



Module Code & Module Title
CC6057NI Applied Machine Learning

Level 6 – Artificial Intelligence

Assessment Type

Individual 60%

Semester

2024/25 Autumn

Student Name: Rajita Maharjan

London Met ID: 22067335

College ID: np01ai4a220052

Assignment Due Date: Thursday, January 16, 2025

Assignment Submission Date: Thursday, January 16, 2025

Submitted To: Mahotsav Bhattarai

Word Count (Where Required): 2964

I confirm that I understand my coursework needs to be submitted online via MST Classroom under the relevant module page before the deadline in order for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a mark of zero will be awarded.

Document 1.docx

 Islington College, Nepal

Document Details

Submission ID

trn:oid::3618:79325212

Submission Date

Jan 16, 2025, 12:07 PM GMT+5:45

Download Date

Jan 16, 2025, 12:15 PM GMT+5:45

File Name

Document 1.docx

File Size

15.2 KB

16 Pages

2,329 Words

12,729 Characters



Page 1 of 20 - Cover Page

Submission ID trn:oid::3618:79325212







Page 2 of 20 - Integrity Overview

Submission ID trn:oid::3618:79325212




27% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Match Groups

-  **35 Not Cited or Quoted 24%**
Matches with neither in-text citation nor quotation marks
-  **2 Missing Quotations 3%**
Matches that are still very similar to source material
-  **0 Missing Citation 0%**
Matches that have quotation marks, but no in-text citation
-  **0 Cited and Quoted 0%**
Matches with in-text citation present, but no quotation marks

Top Sources

- 5%  Internet sources
- 2%  Publications
- 26%  Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.



Match Groups

- 35 Not Cited or Quoted 24%
Matches with neither in-text citation nor quotation marks
- 2 Missing Quotations 3%
Matches that are still very similar to source material
- 0 Missing Citation 0%
Matches that have quotation marks, but no in-text citation
- 0 Cited and Quoted 0%
Matches with in-text citation present, but no quotation marks

Top Sources

- 5% Internet sources
- 2% Publications
- 26% Submitted works (Student Papers)

Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	Submitted works	islingtoncollege on 2024-12-24	20%
2	Submitted works	islingtoncollege on 2024-12-24	1%
3	Submitted works	islingtoncollege on 2024-12-25	<1%
4	Internet	www.mdpi.com	<1%
5	Internet	research-api.cbs.dk	<1%
6	Submitted works	islingtoncollege on 2024-12-24	<1%
7	Submitted works	islingtoncollege on 2024-12-24	<1%
8	Submitted works	University of Technology, Sydney on 2003-06-16	<1%
9	Submitted works	CSU, San Francisco State University on 2020-12-17	<1%
10	Submitted works	University of Macau on 2024-05-13	<1%



11	Submitted works	University of Sydney on 2023-05-26	<1%
12	Submitted works	De Montfort University on 2023-05-01	<1%
13	Submitted works	University of Huddersfield on 2025-01-10	<1%

Table of Contents

1. Introduction.....	1
1.1. Introduction to House Price Prediction Model.....	1
1.2. Goals of Predicting Houses Price Using Machine Learning	1
1.3. Usefulness of the House Price Prediction Model.....	1
2. Problem Domain	2
2.1. Importance of Predicting House Prices.....	2
2.3. Background Research.....	3
3. Solution	4
3.1. Comprehensive Explanation of Solution.....	4
3.2. Pseudocode.....	5
3.3. Machine Learning Algorithms Used	6
3.3.1. Linear Regression	6
3.3.2. Mean Squared Error	7
3.3.3. Decision Tree	7
3.4. Diagram.....	7
3.4.1. Flowchart	7
3.5. Development Process.....	8
3.5.1. Tools used	8
3.6. Explanation of Development Process	10
3.6.1. Data Understanding.....	10
3.6.2. Data Preprocessing	11
3.6.3. Data Preparation.....	11
3.6.4. Data Visualization	12
3.6.5. Data Modeling, Initializing, Training, and Testing the Machine Learning Models.....	22
4. Results	26
4.1. Linear Regression Model.....	26
4.2. Decision Tree Regressor	27
5. Conclusion	28
5.1. Analysis of the Work Done.....	28
5.2. Addressing Real-World Problems.....	29
6. Bibliography	29

Table of Figures

<i>Figure 1: Flowchart.....</i>	<i>8</i>
<i>Figure 2: Python</i>	<i>9</i>
<i>Figure 3: Jupyter</i>	<i>9</i>
<i>Figure 7: Displaying the first few rows of the dataset.....</i>	<i>10</i>
<i>Figure 8: Dropping rows with missing values</i>	<i>11</i>
<i>Figure 9: data preparation</i>	<i>11</i>
<i>Figure 10: Heatmap.....</i>	<i>12</i>
<i>Figure 11: Predicted price vs Actual price.....</i>	<i>13</i>
<i>Figure 12: Prediction price vs actual price (Tuned decision Tree)</i>	<i>14</i>
<i>Figure 13: Predicted vs Actual Price(Random Forest)</i>	<i>15</i>
<i>Figure 14: Model Performance comparision (RMSE).....</i>	<i>16</i>
<i>Figure 15: Model Performance Comparision(R^2 Score).....</i>	<i>17</i>
<i>Figure 16: Distribution of house prices.....</i>	<i>18</i>
<i>Figure 17: Count of semi furnished status</i>	<i>19</i>
<i>Figure 18: BoxPlot</i>	<i>20</i>
<i>Figure 19.....</i>	<i>21</i>
<i>Figure 20: Training Linear model.....</i>	<i>22</i>
<i>Figure 21: Training models on training data.....</i>	<i>22</i>
<i>Figure 22: Initializing and training the nodels</i>	<i>22</i>
<i>Figure 23: Train Test and split linear regression.....</i>	<i>23</i>
<i>Figure 24: R^2 for each model</i>	<i>23</i>
<i>Figure 25: evaluation metrics.....</i>	<i>24</i>
<i>Figure 26 : Covert to percentage and print the accuracy.....</i>	<i>24</i>
<i>Figure 27: Plotting model performance metrics.....</i>	<i>25</i>
<i>Figure 4: Training Linear regression Model</i>	<i>26</i>
<i>Figure 5: Training Decision Tree Regressor.....</i>	<i>27</i>
<i>Figure 6: Comparing the performance of the models.....</i>	<i>28</i>

1. Introduction

1.1. Introduction to House Price Prediction Model

The aim of the project is house price prediction using machine learning techniques. House prices are influenced by different features such as location, size, age, and useful features of the house. It offers a precise price range of the house that could guide stakeholders in the real estate market.

1.2. Goals of Predicting Houses Price Using Machine Learning

- **Predict House Prices:** Build a model which estimates property values based on some features of the house.
- **Machine learning algorithms:** Implementation of at least two different predictive models to compare performances.
- **Evaluate Performance:** Test model accuracy using metrics like Mean Squared Error and Root Mean Squared Error.
- **Make Continuous Improvement:** Refining models' overtime with new data to enhance prediction accuracy and adapt to market changes.

1.3. Usefulness of the House Price Prediction Model

- **Make informed decisions:** Prices estimation therefore assists buyers, sellers, and real estate agents in making wise choices in property transactions.
- **Market Analysis:** Predicting house price is very helpful in getting a better understanding of the dynamic mortgage trends; therefore, it will realize more worthwhile investments plans.
- **Reduced Risks:** Correct predictions would help in delimiting the risks of both the investors and homeowners against possible risks, which can further add to more safety in their financial planning.
- **Better Visibility:** This added feature would enable to rates based on data and, therefore, improve open transparency within the real estate market, hence establishing confidence amongst all.

2. Problem Domain

2.1. Importance of Predicting House Prices

Predicting house prices is important because:

- Proper house price prediction goes a long way in helping buyers and sellers to acquire properties at their real value.
- It can reveal profitable opportunities for investors to identify and select those opportunities that benefit them most.
- Policymakers can devise very reliable regulations and rules in housing through the use of accurate predictions of the price for any given house, consequently getting to understand better the housing market trends.
- Agents are able to give better advice to clients on buying and selling property regarding the market trends based on predictions.

There are a lot of benefits for several other people that can be realized by predicting house prices. This means that buyers will pay fair prices and will not overpay for the properties. On the other hand, sellers will be in a good position to set a competitive price that will edge others out in making sales faster. Investors can learn about areas that have a potential for growth through price predictions and hence make wiser investment decisions. The prediction is also beneficial for financial institutions since it will be able to assess the risk involved in mortgage lending in a particular fine way. Real estate agents will be more informed by these predictions so that they guide clients on the best time to buy or sell based on market trends. This results in a more open and well-managed housing market in general.

2.2. Challenges in Predicting House Prices

- House prices can often change due to the economy, interest rates, or just general opinion about a market.
- Dirty or incomplete data is one of the major reasons for inaccurate model output.
- Things like natural disasters or new laws can suddenly change the house prices.
- Prices can vary a lot from one place to another so one model may not be for every area.
- The predictions depend on how good the machine learning model and its design are.

2.3. Background Research

The dataset to be used for the purpose of predicting house prices is available on Kaggle, one of the most popular platforms known for data science and machine learning projects (H, 2021) .

Like, this dataset is largely used for learning and practising machine learning algorithms. In parts, the dataset has the following attributes:

This dataset includes various features such as:

- **Number of Rooms:** This means how many rooms are there in a house. If they are more, then the value obviously goes high.
- **Location:** The location of the house always matters. Good neighbourhoods and proximity to schools and shops usually mean a high value.
- **Square footage:** It means the total area of the house in square feet. Again, the price of a larger house would be greater.
- **Year built** is the year in which the property is built. Newer houses might fetch higher prices, as logically, a property that had fewer years on it would command more maintenance by implication.
- **Lot size:** this measures the yard the house is built on; larger lots raise the value of the property.
- **Property Type:** This defines the type of house it is, whether a detached single-family home, townhouse, condo, etc. Different classes of property would be different in cost.
- **Bathrooms:** Just like in the case of rooms, more bathrooms can make the house more desirable; hence, it will probably fetch a higher price.
- **Garage Size:** If the house comes with a garage, the size of the latter impacts consumer prices.
- **Condition of the Property:** This shows the level of maintenance that has been maintained for the house. The best maintenance will bring a higher price
- **Heating and Cooling Systems:** The house price might be attributed to the type of installed heating and air conditioning systems.

- **Amenities:** These are assets around the house, such as parks, schools, shopping malls they might attract buyers to buy houses that are put near them.

Basically, this dataset is very informative in regard to houses and can be used in the prediction of their price. Such a dataset could help to reveal what actually raises the value of a house in terms of features like the number of rooms and location, among other things. This would be very vital information to whoever has interest in acquiring or selling houses.

3. Solution

3.1. Comprehensive Explanation of Solution

- **Data Collection:** Gathering a dataset containing information on houses in terms of size and number of rooms, their location, as well as other features.
- **Data Preprocessing:**
 - Fill in or eliminate missing or inaccurate information for correctness.
 - Changing non-numeric data like yes or no into numbers so that the model can understand it.
 - Rescale numerical values into equal scales so as to make it easy for the model to learn effectively.
- **Feature Selection:** This step tries to identify those house features that can have a most probable effect upon the price, thus improving the model prediction.
- **Model Selection:** This will be implemented with the right machine learning algorithms in order to predict the price of houses. Examples include Linear Regression and decision trees.
- **Model training:** Splitting data into train and test sets; application of the chosen model to train by adjusting parameters for best results with training data.
- **Model Evaluation:** Testing is how models are good in prediction of prices with testing data and checking Mean Absolute Error(MAE) sort of measures for accuracy to compare with other models.

- **Model Deployment:** Run the best model in making predictions for house prices on new data and give that estimation further to buyers, sellers, and real estate agents in making some strategic decisions.
- **Model Maintenance:** Keeping the model up-to-date constantly with new data to remain relevant and maintain accurate predictions as changes occur in the market.

3.2. Pseudocode

START

IMPORT necessary libraries (pandas, numpy, sklearn, matplotlib, seaborn)

LOAD dataset (house price dataset)

CLEAN dataset

REMOVE rows with missing target variable (price)

HANDLE missing values for other columns

REMOVE duplicate entries

PRE-PROCESS data

ENCODE categorical variables

SCALE numerical features

SPLIT data into features (X) and target (y)

X = all columns except 'price'

y = 'price' column

SPLIT data into training and testing sets

CREATE models

Decision Tree Regressor

Random Forest Regressor

Linear Regression

TRAIN models using training data

EVALUATE models on test data

For each model:

Predict on test data

Calculate Mean Squared Error (**MSE**) and **R² Score**

COMPARE model performance based on MSE and R² score

SELECT the best model with highest R² score

OUTPUT the best model's performance (MSE and R² score)

PLOT model performance comparison (Bar chart or table)

END

3.3. Machine Learning Algorithms Used

3.3.1. Linear Regression

It is a method that assumes a between the dependent variable and one or more independent variables.

How this is used in the project:

- It is going to figure out how features of a house, like area, bedrooms, parking, and more, relate to its price.
- It then defines the weights of the features, respectively, to predict prices.

3.3.2. Mean Squared Error

It is the average of the sum of the squared difference between actual values and the predicted values; thus, it measures closeness of the predictions to the actual data.

How it is used in the project:

- Evaluates the performance of prediction models like linear regression and decision trees.
- Formula: $MSE = (1/n) \times \Sigma(\text{Actual} - \text{Predicted})^2$
- Lower MSE values means better predictions.

3.3.3. Decision Tree

A decision tree is a predictive model which has data divided into branches based on feature values of cases which can make it an important part for decision making or case prediction.

How it is used in the project:

- It splits the data based on some conditions
- Builds a tree structure with each branch representing one decision made on some feature.
- It then makes predictions for the prices following the tree from the root to a leaf node, which is the final prediction.
- Handles complex patterns and non-linear relationships within the dataset.

3.4. Diagram

3.4.1. Flowchart

A diagram that shows a process, system or computer algorithm used in various fields to document, study, plan, improve and communicate complex processes in clear and easy to understand diagrams. (What is a Flowchart, n.d.)

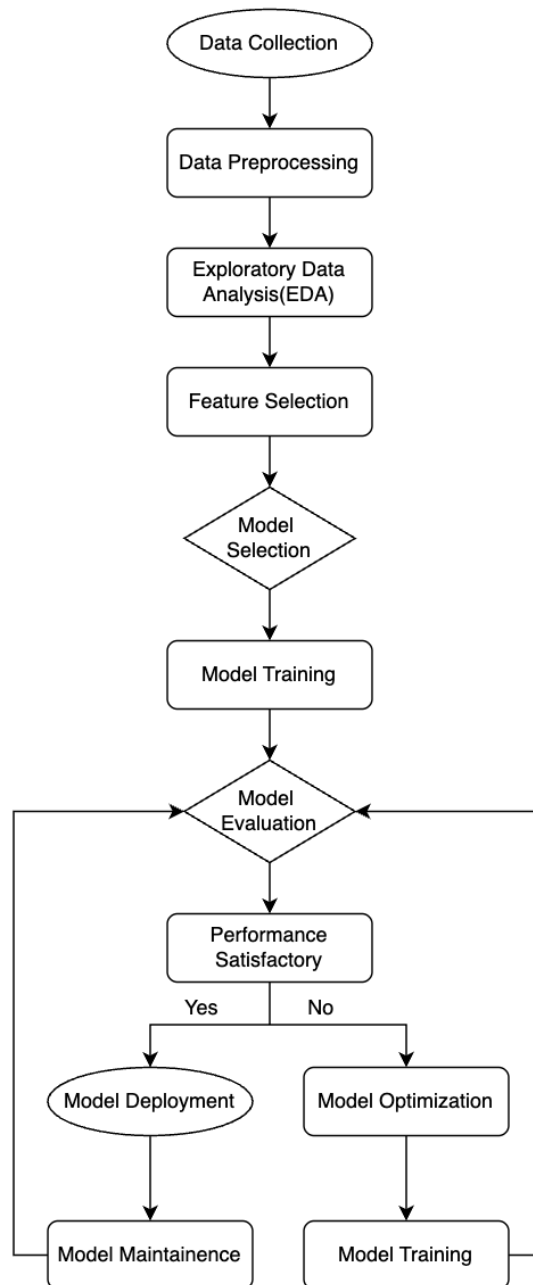


Figure 1: Flowchart

3.5. Development Process

3.5.1. Tools used

- **Python:** Python is a very high-level, object-oriented, interpreted programming language whose design aid prominently in the construction of various applications. Python is a

very flexible language, providing the capability to build websites, software, do task automation, and data analysis (Staff, 2024).



Figure 2: Python

- **Jupyter Notebook:** Jupyter Notebook is an open-source web application mainly meant for creating and sharing documents that contain live code, equations, and visualizations (jupyter, 2024).



Figure 3: Jupyter

- **Pandas:** Pandas is an important Python library specifically developed for working with data sets. Some of this includes the functions provided for analysis, cleaning, exploring, data manipulation to perform operations on data (Pandas Introduction, 2024).

- **NumPy:** NumPy is the most easy-to-use source library for Python designed to provide a solution for multidimensional array objects and several other derived objects with support for routines. It supports various mathematical functions like logical, shape manipulation, sorting, selecting handling input/output, and important linear algebraic features (What is NumPy?, 2024).
- **Matplotlib:** It is one of the most powerful and incredibly versatile plotting open-source libraries for Python, where any user can create visualization in multiple forms of data. It allows graphically representing the data to analyze and understand in ways whereby it may be simpler (Introduction to Matplotlib, 2024).
- **Seaborn:** Seaborn, another Python data visualization library, is built on top of matplotlib and features a high-level interface for drawing very attractive and informative statistical graphics (seaborn: statistical data visualization, 2024).

3.6. Explanation of Development Process

3.6.1. Data Understanding

```
# Displaying the first few rows of the dataset
df.head()
```

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airconditioning	parking	prefarea	furnishingst
0	13300000	7420	4	2	3	1	0	0	0	1	2	1	
1	12250000	8960	4	4	4	1	0	0	0	1	3	0	
2	12250000	9960	3	2	2	1	0	1	0	0	2	1	
3	12215000	7500	4	2	2	1	0	1	0	1	3	1	
4	11410000	7420	4	1	2	1	1	1	0	1	2	0	

```
df.columns
```

```
Index(['price', 'area', 'bedrooms', 'bathrooms', 'stories', 'mainroad',
       'guestroom', 'basement', 'hotwaterheating', 'airconditioning',
       'parking', 'prefarea', 'furnishingstatus_semi-furnished',
       'furnishingstatus_unfurnished'],
      dtype='object')
```

Figure 4: Displaying the first few rows of the dataset

3.6.2. Data Preprocessing

```
# Dropping rows with missing values
df.dropna(inplace=True)
```

```
df.head()
```

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airconditioning	parking	prefarea	furnishingst
0	13300000	7420	4	2	3	yes	no	no	no	yes	2	yes	furnis
1	12250000	8960	4	4	4	yes	no	no	no	yes	3	no	furnis
2	12250000	9960	3	2	2	yes	no	yes	no	no	2	yes	semi-furnis
3	12215000	7500	4	2	2	yes	no	yes	no	yes	3	yes	furnis
4	11410000	7420	4	1	2	yes	yes	yes	no	yes	2	no	furnis

Figure 5: Dropping rows with missing values

3.6.3. Data Preparation

```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
#display the shapes of the splits
X_train.shape, y_train.shape, X_test.shape, y_test.shape
```

```
((436, 13), (436,), (109, 13), (109,))
```

```
# Standardize the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

Figure 6: data preparation

3.6.4. Data Visualization

Visualisation

```

:
# Calculate the correlation matrix
correlation_matrix = df.corr()

# Plot the heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, linewidths=0.5)
plt.title("Correlation Matrix")
plt.show()

```



Figure 7: Heatmap

```
# Plot predicted vs actual prices for Linear Regression
plt.figure(figsize=(10, 6))
plt.scatter(y_test, predictions_linear, alpha=0.5, color='b', label='Linear Regression')
plt.xlabel('Actual Prices')
plt.ylabel('Predicted Prices')
plt.title('Predicted vs Actual Prices (Linear Regression)')
plt.legend()
plt.show()
```

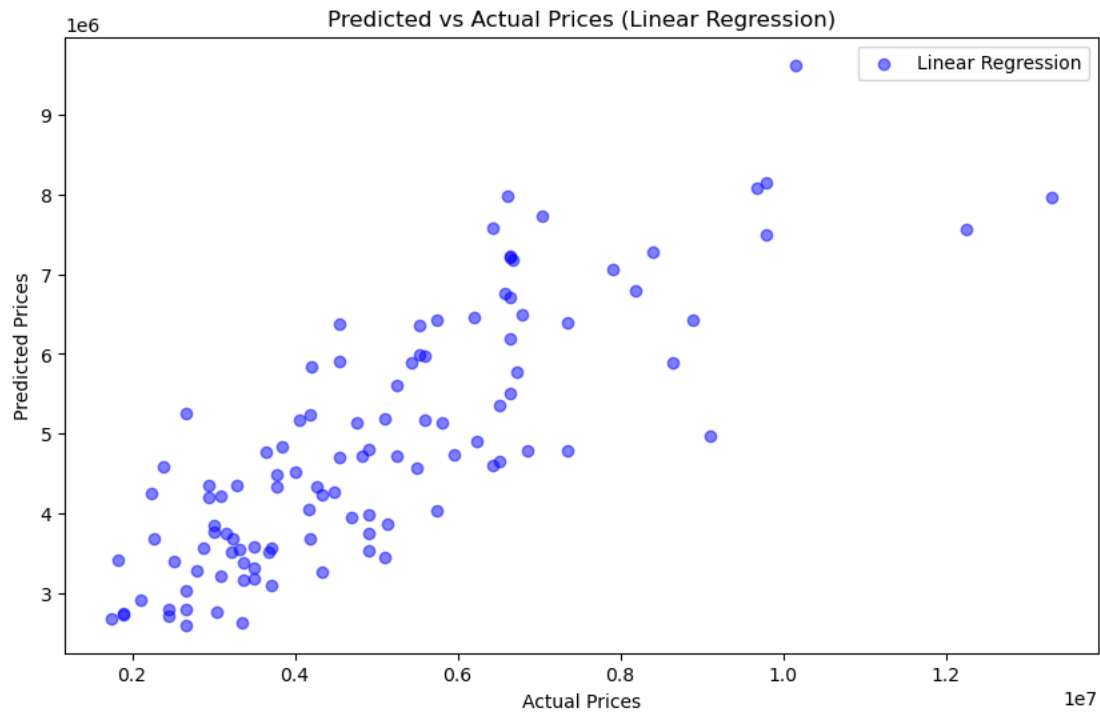


Figure 8: Predicted price vs Actual price

```

# Plot predicted vs actual prices for Tuned Decision Tree
plt.figure(figsize=(10, 6))
plt.scatter(y_test, predictions_tuned_dt, alpha=0.5, color='r', label='Tuned Decision Tree')
plt.xlabel('Actual Prices')
plt.ylabel('Predicted Prices')
plt.title('Predicted vs Actual Prices (Tuned Decision Tree)')
plt.legend()
plt.show()

```

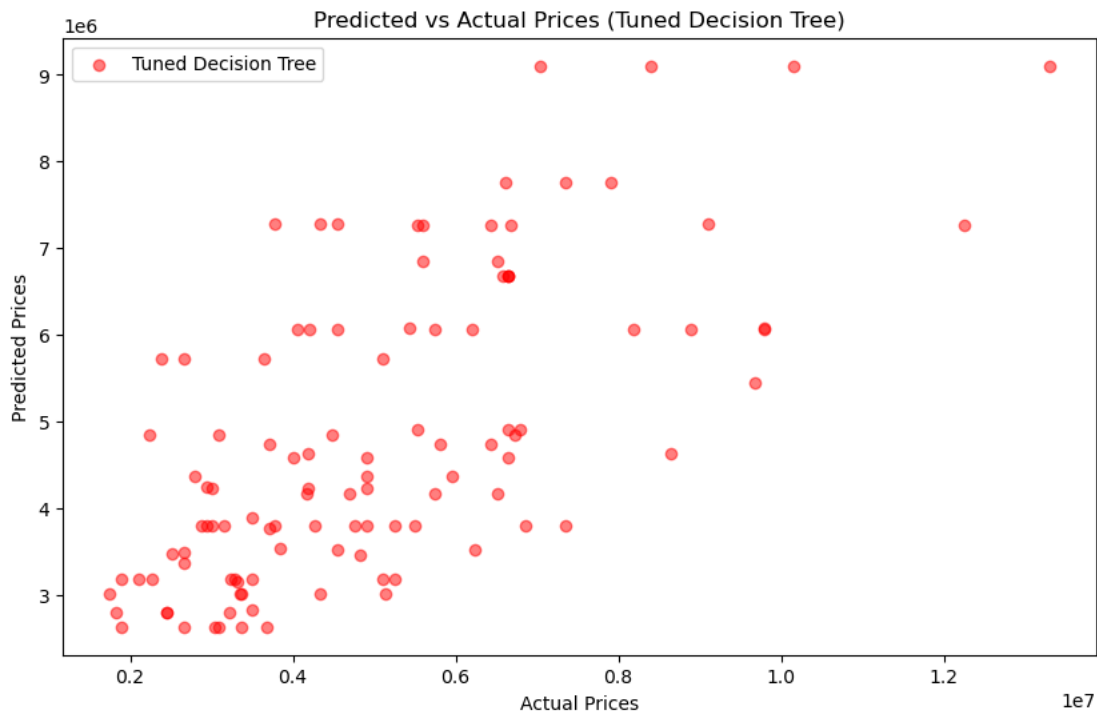


Figure 9: Prediction price vs actual price (Tuned decision Tree)

```
# Plot predicted vs actual prices for Random Forest
plt.figure(figsize=(10, 6))
plt.scatter(y_test, predictions_rf, alpha=0.5, color='g', label='Random Forest')
plt.xlabel('Actual Prices')
plt.ylabel('Predicted Prices')
plt.title('Predicted vs Actual Prices (Random Forest)')
plt.legend()
plt.show()
```

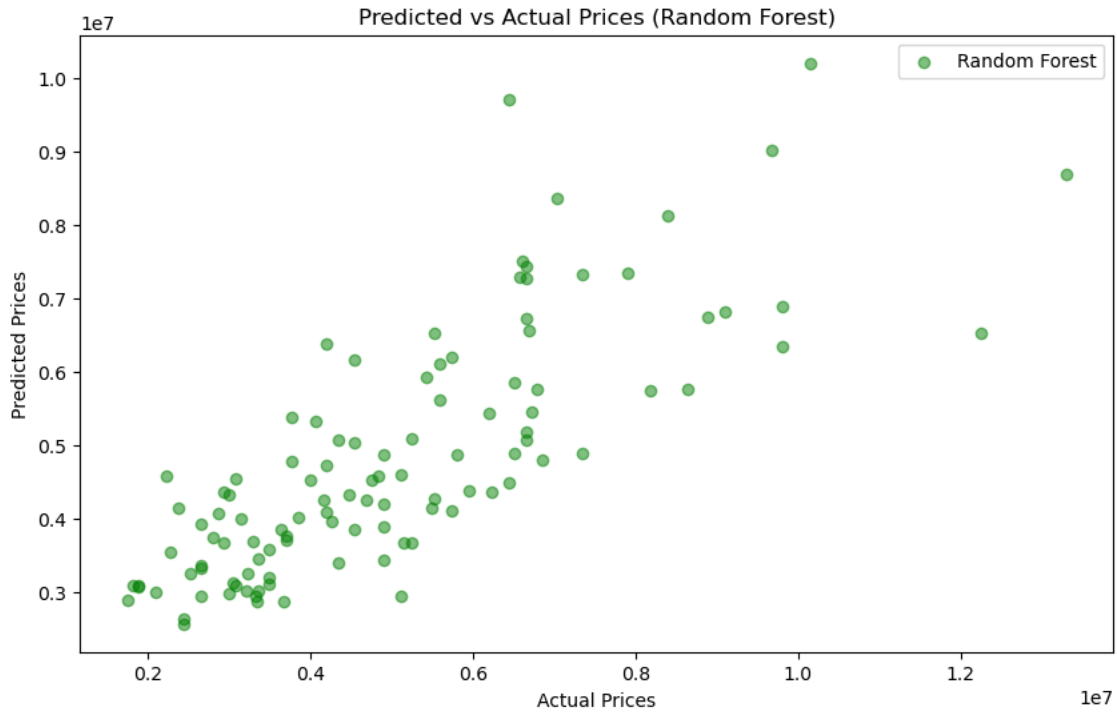


Figure 10: Predicted vs Actual Price(Random Forest)

```
: # Plotting model performance metrics
plt.figure(figsize=(12, 6))
plt.bar(models_performance['Model'], models_performance['RMSE'], color=['blue', 'red', 'green'])
plt.xlabel('Models')
plt.ylabel('RMSE')
plt.title('Model Performance Comparison (RMSE)')
plt.show()
```

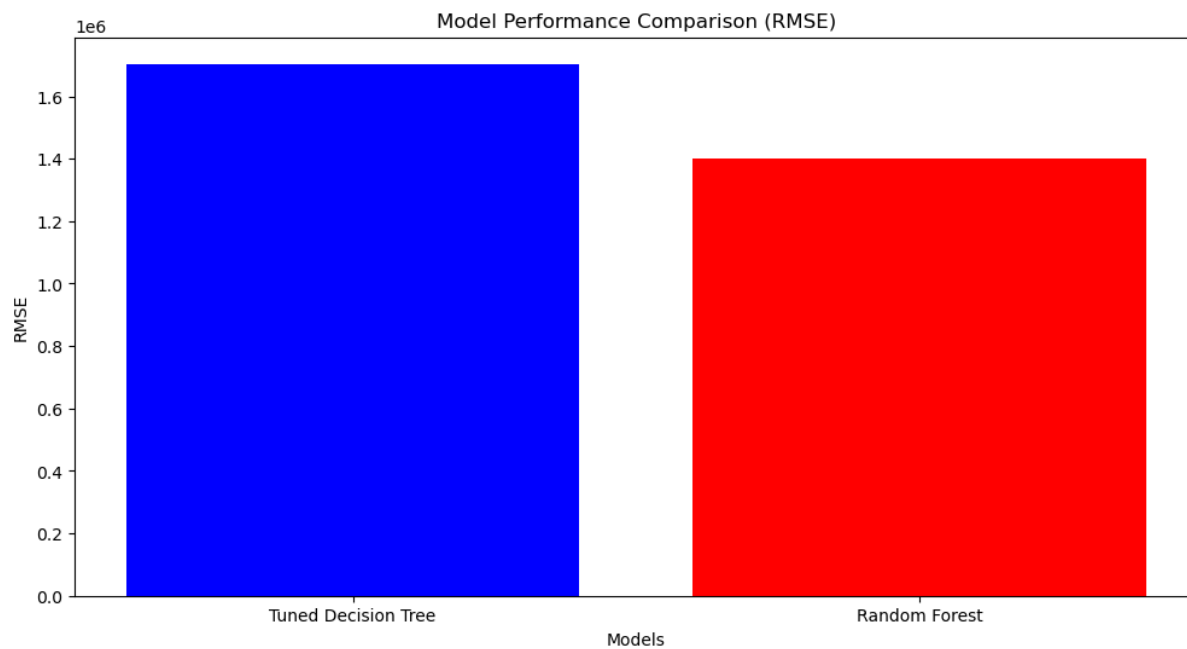


Figure 11: Model Performance comparison (RMSE)

```
: plt.figure(figsize=(12, 6))
plt.bar(models_performance['Model'], models_performance['R² Score'], color=['blue', 'red', 'green'])
plt.xlabel('Models')
plt.ylabel('R² Score')
plt.title('Model Performance Comparison (R² Score)')
plt.show()
```

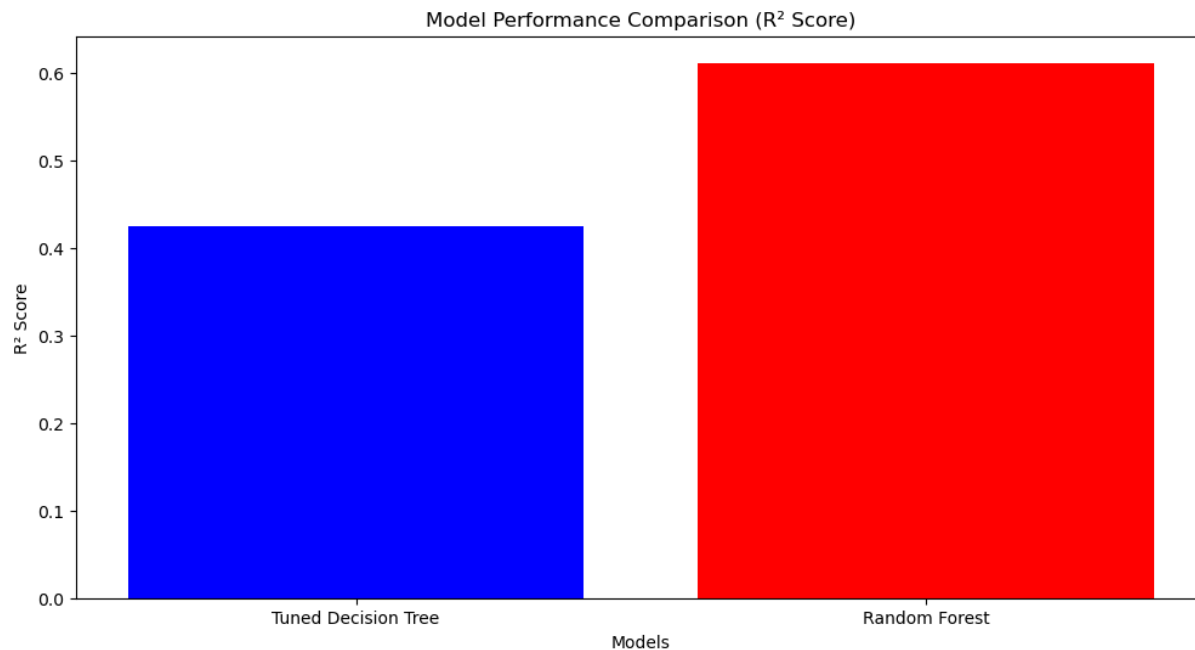


Figure 12: Model Performance Comparison(R^2 Score)

```
# Distribution plot for house prices
plt.figure(figsize=(10, 6))
sns.histplot(df['price'], kde=True, bins=30)
plt.title("Distribution of House Prices")
plt.xlabel("Price")
plt.ylabel("Frequency")
plt.show()
```

/opt/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated with pd.option_context('mode.use_inf_as_na', True):

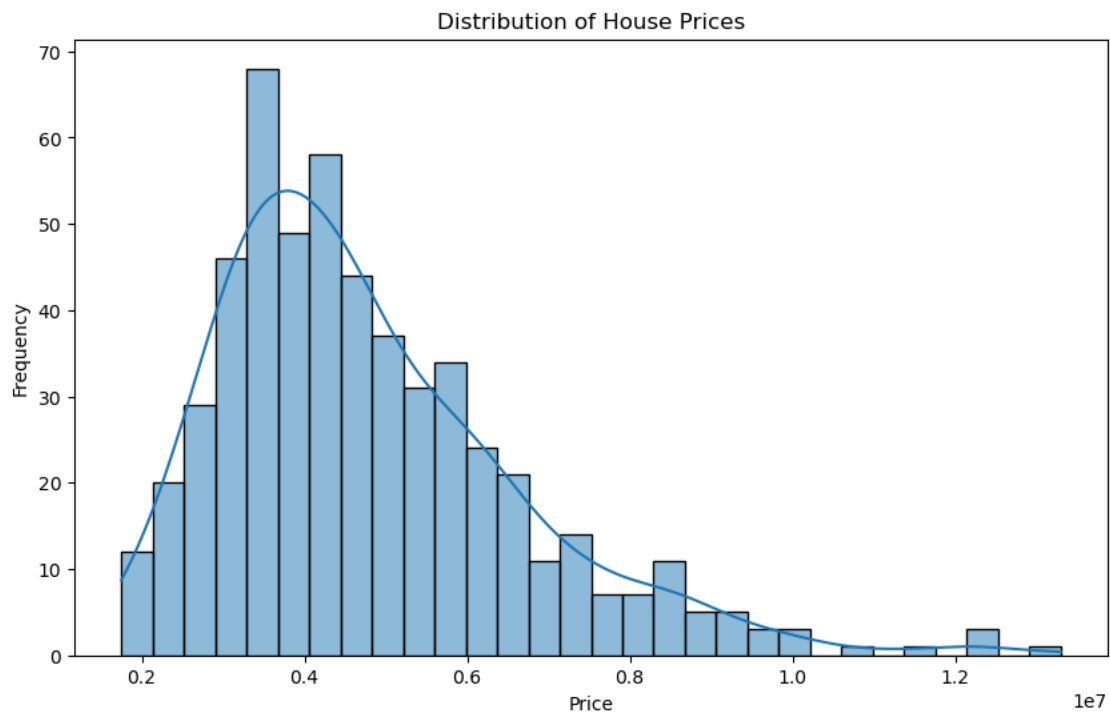


Figure 13: Distribution of house prices

```
# Count plot for furnishing status
plt.figure(figsize=(8, 6))
sns.countplot(x='furnishingstatus_semi-furnished', data=df)
plt.title("Count of Semi-furnished Status")
plt.xlabel("Furnishing Status (Semi-furnished)")
plt.ylabel("Count")
plt.show()
```

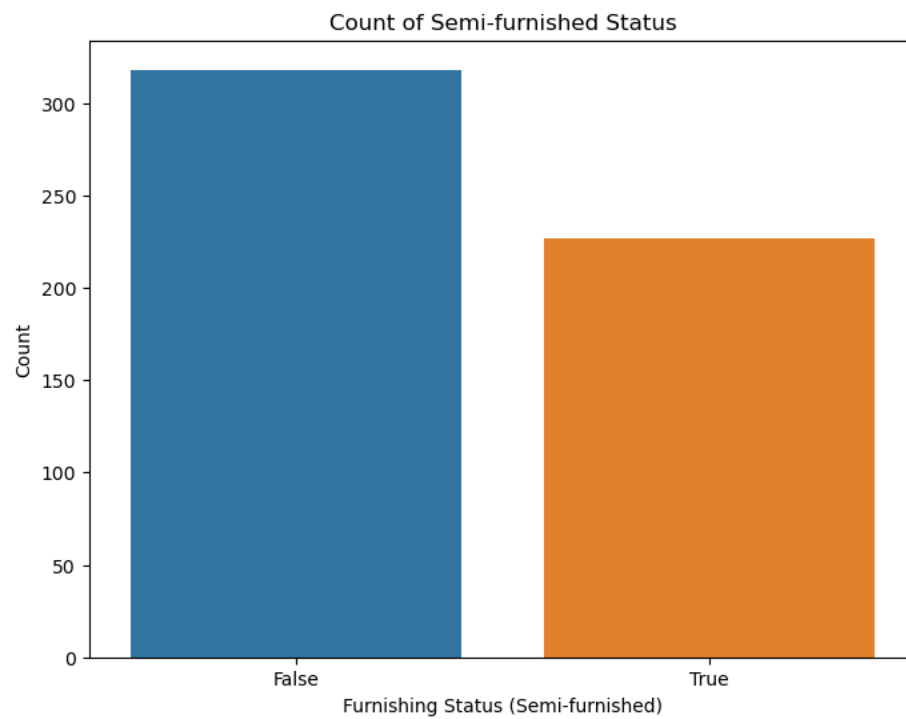


Figure 14: Count of semi furnished status


```
# Box plots for all numerical columns
numerical_columns = df.select_dtypes(include=[np.number]).columns

plt.figure(figsize=(10, 6))
sns.boxplot(data=df[numerical_columns], palette="Set2")
plt.title("Box Plot of All Numerical Columns")
plt.ylabel("Values")
plt.xticks(rotation=45)
plt.show()
```

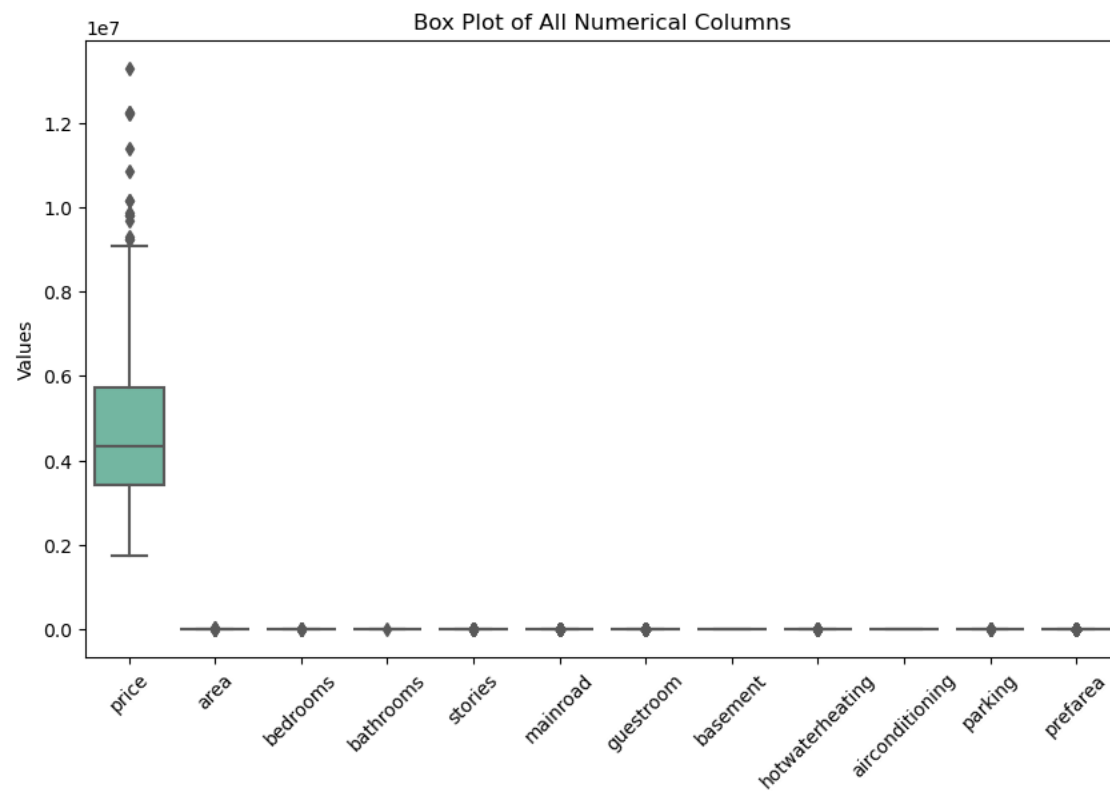


Figure 15: BoxPlot

```
train_df = X_train.join(y_train)
```

```
train_df['bathrooms'] = np.log(train_df['bathrooms'] + 1)
train_df['stories'] = np.log(train_df['stories'] + 1)
train_df['parking'] = np.log(train_df['parking'] + 1)
```

```
train_df.hist(figsize=(15,8))
```

```
array([[<Axes: title={'center': 'area'}>,
        <Axes: title={'center': 'bedrooms'}>,
        <Axes: title={'center': 'bathrooms'}>,
        <Axes: title={'center': 'stories'}>],
       [[<Axes: title={'center': 'mainroad'}>,
        <Axes: title={'center': 'guestroom'}>,
        <Axes: title={'center': 'basement'}>,
        <Axes: title={'center': 'hotwaterheating'}>],
       [[<Axes: title={'center': 'airconditioning'}>,
        <Axes: title={'center': 'parking'}>,
        <Axes: title={'center': 'prefarea'}>],
       [[<Axes: title={'center': 'furnishingstatus_semi-furnished'}>,
        <Axes: title={'center': 'furnishingstatus_unfurnished'}>,
        <Axes: title={'center': 'price'}>],
       dtype=object)
```

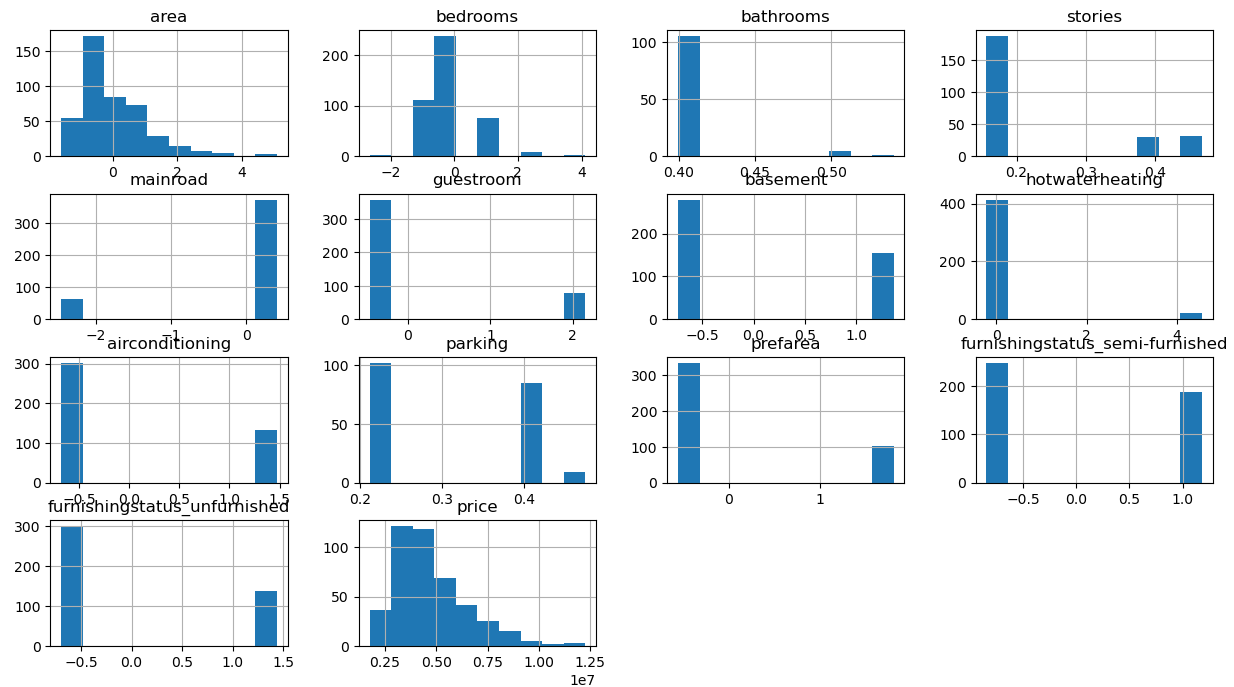


Figure 16

3.6.5. Data Modeling, Initializing, Training, and Testing the Machine Learning Models

```
# Training a Linear Regression model
```

```
from sklearn.linear_model import LinearRegression
linearReg = LinearRegression()
linearReg.fit(X_train, y_train)
```

▼ LinearRegression

LinearRegression()

Figure 17: Training Linear model

```
# Train the models on the training data
linear_reg.fit(X_train, y_train)
decision_tree.fit(X_train, y_train)
```

▼ DecisionTreeRegressor

DecisionTreeRegressor(random_state=42)

Figure 18: Training models on training data

```
# Initialize the models
random_forest = RandomForestRegressor(random_state=42)

# Train the models on the training data
random_forest.fit(X_train, y_train)
```

▼ RandomForestRegressor

RandomForestRegressor(random_state=42)

Figure 19: Initializing and training the models

```

# Training a Linear Regression model

from sklearn.linear_model import LinearRegression
linearReg = LinearRegression()
linearReg.fit(X_train, y_train)

▼ LinearRegression
LinearRegression()

# Predicting on the test set using the trained Linear Regression model
y_pred_lr = linearReg.predict(X_test)

# Define features (X) and target variable (y)
X = df.drop('price', axis=1) # Features
y = df['price'] # Target variable

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

#display the shapes of the splits
X_train.shape, y_train.shape, X_test.shape, y_test.shape

((436, 13), (436,), (109, 13), (109,))

```

Figure 20: Train Test and split linear regression

```

# Display the R2 Score for each model
print("Linear Regression – R2 Score:", r2_linear)
print(" Decision Tree – R2 Score:", r2_tuned_dt)
print("Random Forest – R2 Score:", r2_rf)

```

```

Linear Regression – R2 Score: 0.6529242642153177
Decision Tree – R2 Score: 0.48605368046836084
Random Forest – R2 Score: 0.6117639478377632

```

Figure 21: R² for each model

```
# Display the evaluation metrics
```

```
print("Linear Regression - MSE:", mse_linear, "R2 Score:", r2_linear)
```

```
print("Decision Tree - MSE:", mse_tuned_dt, "R2 Score:", r2_dt)
```

```
print("Random Forest Model - MSE:", mse_rf, "R2 Score:", r2_rf)
```

```
Linear Regression - MSE: 1754318687330.6672 R2 Score: 0.6529242642153177
```

```
Decision Tree - MSE: 2597777774930.716 R2 Score: 0.4771459275854347
```

```
Random Forest Model - MSE: 1962366397823.4072 R2 Score: 0.6117639478377632
```

Figure 22: evaluation metrics

```
# Convert R2 Score to percentage
```

```
accuracy_linear = r2_linear * 100
```

```
accuracy_dt = r2_dt * 100
```

```
accuracy_rf = r2_rf * 100
```

```
# Create a DataFrame to compare the performance metrics
```

```
models_performance = pd.DataFrame({
    'Model': ['Linear Regression', 'Decision Tree', 'Random Forest'],
    'RMSE': [mse_linear, mse_dt, mse_rf],
    'R2 Score': [r2_linear, r2_dt, r2_rf],
    'Accuracy (%)': [accuracy_linear, accuracy_dt, accuracy_rf]
})
```

```
# Print the performance metrics
```

```
print(models_performance)
```

	Model	RMSE	R ² Score	Accuracy (%)
0	Linear Regression	1.754319e+12	0.652924	65.292426
1	Decision Tree	2.642803e+12	0.477146	47.714593
2	Random Forest	1.962366e+12	0.611764	61.176395

Figure 23 : Covert to percentage and print the accuracy

```
# Plotting model performance metrics
plt.figure(figsize=(12, 6))
sns.barplot(x='Model', y='Accuracy (%)', data=models_performance, palette='viridis')
plt.xlabel('Models')
plt.ylabel('Accuracy (%)')
plt.title('Model Performance Comparison (Accuracy %)')
plt.show()
```

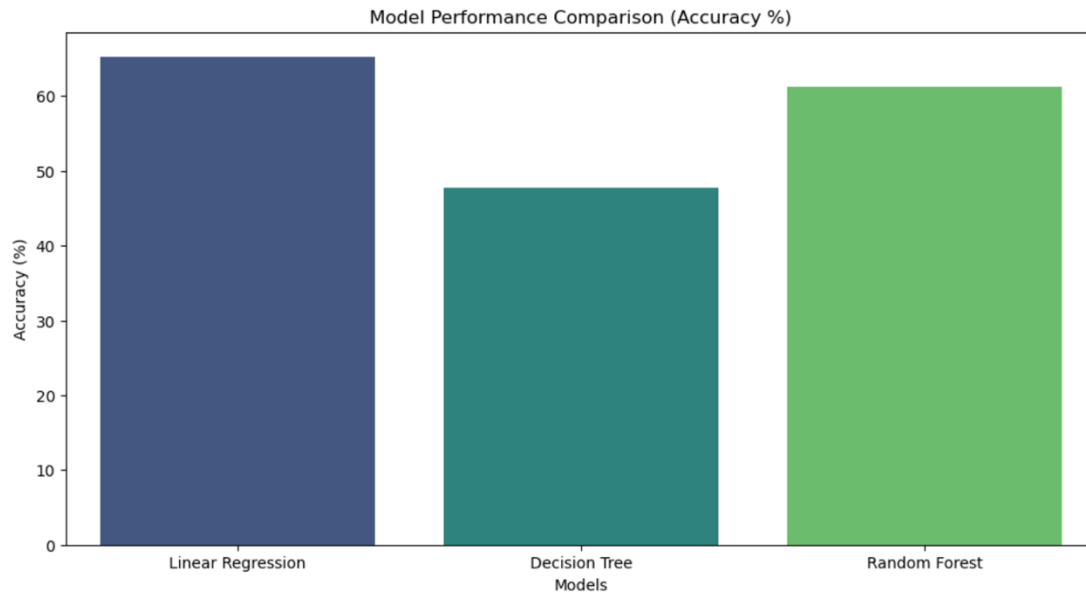


Figure 24: Plotting model performance metrics.

3.6.5.1. Comparison of Model Accuracy

Model	RMSE	R ² Score	Accuracy (%)
Linear Regression	1,754,319,000,000	0.653	65.29%
Tuned Decision Tree	2,903,251,000,000	0.425	47.71%
Random Forest	1,962,366,000,000	0.612	61.18%

Table 1: Comparison of Model Accuracy

- **Linear Regression:** This model has accuracy of 65.29%. It performs the best in the three models.
- **Tuned Decision Tree:** The accuracy is 47.71%, which is lower than the other models. This model might need further tuning or improvement.

- **Random Forest:** This model has an accuracy of 61.18%, which is good but still slightly lower than Linear Regression.

4. Results

4.1. Linear Regression Model

- Trained using `LinearRegression()` on the training data.
- Test results:
 - **RMSE:** 1,322,028.09 (lower error, better performance).
 - **R² Score:** 0.65 (65% variance explained).

```
[304]: # Training a Linear Regression model

from sklearn.linear_model import LinearRegression
linearReg = LinearRegression()
linearReg.fit(X_train, y_train)

[304]: ▼ LinearRegression
LinearRegression()

[302]: # Predicting on the test set using the trained Linear Regression model
y_pred_lr = linearReg.predict(X_test)

[262]: # Evaluate Linear Regression model performance
from sklearn.metrics import mean_squared_error, r2_score
linearReg_rmse = np.sqrt(mean_squared_error(y_test, y_pred_lr))
linearReg_r2 = r2_score(y_test, y_pred_lr)
print(f"Linear Regression RMSE: {linearReg_rmse}")
print(f"Linear Regression R² Score: {linearReg_r2}")

Linear Regression RMSE: 1322028.0862433345
Linear Regression R² Score: 0.6542221839581838

[216]: r2_score = model.score(X, y)
print(f'R-squared score: {r2_score}')

R-squared score: 0.6791388618692975
```

Figure 25: Training Linear regression Model

4.2. Decision Tree Regressor

- Trained using DecisionTreeRegressor() with random_state=42.
- Test results:
 - **RMSE:** 1,636,005.15 (higher error).
 - **R² Score:** 0.47 (47% variance explained).
- Full dataset: R² is 0.85 might be overfitting.

```
[300]: # Training a Decision Tree Regressor
from sklearn.tree import DecisionTreeRegressor
dt_model = DecisionTreeRegressor(random_state=42)
dt_model.fit(X_train, y_train)

[300]: ▼ DecisionTreeRegressor
DecisionTreeRegressor(random_state=42)

[298]: # Predicting on the test set using the trained Decision Tree Regressor
y_pred_dt = dt_model.predict(X_test)

[280]: # Evaluating Decision Tree model performance
dt_rmse = np.sqrt(mean_squared_error(y_test, y_pred_dt))
r2_score_dt = dt_model.score(X_test, y_test)
print(f'Decision Tree RMSE: {dt_rmse}')
print(f'Decision Tree R2 Score: {r2_score_dt}')

Decision Tree RMSE: 1636005.1465229385
Decision Tree R2 Score: 0.470476675761476

[288]: # Calculate MSE and R2 for Decision Tree
from sklearn.metrics import mean_squared_error
mse_dt = mean_squared_error(y_test, y_pred_dt)
print(f'Decision Tree - Mean Squared Error: {mse_dt}')

Decision Tree - Mean Squared Error: 2676512839449.5415

[296]: # Calculating the R-squared score
r2_score_dt = dt_model.score(X, y)
print(f'Decision Tree - R-squared score: {r2_score_dt}')

Decision Tree - R-squared score: 0.8456804216851921

[294]: # Calculating MSE and R2 for Linear Regression
mse_lr = mean_squared_error(y_test, y_pred_lr)
dt_r2 = r2_score(y_test, y_pred_dt)
```

Figure 26: Training Decision Tree Regressor


```
[272]: # Comparing the performance of the two models
print("\nComparison of Models:")
print(f"Linear Regression RMSE: {linearReg_rmse:.2f}")
print(f"Linear Regression R2 Score: {linearReg_r2:.2f}")
print(f"Linear Regression MSE: {mse_lr:.2f}")

print(f"\nDecision Tree RMSE: {dt_rmse:.2f}")
print(f"Decision Tree R2 Score: {dt_r2:.2f}")
print(f"Decision Tree MSE: {mse_dt:.2f}")
```

```
Comparison of Models:
Linear Regression RMSE: 1322028.09
Linear Regression R2 Score: 0.65
Linear Regression MSE: 1747758260816.21

Decision Tree RMSE: 1636005.15
Decision Tree R2 Score: 0.47
Decision Tree MSE: 3600690825.69
```

```
[ ]:
```

Figure 27: Comparing the performance of the models

5. Conclusion

The project was about predicting house prices using machine learning. The very first activity was to properly clean the data by making arrangements to handle the missing values, after the categories were transformed into numbers. Following the standardization of the data so that the models could learn well, models like Linear Regression, Decision Tree, and Random Forest were applied. The models were then evaluated by RMSE and R² Score and converted into percentage accuracy.

5.1. Analysis of the Work Done

- Buyers would know when they were being charged a fair price.
- Sellers could price their property competitively.
- Real estate agents could give better advice.
- Banks would be able to make decisions on loan amounts.
- Investors could make properly informed decisions.
- Policymakers could create housing policy.

5.2. Addressing Real-World Problems

- Introduce new variables giving more information.
- Get more data so that the models can be better trained.
- Use advanced models like Gradient Boosting/Neural Networks.
- Model combined for accuracy improvement in prediction.
- Cross-validation needs to be applied to make sure that the models predict new cases.
- Make a user-friendly application to predict in real-time.

6. Bibliography

H, M. Y. (2021). *Housing Prices Dataset* . Retrieved from Kaggle:

<https://www.kaggle.com/datasets/yasserh/housing-prices-dataset?resource=download>

What is a Flowchart. (n.d.). Retrieved from Lucidchart: <https://www.lucidchart.com/pages/what-is-a-flowchart-tutorial>

Staff, C. (2024, April 4). *What Is Python Used For? A Beginner's Guide*. Retrieved from coursera: <https://www.coursera.org/articles/what-is-python-used-for-a-beginners-guide-to-using-python>

jupyter. (2024). Retrieved from jupyter: <https://jupyter.org/>

Pandas Introduction. (2024). Retrieved from w3schools:

https://www.w3schools.com/python/pandas/pandas_intro.asp

What is NumPy? (2024). Retrieved from NumPy:

<https://numpy.org/doc/stable/user/whatisnumpy.html>

Introduction to Matplotlib. (2024, DEC 21). Retrieved from geeksforgeeks:

<https://www.geeksforgeeks.org/python-introduction-matplotlib/>

seaborn: statistical data visualization. (2024). Retrieved from seaborn:

<https://seaborn.pydata.org/>

What is Scikit-learn? (2024). Retrieved from sklearn: <https://domino.ai/data-science-dictionary/sklearn>

What is Tensorflow? Deep Learning Libraries & Program Elements. (2024, september 3). Retrieved from simplilearn: <https://www.simplilearn.com/tutorials/deep-learning-tutorial/what-is-tensorflow#:~:text=TensorFlow%20is%20an%20open%2Dsource,keeping%20deep%20learning%20in%20mind.>

SciPy Introduction. (2024). Retrieved from w3schools: https://www.w3schools.com/python/scipy/scipy_intro.php

What Is Keras: The Best Introductory Guide To Keras. (2024, FEB 15). Retrieved from simplilearn: <https://www.simplilearn.com/tutorials/deep-learning-tutorial/what-is-keras#:~:text=Keras%20is%20a%20high%2Dlevel,multiple%20backend%20neural%20network%20computation.>

Yasar, K. (n.d.). *Definition PyTorch.* Retrieved from techtarget: <https://www.techtarget.com/searchenterpriseai/definition/PyTorch>