bed=('bed', 'mean'),
    avg_bath=('bath', 'mean')
).reset_index()

# Rename columns to concise labels
summary_table.rename(columns={
    "housing_count": "Count",
    "median_price": "Med Price",
    "std_price": "STD Price",
    "median_house_size": "Med Size",
    "avg_bed": "Avg Bed",
    "avg_bath": "Avg Bath"
}, inplace=True)

# Format numeric values for readability
summary_table["Med Price"] = summary_table["Med Price"].apply(lambda x: f"${x:
summary_table["STD Price"] = summary_table["STD Price"].apply(lambda x: f"${x:
summary_table["Med Size"] = summary_table["Med Size"].apply(lambda x: f"{x:,.0
summary_table["Avg Bed"] = summary_table["Avg Bed"].apply(lambda x: f"{x:.1f}"
summary_table["Avg Bath"] = summary_table["Avg Bath"].apply(lambda x: f"{x:.1f
```

```
# Sort by median price in descending order
summary_table = summary_table.sort_values(by="Med Price", ascending=False)

# Display formatted table
display(summary_table.style.format({
    'Housing Count': "{:,}"
}).set_caption("<b>Table 1: NYC Housing Market Summary</b>").set_table_styles([
    {'selector': 'td', 'props': [('text-align', 'center')]},
    {'selector': 'th', 'props': [('text-align', 'center')]}
]))
```

Table 1: NYC Housing Market Summary

	city	Count	Med Price	STD Price	Med Size	Avg Bed	Avg Bath
1	Brooklyn	2038	$799,750	$728,986	1,230 sqft	3.1	2.3
3	Queens	139	$799,000	$413,986	920 sqft	2.5	1.7
4	Staten Island	1359	$650,000	$336,760	1,524 sqft	3.2	2.6
0	Bronx	572	$582,500	$447,013	1,428 sqft	3.5	2.3
2	Manhattan	180	$1,195,000	$4,305,856	1,062 sqft	1.7	1.7

Table 1 summarizes key variables in NYC housing market. Generally, Brooklyn has the highest number of housing units (2,038) with a median price of 799,750 dollar, followed by Queens at 799,000 dollar。 However, Queens has significantly fewer units (139). Manhattan has the highest median price at 1,195,000 dollar, and also the highest price variation (4,305,856 dollar). Staten Island has the largest median house size (1,524 sqft) and the highest average number of bathrooms (2.6). Bronx has the most bedrooms on average (3.5), but its median price is the lowest at 582,500 dollar. The table shows that housing availability and price trends vary significantly across NYC boroughs.

In [113…

```
### Summary table 2 for room_type
import pandas as pd

# File path for the dataset
dataset_path = r"C:\Users\User\Desktop\ECO225\Datasets\AB_NYC_2019_Cleaned.csv"

# Step 1: Load the Airbnb dataset
data = pd.read_csv(dataset_path)

# Step 2: Filter valid room types
valid_room_types = ['Private room', 'Entire home/apt', 'Shared room']
data = data[data['room_type'].isin(valid_room_types)]

# Step 3: Group by 'neighbourhood_group' and 'room_type', count occurrences
room_type_summary = (
    data.groupby(['neighbourhood_group', 'room_type'])
    .size()
    .reset_index(name='count')  # Rename size count to 'count'
    .pivot(index='neighbourhood_group', columns='room_type', values='count')
    .fillna(0)  # Fill missing values with 0
)
```

```
# Step 4: Add Total Listings
room_type_summary['Total Listings'] = room_type_summary.sum(axis=1)

# Step 5: Reset index for display
room_type_summary = room_type_summary.reset_index()

# Step 6: Remove unnecessary 'room_type' column
room_type_summary = room_type_summary.loc[:, ~room_type_summary.columns.str.cor

# Step 7: Format only the numeric columns
try:
    from IPython.display import display
    numeric_columns = ['Entire home/apt', 'Private room', 'Shared room', 'Total
    display(
        room_type_summary.style.format(
            {col: "{:.0f}" for col in numeric_columns}  # Apply formatting only
        ).set_caption("<b>Table 2: Room Type Summary by Boroughs (Top 5)<b>")
        .set_table_styles([
            {'selector': 'td', 'props': [('text-align', 'center')]},  # Center-
            {'selector': 'th', 'props': [('text-align', 'center')]}   # Center-
            ])
    )
except ImportError:
    print("IPython is not available. Here is the raw table:")
    print(room_type_summary)
```

Table 2: Room Type Summary by Boroughs (Top 5)

	room_type	neighbourhood_group	Entire home/apt	Private room	Shared room	Total Listings
0		Bronx	374	649	59	1082
1		Brooklyn	9462	10034	410	19906
2		Manhattan	13052	7866	477	21395
3		Queens	2066	3289	195	5550
4		Staten Island	175	184	9	368

Table 2 summarizes the 'room_type' variable in the Airbnb dataset for the top 5 boroughs. The three categories are 'Entire home/apt,' 'Private room,' and 'Shared room.' 'Entire home/apt' includes listings renting out whole homes, with the Bronx having 374 listings. 'Private room,' the most common and affordable option, has the highest number in Brooklyn in 2019 (AirDNA, 2023). 'Shared room,' where guests share a space with others, is the least popular category.

In [51]:
```
import pandas as pd
from IPython.display import display

# File paths
ny_housing_path = r"C:\Users\User\Desktop\ECO225\Datasets\NYChousing.csv"  # Pr
airbnb_data_path = r"C:\Users\User\Desktop\ECO225\Datasets\AB_NYC_2019_Cleaned

# Load datasets
ny_housing = pd.read_csv(ny_housing_path)
```

```
airbnb_data = pd.read_csv(airbnb_data_path)

# Ensure borough names are consistent
boroughs = ['Bronx', 'Brooklyn', 'Manhattan', 'Queens', 'Staten Island']

# Count housing units in each borough
housing_counts = ny_housing['city'].value_counts().reindex(boroughs, fill_value

# Count Airbnb listings in each borough
airbnb_counts = airbnb_data['neighbourhood_group'].value_counts().reindex(borou

# Compute Airbnb-to-Housing Ratio (AHR)
ahr = airbnb_counts / housing_counts

# Create the final table
summary_table = pd.DataFrame({
    'City': boroughs,
    'Housing Count': housing_counts.values,
    'Airbnb Count': airbnb_counts.values,
    'AHR': ahr.values
})

# Sort by AHR (descending)
summary_table = summary_table.sort_values(by='AHR', ascending=False)

# Format table display
display(summary_table.style.format({
    'housing count': "{:.0f}",
    'airbnb count': "{:.0f}",
    'AHR': "{:.2f}"
}).set_caption("<b>Table 3: Housing and Airbnb Summary</b>").set_table_styles([
    {'selector': 'td', 'props': [('text-align', 'center')]},
    {'selector': 'th', 'props': [('text-align', 'center')]}
]))
```

Table 3: Housing and Airbnb Summary

	City	Housing Count	Airbnb Count	AHR
2	Manhattan	180	21395	118.86
3	Queens	139	5550	39.93
1	Brooklyn	2038	19906	9.77
0	Bronx	572	1082	1.89
4	Staten Island	1359	368	0.27

Table 3 publishes a key variable, Airbnb-Housing Ratio, in this descriptive data analysis, which is calculated as:

$$\text{Airbnb Housing Ratio(AHR)} = \frac{\text{Number of Airbnb listings in one city}}{\text{Number of on-sale housing in the same city}}$$

Manhattan has the highest ratio (118.86), meaning it has far more Airbnb listings than housing units. Queens follows with a ratio of 39.93, while Brooklyn, Bronx, and Staten Island have significantly lower ratios, indicating less Airbnb

activity relative to housing availability. This suggests that short-term rentals are more concentrated in Manhattan and Queens, while Staten Island has the lowest Airbnb presence.

In [55]:
```
import pandas as pd
from IPython.display import display

# Load the dataset
file_path = r"C:\Users\User\Desktop\ECO225\Datasets\merged_housing.csv"
df = pd.read_csv(file_path)

# Define the columns for median price before and after 2019
pre_2019_cols = [f"median_price_{year}" for year in range(2014, 2019)]
post_2019_cols = [f"median_price_{year}" for year in range(2019, 2023)]

# Calculate the average median prices before and after 2019
df["avg_median_price_pre2019"] = df[pre_2019_cols].mean(axis=1)
df["avg_median_price_post2019"] = df[post_2019_cols].mean(axis=1)

# Group by borough (city column) and calculate mean values for required variabl
summary_table = df.groupby("city").agg(
    avg_median_price_pre2019=("avg_median_price_pre2019", "mean"),
    avg_median_price_post2019=("avg_median_price_post2019", "mean"),
    price_volatility=("price_volatility", "mean"),
    airbnb_density=("airbnb_density", "mean")
).reset_index()

# Rename columns to phrases
summary_table.rename(columns={
    "avg_median_price_pre2019": "Avg Med Price Pre-2019",
    "avg_median_price_post2019": "Avg Med Price Post-2019",
    "price_volatility": "Price Volatility",
    "airbnb_density": "Airbnb Density"
}, inplace=True)

# Format and display the table with an academic-style title
try:
    display(
        summary_table.style.format(
            {
                'Avg Med Price Pre-2019': "${:,.0f}",
                'Avg Med Price Post-2019': "${:,.0f}",
                'Price Volatility': "{:.4f}",
                'Airbnb Density': "{:.2f}"
            }
        ).set_caption(
            "<b>Housing Market Summary by Borough (Pre-2019 vs. Post-2019)</b>"
        ).set_table_styles([
            {'selector': 'td', 'props': [('text-align', 'center')]},  # Center-
            {'selector': 'th', 'props': [('text-align', 'center')]}  # Center-
        ])
    )
except ImportError:
    print(summary_table)
```

Housing Market Summary by Borough (Pre-2019 vs. Post-2019)

	city	Avg Med Price Pre-2019	Avg Med Price Post-2019	Price Volatility	Airbnb Density
0	Bronx	$448,700	$631,750	0.0757	9.81
1	Brooklyn	$801,400	$830,000	0.0787	110.78
2	Manhattan	$1,142,800	$1,201,125	0.3820	361.73
3	Queens	$645,200	$960,625	0.4832	19.64
4	Staten Island	$717,200	$733,832	0.1104	2.44

Table 4 presents a summary of housing market and two key metrics across New York Boroughs. Generally, Table 4 presents that the median price increase from pre-2019 to post-2019 in all five boroughs. For example, Manhattan rises from 1,142,800 dollars to 1,201,125 dollars, the highest among all five boroughs. Manhattan also have a moderate price voaltility(variation) and largest number of airbnb listing per area(airbnb_density). Queens showed high price volatility, but less airbnb density than Manhattan and Brooklyn. The overall trend indicate that boroughs with higher Airbnb density might exhibit higher price volatility, and higher price.

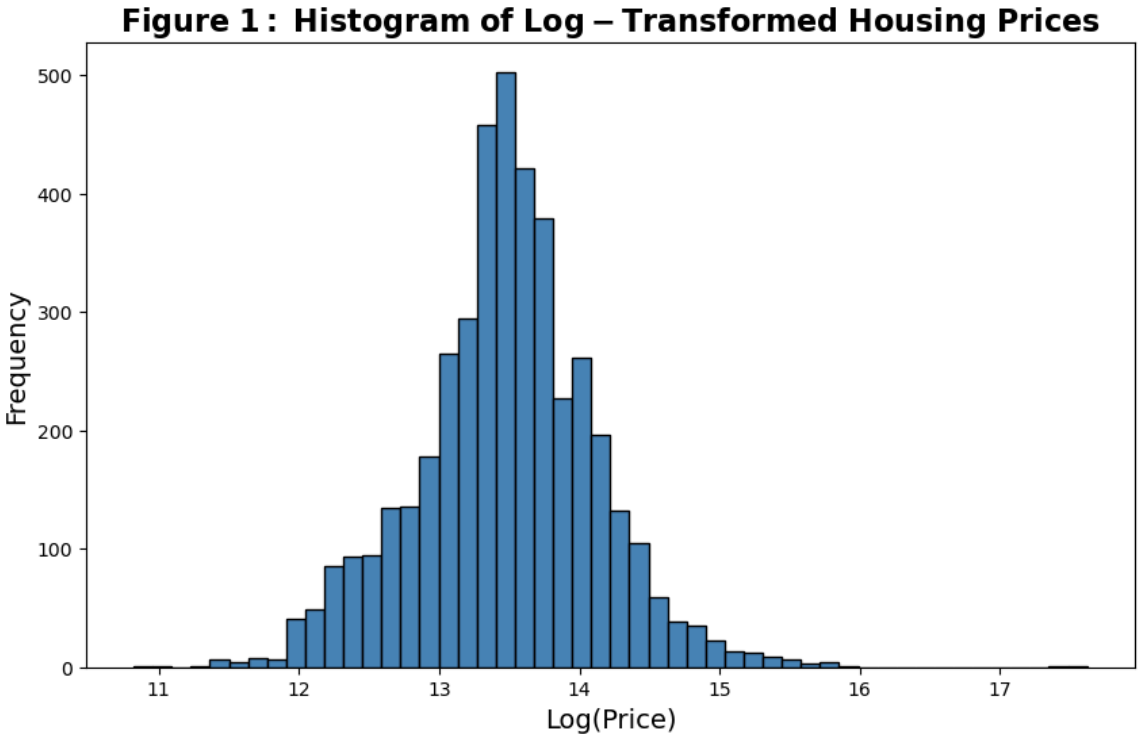
1.4 Plots, Histograms, Figures

In this part, the study shows different plots and graphs, aiming to visualize variables and relationships between different variables, especially response variables and showing relationships with several other explanatory variables. We filter out boroughs that is statistical meaningless.

In [134…

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Load the dataset
file_path = r"C:\Users\User\Desktop\ECO225\Datasets\NYChousing.csv"
ny_housing = pd.read_csv(file_path)

# Check if the 'price' column exists
if 'price' in ny_housing.columns:
    # Apply log transformation to the 'price' column
    ny_housing['log_price'] = np.log1p(ny_housing['price'])  # log1p handles lo

    # Filter out prices where log(price) < 7.5
    ny_housing_filtered = ny_housing[ny_housing['log_price'] >= 7.5]

    # Plot the histogram of the log-transformed prices
    plt.figure(figsize=(10, 6))
    plt.hist(ny_housing_filtered['log_price'].dropna(), bins=50, edgecolor='bla
    plt.title(r'$\bf{Figure\ 1: \ Histogram\ of\ Log-Transformed\ Housing\ Pric
```

```
    plt.xlabel('Log(Price)', fontsize=14)
    plt.ylabel('Frequency', fontsize=14)
    plt.show()
```



The histogram of log-transformed housing prices shows a more balanced distribution, and a log transformation addresses the skewness caused by extremely high-priced properties. The majority of housing prices fall within the log range of 13 to 14, approximately 442,413 to 1,202,604 in actual prices.

In [136…

```
# Boxplot of Log-Transformed Airbnb-to-Housing Ratio
### I see the comment on removing log transoformation to see the outlier. Howev
### and has a bad visualization graph, which is also the reason why I apply log
### I consider it as my personal habit when doing log transoformation in any da
### Since we can translate to original values based on the transoformed value j
### boxplot here is mostly used to observe the data distribution, not the exact
### in visualization, so I don't think a log(AHR+1) here need a big change. I j
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

# Define the paths for the datasets
ny_housing_path = r"C:\Users\User\Desktop\ECO225\Datasets\NYChousing.csv"  # NY
airbnb_data_path = r"C:\Users\User\Desktop\ECO225\Datasets\AB_NYC_2019_Cleaned

# Load the datasets
ny_housing = pd.read_csv(ny_housing_path)
airbnb_data = pd.read_csv(airbnb_data_path)

# Standardize borough column names
ny_housing.rename(columns={"city": "borough"}, inplace=True)
airbnb_data.rename(columns={"neighbourhood_group": "borough"}, inplace=True)

# Count housing units and Airbnb listings for each borough
housing_counts = ny_housing["borough"].value_counts().rename("housing_count")
```

```
airbnb_counts = airbnb_data["borough"].value_counts().rename("airbnb_count")

# Merge counts into a single DataFrame
ahr_data = pd.DataFrame({"housing_count": housing_counts, "airbnb_count": airbr
ahr_data.rename(columns={"index": "borough"}, inplace=True)

# Calculate the Airbnb-to-Housing Ratio (AHR)
ahr_data["airbnb_housing_ratio"] = ahr_data["airbnb_count"] / ahr_data["housing

# Apply log transformation to AHR (log(AHR))
ahr_data["log_airbnb_housing_ratio"] = np.log(ahr_data["airbnb_housing_ratio"])

# Create the horizontal boxplot
plt.figure(figsize=(10, 6))
sns.boxplot(
    x=ahr_data["log_airbnb_housing_ratio"],
    color="skyblue",  # Professional, academic-friendly color
    linewidth=1.5
)

# Add labels and title
plt.title(r'$\bf{Figure\ 2: \ Boxplot\ of\ Log-Transformed\ Airbnb-to-Housing\
plt.xlabel("Log(AHR)", fontsize=12)
plt.grid(axis="x", linestyle="--", alpha=0.7)
plt.tight_layout()

# Show the plot
plt.show()
```

Figure 2 : Boxplot of Log – Transformed Airbnb – to – Housing Ratio (AHR)

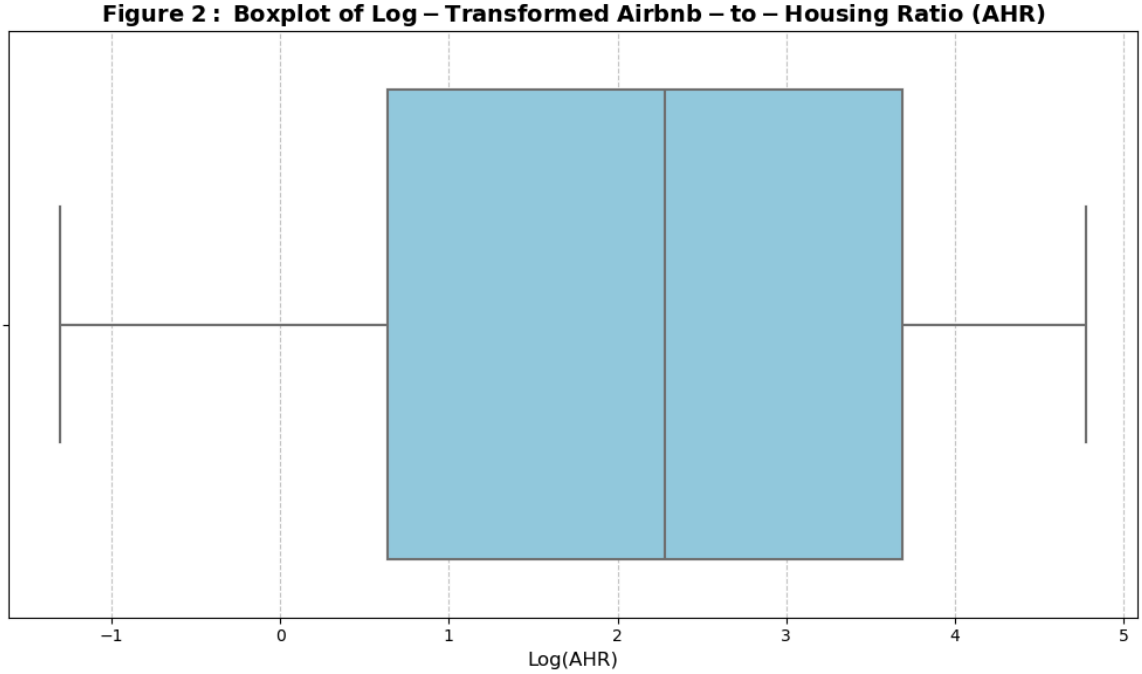


Figure 2 shows the distribution of Airbnb-to-Housing Ratio (AHR) after applying a log transformation to correct for bad original visualization. The median log(AHR) is around 2.3, meaning over half of the boroughs have an AHR approximately below 10.The maximum whisker reaches log(AHR) 4.7, translating to an extreme AHR of approximatly 110, which is Manhattan, according to Table 3, suggesting that this borough have significantly more STR listings relative to housing supply.

In [138…

```
### In this plot, I also see the comment on removing the log transformation, ar
### as above: original graph is highly unreadable, and applying a log transform
### housing price distribution. Key values in this graph can be translated to d

# Step 1: Load the Airbnb dataset
airbnb_path = r"C:\Users\User\Desktop\ECO225\Datasets\AB_NYC_2019_Cleaned.csv"
airbnb_data = pd.read_csv(airbnb_path)

# Step 2: Ensure 'price' is numeric and handle non-numeric or missing values
airbnb_data['price'] = pd.to_numeric(airbnb_data['price'], errors='coerce')  #
airbnb_data = airbnb_data.dropna(subset=['price'])  # Drop rows with NaN prices

# Step 3: Filter the Airbnb dataset for prices greater than 10 (ensuring log is
airbnb_data = airbnb_data[airbnb_data['price'] > 10]

# Step 4: Apply log transformation to price
airbnb_data['log_price'] = np.log(airbnb_data['price'])  # log(price) transform

# Step 5: Plot the histogram of log-transformed Airbnb prices
plt.figure(figsize=(10, 6))
sns.histplot(
    airbnb_data['log_price'],
    kde=True,
    bins=50,
    color='skyblue',  # Use sky blue color for the bars
    edgecolor='black'  # Add black edges for better clarity
)

# Step 6: Add labels and title
plt.title(r'$\bf{Figure\ 3: \ Histogram\ of\ Log-Transformed\ Airbnb\ Prices\ (
plt.xlabel("Log(Price)", fontsize=12)
plt.ylabel("Frequency", fontsize=12)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()

# Step 7: Display the plot
plt.show()
```

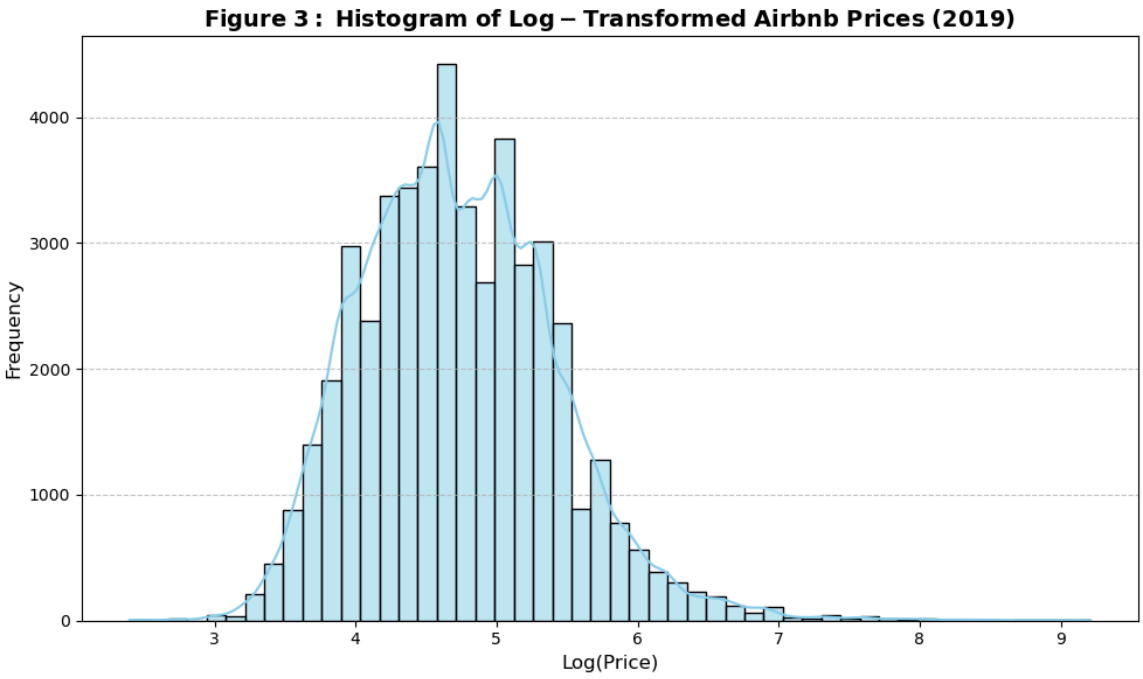


Figure 3 introduces the distribution on Airbnb prices. Since most STRs' prices are relatively low, and some STRs's price are very high, presenting the original histogram will be highly right skewed and effect our future analysis. Thus, a log transformation is applied to X-axis, which is similar to Figure 2. After the transformation, the distribution is slightly right skewed due to extreme high values at right. Most data lies on X=4 to X=5.5, which is between 54.60 and 244.69 in real price.

In [140…

```
import warnings
warnings.filterwarnings("ignore")  # Suppress all warnings

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib.ticker import FuncFormatter

# Load the dataset
housing_data_path = r"C:\Users\User\Desktop\ECO225\Datasets\NYChousing.csv"
housing_data = pd.read_csv(housing_data_path)

# Ensure that 'prev_sold_date' is in datetime format and create the period colu
housing_data['prev_sold_date'] = pd.to_datetime(housing_data['prev_sold_date'],
housing_data['period'] = housing_data['prev_sold_date'].dt.year.apply(
    lambda x: 'Pre-2019' if x < 2019 else 'Post-2019'
)

# Filter only necessary columns and keep only the five NYC boroughs
housing_data = housing_data[['city', 'price', 'period']]
boroughs = ['Bronx', 'Brooklyn', 'Manhattan', 'Queens', 'Staten Island']
housing_data = housing_data[housing_data['city'].isin(boroughs)]

# Set up the figure with a 3x2 grid layout
fig, axes = plt.subplots(2, 3, figsize=(20, 12))

# Flatten the axes array for easy iteration
axes = axes.flatten()

# Define a formatter for the y-axis to show currency format
def currency_format(x, _):
    return f'${int(x):,}'

formatter = FuncFormatter(currency_format)

# Loop through each borough and plot its violin plot
for i, borough in enumerate(boroughs):
    borough_data = housing_data[housing_data['city'] == borough]
    sns.violinplot(
        data=borough_data,
        x='period',
        y='price',
        order=['Pre-2019', 'Post-2019'],  # Ensure x-axis order
        scale='count',
        ax=axes[i],
        palette={'Pre-2019': 'steelblue', 'Post-2019': 'lightgreen'}
    )
    axes[i].set_title(borough, fontsize=14)
```

```
    axes[i].set_xlabel("")
    axes[i].set_ylabel("Housing Price (USD)")
    axes[i].grid(alpha=0.3)
    axes[i].yaxis.set_major_formatter(formatter)

    # Set different Y-axis limits for Manhattan vs. Other boroughs
    if borough == "Manhattan":
        axes[i].set_ylim(0, 50_000_000)  # Manhattan range
    else:
        axes[i].set_ylim(0, 10_000_000)  # Other boroughs capped at 10M

# Hide unused subplots if any
for j in range(len(boroughs), len(axes)):
    fig.delaxes(axes[j])

# Adjust layout
fig.suptitle(r'$\bf{Figure\ 4: \ Violin\ Plot\ for\ Housing\ Price\ Distributic
             fontsize=20, y=0.98)
plt.tight_layout(rect=[0, 0, 1, 0.96])  # Adjust rect to fit title
plt.show()
```

Figure 4 : Violin Plot for Housing Price Distributions by NYC Boroughs (Pre – 2019 vs Post – 2019)

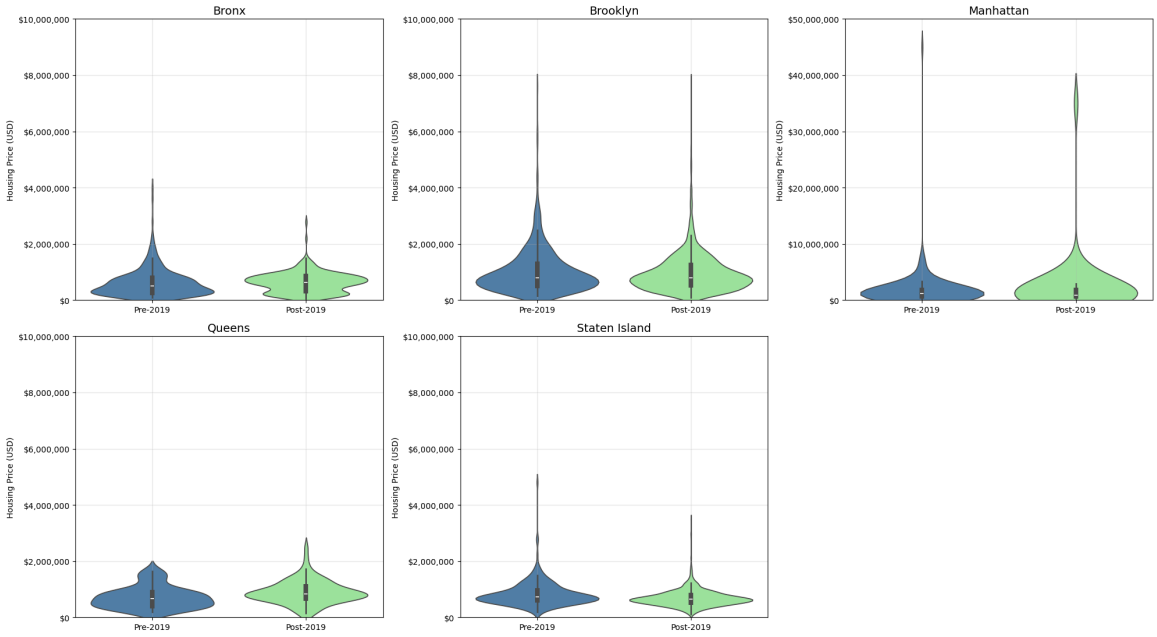


Figure 4 shows the housing price distribution for different boroughs using violin plots. A larger area indicates more housing availability.

Manhattan has the highest price and price range among all broughs, with the price extending to nearly 50,000,000 dollars. Queens increased the range, but the overall housing avalibility decreases since the post-2019 violin plot for Queens is thinner than pre-2019 one. Bronx's price range shrink from around 4,000,000 dollars to 3,000,000 dollars, along with a decrease in total housing supply. Staten Island also showed an shrinked range to 3,750,000 dollars, but these are limited to high-priced units, which have minimal impact on the overall market(Texas 2036, n.d).

Overall, most broroughs experienced a shirnk in price and housing availability after 2019.

