Machine Learning Capstone Project on House Price Prediction using Advanced Regression Techniques

Authors: Indra Chatterjee, Manju Kiruthiga, Ranganathan Sriraman, Sunil Kumar, Shyamla

Project Mentor: Mr. Sanjoy Krishna Ghosh

Project Coordination: Ms. Rashika

Contents

Figures

Acknowledgement

First and foremost, we would like to express our special thanks of gratitude to **Mr. Sanjoy Krishna Ghosh (Project Mentor)** who guided us with this project and has given innumerable, invaluable advice throughout. His guidance helped tremendously to the successful completion of this project.

Besides, we would like to thank Ms. Rashika (Project coordinator), Great Learning institution to have given us this golden opportunity to do a wonderful project on the topic of House Price Prediction using Machine Learning.

Thank you.

Chapter 1: Introduction

1.1 Problem statement

Creation of Intelligent Regression based data model to predict house/home prices on basis of sales data in Seattle region from 2014 to 2015. Data models take into account various features like area, location, amenities and condition

1.2 Background

The dataset given to us is a House price sale data based from Seattle, WA during 2014-15. We will be using **supervised machine learning** algorithms since the given dataset has input variables with known output price. The outcome of our project is to make predictions on the sales prices of the houses of Seattle with the dataset provided.

In machine learning (ML), statistical methods are used to empower machines to learn without being programmed explicitly. The field focuses on letting algorithms learn from the provided data, collect insights, and make predictions on unanalyzed data based on the gathered information.

In general, ML is based on three key models of learning algorithms:

- Supervised learning - a dataset is present with inputs and known outputs
- Unsupervised learning - the machine learns from a dataset that comes with input variables only
- Reinforcement learning - algorithms are used to select an action

1.3 Dataset (Feature descriptions)

The dataset contains House Sale data of Seattle, WA area from 2014-May through 2015-May. The shape of the dataset is 21613*23 (23 attributes and 21613 observations). Below information would describe the features given.

Price is the target attribute and the remaining 22 Features acts as predictors.

Table 1- illustrates the Feature Name and a high-level description of the Features.

S.No	Feature Name	Feature Description
1	cid	A unique 7 to 10 digit ID representing the house property. This column is a transaction key (possibly auto-generated) & do not add much to the price prediction.
2	dayhours	Represents when the date/time when the sale happened. This column do not add much to the price prediction.
3	price	Selling price of the house. This will be our target variable.
4	room_bed	Represents number of bedrooms in the house.
5	room_bath	Represents number of bathrooms in the house.

		• 0.25 bathroom is a bathroom that has either a sink, a shower, toilet or a bathtub • 0.5 bathroom is a half bath/powder room typically a Toilet and sink • 0.75 is a bathroom that contains one sink, one toilet and a shower or a bath
6	living_measure	Represents square footage of living area in the home (carpet area)
7	lot_measure	Represents square footage of the lot
8	ceil	Represents number of floors/levels in the house
9	coast	Is a boolean variable representing whether the house has a water front or not
10	sight	Is a boolean variable representing whether the house has been viewed by potential clients or not
11	condition	Represents overall condition of the house on a scale of 5
12	quality	Grade given to housing unit based on grading system ranges from 1 to 13
13	ceil_measure	Represents square footage of the house except basement
14	basement	Represents square footage of the basement
15	yr_built	Year when this house was built originally, it basically represents the age of the house
16	yr_renovated	Year when the house was renovated. We assume that the last renovation year is captured in the data set
17	zipcode	Represents zipcode of the property
18	lat	Represents latitude of the property
19	long	Represents longitude of the property
20	living_measure15	Represents square footage of living area in the home (carpet area) after renovation 2015
21	lot_measure15	Represents square footage of lot area in the home after renovation 2015
22	furnished	Is a boolean variable representing whether the property is furnished (personal property)
23	total_area	Represents total sum of both living and lot measure

Chapter 2: Planning & Execution

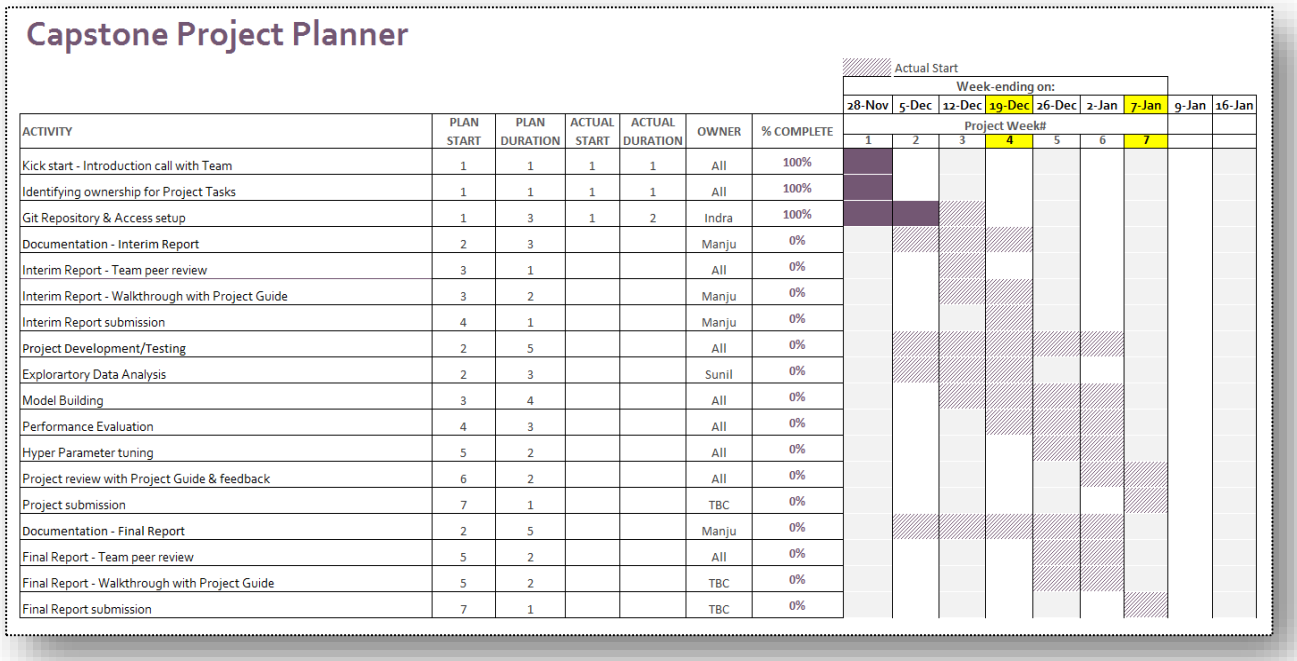
2.1 Project Plan

To provide an overall understanding of the progress that needs to be made in order to complete the project achieving all milestones we have made use a Gantt chart. The project plan will be reviewed and updated periodically, to be maintained in a centralized cloud GitHub repository.

A snapshot of the project plan is given below for illustration purposes.

This Project spans for 7 weeks, starting 28-Nov-2021 and ends on 07-Jan-2022. The progress of the project is tracked weekly. The 2 yellow highlights denote the Milestone delivery.

- 19-Dec is the milestone for the Interim Report submission &
- 07-Jan is the milestone for the Final project code and Final Report submission.



Capstone Project Planner

Actual Start

ACTIVITY	PLAN START	PLAN DURATION	ACTUAL START	ACTUAL DURATION	OWNER	% COMPLETE
Kick start - Introduction call with Team	1	1	1	1	All	100%
Identifying ownership for Project Tasks	1	1	1	1	All	100%
Git Repository & Access setup	1	3	1	2	Indra	100%
Documentation - Interim Report	2	3			Manju	0%
Interim Report - Team peer review	3	1			All	0%
Interim Report - Walkthrough with Project Guide	3	2			Manju	0%
Interim Report submission	4	1			Manju	0%
Project Development/Testing	2	5			All	0%
Explorartory Data Analysis	2	3			Sunil	0%
Model Building	3	4			All	0%
Performance Evaluation	4	3			All	0%
Hyper Parameter tuning	5	2			All	0%
Project review with Project Guide & feedback	6	2			All	0%
Project submission	7	1			TBC	0%
Documentation - Final Report	2	5			Manju	0%
Final Report - Team peer review	5	2			All	0%
Final Report - Walkthrough with Project Guide	5	2			TBC	0%
Final Report submission	7	1			TBC	0%

