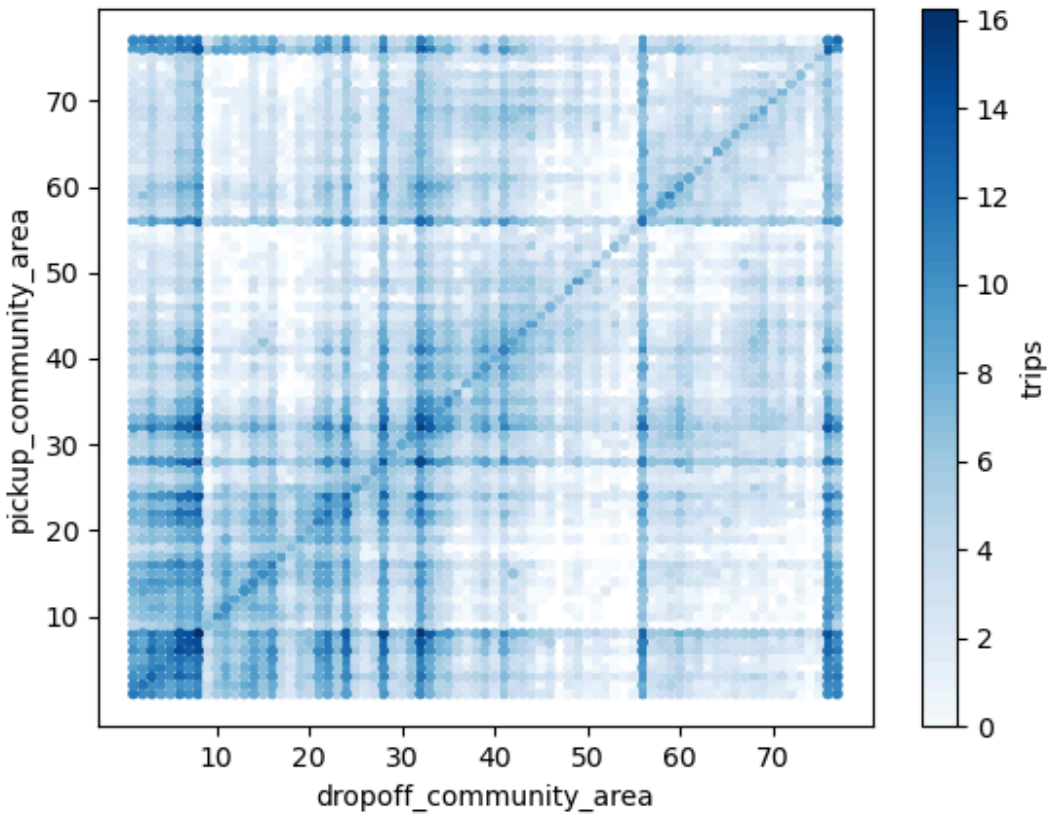
1. TrafficRank
    a. Chicago taxi rides scatterplot

```
import matplotlib.pyplot as plt
import csv
import math

taxi_rides = []
with open('Quiz_3/chicago-taxi-rides.csv', 'r') as file:
    reader = csv.reader(file)
    for row in reader:
        try:
            int(row[0])
            taxi_rides.append(row)
        except:
            continue

x = [int(row[0]) for row in taxi_rides]
y = [int(row[1] )for row in taxi_rides]
z = [math.log(int(row[2])) for row in taxi_rides]
plt.scatter(x, y, c=z, cmap='Blues', s=10)
plt.colorbar(label='trips')
plt.xlabel('dropoff_community_area')
plt.ylabel('pickup_community_area')
plt.xticks([10, 20, 30, 40, 50, 60, 70])
plt.yticks([10, 20, 30, 40, 50, 60, 70])
plt.show()
```



b. Reading the matrix

```
import numpy as np

matrix = np.zeros((77, 77))
for row in taxi_rides:
    matrix[int(row[0])-1, int(row[1])-1] = int(row[2])
```

c. Rankings of Chicago community areas

```
column_sums = matrix.sum(axis=0)
stochastic_matrix = matrix / column_sums

rank = np.full(77, 1/77)
print("iteration 0")
print(rank)

for i in range(6):
    rank = np.dot(stochastic_matrix,rank)
    print("iteration "+str(i+1))
    print(rank)
```

iteration 0
[0.01298701 0.01298701 0.01298701 0.01298701 0.01298701 0.01298701
 0.01298701 0.01298701 0.01298701 0.01298701 0.01298701 0.01298701
 0.01298701 0.01298701 0.01298701 0.01298701 0.01298701 0.01298701
 0.01298701 0.01298701 0.01298701 0.01298701 0.01298701 0.01298701
 0.01298701 0.01298701 0.01298701 0.01298701 0.01298701 0.01298701
 0.01298701 0.01298701 0.01298701 0.01298701 0.01298701 0.01298701
 0.01298701 0.01298701 0.01298701 0.01298701 0.01298701 0.01298701
 0.01298701 0.01298701 0.01298701 0.01298701 0.01298701 0.01298701
 0.01298701 0.01298701 0.01298701 0.01298701 0.01298701 0.01298701
 0.01298701 0.01298701 0.01298701 0.01298701 0.01298701 0.01298701
 0.01298701 0.01298701 0.01298701 0.01298701 0.01298701 0.01298701
 0.01298701 0.01298701 0.01298701 0.01298701 0.01298701]
iteration 1
[0.00683932 0.00900639 0.01764446 0.00840666 0.00729527 0.04750505
 0.02782879 0.15262698 0.00061705 0.00421727 0.01100651 0.00226196
 0.00406792 0.00771278 0.00522479 0.01085633 0.00289082 0.00177196
 0.00318711 0.00144962 0.00764791 0.01722167 0.00260045 0.03393912
 0.00469135 0.00193877 0.00266315 0.08337536 0.00194959 0.00306494
 0.00443036 0.14346879 0.02458763 0.00475774 0.00813502 0.00293976
 0.00336826 0.00652313 0.01039504 0.00372738 0.02581593 0.00717166
 0.00908805 0.01163511 0.00377368 0.00401589 0.00229767 0.00512352
 0.00983541 0.00362686 0.00534709 0.00152714 0.00381129 0.00286569
 0.00243103 0.05767186 0.00275744 0.0043619 0.01030599 0.00304648
 0.00373431 0.00234084 0.00254928 0.00350401 0.00237162 0.00344107
 0.00388152 0.00428417 0.00694634 0.00376051 0.00357764 0.00128418
 0.00300995 0.00082493 0.00201977 0.05800389 0.01411382]
iteration 2
[4.70781526e-03 5.77770219e-03 1.69345940e-02 6.05895482e-03
 6.60962586e-03 5.87086602e-02 4.21287445e-02 2.76600765e-01
 7.29512783e-05 1.47711846e-03 5.81443094e-03 6.09057286e-04
 1.51664489e-03 4.13153428e-03 2.13545881e-03 6.46293008e-03
 7.03713861e-04 2.96262432e-04 8.92256846e-04 3.33521215e-04
 4.11289754e-03 1.22970643e-02 8.22660978e-04 3.28078010e-02
 1.56138782e-03 3.69991694e-04 6.65617689e-04 8.49771461e-02
 5.11915647e-04 8.65741682e-04 1.92257989e-03 2.06670114e-01
 2.66423657e-02 2.22978203e-03 4.17557584e-03 9.39748033e-04
 9.98465611e-04 2.73238626e-03 5.75454099e-03 1.28722204e-03
 1.56483095e-02 3.18353745e-03 4.24797775e-03 6.21072224e-03
 1.22592193e-03 1.31737038e-03 4.95635756e-04 1.95655316e-03
 6.09889125e-03 1.05204315e-03 2.04654043e-03 2.21287647e-04
 1.19506764e-03 6.46521636e-04 5.02989693e-04 2.99995225e-02
 6.62265696e-04 1.49772982e-03 7.41927051e-03 1.09995434e-03
 1.13733527e-03 5.25676700e-04 5.80559441e-04 1.00378126e-03
 5.29970868e-04 9.49279821e-04 1.26703631e-03 1.49852948e-03

3.01247202e-03 1.10896590e-03 1.13882382e-03 2.34150444e-04
9.06994732e-04 7.01131795e-05 4.22297585e-04 5.22875624e-02
1.22805906e-02]
iteration 3
[3.60224511e-03 3.89027237e-03 1.57039239e-02 4.64305436e-03
5.92279096e-03 6.18994104e-02 4.92534633e-02 3.25837945e-01
2.90337358e-05 6.52512702e-04 2.92698777e-03 2.66543437e-04
7.16087406e-04 2.51734677e-03 1.01252438e-03 4.14310668e-03
2.20924290e-04 7.12215641e-05 3.42582402e-04 1.46031511e-04
2.69786509e-03 1.02858985e-02 4.15827360e-04 3.22214524e-02
5.60656013e-04 1.02761460e-04 2.38021256e-04 8.91409977e-02
2.07555003e-04 3.13147398e-04 1.13217673e-03 2.27326377e-01
2.56551753e-02 1.44483129e-03 2.31954421e-03 3.91309164e-04
3.43324934e-04 1.17382575e-03 3.04816981e-03 5.07538477e-04
8.76648655e-03 1.41339451e-03 1.87303676e-03 2.93444147e-03
4.46853035e-04 4.83648587e-04 1.39517633e-04 7.55118226e-04
3.53667213e-03 3.28454182e-04 7.71151381e-04 4.76193219e-05
4.12635546e-04 1.53063205e-04 1.21617883e-04 2.03126280e-02
1.98948766e-04 5.85837307e-04 5.16020831e-03 5.50631974e-04
4.13653537e-04 1.65946685e-04 1.64963941e-04 3.13138111e-04
1.51192053e-04 2.94215916e-04 4.53604618e-04 5.69025053e-04
1.26952349e-03 3.49653210e-04 4.19095087e-04 7.65034175e-05
3.52685626e-04 1.27907090e-05 1.29133775e-04 5.19054806e-02
1.06429664e-02]
iteration 4
[3.02116855e-03 2.89479073e-03 1.48339189e-02 3.92754706e-03
5.56870451e-03 6.26419053e-02 5.21922168e-02 3.45171001e-01
1.95951427e-05 3.93276876e-04 1.60646722e-03 1.69842434e-04
4.42890965e-04 1.84125324e-03 6.08073642e-04 3.13427947e-03
1.08393464e-04 2.93294078e-05 1.93338181e-04 9.88883501e-05
2.17729823e-03 9.63546545e-03 3.04858997e-04 3.22940477e-02
2.50728994e-04 4.48526421e-05 1.31612895e-04 9.13725017e-02
1.21391998e-04 1.60273945e-04 8.79944728e-04 2.35570998e-01
2.51046458e-02 1.16605460e-03 1.54920451e-03 2.04706242e-04
1.39130619e-04 5.72886848e-04 1.83381093e-03 2.29698210e-04
5.67597583e-03 7.09519292e-04 8.78615293e-04 1.35614408e-03
1.84003546e-04 2.03463610e-04 5.06272190e-05 3.04728182e-04
1.99490830e-03 1.14477087e-04 2.96901436e-04 1.65750844e-05
1.60206899e-04 3.98235358e-05 3.62882005e-05 1.72014051e-02
8.75220549e-05 2.83038341e-04 3.55298545e-03 3.79449634e-04
1.96351462e-04 8.24292577e-05 6.49863588e-05 1.21124730e-04
6.09442277e-05 1.11781867e-04 1.82262918e-04 2.42675451e-04
5.57298375e-04 1.28949322e-04 1.81835364e-04 3.65442330e-05
1.64616001e-04 5.39025995e-06 5.35511551e-05 5.20113762e-02
9.63023109e-03]
iteration 5

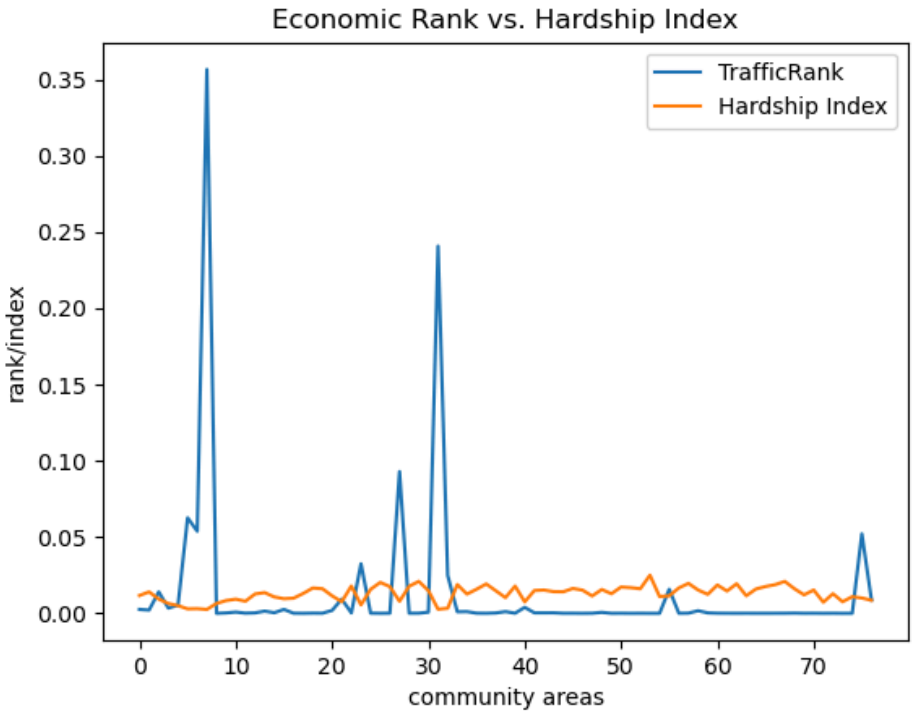
[2.72262280e-03 2.39049411e-03 1.43338273e-02 3.58820955e-03
5.40465946e-03 6.27149246e-02 5.33531831e-02 3.53162750e-01
1.63948848e-05 3.05710693e-04 1.04199231e-03 1.35749652e-04
3.41269894e-04 1.56089681e-03 4.58349600e-04 2.71699763e-03
7.92810994e-05 1.96020227e-05 1.47017791e-04 8.43187097e-05
1.98445758e-03 9.42680811e-03 2.70728527e-04 3.24430568e-02
1.52279019e-04 2.89033647e-05 1.00105313e-04 9.24608972e-02
9.25236261e-05 1.12504978e-04 7.91633623e-04 2.39152571e-01
2.49188325e-02 1.05997734e-03 1.24136755e-03 1.34597444e-04
6.83602021e-05 3.41197190e-04 1.33149708e-03 1.24955155e-04
4.41929168e-03 4.33837801e-04 4.78700103e-04 6.51279357e-04
8.78897834e-05 1.01924690e-04 2.26053675e-05 1.34473312e-04
1.11465885e-03 4.61118602e-05 1.21651469e-04 8.54359638e-06
7.12039739e-05 1.23157411e-05 1.39958960e-05 1.61570959e-02
5.73881722e-05 1.69790436e-04 2.44917500e-03 3.19601390e-04
1.22370689e-04 5.97364578e-05 3.70415227e-05 6.68506064e-05
3.62036092e-05 5.61887617e-05 8.60097472e-05 1.21930032e-04
2.69755838e-04 6.21505035e-05 9.60664977e-05 2.27760628e-05
8.75729316e-05 3.49174122e-06 2.77747259e-05 5.20856550e-02
9.07138461e-03]
iteration 6
[2.57252978e-03 2.13916706e-03 1.40697823e-02 3.43004149e-03
5.32930855e-03 6.26514495e-02 5.38158046e-02 3.56649203e-01
1.51779212e-05 2.74036190e-04 8.07103048e-04 1.22347046e-04
3.00617787e-04 1.44350378e-03 4.00871344e-04 2.54485424e-03
7.06624689e-05 1.68598008e-05 1.30922805e-04 7.92438722e-05
1.91001475e-03 9.35496876e-03 2.59094223e-04 3.25412904e-02
1.19201493e-04 2.35848228e-05 8.90510116e-05 9.29888647e-02
8.16531069e-05 9.54418202e-05 7.56474341e-04 2.40818196e-01
2.48733607e-02 1.01707738e-03 1.11854142e-03 1.06980363e-04
4.16533466e-05 2.49844539e-04 1.12666896e-03 8.39175339e-05
3.91816625e-03 3.23796382e-04 3.17674513e-04 3.44442630e-04
5.05134320e-05 6.28814592e-05 1.22015939e-05 6.88381733e-05
6.25062948e-04 2.21029542e-05 5.60774898e-05 5.81553480e-06
3.69690253e-05 5.01121618e-06 7.20399786e-06 1.57869128e-02
4.78806002e-05 1.19535850e-04 1.70194821e-03 2.95741045e-04
9.32492432e-05 5.26075424e-05 2.78975395e-05 5.09840420e-05
2.84494360e-05 3.73676928e-05 4.98801456e-05 7.48666987e-05
1.52231513e-04 4.06479335e-05 6.27518275e-05 1.72841871e-05
5.24322754e-05 2.77923590e-06 1.74686594e-05 5.21311582e-02
8.77980107e-03]

d. Hardship index comparison

```
hardship_index =
[39.4,47.3,31.5,21.7,16.9,9.9,10.3,8.6,21.8,28.2,31.3,26.7,42.8,45.7,36.2
,32.3,33.4,43.9,55.9,54.3,38.6,25.6,60.3,18.7,53.1,68.3,58.9,26.6,59.8,70
.6,50.1,9.0,11.2,63.5,42.5,53.2,64.9,49.8,34.6,60.2,25.3,50.4,51.4,47.9,4
```

```
7.6,54.9,51.2,38.4,52.6,43.3,58.1,56.7,54.3,84.2,37.0,38.7,56.1,66.1,51.5
,41.9,62.6,49.2,65.3,38.8,53.7,59.2,63.3,70.5,54.3,40.8,51.5,24.5,43.3,25
.6,36.7,33.8,28.9]
total_hardship = sum(hardship_index)
hardship_index = [i/total_hardship for i in hardship_index]

plt.plot(np.arange(77), rank, label='TrafficRank')
plt.plot(np.arange(77), hardship_index, label='Hardship Index')
plt.title('Economic Rank vs. Hardship Index')
plt.xlabel('community areas')
plt.ylabel('rank/index')
plt.legend()
plt.show()
```



Although I was not able to figure out a way to invert the hardship index vector, the plot above reveals a slight inverse relationship between the two lists. Where there is a big spike in the TrafficRank near community areas 6-8 (validated in the scatterplot), there is a clear dip in the hardship index. The same can be seen for communities 28, 32, 56 and 76. After normalizing the hardship index values, they appear much more uniform than the normalized TrafficRank values. I assume the spikes are not as drastically shown in the hardship index because the lower-income residents are evenly distributed across all of the Chicago communities as they make up most of the population within the city limits. Whereas the top 1% come from the suburbs to fill the business districts which have more higher end businesses in the area and less low-income households. Moreover, spikes in hardship index would not necessarily be reflected in economic rank because communities of varying hardships that are residential will have less traffic due to lack of infrastructure.

2. Text Processing
    a. The IDF calculation can only apply to a corpus because it relies on word frequency across documents rather than TF which looks within a document. Words that appear more frequently in the corpus have a smaller IDF score, whereas words that appear less often means it is more important and has a higher IDF score.
    b. The implementation of Scikit-Learn's IDF calculation differs slightly from the standard calculation for a smoothing effect and numerical stability. Adding 1 to the numerator prevents values from getting overly small for words that appear often. Adding 1 to the denominator prevents values from getting overly large as well as division by zero if words appear very rarely or none at all, respectively. The final addition of one after taking the logarithm avoids an IDF of 0 for common words.
    c. Calculating the TF.IDF for any president's speeches

```
import tarfile
import os
from sklearn.feature_extraction.text import TfidfVectorizer
import numpy as np


def get_top_15_words(tar_file_path, extract_path):
    with tarfile.open(tar_file_path, 'r:gz') as tar:
        tar.extractall(path=extract_path)


    base_name = os.path.basename(tar_file_path).split(".")[0]
    dir_path = extract_path + "/" + base_name
    texts = []
    for filename in os.listdir(dir_path):
        with open(os.path.join(dir_path, filename), 'r',
encoding='utf-8') as file:
            texts.append(file.read())


    vectorizer = TfidfVectorizer(stop_words='english')
    tfidf_matrix = vectorizer.fit_transform(texts)
    word_scores = np.array(tfidf_matrix.sum(axis=0)).flatten() # Sum
TF-IDF scores for each word across all documents


    feature_names = vectorizer.get_feature_names_out()
    sorted_indices = word_scores.argsort()[::-1]


    print(f"Top 15 Most Important Words for {base_name.capitalize()}:")
```

```
    for i in range(15):
        word = feature_names[sorted_indices[i]]
        print(f"{i+1}. {word}")
```

d. Qualitative interpretation and analysis of outputs

```
file_path = 'Quiz_3/prez_speeches/carter.tar.gz'
extract_path = 'Quiz_3/'
get_top_15_words(file_path, extract_path)
```

```
Top 15 Most Important Words for Carter:
1. people
2. world
3. energy
4. mr
5. president
6. ve
7. nation
8. american
9. government
10. oil
11. soviet
12. peace
13. united
14. inflation
15. year
```

President Carter was in office from 1977-1981 during the 1979 energy crisis and formed the department of energy. During his presidency, the country saw double-digit inflation numbers, and oil shortages. He was instrumental in the Camp David Accords to bring peace between Egypt and Israel. During the Soviet invasion of Afghanistan also in 1979, Carter announced sanctions on the Soviet Union and demanded a boycott of the 1980 Summer Olympics in Moscow.

After looking carefully at the important words from multiple other presidents' speeches, I would choose to add 'people', 'new', 'mr', 've', 'year', 'years' to the list of stopwords as they convey very little information. The words 'united', 'states', 'country', 'world', 'president', 'government', 'america', 'american', and 'americans' also appear a lot, but these are words that I would expect from any president of the United States.