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**

PAVAN KUMAR CHIMATA, 22033170

**Supervisor:** MAN LAI TANG

**MSc Final Project Declaration**

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

It is my work except were indicated in the report.

I hereby give permission for the report to be made available on module websites provided the source is acknowledged.

**Abstract**

Housing is a fundamental human need, and investing in real estate has always been considered wise due to the general tendency of property values to increase or remain stable. The property market is influenced by numerous factors, making it difficult to precisely estimate house values while avoiding overpayment.

Manual evaluation methods are often flawed due to human errors, leading to inaccurate predictions. This study aims to identify the most significant factors influencing house prices through exploratory data analysis (EDA) and feature engineering, ultimately developing machine learning models capable of accurately predicting property prices using California housing price data.

The study employs several regression models, including quantile regression, linear regression, decision tree regression, random forest regression, and XGBoost regression. Coefficient estimates were obtained using quantile regression and linear regression to understand the impact of different features on median house value.

Among the models tested, the tuned XGBoost model demonstrated the highest accuracy, achieving an R² score of 0.8331 witfh a mean absolute error (MAE) of 27123.84 and a root mean squared error (RMSE) of 39923.81. These findings are especially useful in the real estate industry, where accurate predictions can have a considerable impact on investment decisions, pricing strategies, and market evaluations.

### Contents

1. **Abstract**
2. **Table of Contents**
3. **Introduction**
4. **Literature Review**
5. **Related Work**
6. **Method**

* Review of columns
* Data Type and Quality

1. **Dataset**
   * Description
   * Source
2. **Data Preprocessing**
   * Handling Missing Values
   * Removing Outliers
   * Feature Engineering
   * Encoding Categorical Variables
3. **Exploratory Data Analysis (EDA)**
   * Correlation Heatmap
4. **Machine Learning Models**
   * Quantile Regression
     + Estimation of Coefficients
   * Linear Regression
   * Decision Tree Regression
   * Random Forest Regression
   * XGBoost Regression
   * Hyperparameter Tuning
5. **Evaluation Metrics**
6. **Discussion of Results**
7. **Comparison of Models**
8. **Comparison with Other Papers**
9. **Applying Models to Real-World Scenarios**
10. **Improvements of Models**
11. **Limitations**
12. **Conclusion**

# **Introduction**

Housing is a basic human need, making real estate a crucial sector for investment, as property values generally increase or remain stable. However, the real estate market is complex, influenced by numerous factors, making it challenging to buy a home without overpaying. Accurate predictions are vital for stakeholders like buyers, sellers, and agents, who rely on various data points to make informed decisions. Traditional methods of property valuation, such as appraisals and market analyses, are often flawed due to human error and inefficiency, especially in volatile markets.

With the rise of big data and machine learning, real estate research has significantly evolved. These technologies can handle large datasets and uncover patterns that traditional methods might miss, offering more accurate property price predictions. Machine learning models continuously improve by updating with new data, ensuring predictions remain relevant over time.

The primary goal of this research is to identify the key factors influencing house prices through thorough exploratory data analysis (EDA) and feature engineering. By understanding these factors, the project aims to develop machine learning models that accurately predict property prices using historical data. This involves selecting relevant features, training various models, and evaluating their performance to identify the most effective one. The research ultimately seeks to provide a reliable solution for predicting property prices, addressing a major challenge in the real estate industry.

# **Related Work**

Several studies have explored various machine learning techniques for predicting property prices, highlighting the effectiveness of combining strong regression models and data reduction techniques. Phan's case study on Melbourne used PCA for data optimization and SVM regression, showing improved accuracy in house price forecasting. Rana et al. (2020) found Decision Tree Regression to be the most accurate among several algorithms, achieving 99% accuracy. Lu et al. (2017) proposed a hybrid model combining Lasso Regression and Gradient Boosting, which improved prediction performance but required significant computational power. Other studies, such as those by Mu et al. (2014) and Gu et al. (2011), demonstrated the potential of hybrid machine learning approaches like Genetic Algorithms with SVM for enhancing prediction accuracy. Chen et al. (2017) focused on SVM's ability to handle complex housing market data, further refining predictions with Stepwise regression.

# **Dataset**

The dataset used in this project is obtained from the California Census Data, which is given by the US Census Bureau and available on Kaggle. This dataset offers an extensive feature set encompassing a wide range of attributes that could affect house values for every block group in California. The target variable, the median house value, is one of the dataset's 10 distinct features. The features offer a thorough understanding of California's housing market by addressing a wide range of geographic and socioeconomic issues. This data includes the population of the block group, the average number of rooms per house, the median age of the houses, and the median income of the residents in the block group. It also contains details like the latitude and longitude coordinates and the typical number of bedrooms per property. In this study, the dependent variable is the median house value. By representing the median value of houses in each block group, it offers a uniform standard by which to compare properties in various geographical locations.

About Dataset

**longitude:** A measure of how far west a house is; a higher value is farther west

**latitude:** A measure of how far north a house is; a higher value is farther north

**housingMedianAge:** Median age of a house within a block; a lower number is a newer building

**totalRooms:** Total number of rooms within a block

**totalBedrooms:** Total number of bedrooms within a block

**population:** Total number of people residing within a block

**households:** Total number of households, a group of people residing within a home unit, for a block

**medianIncome:** Median income for households within a block of houses (measured in tens of thousands of US Dollars)

**medianHouseValue:** Median house value for households within a block (measured in US Dollars)

**oceanProximity:** Location of the house w.r.t ocean/sea

**A screenshot of a graph

Description automatically generated**

Statistical Information:

A screenshot of a data

Description automatically generated

**Data Type of Each Feature:**

A screenshot of a computer

Description automatically generated

# **Data Preprocessing**

Data preprocessing involves cleaning and transforming the dataset to ensure it is suitable for model training. This includes handling missing values, normalizing numerical features, and encoding categorical variables.

## **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. Since the median is less influenced by outliers than the mean, it was chosen as a more accurate method for imputing the missing values. The median of the bedroom values was calculated to fill in the gaps.A screenshot of a computer code

Description automatically generatedA white background with black lines

Description automatically generated

## 

## **Feature Engineering**

Additional variables were created to improve the dataset and give the predictive model additional insightful features. The ratios of the total number of rooms, total number of bedrooms, and population to the number of households were computed. These additional variables, "rooms per household," "bedrooms per household," and "population per 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

## 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 was applied to transform the categorical data into a binary matrix, with each category represented by a separate binary feature. 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

# 

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

* PLOTTING DISTRIBUTION PLOT FOR MEDIAN INCOME:

A graph of a distribution of income

Description automatically generated

### Key Observations:

1. **Distribution Shape**:
   * The distribution is right-skewed, with most data points concentrated towards the lower end of the income scale.
   * There is a noticeable peak around an income value of 3 to 4.
2. **Tail**:
   * The distribution has a long tail extending towards higher income values, showing that while most households have a median income below 5, there are some with significantly higher incomes, up to around 15.
3. **Frequency**:
   * The y-axis represents the frequency of households for each income range, with the peak frequency being slightly above 0.25.

* PLOTTING DISTRIBUTION PLOT FOR TOTAL BEDROOMS:

A graph of distribution of bedroom

Description automatically generated

### Key Observations:

1. **Distribution Shape**:
   * The histogram shows the frequency of different values of total\_bedrooms.
   * The KDE line (smooth curve) helps visualize the probability density of the variable.
2. **Skewness**:
   * The distribution is right-skewed (positively skewed), meaning there are more houses with fewer bedrooms, and fewer houses with a very high number of bedrooms.
3. **Range**:
   * The number of bedrooms ranges from 0 to over 6,000.
   * Most of the houses have fewer than 1,000 bedrooms, with a peak (highest frequency) occurring around 400-600 bedrooms.

HISTOGRAM PLOT COMPARING EVERY FEATURE:

A group of blue and white graphs

Description automatically generated

### 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.

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

A diagram of different colored squares

Description automatically generated

*  **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, possibly due to the desirability and scarcity of such locations.
*  **Uniformity**: ISLAND properties tend to have more uniform house prices with fewer extreme outliers, possibly indicating a stable high-value market.

**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, suggesting that these locations are more accessible or affordable.
* **Rarity**: Homes on islands are extremely rare, and those near the ocean or bay are less common but likely more desirable and expensive based on their scarcity.
* **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

## CORRELATION HEATMAP:

1. **Median House Value vs. Median Income**: There is a strong positive correlation (0.69), indicating that higher median incomes are associated with higher median 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 typically have more bedrooms.
3. **Households and Total Rooms**: There is a strong positive correlation (0.92), indicating that areas with more households tend to have more rooms.
4. **Latitude and Longitude**: There is a strong 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**: There is a positive correlation with the median house value (0.11), 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 3-Correlation Heatmap

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

1. **Positive Correlation**: There is a clear positive correlation between median income and median house value. As the median income increases, the median house value also tends to increase.
2. **Cap on House Values**: There appears to be a cap on the median house values at around $500,000. This could indicate that the dataset includes some form of truncation or limit on the house prices. This capping is evident from the horizontal line at the top of the plot.
3. **Income Range**: The median income ranges mostly between 0 and 14 (likely normalized or scaled values).
4. **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).
5. **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

# **Machine Learning Models**

The dataset was divided into features and target variables, with the target variable being the median house value. After that, the dataset was split into training and testing subsets, with 25% of the data going into the testing subset.

To standardize the continuous features, normalization was applied. The training subset's continuous attributes were normalized to have a zero mean and a unit variance. Using the same transformation parameters as the training data, this standardization process was then applied to the testing subset.

The validity and accuracy of the model's performance assessment depend on the feature scaling being consistent throughout the training and testing datasets, which is ensured by this method.

## **Quantile Regression**

Quantile Regression is a statistical technique that extends the capabilities of traditional linear regression models. Quantile regression computes numerous quantiles of the conditional distribution, in contrast to Ordinary Least Squares (OLS) regression, which utilizes the independent variables to estimate the mean of the dependent variable. When the data has outliers or heteroscedasticity, this method gives a more comprehensive view of the relationship between the variables. Since a normally distributed error component with constant variance is assumed in traditional linear regression models, the goal is to estimate the conditional mean of a response variable. This approach may not be as effective when dealing with heteroscedasticity (varying variance) or a skewed response variable distribution, even though it performs well under the assumption of homoscedasticity (constant error variance). By enabling the computation of multiple quantiles (such as the median or other percentiles) of the response variable's conditional distribution, quantile regression gets around these limitations.

The core of Quantile Regression is estimating the conditional quantile functions of the response variable. The quantile function at a specified quantile level τ (where τ is a value between 0 and 1) denotes the τ-th quantile of the response variable, given a set of predictors X. This can be mathematically expressed as:

Here, β(τ) is a vector of regression coefficients corresponding to the τ-th quantile. In contrast to OLS, which minimizes the sum of squared residuals, Quantile Regression minimizes a sum with asymmetric penalties for over- and under-prediction. The optimization problem for estimating β(τ) can be formulated as:

In this expression, represents the check function, which weights the residuals differently based on whether they are above or below the estimated quantile. The indicator function 1u<0 equals 1 if u < 0 and 0 otherwise, allowing the model to assign varying importance to residuals, depending on the quantile being estimated.

### **Estimation of Coefficients:**

To investigate the relationship between the response and predictor variables across different points in the conditional distribution, Quantile Regression models were fitted for several quantiles: 0.01, 0.1, 0.5, 0.95, and 0.99. The Quantile Regression analysis was executed using a **Qreg** function, which takes as input the quantile level, the training data, and the test data. For each quantile, a regression model was trained using the training subset. The model coefficients, along with their confidence intervals, were extracted and recorded. The predicted values for the testing subset were also obtained.

The resulting coefficients from the quantile regression models provided insights into the effect of each predictor variable across different quantiles of the response variable distribution. The intervals between the 1st and 99th quantile predictions were computed to assess the range of expected values, offering a comprehensive view of the variability in the predicted median house values.

#### **Interpretation of Coefficient Estimates:**

The coefficient estimates and their confidence intervals for various quantiles indicate the impact of each predictor on housing prices across different points in the distribution. At the 0.01 quantile, variables such as longitude, latitude, and population per household have a negative effect on housing prices, while median income, rooms per household, and proximity to the ocean positively influence prices. At the 0.10 quantile, the negative impact of longitude and latitude is more pronounced, while median income and proximity to the ocean still have a positive effect. At the 0.50 quantile, the effects remain similar, with median income showing a strong positive influence, and bedrooms per household also contributing positively. At higher quantiles, such as 0.95 and 0.99, the effects of median income and proximity to the ocean become even more significant, while the negative impacts of longitude and latitude increase. Overall, median income and proximity to the ocean are consistently strong positive predictors of housing prices, while longitude, latitude, and population per household typically have negative effects.

### Quantile Regression Model Training

A quantile regression model was fitted particularly to the median quantile, which represents the 50th percentile. The training subset was used to train the quantile regression model, with the supplied quantile set corresponding to the median. The model was then run on the testing subset to provide predictions. These predictions are crucial for comprehending the data's fundamental trend, and they may be compared to actual observed values to determine the model's accuracy.A table with numbers and text

Description automatically generated

## **Linear Regression**

Linear regression is a fundamental statistical technique widely utilized in data analysis and predictive modeling. It is a way for representing the relationship between a dependent variable and one or more independent variables using a linear equation. This strategy is especially useful for understanding how variations in predictor variables affect the response variable, and it offers a simple way to make predictions. The fundamental goal of linear regression is to find the best-fitting straight line that minimizes the difference between actual values and those predicted by the model. The relationship between the dependent variable ‘y ‘and an independent variable ‘x’ is expressed by the following equation:

The term β0​ represents the y-intercept, which is the value of y when x=0. The coefficient β1​ is the slope of the line, indicating the change in y for a one-unit change in x. The term represents the error term, which is the difference between the observed and predicted values.

### Estimation of Coefficients

The coefficients β0​, β1​,…,βp in a linear regression model are estimated using the method of Ordinary Least Squares (OLS). This statistical method aims to find the values of these coefficients that minimize the sum of the squared residuals, thereby providing the best fit for the given data. The OLS method seeks to minimize the following objective function:

where yi are the observed values and ŷi are the predicted values from the model.

The Ordinary Least Squares (OLS) approach was used to estimate the linear regression model's coefficients. A constant term was added to the feature matrix, and it serves as the regression model's intercept. The OLS model was trained using the training dataset, which included the response variable as well as the feature matrix augmented with the constant term. After fitting the model, the estimated coefficients were obtained, which indicate the weights associated with each attribute. These coefficients represent the most accurate linear unbiased estimates of the connection between the dependent variable and the predictors. The coefficients are calculated by minimizing the residual sum of squares between the observed and predicted values from the linear model.

#### Interpretation of Coefficient Estimates

Both longitude and latitude have a negative impact on house prices, indicating that moving further along these coordinates decreases prices. Housing median age positively affects prices, meaning older houses tend to be valued higher. Median income has a strong positive effect, indicating that higher income levels are associated with higher house prices. The number of rooms per household negatively influences prices, while the number of bedrooms per household positively affects them. Population per household slightly decreases house prices. Proximity to the ocean significantly increases house prices, with houses being closer to the ocean, bay, or within one hour of the ocean being more valuable. Being inland has a smaller positive effect on prices.

A black and white text on a white background

Description automatically generated

### **Linear Regression Model Training**

To predict the target variable using the features in the dataset, a linear regression model was built. The response variable and the predictors were both included in the training dataset, which was used to train the model. By determining the ideal weights for each feature, the model was able to determine the link between the independent and dependent variables following the training phase. The model was used to create predictions on the test dataset when training was finished.

## **Decision Tree Regression**

Decision Tree Regression is a machine learning technique that divides data into subsets according to feature values to predict continuous outcomes. It is a non-parametric model that makes predictions using a structure resembling a tree. The target variable predictions are represented by the leaf nodes of the tree, whilst feature-based judgments are represented by the inner nodes. Because of its ease of interpretation and capacity to handle both categorical and numerical data, Decision Tree Regression is highly acclaimed.

### Decision Tree Regression Model Training

To predict the target variable based on dataset features, a decision tree regression model was implemented. The training dataset, which contained both the matched output values and the input feature sets, was utilized to train the model. Throughout the training phase, the model built a decision tree by recursively splitting the data into subsets based on feature values. This allowed it to optimize for the best predictions at each node. The test data was then fed into the model to generate predictions. Based on the feature values in the test data, the model used the decision tree to estimate the target variable, which is represented by these predictions.

## **Random Forest Regression**

Random Forest Regression is an ensemble learning technique for predicting continuous outcomes that combine numerous decision trees. It expands on the concept of decision trees by combining the predictions of many trees to increase accuracy and control overfitting. To create a robust and accurate prediction model, this technique uses bagging (bootstrap aggregation) and feature randomization principles. For regression problems, the Random Forest model's final prediction is the average of the predictions provided by each tree. This averaging procedure smooth out the forecasts and mitigates the effects of individual trees' mistakes.

### Random Forest Model Training

The target variable was predicted using a random forest regression model based on the dataset provided. The model was trained using the training dataset, which contained both input characteristics and target values. During the training process, the model generated an ensemble of decision trees by sampling subsets of the training data and features using a technique known as bagging. Each decision tree in the ensemble was trained independently, and the model combined their predictions to produce a single output. Following training, the model was utilized to predict the target variable in the test dataset. The predictions were made by averaging the outputs of all the individual decision trees in the ensemble, yielding a final predicted value for each observation in the test dataset.

