Natural Language Processing Project

In this NLP project you will be attempting to classify Yelp Reviews into 1 star or 5 star categories based off the text content in the reviews. This will be a simpler procedure than the lecture, since we will utilize the pipeline methods for more complex tasks.

We will use the Yelp Review Data Set from Kaggle.

Each observation 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. (Higher stars is better.) 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.

Let's get started! Just follow the directions below!

Imports

Import the usual suspects. :)

In [ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
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 classification_report, confusion_matrix
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.pipeline import Pipeline
```

The Data

Read the yelp.csv file and set it as a dataframe called yelp.

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

Check the head, info , and describe methods on yelp.

In [ ]:
```
yelp.head()
```

Out[ ]:

	business_id	date	review_id	stars	text	type

	business_id	date	review_id	stars	text	type	
0	9yKzy9PApeiPPOUJEtnvkg	2011-01-26	fWKvX83p0-ka4JS3dc6E5A	5	My wife took me here on my birthday for breakf...	review	rLtl8ZkDX5vH5
1	ZRJwVLyzEJq1VAihDhYiow	2011-07-27	IjZ33sJrzXqU-0X6U8NwyA	5	I have no idea why some people give bad review...	review	0a2KyEL0d3Yb
2	6oRAC4uyJCsJl1X0WZpVSA	2012-06-14	IESLBzqUCLdSzSqm0eCSxQ	4	love the gyro plate. Rice is so good and I als...	review	0hT2KtfLiobPv
3	_1QQZuf4zZOyFCvXc0o6Vg	2010-05-27	G-WvGaISbqqaMHlNnByodA	5	Rosie, Dakota, and I LOVE Chaparral Dog Park!!...	review	uZetl9T0NcRO(
4	6ozycU1RpktNG2-1BroVtw	2012-01-05	1uJFq2r5QfJG_6ExMRCaGw	5	General Manager Scott Petello is a good egg!!!...	review	vYmM4K

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 10 columns):
 #   Column       Non-Null Count  Dtype 
---  ------       --------------  ----- 
 0   business_id  10000 non-null  object
 1   date         10000 non-null  object
 2   review_id    10000 non-null  object
 3   stars        10000 non-null  int64 
 4   text         10000 non-null  object
 5   type         10000 non-null  object
 6   user_id      10000 non-null  object
 7   cool         10000 non-null  int64 
 8   useful       10000 non-null  int64 
 9   funny        10000 non-null  int64 
dtypes: int64(4), object(6)
memory usage: 781.4+ KB
```

In [ ]:
```
yelp.describe()
```

Out[ ]:

	stars	cool	useful	funny
count	10000.000000	10000.000000	10000.000000	10000.000000
mean	3.777500	0.876800	1.409300	0.701300
std	1.214636	2.067861	2.336647	1.907942
min	1.000000	0.000000	0.000000	0.000000
25%	3.000000	0.000000	0.000000	0.000000
50%	4.000000	0.000000	1.000000	0.000000
75%	5.000000	1.000000	2.000000	1.000000
max	5.000000	77.000000	76.000000	57.000000

Create a new column called "text length" which is the number of words in the text column.

In [ ]:
```
yelp['text length'] = yelp['text'].apply(len)
yelp.head()
```

Out[ ]:

	business_id	date	review_id	stars	text	type	
0	9yKzy9PApeiPPOUJEtnvkg	2011-01-26	fWKvX83p0-ka4JS3dc6E5A	5	My wife took me here on my birthday for breakf...	review	rLtl8ZkDX5vH5
1	ZRJwVLyzEJq1VAihDhYiow	2011-07-27	IjZ33sJrzXqU-0X6U8NwyA	5	I have no idea why some people give bad review...	review	0a2KyEL0d3Yl
2	6oRAC4uyJCsJl1X0WZpVSA	2012-06-14	IESLBzqUCLdSzSqm0eCSxQ	4	love the gyro plate. Rice is so good and I als...	review	0hT2KtfLiobP
3	_1QQZuf4zZOyFCvXc0o6Vg	2010-05-27	G-WvGaISbqqaMHlNnByodA	5	Rosie, Dakota, and I LOVE Chaparral Dog Park!!...	review	uZetl9T0NcRO
4	6ozycU1RpktNG2-1BroVtw	2012-01-05	1uJFq2r5QfJG_6ExMRCaGw	5	General Manager Scott Petello is a good egg!!!...	review	vYmM4k

