Czech University of Life Sciences Prague

Faculty of Economics and Management

Department of information engineering

Bachelor Thesis

Data science

Nihar Lathiya

CZECH UNIVERSITY OF LIFE SCIENCES PRAGUE

Faculty of Economics and Management

BACHELOR THESIS ASSIGNMENT

Bc. Nihar Mukeshbhai Lathiya

Systems Engineering and Informatics
Informatics

Thesis title

Data science

Objectives of thesis

Undoubtedly data science is most evolutionary technology of the era. It's all about deriving useful insight from the data in order to solve real world problem. Now a days data mining plays very important and demanding role in business organization, business terms may not have knowledge or required technical skills to manipulate data and create information from it. In this case data science is a field in which big data is a new oil to extract useful information from it and display applicable ideas to members of business organization to increase potential of their business.

The main objective of this thesis is to derive valuable information from Big data to make strategic decisions which can be beneficial for business goals in future.

Methodology

To obtain an objective, it's required to understand the business problem, define objectives for the problem that needs to be tackled. Now for data acquisition, data will be gathered from multiple sources like-web servers, logs, databases and online repositories.

The next step will be data cleaning and complex data transformation which can be performed by ETL tools.

Thus, we can define and refines the selection of features variables that will be used in model development. Now we proceed to the core activity of a data science project which is data modelling by applying force machine learning techniques to the data to identify the moral that best suits business requirements.

Then it's come to create powerful report or dashboards by using visualization tools and communication. This reports or dashboards will be used to get real-time analytics.

The proposed extent of the thesis

30 – 40 pages

Keywords

Data science, Big data, data mining, Machine learning, decision making

Recommended information sources

ECKERSON, Wayne W. Performance dashboards: measuring, monitoring, and managing your business. John Wiley & Sons, 2010.

CHAUDHURI, Surajit; DAYAL, Umeshwar; NARASAYYA, Vivek. An overview of business intelligence technology. Communications of the ACM, 2011, 54.8: 88-98.

KIMBALL, Ralph; ROSS, Margy. The data warehouse toolkit: the complete guide to dimensional modeling. John Wiley & Sons, 2011.

TYRYCHTR, J. – VASILENKO, A. Business Intelligence in Agribusiness – Fundamental Concepts and Research. Brno: KONVOJ, spol. s r. o. , 2015, 100s. ISBN 978-80-7302-170-2.

Expected date of thesis defence

2020/21 SS – FEM

The Bachelor Thesis Supervisor

Ing. Jan Tyrychtr, Ph.D.

Supervising department

Department of Information Engineering

Electronic approval: 19. 11. 2020	Electronic approval: 19. 11. 2020
Ing. Martin Pelikán, Ph.D.	**Ing. Martin Pelikán, Ph.D.**
Head of department	**Dean**

Prague on 22. 02. 2021

Declaration

I declare that I have worked on my bachelor thesis titled "Data science" by myself and I have used only the sources mentioned at the end of the thesis. As the author of the bachelor thesis, I declare that the thesis does not break copyrights of any their person.

In Prague on 2st March, 2021 ___________________________

Acknowledgement

I would like to begin by thanking my thesis supervisor Ing. Jan Tyrychtr, Ph.D. for his grate support and understanding in the completion of this thesis. His irreplaceable encouragement lead me to this success.

I would like to express my sincere thanks to my faculty professors and Czech University of Life Sciences, Prague for support during my graduation.

I would like to finish by thanking my parents for their constant help, support and love.

Data science

Abstract

In this thesis the development of house size prediction model for a real estate website is presented. This model is a function which requires inputs from customers. These input can be customer's desired requirements such as location, price, number of bedrooms and bathrooms. Based on these inputs, the model will predict estimated size of the house from which customer can choose the best.

For this purpose, I used python and its machine learning libraries like NumPy, Pandas, Matplotlib and Scikit Learn as a tool. These tools are very efficient and helpful to perform task of data cleaning and data modelling on a huge dataset. After data modelling there is introduced the machine learning process, which contains various methods, techniques and machine learning algorithm applications. Finally, I created house size prediction function based on the best suitable algorithm.

Machine learning is a huge area of data science studies that could be used in many different ways such as business, health sectors, education etc. This bachelor thesis particularly focused on real estate business and analyse how data science plays revolutionary role in real estate industry.

Keywords

Data science, Big data, Data cleaning, Machine learning, Python, Feature engineering

Datová věda

Abstrakt

V této práci je představen vývoj predikčního modelu velikosti domu pro realitní web. Tento model je funkce, která vyžaduje vstupy od zákazníků. Těmito vstupy mohou být požadavky zákazníka, jako jsou umístění, cena, počet ložnic a koupelen. Na základě těchto vstupů bude model předpovídat odhadovanou velikost domu, ze kterého si zákazník může vybrat nejlepší.

Za tímto účelem je použit nástroj Python a jeho knihovny strojového učení jako NumPy, Pandas, Matplotlib a Scikit Learn. Tyto nástroje jsou velmi účinné a užitečné při provádění úkolů čištění a modelování velkých dat. Po datovém modelování je představen proces strojového učení, který obsahuje různé metody, techniky a aplikaci algoritmu strojového učení. Nakonec vytvářím funkci predikce velikosti domu na základě nejvhodnějšího algoritmu.

Strojové učení je obrovská oblast studií datových věd, kterou lze použít mnoha různými způsoby, jako je obchod, zdravotnictví, vzdělávání atd. Tato bakalářská práce se zaměřila zejména na obchod s nemovitostmi a analyzovala, jak věda o datech hraje revoluční roli v realitním průmyslu.

Klíčová slova

Věda o datech, Velká data, Čištění dat, Strojové učení, Python, Návrh funkcí

Table of Contents

1. Introduction 12

2. Objectives and Methodology 14

2.1 Objectives 14

2.2 Methodology 14

3. Literature review 16

3.1 Machine learning 16

3.1.1 Data 19

3.1.2 Features 20

3.1.3 Machine learning algorithms 21

3.2 Cross-Validation in Model Selection 27

3.2.1 K-fold cross-validation 28

3.3 Hyper Parameter Tuning in Machine Learning 29

3.3.1 Grid search CV 29

3.4 Python for Data Science 29

3.5 Essential Python Libraries for Data Science 30

3.5.1 NumPy: 31

3.5.2 Pandas 33

3.5.3 Matplotlib 34

3.5.4 Scikit-Learn 35

4. Practical part 36

4.1 project tool 36

4.2 Dataset 36

4.3 data cleaning 38

4.4 Feature engineering 43

4.5 Outlier detection and removal 46

4.6 Machine learning model building 54

4.6.1 One-hot-encoding 54

4.6.2 Train-Test split 56

4.6.3 Machine learning algorithms and accuracy estimation 57

4.6.4 Function for home size prediction 60

4.6.5 Exporting model 60

5. Results and discussion 62

5.1 Result interpretation of house size estimation function 62

5.2 Discussion 62

6. Conclusion 64

7. References 65

List of figures

Figure 1: Algorithms employed in Machine Learning. 17
Figure 2: Linear Regression Graph.......... 22
Figure 3: Lasso regression 24
Figure 4: splitting in decision tree 25
Figure 5: pruned decision tree 27
Figure 6: Row dataset 37
Figure 7: Unnecessary columns dropped from dataset 38
Figure 8: detected range values in 'total_sqft' column 41
Figure 9: Converted 'total_sqft' into single values 42
Figure 10: Added new feature - 'price_per_sqft' 43
Figure 11: Dataframe after feature engineering 46
Figure 12: Square feet per bedroom Outliers detection 47
Figure 13: Price outliers of bhk detected 50
Figure 14: Price outliers of bhk removed 51
Figure 15: House density of price per square feet 52
Figure 16: bathroom outliers detected 53
Figure 17: Final dataframe after all outliner removal 54
Figure 18: dummy columns created for location 55
Figure 19: Converted location into numerical category with one-hot-encoding 56
Figure 20: Scores and parameters values comparison of algorithms 59

List of equations

Equation 1: Linear equation..22
Equation 2: Sum of square error...23
Equation 3: Lasso regression..23
Equation 4: variance..26

1. Introduction

Today we live in a world which generates tons of data, which gives birth to today's data driven technologies. Now a days we come across to these technologies in our day-to-day life, whether it is product recommendation on websites or social media, movie or show recommendation on various streaming service and many more. Data is being widely used in every industry such as IT technologies, health sectors, finance and marketing, real estate and so on. Here we can see that we have a great opportunity to tackle problems easily and effectively compared to previous era, which makes earth a better place to live.

Main reason is for rapidly growing revolution is "Big data" available now, which was not the case before. Now it is also possible to predict future from available data pattern of past years. Data is not only about the numbers, pictures, audio, or user information, but it is also about to observe human behaviour and their living pattern which can be used to develop automated technologies such as smart house, self-driving cars, gadgets like Alexa and various artificial intelligence software. All these technologies were unrealistic and untrusted by humans initially, which could be never possible without a problem and creative mind. As being said " necessity is mother of invention". (Plato, 2017)

Things like internet, data science, artificial intelligence (AI), machine learning (ML) cannot be underestimated for improvement of human life and technical development, especially after Covid-19 pandemic when digitalization took place in every industry. If we need an efficient and qualitive solution for any real-world problems, then individual should try possible modern approach to tackle it. This modern approach to use data for decision making, boost business and improve accuracy in various tasks, which is known as "Data science".

Undoubtedly data science is most demanding and profitable technology for any businesses. Any business growth is highly depended on how they collects meaningful data and make wise decisions for their business from it. If we take latest example of WhatsApp new privacy rule to use user data for Facebook's recommendations and advertisement, it might be inspired by Google. Most commercials prefer Google to advertise their product. The reason is Google receives plenty of users data from their various services and platform and the way they use it, is on next level. We can get this idea from experiencing most relevant product recommendation and offers on Google platform, which we browsed in chrome or you tube depending on our current location, but Facebook is still lacking in this area. So, this might be their new strategy to use WhatsApp data for improving their Facebook commercials quality.

If we take a look at real estate industry, it has been totally revolutionized by data science and machine learning technology now a days. We often see price prediction model on their websites to know worth of the house, for example Zillow have one model called ‘Zestimate’. It helps customers as well as the real estate agent for buying or listing house on a particular medium. That potential motivated me to create a unique approach in real estate industry, using data science and machine learning methods.

In this thesis, I will create a size-prediction model for a real-estate website. This model will work with numeric and text data, which will require inputs from user for their desired criteria and will give estimated possible size of house they will get. This model creation will include various data science techniques and machine learning algorithms. This unique approach to help customers for selecting their dream home will help real-estate site to improve their performance as well.

The theoretical knowledge for this task will be covered. I will discuss initial data structure and the change we make in it gradually to prepare it for model training. Different stages for this model training will be discussed with coding and description. Various machine learning algorithms are used to prepare model for high accuracy results.

2. Objectives and Methodology

2.1 Objectives

The thesis is focused on real estate business to create unique experience not only to satisfy their existing customers but create new impact in real estate market and attract more customers for them. Now a days common feature on each reputed real estate site we can see is house price estimation. It is a great innovation of data science field but to make it unique from the rest of websites I came across to house size prediction. It can be totally different and interesting approach to pay attention on customer's specific need. For example, customer always have their fixed budget which is most important and first requirement to focus on. This model will help them to select the best house they can get in their budget. Then it comes to number of bedrooms and how specious they are, number of baths, few desired location to select. Considering, all these factors they can have estimate of size of a house. After analyzing all results, they can choose bigger and specious house for them.

The main objective of this thesis is to derive valuable information from Big data to make strategic decisions which can be beneficial for business goals in future.

To obtain main objective, the development of this bachelor thesis will also tackle the creation of these 4 artifacts as partial objectives:

- Create literature review in the field of data science and machine learning methods.
- Analyze the python libraries for data science.
- Create size prediction model for a real estate website.
- Synthesize the results of practical part and propose recommendations for practice.

2.2 Methodology

The technical overview of the study is based on of scientific books and web resources related to data science techniques, machine learning and python. A systematic review of these aspects will be discussed in the literature review of this thesis. All the publications and literature were helpful during work. There are also citations available for some important parts.

First, I studied business requirements for a data science task, which is to build unique model of house size prediction for their website. Secondly, we will need data to train our model for further prediction. The data for this thesis will be directly downloaded from the Kaggle dataset (Chakraborty, 2017). This data will be used only for the study purpose of this thesis. We will primarily use Jupyter notebook with anaconda distribution as a tool to implement python programming language and windows operating system for our entire task.

As soon as we have data, we can start examining and exploring data. Row data will not be clean in majority of cases including our dataset in this work. For data cleaning we will use python libraries such as NumPy and Pandas. These libraries are very handy to handle numeric values and data frames. As we go further, we will need to visual our data for better understanding. We will use Matplotlib library for plotting graphs for data visualization. Once our data is cleaned, we will move towards feature engineering and outliers detection and we will prepare it for machine learning model.

For our machine learning part we will use supervised machine learning (Kotsiantis, 2007), which means our data set will be labelled with correct prediction values including training dataset. One-hot-encoding method will be used to reduce dimensionality of string data, which is location in our case. Train-test method will be used to apply machine learning algorithm on training data. Scikit learn model selection will be used to implement train test split method and for machine learning algorithm as well. K-fold cross validation and grid search CV will be used to come up with best scoring model and its parameters among three machine learning algorithms, which are linear regression, lasso regression and decision tree.

For saving model we will use pickle (.pkl) file and json file. Which will include final pre-trained model, model's weightage, embedding matrix and data columns. This saved model will be used in real estate website for house size prediction.

3. Literature review

3.1 Machine learning

Machine learning is a well-known concept in the domain of data science, artificial intelligence, and computer science - also known as statistical learning and predictive analytics. Arthur Samuel of IBM firstly used the machine learning term in 1952. In 1950, he wrote the first computer program. The program was a game of checkers, which makes winning strategies and incorporating moves. Later, Samuel also designed various mechanisms allowing his program (Foote, 2019).

As a data scientist, one must become familiar with machine learning concepts because both data science and machine learning overlap. We can say:

- Data science is used to gain insights from data and understanding of data patterns.
- Machine learning is used for making predictions based on available data.

The above predictions employ a set of artificial intelligence techniques that focuses on designing the system and uses statistical experience to improve model parameters tuning, the performance of the index, and improving its predictions, where experience can be previous information or data from broad and specific fields pooled at dataset hubs like Kaggle and made available for use in research. In other data science projects, critical measurement of these algorithms' validity and quality in their variables such as time complexity yields a robust system. Still, an additional notion of sample complexity is required for the algorithm to learn data patterns. In short, theoretical learning guarantees for an algorithm depends on the complexity and size of the training data sample (Mehryar Mohri, 2018).

Machine learning is all about getting computers to make data-driven decisions rather than being explicitly programmed to carry out specific tasks. It enables computing devices to employ embedded programs and generated algorithms to predict instantaneous states just like humans do. Since programming is automation, machine learning is the core of the process. The latter process is a way that makes programming scalable. In conventional programming, data is fed as the input, while programs lie at the core programs to manipulate it and give an output, which is also a dataset. This concept is employed further in Machine

Learning but with improved datasets and robust algorithms to achieve similar goals (Brownlee, 2015).

There are three main types of machine learning algorithms.

However, there exists another technique under the name of Reinforcement learning. All these algorithms have their differences stem from the way they treat the training and testing datasets.

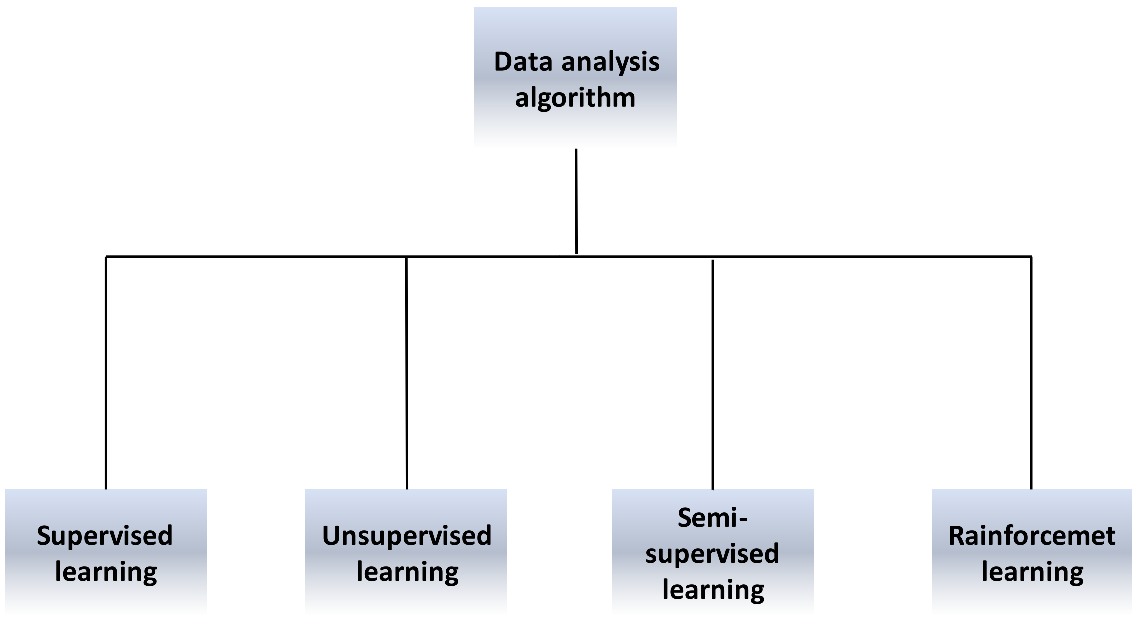


Figure 1: Algorithms employed in Machine Learning.

Source: Author

<u>Supervised machine learning</u>

A supervised learning algorithm comprises an outcome variable, which is also referred to as a dependent variable. This variable needs to be predicted by imposing the independent variables to a learning technique. The technique's overall goal is to generate a model, which resembles a function that can map output from a given set of inputs. Model training runs until the computer finds a suitable model while not compromising the accuracy of the results. This technique's critical points are that it has labelled data, gives direct feedback, and predicts the outcome or future.

Under this category are Random Forest, Linear regression (most common), k- nearest neighbor design tree.

Unsupervised learning

The learning algorithms, in this case, do not contain the outcome or dependent variable for prediction. In these algorithms, data is without a label, it does not give feedback, and it helps find hidden data structure in data. Unsupervised learning algorithms cannot be directly applied to regression or classification problems because we do not know what output data values might be. It makes it impossible to train data than we usually do in supervised learning. It can be used for the clustering population in different groups for specific interventions.

E.G., K- means clustering, Gaussian mixture models, etc.

Reinforcement learning

In this category, the machines are trained to make only specific decisions while ignoring others. It has a reward system and learns a series of actions, implying that the variables in question are manipulated through the trial-and-error mechanism until they fit the desired output. It can be observed that this form of learning uses experience as the core of its decision making. The result is that only the best decisions are employed; hence reliability is optimal.

An example is the Markov decision process.

There are three main components of the machine learning system: data, features, and ML algorithms.

In our study, we seek to employ a Supervised Learning algorithm to achieve our project's objectives. Since we seek to create a model that can easily predict the price of houses in India, we must have data to train and test the model. Supervised learning allows us the degree of freedom to choose the appropriate model that predicts house prices accurately. Data for this project is downloaded from Kaggle (Chakraborty, 2017)

3.1.1 Data

Data can be collected manually or automatically. It can be in any unprocessed structure, text, value, images, audio, etc. data is a crucial factor in data analysis, ML, and A.I. It is not possible to train our model without data.

Therefore, big enterprises spending vast amounts of money on getting to access those data nowadays. Generally, in machine learning, we divide data into two parts- "training data" and "testing data." Since we downloaded data from an online repository, Kaggle.com, we know that this data should be split into the two sections given before. However, as we understand, we cannot split this data until we have cleaned it. Data cleaning ensures the data is in good form, with only the required variables being put into play and observing the dataset's features.

Much time is bound to be spent on data cleaning. This is the most time-consuming part. The modeling data's data should be logically viable without outliers, which are the most common error sources in the modeling paradigm. Data cleaning involves eliminating all the data values with *null* or *NA* as values in their cells. This is done either by replacing the missing values with a median value of the column it resides. The other alternative is the exclusion of missing values (rows) entirely from the dataset. It becomes reasonable to perform the latter decision when the missing variables' population compared to the entire population is small.

Training data :

It is a portion of our data that we show to our model as input and output. Based on this data, we train our model. The training dataset needs to form the majority of the sample. It should be large to overcome the bias introduced by the use of smaller data samples. As per our project, we split the primary dataset into an 80% training set. This was used to train the models on how to predict the price of houses given several input parameters. It is expected that some considerable amount of time would be used during training, and therefore this is to be treated as usual, and the model should be left to do its training.

Testing data:

After our model is thoroughly trained and ready for prediction, we feed the testing data values as an input and get predicted output by our model. This output might not be the same as the actual output of testing data, as our model has not seen it. So, we compare both outputs and check the accuracy of the model (Gupta, 2018). Model testing forms the last step in this project. It requires the minimal pseudo-random sample of the remaining population, which is 20% of the population. However, it should be noted that the sample should be randomized to have a clear picture of what the model portrays. Testing is done using any house from any location except the number of bedrooms and bathrooms with an expectation to have a reasonable house price prediction.

3.1.2 Features

Features are the measurable and observable properties in data that we are interested in analyzing. In datasets, features often appear as columns forming distinctive characteristics in each column. Features can also refer to as "variable" or "attributes." The feature selection process is varies depending on what we need to analyze in our model. Features are building blocks of a dataset. Quality of features if the dataset significantly affects the quality of insight we will gain from datasets. It is possible to improve the quality of feature selection with feature selection and feature engineering. It is typically tricky and tedious, but if it is done well, we can get optimum results of the dataset, containing all the essential features that might have beneficial insights to solve a specific business problem (LLC, 2019). These features are extracted during the data cleaning phase. It is also to state that the dataset characteristics, variables, including individual entries, are observed. These observations form how the required features are selected for the next phase in the data cleaning pipeline.

Once the required variables have been defined, all the other columns are then dropped as our goal is to have a data frame with only the factors on which our model will depend. In case additional variables can help reduce some of the variables, the variables are created, used until they are no longer required, at which they are dropped. As the features are extracted, the output data frame shrinks to the tune of the extracted features. The data shrinks at every stage until finally, we are comfortable using the data frame. In this instance, since we are employing a supervised learning technique, we have to ensure the proper data statistics are in order; for instance, we visualize the data to confirm that it is normally distributed. Once this criterion is met, we further confirm that there exist no anomalies in the data. This is achieved through scatter plots and observing the distribution of houses' prices in the land space.

3.1.3 Machine learning algorithms

Algorithms employed in Machine Learning exist in massive amounts, with many sprouting at the dawn of a day. Generally, data scientist applies more than one algorithms on the model to check which one scores higher and gives much accuracy. In the practical part of this thesis, we are going to focus on the following three algorithms.

These algorithms serve as the litmus test on the viability of a model. In this project, since we seek to find the best fit model, we will not test just a single model but rather impose all the available algorithms as discussed below and find the best model with its parameters. This approach is accompanied by parameter tuning to realize the algorithm viable for the project with its corresponding parameters for optimal model performance. Therefore, we employ hyper parameter tuning in the training of this model to achieve the best parameters and order them with their score index. We then select the algorithm with the best score from the Score index and note the model's tuning parameter.

3.1.3.1 Linear regression

This is an algorithm in the supervised machine learning algorithms family where the value predicted as output is continuous and exhibits a constant slope. It is used for predictive analysis, such as a house's cost, total sales, and calls. The primary goal of this algorithm is usually to answer the following questions:

(1) Does a set of predictor variables (independent) predict the outcome variable satisfactorily? And

(2) Which among the predictor variables have great significance in predicting the outcome variable, and to what extent do they impact the outcome variable?

In linear regression, we establish a relationship between the dependent and independent variables using the best fit line, statistically referred to as a regression line. The following general equation represents the line:

$$\mathbf{y = MX + b}$$

Equation 1: Linear equation

Source*:* (Pierce, 2018)

where, Y is the value under prediction, ***x*** is the independent (predictor) variable***, m*** is the gradient, and b is the y-intercept.

There are two main types of linear regression simple regression and multivariable regression. Simple regression has only one independent variable (x) and one dependent variable (y), but multivariable regression has one dependent variable (y) and more than one independent variable (x). If we take the example of house prices, then price prediction based on only square feet is a simple regression. Price prediction based on square feet, bedrooms, bathrooms, area, balcony, etc. is multivariable linear regression.

Let us visualize linear regression by the following graph:

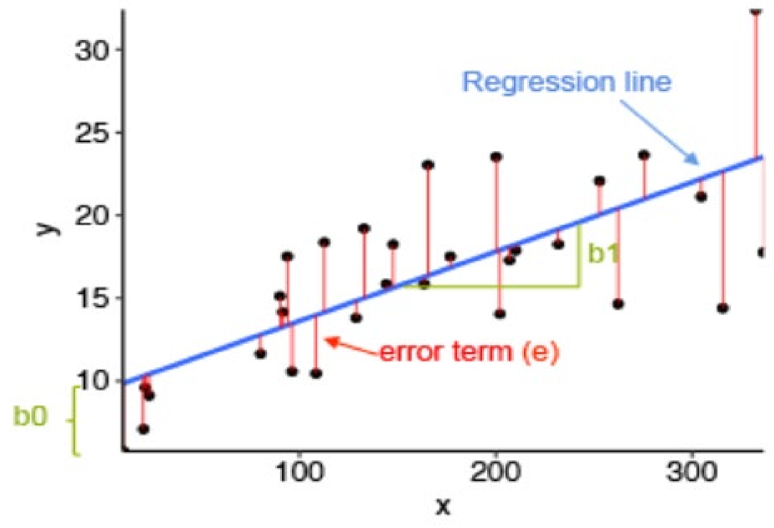


Figure 2: Linear Regression Graph

Source: (kassambara, 2018)

In the figure above, we have two variables, x, and y, which has multiple data points defining the relation between both variables. The blue line is called the regression line passing through all the data points. Now the question is, why it can be only one line? There might also multiple lines passing through data points. Well, that is the motto of linear regression to find the best fit line for our model. The principle behind the regression line is to minimize error, which can be done by the sum of square errors equation:

$$\mathbf{SSE} = \sum_{i=1}^{n}[\Delta]^2$$

$$\Delta = [y_i - (\beta_0 + \beta_1 x_i)]$$

Equation 2: Sum of square error

Source: (Weisberg, 2005)

where, Δ is distance between data points and blue line (regression line) which is defined as red line in figure above.

The equation does the sum of squared individual error of all data points from 1 to N. from the minimization of this equation; we can find only a blue line that can best fit for the model.

3.1.3.2 Lasso Regression

Lasso stands for "Least absolute shrinkage and selection operator." It is a supervised machine learning that employs the concept of shrinkage. Shrinkage is where the data values are compressed towards a central point. The procedure encourages simple and spars model which have fewer parameters. This algorithm is perfectly suitable for the model having high-level multilinearity or when it comes to automation of certain parts of model selection, variable selection, or parameter elimination (Glen, 2015).

The coefficients are established upon minimization of this equation:

$$\sum_{i=1}^{n}[y_i - (\beta_0 + \beta_1 x_i)]^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

Equation 3: Lasso regression

Source*:* (The group lasso for logistic regression, 2008)

Where λ is a constant positive amount of shrinkage, which regulates the imposed penalty's strength and validity.

β_j = slope of regression line.

When λ=0, it is at a steady-state, and the estimates are equal to the one found in the standard linear regression.

As λ increases, more and more coefficients are set to zero and eliminated. So, when l=∞, all coefficients are eliminated.

As λ increases, bias and variance increase.

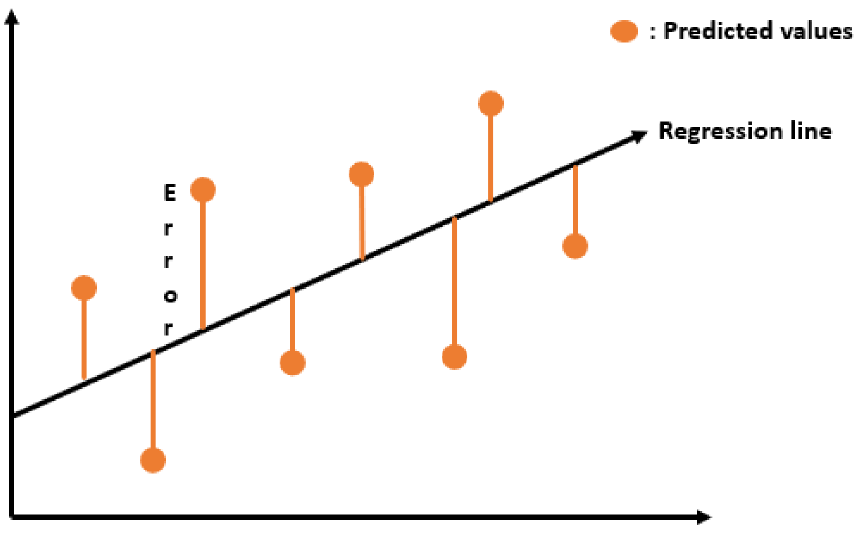


Figure 3: Lasso regression

Source: Author

The technique uses a penalized least squares as a basis for modeling and parameter sub-selection approach. Lasso regression helps us make feature selection by choosing the slop's magnitude value, which means wherever the slop value is close to zero, it will remove those features as they are not much important for our prediction. It will keep only essential features for prediction. It is useful in fitting high-dimensional data exhibiting high correlations in the predictors. It can, therefore, be thought of as a Hybrid variable selection procedure. (Bayesian and LASSO Regressions for Comparative Permeability Modeling of Sandstone Reservoirs, 2018).

3.1.3.3 Decision Tee

The decision tree is usually a supervised machine learning algorithm ideal for task regression and its classification. The decision tree's primary goal is to predict the value or class of specific variables depending on generated decision rules by algorithm from the training dataset. As per the name, this algorithm represents structure like a tree in which we have a variable as a root node to test on. Branches from root nodes are represented as a result of the test, and those branches have leaf nodes representing class or label. There are two decision tree types - **categorical variable decision tree** (targeted variable is categorical) and **continuous variable decision tree** (targeted variable is continuous).

Splitting:

Decision tree decides by splitting their root nodes into sub-nodes. This splitting process continuously goes on until it is left with only homogenous nodes. There are multiple methods of splitting, which depend on the type of targeted variable. For categorical variables, we can use Gini impurity, information gain, and Chi-square. For continuous variables, we can use a reduction invariance. As we will focus on the continuous variable in the practical part, let us understand the reduction invariance.

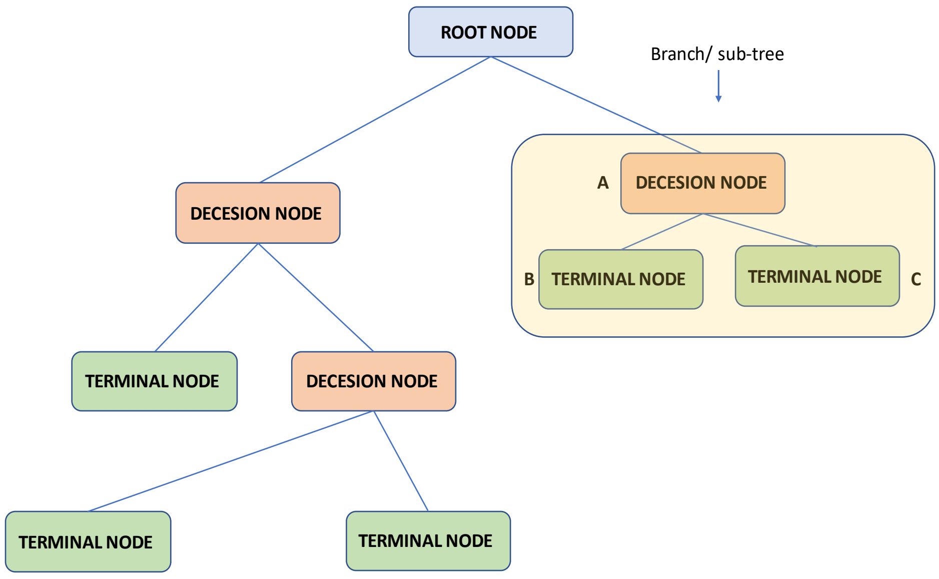


Figure 4: splitting in decision tree

Source: Author

Reduction invariance:

This splitting method can be used for continuous variables only. It uses the generic statistical formula of standard deviation and variance to get the best split.

$$\text{Variance} = \frac{\sum(x-\mu)^2}{n}$$

Equation 4: variance

Source: (Suzuki, 2019)

where μ is the mean of the values, x is the actual value, and n is the number of values.

Variance is used to calculate the purity of a node. As much variance is low as purer a node will be if the node will be entirely pure, then variance value will be 0.

Steps to split a decision tree using reduction invariance (Sharma, 2020):

1. Calculate the variance of each node for each split.
2. Calculate the weighted average variance of each split of child nodes.
3. Pick the split with the lowest variance.
4. Repeat until uniform and homogeneous nodes are realized.

Pruning:

Overfitting is the most usual and significant problem in the decision tree. It is a situation when the model gives 100% accuracy for the training data set but for the testing data set. It might have a more considerable variance between the actual and predicted value. The reason is there is no limit for growth in the decision tree. Sometimes in the worst case, it gives 100% accuracy for training data set by making one leaf for each observation. This situation will affect accuracy while predicting the actual testing data set. Pruning is one of the well-known ways to avoid overfitting. Pruning methods remove the decision nodes from the leaf nodes without affecting the model's overall accuracy. This method uses statistical measures to eliminate the least reliable branches, which leads to faster classification and improvement in the prediction of outputs from the independent test data (Evaluation of Decision Tree Pruning Algorithms for Complexity and Classification Accuracy, 2010). We can easily understand difference between an unpruned tree and a pruned tree with simple example of bank loan approval from following figure.

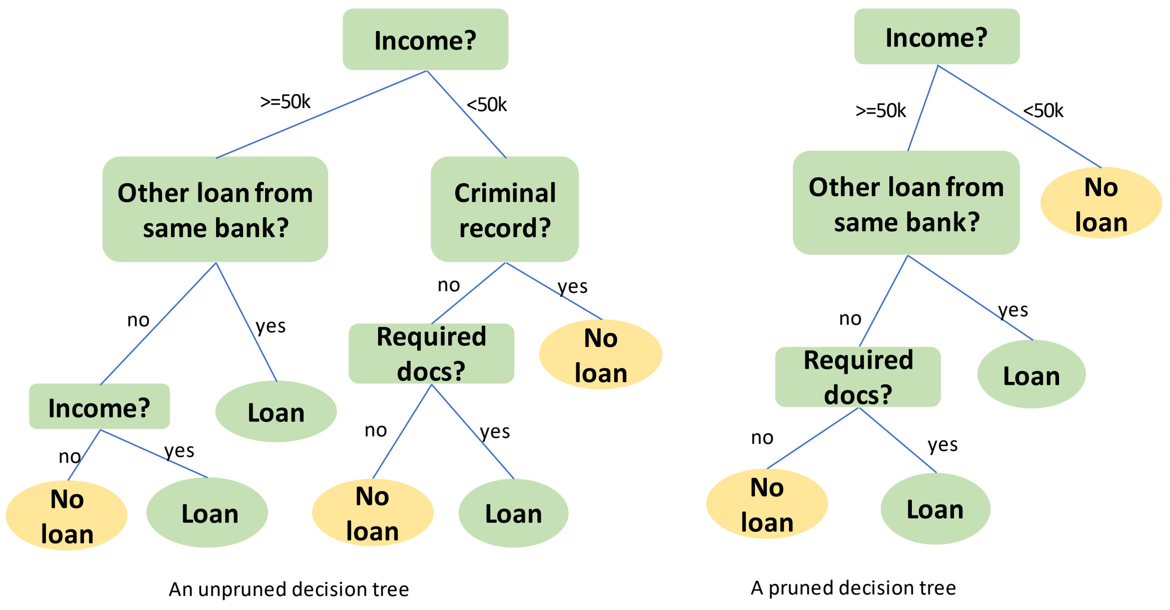


Figure 5: pruned decision tree

Source: Author

3.2 Cross-Validation in Model Selection

Generally, in machine learning, we break down our data set into training and testing data for model creation. Even in the same algorithm, the model will give us different accuracy for the different test sets. Therefore, it is best practice to apply cross-validation to the machine learning model for better accuracy. Cross-validation is a popular statistical technique for algorithm selection. The main goal of the cross-validation is to assess how the model will perform with different data set. The idea behind cross-validation is to split data once or several times to estimate each algorithm's skill or model. The training set is used to train each algorithm, and the validation set is used to estimate the risk. In the end, we select the algorithm with the smallest estimated risk (A survey of cross-validation procedures, 2010).

There are plenty of cross-validation methods such as k-fold cross-validation, holdout method, leave-p-out cross-validation, and leave-one-out cross-validation. We will focus on k-fold cross-validation as it is prevalent, suitable for extensive data set, and we are going to use in our practical part.

Cross-validation Serves to verify that the algorithm selected is robust over random tests with the test data set. The expected score is supposed not to wander far away from the observed values after the hyper-parameter tuning.

3.2.1 K-fold cross-validation

This technique is a widely adopted method for model selection. As per the name, this technique randomly divides data set into K folds approximately in equal size. This K part of the data will be used for the testing dataset, and the rest (k-1) part will be used as a training dataset. The model will be trained and tested for K number of times, and at the end of the process, we will get a K number of scores. The average number of this score will be considered as the average accuracy. We can also say the maximum accuracy of this model, the highest score we received, and the model's minimum accuracy is the lowest score in this process.

In K-fold cross-validation, K is a chosen number by us, which represents the number of folds. The choice of a number depends on our data size and system computation power. The number can be anything ideally between 5 and 10. We must choose numbers carefully because poor choice might lead our model to high variance and high bias. K< 5 might cause issues like that (A Comparative Study of Ordinary Cross-Validation, v-Fold Cross-Validation and the Repeated Learning-Testing Methods, 1989).

Advantages of K-fold CV

- As K value increases, the estimated variance and bias reduces.
- For the K value, the repetition of the process is limited, so less computation time.
- Each data part gets to be trained and tasted precisely.

Disadvantages of K-fold CV

- It takes a considerable amount of time to evolve as the algorithm must rerun from scratch K times.

3.3 Hyper Parameter Tuning in Machine Learning

In machine learning, hyper parameter tuning is an important task to get optimal values of a model's parameter, which gives the maximum accuracy for a particular model. Different datasets have different hyper parameter settings, so it must be tuned for each dataset. A hyper parameter is the machine learning element that automatically cannot be learned by model, but it can be done by a meta-process called - “hyper parameter tuning." Manually, it is difficult to keep track of the hyper parameter and frequently fit into training datasets; simultaneously; it is time-consuming. A grid search CV can solve this problem.

3.3.1 Grid search CV

Grid search CV is one of the well-known methods for hyper parameter tuning. It is a function of the Scikit-learn library, which helps to loop through predefined hyper parameters and fit our model in the training dataset. The method will then list each parameter values' score, and we can select the best parameter from it. According to (Chih-Wei Hsu, 2003), it is highly recommended to use a grid search CV and cross-validation to archive the best parameter values. The grid search CV structure is like a dictionary (keys= parameter names, values= various possibilities for our combinations), and then it passed to our estimator object.

Grid Search CV serves the purpose well in our model as from the python data dictionary. This is the easiest way to validate and classify the parameters used in modeling the final price prediction model.

3.4 Python for Data Science

There is plenty of programming language used in data science projects like Python, Java, R, SAS, SQL, etc. Python is open source, interpreted, and dynamic object-oriented, publicly available in 1991 (Hsu, 2018). It is widely used and suitable for data science tools and applications. According to a “stake overflow survey” in 2019, python is fast-growing and second most loved programming language.

Python holds a unique attribute, and when it comes to performing analytical and quantitative tasks, it is very easy than other programming languages. According to engineers of academia and industry, python APIs are available for deep learning frameworks. Scientific packages have constructed python as incredibly productive and versatile (Bhatia, 2012).

Hence, now we know what importance python has in the data science field, let us focus on some elegant features of python.

- Python supports various platforms such as Windows, Linux, Mac, etc.
- It makes the program easy to read and write. It is also easy to perform various machine learning algorithms and complex scientific calculations, thanks to elegant and simple syntax.
- Python has the ultimate collection of libraries to perform various tasks like data manipulation, data analysis, and data visualization.
- Python is an expressive language that makes possible applications to offer a programmable interface (Eppler, 2015).
- In python, it is simple to an extension of code by appending new modules implemented in other compiled languages like C or C++.

Machine learning scientists prefer python as well in terms of application areas. When it comes to app development for NLP and machine analysis, developers switch to python due to the huge collection of libraries python provides, which helps solve complex business problems efficiently and construct a robust system data application.

3.5 Essential Python Libraries for Data Science

Python libraries are a reusable bunch of functions and methods which we can include in our program to perform several actions without writing code. Python has improved libraries' support in recent years and became the best alternative for data manipulation techniques. It is among the favorites for full-stack developers, which is also highly recommended for general-purpose programming (McKinney, 2012).

In a data science project, we need to go through all the stages like data cleaning, data visualization, model building, etc. Python has plenty of popular libraries for these tasks. Let us focus on some of them, which we are going to use in our practical part.

3.5.1 NumPy:

NumPy (Numeric Python) is one of the most potent and open-source python libraries primarily used for numeric analysis. NumPy deals with numerical data and provides algorithms, data structures, and other utilities to perform scientific calculations and data storage. It is highly recommended to fast operation on arrays, sorting, selecting, mathematical functions, statistical operation, linear algebra, random simulation, etc. It was created by Jim Hugunin which was modified by Travis Oliphant in 2005 to incorporating features of competing NumPy-array into numeric (Oliphant, 2015).

NumPy basics

- Create NumPy arrays and array attributes.
- Array indexing and slicing.
- Reshaping and concatenation.

NumPy arithmetic and statistics basics

- Computations and aggregations
- Comparison and Boolean masks

The NumPy package has a significant object called “ndarray” (n-dimensional array). It is homogeneous and statistical data types and performs many operations in a compiled language (Leo (Liang-Huan) Chin, 2016). Now the question is, why would we use NumPy array when we can just use a python list? The list is very flexible and versatile, and excellent in python, but there are few significant benefits of using NumPy arrays over a python list.

Saves coding time

- No for loops: many vector and matrix operation save coding time

. We do not need to iterate through an array to apply a mathematical operation to each element of that array. We can do it with a single line of code.

Example:

Using python list, we need to use for loop to iterate through that list before you can multiply *= 6 operation to each element:

```
for i in range (len(my_list)):
     my list[i] *= 6
```

Using NumPy array, we can apply that element directly to the entire array with a single code line. NumPy takes care of the rest of the operation behind the scenes, so we do not have to worry about it:

```
my array *= 6
```

Faster execution

- Uses single data type for each element (all must be the same data type) in array to avoid type checking at runtime.
- Uses contiguous blocks of memory.

Uses less memory

- No pointers, so type and item sizes are the same for each column.
- In python list, there is an array with pointers to python object (4B+ per pointer and 16+ for a numerical object).
- Compact data types like unit 8 and float 16.
- Which depends on our task and precision of data.

3.5.2 Pandas

Pandas name is shorthand for "panel data"- a term for data sets with multidimensional structure. Another important machine learning library provides functions and rich data structure to make our data analysis task more manageable, fast, and expressive. Pandas have various methods for combining data, time-series functionality, grouping, and filtering. It also provides indexing functionality, which simplifies reshaping easier, data slicing, performing aggregation, and selecting a subset from a dataset (McKinney, 2012).

Pandas built on top of NumPy. That means pandas require NumPy. Pandas do not require other libraries like Matplotlib and SciPy, but it can be handy if combined with the latter. It is an excellent tool for data wrangling due to its robust design coupled with quick and easy data manipulation features. It has two handy data structures known as **"pandas series"** and **"pandas data frame**." They are core components of pandas that allow us to reshape, merge, split, train, and aggregate data.

Pandas Series

It is a one-dimensional labelled array containing data types. These data types span from strings, doubles, integers, objects, and floats python objects. The axis labels represent the index. In short, it is just a column in memory that is either independent or belongs to a pandas data frame. A unique label is not necessary, but it must be a hashable type. The python object integrates label-based indexing as well. It provisions a host of methods for performing index-operations.

Pandas Data frame

This is a two-dimensional, tabular data structure with more details regarding axis. The concept of a data frame is borrowed from the idea of spreadsheets. It is logically corresponding to a sheet of Excel that includes both rows and columns. The Data frame object contains an ordered collection of columns like a spreadsheet or an excel sheet. Each column holds a unique data type, but different columns may have different types (Bernd Klein, 2011).

Data frame operation in pandas

- Read, view, and extract information;
- Grouping and sorting;
- Deals with duplicate and missing values;
- Selection, filtering, and slicing;
- Pivot table and functions;

The pandas library has been under development way since the python was made public. It is established to narrow and possibly close the gap in the available data analysis tools between python, and the conventional domain-specific statistical computing platforms, software, and database languages (McKinney, january 2011). There are currently fewer pandas library releases, including plenty of new features, enhancements, bug fixing, and API changes. Data analysis among everything else takes the highlights when it comes to the use of pandas. Pandas ensure high functionality and superb flexibility while combined with other libraries and tools.

3.5.3 Matplotlib

It is generally conceptualized as a plotting library for data visualization. It is a python package for 2D plotting that generates a production-graph. Any organization must visualize data and descriptive analysis; matplotlib provides very effective methods for these tasks. It supports both interactive plotting and non-interactive plotting to save the graphics into several formats like .pdf, .png, .jpeg, among others. It can also employ multi-window toolkits (GTK+, wxWidgets) and provide a conglomerate of plot types like line graphs, pie-charts, and bar charts histograms, and other professional-grade figures. Besides, it boasts high customization levels, flexibility, and convenience in use (Tosi, November 2009).

Features of Matplotlib

PyLab interface :

It allows users to create plots using code just like the MathWorks Package MATLABTM figure generates its code.

Matplotlib API:

Acts as the abstract interface over which plots are rendered. It is responsible for tuning the parameters and ensure what is given in the code is translated properly in the intended interface.

Backends:

These play the primary role of interpreting the graphics to other devices connected to the computer or intended for display services.

Most professionals employ Matplotlib in generation postscript files for printing or publishing automatically. Some find the convenience of deploying the graphics on web applications that can dynamically generate specific nature files. Matplotlib library can also be called interactively from Tkinter's python shell on the Windows platform (John Huter, 27 may, 2007).

3.5.4 Scikit-Learn

This is a python library that makes various algorithms and functions that are used in machine learning available. David Cournapeau initially developed it as a google summer of code project in 2007. It is considered one of the best libraries for working with complex data.

Scikit learns built on NumPy, SciPy, and matplotlib. It contains several algorithms for data mining and machine learning tasks like (Hackeling, 2017):

- Dimensionality reduction;
- Data reduction methods (e.g. principal component analysis, feature selection);
- Regression analysis (e.g. linear, logistic, and ridge);
- Classification and clustering models (e.g. random forest, support vector machine, K-means);
- Model tuning and selection (e.g. grid-search, cross-validation);
- It also provides modules for pre-processing data, extracting features, optimizing hyper parameters, and evaluating models to solve real-world problems;

4. Practical part

4.1 project tool

Python is straightforward to learn as it is a described programming language. It has extensive documentation and vibrant online support groups where support is easily found. It is also easy to use with minimal coding and realizing maximum computational power in exploring data, using graphics, and manipulating almost all variables on the go. I have employed python from the interface of Jupyter Notebooks, which is a very convenient tool for python beginners and experts. This brings the convenience of installing python and pandas at a go and accessing them through library import.

4.2 Dataset

First, the datasets are downloaded directly from Google environment. There can be multiple ways to access data but, in this case, I assumed that we are getting data from the organization for which we prepare the size prediction model. I use the data source below:

Dataset Name: Bengaluru-house-price-data

Type: Comma Separated Values (CSV)

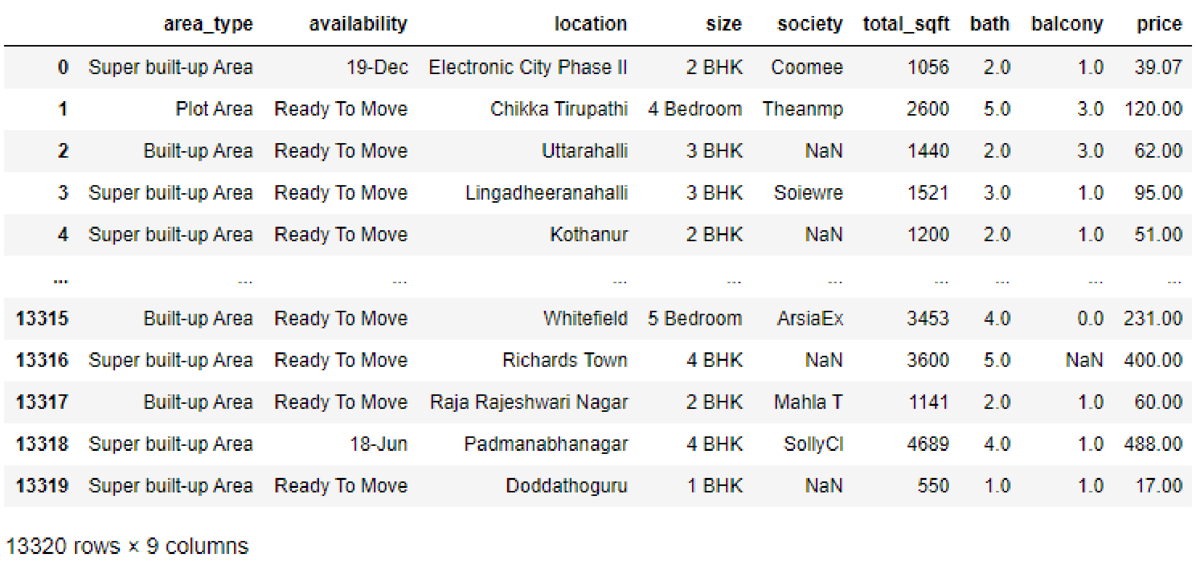
Location: https://www.kaggle.com/amitabhajoy/bengaluru-house-price-data (Chakraborty, 2017)

This dataset is about city 'Bengaluru, India'. So, we assume that we are building size prediction model for real estate organization in Bengaluru. After downloading the dataset, we will impot it into jupyter notebook. First, we will import required libraries in jupyter notebook by executing these commands.

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
%matplotlib inline
import matplotlib
matplotlib.rcParams["figure.figsize"]=(20,10)
```

These libraries will help us for our initial tasks for data cleaning and data visualizing. As we discussed in our literature review NumPy will help us for all numerical operation, Pandas will help us for all data frame operation and Matplotlib will help us for plotting visual graph. Each module assigned with their short form such as np for NumPy, pd for Pandas, plt for pyplot of Matplotlib. So, we don’t have to write full form when we execute commands with libraries. Now, we will read our dataset by executing following command.

```
pd.read_csv("Bengaluru_House_Data.csv")
```



	area_type	availability	location	size	society	total_sqft	bath	balcony	price
0	Super built-up Area	19-Dec	Electronic City Phase II	2 BHK	Coomee	1056	2.0	1.0	39.07
1	Plot Area	Ready To Move	Chikka Tirupathi	4 Bedroom	Theanmp	2600	5.0	3.0	120.00
2	Built-up Area	Ready To Move	Uttarahalli	3 BHK	NaN	1440	2.0	3.0	62.00
3	Super built-up Area	Ready To Move	Lingadheeranahalli	3 BHK	Soiewre	1521	3.0	1.0	95.00
4	Super built-up Area	Ready To Move	Kothanur	2 BHK	NaN	1200	2.0	1.0	51.00
...	...	...	...	...	...	...	...	...	...
13315	Built-up Area	Ready To Move	Whitefield	5 Bedroom	ArsiaEx	3453	4.0	0.0	231.00
13316	Super built-up Area	Ready To Move	Richards Town	4 BHK	NaN	3600	5.0	NaN	400.00
13317	Built-up Area	Ready To Move	Raja Rajeshwari Nagar	2 BHK	Mahla T	1141	2.0	1.0	60.00
13318	Super built-up Area	18-Jun	Padmanabhanagar	4 BHK	SollyCl	4689	4.0	1.0	488.00
13319	Super built-up Area	Ready To Move	Doddathoguru	1 BHK	NaN	550	1.0	1.0	17.00

13320 rows × 9 columns

Figure 6: Row dataset

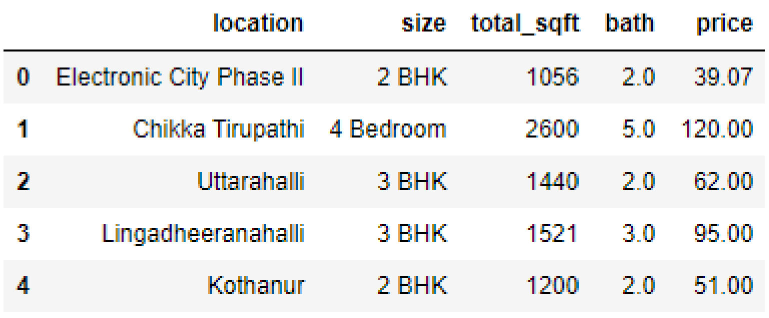
Source: Author

In our dataset we have different types of columns, from them total_sqft is dependent as our end goal is to predict home size. But other columns are independent, their value can influence total_sqft of house. Now, we will assign whole dataset as main data frame executing following command:

```
Main_Data_Frame =pd.read_csv("Bengaluru_House_Data.csv")
mdf0=Main_Data_Frame
```

Step by step we will make improvements in our row dataset and we will assign updated dataset a new name, which is also known as pipeline in data science term. For example, mdf0 is our row dataset, unnecessary columns will be removed from it and assign as mdf1. The reason for removing unnecessary columns is they doesn’t impact much on our dependent variable and by removing these columns, we can reduce size and features of our dataset for ease of further operations. The code for dropping those columns is:

```
mdf1=mdf0.drop(['area_type','society','balcony','availability'],axis
= 'columns')
mdf1.head()
```



	location	size	total_sqft	bath	price
0	Electronic City Phase II	2 BHK	1056	2.0	39.07
1	Chikka Tirupathi	4 Bedroom	2600	5.0	120.00
2	Uttarahalli	3 BHK	1440	2.0	62.00
3	Lingadheeranahalli	3 BHK	1521	3.0	95.00
4	Kothanur	2 BHK	1200	2.0	51.00

Figure 7: Unnecessary columns dropped from dataset

Source: Author

Now, we left with only columns which are very important for prediction house size and these will be the columns which customer will use as input value to check house sizes in different locations. As we can see BHK unite in size columns, it stands for Bedroom-Hall-Kitchen in Indian term. Price is in lakh unit and in Indian currency INR (1,00,000 INR is called 1 lakh). We will proceed to data cleaning task now.

4.3 data cleaning

Data cleaning is part where data scientists spend majority of their time because usually row data are messy and it became critical to handle missing and unexpected values sometimes. That's another reason why we dropped certain columns to keep it short and beginner friendly for this thesis. When it comes to data cleaning, the very first thing we should focus on is null values. To check null values in our data set I will execute following command:

```
mdf1.isnull().sum()
```

output:

```
location       1
size          16
total_sqft     0
bath          73
```

```
price          0
dtype: int64
```

As we can see, we have only 16 null values for size and 73 null values for bath, which are negligible as compared to total rows 13320. We can use standard deviation or average to fill those values randomly, but it might be not accurate for our model. When null values are negligible, we should drop them instead of filling with average. So, in our case I will simply drop those rows by executing following command:

```
mdf2=mdf1.dropna()
mdf2.isnull().sum()
```

output:

```
location      0
size          0
total_sqft    0
bath          0
price         0
dtype: int64
```

We dropped null values from our dataset and assigned name as mdf2 to the new dataset. Now, let us focus on size feature. As it contains values mixture of numbers and strings, we need to deal with it. To see unique values of size, I will execute unique command:

```
mdf2['size'].unique()
```

output:

```
array(['2 BHK', '4 Bedroom', '3 BHK', '4 BHK', '6 Bedroom', '3 Bedroo
m',
       '1 BHK', '1 RK', '1 Bedroom', '8 Bedroom', '2 Bedroom',
       '7 Bedroom', '5 BHK', '7 BHK', '6 BHK', '5 Bedroom', '11 BHK',
       '9 BHK', '9 Bedroom', '27 BHK', '10 Bedroom', '11 Bedroom',
       '10 BHK', '19 BHK', '16 BHK', '43 Bedroom', '14 BHK', '8 BHK',
       '12 Bedroom', '13 BHK', '18 Bedroom'], dtype=object)
```

As we can see from output, some of size values are measured in BHK unite and some of them are as bedroom. Technically both values are same as they indicates to number of bedrooms only. We cannot do numerical operation on string. Thus, we will create new feature BHK to keep only number of bedrooms. In order to create this feature, we need to split size values in two parts in numbers and strings. Then we will use separated numbers in our BHK column. In Python lambda is very popular anonymous function which can takes pythonic expression as parameter. Lambda function is used as an application and whenever it uses, it generally means we want to apply this expression to constructed type. Lambda is widely being used in mapping data and for loops as well (Boudreau, 2020). In our case, I will use following function to split our value:

```
mdf2['bhk']= mdf2['size'].apply(lambda x: int(x.split(' ')[0]))
```

In this function, we created new column bhk in data frame mdf2 by applying lambda function on size column. Simply, this lambda function tokenize whole value of size and split them with space (' ') from which 1st part will be converted into int from string and will be assigned in bhk column. We can check bhk columns unique values by executing following function:

```
mdf2['bhk'].unique()
```

output:

```
array([ 2,  4,  3,  6,  1,  8,  7,  5, 11,  9, 27, 10, 19, 16, 43, 14, 12,
        13, 18], dtype=int64)
```

Now, it looks exactly how we wanted for our further operations and we can use bhk column instead of size. Let us focus on square feet feature. By executing following command, we can see what unique values total_sqft column contains.

```
mdf2.total_sqft.unique()
```

output:

```
array(['1056', '2600', '1440', ..., '1133 - 1384', '774', '4689'],
      dtype=object)
```

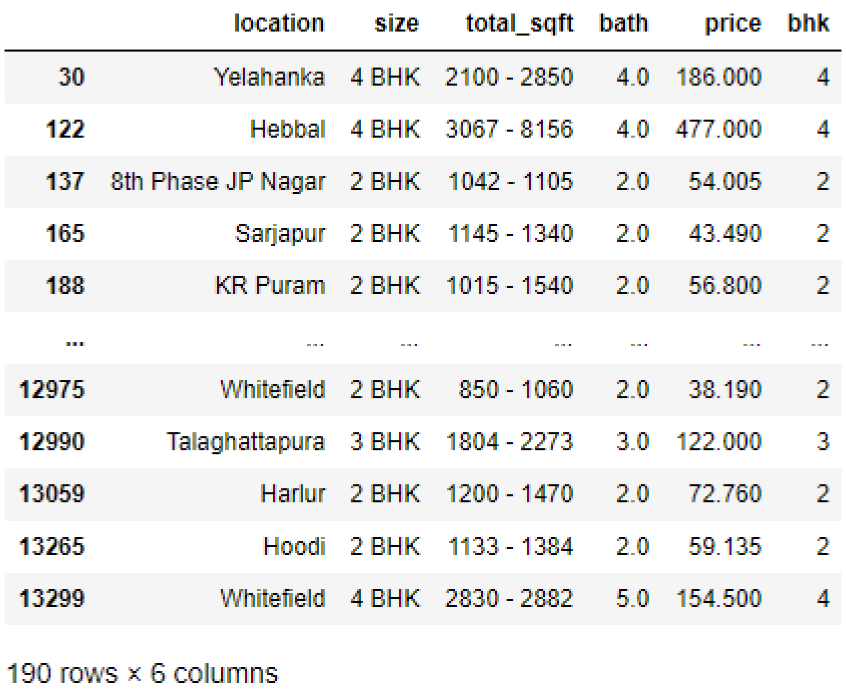
total_sqft feature contains ranges of square feet, which is not appropriate for numerical operation for machine learning. So, we can simply tackle this problem by taking average of range and change it into single value. Following function will be applied to total_sqft column to detect all values which contains range values or any other type of values expect single number.

```
def float_sqft(x):
    try:
        float(x)
    except:
        return False
    return True
```

This function take values as input and check if the value is float or not. If not then it will return as false, otherwise it will remain true. I will apply this function to total_sqft columns by executing following command:

```
mdf2[~mdf2['total_sqft'].apply(float_sqft)]
```

Here, I used negate operator for function to return only values with range values. Output for this operation is represented in following figure.



	location	size	total_sqft	bath	price	bhk
30	Yelahanka	4 BHK	2100 - 2850	4.0	186.000	4
122	Hebbal	4 BHK	3067 - 8156	4.0	477.000	4
137	8th Phase JP Nagar	2 BHK	1042 - 1105	2.0	54.005	2
165	Sarjapur	2 BHK	1145 - 1340	2.0	43.490	2
188	KR Puram	2 BHK	1015 - 1540	2.0	56.800	2
...	...	...	...	...	...	...
12975	Whitefield	2 BHK	850 - 1060	2.0	38.190	2
12990	Talaghattapura	3 BHK	1804 - 2273	3.0	122.000	3
13059	Harlur	2 BHK	1200 - 1470	2.0	72.760	2
13265	Hoodi	2 BHK	1133 - 1384	2.0	59.135	2
13299	Whitefield	4 BHK	2830 - 2882	5.0	154.500	4

190 rows × 6 columns

Figure 8: detected range values in 'total_sqft' column

Source: Author

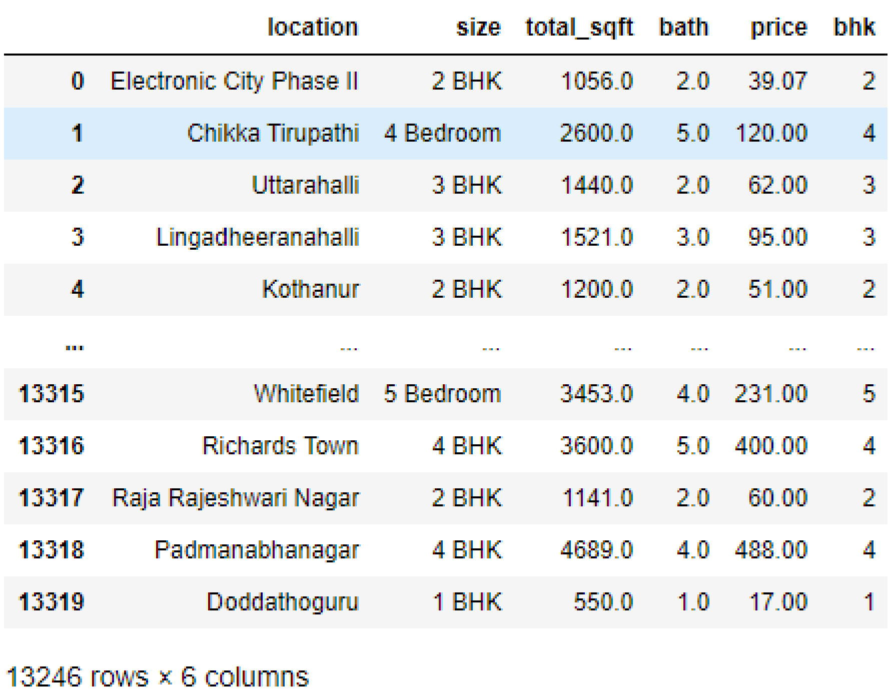
We have 190 rows which have range or other kind of values expect single number. To handle this problem, we will take average where values are in range and other type of values we will drop from dataset. I will create following function to convert range values into average.

```
def sqft_to_numeric_value(x):
    tokens = x.split('-')
    if len(tokens)== 2:
        return (float(tokens[0])+float(tokens[1]))/2
    try:
        return float(x)
    except:
        return None
```

This function works by tokenizing values. It will take values as input and if values are in range, then it splits values in two tokens from (-) operator and return average of both. Else it return to original value in case of single numbers. Now I will apply this function to total_sqft column by executing following function:

```
mdf3=mdf2.copy()
mdf3['total_sqft'] = mdf3['total_sqft'].apply(sqft_to_numeric_value)
mdf3
```

In this command, I created new dataframe mdf3 by copying mfd2 and applying (sqft_to_numeric_value) function. The output of mdf3 represented in following figure:



	location	size	total_sqft	bath	price	bhk
0	Electronic City Phase II	2 BHK	1056.0	2.0	39.07	2
1	Chikka Tirupathi	4 Bedroom	2600.0	5.0	120.00	4
2	Uttarahalli	3 BHK	1440.0	2.0	62.00	3
3	Lingadheeranahalli	3 BHK	1521.0	3.0	95.00	3
4	Kothanur	2 BHK	1200.0	2.0	51.00	2
...	...	...	...	...	...	...
13315	Whitefield	5 Bedroom	3453.0	4.0	231.00	5
13316	Richards Town	4 BHK	3600.0	5.0	400.00	4
13317	Raja Rajeshwari Nagar	2 BHK	1141.0	2.0	60.00	2
13318	Padmanabhanagar	4 BHK	4689.0	4.0	488.00	4
13319	Doddathoguru	1 BHK	550.0	1.0	17.00	1

13246 rows × 6 columns

Figure 9: Converted 'total_sqft' into single values

Source: Author

As we can see, all values in total_sqft seems like single. But to cross check I will examine following command on row number 30 because it had range value, as we can see in figure 8.

```
mdf3.loc[30]
```

output:

```
location       Yelahanka

size               4 BHK
total_sqft          2475
bath                   4
price                186
bhk                    4
Name: 30, dtype: object
```

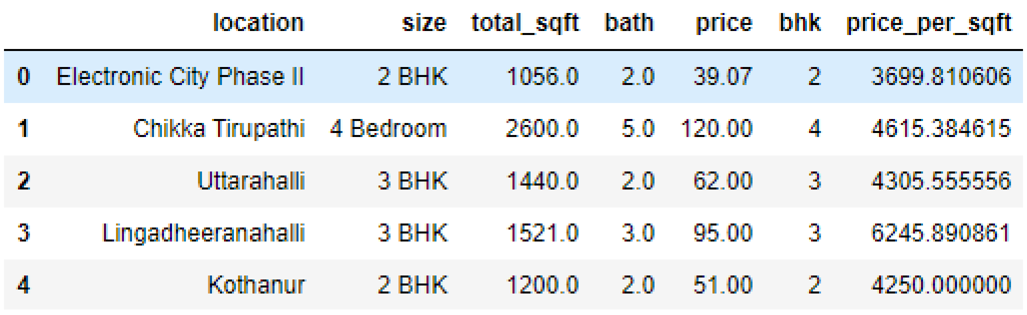
Now, the value of total_sqft is 2475, which is average of 2100-2850. As our dataset is cleaned from null values and ununiformed values, we can proceed to next step of data science life cycle, which is feature engineering.

4.4 Feature engineering

Feature engineering part is focused on creating new variables or features or modify them in a way, which can help us for our next task of outliner detection and removal. We have price of house feature in our dataset but it does not make much sense when we want to examine how price differs from location to location according to area. So, price per square feet is a common feature for any real estate organization. Considering this, I will create new column 'price_per_sqft' by executing following command:

```
mdf4 = mdf3.copy()
mdf4['price_per_sqft'] = mdf4['price']*100000/mdf4['total_sqft']
mdf4.head()
```

This command will create new dataframe mdf4 which is copy of mdf3. But in mdf4 there is new column called 'price_per_sqft', which is division of price and total_sqft columns. Here I multiplied price column with 100000 (1 lakh) which is unit of price in our dataset, so our price per square feet will be normalized unit. Result for this command is represented in following figure:



```
                  location       size  total_sqft  bath   price  bhk  price_per_sqft
0   Electronic City Phase II    2 BHK      1056.0   2.0   39.07    2     3699.810606
1           Chikka Tirupathi  4 Bedroom    2600.0   5.0  120.00    4     4615.384615
2                Uttarahalli    3 BHK      1440.0   2.0   62.00    3     4305.555556
3         Lingadheeranahalli    3 BHK      1521.0   3.0   95.00    3     6245.890861
4                   Kothanur    2 BHK      1200.0   2.0   51.00    2     4250.000000
```

Figure 10: Added new feature - 'price_per_sqft'

Source: Author

Our second feature is location. Let us focus on structure of location column.

```
len(mdf4.location.unique())
```

```
output: 1304
```

we have 1304 types of different locations in our dataset, which is way more for model building process. Because in our future task we will convert all location as columns, that means we will have 1304 more columns. This is known as ‘curse of dimensionality’. According to (Bellman, 2010), curse of dimensionality is the problem caused by the exponential increase in volume associated with adding extra dimensions to Euclidean space.

To tackle this problem, one of the best solution is to come up with ‘other’ category. Which means there will be plenty of locations, which will have only 1 or 2 houses. So basic idea is to move all locations in other category, which have less than 10 homes. For this task, first I will execute following function to know which how many homes are available per location:

```
mdf4.location=mdf4.location.apply(lambda x: x.strip())

location_stats =mdf4.groupby('location')['location'].agg('count').sor
t_values(ascending=False)
location_stats
```

first line of code is lambda function to strip any location, to remove extra spaces from beginning or end. In second line of code, I created variable called location states which will give statistics on location by grouping by locations. The result for this code is below:

```
location
Whitefield           535
Sarjapur  Road       392
Electronic City      304
Kanakpura Road       266
Thanisandra          236
                    ...
LIC Colony             1
Kuvempu Layout         1
Kumbhena Agrahara      1
Kudlu Village,         1
1 Annasandrapalya      1
Name: location, Length: 1293, dtype: int64
```

Now, I will create variable called ‘location_states_less_than_10’ which will contain all location having less than 10 data points or homes by executing following code:

```
location_stats_less_than_10 = location_stats[location_stats <= 10]
```

```
location_stats_less_than_10
```
output:

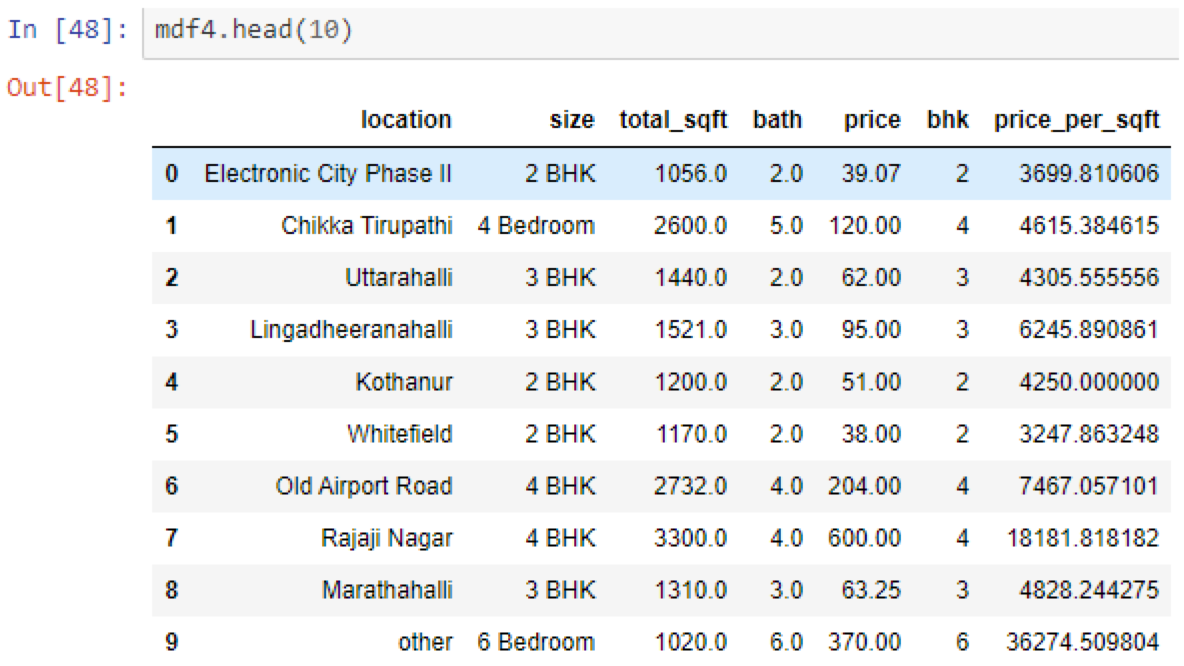
```
location
BTM 1st Stage          10
Basapura               10
Sector 1 HSR Layout    10
Naganathapura          10
Kalkere                10
                       ..
LIC Colony              1
Kuvempu Layout          1
Kumbhena Agrahara       1
Kudlu Village,          1
1 Annasandrapalya       1
Name: location, Length: 1052, dtype: int64
```

As we can see there are 1052 location which have less than 10 homes and now all these location will be represented in other category. To perform this task, I will execute following lambda function:

```
mdf4.location = mdf4.location.apply(lambda x: 'other' if x in locatio
n_stats_less_than_10 else x)
len(mdf4.location.unique())
```

```
output: 242
```

So, now we have only 242 unique location rows which are pretty decent for our future operations. Following figure represents dataframe with latest features.



```
In [48]: mdf4.head(10)
Out[48]:
```

	location	size	total_sqft	bath	price	bhk	price_per_sqft
0	Electronic City Phase II	2 BHK	1056.0	2.0	39.07	2	3699.810606
1	Chikka Tirupathi	4 Bedroom	2600.0	5.0	120.00	4	4615.384615
2	Uttarahalli	3 BHK	1440.0	2.0	62.00	3	4305.555556
3	Lingadheeranahalli	3 BHK	1521.0	3.0	95.00	3	6245.890861
4	Kothanur	2 BHK	1200.0	2.0	51.00	2	4250.000000
5	Whitefield	2 BHK	1170.0	2.0	38.00	2	3247.863248
6	Old Airport Road	4 BHK	2732.0	4.0	204.00	4	7467.057101
7	Rajaji Nagar	4 BHK	3300.0	4.0	600.00	4	18181.818182
8	Marathahalli	3 BHK	1310.0	3.0	63.25	3	4828.244275
9	other	6 Bedroom	1020.0	6.0	370.00	6	36274.509804

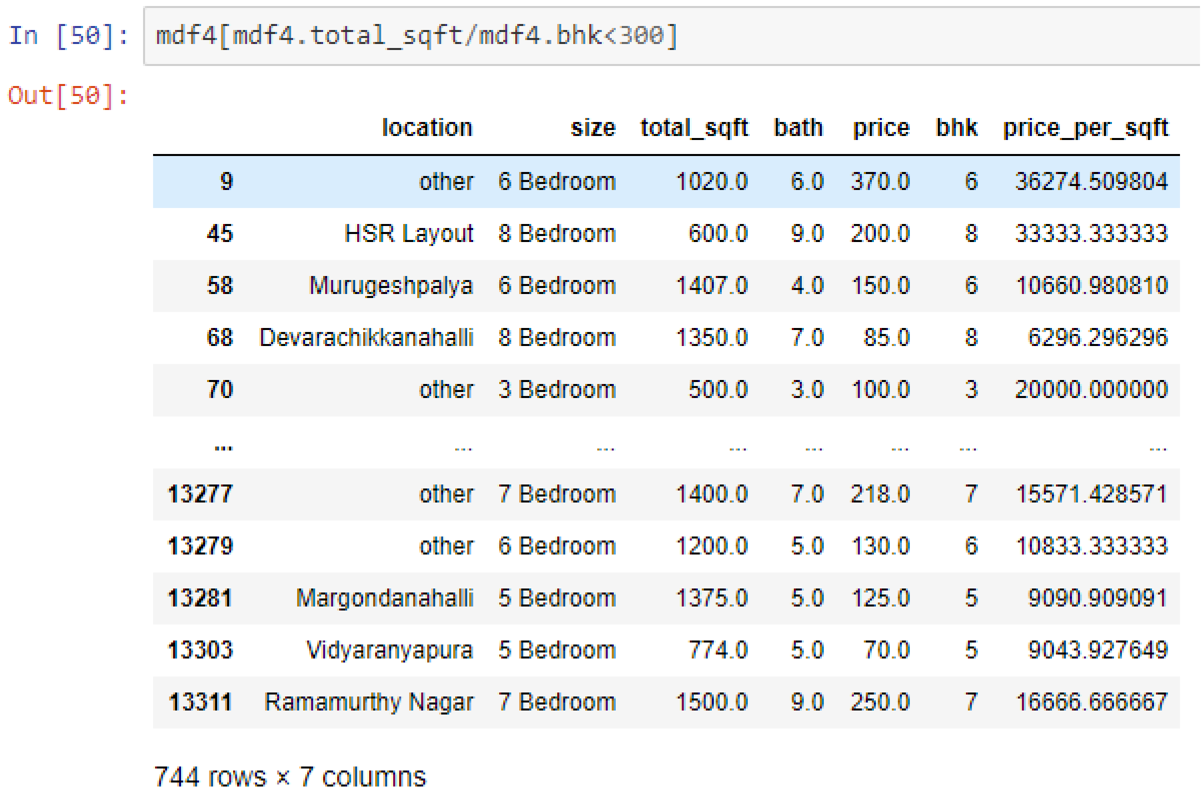
Figure 11: Dataframe after feature engineering

Source: Author

4.5 Outlier detection and removal

Outliers are the data points which are data errors or in some case they are not data errors but they represents the extreme variation in dataset. Although they are valid sometime, still it make sense to remove them otherwise they can create some issues for accuracy of model later on. So, this section will be focus on various type of outliers and their removal techniques.

In real estate domain, there are some basic concepts for square feet area per bedroom which cannot be less than some threshold value. This threshold values are decided by real estate owner or business manager. In our case we assume that we discussed with our business manager and by their opinion, there cannot be any bedroom with less than 300 square feet area. So, our task will be to detect data points with less than 300 square feet per bedroom. I will execute this command and check if our dataset have this kind of errors:



```
In [50]: mdf4[mdf4.total_sqft/mdf4.bhk<300]
Out[50]:
```

	location	size	total_sqft	bath	price	bhk	price_per_sqft
9	other	6 Bedroom	1020.0	6.0	370.0	6	36274.509804
45	HSR Layout	8 Bedroom	600.0	9.0	200.0	8	33333.333333
58	Murugeshpalya	6 Bedroom	1407.0	4.0	150.0	6	10660.980810
68	Devarachikkanahalli	8 Bedroom	1350.0	7.0	85.0	8	6296.296296
70	other	3 Bedroom	500.0	3.0	100.0	3	20000.000000
...	...	...	...	...	...	...	...
13277	other	7 Bedroom	1400.0	7.0	218.0	7	15571.428571
13279	other	6 Bedroom	1200.0	5.0	130.0	6	10833.333333
13281	Margondanahalli	5 Bedroom	1375.0	5.0	125.0	5	9090.909091
13303	Vidyaranyapura	5 Bedroom	774.0	5.0	70.0	5	9043.927649
13311	Ramamurthy Nagar	7 Bedroom	1500.0	9.0	250.0	7	16666.666667

744 rows × 7 columns

Figure 12: Square feet per bedroom Outliers detection

Source: Author

As we can see in above figure, we have 744 rows which have less than 300 square feet per bedroom, which is clearly data errors. To remove this data points from dataset I will execute following command:

```
mdf5= mdf4[~(mdf4.total_sqft/mdf4.bhk<300)]
mdf5.shape
```

```
output: (12502, 7)
```

This command creates new data frame mdf5 which is mdf4 with removed outliers of square feet per bedrooms.

Now, it's time to focus on price per square feet feature which we created in feature engineering chapter. Following command will be describe stats of price_per_sqft column:

```
mdf5.price_per_sqft.describe()
```

output:

```
count      12456.000000
```

```
mean        6308.502826
std         4168.127339
min          267.829813
25%         4210.526316
50%         5294.117647
75%         6916.666667
max       176470.588235
Name: price_per_sqft, dtype: float64
```

From the above statistical data, we observe that min price per square feet is 267.82 which is very rare in city like Bengaluru. Also, maximum price is 176470 which is extremely high but it might be possible for some luxurious house in prime location. Even though these are not errors, we need to remove this data points to make our data normally distributed. Otherwise, these extreme values might lead our model to wrong conclusion. In order to elimination of this values I will create following function:

```
def price_per_sqft_outlier_elimination(df):
    df_out =pd.DataFrame()
    for key, subdf in df.groupby('location'):
        m=np.mean(subdf.price_per_sqft)
        st = np.std(subdf.price_per_sqft)
        reduced_df =subdf[(subdf.price_per_sqft>(m-st)) & (subdf.pric
e_per_sqft<=(m+st))]
        df_out =pd.concat([df_out,reduced_df],ignore_index=True)
    return df_out
```

This function removes extreme values based on standard deviation. In our case we are assuming that our data should be normally distributed and most of data points should lie between mean and one standard deviation. Basic function of this function is it will take a dataframe as an input and group by location. Groping data by location is very important because price differs from location to location. So, per location I am getting 'subdf' (sub dataframe) for which I am calculating m (mean) and std (standard deviation). Then I am filtering data points which are beyond one standard deviation. So, anything above (m - std) and anything below (m + std) I will keep it in my reduced_df and I will keep appending those dataframe per location and it will give me output dataframe as df_out. Now, I will apply this function on mdf5 and create new dataframe mdf6 by following command:

```
mdf6=price_per_sqft_outlier_elimination(mdf5)
mdf6.shape
```

```
output: (10241, 7)
```

As per our output of new dataframe, we removed 2,261 data points with extreme values.

Now, we proceed to check if we have any outliers in our feature of bedrooms. There might be many cases where 2bhk homes have higher price than 3bhk in same location. There might be many reason for that such as property’s age and condition. But for our case we need to check if we have that data points or no in any random location. If we have, than it is necessary to remove or decrease those values for better accuracy of our model. For this task, I will create matplotlib PyPlot function to visualize our data in more convenient way:

```
def plot_scatter_distribution(df,location):
    bhk2 = df[(df.location == location) & (df.bhk == 2)]
    bhk3 = df[(df.location == location) & (df.bhk == 3)]
    matplotlib.rcParams['figure.figsize'] = (15,10)
    plt.scatter(bhk2.total_sqft,bhk2.price,  marker = '*',color ='blu
e', label = '2 BHK', s = 80)
    plt.scatter(bhk3.total_sqft,bhk3.price, color ='orange', label =
'3 BHK', s = 80)
    plt.xlabel("Total Square Feet Area")
    plt.ylabel("Price")
    plt.title(location)
    plt.legend()
    plt.grid()
```

This function will take dataframe and location as input and it will plot scatterplot for 2bhk vs 3 bhk. For example, if we take ‘Sarjapur road’ as location then our command and result will look like this:

```
plot_scatter_distribution(mdf6, "Sarjapur  Road")
```

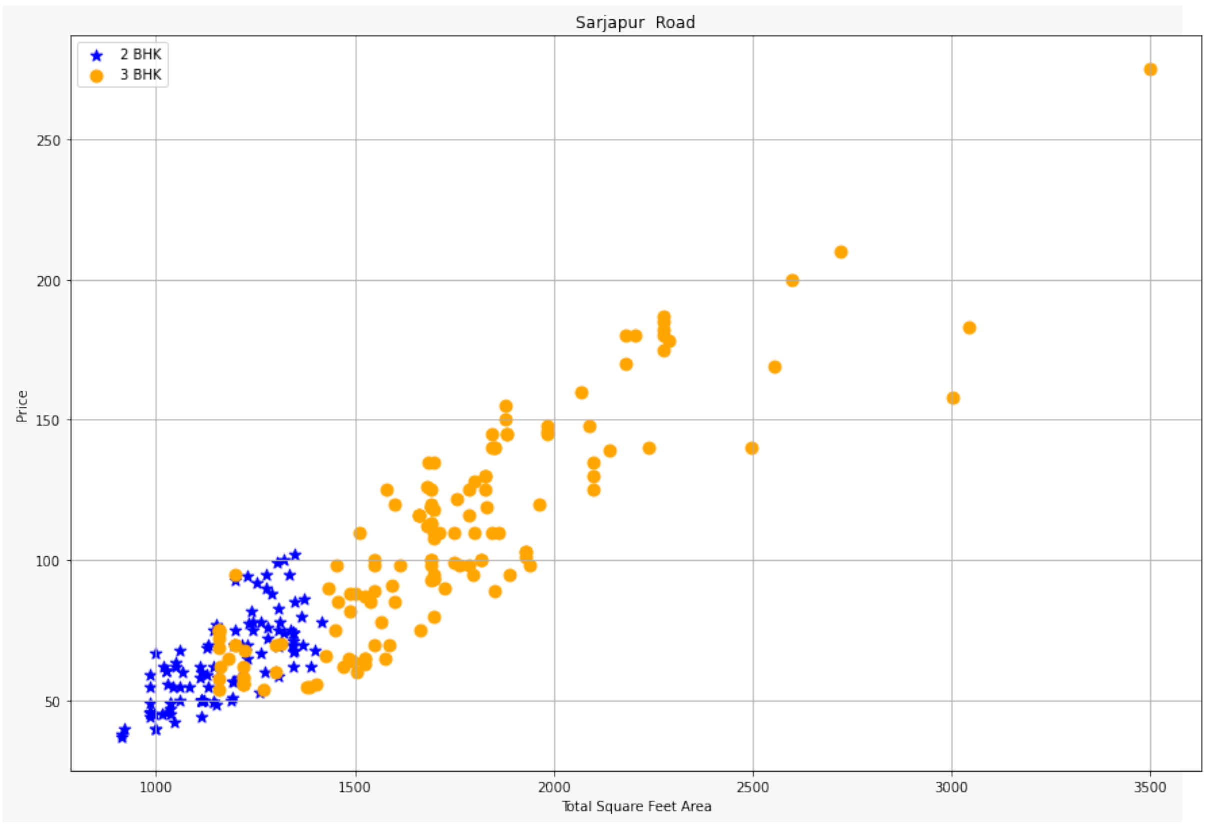


Figure 13: Price outliers of bhk detected

Source: Author

As we can see from above figure there are many data points of 3bhk which has less price than 2bhk. Our goal is to minimize those data points. To tackle this issue, I will create following function:

```
def bedroom_outliers_elimination(df):
    exclude_indices = np.array([])
    for location, location_df in df.groupby('location'):
        bhk_stats = {}
        for bhk, bhk_df in location_df.groupby('bhk'):
            bhk_stats[bhk] ={
                'mean' : np.mean(bhk_df.price_per_sqft),
                'std' : np.std(bhk_df.price_per_sqft),
                'count': bhk_df.shape[0]

            }
        for bhk, bhk_df in location_df.groupby('bhk'):
            stats = bhk_stats.get(bhk-1)
            if stats and stats['count']>5:
                exclude_indices = np.append(exclude_indices,bhk_df[bh
k_df.price_per_sqft<(stats['mean'])].index.values)
    return df.drop(exclude_indices, axis = 'index')
```

This function takes dataframe as an input. First it generate for loop for grouping dataframe (location_df) by location in which it generates another for loop to grouping dataframe (bhk_df) by bhk. For each bhk it will compute mean, standard deviation and count. Then, I created another for loop which will filter data points based on our desired filter (for example, price of 3bhk home should not be less then mean of 2bhk home). Now, I will apply this function to mdf6 and create new dataframe mdf7 by executing following command:

```
mdf7 = bedroom_outliers_elimination(mdf6)
mdf7.shape
```

```
output: (7329, 7)
```

In our new dataframe mdf7, we remain with 7329 rows. To cross check result of our function, I will plot same scatterplot as figure 13, but with new dataframe.

```
plot_scatter_distribution(mdf7, "Sarjapur  Road")
```

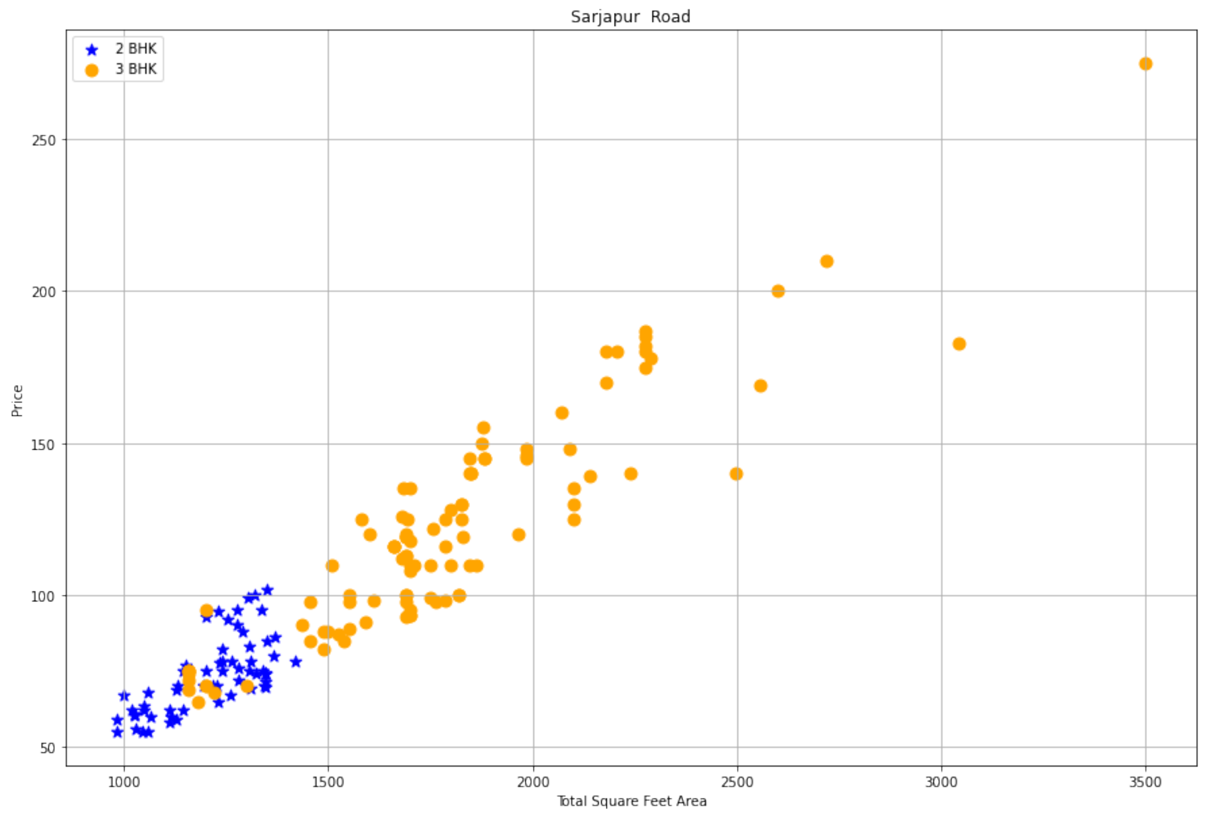


Figure 14: Price outliers of bhk removed

Source: Author

As we can see from above figure, we minimized price outliers of bhk compared to figure 13. It is very critical to remove them all but in my opinion this kind of abnormalities are fine to have because it is not necessary that 3bhk are always expensive than 2bhk in real life.

Now, I will plot a histogram with help of matplotlib to visualize density of homes per square feet by executing following code:

```
matplotlib.rcParams["figure.figsize"] =(20,10)
plt.hist(mdf7.price_per_sqft,rwidth = 0.9, color="teal")
plt.xlabel("Price Per Square Feet")
plt.ylabel("Count")
```

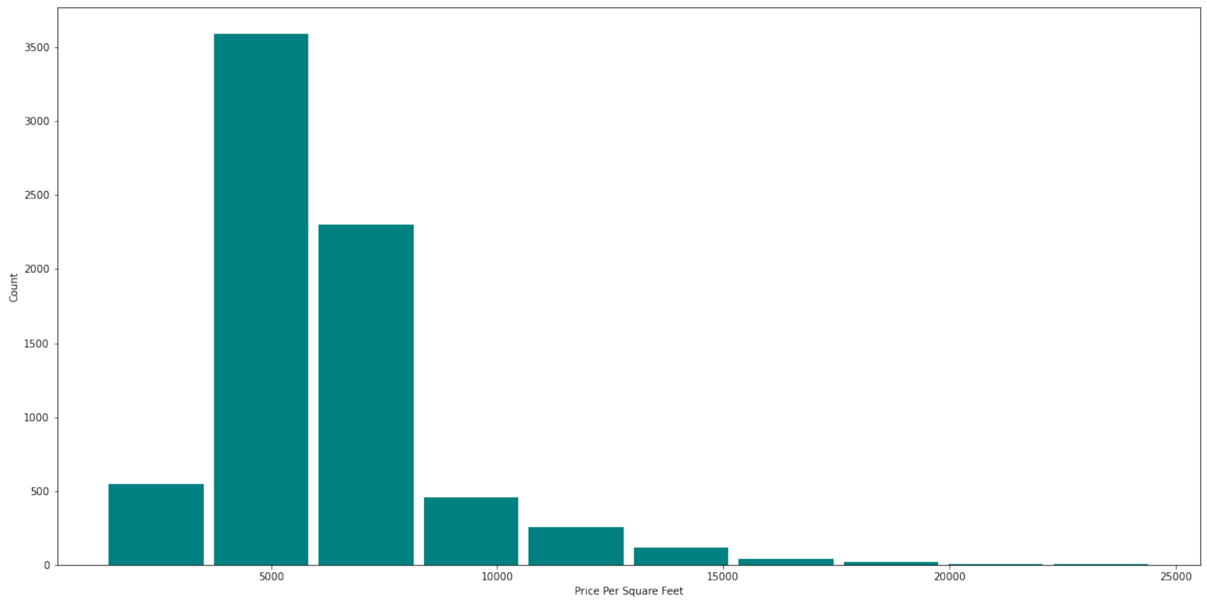


Figure 15: House density of price per square feet

Source: Author

As we can see from above figure, most of our house are in price range of 0-10000 per square feet. This is a gaussian curve that means now our data is normally distributed.

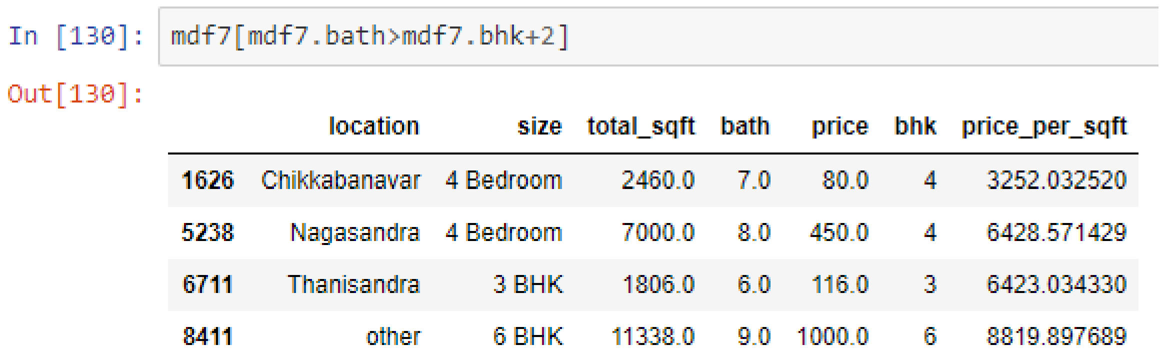
Now, let us explore bathroom feature. To see unique values of bathrooms I will execute following function:

```
mdf7.bath.unique()
```

output:

```
array([ 4.,  3.,  2.,  5.,  8.,  1.,  6.,  7.,  9., 12., 16., 13.])
```

As we can see we have huge amount of bathrooms like 13 and 16 as well. Here we need to check some criteria of how many bathrooms a particular house can have. In that case, business manager comes in frame to decide this criteria. For our task, business requirement is that one house cannot have more bathrooms than bedrooms+2. Because in real life, if we have 3bhk apartment we cannot have 6 bathrooms. If we have these types of house in our dataset, we will consider it as outliers and remove them. Following command and result will help us to check if we have house having more bathrooms than bedrooms+2



```
In [130]: mdf7[mdf7.bath>mdf7.bhk+2]
Out[130]:
```

	location	size	total_sqft	bath	price	bhk	price_per_sqft
1626	Chikkabanavar	4 Bedroom	2460.0	7.0	80.0	4	3252.032520
5238	Nagasandra	4 Bedroom	7000.0	8.0	450.0	4	6428.571429
6711	Thanisandra	3 BHK	1806.0	6.0	116.0	3	6423.034330
8411	other	6 BHK	11338.0	9.0	1000.0	6	8819.897689

Figure 16: bathroom outliers detected

Source: Author

We have only 4 bathrooms outliers. To remove them I will create new dataframe mdf8 by executing following command:

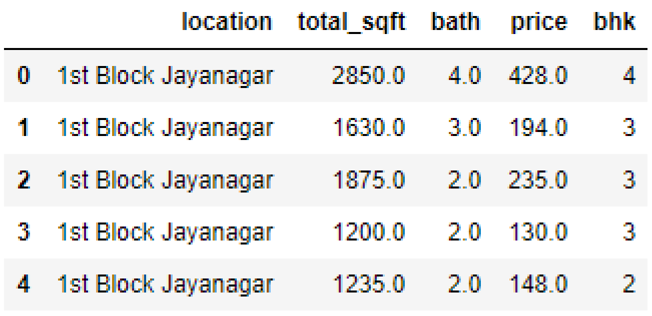
```
mdf8=mdf7[mdf7.bath<mdf7.bhk+2]
mdf8.shape
```

```
output: (7251, 7)
```

Now we removed all outliers from our dataframe. So, I will drop unnecessary columns and create a new dataframe mdf9 by executing following function:

```
mdf9 = mdf8.drop(['size','price_per_sqft'], axis = 'columns')
mdf9.head()
```

output:



	location	total_sqft	bath	price	bhk
0	1st Block Jayanagar	2850.0	4.0	428.0	4
1	1st Block Jayanagar	1630.0	3.0	194.0	3
2	1st Block Jayanagar	1875.0	2.0	235.0	3
3	1st Block Jayanagar	1200.0	2.0	130.0	3
4	1st Block Jayanagar	1235.0	2.0	148.0	2

Figure 17: Final dataframe after all outliner removal

Source: Author

Above figure represents final dataset after all process of data cleaning, feature engineering and outliers removals. This dataframe contains 7251 rows and 5 columns. This dataset is now ready for model building and machine learning process.

4.6 Machine learning model building

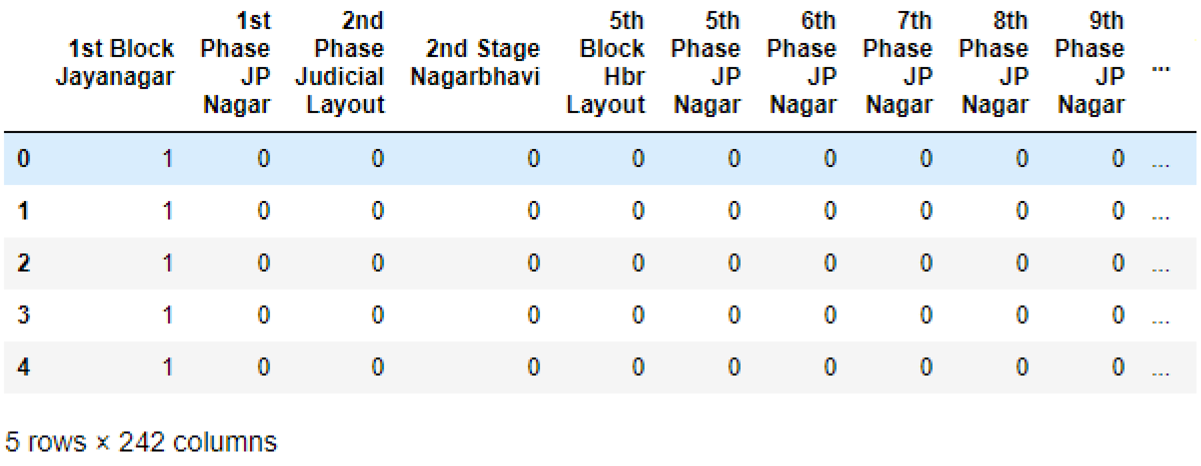
This chapter is focused on building a machine learning model. Various machine learning methods will be applied on our dataset such as 'one-hot-encoding' and 'train test split'. Cross validation and hyperparameter tuning process will be discussed in this section to select best machine learning algorithm and their best parameter as well.

4.6.1 One-hot-encoding

In our final dataset, we have 5 columns from which location is containing text data. For machine learning process, it is necessary to have numerical data only otherwise it cannot be proceed further. One of the best method for transforming categorical feature into numerical feature is 'one-hot-encoding', it is also known as dummies method. To perform one-hot-encoding on our dataset I will execute following command:

```
dummies = pd.get_dummies(mdf9.location)
dummies.head()
```

output:



	1st Block Jayanagar	1st Phase JP Nagar	2nd Phase Judicial Layout	2nd Stage Nagarbhavi	5th Block Hbr Layout	5th Phase JP Nagar	6th Phase JP Nagar	7th Phase JP Nagar	8th Phase JP Nagar	9th Phase JP Nagar	...
0	1	0	0	0	0	0	0	0	0	0	...
1	1	0	0	0	0	0	0	0	0	0	...
2	1	0	0	0	0	0	0	0	0	0	...
3	1	0	0	0	0	0	0	0	0	0	...
4	1	0	0	0	0	0	0	0	0	0	...

5 rows × 242 columns

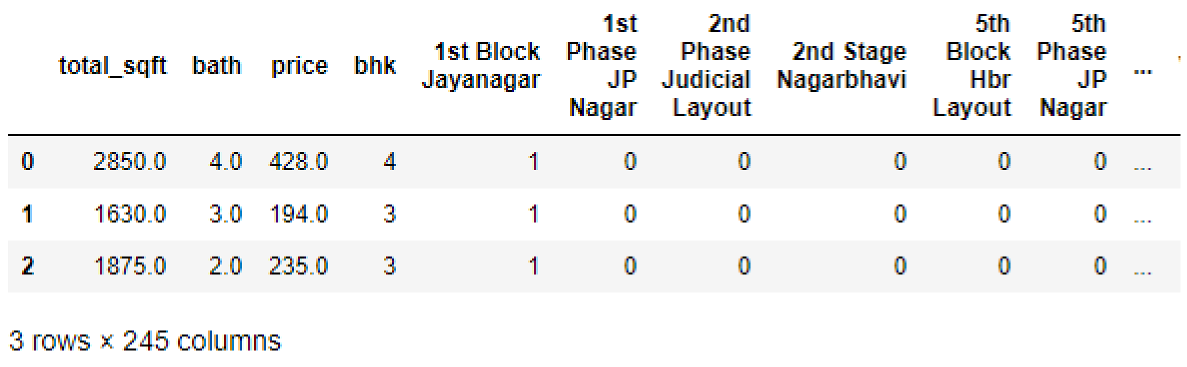
Figure 18: dummy columns created for location

Source: Author

As we can see from above figure, all 242 location categories are converted into dummy columns, which are stored in separate dataframe called ‘dummies’. If data value in particular column represent 1 then it means that home is located there and the rest columns values will be 0. Now, I will join both dataframe dummies and mfd9 together using concat function as shown below:

```
mdf10 = pd.concat([mdf9,dummies.drop('other', axis = 'columns')], axi
s = 'columns')

mdf11 = mdf10.drop('location', axis = 'columns')
mdf11.head(3)
```

In first line of code, I created new dataframe mdf10 which join mdf9 and dummies dataframe together. I also dropped column ‘other’ because in dummies we can live with 1 less column and we can know it’s value from values of other columns. For example, if the rest of columns will be 0, then we can assume that the home will be in other category. In second line I created new dataframe mdf11 by dropping location column from the previous dataframe because now we have all location as columns so it does not make sense to have them in row. The result of above code is represented in following figure:



	total_sqft	bath	price	bhk	1st Block Jayanagar	1st Phase JP Nagar	2nd Phase Judicial Layout	2nd Stage Nagarbhavi	5th Block Hbr Layout	5th Phase JP Nagar	...
0	2850.0	4.0	428.0	4	1	0	0	0	0	0	...
1	1630.0	3.0	194.0	3	1	0	0	0	0	0	...
2	1875.0	2.0	235.0	3	1	0	0	0	0	0	...

3 rows × 245 columns

Figure 19: Converted location into numerical category with one-hot-encoding

Source: Author

Actual shape of the mdf11 is:

```
mdf11.shape
output: (7251, 245)
```

4.6.2 Train-Test split

Train-test method is used to estimate performance of machine learning algorithm. We can also train the model using entire dataset but it is not an idol approach. Because our model already seen whole data so it will give 100% accuracy on machine learning algorithm. That is why train test split method idol strategy when dataset is huge. Basic concept of train-test split is we divide our dataset in two parts- training data and testing data, ratio can be anything, but 80:20 is most common and we will take the same in our case. We will train model with 80% of data and test model accuracy with 20% of data.

Before we divide our dataset into training and testing, I will create dependent and independent variable from dataset. In our case, house size is dependent variable as we are going to predict it based on other variable and rest of the variables will be independent. The code for this is:

```
X = mdf11.drop('total_sqft', axis = 'columns')
y = mdf11.total_sqft
```

X is defined as independent variable and dropped total_sqft column from it. Y is defined as dependent variable, which contains only total_sqft column.

Now, I will import train-test split method from Scikit learn model selection library by executing following code:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test =train_test_split(X,y,test_size = 0.
2, random_state = 10)
```

We divided our data and used 20% as test data and remaining 80% as training data. Random state parameter is set as 10 so train-test data will remain same each time. If we do not provide random state variable or set as 0, then data will be randomly changed each time when we execute this code.