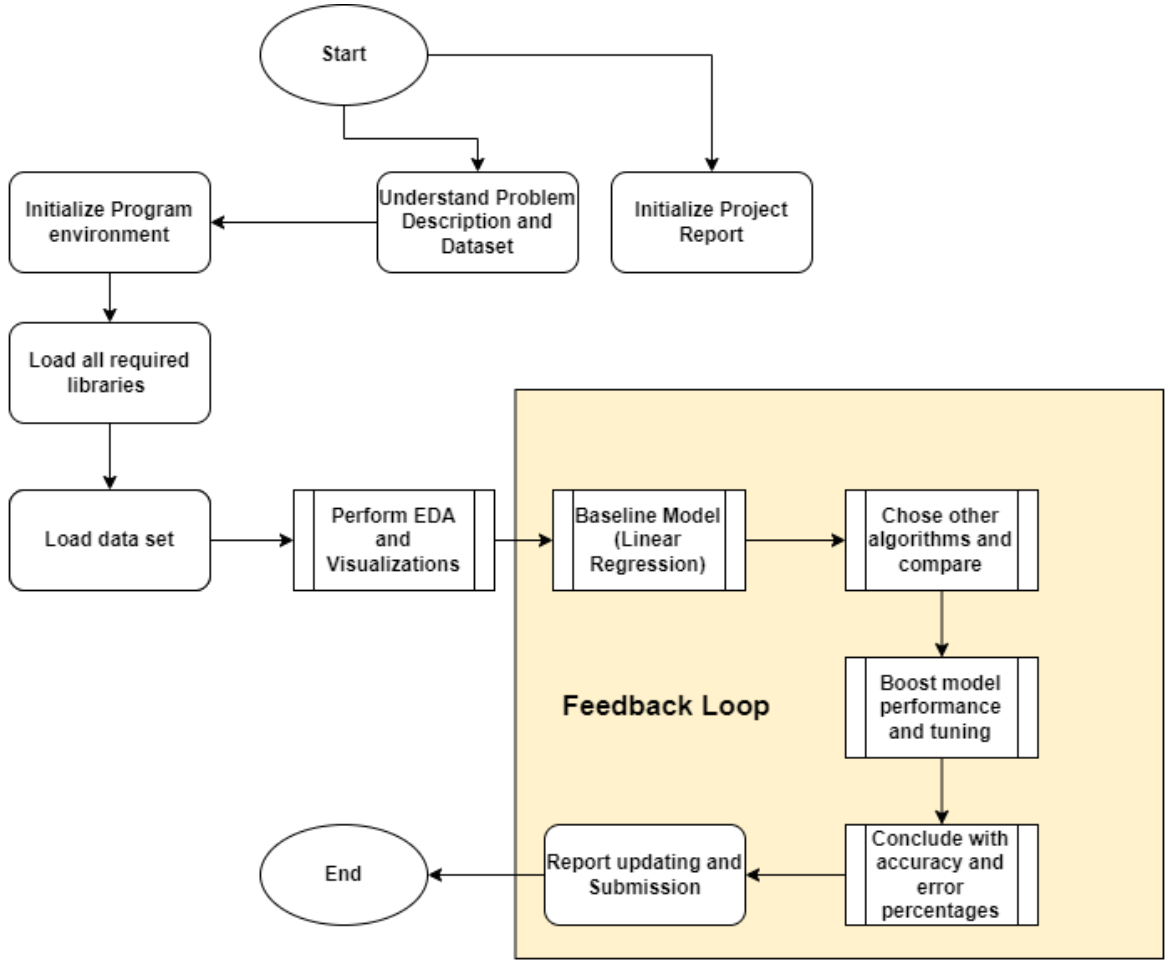
Week-ending on:	28-Nov	5-Dec	12-Dec	19-Dec	26-Dec	2-Jan	7-Jan	9-Jan	16-Jan
Project Week#	1	2	3	4	5	6	7		

Figure 1 Snapshot of the Gantt chart of the project plan for illustration purposes

Gantt project planner_v0.2_Weekl

2.2 Flow Chart

Flow chart below depicts the project life cycle.



Sequence of operations to successfully predict house price for a given region based on sales data for a specific period

Figure 2 Project Flow Chart

Chapter 3: Data Insights - Exploratory Data Analysis

3.1 Five Point Summary

The five-point summary involves the calculation of 5 summary statistical quantities, namely:

- **Median**: The middle value in the sample, called as the 50th percentile or the 2nd quartile.
- **1st Quartile**: The 25th percentile.
- **3rd Quartile**: The 75th percentile.
- **Minimum**: The smallest observation in the sample.
- **Maximum**: The largest observation in the sample.

Below code and output depicts the 5-point summary statistical quantities of the given House price prediction dataset.

```
In [7]: #Summarization of the data or calcuating the 5 point summary i.e. the Min, 25th , 50th (Median), 75th Percentiles, Max value
        houseDataSet.describe().transpose()
```

Out[7]:

	count	mean	std	min	25%	50%	75%	max
cid	21613.0	4.580302e+09	2.876566e+09	1.000102e+06	2.123049e+09	3.904930e+09	7.308900e+09	9.900000e+09
price	21613.0	5.401822e+05	3.673622e+05	7.500000e+04	3.219500e+05	4.500000e+05	6.450000e+05	7.700000e+06
room_bed	21613.0	3.370842e+00	9.300618e-01	0.000000e+00	3.000000e+00	3.000000e+00	4.000000e+00	3.300000e+01
room_bath	21613.0	2.114757e+00	7.701632e-01	0.000000e+00	1.750000e+00	2.250000e+00	2.500000e+00	8.000000e+00
living_measure	21613.0	2.079900e+03	9.184409e+02	2.900000e+02	1.427000e+03	1.910000e+03	2.550000e+03	1.354000e+04
lot_measure	21613.0	1.510697e+04	4.142051e+04	5.200000e+02	5.040000e+03	7.618000e+03	1.068800e+04	1.651359e+06
ceil	21613.0	1.494309e+00	5.399889e-01	1.000000e+00	1.000000e+00	1.500000e+00	2.000000e+00	3.500000e+00
coast	21613.0	7.541757e-03	8.651720e-02	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	1.000000e+00
sight	21613.0	2.343034e-01	7.663176e-01	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	4.000000e+00
condition	21613.0	3.409430e+00	6.507430e-01	1.000000e+00	3.000000e+00	3.000000e+00	4.000000e+00	5.000000e+00
quality	21613.0	7.656873e+00	1.175459e+00	1.000000e+00	7.000000e+00	7.000000e+00	8.000000e+00	1.300000e+01
ceil_measure	21613.0	1.788391e+03	8.280910e+02	2.900000e+02	1.190000e+03	1.560000e+03	2.210000e+03	9.410000e+03
basement	21613.0	2.915090e+02	4.425750e+02	0.000000e+00	0.000000e+00	0.000000e+00	5.600000e+02	4.820000e+03
yr_built	21613.0	1.971005e+03	2.937341e+01	1.900000e+03	1.951000e+03	1.975000e+03	1.997000e+03	2.015000e+03
yr_renovated	21613.0	8.440226e+01	4.016792e+02	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	2.015000e+03
zipcode	21613.0	9.807794e+04	5.350503e+01	9.800100e+04	9.803300e+04	9.806500e+04	9.811800e+04	9.819900e+04
lat	21613.0	4.756005e+01	1.385637e-01	4.715590e+01	4.747100e+01	4.757180e+01	4.767800e+01	4.777760e+01
long	21613.0	-1.222139e+02	1.408283e-01	-1.225190e+02	-1.223280e+02	-1.222300e+02	-1.221250e+02	-1.213150e+02
living_measure15	21613.0	1.986552e+03	6.853913e+02	3.990000e+02	1.490000e+03	1.840000e+03	2.360000e+03	6.210000e+03
lot_measure15	21613.0	1.276846e+04	2.730418e+04	6.510000e+02	5.100000e+03	7.620000e+03	1.008300e+04	8.712000e+05
furnished	21613.0	1.966872e-01	3.975030e-01	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	1.000000e+00
total_area	21613.0	1.718687e+04	4.158908e+04	1.423000e+03	7.035000e+03	9.575000e+03	1.300000e+04	1.652659e+06

Figure 3 Five-point statistical summary of the given dataset

Observation from the above figure:

- Looking through the distribution of features coast, sight, basement, yr_renovated and furnished it is **left skewed**. It is clearly evident from the above summary that the values for minimum, 25 percentile, 50 percentile and 75 percentile is 0. This is also referred as negative skewness in data.

- Some of the attributes seems like **normally distributed since the mean values are most probably equal to median values**: room_bed, room_bath, ceil, condition, yr_built, zipcode, lat and long.

3.2. EDA and Visualizations

The preliminary analysis of data to discover relationships between measures in the data and to gain an insight on the trends, patterns, and relationships among various entities present in the data set with the help of statistics and visualization tools is called Exploratory Data Analysis (EDA). Exploratory data analysis is cross-classified in two different ways where each method is either graphical or non-graphical. And then, each method is either univariate, bivariate or multivariate.