## **XGBoost Regression**

XGBoost Regression is a highly efficient and scalable implementation of gradient boosting techniques that improves prediction performance and processing efficiency. XGBoost, which stands for extreme Gradient Boosting, uses gradient boosting concepts to create and merge numerous weak learners (usually decision trees) into a powerful predictive model. XGBoost Regression uses a boosting strategy in which models are constructed successively to repair the errors of previous models. The objective function in XGBoost is made up of two parts: the loss function and a regularization term. The loss function computes the difference between predicted and actual values, while the regularization term penalizes model complexity to prevent overfitting.

### XGBoost Model Training

The XGBoost regression model was built with the XGBRegressor class from the xgboost library. Initially, the model was trained on the training dataset, with the independent factors utilized to predict the dependent variable. The fit approach was used to estimate the model's optimal parameters. The predicted values were obtained by applying the model to the test dataset. These predictions are the model's estimates of the target variable based on the features in the test data.

# **Hyperparameter Tuning of Random Forest and XGBoost Using Grid Search**

To optimize a machine learning model, hyperparameter tuning is an essential step that entails methodically adjusting hyperparameters to boost model performance. Finding the optimal combination of hyperparameters that yield the best model performance on unknown data is the aim of effective tuning. One common method for adjusting hyperparameters is Grid Search Cross-Validation. With this approach, a grid of hyperparameter values must be specified for evaluation. The model must then be trained and verified for each combination of parameters.

The Random Forest Regression model was subjected to a thorough optimization process to determine the most effective hyperparameters. To analyze multiple model configurations, a grid search strategy was used, which involved systematically testing different values for the number of estimators and the maximum number of features examined for each split. This technique involved doing five-fold cross-validation and evaluating the model based on the negative mean squared error. The hyperparameter adjustment revealed that the ideal setup included 50 estimators and a maximum of four features per split. The model, with these parameters established, was then trained on the dataset and used to make predictions.

Similarly, the XGBoost regression model was modified using a grid search to determine the most effective hyperparameters. The search examined a variety of values for the number of estimators, the learning rate, and the maximum depth of trees. The optimization procedure found that the most effective configuration included 200 estimators, a 0.2 learning rate, and a maximum tree depth of 5. The model, using these ideal parameters, was trained on the dataset and used to make predictions.

# **Evaluation Metrics**

Evaluation metrics are essential for measuring the performance of regression models because they quantify how effectively the model predicts the target variable. Metrics such as R-squared, Mean Absolute Error (MAE), and Root Mean Squared Error (RMSE) provide information about the model's accuracy.

1. Coefficient of Determination (R2)

R2 also known as the coefficient of determination, is a statistical metric that quantifies the proportion of variance in the dependent variable that can be predicted using the independent variables. It is calculated as:

1. Mean Absolute Error (MAE)

Mean Absolute Error measures the average magnitude of errors in a set of predictions, without considering their direction. It is calculated as:

1. Root Mean Squared Error (RMSE)

Root Mean Squared Error quantifies the average magnitude of the errors by taking the square root of the average of the squared errors. It is calculated as:

# **Results**

To evaluate the performance of all regression models, the metrics R-squared (R2), Mean Absolute Error (MAE), and Root Mean Squared Error (RMSE) are utilized. The following table displays the results of all models.

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 4-Comparison of Models R2 Score

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

A diagram of a blue line

Description automatically generated

This scatter plot shows the relationship between the actual and predicted median house values. Each point represents a prediction made by your model, with the actual values on the x-axis and the predicted values on the y-axis. The red dashed line represents the line where the predictions would perfectly match the actual values (i.e., where predicted = actual).

**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.

### Discussion of Machine Learning Model Results

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

* The project aimed to predict house prices using various machine learning models, including **Linear Regression**, **Decision Trees**, **Random Forests**, **Gradient Boosting Machines**, and **Quantile Regression**.
* **Performance Metrics:** The models were evaluated using key performance metrics such as RMSE (Root Mean Square Error), MAE (Mean Absolute Error), and R-squared. These metrics provided insight into the accuracy and reliability of the models in predicting house 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.
* **Insights:** Despite its simplicity, the linear regression model highlighted significant predictors like median\_incomeand ocean\_proximity categories. However, it struggled with capturing the full complexity of the data, as evidenced by the lower R-squared value compared to more advanced models.

#### 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, achieving 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.
* **Insights:** XGBoost ability to focus on difficult-to-predict cases made it particularly effective in capturing the nuanced relationships in the data. However, this model is also more computationally intensive and requires careful tuning of hyperparameters to avoid overfitting.

#### 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.

#### **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.

#### **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:** Future work could explore additional feature engineering, such as interaction terms or non-linear transformations, to capture more complex relationships.
* **Model Interpretability:** While advanced models like GBM offer high accuracy, their complexity can make them harder to interpret. Future work could focus on developing more interpretable models or using techniques like SHAP values to explain model predictions.

# **Conclusion**

In this house price prediction project, various machine learning 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.

**REFERENCES:**

CODE:

import pandas as pd

import seaborn as sns

import numpy as np

import matplotlib.pyplot as plt

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")

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

housing\_data.head()

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

sns.distplot(x = housing\_data['total\_bedrooms'], color = 'lightgreen')

plt.title('Distribution of total bedrooms')

plt.xlabel('Number of Bedrooms')

plt.ylabel('Frequency')

plt.show()

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

housing\_data.hist(bins=50, figsize=(15, 10))

plt.tight\_layout()

plt.show()

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

sns.distplot(x = housing\_data['median\_income'], color = 'tomato')

plt.title('Distribution of Median Income')

plt.xlabel('Income')

plt.ylabel('Frequency')

plt.show()

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

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

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

# Verify the change

housing\_data['ocean\_proximity'].unique()

# Get value counts for 'ocean\_proximity'

ocean\_proximity\_counts = housing\_data['ocean\_proximity'].value\_counts()

# Create the bar plot

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

barplot = sns.barplot(x=ocean\_proximity\_counts.index, y=ocean\_proximity\_counts.values,palette = "cubehelix")

# Add counts on top of the bars

for index, value in enumerate(ocean\_proximity\_counts.values):

plt.text(index, value, str(value), ha='center', va='bottom')

# Add labels and title

plt.xlabel('Proximity')

plt.ylabel('Counts')

plt.title('Ocean Proximity Bar Plot')

# Show the plot

plt.show()

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

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

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

plt.ylabel("Median House Value")

plt.xlabel("Median Income")

#housing\_data2=housing\_data.iloc[:,:-1]

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

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

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

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

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

sns.boxplot(data=housing\_data, x="ocean\_proximity", y="median\_house\_value", palette="Set2")

plt.show()

data = housing\_data

data["rooms\_per\_household"] = data["total\_rooms"]/data["households"]

data["bedrooms\_per\_household"] = data["total\_bedrooms"]/data["households"]

data["population\_per\_household"] = data["population"]/data["households"]

data[['rooms\_per\_household', 'bedrooms\_per\_household', 'population\_per\_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()

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

encoded\_df = pd.DataFrame(encoded\_data, columns=encoder.get\_feature\_names\_out(['ocean\_proximity']))

df = data.join(encoded\_df)

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

df = df.drop(['ocean\_proximity'], axis = 1)

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

sns.heatmap(df.corr(), annot=True, cmap="YlGnBu")

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

y = df["median\_house\_value"]

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.25, random\_state=42)

continuous\_features = ['longitude', 'latitude', 'housing\_median\_age', 'median\_income', 'rooms\_per\_household', 'bedrooms\_per\_household', 'population\_per\_household']

scaler = StandardScaler()

X\_train[continuous\_features] = scaler.fit\_transform(X\_train[continuous\_features])

X\_test[continuous\_features] = scaler.transform(X\_test[continuous\_features])

**Quantile Regression**

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

def Qreg(q, X\_train, y\_train, X\_test):

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

coefs = pd.DataFrame()

coefs['param'] = qr\_model.params

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

coefs['q'] = q

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

pred = pd.Series(qr\_model.predict(X\_test).round(2))

return qr\_model, coefs, pred

qr\_coefs = pd.DataFrame()

qr\_actual\_prediction = pd.DataFrame()

for q in quantiles:

model, coefs, pred = Qreg(q, X\_train, y\_train, X\_test)

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

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

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

print(model.summary())

qr\_coefs

Quantile Regression Predictions with Intervals

qr\_actual\_prediction.columns = quantiles

qr\_actual\_prediction['actual'] = y\_test

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

Quantile Regression Model Training

quantile = 0.5

def Qreg\_single(q, X\_train, y\_train, X\_test):

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

prediction = pd.Series(qr\_model.predict(X\_test).round(2))

return prediction

qr\_pred = Qreg\_single(quantile, X\_train, y\_train, X\_test)

print(f'\nQuantile Regression R² score is {metrics.r2\_score(y\_test, qr\_pred)}')

print(f'\nQuantile Regression Mean Absolute Error is {round(metrics.mean\_absolute\_error(y\_test, qr\_pred), 2)}')

print(f'\nQuantile Regression Mean Squared Error is {round(metrics.mean\_squared\_error(y\_test, qr\_pred), 2)}')

print(f'\nQuantile Regression Root Mean Squared Error is {round(np.sqrt(metrics.mean\_squared\_error(y\_test, qr\_pred)), 2)}')

**2. Linear Regression**

Coefficient Estimates

X\_train\_li = sm.add\_constant(X\_train)

model\_li = sm.OLS(y\_train, X\_train\_li).fit()

print(model\_li.summary())

Extracting Coefficient Estimates

coefficients\_li = model\_li.params

coefficients\_li

**Linear Regression Model Training:**

li\_model = LinearRegression()

li\_model.fit(X\_train, y\_train)

li\_pred = li\_model.predict(X\_test)

print(f'\nLinear Regression R² score is {metrics.r2\_score(y\_test, li\_pred)}')

print(f'\nLinear Regression Mean Absolute Error is {round(metrics.mean\_absolute\_error(y\_test, li\_pred), 2)}')

print(f'\nLinear Regression Mean Squared Error is {round(metrics.mean\_squared\_error(y\_test, li\_pred), 2)}')

print(f'\nLinear Regression Root Mean Squared Error is {round(np.sqrt(metrics.mean\_squared\_error(y\_test, li\_pred)), 2)}')

**3. Decision Tree Regression**

dt\_model = DecisionTreeRegressor()

dt\_model.fit(X\_train, y\_train)

dt\_pred = dt\_model.predict(X\_test)

print(f'\nDecision Tree Regression R² score is {metrics.r2\_score(y\_test, dt\_pred)}')

print(f'\nDecision Tree Regression Mean Absolute Error is {round(metrics.mean\_absolute\_error(y\_test, dt\_pred), 2)}')

print(f'\nDecision Tree Regression Mean Squared Error is {round(metrics.mean\_squared\_error(y\_test, dt\_pred), 2)}')

print(f'\nDecision Tree Regression Root Mean Squared Error is {round(np.sqrt(metrics.mean\_squared\_error(y\_test, dt\_pred)), 2)}')

**4. Random Forest Regression**

rf\_model = RandomForestRegressor()

rf\_model.fit(X\_train, y\_train)

rf\_pred = rf\_model.predict(X\_test)

print(f'\nRandom Forest Regression R² score is {metrics.r2\_score(y\_test, rf\_pred)}')

print(f'\nRandom Forest Regression Mean Absolute Error is {round(metrics.mean\_absolute\_error(y\_test, rf\_pred), 2)}')

print(f'\nRandom Forest Regression Mean Squared Error is {round(metrics.mean\_squared\_error(y\_test, rf\_pred), 2)}')

print(f'\nRandom Forest Regression Root Mean Squared Error is {round(np.sqrt(metrics.mean\_squared\_error(y\_test, rf\_pred)), 2)}')

1. **XGBoost:**

xgb\_model = XGBRegressor()

xgb\_model.fit(X\_train, y\_train)

xgb\_pred = xgb\_model.predict(X\_test)

print(f'\nXGBoost R² score is {metrics.r2\_score(y\_test, xgb\_pred)}')

print(f'\nXGBoost Mean Absolute Error is {round(metrics.mean\_absolute\_error(y\_test, xgb\_pred), 2)}')

print(f'\nXGBoost Mean Squared Error is {round(metrics.mean\_squared\_error(y\_test, xgb\_pred), 2)}')

print(f'\nXGBoost Root Mean Squared Error is {round(np.sqrt(metrics.mean\_squared\_error(y\_test, xgb\_pred)), 2)}')

**Fine Tuning Models:**

Random Forest:

random\_forest\_model = RandomForestRegressor()

param\_random = {

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

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

}

grid\_search\_random = GridSearchCV(estimator = random\_forest\_model, param\_grid = param\_random, cv=5, scoring='neg\_mean\_squared\_error')

grid\_search\_random.fit(X\_train, y\_train)

best\_random\_model = grid\_search\_random.best\_estimator\_

best\_random\_model

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

random\_forest.fit(X\_train, y\_train)

random\_forest\_pred = random\_forest.predict(X\_test)

print(f'\nRandom Forest (Tuned) R² score is {metrics.r2\_score(y\_test, random\_forest\_pred)}')

print(f'\nRandom Forest (Tuned) Mean Absolute Error is {round(metrics.mean\_absolute\_error(y\_test, random\_forest\_pred), 2)}')

print(f'\nRandom Forest (Tuned) Mean Squared Error is {round(metrics.mean\_squared\_error(y\_test, random\_forest\_pred), 2)}')

print(f'\nRandom Forest (Tuned) Root Mean Squared Error is {round(np.sqrt(metrics.mean\_squared\_error(y\_test, random\_forest\_pred)), 2)}')

1. XGBoost

xgboost\_model = XGBRegressor()

param\_xgboost = {

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

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

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

}

grid\_search\_xgboost = GridSearchCV(estimator = xgboost\_model, param\_grid = param\_xgboost, cv = 5, scoring = 'neg\_mean\_squared\_error')

grid\_search\_xgboost.fit(X\_train, y\_train)

best\_xgboost\_model = grid\_search\_xgboost.best\_estimator\_

best\_xgboost\_model

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

xgboost.fit(X\_train, y\_train)

xgboost\_pred = xgboost.predict(X\_test)

print(f'\nXGBoost (Tuned) R² score is {metrics.r2\_score(y\_test, xgboost\_pred)}')

print(f'\nXGBoost (Tuned) Mean Absolute Error is {round(metrics.mean\_absolute\_error(y\_test, xgboost\_pred), 2)}')

print(f'\nXGBoost (Tuned) Mean Squared Error is {round(metrics.mean\_squared\_error(y\_test, xgboost\_pred), 2)}')

print(f'\nXGBoost (Tuned) Root Mean Squared Error is {round(np.sqrt(metrics.mean\_squared\_error(y\_test, xgboost\_pred)), 2)}')

Evaluation Metrics DataFrame

evaluation\_metrics = pd.DataFrame({

'Model': [

'Quantile Regression',

'Linear Regression',

'Decision Tree Regression',

'Random Forest Regression',

'Random Forest Regression (Tuned)',

'XGBoost',

'XGBoost (Tuned)'

],

'Mean Absolute Error': [

round(metrics.mean\_absolute\_error(y\_test, qr\_pred), 2),

round(metrics.mean\_absolute\_error(y\_test, li\_pred), 2),

round(metrics.mean\_absolute\_error(y\_test, dt\_pred), 2),

round(metrics.mean\_absolute\_error(y\_test, rf\_pred), 2),

round(metrics.mean\_absolute\_error(y\_test, random\_forest\_pred), 2),

round(metrics.mean\_absolute\_error(y\_test, xgb\_pred), 2),

round(metrics.mean\_absolute\_error(y\_test, xgboost\_pred), 2)

],

'Mean Squared Error': [

round(metrics.mean\_squared\_error(y\_test, qr\_pred), 2),

round(metrics.mean\_squared\_error(y\_test, li\_pred), 2),

round(metrics.mean\_squared\_error(y\_test, dt\_pred), 2),

round(metrics.mean\_squared\_error(y\_test, rf\_pred), 2),

round(metrics.mean\_squared\_error(y\_test, random\_forest\_pred), 2),

round(metrics.mean\_squared\_error(y\_test, xgb\_pred), 2),

round(metrics.mean\_squared\_error(y\_test, xgboost\_pred), 2)

],

'Root Mean Squared Error': [

round(np.sqrt(metrics.mean\_squared\_error(y\_test, qr\_pred)), 2),

round(np.sqrt(metrics.mean\_squared\_error(y\_test, li\_pred)), 2),

round(np.sqrt(metrics.mean\_squared\_error(y\_test, dt\_pred)), 2),

round(np.sqrt(metrics.mean\_squared\_error(y\_test, rf\_pred)), 2),

round(np.sqrt(metrics.mean\_squared\_error(y\_test, random\_forest\_pred)), 2),

round(np.sqrt(metrics.mean\_squared\_error(y\_test, xgb\_pred)), 2),

round(np.sqrt(metrics.mean\_squared\_error(y\_test, xgboost\_pred)), 2)

],

'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, rf\_pred),

metrics.r2\_score(y\_test, random\_forest\_pred),

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

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

]

})

evaluation\_metrics

**Model's R2 score**

plt.figure(figsize=(12,7))

ax = sns.barplot(x=evaluation\_metrics['Model'], y=evaluation\_metrics['R2 Score'], palette='rocket\_r')

# Add the R2 Score values on top of the bars

for index, value in enumerate(evaluation\_metrics['R2 Score']):

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

plt.xlabel('Models')

plt.ylabel('R2 Score')

plt.title("Comparison of Model's R2 Score")

plt.xticks(rotation=45)

plt.tight\_layout()

plt.show()

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

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

plt.scatter(y\_test, xgboost\_pred, alpha=0.5)

plt.plot([y\_test.min(), y\_test.max()], [y\_test.min(), y\_test.max()], 'r--')

plt.xlabel('Actual Median House Value')

plt.ylabel('Predicted Median House Value')

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

plt.show()