4.6.3 Machine learning algorithms and accuracy estimation

Our model is ready to train by machine learning algorithm now. First, I will try linear regression algorithm and examine score of model. I will import linear regression model from scikit learn linear model and code for this is:

```
from sklearn.linear_model import LinearRegression
lr_clf = LinearRegression()
lr_clf.fit(X_train,y_train)
lr_clf.score(X_test,y_test)
print("The trained Linear Regression Model has a score of : " + str(l
r_clf.score(X_test,y_test)))
```

output:

```
The trained Linear Regression Model has a score of : 0.85253305552204
74
```

We got pretty decent score from linear regression algorithm but it was for only one random split, which might have different score for other split. So, here I would like to import K-fold cross validation method and perform shuffle-split method to get more reliable results of particular algorithm. I have elaborated detail theoretical concept and principal of K-fold cross validation in literature review part in chapter 3.2.1. the code for this task is below:

```
from sklearn.model_selection import ShuffleSplit
from sklearn.model_selection import cross_val_score

cv = ShuffleSplit(n_splits =5, test_size = 0.2, random_state = 0)

print("The output correlation array is as shown below; ")

cross_val_score(LinearRegression(), X, y ,cv = cv)
```

```
output:
The output correlation array is as shown below;

array([0.83006819, 0.74838861, 0.84647633, 0.76505671, 0.74945186])
```

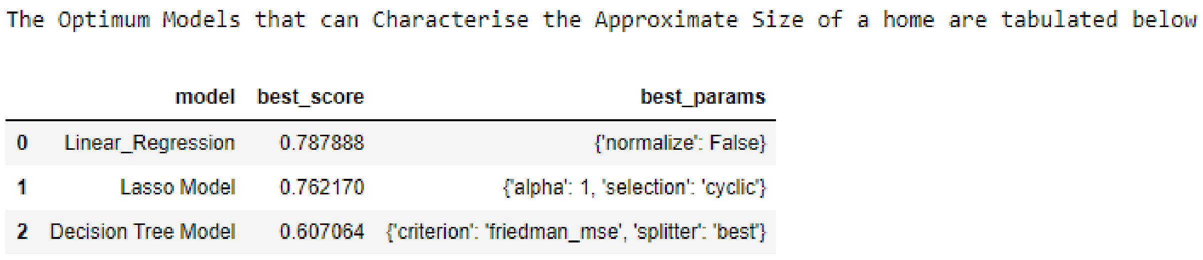
I set shuffle-split as 5 which means our data will be split in to 5 random parts and will be shuffled each time and will return to their individual scores. We have observed from above score that linier regression is giving us very high accuracy in every splits. But in real life project it is very important to compare other algorithms as well to obtain as much higher accuracy as we can. So, my text step will be to compare lasso regression and decision tree regressor algorithm with linear regression. Along with it, I will import grid search cv to come up with best parameter of each algorithm. Theoretical concepts and principal of grid search cv is described in literature review part in chapter 3.3.1. the code for this task is below:

```
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import Lasso
from sklearn.tree import DecisionTreeRegressor

def look_for_best_model_and_tuning_parameters(X,y):
    algos = {
        'Linear_Regression':
        {
            'model' :LinearRegression(),
            'params':
            {
                'normalize': [True, False]
            }
        },


        'Lasso Model' :
        {
            'model': Lasso(),
            'params':
            {
                'alpha' : [1, 2],
                'selection' : ['random', 'cyclic']
            }
        },

        'Decision Tree Model' :
        {
            'model':DecisionTreeRegressor(),
                'params':
                {
                    'criterion' :['mse', 'friedman_mse'],
                    'splitter' : ['best','random']
                }
            }
        }



    scores = []
    cv = ShuffleSplit(n_splits = 5, test_size = 0.2, random_state = 0
)
    for algo_name, config in algos.items():
        gs = GridSearchCV(config['model'], config['params'], cv = cv,
return_train_score=False)
        gs.fit(X,y)
        scores.append({
            'model' : algo_name,
            'best_score' :gs.best_score_,
            'best_params' : gs.best_params_,
        })
    return pd.DataFrame(scores,columns=['model','best_score','best_pa
rams'])
```

In this code, first I imported grid search cv, lasso regression model and decision tree model from scikit learn. Then I defined a function which takes x(independent variable) and y(dependent variable) as input and it goes by each algorithms and their parameters. In last part, I created score object to store scores of each algorithms in a dataframe. Gs will configure each algorithm and their parameters and cv is set for 5 times shuffle-split. I will execute following command to apply this function on our dataset:

```
print("The Optimum Models that can Characterise the Approximate Size
of a home are tabulated below ")
look_for_best_model_and_tuning_parameters(X,y)
```

output:



```
The Optimum Models that can Characterise the Approximate Size of a home are tabulated below
```

	model	best_score	best_params
0	Linear_Regression	0.787888	{'normalize': False}
1	Lasso Model	0.762170	{'alpha': 1, 'selection': 'cyclic'}
2	Decision Tree Model	0.607064	{'criterion': 'friedman_mse', 'splitter': 'best'}

Figure 20: Scores and parameters values comparison of algorithms

Source: Author

4.6.4 Function for home size prediction

As we can see from above figure, linear regression is winner for this model with highest accuracy score of 78.78%. So, I will use linear regression algorithm to build house size prediction model. We have already created lr_clf (linear regression model) previously, so I will just create a function home_size_estimation and use this lr_clf model in it. The code for this task is below:

```
def home_Size_Estimation(location,price,bath,bhk):
    loc_index = np.where(X.columns == location)[0][0]


    x = np.zeros(len(X.columns))
    x[0] = price
    x[1] = bath
    x[2] = bhk
    if loc_index >= 0:
        x[loc_index] = 1
    return lr_clf.predict([x])[0]
```

Above function takes location, price, bath and bhk as input and return estimated size of house.

4.6.5 Exporting model

Finally, our size house prediction model is built. We will interpret results of this function in result and discussion part of this thesis. For now, I will export this model into pickle file by executing following code:

```
import pickle
with open('House_Price_Model.pickle', 'wb') as f:
    pickle.dump(lr_clf,f)
```

It is very simple to export our model in pickle file, we just define model name and dump model classifier in it. This pickle file does not contain actual data but it contains only coefficients, intercepts and other parameters. Other than the model we also need columns information as we can see in our home_size_estimation function location and index of columns are very important. For storing those column, I will use json file and execute following command:

```
import json
columns={
    'data_columns' : [col.lower() for col in X.columns]
}
with open("columns.json", "w") as f:
    f.write(json.dumps(columns))
```

5. Results and discussion

5.1 Result interpretation of house size estimation function

To predict area occupied by a home, we have to call a function 'home_size_estimation'. Parameters of the function are illustrated as below:

Home_size_estimation(location, price, bath, bhk).

This model will ask for inputs from customer and will predict size of home by given values. Let us assume, customers have already their decided budget of 20 lakh INR. As per their family size they are looking for 3 bath and 3 bhk apartment. But their main concern is location as they are confused in three location and they want to know which location will offer them bigger house. So, here we have results for 3 different location:

```
1)home_Size_Estimation('Raja Rajeshwari Nagar',20,3,3)
  Output: 2607.9527865108266


2) home_Size_Estimation('Whitefield',20,3,3)
   Output: 2533.4277211332465


3) home_Size_Estimation('Sarjapur  Road',20,3,3)
   Output: 2433.19471401022
```

As we can see from the output, the customer would go for 'Raja Rajeshwari Nagar' because this area provides bigger size of house then other 2 in the same price. They can also change the criteria and the output will keep changing accordingly as well.

5.2 Discussion

My main motivation for thesis was to create a unique yet very interesting and valuable approach to make improvements in real estate industry with the help of data science techniques. As per our results, I obtained it by building a home size prediction model. As we all know, some reputed real estate websites have house price prediction model but the most valuable thing we look other than price is the size of house. Because generally, customers get approval of their bank loan before buying any property and their budget is fixed and there is no reason to look for house price anymore. So, their main concern will be to choose best and specious house from their desired location. In this case, this feature of size prediction will be very helpful and flexible to decide location of the house.

Even though our model have pretty high accuracy of 78.78%, there are certain limitations and ideas to improve accuracy score. The limitations are features in our data as size does not depends on only bedrooms and baths. There might be many luxurious houses with gardens, garages and terrace which influence house size a lot. Our dataset was only about one city and limited features. But we can take it on next level by increasing data size with multiple cities and analysing their other features as well. We also removed null values and outliers from our dataset to keep it short for this thesis, but in real world these values can be filled with various techniques as per business requirements. There are various algorithm such as XG boost and RNN (recurrent neural network) which can be applied in order to get more accurate score.

After working on this research topic, I came to know a fact that house price or size are just a number which depends on a lots of factor and have their own limitations. For example, house price cannot be only decided by size, feature and location of the house. It is also dependent on quality and age of the house. This problem also can be solved by image processing of the house with the help of technologies of data science, artificial intelligence and neural network. This is very interesting approach to focus on and might be my future research topic.

6. Conclusion

The bachelor thesis objective was to study literature publications and web resources to know modern data science and machine learning techniques along with various python libraries for building a home size prediction model. This bachelor thesis represents life cycle of a real-world data science project which includes various stages like accessing raw data, data cleaning, feature engineering, outliers detection and removal, machine learning methods and algorithm and finally model creation. In the second chapter of this thesis, I discussed goal of the study and in depth methodologies used to achieve desired results from this process.

The third chapter of the study demonstrate literature review to understand theorical concepts of topics like machine learning methods and algorithms, python and its libraries with used references and bibliography. In the fourth chapter of this study, I conducted practical part based on methodologies to obtain objectives of this thesis. The main use of 'home size estimation function' and its results to make wise decision to buy a house are discussed in the fifth chapter of the study results and discussion.

To sum up, data science is a must have technology for development of any organization or just for staying in market as well. Because every day there are new data science techniques and machine learning algorithms are being discovered which makes easy to develop new business strategies and technological development for better human life. In my opinion, every business or industry should approach new data driven technologies. Whether it is a new startup or a giant organization, whether it is product based company or service based company, they should use their data wisely to make smart decisions which not only impact society economically but socially as well. Finally, I conclude my thesis and affirm that I have accomplished all the objectives which were assigned to me.

7. References

{Matplotlib: A 2D graphics environment. **Hunter, J. D. 2007.** 3, s.l. : IEEE COMPUTER SOC, 2007, Vol. 9, pp. 90-95.

A Comparative Study of Ordinary Cross-Validation, v-Fold Cross-Validation and the Repeated Learning-Testing Methods. **Burman, Prabir. 1989.** 3, Davis, California : Oxford university press, 1989, Vol. 76. ISSN: 00063444.

A survey of cross-validation procedures. **Celisse, Sylvain Arlot & Alain. 2010.** 40-79, s.l. : Statist. Surv., 2010, Vol. 4. ISSN : 1935-7516.

Bayesian and LASSO Regressions for Comparative Permeability Modeling of Sandstone Reservoirs. **AI-Mudhafar, watheq J. 2018.** 1, s.l. : Springer US, February 10, 2018, Natural Resources Research, Vol. 28, pp. 47-62. ISSN: 1520-7439.

Bellman, Richard Ernest. 2010. *Dynamic Programming.* NJ, United States : Princeton university press, 2010. ISBN:978-0-691-14668-3.

Bernd Klein, Bodenseo. 2011. Nuerical & scientific computing with Python. *introduction into pandas.* [Online] 2011. https://www.python-course.eu/pandas.php.

Bhatia, Richa. 2012. Analytics India Magazine. *WHY DO DATA SCIENTISTS PREFER PYTHON OVER JAVA?* [Online] Analytics India Magazine, 2012. https://analyticsindiamag.com/why-do-data-scientists-prefer-python-over-java/.

Boudreau, Emmett. 2020. Scientific Python With Lambda. *The exact meaning a proper usage of Python's Lambda function: Python's greatest syntax for scientific programming.* October 22, 2020.

Brownlee, Jason. 2015. Start machine learning. *Basic concepts in machine learning.* December 25, 2015.

Chakraborty, Amitabha. 2017. Bengaluru House price data. *Kaggle.* [Online] October 2017. [Cited: October 31, 2020.] https://www.kaggle.com/amitabhajoy/bengaluru-house-price-data.

Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin. 2003. *A Practical Guide to Support Vector Classification.* Taiwan : research gate, research gate, 2003.

documentation, NumPy. NumPy v1.19 Manual. *NumPy.* [Online] The SciPy community. [Cited: September 16, 2020.] https://numpy.org/doc/stable/.

documentation, pandas. API reference. *Pandas.* [Online] the pandas development team. [Cited: September 17, 2020.] https://pandas.pydata.org/docs/reference/index.html.

Eppler, Jochen Martin. 2015. A convinient interface to the NEST simulator. [book auth.] Eilif Muller. *python in neuroscience.* Nijmegen : Rolf Kotter, 2015.

Evaluation of Decision Tree Pruning Algorithms for Complexity and Classification Accuracy. **Patil, Dipti D. 2010.** 02, s.l. : International Journal of Computer Applications, 2010, International Journal of Computer Applications, Vol. 11.

Foote, Keith D. 2019. A Brief History of Machine Learning. *Data topics.* [Online] Dataversity, March 26, 2019. [Cited: August 05, 2020.] https://www.dataversity.net/a-brief-history-of-machine-learning/.

Glen, Stephanie. 2015. Lasso Regression: Simple Definition. *StatisticsHowTo.* [Online] September 24, 2015. [Cited: August 06, 2020.] https://www.statisticshowto.com/lasso-regression/.

Gupta, Mohit. 2018. ML | Introduction to Data in Machine Learning. *Geeksforgeeks.* [Online] may 01, 2018. [Cited: August 01, 2020.] https://www.geeksforgeeks.org/ml-introduction-data-machine-learning/?ref=lbp.

Hackeling, Gavin. 2017. *Mastering machine learning with scilkit learn.* Birmingham : packt publishing ltd., 2017. ISBN: 978-1-78829-987-9.

Hsu, Hansen. 2018. *2018 MUSEUM FELLOW GUIDO VAN ROSSUM, PYTHON CREATOR & BENEVOLENT DICTATOR FOR LIFE.* s.l. : computer history museum, 2018.

John Huter, Darren dale. 27 may, 2007. *The Matplotlib User's Guide.* 27 may, 2007.

kassambara. 2018. Simple Linear Regression in R. *STHDA Statistical tools for high-throughput data analysis.* [Online] March 10, 2018. [Cited: February 20, 2021.] http://www.sthda.com/english/articles/40-regression-analysis/167-simple-linear-regression-in-r/.

Kotsiantis, S.B. 2007. Supervised machine learning- A review of classification techniques. [book auth.] Ilias G. Maglogiannis. *Emerging Artificial Intelligence Applications in Computer.* Greece : University of Peloponnese, 2007.

Leo (Liang-Huan) Chin, Tanmay Dutta. 2016. *NumPy Essentials.* Birmingham : Packt publishing ltd., 2016. p. 11. ISBN: 978-1-78439-367-0.

LLC, Cogito Tech. 2019. What are Features in Machine Learning and Why it is Important? *Medium.* [Online] Medium, July 15, 2019. [Cited: August 01, 2020.] https://medium.com/@cogitotech/what-are-features-in-machine-learning-and-why-it-is-important-e72f9905b54d.

McKinney, Wes. january 2011. *1pandas: a Foundational Python Library for DataAnalysis and Statistics.* [Article] s.l. : researchgate, january 2011.

—. 2012. *Python for data analysis.* 1st eddition. Sebastopol : O'reilly media, 2012. ISBN: 978-1-449-31979-3.

Mehryar Mohri, Afshin Rostamizadeh, Ameet Talwalkar. 2018. *foundation of machine learning.* 2nd eddition . Cambridge, England : the MIT press, 2018. ISBN: 978-0-262-03940-6.

Oliphant, Travis E. 2015. *Guide to numpy .* 2nd eddition. s.l. : creater space, 2015. ISBN: 978-1-51730-007-4.

Pierce, Rod. 2018. Equation of a Straight Line. *MathsIsFun.com.* [Online] Rod Pierce DipCE BEng, october 12, 2018. [Cited: july 31, 2020.] http://www.mathsisfun.com/equation_of_line.html.

Plato. 2017. *Republic.* s.l. : Pangiun books, 2017. p. 480. ISBN: 0140455116.

scikit-learn, Machine Learning in {P}ython. **Pedregosa, F. and Varoquaux, G. and Gramfort, A. and Michel, V. 2011.** s.l. : Journal of Machine Learning Research, 2011, Vol. 12, pp. 2825--2830.

Sharma, Abhishek. 2020. 4 Simple Ways to Split a Decision Tree in Machine Learning. *decision tree split methods.* [Online] June 30, 2020. [Cited: August 30, 2020.] https://www.analyticsvidhya.com/blog/2020/06/4-ways-split-decision-tree/.

Suzuki, Kunihiro. 2019. *statistic - the fundamentals.* New York : Nova Science Publishers, Incorporated, 2019. p. 162. Vol. 1. ISBN : 9781536144628.

The group lasso for logistic regression. **Meier, Lukas. 2008.** 1, Zurich : J.R. statist, January 04, 2008, Journal of the royal statistical society , Vol. 70, pp. 53-71. 1369-7412/08/70053.

Tosi, Sandro. November 2009. *Matplotlib for python developers.* [ed.] Rakesh Shejwal. Birmingham, UK : Packt publishing ltd., November 2009. ISBN: 978-1-847197-90-0.

Weisberg, Stanford. 2005. *Applied Linear Regression.* 3rd edition. Hoboken, New Jersey : John Wiley & sons, 2005. p. 24. ISBN : 0-471-66379-4.