Uncovering Value: Analyzing the Impact of Neighborhood Amenities on Property Prices

Hassan Ashraf

Contents

Introduction

Success in the booming Washington, D.C., metropolitan real estate market requires an understanding of the factors influencing property prices. This white paper delves into the influence of neighborhood amenities, such as parks, schools, and public transportation, on property values. By employing advanced data analytics techniques, we aim to provide actionable insights to MetroLiving Realty Group (MetroLiving) for identifying maximizing price increases in various Neighborhoods.

Previous research suggests that factors such as unit size, proximity to amenities, and neighborhood characteristics significantly impact property values. Understanding these relationships is crucial for stakeholders such as investors, developers, and policymakers.Our approach encompasses rigorous data cleaning, insightful exploratory analysis, and meticulous model comparison to provide actionable insights for maximizing price increases.

Data Cleaning and Feature Engineering

In the initial phase of our analysis, meticulous attention was dedicated to ensuring the integrity and reliability of the dataset. Key steps in this process included the following:

1. Handling Null Values: 238 Null values in the dependent variable, 'recent sale price' , were identified and addressed promptly. Given the critical role of this variable in our analysis, any instances of missing data were dropped to preserve the integrity of the target variable. There were 13 Null values present in the 'avg_size' variable as well, which were dropped as well.
2. Feature Engineering: Feature engineering emerged as a pivotal aspect of our preprocessing pipeline. We Introduced 3 new variables :

```
df['proximity_to_public_transport'] = 1 / df['distance_to_public_transport']
df['proximity_to_schools'] = 1 / df['distance_to_schools']
df['proximity_to_park'] = 1 / df['distance_to_park']
```

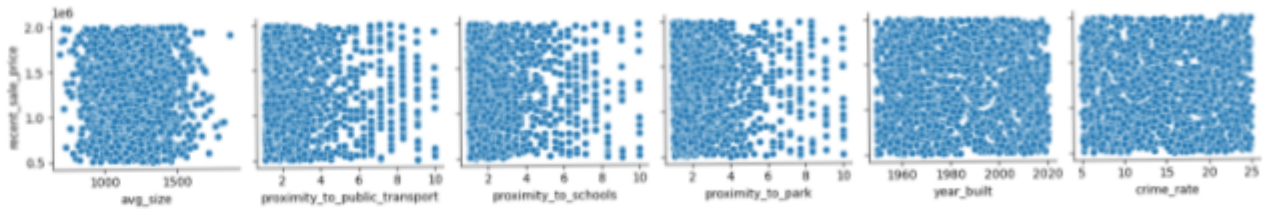
   While introducing new variables, utmost care was exercised to ensure their relevance and meaningfulness in the context of our prediction task.

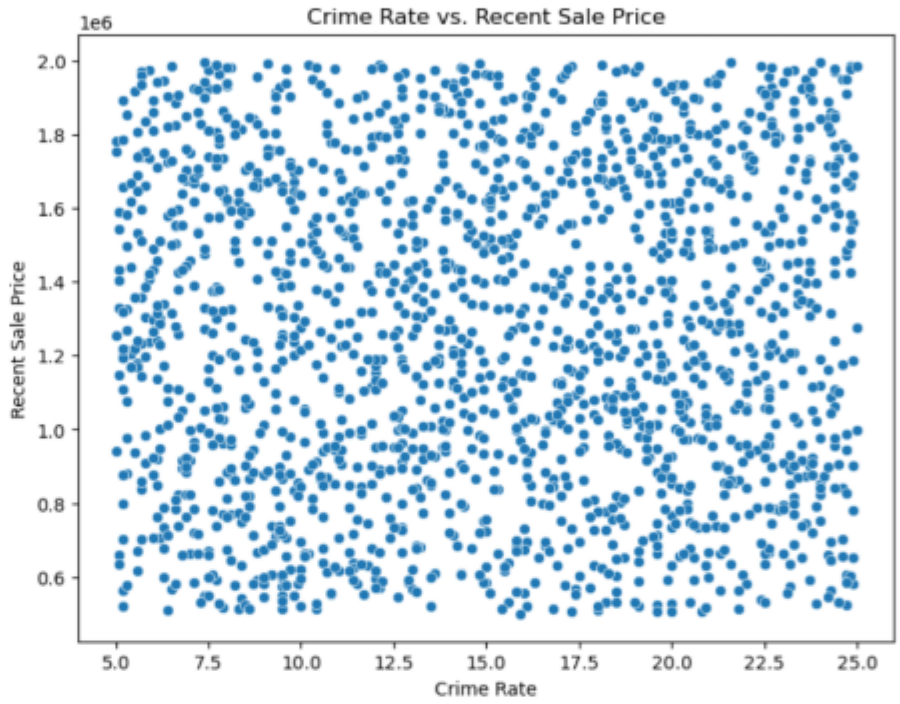
These preliminary steps laid the foundation for subsequent analyses, setting the stage for in-depth exploration and model development aimed at uncovering actionable insights into real estate market trends.

Exploratory Analysis

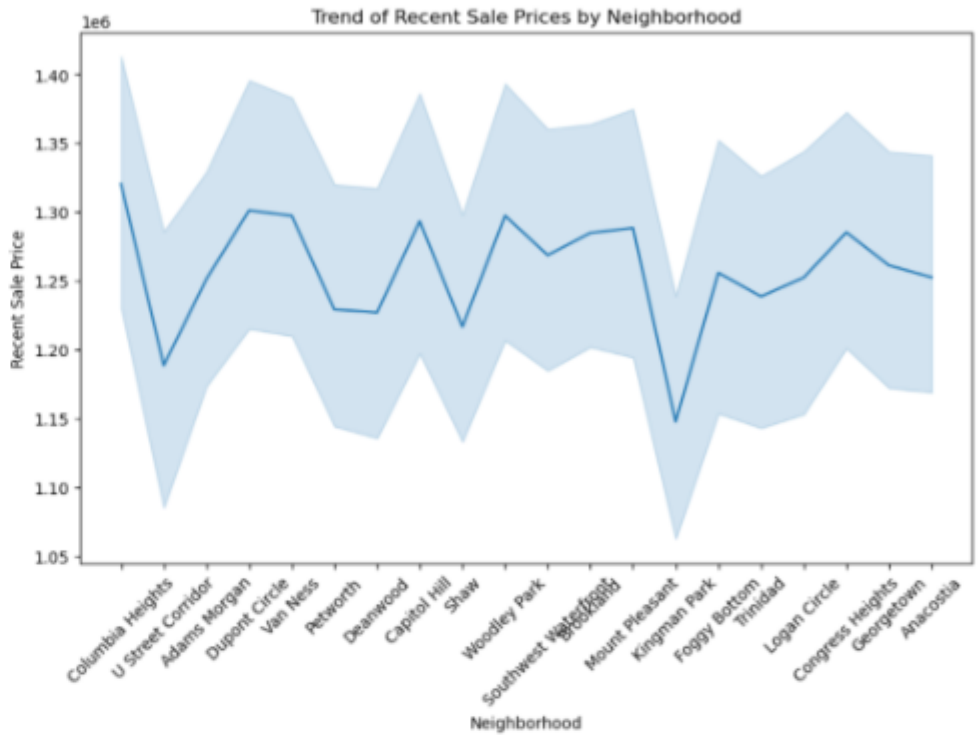
The exploration of our dataset involved an examination of the relationships and patterns inherent within the data. This phase played a pivotal role in informing subsequent modeling decisions and regression technique selection. Key aspects of our exploratory analysis included:

