Yelp Geospatial Location Analysis for Restaurants

August 30, 2019

1 LOCATION ANALYSIS OF RESTAURANTS

PROPERTY VALUE, POPULATION DEMOGRAPHICS, PROXIMITY OF INFLUENCE

LOCATION, LOCATION, LOCATION - > INFLUENTIAL TO THE SUCCESS OF THE RESTAURANT?

[ ]:
```
import numpy as np
import pandas as pd
import json
import csv
import os
import matplotlib.pyplot as plt
import seaborn as sns
import datetime
import nltk
from nltk.corpus import stopwords
import re
from collections import Counter
from sklearn.feature_extraction.text import CountVectorizer
#conda install conda-forge wordcloud
from wordcloud import WordCloud
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix,␣
 ↪accuracy_score
#can't install new tensorflow cuz of macos so use an older version
# import tensorflow as tf
# from tensorflow import keras
import geopandas
import descartes
import requests
import urllib
from shapely.geometry import Point
# from arcgis.gis import GIS
```

2 Phoenix Villages/Neighborhoods

Ahwatukee, Alhambra, Camelback East, Central City, Deer Valley, Desert View, Encanto, Estrella, Laveen, Maryvale, North Gateway, Paradise Valley, Rio Vista, South Mountain

[ ]:
```
import geopandas
import descartes
arizona = geopandas.read_file('Villages/Villages.shp')
arizona.geometry = arizona.geometry.to_crs(epsg = 4326)
plt.figure(figsize=(30,30))
ax = arizona.plot(column='NAME', cmap='Set2')
```

[ ]:
```
#coordinate reference system
crs = {'init': 'epsg:2868'}
#create a point geometry column using Shapely Point
arizonafoodbusiness['geometry'] = arizonafoodbusiness.apply(
    lambda h: Point(float(h['longitude']), float(h['latitude'])), axis = 1)
GEO_AZ= geopandas.GeoDataFrame(arizonafoodbusiness, crs = crs, geometry =␣
 ↪arizonafoodbusiness['geometry'])
```

[ ]:
```
for n in arizona['NAME']:
    neighborhood = arizona[arizona['NAME'] == n]
    ax = neighborhood.plot(column='NAME', cmap='Set2', legend = True)
    plt.xlim(plt.xlim())
    plt.ylim(plt.ylim())
    sns.scatterplot(x='longitude',y= 'latitude',hue='stars', style= 'is_open',␣
 ↪data = GEO_AZ, label = n)
    plt.grid()
```

3 Proximity of Restaurants to each other

-restaurant near amenity?

Does being located near a poor reviewed restaurant affect your own restaurant?

knn

Are negatively reviewed restaurants clustered together? What algorithms do we use to figure that out? distance, nearest neighbor search, k-means

Find the 3 nearest neighbors and conclude whether the 3 nearest neighborhoods are positive and negative. If all three are the same as like the point of interest, then it's 100%. Do that for all points. Then, take the average %

[ ]:
```

```

4 Adding Demographics to the Location

variables = income level, gentrified status

[ ]:
```
LIMICensusTract = ['083000', '082028', '082018',
                   '082010', '082007', '1125.09',
```

```
                    '112508', '112503', '109702',
                    '109601', '109704', '109706',
                    '109703', '109701', '109602',
                    '109604', '112505', '112502','112504',
                    '109801', '109820', '109500', '109400', '109900',
                    '112401', '112402', '112507', '112512', '112513',
                    '109300', '110002', '112302', '112301',
                     '093104', '093101', '093002', '093001',
                     '105702'
                     '107000', '107101', '107102', '109200', '110100',␣
 ↪'112202', '112201',
                     '112601', '112602', '114600',
                     '115500',
                     '105602', '105601', '105900', '106900', '107202',
                     '107201', '109101', '109102', '116900',␣
 ↪'112100','112700',
                     '114500', '114703', '117300', '116607', '115600',␣
 ↪'115700',
                     '105501', '105501', '105502', '106001',␣
 ↪'106002','106003',
                     '106801', '106802', '107300', '109001', '109002',␣
 ↪'109003', '117000',
                     '116800', '114401', '114402',
                     '107400', '108902', '108901','110400', '112900',␣
 ↪'114301', '114302','114800', '116602',
                     '108802', '110501', '110502', '113000', '113100',␣
 ↪'114100', '114200', '114900', '115400', '115802', '115801', '116500', '116702',
                     '106502', '107601', '108602', '108601', '111700',␣
 ↪'113203', '113201', '113202', '114000', '117200', '115300', '115900',␣
 ↪'116000','116400',
                     '110701', '110702', '111602', '111601', '113300',␣
 ↪'113900', '116000', '116100', '115200', '116204', '116205', '319703',␣
 ↪'113801', '113502', '113501', '111501', '111502', '110801',
                     '110901', '110902', '111401', '111402', '111502',␣
 ↪'111501', '113501', '113502', '113602', '113601', '113700', '111300',␣
 ↪'111203', '111202', '111201', '111204', '320100',
                     '105200', '104701', '104702', '104501', '104502',␣
 ↪'104401', '104302', '103615', '103609', '618800' ,'619100', '619200',␣
 ↪'619300', '619400',
                      '103305', '103306', '103304', '614700']
gentrifiedtracts = ['116703', '116602','116000', '116800', '117000',␣
 ↪'110400','112900', '114301', '114200',
                  '114000', '113202', '113203', '113801', '113700', '110702',␣
 ↪'110802', '118602', '619100', '619400', '614700']
```

[ ]:
```
%%time
import requests
import urllib
def getcensustract(lat, lon):
    params = urllib.parse.urlencode({'latitude': lat, 'longitude':lon, 'format':
 ↪'json'})
    url = 'https://geo.fcc.gov/api/census/block/find?' + params
    response = requests.get(url)
    data = response.json()
    return data['Block']['FIPS']
# arizonafoodbusiness['censustract'] = arizonafoodbusiness.apply(lambda x:␣
 ↪getcensustract(x['latitude'], x['longitude']), axis = 1)
# arizonafoodbusiness['latitude'], arizonafoodbusiness['longitude']
```

[ ]:
```
%%time
latlonset = set(zip(round(arizonafoodbusiness['latitude'],3).values,␣
 ↪round(arizonafoodbusiness['longitude'],3).values))
censustractdict ={}
num = 0
for k in latlonset:
    print(num)
    censustractdict[k] = getcensustract(k[0],k[1])
    num +=1
```

[ ]:
```
%%time
arizonafoodbusiness['censustract'] = arizonafoodbusiness.apply(lambda x:␣
 ↪censustractdict.get((round(x['latitude'],3), round(x['longitude'],3))), axis =␣
 ↪1)
```

[ ]:
```
arizonafoodbusiness['shortcensustract'] = arizonafoodbusiness['censustract'].
 ↪apply(lambda h: h[5:11] if h is not None else None)
```

[ ]:
```
def income(tract):
    if tract in gentrifiedtracts:
        return 'Gentrified'
    elif tract in LIMICensusTract:
        return 'LIMI'
    else:
        return None
arizonafoodbusiness['Income Level'] = arizonafoodbusiness['shortcensustract'].
 ↪apply(lambda x: income(x) if x is not None else None)
```

[ ]:
```
arizonafoodbusiness['Gentrified status'] =␣
 ↪arizonafoodbusiness['shortcensustract'].apply(lambda x: 'GENTRIFIED' if x in␣
 ↪gentrifiedtracts else 'NOT GENTRIFIED')
```

[ ]:

[ ]:

[ ]:

[ ]:

[ ]: