California Housing Prices

# Introduction

The California housing prices dataset contains information from the 1990 California census. The data has been obtained from Kaggle.com. The goal is to build a model to predict housing prices in California using the California census data. To achieve this goal, I will be using the hedonic model using multiple linear regression (MLR). Recently, many machine learning techniques are being used for the house price prediction as an alternative to the classical hedonic models. Random forest (RF) is one such technique; I will be using it as a regressor to predict house prices so that results from the random forest regressor and the classical hedonic model can be compared.

# Literature Review

California housing price dataset is being used to build a regression model for predicting house prices. This dataset has been used in a book [1] for demonstrating implementation of machine learning algorithms. Much work has been done in the domain of predicting house prices. Historically, the hedonic price models based on multiple linear regression have been used to predict house prices. The term hedonic was used to describe ‘‘the weighting of the relative importance of various components among others in constructing an index of usefulness and desirability” [2]. Rosen was the first to present the theory of hedonic pricing [3]. Simply put, a hedonic model is a regression model that is used to explain the price of a composite item as a function of its attributes. An item’s total price can be viewed as the sum of the price of all of its attributes. This means that the items prices can be regressed based on its characteristic and learn how each characteristic uniquely determines the overall price of the item [3]. Rosen describes an item as a vector of its characteristics [3]. In the case of a house, these attributes may include structural variables (e.g., number of bedrooms), neighborhood variables (e.g., population) and environmental variables (e.g., ocean proximity) [4]. The ordinary least square linear regression is the standard method used to build hedonic price models. The price prediction function can be represented as a regression function:

Y = X β + ε

Where Y represents a n x 1 vector of n house prices, X is a n x m matrix containing explanatory variables, β is a m x 1 vector of unknown regression coefficients and ε is a vector of the error terms [3].

While hedonic price models have been routinely used in modeling house prices, they have some limitations. They are good for building a straightforward relation between house price and various characteristics, but they may not address issues of outliers, non-linearity and independent variable interactions [5]. Alternatively, many machine learning techniques have been used in real estate price estimation which include artificial neural networks [5,6], decision trees [7], random forest [3] etc. In addition to multiple linear regression, I will be using the random forest technique to predict house prices.

Random Forest is a very powerful ensembling machine learning algorithm which works by creating multiple decision trees and then combining the output generated by each of the decision trees. Random forest use regression trees as the basic learning method. First the number of trees in the forest is decided then each tree is built from a separate bootstrap sample obtained from the dataset. The sampling method used to generate the bootstrap sample is called bagging; in which instances are selected form the dataset with replacement [3]. In addition to this, random forest also change how regression trees are constructed. In contrast to standard trees where all available attributes are used at each node to decide the best attribute to split on; random forest use a subset of attributes at each node to choose the best attribute to split on [8]. This strategy performs very well and removes the bias that may be present due to the decision tree model and also improves the predictive power significantly [9].

Random forest does not require a separate test set to evaluate model accuracy. When the bootstrap sample is selected with replacement there will be some instances of data (approximately one-third of the total instances) which are not in the bootstrap sample. These instances, called out-of-bag (OOB) instances, can be used to estimate the accuracy of the model [9]. Given that enough trees are grown in the random forest, the OOB estimation of error rate is quite accurate [8].

# Dataset

The California housing prices dataset was based on data from the 1990 California census. There are 20,640 instances in the dataset and each row represents one district. A district is the smallest geographical unit for which the US Census Bureau publishes sample data, typically with a population of 600 to 3,000 people [1]. There are 10 attributes of the data that will be used in the model building. A list of all the attributes with description is given in the table below:

|  |  |  |
| --- | --- | --- |
| **No.** | **Column Name** | **Description** |
|  | 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 |
|  | housing\_median\_age | Median age of a house within a district; a lower number is a newer building |
|  | total\_rooms | Total number of rooms within a district |
|  | total\_bedrooms | Total number of bedrooms within a district |
|  | population | Total number of people residing within a district |
|  | households | Total number of households, a group of people residing within a home unit, for a district |
|  | median\_income | Median income for households within a district of houses (measured in tens of thousands of US Dollars) |
|  | median\_house\_value | Median house value for households within a district (measured in US Dollars) |
|  | ocean\_proximity | Location of the house w.r.t ocean/sea |

The task is to build a model to predict the median\_house\_value. Using multiple linear regression, this will be the dependent variable. All other attributes are the independent/predictor variables and will be used for building the regression model. All attributes are numeric except the ocean\_proximity variable which is categorical and consists of the following categories:

* 1H Ocean
* Inland
* Near Ocean
* Near Bay
* Island

The dataset has been downloaded from <https://www.kaggle.com/camnugent/california-housing-prices/home>.

# Approach

Figure 1: Exploratory Data Analysis

## Exploratory Data Analysis

The source code for the project has been uploaded at <https://github.com/AailaM/Capstone>.

The first step is to get a feel of the type of data I’ll be dealing with.

### Overview of Data

This can be done by looking at the structure of the data. A first look at the data tells us that it contains 20640 observations and 10 variables. Of the 10 variables all are numeric except one i.e. ocean\_proximity which is categorical and consists of 5 categories. The five categories with the number of observations in each is given below.

<1H OCEAN INLAND ISLAND NEAR BAY NEAR OCEAN

9136 6551 5 2290 2658

A quick look to the top 5 rows of the data is given below:

| **longitude**  <dbl> | **latitude**  <dbl> | **housing**  **median**  **age**  <dbl> | **total**  **rooms**  <dbl> | **Total**  **bedrooms**  <dbl> | **population**  <dbl> | **House-holds**  <dbl> | **Median**  **income**  <dbl> | **Median**  **House**  **value**  <dbl> | **Ocean**  **proximity**  <fctr> |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| -122.23 | 37.88 | 41 | 880 | 129 | 322 | 126 | 8.3252 | 452600 | NEAR BAY |
| -122.22 | 37.86 | 21 | 7099 | 1106 | 2401 | 1138 | 8.3014 | 358500 | NEAR BAY |
| -122.24 | 37.85 | 52 | 1467 | 190 | 496 | 177 | 7.2574 | 352100 | NEAR BAY |
| -122.25 | 37.85 | 52 | 1274 | 235 | 558 | 219 | 5.6431 | 341300 | NEAR BAY |
| -122.25 | 37.85 | 52 | 1627 | 280 | 565 | 259 | 3.8462 | 342200 | NEAR BAY |

Each row represents a housing district. There are some missing values in the data. Precisely, there are 207 missing values in the total\_bedrooms variable.

### Histogram

A histogram of all the numeric variables is given below to get an idea of the data distribution. It can be seen that many of the histograms are tail heavy distribution; i.e. they extend farther to the right of the median than to the left.

Also, we can see that the variables have very different scales of measurement. For example, housing\_median\_age ranges from 1 to 52, median income ranges from 0.5 to 15, total\_rooms range from 2 to 39320 etc.

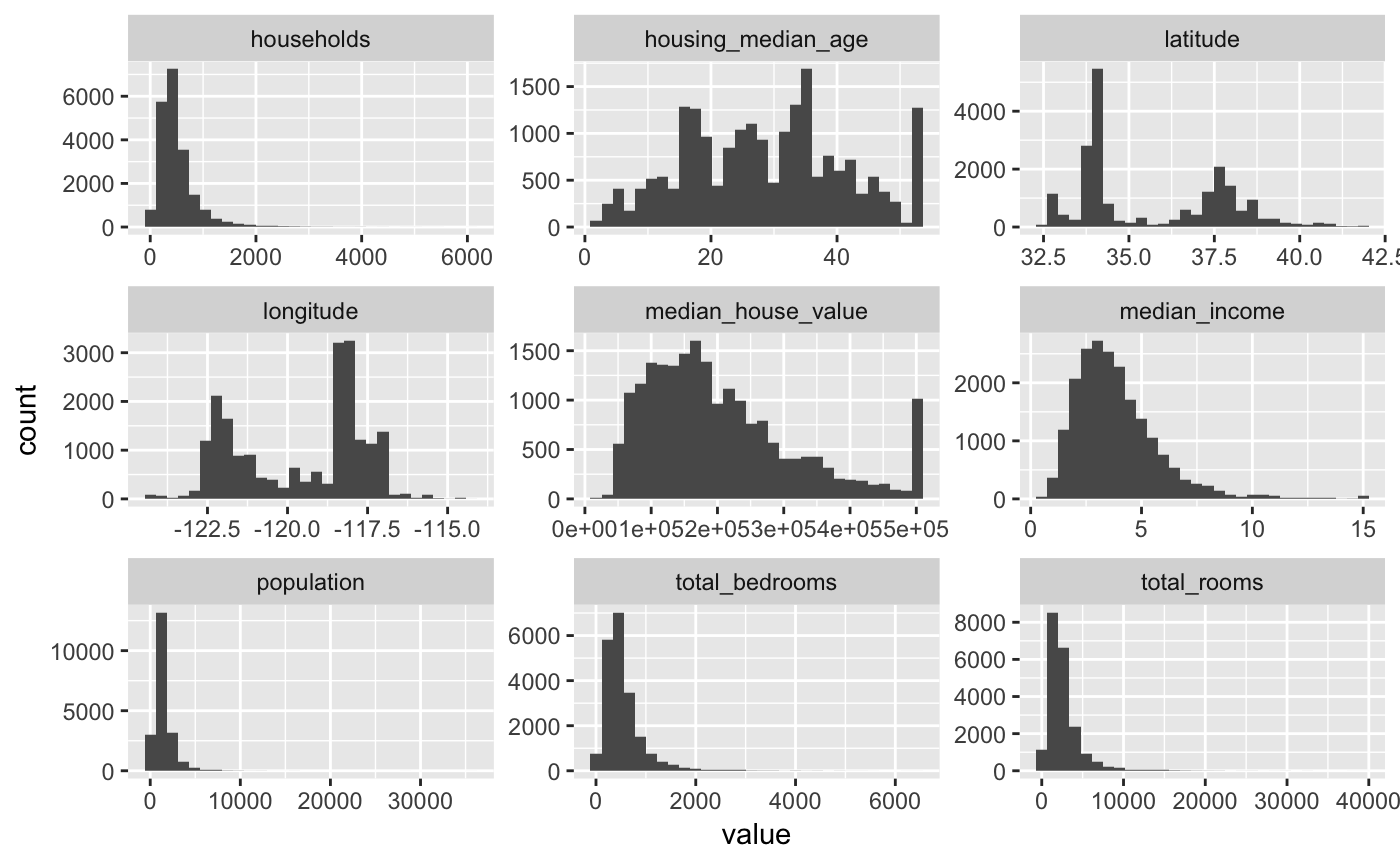


Figure 2 - Histogram of numeric attributes

### Attribute conversions

From the data we can see that many of the attributes contain information for the whole district.

1. Population contains information about the population in the entire district
2. Total bedrooms contain information about the total number of bedrooms in the district
3. Total rooms contain information about the total number of rooms in the district

Since we are interested in predicting the median price of a single house in a district it would be a good idea to convert those attributes to represent per household values. We have the total number of households in a district, so we’ll do the following conversions:

1. Convert district population to population per household i.e. dividing the population by the households in the district
   * People\_per\_household = population / households
2. Convert total rooms in the district to room per household i.e. dividing the total rooms by the households in the district
   * Mean\_rooms = total rooms/ households
3. Convert the total bedrooms to a ratio of bedroom per room in a house i.e. dividing the total bedroom by the total rooms in the district
   * Mean\_bedrooms = total bedrooms/ households

After making these conversions, the variables population, total\_rooms and total\_bedrooms have been removed from the independent variable and only their converted form is used for model building.

### Correlation

Now that we have data relating to each household the next interesting tool to use is to view correlation between the variables. Since the all the data is numeric (except ocean\_proximity which has been converted to type numeric for modeling purposes), pearson correlation coefficient is used. The result of the pearson correlation between each pair of variables is given in the plot below:

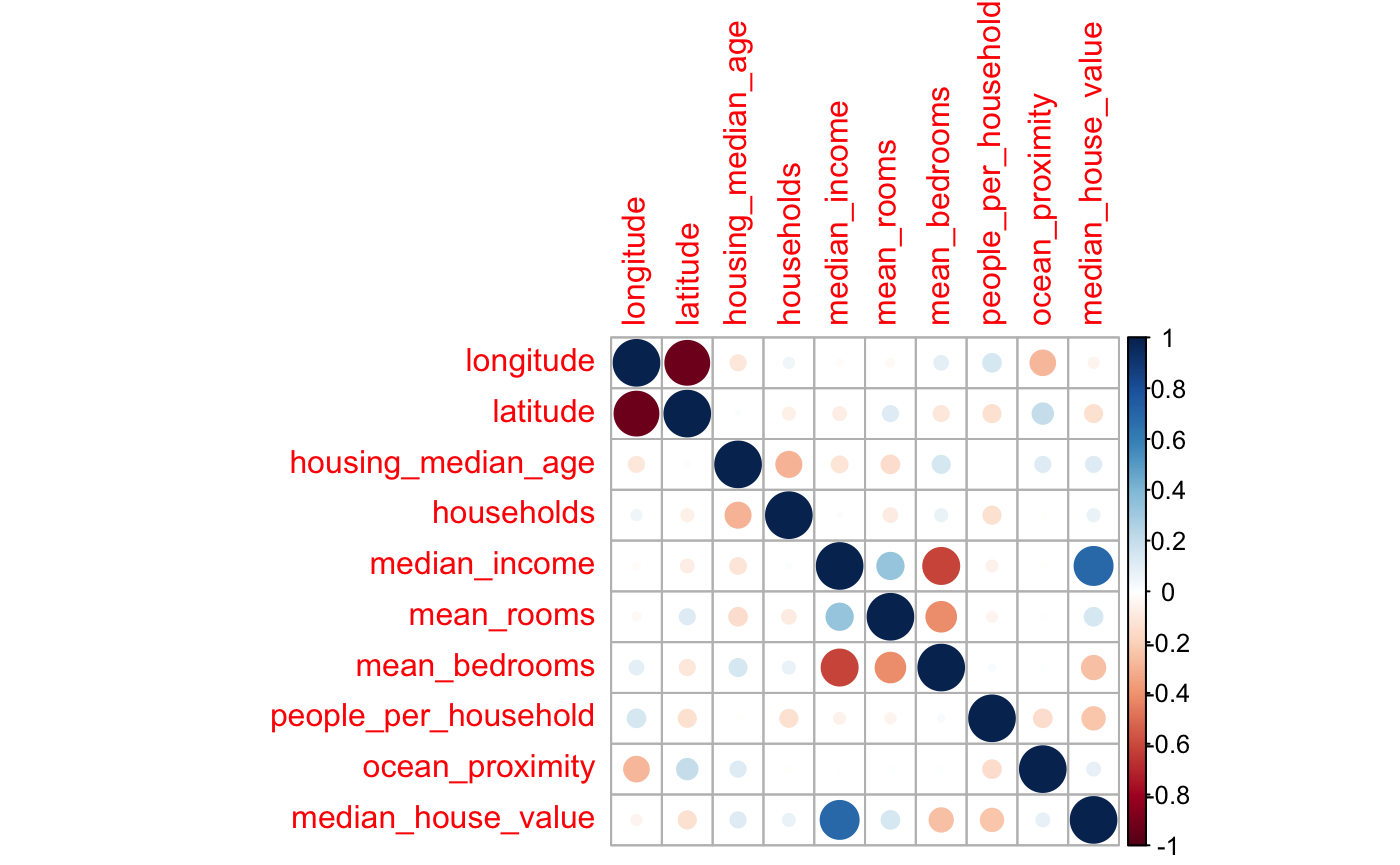


Figure 3 - Correlation plot

The correlation matrix plot in figure 2 above shows the correlation between all pairs of attributes. But we are more interested in the correlation of the dependent variable (median\_house\_value) with all the other attributes.

We can see that:

* there is a strong positive correlation between the house value and the median income of the district
* There is weak negative correlation between the median house value and mean bedrooms per household
* There is weak negative correlation between the median house value and mean number of people per household
* All other attributes have very weak correlation with the house value.

### Box plot

We can make a boxplot of all the independent variables to detect outliers. Figure 3 below shows us the boxplot of all the independent variables.

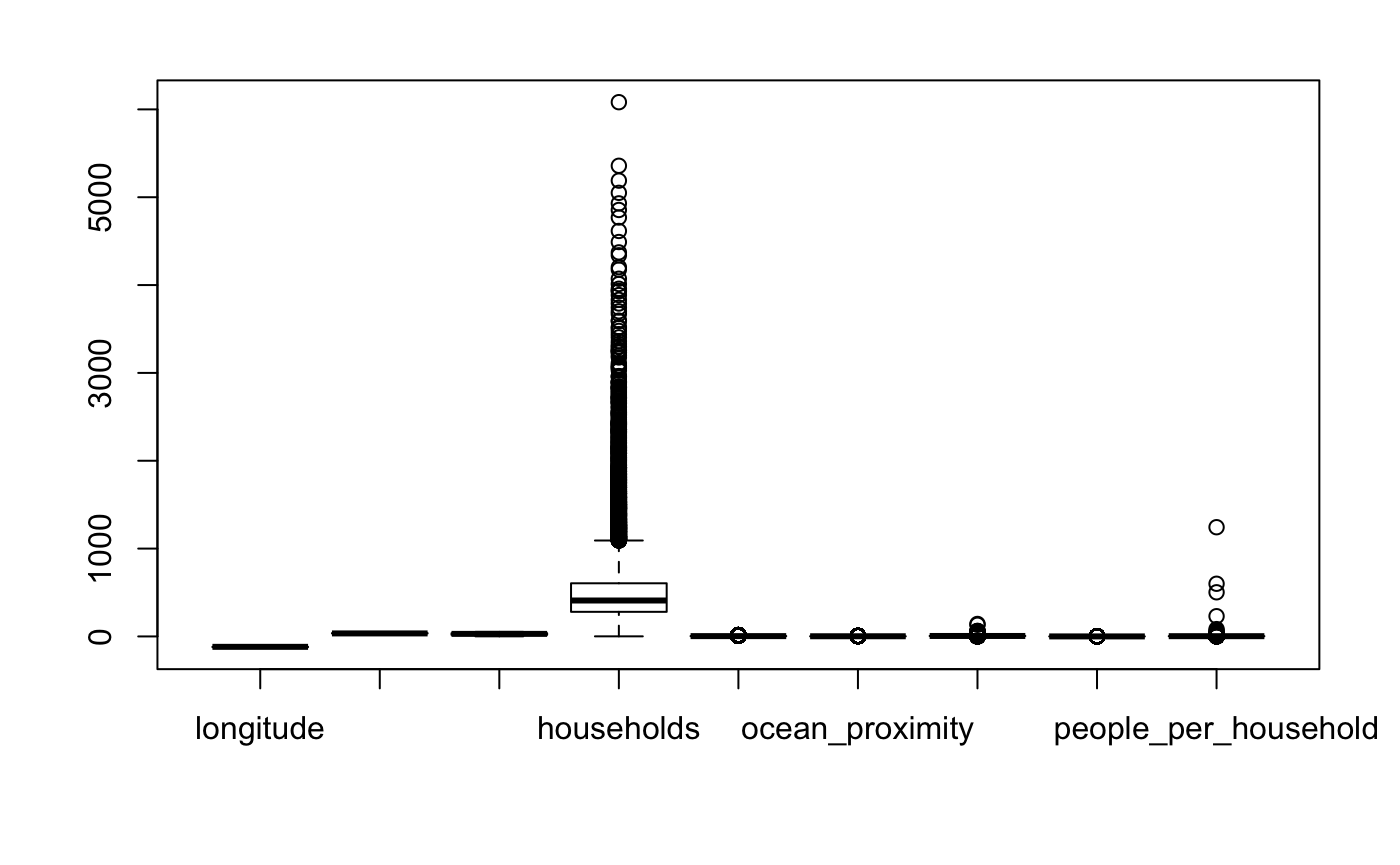


Figure 4 - Boxplot of independent variables

We can see that the households per district has a high number of outliers. We will handle this by removing the districts that contain more than a certain number of households. Also, we can see outliers in the people per household attribute. We can define a threshold of acceptable number of people per household and then remove observations that cross this threshold. A similar process is done for the mean rooms per household.

## Data Cleaning

After getting some idea of the data the next step is to clean the data and prepare it for model building. In this step thing like missing values, outliers etc. will be dealt with.

### Missing Values

There are 207 missing values in the total bedroom attribute. There are multiple options for dealing with missing values. I have used K-nearest neighbor imputation for the missing values. This method looks at the k nearest neighbors of the observation with a missing value and fills it in by taking the average of the values obtained by the k neighbors identified.

### Converting categorical data to numeric

We have one categorical attribute (ocean\_proximity) in the data. We can convert it to type numeric so that it can be used in the regression models. The conversion results in the following:

|  |  |
| --- | --- |
| Level | Label |
| <1H OCEAN | 1 |
| Inland | 2 |
| Island | 3 |
| Near Bay | 4 |
| Near Ocean | 5 |

### Feature scaling

The independent attributes have very varying scales. Feature scaling is required to standardize the range of data for the independent variables. This step is essential for better performance of the machine learning algorithm. Most machine learning algorithms do not perform well on data with variable scales. The normalization function I have used for bringing all independent variable values to a common scale is:

(x - min(x)) / (max(x) - min(x))

This results in all values being scaled to values from 0 to 1.

### Training – Test data split

Now the data is ready to be used for model building. The final step is now to split the data into training and test set. I have used 70% of the data for training and the remaining 30% for testing the model. The sampling procedure used is simple random sample. The training data will be used for model building. This will be further divided into training and validation sets used during cross validation. The test set will only be used after the best model has been chosen using the training and validation set.

## Train Model

Once the data is prepared it is time to build the model. I will be building two models using

* multiple linear regression
* regression tree
* random forest regressor

The basic steps for training a model will be the same for all three techniques.

1. Train and validate model using cross validation
2. Calculate the error on the validation set (RMSE)
3. Hyperparameter tuning to find the best set of parameters for the model given the data
4. Identify the best model
5. Run the model on the test data
6. Calculate the prediction error

Since we have a limited amount of data, we want to use the maximum we can for training the model. Using cross-validation we can ensure that all of the training data is used in the model building and also test the model during the training phase using the validation set. This will help us reduce and identify the problem of overfitting and also give an idea of how the model will perform on unseen data. Validation will help us evaluate the quality of the model during the training phase. Each and every point in the training set is used for validation once in the k-fold cross validation. While each point is used k-1 times for training the model. This helps us reduce underfitting and overfitting. In 10-fold cross validation there will be 10 validation sets. We will calculate the root-mean-square error for each of the 10 validation sets and then find their mean to obtain the mean error for the model.

I am using root-mean-square error (RMSE) as the measure of error for the model. RMSE is reported in the same units as the target variable. So, the lower the value of RMSE the better the model. I will first measure the mean RMSE value on the validation set during the training phase. Then I will try different variations of parameters for the model and calculate the mean RMSE for each of them. This will help me find the model with parameters that result in the least RMSE. These will be the best set of hyperparameters for the model and the resulting model will be our best regression model.

Once we have identified the hyperparameters that result in the best model, we can then run this model on the unseen test set. The predicted prices from the test data will then be compared with the observed prices in the test data to calculate the RMSE on test data.

### Multiple Linear Regression

Multiple linear regression is used to predict the price of the house using the all or a subset of the independent variables. It explains the relationship between the dependent variable (house price) and the independent variables by assigning coefficients to each of the independent variable. These coefficients represent the change in dependent variable for a one unit change in the independent variable while holding other independent variables in the model constant. Having all independent variable on a scale of 0 to 1, we can look at the regression coefficients to get an idea of which are the most important predictor variables.

As a first step we will build a linear regression model using the training set and all of the independent variables. This is achieved using 10-fold cross validation. At the end of the training we have a model and a set of validation errors. Once this is done, the next step is to try different parameters for the model to find the best set of parameters for the model given the data. The best model is the one with the least validation error.

With multiple linear regression the first important decision for model building is to choose the best set of independent variables for building the model. I have tried a few options and they are:

1. Using all independent variables to build the model
2. Using backward elimination to select the most important independent variables
3. Using forward selection to select the most important independent variables
4. Using glmnet package for feature selection
5. Taking a log the of the house price (dependent variable)

After running 10-fold cross validation on each of the above set of parameters the model with the least error on validation set is chosen as the best. The plot below shows a comparison of the validation error obtained from each of the different set of parameters for the linear model.

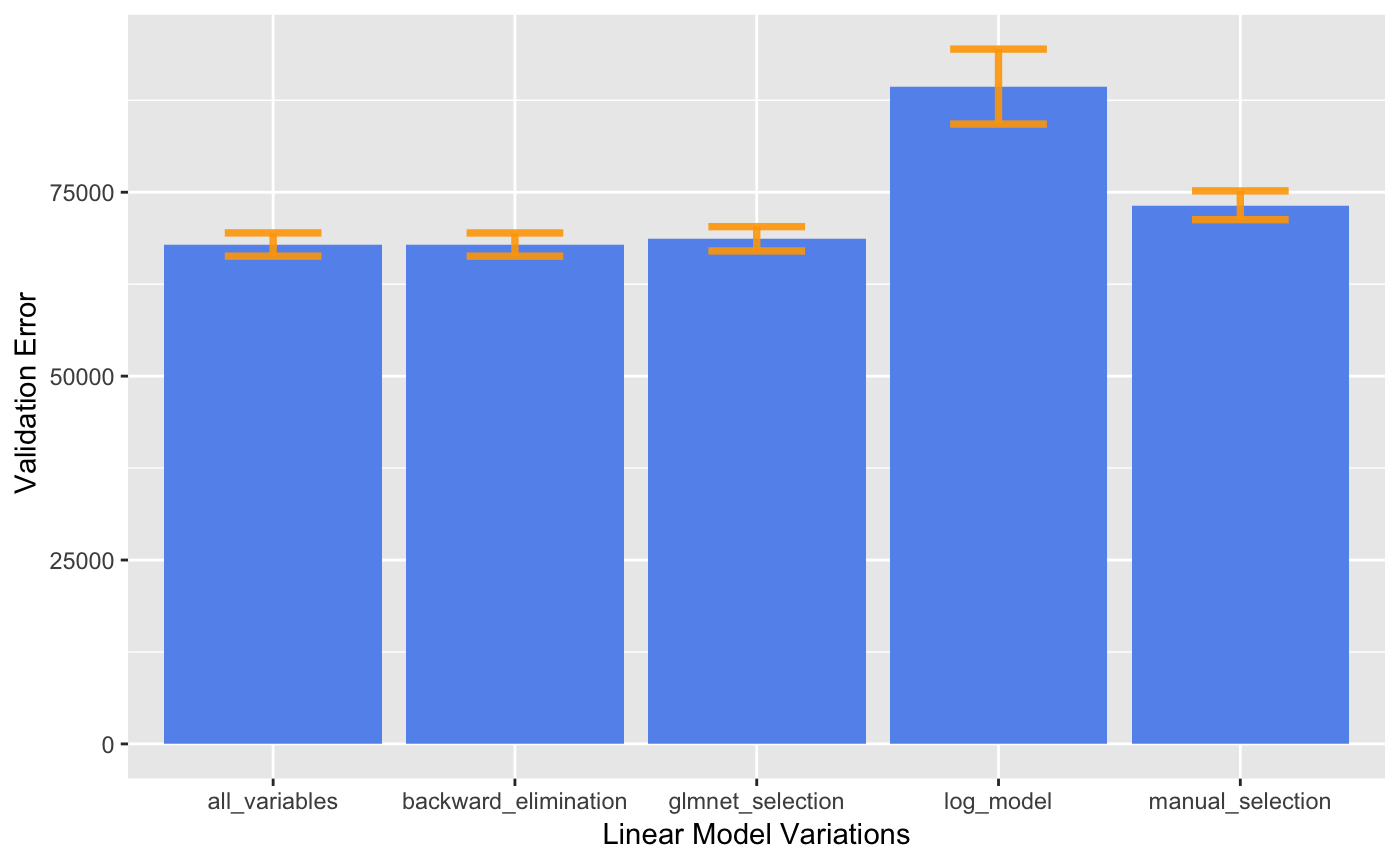


Figure 5 - Linear Model parameter selection and validation error

We can see that choosing all independent variables, choosing variables from backward elimination and glmnet feature selection result in very similar validation error. The least RMSE value obtained is 67888 using backward-elimination/ forward-selection parameter selection. However, the mean validation error is higher with manual feature selection and highest with model built using a log of the house prices.

In our case, the least error is obtained by using all independent variables except ocean\_proximity. This subset of independent variables is obtained from both backward elimination and forward selection. The coefficients from the linear model are given in the table below:

|  |  |
| --- | --- |
| **Independent Variable** | **Coefficient** |
| Intercept | 345019 |
| Longitude | -409363 |
| Latitude | -386118 |
| Housing\_median\_age | 57500 |
| Households | 47288 |
| Median\_income | 616443 |
| Mean\_rooms | 249772 |
| Mean\_bedrooms | 316563 |
| People per household | -476705 |

We can see that atitude, longitude and people per household have a negative coefficient, i.e. an increase in them results in a decrease in price. Other independent variables have positive coefficients. Also, we observe that district median income has the highest coefficient, implying that it has highest impact on the predicted price. People per household and longitude have the next highest coefficients after median income.

After choosing the best model, based on the least error from the validation set, now we are ready to test the model on our unseen test data that put aside earlier using random sampling.

### Regression Tree

Regression tree are a powerful non-linear tool for prediction. A regression tree is like a decision tree except that the target variable is numerical.

I have run 10-fold cross validation to generate different regression tree models by tweaking the hyperparameters for each model. The different variations of the model are given below:

1. Building the regression tree with the training data
2. Building regression tree and prune it
3. Building regression tree by setting control parameters “minsplit” and “cp”
4. Building regression with different values for control parameters “minsplit” and “cp”
5. Building regression tree with control parameters “minsplit” and “cp” and then prune tree

After running 10-fold cross validation on the above stated models, the model with the least validation set error is the one mentioned on no.4. the control parameters “minsplit” is the minimum number of observations that must exist in a node in order for a split to be attempted. The control parameter “cp” is the complexity parameter. A split that does not decrease the overall lack of fit by a factor of cp is not attempted. The main role of this parameter is to save computing time by pruning off splits that are obviously not worthwhile. While pruning is a method to avoid overfitting the data, we see that pruning the tree results in a decrease in the performance. The least value for RMSE obtained is 71271 using minsplit = 1000 and cp = 0.0001.

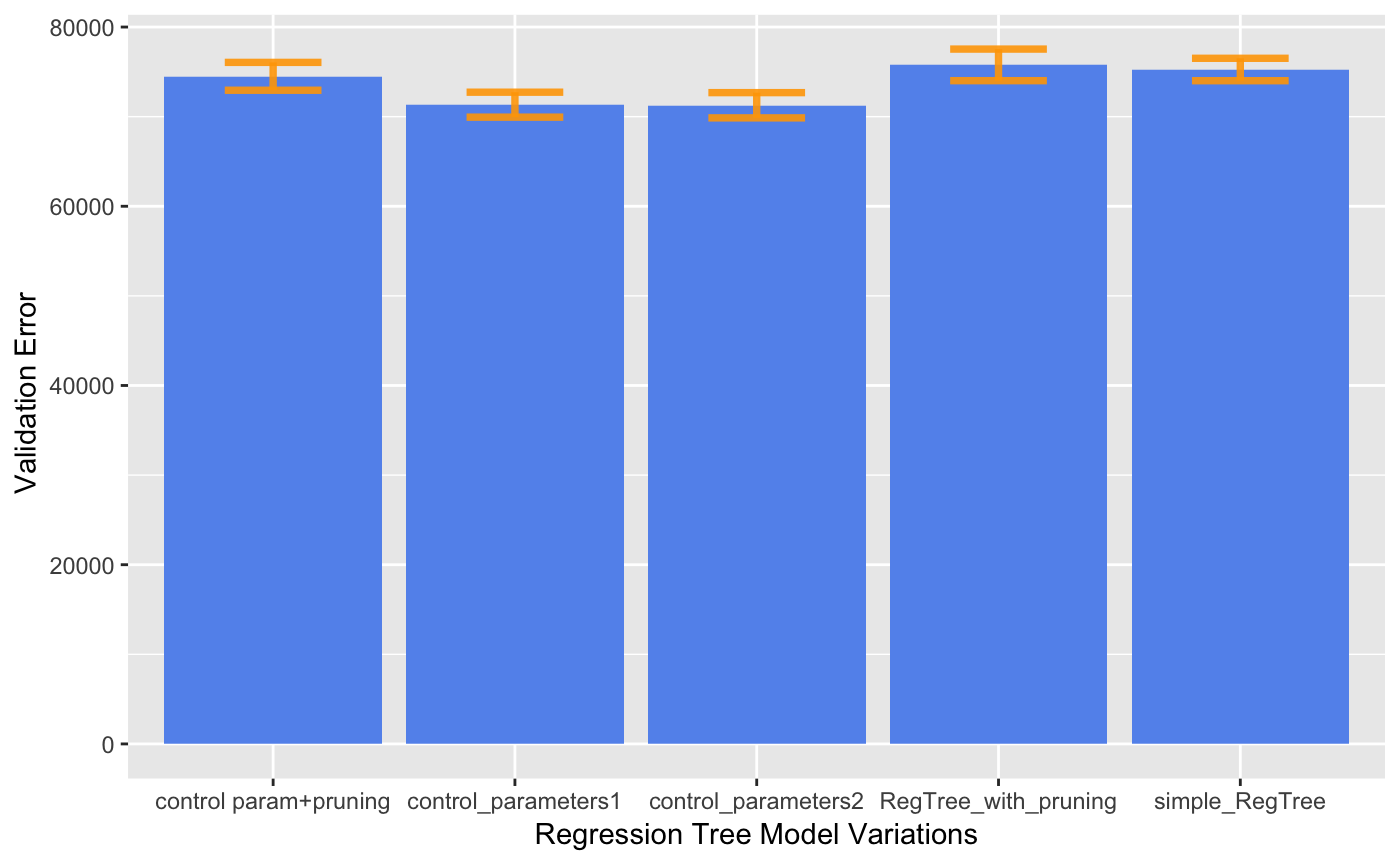


Figure 6 - Regression tree parameter selection and validation error

The resulting tree using the control parameters that give least value for RMSE is given below:

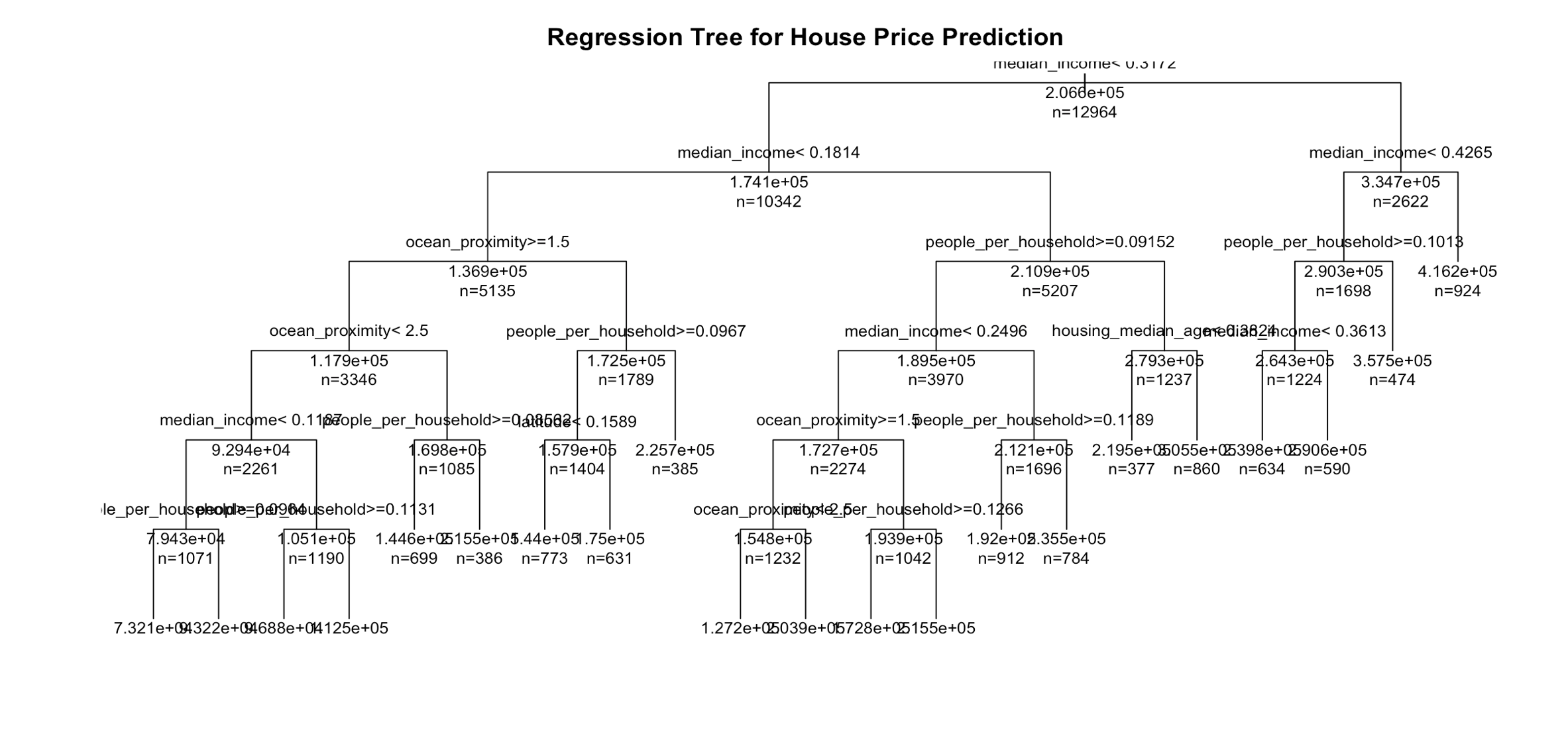


Figure 7 – Regression tree

We can see that district median income, people per household and ocean proximity are important parameters that the regression tree uses to split the data.

### Random Forest

Random forest is a powerful ensemble technique which improves the predictive accuracy by generating a large number of trees (based on random sample of variables) and then combining the results from each tree to get the final outcome.

Like the above two models, I have used 10-fold cross validation to train and validate the model. I have tried different hyperparameters to find the set of hyperparameters that result in the best model (i.e. one with the least RMSE value and low variability of validation error). The different hyperparameters setting used are listed below:

1. Random forest with default parameter values
2. Random forest with ntree = 300
3. Random forest with ntree = 300 and mtry = 5
4. Random forest with ntree = 400 and mtry = 4
5. Random forest with ntree = 800 and mtry = 3

The parameters ntree is the number of trees to grow in the random forest and mtry is the number of variables randomly sampled as candidate at each split. For regression problems the default value of ntree is 500 and mtry is p/3 where p is the number of independent variables used in the model. The larger the number of trees in the random forest the better the predictions. However, increasing values for mtry does not result in the same. The rmse values obtained from the different hyperparameter setting are plotted below:

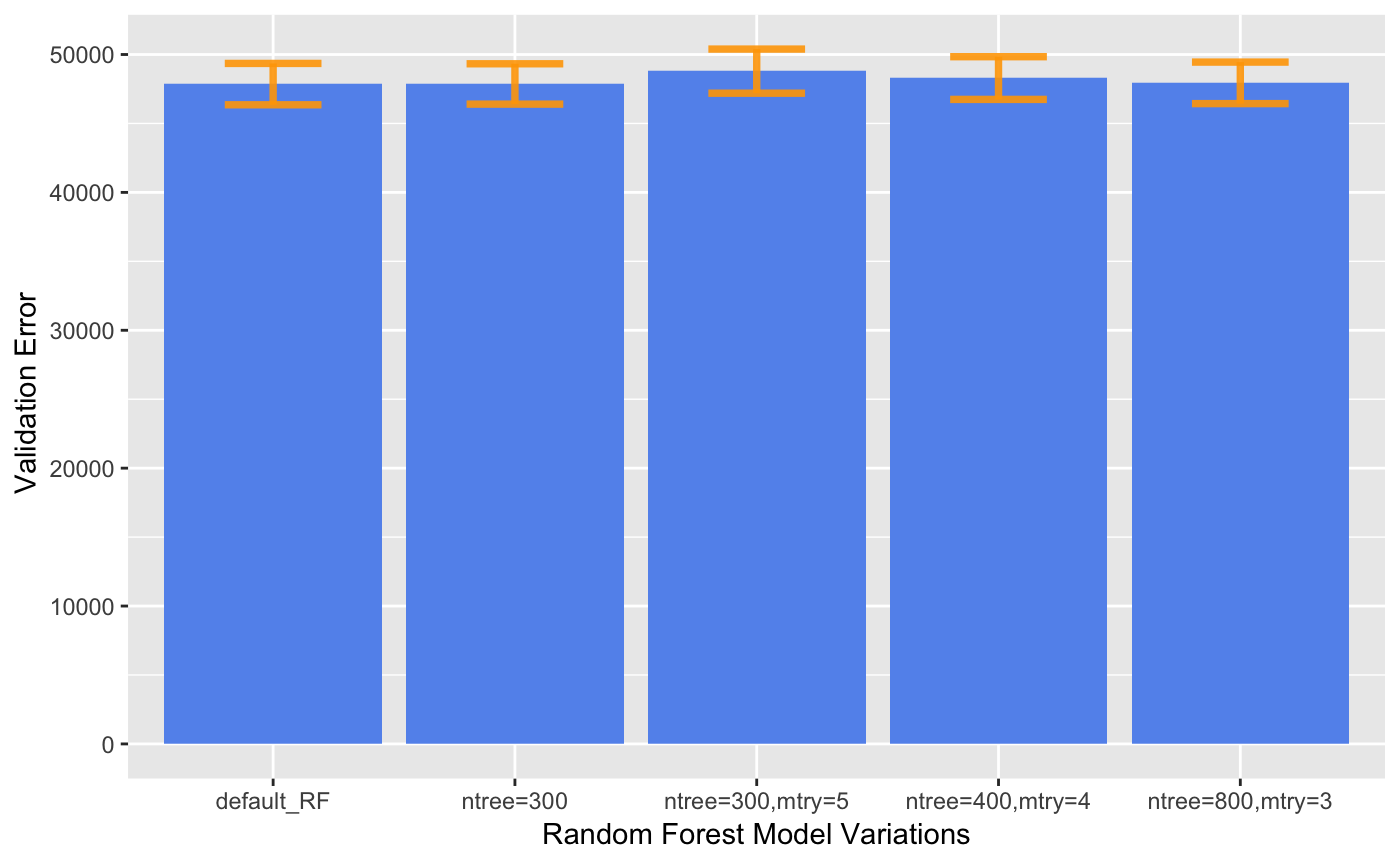


Figure 8 - Random forest parameter selection and validation error

We can see that the mean RMSE values from all models are quite similar with the lowest RMSE value of 47853 for the default parameter used in R (ntree = 500 and mtry = 3). Also, we notice that having number of trees between 300 – 800 does not have a major impact on performance but when we increase mtry the results deteriorate. After using random forest regressor we can see the importance of each variable in the prediction as given in figure below:

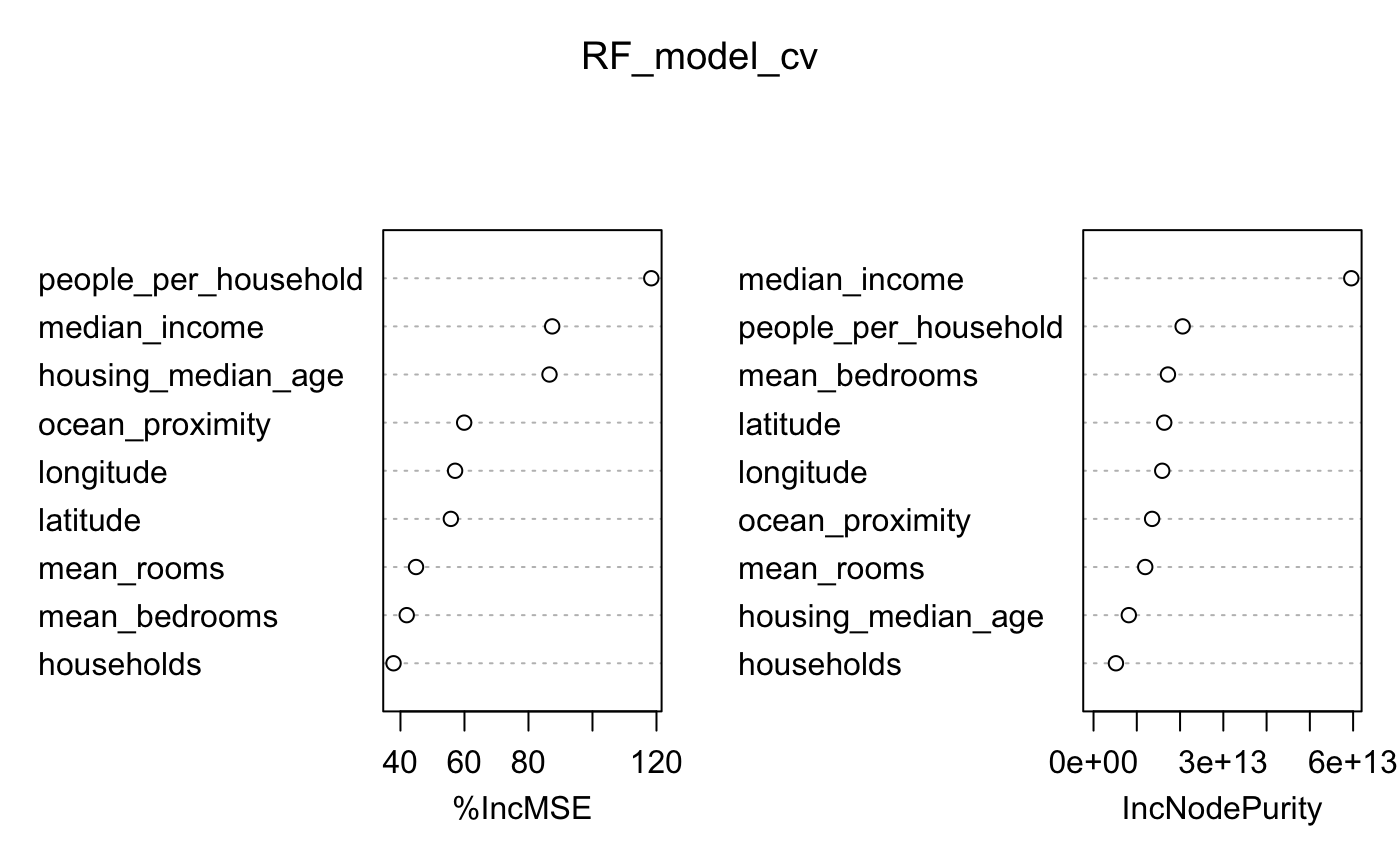


Figure 9 - Random Forest Variable Importance plot

From the variable importance plot we can see that median income and people per household are the two most important variables in the random forest regression model.

## Test Model

After tweaking with the models and choosing the best models it is now time to evaluate the models on the test set. 30% of the data was set aside after data cleaning as test data. now I will share results of running each of the 3 selected models on the test data. Similar metrics will be calculated for all the models to compute the accuracy of prediction. I will calculate the mean absolute percentage error for the predictions obtained as well as the RMSE for the model. The mean absolute percentage error is calculated the computing the absolute percentage error for each of the predictions and then taking the mean of all.

### Linear Regression Model on Test Data

After running the linear model on the test data, a plot of actual vs. predicted values is given in figure below:

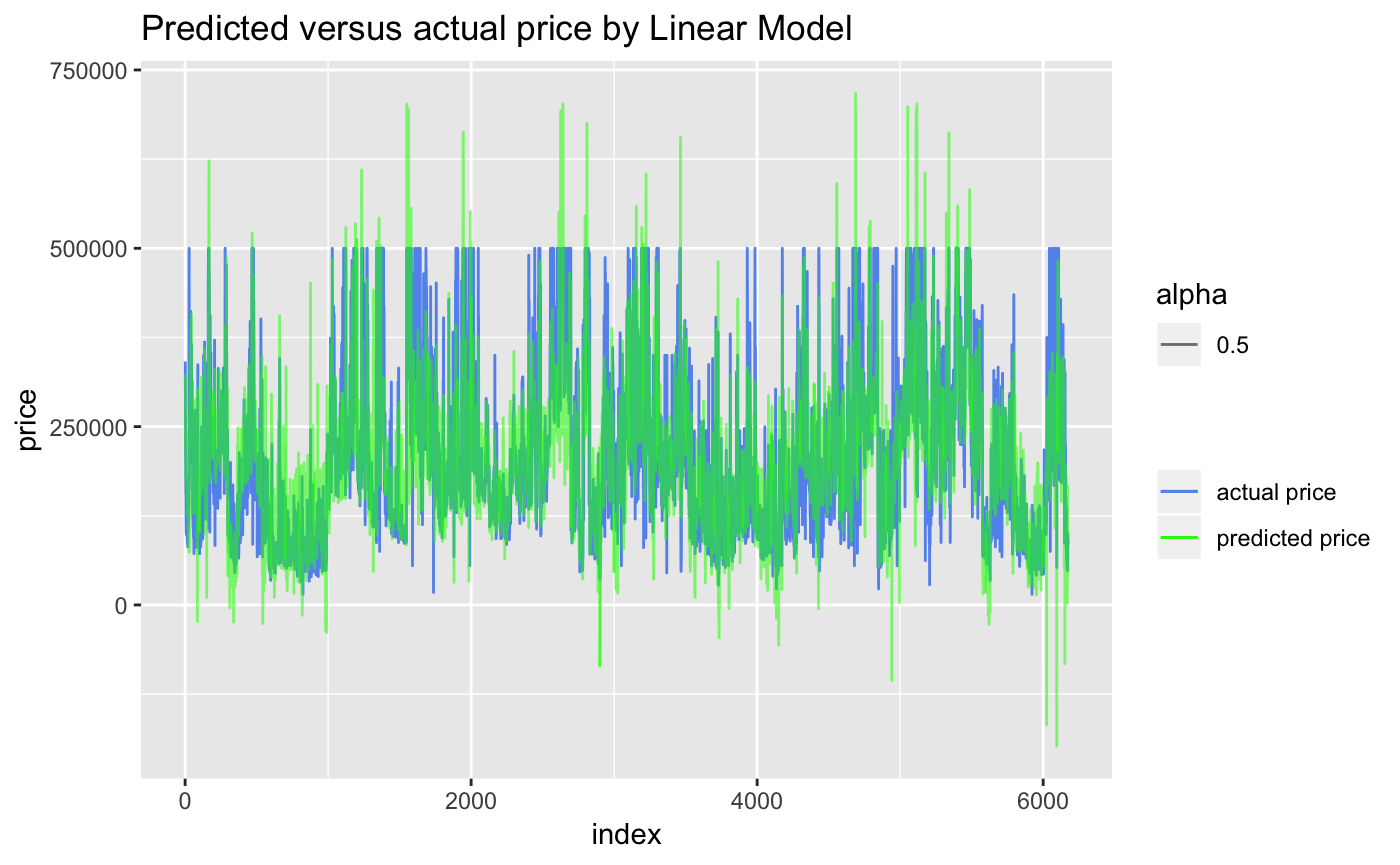


Figure 10 - Plot of actual vs. predicted prices from linear regression model

From the plot above we can see that the predicted price is close to the actual prices but there are many outliers.

The value for mean absolute percentage error is 29.7% and the RMSE calculated on the test set is 67,843.

### Regression Tree Model on Test Data

After running the regression tree model on the test data, a plot of actual vs. predicted values is given in figure below:



Figure 11 - Plot of actual vs. predicted prices from Regression Tree Model

From the figure above we can see that the regression tree model cannot predict the very high or very low prices and hence we can see that the extreme value predictions will not be very good.

The value for mean absolute percentage error is 29.7% and the RMSE calculated on the test set is 72,779

### Random Forest Model on Test Data

After running the random forest model on the test data, a plot of actual vs. predicted values is given in figure below:

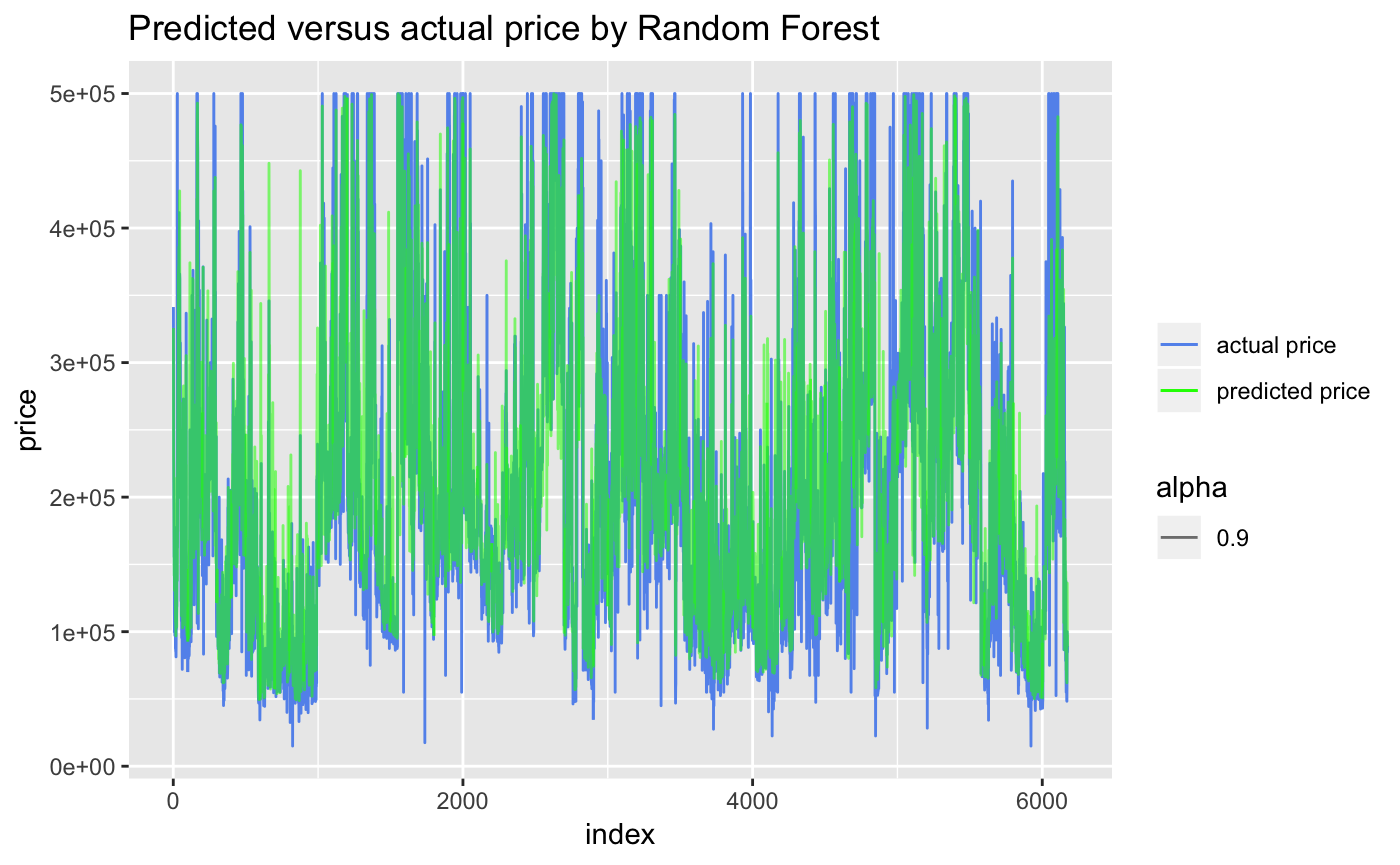


Figure 12 - Plot of actual vs. predicted prices from Random Forest Model

From the figure of actual vs. predicted prices above we can see that the predicted prices are very close to the actual prices. Also, there are not too many outliers.

The value for mean absolute percentage error is 18.2% and the RMSE calculated on the test set is 48,584.

## Evaluate Model

This is the final step, in which the performance of the three models is evaluated. We will compare the values of RMSE and percentage mean absolute error (% MAE), for each of the models in the table below:

|  |  |  |
| --- | --- | --- |
| **Model** | **RMSE** | **% MAE** |
| Linear Regression | 67,843 | 29.7 |
| Regression Tree | 72,779 | 29.7 |
| Random Forest | 48,584 | 18.2 |

From the table we can see that in terms of percentage mean absolute error (%MAE), linear regression model and regression tree model performance is similar; while the random forest regressor performance is substantially better.

However, when we focus on RMSE values we see that there is a difference between the linear regression model and regression tree model, with the linear regression performing better than regression tree. This is because when calculating RMSE the bigger values are penalized much more than the smaller values, resulting in a bigger number of RMSE for regression tree.

In the end the random forest predictions are the best amongst the three models.

# Results

A comparison of the 3 models show that the random forest regressor way out performs the linear regression and regression tree models. A plot of the RMSE from the three regression models is given below:

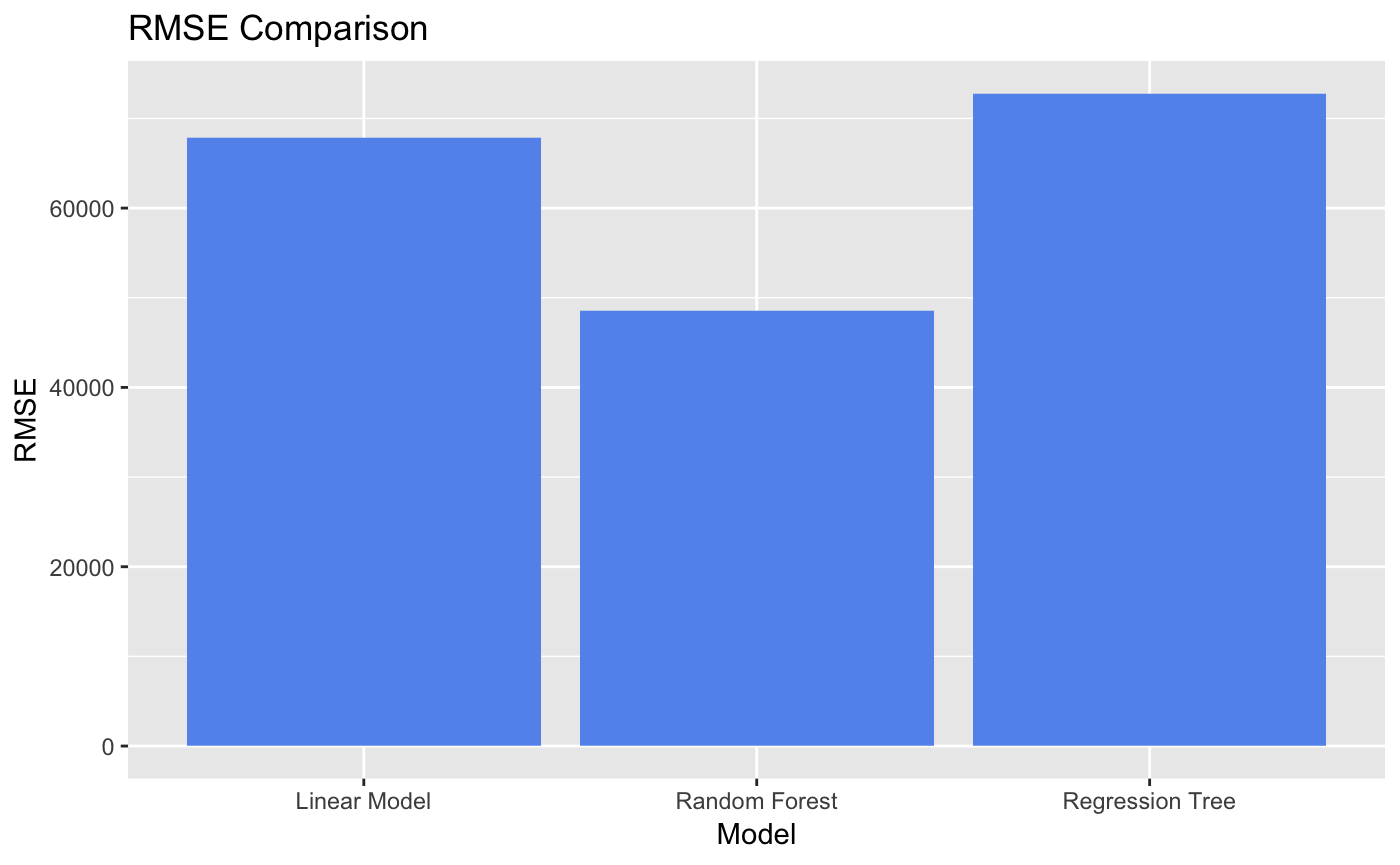


Figure 13 - Comparison of RMSE from the 3 Models

As we can see the that RMSE from the random forest is the best, followed by linear regression model and then the regression tree model.

We can also plot the percentage mean absolute error to see the errors from each model.

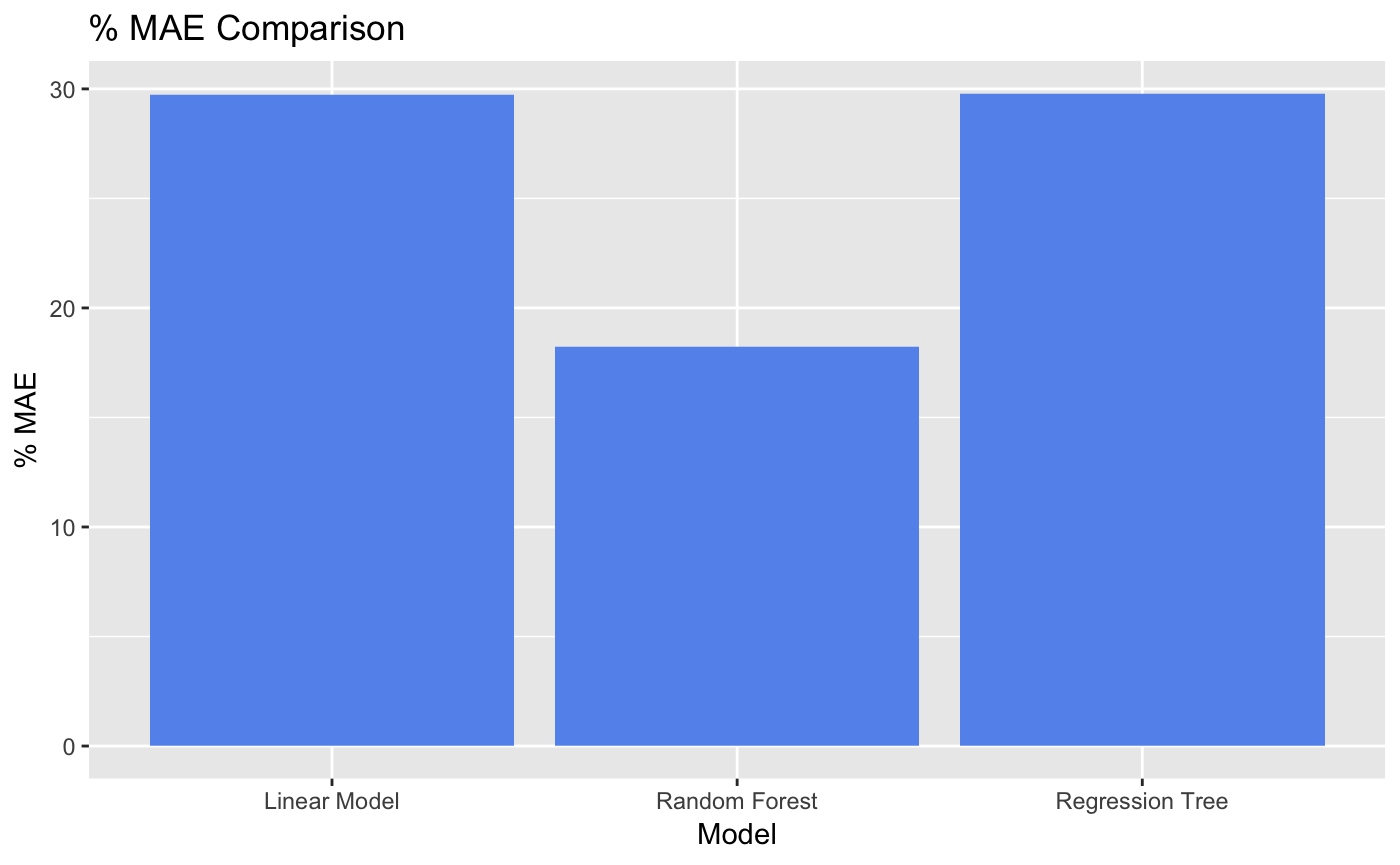


Figure 14 - Comparison of accuracy of the 3 Models

We can conclude that the random forest regressor performance is the best amongst the three regression models. And that it is a good tool for predicting house prices given that the mean absolute error is around 18%.

Another important thing that we learn from the above models is that median income and people per household are the two most important variables in predicting house prices. We see this in all of the three models so we can safely assume these are the most important predictor variables.

# Conclusions

The goal of this project was to build a model to predict house prices in California from data from the 1990 census. The model is built using 9 predictor variables. three different kinds of models have been tested to the best model for predicting house prices given the data.

At the end of the analytical process we can conclude that the random forest regression model is the most suitable for predicting house prices. We can make predictions to about 18% accuracy. Also, we can conclude that a district median income and people per household are the most important predictor variables in the model building process.

# References

1. Aurélien Géron, *Hands-On Machine Learning with Scikit-Learn and TensorFlow*
2. Goodman, A.C. (1998). *Andrew Court and the invention of hedonic price analysis. Journal of Urban Economics*
3. Čeh M., Kilibarda M., Lisec A., Bajat B., *Estimating the Performance of Random Forest versus Multiple Regression for Predicting Prices of the Apartments, 2018*
4. Rosen, S. *Hedonic Prices and Implicit Markets: Product Differentiation in Pure Competition, J. Polit. Econ. 1974*
5. Selim H., *Determinants of House Prices in Turkey: Hedonic regression versus artificial neural network, 2008*
6. Naguyen N., Al Cripps, *Predicting housing value: A comparison of Multiple Regression and Artificial Neural Networks*
7. Park B., Bae J. K., *Using Machine Learning Algorithms for Housing Price Prediction: The case of Fairfax County, Virginia Housing Data*
8. Liaw A., Wiener M., *Classification and Regression by Random Forest, R News 2002*
9. Brieman L., *Random Forests, Machine Learning 2001*