Carlos Facundo Gianello

Midterm Report

Preliminary Analysis:

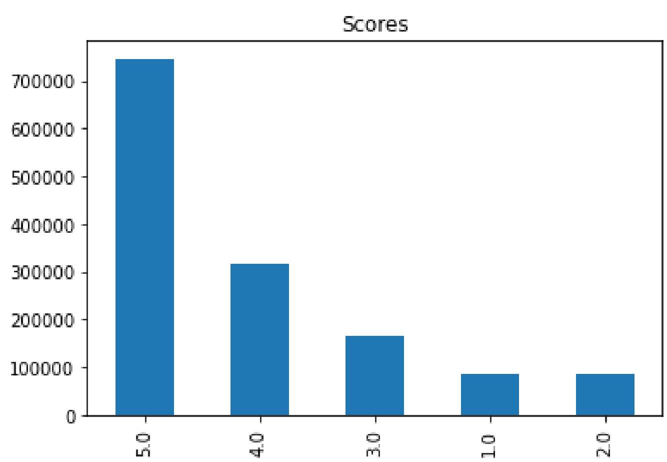
For data we get 3 files, train which contains 1.6 million reviews from Amazon Movies, in this file we can find the associated product id, user id, helpfulness numerator and denominator, score, timestamp, summary and text, and review id. However, some reviews contain missing scores which we must predict.

There is also the test file which contains the 300,000 reviews with ids in train.csv but missing scores. Once we fit the prediction data into our model, we can then fit the test file into the model to fill out this file.

Lastly there is the sample.csv which is the submission file with the review predictions.

Exploration / Feature Extraction:

In the specific review data, we can find some notable features that may impact our prediction. Firstly, reviewers tend to give a score of 5 over other scores which makes that score very popular and can have a negative impact in our prediction.



Thus, we use the following

```
# there are many five star revies, which can bias the classifier, so getting rid of some
fives = df.loc[df['Score'] == 5]
fives = fives.sample(frac=0.5)
df = pd.concat([df.loc[df['Score'] != 5], fives])
```

to decrease the amount of “5” scores. Furthermore, we also encounter some reviews with blank text and summaries which will be turned into blanks for efficient data usage.

```
In [19]: blank_df = pd.DataFrame()
         blank_df = df['Text'].loc[df['Text'].isna()]
         blank_df.head(10)

Out[19]: 5209      NaN
         13114     NaN
         23314     NaN
         32582     NaN
         82557     NaN
         106563    NaN
         134060    NaN
         149540    NaN
         150771    NaN
         158516    NaN
         Name: Text, dtype: object
```

so that we only retrieve relevant information to feed the model.

Another notable thing about the data is the use of faces such as “ :) “ etc. to provide emotion to the review, this can be replaced into the textual definition of the emote so that it can be properly fed to the model.

```
In [34]: df[df.Text.isin([':)'])]
Out[34]:
```

	Id	ProductId	UserId	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time	Summary	Text
9208	9208	0767809246	AFNEAHTMMMQM	0	0	5.0	1404777600	Five Stars	:)

```
In [33]: df[df.Summary.isin([':D'])]
Out[33]:
```

	Id	ProductId	UserId	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time	Summary	Text
17500	17500	0767824571	A3HOTA9SQXBUR1	0	0	5.0	1355270400	:D	I bought this for my husband because it is his...
62799	62799	0783230060	A1BT4S4WMB8IM4	0	0	4.0	1360627200	:D	I recived this item sooner than I thought and ...
85134	85134	0788882988	A2I5GE4ZMWNSBI	1	4	NaN	1212364800	:D	i thought this movie was awesome and the guy w...
624193	624193	B00005JMYF	A1SPSR86QD3SVW	0	1	5.0	1369267200	:D	Hilarious every time! This is a film to quote ...
847765	847765	B0002W4SX6	A3QM79VZN1T98V	0	0	5.0	1358985600	:D	The golden girls was always such a great show ...
911684	911684	B0009S4IEW	A2N2MCN2UNNIHO	0	0	5.0	1402185600	:D	This finished out my complete set. I’ve...
1088966	1088966	B000TAN8ZU	A6VTVGBA6HV5M	0	0	NaN	1378944000	:D	My childhood, right there. I wish they had Joh...
1103271	1103271	B000VNMMQ6	A2I5GE4ZMWNSBI	1	3	5.0	1193443200	:D	this movie was tooo funny!!! and kevin smith w...
1136140	1136140	B00128VA6C	A1EIDQXW1DGA2W	0	0	4.0	1386892800	:D	I love this movie and just wanted it in my mov...
1427971	1427971	B004U2RBGU	A3F546JCJPB6QP	0	0	5.0	1376265600	:D	Hey, it's Trigun! What can I say? Blu-Ray is a...
1527881	1527881	B0077PHME8	A20VAHDMLHVWG1	0	0	4.0	1387843200	:D	I read some of the other negative reviews, but...
1602312	1602312	B009TT0BSE	A8OAC5ANJ1BL	1	3	5.0	1376697600	:D	This was an awesome movie. No skips, no dings,...
1606336	1606336	B00A7GWGLS	A2E43DLAQZPGED	0	2	5.0	1364774400	:D	YAY!!! $39.98 is a great price. I can't wait t...

Then we have things like punctuation and stop words which can just be easily removed to improve tokenization.

Challenges / Tuning:

After tokenizing the data, we are left with individual words that could not be fed into the model, therefore the solution was to use the TF-IDF Vectorizer to encode the words into occurrences and frequencies to be provided to the classifiers for prediction. However, doing this resulted in many occurrences which repeated many times and took a big strain on memory. The solution to this problem was to tune the encoder so that it only checked for relevant occurrences that also did not repeat too much

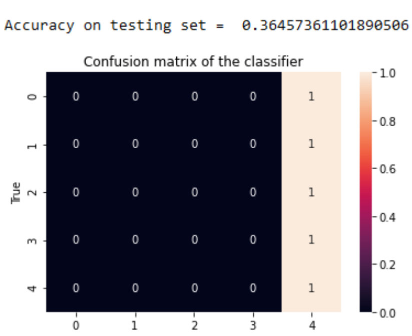
```
count_w = TfidfVectorizer(max_df=0.9, min_df=0.05)
```

After tuning the encoder, we then make the predictions on the training data and finally fit it to the model using the given K Neighbors classifier. The accuracy result was low ~30% and so different classifiers were used to improve certainty. However, the ultimate challenge proved to be time as it was not possible to get all classifiers to predict the data for an accuracy comparison.

Testing / Model:

The original methods in mind for testing predictions were K Neighbors as it was the original classifier and Logistic Regression versus Naïve Bayes, this is because the former is commonly used for things such as detecting whether emails are spam to great success which seemed to fit the necessary predictions for this project. The latter was also expected to perform greatly since it is not sensitive to irrelevant features which could have been missed during the exploration section, it handles text data fairly well and it does not require as much training data. Other classifiers were also used and considered.

Logistic Regression



Decision Tree