A
Minor Project
Report on

House Price Prediction Using Machine Learning With GUI

Submitted in partial fulfillment of the requirements for the award of degree of

Bachelor of Technology
in
Computer Engineering

by
Karan (17001003025)
And
Manoj(17001003030)

Under supervision of
Dr. Mamta Kathuria

Department of Computer Engineering

J. C. BOSE UNIVERSITY OF SCIENCE & TECHNOLOGY, YMCA FARIDABAD-121006

May 2020

CANDIDATE’S DECLARATION

I hereby certify that the work which is being carried out in this Minor Project titled **“House price prediction using Machine Learning with GUI”** in fulfillment of the requirement for the
degree of Bachelor of Technology in Computer Engineering and submitted to “**J. C. BoseUniversity of Science and Technology, YMCA, Faridabad**”, is an authentic record of my own work carried out under the supervision of Dr. Mamta Kathuria.

The work contained in this thesis has not been submitted to any other University or Institute for the award of any other degree or diploma by me.

Karan
(17001003025)
Manoj
(17001003030)

CERTIFICATE

This is to certify that the work carried out in this project titled **"House price prediction using Machine Learning with GUI"** submitted by Karan and Manoj to "**J. C. Bose University of Science and Technology, YMCA, Faridabad**" for the award of the degree of Bachelor of Technology in Computer Engineering is a record of bonafide work carried out by them under my supervision. In my opinion, the submitted report has reached the standards of fulfilling the requirements of the regulations to the degree.

Dr. Mamta Kathuria
(Supervisor)
Professor,
Department of Computer Engg,
J. C. Bose University of Science and Technology, YMCA, Faridabad

Dr. Komal Kumar Bhatia
Chairman,
Department of Computer Engg,
J. C. Bose University of Science and Technology, YMCA, Faridabad

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

1.1 Abstract

Machine learning plays a major role from past years in image detection, spam reorganization, normal speech command, product recommendation and medical diagnosis. Present machine learning algorithm helps us in enhancing security alerts, ensuring public safety and improve medical enhancements. Machine learning system also provides better customer service and safer automobile systems. In the present paper we discuss about the prediction of future housing prices that is generated by machine learning algorithm. For the selection of prediction methods we compare and explore various prediction methods. We utilize lasso regression as our model because of its adaptable and probabilistic methodology on model selection. Our result exhibit that our approach of the issue need to be successful, and has the ability to process predictions that would be comparative with other house cost prediction models. More over on other hand housing value indices, the advancement of a housing cost prediction that tend to the advancement of real estate policies schemes. This study utilizes machine learning algorithms as a research method that develops housing price prediction models. We create a housing cost prediction model In view of machine learning algorithm models for example, XGBoost, lasso regression and neural system on look at their order precision execution. We in that point recommend a housing cost prediction model to support a house vender or a real estate agent for better information based on the valuation of house. Those examinations exhibit that lasso regression algorithm, in view of accuracy, reliably outperforms alternate models in the execution of housing cost prediction.

1.2 Problem statement and related work

In India, there are multiple real estate classified websites where properties are listed for sell/buy/rent purposes such as 99acres , housing , commonfloor , magicbricks and more. However, in each of these websites we can see lot of inconsistencies in terms of pricing of an apartment and there are some cases when similar apartments are priced differently and thus there is lot of in-transparency. Sometimes the consumers may feel the pricing is not justified for a particular listed apartment but there no way to confirm that either. Proper and justified prices of properties can bring in a lot of transparency and trust back to the real estate industry, which is very important as for most consumers especially in India the transaction prices are quite high and addressing this issue will help both the customers and the real estate industry in the long run. We propose to use machine learning and artificial intelligence techniques to develop an algorithm that can predict housing prices based on certain input features.

The business application of this algorithm is that classified websites can directly use this algorithm to predict prices of new properties that are going to be listed by taking some input variables and predicting the correct and justified price i.e. avoid taking price inputs from customers and thus not letting any error creeping in the system. This study on proactive pricing of houses in the Indian context has never been reported earlier in the literature to the best of our knowledge.

However, the problem of house price prediction is quite old and there have been many studies and competitions addressing the same including the classic Boston housing price challenge on Kaggle . As far as housing price prediction in Indian context is

concerned, used machine learning techniques such as XGBoost for predicting housing prices in Bengaluru, India. MachineHack conducted an hackathon on
predicting housing prices in Bengaluru in 2018. The problem statement was to predict the price of houses in Bengaluru given 9 features such as area type, availability, location, price, size, society, total square foot, number of bathrooms and bedrooms. Moreover, there have been other studies for house price prediction in other cities of India such as Mumbai .

1.3 Data

The crucial element in machine learning task for which a particular attention should be clearly taken is the data. Indeed the results will be highly influenced by the data based on where did we find them, how are they formatted, are they consistent, is there any outlier and so on. At this step, many questions should be answered in order to guarantee that the learning algorithm will be efficient and accurate.

The dataset used in this project comes from a Kaggle competition.
This data was collected in 2010 and each of the 25000 entries represents aggregate information about 22 features of homes from various suburbs located in Boston.

The features can be summarized as follows:

id- notation for a house

dateDate- house was sold

pricePrice- is prediction target

bedrooms- Number of Bedrooms/House

bathrooms- Number of bathrooms/bedrooms

sqft_livingsquare - footage of the home

sqft_lotsquare -footage of the lot

floorsTotal - floors (levels) in house

waterfrontHouse - which has a view to a waterfront

view - Has been viewed

condition - How good the condition is (Overall). 1 indicates worn out property and 5 excellent.

gradeoverall - grade given to the housing unit, based on King County grading system. 1 poor,13 excellent.

sqft_abovesquare - footage of house apart from basement

sqft_basementsquare - footage of the basement

yr_built - Built_Year

yr_renovated - Year when house was renovated

zipcode -zip

lat- Latitude coordinate

long - Longitude coordinate

sqft_living15 - Living room area in 2015(implies-- some renovations) This might or might not have affected the lotsize area

sqft_lot15 - lotSize area in 2015(implies-- some renovations)

We'll now open a python 3 Jupyter Notebook and execute the following code snippet to load the dataset and remove the non-essential features. Recieving a success message if the actions were correclty performed.

As our goal is to develop a model that has the capacity of predicting the value of houses, we will split the dataset into features and the target variable. And store them in features and prices variables, respectively.

1.4 Motivation for The Project

Being extremely interested in everything having a relation with the Machine Learning, the independant project was a great occasion to give me the time to learn and confirm my interest for this field. The fact that we can make estimations, predictions and give the ability for machines to learn by themselves is both powerful and limitless in term of application possibilities. We can use Machine Learning in Finance, Medicine, almost everywhere. That’s why we decided to conduct project around the Machine Learning.

1.5 Idea

As a first experience, I wanted to make my project as much didactic as possible by approaching every different steps of the machine learning process and trying to understand them deeply. Known as “toy problem” defining the problems that are not immediate scientific interest but useful to illustrate and practice, I chose to take Real Estate Prediction as approach. The goal was to predict the price of a given apartment according to the market prices taking into account different “features” that will be developed in the following sections.

1.6 Tools

Numpy – To perform basic calculation.

Pandas – To collect dataset and performing changes on dataset.

Sklearn – It is the most important library .All the major work is done under this library like splitting the dataset, training and prediction.

Tkinter – It is used to give the model a graphical interface to the user.

CHAPTER 2 : LITERATURE REVIEW

2.1 Overview

For every project the literature review will give clear idea and it will serve as the base line here most of the authors have concluded that artificial neural networks have the more influence in predicting but in the real world the other algorithms should be also taken into consideration. by conducting this study it helps to know about both the pros and cons and it had helped me to successfully implement the project.

At present each framework may be moved towards innovation for the simplicity from claiming operations. The training framework will be moving towards e-taking. Individuals tend to move from the manual to robotized methodology. That primary goal of the this will be will anticipate that lodging cost with admiration to the plan of the clients. Those exhibit strategies may be An long procedure in which those customers necessities to contact the land operator. The land operators give acceptable A suggestive on the lodging costs prediction. This strategy includes high hazard a direct result the land operator might furnish the bad data of the clients. They employments those straight relapse calculations should figure the cost. This analyses likewise utilized to foresee the best area for the clients to purchasing the houses. The information here utilized is from those Mumbai lodging board since 2009. Eventually, Tom's perusing utilizing this straight relapse he predicted the rate for every square foot. This prediction indicates the square feet of the house will be raised Eventually Towards 2018.(Bhagat et al.; 2016) The mankind's wealthiness is measured Eventually Purchasing a house includes a considerable measure from claiming consolidated choices. Concerning illustration, the same approach offering A house may be additionally troublesome. There needs aid where both the customers and sellers should get them an equal amount of profit. A different model will be carried Eventually that is the Cox regression model for those exact prediction. This model may be propelled starting with survival Investigation. That information utilized for this prediction is from the website named Trulia. He Additionally suggested that it is difficult to get the actual selling time with the website time this is because the observation time is far beyond the data. He also suggests that unsupervised learning methods have more popularity when compared to the other methods. (Li and Chu; 2017) also says that survival regression method helps to predict the values with the help of using the each and very attributes related to the house and its surroundings. (Li and Chu; 2017)A late worth of effort carried out Toward to house value. The value of the house may be influence Toward Different budgetary factors. As we all know that China is one of the most populated countries around the world. Here the author tries to make a prediction to help the banks to provide the home loan for the customers. That prediction compares to Cathy house value list provided by the China. The information is gotten from Taipei lodging segment the over-proliferation after the data collection they use the machine learning algorithm neural network to predict the price the and accuracy of the prediction can find out using the RMSE (ROOT MEAN SQUARE ERROR) and the MAE(MEAN ABSOLUTE PERCENTAGE ERROR).(Li and Chu; 2017)(Willmott; 1981) (Park and Bae; 2015)During the year 2005 there seems to be a high rise in interests on the American housing markets so the America was forced into bankruptcy so it reflects US housing markets they were declined up to 30 to 60 % in the major cities it was continued for many years, after the November 2012 it started to recover because the investment becomes low so there was a demand so the author tries to research and developed a prediction model to get whether the closing price is higher or lower by using the machine learning to obtain the knowledge and to predict the future. Here he uses the KDD model knowledge discovery databases the data here used seems to be merged from the different data sets and uses the WEKA software to find them a multiple algorithms like decision tree is used to finding the relationship in the database, here park and bay uses the RIPPER, C4.5 (J48), naive Bayesian and Ada-Boost every algorithm is used under different conditions RIPPER is used for selecting the majority class and minority class, naive Bayesian is used to divide the data set into different classes by calculating the probability distribution and AdaBoost is used to improve the classification and here performs the two methods one is three way split with 10 folds and 10 folds cross validation by his results achievements RIPPER have the more prediction compared to the other. (Piazzesi and Schneider; 2009)Those foreseeing those value of the product alternately an arrangement may be altogether intricate. The cost prediction is basically utilized within impart business sector. Yet the prediction from claiming offer worth may be precise perplexing due to it dynamic clinched alongside the way. Need to be carried out a neural system model to foreseeing the stock value. This gives an association between those stocks Also benefit. In this model, the creator utilized the stock information need. The following venture will be to ascertain those relapse components based upon the shutting esteem of the stock. Straight relapse will be performed on the first information situated. Right away the duplicate of the first information situated is made What's more Fourier analysis may be connected. Following that Fourier analysis, that standardization of information will be finished. This makes an MLP with a portion neuron. Right away the neural system calculation may be actualized. This calculation gives preferred correctness done prediction Also offers great commotion tolerance. The principal hindrance is that the stake business sector information continues overhauling and the prediction turns into was troublesome. The author(Gu et al.; 2011) says that housing price involves the various economic interest it also includes both the government and the peoples so there is in need of accurate forecasting so the three researchers from key laboratory developed a new model using the genetic algorithm and the support vector machine. They have clearly mentioned the regression theorem of the support vector machine and introduced a new function called kernel with the help of Karush-Kuhn-Tuckers(KTT) conditions. here they have combined the genetic algorithm with the SVM and named it as G-SVM where the kernel functions will be in chromosomes and each will divide into three segments the author is aware of the fitness model so they have calculated the fitness value of for each chromosome so there will less percentage of over fitting model and three operations selection, crossover and mutation operation are performed and the results are obtained . there is a comparison between the grey model and GSVM and GSVM executes the results faster and more precise as suggested by the founders. The authors(Limsombunchai; 2004) try to provide a more accurate prediction on the house prices to improve efficiency to the real estate present in New Zealand he suggests that most peoples in New Zealand have their own houses the sample data is obtained from one of the trusted real estate agency so we can believe that there will not be an error in the data here he compared the hedonic price and the artificial neural network theory when conducting the hedonic price model there is hypothesis based on the previous works it seems to have a positive relationship. In the neural;l network the author uses the trained data in order to avoid the prediction errors the work strategy of neural networks is stated clearly and at last when comparing the results the author says that artificial neural network has more performance when compared to the other one. The author determines the housing prices in the turkey as by the method (Limsombunchai; 2004) the data is taken from the household survey during the year 2004 they suggest that hedonic multiple regression models are mostly used for the price prediction because they tend to fit the data into the model by observing the results there is no multicollinearity between the variables but there is heteroscedasticity due to the white state statistics suggesting that there will be potential problem in the model also some variables are dominating the significant variables in the house price predction while coming to the artificial neural networks they suggest that networks have the capacity of adapting and it is also one the flexible models in this predection feed forward network is used for prediction by comparing their results ANN (ARTIFICIAL NEURAL NETWORK) tends to have more performance when it compared with the hedonic multiple regression as suggested by the(Nghiep and Al; 2001) The authors(Selim; 2009) have compared the multiple regression analysis over the artificial neural networks by using the 60% data for the house pricing prediction several comparisons have been made in their predictive performance they have compared with the different training size and selecting the data in their size ie) the sample data size various for the performance detection. For calculating the error two different equations are used mean absolute percentage error and the absolute percentage error, here the absolute percentage divides the properties into three different stages based on the FE(Forecasting error) percentages Totally six different comparisons are made for more efficiency, here it's clear that if there is enough or sufficient data size artificial neural network can perform better or else the results will be different as said by(Willmott; 1981) The two authors (Wu and Brynjolfsson; 2009) from MIT have conducted about the prediction that how the Google searches the housing price and sales across the world suggesting that in the present world every prediction percentage point is correlated with the next year house sales. The author reveals about the correlation between them housing price and their related searches and the positive relationship between them. The data is taken from the Google search which means the search queries by using the Google trends and with the help of a national association of real-tors the data is collected for all the states present in the united states of America and found the highest number of houses sold during the year 2005 and the recession starts over 2009 by using the auto regressive (AR) model, by using it the relationship between the search queries and housing market indicators they have estimated the baseline for housing price prediction and they are well shown in the figures and suggesting that if there demand to house and there will be demand in house hold appliances.

For classification problems, there is a way to find out the accuracy percentage with the help of the confusion matrices we can find out the accuracy percentage but the regression there is only one possibility to calculate the RMSE root mean squared error here the author says about the error indices an average error of a model calculated using the mean squared error

(MSE) or the root mean squared error (RMSE) . there is a problem in using the correlation coefficient as the significance test, not an appropriate one so that we prefer the RMSE.

2.2 Machine Learning Introduction

The term Machine Learning was coined by Arthur Samuel in 1959, an American pioneer in the field of computer gaming and artificial intelligence and stated that "it gives computers the ability to learn without being explicitly programmed".
And in 1997, Tom Mitchell gave a "well-posed" mathematical and relational definition that "A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E.

Machine Learning is a latest buzzword floating around. It deserves to, as it is one of the most interesting subfield of Computer Science. So what does Machine Learning really mean?

Let's try to understand Machine Learning in layman terms. Consider you are trying to toss a paper to a dustbin.

After first attempt, you realize that you have put too much force in it. After second attempt, you realize you are closer to target but you need to increase your throw angle. What is happening here is basically after every throw we are learning something and improving the end result. We are programmed to learn from our experience.

This implies that the tasks in which machine learning is concerned offers a fundamentally operational definition rather than defining the field in cognitive terms. This follows Alan Turing's proposal in his paper "Computing Machinery and Intelligence", in which the question "Can machines think?" is replaced with the question "Can machines do what we (as thinking entities) can do?"
Within the field of data analytics, machine learning is used to devise complex models and algorithms that lend themselves to prediction; in commercial use, this is known as predictive analytics. These analytical models allow researchers, data scientists, engineers, and analysts to "produce reliable, repeatable decisions and results" and uncover "hidden insights" through learning from historical relationships and trends in the data set(input).

Suppose that you decide to check out that offer for a vacation . You browse through the travel agency website and search for a hotel. When you look at a specific hotel, just below the hotel description there is a section titled "You might also like these hotels". This is a common use case of Machine Learning called "Recommendation Engine". Again, many data points were used to train a model in order to predict what will be the best hotels to show you under that section, based on a lot of information they already know about you.

So if you want your program to predict, for example, traffic patterns at a busy intersection (task T), you can run it through a machine learning algorithm with data about past traffic patterns (experience E) and, if it has successfully "learned", it will then do better at predicting future traffic patterns (performance measure P).
The highly complex nature of many real-world problems, though, often means that

inventing specialized algorithms that will solve them perfectly every time is impractical, if not impossible. Examples of machine learning problems include, “Is this cancer?”, “Which of these people are good friends with each other?”, “Will this person like this movie?” such problems are excellent targets for Machine Learning, and in fact machine learning has been applied such problems with great success.

2.2.1 Classification of Machine Learning

Machine learning implementations are classified into three major categories, depending on the nature of the learning “signal” or “response” available to a learning system which are as follows:-

1. **Supervised learning :** When an algorithm learns from example data and associated target responses that can consist of numeric values or string labels, such as classes or tags, in order to later predict the correct response when posed with new examples comes under the category of Supervised learning. This approach is indeed similar to human learning under the supervision of a teacher. The teacher provides good examples for the student to memorize, and the student then derives general rules from these specific examples.

2. **Unsupervised learning :**Whereas when an algorithm learns from plain examples without any associated response, leaving to the algorithm to determine the data patterns on its own. This type of algorithm tends to restructure the data into something else, such as new features that may represent a class or a new series of un-correlated values. They are quite useful in providing humans with insights into the meaning of data and new useful inputs to supervised machine learning algorithms.
   As a kind of learning, it resembles the methods humans use to figure out that certain objects or events are from the same class, such as by observing the degree of similarity between objects. Some recommendation systems that you find on the web in the form of marketing automation are based on this type of learning.

3. **Reinforcement learning :** When you present the algorithm with examples that lack labels, as in unsupervised learning. However, you can accompany an example with positive or negative feedback according to the solution the algorithm proposes comes under the category of Reinforcement learning, which is connected to applications for which the algorithm must make decisions (so the product is prescriptive, not just descriptive, as in unsupervised learning), and the decisions bear consequences. In the human world, it is just like learning by trial and error.
   Errors help you learn because they have a penalty added (cost, loss of time, regret, pain, and so on), teaching you that a certain course of action is less likely to succeed than others. An interesting example of reinforcement learning occurs when computers learn to play video games by themselves.
   In this case, an application presents the algorithm with examples of specific situations, such as having the gamer stuck in a maze while avoiding an enemy. The application lets the algorithm know the outcome of actions it takes, and learning occurs while trying to avoid what it discovers to be dangerous and to

pursue survival. You can have a look at how the company Google DeepMind has created a reinforcement learning program that plays old Atari's video games. When watching the video, notice how the program is initially clumsy and unskilled but steadily improves with training until it becomes a champion.

4. **Semi-supervised learning :** where an incomplete training signal is given: a training set with some (often many) of the target outputs missing. There is a special case of this principle known as Transduction where the entire set of problem instances is known at learning time, except that part of the targets are missing.

2.2.2 Categorizing on the basis of required Output

Another categorization of machine learning tasks arises when one considers the desired output of a machine-learned system:

1. **Classification :** When inputs are divided into two or more classes, and the learner must produce a model that assigns unseen inputs to one or more (multi-label classification) of these classes. This is typically tackled in a supervised way. Spam filtering is an example of classification, where the inputs are email (or other) messages and the classes are "spam" and "not spam".

2. **Regression :** Which is also a supervised problem, A case when the outputs are continuous rather than discrete.

3. **Clustering :** When a set of inputs is to be divided into groups. Unlike in classification, the groups are not known beforehand, making this typically an unsupervised task.

Machine Learning comes into the picture when problems cannot be solved by means of typical approaches.

2.3 Machine Learning Algorithms

2.3.1 Random Forest Regressor

Random Forest is a trademark term for an ensemble of decision trees. In Random Forest, we've collection of decision trees (so known as "Forest"). To classify a new object based on attributes, each tree gives a classification and we say the tree "votes" for that class. The forest chooses the classification having the most votes (over all the trees in the forest).

Each tree is planted & grown as follows:

1. If the number of cases in the training set is N, then sample of N cases is taken at random but *with replacement*. This sample will be the training set for growing the tree.
2. If there are M input variables, a number m<<M is specified such that at each node, m variables are selected at random out of the M and the best split on these m is used to split the node. The value of m is held constant during the forest growing.
3. Each tree is grown to the largest extent possible. There is no pruning.

2.3.2 Gradient Boosting Regressor

Let’s start by understanding Boosting! Boosting is a method of converting weak learners into strong learners. In boosting, each new tree is a fit on a modified version of the original data set. The gradient boosting algorithm (gbm) can be most easily explained by first introducing the AdaBoost Algorithm.The AdaBoost Algorithm begins by training a decision tree in which each observation is assigned an equal weight. After evaluating the first tree, we increase the weights of those observations that are difficult to classify and lower the weights for those that are easy to classify. The second tree is therefore grown on this weighted data. Here, the idea is to improve upon the predictions of the first tree. Our new model is therefore *Tree 1 + Tree 2*. We then compute the classification error from this new 2-tree ensemble model and grow a third tree to predict the revised residuals. We repeat this process for a specified number of iterations. Subsequent trees help us to classify observations that are not well classified by the previous trees. Predictions of the final ensemble model is therefore the weighted sum of the predictions made by the previous tree models.

Gradient Boosting trains many models in a gradual, additive and sequential manner. The major difference between AdaBoost and Gradient Boosting Algorithm is how the two algorithms identify the shortcomings of weak learners (eg. decision trees). While the AdaBoost model identifies the shortcomings by using high weight data points, gradient boosting performs the same by using gradients in the loss function (*y=ax+b+e , e needs a special mention as it is the error term)*. The loss function is a measure indicating how good are model’s coefficients are at fitting the underlying data. A logical understanding of loss function would depend on what we are trying to optimise. For example, if we are trying to predict the sales prices by using a regression, then the loss function would be based off the error between true and predicted house prices. Similarly, if our goal is to classify credit defaults, then the loss function would be a measure of how good our predictive model is at classifying bad loans. One of the biggest motivations of using gradient boosting is that it allows one to optimise a user specified cost function, instead of a loss function that usually offers less control and does not essentially correspond with real world applications.

2.3.3 Decision Tree regressor

Decision Tree is a decision-making tool that uses a flowchart-like tree structure or is a model of decisions and all of their possible results, including outcomes, input costs and utility. Decision-tree algorithm falls under the category of supervised learning algorithms. It works for both continuous as well as categorical output variables.

The branches/edges represent the result of the node and the nodes have either:

1. Conditions [Decision Nodes]
2. Result [End Nodes]

The branches/edges represent the truth/falsity of the statement and takes makes a decision based on that in the example below which shows a decision tree that evaluates the smallest of three numbers:

Decision tree regression observes features of an object and trains a model in the structure of a tree to predict data in the future to produce meaningful continuous output. Continuous output means that the output/result is not discrete, i.e., it is not represented just by a discrete, known set of numbers or values.

Discrete output example: A weather prediction model that predicts whether or not there'll be rain in a particular day.
Continuous output example: A profit prediction model that states the probable profit that can be generated from the sale of a product.
Here, continuous values are predicted with the help of a decision tree regression model.

CHAPTER 3: IMPLEMENTATION

3.1 Programming language used

3.1.1 Python

Machine Learning is simply making a computer perform a task without explicitly programming it. In today's world every system that does well has a machine learning algorithm at its heart. Take for example Google Search engine, Amazon Product recommendations, LinkedIn, Facebook etc, all these systems have machine learning algorithms embedded in their systems in one form or the other. They are efficiently utilising data collected from various channels which helps them get a bigger picture of what they are doing and what they should do.

Python is a widely used high-level programming language for general-purpose programming. Apart from being open source programming language, python is a great object-oriented, interpreted, and interactive programming language. Python combines remarkable power with very clear syntax. It has modules, classes, exceptions, very high level dynamic data types, and dynamic typing. There are interfaces to many system calls and libraries, as well as to various windowing systems. New built-in modules are easily written in C or C++ (or other languages, depending on the chosen implementation). Python is also usable as an extension language for applications written in other languages that need easy-to-use scripting or automation interfaces.

Python is widely considered as the preferred language for teaching and learning Ml (Machine Learning). Few simple reasons are:

>It's simple to learn. As compared to c, c++ and Java the syntax is simpler and Python also consists of a lot of code libraries for ease of use.

>Though it is slower than some of the other languages, the data handling capacity is great.

>Open Source! – Python along with R is gaining momentum and popularity in the Analytics domain since both of these languages are open source.

>Capability of interacting with almost all the third party languages and platforms.

3.1.2 Python libraries used

3.1.2.1 Sci-kit Learn or Sklearn for Machine Learning

Scikit-learn (formerly **scikits.learn** and also known as **sklearn**) is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, *k*-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

Scikit-learn is largely written in Python, and uses numpy extensively for high-performance linear algebra and array operations. Furthermore, some core algorithms are written in Cython to improve performance. Support vector machines are implemented by a Cython wrapper around LIBSVM; logistic regression and linear support vector machines by a similar wrapper around LIBLINEAR. In such cases, extending these methods with Python may not be possible.

Scikit-learn integrates well with many other Python libraries, such as matplotlib and plotly for plotting, numpy for array vectorization, pandas dataframes, scipy, and many more.

3.1.2.2 Tkinter for GUI

Python has a lot of GUI frameworks, but Tkinter is the only framework that's built into the Python standard library. Tkinter has several strengths. It's **cross-platform**, so the same code works on Windows, macOS, and Linux. Visual elements are rendered using native operating system elements, so applications built with Tkinter look like they belong on the platform where they're run.

Although Tkinter is considered the de-facto Python GUI framework, it's not without criticism. One notable criticism is that GUIs built with Tkinter look outdated. If you want a shiny, modern interface, then Tkinter may not be what you're looking for.

However, Tkinter is lightweight and relatively painless to use compared to other frameworks. This makes it a compelling choice for building GUI applications in Python, especially for applications where a modern sheen is unnecessary, and the top priority is to build something that's functional and cross-platform quickly.

3.2 Importing libraries and modules necessary for this project

Python code in one module gains access to the code in another module by the process of importing it. The import statement is the most common way of invoking the import machinery, but it is not the only way. Functions such as importlib.import_module() and built-in __import__() can also be used to invoke the import machinery.

The import statement combines two operations; it searches for the named module, then it binds the results of that search to a name in the local scope. The search operation of the import statement is defined as a call to the __import__() function, with the appropriate arguments. The return value of __import__() is used to perform the name binding operation of the import statement. See the import statement for the exact details of that name binding operation.

```
#importing libraries

import numpy as np

import pandas as pd

from tkinter import *

from sklearn.model_selection import StratifiedShuffleSplit

from sklearn.pipeline import Pipeline

from sklearn.preprocessing import StandardScaler

from sklearn.impute import SimpleImputer
```

3.3 Load data from CSV files

CSV (comma-separated value) files are a common file format for transferring and storing data. The ability to read, manipulate, and write data to and from CSV files using Python is a key skill to master for any data scientist or business analysis. In this post, we'll go over what CSV files are, how to read CSV files into Pandas DataFrames, and how to write DataFrames back to CSV files post analysis.

Pandas is the most popular data manipulation package in Python, and DataFrames are the Pandas data type for storing tabular 2D data.

The basic process of loading data from a CSV file into a Pandas DataFrame (with all going well) is achieved using the "read_csv" function in Pandas:

```
import pandas as pd
# Read data from file 'filename.csv'
# (in the same directory that your python process is based)
# Control delimiters, rows, column names with read_csv (see later)


housing = pd.read_csv('kc_house_data.csv')
```

3.4 Train and test splitting

For this section we will take the housing dataset and split the data into training and testing subsets. Typically, the data is also shuffled into a random order when creating the training and testing subsets to remove any bias in the ordering of the dataset.

For this will use StratifiedShuffleSplit.

StratifiedShuffleSplit cross-validator provides train/test indices to split data in train/test sets.

This cross-validation object is a merge of StratifiedKFold and ShuffleSplit, which returns stratified randomized folds. The folds are made by preserving the percentage of samples for each class.

```
# splitting of data into train and test set on the basis of waterfront variable

split = StratifiedShuffleSplit(n_splits=1, test_size = 0.2 , random_state=40)
for train_index,test_index in split.split(housing,housing['waterfront']):
    test_set = housing.loc[test_index]
    train_set = housing.loc[train_index]


#splitting of train data on dependent and independent variables

housing = train_set.copy()

housing = train_set.drop("price",axis=1)

housing_labels = train_set['price'].copy()
```

3.5 Pipelining

A machine learning pipeline is used to help automate machine learning workflows. They operate by enabling a sequence of data to be transformed and correlated together in a model that can be tested and evaluated to achieve an outcome, whether positive or negative.

Machine learning (ML) pipelines consist of several steps to train a model. Machine learning pipelines are iterative as every step is repeated to continuously improve the accuracy of the model and achieve a successful algorithm. To build better machine learning models, and get the most value from them, accessible, scalable and durable storage solutions are imperative, paving the way for on-premises object storage.

```
from sklearn.pipeline import Pipeline

from sklearn.preprocessing import StandardScaler

from sklearn.impute import SimpleImputer

#performing pipeline to implement imputer and standardization on data

my_pipeline = Pipeline([

  # Imputation transformer for completing missing values.

   ('imputer', SimpleImputer(strategy="median")),

  # The idea behind StandardScaler is that it will transform your data such that its distribution will
have a mean value 0 and standard deviation of 1.

  ('std_scaler', StandardScaler()),

])

# Transforming data before sending it to pipeline.

housing_num_tr =
my_pipeline.fit_transform(housing.drop(['id','date','yr_built','yr_renovated','zipcode'],axis=1
))

# Preparing data for pipeline.

prepared_data =
my_pipeline.transform(housing_test.drop(['id','date','yr_built','yr_renovated','zipcode'],axis=
1))
```

3.6 Using Regressors

3.6.1 Gradient boosting regressor

Gradient Tree Boosting or Gradient Boosted Decision Trees (GBDT) is a generalization of boosting to arbitrary differentiable loss functions. GBDT is an accurate and effective off-the-shelf procedure that can be used for both regression and classification problems in a variety of areas including Web search ranking and ecology.

The module **sklearn.ensemble** provides methods for both classification and regression via gradient boosted decision trees

GB builds an additive model in a forward stage-wise fashion; it allows for the optimization of arbitrary differentiable loss functions. In each stage a regression tree is fit on the negative gradient of the given loss function.

```
#gradient boosting regressor
def gbr():
    from sklearn import ensemble
    clf1 = ensemble.GradientBoostingRegressor(n_estimators = 400, max_depth = 5,
min_samples_split = 2,learning_rate = 0.1, loss = 'ls')
    t = pd.DataFrame(columns =
['bedrooms','bathrooms','sqft_living','sqft_lot','floors','waterfront','view','condition','grade','
sqft_above','sqft_basement','lat','long','sqft_living15','sqft_lot15'])
    t1 =
t.append(pd.Series([Bedroom.get(),Bathroom.get(),Sqft_living.get(),Sqft_lot.get(),Floors.get(
),Waterfront.get(),View.get(),Condition.get(),Grade.get(),Sqft_above.get(),Sqft_basement.get
(),Lat.get(),Long.get(),Sqft_living15.get(),Sqft_lot15.get()],index=t.columns),ignore_index=Tr
ue)
    t2 = my_pipeline.transform(t1)
    clf1.fit(housing_num_tr, housing_labels)
    s2 = clf1.score(prepared_data,housing_test_labels)
    print(s2)
    pred1 = clf1.predict(t2)
    p1.insert(0,str(pred1))
```

3.6.2 Decision tree regressor

The decision trees is used to fit a sine curve with addition noisy observation. As a result, it learns local linear regressions approximating the sine curve.

We can see that if the maximum depth of the tree (controlled by the `max_depth` parameter) is set too high, the decision trees learn too fine details of the training data and learn from the noise, i.e. they overfit.

```
#decision tree regressor
def dtr():
    from sklearn.tree import DecisionTreeRegressor

    clf3 = DecisionTreeRegressor()

    t = pd.DataFrame(columns = ['bedrooms','bathrooms','sqft_living','sqft_lot','floors','waterfront','view','condition','grade','sqft_above','sqft_basement','lat','long','sqft_living15','sqft_lot15'])
    t1 = t.append(pd.Series([Bedroom.get(),Bathroom.get(),Sqft_living.get(),Sqft_lot.get(),Floors.get(),Waterfront.get(),View.get(),Condition.get(),Grade.get(),Sqft_above.get(),Sqft_basement.get(),Lat.get(),Long.get(),Sqft_living15.get(),Sqft_lot15.get()],index=t.columns),ignore_index=True)
    t2 = my_pipeline.transform(t1)
    clf3.fit(housing_num_tr, housing_labels)
    s3 = clf3.score(prepared_data,housing_test_labels)
    print(s3)
    pred3 = clf3.predict(t2)
    p3.insert(0,str(pred3))
```

3.6.3 Random forest regressor

A random forest is a meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is controlled with the max_samples parameter if bootstrap=True (default), otherwise the whole dataset is used to build each tree.

```
#random forest regressor

def rfr():

    from sklearn.ensemble import RandomForestRegressor



    clf4 = RandomForestRegressor()



    t = pd.DataFrame(columns = 
['bedrooms','bathrooms','sqft_living','sqft_lot','floors','waterfront','view','condition','grade','
sqft_above','sqft_basement','lat','long','sqft_living15','sqft_lot15'])

    t1 = 
t.append(pd.Series([Bedroom.get(),Bathroom.get(),Sqft_living.get(),Sqft_lot.get(),Floors.get(
),Waterfront.get(),View.get(),Condition.get(),Grade.get(),Sqft_above.get(),Sqft_basement.get
(),Lat.get(),Long.get(),Sqft_living15.get(),Sqft_lot15.get()],index=t.columns),ignore_index=Tr
ue)

    t2 = my_pipeline.transform(t1)

    clf4.fit(housing_num_tr, housing_labels)

    s4 = clf4.score(prepared_data,housing_test_labels)

    print(s4)

    pred4 = clf4.predict(t2)

    p4.insert(0,str(pred4))
```

3.7 Graphical User Interface

A GUI (graphical user interface) is a system of interactive visual components for computer software. A GUI displays objects that convey information, and represent actions that can be taken by the user. The objects change color, size, or visibility when the user interacts with them.

Unlike a command-line or CUI, GUI are much easier to learn and use because commands do not need to be memorized. Additionally, users do not need to know any programming languages. Because of their ease of use and more modern appearance, GUI have come to dominate today's market.

Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

Creating a GUI application using Tkinter is an easy task. All you need to do is perform the following steps −

- Import the *Tkinter* module.
- Create the GUI application main window.
- Add one or more of the above-mentioned widgets to the GUI application.
- Enter the main event loop to take action against each event triggered by the user.

```
import Tkinter

top = Tkinter.Tk()

# Code to add widgets will go here...

top.mainloop()
```

Tkinter Widgets

Tkinter provides various controls, such as buttons, labels and text boxes used in a GUI application. These controls are commonly called widgets.

→ Declaration of variables with prefixed set value

```
 # entry variables

Bedroom = IntVar()

Bedroom.set(1)

Bathroom = IntVar()

Bathroom.set(1)

Floors = IntVar()

Floors.set(1)
```

```
Waterfront = IntVar()

Waterfront.set(0)

View = IntVar()

View.set(0)

Grade = IntVar()

Grade.set(3)

Sqft_living = IntVar()

Sqft_living.set(370)

Sqft_lot = IntVar()

Sqft_lot.set(520)

Sqft_above = IntVar()

Sqft_above.set(370)

Sqft_basement = IntVar()

Sqft_basement.set(0)

Condition = IntVar()

Condition.set(1)

Lat = IntVar()

Lat.set(47.15)

Long = IntVar()

Long.set(-122.51)

Sqft_living15  = IntVar()

Sqft_living15.set(399)

Sqft_lot15 = IntVar()

Sqft_lot15.set(651)
```

All Tkinter widgets have access to specific geometry management methods, which have the purpose of organizing widgets throughout the parent widget area. Tkinter exposes the following geometry manager classes: pack, grid, and place.

- The *pack()* Method – This geometry manager organizes widgets in blocks before placing them in the parent widget.

- The *grid()* Method – This geometry manager organizes widgets in a table-like structure in the parent widget.
- The *place()* Method – This geometry manager organizes widgets by placing them in a specific position in the parent widget.

We use the grid method to organize widgets in a table like structure.

Label

The Label widget is used to provide a single-line caption for other widgets. It can also contain images.

It is used to define the user about the variable and entry he is going to fill and values from where to where needs to be fill (min to max),values which are allowed.

```
# labels
NameLb = Label(root, text="Name", fg="yellow", bg="black")
NameLb.grid(row=6, column=0, pady=15, sticky=W)



S1Lb = Label(root, text="Bedroom from 1 to 33", fg="yellow", bg="black")
S1Lb.grid(row=7, column=0, pady=10, sticky=W)

S2Lb = Label(root, text="Bathroom from 1 to 8", fg="yellow", bg="black")
S2Lb.grid(row=8, column=0, pady=10, sticky=W)

S3Lb = Label(root, text="Sqft_living from 370 to 13540", fg="yellow", bg="black")
S3Lb.grid(row=9, column=0, pady=10, sticky=W)

S4Lb = Label(root, text="Sqft_lot from 520 to 1651359", fg="yellow", bg="black")
S4Lb.grid(row=10, column=0, pady=10, sticky=W)

S5Lb = Label(root, text="Floors from 1 to 4", fg="yellow", bg="black")
S5Lb.grid(row=11, column=0, pady=10, sticky=W)
```

```
S6Lb = Label(root, text="Waterfront from 0 to 1", fg="yellow", bg="black")

S6Lb.grid(row=12, column=0, pady=10, sticky=W)


S7Lb = Label(root, text="View from 0 to 4", fg="yellow", bg="black")

S7Lb.grid(row=13, column=0, pady=10, sticky=W)


S8Lb = Label(root, text="Condition from 1 to 5", fg="yellow", bg="black")

S8Lb.grid(row=14, column=0, pady=10, sticky=W)


S9Lb = Label(root, text="Grade from 3 to 13", fg="yellow", bg="black")

S9Lb.grid(row=15, column=0, pady=10, sticky=W)


S10Lb = Label(root, text="Sqft_above from 370 to 9410", fg="yellow", bg="black")

S10Lb.grid(row=16, column=0, pady=10, sticky=W)


S11Lb = Label(root, text="Sqft_basement from 0 to 4820", fg="yellow", bg="black")

S11Lb.grid(row=17, column=0, pady=10, sticky=W)


S12Lb = Label(root, text="Lat from 47.15 to 47.77", fg="yellow", bg="black")

S12Lb.grid(row=18, column=0, pady=10, sticky=W)


S13Lb = Label(root, text="Long from -122.51 to -121.31", fg="yellow", bg="black")

S13Lb.grid(row=19, column=0, pady=10, sticky=W)


S14Lb = Label(root, text="Sqft_living15 from 399 to 6210", fg="yellow", bg="black")

S14Lb.grid(row=20, column=0, pady=10, sticky=W)


S15Lb = Label(root, text="Sqft_lot15 from 651 to 872100", fg="yellow", bg="black")

S15Lb.grid(row=21, column=0, pady=10, sticky=W)
```

Option for user to select the appropriate model to predict the price of the house

1.Gradient boosting

2.Decision tree

3.Random forest

```
l1Lb = Label(root, text="GradientBoostingRegressor", fg="white", bg="red")
l1Lb.grid(row=24, column=0, pady=10,sticky=W)

l3Lb = Label(root, text="DecisionTreeRegressor", fg="white", bg="red")
l3Lb.grid(row=25, column=0, pady=10,sticky=W)

l4Lb = Label(root, text="RandomForestRegressor", fg="white", bg="red")
l4Lb.grid(row=26, column=0, pady=10,sticky=W)
```

Entry

The Entry widget is used to display a single-line text field for accepting values from a user.

Syntax: **entry_widget = tk.Entry(widget, option=placeholder)** where `widget` is the parameter for the parent window/frame while `option` is a placeholder that can have various values like border-width, background color, width & height of button etc.

Spinbox

The Spinbox widget is a variant of the standard Tkinter Entry widget, which can be used to select from a fixed number of values.

The `tk.Spinbox` widget is a `tk.Entry` enhanced with increment and decrement arrows. It can be used for numbers or lists of string values. This widget is a subclass of `Entry`.

```
# entries
NameEn = Entry(root, textvariable=Name)
NameEn.grid(row=6, column=1)
```

```
S1En = Spinbox(root, textvariable=Bedroom,from_=1,to=33)

S1En.grid(row=7, column=1)


S2En = Spinbox(root, textvariable=Bathroom,from_=1,to=8)

S2En.grid(row=8, column=1)


S3En = Spinbox(root, textvariable=Sqft_living,from_=370,to=13540)

S3En.grid(row=9, column=1)


S4En = Spinbox(root, textvariable=Sqft_lot,from_=520,to=1651359)

S4En.grid(row=10, column=1)


S5En = Spinbox(root, textvariable=Floors,from_=1,to=4)

S5En.grid(row=11, column=1)


S6En = Spinbox(root, textvariable=Waterfront,from_=0,to=1)

S6En.grid(row=12, column=1)


S7En = Spinbox(root, textvariable=View,from_=0,to=4)

S7En.grid(row=13, column=1)


S8En = Spinbox(root, textvariable=Condition,from_=1,to=5)

S8En.grid(row=14, column=1)


S9En = Spinbox(root, textvariable=Grade,from_=3,to=13)

S9En.grid(row=15, column=1)


S10En = Spinbox(root, textvariable=Sqft_above,from_=370,to=9410)

S10En.grid(row=16, column=1)
```

```
S11En = Spinbox(root, textvariable=Sqft_basement,from_=0,to=4820)

S11En.grid(row=17, column=1)


S12En = Spinbox(root, textvariable=Lat,from_=47.15,to=47.77)

S12En.grid(row=18, column=1)


S13En = Spinbox(root, textvariable=Long,from_=-122.51,to=-121.31)

S13En.grid(row=19, column=1)


S14En = Spinbox(root, textvariable=Sqft_living15,from_=399,to=6210)

S14En.grid(row=20, column=1)


S15En = Spinbox(root, textvariable=Sqft_lot15,from_=651,to=872100)

S15En.grid(row=21, column=1)
```

Button

The Button widget is used to display buttons in your application.

Syntax: **button_widget = tk.Button(widget, option=placeholder)** where `widget` is the argument for the parent window/frame while `option` is a placeholder that can have various values like foreground & background color, font, command (for function call), image, height, and width of button.

Buttons call their respective function or model and brings the result along with them.

```
b1 = Button(root, text="Price", command=gbr,bg="green",fg="yellow")

b1.grid(row=24, column=3,padx=10)


b3 = Button(root, text="Price", command=dtr,bg="green",fg="yellow")

b3.grid(row=25, column=3,padx=10)
```

```
b4 = Button(root, text="Price", command=rfr,bg="green",fg="yellow")

b4.grid(row=26, column=3,padx=10)
```

Last are the entries where the predicted price of the corresponding model is shown and the accuracy of the model is shown at the console or terminal of the corresponding model.

```
#predicted price

p1 = Entry(root)

p1.grid(row=24, column=1, padx=10)


p3 = Entry(root)

p3.grid(row=25, column=1, padx=10)


p4 = Entry(root)

p4.grid(row=26, column=1, padx=10)
```

References

https://www.tutorialspoint.com/python/python_gui_programming.html

https://www.geeksforgeeks.org/data-preprocessing-machine-learning-python/

https://www.kaggle.com/harlfoxem/housesalesprediction

https://en.wikipedia.org/wiki/Gradient_boosting

https://docs.python.org/3/library/tk.html