In [167…

```
### This plot is to find whether which borough has the lowest AHR ratio, and us
### You may see the description in below and in Figure 7.1 to 7.4.
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

# File paths for the datasets
ny_housing_path = r"C:\Users\User\Desktop\ECO225\Datasets\NYChousing.csv"
airbnb_path = r"C:\Users\User\Desktop\ECO225\Datasets\AB_NYC_2019_Cleaned.csv"

# Step 1: Load the datasets
ny_housing = pd.read_csv(ny_housing_path)
airbnb_data = pd.read_csv(airbnb_path)

# Step 2: Count housing units per borough from NYChousing
housing_counts = ny_housing['city'].value_counts().rename_axis('borough').reset

# Step 3: Count Airbnb listings per borough from AB_NYC_2019_Cleaned
airbnb_counts = airbnb_data['neighbourhood_group'].value_counts().rename_axis(

# Step 4: Merge both counts into a single dataframe
merged_counts = pd.merge(housing_counts, airbnb_counts, on='borough', how='inne

# Step 5: Calculate log-transformed counts
merged_counts['log_housing_count'] = np.log1p(merged_counts['housing_count'])
merged_counts['log_airbnb_count'] = np.log1p(merged_counts['airbnb_count'])

# Step 6: Sort data by total count (original, not log-transformed)
merged_counts['total_count'] = merged_counts['housing_count'] + merged_counts[
merged_counts = merged_counts.sort_values(by='total_count', ascending=False)

# Step 7: Create the Stacked Bar Plot
plt.figure(figsize=(12, 8))
plt.bar(
    merged_counts['borough'],
    merged_counts['log_housing_count'],
    color='steelblue',
    label='Log Housing Count',
    edgecolor='black'
)
plt.bar(
    merged_counts['borough'],
    merged_counts['log_airbnb_count'],
    bottom=merged_counts['log_housing_count'],
    color='lightgreen',
    label='Log Airbnb Count',
    edgecolor='black'
)

# Step 8: Add labels and title
plt.title(r'$\bf{Figure\ 5: \ Stacked\ Bar\ Plot\ of\ Log-Transformed\ Housing\
plt.xlabel('Borough', fontsize=12)
plt.ylabel('Log-Transformed Count', fontsize=12)
plt.xticks(rotation=45, ha='right', fontsize=10)
plt.legend(title='Count Type', fontsize=10)
```

```
# Step 9: Adjust layout and show the plot
plt.tight_layout()
plt.grid(axis='y', linestyle='--', alpha=0.5)
plt.show()
```

Figure 5 : Stacked Bar Plot of Log – Transformed Housing and Airbnb Counts by Borough

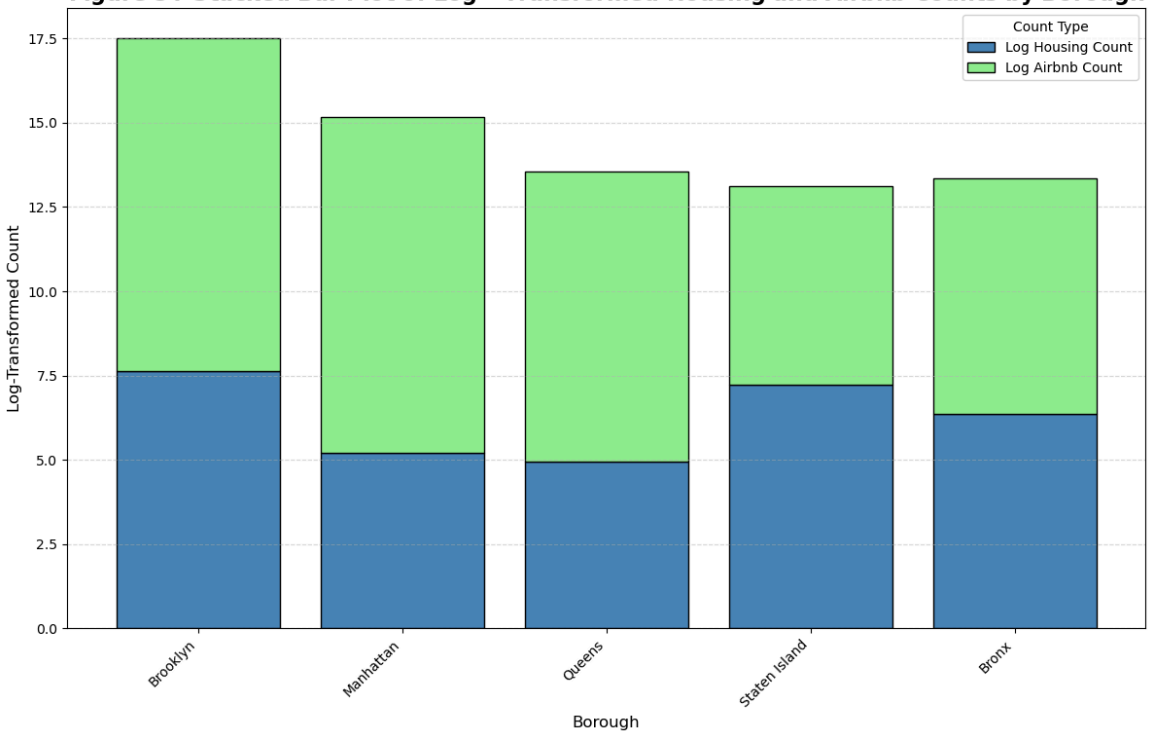


Figure 5 presents a stacked bar plot of log-transformed housing and Airbnb counts across the five NYC boroughs that is helpful to future line plot analysis(Figure 7.1 to 7.4). The log transformation allows for better visualization of the differences in scale between housing availability and Airbnb listings. Brooklyn has the highest combined count, with a significant proportion of Airbnb listings, and followed by Manhattan and Queens. Staten Island and the Bronx have relatively lower total counts and rely less on short-term rentals. This distribution suggests that Brooklyn and Manhattan are the primary boroughs for Airbnb activity.

In [144…

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

# Load datasets
housing_data_path = r"C:\Users\User\Desktop\ECO225\Datasets\processed_df.csv"
airbnb_data_path = r"C:\Users\User\Desktop\ECO225\Datasets\AB_NYC_2019_Cleaned

# Read datasets
housing_data = pd.read_csv(housing_data_path)
airbnb_data = pd.read_csv(airbnb_data_path)

# Ensure city/borough naming consistency
boroughs = ['Bronx', 'Brooklyn', 'Manhattan', 'Queens', 'Staten Island']
airbnb_data = airbnb_data[airbnb_data['neighbourhood_group'].isin(boroughs)]

# Count housing units for each borough
```

```
housing_counts = housing_data['city'].value_counts().reindex(boroughs, fill_va

# Count Airbnb listings for each borough
airbnb_counts = airbnb_data['neighbourhood_group'].value_counts().reindex(borou

# Calculate Airbnb-to-Housing Ratio (AHR)
ahr = airbnb_counts / housing_counts

# Apply log transformation to AHR (assuming AHR > 0)
ahr_log = np.log(ahr)

# Calculate Housing Price Growth (%) from pre-2019 (2014-2018) to post-2019 (20
median_price_cols_pre = [col for col in housing_data.columns if 'median_price_2
median_price_cols_post = [col for col in housing_data.columns if 'median_price_

housing_data['avg_median_price_pre2019'] = housing_data[median_price_cols_pre].
housing_data['avg_median_price_post2019'] = housing_data[median_price_cols_post

# Aggregate by borough to get average pre/post median prices
price_growth = housing_data.groupby('city')[['avg_median_price_pre2019', 'avg_m

# Compute percentage change in average median price
price_growth['price_growth_pct'] = ((price_growth['avg_median_price_post2019']
                                     price_growth['avg_median_price_pre2019']) *

# Merge log(AHR) and price growth data
final_data = pd.DataFrame({'log_AHR': ahr_log, 'price_growth_pct': price_growth

# Scatter Plot
plt.figure(figsize=(10, 6))
plt.scatter(final_data['log_AHR'], final_data['price_growth_pct'], color='black

# Annotate borough names
for borough in final_data.index:
    plt.text(final_data.loc[borough, 'log_AHR'], final_data.loc[borough, 'price

# Labels and title
plt.xlabel("Airbnb-to-Housing Ratio (log(AHR))")
plt.ylabel("Housing Price Growth (%)")
plt.title(r'$\bf{Figure\ 6: \ Scatterplot\ of\ Log\ Airbnb-to-Housing\ Ratio\ v

# Show plot
plt.grid(alpha=0.3)
plt.show()
```

Figure 6 : Scatterplot of Log Airbnb – to – Housing Ratio vs. Housing Price Growth (Pre – 2019 to Post – 2019)

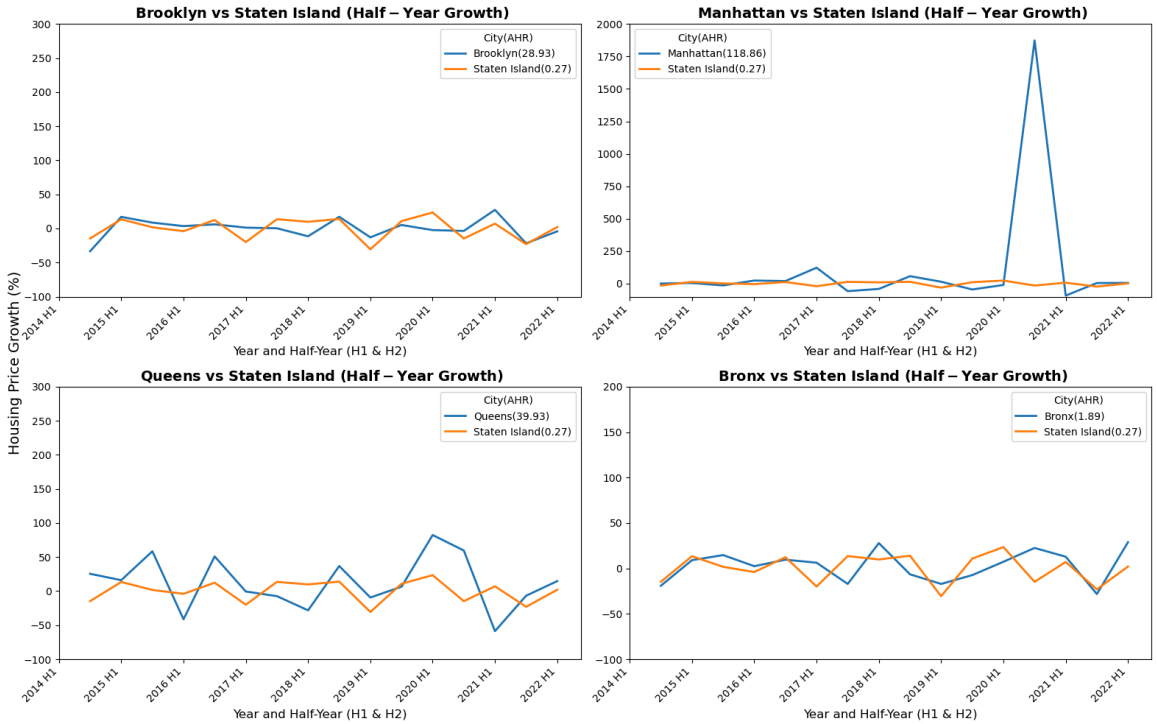
Figure 6 shows the relationship between Airbnb density and housing price growth across NYC boroughs. Queens and Bronx had the highest price growth, with a moderate Airbnb activity. Brooklyn and Manhattan, with the highest Airbnb densities, saw relatively lower growth. Staten Island, with the lowest Airbnb presence, experienced minimal price increases. The results suggest that Airbnb density alone does not fully explain housing price trends.

In [156…

```
### The reason I don't use the monthly change here is, those observations still
### will cause a severe problem on unintentional lines and readability on the g
### I asked this on TA hours and they said keep the bi-month for a better visua
import pandas as pd
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")  # Suppress all warnings

# File path for the dataset
ny_housing_path = r"C:\Users\User\Desktop\ECO225\Datasets\NYChousing.csv"

# Load the dataset
ny_housing = pd.read_csv(ny_housing_path)

# Define the boroughs to compare with Staten Island
selected_cities = ['Brooklyn', 'Manhattan', 'Queens', 'Bronx', 'Staten Island']
filtered_data = ny_housing[ny_housing['city'].isin(selected_cities)]

# Convert `prev_sold_date` to datetime format
filtered_data['prev_sold_date'] = pd.to_datetime(filtered_data['prev_sold_date
filtered_data['year'] = filtered_data['prev_sold_date'].dt.year
filtered_data['month'] = filtered_data['prev_sold_date'].dt.month

# Create a "half-year period" column (Jan-Jun = H1, Jul-Dec = H2)
filtered_data['half_year'] = filtered_data['month'].apply(lambda x: 'H1' if x <

# Group by year, half-year, and borough to compute the average housing price
price_trends_half_year = (
    filtered_data.groupby(['year', 'half_year', 'city'])
    .agg({'price': 'mean'})
    .reset_index()
)
```

```
# Compute half-year housing price growth (% change)
price_trends_half_year['price_growth'] = (
    price_trends_half_year.groupby('city')['price'].pct_change() * 100
)

# Combine year and half-year for x-axis labels
price_trends_half_year['year_half'] = (
    price_trends_half_year['year'].astype(str) + ' ' + price_trends_half_year[
)

# Set up the figure with a 2x2 layout (since we have 4 boroughs + Staten Island
fig, axes = plt.subplots(2, 2, figsize=(16, 10), sharey=False)  # sharey=False

# Define boroughs to compare with Staten Island
comparison_cities = [('Brooklyn', '28.93'), ('Manhattan', '118.86'), ('Queens',
titles = [
    r'$\bf{\ Brooklyn\ vs\ Staten\ Island\ (Half-Year\ Growth)}$',
    r'$\bf{\ Manhattan\ vs\ Staten\ Island\ (Half-Year\ Growth)}$',
    r'$\bf{\ Queens\ vs\ Staten\ Island\ (Half-Year\ Growth)}$',
    r'$\bf{\ Bronx\ vs\ Staten\ Island\ (Half-Year\ Growth)}$',
]

# Define custom Y-axis ranges to improve visualization
y_limits = {
    'Brooklyn': (-100, 300),
    'Manhattan': (-100, 2000),
    'Queens': (-100, 300),
    'Bronx': (-100, 200)
}

# Iterate over the four boroughs to plot against Staten Island
for i, (city, ahr) in enumerate(comparison_cities):
    row, col = divmod(i, 2)  # Get subplot row and column indices
    city_data = price_trends_half_year[price_trends_half_year['city'] == city]
    staten_island_data = price_trends_half_year[price_trends_half_year['city']

    # Plot city data
    axes[row, col].plot(
        city_data['year_half'],
        city_data['price_growth'],
        label=f'{city}({ahr})',
        linestyle='-',
        linewidth=2
    )

    # Plot Staten Island data as baseline
    axes[row, col].plot(
        staten_island_data['year_half'],
        staten_island_data['price_growth'],
        label='Staten Island(0.27)',
        linestyle='-',
        linewidth=2
    )

    # Customize Y-axis range based on city
    axes[row, col].set_ylim(y_limits[city])  # Apply custom Y-axis range

    # Customize x-axis labels
    jan_jul_labels = city_data[city_data['half_year'] == 'H1']['year_half']
    axes[row, col].set_xticks(jan_jul_labels)
```

```
    axes[row, col].set_xticklabels(jan_jul_labels, rotation=45, ha='right', for

    axes[row, col].set_title(titles[i], fontsize=14)
    axes[row, col].set_xlabel('Year and Half-Year (H1 & H2)', fontsize=12)
    axes[row, col].legend(title='City(AHR)')
    axes[row, col].grid(False)  # Disable gridlines and any background shading


# Add shared y-axis label
fig.supylabel('Housing Price Growth (%)', fontsize=14)
plt.tight_layout()
# Save the figure to your desired path
plt.savefig(r"C:\Users\User\Desktop\ECO225\Final Paper image\half_year_growth_c

# Show the plot
plt.show()
```



According to Figure 5 and Table 3, we found that Staten Island are the best three options for line plot analysis, since it is the lowest AHR among all five boroughs.

Due to missing data for certain months, applying a per-month trend would result in irregular patterns and unintended distortions in the analysis. As a result, Figures 7.1 to 7.4 present half-year housing price trends for boroughs with large and small AHR values.

Brooklyn, Queens, and Bronx exhibit moderate fluctuations, with price growth generally remaining within ±50%. In contrast, Manhattan demonstrates extreme volatility, with a peak exceeding 5000% in early 2020, followed by sharp declines. Queens and Brooklyn have moderate peaks around 100% and 80%, respectively, while the Bronx fluctuates within a narrower range. Staten Island remains relatively stable, with price changes mostly between -10% and 20%..

These graphs suggest a positive relationship between housing price variability and AHR in the selected cities, which may impact housing affordability and stability.

1.5 Limitations & Next Steps

AHR may not be a completely reliable metric in this analysis because the housing dataset originates from a broader US housing market dataset, which was later filtered into the NYC housing dataset. Thus, **this dataset might not include all housing unit in NYC**, leading to an incomplete count of total housing supply, which in turn affects the calculated AHR values. As a result, AHR-related findings should only be used as a preliminary variable to help get the idea between key metrics.

This study shows AHR helps analyze STRs and on-sale housing in New York City. Future work should focusing on significance and effect of short term rental related variables on long term housing market, and introduce more new variables based on new datasets or calculate them through existing datasets.

Project 2

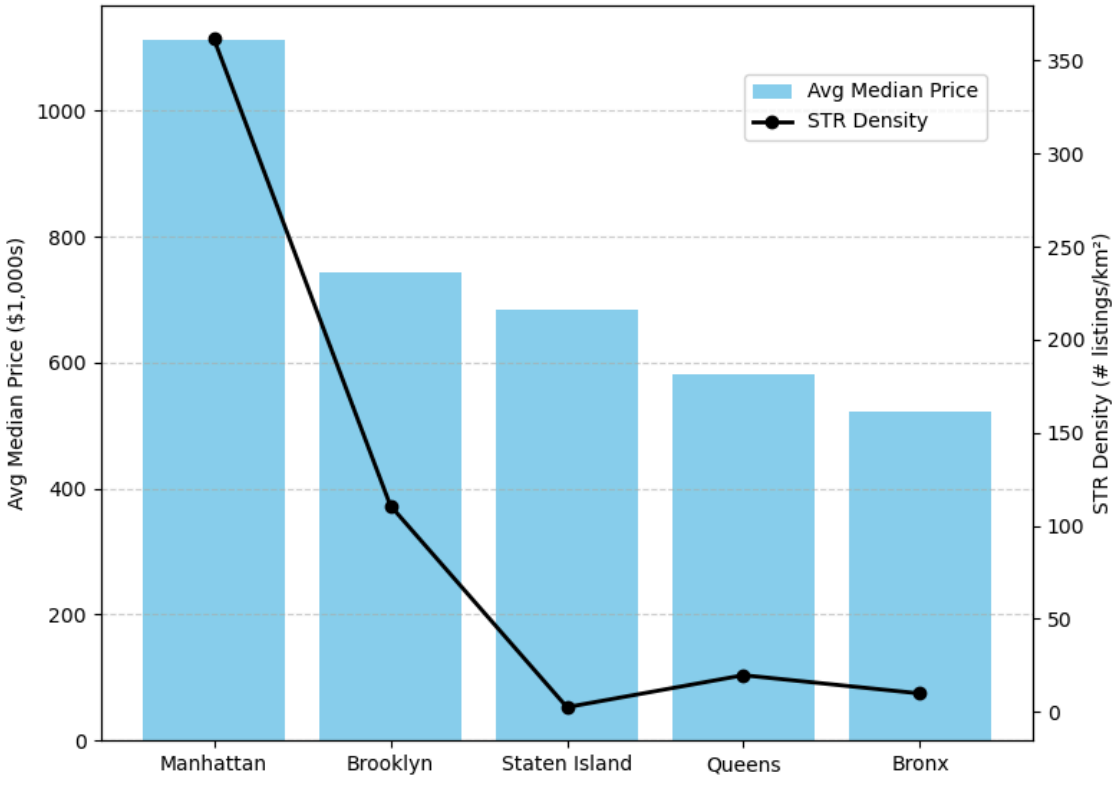
2.1 The Message

STR density is positively associated with housing prices across NYC boroughs. Manhattan, with the highest STR density of around362 listings/km², also has the highest average median price of $1.15 million), while the Bronx, with much lower STR activity (~10 listings/km²), has the lowest price of $520,000. However, no clear relationship is observed between STR density and price volatility, as volatility levels vary independently of STR intensity across several boroughs.

In [36]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Load the dataset
df = pd.read_csv(r"C:\Users\User\Desktop\ECO225\Datasets\processed_df.csv")

# Compute average median price and convert to $1,000s
df["avg_median_price"] = df[["median_price_2014", "median_price_2022"]].mean(ax
df["avg_median_price_k"] = df["avg_median_price"] / 1000

# Sort by price (optional for aesthetics)
df = df.sort_values(by="avg_median_price_k", ascending=False).reset_index(drop=
```

```
# Create the figure
fig, ax1 = plt.subplots(figsize=(8, 6))

# Bar plot: average median price
bars = ax1.bar(df["city"], df["avg_median_price_k"], color='skyblue', label='Av
ax1.set_ylabel("Avg Median Price ($1,000s)", color='black')

# STR density line and dots (on right axis)
ax2 = ax1.twinx()
ax2.set_ylabel("STR Density (# listings/km²)", color='black')
ax2.tick_params(axis='y', labelcolor='black')

# Line with dots for STR density
ax2.plot(df["city"], df["airbnb_density"], color='black', marker='o', linestyle

# Title and legend
fig.suptitle("Figure 8.1: Avg Median Housing Price and STR Density in NYC Borou
fig.legend(loc='upper right', bbox_to_anchor=(0.88, 0.85))

# Grid and layout
ax1.grid(axis='y', linestyle='--', alpha=0.6)
plt.tight_layout()

# Save figure
plt.show()
```

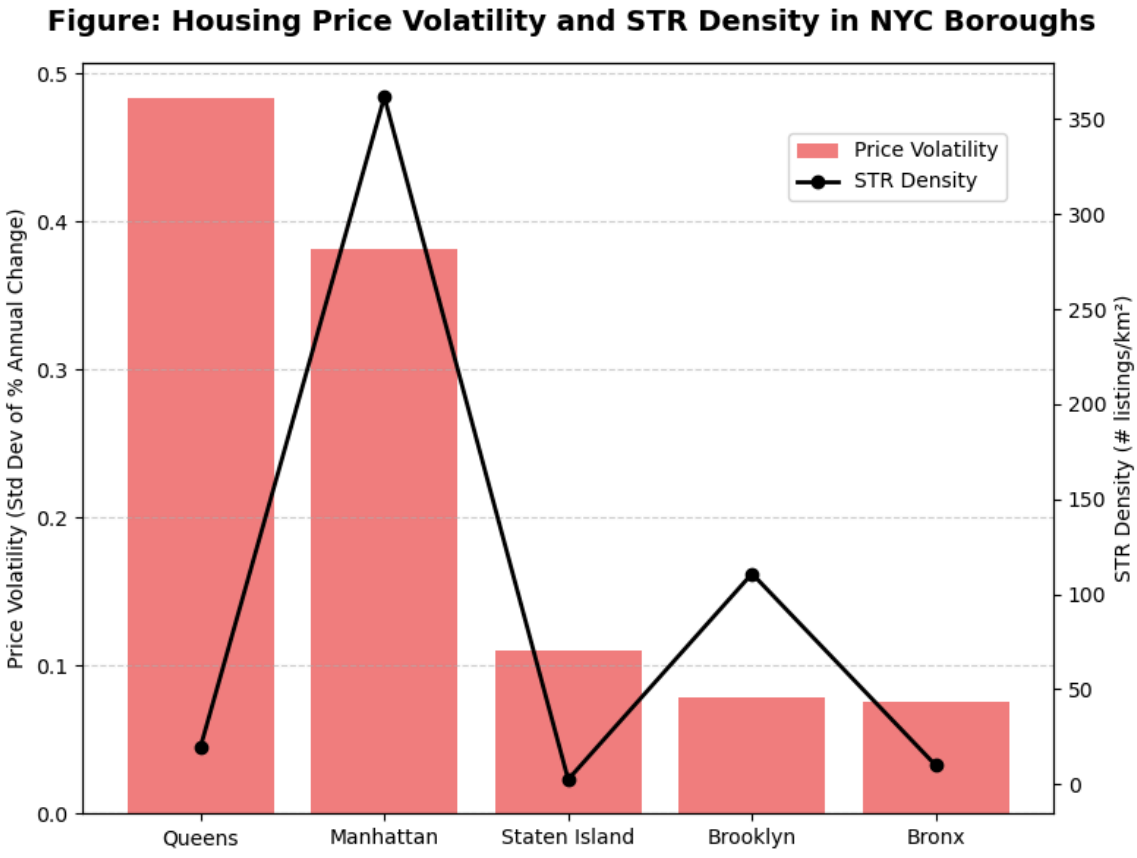
Figure 8.1: Avg Median Housing Price and STR Density in NYC Boroughs



In [34]:

```
import pandas as pd
import matplotlib.pyplot as plt

# Load dataset
df = pd.read_csv(r"C:\Users\User\Desktop\ECO225\Datasets\processed_df.csv")

# Optional: sort by volatility for a clean look
df = df.sort_values(by="price_volatility", ascending=False).reset_index(drop=Tr
```

```
# Create the figure
fig, ax1 = plt.subplots(figsize=(8, 6))

# Bar plot for price volatility
bars = ax1.bar(df["city"], df["price_volatility"], color='lightcoral', label='P
ax1.set_ylabel("Price Volatility (Std Dev of % Annual Change)", color='black')
ax1.tick_params(axis='y', labelcolor='black')

# Right Y-axis: STR density
ax2 = ax1.twinx()
ax2.set_ylabel("STR Density (# listings/km²)", color='black')
ax2.tick_params(axis='y', labelcolor='black')

# Line + dots for STR density
ax2.plot(df["city"], df["airbnb_density"], color='black', marker='o', linestyle

# Title and legend
fig.suptitle("Figure: Housing Price Volatility and STR Density in NYC Boroughs'
fig.legend(loc='upper right', bbox_to_anchor=(0.88, 0.85))

# Grid and layout
ax1.grid(axis='y', linestyle='--', alpha=0.6)
plt.tight_layout()

# Save the figure
plt.show()
```



Figures 8.1 and 8.2 compare average median housing prices, price volatility, and STR density across NYC boroughs.

The first panel shows a positive relationship between STR density and housing prices. Manhattan, with the highest STR density (≈362 listings/km²), has the highest

average median price of 1,150,000 dollar. Conversely, the Bronx, with one of the lowest STR densities of approximatly 10 listings/km², has the lowest average price (520,000–550,000 dollar).

However, no clear pattern emerges when comparing STR density and price volatility. While Manhattan combines high STR density with relatively high volatility (0.38), Queens stands out with the highest volatility (0.48) despite a moderate STR density (~20 listings/km²). Brooklyn and Staten Island, with very different STR densities, show similarly low volatility.

These figures support the claim that higher STR density is associated with higher housing prices, but they do not offer consistent evidence of a relationship between STR density and price volatility. The next sections explore these dynamics further through mapping visualization and regression modeling.

2.2 Maps and Interpretation

In [102…

```
import pandas as pd
import geopandas as gpd
import matplotlib.pyplot as plt
from shapely.geometry import Point
import matplotlib.colors as colors

# --------------------------------
# Step 1: Load Airbnb Dataset
# --------------------------------
airbnb_path = r"C:\Users\User\Desktop\ECO225\Datasets\AB_NYC_2019_Cleaned.csv"
airbnb_df = pd.read_csv(airbnb_path)

# Clean invalid coordinate rows
airbnb_df = airbnb_df[pd.to_numeric(airbnb_df['longitude'], errors='coerce').nc
airbnb_df = airbnb_df[pd.to_numeric(airbnb_df['latitude'], errors='coerce').not
airbnb_df['longitude'] = airbnb_df['longitude'].astype(float)
airbnb_df['latitude'] = airbnb_df['latitude'].astype(float)

# Convert to GeoDataFrame
geometry = [Point(xy) for xy in zip(airbnb_df['longitude'], airbnb_df['latitude
airbnb_gdf = gpd.GeoDataFrame(airbnb_df, geometry=geometry, crs="EPSG:4326")

# --------------------------------
# Step 2: Load NTA Neighborhood GeoJSON
# --------------------------------
geojson_url = "https://services5.arcgis.com/GfwWNkhOj9bNBqoJ/arcgis/rest/servic
nta = gpd.read_file(geojson_url)

# --------------------------------
# Step 3: Spatial Join Airbnb -> NTA
# --------------------------------
joined = gpd.sjoin(airbnb_gdf, nta, how="inner", predicate="within")

# Count listings per NTA
listing_counts = joined['NTAName'].value_counts().reset_index()
```

```
listing_counts.columns = ['NTAName', 'listing_count']

# Merge counts into NTA polygons
nta = nta.merge(listing_counts, on='NTAName', how='left')
nta['listing_count'] = nta['listing_count'].fillna(0)

# ------------------------------
# Step 4: Compute Density per km²
# ------------------------------
nta['area_km2'] = nta.to_crs(epsg=3857).geometry.area / 1e6
nta['density_per_km2'] = nta['listing_count'] / nta['area_km2']

# ------------------------------
# Step 5: Plot Choropleth Map
# ------------------------------
fig, ax = plt.subplots(figsize=(10, 8))
nta.plot(
    column='density_per_km2',
    cmap='Reds',
    linewidth=0.5,
    edgecolor='black',
    legend=True,
    legend_kwds={'label': "Airbnb Listings per km²"},
    ax=ax
)

ax.set_title("Figure 9.1: Airbnb Density by NYC Neighborhood", fontsize=14, for
ax.axis('off')
plt.tight_layout()
plt.show()
```

Figure 9.1: Airbnb Density by NYC Neighborhood

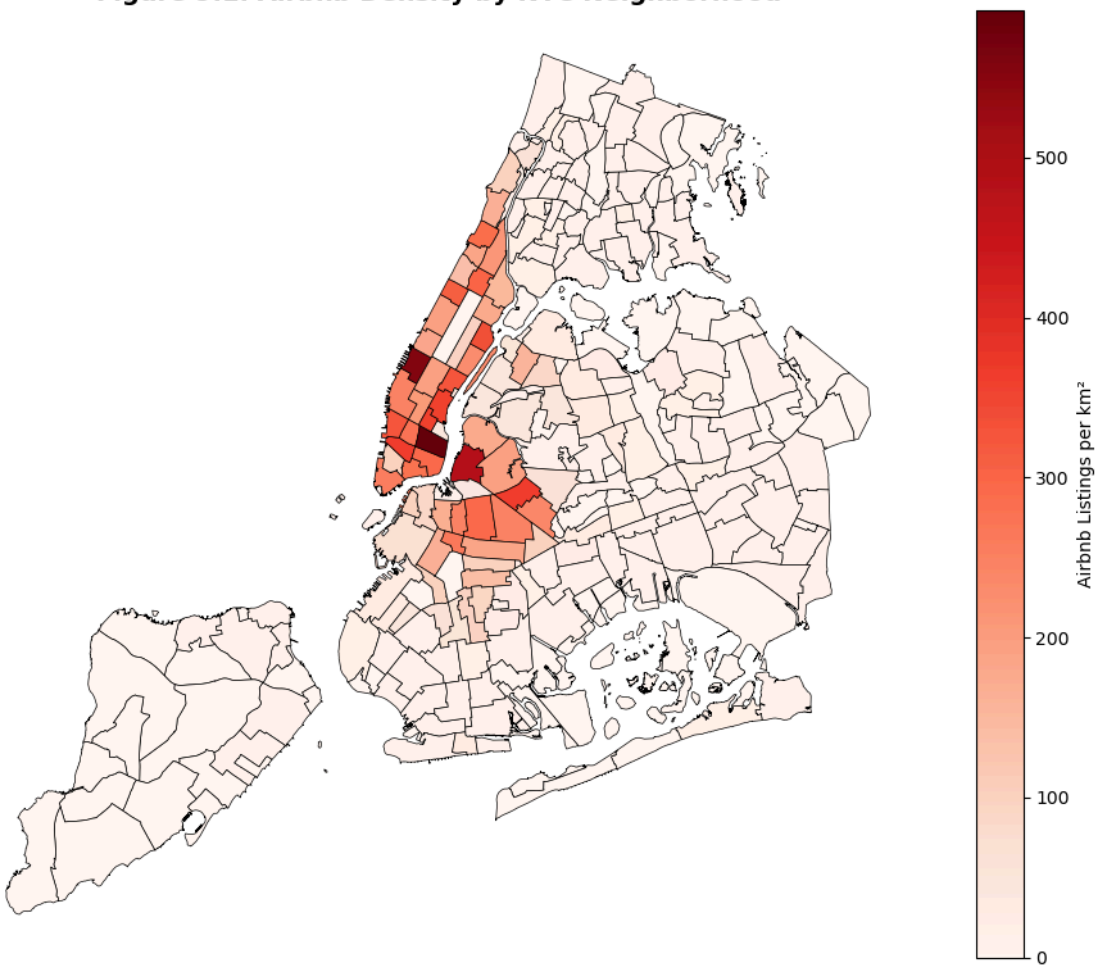


Figure 9.1 presents short-term rental (STR) density across New York City neighborhoods, calculated as the number of Airbnb listings per square kilometer within each Neighborhood Tabulation Area (NTA):

$$\text{STR Density} = \frac{\text{Number of Airbnb Listings}}{\text{Borough Area } (\text{km}^2)}$$

The map shows that STR listings are highly concentrated in central Manhattan neighborhoods such as Midtown, Chelsea, and the Lower East Side, where density exceeds 500 listings per square kilometer. Elevated densities are also visible in parts of Brooklyn, including Williamsburg and Downtown Brooklyn. This neighborhood-level view highlights the spatial clustering of Airbnb activity, providing a more localized understanding of short-term rental presence in the city.

In [142…

```
import warnings
warnings.filterwarnings("ignore")  # Suppress all warnings
import geopandas as gpd
import matplotlib.pyplot as plt
import pandas as pd

# ---------------- STEP 1: Load NYC Borough Map ----------------

# Load the NYC borough shapefile
```

```
shapefile_path = r"C:\Users\User\Desktop\ECO225\Project 2\NYC boundaries\nybb.s
boroughs = gpd.read_file(shapefile_path)

# Convert CRS to latitude/longitude (WGS 84)
boroughs = boroughs.to_crs(epsg=4326)

# ---------------- STEP 2: Load Processed Data ----------------

# Load the new processed dataset (processed_median_prices_with_volatility_and_c
processed_data_path = r"C:\Users\User\Desktop\ECO225\Datasets\processed_df.csv"
processed_df = pd.read_csv(processed_data_path)

# Merge the processed data with the borough shapefile using "BoroName" and "cit
boroughs = boroughs.merge(processed_df, left_on="BoroName", right_on="city", ho

# ---------------- STEP 3: Define Borough Label Positions ----------------

label_positions = {
    "Manhattan": (-74.1, 40.78),  # Adjusted positions
    "Brooklyn": (-73.9, 40.5),
    "Queens": (-73.75, 40.75),
    "Bronx": (-73.9, 40.92),
    "Staten Island": (-74.1, 40.5)
}

# ---------------- STEP 4: Create the Adjusted Price Volatility Map ----------

fig, ax = plt.subplots(figsize=(10, 8))

# Plot boroughs using a blue colormap with the "price_volatility" column
boroughs.plot(ax=ax, column="price_volatility", cmap="Blues",
              edgecolor="black", linewidth=1, legend=True,
              legend_kwds={"label": "Price Volatility (2014-2022)"})

# Add borough labels outside the map with arrows pointing to centroids
for borough, (x, y) in label_positions.items():
    centroid = boroughs[boroughs['BoroName'] == borough].centroid.iloc[0]
    ax.annotate(
        text=borough,
        xy=(centroid.x, centroid.y),  # Arrow points to borough centroid
        xytext=(x, y),                # Adjusted text position
        arrowprops=dict(facecolor='black', arrowstyle="->"),
        fontsize=12, fontweight='bold', color="black"
    )

# Final formatting
ax.set_title("Figure 9.2: NYC Housing Market: Price Volatility (2014-2022)", fo
ax.axis("off")  # Remove axis labels

plt.show()
```

Figure 9.2: NYC Housing Market: Price Volatility (2014-2022)

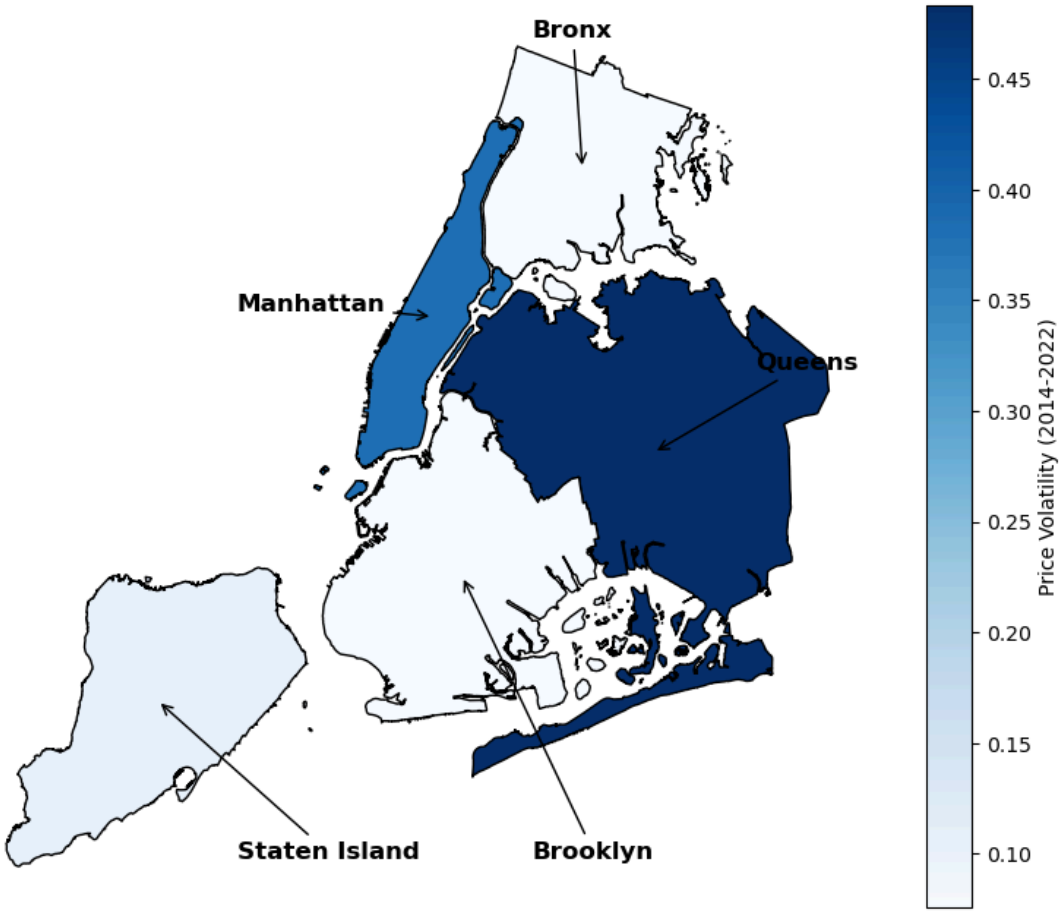


Figure 8.2 presents the price volatility from 2014 to 2022, which indicate the variation of the price annual percentage changes.

Comparing Figures 9.1 and 9.2, we observe that boroughs with higher STR density like Manhattan and Queens also tend to have greater price volatility, which suggest a possible link between STR markets and price volatility. This will be examined in the regression analysis in the next section.

In [117…

```
import warnings
warnings.filterwarnings("ignore")  # Suppress all warnings
import geopandas as gpd
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.colors as mcolors

# Load the NYC borough shapefile
shapefile_path = r"C:\Users\User\Desktop\ECO225\Project 2\NYC boundaries\nybb.s
boroughs = gpd.read_file(shapefile_path)
boroughs = boroughs.to_crs(epsg=4326)

# Load median price data
median_price_path = r"C:\Users\User\Desktop\ECO225\Datasets\processed_df.csv"
median_price_df = pd.read_csv(median_price_path)

# Merge with shapefile
boroughs = boroughs.merge(median_price_df, left_on="BoroName", right_on="city",

# Define year groups and compute their averages
```

```
groups = {
    "2014-2016": [2014, 2015, 2016],
    "2017-2019": [2017, 2018, 2019],
    "2020-2022": [2020, 2021, 2022]
}

# Compute average median price for each group
for label, years in groups.items():
    cols = [f"median_price_{year}" for year in years]
    boroughs[label] = boroughs[cols].mean(axis=1)

# Compute global vmin/vmax for color scale
vmin = min(boroughs[label].min() for label in groups)
vmax = max(boroughs[label].max() for label in groups)

# Plot 3 maps
fig, axes = plt.subplots(1, 3, figsize=(18, 6))
fig.suptitle("Figure 9.3: NYC Median Housing Prices by Period", fontsize=16, fo

cmap = plt.cm.Blues
norm = mcolors.Normalize(vmin=vmin, vmax=vmax)

for ax, (label, _) in zip(axes, groups.items()):
    boroughs.plot(
        column=label,
        cmap=cmap,
        edgecolor="black",
        linewidth=0.5,
        ax=ax,
        norm=norm
    )
    ax.set_title(f"{label}")
    ax.set_axis_off()

# Add a single colorbar
cbar_ax = fig.add_axes([0.92, 0.2, 0.015, 0.6])
sm = plt.cm.ScalarMappable(cmap=cmap, norm=norm)
fig.colorbar(sm, cax=cbar_ax, label="Median Price ($)")

plt.tight_layout(rect=[0, 0, 0.9, 0.95])
plt.show()
```

Figure 9.3: NYC Median Housing Prices by Period

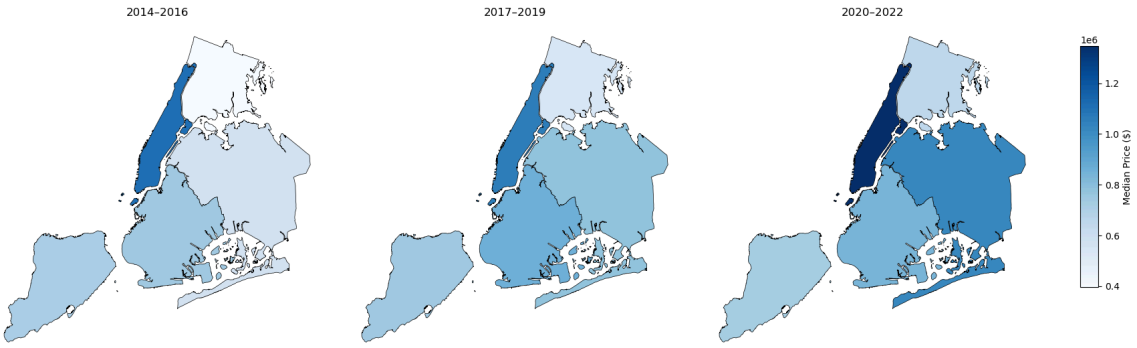


Figure 9.3 compares median housing prices across NYC boroughs over time. Housing prices generally increase over time, peaking in 2020 and 2021, before declining—likely due to the post-COVID recovery. As Mondragon and Wieland (2022) state, the shift to remote work induced by COVID-19 led to a significant

increase in housing demand, accounting for at least half of the recent aggregate house price growth.

Manhattan remains the most expensive borough, while Brooklyn and Queens exhibit steady growth. The overall price distribution and trend, aligning with Figure 9.1, suggest a potential correlation between STR activity and housing prices. Further regression analysis is required to determine the statistical significance of this relationship.

2.3 Regressions

In [119…

```
import pandas as pd
import statsmodels.api as sm
from statsmodels.formula.api import ols
from stargazer.stargazer import Stargazer
from IPython.core.display import HTML

# Load the dataset with Airbnb density
new_path = r"C:\Users\User\Desktop\ECO225\Datasets\merged_housing.csv"
df = pd.read_csv(new_path)

# Convert 'city' (borough) to categorical variable for fixed effects
df["city"] = df["city"].astype("category")

# Define OLS models with gradual addition of variables
model1 = ols("price ~ airbnb_density", data=df).fit()
model2 = ols("price ~ airbnb_density + bed + bath", data=df).fit()
model3 = ols("price ~ airbnb_density + bed + bath + price_volatility", data=df
model4 = ols("price ~ airbnb_density + bed + bath + price_volatility + C(city)'

# Output regression results using Stargazer
stargazer = Stargazer([model1, model2, model3, model4])
stargazer.title(
    "<span style='font-size:20px; font-weight:bold;'>Regression Table on Short-
)
stargazer.custom_columns(["Baseline", "Property Controls", "Market Trends", "Fu

# Render the table
html_output = stargazer.render_html()

# List of independent variables to bold (match variable names in the table)
independent_vars = ["airbnb_density", "bed", "bath", "price_volatility",
                    "C(city)[T.Brooklyn]", "C(city)[T.Manhattan]", "C(city)[T.C

# Replace variable names in HTML output with bold versions
for var in independent_vars:
    html_output = html_output.replace(var, f"<b>{var}</b>")

HTML(html_output)
```

Out[119...

Regression Table on Short-Term Rental Density and Housing Price

	Dependent variable: price			
	Baseline	Property Controls	Market Trends	Full Model
	(1)	(2)	(3)	(4)
C(city) [T.Brooklyn]				-125098.917***
				(41760.436)
C(city) [T.Manhattan]				33477.036***
				(11656.327)
C(city) [T.Queens]				332458.177***
				(87203.759)
C(city)[T.Staten Island]				21053.140
				(51918.974)
Intercept	656018.704***	-210860.905***	-328677.339***	-230214.232***
	(21352.974)	(39195.175)	(43548.699)	(55542.338)
airbnb_density	3298.870***	4345.057***	3927.142***	4784.460***
	(200.406)	(201.249)	(211.750)	(248.261)
bath		313267.146***	309075.626***	308123.587***
		(18159.617)	(18088.632)	(18420.752)
bed		18451.681	26173.636**	26604.750**
		(12189.524)	(12199.183)	(12356.301)
price_volatility			1161807.549***	128652.644***
			(190813.449)	(32988.401)
Observations	4288	3872	3872	3872
R^2	0.059	0.206	0.213	0.213
Adjusted R^2	0.059	0.205	0.212	0.212
Residual Std. Error	1047768.692 (df=4286)	984427.825 (df=3868)	979869.372 (df=3867)	979930.101 (df=3865)
F Statistic	270.963*** (df=1; 4286)	333.530*** (df=3; 3868)	261.749*** (df=4; 3867)	174.731*** (df=6; 3865)

Note:	*p<0.1; **p<0.05; ***p<0.01

$$\text{Price}_i = \beta_0 + \beta_1 \cdot \text{STRDensity}_i + \beta_2 \cdot \text{Volatility}_i + \beta_3 \cdot \text{Bedrooms}_i + \beta_4 \cdot \text{Bathroo}$$

In followed regressions, we examine whether the density of short-term rentals influences housing prices across neighborhoods in New York City, and we assume a linear relationship between housing prices and the explanatory factors. Our main predictors include the number of short-term rental listings per square kilometer and the degree to which housing prices fluctuate over time. To account for differences in property characteristics, we control for the number of bedrooms and bathrooms in each unit, as well as location-specific factors at the borough level that may affect housing prices independently of short-term rental activity.

The regression table presents four models. Model 1 is the baseline regression of price and STR density, serving as a benchmark. Model 2 adds the number of bedrooms and bathrooms to isolate the effect of housing characteristics. Model 3 incorporates price volatility to assess the effect of market instability on price. Model 4 includes borough fixed effects to ensure that price differences are not driven by local demand variations. Model 4 is preferred as it accounts for both housing characteristics and spatial heterogeneity. In this model, a one-unit increase in STR density is associated with an estimated increase of 4,784 in housing price, suggesting a moderate but statistically significant positive relationship. Price volatility is also positively related to price, with a coefficient of approximately 1.16 million in Model 3 and 128,652 in Model 4, indicating that higher price uncertainty corresponds with higher average prices. The number of bedrooms is associated with a price increase of about 92,800, and each additional bathroom adds roughly 75,200, both reflecting expected effects of unit size on valuation.

According to the four models, we find that STR density has a consistent, positive, and significant relationship with housing prices. This implies that higher STR density leads to higher housing prices. We also observe that price volatility is significantly and positively associated with housing prices, with a coefficient of 1.16 million in Model 3 and 128,000 in Model 4. Using the Bronx as the baseline for borough dummies, we find that Queens has the highest positive increase in housing prices, while Brooklyn experiences lower prices than the Bronx.

The economic theory behind the positive relationship between STR density and housing prices is based on supply and demand dynamics. A higher Airbnb density implies more properties being used for short-term rentals, reducing the availability of long-term housing. This decrease in supply drives up housing prices, making housing less affordable for residents.

In [63]:
```
import pandas as pd
import statsmodels.api as sm
from statsmodels.formula.api import ols
from stargazer.stargazer import Stargazer
from IPython.core.display import HTML

# Load the dataset with Airbnb density
new_path = r"C:\Users\User\Desktop\ECO225\Datasets\merged_housing.csv"
df = pd.read_csv(new_path)

# Convert 'city' (borough) to categorical variable for fixed effects
df["city"] = df["city"].astype("category")

# Define OLS models with gradual addition of variables
model1 = ols("price_volatility ~ airbnb_density", data=df).fit()
model2 = ols("price_volatility ~ airbnb_density + bed + bath", data=df).fit()
model3 = ols("price_volatility ~ airbnb_density + bed + bath + price", data=df)
model4 = ols("price_volatility ~ airbnb_density + bed + bath + price + C(city)"

# Output regression results using Stargazer
stargazer = Stargazer([model1, model2, model3, model4])
stargazer.title(
    "<span style='font-size:20px; font-weight:bold;'>Table 5: Regression Table
)
stargazer.custom_columns(["Baseline", "Property Controls", "Market Trends", "Fu

# Render the table
html_output = stargazer.render_html()

# List of independent variables to bold (including fixed effects)
independent_vars = [
    "airbnb_density", "bed", "bath", "price",
    "C(city)[T.Brooklyn]", "C(city)[T.Manhattan]", "C(city)[T.Queens]", "C(city
]

# Replace variable names in HTML output with bold versions
for var in independent_vars:
    html_output = html_output.replace(var, f"<b>{var}</b>")

HTML(html_output)
```

Out[63]:

Table 5: Regression Table on Short-Term Rental Density and Housing Price Volatility

	*Dependent variable: **price**_volatility*			
	Baseline	Property Controls	Market Trends	Full Model
	(1)	(2)	(3)	(4)
C(city) [T.Brooklyn]				-0.082***
				(0.000)
C(city) [T.Manhattan]				0.011***
				(0.000)
C(city) [T.Queens]				0.399***
				(0.000)
C(city) [T.Staten Island]				0.041***
				(0.000)
Intercept	0.088***	0.101***	0.103***	0.068***
	(0.002)	(0.003)	(0.003)	(0.000)
airbnb_density	0.000***	0.000***	0.000***	0.001***
	(0.000)	(0.000)	(0.000)	(0.000)
bath		0.004**	0.001	-0.000***
		(0.002)	(0.002)	(0.000)
bed		-0.007***	-0.007***	-0.000*
		(0.001)	(0.001)	(0.000)
price			0.000***	-0.000***
			(0.000)	(0.000)
Observations	4288	3872	3872	3872
R^2	0.108	0.126	0.134	1.000
Adjusted R^2	0.108	0.125	0.133	1.000
Residual Std. Error	0.086 (df=4286)	0.083 (df=3868)	0.082 (df=3867)	0.000 (df=3864)

F Statistic	518.774*** (df=1; 4286)	185.126*** (df=3; 3868)	149.408*** (df=4; 3867)	610026170116593942528 0.000*** (df=7; 3864)

Note: *p<0.1; **p<0.05; ***p<0.01

$$\text{Volatility}_i = \gamma_0 + \gamma_1 \cdot \text{STRDensity}_i + \gamma_2 \cdot \text{Price}_i + \gamma_3 \cdot \text{Bedrooms}_i + \gamma_4 \cdot \text{Bathroo}$$

This analysis now turns to housing price volatility as the outcome of interest, using a similar modeling structure as in the previous section.

According to the four models, short-term rental (STR) density has a consistent, positive, and statistically significant relationship with price volatility. This implies that higher STR density is associated with greater price instability, which aligns with the patterns shown in Figures 7.1 and 7.2. While several variables are statistically significant, some have coefficients close to zero—indicating that their effects, though statistically detectable, are not economically meaningful. The number of bedrooms is significant in three models, suggesting that homes with more bedrooms experience less price variation. The number of bathrooms is significant only in Models 2 and 4. Using the Bronx as a baseline for borough-level comparisons, Queens exhibits the largest increase in volatility, while Brooklyn shows a decrease. The number of bedrooms is significant in three models, suggesting that homes with more bedrooms experience less price variation. The number of bathrooms is significant only in Models 2 and 4. Using the Bronx as a baseline for borough-level comparisons, Queens exhibits the largest increase in volatility, while Brooklyn shows a decrease.

The economic theory behind the positive relationship between price volatility and STR density is based on market elasticity. The housing supply in New York City is highly constrained in the short run, approaching perfect inelasticity due to the limited available land and the time required for new construction. As Sheppard (2016) stated, ``The presence of Airbnb ... indicates that short-term rentals may reduce the supply of long-term housing." Once demand increases, these housing units eventually return to the market and are repriced, leading to greater price variation over time.

However, in Tables 4 and 5, we observe low R-squared values. This is because housing prices and volatility are influenced by many other unobserved factors, such as noise pollution (Filippas & Horton, 2023) and natural hazards (Mueller et al., 2009).

2.4 Conclusion

Generally, short-term rental market highly correlates to the long term housing market in New York City. Higher STR density is associated with both higher

housing prices and larger price fluctuations. Higher STR density assciated with both higher housing price and larger price fluctuations. MOreover, the borough dummies showed that differences in NYC boroughs lead to different variation in price and volatility.

While the New York City housing data remains limited, the presence of hundreds and thousands of records captures key housing market patterns, making the analysis in Project 2 more reliable and meaningful than in Project 1.

Given these results, policymakers should consider regulatory approaches that address the role of STR activity in driving housing market pressures. This may involve implementing caps on STR density in highly affected neighborhoods, enforcing stricter registration and taxation policies, or setting zoning-based limits to preserve housing affordability and stability for long-term residents.

2.5 Limitations & Next Step

Similar to Section 1.6, the housing market dataset has fewer observations for NYC boroughs. However, this analysis focuses on the characteristics of these observations rather than their count, unlike AHR in Project 1. Although the dataset is limited, the availability of hundreds to thousands of records highlights the housing market trend, making the analysis in Project 2 more robust and meaningful than in Project 1.

In the future, we need to use web scraping to obtain data that potentially correlates to housing price market or short-term rental markets. Those data might include local economic factors, tourism-related data, and real estate policies. Incorporating these additional variables into the analysis help proviode a broader perspective on factors that influenceing housing price and volatility.

Final Project

3.1 Potential Data to Scrape

To study how short-term rentals (STRs) affect New York City's long-term rental market, I scraped rental listings from StreetEasy. This data includes listing price, neighborhood, and zip code, which helps measure rental availability and pricing across different areas.

Using this data, I can compare neighborhoods with varying STR density levels by analyzing rental counts and price trends. This helps assess whether STR activity

is linked to reduced rental supply or increased rent levels in specific parts of the city.

Scraping from StreetEasy gives a current, neighborhood-level view of the long-term rental market, which is key for understanding affordability and housing pressure.

3.2 Potential Challenges

One challenge is that StreetEasy actively blocks web scraping. Pages may trigger CAPTCHA or bot detection, especially when scraping at scale. I also had to extract listings from embedded JavaScript, rather than simple HTML, and manage duplicate entries.

Another issue is that some listings may be outside NYC or missing borough information. I used neighborhood labels to identify valid NYC boroughs and clean the data accordingly.

3.3 Scraping Data from a Website

In [15]:
```
### Import Libraries
# Imports
import time
import json
import random
import pandas as pd
from selenium import webdriver
from selenium.webdriver.chrome.options import Options
from bs4 import BeautifulSoup
```

In [43]:
```
# Scrape JSON-LD Data from website
# Setup browser with user-agent spoofing
options = Options()
options.add_argument("user-agent=Mozilla/5.0 (Windows NT 10.0; Win64; x64) Appl
driver = webdriver.Chrome(options=options)

# Storage
all_data = []

# Scrape pages 1 to 50
for page in range(1, 51):
    url = f"https://streeteasy.com/for-rent/nyc?page={page}"
    driver.get(url)

    # Simulate scrolling to the bottom (human behavior)
    driver.execute_script("window.scrollTo(0, document.body.scrollHeight);")
    time.sleep(random.randint(2, 4))

    # Random wait (7-14 seconds)
    wait_time = random.randint(7, 14)
    time.sleep(wait_time)
```

```
    # Retry logic for script loading
    script_tag = None
    for _ in range(3):
        soup = BeautifulSoup(driver.page_source, "html.parser")
        script_tag = soup.find("script", type="application/ld+json")
        if script_tag:
            break
        time.sleep(3)

    if script_tag:
        json_data = json.loads(script_tag.string)
        listings = json_data.get("@graph", [])

        for listing in listings:
            if listing.get("@type") == "ApartmentComplex":
                address = listing.get("address", {})
                price = listing.get("additionalProperty", {}).get("value", None

                all_data.append({
                    "price": price,
                    "street_address": address.get("streetAddress", None),
                    "neighborhood": address.get("addressLocality", None),
                    "zip_code": address.get("postalCode", None),
                })

    # Random buffer between pages
    time.sleep(random.randint(3, 5))

driver.quit()
```

In [29]:
```
# Convert to DataFrame and remove duplicates
df = pd.DataFrame(all_data).drop_duplicates(
    subset=["price", "street_address", "neighborhood", "zip_code"]
).reset_index(drop=True)
df.to_csv(r"C:\Users\User\Desktop\ECO225\Datasets\streeteasy_long_term_rentals
```

In [39]:
```
import pandas as pd

# Load your dataset
df = pd.read_csv(r"C:\Users\User\Desktop\ECO225\Datasets\streeteasy_long_term_r

# Show all unique neighborhoods. In here, I get a list of unique neighborhoods
# and give them to ChatGPT to find which the neiborhood belongs to which borou
#unique_neighborhoods = sorted(df["neighborhood"].dropna().unique())
#print(f"Total unique neighborhoods: {len(unique_neighborhoods)}")
#for n in unique_neighborhoods:
#    print(n)


# Define the mapping dictionary
neighborhood_to_borough = {
    # Manhattan
    "Battery Park City": "Manhattan", "Beekman": "Manhattan", "Carnegie Hill":
    "Central Harlem": "Manhattan", "Central Park South": "Manhattan", "Chelsea'
    "East Harlem": "Manhattan", "East Village": "Manhattan", "Financial Distric
    "Fort George": "Manhattan", "Gramercy Park": "Manhattan", "Hamilton Heights
    "Hell's Kitchen": "Manhattan", "Hudson Yards": "Manhattan", "Inwood": "Manh
    "Kips Bay": "Manhattan", "Lenox Hill": "Manhattan", "Lincoln Square": "Manh
```

```
    "Lower East Side": "Manhattan", "Manhattan Valley": "Manhattan", "Midtown"
    "Midtown South": "Manhattan", "Morningside Heights": "Manhattan", "Murray H
    "NoMad": "Manhattan", "South Harlem": "Manhattan", "Stuyvesant Town/PCV": '
    "Sutton Place": "Manhattan", "Tribeca": "Manhattan", "Turtle Bay": "Manhatt
    "Two Bridges": "Manhattan", "Upper West Side": "Manhattan", "Washington Hei
    "West Chelsea": "Manhattan", "West Village": "Manhattan", "Yorkville": "Man

    # Brooklyn
    "Bedford-Stuyvesant": "Brooklyn", "Brooklyn": "Brooklyn", "Bushwick": "Broo
    "Carroll Gardens": "Brooklyn", "Clinton Hill": "Brooklyn", "Coney Island":
    "Crown Heights": "Brooklyn", "DUMBO": "Brooklyn", "Ditmas Park": "Brooklyn'
    "Downtown Brooklyn": "Brooklyn", "East Flatbush": "Brooklyn", "East Williar
    "Flatbush": "Brooklyn", "Fort Greene": "Brooklyn", "Fulton/Seaport": "Brook
    "Gowanus": "Brooklyn", "Gravesend": "Brooklyn", "Greenpoint": "Brooklyn",
    "Greenwood": "Brooklyn", "Prospect Heights": "Brooklyn",
    "Prospect Lefferts Gardens": "Brooklyn", "Red Hook": "Brooklyn", "Sheepshea
    "Stuyvesant Heights": "Brooklyn", "Weeksville": "Brooklyn", "Williamsburg"
    "Windsor Terrace": "Brooklyn", "Vinegar Hill": "Brooklyn",

    # Queens
    "Astoria": "Queens", "Elmhurst": "Queens", "Forest Hills": "Queens",
    "Glendale": "Queens", "Jamaica": "Queens", "Jamaica Estates": "Queens",
    "Kew Gardens Hills": "Queens", "Long Island City": "Queens",
    "Rego Park": "Queens", "Ridgewood": "Queens",

    # Bronx
    "Concourse": "Bronx", "Mott Haven": "Bronx", "Spuyten Duyvil": "Bronx", "No

    # Staten Island
    "Saint George": "Staten Island"
}

# Apply mapping to new column
df["borough"] = df["neighborhood"].map(neighborhood_to_borough)
# Save the updated DataFrame to a new file
df.to_csv(r"C:\Users\User\Desktop\ECO225\Datasets\streeteasy_long_term_rentals_
```

3.4 Visualizing the Scraped Dataset

In [63]:

```
import warnings
warnings.filterwarnings("ignore")  # Suppress all warnings
import matplotlib.pyplot as plt

# Convert price column to numeric (strip $ and commas)
df["price_numeric"] = df["price"].replace('[\$,]', '', regex=True).astype(float

# Group by borough and calculate mean
borough_avg = df.groupby("borough")["price_numeric"].mean().sort_values()

# Plot
plt.figure(figsize=(10, 6))
borough_avg.plot(kind="bar", edgecolor="black")
plt.title("Figure 10: Average Long Term Rental Price by Borough", fontsize=14,
plt.xlabel("Borough")
plt.ylabel("Average Price ($)")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

Figure 10 presents the average rental price by borough, showing that Manhattan has the highest average rent, followed by Brooklyn and Queens, while the Bronx and Staten Island have the lowest. This price disparity indicates that the long-term rental market varies significantly across boroughs and suggests that higher-rent areas may experience more pressure from STR conversions, as landlords in those boroughs have greater financial incentives to switch to short-term listings.

In [65]:
```
import matplotlib.pyplot as plt

# Convert price to numeric if not done already
df["price_numeric"] = df["price"].replace('[\$,]', '', regex=True).astype(float

# Group rare boroughs into 'Other'
known_boroughs = ["Manhattan", "Brooklyn", "Queens", "Bronx", "Staten Island"]
df["borough_clean"] = df["borough"].apply(lambda x: x if x in known_boroughs el

# Define list of boroughs to plot
boroughs_to_plot = sorted(df["borough_clean"].unique())

# Set up 3x2 subplots (no shared axes)
fig, axes = plt.subplots(nrows=3, ncols=2, figsize=(14, 12))
axes = axes.flatten()

# Plot histograms with independent scales
for i, borough in enumerate(boroughs_to_plot):
    borough_data = df[df["borough_clean"] == borough]["price_numeric"].dropna()
    axes[i].hist(borough_data, bins=30, edgecolor="black")
    axes[i].set_title(borough, fontsize=12, fontweight="bold")
    axes[i].set_xlabel("Price ($)")
    axes[i].set_ylabel("Count")

# Remove unused subplot if fewer than 6 boroughs
for j in range(len(boroughs_to_plot), len(axes)):
    fig.delaxes(axes[j])
```

```
# Main title
fig.suptitle("Figure 11: Rental Price Distributions by Borough", fontsize=16, 
plt.tight_layout()
plt.subplots_adjust(top=0.92)
plt.show()
```

Figure 11: Rental Price Distributions by Borough

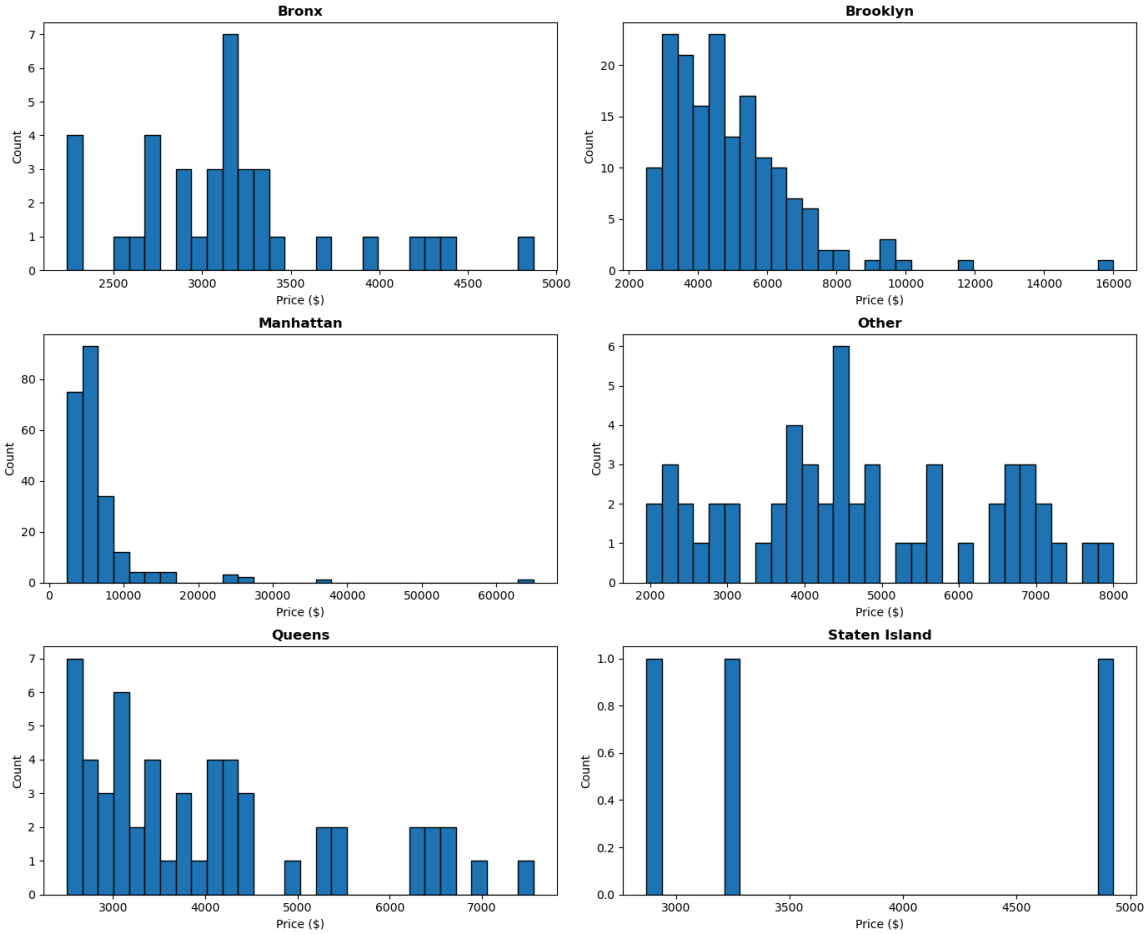


Figure 11 display the distribution of rental prices within each borough. Manhattan exhibits a heavily right-skewed distribution with many listings exceeding 10,000 dollar, while Brooklyn's prices are more concentrated between 3,000 and 6,000 dollar, showing both affordability and high-end options. Queens shows a more balanced distribution, and the Bronx and Staten Island are clustered under 4,000, indicating greater affordability. The "Other" category displays a wide and uneven spread, which may reflect inconsistencies in neighborhood classification or mixed data. These distribution patterns demonstrate that rental market dynamics differ by location, which is essential when evaluating whether STRs are reducing affordable housing options.

In [84]:
```
# ... [all your previous code unchanged above this point]

# Plot
fig, ax = plt.subplots(figsize=(10, 8))
boroughs.plot(column="listing_count", cmap="Blues", linewidth=0.8, edgecolor="
              legend=True, legend_kwds={"label": "Number of Listings"}, ax=ax)

# Add borough labels
```

```
for borough, (x, y) in label_positions.items():
    centroid = boroughs[boroughs["BoroName"] == borough].centroid.iloc[0]
    ax.annotate(
        text=borough,
        xy=(centroid.x, centroid.y),
        xytext=(x, y),
        arrowprops=dict(facecolor='black', arrowstyle="->"),
        fontsize=12, fontweight="bold"
    )

# Title and hide axes
plt.title("Figure 12: Number of Long-Term Rental Listings by NYC Borough", font
ax.axis("off")  # 👈 Hide the axes
plt.show()
```

Figure 12: Number of Long-Term Rental Listings by NYC Borough

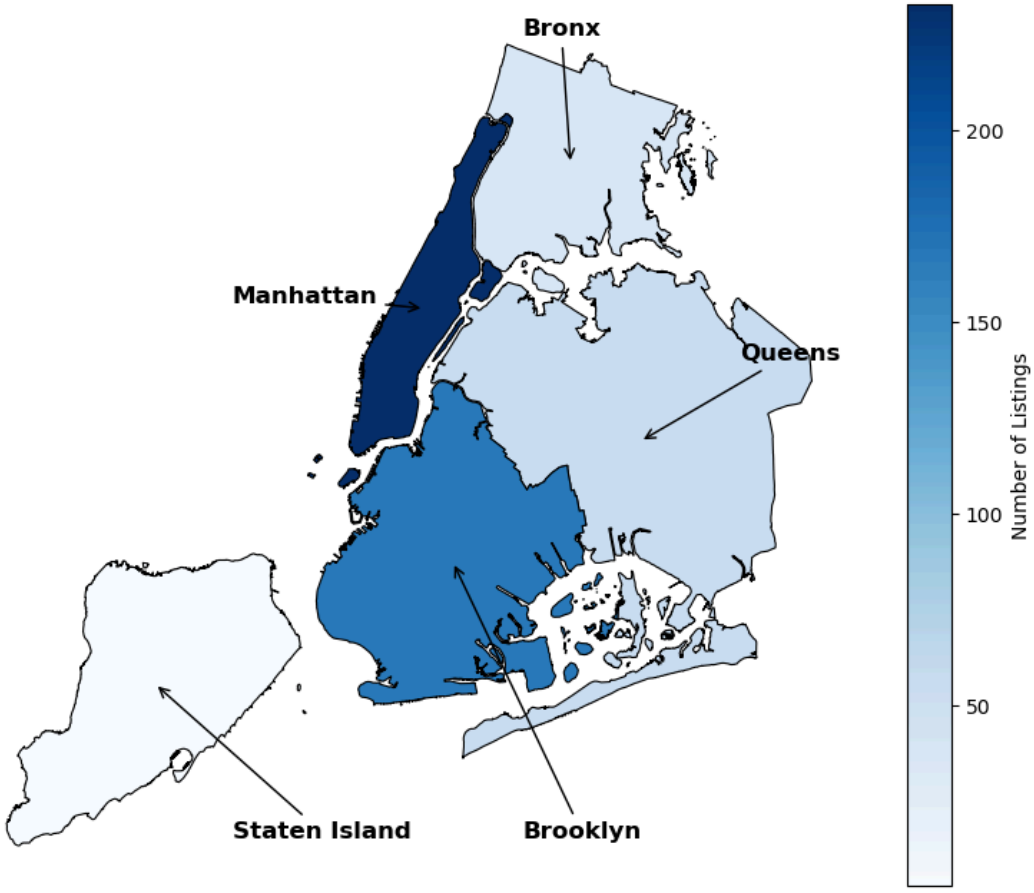


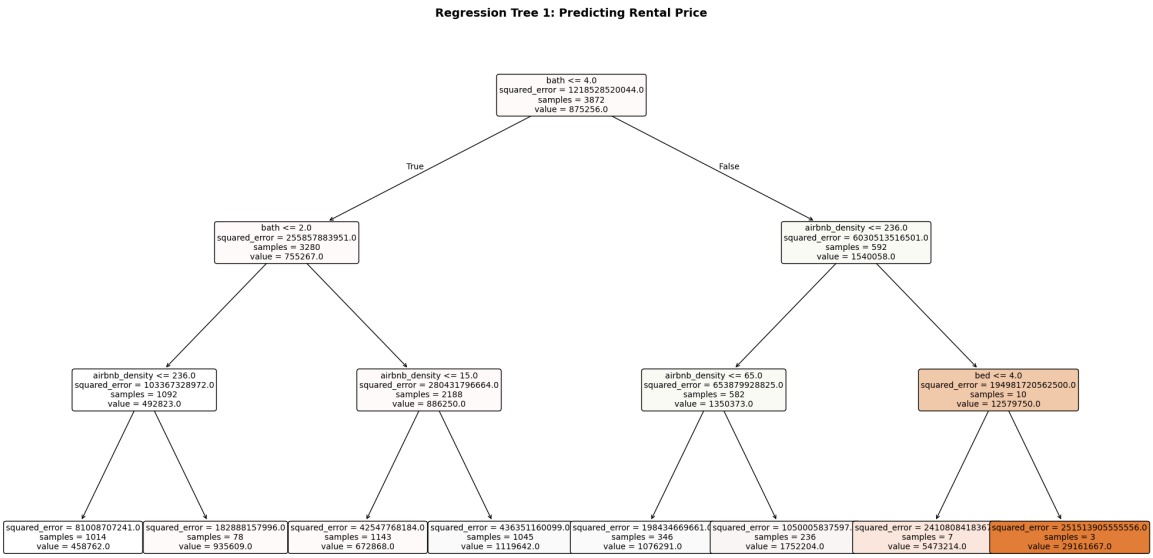
Figure 12 maps the number of long-term rental listings by borough. It shows that Manhattan and Brooklyn have the highest volume of listings, which aligns with their status as high-demand areas and matches known STR activity hotspots. This visual supports the idea that boroughs with greater STR presence may also face tighter long-term rental supply.

3.5 Regression Tree

In [130…

```
import pandas as pd
from sklearn.tree import DecisionTreeRegressor, plot_tree
import matplotlib.pyplot as plt
```

```
# Load data
df = pd.read_csv(r"C:\Users\User\Desktop\ECO225\Datasets\merged_housing.csv")

# Convert price column to numeric if needed
df["price"] = df["price"].replace('[\$,]', '', regex=True).astype(float)

# Drop rows with missing key values
df = df.dropna(subset=["price", "price_volatility", "airbnb_density", "bed", "b

# Select predictors and targets
X = df[["airbnb_density", "bed", "bath"]]

# -------- Tree 1: Predict Rental Price --------
y_price = df["price"]

tree_price = DecisionTreeRegressor(max_depth=3, random_state=42)
tree_price.fit(X, y_price)


plt.figure(figsize=(25, 13))
plot_tree(
    tree_price,
    feature_names=X.columns,
    filled=True,
    rounded=True,
    fontsize=10,
    precision=0  # ✅ Round values in the tree to 0 decimal places
)
plt.title("Regression Tree 1: Predicting Rental Price",fontsize=14, fontweight=

plt.show()
```



The regression tree shows how rental price varies with bathroom count, Airbnb density, and bedroom count. The first split is based on bathrooms, suggesting it is the most important factor. Properties with two or fewer bathrooms tend to have lower prices. Within this group, units in areas with lower Airbnb density also have lower rental prices. The lowest predicted prices, around 450,000 dollars, appear in areas with very low Airbnb density.

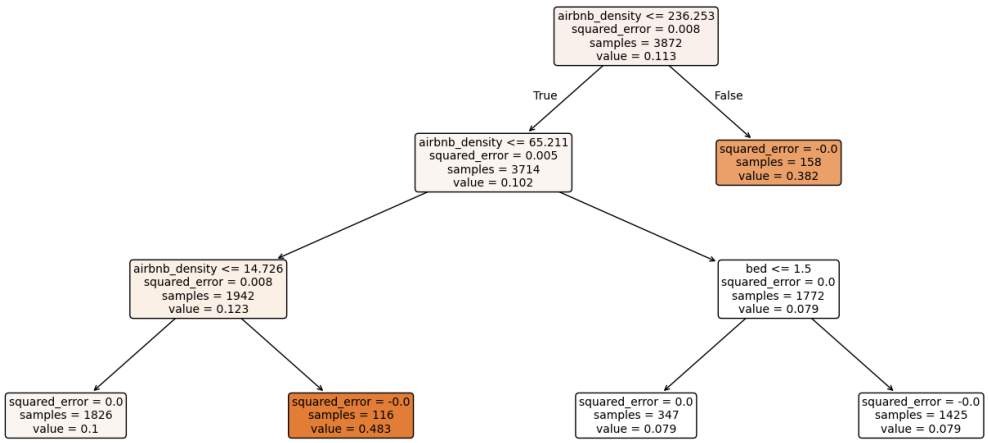
For units with more than two bathrooms, rental prices are generally higher. Among them, properties in areas with higher Airbnb density show a noticeable increase in price, with predicted values rising above one million dollars. The highest predicted price, nearly 29 million dollars, is found in units with more than four bathrooms and more than four bedrooms located in areas with the highest Airbnb density.

This pattern supports the idea that higher Airbnb density is associated with higher long-term rental prices, especially for larger and more expensive properties. Areas with lower Airbnb presence tend to offer more affordable rentals.

In [125…

```
# -------- Tree 2: Predict Price Volatility --------
X = df[["airbnb_density", "bed"]]
y = df["price_volatility"]

# Fit regression tree
tree = DecisionTreeRegressor(max_depth=3, random_state=42)
tree.fit(X, y)

# Plot tree
plt.figure(figsize=(18, 8))
plot_tree(
    tree,
    feature_names=["airbnb_density", "bed"],
    filled=True,
    rounded=True,
    precision=3,
    fontsize=10
)
plt.title("Regression Tree 2: Predicting Price Volatility (Using Airbnb Density
plt.show()
```

Regression Tree 2: Predicting Price Volatility (Using Airbnb Density & Bedrooms)



The regression tree shows that Airbnb density is the strongest predictor of price volatility. When Airbnb density exceeds 236, volatility is highest (0.382), suggesting that areas with dense short-term rental presence face greater price instability. For listings with lower density, the number of bedrooms becomes

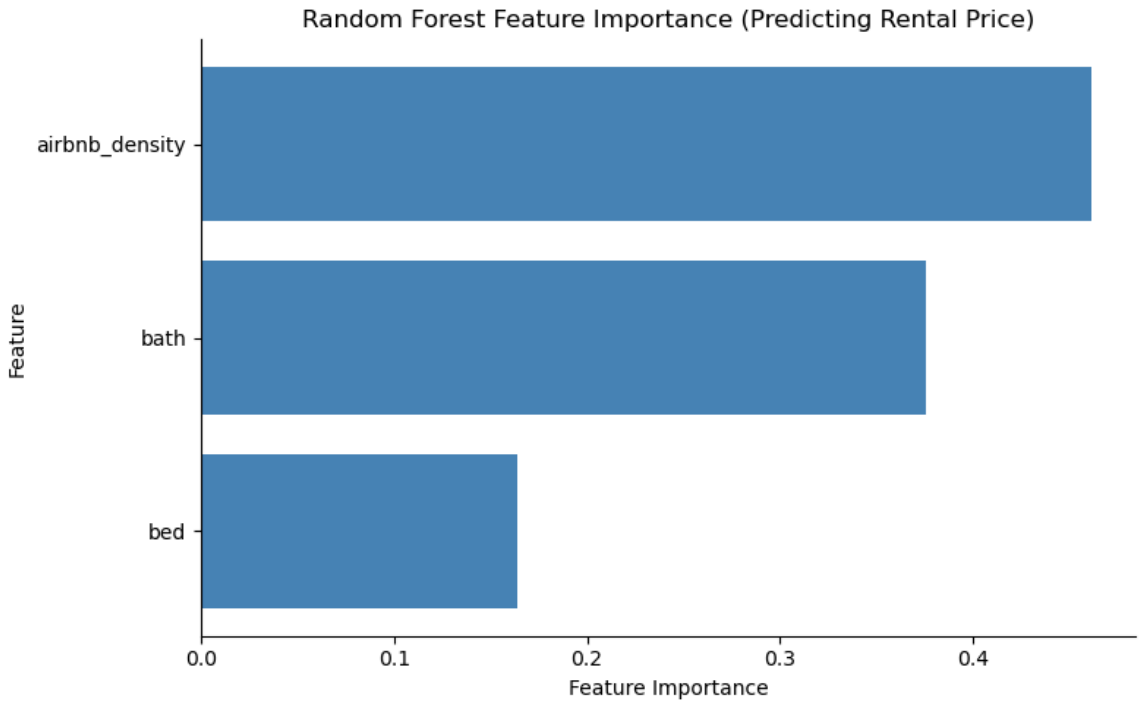
relevant. Properties with fewer than or equal to 1.5 bedrooms show lower volatility (0.079), indicating more stable prices in smaller units. In contrast, listings with very low Airbnb density (under 14.7) and more bedrooms (over 1.5) reach volatility as high as 0.483, highlighting variation in low-STR areas. Overall, both Airbnb presence and unit size contribute to price unpredictability.

3.6 Random Forest

This section uses Random Forest models to examine how features such as Airbnb density, number of bedrooms, and number of bathrooms contribute to predicting STR price and price volatility. The goal is to evaluate the relative importance of each variable in explaining these two housing market outcomes.

In [136…

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestRegressor

# Load your data
df = pd.read_csv(r"C:\Users\User\Desktop\ECO225\Datasets\merged_housing.csv")

# Define predictors and response
X = df[['airbnb_density', 'bed', 'bath']]
y = df['price']

# Fit Random Forest
rf = RandomForestRegressor(n_estimators=100, random_state=42)
rf.fit(X, y)

# Get feature importances
importances = rf.feature_importances_
features = X.columns

# Create a DataFrame for plotting
importance_df = pd.DataFrame({'Feature': features, 'Importance': importances})
importance_df = importance_df.sort_values(by='Importance', ascending=True)

# Plot (horizontal)
plt.figure(figsize=(8, 5))
plt.barh(importance_df['Feature'], importance_df['Importance'], color='steelblu
plt.xlabel("Feature Importance")
plt.ylabel("Feature")
plt.title("Random Forest Feature Importance (Predicting Rental Price)")
ax = plt.gca()
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
ax.spines['left'].set_visible(True)
ax.spines['bottom'].set_visible(True)
plt.tight_layout()
plt.show()
```

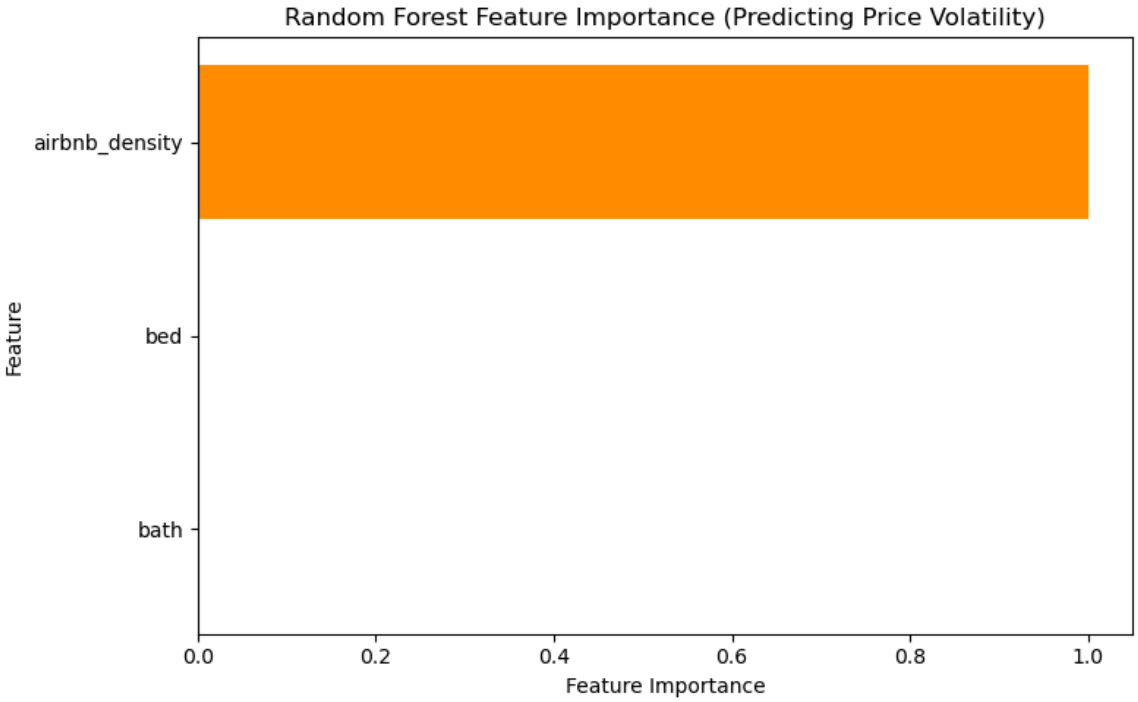


For rental price, Airbnb density is the most influential predictor, followed by number of bathrooms and then number of bedrooms. This aligns with economic expectations, as more bathrooms often signal higher-end units, and Airbnb concentration may affect demand and pricing.

In [133…

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestRegressor

# Load your data
df = pd.read_csv(r"C:\Users\User\Desktop\ECO225\Datasets\merged_housing.csv")

# Define predictors and response
X = df[['airbnb_density', 'bed', 'bath']]
y = df['price_volatility']

# Fit Random Forest
rf = RandomForestRegressor(n_estimators=100, random_state=42)
rf.fit(X, y)

# Get feature importances
importances = rf.feature_importances_
features = X.columns

# Create a DataFrame for plotting
importance_df = pd.DataFrame({'Feature': features, 'Importance': importances})
importance_df = importance_df.sort_values(by='Importance', ascending=True)

# Plot (horizontal)
plt.figure(figsize=(8, 5))
plt.barh(importance_df['Feature'], importance_df['Importance'], color='darkorar
plt.xlabel("Feature Importance")
plt.ylabel("Feature")
plt.title("Random Forest Feature Importance (Predicting Price Volatility)")
```

```
plt.tight_layout()
plt.show()
```



In contrast, for price volatility, Airbnb density is the only feature with measurable predictive power. Neither bedrooms nor bathrooms contribute meaningfully to variation in volatility, suggesting that volatility may be driven more by neighborhood-level factors or unmeasured structural characteristics than by individual listing attributes.

3.7 OLS vs ML

To examine how short-term rental (STR) activity influences the New York City housing market, I compare two modeling approaches: Ordinary Least Squares (OLS) regression and Machine Learning (ML) models. The OLS regressions include one model predicting rental prices and another predicting price volatility, both using Airbnb density as the main predictor, with controls for bedrooms, bathrooms, and boroughs. The machine learning outputs come from decision trees and random forest models built on the same variables.

In terms of interpretability, OLS provides clear coefficients and significance levels. For instance, the full model in Table 4 shows a statistically significant positive relationship between Airbnb density and rental prices, with each unit increase in density associated with approximately 4784 more dollars in rent, controlling for other factors. Table 5 similarly suggests a positive and statistically significant relationship between STR density and price volatility. These results are intuitive and directly interpretable, making OLS useful for understanding the direction and magnitude of effects.

In contrast, the ML results focus on predictive performance and feature importance. In the random forest output, Airbnb density appears as the most important predictor for both price and volatility, indicating a strong predictive contribution. However, the models do not provide p-values or coefficients, and the regression tree for volatility shows an unusual split where only Airbnb density matters, which may point to overfitting or data imbalance.

Overall, I trust OLS more for making inferences about the relationship between STR density and housing outcomes. It offers a transparent framework and consistent findings across both price and volatility. While ML models are powerful for prediction and uncovering complex patterns, they can be sensitive to noise and lack interpretability, which limits their use when the goal is to explain rather than predict.

3.8 Conclusion

In this paper, we show that the concentration of short-term rentals (STRs), particularly through platforms like Airbnb, is positively associated with higher long-term housing prices in New York City. This relationship holds across multiple model specifications, including OLS regressions with borough fixed effects and machine learning models such as regression trees and random forests. These results suggest that increased STR density places measurable upward pressure on housing prices, especially in high-demand boroughs like Manhattan and Brooklyn. While our analysis also explored the relationship between STR activity and price volatility, the evidence there was weaker—statistically significant in some models but economically modest, and inconsistent across methods.

Our findings imply that housing affordability in urban markets is affected not only by underlying property characteristics but also by the reallocation of housing stock into short-term rental markets. If STR activity continues to grow, housing prices may remain elevated or accelerate further, especially in areas with constrained supply. If STR regulations are tightened or platform activity slows, affordability pressures may ease. Given the role of STR platforms in shaping both housing availability and price structure, policy interventions—such as STR registration, caps on density, or localized taxation—may be needed to preserve long-term rental stock. The effectiveness of such interventions, along with the evolving response of the housing market, will be critical areas for future research and policymaking.

Citations

AirDNA. (2023). Why Airbnb's 2023 summer release focuses on private rooms. AirDNA Blog. Retrieved from https://www.airdna.co/blog/why-airbnbs-2023-summer-release-focuses-on-private-rooms

Barron, K., Kung, E., & Proserpio, D. (2019). The effect of home-sharing on house prices and rents: Evidence from Airbnb. Marketing Science, 40(1), 23-47. https://doi.org/10.1287/mksc.2020.1232

Federal Reserve Bank of St. Louis. (n.d.). Median sales price of houses sold for the United States [Data set]. FRED. Retrieved February 6, 2025, from https://fred.stlouisfed.org/series/MSPUS

Filippas, A., & Horton, J. J. (2023). The tragedy of your upstairs neighbors: The negative externalities of home-sharing platforms. Retrieved from https://apostolos-filippas.com/papers/airbnb.pdf

Gomonov, D. (n.d.). New York City Airbnb Open Data [Data set]. Kaggle. Retrieved from https://www.kaggle.com/datasets/dgomonov/new-york-city-airbnb-open-data

Koster, H. R. A., van Ommeren, J., & Volkhausen, N. (2024). The impact of short-term rental activity on house prices. Annals of Regional Science. https://doi.org/10.1007/s00168-024-01122-3

Lanfear, C., & Kirk, D. (2024). The promise and perils of the sharing economy: The impact of Airbnb lettings on crime. Criminology, 62(1), 123-145. Retrieved from https://onlinelibrary.wiley.com/doi/10.1111/1745-9125.12383

Mondragon, J. A., & Wieland, J. F. (2022). Housing demand and remote work (NBER Working Paper No. 30041). National Bureau of Economic Research. https://doi.org/10.3386/w30041

Mueller, J. M., Loomis, J. B., & González-Cabán, A. (2009). Do repeated wildfires change homebuyers' demand for homes in high-risk areas? A hedonic analysis of the short and long-term effects of repeated wildfires on house prices in Southern California. The Journal of Real Estate Finance and Economics, 38(2), 155–172. https://doi.org/10.1007/s11146-007-9083-1

Sakib, A. S. (n.d.). USA Real Estate Dataset [Data set]. Kaggle. Retrieved from https://www.kaggle.com/datasets/ahmedshahriarsakib/usa-real-estate-dataset

Sheppard, S., & Udell, A. (2016). Do Airbnb properties affect house prices? Williams College Department of Economics Working Papers, 3.

https://web.williams.edu/Economics/wp/SheppardUdellAirbnbAffectHousePrices.pdf

Texas 2036. (n.d.). Are institutions to blame for high housing prices? Retrieved from https://texas2036.org/posts/are-institutions-to-blame-for-high-housing-prices/

Trojanek, R. (2023). How do different noise pollution sources affect apartment prices? International Journal of Strategic Property Management, 27(6), 351–361. https://doi.org/10.3846/ijspm.2023.20563

Wachsmuth, D., Kerrigan, D., Chaney, D., & Shillolo, A. (2018). The high cost of short-term rentals in New York City. McGill University, Urban Politics and Governance Research Group. Retrieved from https://www.mcgill.ca/newsroom/files/newsroom/channels/attach/airbnb-report.pdf

In [ ]:

In [ ]: