RATINGS PREDICTION PROJECT

Submitted by:

Snehal Kumawat

INTRODUCTION

- ## Business Problem Framing

Online product reviews are a great source of information for consumers. From the sellers' point of view, online reviews can be used to gauge the consumers' feedback on the products or services they are selling. However, since these online reviews are quite often overwhelming in terms of numbers and information, an intelligent system, capable of finding key insights (topics) from these reviews, will be of great help for both the consumers and the sellers.

- ## Conceptual Background of the Domain Problem

We have a client who has a website where people write different reviews for technical products. Now they are adding a new feature to their website i.e. The reviewer will have to add stars(rating) as well with the review. The rating is out 5 stars and it only has 5 options available 1 star, 2 stars, 3 stars, 4 stars, 5 stars. Now they want to predict ratings for the reviews which were written in the past and they don't have a rating. So, we have to build an application which can predict the rating by seeing the review.

- ## Review of Literature

The following article describes the application of a range of supervised and unsupervised machine learning models to a dataset of product reviews in an effort to predict rating value. The desired output to an input of a text review is a rating on a continuum from 1 to 5.

- ## Motivation for the Problem Undertaken

When you shop in any shopping site, Do you notice that product reviews are in the order of helpfulness of reviews? This helpfulness is based on user's votes. If Amazon users find a product review helpful, users can simply leave positive votes on that review. The review with most positive votes gets placed as "Top Customer Reviews" on the product page by Amazon.

- ## Data Sources and their formats

The data set contains the training set, which has approximately 20163 samples. The data samples contain 2 fields which includes 'full reviews', 'ratings'. Full reviews contains reviews of the product and rating are from 1,2,3,4,5.

.

1. Data Preparation and Cleaning

Fig. 1 captures the research framework for the ranking prediction problem. This section focuses on different algorithms and the various stages that are involved for the proposed rating prediction system such as 'Decision Tree Classifier' and 'Random Forest Classifier' which helps us in predicting ratings and results in a decisive outcome. It includes five major blocks, namely data collection, data preparation, feature processing, mode training, classification and model evaluation. These blocks of the diagram are explained in detail in the next subsections.

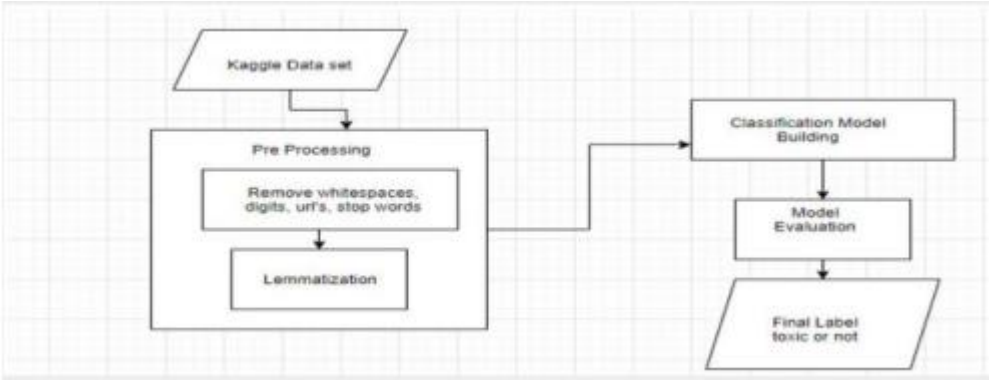


fig 1. Research framework

About Project

The data set contains the training set, which has approximately 20163 samples. The data samples contain 2 fields which includes 'full reviews', 'ratings'. Full reviews contains reviews of the product and rating are from 1,2,3,4,5.

Packages used: Pandas, Numpy, Seaborn, Matplotlib, Scikit and Stats models, NLTK, Lemmeaizer.

Visualization

For the visualisation, I had length of ratings on the independent axis. But instead of counting number of ratings, I counted ratings belonging to each of the different categories.

