# CS512 FUN Projects - Spring 2016

Yi Zhong yz614
Qingqiao Hu qh41
Qi Dong qd33
Computer Science Dep., Rutgers University
Piscataway, NJ, USA
Email: yz614@scarletmail.rutgers.edu qh41@scarletmail.rutgers.edu qd33@scarletmail.rutgers.edu

***Abstract*— Our project aims at building a house recommendation system so that users can receive feedback on house's value and suggestion selling price. The data are received from the website, and the predictions are based on regression unit, which provides data analysis process by converting all the attributions to a new variable.**

## I. Project Description

Our project aims at construct a visual personalized recommendation system which can be used by different types of users in renting-related activities: house buyers, leasing agencies, and landlords. This system allows buyers to find a satisfactory house by evaluating the options, based on personal preferences that depend on both intrinsic attributes of the houses and their geographic relations to other places of interest. It also allows landlords to see the preferences and distributions of potential buyers so that they can adjust the pricing and improve the conditions of their houses to attract buyers more profitably. The main obstacles are automatically gathering the heterogeneous data related to each housing option, and allowing the user to adjust evaluation rules conveniently. We plan to mimic the real world thinking processes of the above-mentioned users and handle the problem step by step.

The project has four stages: Gathering, Design, Infrastructure Implementation, and User Interface.

### *A. Stage1 - The Requirement Gathering Stage.*

A visual interface, such as a web page with a dynamic map and various options, that allows potential home seekers to filter known for-sale houses by their personal preferences; they can set the preferences on the map (for example, places or types of places that they want to be close to or far from) and on the tags and attributes of each house(such as areas, number of bedrooms, number of bathrooms). Landlords can use the same system to query the statistics of house seekers that are interested in their houses, and see if their preferences are satisfied by the house, and if not, figure out how to improve it to attract buyers.

- The general system description:
- Our system is very deliverable because our interface would be very simple to use. Users only need to input their requirements as some instructions on the interface. Our system aims at construct a visual personalized recommendation system which can be used by different types of users in buying-related activities: House seeker, leasing agencies, and landlords. This system allows buyers to find a satisfactory house by evaluating the options, based on personal preferences that depend on both intrinsic attributes of the houses and their geographic relations to other places of interest. It also allows landlords to see the preferences and distributions of potential buyers so that they can adjust the pricing and improve the conditions of their houses to attract buyers more profitably. The main obstacles are automatically gathering the heterogeneous data related to each housing option, and allowing the user to adjust evaluation rules conveniently. We plan to mimic the real world thinking processes of the above-mentioned users and handle the problem step by step.
- The three types of users (grouped by their data access/update rights):
- User 1. House seekers: they want to rent a house or room in a certain area, satisfying some geographic preferences like being close to their workplace or any park, as well as functional preferences like having a number of bedrooms and garages. These preferences are personal and can not necessarily be inferred automatically, so user input is necessary.
- User 2. Landlords: they have houses or rooms to sale, and want to find out how to get the most profit, so they need to find out about the preferences of potential buyers; some of these preferences cannot be satisfied by changing the house itself (like location and number of rooms in most cases) but others like furniture and pricing can be adjusted for a better profit.
- User 3. Leasing agencies: they have more houses than typical landlords so can benefit more from knowing a wide range of renter preferences and improve their conditions accordingly.
- The user's interaction modes: User 1: Use keyboard and mouse to input requirements, and system feeds back houses' Informations according to the requirements.
- The real world scenarios:
  - Scenario1 description: Buy a non-commercial house.
  - System Data Input for Scenario1: Price, location, room area, number of bedrooms, number of bathrooms, parking lot size, and so on.

