CS 6220 Data Mining — Assignment 1

Weihong Yang | https://github.com/wyang10 | yang.weih@northeastern.edu

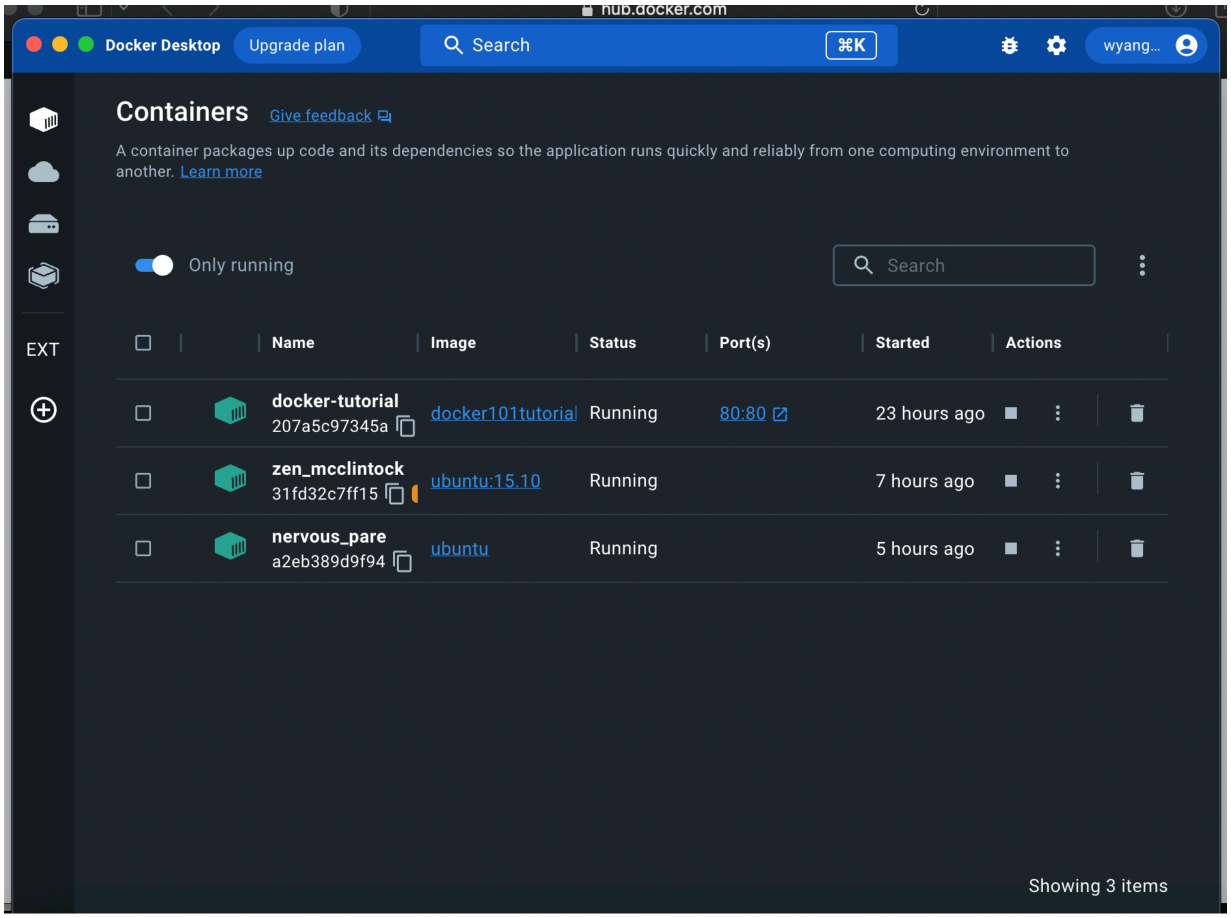
1 Getting Started

1.2

Executing “Hello World” & Screenshot:

In [ ]:
```
# Use the official Python image as the base image in Docker
# FROM python:3.8

# Copy the script that prints "Hello World" into the container
# COPY script.py /

# Run the script when the container starts
# CMD ["python", "/script.py"]


print ("Hello World")


# > docker build -t my-python-image .

# > docker run my-python-image
```

```
Hello World
```



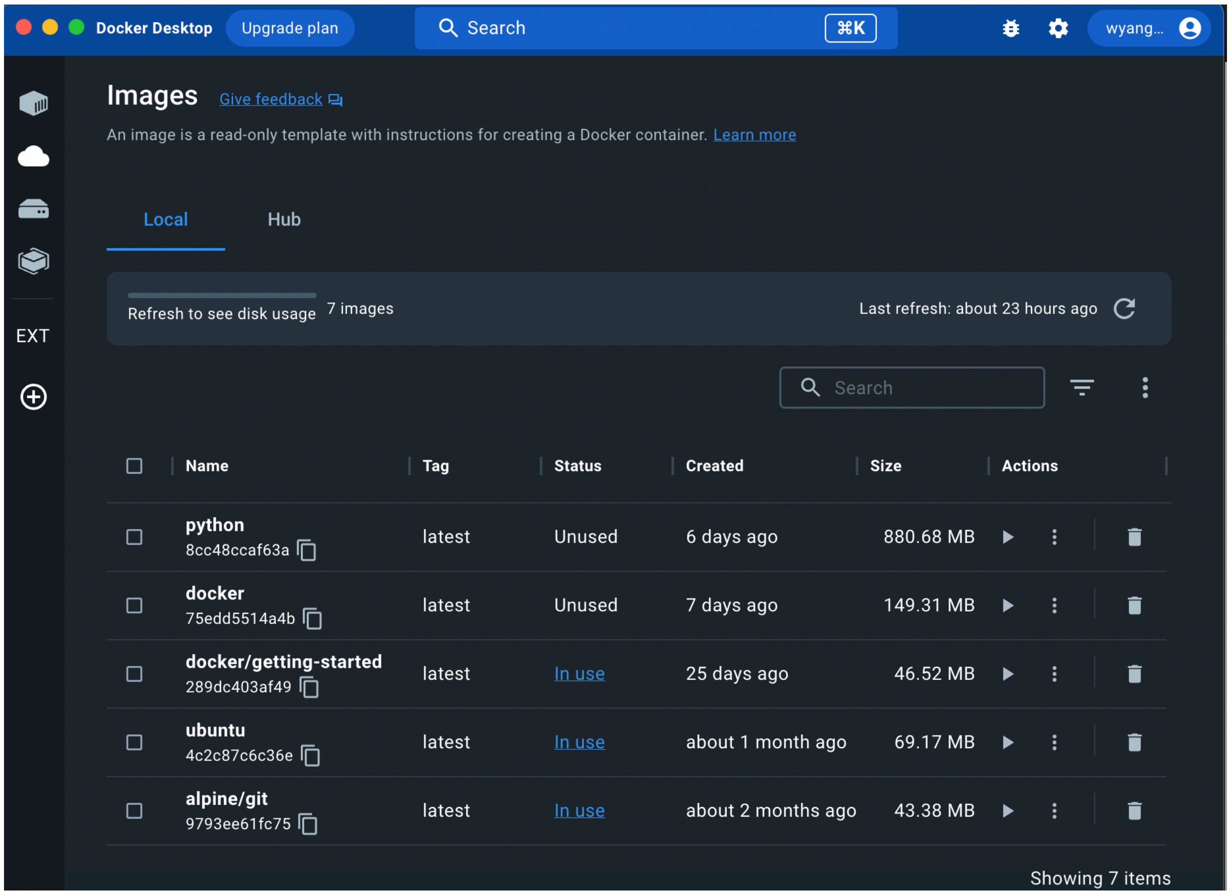
Containers Give feedback

A container packages up code and its dependencies so the application runs quickly and reliably from one computing environment to another. Learn more

Only running

Name	Image	Status	Port(s)	Started
docker-tutorial 207a5c97345a	docker101tutorial	Running	80:80	23 hours ago
zen_mcclintock 31fd32c7ff15	ubuntu:15.10	Running		7 hours ago
nervous_pare a2eb389d9f94	ubuntu	Running		5 hours ago

Showing 3 items



Images Give feedback

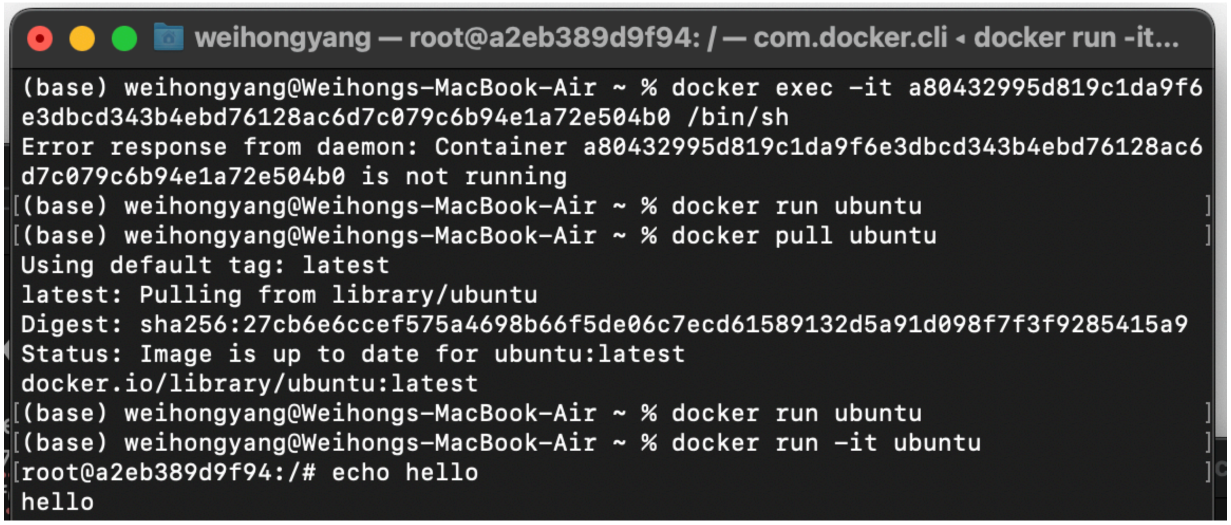
An image is a read-only template with instructions for creating a Docker container. Learn more

Local Hub

Refresh to see disk usage 7 images Last refresh: about 23 hours ago

Name	Tag	Status	Created	Size	Actions
python 8cc48ccaf63a	latest	Unused	6 days ago	880.68 MB	
docker 75edd5514a4b	latest	Unused	7 days ago	149.31 MB	
docker/getting-started 289dc403af49	latest	In use	25 days ago	46.52 MB	
ubuntu 4c2c87c6c36e	latest	In use	about 1 month ago	69.17 MB	
alpine/git 9793ee61fc75	latest	In use	about 2 months ago	43.38 MB	

Showing 7 items


weihongyang — root@a2eb389d9f94: / — com.docker.cli ◂ docker run -it...

```
(base) weihongyang@Weihongs-MacBook-Air ~ % docker exec -it a80432995d819c1da9f6
e3dbcd343b4ebd76128ac6d7c079c6b94e1a72e504b0 /bin/sh
Error response from daemon: Container a80432995d819c1da9f6e3dbcd343b4ebd76128ac6
d7c079c6b94e1a72e504b0 is not running
(base) weihongyang@Weihongs-MacBook-Air ~ % docker run ubuntu
(base) weihongyang@Weihongs-MacBook-Air ~ % docker pull ubuntu
Using default tag: latest
latest: Pulling from library/ubuntu
Digest: sha256:27cb6e6ccef575a4698b66f5de06c7ecd61589132d5a91d098f7f3f9285415a9
Status: Image is up to date for ubuntu:latest
docker.io/library/ubuntu:latest
(base) weihongyang@Weihongs-MacBook-Air ~ % docker run ubuntu
(base) weihongyang@Weihongs-MacBook-Air ~ % docker run -it ubuntu
root@a2eb389d9f94:/# echo hello
hello
```

2 Identifying All Sets

2.1

The cardinality of a set refers to the number of unique elements in the set. The full set of unique items would be the number of different items that were purchased by customers.Function:

- Takes the file path as an argument and opens the file in read mode.
- Uses the csv module to read the file and split each line by the comma delimiter.
- Adds each item to a set.
- Returns the length of the set, which is the cardinality of the dataset.

In [ ]:
```
import csv
import pandas
import matplotlib.pyplot as plt

def cardinality(filename):
  f = open(filename)
  reader = csv.reader(f)
  return reader

reader = cardinality("/content/drive/MyDrive/Colab Notebooks/week-1-assignme
for row in reader:
    print(', '.join(row))
```

```
ketchup,  butter,  diapers
bread,  diapers,  ketchup
butter,  bread,  ketchup
diapers,  butter
ketchup,  butter,  diapers,  beer
diapers,  beer,  butter
diapers,  butter,  bread
bread,  beer,  diapers
ketchup,  chips,  pork,  diapers
butter,  ketchup,  beer
beer,  macaroni,  ketchup,  butter
bread,  ketchup,  butter,  beer
bread,  beer
butter,  beer,  ketchup
diapers,  beer,  butter,  bread
beer,  ketchup,  diapers
butter,  bread,  salmon
butter,  bread
beer,  bread,  diapers
butter,  beer,  ketchup
beer,  diapers,  bread,  ketchup
beer,  ketchup,  tomotes,  butter
bread,  ketchup
bread,  ketchup,  butter
diapers,  butter,  bread
beer,  ketchup
```

```
diapers,  asparagus,  beer
butter,  bread
diapers,  ketchup,  beer,  bread
butter,  ketchup,  beer
beer,  butter,  diapers,  bread
diapers,  bread,  butter,  ketchup
ketchup,  butter,  beans
butter,  diapers,  bread
butter,  bread,  diapers
diapers,  beer
beer,  bread,  ketchup
beer,  diapers,  ketchup
diapers,  beer
ketchup,  beer,  butter,  bread
bread,  diapers
beer,  bread,  butter,  corn
butter,  ketchup
butter,  beer,  diapers,  bread
beer,  bread,  butter
beer,  butter,  bread,  diapers
butter,  ketchup,  beer
beer,  diapers,  butter,  ketchup
butter,  beer,  spinach
bread,  butter
butter,  bread,  ketchup,  diapers
ketchup,  bread,  diapers,  butter
diapers,  beer,  butter,  ketchup
ketchup,  beer
beer,  ketchup,  bread,  diapers
bread,  butter,  beer
beer,  diapers
okra,  beer,  spaghetti
diapers,  ketchup,  beer,  butter
ketchup,  butter,  diapers
beer,  ketchup,  butter,  diapers
bread,  butter,  beer
butter,  okra,  beer,  bread
butter,  bread,  beer
ketchup,  bread,  beer,  leeks
butter,  ketchup,  beer,  bread
butter,  bread,  beer
bread,  diapers,  butter,  ketchup
beer,  diapers,  bread,  butter
butter,  bread
ketchup,  beer
bread,  diapers,  ketchup,  butter
diapers,  ketchup,  bread,  beer
beer,  diapers,  ketchup,  butter
butter,  bread
butter,  diapers,  bread
ketchup,  bread
beer,  butter,  diapers,  ketchup
beer,  butter,  ketchup,  diapers
```

```
diapers, bread, beer
ketchup, butter, diapers
ketchup, diapers
ketchup, bread, diapers
bread, beer, diapers
ketchup, bread, diapers
butter, sausages, bread, diapers
butter, diapers, bread
beer, ketchup
butter, diapers, ketchup, beer
ketchup, bread, butter, raisins
bread, ketchup, diapers, butter
bread, butter
ketchup, butter, squid
bread, butter, beer
beer, butter, ketchup
beer, diapers, bread, butter
beer, ketchup, bread, milk
diapers, ketchup
ketchup, diapers, bread, beer
bread, beer
```

2.2

There are two options for each item: Included or Not included in the subset. For the first item: 2 options For the second item: 2 options ... The total number of possible subsets is the product of all of these options, which is 2^n. So for a set of N unique items, the number of subsets that can be formed is 2^n-1.

2.3

Implementation of a Python module called "all_itemsets":

- Takes a .csv text string file as input
- Returns a list of all possible subsets of unique items

This function:

- Takes the file path as an argument and opens the file in read mode.
- Uses the csv module to read the file and split each line by the comma delimiter.
- Adds each item to a set which stores unique elements.
- Converts the set to a list and using itertools.
- Combinations it creates all possible subsets of size N.
- Returns a list of all the possible subsets.

In [ ]:
```
import csv
import itertools

def all_itemsets(filename):
    unique_items = set()
    with open(filename, 'r') as f:
        reader = csv.reader(f)
        for row in reader:
            for item in row:
                unique_items.add(item)
    unique_items = list(unique_items)
    sets = []
    for i in range(1, len(unique_items)+1):
        sets.extend(list(itertools.combinations(unique_items, i)))
    return sets
```

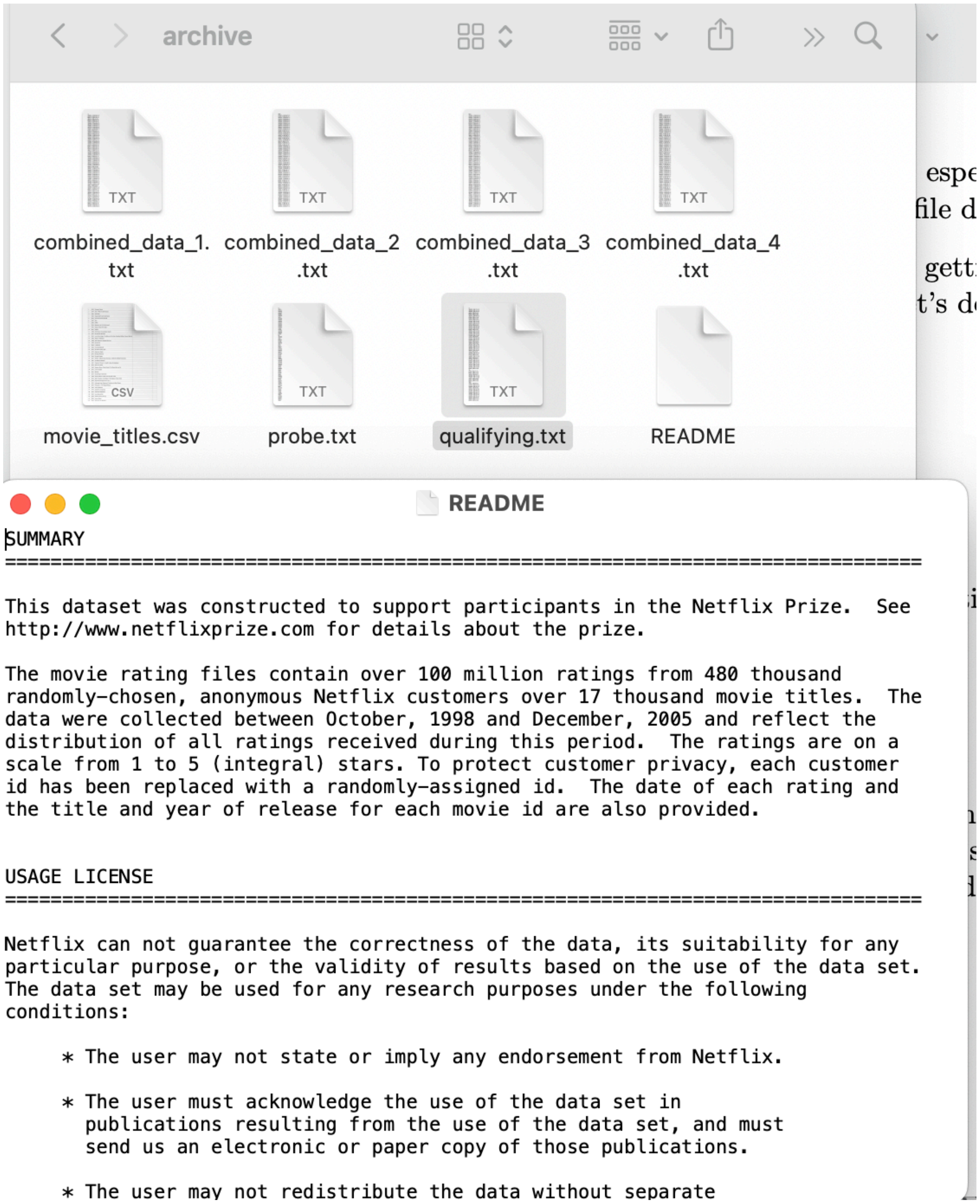
2.4

Implementation of a Python module called "prob_S" that takes a set S and the entire dataset D as input and returns the probability that S occurs:

In [ ]:
```
def prob_S(S, D):
    count = 0
    for i in D:
        if set(S).issubset(set(i)):
            count += 1
    return count/len(D)
```

3 The Netflix Challenge

3.1 Data Verification Screenshot:



```
SUMMARY
================================================================================

This dataset was constructed to support participants in the Netflix Prize.  See
http://www.netflixprize.com for details about the prize.

The movie rating files contain over 100 million ratings from 480 thousand
randomly-chosen, anonymous Netflix customers over 17 thousand movie titles.  The
data were collected between October, 1998 and December, 2005 and reflect the
distribution of all ratings received during this period.  The ratings are on a
scale from 1 to 5 (integral) stars. To protect customer privacy, each customer
id has been replaced with a randomly-assigned id.  The date of each rating and
the title and year of release for each movie id are also provided.


USAGE LICENSE
================================================================================

Netflix can not guarantee the correctness of the data, its suitability for any
particular purpose, or the validity of results based on the use of the data set.
The data set may be used for any research purposes under the following
conditions:

     * The user may not state or imply any endorsement from Netflix.

     * The user must acknowledge the use of the data set in
       publications resulting from the use of the data set, and must
       send us an electronic or paper copy of those publications.

     * The user may not redistribute the data without separate
```

3.2 Data Analysis

3.2.1

To determine the total number of records in the Netflix Prize dataset:

In [ ]:
```
!wc combined_data_1.txt
!head combined_data_1.txt
```

```
wc: combined_data_1.txt: No such file or directory
head: cannot open 'combined_data_1.txt' for reading: No such file or directo
ry
wc: combined_data_2.txt: No such file or directory
head: cannot open 'combined_data_2.txt' for reading: No such file or directo
ry
```

In [ ]:
```
lines = open("combined_data_1.txt")
number_records = 0
for line in lines:
  if ":" in line:
    continue
  else:
    number_records += 1
print(number_records)
```

```
7999141
```

In [ ]:
```
lines = open("/content/drive/MyDrive/Colab Notebooks/week-1-assignment/ Netf
number_records = 0
for line in lines:
  if ":" in line:
    continue
  else:
    number_records += 1
print(number_records)
```

```
26977591
```

In [ ]:
```
lines = open("/content/drive/MyDrive/Colab Notebooks/week-1-assignment/ Netf
number_records = 0
for line in lines:
  if ":" in line:
    continue
  else:
    number_records += 1
print(number_records)
```

```
22601629
```

In [ ]:
```
lines = open("/content/drive/MyDrive/Colab Notebooks/week-1-assignment/ Netf
number_records = 0
for line in lines:
  if ":" in line:
    continue
  else:
    number_records += 1
print(number_records)
```

```
26847523
```

In [ ]:
```
numbers = [7999141, 26977591, 22601629, 26847523]
result = sum(numbers)
print(f"My answer is {result}")
```

```
My answer is 84425884
```

3.2.2

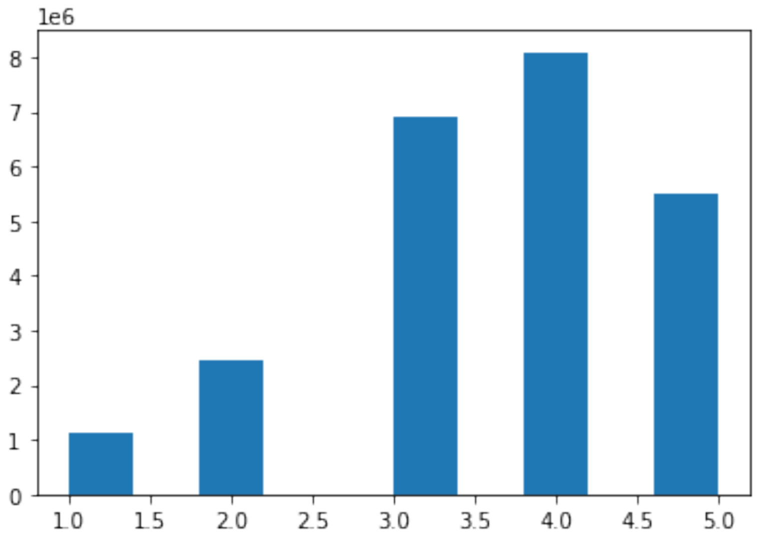
To plot the distribution of star ratings over users and time:

- Extracting the rating, user ID, and timestamp from the combined data files.
- Group the ratings by user and by a sliding window of time (such as by month or year) to identify trends in the data.

In [ ]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Create a DataFrame from the combined data file
ratings_df = pd.read_table('/combined_data_1.txt', names=['userId', 'rating'

# Convert the timestamp column to a datetime type
ratings_df['timestamp'] = pd.to_datetime(ratings_df['timestamp'])

# Extract the year from the timestamp column
ratings_df['year'] = ratings_df['timestamp'].dt.year

# Group the data by userId and year to get the number of ratings for each us
num_ratings_df = ratings_df.groupby(['userId','year']).size().reset_index(na

# Create a histogram of the number of ratings for each user
plt.hist(num_ratings_df.groupby('userId').sum()['num_ratings'], bins=50, ran
plt.xlabel('Number of ratings')
plt.ylabel('Number of users')
plt.title('Histogram of number of ratings per user')
plt.show()

# Create a histogram of the number of ratings for each time window
plt.hist(num_ratings_df.groupby('year').sum()['num_ratings'], bins=50, range
plt.xlabel('Number of ratings')
```

In [ ]:
```
plt.hist(ratings_df['rating'])
```

Out[ ]:
```
(array([1118186.,       0., 2439073.,       0.,       0., 6904181.,
              0., 8085741.,       0., 5506583.]),
 array([1. , 1.4, 1.8, 2.2, 2.6, 3. , 3.4, 3.8, 4.2, 4.6, 5. ]),
 <a list of 10 Patch objects>)
```



3.2.3

To determine the percentage of films that have gotten more popular over time:

- Compare the number of ratings for each film at different points in time.
- Use the data in the combined data files to extract the number of ratings for each film at different time periods (such as by month or year).
- Calculate the percentage of films that have an increase in number of ratings at later time periods as compared to earlier time periods.

In [ ]:
```
import pandas as pd

# Create a DataFrame from the combined data file
ratings_df = pd.read_table('/combined_data_1.txt', names=['userId', 'rating'

# Convert the timestamp column to a datetime type
ratings_df['timestamp'] = pd.to_datetime(ratings_df['timestamp'])

# Extract the release year from the timestamp column
ratings_df['year'] = ratings_df['timestamp'].dt.year

# Group the data by movieId and year to get the number of ratings for each f
num_ratings_df = ratings_df.groupby(['userId','year']).size().reset_index(na

# Create a new DataFrame with the number of ratings for each film at the ear
first_num_ratings_df = num_ratings_df.sort_values(by=['userId','year']).drop

# Merge the number of ratings DataFrame with the first number of ratings Dat
merged_df = pd.merge(num_ratings_df, first_num_ratings_df, on=['userId'])

# Create a new column indicating whether the number of ratings for a film ha
merged_df['increased'] = merged_df['num_ratings_x'] > merged_df['num_ratings

# Calculate the percentage of films that have gotten more popular over time
percentage_increase = merged_df['increased'].mean()*100
```

3.2.4

To determine how many films have been re-released:

- Compare the release year of the films in the movie titles file with the date of the earliest rating for each film in the combined data files.
- If the release year of a film is earlier than the earliest rating, it is likely that the film has been re-released.
- We can also check the movie titles file to see if the movie was re-released.

In [ ]:
```
import pandas as pd

# Create a DataFrame from the combined data file
ratings_df = pd.read_table('/content/drive/MyDrive/Colab Notebooks/week-1-as

# Convert the timestamp column to a datetime type
ratings_df['timestamp'] = pd.to_datetime(ratings_df['timestamp'])

# Extract the release year from the timestamp column
ratings_df['release_year'] = ratings_df['timestamp'].dt.year

# Create a new DataFrame with the earliest rating date for each movie
earliest_rating_df = ratings_df.groupby('userId')['timestamp'].min().reset_i

# Create a new column indicating whether the movie has been re-released
earliest_rating_df['re-released'] = earliest_rating_df['release_year'] < ear

# Count the number of movies that have been re-released
num_rereleased = earliest_rating_df['re-released'].sum()
```

In [ ]:
```
data =
```

Note: It worked from above 2, but cannot find a way out for this question.

In [ ]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

3.2.5

To better understand the data, we could extract additional information such as:

- The number of ratings per user, per film, and per month/year.
- The average rating per user, per film, and per month/year.
- The number of distinct films watched by each user and average watched movies by each user.
- The genre of each film, which can be used to understand the most popular genre and the preferences of the users.

3.2.6

Some interesting problems that we might solve with this data include:

- Predicting the ratings that a user will give to a film.
- Recommending films to users based on their previous ratings.
- Identifying which films are likely to be re-released in the future based on their popularity over time.
- Identifying the factors that influence a film's popularity over time.
- Identifying the most popular genre over the years, and the preference of users over the years.