DATA MINING & ANALYTICS (2023)

Make sure you fill in any place that says `YOUR CODE HERE` or `YOUR ANSWER HERE`, as well as your name below:

```
NAME = "Jeonghyun Lee"
```

Lab 1 - Data Preprocessing

Data transformations are useful for preparing a dataset for answering a particular question. Part of this process involves generating features from the dataset you find relevant to the question at hand. For this lab, we will be using a Yelp reviews dataset. Each row in the dataset depicts one review along with the features of the review (the reviewer, the review text, etc.). The goal of this lab is to eventually convert this reviews dataset into a *reviewers* dataset by creating different features describing each reviewer.

The submission for this assignment should be done *individually*, but you are allowed to work in groups of 2.

Google Colab

Colab is a free online platform provided by Google that allows you to execute python code without any installations on your local machine. Without Colab (using Jupyter notebooks or the command line), you would have to install various packages and manage dependencies.

In Colab, you can simply import them, or even install them (for that particular session). Colab can be accessed at the link: https://colab.research.google.com

IMPORTANT: This lab has been shared with only read permissions to you. Make sure to click File -> "Save a Copy in Drive" so that you can get your own copy that WILL SAVE YOUR PROGRESS in your own Colab environment.

If you download the .ipynb and want to further edit the notebook, you will need to make sure you have Jupyter installed locally so you can view the notebook properly (not as a JSON file).

Environment Setup

Run this cell to setup your environment.

```
# Importing libraries
import numpy as np
import pandas as pd
import math
import os
print('Libraries Imported')
```

```
#DOWNLOADING DATASET IF NOT PRESENT
```

```
!wget -nc http://askoski.berkeley.edu/~zp/yelp_reviews.csv

#!unzip yelp_reviews.zip
print('Dataset Downloaded: yelp_reviews.csv')
df=pd.read_csv('yelp_reviews.csv')
print(df.head())

print('Setup Complete')
```

```
Libraries Imported
--2023-09-06 06:33:57--  http://askoski.berkeley.edu/~zp/yelp_reviews.csv
Resolving askoski.berkeley.edu (askoski.berkeley.edu)... 169.229.192.179
Connecting to askoski.berkeley.edu (askoski.berkeley.edu)|169.229.192.179|:
HTTP request sent, awaiting response... 200 OK
Length: 376638166 (359M) [text/csv]
Saving to: ‘yelp_reviews.csv’

yelp_reviews.csv    100%[===================>] 359.19M  38.3MB/s    in 9.8s

2023-09-06 06:34:07 (36.5 MB/s) - ‘yelp_reviews.csv’ saved [376638166/37663

Dataset Downloaded: yelp_reviews.csv
     type             business_id                 user_id  stars  \
0  review  mxrXVZWc6PWk81gvOVNOUw  mv7shusL4Xb6TylVYBv4CA      4
1  review  mxrXVZWc6PWk81gvOVNOUw  0aN5QPhs-VwK2vusKG0waQ      5
2  review  kK4AzZ0YWI-U2G-paAL7Fg  0aN5QPhs-VwK2vusKG0waQ      5
3  review  mxrXVZWc6PWk81gvOVNOUw  1JUwyYab-uJzEx_FRd81Zg      5
4  review  mxrXVZWc6PWk81gvOVNOUw  2Zd3Xy8hUVmZkNg7RyNjhg      4

                                                text        date  cool_vote
0  Definitely try the duck dish.   I rank it amon...  2011-06-13
1  Big Ass Burger was awesome! Great $5 mojitos. ...  2011-06-25
2             Unbelievable sandwiches! Good service.  2011-06-25
3  Awesome, awesome, awesome! My mom and sister a...  2011-07-18
4  I had the ribs they were great.  The beer sele...  2011-07-19

   useful_votes  funny_votes
0             0            0
1             0            0
2             0            0
3             1            0
4             0            1
Setup Complete
```

Q1: What was the highest number of reviews for any one `business_id`?

- For this task, we will need to group the reviews dataset by `business_id`. This will aggregate data for each business, which is what we need for this task. This can be done using the groupby method. Some pointers of how you could go about this question are listed below:
  - `yelp_businesses = yelp_dataset.groupby('business_id').size()`
  - The `.size()` function counts the number of instances for each `business_id`, which gives us the number of reviews as each instance in this dataset is a review.
  - The following command will sort this list, after which you can take note of the highest value: `sorted_yelp_businesses = yelp_businesses.sort_values(ascending=False, inplace=False)`
  - This approach allows you to see the data structure being used in the sort. A quicker approach to getting the max would be to use the max function: `max(yelp_businesses)`

```
#Make sure you return the answer value in this function
def q1(df):

    # YOUR CODE HERE
    new_business_id = df.groupby('business_id').size()
    max_business_id = max(new_business_id)
    return max_business_id

    raise NotImplementedError()
```

```
#This is a graded cell, do not edit
print(q1(df))
```

```
4128
```

Q2: On average, how many reviews did each business get?

```
#Make sure you return the answer value in this function
def q2(df):
    new_business_id = df.groupby('business_id').size()
    average_business_id = np.mean(new_business_id)
    return average_business_id
    # YOUR CODE HERE
    raise NotImplementedError()
```

```
#This is a graded cell, do not edit
print(q2(df))
```

```
12.63413902163123
```

Q3: What is the average number of reviews per reviewer?

```
#Make sure you return the answer value in this function
def q3(df):

    new_user_id = df.groupby('user_id').size()
    average_user_id = np.mean(new_user_id)
    return average_user_id

    # YOUR CODE HERE
    raise NotImplementedError()
```

```
#This is a graded cell, do not edit
print(q3(df))
```

```
3.188511934933203
```

Q4: Calculate the total number of cool votes per reviewer, then average these totals across reviewers.

```
#Make sure you return the answer value in this function
def q4(df):

    num_cool = df[['user_id', 'cool_votes']]
    num_cool = num_cool.groupby('user_id').sum()
    return np.mean(num_cool)

    # YOUR CODE HERE
    raise NotImplementedError()
```

```
#This is a graded cell, do not edit
print(q4(df))
```

```
cool_votes    1.241728
dtype: float64
/usr/local/lib/python3.10/dist-packages/numpy/core/fromnumeric.py:3430: Fut
  return mean(axis=axis, dtype=dtype, out=out, **kwargs)
```

Q5: Calculate the total number of funny votes per reviewer, then average these totals across reviewers.

```
#Make sure you return the answer value in this function
def q5(df):

    num_funny = df.groupby('user_id')['funny_votes'].sum()
    return np.mean(num_funny)

    # YOUR CODE HERE
    raise NotImplementedError()
```

```
#This is a graded cell, do not edit
print(q5(df))
```

```
1.10126486404605
```

Q6: Calculate the total number of useful votes each business get, then average these totals across business_ids.

```
#Make sure you return the answer in this function
def q6(df):

    num_useful = df.groupby('business_id')['useful_votes'].sum()
    return np.mean(num_useful)

    # YOUR CODE HERE
    raise NotImplementedError()
```

```
#This is a graded cell, do not edit
print(q6(df))
```

```
9.844472147193942
```

Q7: On average, what percentage of a reviewer's votes are cool votes?

(hint1: calculate the percentage of cool votes for each reviewer, then average this percentage across reviewers)

(hint2: you should discard reviewers who have absolutely no votes - from cool, funny, or useful votes - from your calculation)

```
#Make sure you return the answer in this function
#Remember to multiply by 100 for percentages
def q7(df):
  df['total_votes'] = df['cool_votes'] + df['funny_votes'] + df['useful_votes']
  reviewer_votes = df.groupby('user_id').agg({
        'cool_votes': 'sum',
        'funny_votes': 'sum',
        'useful_votes': 'sum',
        'total_votes': 'sum'
    })


  cool_votes = reviewer_votes[reviewer_votes['total_votes'] > 0]
  cool_votes['cool_percentage'] = (cool_votes['cool_votes'] / cool_votes['total_


  return cool_votes['cool_percentage'].mean()

  # YOUR CODE HERE
  raise NotImplementedError()
```

```
#This is a graded cell, do not edit
#Remember to multiply by 100 for percentages
print(round(q7(df),2))
```

```
    19.27
    <ipython-input-23-520532670c80>:14: SettingWithCopyWarning:
    A value is trying to be set on a copy of a slice from a DataFrame.
    Try using .loc[row_indexer,col_indexer] = value instead

    See the caveats in the documentation: https://pandas.pydata.org/pandas-docs
      cool_votes['cool_percentage'] = (cool_votes['cool_votes'] / cool_votes['t
```

Q8: On average, what percentage of a reviewer's votes are funny votes?

(hint1: calculate the percentage of funny votes for each reviewer, then average this percentage across reviewers)

(hint2: you should discard reviewers who have zero total votes from your calculation)

```
#Make sure you return the answer in this function
#Remember to multiply by 100 for percentages
def q8(df):
    df['total_votes'] = df['cool_votes'] + df['funny_votes'] + df['useful_votes'
    reviewer_votes = df.groupby('user_id').agg({
        'cool_votes': 'sum',
        'funny_votes': 'sum',
        'useful_votes': 'sum',
        'total_votes': 'sum'
    })

    funny_votes = reviewer_votes[reviewer_votes['total_votes'] > 0]
    funny_votes['funny_percentage'] = (funny_votes['funny_votes'] / funny_votes[

    return funny_votes['funny_percentage'].mean()
    # YOUR CODE HERE
    raise NotImplementedError()
```

```
#This is a graded cell, do not edit
print(round(q8(df),2))
```

```
18.26
<ipython-input-25-6453eb377085>:13: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs
  funny_votes['funny_percentage'] = (funny_votes['funny_votes'] / funny_vot
```

Q9: On average, what percentage of a reviewer's votes are useful votes?

(hint1: calculate the percentage of useful votes for each reviewer, then average this percentage across reviewers)

(hint2: you should discard reviewers who have zero total votes from your calculation)

```
#Make sure you return the answer in this function
def q9(df):

    df['total_votes'] = df['cool_votes'] + df['funny_votes'] + df['useful_votes'
    reviewer_votes = df.groupby('user_id').agg({
        'cool_votes': 'sum',
        'funny_votes': 'sum',
        'useful_votes': 'sum',
        'total_votes': 'sum'
    })


    useful_votes = reviewer_votes[reviewer_votes['total_votes'] > 0]
    useful_votes['useful_percentage'] = (useful_votes['useful_votes'] / useful_v


    return useful_votes['useful_percentage'].mean()
    # YOUR CODE HERE
    raise NotImplementedError()
```

```
#This is a graded cell, do not edit
print(round(q9(df),2))
```

```
62.47
<ipython-input-27-684b624337bf>:14: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs
  useful_votes['useful_percentage'] = (useful_votes['useful_votes'] / usefu
```

Q10: Find the average review text length (in non-space characters).

```
#Make sure you return the answer in this function
def q10(df):
    average_length = df['text'].str.replace(' ', '').str.len()
    return average_length.mean()
    # YOUR CODE HERE
    raise NotImplementedError()
```

```
#This is a graded cell, do not edit
print(round(q10(df),0))
```

```
499.0
```

Q11: Find the year in which each reviewer wrote the most reviews. Once you have this for each reviewer, subtract the minimum possible year (2004) from each year so that your final feature values are 0, 1, 2, etc.

Note: we are looking for the `answer` to be in the format of a Pandas Series with `user_id` as the index and the year (in 0, 1, 2 format as listed above) as the value.

```
def find_most_reviews_year(df):
    df['date'] = pd.to_datetime(df['date'], errors='coerce')
    df['year'] = df['date'].dt.year

    yearly_reviews = df.groupby(['user_id', 'year']).size().reset_index(name='co
    idx = yearly_reviews.groupby('user_id')['count'].idxmax()
    most_reviews_year = yearly_reviews.loc[idx].set_index('user_id')['year']

    answer = most_reviews_year - 2004
    return answer


answer = find_most_reviews_year(df)
#raise NotImplementedError()
```

```
#This is a graded cell, do not edit
print(answer.sort_index().head())
```

```
user_id
--1Y03CEKR3WDbBjYnsW7A     7
--2QZsyXGz1OhiD4-0FQLQ    10
--82_AVgRBsLw6Dhy8sEnA     4
--8A9o_NeGyt_3kzlXtSdg    11
--8WbseBk1NjfPiZWjQ-XQ    12
Name: year, dtype: int64
```

Q12: Come up with a new feature for each review. This may be derived from existing features. Give your feature the name *my_new_feature*. Display head() of this new feature.

```
# YOUR CODE HERE
my_new_feature = df['text'].str.split().apply(len)
#raise NotImplementedError()
```

```
#This is a graded cell, do not edit
print(my_new_feature.head())
```

```
0    37
1    17
2     4
3    60
4    37
Name: text, dtype: int64
```