3.2.1: Univariate and Bivariate Analysis (Selective features)

Univariate Analysis: 'Uni' means one and variate means variable, so in univariate analysis, there is only one dependable variable. The objective of univariate analysis is to derive the data, define and summarize it, and analyze the pattern present in it. In a dataset, it explores each variable separately. Some patterns that can be easily identified with univariate analysis are Central Tendency (mean, mode and median), Dispersion (range, variance), Quartiles (interquartile range), and Standard deviation.

Distribution of Price

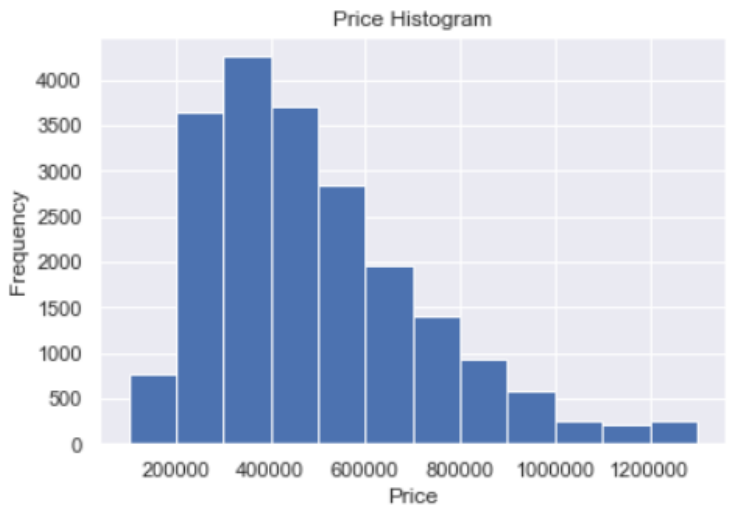


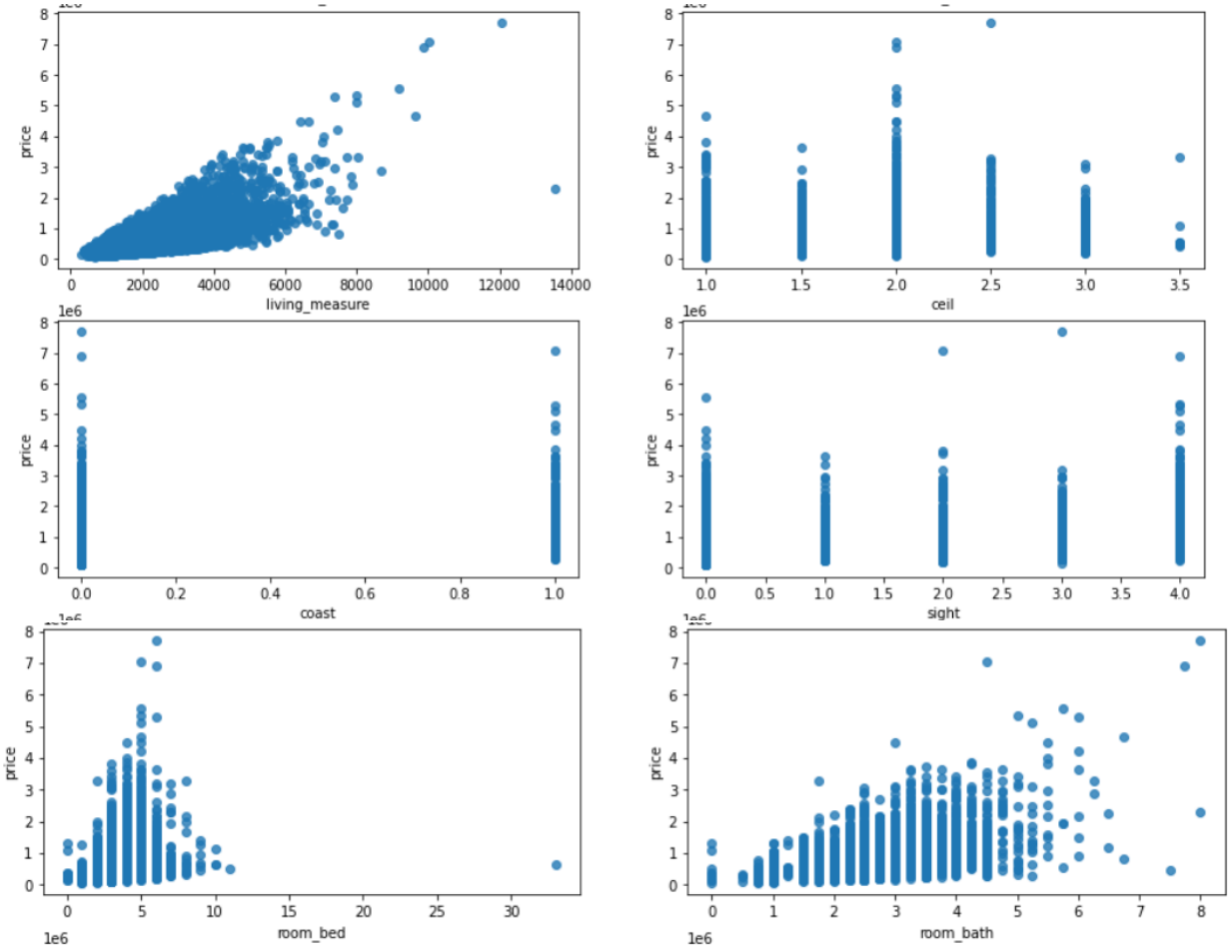
Figure 4 Distribution of Price

Observation: Target variable Price distribution is shown in the picture. It is clearly understandable that the target variable is right skewed which is a positive skewness. Most of the house property prices ranges in 40000

Bivariate Analysis: Bi means two and variate means variable, so here there are two variables. The analysis is related to cause and the relationship between the two variables. There are three types of bivariate analysis. There are three types of bivariate analysis.

- Bivariate Analysis of two Numerical Variables (Numerical-Numerical)
- Bivariate Analysis of two categorical Variables (Categorical-Categorical)
- Bivariate Analysis of one numerical and one categorical variable (Numerical-Categorical)

Plot of Key Features vs. Price



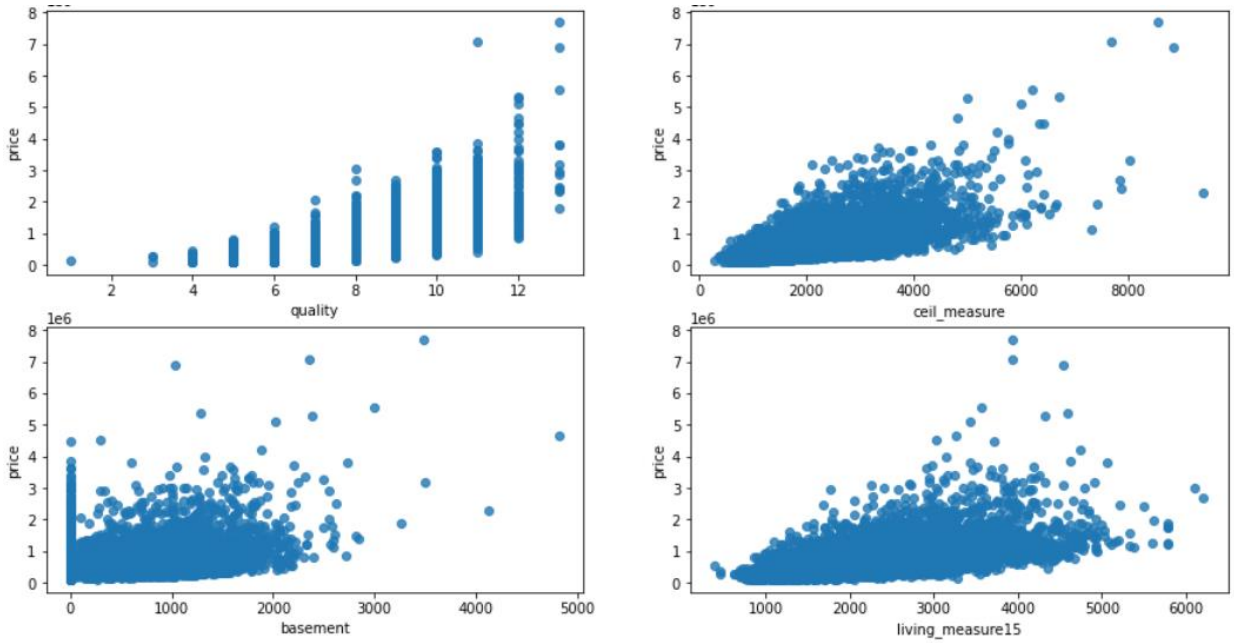


Figure 5 Bivariate Plots regression plots (Price)

Above depicted plots explain:

- Some of the continuous variables like living_measure, ceil_measure and living_measure15 are certainly affecting the target variable whereas basement is not probably affecting the price.
- Referring the categorical variables, quality plays a major role in prediction of prices but condition, sight and coast are going to affect much.

