CS564 Foundations of Machine Learning
Assignment: 1 Part: A

Mukuntha N S (1601CS27)
R J Srivatsa (1601CS35)
Shikhar Jaiswal (1601CS44)

August 17, 2019

1 Assignment Description

The crucial task before applying any machine learning algorithm is a thorough data analysis and data visualization. We are given a dataset (**stackOverflow.csv**), from which a certain amount of information is to be extracted.

2 Procedure

Installation

Install the following dependencies either using pip or through conda in a Python 3.5+ environment:
• jupyter
• pandas
• matplotlib

```
python3 -m pip install jupyter pandas matplotlib
```

or alternatively,

```
conda install -c anaconda jupyter pandas matplotlib
```

Running The Notebook

To run the program, head to the directory $Assignment1/Q1$. Please note that the **dataset should be present in the same directory** as the jupyter notebook. Use the following command to run the notebook:

```
jupyter notebook Q1.ipynb
```

3 Discussion

The primary objective of the assignment is to analyze and visualize the data through the code and the queries asked. The following sub-sections contain the explanation for individual code snippets. For a detailed look at the output, please refer the notebook and the individual files provided.

Notebook Code

Pre-processing and Visualization

Import the Python dependencies and check the data samples and their values.

```
import pandas
import matplotlib
import matplotlib.pyplot as plt

data = pandas.read_csv('./stackOverflow.csv', encoding = 'utf-8')

print(data)
```

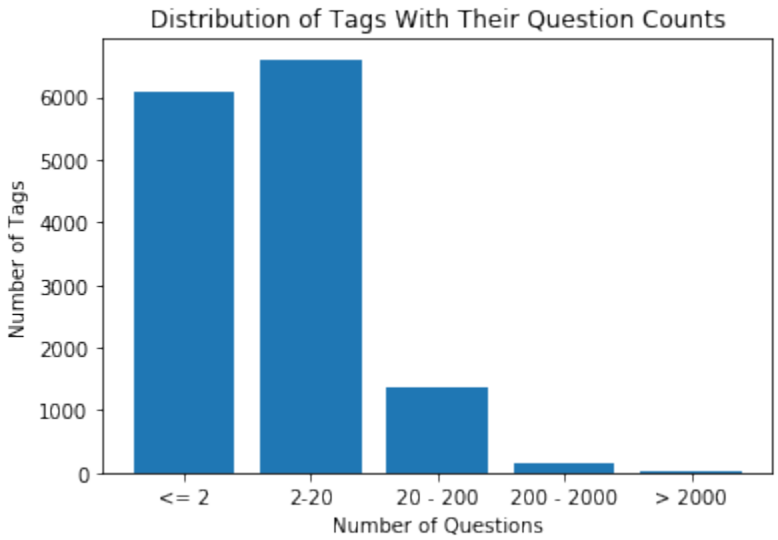
1) The Number Of Questions Asked With Respect To Given Tags

We create a dictionary keeping the unique set of tags as keys, and the question counts as values. We then iterate through the dataset, incrementing the question count whenever we see a previously observed tag. We finally print every tag with its corresponding question count.

```
tag_question_map = dict()

for question_id, tags_list in zip(list(data['qid']), list(data['tags'])):
    tags_list = tags_list.split(',')

    for tag in tags_list:
        if tag in tag_question_map.keys():
            tag_question_map[tag].add(question_id)
        else:
            tag_question_map[tag] = set([question_id])

question1_answer = []
question1_answer.append('tag,number_of_questions\n')

question1_answer = []
tag_counts = [0, 0, 0, 0, 0]
question1_answer.append('tag,number_of_questions\n')

for tag, questions in sorted(tag_question_map.items(), key = lambda x: x[0]):
    question1_answer.append('{},{}\n'.format(tag, len(questions)))

    if len(questions) < 2:
        tag_counts[0] += 1
```

```
    elif len(questions) < 20:
        tag_counts[1] += 1
    elif len(questions) < 200:
        tag_counts[2] += 1
    elif len(questions) < 2000:
        tag_counts[3] += 1
    else:
        tag_counts[4] += 1

index = [0, 1, 2, 3, 4]
plt.bar(index, tag_counts)
plt.xlabel('Number of Questions', fontsize = 10)
plt.ylabel('Number of Tags', fontsize = 10)
plt.xticks(index, ['<= 2', '2-20', '20 - 200', '200 - 2000', '> 2000'], fontsize = 10)
plt.title('Distribution of Tags With Their Question Counts')
plt.show()
```

Plot



Additionally please refer $Q1/q1\_1.csv$ file for the output.