  - Input Data Types for Scenario1: A list of the values and choices above.
  - System Data Output for Scenario1: A list of rooms and their informations that accord with the requirements of users' input.
  - Output Data Types for Scenario1: A table of different attributes of rooms.
  - Scenario2 description: Buy a commercial house.
  - System Data Input for Scenario2: Price, location, room area, number of bedrooms, number of bathrooms, parking lot size, and so on.
  - Input Data Types for Scenario2: A table of the values or choices above.
  - System Data Output for Scenario2: A list of s that accord with the requirements of users' input.
  - Output Data Types for Scenario2: A table of different attributes of houses.
- The user's interaction modes: User 2. Landlords: Use keyboard and mouse to input their house's information, system feeds back a list of houses' informations that similar with the informations of user's input and suggested prices of the house according to our collected data and informations.
- The real world scenarios:
  - Scenario1 description: Sale a non-commercial house.
  - System Data Input for Scenario1: Price, location, room area, number of bedrooms, number of bathrooms, parking lot size, and so on.
  - Input Data Types for Scenario1: A table of the values or choices
  - System Data Output for Scenario1:
    A list of houses' informations that similar with the informations of user's input, and suggestions of the room price.
  - Output Data Types for Scenario1:
    A table of different attributes of rooms.
  - Scenario2 description: Sale a commercial house.
  - System Data Input for Scenario2:
    Price, location, room area, number of bedrooms, number of bathrooms, parking lot size, and so on.
  - Input Data Types for Scenario2: A table of the values or choices
  - System Data Output for Scenario2: A list of houses' informations that similar with the informations of user's input, and suggestions of the house price.
  - Output Data Types for Scenario2: A table of different attributes of houses and texts.

  User 3. Leasing agencies:Use keyboard and mouse to input requirements( these requirements usually much less than User's). After analyse our collected data according to the requirements, system provide different suggestions and recommendations to help users gain more profit.
- The real world scenarios:
  - Scenario1 description: Find rooms' informations
  - System Data Input for Scenario1: Location, room size, room numbers.
  - Input Data Types for Scenario1: A list of the values and choices above
  - System Data Output for Scenario1:
    A list of rooms and their informations that accord with the requirements of users' input and suggestions about how to gain more agency fee.
  - Output Data Types for Scenario1: A table of different attributes of rooms and texts.
  - Scenario2 description: Find houses' informations
  - System Data Input for Scenario2: Location, house type, room numbers, garage numbers.
  - Input Data Types for Scenario2: A table of the values or choices above
  - System Data Output for Scenario2: A list of houses and their informations that accord with the requirements of users' input and suggestions about how to gain more agency fee.
  - Output Data Types for Scenario2: A table of different attributes of houses and texts.
  - Project Time line and Divison of Labor.
    Timeline: 4/1, Finish Stage 2. 4/15 Finish Stage 3. 5/1 Finish the project and report Yi Zhong: algorithm design, documentation, evaluation. Qi Dong: interface implementation. Qingqiao Hu: project report, testing and power point presentation.

### *B. Stage2 - The Design Stage.*

- Short Textual Project Description. The recommendation system works along with users' inputs to give suggestions as a feedback to users about users' houses' prices. We build the system database using crawling, and deposit user's information to the database so that the recommendation system can calculate users' houses to new variables. The feedback includes prediction on house's price, recommended selling price. It is the same to other types of users. For example, landlords can decide when to sell the house for the maximum profit.
- Flow Diagram: see Fig 1
- High Level Pseudo Code System Description.
  - Information Integration Module: Integrating Unit(website-info) We collect the houses' information of New Jersey from the website "weichert" using crawling to obtain the property information. Then we save the information to Excel files. Return Processed data
  - Data Analysis Module: Data Analysis Unit(Processed data) Pass Processed data to Data Analysis Unit, then process it through Regression Unit. The process procedure includes to uniform the file style, to remove the redundant information from the original dataset, to transform the non-digital information to digital information. Deposit it to system database

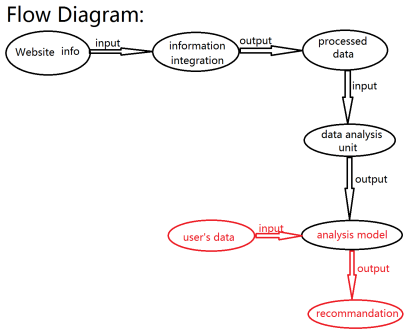


Fig. 1.

  - ∗ User's Preference input() Input User's preference
  - ∗ Regression Unit() Rate user's input house, then give the prediction using linear regression on house's price. Form a suggestion feedback to users

– Algorithms and Data Structures. Algorithms and Data Structures: First we use web crawling algorithm to stretch useful data from website. This algorithm works with website code and get different digital data from website. Secondly we process these data to table structure type which our data analysis unit could use. Finally, we use our model deal with digital data and output useful recommendations.
– As to the data structure, we use list to save houses' information from the website, process the data using string, use data frame to achieve regression and price prediction.
– The main algorithms used in our project are crawling to get the total information from the website, bubble sort which is used for sorting the house list according to the prices provided from the website and regression which is used for providing the standard of price prediction and providing similar houses from the system data-list.
– Flow Diagram Major Constraints.
  - ∗ First integrity constraint: Houses data on the website must contain price, room numbers, year, area.
  - ∗ Second integrity constraint: Input data to the data analysis unit must be table format with numbers.

### *C. Stage3 - The Implementation Stage.*

The programming language we use for this project is Python 2.7, the programming environment is Windows 64-bit. The deliverables for this stage include the following items:

– Data Snippet: See Fig.2
– Regression and Prediction: See Fig.3
– Datacrawling: See Fig.4-6
– DataAnalysis: See Fig.7-8
– Demo and sample findings
  - ∗ Data size: 1MB; Disk Resident: 10MB;

dataanalysis

```
C:\Anaconda2\python.exe C:/Users/YZ/Desktop/512project/dataanalysis.py
<class 'pandas.core.frame.DataFrame'>
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.879
Model:                            OLS   Adj. R-squared:                  0.878
Method:                 Least Squares   F-statistic:                     540.2
Date:                Mon, 25 Apr 2016   Prob (F-statistic):          7.89e-168
Time:                        15:28:18   Log-Likelihood:                -5231.3
No. Observations:                 376   AIC:                         1.047e+04
Df Residuals:                     371   BIC:                         1.049e+04
Df Model:                           5
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [95.0% Conf. Int.]
------------------------------------------------------------------------------
tax           45.9931      2.017     22.803      0.000        42.027    49.959
year         -61.4716     46.722     -1.316      0.189      -153.345    30.402
bedroom    -1.127e+04   1.45e+04     -0.775      0.439     -3.99e+04  1.73e+04
bathroom    6.121e+04   1.94e+04      3.157      0.002      2.31e+04  9.93e+04
lot         2.991e+04    9.1e+04      0.329      0.743     -1.49e+05  2.09e+05
==============================================================================
Omnibus:                      264.447   Durbin-Watson:                   1.469
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             5272.456
Skew:                           2.664   Prob(JB):                         0.00
Kurtosis:                      20.554   Cond. No.                     9.79e+04
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 9.79e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
[  4.60407764e+01   3.87902675e+02  -6.12695986e+03   5.35861884e+04
   4.49184681e+04]
input as: tax,year,bedroom,bathroom,lot3000,2003,6,3,1
3000 2003 6 3 1
y_pred=1084006.66131

Process finished with exit code 0
```

Fig. 2.

