Predicting House Rent Prices in India Using Machine Learning

Predicting House Rent Prices in India Using Machine Learning

Introduction/Background

India has a diverse housing market, from historical buildings to modern apartments, showing growth alongside rising incomes [1]. Yet, the Human Rights Measurement Initiative reports that India meets only 60.9% of its housing potential for its income level. Accurate rent prediction can improve market transparency and affordability.

There have been efforts to use machine learning to predict rental prices by analyzing property attributes. For instance, Mallick et al. [2] used regression models to forecast rental expenses in urban areas. Using ensemble methods, Zhang and Li [3] explored the impact of location and amenities on rent prices. Nevertheless, there is a pressing need for models that cater to the specifics of the Indian housing market.

We will use the "House Rent Prediction Dataset" from Kaggle, containing data on over 4,700 residential properties in India. Features include BHK (bedroom-hall-kitchen), rent, size, floor, area type, locality, city, furnishing status, tenant preferences, bathroom count, and contact information. The dataset is available at: House Rent Prediction Dataset.

Problem Definition

The problem is the lack of predefined models to predict house rent prices in India. The goal is to create machine learning models using essential features to predict rent prices. This will help people in real estate and the housing industry determine fair prices and optimize their budgeting. It will also benefit policymakers by allowing them to make effective plans and take action based on price changes. The motivation is to boost India's housing economy, with the potential to contribute to affordable housing solutions globally across different socio-economic segments.

Methods

The objective is to process real estate rental data and prepare it for machine learning modeling. We aim to create a structured data pipeline that includes data preprocessing, dimensionality reduction, model training, and evaluation. This will ensure the data is well-organized, relevant, and suitable for different algorithms.

Data Preprocessing

Raw Features and Their Processing Approaches

1. **BHK**: Retain this feature as-is, as the number of bedrooms is a direct and interpretable measure for analysis.
2. **Rent**: Retain this feature as-is. This is the **target variable** and does not require transformation.
3. **Size**: Retain this feature as-is. The size (square footage) is a critical quantitative feature.
4. **Floor**: While there are more nuances to this, for now we are just extracting the floor value. For example 1 out of 3 becomes 1 (floor 1)
5. **Area Type**: One-hot encode this categorical variable to preserve distinct types of areas without introducing ordinal bias.
6. **Area Locality**: Dropping this feature and depending on city for location data.
7. **City**: One-hot encode, as this categorical variable may contain cities with diverse economic indicators that should be captured distinctly.
8. **Furnishing Status**: One-hot encode this categorical variable to represent different furnishing states (e.g., furnished, semi-furnished, unfurnished) for clarity.
9. **Bathroom**: Retain as-is. The number of bathrooms is a straightforward, quantitative measure relevant to the property's amenities.
10. **Point of Contact**: Dropping this feature as it doesn't contain any meaningful correlation to rent prices (is only owner or agent).
11. **Tenant Preferred**: One-hot encode to account for varying tenant preferences, which may correlate with specific rental patterns.

12. **Posted On**: Dropping this feature as the range of data in the dataset is limited and therefore doesn't suggest any deeper underlying trends

After all the aforementioned transformations, we get the following information about the features of the dataset.

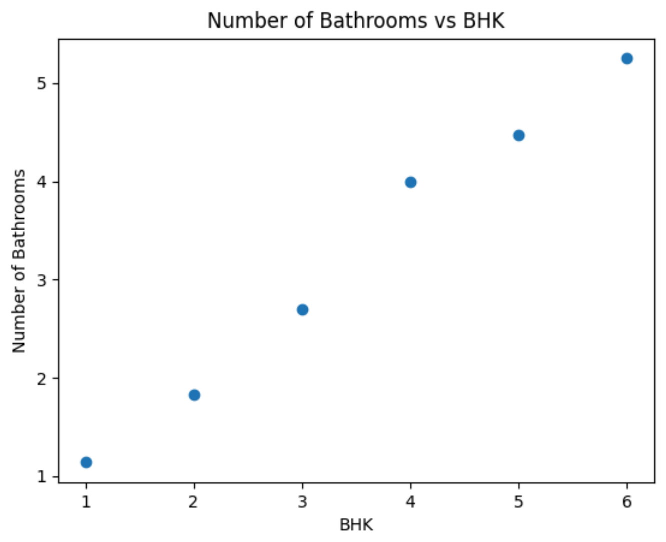
Column	Count	Mean	Std Dev	Min	25%	50%	75%	Max	Missing Values
BHK	4746	2.08386	0.832256	1	2	2	3	6	0
Rent	4746	34993.5	78106.4	1200	10000	16000	33000	3,500,000	0
Size	4746	967.491	634.202	10	550	850	1200	8000	0
Area Type: Built Area	4746	0.000421408	0.020526	0	0	0	0	1	0
Area Type: Carpet Area	4746	0.484197	0.499803	0	0	0	1	1	0
Area Type: Super Area	4746	0.515381	0.499816	0	0	1	1	1	0
City: Bangalore	4746	0.186684	0.389698	0	0	0	0	1	0
City: Chennai	4746	0.187737	0.390543	0	0	0	0	1	0
City: Delhi	4746	0.127476	0.33354	0	0	0	0	1	0
City: Hyderabad	4746	0.182891	0.386618	0	0	0	0	1	0
City: Kolkata	4746	0.110409	0.313432	0	0	0	0	1	0
City: Mumbai	4746	0.204804	0.403601	0	0	0	0	1	0
Furnishing Status: Furnished	4746	0.143279	0.350394	0	0	0	0	1	0
Furnishing Status: Semi-Furnished	4746	0.474294	0.499391	0	0	0	1	1	0
Furnishing Status: Unfurnished	4746	0.382427	0.486031	0	0	0	1	1	0
Tenant Preferred: Bachelors	4746	0.174884	0.379908	0	0	0	0	1	0
Tenant Preferred: Bachelors/Family	4746	0.725664	0.446226	0	0	1	1	1	0
Tenant Preferred: Family	4746	0.0994522	0.2993	0	0	0	0	1	0
Bathroom	4746	1.96587	0.884532	1	1	2	2	10	0
Floor	4746	3.44564	5.76707	0	1	2	3	76	0

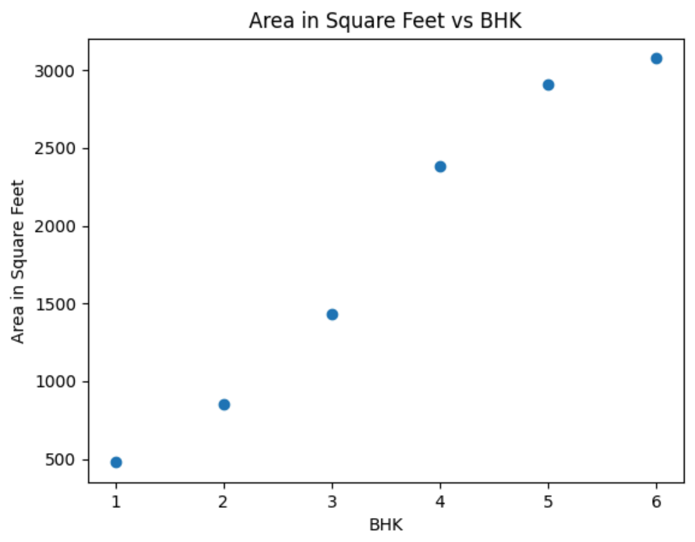
Once transformed, these preprocessing steps produce a structured `N x D` matrix will be prepared, where N is the number of observations, and D is the feature dimension.

Data Split

- Split the processed data into **training** and **testing**, we are using a 80:20 split, to evaluate model generalization accurately.

Observe that there is an issue with the dataset. The columns are linear dependent. It is only logical that the number of bedrooms, bathrooms, and the area all increase together. Plotting this, sure enough, we get graphs which look linearly dependent.





This linear dependence will cause problems. The input matrix is not singular, implying that its inverse does not exist and the determinant is close to 0. Thus, we will require a feature reduction algorithm for the preprocessing step.

Dimensionality Reduction using PCA

To enhance model interpretability, reduce linear dependence, and reduce noise, apply **Principal Component Analysis (PCA)** on the feature set. By transforming features into principal components, PCA identifies linear dependencies and preserves variance:

We selected 12 components for PCA was based on the cumulative variance explained by each component, aiming to capture at least 95% of the total variance. Here's the breakdown:

1. *Cumulative Variance*:
   - By selecting the top 12 principal components, we captured approximately 95.13% of the variance. This threshold was chosen to retain most of the data's variability while reducing the dimensionality, which often helps in simplifying the model and improving computational efficiency.
2. *Variance Explained by Each Component*:
   - Below are the percentages of variance explained by the first 12 components:
     - Component 1: 17.94%
     - Component 2: 13.15%
     - Component 3: 9.22%
     - Component 4: 8.69%
     - Component 5: 7.03%
     - Component 6: 6.42%
     - Component 7: 6.24%
     - Component 8: 5.92%
     - Component 9: 5.60%
     - Component 10: 5.32%
     - Component 11: 4.97%
     - Component 12: 4.62%
   - Together, these components account for around 95.13% of the data's variance. Components beyond the 12th contribute less than 3% each, meaning they add little new information and are less significant for the model.
3. *Dimensionality Reduction*:

  - Reducing from the original 19 features to 12 principal components strikes a balance between simplicity and information retention. This decision improves computational efficiency and minimizes overfitting risks by removing less informative dimensions while preserving most of the variance.

In summary, this step reduces dimensionality without sacrificing much data, optimizing processing time and potentially improving model stability.

Algorithms/Models

Supervised Learning

We implemented three distinct supervised learning models to analyze and predict rental prices. These models were chosen to provide a comparative perspective on performance, ranging from simple linear approaches to more sophisticated ensemble methods.

1. **Linear Regression** We are implementing **Linear Regression** as our first model to capture linear relationships between features and rent price via **supervised learning** [4]. It is expected that this will be a baseline for comparison with other models that are more complicated in nature. We are using scikit-learn to implement the model [5].

    - The goal was to determine how many components provide an optimal balance between model complexity and accuracy. We applied linear regression to data reduced to various numbers of principal components.
    - For the selected 12 components, which capture approximately 95.13% of the variance, the model achieved an R^2 of 0.452 and an MSE of 2,183,889,608.25 on the test set.
2. **XGBoost** Following the linear regression model, we implemented **XGBoost** to improve it, a gradient boosting algorithm known for its efficiency and scalability. XGBoost is a more complex model that creates an ensemble of decision trees and uses boosting to iteratively correct errors made by previous trees, allowing it to model more complex, non-linear relationships in the data.

    - For XGBoost, the optimal number of components was determined to be 12 using PCA, which offered the best balance between model complexity and accuracy. The results for this model were: R^2 of 0.9500, MSE of 304,867,164.7939 and MAE of 8,664.8480 on the test set. The reasoning for this choice of model is its scalability and robustness. XGBoost effectively models complex, non-linear relationships, leading to significantly improved accuracy compared to Linear Regression.
3. **Random Forest** For our third model, we implemented Random Forest, an ensemble learning method that builds multiple decision trees and merges their predictions to improve accuracy and reduce overfitting. Random Forest is particularly effective in handling non-linear data and complex interactions between features.

    - The same PCA-reduced dataset with 12 components was used for consistency here. This produced the results: R^2 of 0.860, MSR of 852,456,500, and MAE of 5947.76.
    - Random Forest is known for its ability to capture non-linear relationships and handle feature interactions while mitigating overfitting due to its ensemble approach. This makes it a complementary model to compare against XGBoost and Linear Regression.

Unsupervised Learning

We implemented **KMeans clustering** as an unsupervised learning method to explore the inherent structure of the housing rental data. We are very curious to see how unsupervised learning approaches perform in numerical modeling workloads. KMeans is particularly useful for identifying distinct groups within a dataset based on feature similarities, which can provide valuable insights into the characteristics of different rental segments. Given parameter `k`, our model performs KMeans unsupervised learning by clustering the datapoints based on the input. Each cluster is then represented by the mean rent of its datapoints.

There was one additional preprocessing step. We had to perform data normalization for the KMeans to ensure that columns which are unbounded do not weigh more than columns which are one-hot encoded and upper-bounded to 1.

In the context of our housing rental dataset, we utilized KMeans to identify clusters of properties that share similar characteristics based on features such as BHK, size, rent, and furnishing status. This clustering can help in understanding how different property attributes correlate with rental prices and can reveal patterns that may not be immediately apparent through supervised learning methods.

We experimented with various values of `k` to determine the optimal number of clusters. The evaluation of the clustering performance was conducted using metrics such as the **Silhouette Score**, which measures how similar an object is to its own cluster compared to other clusters, and the **Within Cluster Sum of Squares (WCSS)** which helps decide the optimal hyperparameter. A higher Silhouette Score indicates better-defined clusters. The optimal number of clusters occurs when the WCSS value levels off.

Results and Discussion

Results and Discussion: Discuss the results of your methods. Present visualizations and quantitative scoring metrics to analyze the performance of your models. Compare each approach and explain the tradeoffs, strengths, and limitations of each approach. What does your visualization/metric tell you? Why did each model perform well/poorly? How do the models compare to each other? What are the next steps if any?

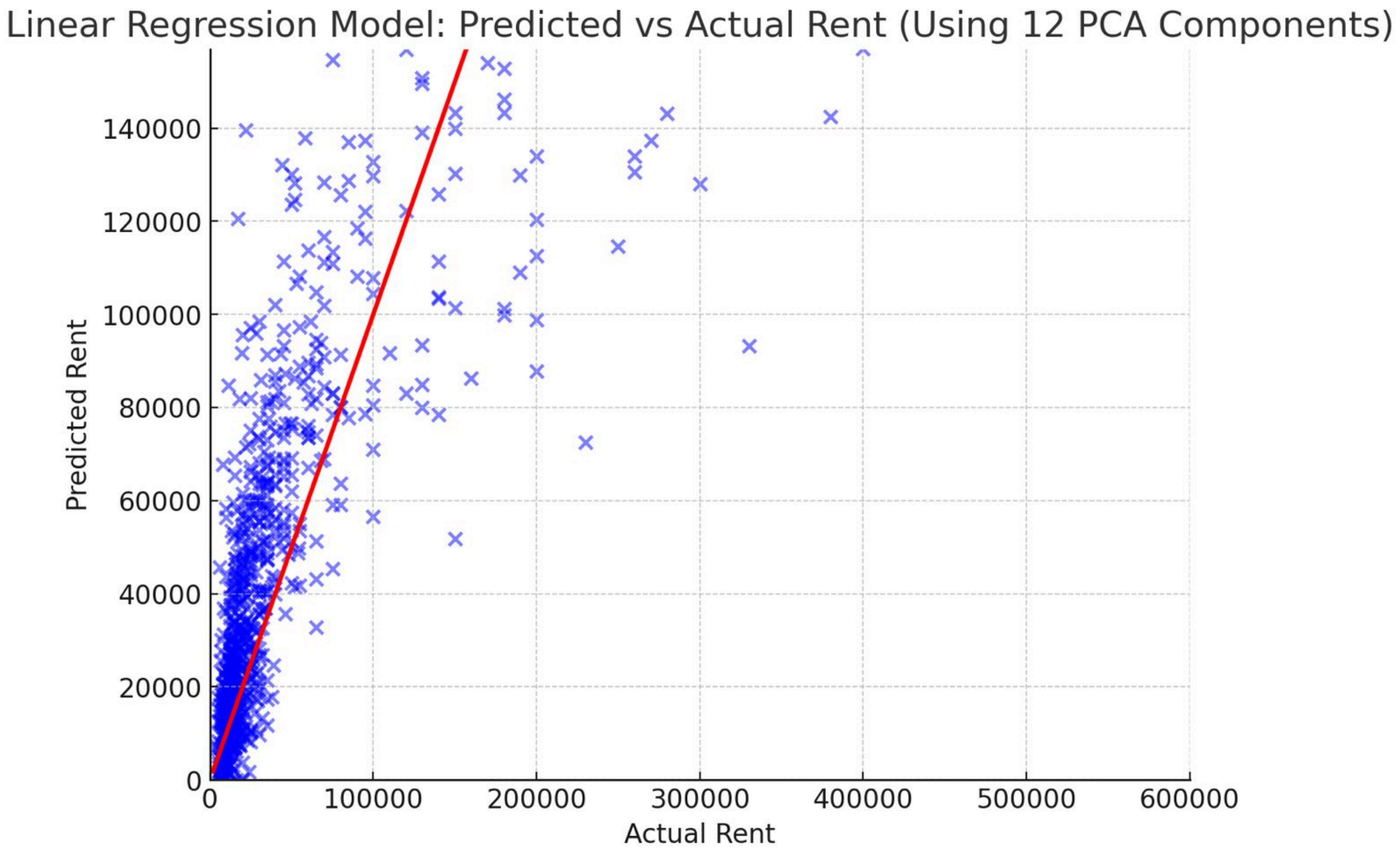
Supervised Learning

We implemented three distinct supervised learning models to analyze and predict rental prices. These models were chosen to provide a comparative perspective on performance, ranging from simple linear approaches to more sophisticated ensemble methods.

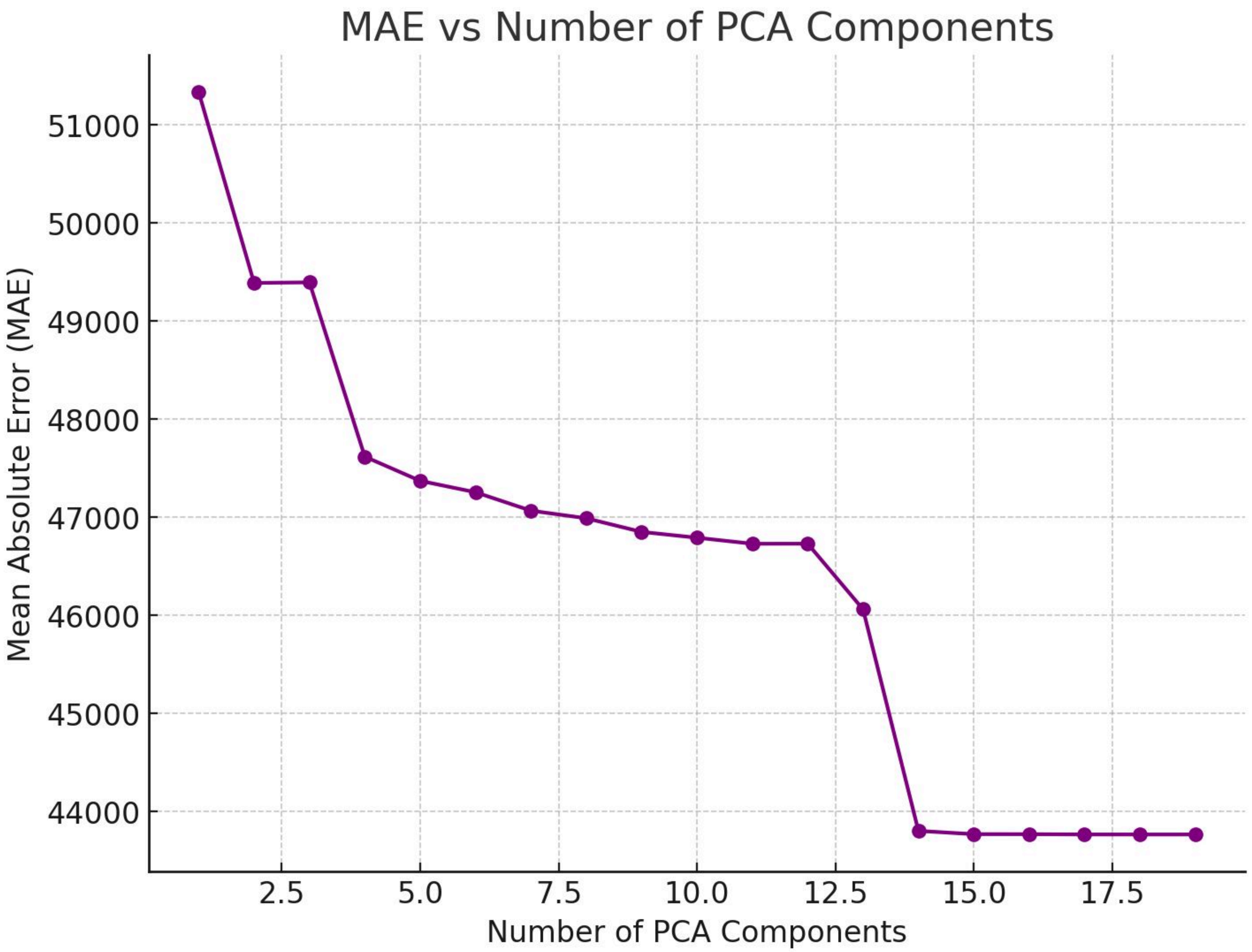
Linear Regression

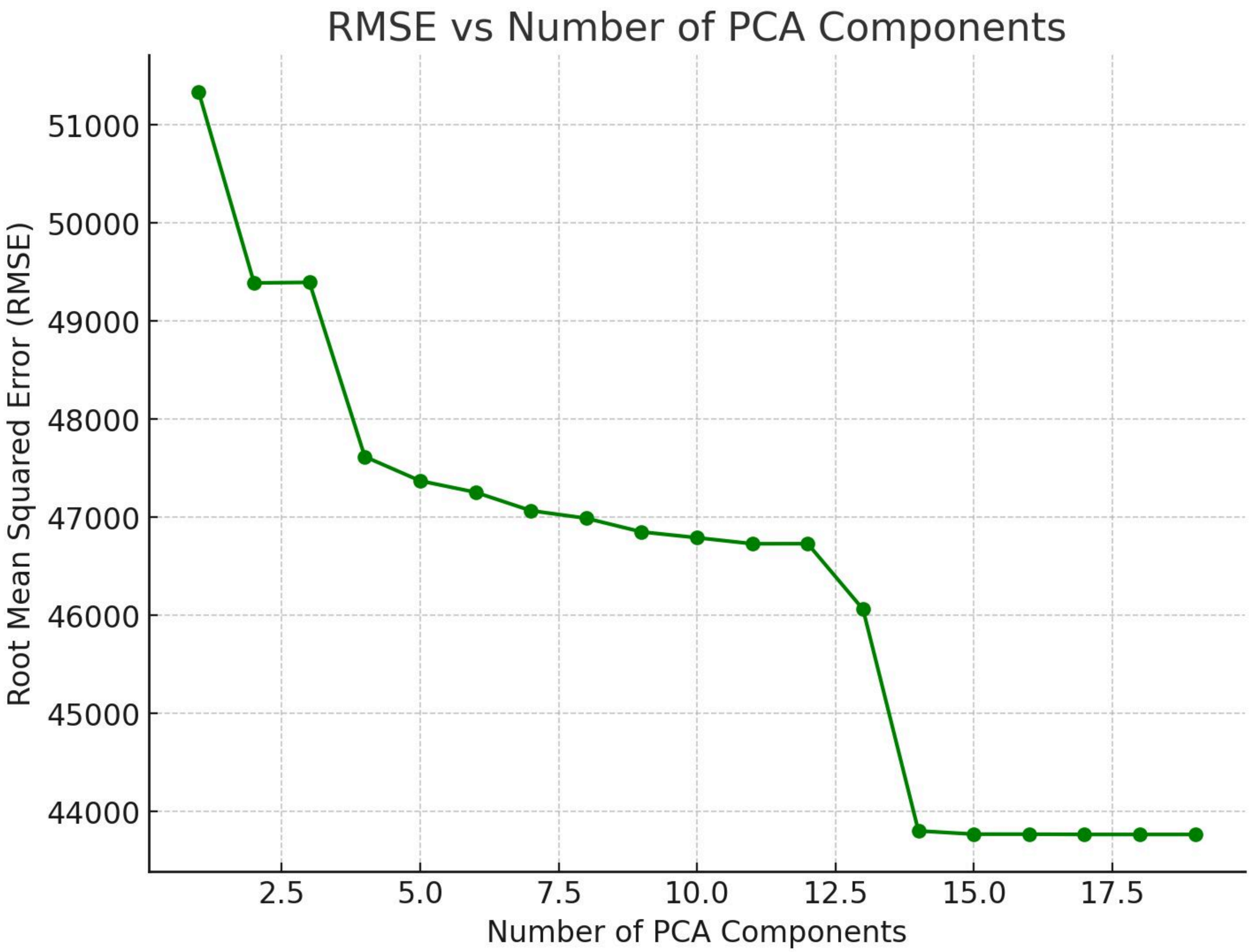
We implemented **Linear Regression** as our baseline model to capture linear relationships between features and rent prices. Using 12 PCA components that retained approximately 95.13% of the variance, the model achieved the following results:

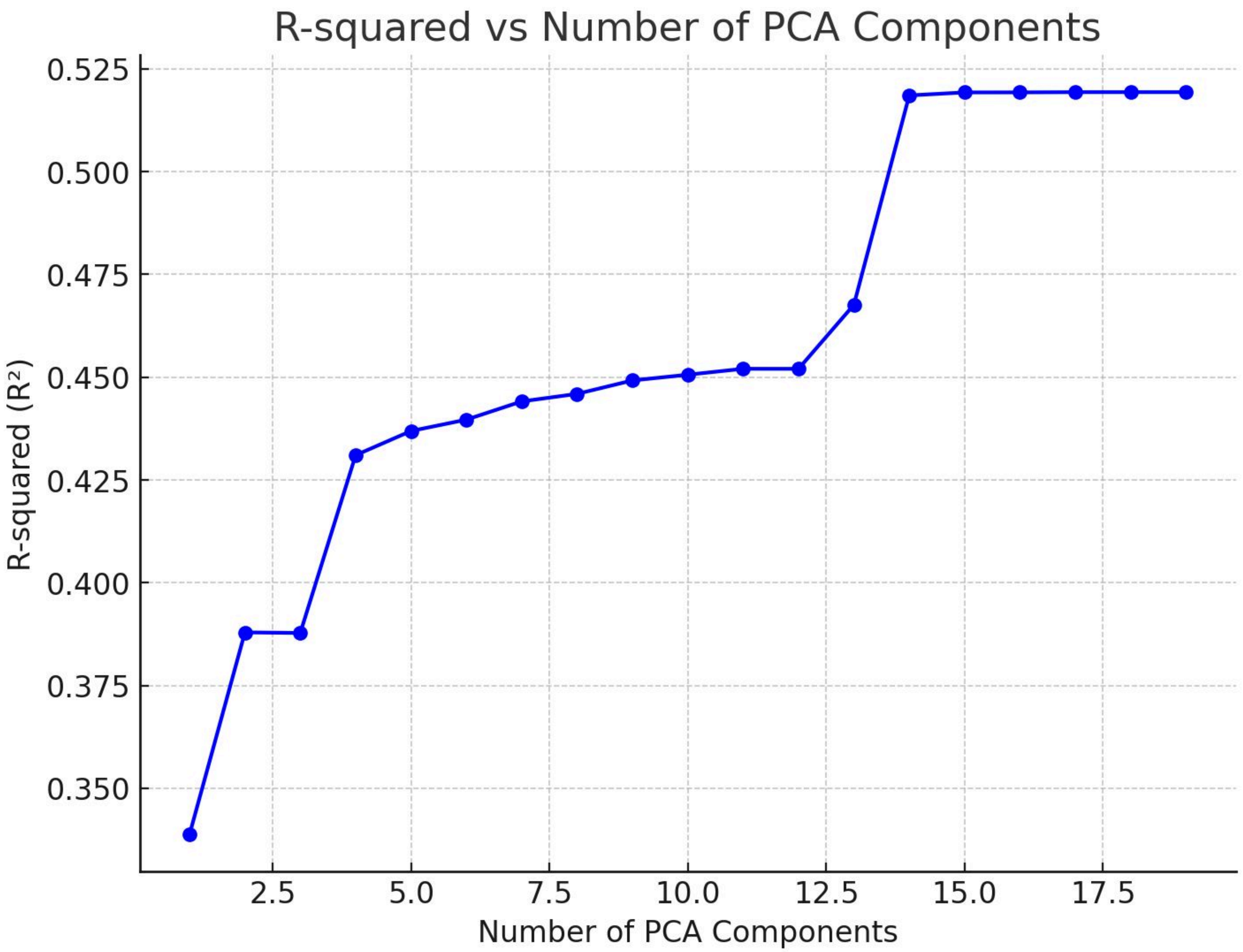
- **R-squared (R^2)**: 0.452
- **Mean Squared Error (MSE)**: 2,183,889,608.25
- **Mean Absolute Error (MAE)**: 21,937.11



As seen in the predicted vs. actual values chart, there is a clear positive correlation, but outliers highlight the model's inability to capture complex, non-linear relationships. This limitation is likely due to the lack of locality-based features, which could have provided additional granularity.


MAE vs Number of PCA Components
Mean Absolute Error (MAE)
51000
50000
49000
48000
47000
46000
45000
44000
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Number of PCA Components


RMSE vs Number of PCA Components
Root Mean Squared Error (RMSE)
51000
50000
49000
48000
47000
46000
45000
44000
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Number of PCA Components

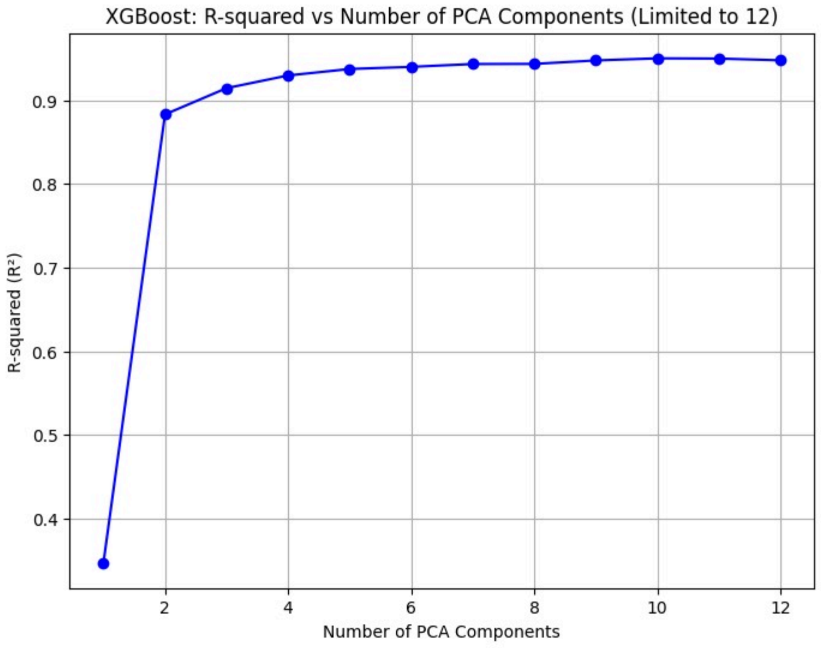


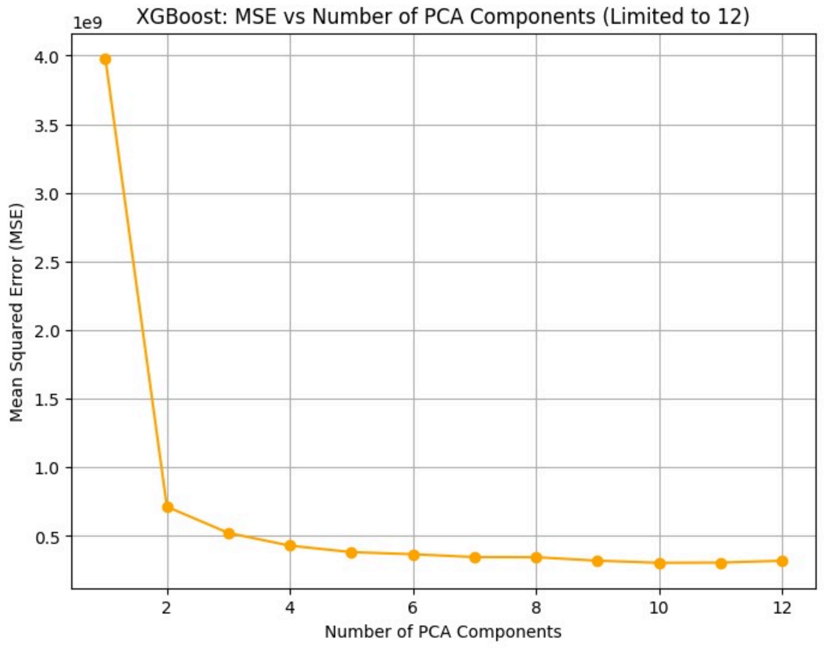
The performance metrics (MAE, RMSE, and R^2) across PCA components indicate that while 12 features preserved most of the variance, using 14 components might yield better results. Future iterations will refine feature selection and assess these discrepancies.

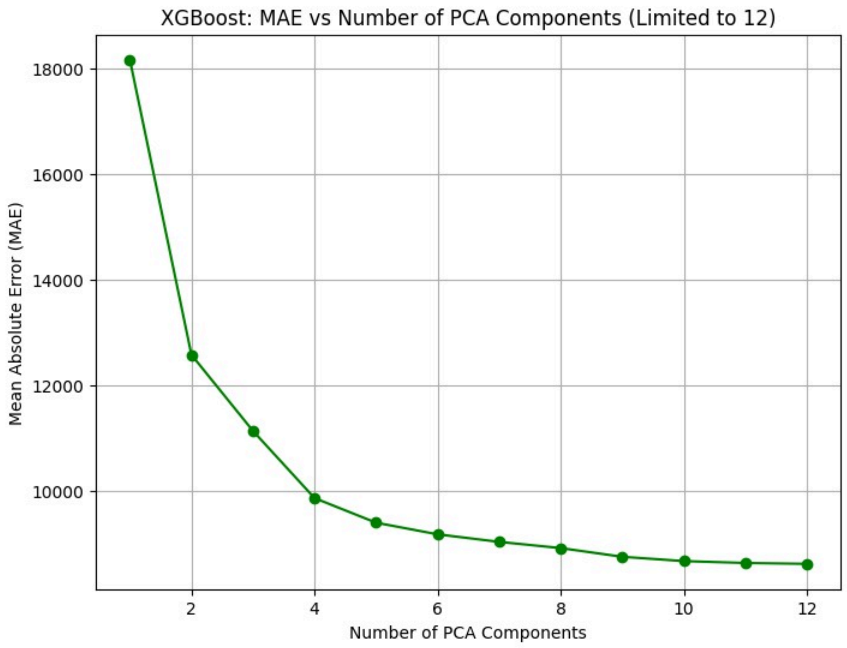
XGBoost

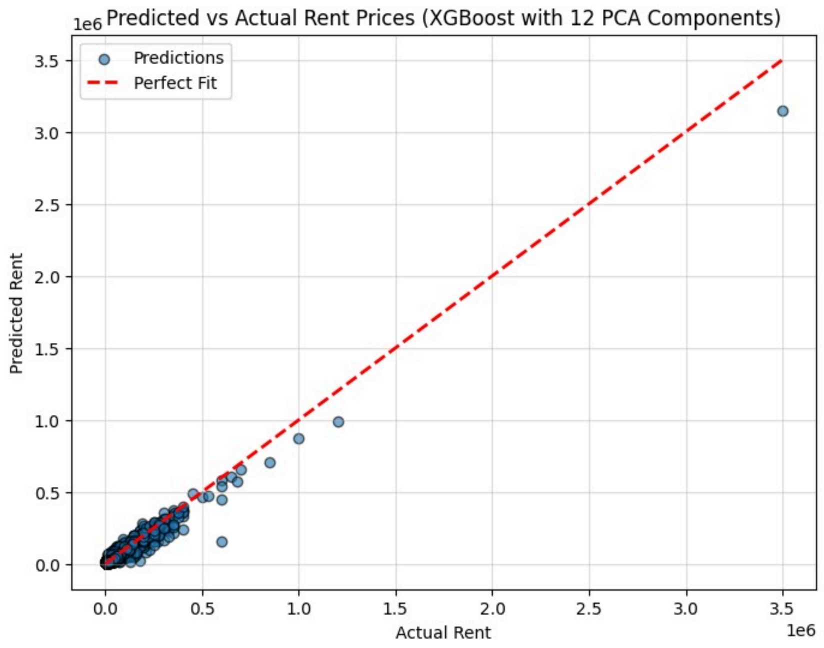
XGBoost, a gradient boosting algorithm, significantly outperformed Linear Regression by effectively capturing non-linear relationships and complex feature interactions. Using the same PCA-reduced dataset (12 components), the model achieved the following results:

- **R-squared (R^2)**: 0.9500
- **Mean Squared Error (MSE)**: 304,867,164.79
- **Mean Absolute Error (MAE)**: 8,664.85







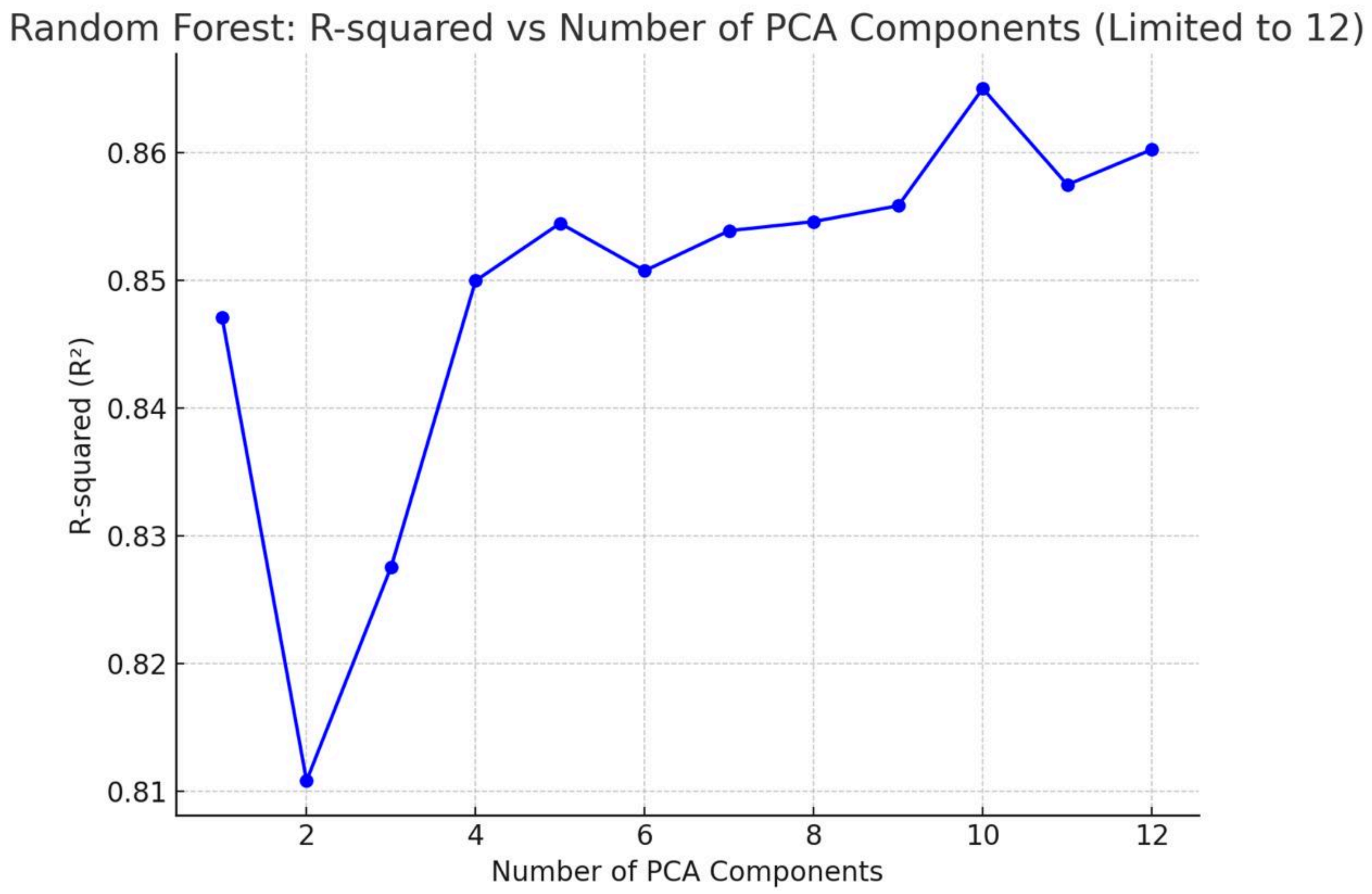


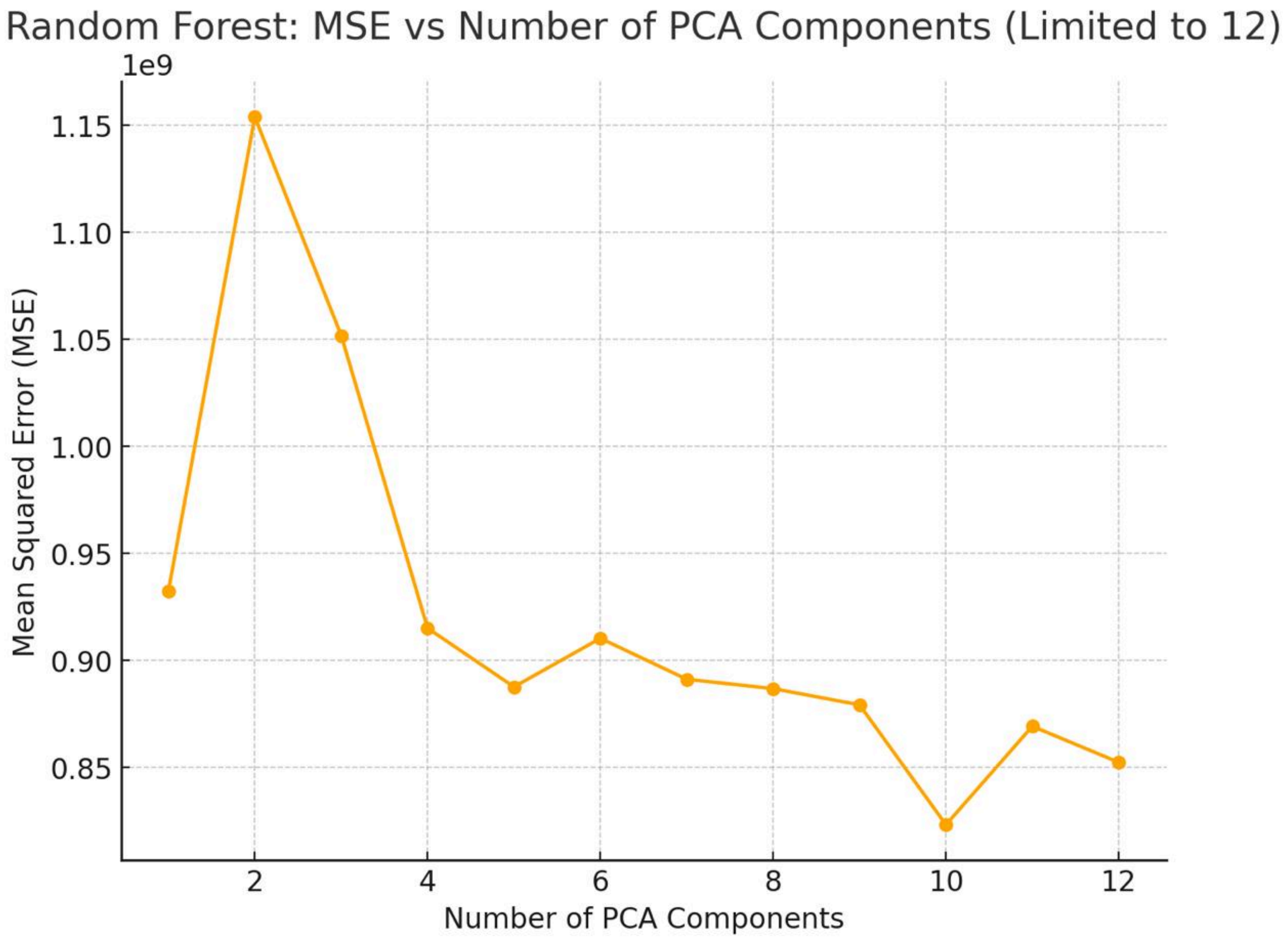
The visualizations above highlight XGBoost's superior performance across all metrics. Its ensemble learning approach iteratively reduces residual errors, resulting in a much tighter alignment between predicted and actual values compared to Linear Regression. The MAE and MSE values demonstrate its robustness in minimizing both typical and large errors.

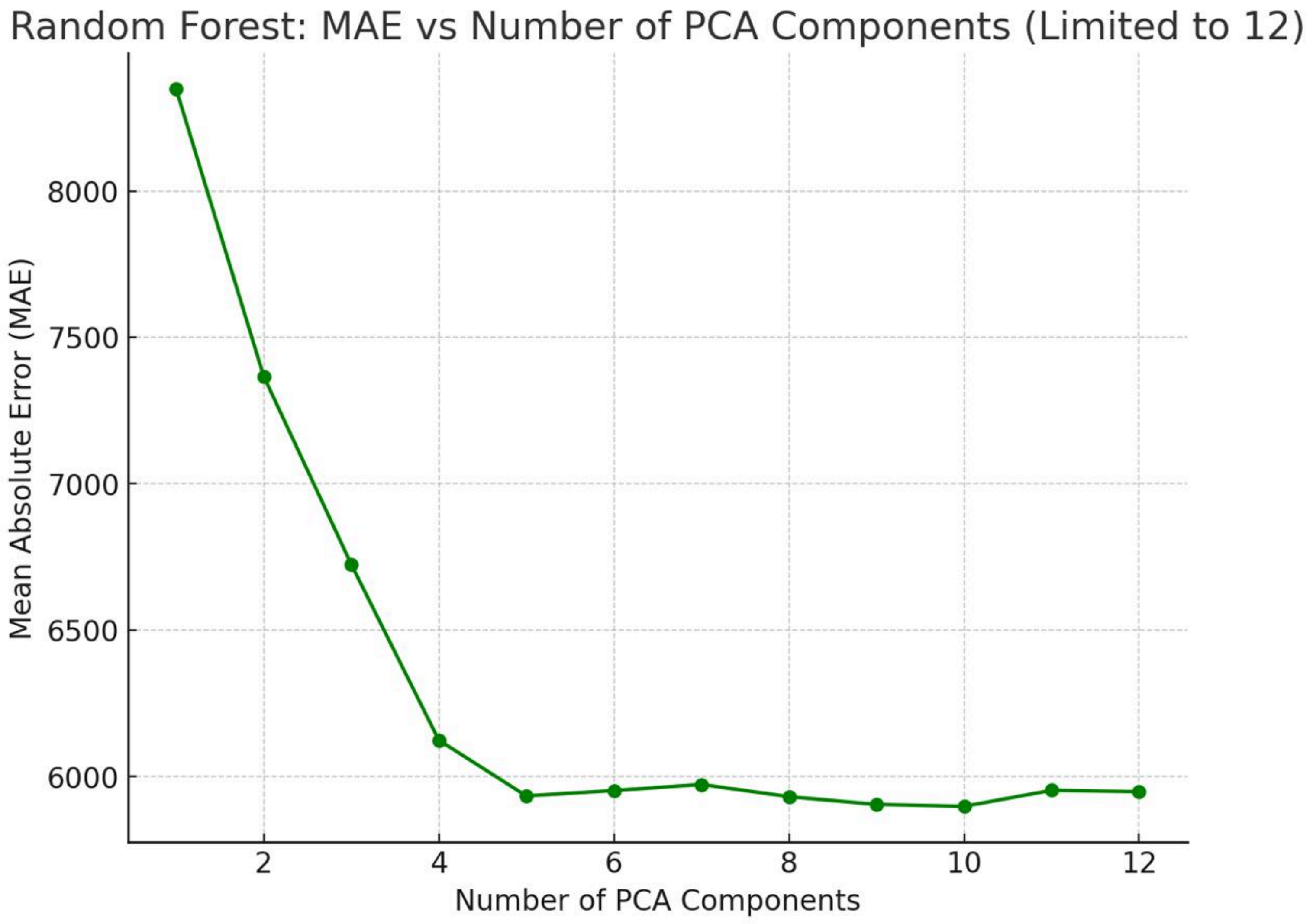
Random Forest

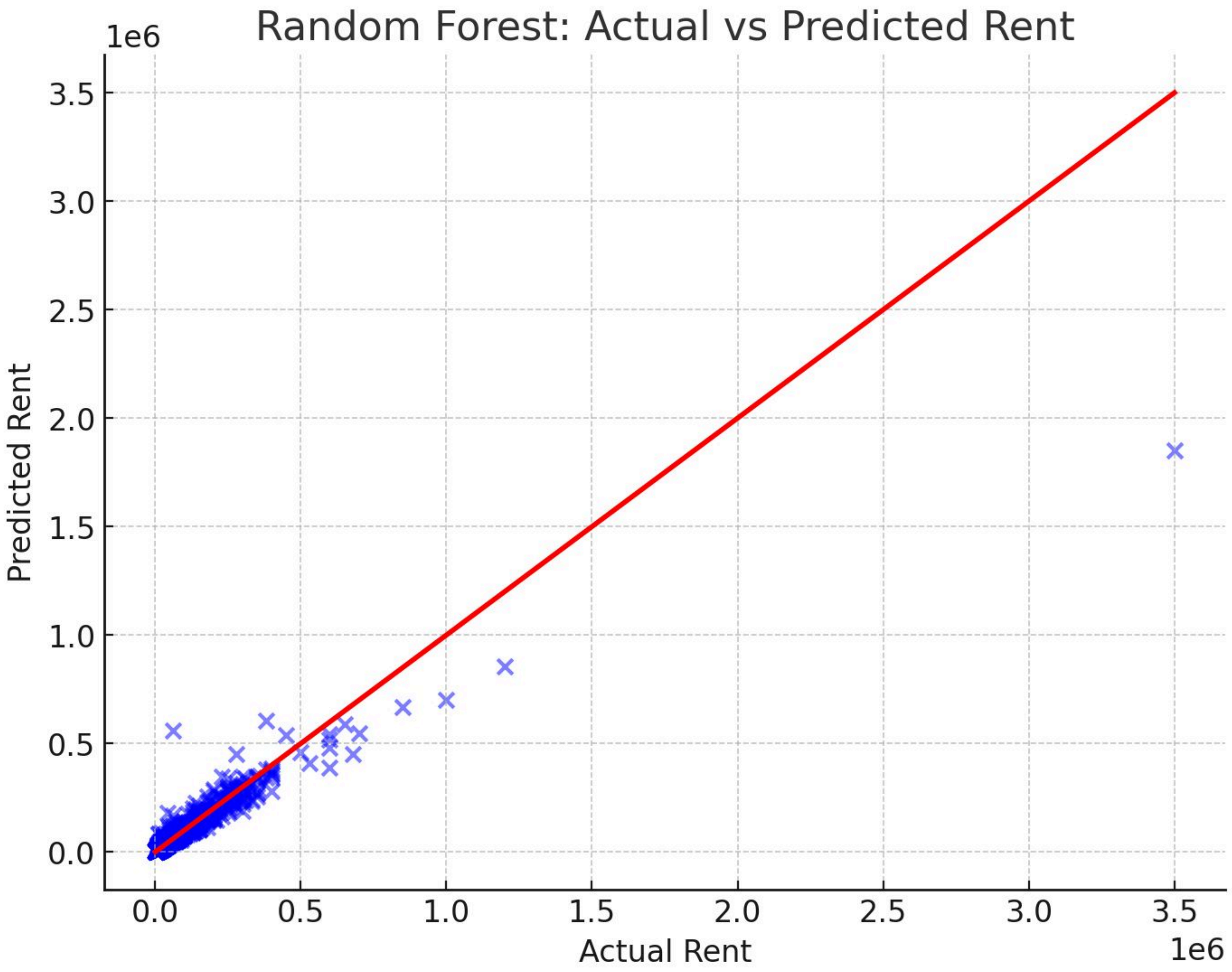
We also implemented **Random Forest**, an ensemble method that averages predictions across multiple decision trees to improve accuracy and reduce overfitting. Using 12 PCA components, Random Forest produced the following results:

- **R-squared (R^2)**: 0.860
- **Mean Squared Error (MSE)**: 852,456,500.00
- **Mean Absolute Error (MAE)**: 5,947.76


Random Forest: R-squared vs Number of PCA Components (Limited to 12)
R-squared (R²)
0.86
0.85
0.84
0.83
0.82
0.81
2
4
6
8
10
12
Number of PCA Components


Random Forest: MSE vs Number of PCA Components (Limited to 12)
1e9
1.15
1.10
1.05
1.00
0.95
0.90
0.85
Mean Squared Error (MSE)
2
4
6
8
10
12
Number of PCA Components


Random Forest: MAE vs Number of PCA Components (Limited to 12)
Mean Absolute Error (MAE)
8000
7500
7000
6500
6000
2
4
6
8
10
12
Number of PCA Components



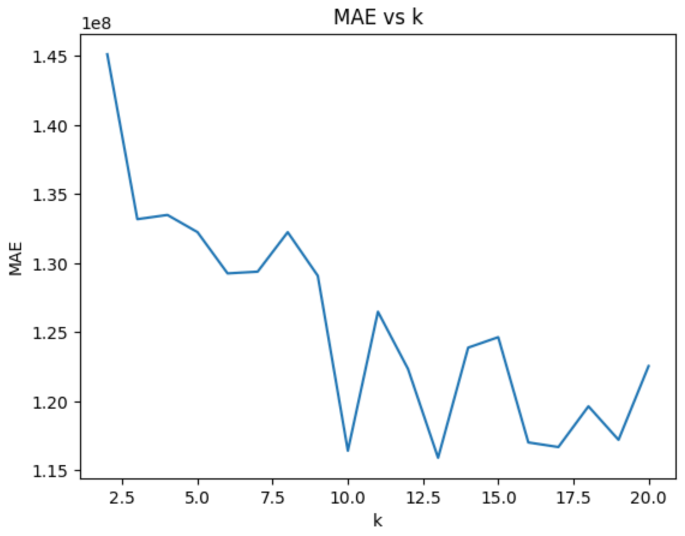
Random Forest excelled in minimizing MAE, making it particularly effective for typical rental price predictions. However, its higher MSE compared to XGBoost suggests it struggles slightly with extreme values, making it less accurate overall. Despite this, its performance in reducing variance demonstrates its utility in scenarios where stability and interpretability are prioritized.

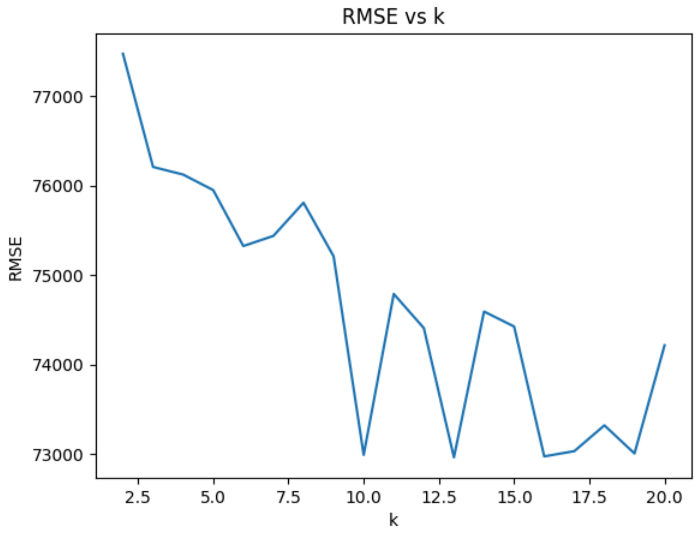
Unsupervised Learning

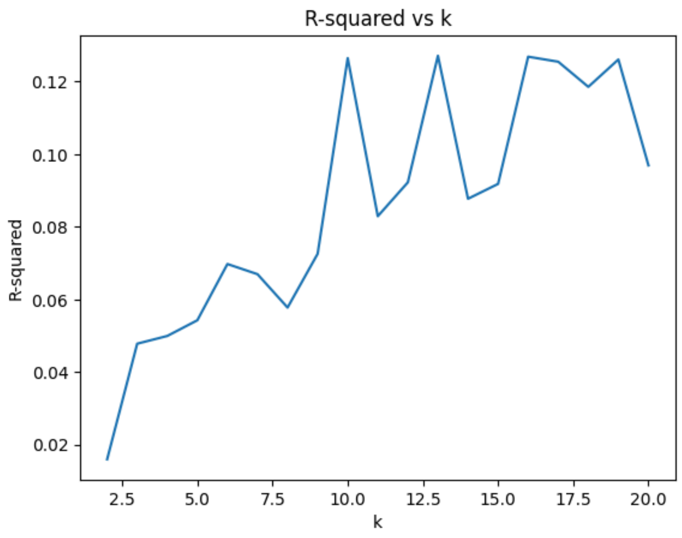
We implemented **KMeans clustering** as an unsupervised learning approach to explore the inherent structure of rental data. Clustering properties based on features like BHK, size, rent, and furnishing status revealed valuable insights into rental market segmentation.

Below is a table and graphs which show our results for the KMeans pipeline.

Clusters	MAE	R-squared	RMSE	Silhouette Score	WCSS
2	145121951.38	0.0159	77473.07	0.1818	9527.46
3	133183423.08	0.0478	76208.72	0.2037	7690.54
4	133490815.94	0.0499	76123.40	0.2318	6784.99
5	132245132.71	0.0543	75949.88	0.2320	6516.71
6	129258604.98	0.0698	75325.06	0.2603	5796.06
7	129385249.54	0.0670	75438.42	0.2379	5680.69
8	132243004.50	0.0578	75809.12	0.2514	5240.72
9	129086417.94	0.0726	75210.89	0.2911	4910.86
10	116419730.29	0.1265	72991.91	0.2798	4778.49
11	126486104.83	0.0829	74790.07	0.2818	4784.45
12	122343192.57	0.0923	74408.10	0.3088	4220.52
13	115909892.31	0.1271	72964.84	0.3309	4219.48
14	123885938.29	0.0877	74593.59	0.3152	3918.67
15	124643125.02	0.0918	74425.98	0.3496	3810.89
16	117023446.06	0.1269	72975.44	0.3733	3598.88
17	116689685.08	0.1255	73033.57	0.3799	3424.12
18	119640400.41	0.1186	73322.06	0.4279	3074.35
19	117211270.11	0.1261	73007.90	0.4305	3096.36
20	122555308.93	0.0969	74216.64	0.4385	2894.98

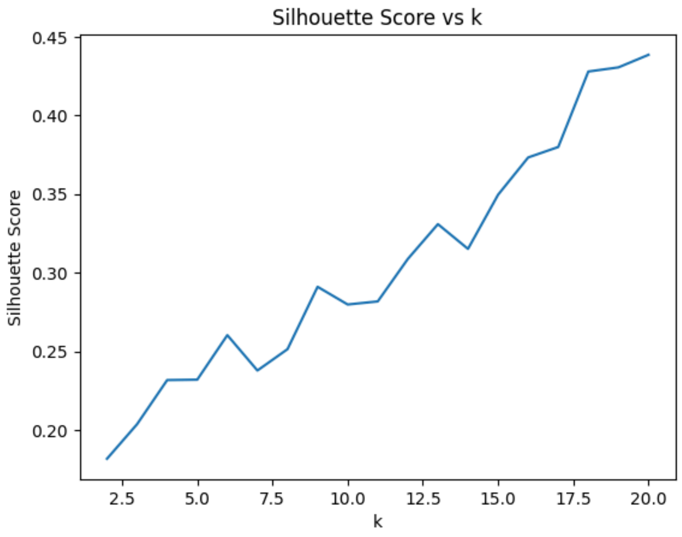

MAE vs k
1e8
MAE
k
1.45
1.40
1.35
1.30
1.25
1.20
1.15
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0

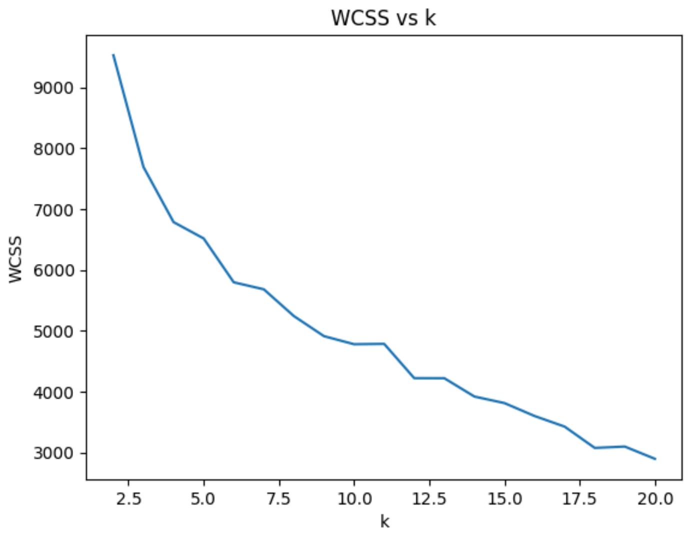

RMSE vs k
RMSE
k
77000
76000
75000
74000
73000
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0



Observe that in the data above, the error-oriented metrics like RMSE and MAE increase with the number of clusters, while accuracy-oriented metrics like the R-squared value increases with the number of clusters. This makes sense because each cluster is represented by the mean rent of the data points within it. More points would mean that more means are captured, increasing the granularity of the outputs.

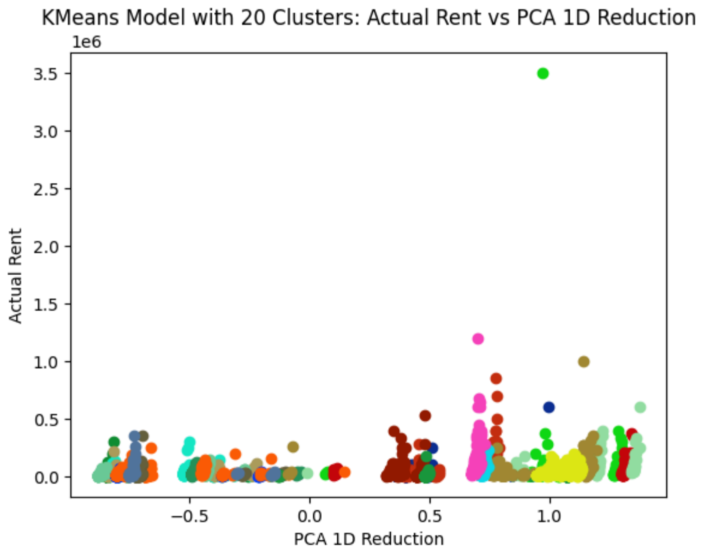
There are large increases and decreases in these graphs. We are still trying to decipher why there occur, but it likely has some explanation in the PCA reductions.



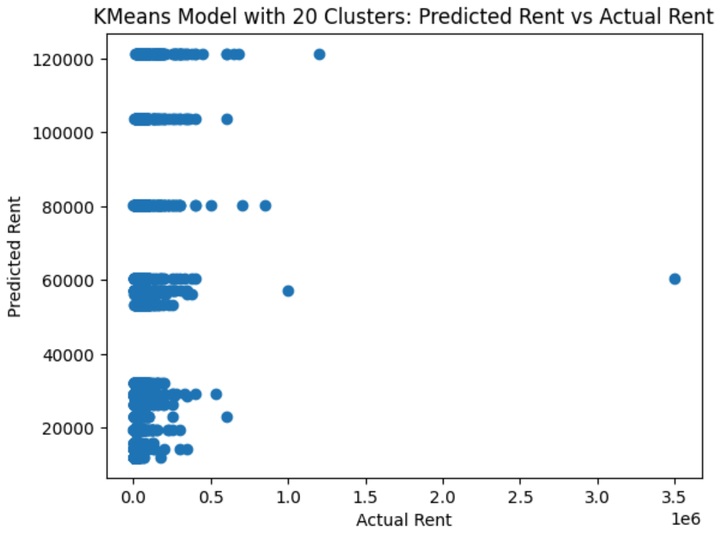


Observe that in the data above, the WCSS and Silhouette Score decrease and increase respectively, with no neat leveling off. We believe that this is explained by the nature of the KMeans algorithm. The rent prediction workload is inherently numerical and not classification-based. Thus, there is no real “optimal” value for hyperparameter `k`. While we predicted this result, it was fascinating to see it manifest in reality.

In terms of actual rent predictions, below are graphs which show the predictions for `k=20`.



Observe that the clustering is very similar to PCA groupings of the data. We believe that the overlap would be even stronger for higher dimensions of PCA.



Obviously, the predicted rent only has 20 distinct values. Yet, there seems to be a weak, positive correlation between the predicted rent and actual rent.

Comparative Analysis

Model	R²	MSE	MAE
Linear Regression	0.452	2,183,889,608.25	21,937.11
XGBoost	0.9500	304,867,164.79	8,664.85
Random Forest	0.860	852,456,500.00	5,947.76
KMeans 20 Cluster	0.0969	5,508,109,652.89	122,555,308.93

Key Insights:

1. **Linear Regression**: Performs poorly due to its inability to capture non-linear relationships. Useful as a baseline model.
2. **XGBoost**: Excels in overall accuracy and robustness, making it the best-performing model.
3. **Random Forest**: Balances accuracy and error reduction, with the lowest MAE, but slightly lags in overall accuracy (R^2).
4. **KMeans**: Strongly correlated with PCA data. This model has the worst error and accuracy, but understandably so, because it is mainly used for classification-oriented workloads. KMeans' limitation is that it cannot be applied to numerical scenarios.

Next Steps

1. **Enhance Feature Engineering**:
   - Integrate demographic data (e.g., locality-level median income) to improve prediction accuracy.
   - Experiment with PCA dimensions, particularly 14 components, for better variance retention.
2. **Model Optimization**:
   - Perform hyperparameter tuning for XGBoost and Random Forest to further optimize results.
3. **Advanced Techniques**:
   - Explore Neural Networks or hybrid approaches to capture deeper relationships in the data.

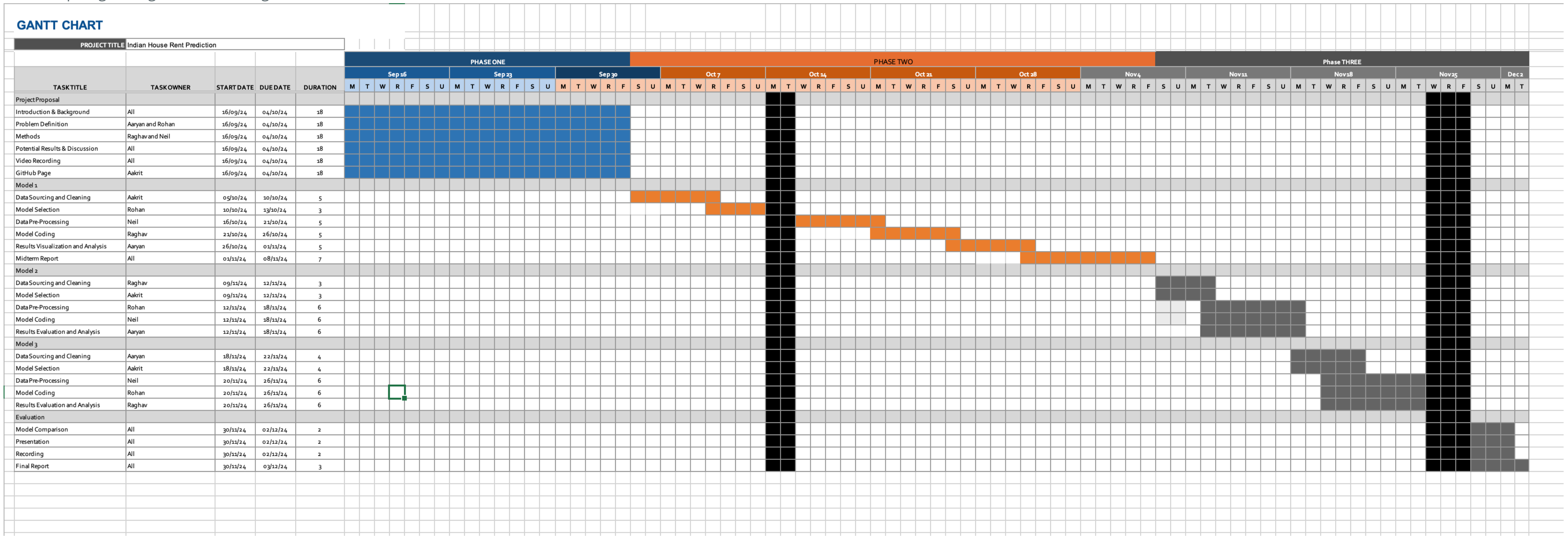
4. **Refine Clustering**:
    - Incorporate KMeans insights into supervised models for improved interpretability and performance.

These outlined next steps aim to improve the predictive accuracy of the models and derive deeper insights into rental market dynamics.

Additional Sections

Gantt Chart

refer: https://github.gatech.edu/asinghal90/CS_7641_Team_24/blob/002800fadc1bafb3580d51ecb3f6582a663d5488/Gantt_Chart.xlsx or:



GANTT CHART

PROJECT TITLE Indian House Rent Prediction

TASK TITLE	TASK OWNER	START DATE	DUE DATE	DURATION
Project Proposal				
Introduction & Background	All	16/09/24	04/10/24	18
Problem Definition	Aaryan and Rohan	16/09/24	04/10/24	18
Methods	Raghav and Neil	16/09/24	04/10/24	18
Potential Results & Discussion	All	16/09/24	04/10/24	18
Video Recording	All	16/09/24	04/10/24	18
GitHub Page	Aakrit	16/09/24	04/10/24	18
Model 1				
Data Sourcing and Cleaning	Aakrit	05/10/24	10/10/24	5
Model Selection	Rohan	10/10/24	13/10/24	3
Data Pre-Processing	Neil	16/10/24	21/10/24	5
Model Coding	Raghav	21/10/24	26/10/24	5
Results Visualization and Analysis	Aaryan	26/10/24	01/11/24	5
Midterm Report	All	01/11/24	08/11/24	7
Model 2				
Data Sourcing and Cleaning	Raghav	09/11/24	12/11/24	3
Model Selection	Aakrit	09/11/24	12/11/24	3
Data Pre-Processing	Rohan	12/11/24	18/11/24	6
Model Coding	Neil	12/11/24	18/11/24	6
Results Evaluation and Analysis	Aaryan	12/11/24	18/11/24	6
Model 3				
Data Sourcing and Cleaning	Aaryan	18/11/24	22/11/24	4
Model Selection	Aakrit	18/11/24	22/11/24	4
Data Pre-Processing	Neil	20/11/24	26/11/24	6
Model Coding	Rohan	20/11/24	26/11/24	6
Results Evaluation and Analysis	Raghav	20/11/24	26/11/24	6
Evaluation				
Model Comparison	All	30/11/24	02/12/24	2
Presentation	All	30/11/24	02/12/24	2
Recording	All	30/11/24	02/12/24	2
Final Report	All	30/11/24	03/12/24	3

Contribution Table

Name	Contributions
Raghav	Algorithm Analysis/Visualizations
Aaryan	Model Development
Neil	Model Development
Aakrit	Data Preprocessing
Rohan	Data Preprocessing

References

[1] Human Rights Measurement Initiative, “Right to Housing in India,” Rightstracker.org. [Online]. Available: https://rightstracker.org/. Accessed: Oct. 4, 2024.

[2] S. Mallick, et al., “Rental Price Prediction Using Machine Learning Algorithms,” Int. J. Comput. Appl., vol. 182, no. 39, pp. 25-30, 2019.

[3] Y. Zhang and X. Li, “Predicting Rental Prices Using Ensemble Learning Methods,” in Proc. IEEE Int. Conf. Data Sci. Adv. Anal., 2020, pp. 300-309.

[4] A. Ng, “Feature selection, L1 vs. L2 regularization, and rotational invariance,” in Proceedings of the 21st International Conference on Machine Learning (ICML), 2004, pp. 78-85.

[5] F. Pedregosa et al., “Scikit-learn: Machine Learning in Python,” Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011.