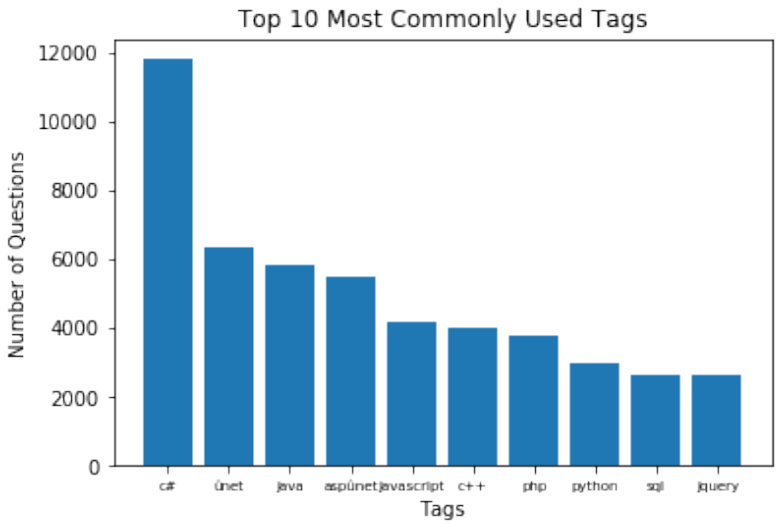
2) Most Commonly Used Tags And The Trend In Data Science Tags

We sort the key-value pairs we obtain from the previous dictionary by the question counts, and print every tag with its corresponding question count.

```
question2_answer = []
labels = []
data_science = ['python', 'sql', 'excel', 'matlab', 'sas', 'r']
trend = [0, 0, 0, 0, 0, 0]
```

```
question_counts = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
question2_answer.append('tag,number_of_questions\n')
i = 0

for tag, questions in sorted(tag_question_map.items(), key = lambda x: -len(x[1])):
    if i < 10:
        labels.append(tag)
        question_counts[i] = len(questions)
        i += 1

    if tag in data_science:
        trend[data_science.index(tag)] = len(questions)

    question2_answer.append('{},{}\n'.format(tag, len(questions)))

index = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
plt.bar(index, question_counts)
plt.xlabel('Tags', fontsize = 10)
plt.ylabel('Number of Questions', fontsize = 10)
plt.xticks(index, labels, fontsize = 7)
plt.title('Top 10 Most Commonly Used Tags')
plt.show()
```

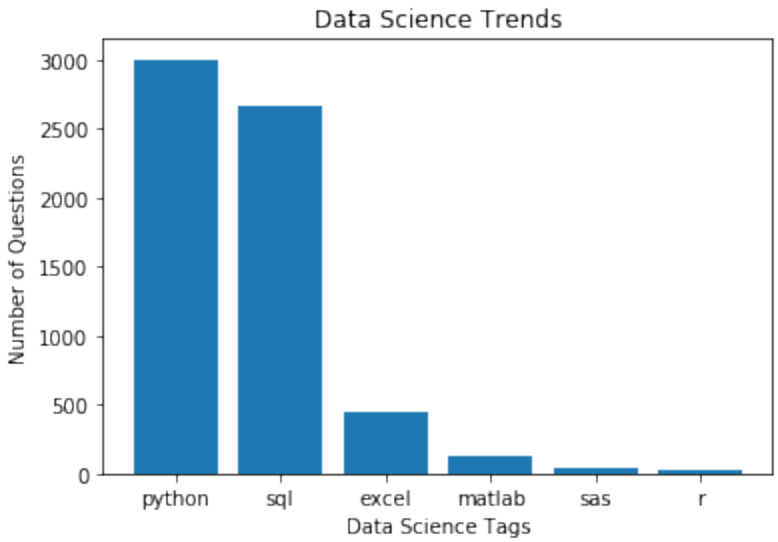
Plot




```
index = [0, 1, 2, 3, 4, 5]
plt.bar(index, trend)
plt.xlabel('Data Science Tags', fontsize = 10)
plt.ylabel('Number of Questions', fontsize = 10)
plt.xticks(index, data_science, fontsize = 10)
```

```
plt.title('Data Science Trends')
plt.show()
```

