Design Document

First Stop for First-time home buyers

Team:Sha Li, Mobing Zhuang, Gary Gregg, Deepa Agrawal | Course project for Data 515A | Team Repo

Overview

This project is built using Python library for interactive visualization **Bokeh** and Python Machine Learning package **Sklearn**. This tool is built for first-time home buyers to set up expectations, plan budgets and make an informed decision on expenses before they go through the exhaustive house-hunting process given current real-estate market status.

Reference Documents

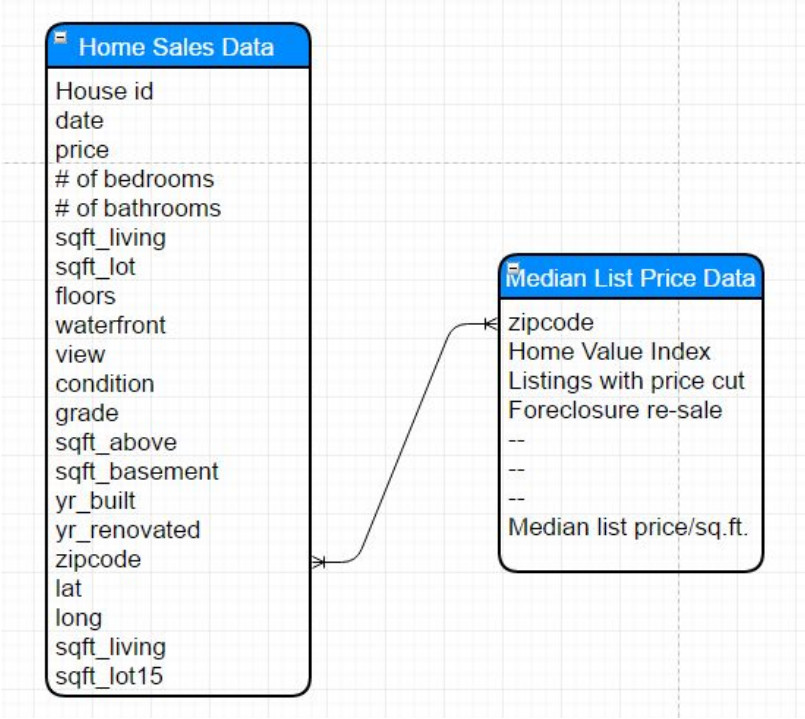
1. Link to Functional Specification.
2. Link to Project Plan.
3. Packages we would be using:
    - http://scikit-learn.org/stable/modules/linear_model.html#ridge-regression
    - http://scikit-learn.org/stable/modules/linear_model.html#lasso
    - http://bokeh.pydata.org/en/latest/docs/dev_guide.html#devguide

Data

Datasets:

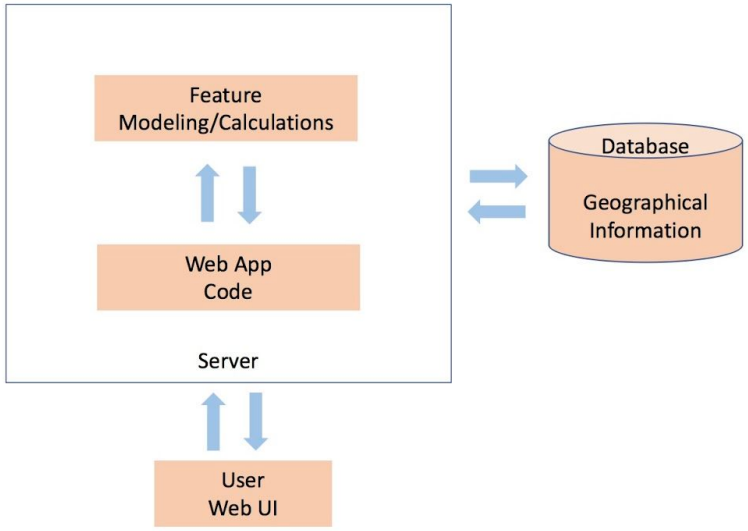
- King County Home Sales Data (2014-15)
- King County Home Values with Median List Price

Source: https://www.zillow.com/king-county-wa/home-values/



Overview: We are using King County housing dataset for the year 2014-2015 with prices for about 21k houses. This dataset has 18 unique house features which can used to fit the model. We have augmented this dataset with King County data with median list price per sq ft. The list price would help us calculating the right bidding price for a house.

Architecture



Components

1. **Statistical model for house price prediction.** The statistical model for house price prediction shall use the the **LassoCV** and **RidgeCV** functions in the **sklearn.linear_model** package to construct models that perform well given the home characteristics submitted to them. All variables aside from “price” shall be regarded as potential predictors, and “price” shall be the response of the model. To assess the quality of the model, available data may be divided into appropriately sized **training** and **test** sets, and these shall be varied randomly to insure that the models thus constructed have an optimum trade off of low **variance** versus low **bias**. As necessary, other model types shall be examined, and deployed using a similar quality assurance strategy.

2. **Mathematical model for house bidding price calculation.** The mathematical model for house bidding price calculation shall use the **statistical model for house price prediction** component to arrive at a baseline estimate of value for the property under consideration. This component shall use the estimate as a median price for the home. It shall then use an appropriate function to give a range for bidding. It is expected

that the function to construct the bidding range shall be linear or exponential in nature with, the median house price as input.

3. **Mathematical model for monthly expense calculation.** The mathematical model for monthly expense calculation shall be a function that takes into account the following considerations:

- Median price estimate from the **statistical model for house price prediction component**
- Current mortgage rates
- Formula to arrive at an estimate of property taxes for a home with the given value
- Formula to arrive at an estimate of insurance using known rates for a home with the given value
- Formula to estimate the monthly cost of utilities such as electricity, water and gas
- Formula to estimate the monthly cost of residential services, such as waste collection and recycling

4. **Database (using sql or google fusion table)**

- Function: to store geographical data for showing maps on web UI.
- Input: Price
- Output: latitude and longitude.

5. **A Html landing page which contains our web app homepage design, user interactions buttons, embedment for bokeh back-end server.**

- Function: to serve as user interface
- Input: user input in the format of drop down menu, navigation, submit forms et, al.
- Output: a) predicted price, suggested bidding price and monthly cost; b) a map showing houses within the predicted price range.

6. **Bokeh back-end server including but not limited to bokeh widget, bokeh components, Geojson and google mapping and styling.**

- Function: to serve as back-end server to receive user's' inputs, run mathematical models and return results to front-end UI.
- Bokeh documentation:http://bokeh.pydata.org/en/latest/index.html

Interactions

In order to realize real-time interactions, we have designed three main interfaces between our components:

a) the interactions between users and user interface;
b) the interactions between front-end web UI with back-end Bokeh server
c) the Interactions between Bokeh server with databases.

The working mechanism of the interactions between components will be described in details in the following use cases.

1) **Use Case Description:**

Prospective buyer desires to determine a price range for a home with a selection of features.

Interactions:

After the user entered the desired parameters of the house in the user interface of the web application, there will mainly be two interactions. Firstly, the Bokeh server would predict the housing price based on the trained model and return the result to the front-end. Thus, the user can review the results, try multiple parameter sets, and eventually be able to make better decisions during house-hunting. Secondly, the web server will communicate with the database and present the location of the potential interested houses in the map. The user will be able to hover over the map to know the exact location of the house, as well as other information such as, the school district and the zip code.

2) **Use Case Description:**

Prospective buyer desires to determine an appropriate bidding range for homes for which they have interest.

Interactions:

The interactions in this case are very similar with the first case. After the parameters are entered though the user interface (including the listing price), there will mainly be two interactions. Firstly, the Bokeh server would perform prediction on the bidding price based on the parameters entered and then return the result to the front-end. The second interaction here is the same with the first case. Except that when the user hovers over the map, the recommended bidding price for each potentially interested house will also be displayed.

3) **Use Case Description:**

Prospective seller desires to determine an appropriate listing price for a home with a selection of features.

Interactions:

Very similar to use case #1, after the user entered the desired parameters of the house in the user interface of the web application, there will mainly be two interactions. 1) The Bokeh server predict the housing price based on our machine learning model and return the result to the front-end UI for user to review. 2) The Bokeh server interfaces with the database and present the locations of the houses.

4) **Use Case Description:**

Prospective buyer desires to determine exact monthly expenses she will incur on the house of her choice.

Interactions:

Upon initiation(On selecting the tab for 'My Monthly Expenses'), the user interface will present to the prospective user a collection of appropriate and intuitive data-entry widgets, arranged in a logical format. The user will enter her inputs(which is the listed price or house features) and submit to the user interface with a pushbutton (or similar finalization widget). The user interface will submit the input parameters to the Bokeh back-end server, which will use a mathematical model to determine the monthly expenses for the user. These will be returned, and displayed to the user back to the user interface.

Testing

A different testing strategy shall be employed for each component of the system. Accordingly, here we address separate strategies for testing each of:

- **User/Web UI**
- **Web Application Code**
- **Feature & Modeling Calculations**

The component breakdown of the application closely resembles, and is inspired by the well-known **Model-View-Controller** pattern, and shall be tested based on strategies that have been proven successful for similarly structured software systems.

User/Web UI. Bokeh is a Python interactive visualization library that targets modern web browsers for presentation. Bokeh has its own testing utility suite located in the **bokeh.server.test.utils** package. Tools in this package shall be extensively employed to insure quality, and proper functionality of all user and web UI components. In particular, the tools provided in the **test_server**, **test_callbacks** and **test_server** libraries shall be used. Additionally, **Selenium** may be used as a testing tool. **Selenium** is a portable, software-testing framework for web applications. Selenium provides a record/playback tool for authoring tests without the need to learn a test scripting language. It also provides a test domain-specific language (**Selenese**) to write tests in a number of popular programming languages, include Python.

Web Application Code. Many of the same test tools used for the **User/Web UI** component of the system shall be applicable here, including use of the **bokeh.server.test.utils** package, and the functionality provided in the **test_server**, **test_callbacks** and **test_server** libraries. For the interface to the **Feature & Modeling Calculations** component, stub callbacks may be employed to insure that the parameters passed to features & modeling are correct and compliant with the desired interface. To the extent that it insures correct functionality, **Selenium** may be used here to drive testing through to the **Web Application Component**. Additionally, the unit test capability included in the native Python unit test framework shall be employed to insure correct functionality at the lowest levels.

Feature & Modeling Calculations. Software in this component shall be unit tested at the lowest level utilizing the native Python unit test framework. Specifically, unit tests shall be created that simulate input to the component from the **Web Application Code** layer. Ad-hoc testing shall be employed as required on a step-wise basis during development using the tools provided in the **Pycharm** integrated development environment, including step-wise debugging and watchpoints. To the extent possible, all ad-hoc testing shall be subsequently captured in unit tests using the native Python unit test framework. All publicly and internally exposed methods in this component shall have meaningful unit tests. For integration testing, **Selenium** may be employed at the final stage to drive through testing from the **User/Web UI** component to the **Web Application Code** component to the **Feature & Modeling Calculations** Component.