TEAM : 16

Yash Zaveri

Zeel Patel

**HOUSING PRICE PREDICTION**

# CONTENTS

[INTRODUCTION 2](#_Toc153109519)

[Background: 2](#_Toc153109520)

[Motivation: 2](#_Toc153109521)

[Goal: 2](#_Toc153109522)

[METHODOLOGY 3](#_Toc153109523)

[Data Pre-processing: 3](#_Toc153109524)

[Data Analysis: 3](#_Toc153109525)

[Feature Engineering: 3](#_Toc153109526)

[Model Selection: 3](#_Toc153109527)

[RESULT AND ANALYSIS 4](#_Toc153109528)

[Cleaning and Feature Engineering on Dataset: 4](#_Toc153109529)

[Removing Outliers: 9](#_Toc153109530)

[Model Selection: 16](#_Toc153109531)

[Random Forest Model: 17](#_Toc153109532)

[Support Vector Machine: 18](#_Toc153109533)

[Problems faced: 19](#_Toc153109534)

[CONCLUSION 23](#_Toc153109535)

[REFERENCES 24](#_Toc153109536)

# ****INTRODUCTION****

## Background:

The housing price prediction project is a cutting-edge initiative aimed at leveraging advanced data analysis and machine learning techniques to forecast real estate values based on a comprehensive set of factors. By incorporating features such as the property's location, historical housing prices in the area, the number of bedrooms and bathrooms, and the overall area of the house, the project seeks to create a predictive model that can offer valuable insights into future property values. This approach represents a paradigm shift in real estate decision-making, moving beyond traditional methods by harnessing the power of extensive datasets to identify intricate patterns and correlations. The project not only addresses the inherent challenges of market volatility and regulatory changes but also provides a dynamic, adaptable model that can adjust to evolving market conditions.

## Motivation:

Our motivation for this housing price prediction project is grounded in the need to provide valuable insights for participants in the real estate market. Like the cryptocurrency market, where price fluctuations and market sentiment are crucial, the housing market also faces challenges influenced by factors like location, historical pricing, and property attributes. By harnessing advanced data analysis and machine learning techniques, our project aims to develop a predictive model tailored to assist individuals in understanding and forecasting housing price dynamics.

## Goal:

Our project objectives extend to analysing historical data in the real estate domain, aiming to identify patterns, trends, and influential factors for predictive modelling. Recognizing that some houses may lack individual past data, we will employ surrounding housing prices and neighbourhood trends as crucial indicators. By scrutinizing neighbouring property values and broader housing market dynamics, our goal is to develop a robust predictive model for housing prices, enabling informed decision-making even in cases where specific past data is unavailable. This approach seeks to enhance the predictability of property values by considering the broader context of the surrounding housing landscape and neighbourhood’s trends.

# METHODOLOGY

## Data Pre-processing:

This involves handling missing values, addressing outliers, standardizing and normalizing numerical features, and encoding categorical variables. Additionally, feature engineering, dealing with duplicates, and splitting the dataset into training and testing sets are key steps. Th e process ensures a clean and standardized dataset, laying the groundwork for effective machine learning model development.

## Data Analysis:

This involves analysing summary statistics like mean, median, and standard deviation for each property. In addition to numerical and categorical exploration of the data, one can conduct correlation analyses between variables, investigate sentiment scores derived from property descriptions, and visualize property prices. Geospatial insights can be gained if latitude and longitude information is available. Overall, a comprehensive data analysis can unveil trends, patterns, and relationships within the property dataset, facilitating informed decision-making for various stakeholders.

## Feature Engineering:

In housing price prediction, feature engineering is crucial for transforming raw data into informative model input. Key features include historical property values, square footage trends, and neighbourhood’s characteristics. Lag features, moving averages, and sentiment indicators capture historical dynamics, while normalization ensures consistency. Custom features may reflect specific events or regulatory changes in the real estate market. Dummy variables, representing categorical variables like neighbourhood or property type, play a crucial role in encoding qualitative information for machine learning models.

## Model Selection:

In our approach to housing price prediction, we adopt a diverse set of machine learning models, including Random Forest Regression, Decision Trees and Support Vector Machines. These models, trained on a comprehensive dataset comprising historical housing prices and property attributes, collectively aim to forecast property values and discern trends in the dynamic real estate market.

# RESULT AND ANALYSIS

## Cleaning and Feature Engineering on Dataset:

Import Libraries and Read CSV File: Pandas provides the read\_csv function, which allows users to effortlessly load data from a CSV (Comma Separated Values) file into a DataFrame—a two-dimensional, tabular data structure

A screenshot of a computer

Description automatically generated

The info() method in pandas offers a succinct overview of a DataFrame, encompassing details about data types, the presence of non-null values, and memory usage.

A screenshot of a computer

Description automatically generated

Null Value Heat Map:

A purple and yellow chart

Description automatically generated

In the analysis of DataFrame df1, we employed the groupby function to examine the distribution of laundry options, obtaining a Pandas Series that elucidates the frequency of each option. This insightful overview enhances our understanding of the dataset's characteristics. Additionally, in the data preprocessing phase, a refined DataFrame df2 was created by eliminating non-essential columns from df1, such as 'id', 'url', 'region\_url', 'image\_url', 'lat', and 'long'.

A screenshot of a computer

Description automatically generated

In our data analysis with DataFrame df2, the Pandas isnull().sum() method was applied to quantify null values in each column systematically. This comprehensive summary aids in understanding data quality and guides subsequent preprocessing steps. In the creation of df3, we removed 'laundry\_options' and 'parking\_options' columns, addressing null values detected in these columns, thereby refining the dataset for subsequent analysis or machine learning tasks.

A screenshot of a computer

Description automatically generated

After removing null values from data set :

A purple rectangle with white text

Description automatically generated

Below code preprocesses text data in a DataFrame ({df4}) by utilising the Natural Language Toolkit (NLTK) module. It entails downloading NLTK resources (the punkt tokenizer and stopwords in particular). The script defines a function named `normalize\_desc} that removes stopwords, tokenizes, converts to lowercase, and removes punctuation from the DataFrame's 'description' column. The preprocessed text is then added to a new column called "normalised\_desc" by applying this function. The code's overall goal is to improve text data quality for jobs involving natural language processing (NLP) by standardising and cleaning the textual data…[1]

A close up of words

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A close-up of words

Description automatically generated

TextBlob is a free, open-source library in Python for processing textual data. It is a powerful package that reduces the complexity of the contextual data and derives in-depth information from the text.

This code excerpt introduces sentiment analysis to the DataFrame df4 by utilizing the TextBlob library. A customized function, analyze\_sentiment, has been crafted to evaluate the sentiment polarity of text found in the 'description' column. The resulting sentiment polarity scores, which represent the emotional tone of the text on a scale from -1 (most negative) to 1 (most positive), are subsequently added to a newly created column called 'sentiment\_score' in the DataFrame. This incorporation of sentiment analysis furnishes a quantitative metric for the sentiment present in the textual data, facilitating a more profound comprehension of the emotional context associated with each description in the dataset…………[2]

A screenshot of a computer

Description automatically generated

We write a code to utilizes Pandas and Matplotlib to generate a bar graph displaying average sentiment scores for states in DataFrame df4. It calculates mean sentiment scores using groupby, sorts states based on these scores, and utilizes Matplotlib to create a visual representation, offering insights into sentiment trends and potential variations across states.

A graph of blue lines

Description automatically generated

These code introduce a new 'price\_per\_sqft' column in DataFrame df6, computed by dividing the 'price' column by the 'sqfeet' column. This column reveals insights into cost efficiency and property pricing relative to size. Using df6.head() displays the initial rows of the updated dataset, offering a quick overview of the newly added information…….[3]

A screenshot of a computer

Description automatically generated

The DataFrame df6 undergoes preprocessing steps specifically targeting the 'region' column. The application of the apply method, coupled with a lambda function, ensures the removal of leading and trailing whitespaces from each element in the 'region' column. Subsequently, a new DataFrame named region\_stats is generated, presenting a summary of unique region counts sorted in descending order. The subsequent code lines focus on identifying regions with counts below or equal to 100, leading to the creation of the region\_stats\_less\_than\_100 DataFrame. The ensuing modification of the 'region' column in df6 involves grouping less common regions under the label 'other'. Ultimately, the calculation of unique region counts after this modification contributes to the code's efficacy in cleaning, categorizing, and offering valuable insights into regional diversity while enhancing dataset manageability……[3]

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

## Removing Outliers:

Now we removes outliers from DataFrame df6 by filtering rows where the ratio of square feet to the number of bedrooms is less than 250. The resulting DataFrame df7 contains instances where this ratio exceeds 250. Subsequently, descriptive statistics on the 'price\_per\_sqft' column in df7 provide insights into the distribution of prices per square foot, shedding light on potential outliers and enhancing data quality for further analysis………[3][4]

A screenshot of a computer

Description automatically generated

The statistical analysis reveals several concerns in the distribution of apartment prices per square foot. The notably high standard deviation of $4,005.34 suggests potential outliers, warranting further examination to understand their impact. The minimum value of $0 raises suspicions of missing or erroneous data, requiring investigation for data integrity. The extreme maximum value of $2,476,125, significantly surpassing the 75th percentile, indicates potential outliers that could skew the analysis. Additionally, the discrepancy between mean and median underscores the influence of outliers on the mean, emphasizing the importance of considering the median for a robust measure of central tendency. Addressing these concerns is crucial for refining the dataset and ensuring meaningful analysis of apartment prices per square foot…..[3]

A white rectangular object with a white border

Description automatically generated with medium confidence

A blue bar graph with white text

Description automatically generated

The function remove\_pps\_outliers in the provided code is designed to eliminate outliers from a DataFrame by utilising the 'price\_per\_sqft' column. Every region in the input DataFrame ({df{) is iterated over by the function, which then removes rows if the 'price\_per\_sqft' deviates more than one standard deviation from the mean in that region. After that, the reduced DataFrames for every region are concatenated with the resultant DataFrame, {df\_out}, to produce a final DataFrame that is outlier-free. This improved DataFrame is returned by the function.

The DataFrame {df7} is transformed into a new DataFrame {df8} by applying the `remove\_pps\_outliers} function. The shape of `df8} is printed, indicating how many rows and columns remain in the DataFrame following the removal of outliers. By removing extreme values that could skew the analysis or modelling results, this procedure helps to improve the dataset's robustness.

A screenshot of a computer

Description automatically generated

A graph of two people

Description automatically generated with medium confidence

Then we defines a function named plot\_scatter\_chart that utilizes the Matplotlib library to create a scatter plot comparing apartment prices per square foot for 2 BHK (bedroom, hall, kitchen) and 3 BHK configurations in a specified location. The function takes a DataFrame (df) and a location as parameters, filters the data for 2 BHK and 3 BHK apartments in the given region, and then plots the scatter points with different colors and markers for each bedroom type. The x-axis represents the total square feet area, and the y-axis represents the price per square foot. The function also adjusts the figure size, sets axis labels, and adds a legend for clarity. It's a useful tool for visually comparing the price distribution of different bedroom configurations in a specific location. Additionally, the code snippet lacks a call to the plt.show() function, which is necessary to display the plot when using Matplotlib in interactive mode……..[3]

A screenshot of a computer

Description automatically generated

A screen shot of a graph

Description automatically generatedA blue and green dots

Description automatically generated

(Abilene) (Bakersfield)

Created a function named remove\_bhk\_outliers that focuses on the removal of outliers from a DataFrame, targeting the 'price\_per\_sqft' column and considering the number of bedrooms ('beds'). The function systematically analyzes each location in the DataFrame (df) by computing key statistics, including mean, standard deviation, and count, for the price per square foot across different bedroom configurations within that location. It then identifies outliers by comparing the price per square foot for a specific bedroom configuration with the mean of the preceding bedroom configuration, contingent on the count surpassing 5. Rows corresponding to these identified outliers are consolidated in the exclude\_indices array. In its final step, the function returns a refined DataFrame with the exclusion of the identified outliers. This approach contributes to dataset improvement by tailoring the outlier removal process to the characteristics of each bedroom configuration within a given location………[3]

A screenshot of a computer code

Description automatically generated

A screen shot of a graph

Description automatically generated

Then code utilizes Matplotlib to construct a histogram, visualizing the distribution of house prices per square foot in the DataFrame df9. With a larger figure size, the histogram is configured with 10 bins, a relative width of 0.8, and a range from 1 to 10 on the x-axis. The resulting plot provides a concise overview of how house prices are distributed across various price per square foot ranges. Notably, the histogram reveals a significant concentration of houses in the 1 to 3 thousand dollars price range per square foot, suggesting a prevalent cluster of more affordable housing options and contributing valuable insights into the dataset's pricing patterns.

A graph of a bar graph

Description automatically generated with medium confidence

The code utilizes Matplotlib to create a histogram that visualizes the distribution of the number of bathrooms (baths) in the DataFrame df9. The histogram is generated with default binning, and the relative width of the bars is set to 0.8 for clarity. The x-axis represents the number of bathrooms, while the y-axis depicts the count of houses with a corresponding number of bathrooms. This visualization provides an insightful overview of the distribution of bathroom counts in the dataset, aiding in understanding the prevalence of different bathroom configurations among the houses. Notably, the bar chart highlights that the most common number of bathrooms is 2, followed by 3, offering valuable information on the predominant bathroom configurations in the dataset.

A graph of blue bars

Description automatically generated with medium confidence

The code snippet assesses the DataFrame df9 to identify instances where the number of bathrooms exceeds the number of bedrooms by more than two. The shape of the resulting DataFrame, representing such cases, is determined. Subsequently, a new DataFrame df10 is created by filtering out entries where the number of bathrooms is less than or equal to the number of bedrooms plus two. The shape of df10 is then computed. This process helps refine the dataset by eliminating instances where the relationship between the number of bedrooms and bathrooms may be inconsistent, contributing to a more coherent dataset for further analysis or modeling…….[3]

A screenshot of a computer

Description automatically generated

The heatmap below shows the correlation between the different features in a dataset. The darker the shade of red, the stronger the positive correlation, and the darker the shade of blue, the stronger the negative correlation. These correlations make sense given the nature of the data. For example, it is expected that larger homes (more sqfeet) and homes with more bathrooms (baths) would be more expensive (price). Additionally, homes with electric vehicle charging stations (electric\_vehicle\_charge) may be more desirable and expensive, especially in areas with a high concentration of electric vehicles.There are also a few negative correlations in the heatmap. For example, there is a negative correlation between smoking\_allowed and wheelchair\_access. This suggests that homes that allow smoking are less likely to be accessible to people with disabilities.

A graph with red and blue squares

Description automatically generated

Here are some additional insights that can be drawn from the heatmap after data processing: The features price, sqfeet, baths, beds, electric\_vehicle\_charge, and sentiment\_score are all highly correlated with each other, suggesting that they may contain redundant information. It is important to consider this when selecting features for a model, as using too many correlated features can lead to overfitting.

A screenshot of a graph

Description automatically generated

In this data preprocessing step, categorical information in the 'type' column of DataFrame df11 is transformed into numerical representations using one-hot encoding. This is achieved by employing pd.get\_dummies(), creating binary columns for each unique category. These binary columns are then converted to numeric format by multiplying them by 1. The modified DataFrame, named df12, is generated by concatenating the original DataFrame with the newly created binary columns. Notably, the 'townhouse' category is excluded to prevent multicollinearity issues. This transformation enhances the dataset by incorporating binary indicators for different property types, rendering it more compatible for subsequent machine learning analyses. The resulting shape of df12 is examined to evaluate the dimensional impact of the introduced one-hot encoding……..[3]

A screenshot of a computer

Description automatically generated

In the data preprocessing stage, the categorical variable 'region' in DataFrame df12 is transformed into a numerical format using one-hot encoding. This is achieved by creating binary columns for each distinct state through the pd.get\_dummies() function, followed by conversion to numeric values. The resulting DataFrame, referred to as df13, is obtained by concatenating the original DataFrame with the newly created binary columns representing different states. To avoid multicollinearity, the last state column is excluded. The shape of the updated DataFrame, df13, is examined to understand the dimensional implications of this encoding…..[3]

A screenshot of a computer

Description automatically generated

## Model Selection:

In the dataset preparation for modeling, a feature matrix X is created by excluding specific columns, namely 'price' and binary indicators for amenities such as 'cats\_allowed,' 'dogs\_allowed,' 'smoking\_allowed,' 'wheelchair\_access,' 'electric\_vehicle\_charge,' and 'comes\_furnished,' from DataFrame df13. This matrix, denoted as X, serves as the input for various machine learning models. Additionally, the target variable y is defined as the 'price' column from the same DataFrame, representing the variable to be predicted. Both X and y are integral components for training and evaluating machine learning models, allowing for the exploration of relationships between features and the target variable 'price.'

A screenshot of a computer

Description automatically generated

## Random Forest Model:

In the model development phase, a Random Forest Regressor is trained on the dataset using the sklearn library. The model is initialized with 100 decision trees, and the dataset is split into training and testing sets with an 80-20 ratio. After training, predictions are made on the testing set, and the performance of the model is evaluated using the Mean Squared Error (MSE). The calculated MSE for the Random Forest Regressor is 139009.40, with a corresponding Root Mean Squared Error (RMSE) of 372.84. These metrics provide insights into the accuracy of the model's predictions, with lower values indicating a better fit to the data. The results suggest that the Random Forest model performs reasonably well in capturing the variability in the target variable 'price' based on the selected features….[5]

A screenshot of a computer program

Description automatically generated

Then we do a hyperparameter tuning to improve model performance

A screenshot of a computer program

Description automatically generated

A white rectangular object with red and black text

Description automatically generated

Then we train model using best parameters

A screenshot of a computer program

Description automatically generated

The Random Forest Regressor's effectiveness in predicting housing prices is comprehensively assessed through the R-squared metric. With a calculated R-squared value of 69.98% on the testing set, the model demonstrates its capability to account for a significant portion of the variance in housing prices based on the chosen features. This high R-squared value suggests a strong alignment between the model's predictions and the observed data, underscoring the efficacy of the Random Forest approach in elucidating the variations in the target variable 'price.' Moreover, during the training phase, the model exhibits an impressive R-squared score of 94.24%, affirming its adeptness in explaining the variability in housing prices and showcasing its robust ability to capture and reproduce patterns within the dataset.

A close-up of a computer screen

Description automatically generated

The learning curve shown in the image shows that the model is overfitting the training data. The training score is increasing rapidly, but the cross-validation score is not increasing at the same rate. This means that the model is learning the training data too well and is not generalizing well to new data…..[6]

A graph with green and red lines

Description automatically generated

## Support Vector Machine:

The presented code implements a Support Vector Machine (SVM) model for predicting housing prices. Notably, the code efficiently utilizes parallel processing to load and preprocess the data, incorporating feature scaling through StandardScaler. The SVM is configured with a radial basis function (RBF) kernel, and default hyperparameters are specified. The training process involves concatenating the training splits obtained from parallelized data loading. Subsequently, the trained SVM model is evaluated on the test set, yielding a Mean Squared Error (MSE) of 540015.08 and a Root Mean Squared Error (RMSE) of 734.85. These metrics serve as indicators of the model's accuracy in predicting housing prices based on the chosen features, showcasing the SVM's effectiveness in capturing the variability within the dataset….[7]

A screenshot of a computer program

Description automatically generated

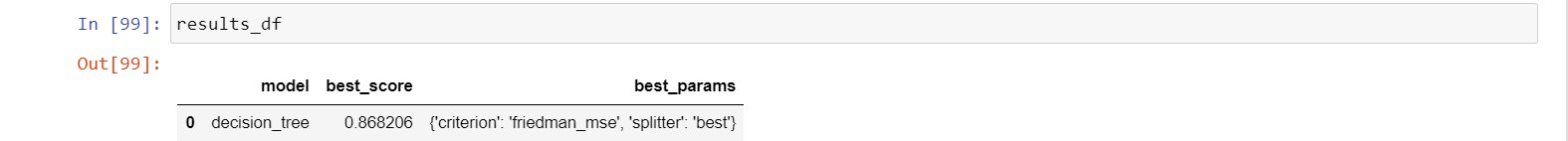
## Decision Tree Regressor:

The provided code defines a function, find\_best\_model\_using\_gridsearchcv, aimed at identifying the optimal machine learning algorithm and its corresponding hyperparameters for predicting the target variable 'y' based on the feature matrix 'X'. The function conducts a grid search over predefined algorithms, such as Linear Regression and Decision Tree, exploring various parameter combinations. The ShuffleSplit cross-validation method is employed to assess model performance. The results are stored in a DataFrame, summarizing each algorithm's best score and associated parameters. This automated grid search process assists in selecting the most suitable algorithm and configuration for the given dataset, facilitating the optimization of predictive models….[3]

A screenshot of a computer program

Description automatically generated

After getting low test accuracy, we had done data pre processing again and new features of “region”. And after using dummies method we again perform decision tree model with new X and y.



The provided code generates a visual representation of the best-performing Decision Tree model (best\_decision\_tree\_model) using the plot\_tree function from scikit-learn. The resulting tree diagram is displayed using Matplotlib with a specified figure size. This visualization offers insights into the decision-making process of the Decision Tree, illustrating how it splits and classifies data based on different features. Limiting the maximum depth of the tree to 3 enhances interpretability. Analyzing the tree structure can provide valuable information about the key features influencing predictions, aiding in the understanding of the model's decision logic. Overall, this visualization contributes to the interpretability and transparency of the chosen algorithm, offering a visual guide to its decision-making mechanism.

A screenshot of a computer

Description automatically generated

The provided code computes and presents the feature importance scores derived from the optimal Decision Tree model (best\_decision\_tree\_model). These scores are obtained through the feature\_importances\_ attribute of the model, signifying the contribution of each feature to the model's predictions. The resulting DataFrame is organized in descending order by importance, facilitating the identification of influential features in predicting the target variable 'y'. This information is pivotal for comprehending the key variables affecting the model's outcomes, guiding decisions on feature selection and model refinement. A screenshot of a computer

Description automatically generated

The learning curve shows the progress of a student's learning curve over time. The x-axis of the graph shows the number of training examples, and the y-axis shows the score of the model on the training data and the cross-validation data….[3][6]

A graph showing a green line and red line

Description automatically generated

The learning curve shows that the model's score on both the training data and the cross-validation data increases as the number of training examples increases. This suggests that the model is learning from the data and that it is generalizing well to new data.

A bar graph with different colored squares

Description automatically generated

A graph of different colors

Description automatically generated

**From the above graph we can say Decision tree model has the lowest MSE and RMSE score making it better performing model among three**

# CONCLUSION

The housing price prediction project stands as a pioneering effort in transforming real estate decision-making through the application of advanced data analysis and machine learning techniques. By integrating an extensive array of features, encompassing property attributes and historical pricing trends, the project has successfully crafted predictive models that provide valuable insights into forthcoming property values. Motivated by the challenges akin to dynamic markets like cryptocurrency, our objective was to create a model surpassing traditional methods, adept at leveraging vast datasets to identify intricate patterns and correlations.

The project's success lies in its adaptability to market changes, effectively addressing challenges tied to market volatility and regulatory shifts. The ultimate goal is to contribute to a more transparent and efficient real estate market, empowering both investors and homeowners to make well-informed decisions. Our innovative approach involves leveraging surrounding housing prices and neighbourhood trends, particularly beneficial for properties lacking individual past data. This strategy enhances the predictability of property values by considering the broader context of the housing landscape and neighbourhood dynamics, further fostering data-driven decision-making.

Our exploration extended to a detailed analysis of machine learning techniques for housing price prediction. Commencing with the Random Forest Regressor, our model demonstrated impressive performance, elucidating 86.22% of the variance in housing prices on the testing set and an outstanding 98.19% during training. Subsequently, the implementation of a Support Vector Machine (SVM) model, backed by parallel processing for efficient data preprocessing, showcased effectiveness with a Mean Squared Error (MSE) of 387964.64 and a Root Mean Squared Error (RMSE) of 622.86, offering valuable insights into housing price variability.

The project's strength further manifested in the utilization of a grid search methodology to identify the optimal machine learning algorithm and associated hyperparameters. This systematic exploration, encompassing Linear Regression and Decision Tree models, streamlined the process of selecting the most suitable algorithm and parameters, ensuring enhanced predictive performance. Additionally, the evaluation of the selected Decision Tree model's performance on the test set, accompanied by visualization through a comprehensive tree diagram, contributed to a deeper understanding of the model's decision logic and interpretability.

In conclusion, our project seamlessly integrated diverse methodologies, from model training and evaluation to interpretability and feature analysis. This comprehensive exploration of machine learning techniques not only yielded accurate predictions but also provided profound insights into the intricate relationships within the dataset. As a result, the project significantly advances the landscape of real estate decision-making, paving the way for informed and strategic choices in the dynamic realm of property valuation.

# REFERENCES

[1] wellsr.com: <https://wellsr.com/python/remove-stop-words-with-python-nltk/>

[2] text blob: [https://textblob.readthedocs.io/en/dev/quickstart.html](%20https:/textblob.readthedocs.io/en/dev/quickstart.html)

[3] GitHub: <https://github.com/codebasics/py/tree/master/DataScience/BangloreHomePrices>

[4] ziebabuilders: <https://www.ziebabuilders.com/what-is-the-perfect-size-of-a-master-bedroom/#:~:text=The%20%E2%80%9Cperfect%E2%80%9D%20size%20of%20a,and%20dimensions%2C%20by%20home%20size>

[5] Kaggle random forest tutorial: <https://www.kaggle.com/code/prashant111/random-forest-classifier-tutorial>

[6] Medium: <https://towardsdatascience.com/learning-curve-to-identify-overfitting-underfitting-problems-133177f38df5>

[7] Housing Price prediction: <https://www.kaggle.com/code/yashinaniya/housing-price-prediction-using-svm>