Yelp Regression Analysis on Measurable Metrics

August 31, 2019

1 Yelp Regression Analysis on Measurable Metrics

Measurable Metrics include total of check ins , average rating, total number of reviews, total number of positive reviews, total number of negative reviews

[1]:
```
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
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
```

[ ]:
```
%%time
# Reading the business dataset
# columns include address, attributes, business_id,
# categories, city, hours, is_open, latitude, longitude,
# name, postal_code, review_count, stars, state
```

```
business = pd.read_csv('yelp_business.csv')
checkin = pd.read_csv('yelp_checkin.csv')
user = pd.read_csv('yelp_user.csv')
review = pd.read_csv('yelp_review.csv')

# Found data on outside of US. We should exclude international businesses.

stateinitials = ['AL', 'AK', 'AZ', 'AR', 'CA',
                 'CO', 'CT', 'DE', 'FL', 'GA', 'HI',
                 'ID', 'IL', 'IN', 'IA', 'KS', 'KY', 'LA',
                 'ME', 'MD', 'MA', 'MI', 'MN', 'MS',
                 'MO', 'MT', 'NE', 'NV', 'NH', 'NJ', 'NM',
                 'NY', 'NC', 'ND', 'OH', 'OK', 'OR', 'PA',
                 'RI', 'SC', 'SD', 'TN', 'TX', 'UT', 'VT',
                 'VA', 'WA', 'WV', 'WI', 'WY']
# Filter for only USA LOCATIONS
usbusiness = business[business['state'].isin(stateinitials)]
# Filter for Food and Restaurants Including Categories

foodbusiness = usbusiness[usbusiness['categories'].apply(lambda x: 'Food' in␣
 ↪str(x) or 'Restaurant' in str(x))]

#number of businesses in the food section
#AZ = 13826, NV = 9263, OH = 6031, NC = 4969
# foodbusiness['state'].value_counts()

# the food section includes supermarkets and nonrestaurant food related␣
 ↪businesses
nonrestaurantlist = ['Shopping', 'Shopping,',
                     'Services', 'Groceries', 'Event Planning',␣
 ↪'Convenience','Convenience Stores',
                    'Gas Station', 'Gas', 'Grocery', 'Station', 'Banks,']
# filter out sole food serving places
foodbusiness = foodbusiness[foodbusiness['categories'].apply(
    lambda x: not any(s in nonrestaurantlist for s in str(x).split(';')))]

#change categories into a list not a string
foodbusiness['categorieslist'] = foodbusiness['categories'].apply(lambda x:␣
 ↪str(x).split(';'))
# # number of food businesses for each state
# # AZ 12331, NV 8399, OH = 5535
# foodbusiness['state'].value_counts()

#selecting only AR businesses
arizonafoodbusiness = foodbusiness[foodbusiness['state'] == 'AZ']
#selecting only NV businesses
nevadafoodbusiness = foodbusiness[foodbusiness['state'] == 'NV']
```

```
restaurantgroups = ['Afghan',
'African', 'Senegalese', 'South African', 'American (New)', 'American␣
 ↪(Traditional)',
'Arabian','Argentine','Armenian','Asian Fusion','Australian','Austrian',
'Bangladeshi','Barbeque','Basque','Belgian','Brasseries','Brazilian',
'Breakfast & Brunch','Pancakes','British,Buffets',
'Bulgarian','Burgers','Burmese','Cafes','Themed Cafes','Cafeteria','Cajun/
 ↪Creole','Cambodian','Caribbean','Dominican','Haitian','Puerto␣
 ↪Rican','Trinidadian','Catalan','Cheesesteaks','Chicken Shop','Chicken␣
 ↪Wings','Chinese',
'Cantonese','Dim Sum','Hainan','Shanghainese','Szechuan','Comfort Food',
'Creperies','Cuban','Czech','Delis','Diners','Dinner␣
 ↪Theater','Eritrean','Ethiopian','Fast Food','Filipino','Fish &␣
 ↪Chips','Fondue','Food Court','Food␣
 ↪Stands','French','Mauritius','Reunion','Game Meat','Gastropubs','Georgian',
'German','Gluten-Free','Greek','Guamanian','Halal','Hawaiian','Himalayan/
 ↪Nepalese',
'Honduran','Hong Kong Style Cafe','Hot Dogs','Hot␣
 ↪Pot','Hungarian','Iberian','Indian','Indonesian','Irish','Italian','Calabrian','Sardinian','S
 ↪Belt Sushi','Izakaya','Japanese Curry','Ramen','Teppanyaki',
'Kebab','Korean','Kosher','Laotian','Latin␣
 ↪American','Colombian','Salvadoran','Venezuelan','Live/Raw␣
 ↪Food','Malaysian','Mediterranean','Falafel','Mexican',
'Tacos','Middle Eastern','Egyptian','Lebanese','Modern European','Mongolian',
'Moroccan','New Mexican Cuisine','Nicaraguan','Noodles','Pakistani','Pan␣
 ↪Asian','Persian/Iranian','Peruvian','Pizza','Polish','Polynesian','Pop-Up␣
 ↪Restaurants','Portuguese','Poutineries','Russian','Salad','Sandwiches','Scandinavian','Scotti
 ↪Food','Soup','Southern','Spanish','Sri Lankan','Steakhouses','Supper␣
 ↪Clubs','Sushi Bars','Syrian','Taiwanese','Tapas Bars',
'Tapas/Small␣
