In [19]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import math
import warnings
warnings.filterwarnings('ignore')
warnings.filterwarnings("ignore", category=DeprecationWarning)
warnings.filterwarnings("ignore",category=UserWarning)
sns.set_style("whitegrid")
%matplotlib inline
np.random.seed(7)

csv = "amazon.csv"
df = pd.read_csv(csv)
df.head(10)
```

Out[19]:

	id	dateAdded	dateUpdated	name	asins
0	AVpgNzjwLJeJML43Kpxn	2015-10-30T08:59:32Z	2019-04-25T09:08:16Z	AmazonBasics AAA Performance Alkaline Batterie...	B00QWO9P0O,B00LH3DMUC
1	AVpgNzjwLJeJML43Kpxn	2015-10-30T08:59:32Z	2019-04-25T09:08:16Z	AmazonBasics AAA Performance Alkaline Batterie...	B00QWO9P0O,B00LH3DMUC
2	AVpgNzjwLJeJML43Kpxn	2015-10-30T08:59:32Z	2019-04-25T09:08:16Z	AmazonBasics AAA Performance Alkaline Batterie...	B00QWO9P0O,B00LH3DMUC
3	AVpgNzjwLJeJML43Kpxn	2015-10-30T08:59:32Z	2019-04-25T09:08:16Z	AmazonBasics AAA Performance Alkaline Batterie...	B00QWO9P0O,B00LH3DMUC
4	AVpgNzjwLJeJML43Kpxn	2015-10-30T08:59:32Z	2019-04-25T09:08:16Z	AmazonBasics AAA Performance Alkaline Batterie...	B00QWO9P0O,B00LH3DMUC
5	AVpgNzjwLJeJML43Kpxn	2015-10-30T08:59:32Z	2019-04-25T09:08:16Z	AmazonBasics AAA Performance Alkaline Batterie...	B00QWO9P0O,B00LH3DMUC
6	AVpgNzjwLJeJML43Kpxn	2015-10-30T08:59:32Z	2019-04-25T09:08:16Z	AmazonBasics AAA Performance Alkaline Batterie...	B00QWO9P0O,B00LH3DMUC

	id	dateAdded	dateUpdated	name	asins
7	AVpgNzjwLJeJML43Kpxn	2015-10-30T08:59:32Z	2019-04-25T09:08:16Z	AmazonBasics AAA Performance Alkaline Batterie...	B00QWO9P0O,B00LH3DMUC
8	AVpgNzjwLJeJML43Kpxn	2015-10-30T08:59:32Z	2019-04-25T09:08:16Z	AmazonBasics AAA Performance Alkaline Batterie...	B00QWO9P0O,B00LH3DMUC
9	AVpgNzjwLJeJML43Kpxn	2015-10-30T08:59:32Z	2019-04-25T09:08:16Z	AmazonBasics AAA Performance Alkaline Batterie...	B00QWO9P0O,B00LH3DMUC

10 rows × 24 columns

In [3]:
```
data = df.copy()
data.describe()
```

Out[3]:

	reviews.id	reviews.numHelpful	reviews.rating
count	4.100000e+01	16115.000000	28332.000000
mean	1.840066e+08	0.529321	4.514048
std	2.337036e+07	9.345017	0.934957
min	1.116244e+08	0.000000	1.000000
25%	1.843344e+08	0.000000	4.000000
50%	1.885078e+08	0.000000	5.000000
75%	1.988160e+08	0.000000	5.000000
max	2.085304e+08	621.000000	5.000000

In [4]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 28332 entries, 0 to 28331
Data columns (total 24 columns):
 #   Column               Non-Null Count  Dtype 
---  ------               --------------  ----- 
 0   id                   28332 non-null  object
 1   dateAdded            28332 non-null  object
 2   dateUpdated          28332 non-null  object
 3   name                 28332 non-null  object
 4   asins                28332 non-null  object
 5   brand                28332 non-null  object
 6   categories           28332 non-null  object
 7   primaryCategories    28332 non-null  object
 8   imageURLs            28332 non-null  object
 9   keys                 28332 non-null  object
 10  manufacturer         28332 non-null  object
 11  manufacturerNumber   28332 non-null  object
 12  reviews.date         28332 non-null  object
 13  reviews.dateSeen     28332 non-null  object
```

```
 14  reviews.didPurchase  9 non-null      object
 15  reviews.doRecommend  16086 non-null  object
 16  reviews.id           41 non-null      float64
 17  reviews.numHelpful   16115 non-null  float64
 18  reviews.rating       28332 non-null  int64
 19  reviews.sourceURLs   28332 non-null  object
 20  reviews.text         28332 non-null  object
 21  reviews.title        28332 non-null  object
 22  reviews.username     28332 non-null  object
 23  sourceURLs           28332 non-null  object
dtypes: float64(2), int64(1), object(21)
memory usage: 5.2+ MB
```

In [5]:
```
data["asins"].unique()
```

Out[5]:
```
array(['B00QWO9P0O,B00LH3DMUO', 'B00DIHVMEA,B00EZ1ZTV0',
       'B01E6AO69U,B00L9EPT8O', 'B01J24C0TI', 'B073SQYXTW', 'B00ZV9RDKK',
       'B00QWO9P0O,B01IB83NZG,B00MNV8E0C', 'B00WRDS8H0',
       'B00EEBS9O0,B01CHQHIJK', 'B01B66989K,B00CD8ADKO,B00LA9H6UM',
       'B00DUGZFWY', 'B00F5CKWBA,B00KPQCWAU',
       'B0002LCUZK,B010CEC6MI,B01B25NN64', 'B074MCBG25,B075357QFB',
       'B00QFQRELG', 'B006GWO7UA', 'B01L7XWEQQ', 'B006BGZJJ4',
       'B00Y3QOH5G', 'B01BH83OOM', 'B00ZV9PXP2', 'B00NH144GK,B00LA9H1E8',
       'B00OP6SMCI', 'B00BGIQS1A', 'B006LW0W5Y', 'B0751RGYJV',
       'B00IOY8XWQ', 'B010RLCH2U', 'B01GAGYVU2', 'B0752151W6',
       'B018SZT3BK,B01AHB9CN2', 'B018Y226XO', 'B01AHB9CYG', 'B01ACEKAJY',
       'B00REQKWGA', 'B00IOYAM4I', 'B01IO618J8', 'B018Y22BI4',
       'B01AHBBG04', 'B01AHBDCKQ', 'B00VINDBJK',
       'B0189XZRTI,B0189XYY0Q,B0189XZ0KY', 'B01J94SWWU', 'B00QAVO43C',
       'B06VTJWRJW', 'B00QAVNWSK', 'B01J94SCAM', 'B01J94SBEY',
       'B01J94YIT6', 'B01J94T1Z2', 'B018Y224PY', 'B00VKTZFB4',
       'B018Y225IA', 'B00ZS0G0PG', 'B06XD5YCKX', 'B018Y22C2Y',
       'B01AHB9C1E', 'B018Y23MNM', 'B017JG41PC', 'B00XNQECFM',
       'B00UH4D8G2', 'B06XCWLL12', 'B005OOKNP4', 'B001NIZB5M',
       'B018T075DC'], dtype=object)
```

In [143…
```
data.columns
```

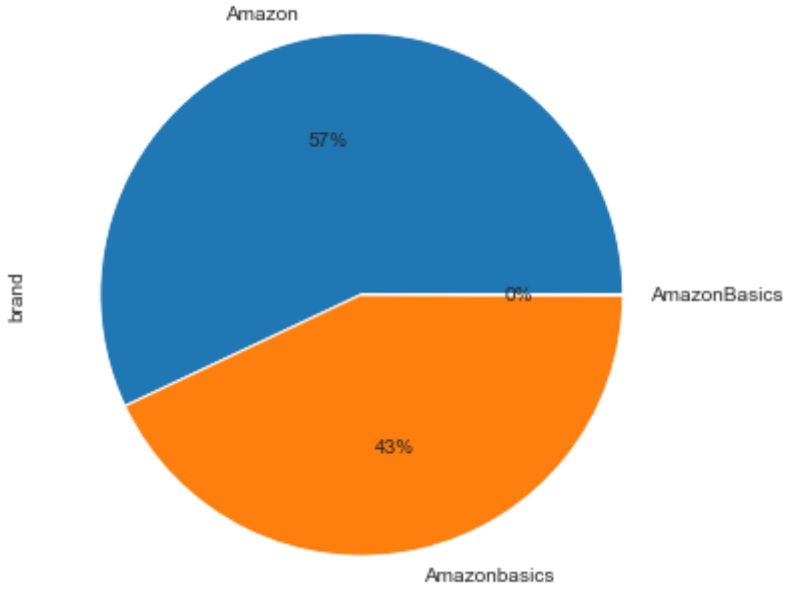
Out[143…
```
Index(['name', 'brand'], dtype='object')
```

In [6]:
```
asins_unique = len(data["asins"].unique())
print("Number of Unique ASINs: " + str(asins_unique))
```

```
Number of Unique ASINs: 65
```

In [145…
```
data.brand.value_counts().plot(kind='pie', autopct='%1.0f%%')
```

Out[145…
```
<AxesSubplot:ylabel='brand'>
```



In [9]:
```
split = StratifiedShuffleSplit(n_splits=5, test_size=0.2)
for train_index, test_index in split.split(dataAfter,
                                           dataAfter["reviews.rating"]):
    strat_train = dataAfter.reindex(train_index)
    strat_test = dataAfter.reindex(test_index)
```

In [10]:
```
len(strat_train)
```

Out[10]:
```
22665
```

In [11]:
```
strat_train["reviews.rating"].value_counts()/len(strat_train)
```

Out[11]:
```
5    0.702272
4    0.199338
3    0.042577
1    0.034061
2    0.021752
Name: reviews.rating, dtype: float64
```

In [12]:
```
len(strat_test)
```

Out[12]:
```
5667
```

In [13]:
```
strat_test["reviews.rating"].value_counts()/len(strat_test)
```

Out[13]:
```
5    0.702312
4    0.199400
3    0.042527
1    0.034057
2    0.021705
Name: reviews.rating, dtype: float64
```

In [14]:
```
reviews = strat_train.copy()
reviews.head(2)
```

Out[14]:

	id	dateAdded	dateUpdated	name

	id	dateAdded	dateUpdated	name	
26762	AVqkIhxunnc1JgDc3kg_	2017-03-06T14:59:43Z	2019-02-23T02:49:38Z	Fire HD 8 Tablet with Alexa, 8 HD Display, 16 ...	B018T(
5922	AVpgNzjwLJeJML43Kpxn	2015-10-30T08:59:32Z	2019-04-25T09:08:16Z	AmazonBasics AAA Performance Alkaline Batterie...	B00QWO9P0O,B00LH3I

2 rows × 24 columns

In [15]:
```
len(reviews["name"].unique()), len(reviews["asins"].unique())
```

Out[15]:
```
(64, 64)
```

In [16]:
```
reviews.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 22665 entries, 26762 to 7111
Data columns (total 24 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   id                    22665 non-null  object 
 1   dateAdded             22665 non-null  object 
 2   dateUpdated           22665 non-null  object 
 3   name                  22665 non-null  object 
 4   asins                 22665 non-null  object 
 5   brand                 22665 non-null  object 
 6   categories            22665 non-null  object 
 7   primaryCategories     22665 non-null  object 
 8   imageURLs             22665 non-null  object 
 9   keys                  22665 non-null  object 
 10  manufacturer          22665 non-null  object 
 11  manufacturerNumber    22665 non-null  object 
 12  reviews.date          22665 non-null  object 
 13  reviews.dateSeen      22665 non-null  object 
 14  reviews.didPurchase   7 non-null      object 
 15  reviews.doRecommend   12896 non-null  object 
 16  reviews.id            34 non-null     float64
 17  reviews.numHelpful    12920 non-null  float64
 18  reviews.rating        22665 non-null  int64  
 19  reviews.sourceURLs    22665 non-null  object 
 20  reviews.text          22665 non-null  object 
 21  reviews.title         22665 non-null  object 
 22  reviews.username      22665 non-null  object 
 23  sourceURLs            22665 non-null  object 
dtypes: float64(2), int64(1), object(21)
memory usage: 4.3+ MB
```

In [17]:
```
reviews.groupby("asins")["name"].unique()
```

Out[17]:
```
asins
B0002LCUZK,B010CEC6MI,B01B25NN64    [Expanding Accordion File Folder Plastic Por
ta...
B001NIZB5M                          [Amazon Kindle Replacement Power Adapter (Fi
ts...
B005OOKNP4                          [AmazonBasics Bluetooth Keyboard for Android
```

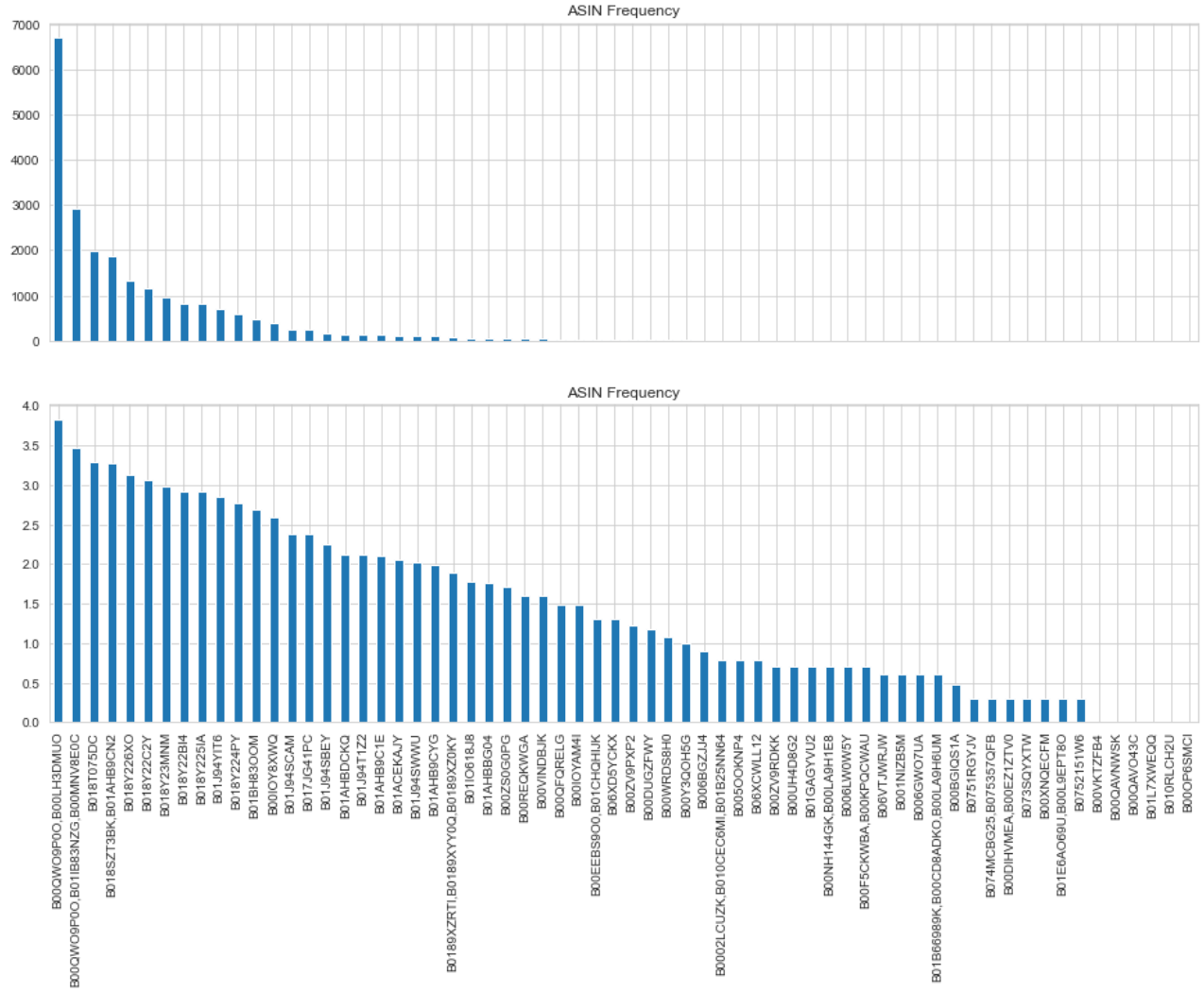
```
D...
B006BGZJJ4                          [Amazon Kindle Charger Power Adapter Wall Ch
ar...
B006GWO7UA                          [Kindle PowerFast International Charging Kit
(...
                                                                ...
B06XD5YCKX                          [All-New Kindle Oasis E-reader - 7 High-Reso
lu...
B073SQYXTW                                                  [Echo Spot Pair Kit (Bl
ack)]
B074MCBG25,B075357QFB               [Cat Litter Box Covered Tray Kitten Extra La
rg...
B0751RGYJV                          [Amazon Echo (2nd Generation) Smart Assistan
t ...
B0752151W6                          [All-new Echo (2nd Generation) with improved
s...
Name: name, Length: 64, dtype: object
```

In [18]:
```
different_names = reviews[reviews["asins"] ==
                          "B00L9EPT8O,B01E6AO69U"]["name"].unique()
for name in different_names:
    print(name)
```

In [19]:
```
reviews[reviews["asins"] == "B00L9EPT8O,B01E6AO69U"]["name"].value_counts()
```

Out[19]:
```
Series([], Name: name, dtype: int64)
```

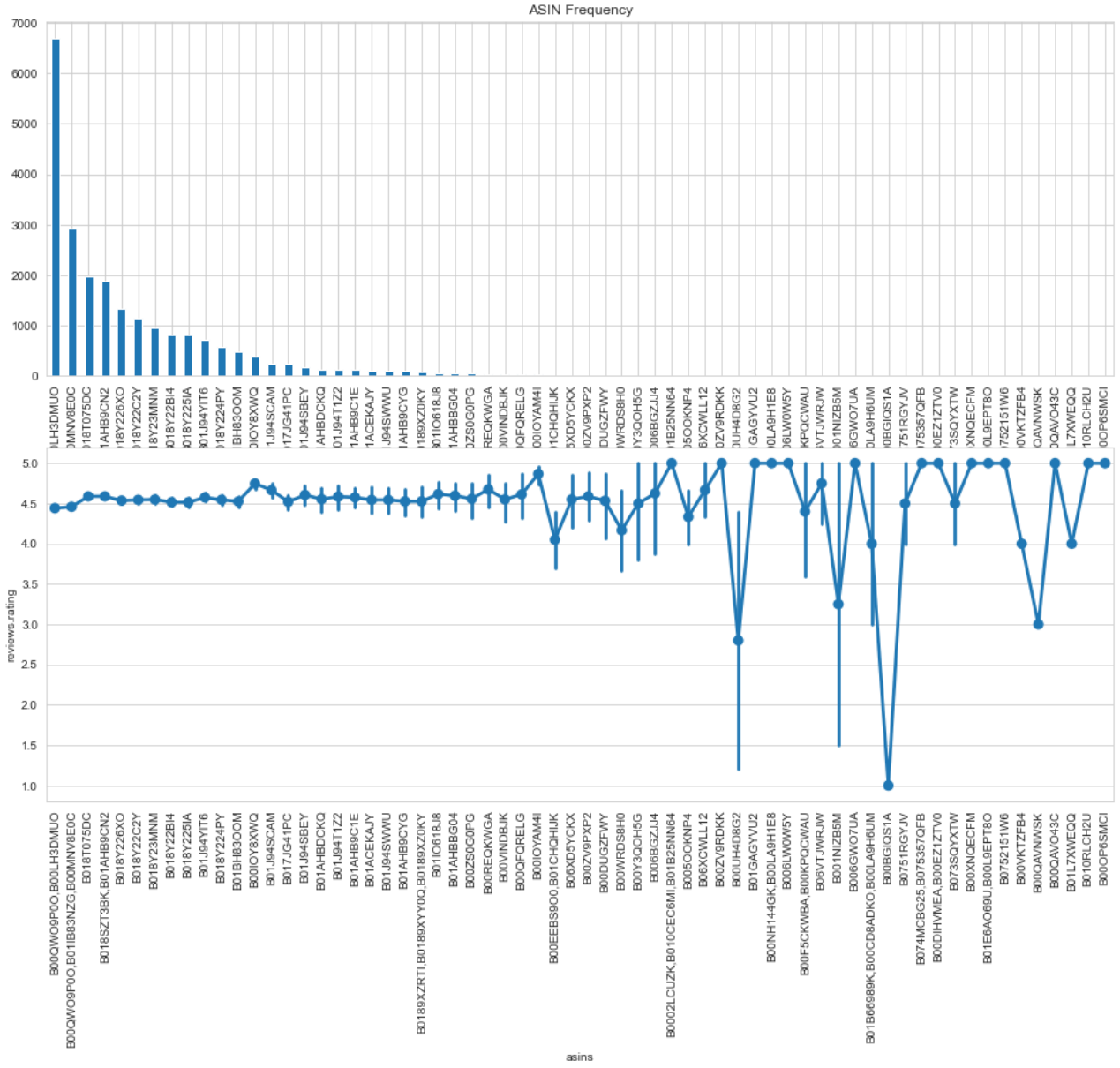
In [22]:
```
fig = plt.figure(figsize=(16,10))
ax1 = plt.subplot(211)
ax2 = plt.subplot(212, sharex = ax1)
reviews["asins"].value_counts().plot(kind="bar", ax=ax1, title="ASIN Frequency")
np.log10(reviews["asins"].value_counts()).plot(kind="bar", ax=ax2,
                                               title="ASIN Frequency")
plt.show()
```



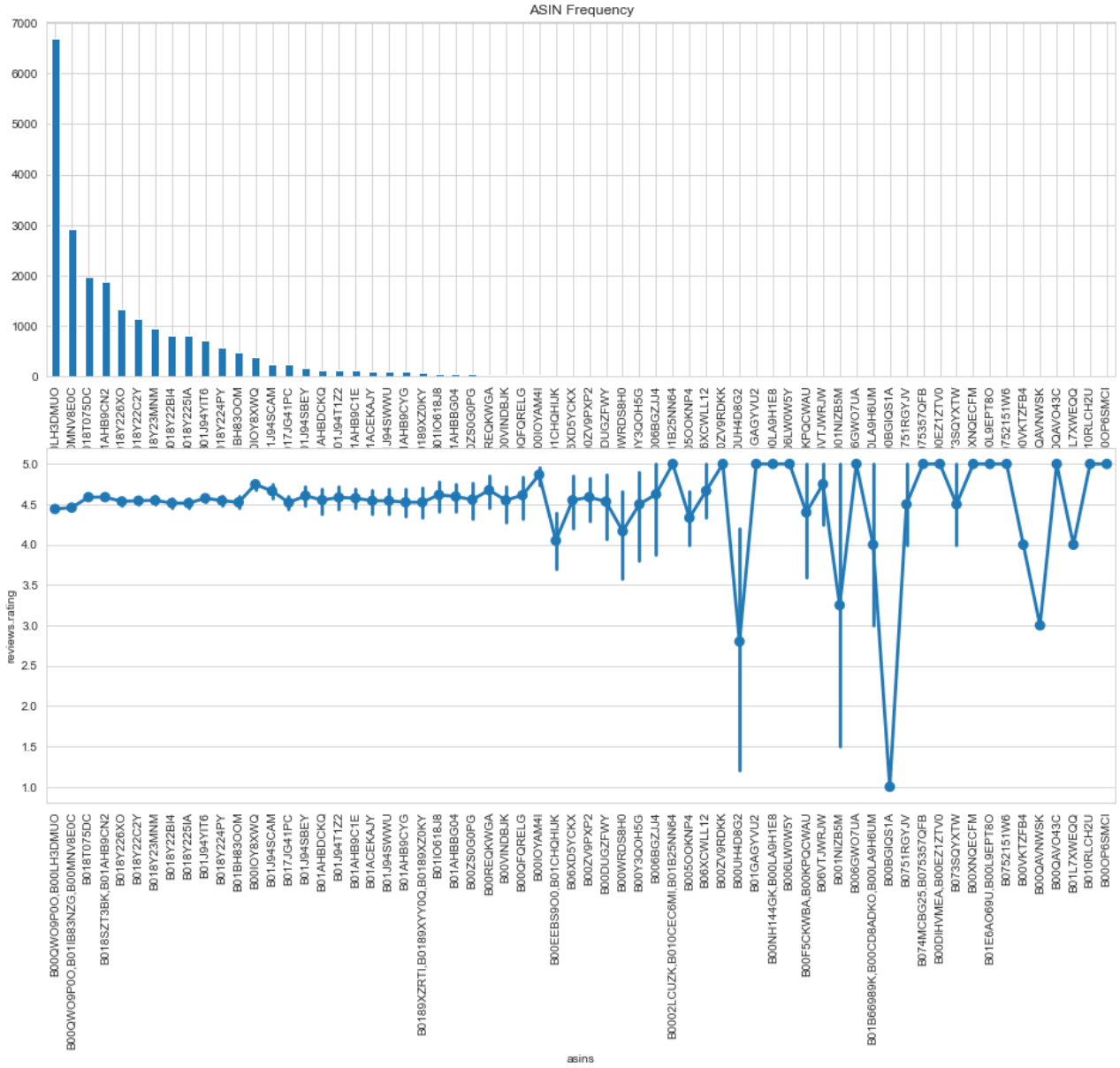
In [23]:
```
reviews["reviews.rating"].mean()
```

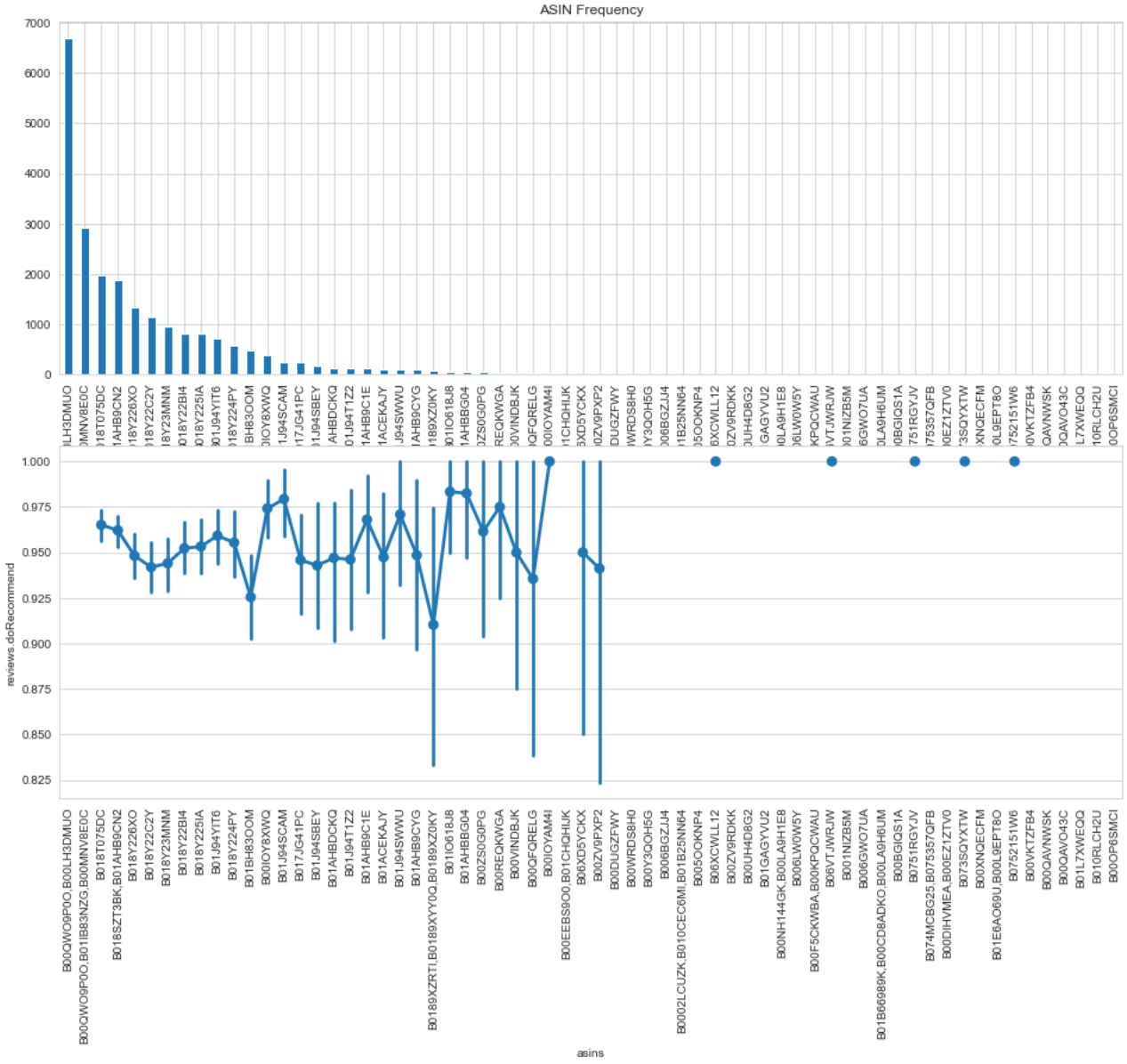
Out[23]:
```
4.514008382969336
```

In [24]:
```
asins_count_ix = reviews["asins"].value_counts().index
plt.subplots(2,1,figsize=(16,12))
plt.subplot(2,1,1)
reviews["asins"].value_counts().plot(kind="bar", title="ASIN Frequency")
plt.subplot(2,1,2)
sns.pointplot(x="asins", y="reviews.rating", order=asins_count_ix, data=reviews)
plt.xticks(rotation=90)
plt.show()
```



In [26]:
```
asins_count_ix = reviews["asins"].value_counts().index
plt.subplots(2,1,figsize=(16,12))
plt.subplot(2,1,1)
reviews["asins"].value_counts().plot(kind="bar", title="ASIN Frequency")
plt.subplot(2,1,2)
sns.pointplot(x="asins", y="reviews.rating", order=asins_count_ix, data=reviews)
plt.xticks(rotation=90)
plt.show()
plt.subplots (2,1,figsize=(16,12))
plt.subplot(2,1,1)
reviews["asins"].value_counts().plot(kind="bar", title="ASIN Frequency")
plt.subplot(2,1,2)
sns.pointplot(x="asins", y="reviews.doRecommend", order=asins_count_ix,
              data=reviews)
plt.xticks(rotation=90)
plt.show()
```


ASIN Frequency
reviews.rating
asins



In [27]:
```
corr_matrix = reviews.corr()
corr_matrix
```

Out[27]:

	reviews.id	reviews.numHelpful	reviews.rating
reviews.id	1.000000	-0.545603	0.063057
reviews.numHelpful	-0.545603	1.000000	-0.026607
reviews.rating	0.063057	-0.026607	1.000000

In [28]:
```
reviews.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 22665 entries, 26762 to 7111
Data columns (total 24 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   id                    22665 non-null  object
 1   dateAdded             22665 non-null  object
 2   dateUpdated           22665 non-null  object
 3   name                  22665 non-null  object
 4   asins                 22665 non-null  object
```

```
 5   brand                 22665 non-null  object
 6   categories            22665 non-null  object
 7   primaryCategories     22665 non-null  object
 8   imageURLs             22665 non-null  object
 9   keys                  22665 non-null  object
 10  manufacturer          22665 non-null  object
 11  manufacturerNumber    22665 non-null  object
 12  reviews.date          22665 non-null  object
 13  reviews.dateSeen      22665 non-null  object
 14  reviews.didPurchase   7 non-null      object
 15  reviews.doRecommend   12896 non-null  object
 16  reviews.id            34 non-null     float64
 17  reviews.numHelpful    12920 non-null  float64
 18  reviews.rating        22665 non-null  int64
 19  reviews.sourceURLs    22665 non-null  object
 20  reviews.text          22665 non-null  object
 21  reviews.title         22665 non-null  object
 22  reviews.username      22665 non-null  object
 23  sourceURLs            22665 non-null  object
dtypes: float64(2), int64(1), object(21)
memory usage: 4.9+ MB
```

In [29]:
```
counts = reviews["asins"].value_counts().to_frame()
counts.head()
```

Out[29]:

	asins
B00QWO9P0O,B00LH3DMUO	6692
B00QWO9P0O,B01IB83NZG,B00MNV8E0C	2936
B018T075DC	1979
B018SZT3BK,B01AHB9CN2	1875
B018Y226XO	1331

In [30]:
```
avg_rating = reviews.groupby("asins")["reviews.rating"].mean().to_frame()
avg_rating.head()
```

Out[30]:

	reviews.rating
asins	
B0002LCUZK,B010CEC6MI,B01B25NN64	5.000000
B001NIZB5M	3.250000
B005OOKNP4	4.333333
B006BGZJJ4	4.625000
B006GWO7UA	5.000000

In [31]:
```
table = counts.join(avg_rating)
table.head(30)
```

Out[31]:

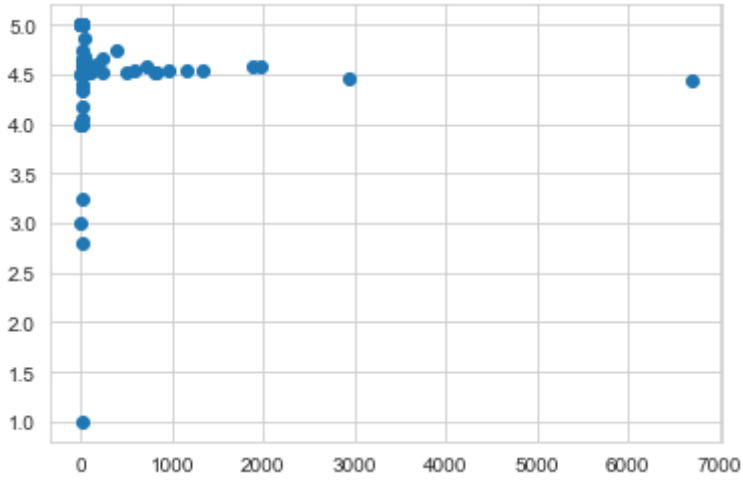
	asins	reviews.rating
B00QWO9P0O,B00LH3DMUO	6692	4.442170
B00QWO9P0O,B01IB83NZG,B00MNV8E0C	2936	4.458787
B018T075DC	1979	4.588176

	asins	reviews.rating
B018SZT3BK,B01AHB9CN2	1875	4.588267
B018Y226XO	1331	4.535687
B018Y22C2Y	1153	4.544666
B018Y23MNM	965	4.548187
B018Y22BI4	817	4.515300
B018Y225IA	813	4.514145
B01J94YIT6	711	4.575246
B018Y224PY	583	4.545455
B01BH83OOM	488	4.524590
B00IOY8XWQ	387	4.746770
B01J94SCAM	242	4.665289
B017JG41PC	240	4.520833
B01J94SBEY	175	4.605714
B01AHBDCKQ	132	4.553030
B01J94T1Z2	130	4.584615
B01AHB9C1E	125	4.576000
B01ACEKAJY	114	4.543860
B01J94SWWU	103	4.543689
B01AHB9CYG	97	4.525773
B0189XZRTI,B0189XYY0Q,B0189XZ0KY	78	4.525641
B01IO618J8	60	4.616667
B01AHBBG04	57	4.596491
B00ZS0G0PG	52	4.557692
B00REQKWGA	40	4.675000
B00VINDBJK	40	4.550000
B00QFQRELG	31	4.612903
B00IOYAM4I	30	4.866667

In [32]:
```
plt.scatter("asins", "reviews.rating", data=table)
table.corr()
```

Out[32]:

	asins	reviews.rating
asins	1.000000	0.013432
reviews.rating	0.013432	1.000000



In [33]:
```
`
```

Out[33]:
```
26762    Positive
5922     Negative
15017    Positive
2977     Positive
22761     Neutral
26482    Positive
7861     Positive
21027    Positive
16242    Positive
13791    Positive
6434     Positive
16813    Positive
22       Positive
26772    Positive
18775    Positive
648      Positive
15522    Positive
16700    Positive
4423     Positive
11506    Positive
Name: Sentiment, dtype: object
```

In [35]:
```
# Prepare data
X_train = strat_train["reviews.text"]
X_train_targetSentiment = strat_train["Sentiment"]
X_test = strat_test["reviews.text"]
X_test_targetSentiment = strat_test["Sentiment"]
print(len(X_train), len(X_test))
```

```
22665 5667
```

In [36]:
```
X_train = X_train.fillna(' ')
X_test = X_test.fillna(' ')
X_train_targetSentiment = X_train_targetSentiment.fillna(' ')
X_test_targetSentiment = X_test_targetSentiment.fillna(' ')

# Text preprocessing and occurance counting
from sklearn.feature_extraction.text import CountVectorizer
count_vect = CountVectorizer()
X_train_counts = count_vect.fit_transform(X_train)
X_train_counts.shape
```

Out[36]:
```
(22665, 9704)
```

In [37]:
```
from sklearn.feature_extraction.text import TfidfTransformer
tfidf_transformer = TfidfTransformer(use_idf=False)
X_train_tfidf = tfidf_transformer.fit_transform(X_train_counts)
X_train_tfidf.shape
```

Out[37]:
```
(22665, 9704)
```

In [38]:
```
from sklearn.naive_bayes import MultinomialNB
from sklearn.pipeline import Pipeline
clf_multiNB_pipe = Pipeline([("vect", CountVectorizer()),
                             ("tfidf", TfidfTransformer()),
                             ("clf_nominalNB", MultinomialNB())])
clf_multiNB_pipe.fit(X_train, X_train_targetSentiment)
```

Out[38]:
```
Pipeline(steps=[('vect', CountVectorizer()), ('tfidf', TfidfTransformer()),
                ('clf_nominalNB', MultinomialNB())])
```

In [48]:
```
import numpy as np
predictedMultiNB = clf_multiNB_pipe.predict(X_test)
np.mean(predictedMultiNB == X_test_targetSentiment)
```

Out[48]:
```
0.9025939650608787
```

In [49]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import Pipeline
clf_logReg_pipe = Pipeline([("vect", CountVectorizer()),
                            ("tfidf", TfidfTransformer()),
                            ("clf_logReg", LogisticRegression())])
clf_logReg_pipe.fit(X_train, X_train_targetSentiment)

import numpy as np
predictedLogReg = clf_logReg_pipe.predict(X_test)
np.mean(predictedLogReg == X_test_targetSentiment)
```

Out[49]:
```
0.9313569790012353
```

In [50]:
```
from sklearn.svm import LinearSVC
clf_linearSVC_pipe = Pipeline([("vect", CountVectorizer()),
                               ("tfidf", TfidfTransformer()),
                               ("clf_linearSVC", LinearSVC())])
clf_linearSVC_pipe.fit(X_train, X_train_targetSentiment)

predictedLinearSVC = clf_linearSVC_pipe.predict(X_test)
np.mean(predictedLinearSVC == X_test_targetSentiment)
```

Out[50]:
```
0.9454737956590789
```

In [51]:
```
from sklearn.tree import DecisionTreeClassifier
clf_decisionTree_pipe = Pipeline([("vect", CountVectorizer()),
                                  ("tfidf", TfidfTransformer()),
                                  ("clf_decisionTree", DecisionTreeClassifier())
                                 ])
clf_decisionTree_pipe.fit(X_train, X_train_targetSentiment)

predictedDecisionTree = clf_decisionTree_pipe.predict(X_test)
np.mean(predictedDecisionTree == X_test_targetSentiment)
```

Out[51]:
```
0.9361214046232574
```

In [52]:
```
from sklearn.ensemble import RandomForestClassifier
clf_randomForest_pipe = Pipeline([("vect", CountVectorizer()),
                                  ("tfidf", TfidfTransformer()),
                                  ("clf_randomForest", RandomForestClassifier())
                                 ])
clf_randomForest_pipe.fit(X_train, X_train_targetSentiment)

predictedRandomForest = clf_randomForest_pipe.predict(X_test)
np.mean(predictedRandomForest == X_test_targetSentiment)
```

Out[52]:
```
0.949532380448209
```

In [54]:
```
predictedGS_clf_LinearSVC_pipe = gs_clf_LinearSVC_pipe.predict(X_test)
np.mean(predictedGS_clf_LinearSVC_pipe == X_test_targetSentiment)
```

Out[54]:
```
0.9638256573142756
```

In [55]:
```
for performance_analysis in (gs_clf_LinearSVC_pipe.best_score_,
                             gs_clf_LinearSVC_pipe.best_estimator_,
                             gs_clf_LinearSVC_pipe.best_params_):
        print(performance_analysis)
```