dataanalysis

```
C:\Anaconda2\python.exe C:/Users/YZ/Desktop/512project/dataanalysis.py
<class 'pandas.core.frame.DataFrame'>
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.879
Model:                            OLS   Adj. R-squared:                  0.878
Method:                 Least Squares   F-statistic:                     540.2
Date:                Mon, 25 Apr 2016   Prob (F-statistic):          7.89e-168
Time:                        15:28:18   Log-Likelihood:                -5231.3
No. Observations:                 376   AIC:                         1.047e+04
Df Residuals:                     371   BIC:                         1.049e+04
Df Model:                           5
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [95.0% Conf. Int.]
------------------------------------------------------------------------------
tax           45.9931      2.017     22.803      0.000        42.027    49.959
year         -61.4716     46.722     -1.316      0.189      -153.345    30.402
bedroom    -1.127e+04   1.45e+04     -0.775      0.439     -3.99e+04  1.73e+04
bathroom    6.121e+04   1.94e+04      3.157      0.002      2.31e+04  9.93e+04
lot         2.991e+04    9.1e+04      0.329      0.743     -1.49e+05  2.09e+05
==============================================================================
Omnibus:                      264.447   Durbin-Watson:                   1.469
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             5272.456
Skew:                           2.664   Prob(JB):                         0.00
Kurtosis:                      20.554   Cond. No.                     9.79e+04
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 9.79e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
[  4.60407764e+01   3.87902675e+02  -6.12695986e+03   5.35861884e+04
   4.49184681e+04]
input as: tax,year,bedroom,bathroom,lot3000,2003,6,3,1
3000 2003 6 3 1
y_pred=1084006.66131

Process finished with exit code 0
```

Fig. 3.

dataanalysis.py × Data_CrawlingWrangling.py ×

```
import re
import urllib
import time
import csv

def getdata(num):
    datalist = []
    htmlFile = urllib.urlopen(url)
    htmlText = htmlFile.read()
  #  regex = '<h3>Property Information</h3>' + '\n.*' + '<ul>(.*)MLS/Web'
# regex = '<h3>Property Information</h3>.*\n.*\n<ul>.*\n.*\n' + '.*<li>(.*)</li>.*\n.*\n' * 12
    regex = '<h3>Property Information</h3>([^\t]*?)</ul>'

    datalist.append(re.compile(regex).findall(htmlText))
    datalist = re.sub(',', "", str(datalist))
    datalist = re.sub('', "", str(datalist))
   # print datalist
    datalist = re.sub('(Fees.+?)</li>', "", str(datalist))
    finallist = []
    if  re.findall(r'Commercial', str(datalist))==[]:
        price = re.findall(r'\$+(\d+)</li>+', str(datalist))
        if len(price) == 2:
            for i in price:finallist.append(i)

            year = re.findall(r"Builtin(.+?)</li>", str(datalist))
            if year != []:
                finallist.append(year[0])

                bedroom = re.findall(r'<li>+(\d+)Bedroom+', str(datalist))
                if bedroom != []:
                    finallist.append(bedroom[0])

                bathroom = re.findall(r'<li>+(\d+)FullBath', str(datalist))
                halfbathroom = re.findall(r'Bath+(\d+)HalfBath', str(datalist))

                if bathroom != []:
                    if halfbathroom != []:
                        finallist.append(bathroom[0]+'.'+halfbathroom[0])
                    else:
                        finallist.append(bathroom[0])


                lot = re.findall(r'Lots', str(datalist))
                if lot != []:finallist.append('1')
                else:finallist.append('0')
            # # print datalist
            #  datalist = re.sub('(MLS.+?)</li>', "", str(datalist))
            #  datalist = re.sub('(Unit.+?)</li>', "", str(datalist))
            # # datalist = re.sub('<li>(.+?)Story</li>', "", str(datalist))
```

Fig. 4.

dataanalysis.py × Data_CrawlingWrangling.py ×

```
        # finallreplace('\r','')
    textlist = datalist_clean(datalist)
  # print textlist
    htmlFile.close()
    return finallist,textlist

def datalist_clean(datalist):
    with open(test, "w") as w:
            w.write(str(datalist))
    with open(test, "r") as r:
            data = r.read()
        #   print type(data)
            data = data.replace('\\r\\n','')
            data = data.replace('<ul>', '')

            data = data.replace('[[\'', '')
            data = data.replace('\']]', '')
            data = data.replace(' ','')
            data = data.replace(',', '')
            data = data.replace('/', '')
            data = data.replace('-', '')
          #  data = data.replace('$', '')
            data = data.replace('Commercial','*')
            data = data.split("<li>")
        #     data = [x for x in data if x != '']
        # #   print type(data)
        #     b = 1
        #     for x in data:
        #         if 'Lots' in x:
        #             b = 0
        #     if b:
        #         data.insert(len(data),0)
        #     else:
        #         data.insert(len(data), '1')
        #
        #
        #     a = 1
        #
        #     for x in data:
        #         if 'Tax' in x:
        #             a = 0
        #     if a:
        #         data.insert(3,0)
        #
        #     datastr = str(data)
        #     print datastr
        #     for i in 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ':
        #         datastr = datastr.replace(i,'')
        #     print datastr
```

Fig. 5.

```
with open(house,"wb") as h:
        h_csv = csv.writer(h)
        title = ['price','tax','year','bedroom','bathroom','lot','address']
        h_csv.writerow(title)
        for a in range(2728,2750):
            num = str(a)
            url = "http://www.weichert.com/6196" + num
            finallist,texlist = getdata(url)
            if len(finallist)==6:
                finallist.append(url)
                finallist = finallist + texlist
                print finallist
                h_csv.writerow(finallist)
        #     data = datalist_clean(datalist)

            # if data[0] == '[[]]':
            #     continue
            # else:
            #
            #     if data[1] == '*':
            #         c_csv = csv.writer(c)
            #         c_csv.writerow(data)
            #     else:
            #         h_csv = csv.writer(h)
            #         h_csv.writerow(data)
```

Fig. 6.

dataanalysis.py × Data_CrawlingWrangling.py ×

```
import pandas as pd
import csv
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.cross_validation import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
import statsmodels.formula.api as sm
import operator

def sort_by_column(csv_cont, col, reverse=False):
    header = csv_cont[0]
    body = csv_cont[1:]
    if isinstance(col, str):
        col_index = header.index(col)
    else:
        col_index = col
    body = sorted(body,
                  key=operator.itemgetter(col_index),
                  reverse=reverse)
    body.insert(0, header)
    return body

def csv_to_list(csv_file, delimiter=','):
    with open(csv_file, 'r') as csv_con:
        reader = csv.reader(csv_con, delimiter=delimiter)
        return list(reader)

def convert_cells_to_floats(csv_cont):
    for row in range(len(csv_cont)):
        for cell in range(len(csv_cont[row])):
            try:
                csv_cont[row][cell] = float(csv_cont[row][cell])
            except ValueError:
                pass



def write_csv(dest, csv_cont):
    """ New CSV file. """
    with open(dest, 'wb') as out_file:
        writer = csv.writer(out_file, delimiter=',')
        for row in csv_cont:
            writer.writerow(row)

csv_cont = csv_to_list('house2.csv')

# print('nnOriginal CSV file:')
# print csv_cont
```

Fig. 7.

* We collect houses' information from "weichert", calculate the prediction price through multiple linear regression. After collecting all the information from the website, we find out that the built-year of houses have no effects on houses' prices. And this is weird, because in common sense, the elder the house is, the lower the price will be, considering higher cost on maintenance, lower security standard. Maybe the website does not put the built-year into the calculation standard.

### *D. Stage4 - User Interface.*

Please insert your deliverables for Stage4 as follows:

- Unzip the interface web page and related data into a directory, run the command "python -m SimpleHTTPServer 8000" (if using python 2) or "python -m http.server 8000"(if using python 3) in the directory,

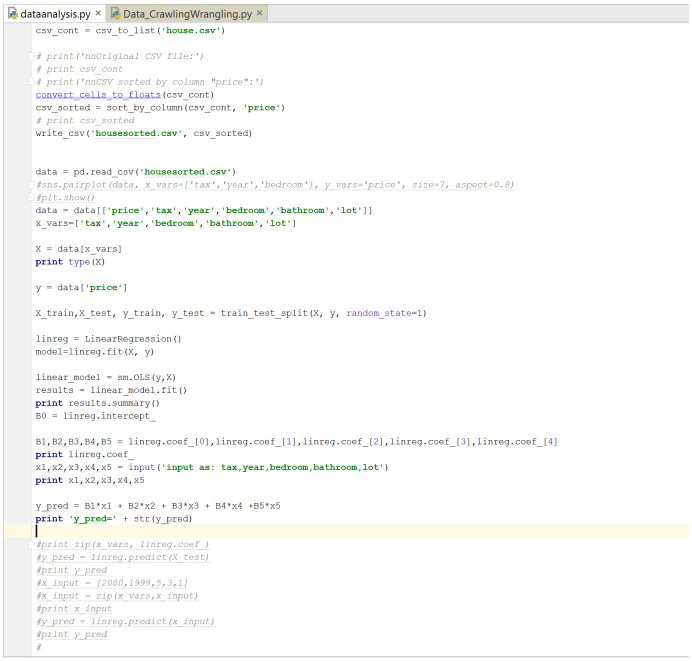


Fig. 8.

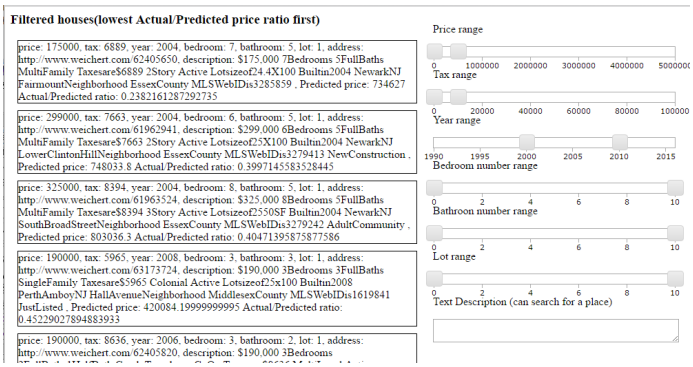


Fig. 9.

then use a web browser and navigate to localhost: 8000/ house data filter.html. The initial screen shot:

– Two different sample navigation user paths through the data exemplifying the different modes of interaction and the corresponding screen shots.

1. The user wants to find cheap small houses, so slide the bedroom, bathroom, price and tax sliders to the left, then the web page will display the houses that seem like deals according to the actual price to predicted price ratio. The predicted price is calculated with the coefficients obtained from linear regression.

1

2. The user wants to find any houses that are deals in the Edison region, so type the text ”Edison” (case sensitive) into the Text description search box, and slide all sliders as wide apart as possible. The top houses displayed are the ones that seem to be deals

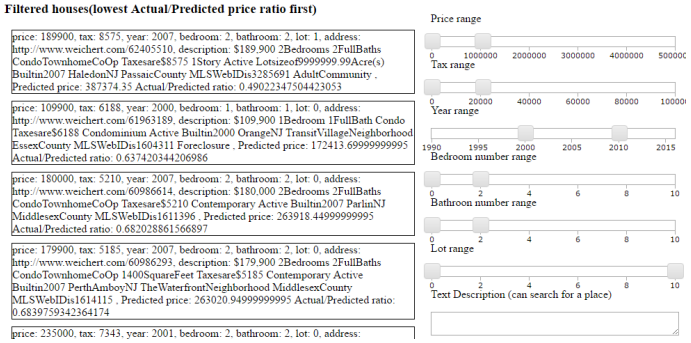


Fig. 10.

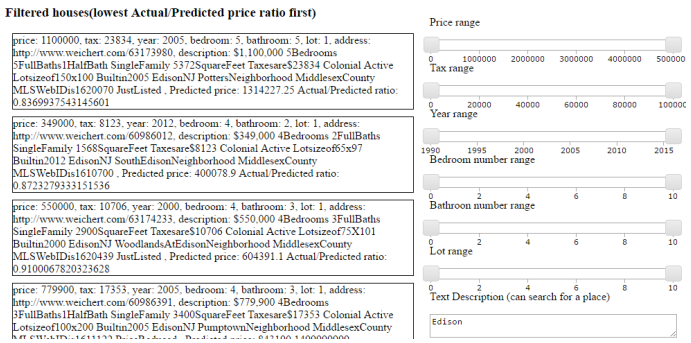


Fig. 11.

according to the linear regression.

– The error messages popping-up when users access and/or updates are denied (along with explanations and examples):
  * The error message: There are no access denied error messages, because the data gathering/storing and the user interface are different parts in our application. The user interface has all access to data already scraped from the Web.
  * The error message explanation (upon which violation it takes place): The houses display being empty means that the request cannot be met within the current data set.
  * The error message example according to user(s) scenario(s): If the user inputs a place name incorrectly (for example, type ”Edison” as ”Eddison”) then most likely there will be no results displayed. What’s more, since the result is updated in real time for every character of text, the user can notice the mistake immediately when he types ”Edd”.
– The information messages or results that pop-up in response to user interface events.
  * The information message: The ”message” is the left side’s houses display area being empty.
  * The information message explanation and the corresponding event trigger Because the user starts typing ”Edison” as ”Eddison” in the text search box, the houses display becomes empty because

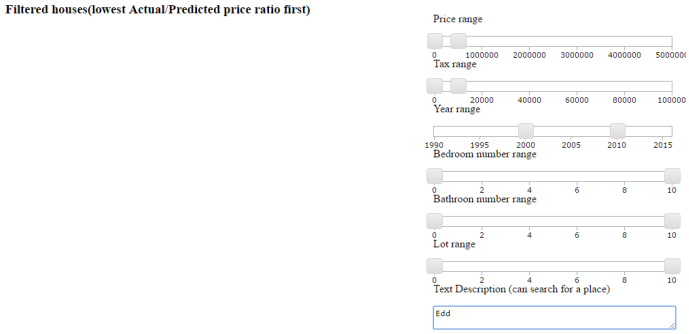


Fig. 12.

”Edd” doesn’t match any data records.

- The error message example in response to data range constraints and the corresponding user’s scenario If the user inputs conditions that cannot be met, the displayed houses area will immediately be empty, and the user can simply correct it and do not need to click on a pop-up. The interface showing no houses match a mistake in input:

– The interface mechanisms that activate different views.

- The interface mechanism: There is only one view, and the filtering of information is controlled by sliders and search boxes.

## II. Project Highlights.

- Our project aims at building a house-buying system for users like house-seekers, landlords, leasing agencies. The recommendation system provides predictions on house’s prices according to all the preferences. The system database are built from the website ”Weichert”, and price prediction is fulfilled using multiple linear regression which ensures the accuracy of the system. Upon using the system interface, users can get the similar houses directly when they move the preference sliders, and it’s simple to use.

1) Title: House buying system(Yi Zhong, Qingqiao Hu, Qi Dong)
2) Project Goal: Building a recommendation system on house sale.
3) Outline of the presentation:
4) Description: Our project is based on the information from âWeichertâ, a website of house renting or buying, and we focus on the houses of New Jersey. We collect the house’s information, and save them to our system database. We use linear regression to give prediction to the house price according to the preferences provided by users, and provide the concrete property information of houses for users to choose from. â
5) Pictures are included in stage 4.
6) Project Stumbling Blocks: Information collection, Data processing, Linear Regression, UI Design
7) Future Extensions: Expanding the System Database to the US, even the world, Providing more preferences for users to choose from, Building a virtualization of the houses, so that users can get a clear view of similar houses according to the dynamic graph
8) References: Weichert