|  |
| --- |
| **Predicting House Prices using Machine Learning Models** |

**Project Overview:**

In this project, the machine learning algorithms Linear Regression, k-Nearest-Neighbours regression (k-NN), Random Forest (RF) regression and Support Vector Regression (SVM) were used to predict house prices from a set of features in the Housing data set.

The goal was to compare their relative performance at this task.

I have used scikit learn (sklearn) Python library to build the respective models and calculated the error between the actual and predicted house price using four different metrics.

Hyperparameters for the algorithms used were optimally selected.

An optimal subset of hyperparameters for the algorithms was selected through the grid search algorithm for the best prediction.

The KNN was found to consistently perform better than the Random Forest algorithm in terms of smaller errors and be better suited as a prediction model for the house price problem.

The practical usefulness of the prediction is rather limited to making basic valuations.

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| Sl. No. | Title | Page No. |
| 1 | Introduction | 4 |
| 2 | Problem statement | 4 |
| 3 | Source and Size of Dataset | 4 |
| 4 | Overview of the Dataset | 5 |
| 5 | Python Libraries Used | 6 |
| 6 | Machine Learning Algorithms Used | 6-7 |
| 7 | Handling Missing values, duplicates and Categorical columns in the dataset | 7-8 |
| 8 | Exploratory Data Analysis | 8-18 |
| 9 | Train Test Split | 19 |
| 10 | Feature Scaling | 19 |
| 11 | Model Building | 20 |
| 12 | Model Evaluation using error Metrics | 21-22 |
| 13 | Feature Selection | 23-24 |
| 14 | Hyperparameter Tuning | 24-27 |
| 15 | Training model with hyperparameters | 27-28 |
| 16 | Evaluation of Tuned Model | 29-30 |
| 17 | Code Sample | 31-33 |
| 18 | Conclusion | 34 |

**Introduction**

Accurately determining the value of a residence poses a significant challenge for various stakeholders, including homeowners, prospective buyers, real estate agents, lenders, and investors.

The prediction of a property's selling price encompasses various methodologies, often relying on regression techniques. These techniques typically involve one or more predictor variables as inputs and a singular target variable as the output.

Within the scope of this project, I have undertaken a comparative analysis of different machine learning models to assess their effectiveness in predicting house selling prices. This evaluation is based on diverse features, including area size, the number of bedrooms and bathrooms, and amenities like proximity to a main road, availability of air conditioning, hot water heating facilities, and more.

**Problem statement**

To what extent can housing prices be forecasted using Regression algorithms such as Linear regression, Decision Tree Regressor, Support Vector Regressor, k-Nearest Neighbors and Random Forest regression?

Our dataset comprises real estate market information, encompassing features such as bedrooms, area, bathrooms and so on.

The Objective is to:

1. Implement data preprocessing and preparation techniques to acquire clean and refined data.

2. Develop machine learning models capable of predicting house prices based on various features.

3. Conduct a comprehensive analysis and comparison of model performances to identify and select the most effective model.

**Source and Size of the Dataset:**

I have collected the Dataset from official website of Kaggle.

The size of the dataset Housing.csv is 30 KB.

**Overview of the Dataset:**

The dataset offers essential features for forecasting house prices, encompassing variables such as area, bedrooms, bathrooms, stories, amenities like air conditioning and parking, and details on furnishing status. Its comprehensive nature allows for in-depth analysis and modeling, facilitating a nuanced understanding of the factors influencing house prices and the development of precise predictions within real estate markets.

The dataset comprises of 13 columns, namely: area, price, bedrooms, bathrooms, parking, basement, stories, preferred area, air conditioning, hot water heating, main road, guest room, and furnishing status.

**Brief of the Features/columns of the Dataset:**

This dataset provides comprehensive information for house price prediction, with 13 column names:

* **Price:** The price of the house.
* **Area:** The total area of the house in square feet.
* **Bedrooms:** The number of bedrooms in the house.
* **Bathrooms:** The number of bathrooms in the house.
* **Stories:** The number of stories in the house.
* **Mainroad:** Whether the house is connected to the main road (Yes/No).
* **Guestroom:** Whether the house has a guest room (Yes/No).
* **Basement:** Whether the house has a basement (Yes/No).
* **Hotwaterheating:** Whether the house has a hot water heating system (Yes/No).
* **Airconditioning:** Whether the house has an air conditioning system (Yes/No).
* **Parking:** The number of parking spaces available within the house.
* **Prefarea:** Whether the house is located in a preferred area (Yes/No).
* **Furnishingstatus:** The furnishing status of the house (Fully Furnished, Semi-Furnished, Unfurnished).

**Python Libraries Used:**

* **NumPy:**  A powerful library for numerical operations and multidimensional array manipulation in Python, essential for scientific computing and data analysis.
* **Pandas:** A versatile data manipulation and analysis library that provides data structures like DataFrame for efficient handling and exploration of structured data.
* **Matplotlib:** A widely-used plotting library that enables the creation of various static, animated, and interactive visualizations in Python.
* **Seaborn:** Built on top of Matplotlib, Seaborn is a statistical data visualization library that simplifies the creation of informative and aesthetically pleasing graphics.
* **Scikit-learn:** A comprehensive machine learning library that provides simple and efficient tools for data analysis and modeling, offering various algorithms for classification, regression, clustering, and more.

**Machine Learning algorithms Used**

Machine learning algorithms are computational procedures designed to enable computers to learn patterns and make predictions or decisions from data. They use statistical techniques to iteratively improve their performance on a specific task without being explicitly programmed, adapting and evolving as they process new information.

In this project, I have used Regression algorithms to predict House price based on the given features in the dataset.

**Why we have used Regression algorithms?**

Regression algorithms are utilized to model and analyze the relationships between variables, especially when the goal is to predict a continuous numerical outcome. These algorithms are valuable in understanding how independent variables impact the dependent variable, making them essential for tasks like predicting house prices, stock values, or any situation where understanding and forecasting numerical trends is crucial.

**Algorithms that I have used in this project:**

**1. Linear Regression:** A simple and widely-used algorithm that establishes a linear relationship between input features and a continuous target variable, making it suitable for predicting numerical outcomes.

**2. Random Forest Regressor:** An ensemble learning algorithm that constructs a multitude of decision trees and aggregates their predictions, providing robust and accurate predictions for regression tasks through its adaptability and resistance to overfitting.

**3. K-Neighbors Regressor:** A non-parametric algorithm that predicts the target variable based on the average of its k-nearest neighbors, making it effective for capturing local patterns in data and suitable for regression tasks.

**4. Decision Tree Regressor:** A tree-structured model that recursively splits data into subsets, enabling the prediction of continuous target variables by navigating the tree from root to leaf based on input features.

**5. Support Vector Machine Regressor:** An algorithm that finds the optimal hyperplane to best represent the relationship between input features and the target variable, making it effective for regression tasks by maximizing the margin between data points.

**Handling missing data/duplicates and categorical column in the Dataset:**

Shape of the Dataset is (545,13)

Statistical Overview of the Data:



**Handling Missing values and Duplicates:**

In this dataset, there is no missing value and no duplicates.

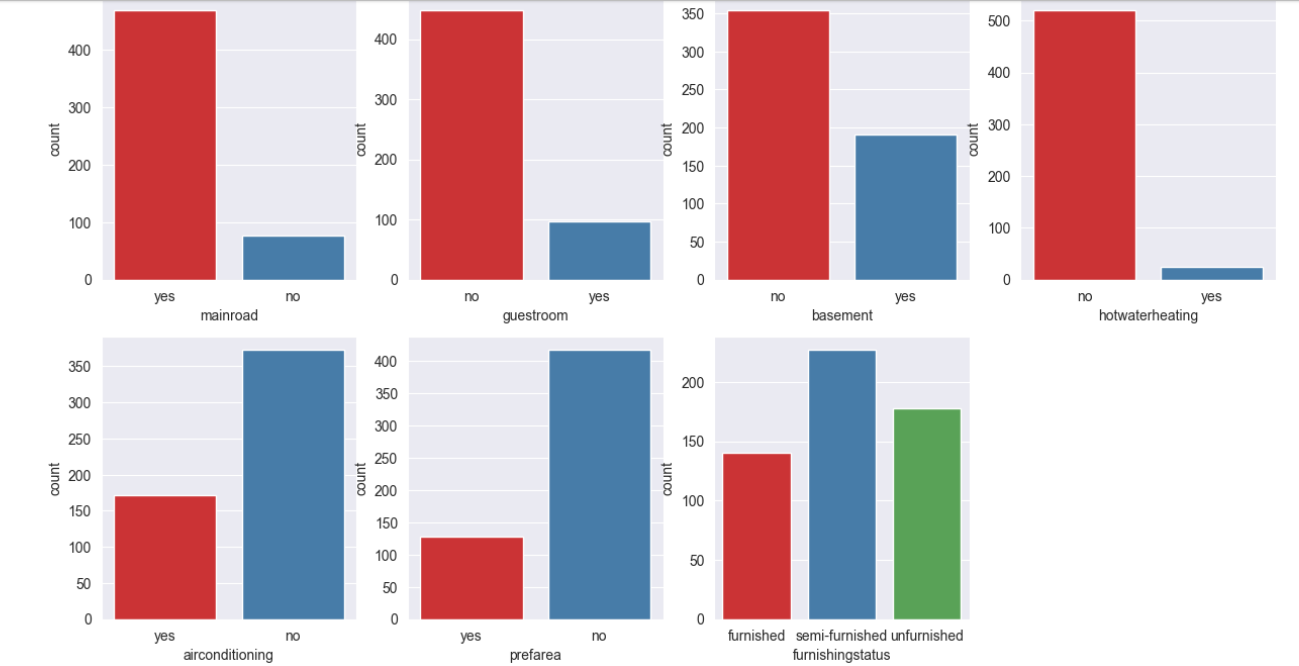
**Handling categorical data:**

I have replaced Yes = 1 and No=0 and

Semi-Furnished-0, Unfurnished -1, furnished-2

**Exploratory Data Analysis:**

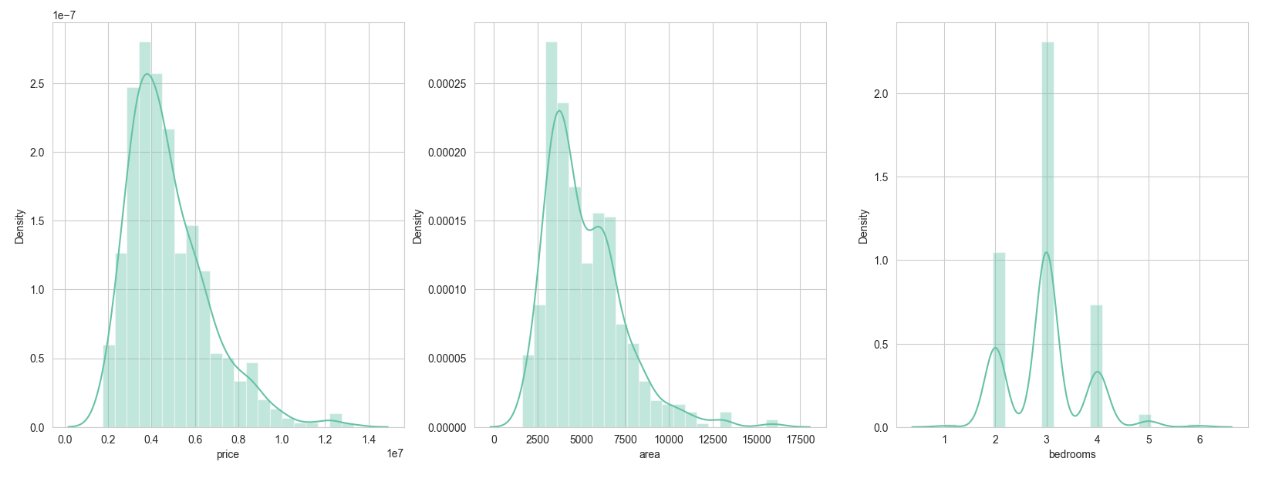
**Count plot of all Categorical columns**

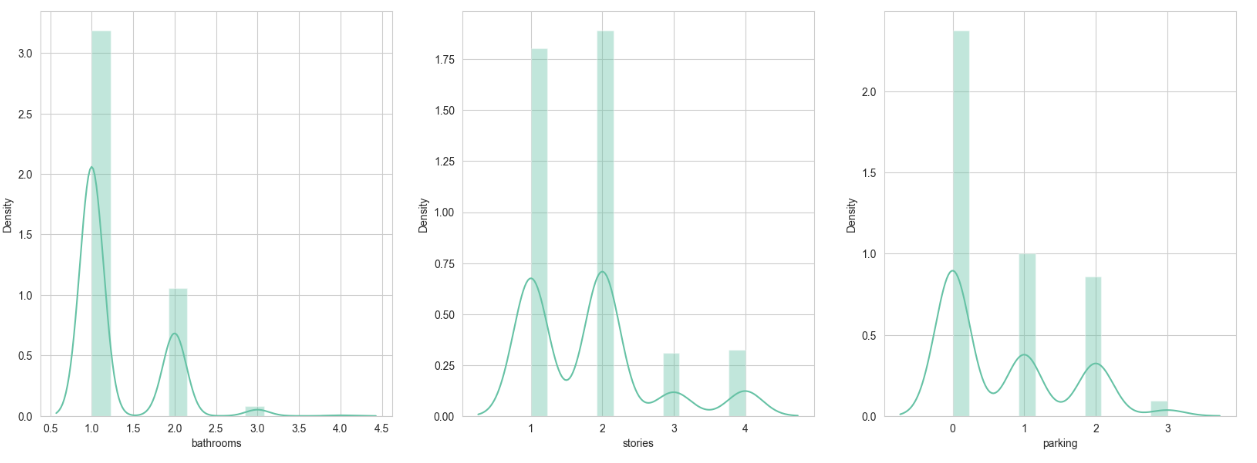
****

**Conclusion from above plots**

* Majority of houses which are connected to main road.
* There are fewer number of houses which has guestroom.
* There are more houses which do not have basement as compared to number of houses which has a basement
* There are fewer number of houses which has hot water heating facility.
* Number of houses which has air conditioning facility is less than houses which do not have air conditioning facility
* Number of houses that are located in preferred area is less than those which are not in preferred area
* Count of Semi-furnished house is largest followed by unfurnished with second highest number and furnished with least number.

**Distribution plot of Numerical columns**



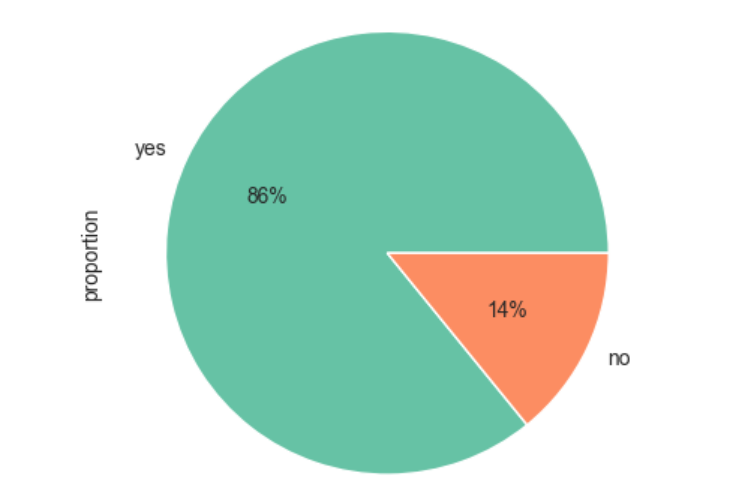


**Conclusion from above plots:**

From above distribution plots, I can conclude that data is skewed in each of the above features

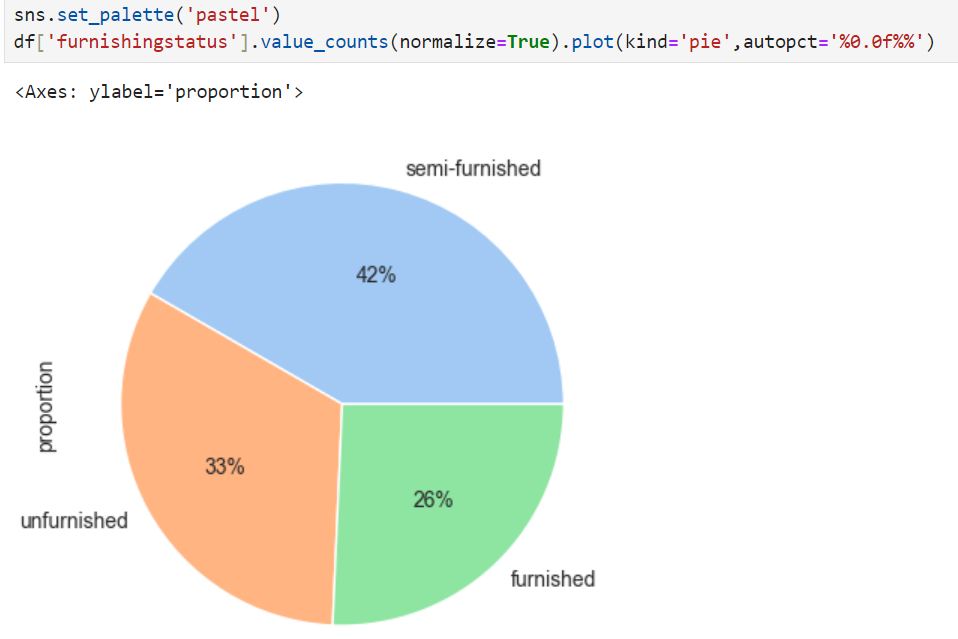
There are outliers in the features.

**Below is a pie plot depicting ration of houses which are near by main road and which houses are not near main road**



**Conclusion from above plot:**

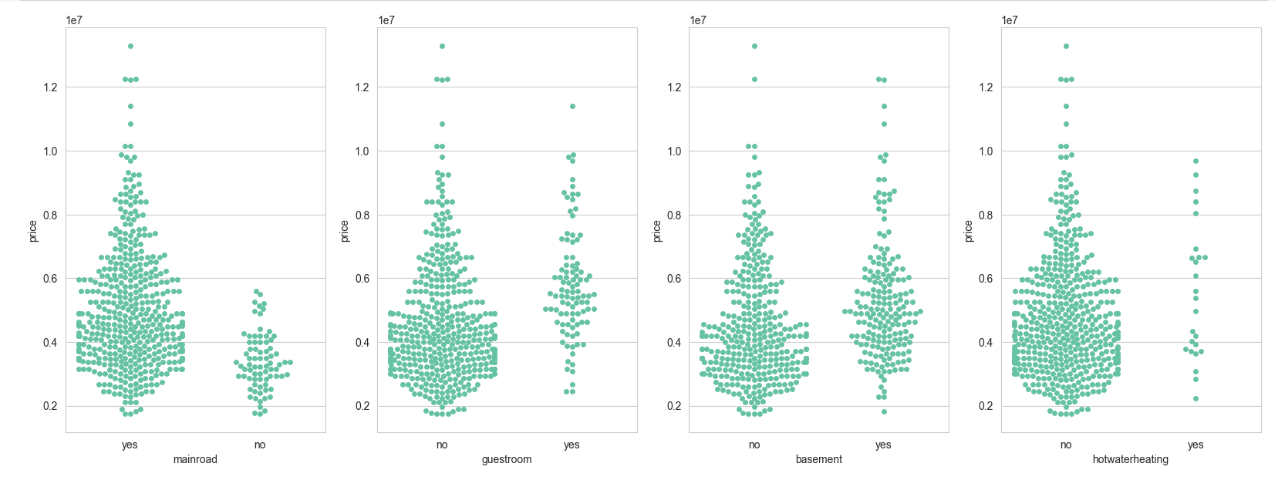
* 86% of houses are connected to main road
* 14% of the houses are not connected to main road

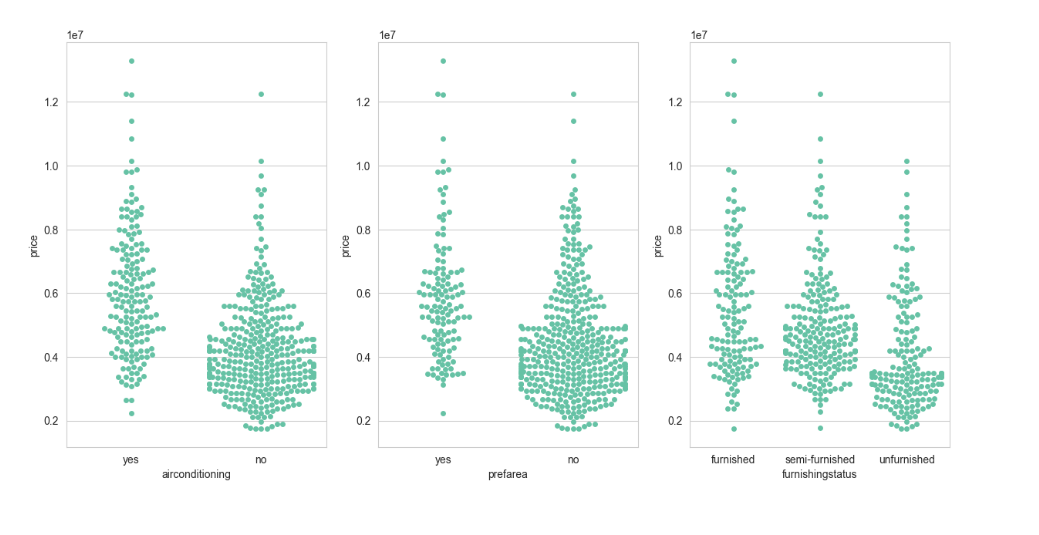
****

**Conclusion**

* 42% houses are semi-furnished
* 33% houses are unfurnished
* 26% houses are furnished

**Relation between categorical columns and Target column Price**

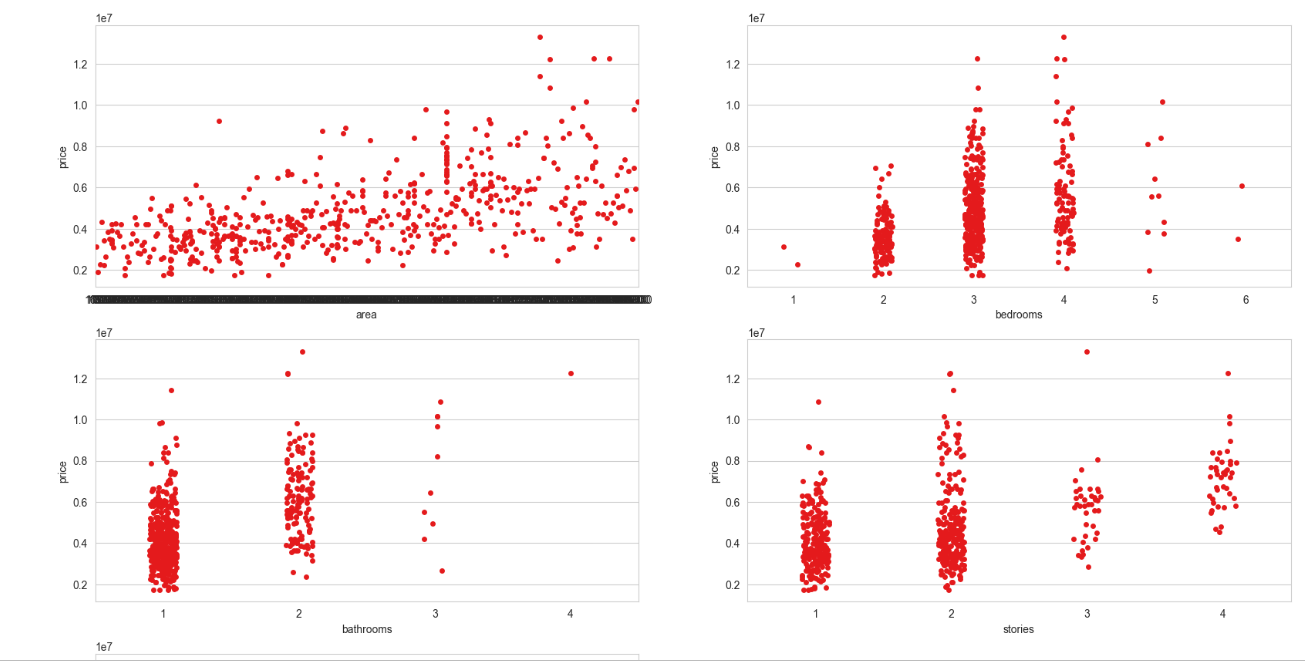


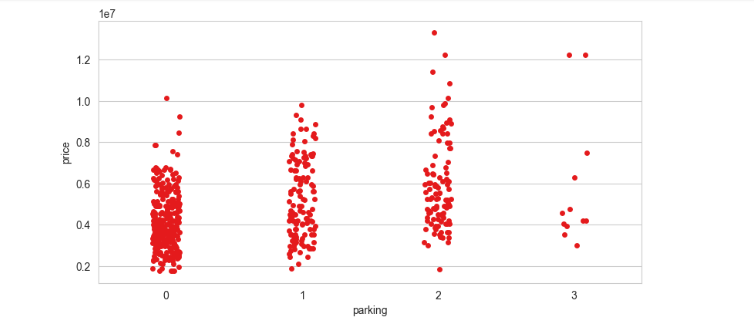


**Conclusion:**

* From above scatterplots, I can conclude there are more houses which is near main road compared to houses which is not nearby main road.
* House price is higher for house which is near mainroad
* There are higher number of houses which do not have a guestroom,basement and hotwater heating facility.
* Presence of Guestroom,basement and hotwaterheating facility in a house does not have a significant effect on house price.
* There are higher number of houses which do not have airconditioning facility, preferred area and semi-furnished.
* House having airconditioning facility has higher price compared to house which do not airconditioning facility.
* Furnished houses are more costly than semi-furnished and unfurnished houses.

**Relation between Numerical columns and target column Price**

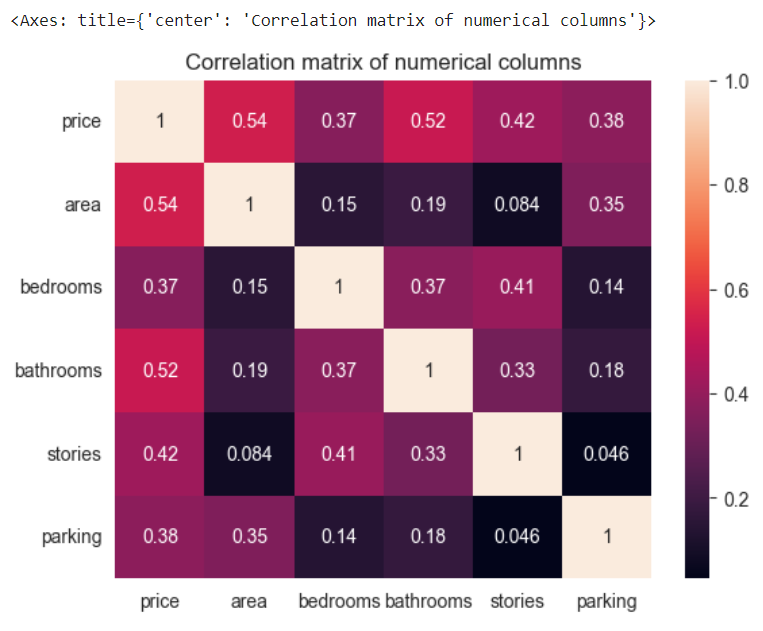




**Conclusion:**

* From above plots, I can conclude that:
* House with large area has higher price as compared to houses with small area.
* Majority of the houses has 3 bedrooms
* House price increases with increase in number of bedrooms.
* House price starts decreasing when number of bedrooms is more than 4.
* Majority of houses has 1 bathroom
* Number of bathrooms does not have a significant effect on house price.
* Number of stories does not affect the house price.
* House price is higher for houses which has a parking.

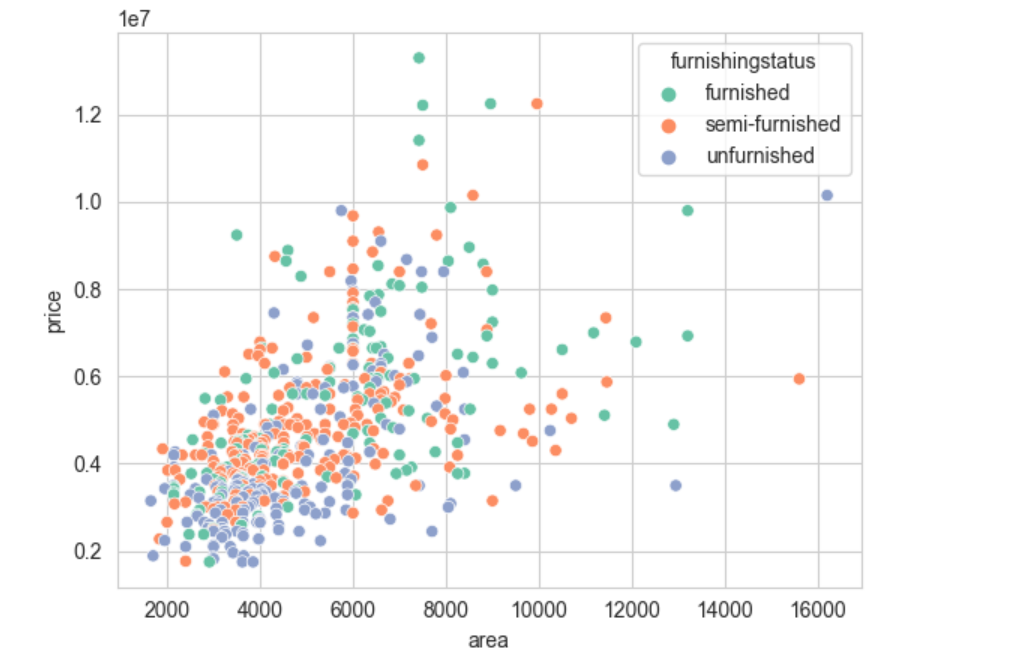
**Correlation between numerical features and target feature-price**



**Conclusion**

* Correlation between price and area is relatively high- 0.54
* Correlation between price and bathrooms is next high value-0.52
* Correlation between price and stories is 0.42
* Although these corelations are strong enough, still I can say price and area are highly co-related as compared to other features

**Price of a house based on area and furnishing status**

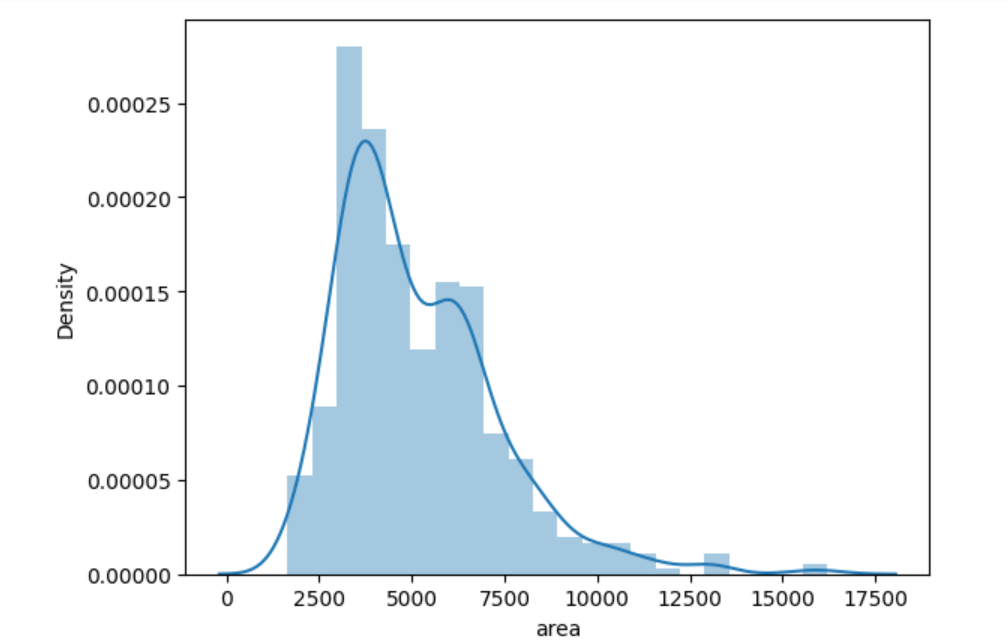


**Conclusion:**

* Majority of houses are semi-furnished
* Price increases with increase in house area
* Number of furnished house is less than unfurnished and semi-furnished houses.
* Area of Furnished house is more as compared to semi-furnished and unfurnished
* Price is highest for furnished house
* Price is lowest for unfurnished house
* Moderate price for semi-furnished house

**Removal of Outliers:**

**If I plot a distribution graph of column “area”, distribution of data points is skewed towards right. These are possibly outliers.**

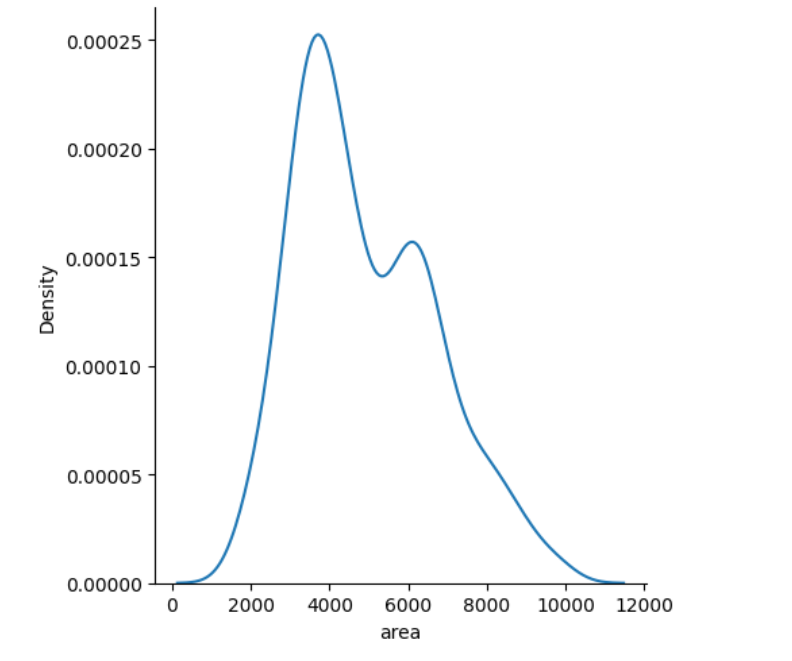
****

To remove outliers, I have considered a threshold value of 10000 and dropped rows having value more than 10000 and less than 2500

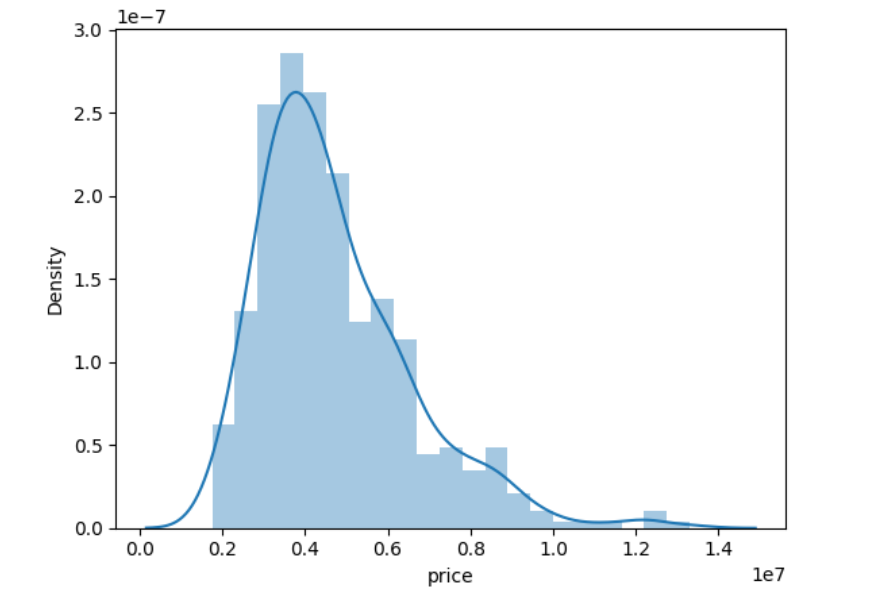
df.drop(df[(df['area']>10000)].index, axis=0,inplace=True)

Dropped rows containing values above threshold value

After dropping the outliers, the distribution is more standard.



Similarly, if I plot distribution graph of column “Price”, data is skewed towards right.

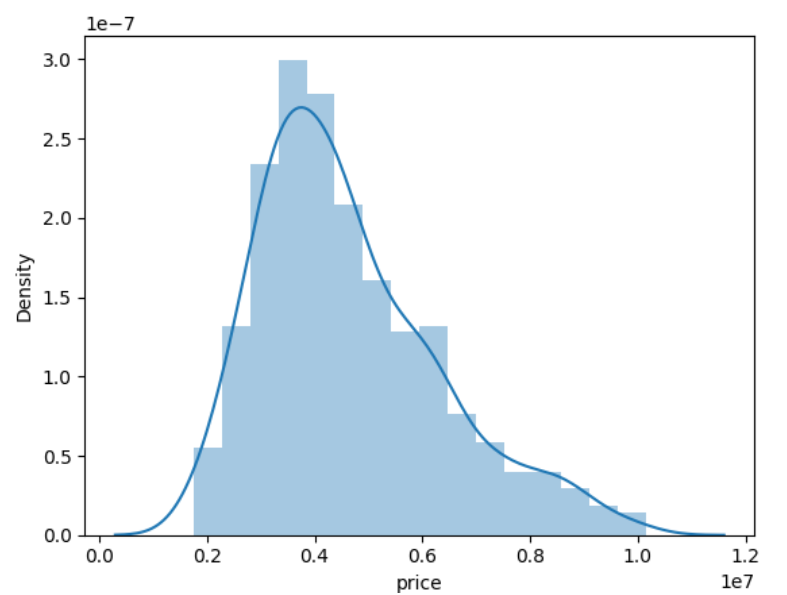


* I can see there are outliers in the column price. I will consider a threshold value of 10150000 and drop rows containing values more than 10150000



* Dropped the rows containing values greater than 10150000

Now if the distribution of data is more standard



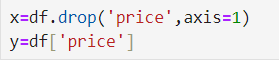
I have removed the outliers from the area and price columns of the dataset.

Next, I will handle Categorical columns by using below code



After handling missing data, duplicates, outliers and categorical columns, I have a clean data to start building the Machine Learning models.

Here I have separated the independent and dependent variables



**Train-Test Split**

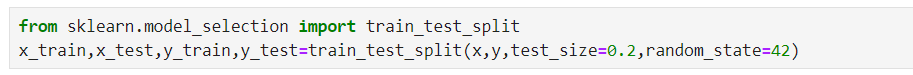
Train-test split is a common practice in machine learning where the available dataset is divided into two subsets: the training set and the testing set. The training set is used to train and build the model, while the testing set is kept separate and used to evaluate the model's performance on unseen data, ensuring a reliable assessment of its generalization ability and predictive accuracy. This approach helps detect overfitting and ensures the model's effectiveness in making predictions on new, unseen data.

Here in this step, I have divided the entire dataset into two parts:

Training data- I have trained the model using the training set.

Testing data- I have then tested the model using the test set.

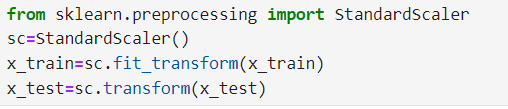
Then perform train-test split using Scikit Learn library of Python



**Feature Scaling**

Feature scaling is a preprocessing step in machine learning that involves transforming the numerical features of a dataset to a standardized or normalized scale. This is crucial because many machine learning algorithms are sensitive to the magnitude of input features, and scaling ensures that all features contribute equally to model training. Common techniques include Min-Max scaling, Standardization (Z-score normalization), and Robust scaling, which adjust feature values to a common range or distribution, enhancing the stability and convergence of the algorithms during training.

In this project, I have used Standard Scaler from Scikit Learn Library of Python.



**Model Building**

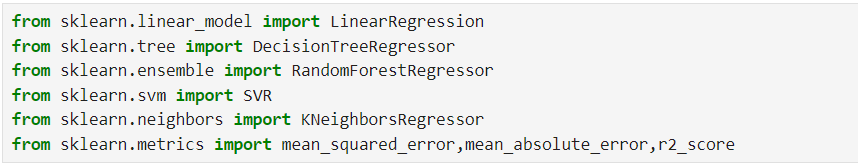
Model building in machine learning refers to the process of creating a predictive model based on a given dataset. It involves selecting an appropriate algorithm, preprocessing the data, splitting it into training and testing sets, and training the model on the training data. The model is then evaluated on the testing set to assess its performance.

In this project, I have used regression algorithms to build my model.

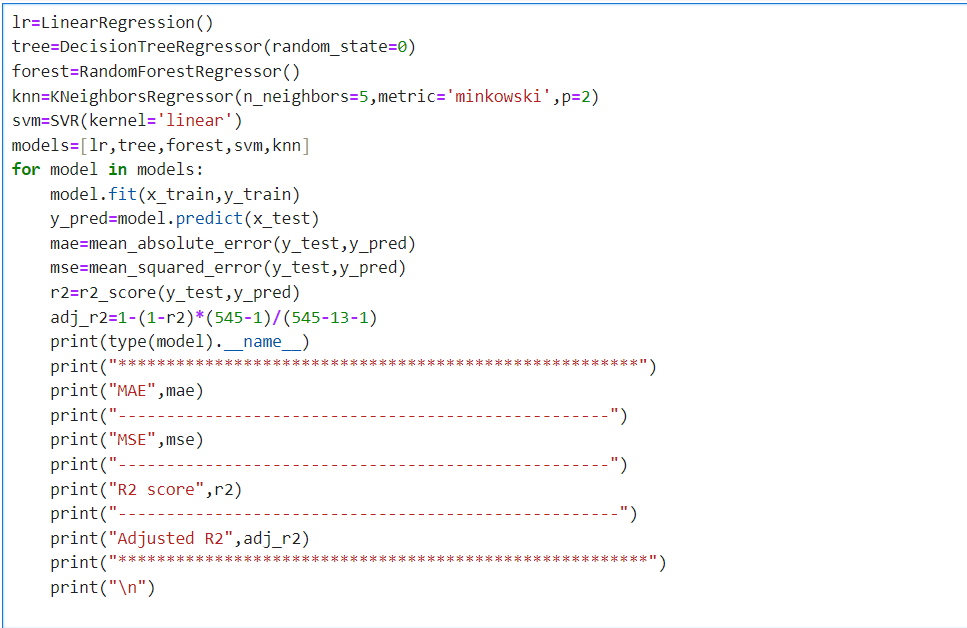
I have evaluated multiple models and concluded with the model which gave best result.

Build the model:

Importing the required models and error metrics from sklearn library of Python



Train the model using training set of data and evaluate the model using error metrics like Mean Squared error, Mean absolute error, r2\_score, adj\_r2 score.



**Model Evaluation using error metrics:**

In this project, I have used below four error metrics to evaluate our models.

1. Mean Absolute Error (MAE):

MAE is a metric that measures the average absolute difference between the predicted and actual values. It provides a straightforward indication of the model's accuracy, with lower MAE values indicating better performance.

2. Mean Squared Error (MSE):

MSE calculates the average of the squared differences between predicted and actual values. It penalizes larger errors more heavily than MAE, making it sensitive to outliers. A lower MSE signifies better model performance.

3. R-squared (R2) Score:

R2 Score quantifies the proportion of the variance in the dependent variable that is predictable from the independent variables. It ranges from 0 to 1, with higher values indicating a better fit. A score of 1 means the model perfectly predicts the target variable.

4. Adjusted R-squared (Adj R2) Score:

Adj R2 adjusts the R2 score based on the number of predictors in the model. It penalizes the addition of irrelevant predictors, providing a more accurate assessment of a model's explanatory power. It is particularly useful in multiple regression scenarios with multiple predictors.

MAE and MSE assess prediction accuracy, with MSE being more sensitive to large errors. R2 Score measures the proportion of explained variance, while Adjusted R2 Score refines this metric to account for the impact of irrelevant predictors in multiple regression models.

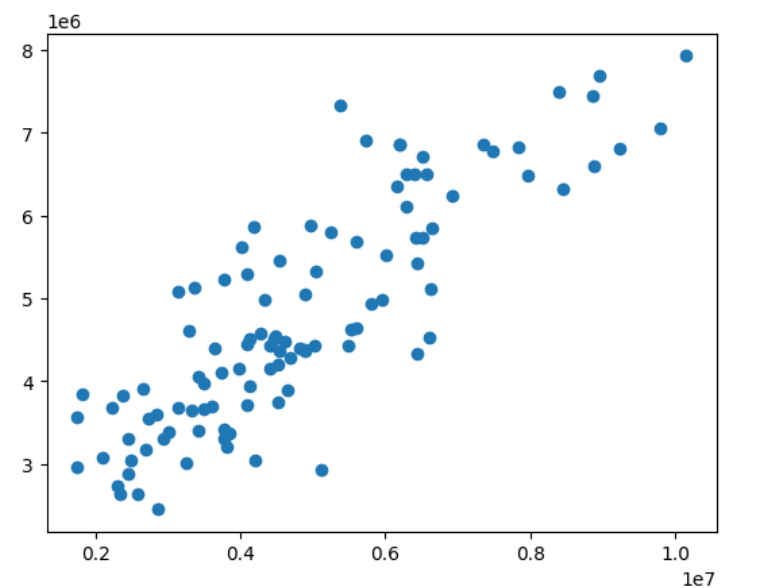
**Result of Evaluation:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model Name | Mean Absolute error | Mean squared Error | R2\_score | Adj\_R2\_score |
| Linear regression | 800016.8650918446 | 1058877130506.5159 | 0.7071147043895634 | 0.6999442545911911 |
| Decision Tree  Regressor | 1178938.6666666667 | 2570880100320.0 | 0.2888948524168936 | 0.271485498521262 |
| Random forest Regressor | 841778.8622222221 | 1260483653152.813 | 0.6513503628233163 | 0.6428146843236988 |
| Support vector Machine Regressor | 1521856.2437009262 | 4019313946543.022 | -0.11174178709600513 | -0.13895957096087908 |

**From above we can conclude that:**

* Linear Regression, K-Neighbors regressor and Random Forest Regressor are getting better accuracy compared to other two models
* For Linear regression, r2\_score is 0.7071147043895634, for random forest r2\_score is 0.6333285440650249 and for K-Neighbors r2\_score is 0.7029723116526394
* I can conclude Linear regression gives better result than all other models

**Plotting actual and Predicted values**



In above plot, there is a linear relation between actual and predicted value.

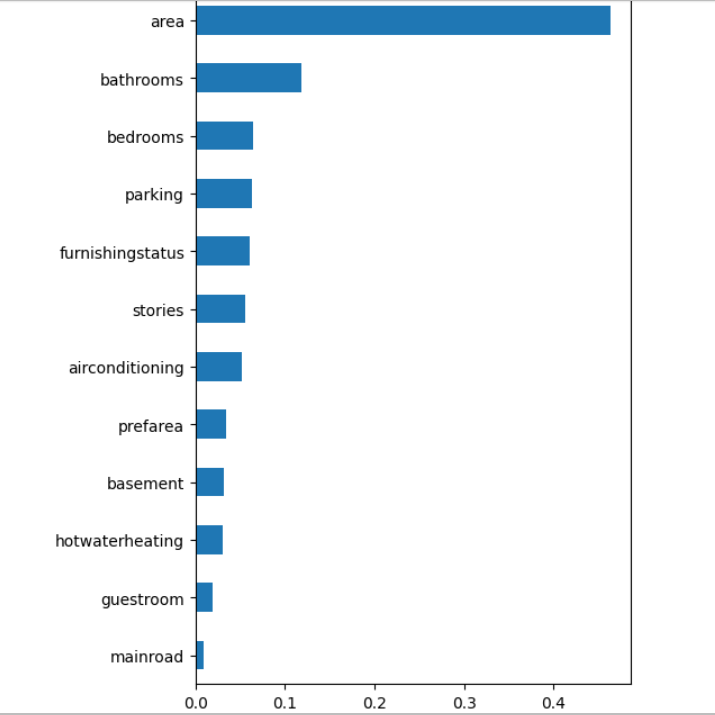
**Feature Selection**

Feature selection is a process in machine learning that involves choosing a subset of relevant features from the original set of variables in a dataset. The primary objectives of feature selection are to improve model performance, reduce computational complexity, and enhance interpretability.

I have done feature selection by using the attribute feature\_importances\_.

This attribute gives us the important features. We can select the more relevant features and exclude the remaining features while building the model. In this way we can improve the accuracy of the model

****

****

* From above plot, I can conclude that features guestroom and mainroad has less importance as compared to other features
* Features- prefarea, hotwaterheating and basement has similar importance.
* Features airconditioning and stories has similar importance
* Features parking and furnishing status has similar importance

After feature importance, we evaluate the model again

* But I did not get better testing result for any algorithms
* So, I have performed Hyperparameter tuning to improve the model further
* For Hyperparameter tuning, I have Grid search Cross validation technique

**Hyperparameter Tuning**

Hyperparameter tuning is the process of optimizing the hyperparameters of a machine learning model to improve its performance. Hyperparameters are external configurations that are not learned from the data but are set prior to the training process. The goal of tuning is to find the best combination of hyperparameters that maximizes the model's accuracy or other performance metrics.

There are various techniques available for hyperparameter tuning in machine learning models

In this project, I have used Grid search Cross Validation technique.

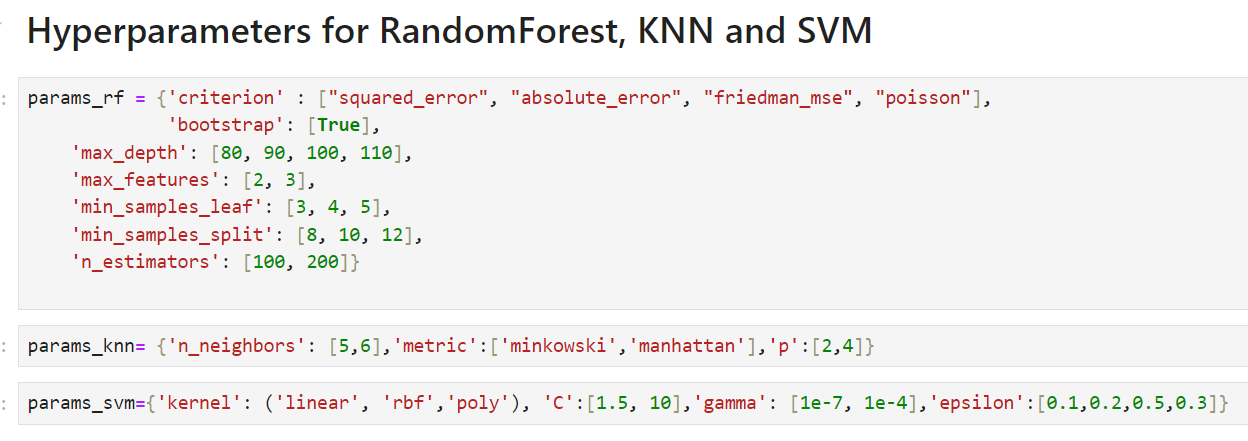
Grid Search CV (Cross-Validation) is a hyperparameter tuning technique used in machine learning to systematically search through a predefined grid of hyperparameter values for a model. The process involves evaluating the model's performance for each combination of hyperparameters using cross-validation.

Steps involved in Grid Search CV hyperparameter tuning technique:

* **Import GridSearchCV from sklearn library of python**



* **Specify the hyperparameters for respective machine learning models.** 
  + For Random Forest, I have taken a set of values for parameters criterion, bootstrap, max\_depth, max\_features, min\_sampels\_leaf, min\_samples\_split and n\_estimators
  + For K-Neighbors regressor, I have taken a set of values for parameters n\_neighbors, metric and p.
  + For SVM Regressor, I have taken values for parameters kernel ,C , gamma and epsilon



* **Model and Evaluation Metric:**

Select the machine learning model you want to tune and the evaluation metric you want to optimize.

I have tuned Random forest, KNN and SVM models.

* **Grid Search:**

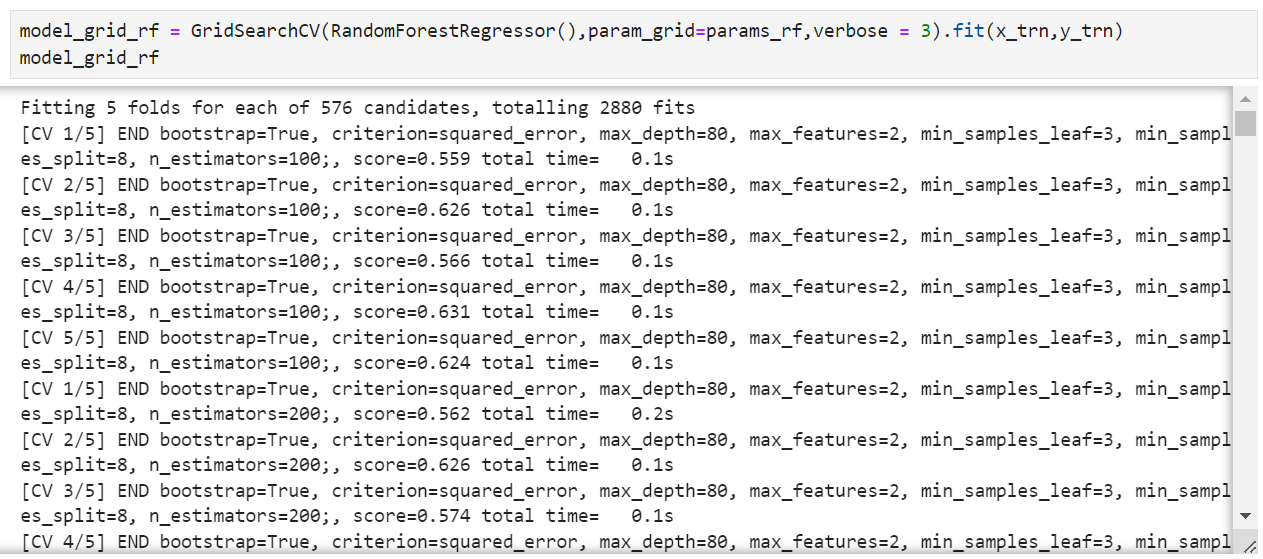
Iterate through all possible combinations of hyperparameters in the grid and train the model for each combination using cross-validation.

**Tuning model with Hyperparameters**

After selecting the set of hyperparameters with which I want to tune my model.

I have used Grid search CV method and tuned my models.

Below is the code snippet from this project that shows the steps I have used to tune Random Forest Regressor model.



After tuning the Random forest model, checked test score

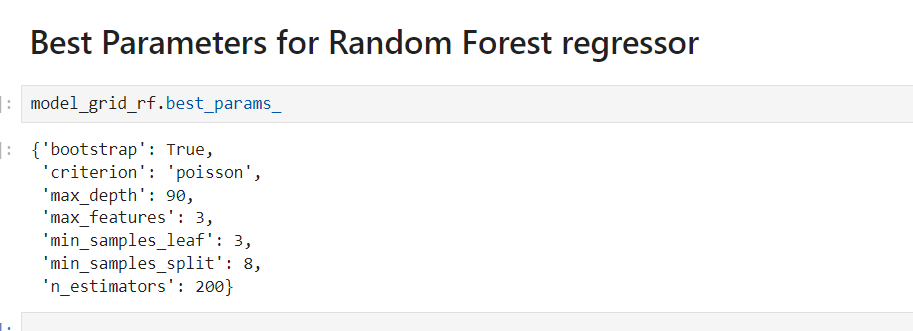


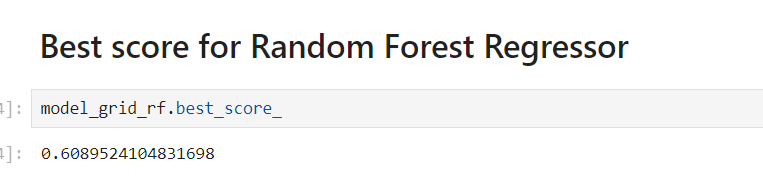
After tuning, random forest regressor testing score is 0.6532709635053864

* **Select Best Parameters:**

Identify the set of hyperparameters that result in the best performance according to the chosen evaluation metric.

I checked best combination of parameters.





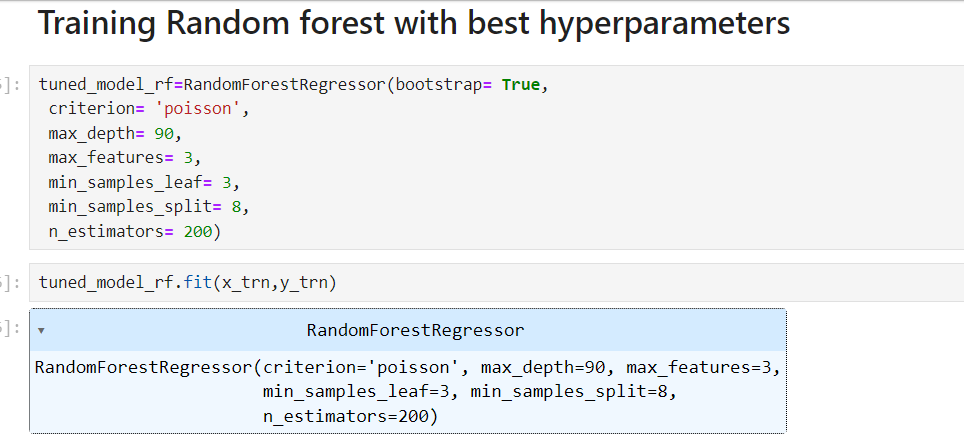
* **Final Model:**

Train the final model using the selected optimal hyperparameters on the entire training dataset.

**Training models using Hyperparameter**

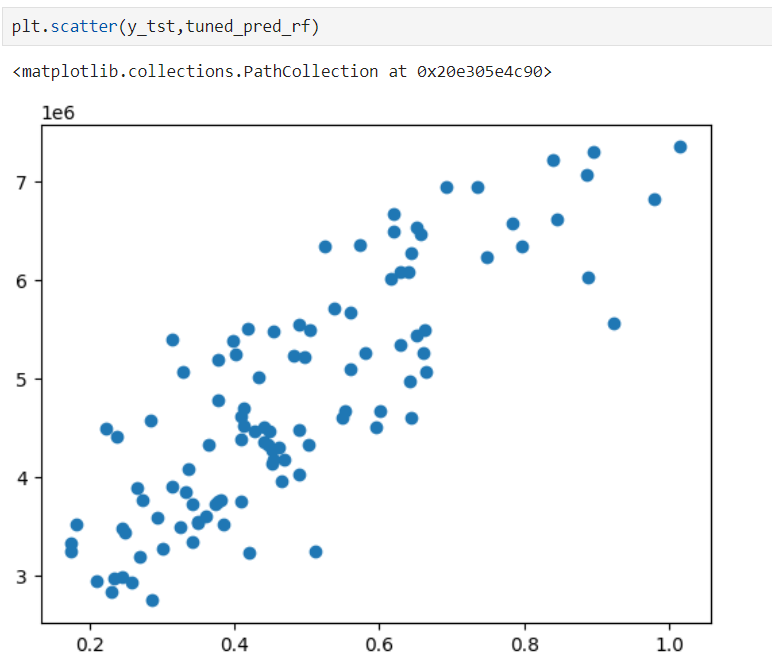
After getting the best parameters, I have trained the model using best parameters and evaluate the model accuracy using MAE, MSE, r2\_score, adj\_r2 score.

Below is a code snippet from my project that shows steps I followed to train my random forest model using hyperparameters.



Grid Search CV automates the process of hyperparameter tuning and helps in finding the combination that maximizes the model's performance. While it exhaustively searches through all combinations, it can be computationally expensive, especially with large hyperparameter spaces. Randomized Search CV is an alternative that randomly samples combinations, providing a more computationally efficient option.

**Plotting observed data and tuned predicted data**



From above plot, I can say I have a better graph with data points depicting a linear relationship

After training the models with best parameters, evaluated the model again using error metrics.

**Evaluation of Tuned model**

In this project, I have evaluated the tuned model using error metrics:

* Mean absolute error
* Mean Squared error
* R2\_Score
* Adj\_R2 Score



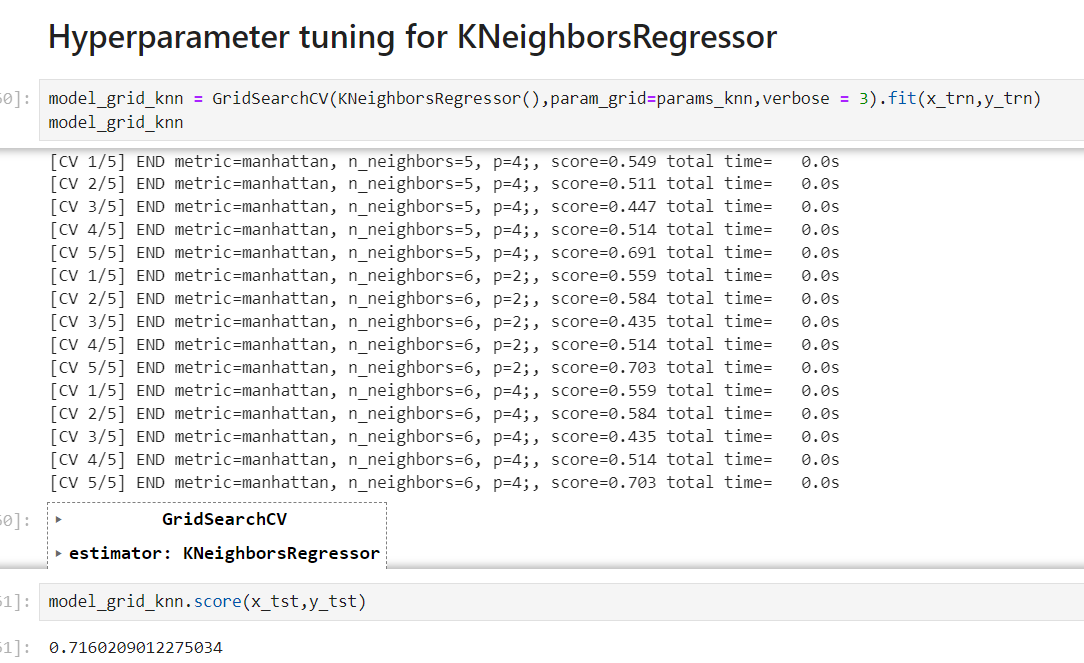


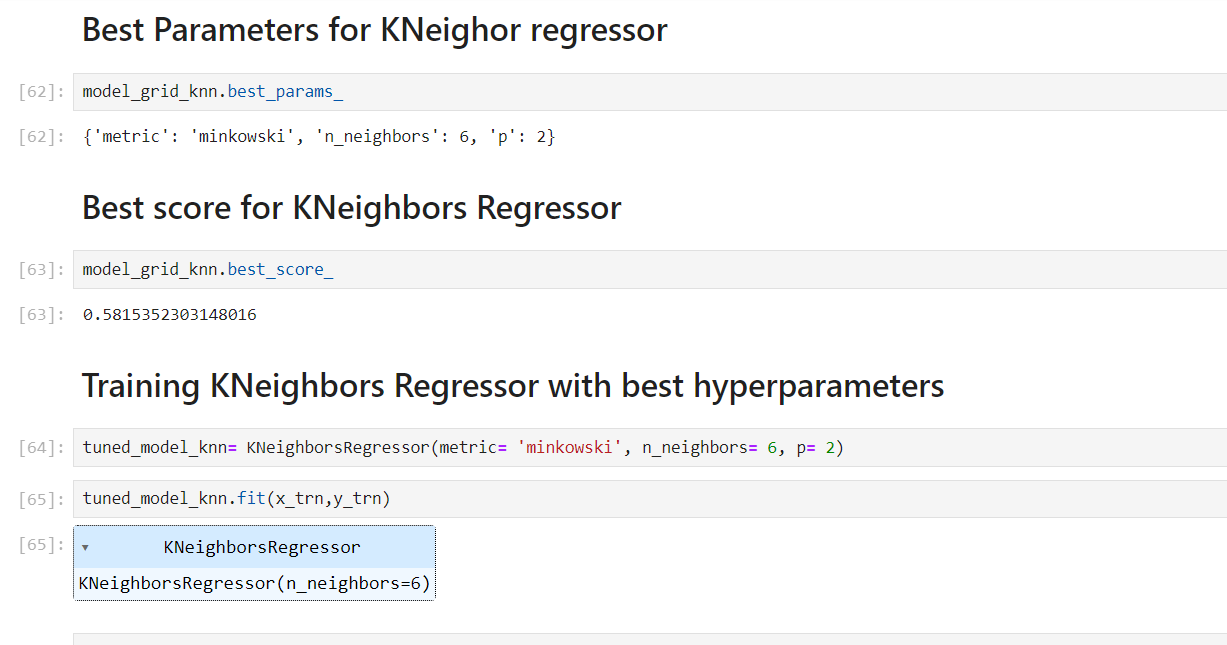
**Result of evaluation**

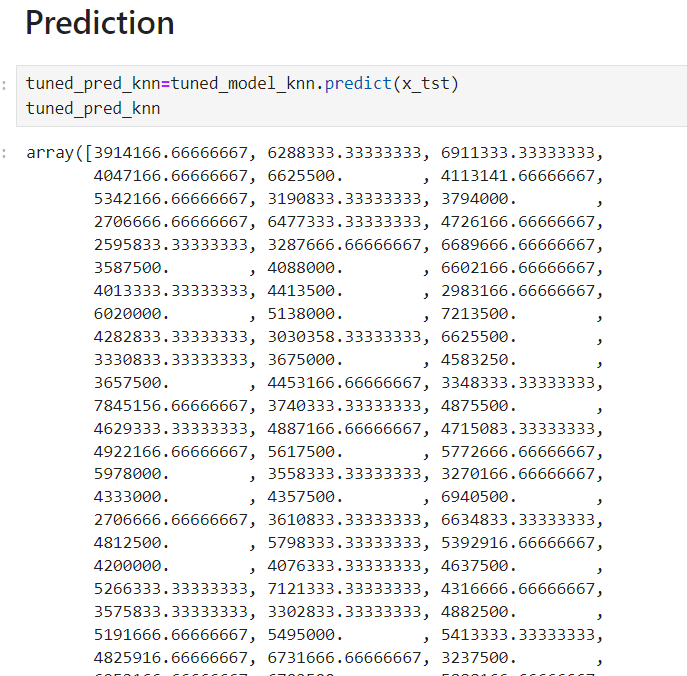
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model Name** | **Mean absolute error** | **Mean squared error** | **R2\_score** | **Adj\_R2\_score** |
| **Random Forest Regressor** | 849559.1086973929 | 1258605072668.2327 | 0.6518699779747845 | 0.6433470207500618 |
| **KNN** | 787451.7777777779 | 1026678285795.5553 | 0.7160209012275034 | 0.7090684939129226 |
| **SVM** | 1517262.4370092608 | 3995271105312.47 | -0.10509153990655395 | -0.13214651169334335 |

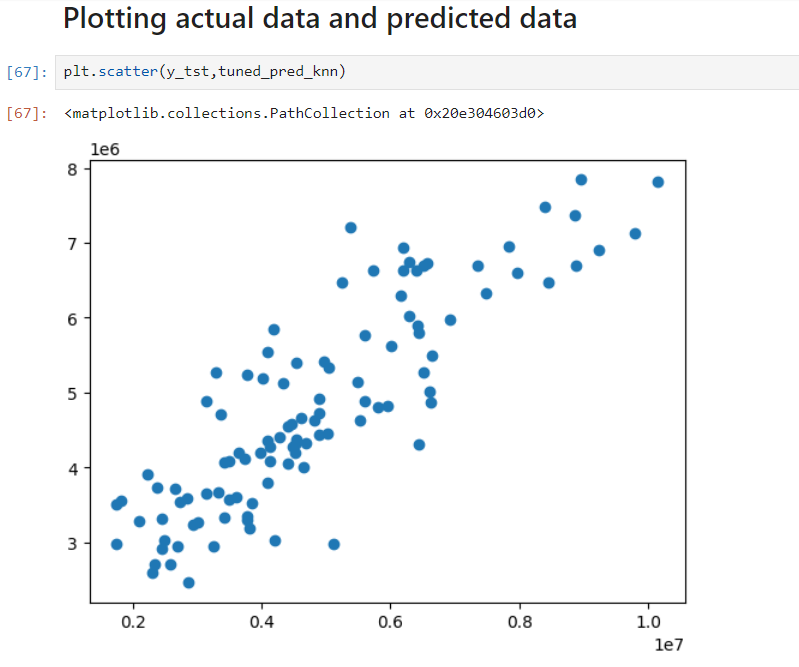
**Code samples:**

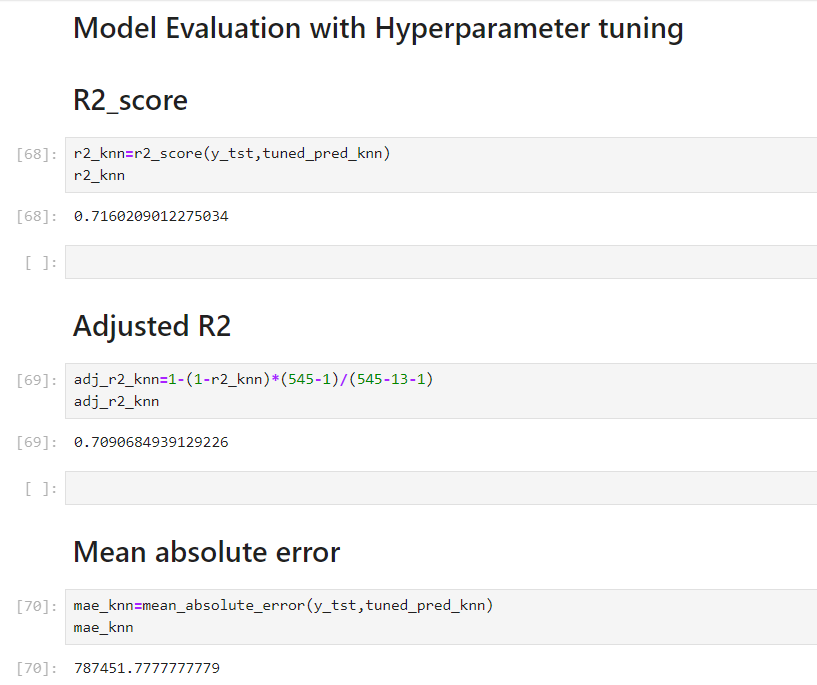
Attached some of the code samples of model building and evaluation

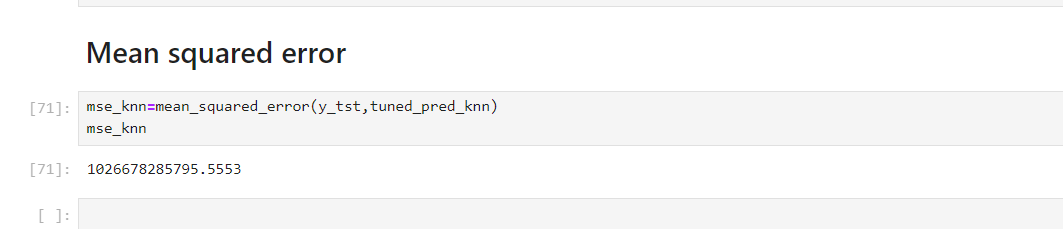
****

****

****

****

****

****

**Conclusion**

From this project, I conclude that:

* Before hyperparameter tuning, I was getting good accuracy only in Linear Regression
* Linear regression r2\_score is 0.7071147043895634
* I got better result for Random forest and KNN after hyperparameter tuning as compared to before hyperparameter tuning
* K-Neighbors regressor gives better result as compared to other algorithms after hyperparameter tuning
* For K-neighbors regressor r2\_score is 0.7160209012275034
* For Random Forest is r2\_score is 0.6445585363759831.
* For SVM, r2\_score is -0.10509153990655395
* Linear regression, KNN models have relatively higher accuracy compared to other models.
* Support Vector Regressor has a negative accuracy, indicating poor performance.
* Random Forest Regressor has moderate accuracy.
* Its visible that KNN has better r2\_score value.
* KNN gives better prediction than other algorithms
* There is a lot of scope of improvement for this model