STAT-5330 Project: Analyzing Real Estate Market Trends and Price Prediction

Yidan Chen

Abstract

In this project, We present a clustering analysis of New York City's real estate market, focusing on geographic and pricing patterns. Properties are grouped into distinct clusters using k-means, revealing differences in housing characteristics across boroughs.

Contents

1 Introduction **1**

2 Literature review **1**

3 Analysis & Results **2**

4 Discussion and Conclusion **6**

A Code **7**

1 Introduction

The housing market is a cornerstone of economic activity and urban development. In New York City, where housing prices vary dramatically, understanding the factors that influence these prices is crucial for buyers, sellers, and policymakers. This study addresses two key questions: **What are the most important factors affecting housing prices?** and **Can we develop an accurate model to predict home prices based on property characteristics?**

To answer these questions, we use a dataset of real estate listings from New York City. The analysis begins with clustering methods, which segment the housing market based on property characteristics and location, highlighting distinct price patterns. We then apply regression analysis to identify significant price determinants, including property size, bedrooms, bathrooms, property type, and location. These methods help us compare multiple predictive models, evaluating their performance using metrics such as AIC and Adjusted R^2.

2 Literature review

Clustering techniques are widely used in housing market studies to identify market segments by grouping properties with similar price trends. [1][2] This study extends these methods by employing location-based clustering to analyze spatial price variations in New York City. Unlike traditional zoning by boroughs (e.g., Manhattan or Brooklyn), we use geographic coordinates to define clusters, providing a finer-grained perspective on pricing patterns.

Regression analysis is a key tool for predicting housing prices. Prior research has demonstrated the importance of attributes like square footage and property type, often using interaction terms to improve accuracy. For instance, interactions between property size and neighborhood characteristics have been shown to enhance predictions. [3] Our approach combines these predictors with location-based clusters to capture both property-level and spatial effects.

3 Analysis & Results

We began by exploring the dataset to understand key variable distributions and relationships. Data cleaning identified anomalies, such as high-priced outliers, duplicate properties with differing prices, and fractional values for bathrooms and bedrooms. To address skewness, we excluded properties above the 95th price percentile. Initially, the price distribution was right-skewed, dominated by lower-priced properties and a few outliers. After removing duplicates and applying a logarithmic transformation to prices, the distribution became more symmetrical, improving its suitability for modeling.

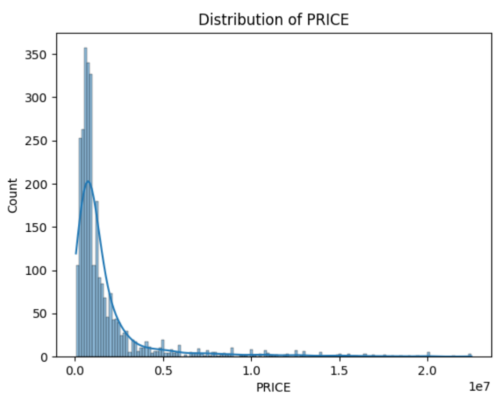


Figure 1: Initial Price Distribution

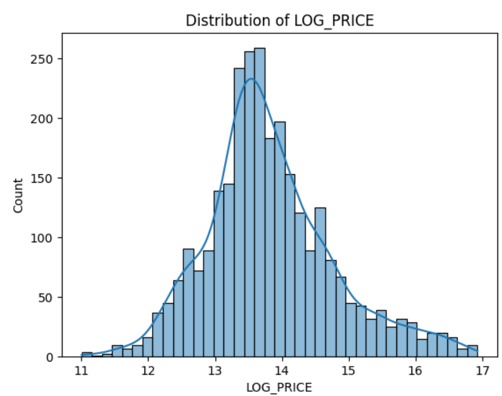
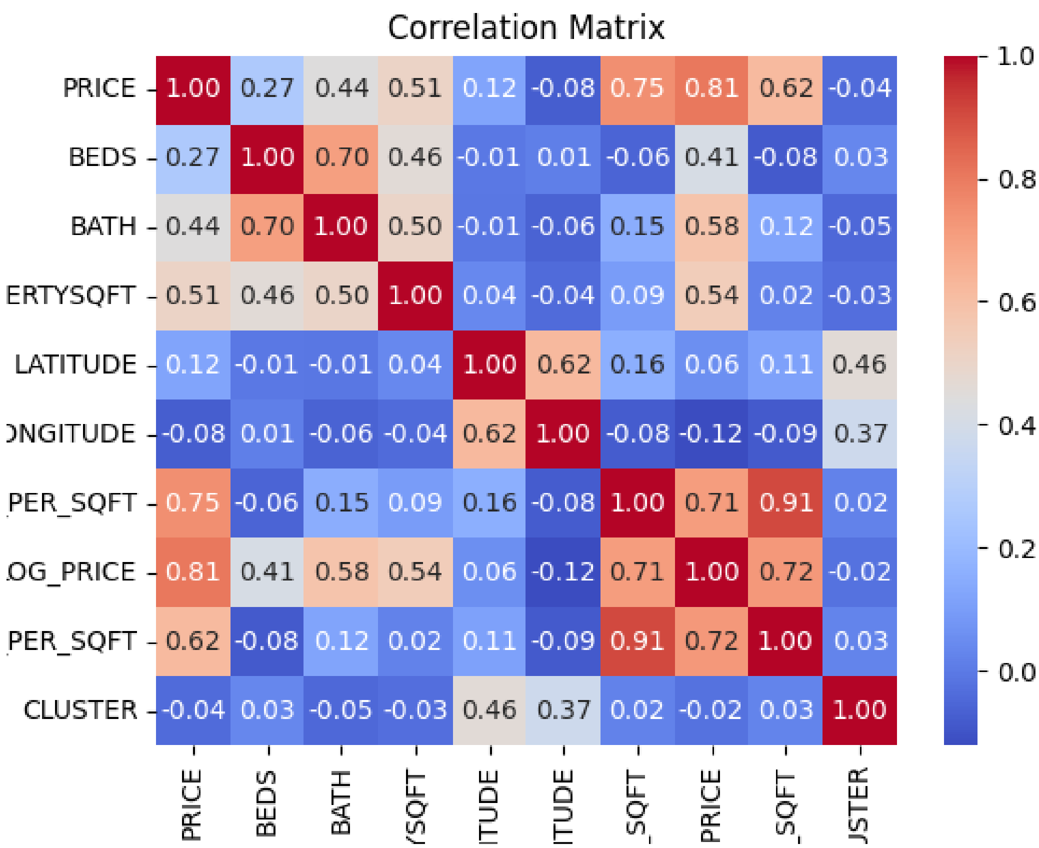
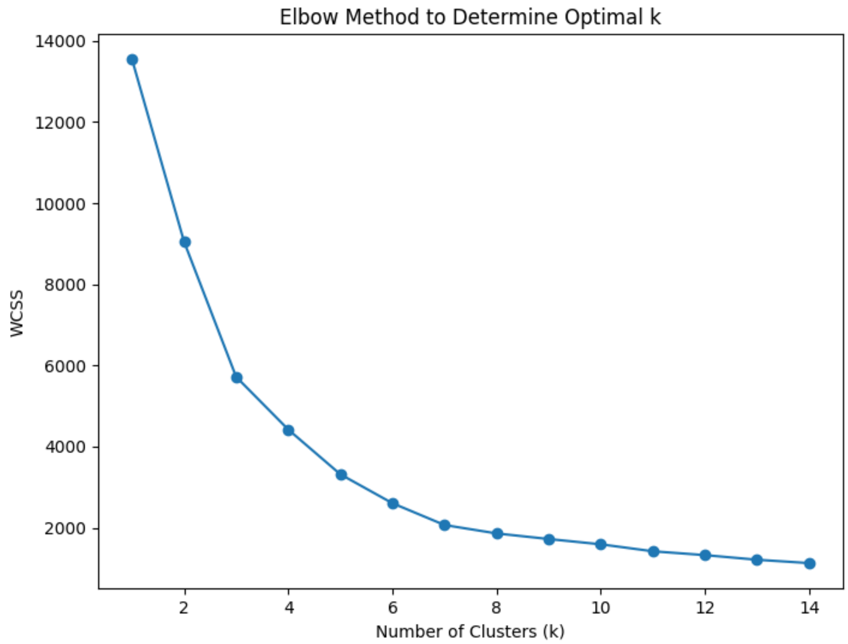
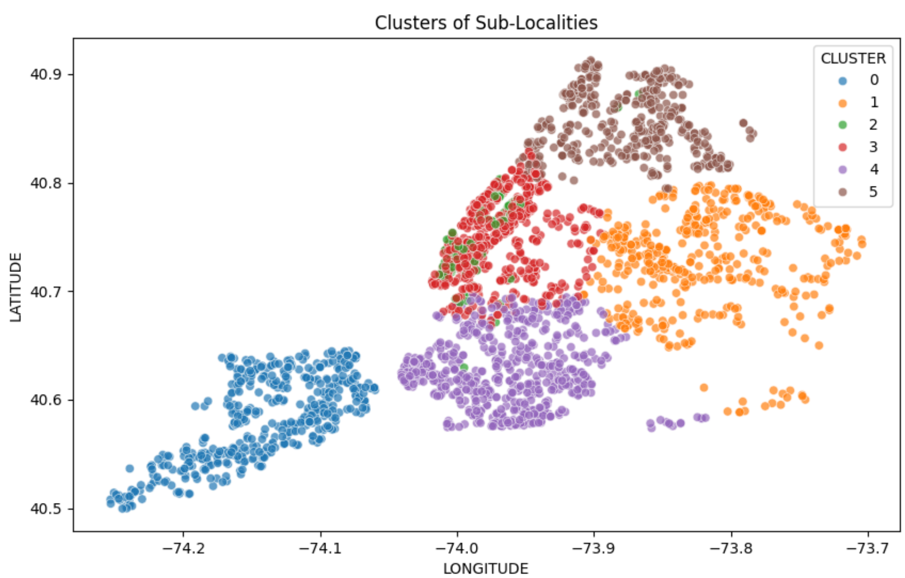


Figure 2: Logarithmic transformation



We then investigated the relationship between price and geographic location (latitude and longitude). Using K-means clustering, we segmented properties into geographic clusters based on their coordinates. This approach provided a more precise spatial segmentation compared to administrative sub-local divisions, revealing distinct price patterns across different areas. High-priced clusters typically aligned with central urban areas, while lower-priced clusters were located on the peripheries. Furthermore, clustering properties based on price and size revealed a clear segmentation of larger, high-value properties versus smaller, more affordable options. These clusters were visualized using scatter plots (price vs. property square footage) and geographic maps. Consistent color coding across clusters facilitated the identification of spatial and numerical patterns.



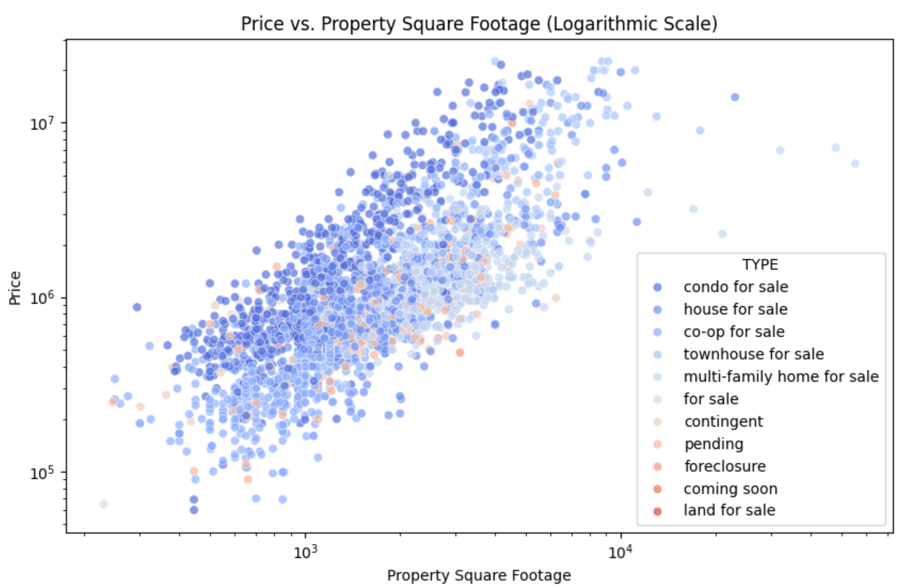


In the regression analysis, we utilized **R** to construct and refine models predicting property prices. Initially, we compared models incorporating `SUBLOCALITY` (administrative divisions) and `CLUSTER` (derived from K-means clustering of latitude and longitude). The cluster-based model demonstrated a better fit with a lower AIC, emphasizing the effectiveness of spatial clustering in capturing price variations.

Through backward selection, we refined the model to its final form:

$$PRICE \sim PROPERTYSQFT + BEDS + BATH + TYPE + CLUSTER.$$

This model had the lowest AIC and identified key factors influencing prices: First is Square footage (PROPERTYSQFT), whcih is a highly significant factor, showing a strong positive influence on price. Then they are Bathrooms (BATH) and bedrooms (BEDS), both significantly contributed to price predictions, with bathrooms having a larger impact. For certain clusters, like CLUSTER2 and CLUSTER3, were highly significant, capturing spatial price differences, while others (e.g., CLUSTER1) were not as impactful. Property types (TYPE): Categories like "condo for sale" and "townhouse for sale" significantly influenced prices, whereas others, like "coming soon," had limited impact.



This refined model validated the importance of square footage, property characteristics, and spatial clustering in understanding and predicting housing prices, while identifying factors with limited significance.

4 Discussion and Conclusion

This project integrates clustering and regression to evaluate how clustering enhances price prediction accuracy. By exploring various models, we uncovered key factors influencing housing prices in New York City and demonstrated the value of combining these techniques. Square footage (`PROPERTYSQFT`), bathrooms (`BATH`), and bedrooms (`BEDS`) emerged as key predictors, with property type and location-based clusters providing additional insights. Clustering effectively revealed geographic price variations beyond traditional administrative boundaries.

Limitations include assumptions of cluster homogeneity, linearity in regression, and data quality challenges such as outliers. Future work could explore non-linear models and incorporate socioeconomic or temporal data to improve predictions.

References

[1] T. Skovajsa. (2023). A Review of Clustering Methods for Housing Market Segmentation. Review of European and Comparative Real Estate, 11(3), 22-35. Retrieved from https://sciendo.com/pdf/10.2478/remav-2023-0022

[2] S. Hwang and J.-C. Thill. (2007). Delineating Urban Housing Submarkets with Fuzzy Clustering. Proceedings of the 15th ACM International Symposium on Geographic Information Systems, 176-183. Retrieved from https://gis.depaul.edu/shwang/research/ACMGIS07_Paper176_HwangThill_20070926.pdf

[3] L. Zhang and P. L. Jin. (2023). Integrating Clustering and Regression for Spatial Segmentation in Housing Markets. arXiv Preprint. Retrieved from https://arxiv.org/html/2405.08398v2

[4] G. S. Sirmans, D. A. Macpherson, and E. N. Zietz. (2005). The Composition of Hedonic Pricing Models. Journal of Real Estate Literature, 13(1), 3-43.

A Code

Listing 1: Python Script for Analysis

```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import folium
from folium.plugins import MarkerCluster
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import plotly.express as px

# Load the dataset
df = pd.read_csv("NY_House_Dataset.csv")

# === Step 1: Remove Outliers in PRICE ===
# Remove properties above the 99th percentile in price
upper_price_limit = df["PRICE"].quantile(0.99)
df = df[df["PRICE"] <= upper_price_limit]

# Visualize PRICE distribution after removing outliers
plt.figure(figsize=(8, 4))
df["PRICE"].plot(kind="box")
plt.title("Price Distribution After Outlier Removal")
plt.savefig('Price_Distribution_After_Outlier_Removal.png')
plt.show()

# === Step 2: Remove Invalid or Zero Square Footage ===
df = df[df["PROPERTYSQFT"] > 0]

# === Step 3: Handle Bedrooms (BEDS) and Bathrooms (BATH) ===
# Round BEDS and BATH to nearest integer
df['BEDS'] = df['BEDS'].round(0).astype(int)
df['BATH'] = df['BATH'].round(0).astype(int)

# Remove unrealistic BEDS and BATH values
df = df[(df['BEDS'] >= 1) & (df['BEDS'] <= 15)]
df = df[(df['BATH'] >= 1) & (df['BATH'] <= 15)]

# === Step 4: Handle Invalid or Missing Street Names ===
# Identify rows with numeric LONG_NAME values and replace them with a placeholder
```

```python
invalid_street = df['LONG_NAME'].str.isnumeric()
df.loc[invalid_street, 'LONG_NAME'] = 'unknown'

# Replace empty or null strings in STREET_NAME with 'unknown'
df['STREET_NAME'] = df['STREET_NAME'].fillna('unknown').str.strip().str.lower()
df['LONG_NAME'] = df['LONG_NAME'].fillna('unknown').str.strip().str.lower()

# === Step 5: Remove Duplicates ===
df = df.drop_duplicates(subset=["ADDRESS", "PRICE", "TYPE"])

# === Step 6: Filter Properties by Geographic Boundaries (NYC) ===
nyc_lat_min, nyc_lat_max = 40.4, 41.0
nyc_lon_min, nyc_lon_max = -74.3, -73.6
df = df[(df["LATITUDE"].between(nyc_lat_min, nyc_lat_max)) &
        (df["LONGITUDE"].between(nyc_lon_min, nyc_lon_max))]

# === Step 7: Standardize Text Columns ===
text_cols = ["BROKERTITLE", "TYPE", "ADDRESS", "STATE", "MAIN_ADDRESS",
             "ADMINISTRATIVE_AREA_LEVEL_2", "LOCALITY", "SUBLOCALITY",
             "STREET_NAME", "LONG_NAME", "FORMATTED_ADDRESS"]
for col in text_cols:
    df[col] = df[col].astype(str).str.strip().str.lower()

# === Step 8: Add Derived Features ===
df["PRICE_PER_SQFT"] = df["PRICE"] / df["PROPERTYSQFT"]
df = df[df["PRICE_PER_SQFT"] > 0]  # Ensure no invalid values
df["PRICE_PER_SQFT"] = df["PRICE_PER_SQFT"].round(2)
df["LOG_PRICE"] = np.log(df["PRICE"]).round(2)
df["LOG_PRICE_PER_SQFT"] = np.log(df["PRICE_PER_SQFT"]).round(2)

# === Step 9: Save the Cleaned Dataset ===
# df.to_csv("Cleaned_NY_House_Dataset.csv", index=False)
# print(f"Cleaned dataset saved with {df.shape[0]} rows and {df.shape[1]} columns.")

```

```
# Print a summary of the cleaned data
# print(f"Cleaned data saved with {df.shape[0]} rows and {df.shape[1]} columns.")
# .info()
# print(df.describe())


# Analyze the data
locality_price_stats = df.groupby('SUBLOCALITY')['PRICE'].agg(['mean', 'median', 'count', 'std']).reset_index()
locality_price_stats = locality_price_stats.sort_values(by='mean', ascending=False)
# print("Top 10 most expensive sub-localities", locality_price_stats.head(5))  # Top 10 most expensive sub-localities


# Prepare data for clustering
clustering_features = df[['LATITUDE', 'LONGITUDE', 'PRICE']].copy()
scaler = StandardScaler()
clustering_features_scaled = scaler.fit_transform(clustering_features)

# Data preparation
clustering_features = df[['LATITUDE', 'LONGITUDE', 'PRICE']]
scaler = StandardScaler()
clustering_features_scaled = scaler.fit_transform(clustering_features)

# Elbow method
wcss = []
for k in range(1, 15):
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(clustering_features_scaled)
    wcss.append(kmeans.inertia_)

# Apply K-Means
kmeans = KMeans(n_clusters=6, random_state=42)
df['CLUSTER'] = kmeans.fit_predict(clustering_features_scaled)

threshold = 50  # Example: Remove square footage values with more than 50 occurrences

# Identify frequent square footage values
```

```
frequent_sqrft_values = df['PROPERTYSQFT'].value_counts()
frequent_sqrft_values = frequent_sqrft_values[
    frequent_sqrft_values > threshold].index

# Remove rows with these square footage values
df = df[~df['PROPERTYSQFT'].isin(frequent_sqrft_values)]
df.to_csv("NY_House_Dataset_with_clusters.csv", index=False
    )

cluster_summary = df.groupby('CLUSTER').agg({
    'PRICE': ['mean', 'median'],
    'PRICE_PER_SQFT': ['mean', 'median'],
    'BEDS': 'mean',
    'BATH': 'mean'
})

# Expand display options
pd.set_option('display.max_rows', 100)  # Set to a large
    value to show more rows
pd.set_option('display.max_columns', 100)  # Set to a large
     value to show more columns
borough_cluster_summary = df.groupby(['CLUSTER', '
    SUBLOCALITY']).size().unstack(fill_value=0)


# distribution of PRICE
sns.histplot(df["PRICE"], kde=True)
plt.title("Distribution of PRICE")
plt.show()
# distribution of PRICE_PER_SQFT
sns.histplot(df["PRICE_PER_SQFT"], kde=True)
plt.title("Distribution of PRICE_PER_SQFT")
plt.show()
# distribution of LOG_PRICE
df["LOG_PRICE"] = np.log1p(df["PRICE"])
sns.histplot(df["LOG_PRICE"], kde=True)
plt.title("Distribution of LOG_PRICE")
plt.show()

# distribution of LOG_PRICE_PER_SQFT
corr = df.corr(numeric_only=True)
sns.heatmap(corr, annot=True, fmt=".2f", cmap="coolwarm")
plt.title("Correlation Matrix")
plt.show()

# Plot the elbow curve
plt.figure(figsize=(8, 6))
```

```
plt.plot(range(1, 15), wcss, marker='o')
plt.title("Elbow Method to Determine Optimal k")
plt.xlabel("Number of Clusters (k)")
plt.ylabel("WCSS")
plt.show()

# Visualize Clusters
plt.figure(figsize=(10, 6))
sns.scatterplot(data=df, x='LONGITUDE', y='LATITUDE', hue='CLUSTER', palette='tab10', alpha=0.7)
plt.title("Clusters of Sub-Localities")
plt.show()


# Filter sub-localities with a significant number of listings to avoid clutter
sublocality_filter = locality_price_stats[locality_price_stats['count'] > 10]['SUBLOCALITY']
filtered_df = df[df['SUBLOCALITY'].isin(sublocality_filter)]

plt.figure(figsize=(12, 10))
sns.boxplot(data=filtered_df, x='SUBLOCALITY', y='PRICE', showfliers=False)
plt.xticks(rotation=90)
plt.title("Distribution of Prices by Sub-Locality")
plt.show()

plt.figure(figsize=(12, 10))
sns.boxplot(data=filtered_df, x='TYPE', y='PRICE', showfliers=False)
plt.xticks(rotation=90)
plt.title("Distribution of Prices by TYPR")
plt.show()

# Create a map


# df['CLUSTER'] = df['CLUSTER'].astype(str)
# Define colors for each cluster
cluster_colors = {
    0: 'blue',
    1: 'orange',
    2: 'green',
    3: 'red',
    4: 'purple',
    5: 'brown'
```

```
}

# Create the map
map = folium.Map(location=[40.7128, -74.0060], zoom_start
    =12)

# Create a MarkerCluster for each cluster
for cluster_label, color in cluster_colors.items():
    # Filter the dataset for this cluster
    cluster_data = df[df['CLUSTER'] == cluster_label]

    # Create a MarkerCluster for this cluster
    cluster = MarkerCluster(icon_create_function=f'''
        function(cluster) {{
            return L.divIcon({{
                html: '<div style="background-color:{color
                    }; color:white; border-radius:50%;
                    padding:5px;">' + cluster.getChildCount
                    () + '</div>',
                className: 'marker-cluster',
                iconSize: [30, 30]
            }});
        }}
    ''').add_to(map)

    # Add CircleMarkers for this cluster
    for i, row in cluster_data.iterrows():
        folium.CircleMarker(
            location=(row['LATITUDE'], row['LONGITUDE']),
            radius=5,
            color=color,
            fill=True,
            fill_opacity=0.7,
            tooltip=f"Price: ${row['PRICE']:,.0f}, Cluster:
                 {cluster_label}"
        ).add_to(cluster)

# Save the map as an interactive HTML file
map.save("NY_Housing_Map_Clusters.html")
# print("Map saved as 'NY_Housing_Map_Clusters.html'")


print('cluster_summary:'
      , cluster_summary)
# print('borough_cluster_summary',borough_cluster_summary)
```

```python
from scipy.stats import f_oneway

# Test if PRICE differs significantly across clusters
anova_result = f_oneway(
    *[df[df['CLUSTER'] == cluster]['PRICE'] for cluster in df['CLUSTER'].unique()]
)
print(f"ANOVA result for PRICE across clusters: {anova_result}")

from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score

# Select features for clustering
X = df[['LATITUDE', 'LONGITUDE', 'PRICE']]

# Drop rows with missing values to ensure consistency
X = X.dropna()

# Fit KMeans on the cleaned dataset
kmeans = KMeans(n_clusters=5, random_state=42).fit(X)

# Ensure the same dataset is used for silhouette score
score = silhouette_score(X, kmeans.labels_)
print(f'Silhouette Score: {score}')
print(f"Shape of X: {X.shape}")
print(f"Length of labels: {len(kmeans.labels_)}")
```