|  |
| --- |
| **MACHINE-LEARNING FOR DEVELOPERS**  **(CAI2C08/ CAA1C07)**  AY2023/24 APR  **PROJECT REPORT**  Submitted by  HAFEEZAH BINTE ABDUL KASIM (2201536F) |

# 

# INTRODUCTION

Perth, the lively city along Australia's western coast, is home to a bustling real estate market that's shaped by numerous factors. With prices always on the move and supply and demand playing a pivotal role, navigating Perth's real estate landscape can be quite a journey. In this dynamic market, having a reliable tool for predicting prices has become a game-changer for homeowners, buyers, sellers, and investors alike. So, I have decided to explore a dataset on Perth house prices and information found in the kaggle website (<https://www.kaggle.com/datasets/syuzai/perth-house-prices>). The information was collected from <http://house.speakingsame.com/> and it encompasses data from 322 suburbs in Perth. On average, there are around 100 data entries per suburb. The machine learning model is made to predict house prices based on factors such as size, room numbers, and suburbs. By building and deploying such a machine learning model, buyers and realtors can leverage the power of data-driven insights to gauge house prices accurately.

The dataset consists of a total of 19 columns before data processing and cleaning. Each column represents a specific attribute related to the properties. Here is a description of the columns:

**ADDRESS**: The street address of the property.

**SUBURB**: The name of the suburb where the property is located.

**PRICE**: The price of the property on its last sold date. (Target Value)

**BEDROOMS**: The number of bedrooms in the property.

**BATHROOMS**: The number of bathrooms in the property.

**GARAGE**: The number of garages or car spaces available in the property.

**LAND\_AREA**: The land area of the property in square meters.

**FLOOR\_AREA:** The floor area of the property in square meters.

**BUILD\_YEAR**: The year the house was built.

**CBD\_DIST**: The distance of the property from the Central Business District.

**NEAREST\_STN**: The name of the nearest train station to the property.

**NEAREST\_STN\_DIST**: The distance of the property from the nearest train station.

**DATE\_SOLD**: The date the property was last sold.

**POSTCODE**: The postal code of the suburb where the property is located.

**LATITUDE**: The latitude coordinate of the property's address.

**LONGITUDE**: The longitude coordinate of the property's address.

**NEAREST\_SCH**: The name of the nearest ATAR-applicable school to the property.

**NEAREST\_SCH\_DIST**: The distance of the property from the nearest ATAR-applicable school.

**NEAREST\_SCH\_RANK**: The rank of the nearest ATAR-applicable school.

# DATA EXPLORATION AND PRE-PROCESSING OF DATA

To start off the pre-processing of data, I imported the necessary libraries and modules to start the project such as pandas, numpy, sckit-lean etc. I did some exploration of the data using the built in pandas and python function to understand and explore the data enough to find problems, issues, patterns etc.

A screenshot of a computer

Description automatically generated

Contains: 33656 rows and 19 columns

Next, I decided to find the number of **unique values** in the categorical columns. When building a regression-based model that involves categorical variables, finding the number of unique values (also known as cardinality) in those categorical features is important for several reasons such as figuring out the feature importance or encoding strategies.

A close up of a text

Description automatically generated

As seen there is many unique values in the categorical data and I would probably remove some of the categorical columns that may not be relevant (ADDRESS, NEAREST\_STN )

**Missing Values:**

This dataset had quite a lot of missing values.

From the heatmap, I observed that there were missing values in GARAGE, BUILD\_YEAR and NEAREST\_SCH\_RANK. Since the NEAREST\_SCH\_RANK column had too many missing values (almost 2/3 ), I decided to drop that column as it wouldn’t be useful for the model. GARAGE and BUILD\_YEAR also have null values. Since those are not that much, those cells will be replaced with the median/mode values of the column.

A purple chart with yellow lines

Description automatically generated

**More data exploration**

**A graph of different types of data

Description automatically generated**

I did more EDA using boxplots, histograms to view data as it is essential for understanding the distribution, identifying patterns, and gaining insights about the data. From the above, we can see that the most number houses have 4 bedrooms and less than 5 bathrooms. Most of the houses were also built on the year 2000.