 ↪Plates','Tex-Mex','Thai','Turkish','Ukrainian','Uzbek','Vegan','Vegetarian','Vietnamese','Waf
'Wraps']

arizonafoodbusiness['maincuisine'] = arizonafoodbusiness['categorieslist'].apply(
    lambda x: [a for a in x if a in restaurantgroups]).apply(
    lambda h: h[0] if len(h) != 0 else np.nan)

# filter to include only the business id of the food business
foodcheckin = checkin[checkin['business_id'].isin(foodbusiness['business_id'])]
# filter only arizona food businesses
arizonafoodcheckin = checkin[checkin['business_id'].
 ↪isin(arizonafoodbusiness['business_id'])]
# sum of checkins per business for food business
sumfoodcheckin = foodcheckin.groupby('business_id').agg(
    {'checkins': 'sum'}).reset_index()
```

```
sumfoodcheckin = dict(zip(sumfoodcheckin['business_id'],
                          sumfoodcheckin['checkins']))
#adding it to the az and food df
arizonafoodbusiness['sumcheckin'] = arizonafoodbusiness['business_id'].
 ↪map(sumfoodcheckin)
foodbusiness['sumcheckin']= foodbusiness['business_id'].map(sumfoodcheckin)

# change the date into year, day
review['dateobject'] = review['date'].apply(lambda x: datetime.datetime.
 ↪strptime(x, '%Y-%m-%d'))
review['year' ] = review['dateobject'].apply(lambda h: h.year)
review['day'] = review['dateobject'].apply(lambda h: h.weekday())

# #include reviews of only food businesses in our study
foodreviews = review[review['business_id'].isin(foodbusiness['business_id'])]
#include reviews of only food businesses in Arizona
arizonafoodreviews = review[review['business_id'].
 ↪isin(arizonafoodbusiness['business_id'])]

# include reviews of only food businesses in Nevada
nevadafoodreviews = review[review['business_id'].
 ↪isin(nevadafoodbusiness['business_id'])]

# food reviews from users living in Arizona
userarizona = arizonafoodreviews['user_id'].unique()

#business id to name
businessidtoname = dict(zip(arizonafoodbusiness['business_id'],␣
 ↪arizonafoodbusiness['name']))
#business name to lon
businessidtolong = dict(zip(arizonafoodbusiness['business_id'],␣
 ↪arizonafoodbusiness['longitude']))
#business name to lat
businessidtolat = dict(zip(arizonafoodbusiness['business_id'],␣
 ↪arizonafoodbusiness['latitude']))

# add business name to AZ food reviews
arizonafoodreviews['business_name'] = arizonafoodreviews['business_id'].
 ↪map(businessidtoname)

#add lowername to the AZ food businesses
arizonafoodbusiness['lowername'] = arizonafoodbusiness['name'].apply(lambda x: x.
 ↪replace("'", "").lower())

#add set of lon and lat to AZ food reviews
```

```
arizonafoodreviews['lon'] = arizonafoodreviews['business_id'].
 ↪map(businessidtolong)
arizonafoodreviews['lat'] = arizonafoodreviews['business_id'].
 ↪map(businessidtolat)


# users profile from Arizona
arizonauserprofile = user['user_id'].isin(userarizona)
# elite users on Yelp
eliteusers = user[user['elite'] != 'None']
#getting the start year for yelping since
user['startyear'] = user['yelping_since'].apply(lambda x: x[0:4])

#remove outlier - gZGsReGOVeX4uKViHTB9EQ in Arizona
arizonafoodbusiness = arizonafoodbusiness[arizonafoodbusiness['business_id'] !=␣
 ↪'gZGsReGOVeX4uKViHTB9EQ']

# checkin values of the arizona food business
arizonafoodbusiness['sumcheckin'] = arizonafoodbusiness['business_id'].
 ↪map(sumfoodcheckin)

closedazbusiness = arizonafoodbusiness[arizonafoodbusiness['is_open'] == 0]
openazbusiness = arizonafoodbusiness[arizonafoodbusiness['is_open'] == 1]


#labeling review positive, negative, or neutral
def positiveornegative(star):
    if star > 3:
        return 'positive'
    elif star < 3:
        return 'negative'
    else:
        return 'neutral'
arizonafoodreviews['label'] = arizonafoodreviews['stars'].map(positiveornegative)

import gc
gc.collect()

#the code goes 70x faster when it's cached right here than to be called␣
 ↪repeatedly
stopw = stopwords.words('english')
# remove punctuations, lowercase,
#remove numbers r'\d+
#remove stop words, and split string into a list of words
arizonafoodreviews['clean text'] = arizonafoodreviews['text'].apply(
    lambda word: re.sub(r'\d+', '', re.sub(r'[^\w\s]', '', word.replace(
    '\n', ' '))))).apply(
```

```
    lambda word: [w for w in word.lower().split(' ') if w not in stopw and w !=␣
 ↪''])

positiveazfoodreviews = arizonafoodreviews[arizonafoodreviews['label'] ==␣
 ↪'positive']
negativeazfoodreviews = arizonafoodreviews[arizonafoodreviews['label'] ==␣
 ↪'negative']
```

2 Create new features for Regression

[3]:
```
# create new features
def pospercentage(business_id):
    data = arizonafoodreviews[arizonafoodreviews['business_id'] == business_id]
    positivecount = np.count_nonzero(data['stars'] > 3)
    total = data.shape[0]
    return positivecount/total * 100
arizonafoodbusiness['positive %'] = arizonafoodbusiness['business_id'].
 ↪map(pospercentage)
```

[4]:
```
def negpercentage(business_id):
    data = arizonafoodreviews[arizonafoodreviews['business_id'] == business_id]
    negcount = np.count_nonzero(data['stars'] < 3)
    total = data.shape[0]
    return negcount/total * 100
arizonafoodbusiness['negative %'] = arizonafoodbusiness['business_id'].
 ↪map(negpercentage)
```

[5]:
```
arizonafoodbusiness['totalreviews'] = arizonafoodbusiness['business_id'].
 ↪map(arizonafoodreviews['business_id'].value_counts())
arizonafoodbusiness['sumcheckin'] = arizonafoodbusiness['sumcheckin'].fillna(0.0)
```

[6]:
```
X = arizonafoodbusiness[['negative %', 'positive %', 'totalreviews',␣
 ↪'sumcheckin']]
X = arizonafoodbusiness[['sumcheckin']]
Y = arizonafoodbusiness['stars'].values
```

what are we predicting? What is Y? We will set stars as Y.

[7]:
```
from sklearn.metrics import accuracy_score # only works for classification
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.linear_model import LinearRegression
# model = LinearRegression()
# X_train, X_test, y_train, y_test =train_test_split(X, Y, test_size = 0.33,␣
 ↪random_state = 23)

# model.fit(X_train, y_train)
# predictions = model.predict(X_test)
# mean_squared_error(predictions, y_test)
```

[8]:
```
def linear_regression(xvariables, yvariables):
    X = arizonafoodbusiness[xvariables]
    Y = arizonafoodbusiness[yvariables].values
    model = LinearRegression()
    X_train, X_test, y_train, y_test =train_test_split(X, Y, test_size = 0.33,␣
 ↪random_state = 23)

    model.fit(X_train, y_train)
    predictions = model.predict(X_test)
    print(xvariables)
    return 'R2: ', r2_score(y_test,predictions),'Coefficients: ', model.coef_,␣
 ↪'Mean Squared error: ', mean_squared_error(y_test, predictions)
```

[9]:
```
linear_regression(['positive %', 'negative %'], 'stars')
```

```
['positive %', 'negative %']
```

[9]:
```
('R2: ',
 0.9417972605797,
 'Coefficients: ',
 array([ 0.01975644, -0.01540727]),
 'Mean Squared error: ',
 0.040868623396011716)
```

[10]:
```
linear_regression(['totalreviews'], 'stars')
```

```
['totalreviews']
```

[10]:
```
('R2: ',
 0.04078358613119293,
 'Coefficients: ',
 array([0.00120258]),
 'Mean Squared error: ',
 0.6735396780998308)
```

[11]:
```
linear_regression(['sumcheckin'], 'stars')
```

```
['sumcheckin']
```

[11]:
```
('R2: ',
 0.017372009082638695,
 'Coefficients: ',
 array([0.00019648]),
 'Mean Squared error: ',
 0.6899787484087855)
```

[12]:
```
linear_regression(['totalreviews', 'sumcheckin'], 'stars')
```

```
['totalreviews', 'sumcheckin']
```

[12]:
```
('R2: ',
 0.04357573037103435,
 'Coefficients: ',
 array([ 0.00166164, -0.00012627]),
 'Mean Squared error: ',
 0.6715790987088611)
```

[13]:
```
linear_regression(['positive %'], 'stars')
```

```
['positive %']
```

[13]:
```
('R2: ',
 0.9074331280153802,
 'Coefficients: ',
 array([0.03254211]),
 'Mean Squared error: ',
 0.06499832598543956)
```

[14]:
```
linear_regression(['negative %'], 'stars')
```

```
['negative %']
```

[14]:
```
('R2: ',
 0.8824511985510124,
 'Coefficients: ',
 array([-0.03492727]),
 'Mean Squared error: ',
 0.0825400616005313)
```

[15]:
```
linear_regression(['sumcheckin'], 'totalreviews')
```

```
['sumcheckin']
```

[15]:
```
('R2: ',
 0.6498637039709836,
 'Coefficients: ',
 array([0.19423617]),
 'Mean Squared error: ',
 6343.905009066146)
```

[16]:
```
linear_regression(['sumcheckin'], 'positive %')
```

```
['sumcheckin']
```

[16]:
```
('R2: ',
 0.01821369904892267,
 'Coefficients: ',
 array([0.00572923]),
```

```
'Mean Squared error: ',
589.6532373321794)
```

[ ]: