Machine Learning with NLP ON Review Rating Data

I. Definition

Project Overview

Is it possible to predict the star rating of an Amazon review given the text of that review? I want to try to answer that question using a collection of 31000 Amazon reviews of Electronic Devices as my dataset. I think it is interesting to see if the sentiment of a review can be decided from the text of the review itself.

The problem of determining the sentiment of text is not new. Research was done to try to extract sentiment from movie reviews 1. They tried to predict whether the review was positive or negative based on the text. They found that one of the main difficulties with sentiment extraction is that it requires more understanding of the text than traditional topic-based categorization. Topic-based categorization relies on finding keywords in text, however sentiment can be expressed more subtly.

Problem Statement

The online store Amazon.com and Flipkart gives its users the ability review products in the form of a written review and a star rating between 1-5 stars. In this report, we try to predict the star rating given the written review.

To build our model we must complete the following tasks:

1. Download and pre-process Amazon,Flipkart reviews of Electronic Devices.

2. Extract features from the reviews.

3. Train models to predict ratings.

4. Refine the models.

5. Compare/contrast final models.

6. Discuss further possible improvements.

Metrics

The metric we will use to measure the performance of our models is the F1-score. An F1

score is defined as follows:

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives}$$

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

An F1-score is a popular metric to measure the performance of classifiers. An F1-score can be interpreted as a weighted average of precision and recall. 2 The range of an F1-score is between 0 and 1, with the best score being 1.

Our dataset is asymmetrical. 54% of the Amazon ratings in our dataset are five stars; this means if a model always guesses five stars it would be right 54% of the time. We don't want to favor models that guess five stars too often. This is why we are using the f1-score as our metric; it takes both false positives and false negatives into account. Using F1 scores to evaluate our models will allow us to favor models that have both high precision and recall.

II. Analysis

Data Exploration

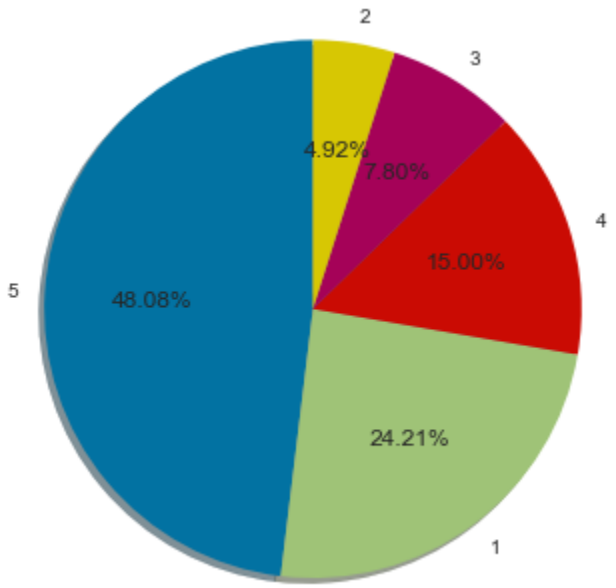
extracted 31000 reviews of Electronic Devices from Amazon and Flipkart. The data was extracted on August 2022.

There are a few abnormalities in the data. First, there are empty reviews. It's impossible to leave a review without a rating, but it is possible to leave a rating without a review. When we clean the data we should remove these empty reviews.

Next, it's also important to note that most reviews (over 50%) are 5-star reviews. We need to take that into account when splitting our data into training and testing sets. It's important that we stratify the data, so we don't get uneven datasets.

Table 1 is a random sample from the dataset. We will look at these example reviews later to see how each model does against this sample.

Table 1: Example Reviews



Reviews

Excellent product! It’s working in Chile with Argentinian and Italian SIM cards. Very good battery life (almost 7 days with regular use).

Purchased this phone for my son after breaking his iphone. We’re pleased with the RCA and he hasn’t had any issues with the phone, usage or connectivity and we’re on the AT&T network.

My daughter loves her new phone.

Stopped working after 6 days. Amazon refunded money with no issues.

No lo recomiendo. No funciona en Venezuela. No viene con idioma español solo ingles y francés.Not recommend. Does not work in Venezuela.

Rating Reviews

3 I love the keyboard on the phone/and wifi, but the volume is not that great and earpiece did not work with phone.

Exploratory Visualization

Figure 1 shows the distribution or ratings. Here we can see that there are many more 5-star ratings than other ratings. It's important that we take that into account when creating our training and testing sets or we may end up with too many 5-star reviews in one set.

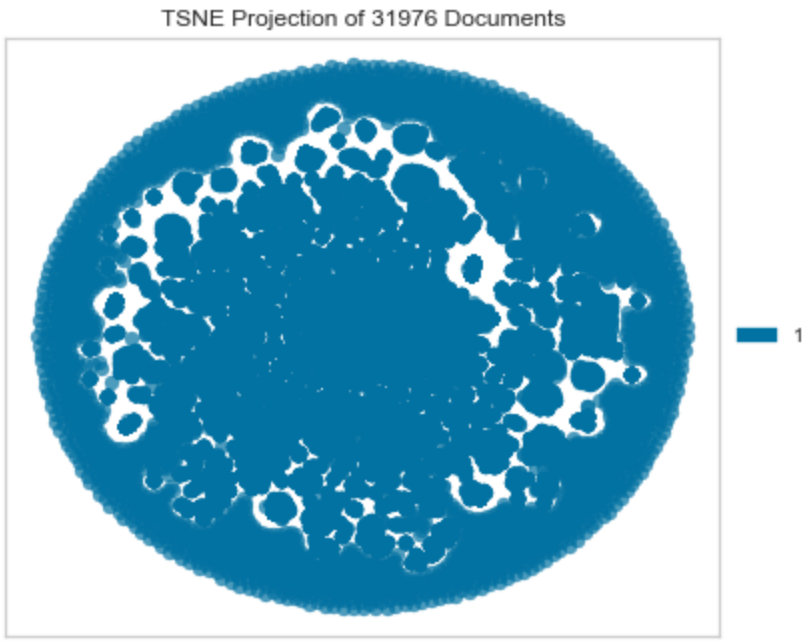


Figure 1: Rating Distribution

Figure 2 shows the distribution of review lengths. Notice that most of the review are relativity short, with very few above 200 characters. This is why it's important to take into account document length during feature extraction. We don't want long review to seem more relevant just because they are long.

Algorithms and Techniques

We need an algorithm for feature extraction and some algorithms for classification. First, we will look at an algorithm for feature extraction, TF-IDF, and then we will look at three

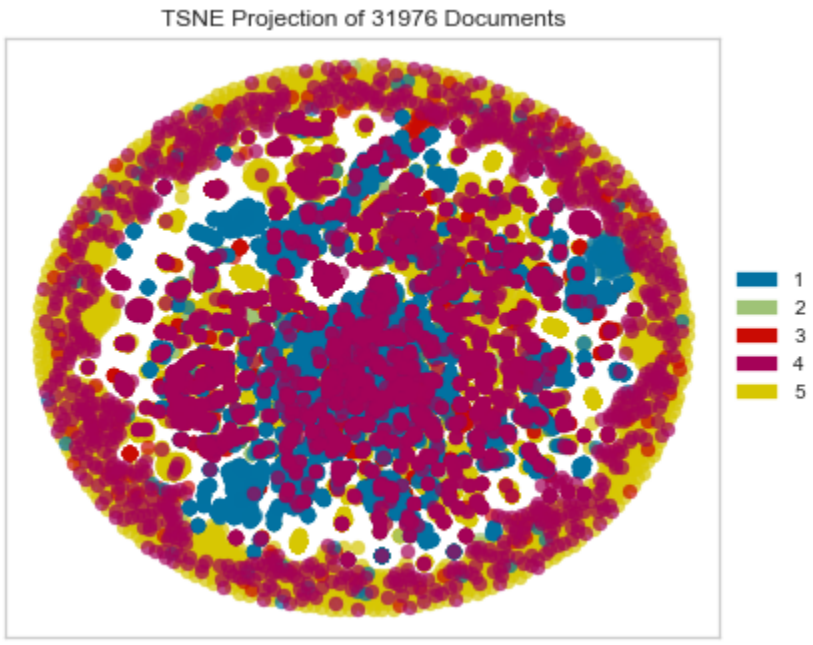


Figure 2: Review Length Distribution

algorithms we will use as candidates for our classification algorithm: Logistic Regression, Gaussian Naive Bayes, and Stochastic Gradient Descent.

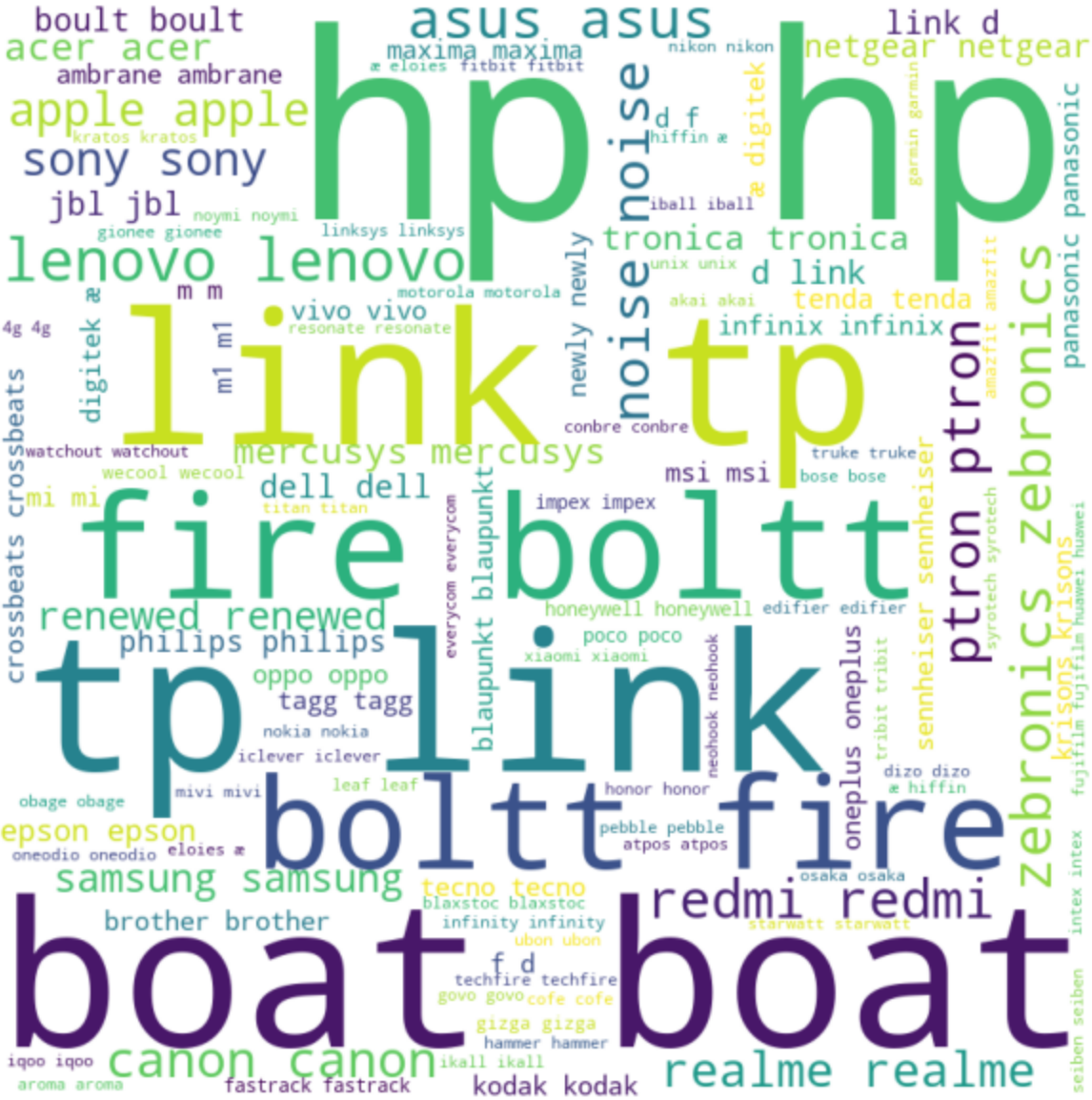
I chose these algorithms for performance reasons (these algorithms must run on a MacBook Pro with no GPU), these algorithms scale well to large datasets and large feature sets.