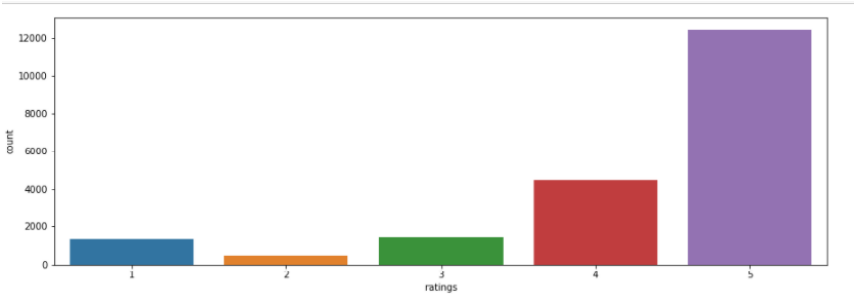


fig2.comments belonging to different categories

1. Data Preprocessing

Text preprocessing is an important step for any natural language processing task. It transforms text from its raw format into something that is more processable for computer algorithms. The

goal of our preprocessing is to achieve normalization and noise removal from the dataset befor e we experiment with different embeddings and models.

1. **Preparation for removal of punctuation marks:** I imported the string library comprisi ng all punctuation characters and appended the numeric digits to it, as those were requir ed to be removed too.

```
import string
print(string.punctuation)
punctuation_edit = string.punctuation.replace('\'','') +"0123456789"
print (punctuation_edit)
outtab = "                                         "
trantab = str.maketrans(punctuation_edit, outtab)
```

```
!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~
!"#$%&()*+,-./:;<=>?@[\]^_`{|}~0123456789
```

2. Updating the list of stop words : Stop words are those words that are frequently used in both written and verbal communication and thereby do not have either a positive/negative impact on our statement. E.g. is, this, us, etc.

Python has a built-in dictionary of stop words. I used the same and also appended the single letters like ‘b’, ‘c’ …. to it, which might be pre-existing or have generated during data preprocessing.

```
from stop_words import get_stop_words
stop_words = get_stop_words('english')
stop_words.append('')

for x in range(ord('b'), ord('z')+1):
    stop_words.append(chr(x))
```

3. Stemming and Lemmatising : Stemming is the process of converting inflected/derived words to their word stem or the root form. This helps in achieving the training process with a better accuracy.

Lemmatising is the process of grouping together the inflected forms of a word so they can be analyzed as a single item. I used the **word-net library** in nltk for this purpose. Stemmer and Lemmatizer were imported from nltk

```
import nltk
from nltk.stem import PorterStemmer, WordNetLemmatizer
```

```
#create objects for stemmer and lemmatizer
lemmatiser = WordNetLemmatizer()
stemmer = PorterStemmer()
#download words from wordnet library
nltk.download('wordnet')
```

```
for i in range(len(comments)):
    comments[i] = comments[i].lower().translate(trantab)
    l = []
    for word in comments[i].split():
        l.append(stemmer.stem(lemmatiser.lemmatize(word,pos="v")))
    comments[i] = " ".join(l)
```

4. Applying Count Vectorizer : Count Vectorizer is used for converting a string of wo rds into a matrix of words. Column headers have the words themselves and the cell val ues signify the frequency of occurrence of the word.

```
#import required library
from sklearn.feature_extraction.text import CountVectorizer

#create object supplying our custom stop words
count_vector = CountVectorizer(stop_words=stop_words)
#fitting it to converts comments into bag of words format
tf = count_vector.fit_transform(comments).toarray()
```

After doing all this process the clean comments are as follows and length column cont ains original length of comments and clean_length column contains length of clean co mments.

	id	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe	length	clean_length
0	0000997932d777bf	explanation edits made username hardcore metal...	0	0	0	0	0	0	264	180
1	000103f0d9cfb60f	d'aww! match background colour i'm seemingly s...	0	0	0	0	0	0	112	111
2	000113f07ec002fd	hey man, i'm really trying edit war. guy const...	0	0	0	0	0	0	233	149
3	0001b41b1c6bb37e	can't make real suggestion improvement wondere...	0	0	0	0	0	0	622	397
4	0001d958c54c6e35	you, sir, hero. chance remember page that's on?	0	0	0	0	0	0	67	47

5. WordCloud

Word Cloud is a data visualization technique used for representing text data in which t he size of each word indicates its frequency or importance. Word clouds are widely us ed for analyzing data from social network websites. For generating word cloud in Pyth on, modules needed are – matplotlib, pandas and wordcloud.

cloud graph for rating 5

When Rating Score is 5 or 4, most of the frequently used words in the review texts are positive, s uch as good, good product, nice, best, great, awesome.

Feature Scaling

The engineered features are normalized from 0.0 to 1.0. The tf-idf features are not scaled.

3. Methodology

Finding the Best Algorithm To compare the relative performances of each algorithm, I'm going to test them on the same preprocessed data, a tf-idf vectorized data with features. The target is any_label, and the scores are F1 scores with five-fold cross validation.

4. Classification Model Building:

Classification is the process of predicting the class of given data points. Classes are sometimes called as targets/ labels or categories. Classification predictive modeling is the task of approximating a mapping function (f) from input variables (X) to discrete output variables (y).

Splitting the Dataset

As usual for supervised machine learning problems, we need a training dataset to train our model and a test dataset to evaluate the model.

Decision Tree Classifier

A decision tree is a simple representation for classifying examples. For this section, assume that all of the i nput features have finite discrete domains, and there is a single target feature called the "classification".

```
Training accuracy is 0.8615245456788093
Test accuracy is 0.7769841848261788
```

K-Neighbors Classifier

The K in the name of this classifier **represents the k nearest neighbors**, where k is an integer value specified by the user.

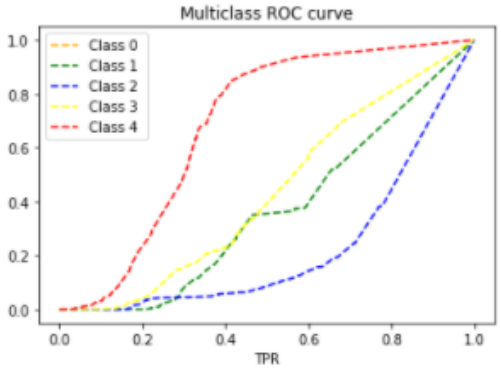
Using KNeighborsClassifier and cross validation we got

```
Training accuracy is 0.7310870816544692
Test accuracy is 0.6999489833102543
```

Random Forest Classifier

A random forest classifier. A random forest is **a meta estimator that fits a number of decision tree c lassifiers on various sub-samples of the dataset** and uses averaging to improve the predictive accura cy and control over-fitting. The number of trees in the forest.

We got
```
Training accuracy is 0.8615245456788093
Test accuracy is 0.815173821150062
```



Auc ROC curve

The best model is **Random Forest Classifier**

Since the difference between the percentages score of cross validation and r2_score is optimum. After that we save the best model using pickle method.

5. Conclusion

It is seen that the most effective model in predicting ratings is Random Forest Classifier. Afte r predicting test dataset we get the following results.

And here is output for test dataset values.

	original	predicted
0	4	4
1	2	3
2	4	4
3	4	5
4	5	5
...	...	...
13716	5	5
13717	2	2
13718	3	3
13719	5	4
13720	5	5

13721 rows × 2 columns

6. Limitations of this work and Scope for Future Work

The individual features, feature combinations within each category, and all feature comb inations that lead to optimal performance have been studied. The usefulness of a review is lik ely to depend on numerous factors that are difficult to isolate and study. The empirical results

set comparable and reproducible baselines for review helpfulness prediction, and highlight the feature combination patterns that lead to general good prediction performance.

We suggest a plan to improve the NLP classifiers: first by using other algorithms which such as Support Vector Clustering (SVC) and Convolutional Neural Networks (CNN).
b) We also suggest using SVM for text processing and text classification. It requires a grid sea rch for hyper-parameter tuning to get the best results.