Plot



Additionally please refer $Q1/q1\_2.csv$ file for the output.

3) Average Time Taken To Answer A Question

We find the column holding the difference between the answer time and the question time, and compute the mean of all the values. We finally print out the same mean.

```
average_answer_time = (data['at'] - data['qt']).mean()
question3_answer = ['{}\n'.format(average_answer_time)]

print('Average time taken to answer a question (considering all answers): \
       {}s'.format(average_answer_time))
```

Output: Average time taken to answer a question (considering all answers): 133765.87433786143s
Additionally please refer $Q1/q1\_3.csv$ file for the output.

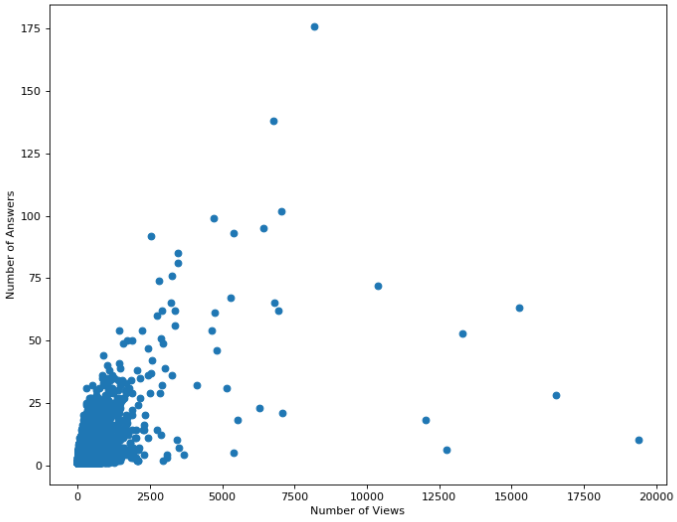
4) Number Of Views Related To The Number Of Answers

We simply print out a scatter plot with the number of views on the x-axis and the number of answers on the y-axis.

```
view_vs_answer = [(view, answer) for _, view, answer in (set(zip(data['qid'],
                  data['qvc'], data['qac'])))]

plt.figure(num = None, figsize = (10, 8), dpi = 80, facecolor = 'w', edgecolor = 'k')
```

```
plt.xlabel('Number of Views')
plt.ylabel('Number of Answers')
plt.scatter(*zip(*view_vs_answer))
plt.savefig("q1_4.png")
plt.show()
```

Plot



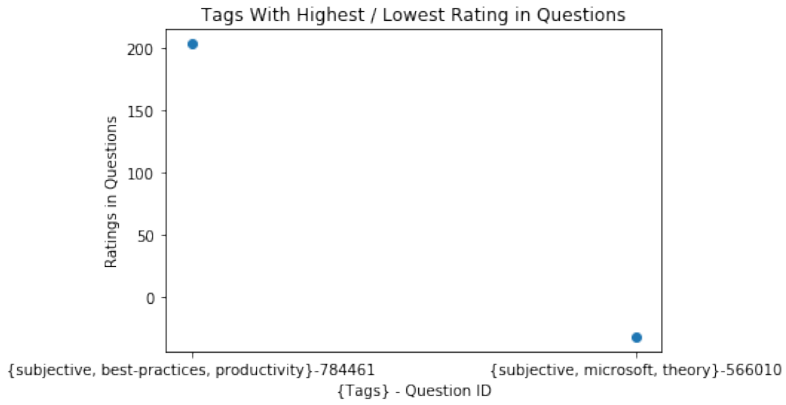
5) Tags That Get Highest / Lowest Rating In Questions

We create a dictionary keeping the unique set of tags as keys, and the tuple of question IDs and question scores as values. We then iterate through the dataset, updating the highest / lowest question ID and score pairs whenever we see a higher / lower observed score for the same tag. We finally print every tag with its corresponding highest / lowest question ID and question score.

```
tag_max_question_rating_map = dict()
tag_min_question_rating_map = dict()

tag_max_answer_rating_map = dict()
tag_min_answer_rating_map = dict()

for question_id, answer_id, tags_list, question_score, answer_score in
    zip(list(data['qid']), list(data['aid']), list(data['tags']), list(data['qs']),
        list(data['as'])):
    tags_list = tags_list.split(',')

    for tag in tags_list:
```

```
        if tag in tag_max_question_rating_map.keys():
            if tag_max_question_rating_map[tag][1] < question_score:
                tag_max_question_rating_map[tag] = (question_id, question_score)
            if tag_min_question_rating_map[tag][1] > question_score:
                tag_min_question_rating_map[tag] = (question_id, question_score)
            if tag_max_answer_rating_map[tag][1] < answer_score:
                tag_max_answer_rating_map[tag] = (answer_id, answer_score)
            if tag_min_answer_rating_map[tag][1] > answer_score:
                tag_min_answer_rating_map[tag] = (answer_id, answer_score)
        else:
            tag_max_question_rating_map[tag] = (question_id, question_score)
            tag_min_question_rating_map[tag] = (question_id, question_score)
            tag_max_answer_rating_map[tag] = (answer_id, answer_score)
            tag_min_answer_rating_map[tag] = (answer_id, answer_score)

tag_max_question_rating = sorted(tag_max_question_rating_map.items(),
                                 key = lambda x: -x[1][1])[:3]
tag_min_question_rating = sorted(tag_min_question_rating_map.items(),
                                 key = lambda x: x[1][1])[:3]

question5_answer = []
question5_answer.append('tag,qid_highest,highest_question_rating,qid_lowest,\
                         lowest_question_rating\n')

for tag in sorted(tag_max_question_rating_map.keys()):
    question5_answer.append('{},{},{},{},{}\n'.format(tag,
                                                      tag_max_question_rating_map[tag][0],
                                                      tag_max_question_rating_map[tag][1],
                                                      tag_min_question_rating_map[tag][0],
                                                      tag_min_question_rating_map[tag][1],))

index = [0, 1]
plt.scatter(index, [tag_max_question_rating[0][1][1], tag_min_question_rating[0][1][1]])
plt.xlabel('{Tags} - Question ID', fontsize = 10)
plt.ylabel('Ratings in Questions', fontsize = 10)
plt.xticks(index, ['{' + tag_max_question_rating[0][0] + ', ' +
                   tag_max_question_rating[1][0] + ', ' + tag_max_question_rating[2][0] +
                   '}-' + str(tag_max_question_rating[0][1][0]), '{' +
                   tag_min_question_rating[0][0] + ', ' + tag_min_question_rating[1][0] +
                   ', ' + tag_min_question_rating[2][0] + '}-' +
                   str(tag_min_question_rating[0][1][0])], fontsize = 10)
plt.title('Tags With Highest / Lowest Rating in Questions')
plt.show()
```

Plot



Additionally please refer $Q1/q1\_5.csv$ file for the output.

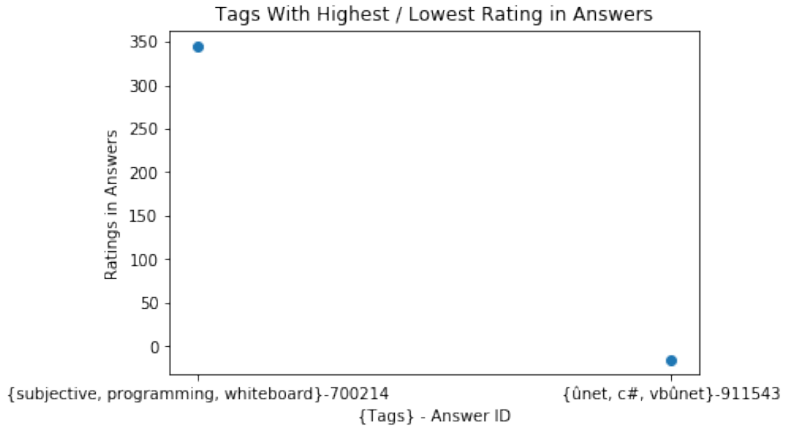
6) Tags That Get Highest / Lowest Rating In Answers

We create a dictionary keeping the unique set of tags as keys, and the tuple of answer IDs and answer scores as values. We then iterate through the dataset, updating the highest / lowest answer ID and score pairs whenever we see a higher / lower observed score for the same tag. We finally print every tag with its corresponding highest / lowest answer ID and answer score.

```
tag_max_answer_rating = sorted(tag_max_answer_rating_map.items(),
                               key = lambda x: -x[1][1])[:3]
tag_min_answer_rating = sorted(tag_min_answer_rating_map.items(),
                               key = lambda x: x[1][1])[:3]

question6_answer = []
question6_answer.append('tag,aid_highest,highest_answer_rating,aid_lowest,\
                         lowest_answer_rating\n')

for tag in sorted(tag_max_question_rating_map.keys()):
    question6_answer.append('{},{},{},{},{}\n'.format(tag,
                                                      tag_max_answer_rating_map[tag][0],
                                                      tag_max_answer_rating_map[tag][1],
                                                      tag_min_answer_rating_map[tag][0],
                                                      tag_min_answer_rating_map[tag][1],))

index = [0, 1]
plt.scatter(index, [tag_max_answer_rating[0][1][1], tag_min_answer_rating[0][1][1]])
plt.xlabel('{Tags} - Answer ID', fontsize = 10)
plt.ylabel('Ratings in Answers', fontsize = 10)
plt.xticks(index, ['{' + tag_max_answer_rating[0][0] + ', ' + tag_max_answer_rating[1][0] +
                   ', ' + tag_max_answer_rating[2][0] + '}-' +
                   str(tag_max_answer_rating[0][1][0]), '{' + tag_min_answer_rating[0][0] +
                   ', ' + tag_min_answer_rating[1][0] + ', ' + tag_min_answer_rating[2][0] +
```

```
                    '}-' + str(tag_min_answer_rating[0][1][0])], fontsize = 10)
plt.title('Tags With Highest / Lowest Rating in Answers')
plt.show()
```

Plot



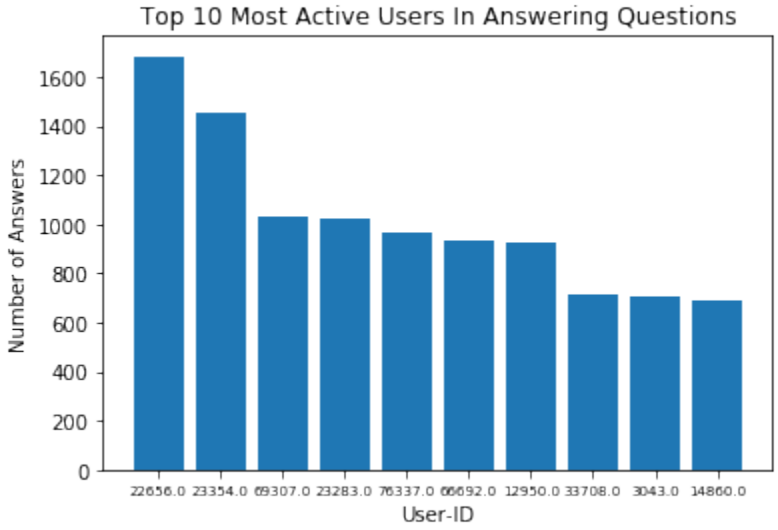
Additionally please refer $Q1/q1\_6.csv$ file for the output.

7) Most Active / Inactive In Answering The Questions

We make use of *Counter* object to sort the data by the count of answers made by a particular user. We finally print the top 10 most active and most inactive users by their user ID with the corresponding answer count.

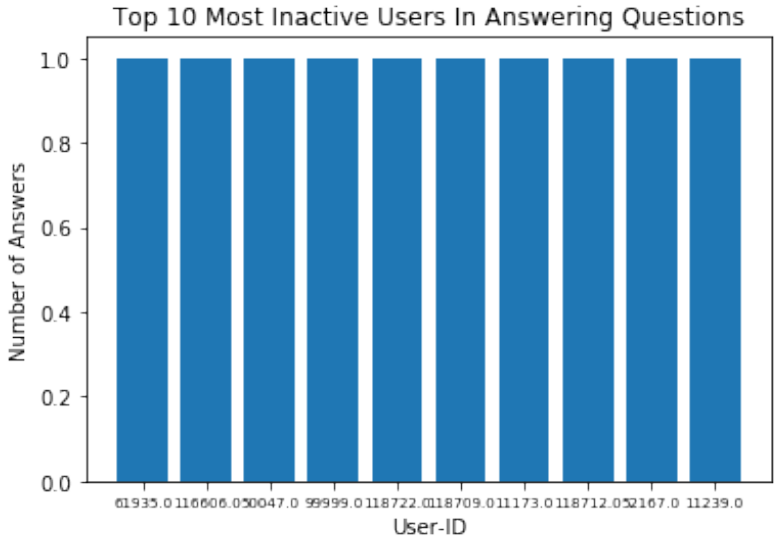
```
from collections import Counter

question7_answer = []
question7_answer.append('Top 10 Most Active Users In Answering Questions:\n')
sorted_answer_counts = Counter(data['j']).most_common()
user_ids = []
answer_counts = []

for user, answer_count in sorted_answer_counts[:10]:
    user_ids.append(user)
    answer_counts.append(answer_count)
    question7_answer.append("User-ID: {}, Number of Answers: \
                            {}\n".format(user, answer_count))

index = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
plt.bar(index, answer_counts)
plt.xlabel('User-ID', fontsize = 5)
plt.ylabel('Number of Answers', fontsize = 5)
plt.xticks(index, user_ids, fontsize = 5, rotation = 30)
plt.title('Top 10 Most Active Users In Answering Questions')
plt.show()
```

Plot




```
question7_answer.append('\nTop 10 Most Inactive Users In Answering Questions:\n')
user_ids = []
answer_counts = []

for user, answer_count in sorted_answer_counts[-10:][::-1]:
    user_ids.append(user)
    answer_counts.append(answer_count)
    question7_answer.append("User-ID: {}, Number of Answers: \
                            {}\n".format(user, answer_count))

plt.bar(index, answer_counts)
plt.xlabel('User-ID', fontsize = 5)
plt.ylabel('Number of Answers', fontsize = 5)
plt.xticks(index, user_ids, fontsize = 5, rotation = 30)
plt.title('Top 10 Most Inactive Users In Answering Questions')
plt.show()
```

Plot



Additionally please refer $Q1/q1\_7.csv$ file for the output.

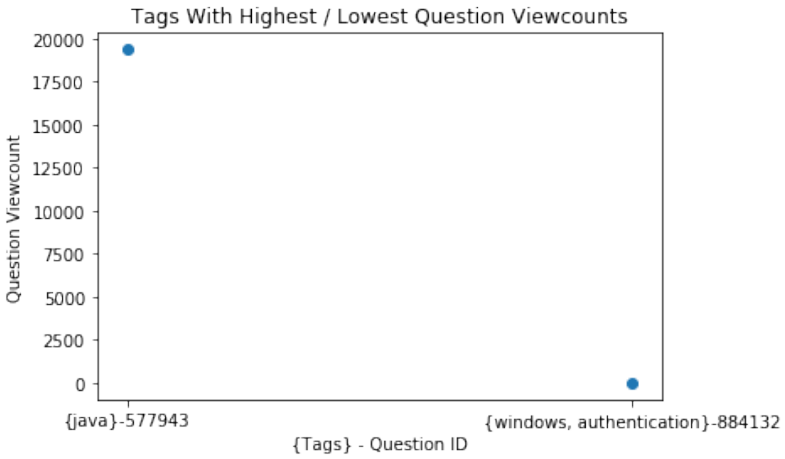
8) Tags Drawing The Highest / Lowest Views

We create a dictionary keeping the unique set of tags as keys, and the tuple of question IDs and question view-counts as values. We then iterate through the dataset, updating the highest / lowest question ID and question view-count pairs whenever we see a higher / lower observed view-count for the same tag. We finally print every tag with its corresponding highest / lowest question ID and question view-count.

```
tag_max_viewcount_map = dict()
tag_min_viewcount_map = dict()

for question_id, question_viewcount, tags_list in zip(list(data['qid']), list(data['qvc']),
                                                      list(data['tags'])):
    tags_list = tags_list.split(',')

    for tag in tags_list:
        if tag in tag_max_viewcount_map.keys():
            if tag_max_viewcount_map[tag][1] < question_viewcount:
                tag_max_viewcount_map[tag] = (question_id, question_viewcount)
            if tag_min_viewcount_map[tag][1] > question_viewcount:
                tag_min_viewcount_map[tag] = (question_id, question_viewcount)
        else:
            tag_max_viewcount_map[tag] = (question_id, question_viewcount)
            tag_min_viewcount_map[tag] = (question_id, question_viewcount)

tag_max_viewcount = sorted(tag_max_viewcount_map.items(), key = lambda x: -x[1][1])[0]
tag_min_viewcount = sorted(tag_min_viewcount_map.items(), key = lambda x: x[1][1])[:2]
```

```
question8_answer = []
question8_answer.append('tag,qid_highest,highest_question_viewcount,qid_lowest,\
                         lowest_question_viewcount\n')

for tag in sorted(tag_max_viewcount_map.keys()):
    question8_answer.append('{},{},{},{},{}\n'.format(tag,
                                                      tag_max_viewcount_map[tag][0],
                                                      tag_max_viewcount_map[tag][1],
                                                      tag_min_viewcount_map[tag][0],
                                                      tag_min_viewcount_map[tag][1]))

index = [0, 1]
plt.scatter(index, [tag_max_viewcount[1][1], tag_min_viewcount[0][1][1]])
plt.xlabel('{Tags} - Question ID', fontsize = 10)
plt.ylabel('Question Viewcount', fontsize = 10)
plt.xticks(index, ['{' + tag_max_viewcount[0] + '}-' + str(tag_max_viewcount[1][0]),
                   '{' + tag_min_viewcount[0][0] + ', ' + tag_min_viewcount[1][0] +
                   '}-' + str(tag_min_viewcount[0][1][0])], fontsize = 10)
plt.title('Tags With Highest / Lowest Question Viewcounts')
plt.show()
```

Plot



Additionally please refer *Q*1/*q*1_8.*csv* file for the output.

Write to File

We finally write out all the parts of the answer out to individual files.

```
filenames = [(question1_answer, 'q1_1.csv'),
             (question2_answer, 'q1_2.csv'),
             (question3_answer, 'q1_3.csv'),
             (question5_answer, 'q1_5.csv'),
```

```
            (question6_answer, 'q1_6.csv'),
            (question7_answer, 'q1_7.txt'),
            (question8_answer, 'q1_8.csv')]

for answer_list, fname in filenames:
    with open(fname, 'w', encoding = 'utf-8') as f:
        f.writelines(answer_list)
```