EDA

Let's explore the data

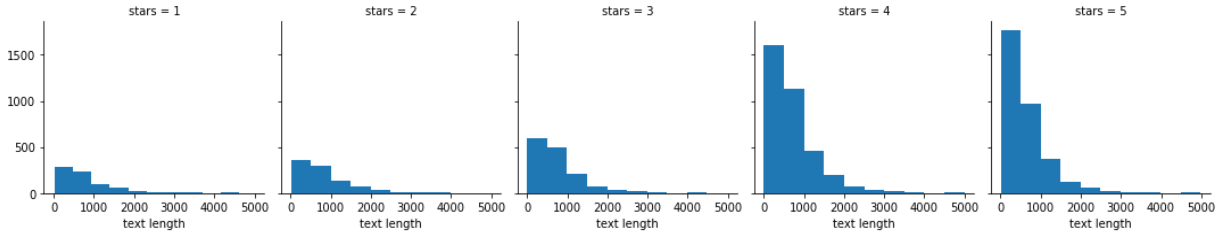
Imports

Import the data visualization libraries if you haven't done so already.

In [ ]:

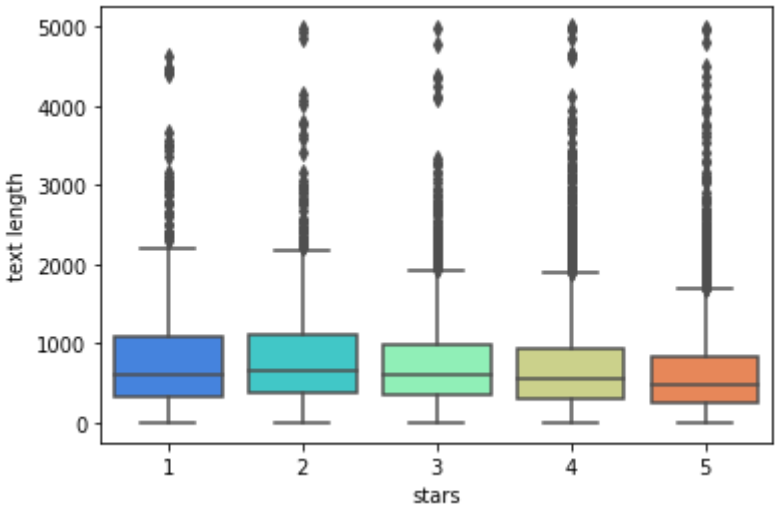
Use FacetGrid from the seaborn library to create a grid of 5 histograms of text length based off of the star ratings. Reference the seaborn documentation for hints on this

In [ ]:
```
g = sns.FacetGrid(yelp,col='stars')
g.map(plt.hist,'text length')
plt.show()
```



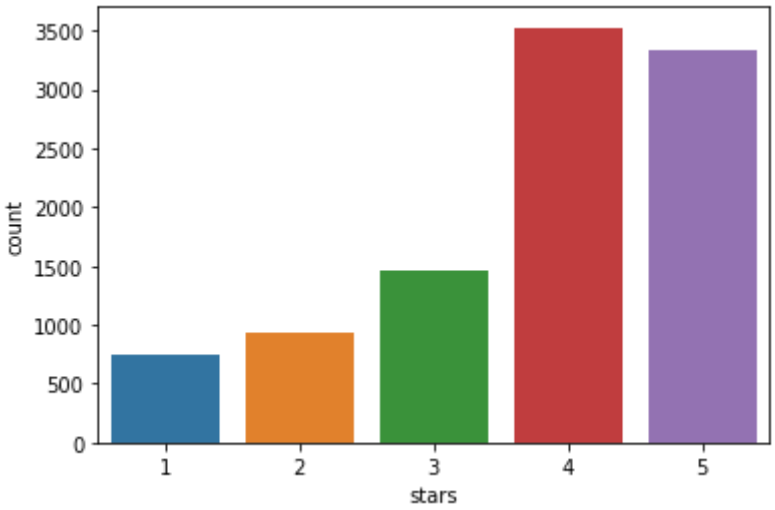
Create a boxplot of text length for each star category.

In [ ]:
```
sns.boxplot(x='stars',y='text length',data=yelp,palette='rainbow')
plt.show()
```



Create a countplot of the number of occurrences for each type of star rating.

In [ ]:
```
sns.countplot(x = 'stars', data=yelp)
plt.show()
```



Use groupby to get the mean values of the numerical columns, you should be able to create this dataframe with the operation:

In [ ]:
```
stars = yelp.groupby('stars').mean()
print(stars)
```

```
           cool    useful     funny  text length
stars
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
```

Use the corr() method on that groupby dataframe to produce this dataframe:

In [ ]:
```
stars.corr()
```

Out[ ]:

	cool	useful	funny	text length
cool	1.000000	-0.743329	-0.944939	-0.857664
useful	-0.743329	1.000000	0.894506	0.699881
funny	-0.944939	0.894506	1.000000	0.843461
text length	-0.857664	0.699881	0.843461	1.000000

Then use seaborn to create a heatmap based off that .corr() dataframe:

In [ ]:
```
sns.heatmap(stars.corr(),cmap='coolwarm',annot=True)
plt.show()
```

NLP Classification Task

Let's move on to the actual task. To make things a little easier, go ahead and only grab reviews that were either 1 star or 5 stars.

Create a dataframe called yelp_class that contains the columns of yelp dataframe but for only the 1 or 5 star reviews.

In [ ]:
```

```

```
yelp_class = yelp[(yelp.stars == 1) | (yelp.stars ==5)]
```

Create two objects X and y. X will be the 'text' column of yelp_class and y will be the 'stars' column of yelp_class. (Your features and target/labels)

In [ ]:
```
X = yelp_class.text
y = yelp_class.stars
```

Import CountVectorizer and create a CountVectorizer object.

In [ ]:
```
bow_transformer = CountVectorizer()
```

Use the fit_transform method on the CountVectorizer object and pass in X (the 'text' column). Save this result by overwriting X.

In [ ]:
```
X = bow_transformer.fit_transform(X)
print(X)
```

```
  (0, 11265)    2
  (0, 18735)    1
  (0, 17406)    1
  (0, 10635)    1
  (0, 8161)     2
  (0, 11821)    3
  (0, 1984)     1
  (0, 6864)     1
  (0, 2364)     1
  (0, 937)      8
  (0, 9114)     9
  (0, 18517)    8
  (0, 6123)     3
  (0, 17126)    10
  (0, 18585)    1
  (0, 12505)    1
  (0, 18672)    1
  (0, 10300)    2
  (0, 15422)    1
  (0, 12016)    1
  (0, 12054)    1
  (0, 17134)    4
  (0, 7715)     1
  (0, 921)      1
  (0, 407)      1
  :     :
  (4085, 9817)  1
  (4085, 10088) 1
  (4085, 9127)  1
  (4085, 10873) 1
  (4085, 2154)  1
  (4085, 183)   1
  (4085, 10907) 1
  (4085, 17416) 1
  (4085, 6370)  1
  (4085, 3314)  1
  (4085, 3329)  1
  (4085, 14423) 1
  (4085, 1442)  1
  (4085, 16134) 1
  (4085, 15944) 1
  (4085, 6651)  1
  (4085, 12298) 1
  (4085, 543)   1
  (4085, 17700) 1
```

```
(4085, 2515)  1
(4085, 189)   1
(4085, 9624)  1
(4085, 5510)  1
(4085, 836)   1
(4085, 15962) 1
```

Train Test Split

Let's split our data into training and testing data.

Use train_test_split to split up the data into X_train, X_test, y_train, y_test. Use test_size=0.3 and random_state=101

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

Training a Model

Time to train a model!

Import MultinomialNB and create an instance of the estimator and call is nb

In [ ]:
```
nb = MultinomialNB()
```

Now fit nb using the training data.

In [ ]:
```
nb.fit(X_train,y_train)
```

Out[ ]:
```
▾ MultinomialNB

MultinomialNB()
```

Predictions and Evaluations

Time to see how our model did!

Use the predict method off of nb to predict labels from X_test.

In [ ]:
```
predictions = nb.predict(X_test)
```

Create a confusion matrix and classification report using these predictions and y_test

In [ ]:
```
print(confusion_matrix(y_test, predictions))
print("\n")
print(classification_report(y_test, predictions))
```

```
[[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

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

Great! Let's see what happens if we try to include TF-IDF to this process using a pipeline.

Using Text Processing

Import TfidfTransformer from sklearn.

Import Pipeline from sklearn.

Now create a pipeline with the following steps:CountVectorizer(), TfidfTransformer(),MultinomialNB()

In [ ]:
```
pipeline = Pipeline([
    ('bow', CountVectorizer()),# string ke jumlah bilangan bulat token
    ('tfidf', TfidfTransformer()),# bilangan bulat dihitung hingga skor TF-IDF terti
    ('classifier', MultinomialNB())# melatih vektor TF-IDF dengan pengklasifikasi Na
])
```

Using the Pipeline

Time to use the pipeline! Remember this pipeline has all your pre-process steps in it already, meaning we'll need to re-split the original data (Remember that we overwrote X as the CountVectorized version. What we need is just the text

Train Test Split

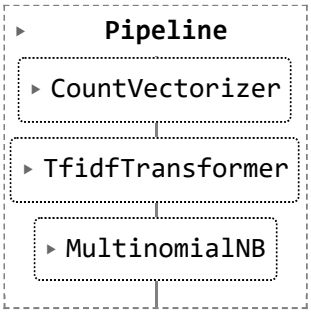
Redo the train test split on the yelp_class object.

In [ ]:
```
X = yelp_class.text
y = yelp_class.stars
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_st
```

Now fit the pipeline to the training data. Remember you can't use the same training data as last time because that data has already been vectorized. We need to pass in just the text and labels

In [ ]:
```
pipeline.fit(X_train, y_train)
```

Out[ ]:



Predictions and Evaluation

Now use the pipeline to predict from the X_test and create a classification report and confusion matrix. You should notice strange results.

In [ ]:

```
pipe_predictions = pipeline.predict(X_test)
print(confusion_matrix(y_test, pipe_predictions))
print("\n")
print(classification_report(y_test, pipe_predictions))
```

```
[[  0 228]
 [  0 998]]


              precision    recall  f1-score   support

           1       0.00      0.00      0.00       228
           5       0.81      1.00      0.90       998

    accuracy                           0.81      1226
   macro avg       0.41      0.50      0.45      1226
weighted avg       0.66      0.81      0.73      1226

c:\Users\Christian\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn
\metrics\_classification.py:1334: UndefinedMetricWarning: Precision and F-score are
ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_divi
sion` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
c:\Users\Christian\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn
\metrics\_classification.py:1334: UndefinedMetricWarning: Precision and F-score are
ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_divi
sion` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
c:\Users\Christian\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn
\metrics\_classification.py:1334: UndefinedMetricWarning: Precision and F-score are
ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_divi
sion` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

Lesson learned

In [ ]:

```
#1 Text processing dengan CountVectorizer dan NaiveBayes memberikan hasil yang lebih
```

Insight

In [ ]:

```
#1 CountVectorizer dan NaiveBayes memberikan hasil bagus dengan akurasi 93% dan reca
#2 MultinomialNB dan TfidfTransformer memberikan hasil yang kurang bagus dengan nila
```

Summary

In [ ]:

```
#1 Penggunaan CountVectorizer dan NaiveBayes memberikan hasil yang lebih baik diband
```