```
0.957511581733951
Pipeline(steps=[('vect', CountVectorizer(ngram_range=(1, 2))),
                ('tfidf', TfidfTransformer()), ('clf_linearSVC', LinearSVC())])
{'tfidf__use_idf': True, 'vect__ngram_range': (1, 2)}
```

In [56]:
```
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score

print(classification_report(X_test_targetSentiment,
                            predictedGS_clf_LinearSVC_pipe))
print('Accuracy: {}'. format(accuracy_score(X_test_targetSentiment,
                             predictedGS_clf_LinearSVC_pipe)))
```

```
              precision    recall  f1-score   support

    Negative       0.86      0.78      0.82       316
     Neutral       0.88      0.51      0.64       241
    Positive       0.97      1.00      0.98      5110

    accuracy                           0.96      5667
   macro avg       0.90      0.76      0.81      5667
weighted avg       0.96      0.96      0.96      5667

Accuracy: 0.9638256573142756
```

In [58]:
```
from sklearn.model_selection import train_test_split # function for splitti
import nltk
from nltk.corpus import stopwords
from nltk.classify import SklearnClassifier
from wordcloud import WordCloud,STOPWORDS
```

In [60]:
```
import matplotlib.pyplot as plt
```

In [97]:
```
 from subprocess import check_output
```

In [147…
```
pd.set_option('display.max_columns', 999)
train.head()
```

Out[147…

	name
27446	Fire HD 8 Tablet with Alexa, 8 HD Display, 16 ...
26722	Fire HD 8 Tablet with Alexa, 8 HD Display, 16 ...
23085	Fire Kids Edition Tablet, 7 Display, Wi-Fi, 16...
3112	AmazonBasics AAA Performance Alkaline Batterie...
25164	Kindle E-reader - White, 6 Glare-Free Touchscr...

In [162…
```
import pandas as pd
data1 = pd.read_csv('amazonreviews.csv')

print(df.columns)
df.columns
data1 = data1[['reviewstext', 'reviewsrating']]
```

```
Index(['id', 'dateAdded', 'dateUpdated', 'name', 'asins', 'brand',
       'categories', 'primaryCategories', 'imageURLs', 'keys', 'manufacturer',
       'manufacturerNumber', 'reviewsdate', 'reviewsdateSeen',
       'reviewsdidPurchase', 'reviewsdoRecommend', 'reviewsid',
       'reviewsnumHelpful', 'revrating', 'reviewssourceURLs', 'textrev',
       'reviewstitle', 'reviewsusername', 'sourceURLs'],
      dtype='object')
```

In [163…
```
train, test = train_test_split(data1,test_size = 0.1)
train = train[train.reviewsrating!=3]
```

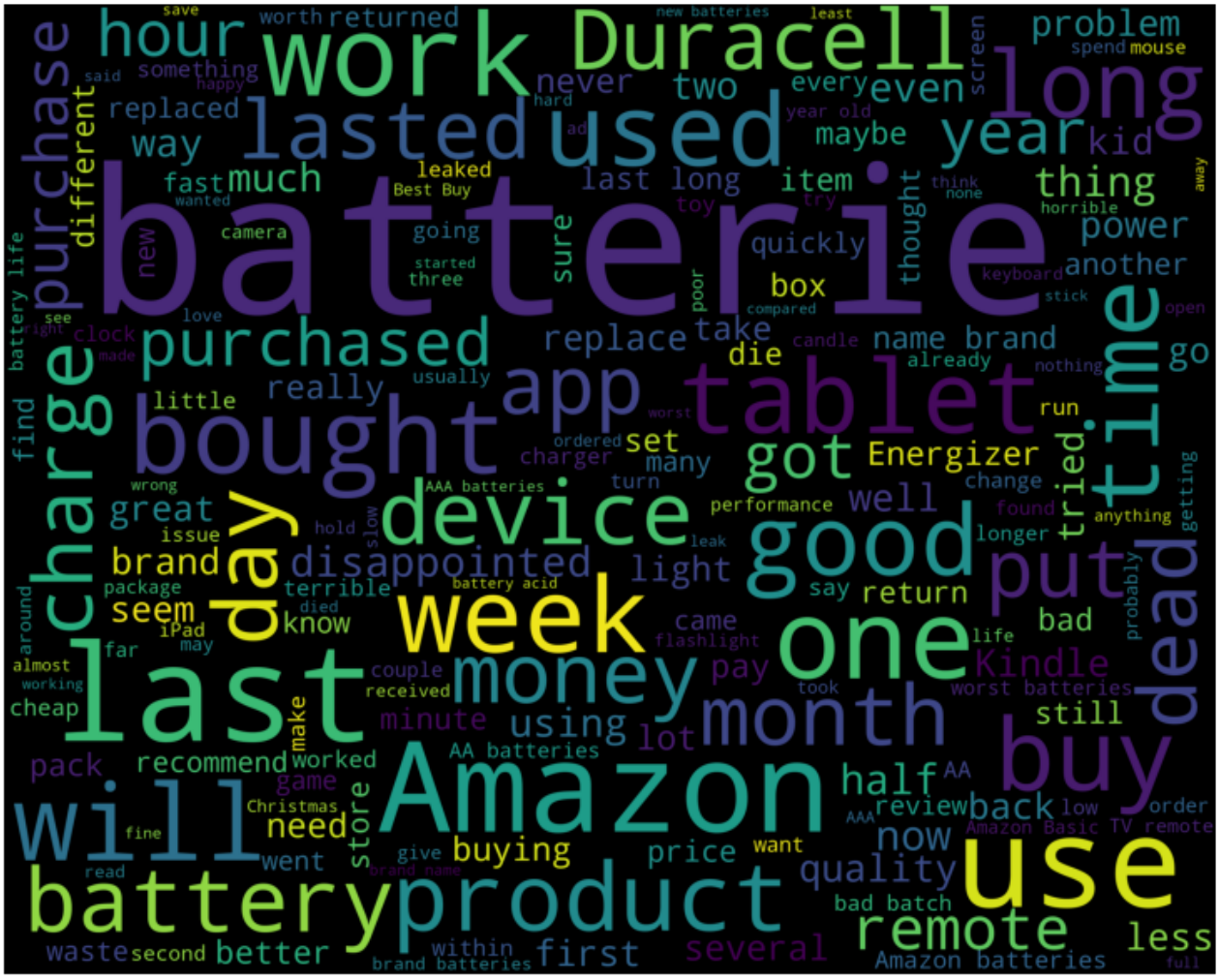
In [165…
```
train_pos = train[ train['reviewsrating'] >= 4]
train_pos = train_pos['reviewstext']
train_neg = train[ train['reviewsrating'] <=2]
train_neg = train_neg['reviewstext']
```

In [166…
```
from wordcloud import WordCloud,STOPWORDS


def wordcloud_draw(data1, color = 'black'):
    words = ' '.join(data1)
    cleaned_word = " ".join([word for word in words.split()
                             if 'http' not in word
                                 and not word.startswith('@')
                                 and not word.startswith('#')
                                 and word != 'RT'
                             ])
    wordcloud = WordCloud(stopwords=STOPWORDS,
                      background_color=color,
                      width=2500,
                      height=2000
                     ).generate(cleaned_word)
    plt.figure(1,figsize=(13, 13))
    plt.imshow(wordcloud)
    plt.axis('off')
    plt.show()

print("Positive words")
wordcloud_draw(train_pos,'white')
```

```
print("Negative words")
wordcloud_draw(train_neg)
```

Positive words

Negative words



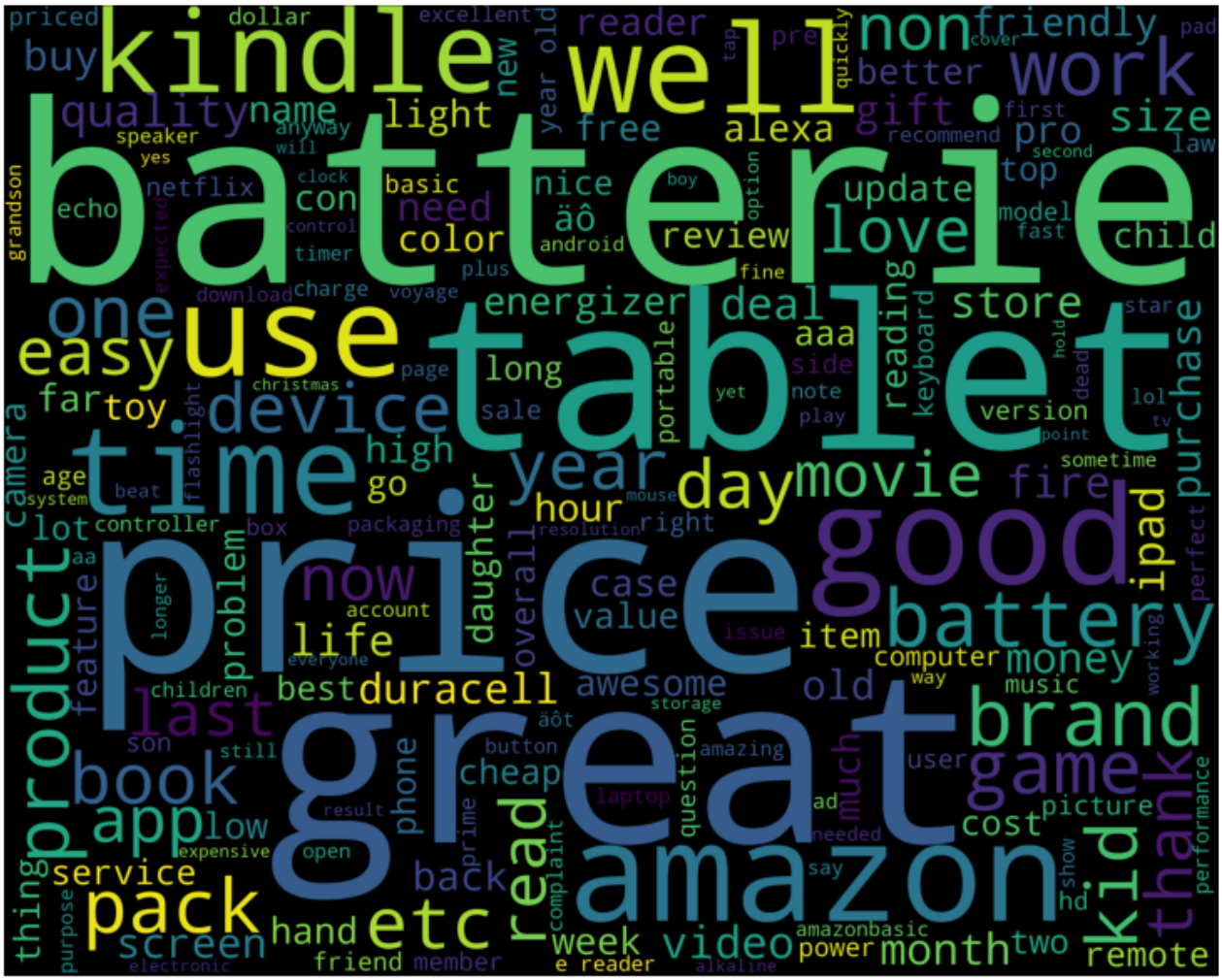
In [172…
```
amazonreviews = []
stopwords_set = set(stopwords.words("english"))

for index, row in train.iterrows():
    words_filtered = [e.lower() for e in row.reviewstext.split() if len(e) >= 3]
    words_cleaned = [word for word in words_filtered
        if 'http' not in word
        and not word.startswith('@')
        and not word.startswith('#')
        and word != 'RT']
    words_without_stopwords = [word for word in words_cleaned if not word in sto
    amazonreviews.append((words_without_stopwords, row.reviewstext))

test_pos = test[ test['reviewsrating'] >= 4]
test_pos = test_pos['reviewstext']
test_neg = test[ test['reviewsrating'] <= 2]
test_neg = test_neg['reviewstext']
```

In [173…
```
# Extracting word features
def get_words_in_amazonreviews(amazonreviews):
    all = []
    for (words, reviewsrating) in amazonreviews:
        all.extend(words)
    return all

def get_word_features(wordlist):
    wordlist = nltk.FreqDist(wordlist)
```

```
    features = wordlist.keys()
    return features
w_features = get_word_features(get_words_in_amazonreviews(amazonreviews))

def extract_features(document):
    document_words = set(document)
    features = {}
    for word in w_features:
        features['contains(%s)' % word] = (word in document_words)
    return features
```

In [174…

```
wordcloud_draw(w_features)
```



In [ ]:

```
training_set = nltk.classify.apply_features(extract_features,amazonreviews)
classifier = nltk.NaiveBayesClassifier.train(training_set)
```

In [ ]:

```
neg_cnt = 0
pos_cnt = 0
for obj in test_neg:
    res =  classifier.classify(extract_features(obj.split()))
    if(res == 'Negative'):
        neg_cnt = neg_cnt + 1
for obj in test_pos:
    res =  classifier.classify(extract_features(obj.split()))
    if(res == 'Positive'):
        pos_cnt = pos_cnt + 1
```

```
print('[Negative]: %s/%s '  % (len(test_neg),neg_cnt))
print('[Positive]: %s/%s '  % (len(test_pos),pos_cnt))
```

In [ ]: