LARISSA LIU

House price predict app

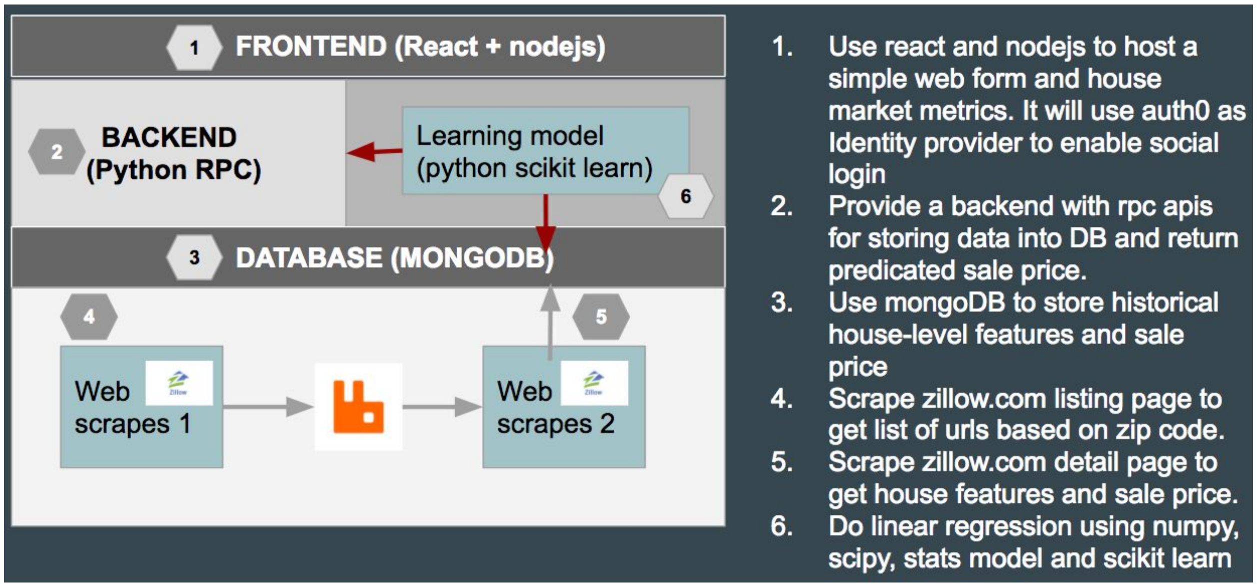
Learning regression model to predict house price

Introduction

This application will help sellers to predict the sale price of their house based on those house-level features they provided. Seller will also be able to add/update certain house renovation ideas there to see how much those renovation could help on improving the sale price.

In order to achieve a high level prediction accuracy, the application will scrape Zillow.com, an online real estate database to extract real estate listings available. This real estate scraper will extract details of property listings based on zip code. (for demo purpose, it will only focus on real estate market at Dallas, TX). this application will also use linear regression , one of most popular technique , to implement the learning model.

Architecture diagram

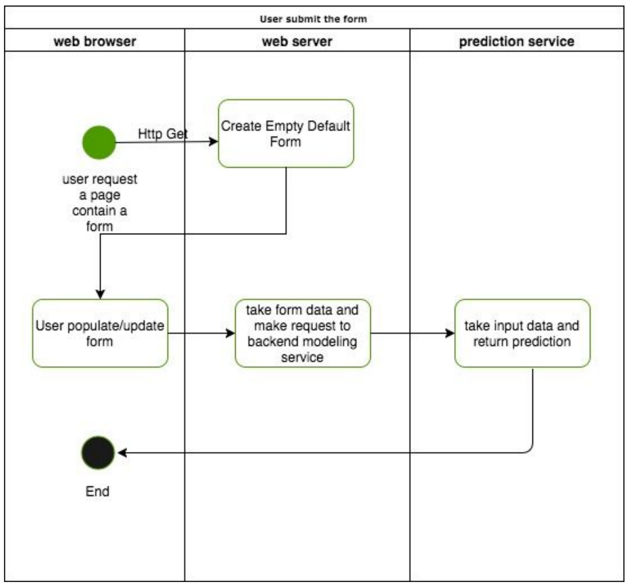


1. Use react and nodejs to host a simple web form and house market metrics. It will use auth0 as Identity provider to enable social login
2. Provide a backend with rpc apis for storing data into DB and return predicated sale price.
3. Use mongoDB to store historical house-level features and sale price
4. Scrape zillow.com listing page to get list of urls based on zip code.
5. Scrape zillow.com detail page to get house features and sale price.
6. Do linear regression using numpy, scipy, stats model and scikit learn

Modules design

User populate/update house features

Client : using ReactJS, just has one component called ‘houseForm’. Server: Using Express.


User submit the form
web browser
web server
prediction service
Http Get
Create Empty Default Form
user request a page contain a form
User populate/update form
take form data and make request to backend modeling service
take input data and return prediction
End

Scrape zillow house list page and save to cloud_amqp

Service name: house_monitor

Used python module: lxml, requests, redis, re, cloud_amqp_client

Description: based on county and zip code provided from a text file to scrape sold houses basic information per county + zip code and send to the queue. Have append '?fullpage=true' at the end of detail page url.

value	xpath
List of houses	`//div[@id='search-results']//article`
address	`.//span[@itemprop='address']//span[@itemprop='streetAddress']//text()`
city	`.//span[@itemprop='address']//span[@itemprop='addressLocality']//text()`
state	`.//span[@itemprop='address']//span[@itemprop='addressRegion']//text()`
postal_code	`.//span[@itemprop='address']//span[@itemprop='postalCode']//text()`
price	`.//span[@class='zsg-photo-card-status']//text()`
url	`.//a[contains(@class,'overlay-link')]/@href`

Scrape zillow house detail page and save to Mongodb

Backend Service name: houses_fetcher

Used module: beautifulSoup, request, lxml, re

Description: get one house metadate at a time and use url of that house to request detail page. Then scrape more house features and save those info as a single object into mongodb.

Get sale price

If price property in the message from queue, then do nothing. If price property equals to 0, then use regular expression to download all Ajax request url in the detail page. And get the second in the url lists. Use that url to make a request then use BeautifulSoup to parse the price history table. Find the first 'pending sale' price. And assign it as sale price.

Get flooring

Use XPATH (`//span[@class='hdp-fact-name' and text()='Flooring: ']/following-sibling::span[1]//text()`) to grab flooring field in the detail page. Give different type of floor a number. Those type are HardWood(4), Laminate(3),Carpet(2),Tile(1),No Flooring(0).

Get gutters

Use XPATH(`//span[@class='hdp-fact-name' and text()='Exterior Features: ']/following-sibling::span[1]//text()`). If there is content, then set as 1, otherwise as 0. Assign to 'gutters' property

Get Fencing

Use XPATH(`//span[@class='hdp-fact-name' and text()='Fencing: ']/following-sibling::span[1]//text()`). If there is content, then set as 1, otherwise as 0. Assign to 'fencing' property

Get number of Beds

Use XPATH(`//span[@class='addr_bbs' and contains(text(),'beds')]//text()`). If there is content, then cast it as integer and assign to 'beds' property, otherwise assigned the property as 0.

Get number of Baths

Use XPATH(`//span[@class='addr_bbs' and contains(text(),'baths')]//text()`). If there is content, then cast it as integer and assign to 'baths' property, otherwise assigned the property as 0.

Get number of sqft

Use XPATH(`//span[@class='addr_bbs' and contains(text(),'sqft')]//text()`). If there is content, then cast it as integer and assign to 'baths' property, otherwise assigned the property as 0.

Get year built

Use XPATH(`.//p[contains(text(),'Year Built')]//following-sibling::div[1]//text()`). If there is content, then cast it as integer and assign to 'built_yr' property, otherwise Use another XPATH (`.//span[@class='hdp-fact-value' and contains(text(),'Built in ')]//text()`) try to look for build year. If there is content, then cast it as integer and assign to 'built_yr' property, otherwise assigned the property as 0.

Get lots

Use XPATH(`//span[@class='addr_bbs' and contains(text(),'baths')]//text()`). If there is content, then cast it as integer and assign to 'built_yr' property, otherwise Use another XPATH (`.//span[@class='hdp-fact-name' and text()='Lot: ']/following-sibling::span[1]//text()`) try to look for build year. If there is content, then cast it as integer and assign to 'built_yr' property, otherwise assigned the property as 0.

Use tensorflow estimator to learning linear regression model on the house price

Component1 : importData.py

Load data, pre-process data and then generate training and testing data set

Main function signature:

load_data(y_name=”price”, train_fraction=0.7, seed=None)

Parameter:

y_name: the column to return as the label

Train_fraction: the fraction of the dataset to use for training.

seed: the random seed to use when shuffling the data. ‘Non; generates a unique shuffle every run

Returns:

A pair of paris where the first part is the training data, the second is the test data:

Example:

`(x_train, y_train), (x_test, y_test) = load_data(...)`

`x` contains a pandas DataFrame of features, while ‘y’ contains the label array

Component2 : linear_regression.py

Using training data to learning linear regression model and using testing data to verifying the accuracy of the model. Export model into a directory and return model as response.

Main function signature: build_model()

Returns:

The model_fn with following signature: `def model_fn(features, labels, mode, config)`