PROJECT REPORT

HOUSE PRICE PREDICTION

Anandu R (CUTM00802)
Boney Simon (CUTM00790)
Jashith V.P (CUTM01124)
Philip George(CUTM00824)

Guided By - Manish Jain

Contents

Executive Summary

The latest research on prediction markets to further their utilization by economic forecasters. Thus, there is a need to predict the efficient house pricing for real estate customers with respect to their budgets and priorities. This project efficiently analyses previous market trends and price ranges, to predict future prices. This topic brings together the latest research on prediction markets to further their utilization by economic forecasters. It provides a description of prediction markets, and also the current markets which are useful in understanding the market which helps in making useful predictions.

This project uses gradient boosted regression algorithm to predict prices by analyzing current house prices, thereby forecasting the future prices according to the user's requirements. With a large amount of unstructured resources and documents, the Real estate industry has become a highly competitive business. The data mining process in such an industry provides an advantage to the developers by processing those data, forecasting future trends, and thus assisting them to make favorable knowledge-driven decisions.

1 Introduction

1.1 Aim

The aim of the project is to provide estimates of the price and popularity of a residence based on user parameters.

1.2 Technologies

The project is divided into four stages namely, data cleaning, Exploratory data analysis, modelling, application development, and deployment. We use R programming for the first three steps where in the following libraries are used:

Data Cleaning

- **dplyr** - It provides tools that enables data handling, data mining, allows us to pipeline various operations to streamline workflow and derive insights from the data. dplyr is a grammar of data manipulation, providing a consistent set of verbs that allow most common data manipulation challenges.
- **reshape2** - In order to reshape the data to aggregate/summarise the values or to flatten the information to drill down further into the data. This package that allows us to easily transform our data into whatever structure we may need. Many of us are used to seeing our data structured so that corresponds to a single participant and each column corresponds to a variable. This type of data structure is known as **wide** format. However, many of the packages in R require that we stretch our data so that a single participant may occupy multiple rows. This type of data structure is known as **long** format
- **Tidyverse** - The package is intended to make statisticians and data scientists more productive by guiding them through workflows that facilitate communication, and result in reproducible work products. Fundamentally, the tidyverse is about the connections between the tools that make the workflow possible.
- **ggplot2** - It is a plotting package that makes it simple to create complex plots from data in a data frame. It provides a more programmatic interface for specifying what variables to plot, how they are displayed, and general visual properties.
- **ggmap** - The ggmap package extends the ever-amazing ggplot core features to allow for spatial and geographic visualization.

- **lubridate** - Lubridate is an R package that makes it easier to work with dates and times. Lubridate's parse functions handle a wide variety of formats and separators, which simplifies the parsing process.

Exploratory data analysis

- **dplyr** - *Same as in data cleaning*
- **purrr** - purrr enhances R's functional programming (FP) toolkit by providing a complete and consistent set of tools for working with functions and vectors. If you've never heard of FP before, the best place to start is the family of map() functions which allow you to replace many for loops with code that is both more succinct and easier to read.
- **tidyr** - To tidy messy data, you first identify the variables in the dataset, then use the tools provided by tidyr to move them into columns. tidyr provides three main functions for tidying messy data: gather(), separate() and spread().
- **corrplot** - The corrplot package is a graphical display of a correlation matrix, confidence interval. It also contains some algorithms to do matrix reordering. In addition, corrplot is good at details, including choosing color, text labels, color labels, layout, etc.
- **ggplot2** - *Same as in data cleaning*

Predictive modelling

- **caret** - The caret package (short for Classification And REgression Training) contains functions to streamline the model training process for complex regression and classification problems.caret has several functions that attempt to streamline the model building and evaluation process, as well as feature selection and other techniques.
- **xgboost** - It supports various objective functions, including regression, classification and ranking. The package is made to be extendible, so that users are also allowed to define their own objective functions easily.
- **data.table** - Data.table is an extension of data.frame package in R. It is widely used for fast aggregation of large datasets, low latency add/update/remove of columns, quicker ordered joins, and a fast file reader.
- **dplyr** - *Same as in data cleaning*

- **tidyverse** - *Same as in data cleaning*

Application development

The application is made entirely on python using dash framework and rendered in HTML. It uses plotly's express graph generation packages, as well as gradient boosted linear regression model to predict the price of houses based on the user inputs. The application is formatted and rendered in HTML using the bootstrap stylesheet which allows for creating clean and responsive designs.