TF-IDF

We are going to use **term frequency–inverse document frequency** (TF-IDF)4 as our feature extraction strategy. TF-IDF takes into account document length and term frequency when building features. We don't want long reviews to seem more relevant just because they are long, we also don't want to overvalue words that appears many times but do not provide very much information. TF-IDF weights the words based on it's frequency in the document and the document length.

TF/IDF = Term Frequency × Inverse Document Frequency

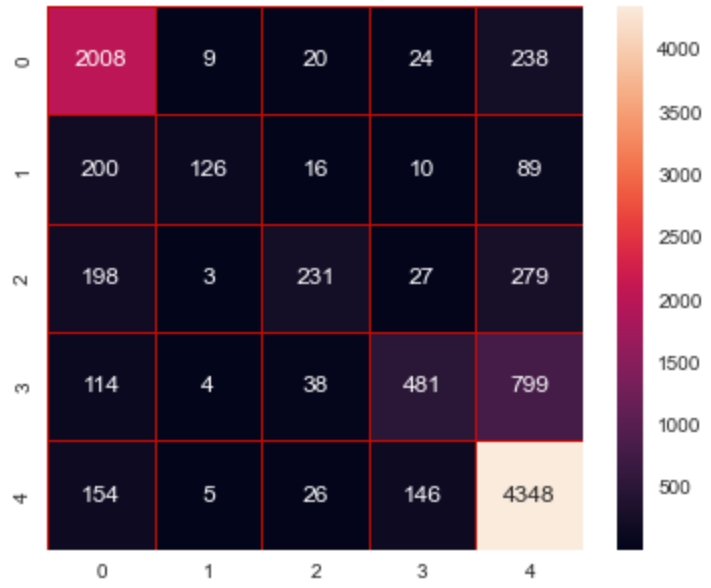
TF-IDF uses a bag of words with weights to represent a document. However, this will miss some relationships between words. For example "not good" and "very good" both contain the word "good" but one is has the inverse meaning. We can help fix this problem using the ngram_range option in sklearns implementation of TD-IDF. ngram_range allows us to set the number of words that can be considered together. Setting this correctly will take some guessing and checking.


boat boat
hp hp
tp link
link tp
fire boltt
boltt fire
zebronics zebronics
ptron ptron
asus asus
lenovo lenovo
samsung samsung
realme realme
redmi redmi
canon canon
boult boult
acer acer
ambrane ambrane
apple apple
sony sony
jbl jbl
maxima maxima
netgear netgear
link d
noise noise
tronica tronica
d link
tenda tenda
infinix infinix
vivo vivo
mercusys mercusys
dell dell
msi msi
mi mi
renewed renewed
philips philips
oppo oppo
tagg tagg
nokia nokia
iclever iclever
mivi mivi
leaf leaf
honor honor
oneplus oneplus
epson epson
pebble pebble
atpos atpos
tecno tecno
brother brother
blaxstoc blaxstoc
infinity infinity
ubon ubon
techfire techfire
govo govo
cofe cofe
gizga gizga
hammer hammer
ikall ikall
kodak kodak
fastrack fastrack
aroma aroma
iqoo iqoo
oneodio oneodio
eloies æ
obage obage
watchout watchout
wecool wecool
titan titan
impex impex
honeywell honeywell
edifier edifier
poco poco
xiaomi xiaomi
osaka osaka
starwatt starwatt
dizo dizo
æ hiffin
sennheiser sennheiser
panasonic panasonic
amazfit amazfit
syrotech syrotech
krisons krisons
fujifilm fujifilm
huawei huawei
intex intex
seiben seiben
truke truke
bose bose
conbre conbre
motorola motorola
akai akai
unix unix
linksys linksys
gionee gionee
noymi noymi
iball iball
hiffin æ
garmin garmin
kratos kratos
æ eloies fitbit fitbit
nikon nikon
digitek
crossbeats
everycom everycom
blaupunkt
neohook neohook
tribit tribit
resonate resonate
m1 m1
m m
4g 4g
d f
f d

Logistic Regression

We are going to use sklearns implementation of logistic regression 5 as our benchmark.

```
******************RESULTS*******************
The accuracy score of train is : 78.98851807175087
The accuracy score test is : 74.9921817992286
The cross validation score is : 74.16786482713853
Confusion Matrix:
 [[2008    9   20   24  238]
 [ 200  126   16   10   89]
 [ 198    3  231   27  279]
 [ 114    4   38  481  799]
 [ 154    5   26  146 4348]]
Classification              precision    recall  f1-score   support

           1       0.75      0.87      0.81      2299
           2       0.86      0.29      0.43       441
           3       0.70      0.31      0.43       738
           4       0.70      0.33      0.45      1436
           5       0.76      0.93      0.83      4679

    accuracy                           0.75      9593
   macro avg       0.75      0.55      0.59      9593
weighted avg       0.75      0.75      0.72      9593

************************************************
```



Logistic regression is a classification algorithm (not a regression algorithm). Logistic regression takes a vector of features and transforms it into a probability by feeding it into a sigmoid function. Given a set of training points logistic regression will try to find a vector of weights that when multiplied to the set of features and fed into the sigmoid function we get an accurate prediction.

Even though logistic regression can only handle binary classification problems, sklearns implementation gives us the option to use a *One-vs-Rest* technique; it trains against each class one at a time with all other samples counted as negative examples. This makes logistic regression a linear model capable of multiclass classification.

Random Forest Classifier

**********RESULTS**********

The accuracy score of train is : 91.98498860742528

The accuracy score test is : 83.09183779839466

The cross validation score is : 81.48595153104048

Confusion Matrix:

[[2038 20 25 30 186]

[79 286 13 10 53]

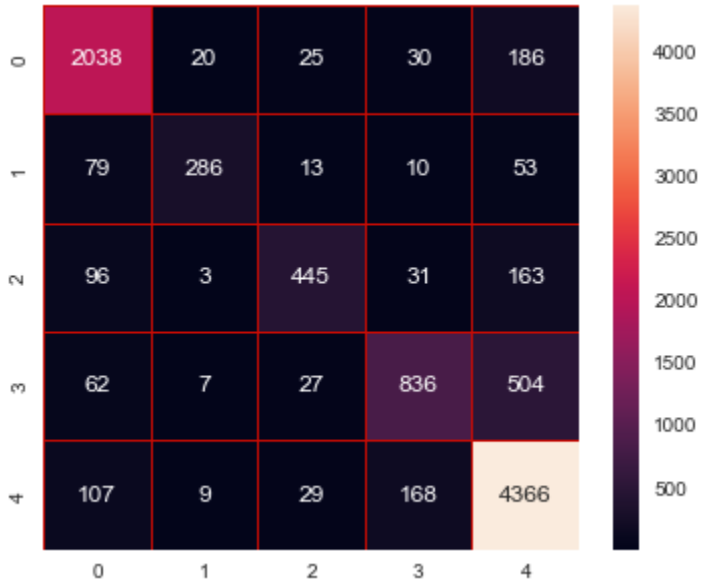
[96 3 445 31 163]

[62 7 27 836 504]

[107 9 29 168 4366]]

Classification	precision	recall	f1-score	support
1	0.86	0.89	0.87	2299
2	0.88	0.65	0.75	441
3	0.83	0.60	0.70	738
4	0.78	0.58	0.67	1436
5	0.83	0.93	0.88	4679
accuracy			0.83	9593
macro avg	0.83	0.73	0.77	9593
weighted avg	0.83	0.83	0.82	9593

****



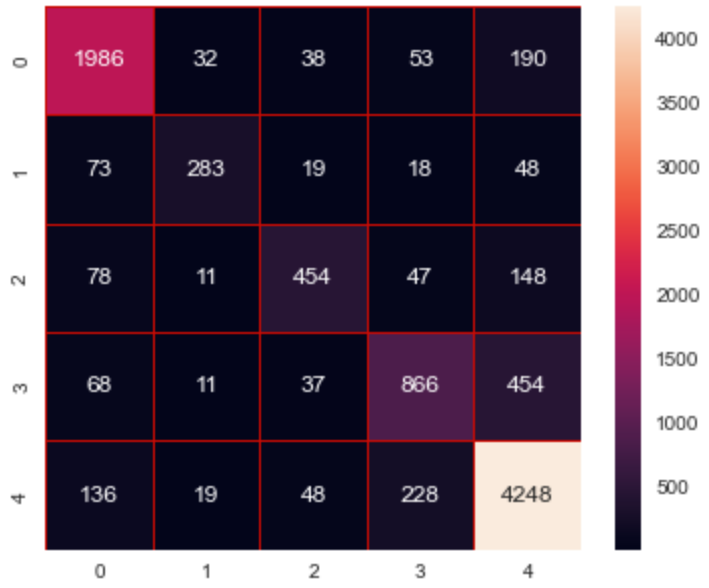
Benchmark

We are going to use sklearns implementation of logistic regression 6 as our benchmark. For the benchmark, we are not going to do any parameter tuning; we are going to use sklearns default parameters. As we will see, logistic regression does fairly well right out of the box.

```
dec_clf = DecisionTreeClassifier()
dec_clf.fit(X_train, y_train)
y_pred_train = dec_clf.predict(X_train)
y_pred = dec_clf.predict(X_test)
```

```
print("******************RESULTS******************")
print("The accuracy score of train is :", accuracy_score(y_train,
y_pred_train)*100)
print("The accuracy score test is :", accuracy_score(y_test, y_pred)*100)
cv_score = cross_val_score(dec_clf,X_train, y_train,cv=5)
print("The cross validation score is :", cv_score.mean()*100)

print("Confusion Matrix: \n", confusion_matrix(y_test, y_pred))
print("Classification ", classification_report(y_test, y_pred))
print("**********************************************")

plt.figure(figsize=(6,5))
sns.heatmap(confusion_matrix(y_test, y_pred),annot=True,fmt =
"d",linecolor="r",linewidths=1)
plt.show()
```

```
******************RESULTS******************
The accuracy score of train is : 91.98498860742528
The accuracy score test is : 81.69498592723862
The cross validation score is : 80.19032636873241
Confusion Matrix:
 [[1986   32   38   53  190]
 [  73  283   19   18   48]
 [  78   11  454   47  148]
 [  68   11   37  866  454]
 [ 136   19   48  228 4248]]
Classification              precision    recall  f1-score   support

           1       0.85      0.86      0.86      2299
           2       0.79      0.64      0.71       441
           3       0.76      0.62      0.68       738
           4       0.71      0.60      0.65      1436
           5       0.83      0.91      0.87      4679

    accuracy                           0.82      9593
   macro avg       0.79      0.73      0.75      9593
weighted avg       0.81      0.82      0.81      9593

**********************************************
```




```
benchmark_linear_model = LogisticRegression().fit(X_train, y_train)
f1_score(y_test, benchmark_linear_model.predict(X_test),
average='weighted')
```

III. Methodology

Data Preprocessing

There are several preprocessing steps we need to take before training. First, sometimes people leave ratings without reviews. We are going to ignore empty reviews:

```
data = data[data['Reviews'].isnull()==False]
```

Next, some of the reviews have some formatting that we want to get rid of. We also want to remove all characters not in the English alphabet (including punctuation) and set everything to lowercase:

At this point each review is a bunch of lowercase words. Now we need to do some kind of feature extraction. As stated in *Algorithms and Techniques*, we are going to use TF-IDF as our extraction algorithm

We also look at the word cloud in figure 3 and see some similarities between the two. 7BeautifulSoup is a Python library for pulling data out of HTML and XML files. We are using it here to

Implementation

We are going to use a *Multinomial Naive Bayes* classifier and a *Stochastic Gradient Descent* classifier to predict ratings. We are going to use sklearn's implementation of these algorithms. Both are relatively simple to use; the tricky part is parameter tuning which we will get to in *Refinement*.

If we use sklearn's default implementation of these algorithms we can achieve the following F1-Scores:

Algorithm: MultinomialNB SGDClassifier F1-Score: 0.6460 0.6100

Refinement

Parameter tuning was done using sklearn's GridSearchCV algorithm; GridSearchCV does an exhaustive search over specified parameter values.

Hyper parameter tuning

In [81]:

```
param_grid = [
```

```
   {"n_estimators": range(20, 100, 2),
    "max_depth": [1, 2, 4, 6, 9, 12],
    "min_samples_split": [2, 6, 8],
    "min_samples_leaf": [1, 2, 3],
    }
]


ran_clf  =  RandomForestClassifier(random_state = 89)
rf_grid = GridSearchCV(ran_clf,param_grid,cv = 2,)
rf_grid.fit(X_train,y_train)
params = rf_grid.best_params_
print("Best Params  ",params)
```

Best Params {'max_depth': 12, 'min_samples_leaf': 1, 'min_samples_split': 6, 'n_estimators': 22}

In [82]:

```
rand_clfr = RandomForestClassifier(n_estimators =
26,min_samples_split=2,min_samples_leaf=1,max_features='auto',max_depth=12,
                                   criterion='entropy',bootstrap = True)
rand_clfr.fit(X_train,y_train)
rf_pred = rand_clfr.predict(X_test)
print('Final Random Forest Classifier Model')
print('Accuracy Score :', accuracy_score(y_test, rf_pred))
print('\n')
print('Confusion matrix of Random Forest Classifier :',confusion_matrix(y_test,
rf_pred))
print('\n')
print('Classification Report of Random Forest Classifier:
',classification_report(y_test, rf_pred))
```

Final Random Forest Classifier Model
Accuracy Score : 0.5218388408214323

Confusion matrix of Random Forest Classifier : [[334 0 0 0 1965]
[12 0 0 0 429]
[14 0 0 0 724]
[1 0 0 0 1435]
[7 0 0 0 4672]]

Classification Report of Random Forest Classifier: precision recall f1-score support

1	0.91	0.15	0.25	2299
2	0.00	0.00	0.00	441
3	0.00	0.00	0.00	738

4	0.00	0.00	0.00	1436
5	0.51	1.00	0.67	4679
accuracy			0.52	9593
macro avg	0.28	0.23	0.18	9593
weighted avg	0.46	0.52	0.39	9593

IV. Results

Model Evaluation and Validation

We save 30% of the data for testing. Now we can compare the models by seeing their performance on this data. We can look at their F1-scores, and, to help us visualize their performance, their confusion matrices.

These confusion matrices tell us that the models have an easier time with the 5-star and 1-star reviews than the other reviews. Reviews with 5-star and 1-star are probably more straightforward because people who write these reviews use very distinct words.

It's clear from both the F1-score and confusion matrices that the stochastic gradient descent model outperforms the others.

Justification

The final stochastic gradient descent learning model (SGDClassifier) has a F1-Score of 0.8568, this is much better than the benchmark model which has a score of 0.7862. We can look at the example reviews from *Table 1* and see how each classifier did:

Review

Excellent product! It's working in Chile with Arge. . . Purchased this phone for my son after breaking his. . . My daughter loves her new phone.

Stopped working after 6 days. Amazon refunded mone. . .

We can see that the classifiers have a hard time differentiating between 4 and 5 star reviews and 2 and 1 star reviews. One of the reviews has Spanish words, almost all the reviews are English so it's not surprising that that all three classifier got that predictions wrong.

The final model does a good job of finding the general sentiment of a review. If you want to determine if a review is positive or negative, then this model is probably good enough. If you want a model that can differentiate between similar star ratings (like 3-star and 4-star reviews), then more work needs to be done.

V. Conclusion

Free-Form Visualization Word Cloud

When we look at the top phrases from TF-IDF we should see some similarities with the following generated word cloud:

Reflection

1. The data was downloaded and cleaned.

2. Features were extracted from the review text.

3. The data was split into testing and training sets.

4. A benchmark classifier was created.

5. Several classifiers were trained against the data using automated parameter tuning. 6. The models were tested against the testing data.

7. The models were compared using F1-scoring as a metric.

I found feature-extraction and parameter tuning as the most complicated steps in this project. The performance of my models was not very good until

Improvement

All three models have low recall on reviews with star ratings between 2 and 4. Even for a human, it's difficult to differentiate between these reviews. The difference between a 3-star and 4-star review is very subtle. A good place to look for improvements are ways to improve recall for these mid-star reviews.

A solution not explored in this report is a neural network. Many people have found success using neural networks for text classification problems. A good candidate would be the use of the python library. neural network library that would be useful in building a neural network model to predict ratings.