Apartment Rent Prediction Project Milestone 1

Team: CS_13

Team Members

Name	ID	section	Department
مصطفي حسين محمود	2021170519	6	CS
وليد محمد مصلح	2021170616	7	
كيرلس سمعان رزق	2021170411	5	
منة الله عبد العاطي محمد	2021170548	7	
منة الله ابراهيم حسن	2021170550	7	
هاجر عصام محمود محمود	2021170600	7	

1) Preprocessing

Step 1: Extracting Feature from another

extracting number of beds from the ‘body column and adding the maximum number of beds mentioned in body column in a new column ‘beds’ by the function
`extract_beds(text)`
the column is dropped as the correlation is lower than 0.2

Step 2: Dealing with missing values

- getting the number of null values in y (target) using:
  `y.isnull().sum()`
  Then drop the row from the data using:
  `proj = proj.dropna(subset=["price_display"])`
  Note that ‘proj’ is the whole dataset.

- getting the number of null values in x (features) using this code:
  `x.isnull().sum()`

- Then replace it by using **replace_nulls(df)** that:
  - Filling missing values with the mean for numeric columns.
  - Filling missing values with the mode for string columns.

Step 3: Detecting and Handling the outliers

There are two functions are used there:

- **outlier_detection(df)**: This function calculates outliers for numeric columns using the Interquartile Range (IQR) method and prints the count of outliers for each numeric column.

- **replace_outliers(df)**: This function replaces outliers with the third quartile value (Q3) for each numeric column.

This step is canceled as the accuracy increases without it (sometimes the data has important value to add even it looks like it is an outlier)

Step 4: Data Encoding

encoding the categorical variables into a numerical format to allow the algorithms to process them effectively.

Encoding is done using 3 techniques:

- **Target Encoding for columns**
  {'cityname', ' amenities', 'address'}
  These columns have the most effect on the target column so this is the most suitable type of encoding for these columns.

- **One Hot Encoding for 'pets_allowed'**
  This column has few classes so this is the most suitable encoding type for it. And after encoding we tried to reduce the number of encoded columns into two columns one for cats and one for dogs.

- **Label Encoding for the remaining features**
  This is the most suitable type of encoding for these columns as applying any other encoding types can confuse the prediction process.

Step 5: Feature Scaling

Doing it ensures that all features have the same scale, preventing features with larger scales from dominating those with smaller scales during model training. It equalizes the importance of features and helps the algorithm converge faster. Using **MinMaxScaler().**

Step 6: Feature Selection and Removal

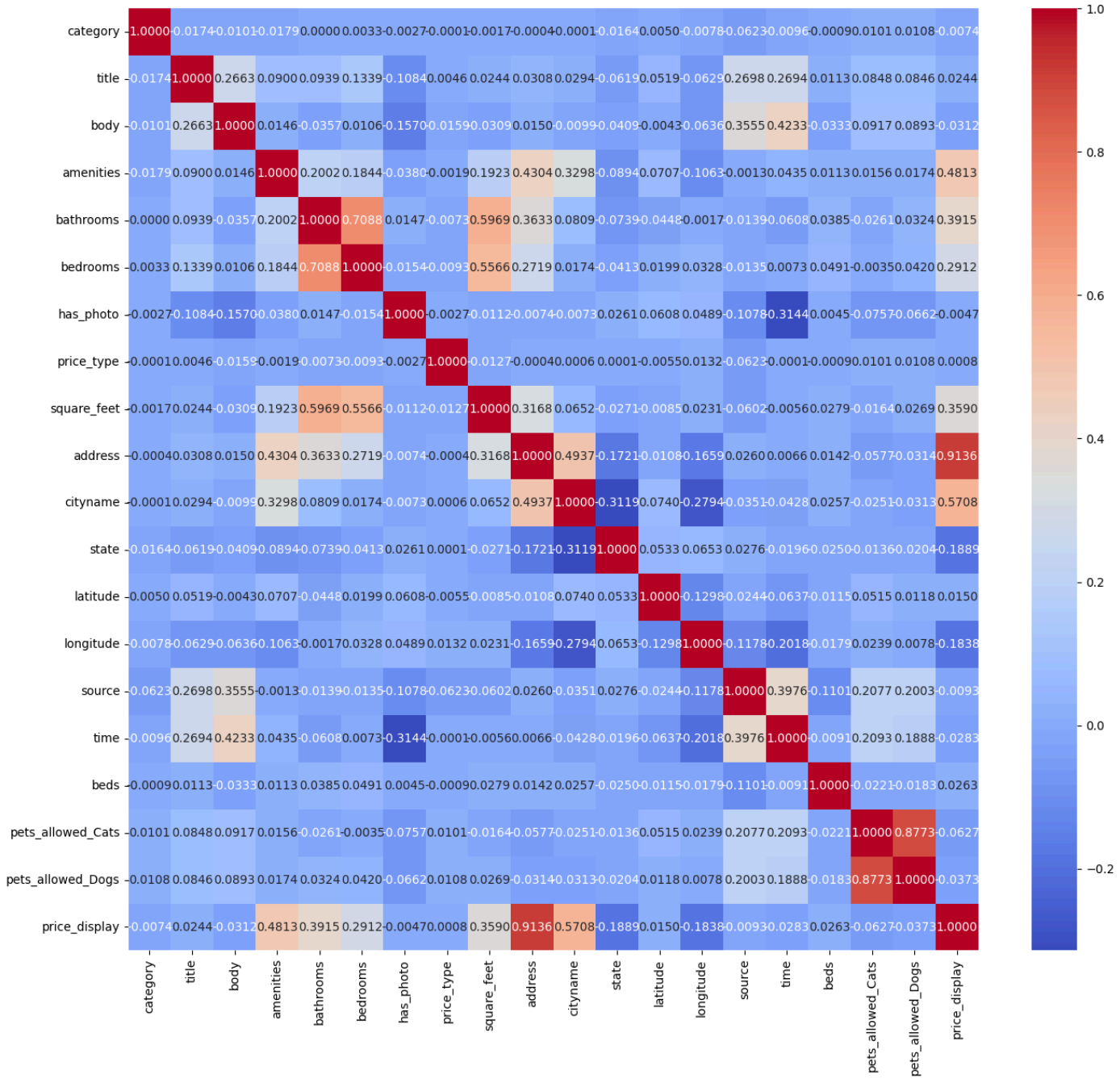
After applying **x.unique()** We found that the 'currency' and 'fee' columns have only one unique value each after pre-processing, constant columns are redundant and do not add any additional information so we dropped them. And for the 'id' column the unique values equal the number of data records which either do not add any additional information so we dropped it using:

```
x.drop(['id', 'currency', 'fee'], axis=1)
```

Step 7: Correlation Analysis

Using **correlation(x,y)**to compute the correlation matrix between features and the target.

It visualizes the correlation matrix using a heatmap and selects features with correlation coefficients greater than **0.2** with the target.



2) Applying Model

We evaluate the performance of various regression models using the provided dataset and features.

The models considered include:

- Linear Regression
- Ridge Regression
- Lasso Regression
- Polynomial Regression
- Random Forest
- Decision Tree

Model Performance:

Model Name	Train MSE	Train R2 Score	Test MSE	Test R2 Score
Linear Regression	160053.91	0.88	183729.50	0.76
Ridge Regression	161949.37	0.88	181029.15	0.77
Lasso Regression	160062.32	0.88	183681.34	0.76
Polynomial Regression	160053.91	0.88	183729.50	0.76
Random Forest	37350.14	0.97	51242.36	0.93
Decision Tree	25273.89	0.98	79016.24	0.90

The **Random Forest model** achieved the highest R2 Score on the test set, indicating **strong predictive performance**.

Features Used:

- amenities
- bathrooms
- bedrooms
- square_feet
- address
- cityname

Model Iteration and Evaluation:

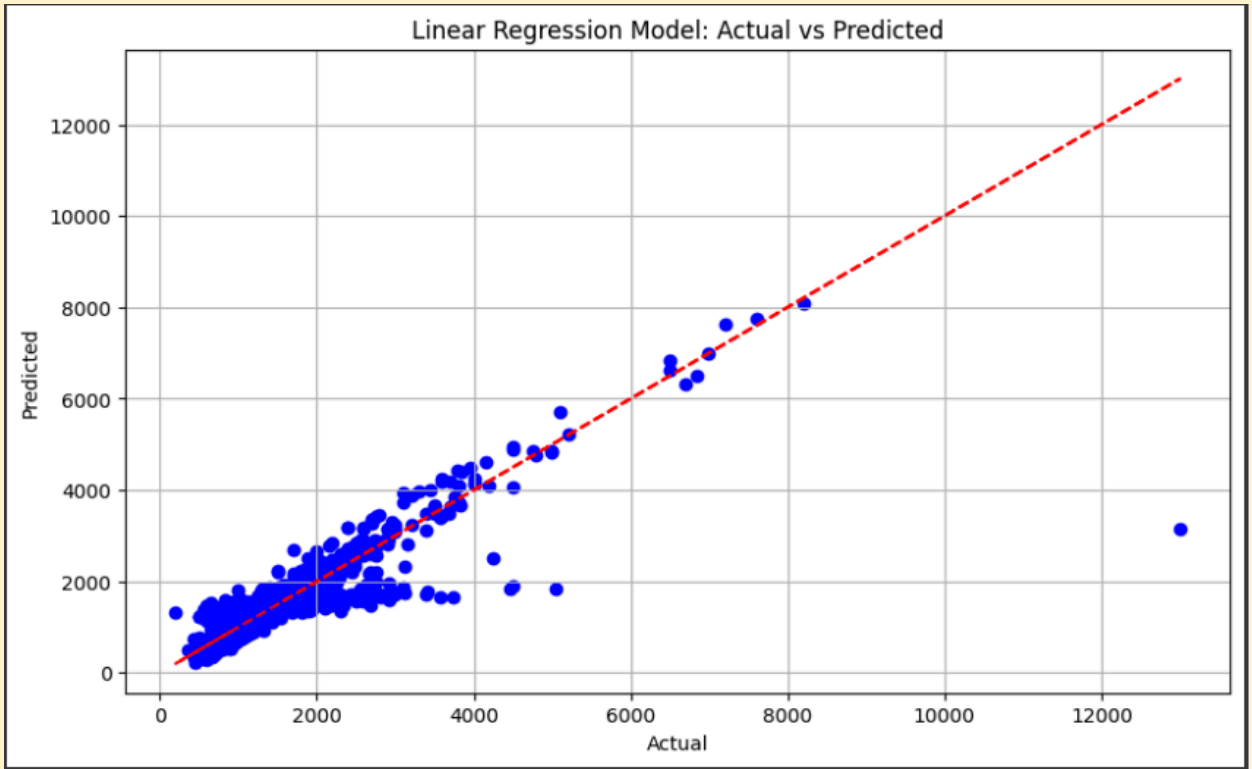
The code iterates over each model, applies Recursive Feature Elimination with Cross-Validation (RFECV) for selected models, fits the model on features, evaluates MSE and R2 Score, splits data into 80% training and 20% testing sets (random state 42), and calculates metrics for both sets.

Training and Testing Set Sizes:

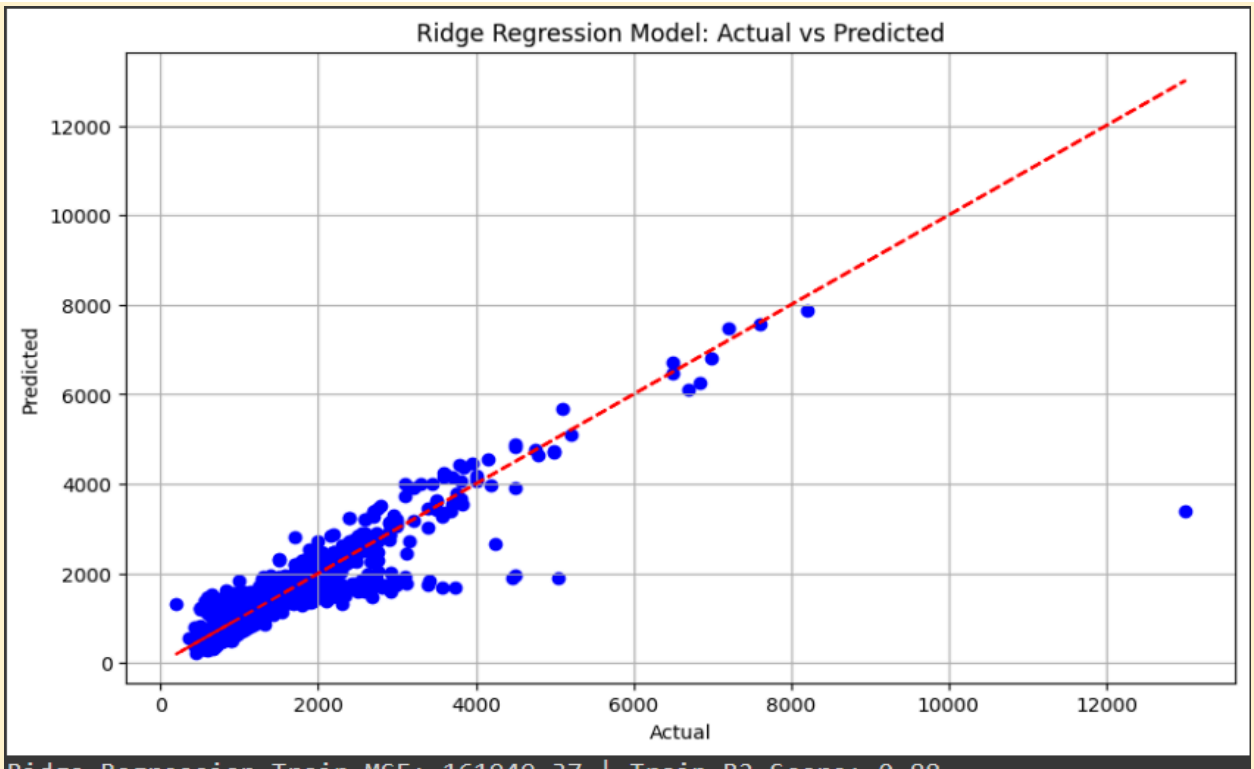
- The training set size is set to 80% of the data, while the testing set size is 20%.
- The random state of 42 ensures consistency in the split for reproducibility.

Performance Evaluation and Visualization:

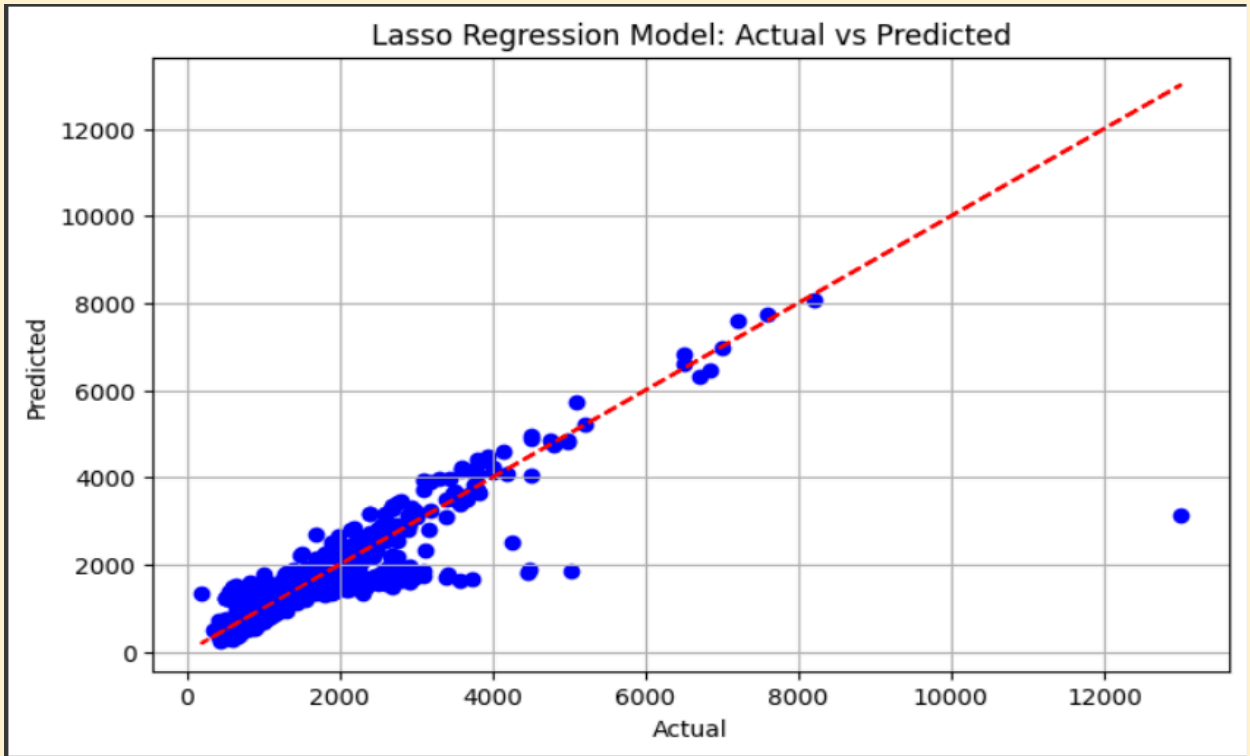
- The code calculates MSE and R2 scores for the training and testing sets.
- It then generates scatter plots comparing the actual vs. predicted values for each model, providing a visual assessment of predictive performance. These scatter plots are presented below:



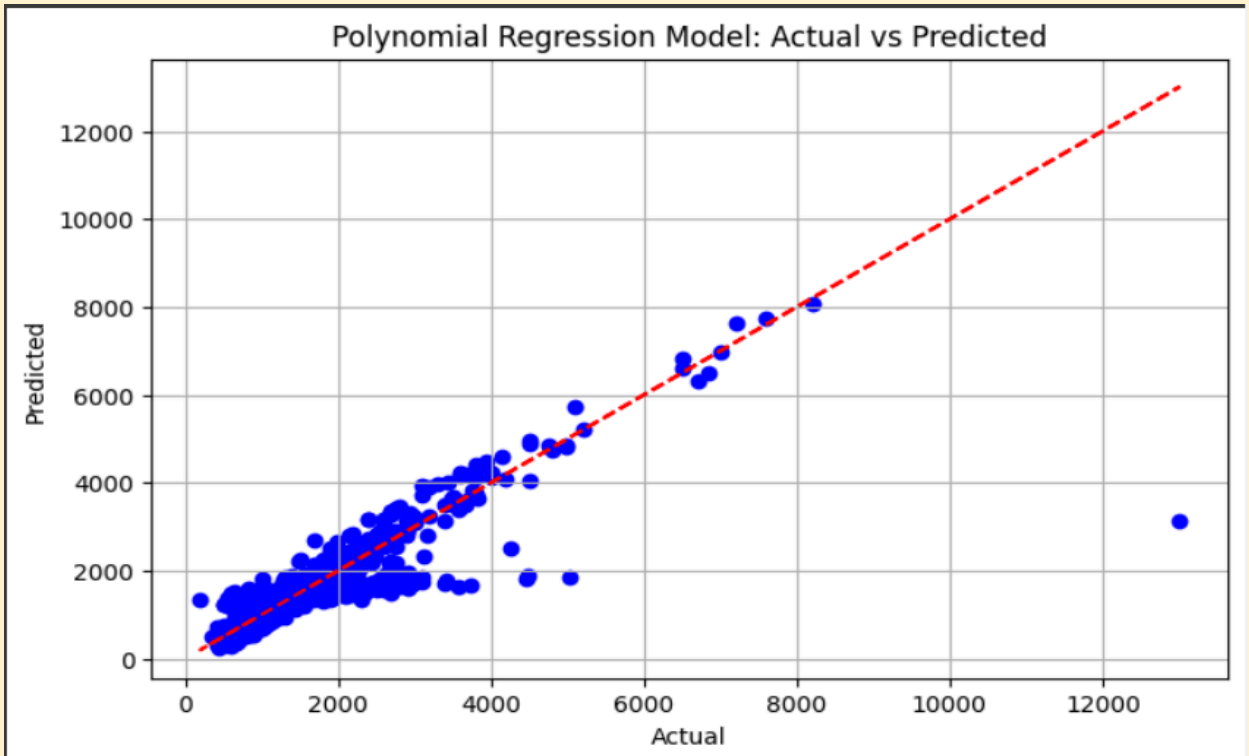

```
Linear Regression Train MSE: 160053.91 | Train R2 Score: 0.88
Linear Regression Test MSE: 183729.50 | Test R2 Score: 0.76
```



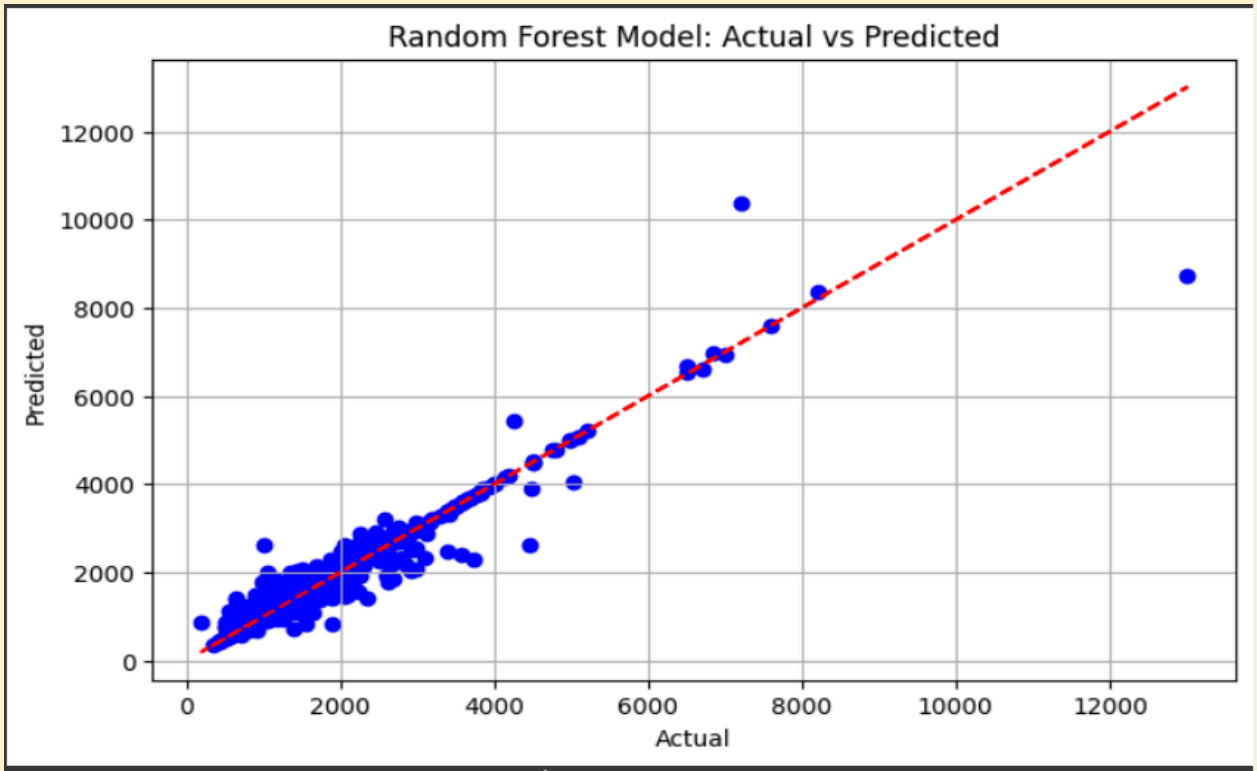

```
Ridge Regression Train MSE: 161949.37 | Train R2 Score: 0.88
Ridge Regression Test MSE: 181029.15 | Test R2 Score: 0.77
```



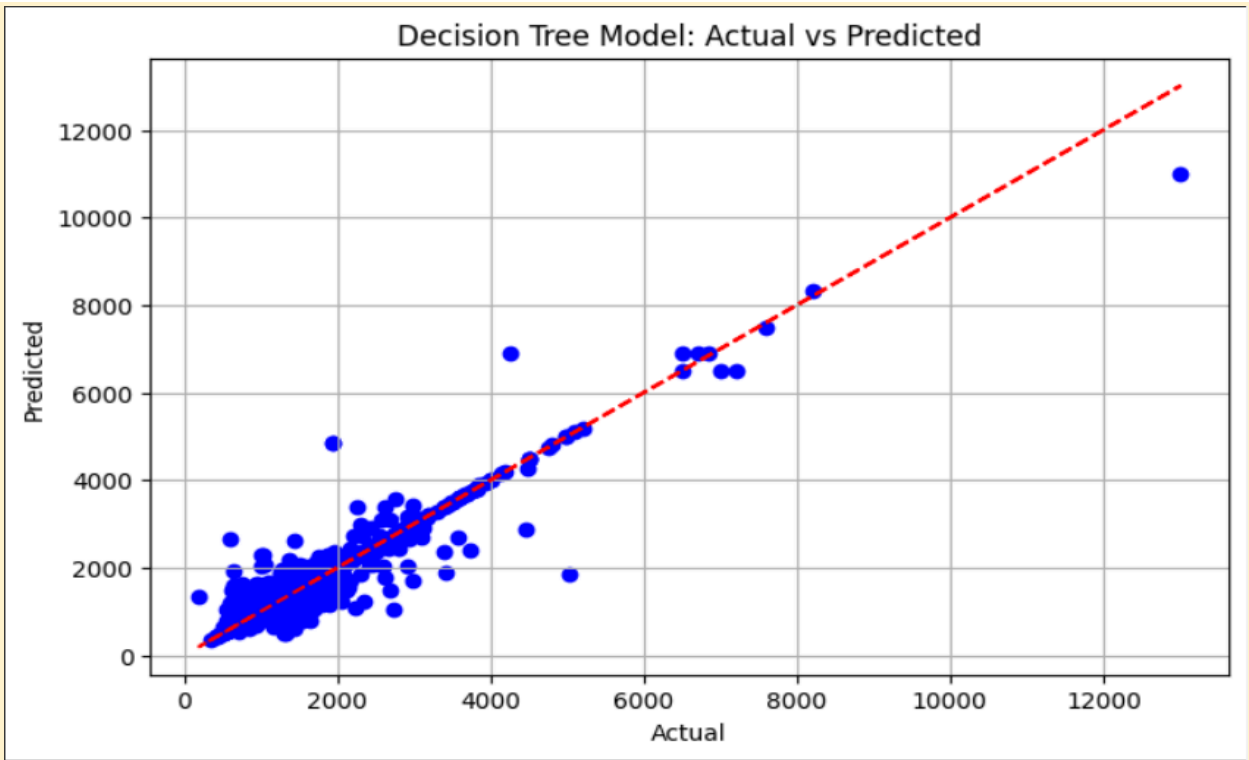

```
Lasso Regression Train MSE: 160062.32 | Train R2 Score: 0.88
Lasso Regression Test MSE: 183681.34 | Test R2 Score: 0.76
```




```
Polynomial Regression Train MSE: 160053.91 | Train R2 Score: 0.88
Polynomial Regression Test MSE: 183729.50 | Test R2 Score: 0.76
```




```
Random Forest Train MSE: 37350.14 | Train R2 Score: 0.97
Random Forest Test MSE: 51242.36 | Test R2 Score: 0.93
```




```
Decision Tree Train MSE: 25273.89 | Train R2 Score: 0.98
Decision Tree Test MSE: 79016.24 | Test R2 Score: 0.90
```

Conclusion:

In the Apartment Rent Prediction Project, we aimed to predict the rent of apartments based on various features. We started by preprocessing the data, which involved feature extraction, handling missing values, data encoding, feature scaling, feature selection and removal, and correlation analysis.
For handling missing values, we dropped rows with null values in the target column and replaced missing values in the features with the mean for numeric columns and the mode for string columns. Outlier detection and handling were initially performed using the IQR method, but this step was later canceled as it did not improve the accuracy and some outliers contained important values for predicting. Data encoding was done using target encoding for columns with the most effect on the target column, one-hot encoding for the 'pets_allowed' column, and label encoding for the remaining features. Feature scaling was performed using MinMaxScaler() to ensure all features were in the same range. Correlation analysis was used to select features with correlation coefficients greater than **0.2** with the target.
Several regression models were evaluated, including Linear Regression, Ridge Regression, Lasso Regression, Polynomial Regression, Random Forest, and Decision Tree. The Random Forest model achieved the highest R2 Score on the test set, indicating strong predictive performance. The features used were 'amenities', 'bathrooms', 'bedrooms', 'square_feet', 'address', and 'cityname'.
Our initial intuition was that outlier handling would improve the accuracy of our model. However, this was disproved. Our intuition about the Random Forest model being the best performer was proved correct, as it achieved the highest R2 Score on the test set.
Overall, this phase of the project involved a thorough preprocessing of the data and evaluation of various regression models, leading to a strong predictive model for apartment rent.