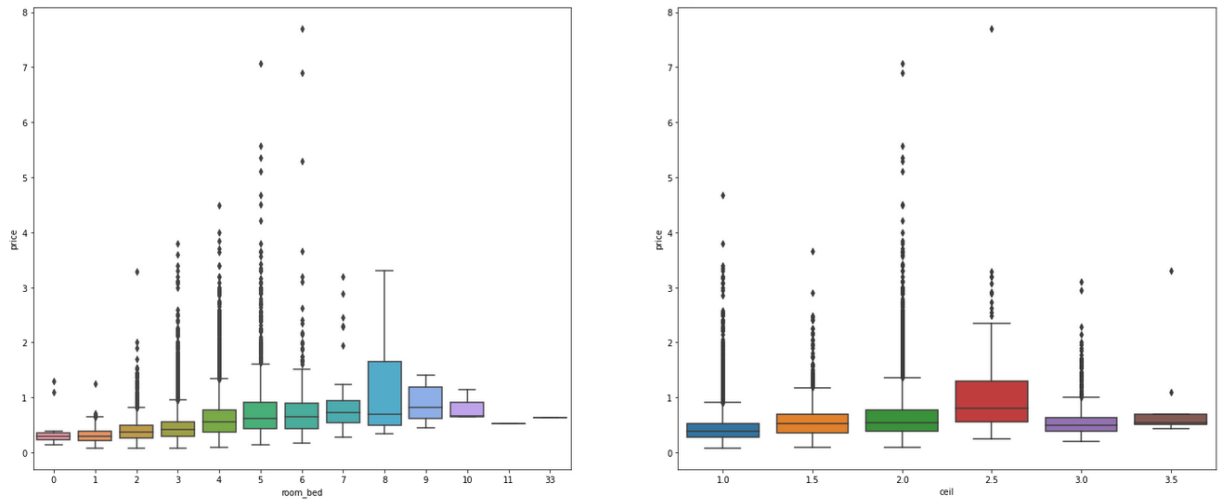
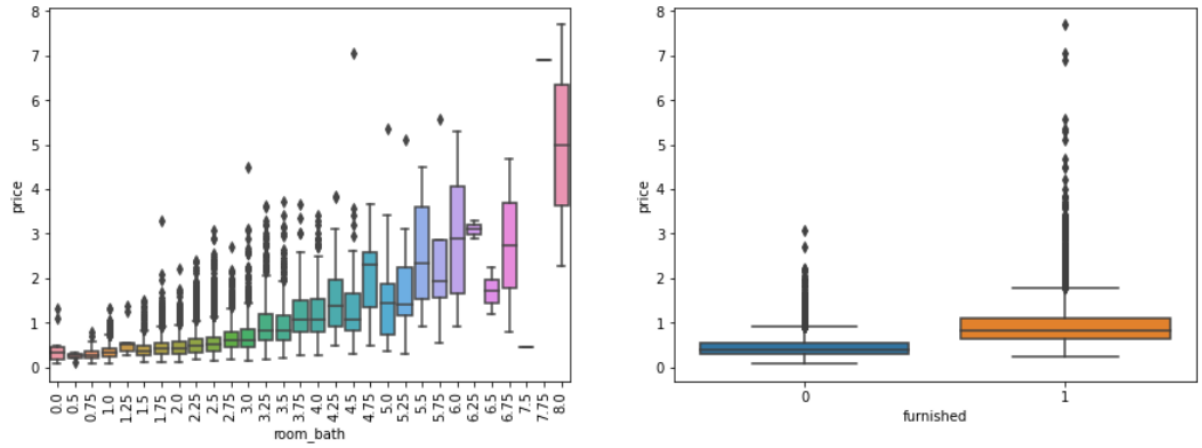


Figure 6 Box Plot (Showing outliers)



From the above charts, it can be seen that there are very few houses which have some features or price appears far from others like 33 bedrooms or price around 7000000. However, determining their possible negative effect will be time consuming and in the real data sets there will always be some outliers like some luxury house prices in this dataset. That's why it is better to not planning for remove outliers

3.2.2: Geographical data & Observations

In the Dataset given, we have Geographical information like Latitude, Longitude & Zipcode for each property. We examined further to see which Geographical area this data represents. The Geo information might not be directly relevant to the model building, but it was interesting to find out that the data is from properties in Seattle, WA, USA area. (zip codes 98001 to 98199). We plotted the data using the library, Folium over the Seattle map to see the distribution of the properties. The density of the properties of the given dataset can be clearly seen in Picture 5.

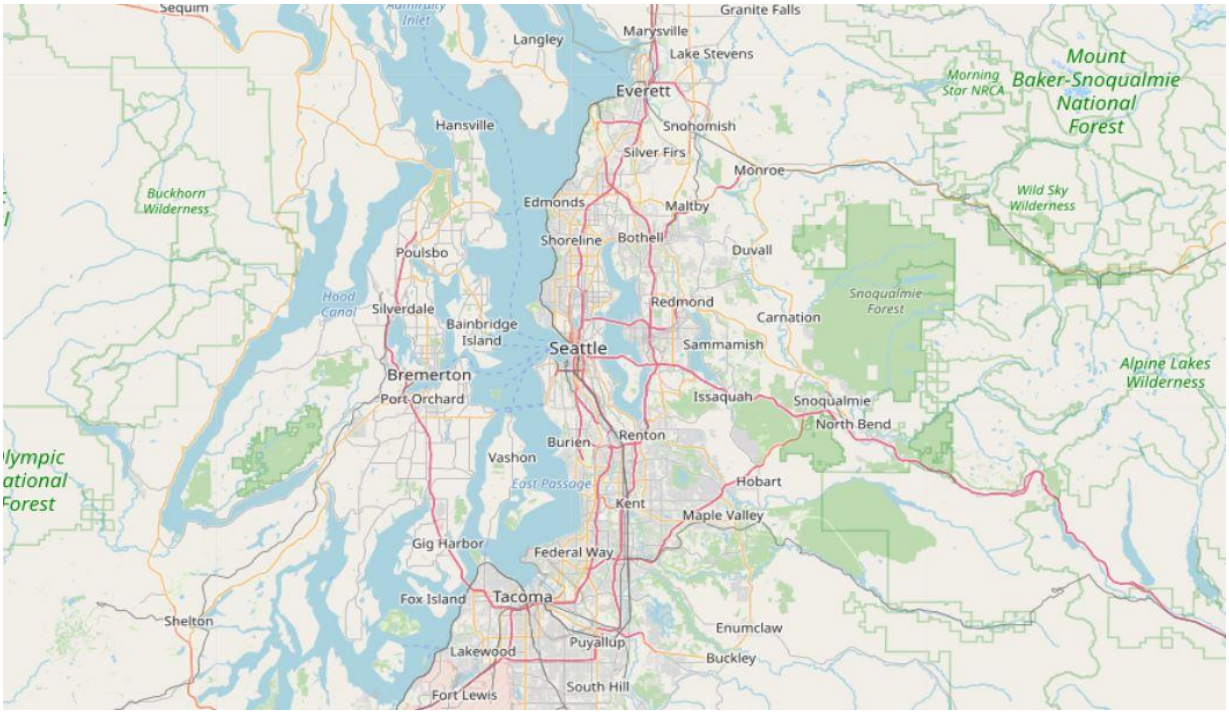


Figure 7 Plain Map of Seattle city.

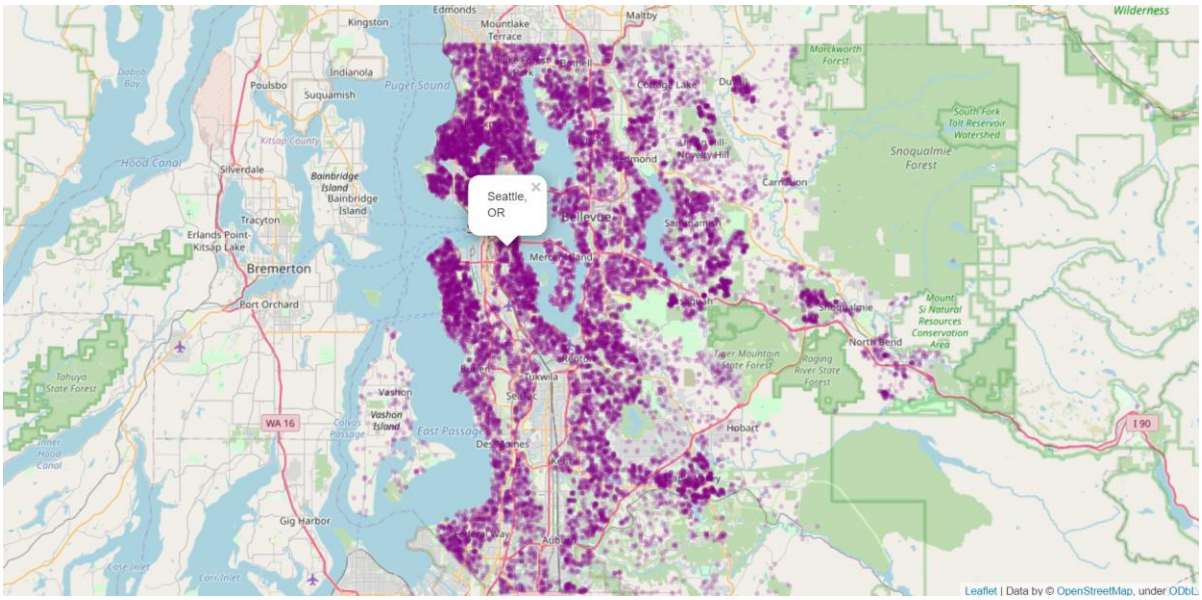


Figure 8 Dataset plotted on a map

3.2.3: Correlation plots (Heat Map)

Correlation is a statistical term describing the degree to which two variables move in coordination with one another. If the two variables move in the same direction, then those variables are said to have a positive correlation. If they move in opposite directions, then they have a negative correlation.

Here we are going with Pearson Correlation. There are other types of correlation also available like

- Positive, Negative or Zero Correlation
- Linear or Curvilinear Correlation
- Scatter Diagram Method
- Pearson's Product Moment Co-efficient of Correlation
- Spearman's Rank Correlation Coefficient

The correlation coefficient is a numerical value ranging from -1 to 1. Below table indicates what correlation coefficient value means to a given data.

Table 2- illustrates correlation coefficient

correlation coefficient	**Means**
1	The Features are highly Positively correlated with each other
0	The Features are not correlated with each other
-1	The Features are highly Negatively correlated with each other

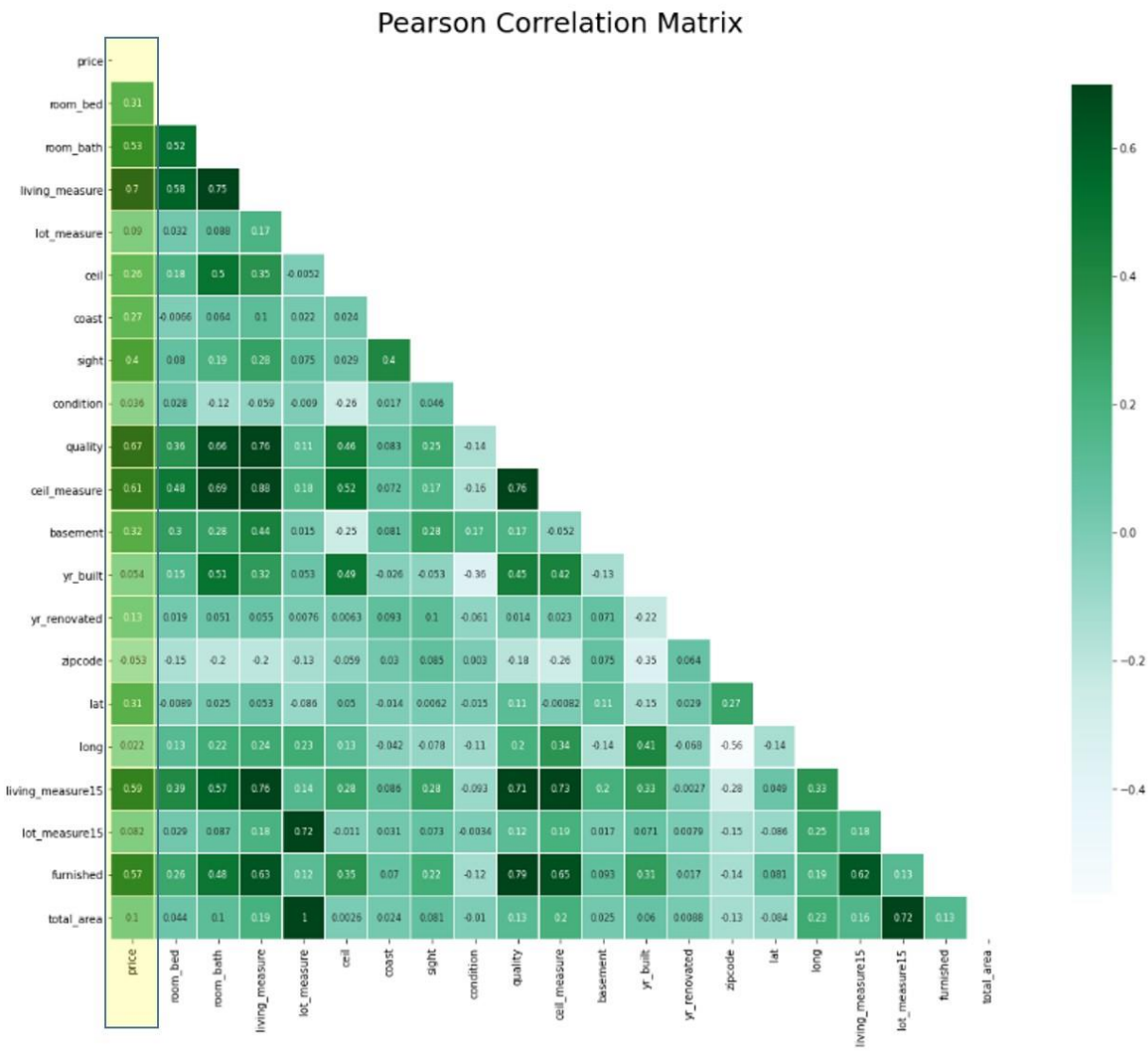


Figure 9 Heat Map (Correlation)

Observation from the Correlation heat map chart (Target Variable Vs. Features):

- **Highly positively correlated Features** – Features like living_measure (0.7), quality (0.67), ceil_measure (0.61), living_measure15 (0.59), furnished (0.57), rooms_bath (0.53) are more than 0 and closer to 1, hence highly positively correlated with the Target feature Price.
- **Not correlated Features** – Features like lot_measure (0.09), condition (0.03), zipcode (0.05), longitude (0.02), yr_built (0.05), lot_measure15 (0.08), total_area (0.1) are very close to 0, hence Not correlated with Price
- **Highly Negatively correlated Features** – From the correlation chart, we do not find any Negative correlation for features with price.
- Interesting observation here is on the yr_built feature. We usually assume that age of a property would be a key factor in determining the price. However, the yr_built has a correlation closer to 0. During the Development, we will examine further during the model

building to see if any new column ("building_age") can be derived based on yr_built and yr_renovation to improve the model efficiency.

- **Multicollinearity** occurs when two or more independent variables are highly correlated with one another in a regression model. This means that an independent variable can be predicted from another independent variable in a regression model. From the above heat map, it is clearly visible that there is a possibility for Multicollinearity hence we checked for VIF (Variable Inflation Factor)

	Feature	VIF
0	room_bed	2.335786e+01
1	room_bath	2.880139e+01
2	living_measure	inf
3	lot_measure	inf
4	ceil	1.689321e+01
5	coast	1.213027e+00
6	sight	1.552546e+00
7	condition	3.474618e+01
8	quality	2.102353e+02
9	ceil_measure	inf
10	basement	inf
11	yr_built	9.650585e+03
12	yr_renovated	1.195189e+00
13	zipcode	1.633655e+06
14	lat	1.390993e+05
15	long	1.361455e+06
16	living_measure15	2.724703e+01
17	lot_measure15	2.598952e+00
18	furnished	3.506156e+00
19	total_area	inf

Figure 10 A snapshot of the Variable inflation factor of the Features.

The dataset we have has only 22 Features and 1 Target column. Since the number of Features are only 22, not high we should be good to use all the Features given in the Dataset. Out of the 22 Features, we can clearly see that cid and dayhours are just audit columns and do not add any value to the model in terms of prediction. Based on above correlation heat map/VIF and summary listed, we identify the below features with a correlation less than 0.25 with target variable 'price' as potential columns to be excluded from the data. However, we will be experimenting the different Features to be used/eliminated in the models and finalize the Feature list.

cid
dayhours
zipcode
lot_measure
lot_measure15
long
condition
yr_built
yr_renovated
total_area

3.3 Outlier Summary (Identification and handling)

Based on the EDA, we can see some glaring observations.

Lot_measure: In general, 43560 sqft. makes 1 acre. From the below image & the data given, we can notice that there are many multi-acre properties in the dataset. Such big lot house are not very common family houses in a neighborhood within a city. We will be filtering the data to exclude homes that have >10-acre lot space i.e. lot_measure > 43560 sqft.

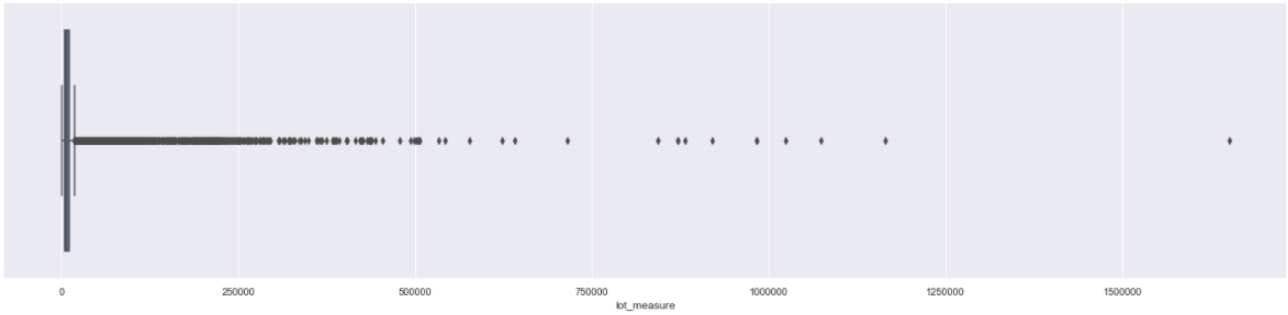
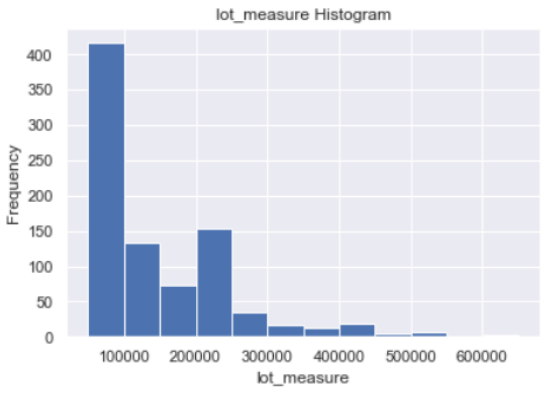


Figure 11 A Box plot illustrating the outlier in lot_measure feature..

Before handling lot_measure anomaly



After handling lot_measure anomaly

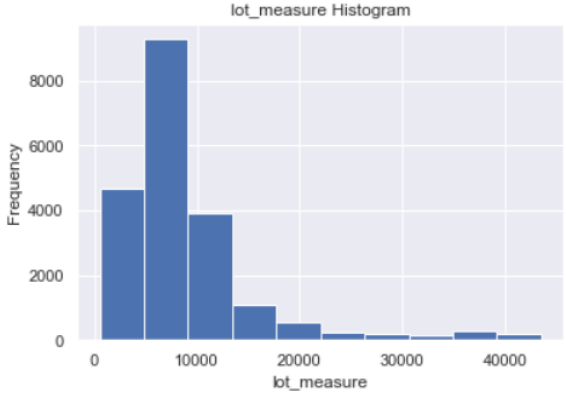


Figure 12 Histogram of lot_measure vs. frequency before & after the anomaly handling

room_bed: In the dataset given, for the record # 750, the no.of Bedroom is given as 33 while the living_measure and lot_measure are 1620 & 6000 sqft respectively and the no.of Bath is 1.75. For this lot size, 33 bedrooms seem abnormal and unrealistic. This could very well be a typo error during data collection. We can either impute this record (with average) or filter out this record. Since this is just 1 record with such anomaly, we are filtering out this record to avoid any skewness

Chapter 4: Model Selection and Building

4.1 Selection of Model

The problem at hand is prediction of house property prices. Here the Target variable is given & the Price feature is a continuous variable. Hence this would be a **Supervised - Regression problem**.

For the baseline model to bench mark the performance we used **LazyRegressor** algorithm, an offering from the *Lazy Predict* library.

Lazy predict library helps build a lot of basic models without much code and helps understand which models works better without any parameter tuning. Lazy predict library is a quick reference to understand the different algorithm performance to identify a range of the performance score and to bench mark the performance scores. Lazy predict offers algorithm for both Classification and Regression problems.

4.2 Model Baseline

4.2.1 Data Segregation

The model building process starts with separating features for our model from the target variable. Notice that in our case all columns except *'Price'* are features that we want to use for the model.

- ✓ Our target variable is '*Price'* - y_data
- ✓ After dropping cid and dayhours columns the remaining features are candidates for Predictors – x_data

4.2.2 Feature Scaling

Feature scaling is a method used to normalize the range of independent variables or features of data. In data processing, it is also known as data normalization and is generally performed during the data preprocessing step. As part of the Baseline model, Models were built with and without scaling for observation purposes to see if any differences were noticed. In this case, the scaling did not alter the performance measure of the models while testing with LazyRegressor.

We used **MinMaxScaler function from sklearn** python library for feature scaling.

4.2.3 Model Training and Testing

We are using **train_test_split** from the function **scikit-learn** to divide features data (x_data) and target data (y_data) even further into train and test. We can control the train_test split fraction by using the ***test_size*** parameter. Note that we had it set to 0.3 in our example. It can be any number between 0.0 and 1.0. As part of the model tuning and performance improvement, we will be experimenting with different test size like 0.25, 0.2 to see how the overall model behavior is.

4.2.4 Baseline

For the baseline, we experimented with 4 models.

Models 1 & 2 were built and tested with 19 Features (except cid and dayhours). The models were tried with and without scaling and the Max R Squared score, Best model from the LazyRegressor was observed to be **XGBRegressor**.

Models 3 & 4 were built and tested with 11 Features. As per the EDA analysis the below Features were excluded and model was built using LazyRegressor.

lot_measure (0.09)
condition (0.03)
zipcode (0.05)
longitude (0.02)
yr_built (0.05)
lot_measure15 (0.08)
total_area (0.1)
The co-relation coefficients are given in the bracket for reference.

The models were tried with and without scaling and the Max R Squared score, best model from the LazyRegressor was observed to be **XGBRegressor**.

	Model 1 & 2 (with X=20 Features)**		**Model 3 & 4 (with X=12 Features)****	
	without scaling	with scaling	without scaling	with scaling
Max R-Squared	0.89	0.89	0.83	0.83
Best Model (based on R-Square, RMSE & Execution time)	XGBRegressor	XGBRegressor	XGBRegressor	XGBRegressor

	Features used**	
Feature Names	**Models 1 & 2**	**Models 3 & 4**
cid	N	N
dayhours	N	N
room_bed	Y	Y
room_bath	Y	Y
living_measure	Y	Y
lot_measure	Y	N
ceil	Y	Y
coast	Y	Y
sight	Y	Y
condition	Y	N
quality	Y	Y
ceil_measure	Y	Y
basement	Y	Y
yr_built	Y	N
yr_renovated	Y	N
zipcode	Y	N
lat	Y	N
long	Y	N
living_measure15	Y	Y
lot_measure15	Y	N
furnished	Y	Y
total_area	Y	N

The overall score of these models with and without Scaling is not too different, however the Execution time was less while using scaling.

The LazyRegressor Report for Models 1 & 2 (with scaling) is as below:

Model	Adjusted R-Squared	R-Squared	RMSE	Time Taken
XGBRegressor	0.89	0.89	123600.79	1.28
LGBMRegressor	0.88	0.88	129032.10	0.29
RandomForestRegressor	0.88	0.88	129264.08	13.59
HistGradientBoostingRegressor	0.87	0.87	133068.31	1.54
BaggingRegressor	0.86	0.86	137264.61	1.83
GradientBoostingRegressor	0.85	0.86	141740.71	4.90
ExtraTreesRegressor	0.85	0.85	144110.72	7.81
KNeighborsRegressor	0.76	0.76	182528.61	2.44
DecisionTreeRegressor	0.75	0.75	185421.38	0.29
PoissonRegressor	0.72	0.72	195321.56	0.12
GammaRegressor	0.70	0.70	202539.80	0.06
LassoLars	0.68	0.68	209838.17	0.05
LassoCV	0.68	0.68	209838.78	2.00
Lasso	0.68	0.68	209839.70	3.06
Lars	0.68	0.68	209840.06	0.08
LassoLarsIC	0.68	0.68	209840.06	0.05
LarsCV	0.68	0.68	209840.06	0.15
LassoLarsCV	0.68	0.68	209840.06	0.13
Ridge	0.68	0.68	209840.61	0.03
RidgeCV	0.68	0.68	209845.63	0.07
BayesianRidge	0.68	0.68	209848.57	0.06
LinearRegression	0.68	0.68	209897.25	0.05
TransformedTargetRegressor	0.68	0.68	209897.25	0.05
SGDRegressor	0.68	0.68	210209.97	0.09
ElasticNet	0.65	0.65	219965.29	0.14
ExtraTreeRegressor	0.65	0.65	220789.50	0.11
HuberRegressor	0.64	0.64	224092.98	0.20
PassiveAggressiveRegressor	0.63	0.63	226437.59	0.84
OrthogonalMatchingPursuitCV	0.62	0.62	228296.52	0.10
GeneralizedLinearRegressor	0.62	0.62	230440.85	0.12
TweedieRegressor	0.62	0.62	230440.85	0.05
OrthogonalMatchingPursuit	0.57	0.57	244205.07	0.04
RANSACRegressor	0.46	0.46	273607.39	0.22
AdaBoostRegressor	0.32	0.32	307401.65	1.93
ElasticNetCV	0.02	0.02	368193.96	0.29
DummyRegressor	-0.00	-0.00	372251.50	0.03
NuSVR	-0.03	-0.02	376386.61	12.17
SVR	-0.06	-0.05	382230.34	12.24
MLPRegressor	-1.09	-1.08	536915.03	22.17
KernelRidge	-1.41	-1.40	576364.41	27.19
LinearSVR	-1.96	-1.95	639149.39	0.04
GaussianProcessRegressor	-5.21	-5.19	926396.72	84.05

Figure 133 Lazy Regressor Report-1

The LazyRegressor Report for Models 3 & 4 (with scaling & with reduced Features) is as below:

Model	Adjusted R-Squared	R-Squared	RMSE	Time Taken
XGBRegressor	0.83	0.83	153634.83	1.08
RandomForestRegressor	0.83	0.83	154177.81	8.47
HistGradientBoostingRegressor	0.82	0.82	157782.26	1.45
GradientBoostingRegressor	0.82	0.82	158838.70	2.36
LGBMRegressor	0.82	0.82	159198.79	0.32
ExtraTreesRegressor	0.81	0.81	162259.93	4.25
BaggingRegressor	0.81	0.81	163398.85	0.65
PoissonRegressor	0.73	0.73	194292.41	0.11
KNeighborsRegressor	0.71	0.71	198914.50	1.10
GammaRegressor	0.67	0.67	212985.72	0.05
DecisionTreeRegressor	0.67	0.67	214596.75	0.11
Lasso	0.64	0.64	223664.47	2.32
LassoLarsIC	0.64	0.64	223664.82	0.05
Lars	0.64	0.64	223664.82	0.10
Ridge	0.64	0.64	223665.47	0.03
LassoLars	0.64	0.64	223666.35	0.05
LassoLarsCV	0.64	0.64	223668.05	0.15
LassoCV	0.64	0.64	223669.47	0.70
RidgeCV	0.64	0.64	223671.32	0.05
BayesianRidge	0.64	0.64	223673.37	0.04
LarsCV	0.64	0.64	223682.81	0.15
LinearRegression	0.64	0.64	223742.53	0.05
TransformedTargetRegressor	0.64	0.64	223742.53	0.04
SGDRegressor	0.64	0.64	223814.91	0.08
ExtraTreeRegressor	0.62	0.62	228235.01	0.11
OrthogonalMatchingPursuitCV	0.62	0.62	228296.52	0.09
ElasticNet	0.61	0.61	231393.50	0.07
HuberRegressor	0.59	0.59	237556.02	0.17
TweedieRegressor	0.59	0.59	239533.23	0.04
GeneralizedLinearRegressor	0.59	0.59	239533.23	0.25
PassiveAggressiveRegressor	0.58	0.59	239727.85	0.57
RANSACRegressor	0.51	0.51	261411.65	0.18
OrthogonalMatchingPursuit	0.50	0.50	263410.05	0.03
AdaBoostRegressor	0.35	0.35	299688.26	0.85
ElasticNetCV	0.02	0.02	368363.78	0.17
DummyRegressor	-0.00	-0.00	372251.50	0.02
NuSVR	-0.02	-0.02	376269.04	12.19
SVR	-0.06	-0.05	382023.72	12.40
MLPRegressor	-1.22	-1.21	553626.21	26.92
KernelRidge	-1.45	-1.44	581884.58	28.77
LinearSVR	-1.95	-1.95	639149.39	0.06
GaussianProcessRegressor	-3184.85	-3178.65	20990651.67	78.22

Figure 144 Lazy Regressor Report-2

Based on the different experiments done with the LazyRegressor, we can determine that XGBRegressor is the "Best" performing model for this given dataset. **XGBRegressor** is the chosen model.

We are getting R-Squared score range of 83 to 89% using XGBRegressor model.

Chapter 5: Model Tuning and Cross Validation

Hyperparameter tuning: All machine learning models have hyperparameters which you must tune or set such that the given model is customized according to the dataset. In definition hyperparameters are model configuration argument specified by the developer to guide the learning process for a specific dataset. Cross-validation is often used to estimate this generalization performance.

Cross-Validation is a resampling technique with the fundamental idea of splitting the dataset into 2 parts- training data and test data. Train data is used to train the model and the unseen test data is used for prediction. If the model performs well over the test data and gives good accuracy, it means the model hasn't over fitted the training data and can be used for prediction.

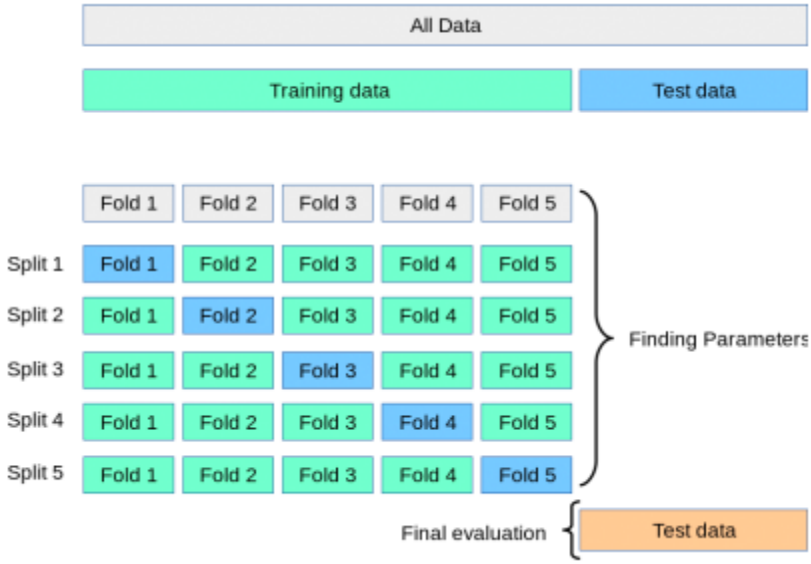


Figure 155 Cross validation overview

In this chapter we would be dealing with hyper parameter tuning in order to increase our regression model accuracy. To select the best set of hyperparameters for a model for a given dataset is often challenging.

There are different types of CV techniques available and here we will look at grid and Randomized search in this section.

5.1 Grid Search CV

Defines a search space as a grid of hyperparameter values and evaluate every position in the grid. It helps to loop through predefined hyperparameters and fit the estimator (model) on training set. So, in the end, we can select the best parameters from the listed hyperparameters.

```
run_Grid_Search_CV(houseDataSet.drop(columns=['cid', 'dayhours'],axis=1), 'price')
```

```
{'model': 'XGBRegressor', 'Y Predictions': array([365495., 767014., 982501., ..., 240596., 223385., 447881.],
      dtype=float32), 'Best accuracy': '87%', 'Best Params': {'gamma': 0, 'max_depth': 5, 'sampling_method': 'uniform', 'subsample': 1},
'Test set accuracy': '88%'}
```

Here we used Grid Search CV on the given dataset which is giving the optimal hyperparameters to achieve the best accuracy score for the model. The Grid search CV gave us a best score of 88%

5.2 Random Search CV

Random search is a technique where random combinations of the hyperparameters are used to find the best solution for the built model. This works best under the assumption that not all hyperparameters are equally important

```
run_Random_Search_CV(houseDataSet.drop(columns=['cid', 'dayhours'],axis=1), 'price')
```

```
best score of Randomized Search : Best accuracy:88%
```

We have explored both Grid search CV & Random search CV in the previous step. Both Grid search CV and Random search CV gave us a similar score. However, the execution time i.e. performance of the Random search CV was quicker compared to the Grid search CV.

5.3 Principal component analysis (PCA)

Principal Component Analysis, or PCA, is a dimensionality-reduction method that is often used **to reduce the dimensionality of large data sets**, by transforming a large set of variables into a smaller one that still contains most of the information in the large set. PCA helps in identifying relationships among different variables & then coupling them.

The dataset given to us has only 21 features (cid, dayhours columns were removed because they were audit columns and not offering much insights for the prediction). The 21 features in comparison is not a high number of dimensions. However, we tried PCA technique on the dataset given to identify feature importance.

```
run_model_pipeline(houseDataSet.drop(columns=['cid', 'dayhours'],axis=1),'price', isPCA=True)
```

```
Pipeline (with PCA) Test score: 0.78
```

```
run_model_pipeline(houseDataSet.drop(columns=['cid', 'dayhours'],axis=1),'price', isPCA=False)
```

```
Pipeline (without PCA) Test score: 0.89
```

The observation while testing is that, with all 21 features without PCA, our model score was 89% while with PCA using 17 Features, the model score was 78%. Here the number of features reduced/eliminated is only 4 however the difference or improvement in the model performance was significant. Considering that the number of features reduced and the model score difference is negligible, **we decided to proceed with a model with all 21 Features without PCA**.

	With PCA	Without PCA
Number of Features	17	21
Accuracy score	78%	89%

5.4 Pipelining

A machine learning pipeline is the end-to-end construct that orchestrates the flow of data into, and output from, a machine learning model (or set of multiple models). It includes raw data input, features, outputs, the machine learning model and model parameters, and prediction outputs.

Machine learning (ML) pipelines consist of several steps to train a model. Machine learning pipelines are iterative as every step is repeated to continuously improve the accuracy of the model and achieve a successful algorithm.

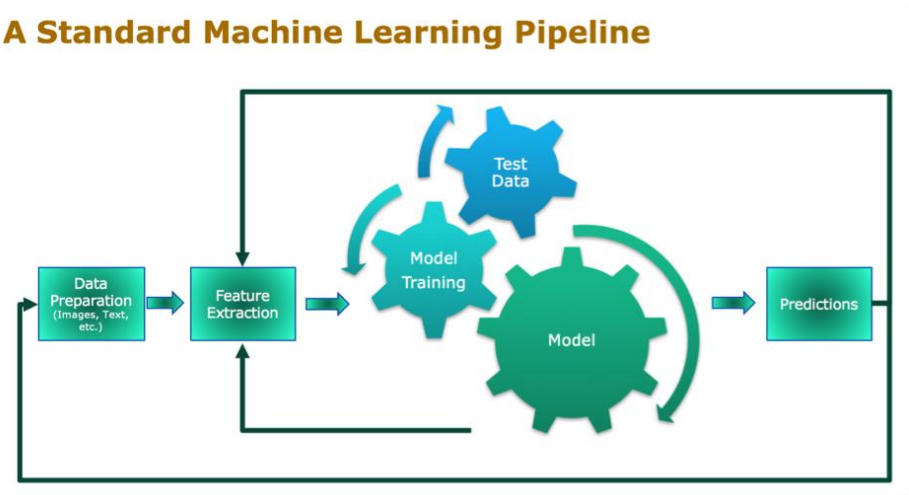


Figure 166 Pipeline overview

As highlighted in the above section, we tried implementing 2 Pipelines.

- Pipeline 1: Standard scalar --> PCA --> Model (XGB)
- Pipeline 2: Standard scalar --> Model (XGB)

For our final model, we are opting with Pipeline2 for the reasons mentioned above.

```
#Creating a Pipeline here with MinMaxScaler --> PCA() --> Model (XGB Regressor)
if isPCA:
    pipe = Pipeline([(" scaler", MinMaxScaler()), ('pca', PCA()), (" xgb", XGBRegressor())])
else:
    pipe = Pipeline([(" scaler", MinMaxScaler()), (" xgb", XGBRegressor())])
```

We created a Boolean variable isPCA to handle the above mentioned Pipeline execution of with and without PCA.

Chapter 6: Pickling and Model deployment

Pickle is a module in Python used for serializing and de-serializing Python objects. This converts Python objects like lists, dictionaries, etc. into byte streams. You can convert the byte streams back into Python objects through a process called **unpickling**. Pickling is also known as serialization, flattening, or marshalling.

We are pickling the XGB regressor model which is the best model after CV analysis in our code.

Model Pickling, XGB Regressor is our best model / linear regressor algorithm

In [38]:
```
# Defining constants

PICKLE_MODEL_FILE_NAME="xbgRegressor_model.pickle"
PICKLE_MODEL_INSTANCE=regression_model
```

In [39]:
```
# Writing or creating Pickle file here

with open(PICKLE_MODEL_FILE_NAME, 'wb') as model_pickle_file:
  pickle.dump(PICKLE_MODEL_INSTANCE, model_pickle_file)

print('Model Pickled successfully')
```

```
Model Pickled successfully
```

Unpickling the pickled model and checking the test scores.

Reading Pickle file to run predictions

In [40]:
```
test_model = pickle.load(open(PICKLE_MODEL_FILE_NAME, 'rb'))
print(test_model)
print(f'Accuracy score of model prediction data  : {test_model.score(X_test, Y_test)}')
```

```
XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
             importance_type='gain', interaction_constraints='',
             learning_rate=0.300000012, max_delta_step=0, max_depth=6,
             min_child_weight=1, missing=nan, monotone_constraints='()',
             n_estimators=100, n_jobs=0, num_parallel_tree=1, random_state=0,
             reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
             tree_method='exact', validate_parameters=1, verbosity=None)
Accuracy score of model prediction data  : 0.8857067989988064
```

Chapter 7: Conclusion

7.1 Summary

In the beginning of this Capstone project, our aim was to build an Intelligent Regression model to predict the House prices on the basis of various features related to the sales data in Seattle region from 2014-15 that were provided to us. We identified the Dependent and the Target variables, carried out EDA (Exploratory Data Analysis) and Data visualization on the data to identify the patterns, relationships, multi-collinearity, correlation and anomalies of features with respect to the Target variable. We used Folium library to plot the Geo-graphical information provided in the dataset to visualize the distribution of the properties over a Seattle city map.

Using the Lazy Regressor library, we identified a Baseline model i.e. XGBRegressor based on the model performance and execution time. Once the Baseline was identified, we used different Feature Engineering techniques like Cross validation – Grid search and Random search, PCA, Pipeline to fine tune and identify the best parameters for the model to achieve a max score. We were able to get a score of about 89% for our final model.

We used tools like Gantt chart for project planning and Git-hub for Collaboration and version control.

Overall we had a good learning experience understanding the nuances of executing a Machine Learning project.

7.2 Model Performances (Execution times)

The team members used our Personal laptops (mostly i3 or i5 processors) for the Model building and Performance tuning. While using Lazy Regressor, Cross validation and performance tuning exercises we noticed that the over-all execution time was high. These are CPU intensive operations and with a limited processing capacity, the high execution time was understandable. The number of Features given in the dataset was 23, however if the numbers were high then we would have had performance challenges with the limited processing capacity. One thing we would like to explore in the future is to subscribe for a Cloud based GPU with a leading Cloud service provider with higher processing capacity to handle the CPU intensive operations.

7.3 Limitations

During the Capstone project, the team observed the below limitations with the dataset given to us. This is just a recommendation from our end that may be considered for future Capstone project as necessary. The dataset given to us had 23 features and the complexity of the data is not high. During EDA, we observed Multi-collinearity among some of the Features and this is understandable because of the nature of the data. It would have been interesting to get additional features with different multi-collinearity, correlation aspects.

Also, with only 23 features, the complexity of Feature selection exercise, PCA was easy. A dataset with higher number of Features could have been challenging and played a better learning experience.

7.4 Future improvements

The dataset provided to us for the Capstone project was a historic one (2014-15) and is also specific to a Region i.e. Seattle, USA.

- One of the future improvement could be to expand this data for other Regions.
- We may consider including other factors like Cost Inflation index, consumer price index, etc. to give an additional dimension with respect to the Target variable, Price.

Bibliography

References

1. https://olympus.mygreatlearning.com/courses
2. Pickling Machine Learning Models. Train your ML models once and use them… | by Aryan Deore | Better Programming
3. Mastering Exploratory Data Analysis(EDA) For Data Science Enthusiasts (analyticsvidhya.com)
4. https://www.kaggle.com/alexisbcook/interactive-maps
5. https://anaconda.org/conda-forge/folium
6. Linear Regression Algorithm To Make Predictions Easily (analyticsvidhya.com)
7. https://www.mygreatlearning.com/blog/gridsearchcv/
8. https://xgboost.readthedocs.io/en/stable/parameter.html