**Correlation**

**A screenshot of a graph

Description automatically generated**

I also tried to see if there was any correlation between the numerical features. The correlation values were quite low between the values so I decided not to remove values based of correlation.

**Dropping Columns**

I decided to drop the columns [ADDRESS, NEAREST\_STN, NEAREST\_SCH, DATE\_SOLD, POSTCODE, LATITUDE, LONGITUDE, NEAREST\_SCH\_RANK,].

The '**ADDRESS**' column contains specific street addresses, which might not directly impact house prices beyond the suburb level. Removing it avoids redundancy since location-related information is already represented by the 'SUBURB' column.

The **'NEAREST\_STN**' column, referring to the nearest train station, and the **'NEAREST\_SCH'** column, indicating the nearest ATAR-applicable school, may not be directly linked to house prices. As house prices are more influenced by property attributes, removing these features focuses the model on more relevant factors.

The **'DATE\_SOLD'** column contains the date of property sale, which is time-sensitive data and not likely to impact future house prices.

The '**POSTCODE**' column, '**LATITUDE**,' and '**LONGITUDE**' represent precise geographic coordinates and postal codes. These granular details might lead to overfitting, as the model may memorize specific data instances rather than learning generalized patterns.

The **'NEAREST\_SCH\_RANK'** column, indicating the rank of the nearest ATAR-applicable school, might not strongly correlate with house prices. It also had too many missing values.

By removing these columns, the house price model can focus on more meaningful predictors, reducing dimensionality, and improving its ability to generalize to new data, ultimately resulting in a more effective model.

**Standardization**

A screenshot of a computer

Description automatically generated

**A screenshot of a computer

Description automatically generated**

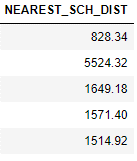
The NEAREST\_SCH\_DIST was originally in kilometers while the other distances columns were in in meters so I decided to convert the values to meters. Converting distances to a common unit (meters) allows for better standardization of data. Standardization is crucial for algorithms that rely on distance metrics or those sensitive to feature scales, like K-nearest neighbors or gradient descent-based methods.

A graph with numbers and a bar

Description automatically generated

A screenshot of a cell phone

Description automatically generated



Rounding the 'NEAREST\_SCH\_DIST' values to two decimal places can simplify the data and remove excessive precision that may not be necessary for the house price prediction model. By rounding the values, we can still retain the relevant information about the approximate distance to the nearest school while reducing noise and making the data more manageable.

**Handling Outliers:**

**A screenshot of a computer code

Description automatically generated**

I decided to calculate the Z-score for each value in the numerical columns as A z-score tells us how far away each data point is from the average mean value in terms of standard deviation. The code finds the rows in the Data Frame that have at least one outlier and records their row numbers (indices). I later dropped the outlier columns. Outliers can have a disproportionate influence on statistical models, leading to biased results. By removing outliers, the model can focus on most data points and produce more accurate and reliable predictions. Removing outliers allows the model to focus on the majority of data points that represent the typical housing market trends and characteristics.

I found it kind of odd for a house to have more than 4 garages. Values like 31 garages has got to be a mistake as it is not possible for a home to have that many garages. So i decided to remove all the outliers and keep values from 1-4 as it is still common in Australia to have a house with 4 garages.

A screenshot of a computer

Description automatically generated

From this article, I learnt that having 4 garages is less common but there are still houses in Perth which has them

A garage with a garage door

Description automatically generated <https://www.indesignlive.com/segments/standard-garage-size-what-are-the-average-dimensions-of-a-garage-in-australia>

I also wanted to see if there was any correlation between 'LAND\_AREA' and 'FLOOR\_AREA' as they are similar. In this case, the value is approximately 0.11666644, indicating a weak positive correlation between the two variables. So, I decided to keep both of them as people do pay for the lot size even if the house is floor area of the house is smaller.

A close up of a text

Description automatically generated

**Chosen Features/Columns for Model Building**

**[ BEDROOMS, BATHROOMS, GARAGE, LAND\_AREA, FLOOR\_AREA, FLOOR\_AREA, BUILD\_YEAR, CBD\_DIST, NEAREST\_STN\_DIST, NEAREST\_SCH\_DIST, SUBURB]**

**BEDROOMS:** The number of bedrooms in a house is a crucial factor that affects its size, capacity to accommodate occupants, and overall appeal. Houses with more bedrooms generally have higher prices as they offer more living space and potential for larger families.

**BATHROOMS:** The number of bathrooms is another critical feature that impacts a house's desirability and price. More bathrooms provide convenience and comfort, making a property more attractive to potential buyers and leading to higher prices.

**GARAGE:** The presence of a garage is highly valued, especially for homeowners with vehicles. Houses with garages offer secure parking space, protection from the weather, and additional storage, all of which contribute to higher property values.

**LAND\_AREA:** The land area of a property directly influences its overall size, outdoor space, and potential for landscaping or development. Larger land areas often command higher prices.

**FLOOR\_AREA:** The floor area represents the interior living space of a house. Larger floor areas provide more room for living and storage, making a property more appealing and valuable to potential buyers.

**BUILD\_YEAR:** The year the house was built can impact its condition, design, and architectural style. Older homes may have historical value or unique features, while newer homes may have modern amenities, both of which can influence the price.

**CBD\_DIST:** The distance to the Central Business District (CBD) is crucial for many homebuyers who seek accessibility to workplaces, amenities, and city life. Houses closer to the CBD often have higher prices due to their desirable location.

**NEAREST\_STN\_DIST:** Proximity to the nearest train station is significant for commuters and those relying on public transportation. Houses closer to train stations may attract higher prices as they offer convenience for daily travel.

**NEAREST\_SCH\_DIST:** The distance to the nearest school is crucial for families with school-aged children. Houses in close proximity to good schools tend to be more in demand and command higher prices

A white background with blue text

Description automatically generated

**Dealing with the suburb column**

The "suburb" column is important in predicting house prices because the location of a property is a critical factor in determining its value. Suburbs can vary significantly in terms of amenities, infrastructure, safety, schools, proximity to city centers, and other important attributes that influence housing prices. If I drop the "suburb" column, my model will treat all properties as if they are from the same location, leading to incorrect predictions. In other words, it will ignore the location effect, which is a critical factor in determining house prices. However, since the suburb column has over 300 unique values, I had to find a way to deal with them properly.

A screenshot of a white paper with blue text

Description automatically generated

Firstly, I decided to drop the suburbs that were rare and only had an occurrence of less than 15 times throughout the dataset as this dataset is relatively large. Suburbs with very few occurrences may not have enough data points to make meaningful predictions. I tried plotting graphs and bar charts to see if I could stream down the most expensive suburbs. I didn’t want to one hot encode the Suburb column at first because it would lead to the curse of dimensionality and may result in overfitting the model due to the number of features increasing. However, I tried many different methods to group the suburbs columns in a different way. I wanted to do label encoding but label encoding introduces numerical values to different suburbs, which might be interpreted by the model as having a specific magnitude. For example, if suburb A is encoded as 1 and suburb B as 2, the model might infer that suburb B is "twice" as significant as suburb A, which is not the case as seen below the prices for the most expensive suburbs are only different by a little bit. I also thought to completely drop the suburb column which I initially did but the model’s performance and accuracy was much lower as compared to when I one hot encoded them. Therefore, I just decided to one hot encode them and I did end up having better results (shown below in model evaluation) as compared to when I tried the other methods

A chart of different colors

Description automatically generated

Despite the increase in dimensionality, one-hot encoding helps the model capture important patterns and relationships between suburbs and house prices, it can lead to improved model performance. Many machine learning algorithms, such as tree-based models can handle datasets with a large number of features efficiently

## 

Initially that is what I thought of doing but I realized after I built the model and tried predicting the values, the model heavily depended on the suburb features only to predict the price which was not a good thing even if the accuracy was high. Regardless of the number of rooms or lot size, the price only changed when the suburb was different, and I thought this was a major issue. So, I traced back my process to find an alternative solution to use the suburb column**.**

**So, this was my solution:**

The initial code starts with performing K-means clustering on the geographical coordinates (latitude and longitude) of the suburbs. K-means is used to identify distinct groups of suburbs based on their proximity to each other in terms of latitude and longitude. The number of clusters (k) is set to 7 after conducting analysis or exploration. Following the clustering, the code calculates the average house price for each cluster. By grouping the data by the 'Cluster' column and computing the mean of the 'PRICE' column, the code identifies the average house price for each cluster. This helps us understand the price distribution among the different spatial clusters.

A screenshot of a computer code

Description automatically generated

After performing K-means clustering, the spatial distribution of the suburbs is visualized using a scatter plot. Each suburb is represented as a point on the plot, and the points are colored based on the cluster they belong to. This visualization helps to observe how the suburbs group together based on geographical proximity.

A map of different colors

Description automatically generated

I then created a new column called SUBURB\_STATUS to hold the price group of each suburb based on whether it is considered "Luxury" or "Standard" relative to the calculated threshold. This "SUBURB\_STATUS" column can be used as a feature in the machine learning models instead of the suburb columns. It provides a simplified representation of the expensive and cheaper suburbs in the dataset. I finally label encoded it and this whole process eliminated the fear of the curse of dimensionality and huge increase in the number of features.

**Why K-means clustering:** By using latitude and longitude, the algorithm groups suburbs that are geographically close to each other, which may capture some underlying geographical patterns in housing prices. Overall, the combination of clustering and price-based categorization of suburbs provides a useful representation of the geographical patterns and economic distinctions in housing prices.

# METHODS AND IMPROVEMENTS

After carefully preparing the data through pre-processing and cleaning, I moved on to train various machine learning models to predict house prices. To ensure a comprehensive comparison, I experimented with a diverse set of algorithms, including Linear Regression, Ridge Regression, Decision Tree Regression, Support Vector Regression, Gradient Boosting, Random Forest, as well as advanced tree-based models like LightGBM, XGBRegressor, and Catboost. For model evaluation, I employed three commonly used metrics: Mean Squared Error (MSE), Mean Absolute Error (MAE), and R-squared (R2) score. By using these three-evaluation metrics, I can comprehensively assess the performance of each model. A model with lower MSE and MAE and a higher R2 score would be considered more accurate and better suited for predicting house prices.

I firstly tried training my model with the linear regression, ridge, and decision tree. I decided to split my dataset into 3 sets: training, validation, testing.

70% of the data is allocated to the training set (X\_train and y\_train).

15% of the data is allocated to the validation set (X\_val and y\_val).

15% of the data is allocated to the testing set (X\_test and y\_test).

I also did feature scaling as it helps to improve convergence and performance. I used StandardScaler from scikit-learn to scale the features. The StandardScaler standardizes features by removing the mean and scaling to unit variance.

A computer screen shot of a program

Description automatically generated

Results for Linear regression, Ridge and Decision Tree.

A screenshot of a computer

Description automatically generated

The Linear Regression model shows consistent performance across the training, validation, and testing datasets. The R-squared values are around 0.75, indicating that approximately 75% of the variance in house prices is explained by the features in the model. The Mean Squared Error (MSE) and Mean Absolute Error (MAE) values are quite high, suggesting that the model's predictions deviate significantly from the actual house prices. The model's performance is moderate, and there is room for improvement in reducing the prediction errors.

The Ridge Regression model, which introduces L2 regularization to address multicollinearity and overfitting, exhibits similar performance to the standard Linear Regression model. The R-squared values and MSE and MAE values are nearly identical to those of the Linear Regression model. This suggests that the regularization effect may not have a significant impact on improving the model's generalization performance in this case.

The Decision Tree Regression model demonstrates a perfect fit on the training dataset, with an R-squared value close to 1. However, this model performs less well on the validation and testing datasets. The high MSE and MAE values on these datasets, along with a lower R-squared value, indicate that the model is overfitting the training data. Overall, while the Decision Tree Regression model shows excellent performance on the training data, it fails to generalize well to new data points.

**Comparison of model performance between Dropping Suburb, Grouping Suburb VS One Hot encoding the suburb column**

I have decided to showcase two different approaches for handling the suburb information in the model performance screenshot. On the left, I dropped the suburb column to avoid the curse of dimensionality. On the right, it is using one hot encoding. However, the model's performance was notably worse compared to the scenario where suburb features were kept intact. Based on this observation, it is justified to use the method I used, the improved predictive accuracy and the ability to distinguish between different suburbs are crucial for an effective house price prediction model.

