Assignment 3

Part 1

```
def parse(path):
    g = gzip.open(path, 'rb')
    for l in g:
        yield json.loads(l)

def getDF(path):
    i = 0
    df = {}
    folder = parse(path)
    for d in folder:
        print(i)
        df[i] = d
        i += 1
    return df
```

```
df = getDF('Electronics_5.json.gz')
data_frame = pd.DataFrame.from_dict(df, orient='index')
data_frame.to_csv('data_frame.csv', index=False)
```

```
df = getDF('meta_Electronics.json.gz')
meta_df = pd.DataFrame.from_dict(df, orient='index')
meta_df.to_csv('meta_data.csv', index=False, escapechar='\\')
```

```
data = pd.read_csv("data_frame.csv")
meta_data = pd.read_csv("meta_data.csv")
```

Part 2

```python
product_indices = []
product = 'USB Cables'
for i in range(len(meta_data)):
    if product in meta_data['category'][i]:
        product_indices.append(i)
```

Part 3

1. Removed rows with NaN values in 'title' and 'asin' columns.
2. Removed duplicate rows.
3. Merged the meta-data and the review dataset with 'asin' as the joining column.

```python
product_meta_data_df = meta_data.iloc[product_indices]
product_meta_data_df = product_meta_data_df.dropna(subset=['title', 'asin'])
product_meta_data_df = product_meta_data_df.drop_duplicates()

product_df= pd.merge(data, product_meta_data_df, on='asin', how='inner')
product_df = product_df.dropna(subset=['overall'])
product_df = product_df.drop_duplicates()
```

Number of rows for the product: 124237

Part 4

```
a. Number of Reviews: 124237
b. Average Rating Score: 4.365503369037961
c. Number of Unique Products: 2591
d. Number of Good Ratings: 107710
e. Number of Bad Ratings: 13096
f. Number of Reviews corresponding to each Rating:
overall
1.0     8621
2.0     4475
3.0     6809
4.0    15124
5.0    85777
Name: count, dtype: int64
```

Part 5

1. Functions used for pre-processing the data

```
> def expand_acronyms(text): ...
> def remove_html_tags(text): ...
> def remove_accented_chars(text): ...
> def remove_special_characters(text): ...
> def lemmatize_text(text): ...
  def normalize_text(text):
      text = text.lower()
      text = expand_acronyms(text)
      text = remove_html_tags(text)
      text = remove_accented_chars(text)
      text = remove_special_characters(text)
      text = lemmatize_text(text)
      return text
```

Part 6

A. Top 20 most reviewed brands:

brand	
AmazonBasics	15395
Cable Matters	8674
Mediabridge	5585
StarTech	5052
Anker	3914
Belkin	3622
INSTEN	2682
Monoprice	2526
Generic	2093
iSeekerKit	1631
C2G	1466
Sabrent	1394
Samsung	1302
C&E	1212
Mission Cables	1169
Tripp Lite	1168
TRENDnet	1124
CableCreation	1045
UGREEN	1037
Eversame	989

B. Top 20 least reviewed brands:

brand	
Innovate Motorsports	5
CoverON	5
Philips	5
D-Link	5
maxinbuy	5
F.DORLA	5
KKmoon	5
Clarion	5
Weiup	5
SODIAL	5
LINESO	5
WIT Inc.	5
X-EDITION	5
TEVIWIN	5
HQRP	5
TTMSTUFF	5
BRIGHTSHOW	5
MRC	4

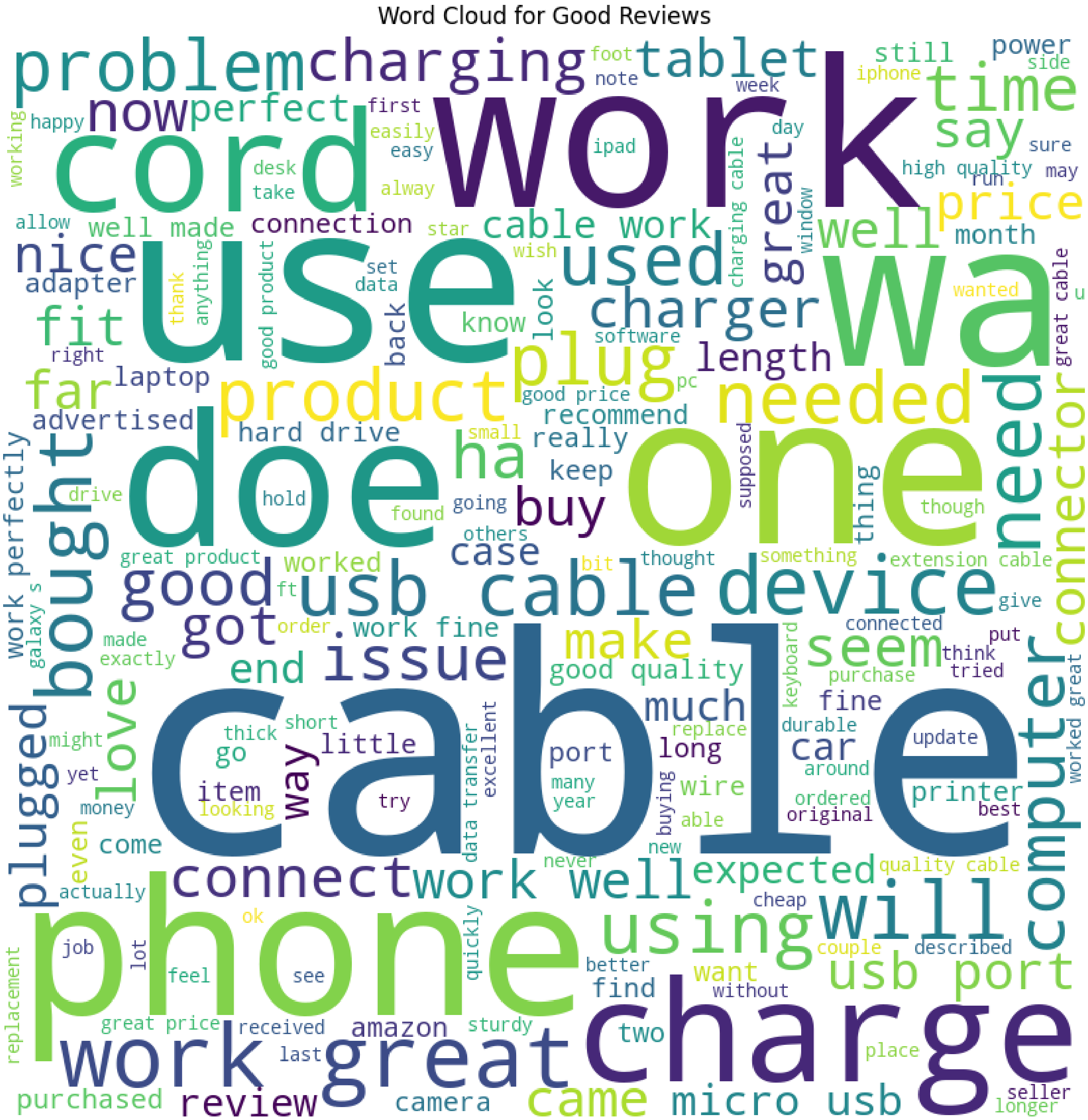
Danibos 4
Star Toys 1

C. Most positively reviewed USB Cable:
title
(2 Pack) USB 2.0 a to Mini 5 Pin B Cable for External Hdds/camera/card Readers (75cm - 2 Foot - 0.75m) 5.0

D.

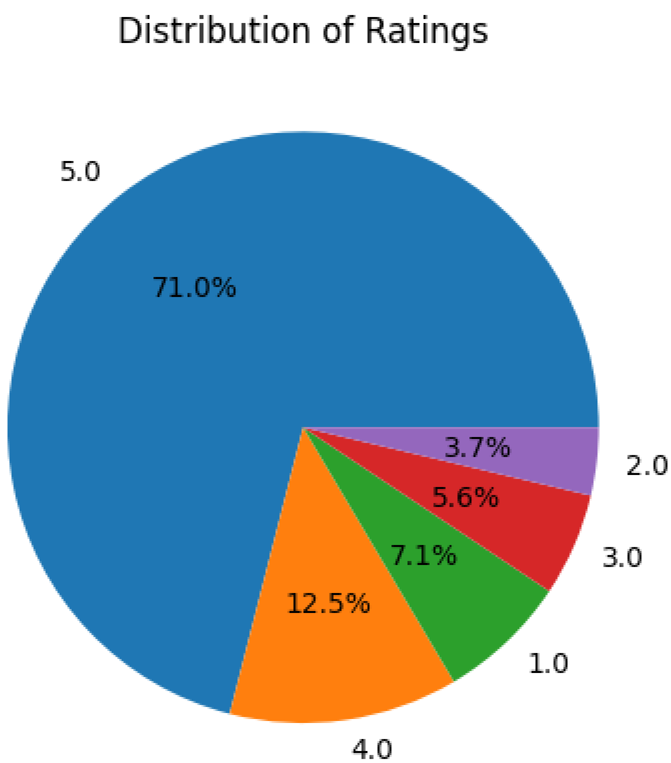
```
1999-12-31      216
2000-12-31       12
2001-12-31     2403
2002-12-31      535
2003-12-31     1217
2004-12-31     3250
2005-12-31     2605
2006-12-31     1595
2007-12-31     1638
2008-12-31     5356
2009-12-31     2991
2010-12-31     4876
2011-12-31     8620
2012-12-31     4931
2013-12-31    10326
2014-12-31     9444
2015-12-31    18976
2016-12-31     7620
2017-12-31     2817
2018-12-31     1533
```

E. Good Review Ratings Cloud



Bas Review Ratings Cloud

Word Cloud for Bad Reviews



F.

G. Year with maximum reviews: 2015.0

H. Year with the highest number of customers: 2015.0

Part 7

1. I have used tf-idf vectorizer to create word embeddings.

```
tfidf_vectorizer = TfidfVectorizer()
X = tfidf_vectorizer.fit_transform(product_df['reviewText'])
```

Part 8

```
good_threshold = 4.0
bad_threshold = 3.0

product_df = product_df.dropna(subset=['overall', 'reviewText'])
product_df['Rating Class'] = product_df['overall'].apply(lambda x: 'Good' if x >= good_threshold else ('Average' if x == bad_threshold else 'Bad'))
```

Part 9

1. I have sampled the good reviews to reduce the size of the dataset for faster computations.

```
label_encoder = LabelEncoder()
y = label_encoder.fit_transform(product_df['Rating Class'])

label_mapping = dict(zip(label_encoder.classes_, label_encoder.transform(label_encoder.classes_)))
print("Label Mapping:")
print(label_mapping)

tfidf_vectorizer = TfidfVectorizer()
X = tfidf_vectorizer.fit_transform(product_df['reviewText'])


indices_y0 = np.where(y == 0)[0]
indices_y1 = np.where(y == 1)[0]
indices_y2 = np.where(y == 2)[0]

num_indices_to_select = int(0.1 * len(indices_y2))

selected_indices_y2 = np.random.choice(indices_y2, size=num_indices_to_select, replace=False)

sampled_indices = np.concatenate([indices_y0, indices_y1, selected_indices_y2])

sampled_X = X[sampled_indices]
sampled_y = y[sampled_indices]
# print(sampled_y.shape, sampled_X.shape)

X_train, X_test, y_train, y_test = train_test_split(sampled_X, sampled_y, test_size=0.25, random_state=42)
```

Part 10

Bad Class

Model Name	Precision	Recall	F1-Score	Support
Logistic Regression	0.23	0.04	0.07	1695
SVC	0.34	0.01	0.02	1695
Random Forest	0.24	0.04	0.06	1695
Decision Trees	0.23	0.19	0.21	1695
Naive Bayes	0	0	0	1695

Average Class

Model Name	Precision	Recall	F1-Score	Support
Logistic Regression	0.45	0.72	0.56	3305
SVC	0.45	0.84	0.58	3305
Random Forest	0.45	0.79	0.57	3305
Decision Trees	0.45	0.50	0.47	3305
Naive Bayes	0.44	0.99	0.61	3305

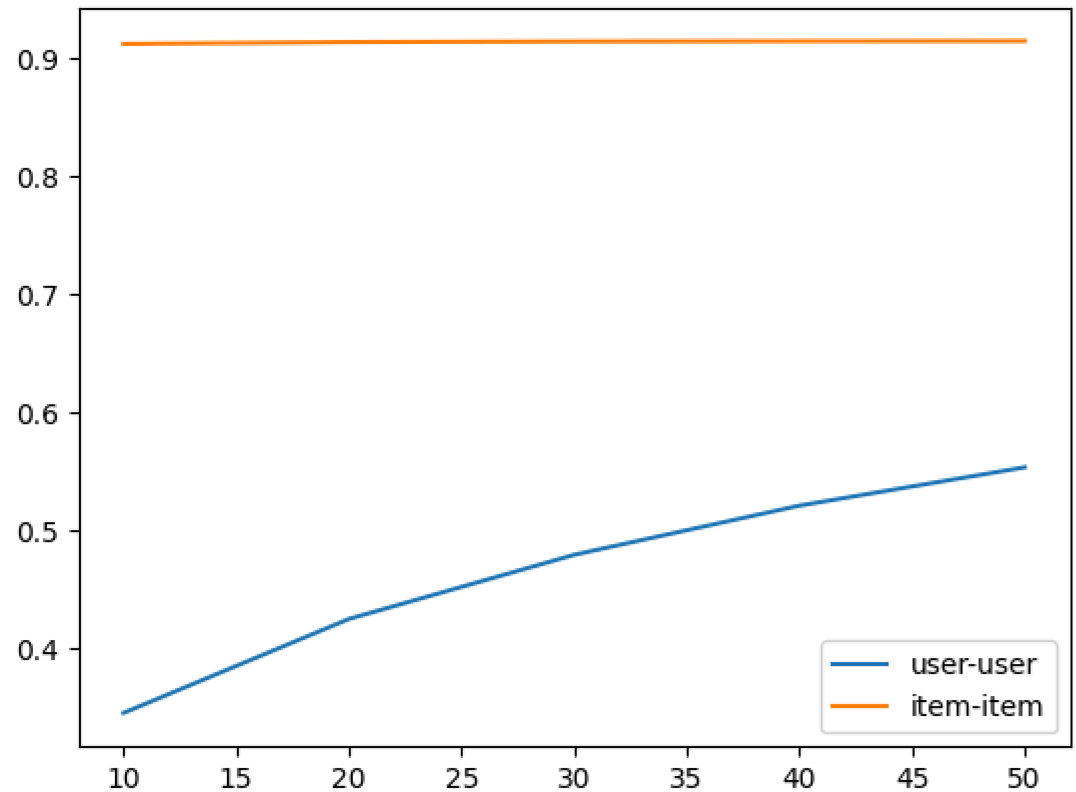
Good Class

Model Name	Precision	Recall	F1-Score	Support
Logistic Regression	0.36	0.27	0.31	2499
SVC	0.37	0.19	0.25	2499
Random Forest	0.35	0.19	0.25	2499
Decision Trees	0.34	0.32	0.33	2499
Naive Bayes	0.34	0.02	0.03	2499

Analysis:

- Bad Class: The Decision Trees model has the highest F1-Score (0.21), followed by Logistic Regression (0.07). Naive Bayes performed poorly with an F1-Score of 0.
- Average Class: Naive Bayes has the highest F1-Score (0.61), followed closely by SVC (0.58). Decision Trees have the lowest F1-Score (0.47).
- Good Class: Logistic Regression has the highest F1-Score (0.31), followed by SVC (0.25). Naive Bayes performed poorly with an F1-Score of 0.03.
- In summary, Naive Bayes generally performed well in the Average class but poorly in the other two classes. SVC and Logistic Regression showed consistent performance across different classes, while Decision Trees had mixed results. Overall, the performance varied across models and classes, suggesting the importance of selecting the appropriate model for each specific class.

Part 11



User-User Collaborative filtering - [0.3456919945725916, 0.42525629428614503, 0.4795806321925725, 0.5212187170209559, 0.5536687773254937]
Item-Item Collaborative filtering - [0.9115651135005973, 0.9131600955794504, 0.9136917562724014, 0.9139575866188769, 0.9141170848267621]

Part 12

```
Top 10 products by user sum ratings:
title
AmazonBasics USB 3.0 Cable - A-Male to Micro-B - 3 Feet (0.9 Meters)                                                    10962.0
AmazonBasics Micro-USB to USB 2.0 Cable - 6 Feet (1.8 Meters)                                                            10761.0
AmazonBasics USB 3.0 Extension Cable - A-Male to A-Female - 3 Feet (2 Pack)                                               8305.0
Mediabridge USB 2.0 - A Male to B Male Cable (10 Feet) - High-Speed w/ Gold-Plated Connectors - Black (Part# 30-001-10B )  7909.0
Insten Micro USB OTG to USB 2.0 Adapter Cable Compatible With Samsung Galaxy S7/S6/S6 Edge/Note 4/3                        6637.0
10 ft Micro USB Cable - A to Micro B                                                                                      6532.0
Mediabridge USB 2.0 - USB Extension Cable (6 Feet) - A Male to A Female with Gold-Plated Contacts                          6344.0
Anker 6ft / 1.8m Nylon Braided Tangle-Free Micro USB Cable with Gold-Plated Connectors for Android, Samsung, HTC, Nokia, Sony and More (Red)     6045.0
Cable Matters 2-Pack Micro USB 3.0 Cable (Micro USB 3 Cable A to Micro B) in Black 3 Feet                                 5193.0
AmazonBasics USB 2.0 Cable - A-Male to Mini-B - 6 Feet (1.8 Meters)                                                       5025.0
Name: overall, dtype: float64
```