A red rectangular object with white border

Description automatically generated

**University of Hertfordshire**

Department of Physics, Astronomy and Mathematics MSc Data Science

Project report: 7PAM2002 – Data Science Project

**House Price Prediction with Machine Learning Regression Techniques**

**Supervisor:** MAN LAI TANG

**Data Submitted:**

**Word Count: 6283**

**DECLARATION STATEMENT**

This report is submitted in partial fulfilment of the requirement for the degree of Master of  
Science in Data Science at the University of Hertfordshire.

I have read the guidance to students on academic integrity, misconduct and plagiarism  
information at Assessment Offences and Academic Misconduct and understand the  
University process of dealing with suspected cases of academic misconduct and the possible  
penalties, which could include failing the project module or course.

I certify that the work submitted is my own and that any material derived or quoted from  
published or unpublished work of other persons has been duly acknowledged. (Ref. UPR  
AS/C/6.1, section 7 and UPR AS/C/5, section 3.6). I have not used chatGPT, or any other  
generative AI tool, to write the report or code (other than were declared or referenced).

I did not use human participants or undertake a survey in my MSc Project.  
I hereby give permission for the report to be made available on module websites provided  
the source is acknowledged.

**Student Name:** pavan kumar chimata  
**Student signature:** ch pavan kumar **Student SRN number:** 22033170

**UNIVERSITY OF HERTFORDSHIRE  
SCHOOL OF PHYSICS, ENGINEERING AND COMPUTER SCIENCE**

**Abstract**

Housing is an essential human need, and real estate investment has long been regarded as a prudent choice due to the general trend of property values to either increase or remain stable. However, the property market is complex, influenced by a multitude of factors that complicate the precise estimation of house values and the avoidance of overpayment. Traditional methods of property evaluation often suffer from human error, resulting in inaccurate predictions. This study seeks to identify the key factors influencing house prices through exploratory data analysis (EDA) and feature engineering, with the aim of developing machine learning models that can accurately predict property prices using California housing data.

This research used different Regression models like quantile ,linear regressions and random forest, decision tree , XGBoost regressions. By using quantile and linear regression coefficient was estimates to see how each individual feature affecting the dependent feature. R² score with 0. 8331,

MAE with 27,123.84, and a RMSE with 39,923.81. These results are particularly valuable for the real estate industry, where precise predictions can significantly influence investment decisions, pricing strategies, and market assessments.

**Contents**

1. Introduction .................................................................................................. 4

1.1 Research Objective ................................................................................... 4

2. Literature Review .......................................................................................... 5

2.1 Research Question ........................................................................... ........ 5

2.2 Methodology ............................................................................................. 5

2.3 Related Work ............................................................................................. 6

3. Dataset ............................................................................................................ 7

3.1 Review of Columns ..................................................................................... 7

3.1.1 Statistical Information .......................................................................... 8

3.1.2 Data Type and Quality .......................................................................... 8

3.1.3 Data Ethical Considerations .................................................................. 9

3.1.4 Data Preprocessing ............................................................................... 9

3.1.5 Handling Missing Values ....................................................................... 9

3.1.6 Feature Engineering ............................................................................ 10

3.1.7 Encoding Categorical Variables ............................................................ 11

3.2 Exploratory Data Analysis (EDA) ................................................................ 12

3.2.1 Median Income Distribution Plot ......................................................... 12

3.2.2 Total Bedroom Distribution Plot .......................................................... 13

3.2.3 Histogram Plot ..................................................................................... 13

3.2.4 Boxplot for Ocean Proximity ................................................................ 14

3.2.5 Bar Plot for Ocean Proximity ................................................................ 15

3.2.6 Correlation Heatmap ............................................................................ 16

3.2.7 Scatter Plot ........................................................................................... 16

4. Machine Learning Models .............................................................................. 19

4.1 Quantile Regression ................................................................................... 19

4.1.1 Estimation of Coefficients .................................................................... 20

4.2 Linear Regression ....................................................................................... 21

4.2.1 Estimation of Coefficients .................................................................... 21

4.3 Comparing Coefficient Values for Quantile and Linear Regression ........... 23

4.4 Decision Tree Regression .......................................................................... 24

4.4.1 Training a Decision Tree Regression Model ........................................ 24

4.5 Random Forest Regression ....................................................................... 24

4.5.1 Training a Random Forest Model ........................................................ 24

4.6 XGBoost Regression .................................................................................. 25

4.6.1 Training an XGBoost Model ................................................................. 25

4.7 Hyperparameter Tuning ........................................................................... 25

4.8 Evaluation Metrics ................................................................................... 25

5. Discussion of Results ..................................................................................... 27

5.1 Observations ............................................................................................ 29

5.2 Discussion of Machine Learning Model Results ....................................... 29

5.3 Comparison of Models & Final Model Selection ..................................... 30

5.4 Limitations & Future Work ...................................................................... 31

6. Conclusion ................................................................................................... 32

7. References ................................................................................................... 33

8. Appendix ...................................................................................................... 34

Chapter 1. **Introduction**

Housing is an essential human need, and real estate is often viewed as a sound investment because property values tend to either appreciate or remain steady over time. However, the real estate market is complex, influenced by numerous factors, making it challenging to accurately estimate property values and avoid overpaying. Traditional methods of property valuation, such as appraisals and market analyses, are often susceptible to human error, leading to inaccurate predictions. This project focuses on impact of various parameters like bedrooms, location ,household income etc on house price by using various machine learning methods.by analysing the data with EDA process and various regression techniques house price can be predicted accurately as much as possible. This project also estimated coefficient values with quantile and linear regression to understand how various features affecting house price values. Apart from all the models used in project the tuned XGBoost model achieved the highest accuracy, with R² score of 0.8331, a MAE of 27,123.84, and a RMSE of 39,923.81. These findings are particularly valuable in the real estate industry, where precise predictions can significantly influence investment decisions, pricing strategies, and market assessments.

Background

Housing is a fundamental necessity, making the real estate sector vital for investment, given that property values generally appreciate or stay stable. Nonetheless, the real estate market is intricate, with numerous factors influencing it, which complicates the process of purchasing a property without overpaying. Accurate price predictions are crucial for stakeholders like buyers, sellers, and agents, who depend on various data points to make informed decisions. Traditional valuation methods, such as appraisals and market analyses, often suffer from human error and inefficiency, especially in volatile markets. These technologies can manage large datasets and reveal patterns that traditional methods might overlook, leading to more accurate property price predictions. Machine learning models continuously evolve by incorporating new data, ensuring that predictions remain relevant over time.

**1.1 Research Objective**

This target of this research influencing his to identify the situations that making house prices increase and decrease . it is done through comprehensive exploratory data analysis (EDA) and feature engineering. By understanding these factors, the study aims to develop machine learning models that can accurately predict property prices using historical data. This involves selecting relevant features, training various models, and evaluating their performance to determine the most effective one. The research aims to offer a reliable solution for predicting property prices, addressing a significant challenge in the real estate industry.

Chapter 2 **Literature Review**

The increased technology in Artificial intelligence and machine learning gave real estate a big advantage. Several studies have explored this area using various machine learning models.

**References**

1. For instance, Rana et al. (2020) compared different regression algorithms, including XGBoost, Decision Tree, SVR, and Random Forest, for price prediction. They found that the Decision Tree algorithm was the most accurate, achieving a 99% accuracy rate.
2. T.D. Phan (2018) focused on Melbourne's housing market, employing techniques such as PCA and SVM to enhance prediction accuracy.
3. Lu et al. (2017) proposed For Kaggle "House Prices: Advanced Regression Techniques" challenge dataset hybrid Lasso-Gradient Boosting perform well . This project aims to employ similar machine learning methods to predict house prices effectively, catering to customers' budgets and needs.

**2.1 Research Questions**

The main Agenda of the research is to predict house price by various ML algorithms. Which will benefit the stakeholders such as buyers, sellers, and real estate agents.so that when buying a property or house machine learning will be able to predict the accurate price ,so that sellers won’t sell for low price and buyers won’t buy for high price.

Objectives

* Develop machine learning models capable of accurately predicting house prices using historical data.
* Identify the key factors that influence housing prices through EDA and Feature Engineering.
* Determine the most relevant features that affect housing values.

**2.2 Methodology**

1. **Machine Learning Algorithms**: The study deals with various ML models to identify most effective and suitable model for predicting house price.
2. **Improving Accuracy**: The research employs various techniques, such as data preprocessing, feature engineering, and hyperparameter tuning, to enhance the accuracy of housing price prediction models.

**Related Work**

Several studies have examined various machine learning techniques for predicting property prices, demonstrating the effectiveness of combining robust regression models and data reduction methods. Phan's case study on Melbourne used PCA for data optimization and SVM regression, resulting in improved house price forecasting accuracy. Rana et al. (2020) identified Decision Tree Regression as the most accurate algorithm, achieving 99% accuracy. Lu et al. (2017) proposed a hybrid model combining Lasso Regression and Gradient Boosting, which enhanced prediction performance but required significant computational resources. Other studies, such as those by Mu et al. (2014) and Gu et al. (2011), highlighted the potential of hybrid machine learning approaches, such as combining Genetic Algorithms with SVM, to improve prediction accuracy. Chen et al. (2017) focused on SVM's ability to manage complex housing market data, refining predictions with Stepwise regression.

Dataset

The dataset for this project comes from the California Census Data, provided by the US Census Bureau and available on Kaggle. This dataset offers a comprehensive feature set, covering a wide range of attributes that could impact house values for every block group in California. The target variable, the median house value, is one of 10 distinct features in the dataset. These features provide a deep understanding of California's housing market by addressing various geographic and socioeconomic factors. The data includes population details for each block group, the average rooms per house, the median house age, and the median income of people who living at the area. Additionally, it contains geographical details such as latitude and longitude, and also the average number of bedrooms per property. In this study, the median house value is the dependent variable, representing the median value of houses within each block group, which offers a consistent benchmark for comparing properties across different geographic areas.

**Chapter 3 DATASET**

* **longitude:** describes the location of houses distance from west and East
* **latitude:** describes the north and south location distance of houses.
* **housingMedianAge:** it shows the Median house age in years
* **totalRooms:** gives information about number of rooms in a block
* **totalBedrooms:** it shows or gives information about total number of bedrooms in block.
* **population:** number of people living in a block
* **households:** gives information about number of families living in a block
* **medianIncome:** median income earned by household or families (measured in tens of thousands of USD)
* **medianHouseValue:** gives income about Median house value in a block (measured in US Dollars)
* **oceanProximity:** gives information about house location which are near ocean proximity

**3.1.1 Review of Columns**

**A screenshot of a graph

Description automatically generated**

**3.1.2 Statistical Information:**

A screenshot of a data

Description automatically generated

**3.1.3 Data Ethical Consideration:**

** Permission to Use Data:** Everyone can use the data in this website because it’s an public website that allows everyone to access and download the data and use it. Licence was mentioned at this link. [**https://ev.turnitinuk.com/app/carta/en\_us/?u=19488046&s=1&ro=103&student\_user=1&o=236178042&lang=en\_us**](https://ev.turnitinuk.com/app/carta/en_us/?u=19488046&s=1&ro=103&student_user=1&o=236178042&lang=en_us)

** Data Source Verification:**

* Data is collected form reputable website called Kaggle, here is the website link

[**https://www.kaggle.com/datasets/nazishjaveed/california-house-price-prediction**](https://www.kaggle.com/datasets/nazishjaveed/california-house-price-prediction)

**3.1.4 Data Type of Each Feature :**

A screenshot of a computer

Description automatically generated

# **3.1.5 Preprocessing of Data**

Data was pre-processed to help models predict efficiently. To happen data should not have any null values in it and it involves removing duplicate values and scaling the data and then applying one-Hot encoding to convert categorical variable into numerical values

## **3.1.6 Handling Missing Values**

The dataset contains 20,640 instances, but only 20,433 entries had valid data for the total number of bedrooms, leaving 207 instances with missing values. The distribution of bedroom numbers was positively skewed, with a longer tail on the right. The missing values are filled with median values because median is not much influenced by outliers.A screenshot of a computer code

Description automatically generatedA white background with black lines

Description automatically generated

## 

## **3.1.8 Feature Engineering**

Additional variables were created to improve the dataset and give the predictive model additional insightful features. The ratios of the total rooms, total bedrooms, and population of households were computed. These additional variables, "rooms/household," "bedrooms/ household," and "population/ household," capture the average distribution of rooms, bedrooms, and population inside each family, offering a more nuanced picture of the data. The original columns "total rooms," "total bedrooms," "population," and "households" were eliminated from the dataset once these derived variables were created. The activity aimed to streamline the feature set and lessen redundancy.

A table with numbers and text

Description automatically generated

## 

## **3.1.9 Encoding Categorical Variables**

The nominal feature "ocean proximity" was converted into a numerical format to make it compatible with machine learning algorithms. One-Hot encoding applied to categorical values this metho convert categorical values into numerical matrix to be precise a binary matrix were created. This process created additional columns, each corresponding to a distinct category within the "ocean proximity" variable. These new binary columns were integrated back into the dataset, and the original "ocean proximity" column was removed to eliminate redundancy and maintain a purely numerical feature set.

A screenshot of a computer

Description automatically generated

# **3.2 Exploratory Data Analysis (EDA)**

**3.2.1 PLOTTING DISTRIBUTION PLOT FOR MEDIAN INCOME:**

### Key Observations:

1. **Distribution Shape**:
   * The distribution is right-skewed
   * There is a noticeable peak around an income value of 3 to 4. while most households have a median income below 5.

A graph of a distribution of income

Description automatically generated

**3.2.2 PLOTTING DISTRIBUTION PLOT FOR TOTAL BEDROOMS:**

### Key Observations:

1. **Distribution Shape**:
   * The number of bedrooms ranges from 0 to over 6 ,with a peak (highest frequency) occurring around 4-6 bedrooms.
   * The distribution is right-skewed (positively skewed), meaning there are more houses with fewer bedrooms.

A graph of distribution of bedroom

Description automatically generated

**3.2.3** **HISTOGRAM PLOT COMPARING EVERY FEATURE:**

### Key Takeaways:

* **Skewness**: Many distributions, such as total rooms, total bedrooms, population, households, and median income, are right-skewed. This indicates the presence of outliers with very high values.
* **Median House Age**: The annotation mentions that the median age of houses is mostly between 15 to 37 years, which is visible in the distribution with several peaks within this range.

A group of blue and white graphs

Description automatically generated

**3.2.4 PLOTTING BOXPLOT TO COMPARE OCEAN PROXIMITY AND MEDIAN HOUSE VALUE**:

 **Proximity to water** generally increases house values, with the highest values seen for properties on islands and near the ocean.

 **Inland properties** have the lowest median house values, indicating that proximity to the ocean is a significant factor in determining house prices.

 **Variation**: There is considerable variation in house prices within categories close to the ocean, particularly NEAR BAY and <1H OCEAN.

A diagram of different colored squares

Description automatically generated

**3.2.5 COUNTING OCEAN PROXIMITY USING BAR PLOT:**

### **Summary:**

* **Prevalence**: The most common locations for homes in this dataset are within one hour from the ocean and inland areas.
* **Rarity**: Homes on islands are extremely rare, and those near the ocean or bay are less common.
* **Market Dynamics**: The distribution shows that while proximity to water is valuable, most homes are located within a reasonable distance (like 1 hour) rather than directly by the ocean, which could reflect cost or availability constraints

A bar graph with different colored squares

Description automatically generated

## **3.2.6 CORRELATION HEATMAP:**

1. **Median House Value vs. Median Income**: The positive correlation (0.69), tells that higher incomes are linked with higher house values.
2. **Total Bedrooms and Total Rooms**: These two variables have a very high correlation (0.93), which makes sense as houses with more rooms have more bedrooms.
3. **Households and Total Rooms**: the positive correlation with 0.92 explains that places where large households are staying likely to have more rooms.
4. **Latitude and Longitude**: There is a negative correlation (-0.92), likely reflecting geographical patterns within the dataset.
5. **Ocean Proximity Variables**: The correlations between different ocean proximity categories are low to moderately negative, indicating that these categories are somewhat mutually exclusive but not perfectly so.
6. **Housing Median Age**: The 0.11 correlation suggesting that older homes might be valued slightly higher, though this correlation is not very strong.

A screenshot of a graph

Description automatically generated

Figure 1-Correlation Heatmap

**3.2.7 PLOTTING SCATTER PLOT B/W MEDIAN INCOME & MEDIAN HOUSE VALUE:**

As the median income increases, the median house value also tends to increase.

1. **Concentration**: Most of the data points are concentrated in the lower to middle range of income values (0 to 6) and house values (up to around $300,000).
2. **Outliers**: There are some outliers with higher income values and correspondingly high house values, although they are less frequent.

A blue dotted chart with white text

Description automatically generated

# 

# Chapter **4. Machine Learning Models**

The Dataset split into two variables which are predictor variables and the target variable, with the latter being the median house price. Then 75% of data considered as training data and 25% of the data is assumed as testing data.

Normalization was applied to standardize the continuous predictor variables. Specifically, the continuous features in the training set were normalized to have a mean is zero and standard deviation as one. The same normalization parameters, derived from the training data, were subsequently used to scale the continuous features in the testing set.

This approach ensures consistency in feature scaling across both training and testing datasets, which is very important for maintaining the integrity and reliability of model's performance evaluation.

## 

## **4.1 Quantile Regression**

Quantile Regression is a robust statistical technique that extends beyond traditional linear regression by estimating various quantiles within the conditional distribution of the target variable. Unlike Ordinary Least Squares (OLS) regression, which centers on predicting the mean of the dependent variable based on the independent variables, quantile regression allows for the estimation of different points (quantiles) in the distribution. This makes it especially useful when dealing with data that includes outliers or displays heteroscedasticity (unequal variance), as it does not rely on the assumption of constant error variance that underpins OLS regression. While OLS is effective when homoscedasticity is present, it may falter when confronted with data exhibiting varying variance or a skewed distribution in the response variable. By estimating multiple quantiles, such as the median or other specific percentiles, quantile regression provides a richer and more nuanced understanding of the relationships between variables.

At its core, Quantile Regression focuses on estimating conditional quantile functions of the response variable. The quantile function, denoted as Qy(τ|X), at a specific quantile level τ (where τ ranges from 0 to 1), represents the τ-th quantile of the response variable given a set of predictors X. This relationship is mathematically expressed as:

Qy(τ∣X)=X⊤β(τ)

In this equation, β(τ) represents the vector of regression coefficients corresponding to the τ-th quantile. Unlike OLS, which minimizes the sum of squared residuals, Quantile Regression minimizes a sum that applies asymmetric penalties depending on whether the residual is above or below the estimated quantile. The optimization problem for estimating β(τ) is formulated as:

min⁡β∑i=1nρτ(yi−Xi⊤β)minβ∑i=1nρτ(yi−Xi⊤β)

Here ρτ​(u)=u(τ−1(u<0)), where 1(u<0) is known as the check function,

ρτ​(u)=u(τ−1(u<0))

Here:

* u=yi−Xi⊤. is the residual.
* 1(u<0) is the indicator function that equals 1 if u<0 and 0 otherwise.
* The expression can also be written explicitly as:

ρτ(u)={τu if u≥0

(τ−1)u if u<0

**4.1.1 Coefficient Estimation**:

To investigate the relationship between the independent and dependent variables across various points of the conditional distribution, Quantile Regression models were fitted at different quantile levels: 0.01, 0.10, 0.50, 0.95, and 0.99. The analysis was carried out using the `Qreg` function, which takes the quantile level, training data, and testing data as inputs. For each quantile, a regression model was trained with the training data, and the resulting model coefficients and their confidence intervals were calculated. Predictions were then generated for the testing subset.

The coefficients obtained from these quantile regression models offer valuable insights into how each predictor variable affects different quantiles of the response variable's distribution. By calculating the range between the predictions at the 1st and 99th quantiles, the variability in the predicted median house values can be assessed, providing a comprehensive view of the expected distribution of house values.

**Interpretation of Coefficient Estimates**:

The estimated coefficients and their confidence intervals across various quantiles highlight the varying influence of each predictor on housing prices. At the 0.01 quantile, factors like longitude, latitude, and population per household are observed to negatively influence housing prices, while median income, rooms per household, and proximity to the ocean have a positive impact. As the quantile level increases to 0.10, the negative effects of longitude and latitude become more pronounced, yet the positive influence of median income and proximity to the ocean remains strong. Similar trends persist at the 0.50 quantile, with median income continuing to exert a significant positive influence, and bedrooms per household contributing positively as well. At higher quantiles, such as 0.95 and 0.99, the positive effects of median income and proximity to the ocean become even more dominant, while the negative impacts of longitude and latitude intensify. Overall, median income and proximity to the ocean consistently emerge as strong positive predictors of housing prices, whereas longitude, latitude, and population per household generally exert negative influences.

**Quantile Regression Model Training:**

A quantile regression model was specifically trained on the median quantile, representing the 50th percentile. The quantile model was trained with training dataset at 0.5 quantile , The trained model was applied to test data to predict dependent variable. These predictions are crucial for understanding the trends and patterns in the data and can be compared to the actual observed values to achieve the model's accuracy.

A table with numbers and text

Description automatically generated

## **4.2 Linear Regression**

**Linear Regression Overview:**

A basic statistical method that is frequently used in data analysis and predictive modelling is linear regression. It uses a linear equation to establish a link between a dependent variable and one or more independent variables. This technique offers a straightforward but efficient way to make predictions, and it is especially helpful for examining how changes in predictor variables impact the response variable. Finding the best-fitting line that minimises the difference between the actual and anticipated values is the primary objective of linear regression. The relationship between the dependent variable yy and the independent variable xx is represented by the equation:

y=β0+β1x+ϵy=β0​+β1​x+ϵ

**4.2.1Estimation of Coefficients:**

The Ordinary Least Squares (OLS) method is used to estimate the coefficients β0 is a intercept ,and β1 the slope in a linear regression model. To ensure the greatest possible fit for the data, this method seeks to identify the coefficient values that minimise the sum of squared residuals. The following objective function is what the OLS approach aims to minimise:

:

Minimiz e  i=1n(yi−y^i)2Minimize i=1∑n​(yi​−y^​i​)2

Where yi = observed values and,

Y^I = predicted values by model

The OLS method was used to estimate the coefficients in the linear regression model. To act as the intercept, a constant term was added to the feature matrix. The response variable and the feature matrix that had the constant term added to it made up the training data that the model was then trained on. Following model fitting, the estimated the estimated coefficients were obtained, representing the weights assigned to each attribute. These coefficients provide the most accurate linear, unbiased estimates of the relationship between the dependent variable and the predictors. They are calculated by minimizing the residual sum of squares between the observed and predicted values generated by the linear model.

Minimize

∑n​(yi​−(β0​+β1​xi​))2

To estimate the coefficients in the linear regression model, the OLS method was applied. A constant term was included in the feature matrix to serve as the intercept. The model was then trained using the training data, which consisted of the response variable and the feature matrix augmented with the constant term. After fitting the model, the estimated coefficients were obtained, representing the weights assigned to each attribute. These coefficients provide the most accurate linear, unbiased estimates of the relationship between the dependent variable and the predictors. They are calculated by minimizing the residual sum of squares between the observed and predicted values generated by the linear model.

**Interpretation of Coefficient Estimates:**

Both longitude and latitude negatively impact house prices, suggesting that moving further along these coordinates leads to lower prices. The median age of houses positively influences prices, indicating that older homes tend to have higher values. Median income has a significant positive effect, showing that higher income levels are associated with higher house prices. The number of rooms per household negatively affects prices, while the number of bedrooms per household has a positive impact. Population per household slightly decreases house prices. Proximity to the ocean greatly increases house prices, with properties closer to the ocean, bay, or within an hour of the ocean being more valuable. Being inland has a smaller but positive effect on prices.

A black and white text on a white background

Description automatically generated

**Linear Regression Model Training:**

Using the properties in the dataset, a linear regression model was created to predict the target variable. A dataset including the response variable, and the predictor variables was used to train the model. The model determined the ideal weights for every feature during training, defining the connection between the independent and dependent variables. The model was used to generate predictions on the test dataset once it had been trained.

**4.3Comparing Co-Efficient Values For Quantile And Linear Regression:**

A screenshot of a computer

Description automatically generated

**Intercept (const):**

* **Interpretation**: The intercept (const) is the predicted value of the dependent variable (in this case, house prices) when all the independent variables are set to zero. It serves as a baseline or starting point for the prediction.

The **const** term shows a negative intercept for both the linear and quantile regression models, indicating that, without considering the effect of any features, the baseline predicted house price would be negative. This situation implies that the actual price predictions are heavily dependent on the features provided and that the intercept alone isn't meaningful in isolation. Instead, it highlights how the model adjusts to the presence of other variables, leading to meaningful, positive predictions when those features are factored in.

**Summary:**

The table provides a comparative view of how different features impact house prices in linear and quantile regression models. While the direction of impact (positive or negative) is consistent across models, the magnitude of these impacts and the confidence intervals show some variability, reflecting the differences between the mean-based and quantile-based approaches. This analysis is crucial for understanding the nuances in the relationship between housing characteristics and prices.

**4.4 Decision Tree Regression:**

A machine learning technique called decision tree regression divides the dataset into smaller groups according to feature values to predict continuous values. This non-parametric model works in a manner like a tree structure, with the leaf nodes representing the expected continuous output and each inside node representing a decision based on a specific attribute. Decision Tree Regression is valued for its capacity to handle both numerical and categorical data, as well as its straightforward interpretability.

**4.4.1 Training a Decision Tree Regression Model:**

Based on the features in the dataset, a Decision Tree Regression model was used to forecast the target variable. A dataset containing the input features and the matching output values was used to train the model. Recursively dividing the data into subsets according to various feature values, the model constructed a decision tree during the training phase with the goal of maximising prediction accuracy at each node. After being trained, the model was used to the test set of data, estimating the target variable using the tree structure.

**4.5 Random Forest Regression:**

By merging the results of several decision trees, Random Forest Regression is an ensemble learning technique that improves prediction accuracy. By averaging the predictions from multiple trees, this method lowers the risk of overfitting and boosts prediction reliability. In order to reduce errors, each tree in a Random Forest is built using a technique called bagging (bootstrap aggregation). The final forecast is the mean of the predictions from each individual tree.

**4.5.1 Training a Random Forest Model:**

The Random Forest Regression model was trained to predict the target variable using the features of the dataset. The training process involved using a dataset that contained both the independent variables and the target values. During training, the model generated an ensemble of decision trees by sampling different subsets of the data and features. Each tree was trained independently, and their predictions were averaged to produce a final output. This trained model was subsequently used to predict the target variable in the test dataset by averaging the outputs of all the trees.

**4.6 XGBoost Regression:**

A potent and scalable gradient boosting solution, XGBoost Regression aims to increase processing speed and prediction accuracy. Extreme Gradient Boosting, or XGBoost, combines several weak learners—usually decision trees—in a sequential manner to create a powerful predictive model. The loss function, which gauges the discrepancy between expected and actual values, and the regularisation term, which penalises overly complicated models, are components of the objective function that the model optimises.

**4.6.1 Training an XGBoost Model:**

The XGBoost regression model was created using the XGBRegressor class from the XGBoost library. The model was initially trained on the dataset, with the independent variables used to predict the dependent variable. The fit method was employed to optimize the model's parameters. After training, the model was used to generate predictions on the test dataset, providing estimates of the target variable based on the features in the test data.

**4.7 Hyperparameter Tuning for Random Forest and XGBoost Using Grid Search:**

A critical stage in machine learning model optimisation is hyperparameter tuning, which is methodically modifying the hyperparameters to enhance model performance. A popular method for fine-tuning hyperparameters is Grid Search Cross-Validation, which involves testing several hyperparameter combinations to see which model performs best.

A thorough grid search was done to determine the best hyperparameters for the Random Forest Regression model. Several configurations, such as the number of estimators and the maximum number of features taken into account for each split, were tested during this procedure. The model was assessed using the negative mean squared error in the grid search, which also included five-fold cross-validation. The model was then trained and predictions were made using the best configuration, which was found to be 50 estimators with a maximum of four features per split.

Likewise, a grid search was employed to optimise the hyperparameters of the XGBoost regression model. The maximum depth of the trees, the learning rate, and the number of estimators were all examined with different values in the grid search. The model was then trained and predictions were produced using the ideal configuration, which had 200 estimators, a learning rate of 0.2, and a maximum tree depth of 5.

**4.8 Evaluation Metrics:**

Regression model performance is evaluated using evaluation metrics, which give a numerical indication of the model's ability to predict the target variable. Metrics like Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and R-squared provide important information about how accurate the model is.

**Coefficient of Determination (R²):**

**R²**, also known as the coefficient of determination, measures the proportion of variance in the dependent variable that is explained by the independent variables. It is calculated as: R² = 1 - rac{SS\_{res}}{SS\_{tot}}, where SS\_{res} is the residual sum of squares, and SS\_{tot} is the total sum of squares. This metric provides a clear understanding of how well the model fits the data.

**Mean Absolute Error (MAE):** Regardless of direction, MAE calculates the average magnitude of mistakes in a series of forecasts. It is computed as:

**Root Mean Squared Error (RMSE):** By taking the square root of the average of the squared errors, RMSE calculates the average magnitude of the mistakes. It is computed as follows .

# Chapter **5. Discussion of Results**

The metrics R-squared (R2), Mean Absolute Error (MAE), and Root Mean Squared Error (RMSE) are used to assess each regression model's performance. The outcomes of every model are shown in the table that follows.

**Quantile Regression** yielded a Mean Absolute Error of 44,393.51, a Root Mean Squared Error of 64,285.92, and an R² Score of 0.5672. These results indicate a moderate level of prediction accuracy, with relatively high error in both absolute and squared terms and a moderate coefficient of determination.

**Linear Regression** achieved a Mean Absolute Error of 46,017.68, a Root Mean Squared Error of 62,765.73, and an R² Score of 0.5874. this model demonstrated slightly less accuracy compared to **Quantile Regression**, with a marginally higher MAE, it has a lower RMSE and a higher R² Score, indicating it explains a greater proportion of the variance in the data.

**Decision Tree Regression** resulted in a Mean Absolute Error of 41,533.81, a Root Mean Squared Error of 62,406.15, and an R² Score of 0. 5921.The Decision Tree model outperformed Quantile and Linear Regression in terms of error metrics, with a small improvement in R² Score.

**Random Forest Regression** exhibited a Mean Absolute Error of 29,413.11, a Root Mean Squared Error of 43,606.92, and an R² Score of 0.8008. This model significantly outperformed the previously mentioned models, indicating a substantial reduction in prediction errors and a high R² Score, reflecting a better model fit.

XGBoost demonstrated the best performance with a Mean Absolute Error of 27,622.86, a Root Mean Squared Error of 40,550.39, and an R² Score of 0.8278. This model has the lowest errors in all metrics and the best R² Score, indicating its usefulness in accurately predicting housing prices.

A table with numbers and symbols

Description automatically generated

When comparing the performance of the regression models, the Random Forest and XGBoost models demonstrated the highest R-squared scores, indicating superior predictive accuracy relative to other models. Moreover, hyperparameter tuning further enhanced these models' performance. The tuned Random Forest model improved its R-squared score to 0.8160, and the tuned XGBoost model achieved the highest R-squared score of 0.8331. This shows that fine-tuning the hyperparameters resulted in more accurate predictions and improved model performance.

A graphical representation of the regression models was employed to compare the R-squared

scores, providing a clearer understanding of each model's performance.

A graph of different colored bars

Description automatically generated

Figure 2-Comparison of Models R2 Score

**Predicted vs Actual Median House Values using XGBoost (Tuned):**

A diagram of a blue line

Description automatically generated

The above plot is a scatter plot on x-axis there are actual values and on y-axis there are predicted values of median house values. The red dashed line represents the prediction sthat perfectly matched actual values.

The relationship between the median house values—actual and predicted—is displayed in this scatter plot. With the actual values on the x-axis and the anticipated values on the y-axis, each point represents a prediction produced by your model. The line where the actual values and the predictions would coincide exactly (i.e., where predicted = actual) is indicated by the red dashed line.

**5.2 Observations:**

* **Positive Correlation:** The upward trend suggests that your model is generally able to predict the house values correctly, as higher actual values correspond to higher predicted values.
* **Variance:** There is some spread around the line, indicating that while many predictions are close, some are off. The spread increases with higher house values, which suggests that the model might be less accurate for higher-priced houses.
* **Outliers:** Some points deviate significantly from the red line, particularly on the higher end of the value range, which may indicate potential outliers or areas where the model's predictions could be improved.

### **5.3 Discussion of Machine Learning Model Results:**

#### 1. **Model Performance Overview**

#### • The study used a variety of machine learning models, such as Gradient Boosting Machines, Random Forests, Decision Trees, Linear Regression, and Quantile Regression, to forecast housing prices.

* • Performance measures: Root Mean Square Error (RMSE), Mean Absolute Error (MAE), and R-squared were among the important performance measures used to assess the models. These measures shed light on the models' dependability and accuracy in estimating home prices.

#### 2. **Linear Regression Model**

* **Performance:** The linear regression model served as a baseline for comparison with more complex models. Although it provided a straightforward interpretation of the relationship between features and the target variable, its predictive performance was relatively limited due to the model's inability to capture non-linear relationships.
* **Findings: The linear regression model, despite its simplicity, revealed important factors such as categories for ocean proximity and median income. However, as seen by the lower R-squared value in comparison to more sophisticated models, it had trouble capturing the entire complexity of the data.**

#### 3. **Decision Tree Model**

* **Performance:** The decision tree model improved upon linear regression by capturing non-linear relationships between features and house prices. However, it was prone to overfitting, especially given the high variance observed in the training and testing errors.
* **Insights:** The model emphasized the importance of certain features such as median\_income and housing\_median\_age, but the lack of generalization reduced its effectiveness when applied to new data.

#### 4. **Random Forest Model**

* **Performance:** The random forest model significantly improved prediction accuracy, with a lower RMSE and higher R-squared value compared to both linear regression and decision trees. The ensemble approach of random forests helped mitigate overfitting and provided more stable predictions.
* **Insights:** Feature importance analysis showed that median\_income, latitude, and longitude were the most influential predictors, aligning with the findings from the correlation heatmap. The model's ability to aggregate multiple decision trees resulted in better generalization.

#### 5. **XGBoosting Machines:**

* **Performance:** XGBoost further enhanced prediction performance, which results in the lowest RMSE and highest R-squared among all models. The iterative boosting process allowed the model to correct errors from previous iterations, leading to more accurate predictions.

#### 6. **Quantile Regression Model**

* **Performance:** The quantile regression model was used to estimate the median house prices, providing robust predictions that are less sensitive to outliers. This approach was particularly useful for understanding the distribution of house prices and the impact of features across different quantiles.
* **Insights:** The results showed that while median income remained a strong predictor, the model also highlighted the varying effects of ocean\_proximity across different price levels. For example, proximity to the ocean had a more pronounced effect on higher quantiles, reflecting the premium prices associated with waterfront properties.

#### **5.4 Comparison and Final Model Selection**

* **Best Model:** Based on the evaluation metrics, the XGBoost Machine emerged as the best-performing model for house price prediction, offering the highest accuracy and robustness.
* **Trade-offs:** While XGBoost provided superior performance, the random forest model also offered strong results with better interpretability and rits not educed computational demands. Depending on the application, either of these models could be deployed for predicting house prices.

#### **5.5 Limitations and Future Work**

* **Overfitting:** Despite efforts to mitigate overfitting, some models (particularly decision trees) exhibited signs of overfitting, indicating the need for further tuning or regularization.
* **Feature Engineering:** To capture more complicated interactions, future work should investigate additional feature engineering techniques such interaction terms or non-linear transformations..
* **Model Interpretability:** Even if sophisticated models with excellent accuracy, such as GBM, might be difficult to understand due to their complexity. Subsequent research endeavours may concentrate on crafting more comprehensible models or employing methods such as SHAP values to elucidate model forecasts.

# 

# 

# 

Chapter 6. **Conclusion**

In this house price prediction project, various ML models were explored and evaluated to identify the most effective approach for accurately predicting housing prices. Among these models, **XGBoost** stood out as the top performer, delivering the highest accuracy. XGBoost’s strength lies in its ability to model complex, non-linear relationships within the data, effectively utilizing features like median\_income, ocean\_proximity, and latitude to predict housing prices with precision.

Other models, such as Random Forest, also demonstrated solid performance, with the added advantage of being more interpretable and less prone to overfitting. However, XGBoost’s superior accuracy made it the best choice for this task.

This project underscores the importance of model selection based on specific objectives. While XGBoost provided the most accurate predictions, its complexity and computational requirements must be balanced against the needs of the application. Future work could focus on optimizing these models further, exploring additional relevant features, and mitigating any remaining limitations to continue improving prediction accuracy.

Chapter 7. **REFERENCES**

1. **Truong, Q., Nguyen, M., Dang, H., and Mei, B. (2020). *Housing Price Prediction using Improved Machine Learning Techniques*. Procedia Computer Science, 174, pp. 433–442. DOI:**[**https://doi.org/10.1016/j.procs.2020.06.111**](https://doi.org/10.1016/j.procs.2020.06.111)**.**
2. **Zulkifley, N.H., Ibrahim, I., Rahman, S.A., and Ubaidullah, N.H.A. (2020). *A review of the literature on machine learning model for house price prediction*. International Journal of Computer Science and Modern Education, 12(6), pp. 46–54. DOI:**[**https://doi.org/10.5815/ijmecs**](https://doi.org/10.5815/ijmecs)**.**
3. **Phan, T.D. (2018). *Housing Price Prediction Using Machine Learning Algorithms: The Case of Melbourne City, Australia*. In: Proceedings of the 2018 International Conference on Machine Learning and Data Engineering (iCMLDE), Sydney, NSW, Australia, pp. 35–42. DOI:**[**https://doi.org/10.1109/iCMLDE.2018.00017**](https://doi.org/10.1109/iCMLDE.2018.00017)**.**
4. **Madhuri, C.R., Pujitha, M.V., and Anuradha, G. (2019). *House price prediction using regression techniques: A comparison study*. In: Proceedings of the 2019 International Conference on Smart Structures and Systems (ICSSS), IEEE, pp. 1–5. DOI:**[**https://doi.org/10.1109/ICSSS.2019.8882834**](https://doi.org/10.1109/ICSSS.2019.8882834)**.**
5. **Mayuri, Shaha S., Aditi, M., Mishra, N., Nerurkar, A., and Bhagat, S. (2023). *Machine Learning-Based Prediction of Home Prices*. SSRN. DOI:**[**https://doi.org/10.2139/ssrn.4413863**](https://doi.org/10.2139/ssrn.4413863)
6. **Xin, S., and Khalid, K. (2018). *Lasso and Ridge Regression used to model housing prices*. Engineering & Technology International Journal, 7, p. 498. DOI:**[**https://doi.org/10.14419/ijet.v7i4.30.22378**](https://doi.org/10.14419/ijet.v7i4.30.22378)**.**

Chapter 8**.Appendix:**

import pandas as pd

import seaborn as sb

import numpy as np

import matplotlib.pyplot as plot

from sklearn.preprocessing import OneHotEncoder

from sklearn.preprocessing import StandardScaler

from sklearn.model\_selection import train\_test\_split

import statsmodels.api as sm

import statsmodels.regression.quantile\_regression as Q\_reg

from statsmodels.regression.quantile\_regression import QuantReg

from sklearn.linear\_model import LinearRegression

from sklearn.ensemble import RandomForestRegressor

from xgboost import XGBRegressor

from sklearn.tree import DecisionTreeRegressor

from sklearn.model\_selection import GridSearchCV

from sklearn import metrics

from sklearn.metrics import r2\_score, mean\_absolute\_error, mean\_squared\_error

import warnings

warnings.filterwarnings("ignore")

housingData = pd.read\_csv("housing\_dataset.csv")

housingData.head()

plot.figure(figsize = (12,6))

sb.distplot(x = housingData['total\_bedrooms'], color = 'lightgreen')

plot.title(' total bedrooms Distribution')

plot.xlabel('Total Number of Bedrooms')

plot.ylabel('Frequency')

plot.show()

housingData['total\_bedrooms'].fillna(value = housingData['total\_bedrooms'].median(), inplace = True)

housingData.hist(bins=50, figsize=(15, 10))

plot.tight\_layout()

plot.show()

plot.figure(figsize = (12,6))

sb.distplot(x = housingData['median\_income'], color = 'tomato')

plot.title(' Median Income Distribution')

plot.xlabel('Income in Tens of Thousands Dollars')

plot.ylabel('Frequency')

plot.show()

# Change '<1H OCEAN' to '1Hocean' in the 'ocean\_proximity' column

# Replace '<1H OCEAN' with '1Hocean'

# Change '<1H OCEAN' to '1Hocean' in the 'ocean\_proximity' column

# Replace '<1H OCEAN' with '1Hocean'

housingData['ocean\_proximity'] = housingData['ocean\_proximity'].replace('<1H OCEAN', '1Hocean')

# Verify the change

housingData['ocean\_proximity'].unique()

# Get value counts for 'ocean\_proximity'

oceanProximityCounts = housingData['ocean\_proximity'].value\_counts()

# Create the bar plot

plot.figure(figsize=(10, 6))

barplot = sb.barplot(x=oceanProximityCounts.index, y=oceanProximityCounts.values,palette = "cubehelix")

# Add counts on top of the bars

for i, v in enumerate(oceanProximityCounts.values):

plot.text(i, v, str(v), ha='center', va='bottom')

# Add labels and title

plot.xlabel('Proximity')

plot.ylabel('Counts')

plot.title('Ocean Proximity Bar Plot')

# Show the plot

plot.show()

plot.figure(figsize = (12,6))

plot.title("Median House Values vs Median Income")

sb.scatterplot(data = housing\_data, y = 'median\_house\_value', x = 'median\_income', alpha = 0.5, color = 'g')

plot.ylabel("Median House Value")

plot.xlabel("Median Income")

housing\_data2 = pd.get\_dummies(housing\_data, columns=['ocean\_proximity'], drop\_first=False)

plot.figure(figsize=(12,6))

sb.heatmap(housing\_data2.corr(), annot=True, cmap="YlGnBu")

plot.figure(figsize = (12,6))

plot.suptitle('Ocean Proximity vs Median House Value')

sb.boxplot(data=housingData, x="ocean\_proximity", y="median\_house\_value", palette="Set2")

plot.show()

data = housingData

data["rooms/household"] = data["total\_rooms"]/data["households"]

data["bedrooms/household"] = data["total\_bedrooms"]/data["households"]

data["population/household"] = data["population"]/data["households"]

data[['rooms/household', 'bedrooms/household', 'population/household']].describe()

data = data.drop(['total\_rooms','total\_bedrooms','population', 'households'], axis = 1)

data.head()

data = data.loc[data['ocean\_proximity'] != 'ISLAND'].reset\_index(drop=True)

data['ocean\_proximity'].value\_counts()

encoder = OneHotEncoder()

encodedData = encoder.fit\_transform(data[['ocean\_proximity']]).toarray()

encodedDf = pd.DataFrame(encodedData, columns=encoder.get\_feature\_names\_out(['ocean\_proximity']))

#df = data.join(encoded\_df)

#df.reset\_index(drop=True, inplace=True)

combined\_df = data.join(encodedDf)

combined\_df.reset\_index(drop=True, inplace=True)

final\_df = combined\_df.drop(columns=['ocean\_proximity'])

final\_df.head()

plot.figure(figsize=(12,6))

sb.heatmap(final\_df.corr(), annot=True, cmap="YlGnBu")

X = final\_df.drop(["median\_house\_value"], axis=1)

y = final\_df["median\_house\_value"]

X.columns = [str(col).replace('[', '').replace(']', '').replace('<', '') for col in X.columns]

XTrain, XTest, yTrain, yTest = train\_test\_split(X, y, test\_size=0.25, random\_state=42)

continuousFeatures = ['longitude', 'latitude', 'housing\_median\_age', 'median\_income', 'rooms/household', 'bedrooms/household', 'population/household']

scaler = StandardScaler()

XTrain[continuousFeatures] = scaler.fit\_transform(XTrain[continuousFeatures])

XTest[continuousFeatures] = scaler.transform(XTest[continuousFeatures])

**Quantile Regression**

quantiles = [0.01, 0.1, 0.5, 0.95, 0.99]

def Qreg(q, XTrain, yTrain, XTest):

qrmodel = sm.QuantReg(yTrain, XTrain).fit(q=q)

coefs = pd.DataFrame()

coefs['param'] = qrmodel.params

coefs = pd.concat([coefs, qrmodel.conf\_int()], axis=1)

coefs['q'] = q

coefs.columns = ['beta', 'beta\_lower', 'beta\_upper', 'quantile']

pred = pd.Series(qrmodel.predict(XTest).round(2))

return qrmodel, coefs, pred

qr\_coefs = pd.DataFrame()

qr\_actual\_prediction = pd.DataFrame()

for qt in quantiles:

model, coefs, pred = Qreg(qt, XTrain, yTrain, XTest)

qr\_coefs = pd.concat([qr\_coefs, coefs])

qr\_actual\_prediction = pd.concat([qr\_actual\_prediction, pred], axis=1)

print(f"\nQuantile: {qt}\n")

print(model.summary())

qr\_coefs

qr\_actual\_prediction.columns = quantiles

qr\_actual\_prediction['actual'] = yTest

qr\_actual\_prediction['interval'] = qr\_actual\_prediction[0.99] - qr\_actual\_prediction[0.01]

qr\_actual\_prediction = qr\_actual\_prediction.sort\_values('interval').reset\_index(drop=True)

qr\_actual\_prediction

prediction

Quantile Regression Model Training

quantile = 0.5

def Qreg\_single(q, XTrain, yTrain, XTest):

qrModel = sm.QuantReg(yTrain, XTrain).fit(q=q)

prediction = pd.Series(qrModel.predict(XTest).round(2))

return prediction

qr\_pred = Qreg\_single(quantile, XTrain, yTrain, XTest)

# Compute and display the R² score for Quantile Regression

quantileR2 = metrics.r2\_score(yTest, qr\_pred)

print(f'\nR² score for QR: {quantileR2}')

# Compute and display the Mean Absolute Error (MAE) for Quantile Regression

quantileMAE = metrics.mean\_absolute\_error(yTest, qr\_pred)

print(f'\nMAE : {round(quantileMAE, 2)}')

# Compute and display the Mean Squared Error (MSE) for Quantile Regression

quantileMSE = metrics.mean\_squared\_error(yTest, qr\_pred)

print(f'\nMSE : {round(quantileMSE, 2)}')

# Compute and display the Root Mean Squared Error (RMSE) for Quantile Regression

quantileRMSE = np.sqrt(quantile\_mse)

print(f'\nRMSE: {round(quantileRMSE, 2)}')

**2. Linear Regression**

Coefficient Estimates

X\_trainLi = sm.add\_constant(XTrain)

modelLi = sm.OLS(yTrain, X\_trainLi).fit()

print(modelLi.summary())

Extracting Coefficient Estimates

coefficientsLi = model\_li.params

coefficientsLi

**Linear Regression Model Training:**

li\_model = LinearRegression()

li\_model.fit(XTrain, yTrain)

liPred = li\_model.predict(XTest)

r2 = metrics.r2\_score(yTest, liPred)

print(f'\nR-squared for LR: {r2}')

mae = metrics.mean\_absolute\_error(yTest, liPred)

print(f'\nMAE for LR: {round(mae, 2)}')

mse = metrics.mean\_squared\_error(yTest, liPred)

print(f'\nMSE for LR: {round(mse, 2)}')

rmse = np.sqrt(mse)

print(f'\nRMSE for LR: {round(rmse, 2)}')

**3. Decision Tree Regression**

dtModel = DecisionTreeRegressor()

dtModel.fit(XTrain, yTrain)

dtPred = dtModel.predict(XTest)

MAE = metrics.mean\_absolute\_error

MSE = metrics.mean\_squared\_error

r2\_score = metrics.r2\_score(y\_test, dt\_pred)

# Calculate Decision Tree Regression metrics

mseDtree = metrics.mean\_squared\_error(yTest, dtPred)

rmseDtree = np.sqrt(mseDtree)

maeDtree = metrics.mean\_absolute\_error(yTest, dtPred)

r2Dtree = metrics.r2\_score(yTest, dtPred)

print(f'\nDecision Tree Regression Metrics:')

print(f'R²: {r2Dtree:.4f}')

print(f'MAE: {maeDtree:.2f}'

print(f'MSE: {mseDtree:.2f}')

print(f'RMSE: {rmseDtree:.2f}')

**4. Random Forest Regression**

r2Rforest = metrics.r2\_score(yTest, rfPred)

maeRforest = metrics.mean\_absolute\_error(yTest, rfPred)

mseRforest = metrics.mean\_squared\_error(yTest, rfPred)

rmseRforest = np.sqrt(mseRforest)

print(f'\nRandom Forest Regression Metrics:')

print(f'R² Score: {r2Rforest:.4f}')

print(f'MAE: {maeRforest:.2f}')

print(f'MSE: {mseRforest:.2f}')

print(f'RMSE: {rmseRforest:.2f}')

**XGBoost:**

xgbModel = XGBRegressor()

xgbModel.fit(XTrain, yTrain)

xgbPred = xgbModel.predict(XTest)

# Calculate XGBoost Regression metrics

r2Xgb = metrics.r2\_score(yTest, xgbPred)

maeXgb = metrics.mean\_absolute\_error(yTest, xgbPred)

mseXgb = metrics.mean\_squared\_error(yTest, xgbPred)

rmseXgb = np.sqrt(mseXgb)

# Display the metrics

print(f'\nXGBoost Regression Metrics:')

print(f'R² Score: {r2Xgb:.4f}') # R² with 4 decimal places

print(f'MAE: {maeXgb:.2f}') # MAE with 2 decimal places

print(f'MSE: {mseXgb:.2f}') # MSE with 2 decimal places

print(f'RMSE: {rmseXgb:.2f}') # RMSE with 2 decimal places

**Fine Tuning Models:**

randomForestModel = RandomForestRegressor()

param\_random = {

'n\_estimators': [15, 30, 50],

'max\_features': [4, 8, 12, 16]

}

gridSearchRandom = GridSearchCV(estimator = randomForestModel, param\_grid = param\_random, cv=5, scoring='neg\_mean\_squared\_error')

gridSearchRandom.fit(XTrain, yTrain)

bestRandomModel = gridSearchRandom.best\_estimator\_

bestRandomModel

randomForest = RandomForestRegressor(n\_estimators=50, max\_features=4)

randomForest.fit(XTrain, yTrain)

randomForestPred = randomForest.predict(XTest)

# Calculate Random Forest (Tuned) metrics

r2\_RForestTuned = metrics.r2\_score(yTest, randomForestPred)

mae\_RForestTuned = metrics.mean\_absolute\_error(yTest, randomForestPred)

mse\_RForestTuned = metrics.mean\_squared\_error(yTest, randomForestPred)

rmse\_RForestTuned = np.sqrt(mse\_RForestTuned)

# Display the metrics

print(f'\nRandom Forest (Tuned) Regression Metrics:')

print(f'R² Score: {r2\_RForestTuned:.4f}') # R² with 4 decimal places

print(f'MAE: {mae\_RForestTuned:.2f}') # MAE with 2 decimal places

print(f'MSE: {mse\_RForestTuned:.2f}') # MSE with 2 decimal places

print(f'RMSE: {rmse\_RForestTuned:.2f}') # RMSE with 2 decimal places

xgboostModel = XGBRegressor()

param\_xgboost = {

'n\_estimators': [100, 200],

'learning\_rate': [0.01, 0.1, 0.2],

'max\_depth': [3, 5, 7]

}

gridSearchXgboost = GridSearchCV(estimator = xgboostModel, param\_grid = param\_xgboost, cv = 5, scoring = 'neg\_mean\_squared\_error')

gridSearchXgboost.fit(XTrain, yTrain)

bestXgboostModel = gridSearchXgboost.best\_estimator\_

bestXgboostModel

xgboost = XGBRegressor(n\_estimators=200, max\_depth=5, learning\_rate=0.2)

xgboost.fit(XTrain, yTrain)

xgboostPred = xgboost.predict(XTest)

# Calculate XGBoost (Tuned) metrics

r2\_XgbTuned = metrics.r2\_score(yTest, xgboostPred)

mae\_XgbTuned = metrics.mean\_absolute\_error(yTest, xgboostPred)

mse\_XgbTuned = metrics.mean\_squared\_error(yTest, xgboostPred)

rmse\_XgbTuned = np.sqrt(mse\_XgbTuned)

# Display the metrics

print(f'\nXGBoost (Tuned) Regression Metrics:')

print(f'R² Score: {r2\_XgbTuned:.4f}') # R² with 4 decimal places

print(f'MAE: {mae\_XgbTuned:.2f}') # MAE with 2 decimal places

print(f'MSE: {mse\_XgbTuned:.2f}') # MSE with 2 decimal places

print(f'RMSE: {rmse\_XgbTuned:.2f}') # RMSE with 2 decimal places

Evaluation Metrics DataFrame

# Compute the evaluation metrics

metrics\_data = {

'Model': [

'Quantile-Regression',

'Linear-Regression',

'Decision-Tree Regression',

'Random-Forest-Regression',

'Random-Forest-Regression (Tuned)',

'XGBoost',

'XGBoost (Tuned)'

],

'Mean Absolute Error': [

metrics.mean\_absolute\_error(y\_test, qr\_pred),

metrics.mean\_absolute\_error(y\_test, li\_pred),

metrics.mean\_absolute\_error(y\_test, dt\_pred),

metrics.mean\_absolute\_error(y\_test, rfPred),

metrics.mean\_absolute\_error(y\_test, randomForestPred),

metrics.mean\_absolute\_error(y\_test, xgbPred),

metrics.mean\_absolute\_error(y\_test, xgboostPred)

],

'Mean Squared Error': [

metrics.mean\_squared\_error(y\_test, qr\_pred),

metrics.mean\_squared\_error(y\_test, li\_pred),

metrics.mean\_squared\_error(y\_test, dt\_pred),

metrics.mean\_squared\_error(y\_test, rfPred),

metrics.mean\_squared\_error(y\_test, randomForestPred),

metrics.mean\_squared\_error(y\_test, xgbPred),

metrics.mean\_squared\_error(y\_test, xgboostPred)

],

'Root Mean Squared Error': [

np.sqrt(metrics.mean\_squared\_error(y\_test, qr\_pred)),

np.sqrt(metrics.mean\_squared\_error(y\_test, li\_pred)),

np.sqrt(metrics.mean\_squared\_error(y\_test, dt\_pred)),

np.sqrt(metrics.mean\_squared\_error(y\_test, rfPred)),

np.sqrt(metrics.mean\_squared\_error(y\_test, randomForestPred)),

np.sqrt(metrics.mean\_squared\_error(y\_test, xgbPred)),

np.sqrt(metrics.mean\_squared\_error(y\_test, xgboostPred))

],

'R2 Score': [

metrics.r2\_score(y\_test, qr\_pred),

metrics.r2\_score(y\_test, li\_pred),

metrics.r2\_score(y\_test, dt\_pred),

metrics.r2\_score(y\_test, rfPred),

metrics.r2\_score(y\_test, randomForestPred),

metrics.r2\_score(y\_test, xgbPred),

metrics.r2\_score(y\_test, xgboostPred)

]

}

# Convert metrics data to DataFrame with formatted values

Metrics\_output = pd.DataFrame(metrics\_data)

# Format the DataFrame values

Metrics\_output['Mean Absolute Error'] = Metrics\_output['Mean Absolute Error'].apply(lambda x: f'{x:.2f}')

Metrics\_output['Mean Squared Error'] = Metrics\_output['Mean Squared Error'].apply(lambda x: f'{x:.2f}')

Metrics\_output['Root Mean Squared Error'] = Metrics\_output['Root Mean Squared Error'].apply(lambda x: f'{x:.2f}')

Metrics\_output['R2 Score'] = Metrics\_output['R2 Score'].apply(lambda x: f'{x:.4f}')

Metrics\_output

**Model's R2 score**

Metrics\_output['R2 Score'] = pd.to\_numeric(Metrics\_output['R2 Score'])

# Plotting

plot.figure(figsize=(10,7))

ax = sb.barplot(x='Model', y='R2 Score', data=Metrics\_output, palette='rocket\_r')

# Add the R2 Score values on top of bars

for i, v in enumerate(Metrics\_output['R2 Score']):

ax.text(i, v, f'{value:.6f}', ha='center', va='bottom')

plot.xlabel('Models')

plot.ylabel('R² Score')

plot.title(" R² Score comparison between models")

plot.xticks(rotation=45)

plot.tight\_layout()

plot.show()

**Predicted vs Actual Median House Values using XGBoost (Tuned)**

plot.figure(figsize=(11, 6))

plot.scatter(yTest, xgboostPred, alpha=0.6)

plot.plot([yTest.min(), yTest.max()], [yTest.min(), yTest.max()], 'r--')

plot.xlabel('Actual Median House Value')

plot.ylabel('Predicted Median House Value')

plot.title('Predicted vs Actual Median House Values')

plot.show()

**Comparing Co-Efficient Values For Quantile And Linear Regression**

# Assuming 'data' is your DataFrame containing the dataset

# Log transformation on 'median\_income'

data['log\_median\_income'] = np.log(data['median\_income'])

# Update feature list

X = data.drop(["median\_house\_value"], axis=1)

X['log\_median\_income'] = np.log(X['median\_income'])

X = X.drop('median\_income', axis=1)

# Add a constant term

X = sm.add\_constant(X)

# Log-transform the target variable

data['logMedianHouseValue'] = np.log(data['median\_house\_value'])

y = data['logMedianHouseValue']

X = X.apply(pd.to\_numeric, errors='coerce') # Convert to numeric, forcing any problematic data to NaN

X = X.fillna(0) # filling null values with zero

# Also ensure y is numeric

y = pd.to\_numeric(y, errors='coerce')

y = y.fillna(0) # Handle missing values in the target

# Spliting data into train and test data

XTrain, XTest, yTrain, yTest = train\_test\_split(X, y, test\_size=0.25, random\_state=42)

# Fit lR model

linear\_model = sm.OLS(yTrain, XTrain).fit()

# Get linear regression coefficients

linear\_coefs = linear\_model.params

linear\_conf\_int = linear\_model.conf\_int()

linear\_conf\_int.columns = ['beta\_lower', 'beta\_upper']

# Combine coefficients and confidence intervals

linear\_regression\_results = pd.concat([linear\_coefs, linear\_conf\_int], axis=1).reset\_index()

linear\_regression\_results.columns = ['feature', 'beta\_linear', 'beta\_lower\_linear', 'beta\_upper\_linear']

def run\_quantile\_regression(X, y, quantile):

qr\_model = sm.QuantReg(y, X).fit(q=quantile)

coefs = pd.DataFrame({

'feature': X.columns,

'beta\_quantile': qr\_model.params,

'beta\_lower\_quantile': qr\_model.conf\_int()[0],

'beta\_upper\_quantile': qr\_model.conf\_int()[1],

})

return coefs

# Running quantile regression for 0.5 quantile (median)

quantile\_regression\_results = run\_quantile\_regression(X, y, 0.5)

# Intersection of Linear and Quantile Regression Coefficients

intersection = pd.merge(linear\_regression\_results, quantile\_regression\_results, on='feature', suffixes=('\_linear', '\_quantile'))

# Display the results

print("\nIntersection of Coefficients:")

print(intersection[['feature', 'beta\_linear', 'beta\_quantile', 'beta\_lower\_linear', 'beta\_upper\_linear', 'beta\_lower\_quantile', 'beta\_upper\_quantile']])