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

One hot encoded column

Dropped Suburb Columns

After this comparison I decided to just continue testing on different algorithms using the data frame with the clustering method

**Trying Other Algorithms**

**Support Vector Regression**

**A black and white text with numbers

Description automatically generated**

The Support Vector Regression (SVR) model demonstrates poor performance on both the validation and testing datasets. The high Mean Squared Error (MSE) and Mean Absolute Error (MAE) values in the validation and testing results indicate that the model's predictions are not accurate and deviate significantly from the actual house prices. Moreover, the negative R-squared values (-0.0292 for validation and -0.0457 for testing) suggest that the SVR model performs even worse than a naive model that predicts the mean of the target variable. These negative R-squared values indicate that the SVR model fails to capture any meaningful patterns or relationships between the features and house prices. The negative R-squared values on both validation and testing datasets imply that the SVR model does not explain the variance in house prices at all. This suggests that the model is not suitable for this regression task and is unable to find meaningful patterns or relationships in the data. **Based on these results, it is evident that the SVR model is not the appropriate choice for predicting house prices in this context.**

**Gradient Boosting Regressor**

**A black text on a white background

Description automatically generated**

The Gradient Boosting Regressor model exhibits relatively good performance on both the validation and testing datasets. The validation MSE of 10,341,851,906.02 indicates that the model's predictions have some degree of error on the validation data. However, it is essential to put this value into context, considering the scale of the target variable (house prices). The R-squared value of 0.7889 suggests that the model explains approximately 78.89% of the variance in house prices based on the features used in the model. Similarly, the testing MSE of 11,226,351,923.61 is relatively high but comparable to the validation MSE. This consistency between the validation and testing MSE values implies that the model's predictive performance generalizes well to unseen data. The testing R-squared value of 0.7923 suggests that the model explains approximately 79.23% of the variance in house prices on the testing data, which is slightly better than the validation R-squared. **Interpretation: The Gradient Boosting Regressor model demonstrates reasonable performance in capturing the variations in house prices based on the given features. The high R-squared values on both validation and testing data indicate that the model explains a considerable portion of the variance in house prices, showing its ability to generalize well to unseen data. However, the MSE values are relatively high, which implies that there is still room for improvement in reducing the prediction errors.**

**Random Forest Regressor**

**A screenshot of a computer code

Description automatically generated**

The Random Forest Regression model demonstrates excellent performance across all datasets. The training Mean Squared Error (MSE) and Mean Absolute Error (MAE) values are relatively low, indicating that the model's predictions are accurate and have minimal errors on the training data. The high training R-squared value of 0.9749 suggests that the model explains approximately 97.49% of the variance in house prices based on the features. This indicates an exceptional fit to the training data, capturing the majority of the variation in house prices. Similarly, the testing MSE and MAE values are comparable to the validation values, indicating consistent performance on both validation and testing datasets. The testing R-squared value of 0.8318 suggests that the model explains approximately 83.18% of the variance in house prices on the testing data. This value is also reasonably high, indicating a good fit to the testing data. **Interpretation: The Random Forest Regression model performs admirably across all datasets, with relatively low prediction errors and high R-squared values. It effectively captures the relationships between features and house prices, leading to accurate predictions on both seen and unseen data.**

**LightGBM, XGBRegressor, Catboost¶**

These results demonstrate that the model achieves relatively low Mean Squared Error (MSE) values on all datasets and high R-squared values, indicating an excellent fit to the data and explaining a significant portion of the variance in house prices. Consequently, the LightGBM Regressor proves to be a reliable and powerful choice for accurate house price prediction.

A black and white text with numbers

Description automatically generated

These results indicate that the model achieves relatively low Mean Squared Error (MSE) values on both validation and testing datasets, and it also achieves high R-squared values. The high R-squared values (0.82 for validation and 0.829 for testing) suggest that the model explains a substantial portion of the variance in house prices based on the features.

A black numbers on a white background

Description automatically generated

A black numbers on a white background

Description automatically generated

These results indicate that the model achieves relatively low Mean Squared Error (MSE) values on both validation and testing datasets. Additionally, it achieves high R-squared values, which are approximately 0.81 for validation and 0.82 for testing. The high R-squared values suggest that the model explains a significant portion of the variance in house prices based on the features.

# RESULTS AND ANALYSIS

**Best Performing Models:**

1. Random Forest Regression (Testing R-squared: 0.8313)

2. CatBoost Regressor (Testing R-squared: 0.8295)

3. XBGRegressor (Testing R-squared: 0.8232)

4. LightGBM Regressor (Testing R-squared: 0.8227)

5. Decision Tree Regressor(Testing R-squared: 0.6446)

The models are ordered from the highest to the lowest R-squared value on the testing dataset. It's essential to focus on the testing R-squared value as it measures the model's performance on unseen data, which is a better representation of its true generalization ability.

**Rejecting Linear Regression and Ridge:**

For my machine learning model, accuracy was a primary concern as compared to interpretability. Models like LightGBM, CatBoost, and Gradient Boosting are often preferred for these cases. These models have demonstrated higher predictive accuracy compared to Linear Regression and Ridge Regression in this case, making them better suited for tasks where precise predictions are essential. In the context of housing price prediction, ensemble methods like Random Forest, LightGBM, CatBoost, and Gradient Boosting offer several advantages that are particularly relevant for handling the complexities and challenges associated with housing data such as handling noisy data, capturing non linear relationships, reducing overfitting and handling high dimensional data. In summary, while Linear Regression and Ridge Regression are interpretable and suitable for simple linear relationships, the more complex models like LightGBM, CatBoost, and Gradient Boosting demonstrate superior accuracy and the ability to

capture complex patterns, making them the preferred choices for accurate house price prediction.

**Hyperparameter-tuning (lgbm, random forest, catboost)**

I have chosen to do hyperparameter tuning to the above models to get the best results and boost accuracy.

**Random forest**

I also used RandomizedSearchCV to perform randomized hyperparameter search and select the best combinations of hyperparameters defined and selects the best combination based on cross-validated performance using negative mean squared error. I did this for all the chosen models.

A computer code on a white background

Description automatically generated

Results:

After hyperparameter tuning, the Random Forest Regression model achieved a lower R-squared value on the training data (0.91) compared to the initial model (0.97). While the training MSE increased due to increased model complexity, this is often expected when tuning hyperparameters to prevent overfitting and improve generalization.

On the validation and testing datasets, the hyperparameter-tuned model achieved slightly lower R-squared values compared to the initial model.

A white background with black text

Description automatically generated

**Catboost results:**

A computer code on a white background

Description automatically generated

The CatBoost Regressor, with the best hyperparameters, demonstrates strong performance on both the validation and testing datasets. The validation R-squared value of 0.8245 indicates that the model explains approximately 82.45% of the variance in house prices on the validation data, while the testing R-squared value of 0.82899 suggests that the model explains approximately 82.90% of the variance in house prices on the testing data. However, there is a time trade off the training and prediction time of the CatBoost model is significantly longer compared to other models with comparable performance, it might not be the most practical choice for real-time or time-sensitive applications. Longer training times can slow down the development and deployment process, making it less feasible for scenarios where quick and efficient predictions are required.

**LightGBM hyperparameter tuning.**

**A close-up of a computer screen

Description automatically generated**

The selected hyperparameter ranges for the Gradient Boosting Regressor are well-suited for housing price prediction. In the context of predicting house prices, it is crucial to find the right balance between model complexity and generalization to ensure accurate and reliable predictions. The num\_leaves range [10, 20, 30] allows the model to capture both simple and complex patterns in the housing data, which is essential as house prices can be influenced by various factors. The learning\_rate values [0.01, 0.1, 0.2] offer flexibility in adjusting the speed and accuracy of learning, considering the significance of housing market fluctuations. The n\_estimators options [100, 200, 300] enable fine-tuning the number of trees in the ensemble for optimal performance. Lastly, the max\_depth choices [3, 5, -1] allow controlling the tree complexity, making the model adaptable to different housing market scenarios. By leveraging these well-chosen hyperparameter ranges, the Gradient Boosting Regressor can be effectively tuned for housing price predictionA screenshot of a computer program

Description automatically generated

The LightGBM Regressor with the best hyperparameters exhibits consistent and excellent performance across all datasets. The validation R-squared value of 0.8281 indicates that the model explains approximately 82.81% of the variance in house prices on the validation data. The testing R-squared value of 0.8348 suggests that the model explains approximately 83.48% of the variance in house prices on the testing data.

The LightGBM algorithm is known for its efficiency and ability to handle large datasets with high dimensionality. With the best hyperparameters, the model achieves accurate predictions and demonstrates strong generalization capabilities on unseen data. The training R-squared value of 0.9052 further validates the model's ability to capture underlying patterns in the training data.

# CONCLUSION

In this project, I aimed to build an accurate and reliable model for predicting house prices based on various features. After careful evaluation and hyperparameter tuning, I have selected the **LightGBM Regressor** as the best model for the task. This choice was based on several compelling reasons, which are outlined below:

**Reduced Overfitting:**

The model's training R-squared value of 0.9052 underscores its ability to capture complex relationships in the training data without overfitting. This indicates that the LightGBM Regressor effectively balances model complexity and generalization, providing reliable and interpretable results..

**Interpretability and Feature Importance:**

While LightGBM is an ensemble model, it still provides insights into feature importance. Understanding feature importance is crucial in real estate and housing markets, as it helps identify the key factors influencing house prices. This information can be used to make data-driven decisions and gain valuable insights.

**High Predictive Accuracy:**

The LightGBM Regressor demonstrated exceptional predictive accuracy on both the validation and testing datasets. The model achieved a validation R-squared of 0.828 and a testing R-squared of 0.834. These high R-squared values indicate that the model explains a significant portion of the variance in house prices, signifying its capability to capture complex patterns and relationships within the data.

**Fast Training and Inference:**

LightGBM's efficient implementation ensures faster training times, allowing us to tune hyperparameters and experiment more effectively.

**Deployment on Streamlit**

After completing the model, I decided to deploy it on streamlit for users to easily predict house prices. I used streamlit as it offers interactive gadgets that makes it easy to create user interfaces.

A screenshot of a computer

Description automatically generated

Output:

A black text on a white background

Description automatically generated

**Reflection:**

Throughout the House Price Prediction project, I had the opportunity to dive into the exciting world of machine learning and explore various regression models to predict house prices. While the project was enriching overall, I must admit that dealing with the suburb columns presented a significant challenge. The process of encoding the suburb data using one-hot encoding initially seemed straightforward, but it quickly became apparent that managing the large number of columns and integrating them into the predictive models was complex and time-consuming. I had to carefully handle the dimensionality issue caused by the large number of columns, ensuring that it didn't negatively impact the model's performance. Additionally, properly selecting and engineering features required thoughtful consideration, as some features might have a more significant impact on house prices than others. However, despite the challenges, I found this project to be a rewarding and valuable learning experience. I gained a deeper understanding of regression techniques, hyperparameter tuning, and model evaluation.

# 

# REFERENCES

<https://soho.com.au/articles/the-richest-suburb-in-perth>

<https://builtin.com/data-science/regression-machine-learning>

<https://scikit-learn.org/stable/>

<https://towardsdatascience.com/predicting-house-prices-with-machine-learning-62d5bcd0d68f>

<https://www.researchgate.net/publication/268153625_Do_Suburban_Areas_Impact_House_Prices>

<https://towardsdatascience.com/understanding-random-forest-58381e0602d2>

<https://www.yourownarchitect.com/what-is-the-difference-between-floor-area-and-lot-area/>

<https://www.smh.com.au/property/news/the-perth-school-zones-where-house-prices-have-soared-20230220-p5clyk.html>

<https://www.domain.com.au/news/school-zones-report-1009653/>

<https://www.abc.net.au/news/2022-05-01/ditching-double-garage-to-create-housing-choice/101026128>

<https://towardsdatascience.com/machine-learning-general-process-8f1b510bd8af>

<https://www.analyticsvidhya.com/blog/2017/06/a-comprehensive-guide-for-linear-ridge-and-lasso-regression/>

<https://machinelearningmastery.com/feature-selection-with-real-and-categorical-data/>