# Yelp_Reviews

September 10, 2020

## 1 Kaushal Rao - NLP with Yelp Reviews

#### 1.0.1 In this NLP project we will be attempting to classify Yelp Reviews into the 1 star, 2 star, 3 star, 4 star, or 5 star categories based off the review's text content, using the Multinomial Naive Bayes statistical model.

We will use the Yelp Review Data Set from Kaggle.

Each observation/row in this dataset is a review of a particular business by a particular user. The "stars" column is the number of stars (1 through 5) assigned by the reviewer to the business. In other words, it is the rating of the business by the person who wrote the review. The "cool" column is the number of "cool" votes this review received from other Yelp users. All reviews start with 0 "cool" votes, and there is no limit to how many "cool" votes a review can receive. In other words, it is a rating of the review itself, not a rating of the business. The "useful" and "funny" columns are similar to the "cool" column.

```
In [1]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import seaborn as sns
        sns.set_style('white')
        %matplotlib inline
        from sklearn.feature_extraction.text import CountVectorizer
        from sklearn.model_selection import train_test_split
        from sklearn.naive_bayes import MultinomialNB
        from sklearn.metrics import confusion_matrix,classification_report
        from sklearn.feature_extraction.text import  TfidfTransformer
        from sklearn.pipeline import Pipeline
        # importing relevant modules
```

### 1.1 Importing the Data

```
In [2]: yelp = pd.read_csv('yelp.csv')
        yelp.head()
```

```
Out[2]:              business_id        date               review_id  stars  \
        0  9yKzy9PApeiPPOUJEtnvkg  2011-01-26  fWKvX83p0-ka4JS3dc6E5A      5
        1  ZRJwVLyzEJq1VAihDhYiow  2011-07-27  IjZ33sJrzXqU-0X6U8NwyA      5
        2  6oRAC4uyJCsJl1X0WZpVSA  2012-06-14  IESLBzqUCLdSzSqm0eCSxQ      4
```

```
3  _1QQZuf4zZOyFCvXc0o6Vg  2010-05-27  G-WvGaISbqqaMHlNnByodA      5
4  6ozycU1RpktNG2-1BroVtw  2012-01-05  1uJFq2r5QfJG_6ExMRCaGw      5

                                                text    type  \
0  My wife took me here on my birthday for breakf...  review
1  I have no idea why some people give bad review...  review
2  love the gyro plate. Rice is so good and I als...  review
3  Rosie, Dakota, and I LOVE Chaparral Dog Park!!...  review
4  General Manager Scott Petello is a good egg!!!...  review

                  user_id  cool  useful  funny
0  rLtl8ZkDX5vH5nAx9C3q5Q     2       5      0
1  0a2KyELOd3Yb1V6aivbIuQ     0       0      0
2  0hT2KtfLiobPvh6cDC8JQg     0       1      0
3  uZetl9T0NcROGOyFfughhg     1       2      0
4  vYmM4KTsC8ZfQBg-j5MWkw     0       0      0
```

```
In [3]: yelp.info()
```

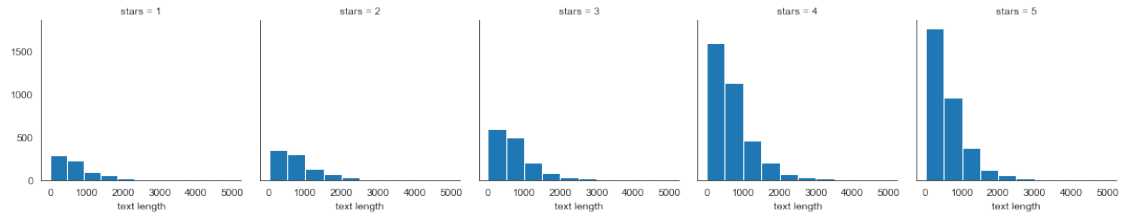
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 10 columns):
business_id    10000 non-null object
date           10000 non-null object
review_id      10000 non-null object
stars          10000 non-null int64
text           10000 non-null object
type           10000 non-null object
user_id        10000 non-null object
cool           10000 non-null int64
useful         10000 non-null int64
funny          10000 non-null int64
dtypes: int64(4), object(6)
memory usage: 781.4+ KB
```

```
In [4]: yelp['text length'] = yelp['text'].apply(len)
        # creating a new column called 'text length' which is the # of words in the 'text' col
```

### 1.2 Exploratory Data Analysis (EDA)

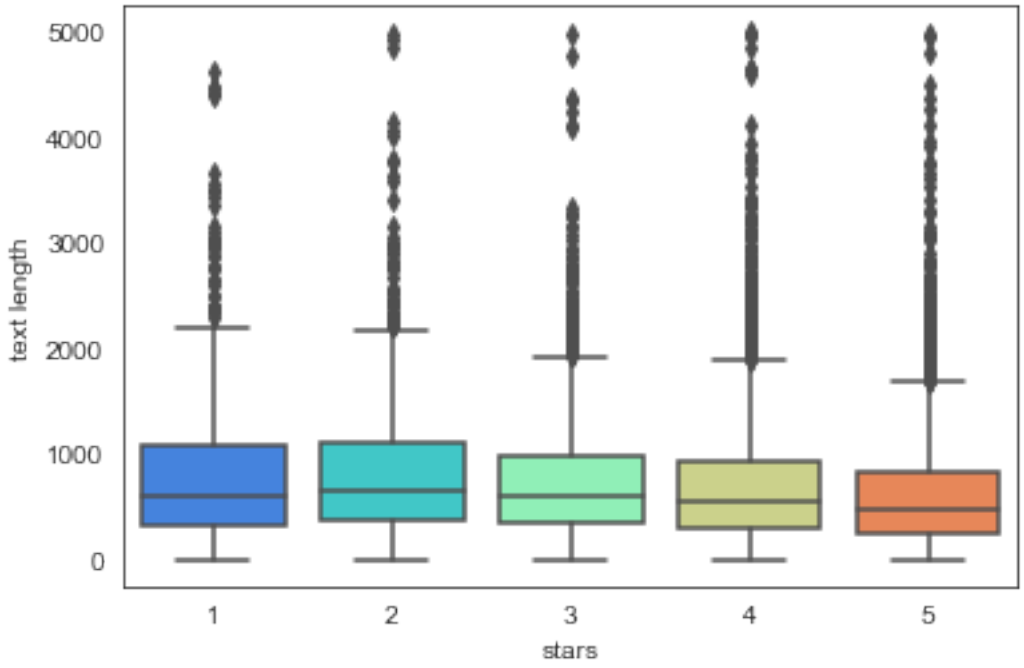
```
In [5]: g = sns.FacetGrid(yelp,col='stars')
        g.map(plt.hist,'text length')
        # grid of 5 histograms of text length based off of the star ratings
        # we can see that reviews that are 4 or 5 stars are more likely to be shorter (proport
```

```
Out[5]: <seaborn.axisgrid.FacetGrid at 0x1a24dbd240>
```



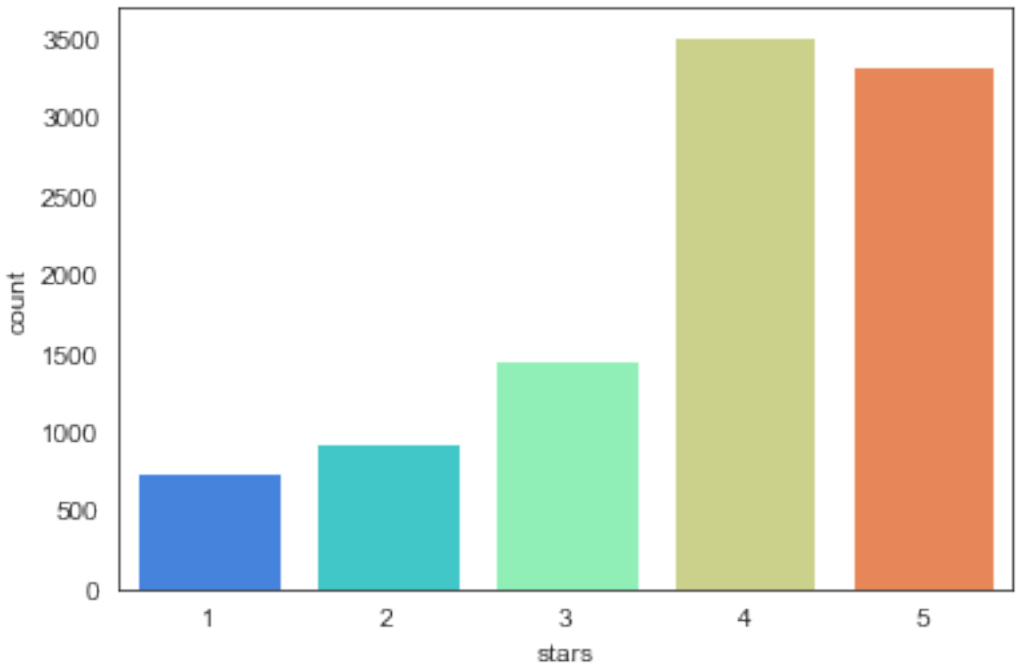

In [6]:
```
sns.boxplot(x='stars',y='text length',data=yelp,palette='rainbow')
# boxplot of text length for each star category
# median tends to be around the same for each
```

Out[6]: <matplotlib.axes._subplots.AxesSubplot at 0x1a25293ef0>




In [7]:
```
sns.countplot(x='stars',data=yelp,palette='rainbow')
# countplot of the number of occurences for each type of star rating
```

Out[7]: <matplotlib.axes._subplots.AxesSubplot at 0x1a25447470>



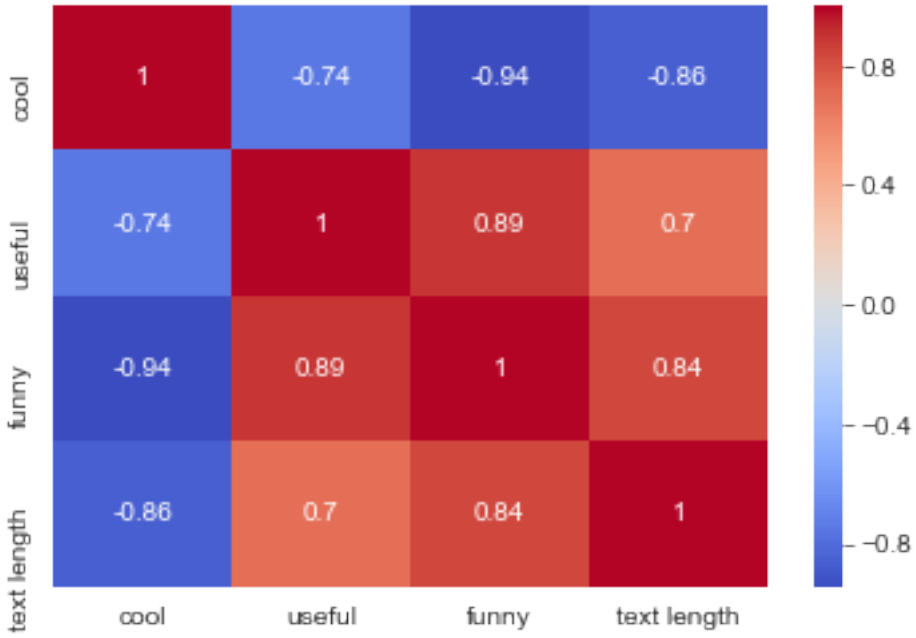

```
In [8]: stars = yelp.groupby('stars').mean()
        stars
        # 1 star reviews score higher in 'useful' and 'funny'
        # 4/5 star reviews score higher in 'cool' and are shorter in text
```

Out[8]:

| stars | cool | useful | funny | text length |
|---|---|---|---|---|
| 1 | 0.576769 | 1.604806 | 1.056075 | 826.515354 |
| 2 | 0.719525 | 1.563107 | 0.875944 | 842.256742 |
| 3 | 0.788501 | 1.306639 | 0.694730 | 758.498289 |
| 4 | 0.954623 | 1.395916 | 0.670448 | 712.923142 |
| 5 | 0.944261 | 1.381780 | 0.608631 | 624.999101 |

```
In [9]: sns.heatmap(stars.corr(),cmap='coolwarm',annot=True)
        # correlation heatmap of the different columns
        # 'useful' has positive correlation with 'funny' and 'text length'
        # 'useful' has negative correlation with 'cool'
```

```
Out[9]: <matplotlib.axes._subplots.AxesSubplot at 0x1a2559c438>
```




### 1.3 NLP Classification

```
In [10]: # for now, we will just grab reviews that are either 1 or 5 stars
         yelp_class = yelp[(yelp.stars==1) | (yelp.stars==5)]
```

```
In [11]: X = yelp_class['text'] #features
         y = yelp_class['stars'] #labels
         cv = CountVectorizer()
         # create a CountVectorizer object
         # to convert the collection of text documents to a matrix of token counts
```

```
In [12]: X = cv.fit_transform(X)
         # creating sparse matrix
```

#### 1.3.1 Train-Test Split

```
In [13]: X_train, X_test, y_train, y_test = train_test_split(X, y,test_size=0.3,random_state=1
```

#### 1.3.2 Training the Model

```
In [14]: nb = MultinomialNB()
         nb.fit(X_train,y_train)
         # fitting the multinomial Naive Bayes classifier with training data
```

```
Out[14]: MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)
```

### 1.3.3 Predictions and Evaluation

```
In [15]: predictions = nb.predict(X_test)
         print(confusion_matrix(y_test,predictions))
         print('\n')
         print(classification_report(y_test,predictions))
         # model performed pretty well at distinguishing 1 star from 5 star reviews
```

```
[[159  69]
 [ 22 976]]


              precision    recall  f1-score   support

           1       0.88      0.70      0.78       228
           5       0.93      0.98      0.96       998

   micro avg       0.93      0.93      0.93      1226
   macro avg       0.91      0.84      0.87      1226
weighted avg       0.92      0.93      0.92      1226
```

## 1.4 Using Text Processing

```
In [16]: pipeline = Pipeline([
             ('bow', CountVectorizer()), #strings to token integer counts
         #    ('tfidf', TfidfTransformer()), #integer counts to weighted TF-IDF scores
             ('classifier', MultinomialNB()), #train on TF-IDF vectors w/ MN Naive Bayes
         ])
         # pipeline for storing preprocessing steps
         # TF-IDF scores are a weight based on term frequency and inverse document frequency
         # in this case, TF-IDF scores actually made the model worse so its commented out
```

### 1.4.1 Train-Test Split

```
In [17]: X = yelp_class['text'] #features
         y = yelp_class['stars'] #labels
         X_train, X_test, y_train, y_test = train_test_split(X, y,test_size=0.3,random_state=1
```

### 1.4.2 Predictions and Evaluation

```
In [18]: pipeline.fit(X_train,y_train)
         # fit pipeline on training data
```

```
Out[18]: Pipeline(memory=None,
             steps=[('bow', CountVectorizer(analyzer='word', binary=False, decode_error='stri
                dtype=<class 'numpy.int64'>, encoding='utf-8', input='content',
                lowercase=True, max_df=1.0, max_features=None, min_df=1,
```

```
                ngram_range=(1, 1), preprocessor=None, stop_words=None,
                strip_accents=None, token_pattern='(?u)\\b\\w\\w+\\b',
                tokenizer=None, vocabulary=None)), ('classifier', MultinomialNB(alpha=1.0, cla
```

In [19]:
```
predictions = pipeline.predict(X_test)
# predictions
```

In [20]:
```
print(confusion_matrix(y_test,predictions))
print(classification_report(y_test,predictions))
# a great model for 5-class classification!
```

```
[[155  73]
 [ 15 983]]
              precision    recall  f1-score   support

           1       0.91      0.68      0.78       228
           5       0.93      0.98      0.96       998

   micro avg       0.93      0.93      0.93      1226
   macro avg       0.92      0.83      0.87      1226
weighted avg       0.93      0.93      0.92      1226
```