Capstone Project

Machine Learning Engineer Nanodegree

Mohamed Murad

9 August 2018

Reviews Sentiment Analysis

Definition:

1. Project Overview

Sentiment Analysis domain is a domain in which we analyze inputs such as reviews, messages, comments..etc to extract the sentiment from these given data and predict if the sentiment is positive or negative.

In Reviews Sentiment Analysis problem, we predict the sentiment of a given review whether it is a positive or a negative review.

In this project we use a dataset[i] combined from three sources, Amazon, Yelp and imdb. we combined reviews from these three sources to get a dataset of about 2748 examples of labeled reviews. The data set is published on Kaggle, and I decided to use it in this common problem.

2. Problem Statement

The goal is to train a model which it is able to predict if a given review is a positive review or a negative review, to solve this problem we take the following steps:

1- load the data and combine them in a single data frame.
2- process the inputs using count vectorizer.
3- train two different classifiers, Multinomial Naive Bayes and Bernoulli Naive Bayes.
4- Use Grid Search for tunning parameters to enhance the accuracy.
5- Discuss the results and make a decision for which classifier is better.

The final model is expected to take any review as a text input and return a 0 or 1 output. 0 if the review is predicted as negative and 1 if positive.

3. Metrics

We use accuracy_score[ii] Metric to compare with the benchmark accuracy and test the performance of our solution.

We selected this metrics specifically as we find out that our dataset is balanced, we found that about 49.6% of the examples are negative and about 50.4% of the examples are positive. There is a little difference but we can neglect it and consider our data set is balanced.
And as our dataset is considered Balanced, so the best choice Metric is Accuracy_Score.

Analysis:

1. Data Exploration

Our dataset is consists of one input feature and one output label.
The input feature is a text sentences with the name “Review”.
And Output which named as “sentiment” has two possible values; 0 as negative and 1 as positive.

	review	sentiment
0	So there is no way for me to plug it in here i...	0
1	Good case, Excellent value.	1
2	Great for the jawbone.	1
3	Tied to charger for conversations lasting more...	0
4	The mic is great.	1
5	I have to jiggle the plug to get it to line up...	0
6	If you have several dozen or several hundred c...	0
7	If you are Razr owner...you must have this!	1
8	Needless to say, I wasted my money.	0
9	What a waste of money and time!.	0

By explore our dataset we find that we have about 1362 negative labeled reviews and 1386 positive reviews.

```
postive reviews percentage:  0.49563318777292575
negative reviews percentage:  0.5043668122270742
```

The percentage of these examples is about 49.6% and 50.4% respectively.

So we find the accuracy metrics is good for our problem.

To explore one example from our dataset, let's take the 4th example
"the mic is great" labeled as 1.
it's obvious that the review is positive.
Our input features need to be transformed to another representation so that we can benefit from these text data.
e.g. If we used bag of words, our input feature will be an array of zeros and ones.
But the output need no transformation, we will use it as it is. 0 for negative review and 1 for positive.

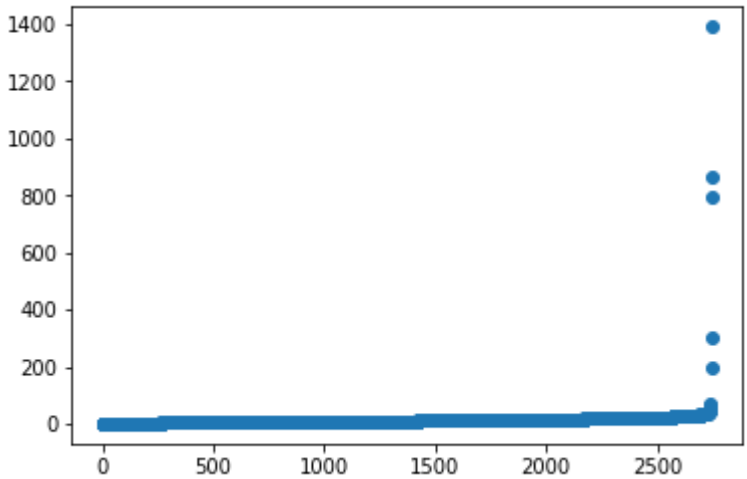
If we take another example as "the mic is not great" labeled as 0.
So we should take into consideration the negation of a sentence and not just the positive and negative words. We can solve this problem with ngrams for example.

let's do some statistics.

```
max number of words:  1390
min number of words:  1
mean is:  13.006550218340612
meadian is:  10.0
standard deviation is:  35.81287227056574
```

So we find that our range of number of words in reviews is [1:1390], the mean is about 13, std is about 35.8 and median is 10.

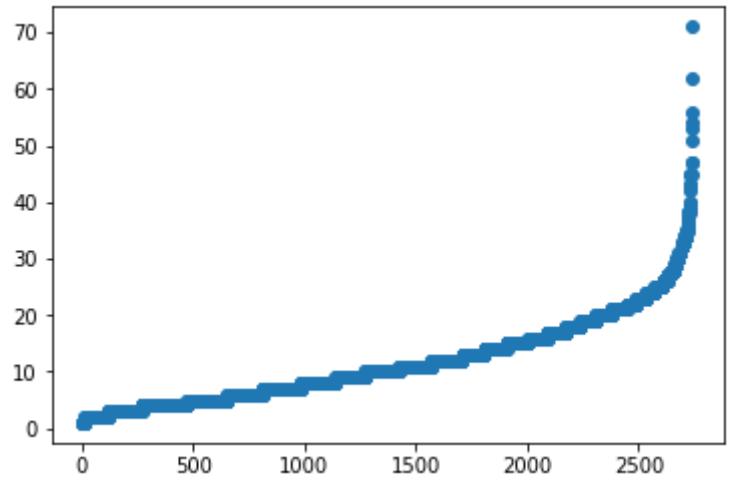
the following scatter plot show the number of words correspond to each review.



X-axis: is the review number, y-axis is the corresponding number of words in the review.

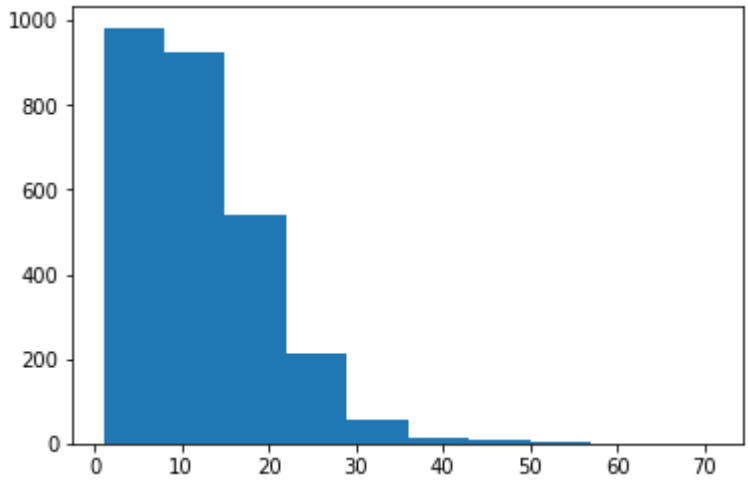
We notice that all reviews have number of words between 1 and 100 and the last few reviews consists of more than 100 words which reach to 1490 words in the last review.
So we have very long review.
let’s remove them and plot again.

The following plot is the same but after removing the last 5 reviews.



We notice that now there are no abnormals length of reviews and there is no need to remove any other examples. Now there are no very long nor very short reviews.
And all reviews consists of number of words in the range [0:71].

Now lets plot the distribution.



The above histogram show that most of reviews have few number of words, most of the reviews have number of words between 1 and 35.

2. Exploratory Visualization

We need not to make many visualizations in our dataset, as we have only one input feature and one output label.
But to use our dataset we need to find if the examples is shuffled or not.

This scatter show the sentiment of each example of our dataset.
Note: you can zoom in it and out in the notebook.
The x-axis represents the example number, the y-axis represents the sentiment of that example, 0 or 1.

from this plot we find that the data is shuffled and no bias will happen when we work on our dataset.

3. Algorithms and techniques

we use two different algorithms to solve our problems.
Both algorithms we used are types of Naive Bayes, Naive Bayes is a supervised learning methods based on applying Bayes' theorem with strong (naive) feature independence assumptions.

1. Multinomial Naive Bayes[iii]:
"Naive Bayes classifier for multinomial models
The multinomial Naive Bayes classifier is suitable for classification with discrete

features (e.g., word counts for text classification). The multinomial distribution normally requires integer feature counts.” doc.

$$\hat{\theta}_{yi} = \frac{N_{yi} + \alpha}{N_y + \alpha n}$$

2. Bernoulli Naive Bayes[iv]:

“Naive Bayes classifier for multivariate Bernoulli models.
Like Multinomial NB, this classifier is suitable for discrete data. The difference is that while Multinomial NB works with occurrence counts, BernoulliNB is designed for binary/boolean features.”doc.

$$P(x_i \mid y) = P(i \mid y)x_i + (1 - P(i \mid y))(1 - x_i)$$

Alpha parameter is used in both, which it represents smoothing, when alpha is 0 there is no smoothing. Default is 1.

Technique we used are,

1. bag of words:
to convert our text input features to an array of zeros and ones, so that we can use this feature.
We do that using CountVectorizer which exist in sklearn.feature_extraction.text.
It do everything for us.

The default parameters are:
stop_words=None,
lowercase=True,
ngram_range=(1,1),
max_features=None

2. GridSearch:
this technique used to tune parameters so that we get better accuracy.
The parameter we deal with in this class are, scoring, estimator, and parameters_grid we should pass these parameters with values.
parameters_grid consists of all paramters of the model we need to tune with all possible ranges of values.

4. Benchmark

Our selected benchmark is LinearSVC,

“Similar to SVC with parameter kernel=’linear’, but implemented in terms of

liblinear rather than libsvm, so it has more flexibility in the choice of penalties and loss functions and should scale better to large numbers of samples.” doc [v]

we first preprocessed our features using count vectorizer, and then trained our benchmark with our dataset and tested it.

We got a result accuracy of about 78%

So, in our solution we need to pass this accuracy.

Methodology:

1. Data Preprocessing

as we discussed before, we can’t use the text input feature as it is,we need to transform it to another representation.
One of the best suitable representation is bag of words.
Bag of words used to convert raw text data to useful numerical values, it extract the features needed to be fed to our algorithms.
It tokenizing, counting and normalizing in order.

We get an array of all possible words with 1 at used words and 0 at not used words. This result is replaced by the raw text input feature and this is what we need next in the training phase.

We avoid stop words in our transformation, besides we convert all inputs to lower case first.
We take the n-grams into consideration also.

The following are the new representation of our first example in training and test set.

```
print("first training example new representation\n",training_data[0])
print("first testing example new representation\n",testing_data[0])
```

```
first training example new representation
   (0, 6726)    1
  (0, 240)      1
first testing example new representation
   (0, 288)     1
  (0, 4682)     1
  (0, 6187)     1
```

There is no need of any other preprocessing.

2. Implementation:

1. bag of words:
to Implement bag of words technique, we used Count Vectorizer which it is in sklearn.feature_extraction.text.

We need to convert all sentences to lower case first so that we avoid the effect of the word case (upper or lower)
we set the parameter lowercase=True in instantiating the CountVectorizer().

To avoid words such as “in, I, he, we, to...etc”. any ineffective words in sentiment, we set stop_words='english' also while instantiating.

In case of negation or any effect that generated from two or more words, we consider n-grams parameter, we set it to (1,3), to consider all combinations of 3 words at maximum.

Now our instantiated count_vectorizer is ready, we fit our training set to it by using fit_transform method to transform the training set after fitting, passing input features after processing and their labels.

We transform also our testing sets using our count_vectorizer.

Now we finished our bag of words transformation using count_vectorizer.

Our testing and training sets now is ready to be used in our algorithms solutions.

2. Algorithms:
we need to solve our problems using two different algorithms. MultinomialNB and BernoulliNB.

We simply instantiate MultinomialNB, fit it using new training set and the corresponding labels, then to test accuracy we predict using the new testing set.

3. plot_confusion_matrix:
there is an implemented function used to print a confusion matrix of the results, there is no difficulties in implementing it.
Just use the function and pass needed parameters to plot a confusion matrix.

4. metrics:
to use our selected metrics “accuracy”, we simply import it from sklearn.metrics, it’s

named as accuracy_score.

To run it, we pass two parameters to accuracy_score, y_labels, and predected_y_labels. The calculated accuracy will be returned.

3. Refinement:

we used Grid Search to tune our model's parameters.

To use GridSearchCV, we first import it from sklearn.grid_search, we instantiate it, and pass the three needed paramters as discussed before.

We are interested to tune only the alpha paramters,
our range of values are: 'alpha' :
[0.001,0.01,0.1,0.2,0.3,0.4,0.5,0.6,0.75,0.8,0.9,1,1.1,1.2,1.3,1.4,1.5,1.6,……..,2.5,3]

when we run our grid search the accuracy is calculated for each of these alpha values.

We return the best classifer which it's the classifier that give the best accuracy by using grid_fit.best_estimator_

Our accuracy for both of our algorithms before refinment was 79.091% and 80.182% but after refinement the accuracy became `79.455% and 80.909% respectively.`

`So, there is about 1% of enhancement.`

Results:

1. Model Evaluation and Validation

Our final model is Bernoulli Naive Bayes.
We tried two algorithms of Naive Bayes to solve our problem, Multinomial and Bernoulli.
We fit our model with training set, and made predictions using testing set to calculate the score.
We got an accuracy of `79.091% and 80.182%` respectively.
After tuning parameters we got an accuracy of `79.455% and 80.909%` respectively.
That passes the bench mark accuracy.
We select Bernoulli model as it give a better accuracy.

	benchmark	MultinomialNB	BernoulliNB
accuracy	79.273%	79.091%	80.182%
Optimized	-----	79.455%	80.909%

The following plot show the final results.

We tested our final model to make sure it’s working for general input.

When we tried: 'I am very excited with your product', it give positive and that’s correct.
When we tried: 'the movie is very slow, I did not enjoy it', it give negative and that is also correct.

Our two input examples predicted true, but it’s obviously that the accuracy is not very advanced to predict more complicated reviews. I think more huge dataset will make our accuracy pass 90%

So, I think the model is robust for the problem but it needs to be trained on a bigger dataset of about 100000 examples of reviews at least not just 2748.

But anyway most of the results from this results can be trusted with this acceptable accuracy.

2. Justification

the following is a summary of our results:

	benchmark	MultinomialNB	BernoulliNB
accuracy	79.273%	79.091%	80.182%
Optimized	-----	79.455%	80.909%

So, It's obvious that we can overcome the benchmark results by using Mutlinomial Naive bayes and Bernoulli Naive bayes.

The benchmark accuracy is 79.273% but Multinomial are less but very close to the benchmark accuracy and Bernoulli Naive Bayes algorithm are better than the benchmark algorithm in solving our problems (as they give a higher accuracy, 79.091% and 80.182% respectively).

So we easily managed to pass the benchmark accuracy without any optimization using Bernouli Naive Bayes.

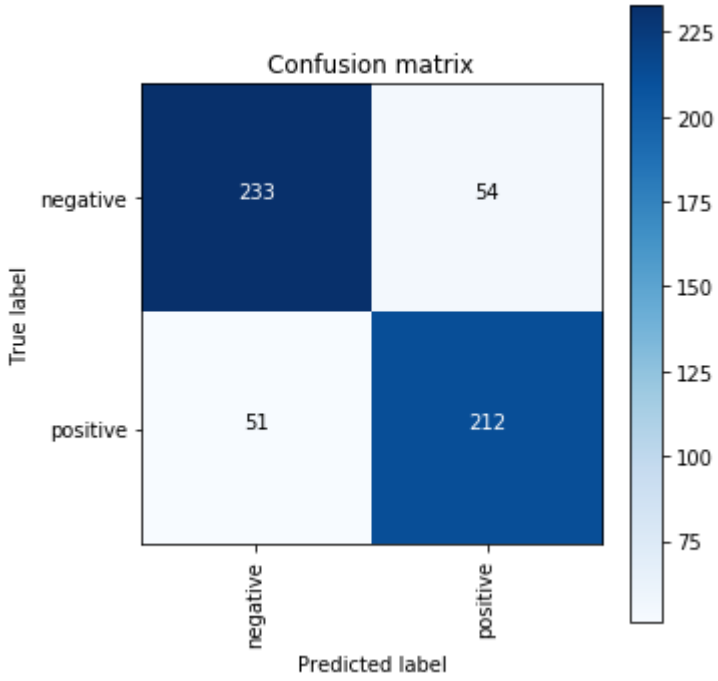
When we use GridSearch to optimize our two algorithms, we got, 79.455% and 80.909% for the optimized Mutinomial and Bernoulli Naive Bayes models which they are better than the benchmark accuracy.

To compare the Two used algorithms, we find Bernoulli Naive Bayes give better accuracy in solving our preoblems. that is obvious because our problem's output is binary (negative or positive) and Bernoulli Naive Bayes is a good choice for these kinds of problems.

Conclusion:

1. Free Form Visualization

The following plot is a confusion matrix of our final model after tunning.



It’s obvious on it how most of the examples predicted correctly.

e.g. 212 of the 550 test set is predicted as positive correctly and 233 predicted as negative correctly.

So I consider the model can be trusted and it is a significant enough to solve our problem with acceptable accuracy of 80.909%.

2. Reflection

To summarize the solution, we first convert the only input feature from text representation to array of numbers using bag of words so that we can benefit from the input feature, then we instantiated our two selected algorithms, trained them, tested them, tuning them and tested them again, make a comparison and found that Bernoulli Naive Bayes is more suitable.

The most interesting aspects aspects of this project is how to depends only on the text review to predict the sentiment. There are no any other features such as rating.

So the first challenge was to convert this feature to another representation.

I think there are tens of techniques we can apply on this feature to get the most of it and get better and better accuracy, but I am satisfied here with avoiding stop words, neglect word sensitivity and taking n-grams into consideration during vectorization.

But I think there is no any difficult aspects overall on the project.

Actually I expected the final model accuracy to be at least 85% but I relized I should have bigger dataset to achieve that.
So if our dataset is about 100000 examples, I’m sure this model will pass the 85% accuracy.

3. Imporvment

In applying bag of words, there are many techniques to do that, e.g. TfidfTransformer. We should make a comparison between them to find which one is the most suitable to our problem.

I also found RNN is very good for our problem and I tried to achieve it but unfortunately I faced many problems in implementing it, So I think if I could apply it, it will give very good accuracy in our problem.

i https://www.kaggle.com/marklvl/sentiment-labelled-sentences-data-set
ii http://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html#sklearn.metrics.accuracy_score
iii http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html#sklearn.naive_bayes.MultinomialNB
iv http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.BernoulliNB.html#sklearn.naive_bayes.BernoulliNB
v http://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html