- **scipy** - SciPy is a scientific computation library, It provides more utility functions for optimization, stats and signal processing.
- **dash** - Dash is a productive Python framework for building web analytic applications. Written on top of Flask, Plotly.js, and React.js, Dash is ideal for building data visualization apps with highly custom user interfaces in pure Python. It's particularly suited for anyone who works with data in Python.
- **plotly** - high-level interface for data visualization
- **urllib** - Urllib module is the URL handling module for python. It is used to fetch URLs (Uniform Resource Locators). It uses the urlopen function and is able to fetch URLs using a variety of different protocols.
- **json** - To read, encode and decode JSON data
- **datetime, time** - The datetime module supplies classes for manipulating dates and times. The time module allows to vary the program execution timing to control the speed of execution of steps
- **re** - A regular expression (or RE) specifies a set of strings that matches it; the functions in this module let you check if a particular string matches a given regular expression

Deployment

- **github** - GitHub is a Git repository hosting service. GitHub provides a Web-based graphical interface. It also provides access control and several collaboration features, such as a wikis and basic task management tools for every project.
- **heroku** - Heroku is a container-based cloud Platform as a Service (PaaS). Developers use Heroku to deploy, manage, and scale modern apps. Our platform is elegant, flexible, and easy to use, offering developers the simplest path to getting

their apps to market.

1.3 Hardware Architecture

At the deployment stage of the workflow we're met with the need to find a suitable cloud web hosting platform, which runs containerized applications on compute instances knows as dynos on Heroku

- **dynos** - The Heroku Platform uses the container model to run and scale all Heroku apps. The containers used at Heroku are called "dynos." Dynos are isolated, virtualized Linux containers that are designed to execute code based on a user-specified command. An app deployed on Heroku can scale to any specified number of dynos based on its resource demands. Heroku's container management capabilities provide you with an easy way to scale and manage the number, size, and type of dynos your app may need at any given time.

2 System

2.1 Requirements

The requirements can be described from two perspectives.One from the server end and the other from the user end.

- The server end requirements are server-specifics that comprise of elements to run the machine learning model and predict the price for the user inputs.
- The user end requirements involve components that allow him/her to make use of the House price prediction web application for his/her need.

2.1.1 Functional requirements

- **server end** - The web application was deployed in heroku server. The server makes use of a Python instance, and per month 1000 active hours is available for the current account. The .sav file and the ui source code was uploaded into heroku. This was then used for prediction using the user input from the ui.

  Even though the deployment was done in Python, all the stages in the model development comprising Exploratory Data Analysis, Feature Selection, Model Testing and development was done using R programming.

  So after the model was saved in .rda format, later the coefficients were extracted and stored in .sav format.
- **user end** - The user do not require any special functionalities except basic requirements from his/her end to run the model and get the predicted value as the whole thing is hosted in the cloud.

2.1.2 User requirements

The users are expected to use the application primarily to find the price of house given various parameters such as district, building type, number of living rooms, bathrooms and so on, based on the user's requirements he/she can obtain the desired results.

Furthermore, the user has an interactive dashboard which they can use to find the best district to plan their housing in, based on KPIs which provide information such as which is the most costliest district to live in, where can we find houses that's on average having a large square area and so on.

The users can also obtain details regarding the distribution of various parameters based on their inputs which triggers an active filtration of the data to display the results as needed.

2.1.3 Environmental requirements

The user requires either a mobile device or Desktop Computer/ Laptop Computer of any configuration with proper internet connectivity. The app is both mobile and computer compatible. The user requires the below or the latest browser versions of the following particulars:

- IE11 on Windows 10
- IE10 on Windows 8
- Microsoft Edge 15 on Windows 10
- Firefox 54 on Windows 10
- Chrome 59 on Windows 10 and OSX 10.10
- Opera 46 on Windows 10
- Safari 10.1 on OSX 10.10

Also the ram requirements is 4 GB and a minimum internet bandwidth of 4Mbps is also required.

2.2 Design and Architecture

Data

The raw data set is downloaded from kaggle, and is of housing price of Beijing from 2011 to 2017. The data set can be downloaded from here.It includes URL, ID, Lng, Lat, CommunityID, TradeTime, DOM(days on market), Followers, Total price, Price, Square, Living Room, number of Drawing room, Kitchen and Bathroom, Building Type, Construction time. renovation condition, building structure, Ladder ratio(which is the proportion between number of residents on the same floor and number of elevator of ladder. It describes how many ladders a resident have on average), elevator, Property rights for five years (It's related to China restricted purchase of houses policy), Subway, District, Community average price.

Most data is traded in 2011-2017, some of them is traded in Jan,2018, and some is even earlier(2010,2009)

- **url** :- the url which fetches the data.
- **id** :- the id of transaction.
- **Lng** :- and Lat coordinates, using the BD09 protocol.
- **Cid** :- community id.
- **tradeTime** :- the time of transaction.
- **DOM** :- active days on market.
- **followers** :- the number of people follow the transaction.
- **totalPrice** :- the total price
- **price** :- the average price by square
- **square** :- the square of house
- **livingRoom** :- the number of living room
- **drawingRoom** :- the number of drawing room
- **kitchen** :- the number of kitchen
- **bathroom** :- the number of bathroom
- **floor** :- the height of the house.
- **builingType** :- including tower 1, bungalow 2 combination of plate and tower 3, plate 4.
- **constructionTime** :- the time of construction
- **renovationCondition** :- including other(1), rough(2), Simplicity(3), hardcover(4).

- **buildingStructure** :- including unknown(1), mixed(2), brick and wood(3), brick and concrete(4),steel(5) and steel-concrete composite (6).
- **ladderRatio** :- the proportion between number of residents on the same floor and number of elevator of ladder. It describes how many ladders a resident have on average.
- **elevator** :- have (1) or not have elevator(0).
- **fiveYearsProperty** :- if the owner have the property for less than 5 years

Cleaning

Data cleaning is the process of preparing data for analysis by removing or modifying data that is incorrect, incomplete, irrelevant, duplicated, or improperly formatted.This data is usually not necessary or helpful when it comes to analyzing data because it may hinder the process or provide inaccurate results. Let us look through the data for inconsistencies and clean it.

- The raw data contains Chinese symbols which are removed.
- Columns id and url are removed since it has noting to with the analysis.
- Data type of variable are changed into suitable format.
- Missing values in the data set are removed except for the column DOM since almost 50% of the data is missing.

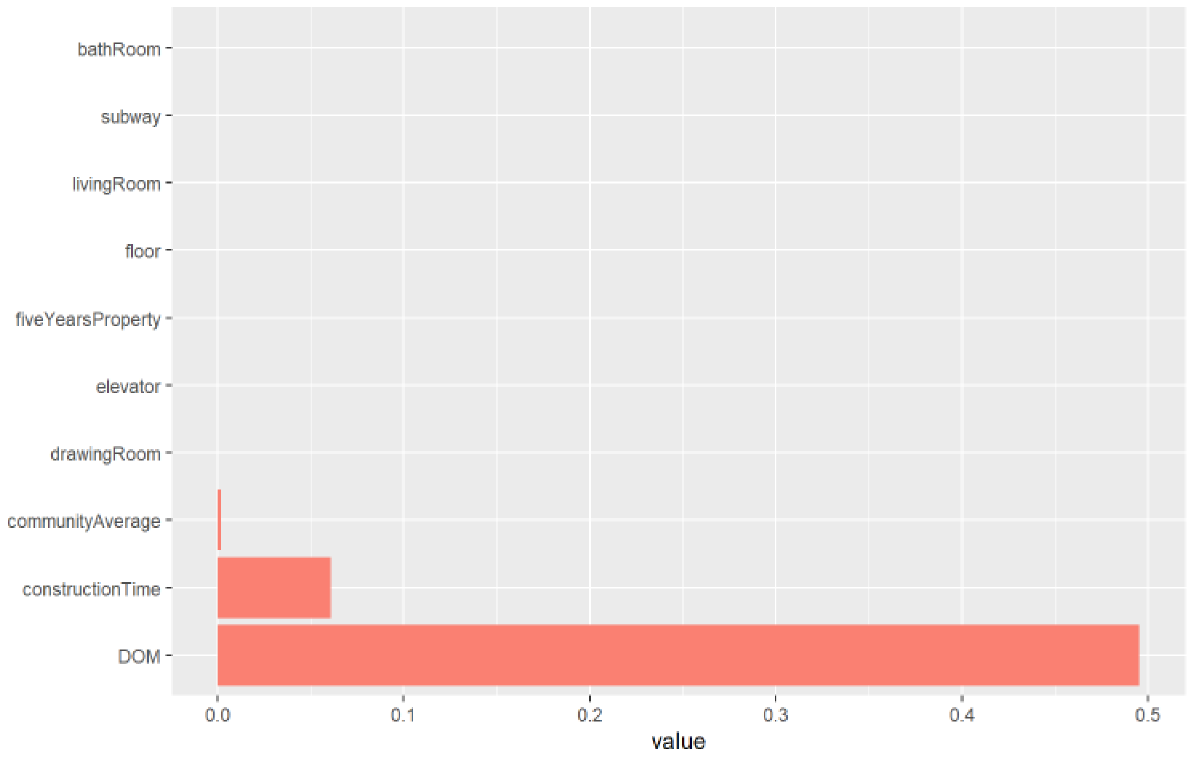


Figure 2.1: Percentage Missing Values

- The DOM variable is median imputed.
- The factor variables are renamed with appropriate names.

The cleaned data set can be downloaded from here.

Exploratory data analysis

This is the most exciting part of the project, whereby the analyst gets to play the role of a detective and squeeze the maximum knowledge on the data.

Severe data quality issues were identified as part of the process.

Firstly missing data issues were identified. It was surprising to find that about 50 per-cent of data was missing from the feature "DOM" which stands for Days On Market, whereas less than 1 per-cent of the data were missing for other features.

The distribution of the target variable "Price" was found to be right-skewed as expected, because as prices increase no of houses sold at that higher price reduce. This was validated by means of QQ-Plot and a Kernel Density Plot.

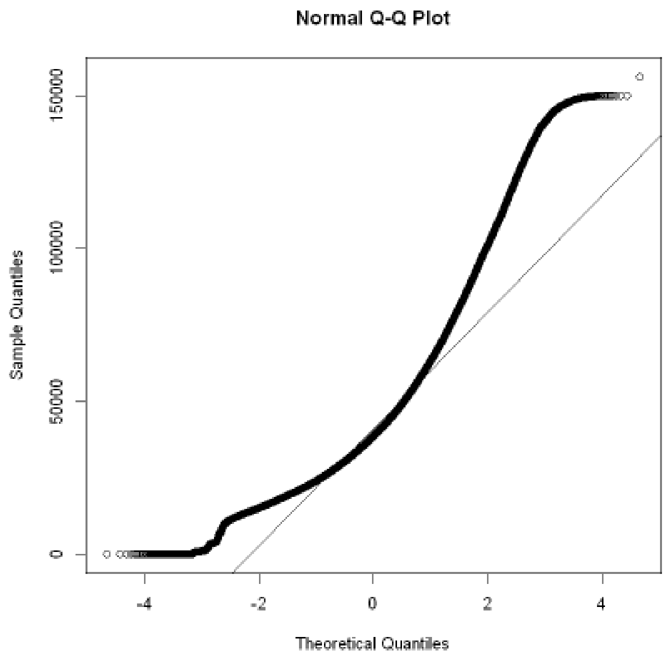


Figure 2.2: QQ Plot- Target Variable Distribution

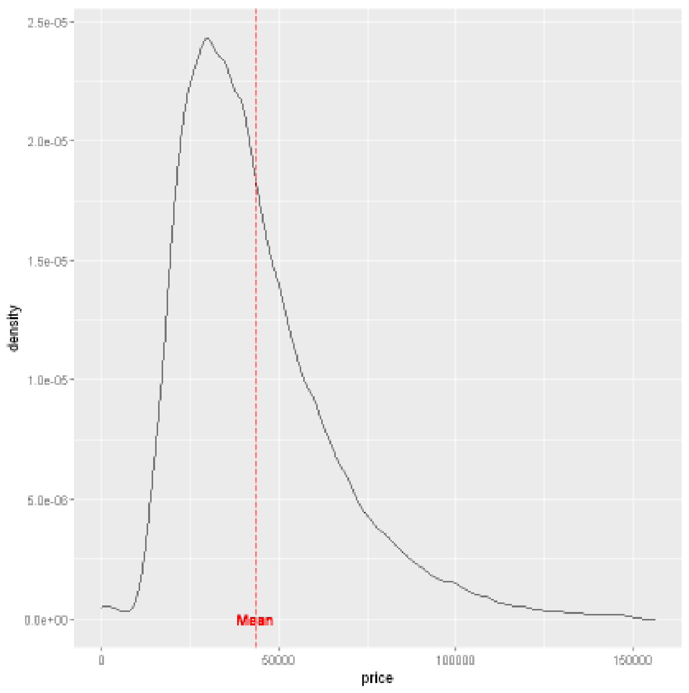


Figure 2.3: KDE- Target Variable Distribution

Exploring through other features, it was found that there were a total of 4035 communities present in Beijing varied through 13 districts.

Also number of building type in data set was found to be 13, but the metadata mentions only 4 types, this issue was found to be encoding issues when converting from the Chinese to English.

Majority of the transactions (more than 70 per-cent) was in central Beijing. House purchases in central Beijing is highly volatile .

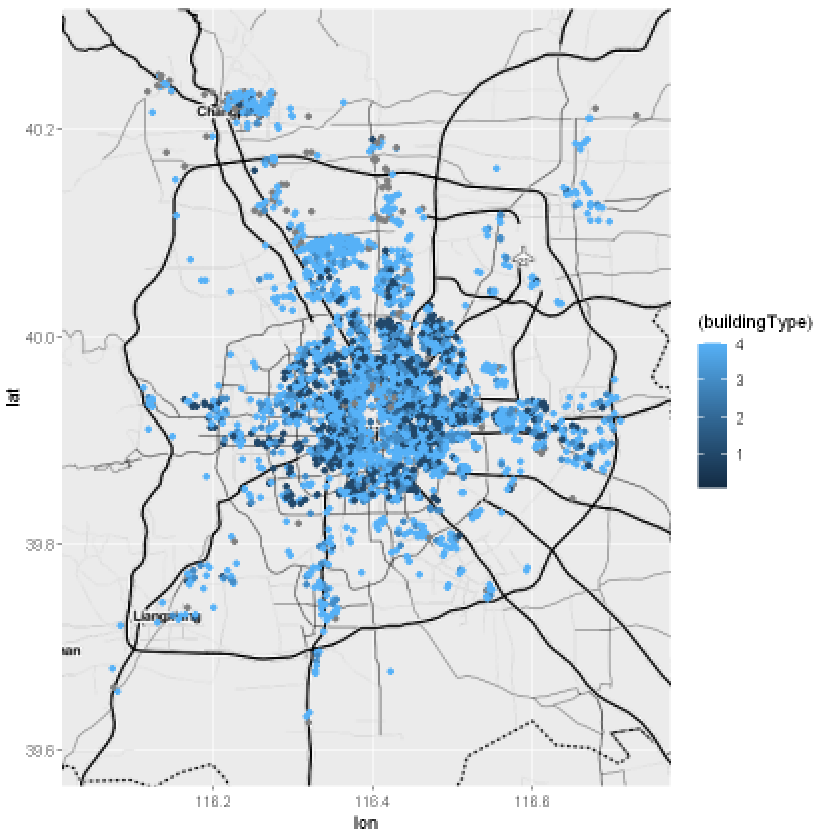


Figure 2.4: Transaction Concentration

For better understanding of data set, additional feature such as year was created. It was found that sample size for year 2018 was only 219 compared to a total of 318851 entries. So the sampling had to done properly to ensure that a best fit curve with maximum accuracy was obtained from the data set for the prediction. This has a direct implication on the model to be selected for the prediction.

These are some of the insights on data identified as part of Exploratory Data Analysis.

Modelling

For modeling the data we had to limit the parameters to ones a user would use to search for an apartment. This presents a challenge as its not going to be as accurate as a model with all the parameters included.

Several different model architectures were tested and some were rejected because of memory constraints.

In the end Xgboost was selected as it had the best performance with the final parameter set.

2.3 Testing

2.3.1 Test Plan Objectives

The objective is to evaluate model performance during development and select the best performing model. The metric used to evaluate model performance in the RMSE (Root Mean Squared Error).

2.3.2 Test Strategy

Model architectures are evaluated buy training them on 65% (training set) of the data set and the RMSE is calculated using the remaining 35% (testing set).

2.3.3 Performance Test

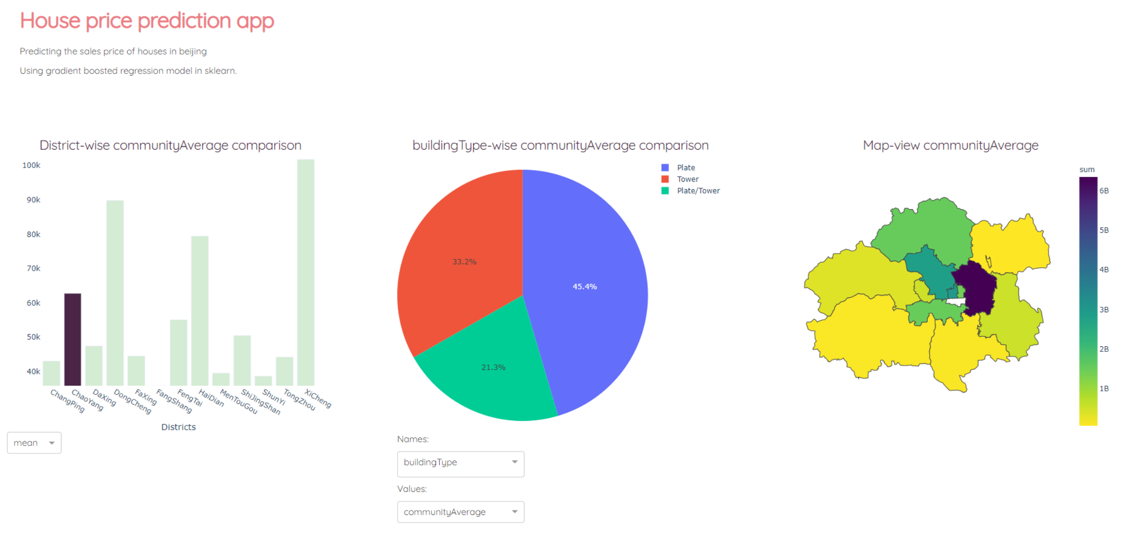
The initial model contained all parameters to get a baseline of best case performance to compare future models to. The parameters used where chosen based on the project requirements and the performance of each is as shown.

Model	**RMSE**
Baseline	100.96
Tree	158.46
Linear	145.97
Xgboost	118.46

Xgboost was taken as the final model.

Application deployment The performance of the application depends vastly on the dynos(cloud instances on heroku) assinged to the application, in case of high traffic it does not perform efficiently i.e it does not scale for more than one instance that serves all application requests around the world being deployed using free dynos.

2.4 Graphical User Interface (GUI) Layout



House price prediction app

Predicting the sales price of houses in beijing

Using gradient boosted regression model in sklearn.

mean

Names: buildingType

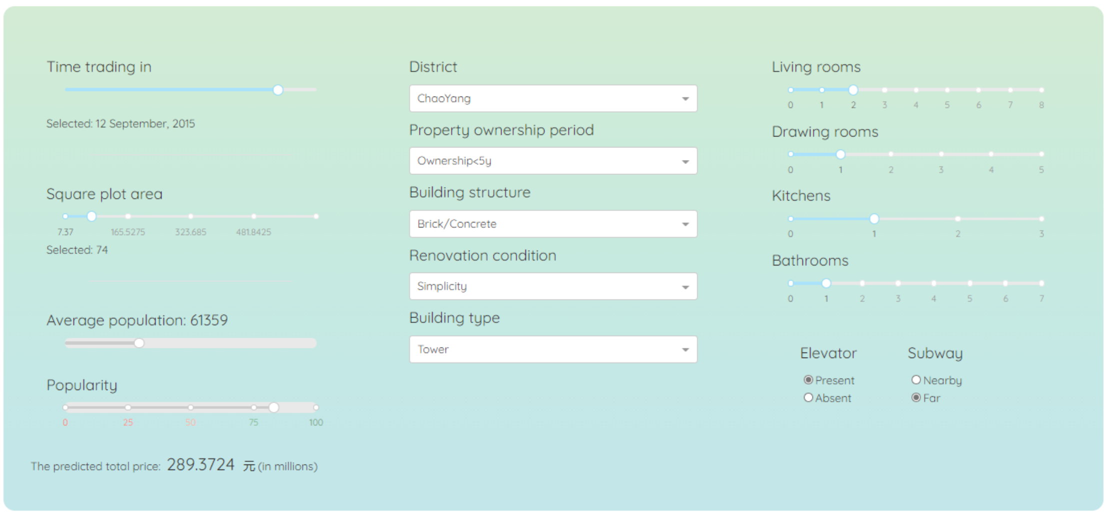
Values: communityAverage

Figure 2.5: Dashboard

The Front end is done entirely using python using the dash packages which includes dash_core_components and dash_html_components for building the UI and formatting of the page can be done using bootstrap css.

The interface is divided into 3 rows:

- Header - Title greeting along with a short description about the application
- Dashboard analytics - Charts plotted on various KPIs that can be used to gain insights on various features about the data
- Dashboard prediction - With dropdowns, sliders and radio button fields to obtain input from user which is then used to run the prediction in the backend and return the predicted price of house in million chinese yen.


Time trading in

Selected: 12 September, 2015

Square plot area

Selected: 74

Average population: 61359

Popularity

The predicted total price: 289.3724 元 (in millions)

District: ChaoYang

Property ownership period: Ownership<5y

Building structure: Brick/Concrete

Renovation condition: Simplicity

Building type: Tower

Living rooms: 2

Drawing rooms: 1

Kitchens: 1

Bathrooms: 1

Elevator: [✓ Present] Absent

Subway: Nearby [✓ Far]

Figure 2.6: Dashboard

UI elements

Number of floors will not be considered for the modelling at present since some people could be living on higher floors apartments with many floors, and some in houses that has 1 or 2 floor, this could possibly throw off the model unless handled accordingly.

By taking the median value of each of the features:

- square
- livingRoom
- drawingRoom
- kitchen
- bathRoom
- buildingType
- renovationCondition
- buildingStructure
- elevator
- fiveYearsProperty
- subway
- district

We can derive the metric - ’popularity’ based on the values selected which is calculated by measuring how far away from the 50th percentile data the selected data is and then calculating the ordinary least squares error - through which the percentage ’popularity’

Obtained Beijing topographical map from Nicole9519’s geojson data which had to be formatted to be used in the dash board, the formatting/cleaning is done using geopandas and geojson.

3 Conclusions

The House price prediction web application helps to predict price for a house in Beijing , given a set of features from the user. In addition to this to enhance the overall user experience, visualisations of the data, regarding the building such as district wise comparison, map view of total price, and community average are provided.

The data set was obtained from kaggle. Exploratory Data Analysis was done to give insights on data quality issues. These issues were addressed in the data cleaning process.The gradient boosted regression model was used for the model development and 95 per cent accuracy was obtained. The model was later deployed for prediction on heroku web application hosting platform.

The project with huge potential in the Real Estate Industry will cater to the needs as expected.

4 Further development or research

In the future a more advanced model trained using neural networks could be used to improve the accuracy and more graphical components for better insights into the data could be done.
The current data set has only data from 13 districts, the information about the 3 remainder districts is missing. Data from other regions could be included in model fitting to provide the service to other regions as well.

Bibliography

[1] Housing price of Beijing from 2011 to 2017, fetching from Lianjia.com

[2] Gradient Boosting Algorithm for Machine Learning, Machine learning Mastery

[3] XGBoost Documentation

[4] Dash User Guide