Technology Review:
Housing Recommendation

cse583

Student:Yuchen Wang
Yuhan Gao
Jinlin Xiang
Xintong Xu

OUTLINE

- Project overview
- User case
- Python Libraries

Project overview

- Create a web-bases marketplace for people to list, discover and book unique accommodations around the world
- Travelers can find the all rooms in target area and get the recommendations
- Landlords could get the suggested prices of their new rooms

Python Library

- Python library we used:
  - Dataframe: pandas, numpy
  - Machine Learning : sklearn
  - Visualization: matplotlib, folium
- API we used:
  - Mapbox

Use case 1

Detailed listing information on map view

- Dataset with listing information and location
- User interface that allows users to select area
- Map view that allows users to visualize all rooms location in this area

Use case 2

Recommend top 10 rooms for customers

- Dataset with guests' review scores
- User interface that allows users to input the info about the room
- User interface that allows users to select the order of recommendations
- Map view that allows users to visualize the recommendations' location

MapBoxGL(API) :

```
mapboxgl.accessToken = 'pk.eyJ1IjoiZ2FuZ2JlaXFpMTk5MTU5IiwiYSI6ImNqMm1kcTNjczAwdWUyd253Z3IxcW5xMHYifQ.iws1nZ75QX-5DETQPedmnQ';
var map = new mapboxgl.Map({
        container: 'map',
        style: 'mapbox://styles/mapbox/dark-v9',
        center: [-122.325769,47.620057],
        zoom: 10.1
});

var mapId = "wangbeiqi199159.cj2qzjlcd001p2wmxq7lm6ocm-0cldu"

map.on('load', function () {


        map.addLayer({
            'id': 'entire_fills',
            'type': 'circle',
            'source': {
                type: "geojson",
                data: listings
            },
            'paint': {
                'circle-radius': {
                    'base': 1.75,
                    'stops': [[12, 2.3], [20, 200]]
                },

                'circle-color': "#fbb03b",
                "circle-opacity": 0.85,
            },
            "filter": ["==", "room_type", "Entire home/apt"]
        });

        map.addLayer({
            'id': 'private_fills',
            'type': 'circle',
            'source': {
                type: "geojson",
                data: listings
            },
            'paint': {
                'circle-radius': {
                    'base': 1.75,
                    'stops': [[12, 2.3], [20, 200]]
                },

                'circle-color': "#3bb2d0",
                "circle-opacity": 0.85,
            },
            "filter": ["==", "room_type", "Private room"]
        });

        map.addLayer({
            'id': 'shared_fills',
            'type': 'circle',
            'source': {
                type: "geojson",
Column 1
```

Folium(py lib):

```
import folium
import pandas as pd
# San Seattle latitude and longitude values
latitude = 47.620057
longitude = -122.325769
# Create map and display it
sea_map = folium.Map(location=[latitude, longitude], zoom_start=10.1)
cdata = pd.read_csv('../data/listings.csv')
data_entire = cdata[cdata.room_type == 'Entire home/apt']
limit = 1000
data_entire_limit = data_entire.iloc[0:limit, :]
data_private = cdata[cdata.room_type == 'Private room']
data_private_limit = data_private.iloc[0:limit, :]
data_others = cdata[(cdata.room_type != 'Entire home/apt') & (cdata.room_type != 'Private room')]
data_others_limit = data_others.iloc[0:limit, :]
sea_map = folium.Map(location=[latitude, longitude], zoom_start=10.1)


for lat, lng, in zip(data_others_limit.latitude, data_others_limit.longitude):
        folium.CircleMarker(
            location = [lat, lng],
            radius=0.5,
            color='red',
        ).add_to(sea_map)

for lat, lng, in zip(data_private_limit.latitude, data_private_limit.longitude):
        folium.CircleMarker(
            location = [lat, lng],
            radius=0.5, # define how big you want the circle markers to be
            color='blue',
        ).add_to(sea_map)
for lat, lng, in zip(data_entire_limit.latitude, data_entire_limit.longitude):
        folium.CircleMarker(
            location = [lat, lng],
            radius=0.5, # define how big you want the circle markers to be
            color='yellow',
        ).add_to(sea_map)
sea_map
```

Demo:

MapBoxGL
Folium

Demos:

	Folium	**MapBox**
Pros	Python package is easier to use for back-end developer. Folium can read dataframe directly. There is no need to transfer data format.	MapBox can run on JavaScript environment, easy to implement for front end developer. It also saves space because it create map in HTML on MapBox server.
Cons	To present on web page, it has to save map into local disk, cost more disk space. It not easy to implement for front end developer.	JavaScript is more complicated to code compared to python.

Use case 3

Recommend room price for landlord

- Database with price of houses around it
- User interface that allows users to input the info about the room to improve the accuracy of predicted price
- Map view that allows users to visualize all houses or the similar rooms around it

Demos:

DEMO: Pandas/ sklearn/matplotlib

Code:

In [9]:
```
basic_features = ['bedrooms', 'bathrooms', 'sqft_living',
'sqft_lot', 'floors', 'zipcode']
```

In [10]:
```
advanced_features = basic_features + [
    'condition',      # condition of the house
    'grade',          # measure of qality of construction
    'waterfront',     # waterfront property
    'view',           # type of view
    'sqft_above',     # square feet above ground
    'sqft_basement',  # square feet in basementab
    'yr_built',       # the year built
    'yr_renovated',   # the year renovated
    'lat',            # the longitude of the parcel
    'long',           # the latitide of the parcel
    'sqft_living15',  # average sq.ft. of 15 nearest neig
hbors
    'sqft_lot15',     # average lot size of 15 nearest ne
ighbors
]
```

Demos:

DEMO: Pandas/ sklearn/matplotlib

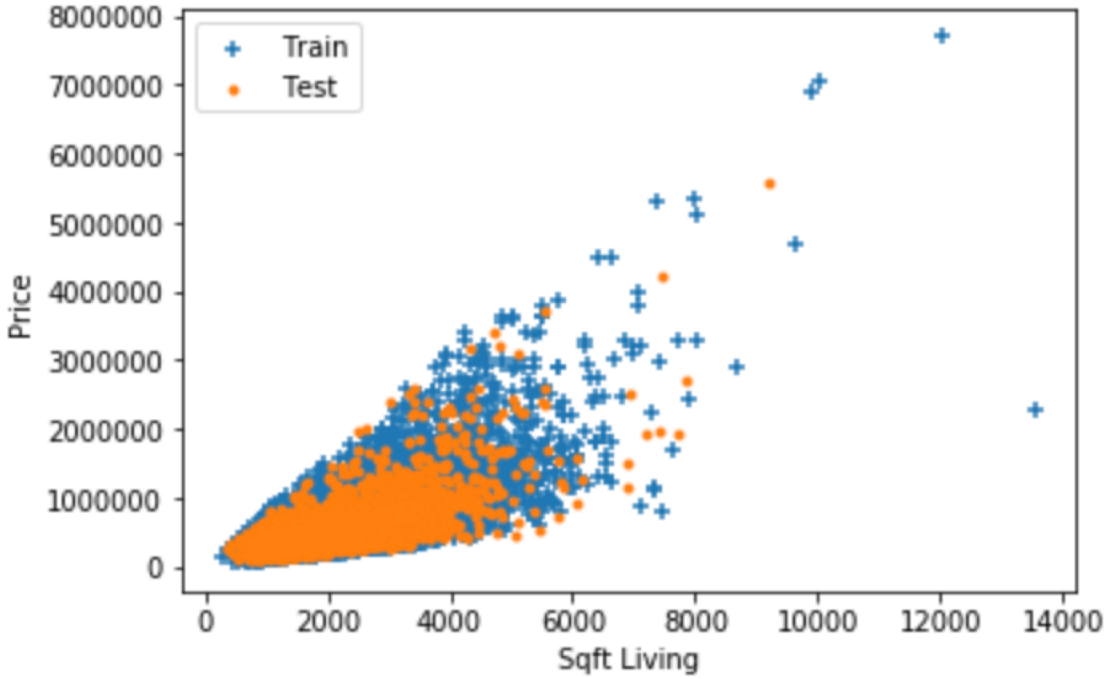
Code:

In [11]:
```
from sklearn import datasets, linear_model
regr = linear_model.LinearRegression()
# Train the model using the training sets
basic_model=regr.fit(train_data[basic_features], train_da
ta['price'])
advanced_model=regr.fit(train_data[advanced_features], tr
ain_data['price'])
```

Demos:

DEMO: Pandas/ sklearn/matplotlib

Output:



Demos:

1. DEMO1: Pandas/ sklearn/matplotlib
2. DEMO2: MapboxGL
3. DEMO3: MapBox