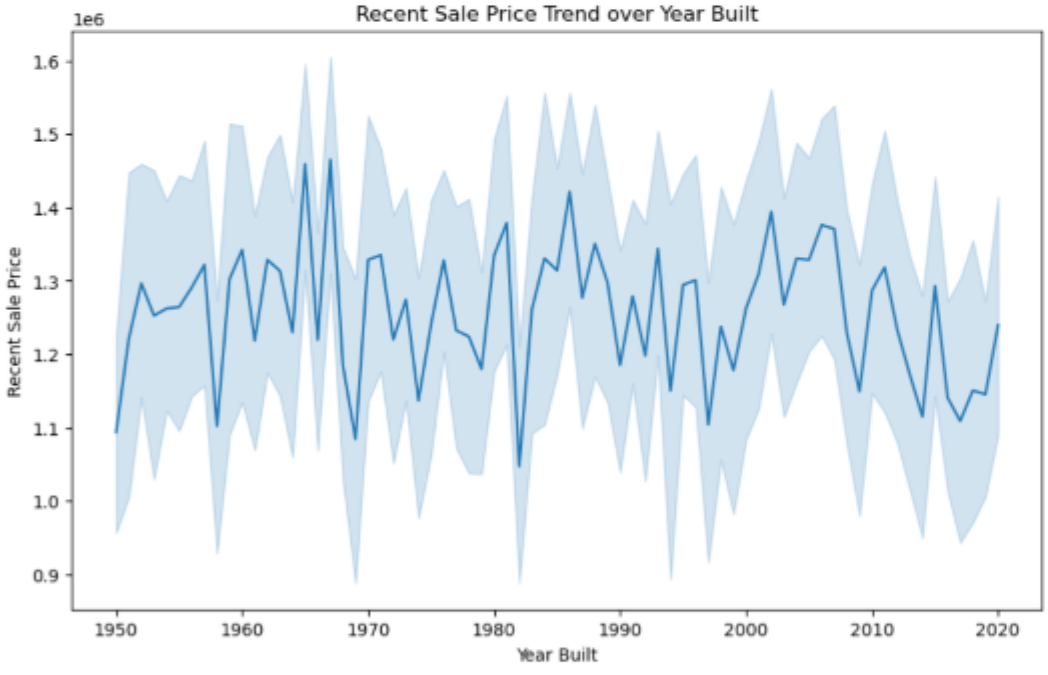
1. **Identification of Non-linear Relationships:** Through comprehensive visualization and analysis, we sought to identify non-linear relationships between variables. This involved scrutinizing scatter plots, correlation matrices, and other graphical representations to discern potential curvature or non-linearity in the data. Such insights were instrumental in guiding our choice of regression techniques, ensuring they were well-suited to capture the inherent complexity of the relationships at play.
2. **Assessment of Correlation**: Another crucial aspect of our exploration involved assessing the degree of correlation between variables. By examining correlation matrices and heatmaps, we gained valuable insights that the variables are not Correlated as all the correlation values are Near zero. This analysis helped identify potential multicollinearity issues and informed feature selection and engineering strategies.

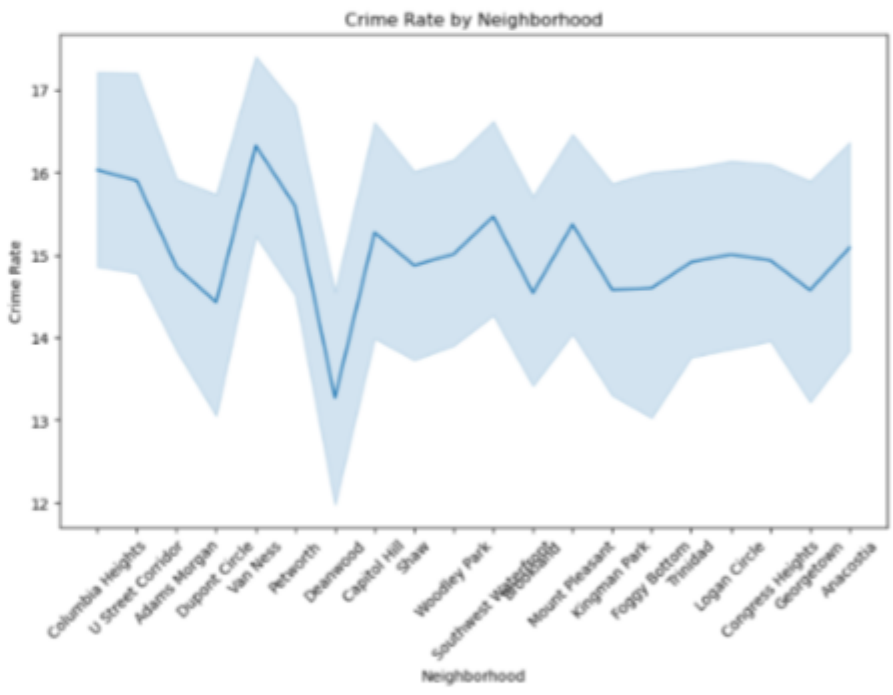


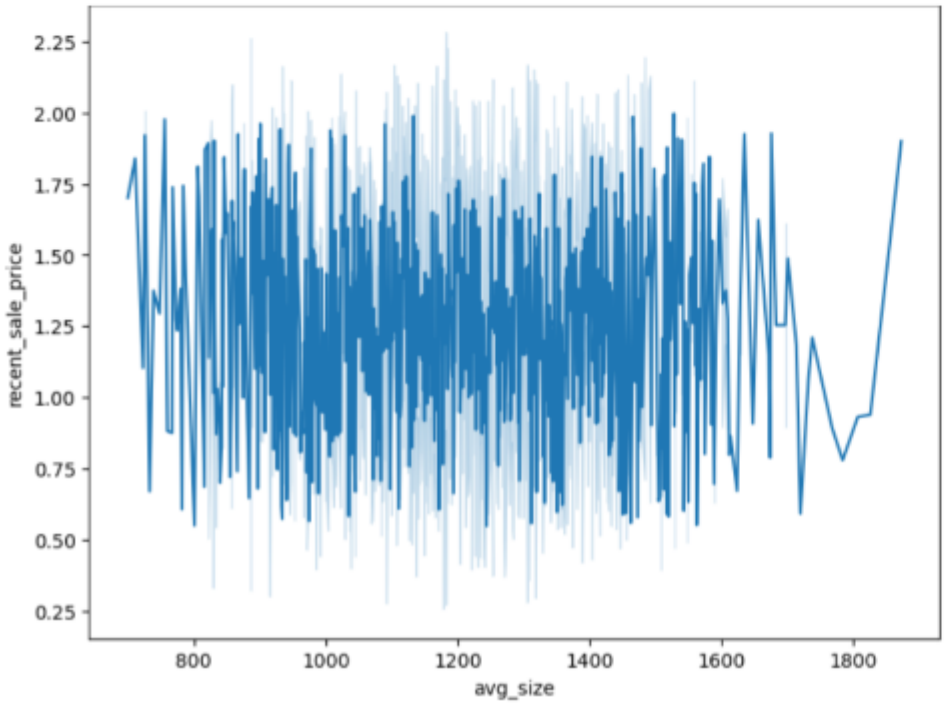


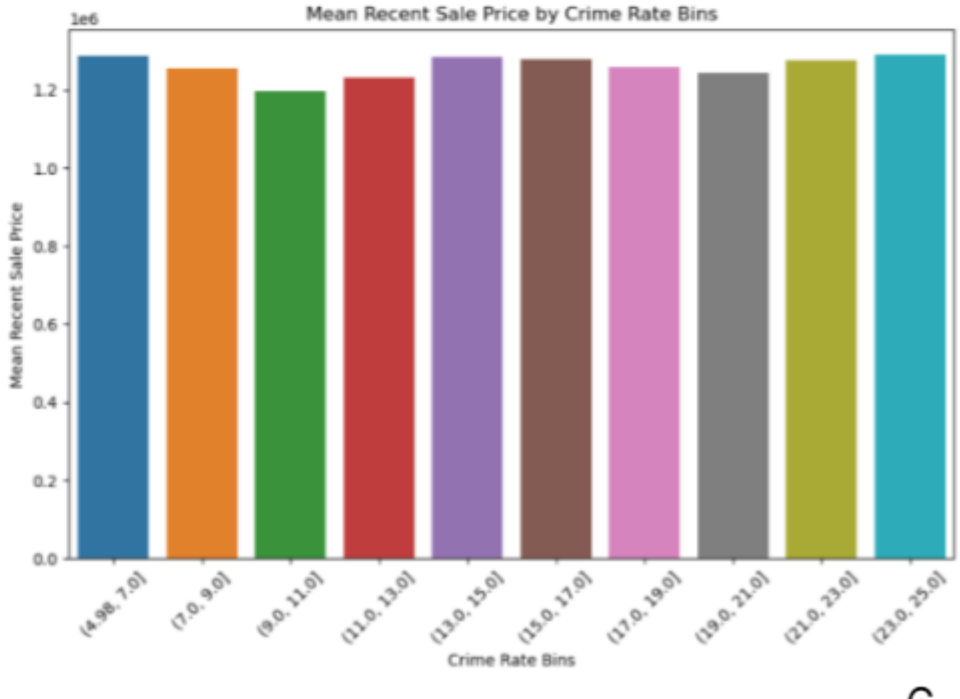
Exploratory Analysis



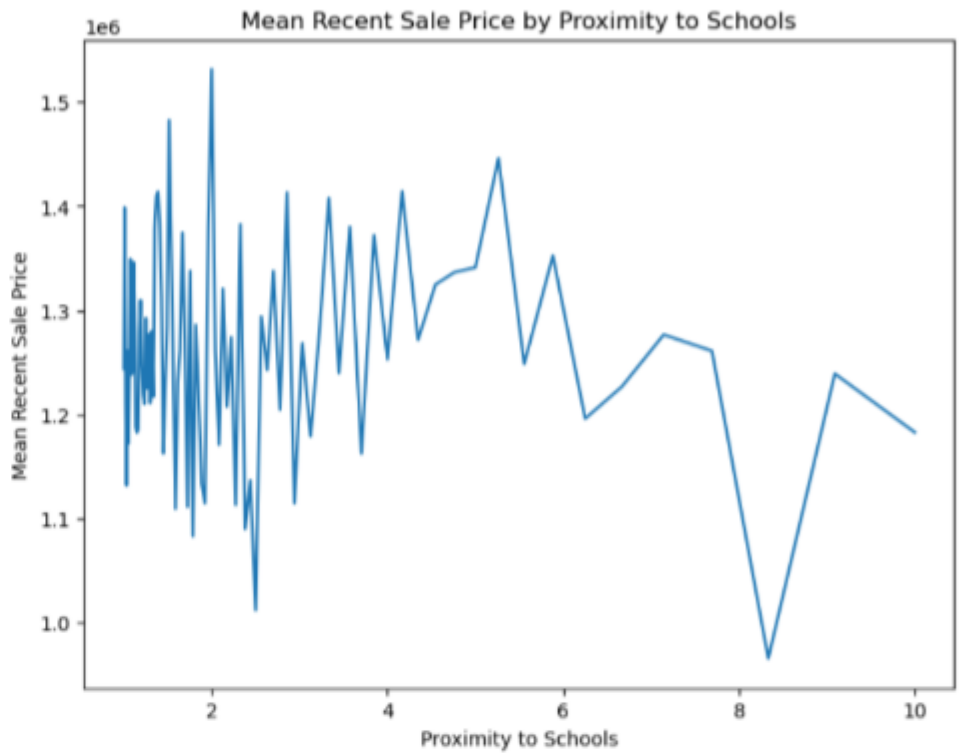


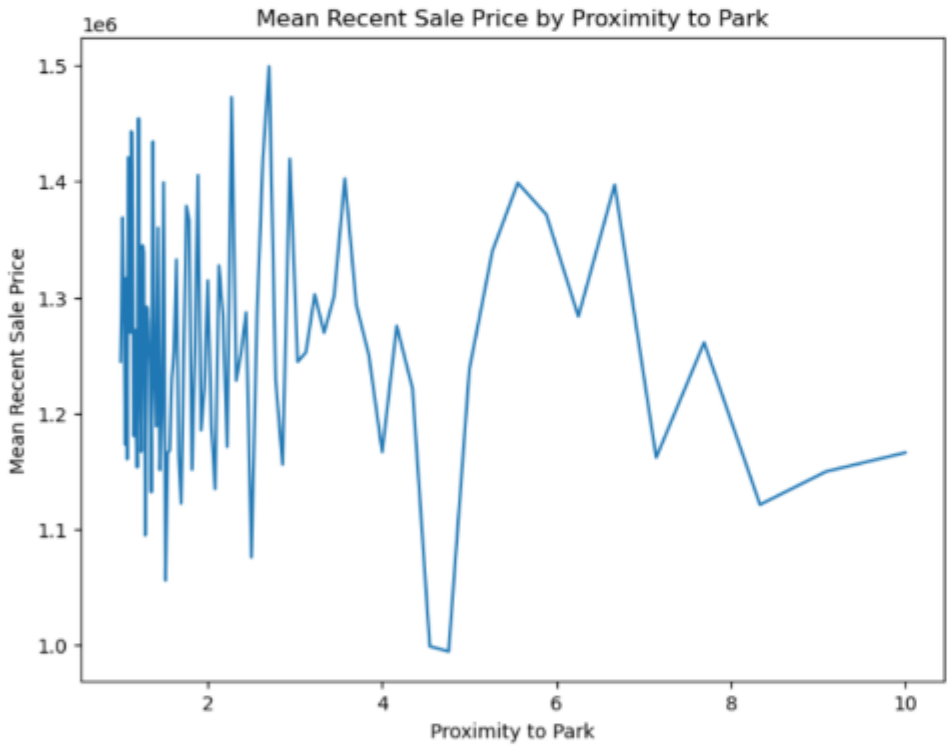


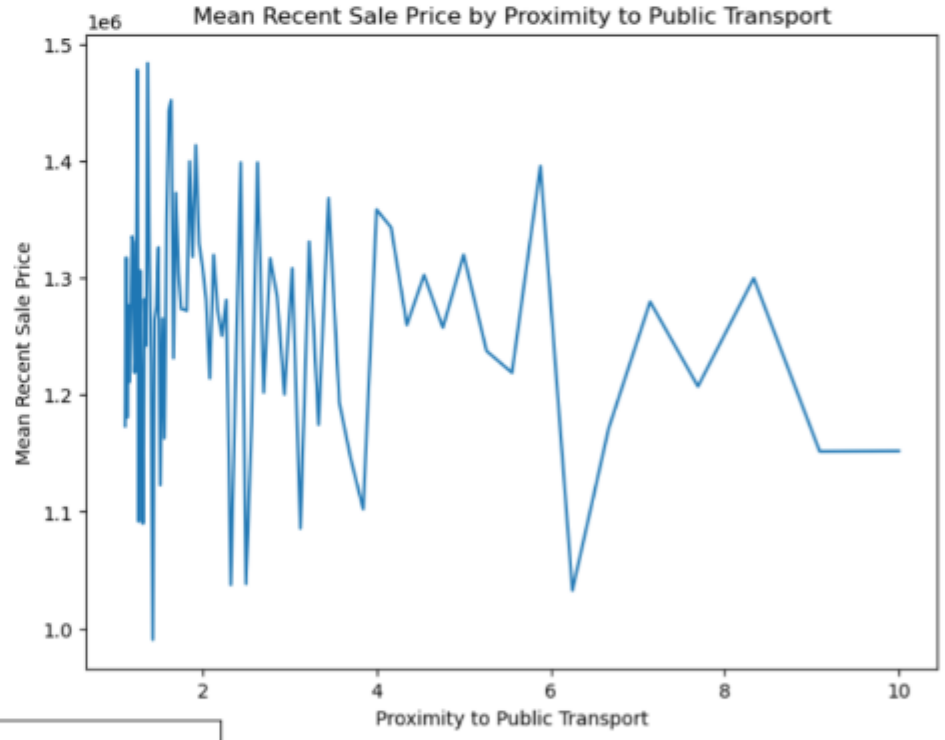


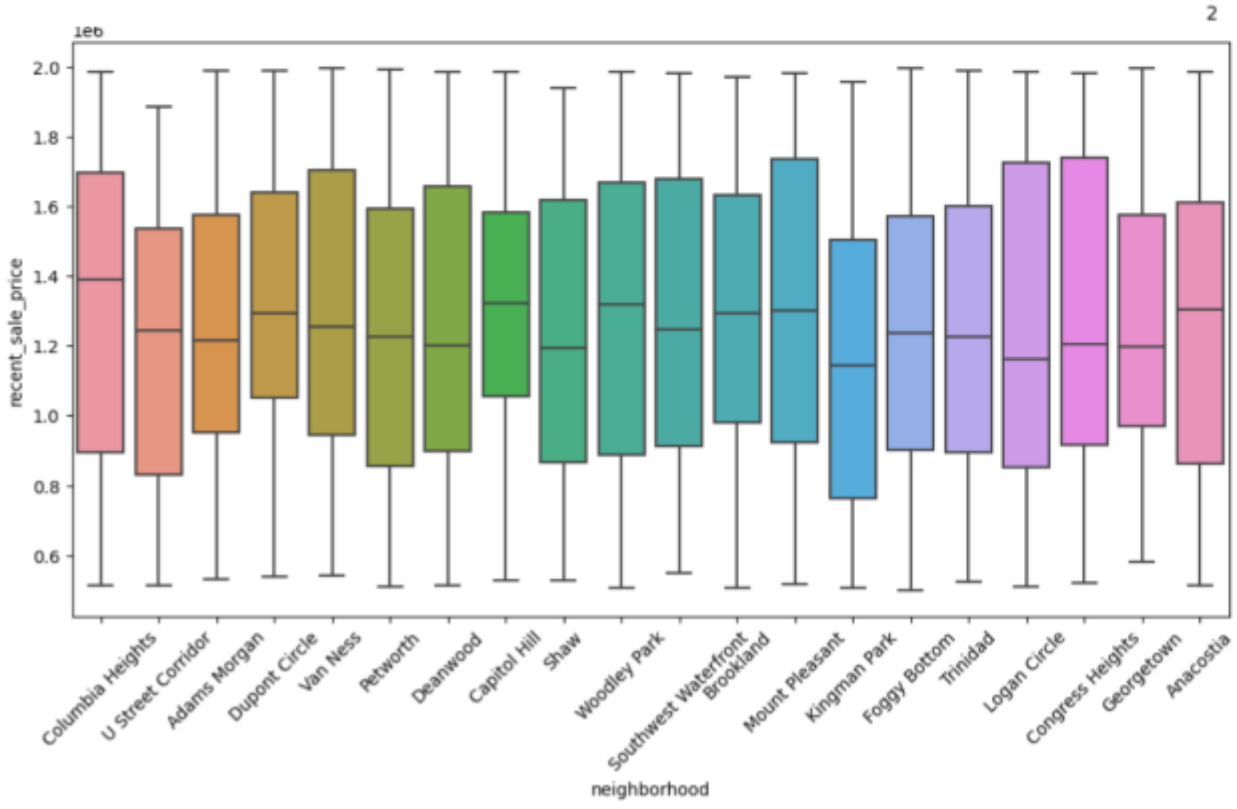


Exploratory Analysis









Exploratory Analysis

During our exploratory data analysis, we uncovered valuable insights that shaped our understanding of the dataset and guided subsequent modeling decisions. Here are the key findings from our exploration:

1. **Non-linear Nature of the Data:**
   - Graphical analysis, including line plots and scatter plots, revealed that the relationship between certain predictors and the target variable (recent sale price) was non-linear. For instance, when plotting average size against recent sale price, we observed random clustered fluctuations instead of a discernible linear pattern. Similarly, the line plot depicting the relationship between year built and recent sale price showed no clear trend or pattern, indicating the absence of a linear relationship.
2. **Lack of Correlation Between Variables:**
   - Our analysis also revealed a lack of significant correlation between variables. This was evident from scatter plots and correlation matrices, which did not exhibit strong linear relationships between predictors and the target variable.
3. **Property Type and Sale Prices:**
   Box plots comparing recent sale price distributions across different property types highlighted interesting insights. While townhouses exhibited slightly lower recent sale prices compared to condominiums and single-family homes, the latter two categories appeared to have similar recent sale price distributions. It is also worthy to know that the neighborhoods where avg size was generally higher resulted in higher recent sale prices as well , indicating that there might be a direct relation between the two as well.
4. **Crime Rate and Neighborhoods:**
   - The analysis of crime rate trends across neighborhoods revealed a potential relationship between crime rates and recent sale prices. Neighborhoods with decreasing crime rates tended to have higher recent sale prices. However, scatter plots depicting recent sale prices against crime rates did not exhibit a clear linear pattern, suggesting that the relationship may be more complex.
5. **Effect of Proximity Factors:**
   - Proximity to amenities such as public transport, schools, and parks emerged as influential factors affecting recent sale prices. Among these, proximity to public transport appeared to have the most significant impact, followed by proximity to schools and parks. Neighborhoods with closer proximity to these amenities tended to command higher recent sale prices. For example neighborhoods like southwest waterfront brookland which was closer to all three resulted in joint top recent sales price along with Mount pleasant.
6. **Neighborhood Disparities:**
- The disparity in recent sale prices between neighborhoods like Kingman Park and Woodley Park, despite Kingman Park boasting better proximities to all three variables, underscores the influence of factors beyond mere amenity proximity. This observation highlights the multifaceted nature of property pricing determinants. While proximity to amenities undoubtedly contributes to property values, other factors such as neighborhood characteristics and market dynamics exert significant influence as well. Therefore, a comprehensive understanding of these diverse factors is essential for accurately assessing property prices and navigating the complex real estate market landscape.

By delving into these insights, we gained a deeper understanding of the dataset's nuances and complexities. These findings served as a foundation for our subsequent modeling endeavors, ensuring that our regression techniques were appropriately tailored to capture the intricacies of the real estate market dynamics in the Washington, D.C., metropolitan area.

Regression Analysis

We embarked on a comprehensive regression analysis to uncover the intricate relationship between property attributes and recent sale prices within an urban area. Our study delved into various regression techniques, each offering unique insights into the complexities of real estate markets.
The following regression techniques were employed:
1) **Multiple Linear Regression (MLR):** Investigated the linear relationship between property attributes (e.g., unit size, proximity to amenities) and sale prices.
2) **Polynomial Regression:** Explored non-linear relationships between variables by introducing polynomial terms into the model.
3) **Ridge and Lasso Regression:** Addressed multicollinearity and overfitting issues by applying regularization techniques.
4) **Neural Network Regression**: Utilized a neural network model to capture complex non-linear relationships.
5) **Random Forest Regression:** Employed an ensemble learning approach to capture interactions among predictors.

Regression Analysis

Multiple linear Regression:

MLR is a standard regression technique suitable for exploring linear relationships between predictors and the target variable. Our Initial Exploratory analysis suggested the presence of linear associations between some property attributes and sale prices.
Steps Taken: -Data cleaning and preprocessing to handle missing values and ensure data integrity.
- Feature engineering to create meaningful variables relevant to the prediction task.
-Model fitting using MLR with property attributes as predictors and recent sale prices as the target variable.
Findings: Despite initial expectations, MLR yielded limited success. The model exhibited a weak correlation between predictors and sale prices, as indicated by an R-squared value of approximately 0.003. Moreover, both the mean absolute error (MAE) of approximately $371,330 and mean squared error (MSE) of approximately $183 billion were notably high, suggesting significant discrepancies between predicted and actual prices.

In [87]:
```
from sklearn.linear_model import LinearRegression

X = df[['units', 'avg_size', 'proximity_to_public_transport', 'proximity_to_schools', 'proximity_to_park' , 'year_built', 'crime_rate']]
y = df['recent_sale_price']


model_multiple_linear_regression = LinearRegression()
model_multiple_linear_regression.fit(X, y)


print('Intercept:', model_multiple_linear_regression.intercept_)
print('Coefficients:', model_multiple_linear_regression.coef_)

from sklearn.metrics import r2_score


predictions = model_multiple_linear_regression.predict(X)


r_squared = r2_score(y, predictions)


print('R-squared:', r_squared)


from sklearn.metrics import mean_absolute_error, mean_squared_error


mae = mean_absolute_error(y, predictions)


mse = mean_squared_error(y, predictions)


print('Mean Absolute Error:', mae)
print('Mean Squared Error:', mse)
```

```
Intercept: 2468963.6708604773
Coefficients: [  326.90551452    -9.79526605 -6244.09158944 -2092.64983014
 -6076.8968629   -597.62820523  1348.42057411]
R-squared: 0.002700395453252291
Mean Absolute Error: 371330.1344120291
Mean Squared Error: 183176186226.81674
```

Regression Analysis

Polynomial Regression:

Observations from MLR hinted at non-linear relationships between predictors and the target variable. - Polynomial regression allows for the exploration of non-linear associations by introducing polynomial terms into the model.
Findings: Despite incorporating polynomial terms up to the second degree, the model's performance did not improve significantly. In fact, the negative R-squared value and high MSE underscored its inability to effectively capture underlying patterns within the data, signaling a poor fit.
Results :
Mean Squared Error (MSE) - Polynomial Regression: 183338365125.71555
R-squared - Polynomial Regression: -0.16968531918608343

In [147…

```
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.pipeline import Pipeline
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np



X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)


degree = 2


polynomial_regression = Pipeline([
    ('poly_features', PolynomialFeatures(degree=degree)),
    ('linear_regression', LinearRegression())
])


polynomial_regression.fit(X_train, y_train)


y_pred_poly = polynomial_regression.predict(X_test)


mse_poly = mean_squared_error(y_test, y_pred_poly)
r2_poly = r2_score(y_test, y_pred_poly)

print("Mean Squared Error (MSE) - Polynomial Regression:", mse_poly)
print("R-squared - Polynomial Regression:", r2_poly)
```

```
Mean Squared Error (MSE) - Polynomial Regression: 183338365125.71555
R-squared - Polynomial Regression: -0.16968531918608343
```

Regression Analysis

Ridge and Lasso Regression:

Multicollinearity and overfitting issues observed in previous models necessitated regularization techniques. Ridge and Lasso regression methods are effective in addressing multicollinearity and reducing model complexity.
Steps Taken: - Implemented Ridge and Lasso regression models using sklearn's Ridge and Lasso classes. Tuned hyperparameters (e.g., alpha) to optimize model performance. Evaluated model fit using training and testing datasets.
Results:
- Ridge Regression: - Training R-squared: 0.0148, Testing R-squared: -0.0191 - Training MSE: $160,531,316,961.72, Testing MSE: $159,733,095,775.37 - Training MAE: $330,501.58, Testing MAE: $332,631.97
Lasso Regression: - Training R-squared: 0.0148, Testing R-squared: -0.0193 - Training MSE: $160,531,042,131.07, Testing MSE: $159,773,053,361.71 - Training MAE: $330,538.85, Testing MAE: $332,704.47
Findings: Both Ridge and Lasso regression models yielded marginal improvements over MLR, but the overall model fit remained poor. Despite regularization, the models failed to adequately capture the complexities in the data, suggesting the presence of unmodeled factors influencing sale prices.

In [98]:
```
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.linear_model import Ridge
from sklearn.model_selection import train_test_split


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)


numeric_features = X.select_dtypes(include=['int64', 'float64']).columns
categorical_features = X.select_dtypes(include=['object']).columns


numeric_transformer = Pipeline(steps=[
    ('scaler', StandardScaler())
])

categorical_transformer = Pipeline(steps=[
    ('onehot', OneHotEncoder(handle_unknown='ignore'))
])


preprocessor = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer, numeric_features),
        ('cat', categorical_transformer, categorical_features)
    ])


ridge_pipeline = Pipeline([
    ('preprocessor', preprocessor),
    ('ridge', Ridge(alpha=1.0))
])


ridge_pipeline.fit(X_train, y_train)


ridge_coefs = ridge_pipeline.named_steps['ridge'].coef_
```

Regression Analysis

Ridge and Lasso Regression:

```
ridge_coefs = ridge_pipeline.named_steps['ridge'].coef_


numeric_feature_names = preprocessor.transformers_[0][2]
categorical_feature_names = ridge_pipeline.named_steps['preprocessor'] \
                                .named_transformers_['cat'] \
                                .named_steps['onehot'] \
                                .get_feature_names_out(input_features=categorical_features)


feature_names = np.concatenate([numeric_feature_names, categorical_feature_names])


coefficients = pd.DataFrame({'Feature': feature_names, 'Coefficient': ridge_coefs})
print(coefficients)
```

```
print("Ridge Regression Coefficients:")
print(ridge_pipeline.named_steps['ridge'].coef_)


print("\nRidge Regression Intercept:")
print(ridge_pipeline.named_steps['ridge'].intercept_)
```

```
Ridge Regression Coefficients:
[  4241.58790097   5087.01765842 -19992.31800829  -8438.28019253
   5562.9664754   -6420.14586566    431.05218157 -30555.96326826
 -14969.56321551  -6007.06120248 -24176.98297856 -18099.12585456
   8338.02389688  33032.63039521  61656.04894667  26871.03510551
  -9938.94977046  35007.56859982  10845.00056538 -16154.16737124
 -88380.6062706   29051.13374132  53640.15989048 -36925.70700003
 -28395.48693403  20990.69159521 -23594.25593098 -90271.85526579
  28253.10864335  28251.73599636   7390.38452337  15741.96415598
 -23132.34867934]

Ridge Regression Intercept:
1254497.3264722212
```

```
                                Feature   Coefficient
0                                 units   4241.587901
1                              avg_size   5087.017658
2          distance_to_public_transport -19992.318008
3                    distance_to_schools  -8438.280193
4                       distance_to_park   5562.966475
5                             year_built  -6420.145866
6                             crime_rate    431.052182
7         proximity_to_public_transport -30555.963268
8                   proximity_to_schools -14969.563216
9                       proximity_to_park  -6007.061202
10            neighborhood_Adams Morgan -24176.982979
11               neighborhood_Anacostia -18099.125855
12               neighborhood_Brookland   8338.023897
13            neighborhood_Capitol Hill  33032.630395
14        neighborhood_Columbia Heights  61656.048947
15        neighborhood_Congress Heights  26871.035106
16                neighborhood_Deanwood  -9938.949770
17           neighborhood_Dupont Circle  35007.568600
18            neighborhood_Foggy Bottom  10845.000565
19              neighborhood_Georgetown -16154.167371
20            neighborhood_Kingman Park -88380.606271
21            neighborhood_Logan Circle  29051.133741
22          neighborhood_Mount Pleasant  53640.159890
23                neighborhood_Petworth -36925.707000
24                   neighborhood_Shaw -28395.486934
25  neighborhood_Southwest Waterfront  20990.691595
26                neighborhood_Trinidad -23594.255931
27      neighborhood_U Street Corridor -90271.855266
28                neighborhood_Van Ness  28253.108643
29            neighborhood_Woodley Park  28251.735996
30            property_type_Condominium   7390.384523
31   property_type_Single Family Home  15741.964156
32             property_type_Townhouse -23132.348679
```

```
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error


y_train_pred = ridge_pipeline.predict(X_train)
y_test_pred = ridge_pipeline.predict(X_test)


train_r2 = r2_score(y_train, y_train_pred)
test_r2 = r2_score(y_test, y_test_pred)


train_mse = mean_squared_error(y_train, y_train_pred)
test_mse = mean_squared_error(y_test, y_test_pred)


train_mae = mean_absolute_error(y_train, y_train_pred)
test_mae = mean_absolute_error(y_test, y_test_pred)


print("Training R-squared:", train_r2)
print("Testing R-squared:", test_r2)
print("\nTraining Mean Squared Error (MSE):", train_mse)
print("Testing Mean Squared Error (MSE):", test_mse)
print("\nTraining Mean Absolute Error (MAE):", train_mae)
print("Testing Mean Absolute Error (MAE):", test_mae)
```

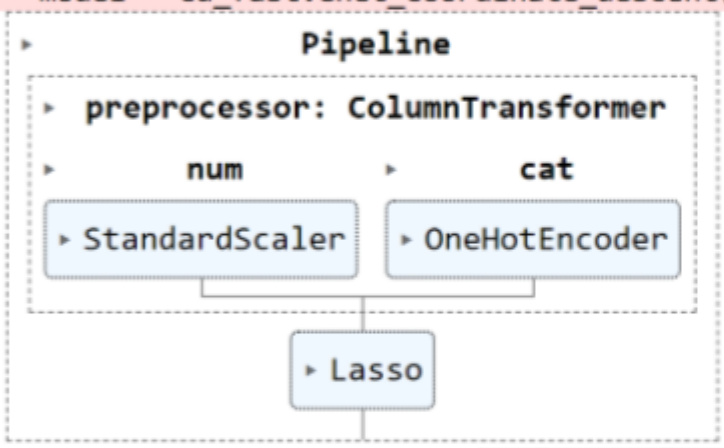
```
Training R-squared: 0.01480160617945514
Testing R-squared: -0.019085432492405463

Training Mean Squared Error (MSE): 160531316961.72372
Testing Mean Squared Error (MSE): 159733095775.36615

Training Mean Absolute Error (MAE): 330501.5824404619
Testing Mean Absolute Error (MAE): 332631.973222693
```

Regression Analysis

Ridge and Lasso Regression:

```
from sklearn.linear_model import Lasso


lasso_pipeline = Pipeline([
    ('preprocessor', preprocessor),
    ('lasso', Lasso(alpha=1.0))
])


lasso_pipeline.fit(X_train, y_train)
```

```
D:\PYTHON - ANACONDA\Lib\site-packages\sklea
the scale of the features or consider increa
  model = cd_fast.enet_coordinate_descent(
```




```
processed_columns = list(lasso_pipeline.named_steps['preprocessor']
                         .named_transformers_['cat']
                         .named_steps['onehot']
                         .get_feature_names_out(categorical_features))


all_feature_names = list(numeric_features) + processed_columns


lasso_coefficients = lasso_pipeline.named_steps['lasso'].coef_


coefficients_df = pd.DataFrame({'Feature': all_feature_names, 'Coefficient': lasso_coefficients})


print(coefficients_df.sort_values(by='Coefficient', key=abs, ascending=False))


train_preds = lasso_pipeline.predict(X_train)


test_preds = lasso_pipeline.predict(X_test)


from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error


train_r2 = r2_score(y_train, train_preds)
print("Training R-squared:", train_r2)


test_r2 = r2_score(y_test, test_preds)
print("Testing R-squared:", test_r2)


train_mse = mean_squared_error(y_train, train_preds)
print("Training Mean Squared Error (MSE):", train_mse)


test_mse = mean_squared_error(y_test, test_preds)
print("Testing Mean Squared Error (MSE):", test_mse)
```

```
print("Testing Mean Absolute Error (MAE):", test_mae)
```

```
                                Feature   Coefficient
27     neighborhood_U Street Corridor -93494.465789
20          neighborhood_Kingman Park -91339.212019
14       neighborhood_Columbia Heights  60537.678310
22         neighborhood_Mount Pleasant  52370.775665
23              neighborhood_Petworth -39239.571033
17          neighborhood_Dupont Circle  33587.745014
13           neighborhood_Capitol Hill  31645.224161
7       proximity_to_public_transport -30692.676341
24                  neighborhood_Shaw -30562.598600
21           neighborhood_Logan Circle  27496.733445
28               neighborhood_Van Ness  26711.399089
29           neighborhood_Woodley Park  26702.848197
10           neighborhood_Adams Morgan -26307.190419
26              neighborhood_Trinidad -25772.325956
15      neighborhood_Congress Heights  25219.638622
31   property_type_Single Family Home  22639.640280
11             neighborhood_Anacostia -20168.851738
2        distance_to_public_transport -20124.552823
25  neighborhood_Southwest Waterfront  19343.145506
19            neighborhood_Georgetown -18280.968585
32            property_type_Townhouse -16313.562083
8               proximity_to_schools -15022.633723
30          property_type_Condominium  14222.921766
16              neighborhood_Deanwood -11941.103919
18           neighborhood_Foggy Bottom   9110.878070
3                 distance_to_schools  -8496.931232
12             neighborhood_Brookland   6542.955689
5                          year_built  -6429.707350
9                   proximity_to_park  -6011.485785
4                    distance_to_park   5582.990966
1                            avg_size   5081.627157
0                               units   4259.733619
6                           crime_rate    417.882821
Training R-squared: 0.014803292845458782
Testing R-squared: -0.019340358961843274
Training Mean Squared Error (MSE): 160531042131.07382
Testing Mean Squared Error (MSE): 159773053361.70786
Training Mean Absolute Error (MAE): 330538.84752715856
Testing Mean Absolute Error (MAE): 332704.4666508229
```

Regression Analysis

Neural Network Regression:

MLR and polynomial regression failed to capture the non-linear relationships observed in the data. Neural networks are capable of learning complex non-linear patterns and interactions.
Results: R-squared (R^2) value: -0.7240 - Mean Absolute Error (MAE): $458,895.70 - Mean Squared Error (MSE): $316,657,661,688.88
Findings: The neural network regression model performed poorly, with a negative R-squared value indicating a significant deviation from the expected outcomes. This suggests that the model failed to capture the underlying patterns in the data, highlighting the limitations of neural networks in this context.

```
from sklearn.neural_network import MLPRegressor


X = df[['units', 'avg_size', 'proximity_to_public_transport', 'proximity_to_schools', 'proximity_to_park' , 'year_built', 'crime_rate']]
y = df['recent_sale_price']


model_neural_network = MLPRegressor()
model_neural_network.fit(X, y)


predictions = model_neural_network.predict(X)


print('Predictions:', predictions[:5])


from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score


mae = mean_absolute_error(y, predictions)
mse = mean_squared_error(y, predictions)
r2 = r2_score(y, predictions)


print('Mean Absolute Error:', mae)
print('Mean Squared Error:', mse)
print('R-squared:', r2)
```

```
Predictions: [989410.00098055 842980.06261391 902335.02376661 797199.59631309
 905213.46671496]
Mean Absolute Error: 458895.696446196
Mean Squared Error: 316657661688.8813
R-squared: -0.7240372085702156
```

Regression Analysis

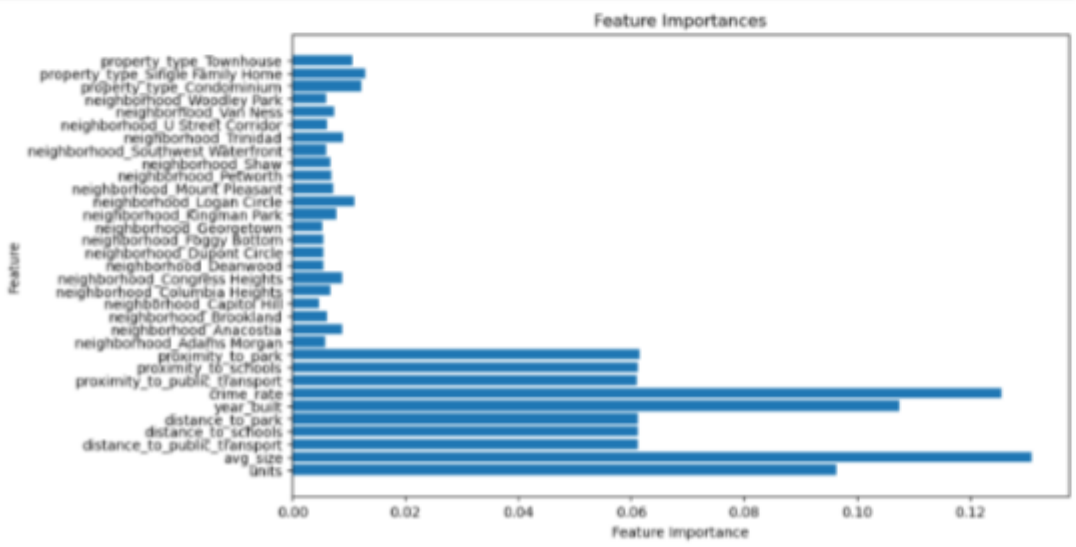
Random Forrest Regression:

Previous models struggled to capture the complex interactions among predictors. Random forest regression is an ensemble learning technique capable of handling non-linear relationships and capturing complex interactions.
Steps Taken: - Implemented a random forest regression model using sklearn's RandomForestRegressor. Specified hyperparameters such as the number of trees and maximum depth. Trained the model and evaluated its performance on testing data.
Results: Mean Squared Error (MSE): $160,201,542,787.95 - R-squared (R^2) value: -0.0221
Findings: Despite its flexibility in capturing complex relationships, the random forest regression model exhibited poor performance. The negative R-squared value and high MSE suggest that the model failed to accurately predict sale prices, indicating the presence of unexplained variability in the data.

In [102…

```
from sklearn.ensemble import RandomForestRegressor


rfr_pipeline = Pipeline([
    ('preprocessor', preprocessor),
    ('random_forest', RandomForestRegressor(n_estimators=100, random_state=42))
])


rfr_pipeline.fit(X_train, y_train)


y_pred_rfr = rfr_pipeline.predict(X_test)


processed_columns = list(rfr_pipeline.named_steps['preprocessor']
                         .named_transformers_['cat']
                         .named_steps['onehot']
                         .get_feature_names_out(categorical_features))


all_feature_names = list(numeric_features) + processed_columns


feature_importances = rfr_pipeline.named_steps['random_forest'].feature_importances_


importance_df = pd.DataFrame({'Feature': all_feature_names, 'Importance': feature_importances})


print(importance_df.sort_values(by='Importance', ascending=False))



mse_rfr = mean_squared_error(y_test, y_pred_rfr)
r2_rfr = r2_score(y_test, y_pred_rfr)

print("Mean Squared Error (MSE) - RFR:", mse_rfr)
print("R-squared - RFR:", r2_rfr)
```

```
plt.figure(figsize=(10, 6))
plt.barh(feature_names, feature_importances)
plt.xlabel('Feature Importance')
plt.ylabel('Feature')
plt.title('Feature Importances')
plt.show()
```



	Feature	Importance
1	avg_size	0.130950
6	crime_rate	0.125512
5	year_built	0.107460
0	units	0.096296
9	proximity_to_park	0.061457
3	distance_to_schools	0.061268
8	proximity_to_schools	0.061208
4	distance_to_park	0.061193
2	distance_to_public_transport	0.061118
7	proximity_to_public_transport	0.060920
31	property_type_Single Family Home	0.012846
30	property_type_Condominium	0.012149
21	neighborhood_Logan Circle	0.010906
32	property_type_Townhouse	0.010688
26	neighborhood_Trinidad	0.008974
15	neighborhood_Congress Heights	0.008879
11	neighborhood_Anacostia	0.008799
20	neighborhood_Kingman Park	0.007731
28	neighborhood_Van Ness	0.007354
22	neighborhood_Mount Pleasant	0.007295
23	neighborhood_Petworth	0.006862
24	neighborhood_Shaw	0.006665
14	neighborhood_Columbia Heights	0.006662
12	neighborhood_Brookland	0.006249
27	neighborhood_U Street Corridor	0.006221
29	neighborhood_Woodley Park	0.006033
25	neighborhood_Southwest Waterfront	0.005926
10	neighborhood_Adams Morgan	0.005871
18	neighborhood_Foggy Bottom	0.005533
17	neighborhood_Dupont Circle	0.005456
16	neighborhood_Deanwood	0.005394
19	neighborhood_Georgetown	0.005344
13	neighborhood_Capitol Hill	0.004780

```
Mean Squared Error (MSE) - RFR: 160201542787.94522
R-squared - RFR: -0.022074090065819618
```

Conclusion

In the real estate domain, accurately forecasting sale prices poses a multifaceted challenge that necessitates thorough analysis and modeling. Our investigation entailed a comprehensive exploration employing various regression methodologies, each tailored to unravel the intricate interplay of factors influencing property values. Despite leveraging advanced techniques such as polynomial regression, neural networks, and random forest regression, the outcomes of each approach fell short of expectations.

Polynomial regression, aimed at capturing subtle nonlinear relationships, failed to yield the expected enhancements. Despite its capability to accommodate intricate patterns via polynomial terms, the model's performance remained inadequate. This suggests that the complexities inherent in real estate pricing surpass mere polynomial functions, requiring a more nuanced strategy.

Similarly, neural networks, renowned for their adeptness at deciphering complex patterns, encountered difficulties in extracting meaningful insights from the dataset. The model's negative R-squared value and elevated mean absolute error underscore its inability to unearth the underlying relationships between predictors and sale prices. Despite their flexibility, neural networks may have struggled to reconcile the myriad variables influencing real estate values.

Even the ensemble learning approach of random forest regression, lauded for its adaptability to intricate data structures, fell below expectations. The model's incapacity to attain a positive R-squared value and the presence of a high mean squared error indicate its failure to precisely capture the nuanced dynamics of the real estate market.

These findings collectively underscore the complexity and multifaceted nature of real estate pricing dynamics. They emphasize the imperative for ongoing research and exploration of alternative modeling approaches capable of capturing the intricate interdependencies among variables influencing property values. Furthermore, they stress the significance of considering external factors, such as economic trends, market sentiment, and regulatory changes, which may exert substantial influence on real estate prices but are often overlooked in conventional regression models.

In essence, while the pursuit of accurately predicting real estate sale prices may encounter obstacles, it also presents opportunities for innovation and discovery. By embracing a multidisciplinary approach and harnessing emerging technologies, we can unearth new insights and refine our comprehension of the complex dynamics shaping the real estate landscape.