Instructor: Nasir Mahmood **Date: 30 November/2020**

Student: Fahad Ahmad

INFO70041_Assignment 5

1. **Import data(yelp.csv)**
   I used Kaggle for applying Natural learning Processing (NLP) algorithm to the yelp dataset.

[3]:
```
#Import the data, used Kaggle
yelpdata = pd.read_csv('../input/yelpdata1/yelp.csv')
yelpdata.head()
```

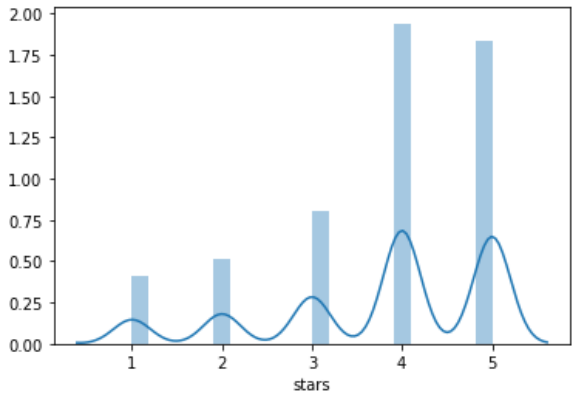
Out[3]:

	business_id	date	review_id	stars	text	type	user_id	cool	useful	funny
0	9yKzy9PApeiPPOUJEtnvkg	2011-01-26	fWKvX83p0-ka4JS3dc6E5A	5	My wife took me here on my birthday for breakf...	review	rLtl8ZkDX5vH5nAx9C3q5Q	2	5	0
1	ZRJwVLyzEJq1VAihDhYiow	2011-07-27	IjZ33sJrzXqU-0X6U8NwyA	5	I have no idea why some people give bad review...	review	0a2KyEL0d3Yb1V6aivbluQ	0	0	0
2	6oRAC4uyJCsJl1X0WZpVSA	2012-06-14	IESLBzqUCLdSzSqm0eCSxQ	4	love the gyro plate. Rice is so good and I als...	review	0hT2KtfLiobPvh6cDC8JQg	0	1	0
3	_1QQZuf4zZOyFCvXc0o6Vg	2010-05-27	G-WvGaISbqqaMHlNnByodA	5	Rosie, Dakota, and I LOVE Chaparral Dog Park!!...	review	uZetl9T0NcROGOyFfughhg	1	2	0
4	6ozycU1RpktNG2-1BroVtw	2012-01-05	1uJFq2r5QfJG_6ExMRCaGw	5	General Manager Scott Petello is a good egg!!!...	review	vYmM4KTsC8ZfQBg-j5MWkw	0	0	0

2. **Visualize the voting columns by either histogram, seaborn, or countplot.**

   Frequency distribution for star ratings shown below in histogram plot:

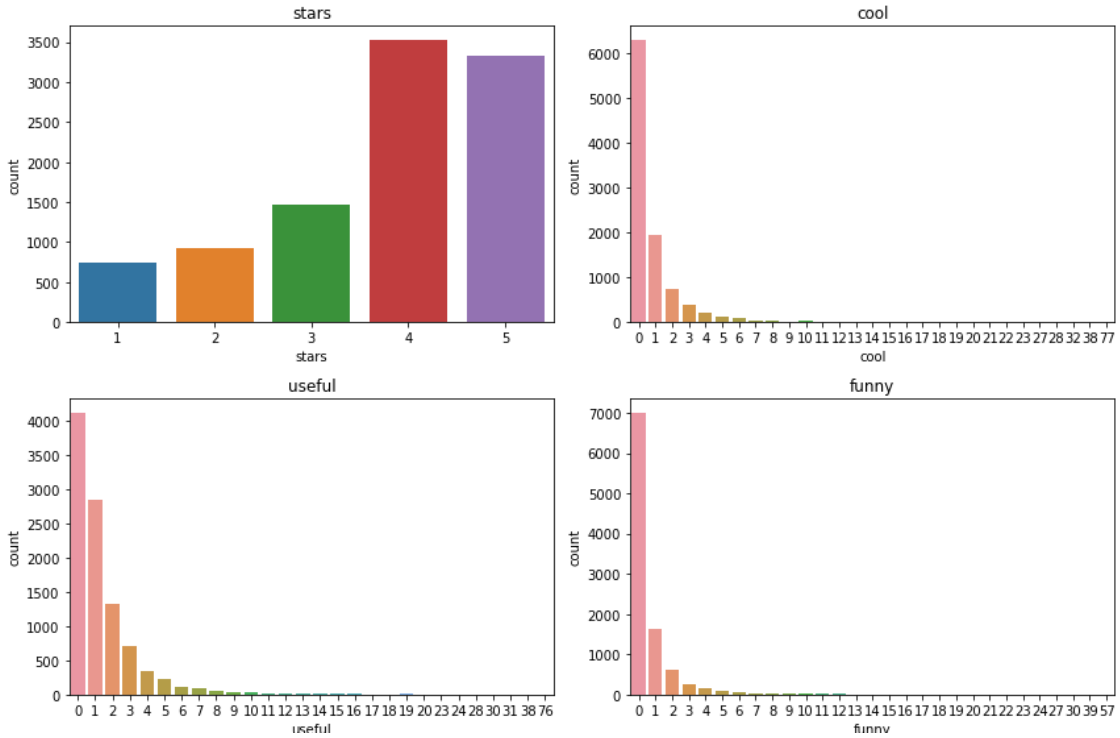
   sns.distplot(yelpdata["stars"])



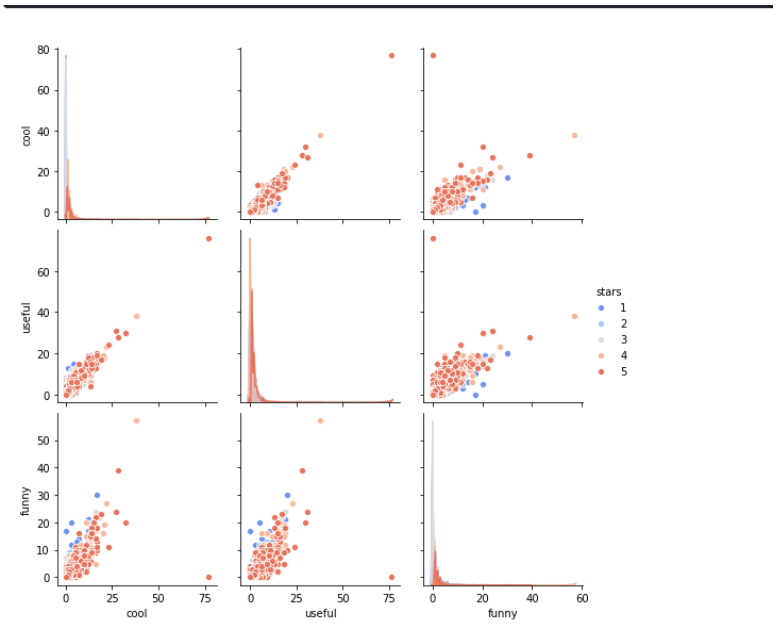
+ Code + Markdown

```
]:
#Plot the voting columns, star rating, cool rating, useful rating, funny rating
warnings.filterwarnings("ignore")

fig = plt.figure(figsize=(12,8))
axes1 = plt.subplot(2,2,1)
axes1 = sns.countplot(x='stars', data=yelpdata)
axes1.set_title('stars')
axes1.set_ylabel('count')

axes2 = plt.subplot(2,2,2)
axes2 = sns.countplot(x='cool', data=yelpdata)
axes2.set_title('cool')
axes2.set_ylabel('count')

axes3 = plt.subplot(2,2,3)
axes3 = sns.countplot(x='useful', data=yelpdata)
axes3.set_title('useful')
axes3.set_ylabel('count')

axes4 = plt.subplot(2,2,4)
axes4 = sns.countplot(x='funny', data=yelpdata)
axes4.set_title('funny')
axes4.set_ylabel('count')

plt.tight_layout()
```

```
axes4.set_ylabel('count')

plt.tight_layout()
```



sns.pairplot(yelpdata, hue='stars', palette='coolwarm')



3. **Make the dataframe (feature = text and label = stars) for the reviews with stars greater than zero.**

```
12]: #Create a new dataframe only with stars and text; stars column renamed to label and text renamed to
     #feature
     yelpclassdata = yelpdata.rename(columns = {"stars":"label","text":"feature"})
     yelpclassdata.head()
```

it[12]:

	business_id	date	review_id	label	feature	type	user_id	cool	useful	funny
0	9yKzy9PApeiPPOUJEtnvkg	2011-01-26	fWKvX83p0-ka4JS3dc6E5A	5	My wife took me here on my birthday for breakf...	review	rLtl8ZkDX5vH5nAx9C3q5Q	2	5	0
1	ZRJwVLyzEJq1VAihDhYiow	2011-07-27	IjZ33sJrzXqU-0X6U8NwyA	5	I have no idea why some people give bad review...	review	0a2KyEL0d3Yb1V6aivbluQ	0	0	0
2	6oRAC4uyJCsJl1X0WZpVSA	2012-06-14	lESLBzqUCLdSzSqm0eCSxQ	4	love the gyro plate. Rice is so good and I als...	review	0hT2KtfLiobPvh6cDC8JQg	0	1	0
3	_1QQZuf4zZOyFCvXc0o6Vg	2010-05-27	G-WvGaISbqqaMHlNnByodA	5	Rosie, Dakota, and I LOVE Chaparral Dog Park!!...	review	uZetl9T0NcROGOyFfughhg	1	2	0
4	6ozycU1RpktNG2-1BroVtw	2012-01-05	1uJFq2r5QfJG_6ExMRCaGw	5	General Manager Scott Petello is a good egg!!!...	review	vYmM4KTsC8ZfQBg-j5MWkw	0	0	0

```
13]: #Take only the columns 'feature' and 'label' from the data
```

4. **Tokenize the text and generate the word vector.**

Words in "feature" column tokenized and placed under the "feature1" column. This "feature1" column will then be the "X" data to be split into training and test sets.

```
#iterating through each text
for txt in yelpclassdata['feature']:

    #splits txt into words, drop any non-alphabet characters
    w_list = re.sub(r'[^a-z]',' ', txt.lower()).split()

    #add another column with tokens of that text as features
    yelpclassdata.loc[i,'feature1'] = ' '.join(w_list)

    #Remove stopwords from w_list #If in there don't include it; English function has a list of stop words
    w_list = [w for w in w_list if w not in stopwords.words('english')]

    #List of all the tokens without stopwords
    word_list += w_list

    #Incrementing the index variable
    i += 1

yelpclassdata.head()
```

:[18]:

	feature	label	feature1
0	My wife took me here on my birthday for breakf...	5	my wife took me here on my birthday for breakf...
1	I have no idea why some people give bad review...	5	i have no idea why some people give bad review...
2	love the gyro plate. Rice is so good and I als...	4	love the gyro plate rice is so good and i also...
3	Rosie, Dakota, and I LOVE Chaparral Dog Park!!...	5	rosie dakota and i love chaparral dog park it ...
4	General Manager Scott Petello is a good egg!!!...	5	general manager scott petello is a good egg no...

Word vector generated with the code below.

[19]:

```
#From INFO70041 walkthrough
#Get frequency distribution of words
all_words_fd =nltk.FreqDist(word_list)
#Get most common 3000 of them and store in word_common
word_common = all_words_fd.most_common(3000)
#Store word_features as the first element in thecouple
word_features = [w[0] for w in word_common]

print(word_common[0:5])
print(word_features)
```

```
[('good', 6801), ('place', 6662), ('food', 6184), ('great', 5127), ('like', 5041)]
['good', 'place', 'food', 'great', 'like', 'one', 'get', 'go', 'time', 'really', 'service', 'would', 'back', 'also', 'love', 'little', 'nice', '
'got', 'pretty', 'much', 'restaurant', 'chicken', 'try', 'ordered', 'menu', 'people', 'first', 'know', 'bar', 'order', 'could', 'think', 'better
'staff', 'night', 'way', 'going', 'cheese', 'pizza', 'right', 'two', 'delicious', 'made', 'came', 'say', 'want', 'salad', 'lunch', 'come', 'new'
ke', 'eat', 'see', 'experience', 'since', 'sure', 'definitely', 'happy', 'around', 'wait', 'something', 'times', 'ever', 'next', 'find', 'everyt
gh', 'give', 'meal', 'area', 'said', 'bad', 'table', 'many', 'location', 'dinner', 'thing', 'lot', 'hour', 'last', 'another', 'phoenix', 'side',
orite', 'sandwich', 'hot', 'stars', 'tasty', 'burger', 'drinks', 'feel', 'drink', 'sweet', 'things', 'enough', 'awesome', 'fries', 'store', 'bee
'minutes', 'atmosphere', 'worth', 'bread', 'need', 'coffee', 'different', 'work', 'tried', 'took', 'clean', 'ok', 'old', 'breakfast', 'places',
'recommend', 'excellent', 'years', 'actually', 'huge', 'nothing', 'friend', 'check', 'friends', 'found', 'asked', 'visit', 'kind', 'free', 'prob
'thought', 'scottsdale', 'special', 'server', 'anything', 'however', 'maybe', 'quite', 'super', 'sushi', 'perfect', 'rice', 'full', 'large', 'co
un', 'beef', 'told', 'cream', 'usually', 'review', 'dish', 'let', 'away', 'outside', 'inside', 'far', 'else', 'oh', 'fried', 'enjoy', 'spot', 'u
yone', 'decided', 'ask', 'half', 'high', 'left', 'pork', 'without', 'overall', 'spicy', 'soup', 'served', 'getting', 'least', 'family', 'red',
cent', 'water', 'dishes', 'loved', 'enjoyed', 'put', 'couple', 'almost', 'may', 'hard', 'makes', 'course', 'music', 'eating', 'yes', 'ice', 'pla
t', 'fan', 'use', 'fantastic', 'green', 'chips', 'week', 'part', 'fish', 'wonderful', 'restaurants', 'dining', 'must', 'reviews', 'local', 'shop
'tasted', 'several', 'valley', 'shrimp', 'kids', 'live', 'steak', 'customer', 'care', 'especially', 'stuff', 'style', 'chocolate', 'hours', 'eit
ess', 'done', 'year', 'stop', 'tables', 'door', 'quick', 'guy', 'dessert', 'waitress', 'finally', 'end', 'although', 'sit', 'name', 'disappointe
'comes', 'decor', 'bring', 'drive', 'trying', 'walk', 'gave', 'line', 'might', 'cooked', 'thai', 'yelp', 'yummy', 'salsa', 'fact', 'called', 'pa
```

5. **Divide the dataframe into train and test sets.**

[22]:
```
# define X and y
#Pass in tokenized data into X
X = yelpclassdata["feature1"]
y = yelpclassdata["label"]
```

[23]:
```
#Implement tokenization by creating object
#obj = PreProcessText()
#X = obj.token_words(X)
#print(X)
```

+ Code + Markdown

[24]:
```
# split X and y into training and testing sets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3,random_state = 101)
```

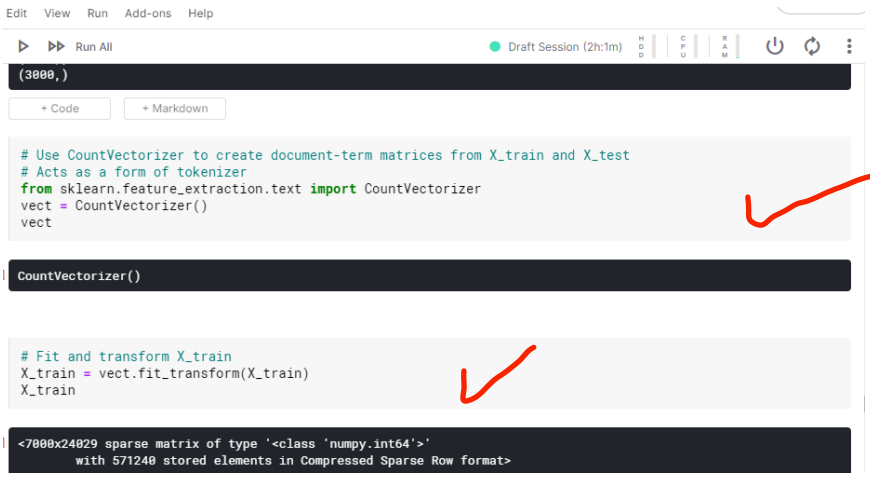
6. **Fit a classifier on the train data.**

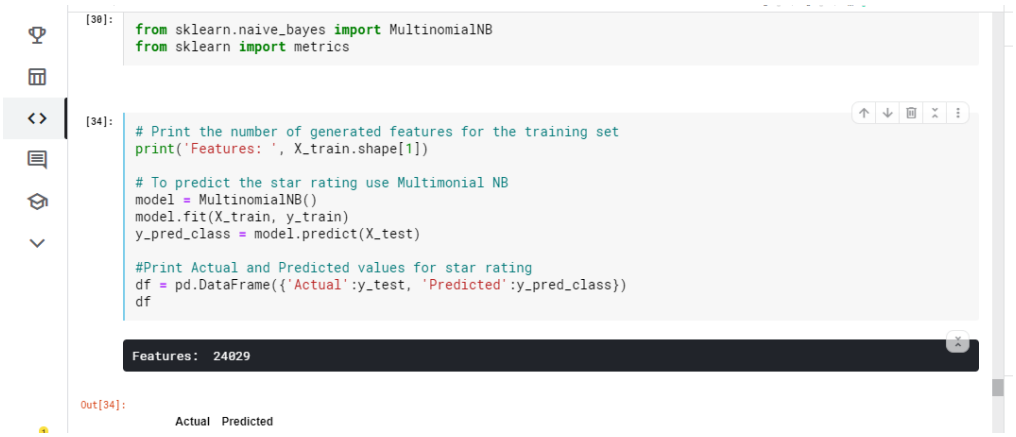
First, pass in the tokenized words into C

define X and y
#Pass in tokenized words into X
X = yelpclassdata["feature1"]

y = yelpclassdata["label"]

Then use CounteVectorizer for feature representation, that is to create document-term matrices from X_train and X_test

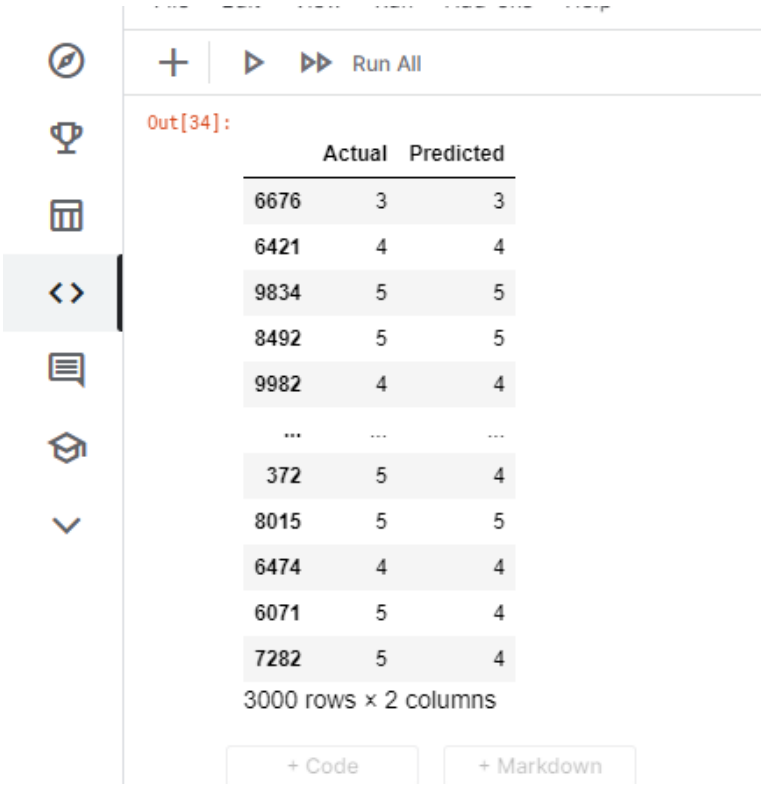




```
[30]:
from sklearn.naive_bayes import MultinomialNB
from sklearn import metrics

[34]:
# Print the number of generated features for the training set
print('Features: ', X_train.shape[1])

# To predict the star rating use Multimonial NB
model = MultinomialNB()
model.fit(X_train, y_train)
y_pred_class = model.predict(X_test)

#Print Actual and Predicted values for star rating
df = pd.DataFrame({'Actual':y_test, 'Predicted':y_pred_class})
df

Features:  24029

Out[34]:
```

Actual Predicted

7. **Predict the label of the test data.**



Run All

Out[34]:

	Actual	Predicted
6676	3	3
6421	4	4
9834	5	5
8492	5	5
9982	4	4
...	...	...
372	5	4
8015	5	5
6474	4	4
6071	5	4
7282	5	4

3000 rows × 2 columns

+ Code + Markdown

8. **Generate the confusion matrix or accuracy of the predicted and real label of the test set.**

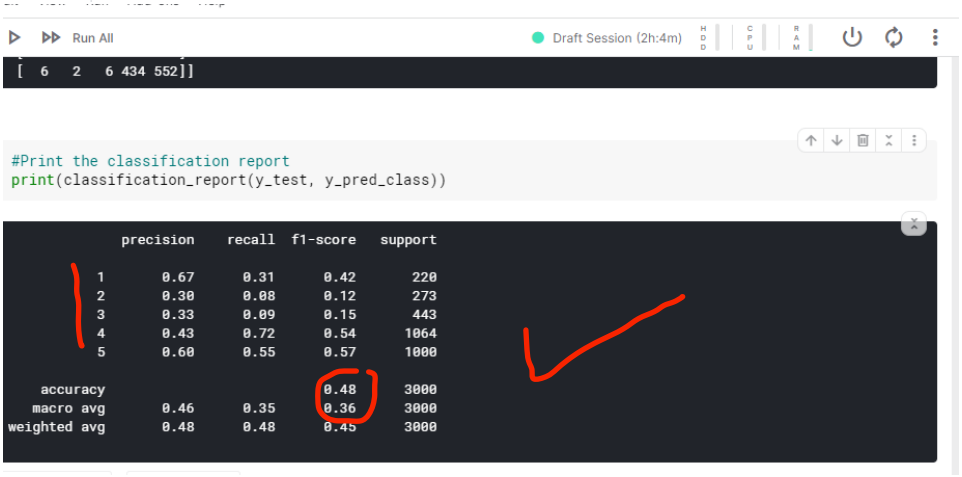
Accuracy of the prediction from the Naïve Bayes model for star rating or predicting labels for features is 48.2%.

Confusion matrix and Classification report located below.

[32]:

```
# Print the accuracy of its predictions
print('Accuracy: ', metrics.accuracy_score(y_test, y_pred_class))

# Print the confusion matrix
print('Confusion Matrix: ')
print(metrics.confusion_matrix(y_test, y_pred_class))
```

```
Accuracy:  0.4826666666666667
Confusion Matrix:
[[ 68  34  13  74  31]
 [ 16  21  42 162  32]
 [  5  11  42 341  44]
 [  7   3  24 765 265]
 [  6   2   6 434 552]]
```



From the classification rating, can see that Precision was best for the 1 star rating and lowest for 2 star and 3 star ratings. Ability to not label the text incorrectly, was best for 1 star and 5 star ratings.