**Medium Blog Posts Popularity prediction**

Mihir Deshpande

School of Information Studies

Syracuse University

Syracuse, NY 13244 U.S.A.

mdeshpan@syr.edu

**Abstract**

Medium is an online platform that publishes curated content over a range of topics. There are many contributors(authors) that post their blogs on medium. Users of the platform give “claps” if they like what they see. This is the mechanism implemented by medium to gauge the popularity of a post. This project deals with predicting the popularity of a post using two most popular text classification algorithms.

**1) Introduction**

The dataset is sourced from Kaggle, an online platform to find datasets and enter competitions for the data science community. The data is a collection of 337 full length blog posts on the topics of Data Science, Machine Learning and Artificial Intelligence. The motivation for popularity prediction for any platform that deals with blog posts or text content is that it can push those posts that it can classify having high popularity to the users of the platform enhancing the experience and driving engagement. The algorithms used to predict the popularity are Naïve-Bayes classifier and Support Vector Machines

**1.1) Naïve Bayes Classifier**

Naïve Bayes classifier is one of the popular machine learning classifiers. It is a generative classifier. It uses Bayes rule to estimate posterior probabilities for different classes using the prior probabilities and class conditional probabilities. It is ‘naïve’ because it assumes independence of attributes. The class with the highest posterior probability is predicted to be the class of the instance. It is very popular for text mining applications because itis robust to noise and irrelevant attributes. It is also very fast. There are two different implementations of the naïve Bayes algorithm, multinomial naïve Bayes and Bernoulli naïve Bayes. For this project, multinomial naïve Bayes was used.

**1.2) Support vector Machine**

It is another popular machine learning algorithm proven empirically to give very good performance over a range of applications. It is a discriminative model with the goal of finding a decision boundary that linearly separates the classes over the feature space. It uses some mathematical techniques called kernel tricks when the classes are not linearly separable. The goal of the kernel trick is to project the features in a higher dimensional space where the classes are linearly separable. There are different types of kernel functions that can be used for this purpose. For many text mining applications, linear kernel is most commonly used as it is possible to find a hyperplane in a very large dimensional space, which is usually the case with text classification problems. SVM is also robust to outliers.

**2) Data Preparation**

**2.1) Basic Data Cleaning**

It was noticed that the text had a lot of ‘\n’ and numbers. These were removed using regular expression with the python package ‘re’.

**2.2) Discretization**

The claps variable is a string data type with a presence of ‘K’ attributing thousand. That was appropriately treated and converted to a numeric data type. This continuous numeric variable must be discretized further and converted into categories to model the problem as a classification problem. There were a few choices for the intervals to be decided. One choice could have been random demarcation based on intuition. But the challenge with that is to make the categories balanced. Another clear strategy is to look at the data distribution and decide the categories based on that. This will make the categories balanced. There are two ways of discretizing based on distribution. One is to look at quartiles. This will give us 4 categories with 25% data in each category. With this approach we will get the categories ‘<150’, ‘150-2K’, ‘2K-7K’, ‘7K-above’. Another is to discretize based on the median. This will give us two categories with 50% in each category. With this approach, we get the categories ‘Under 2K’ and ‘Above 2K’. The performance of the classifiers on both these choices was tested.

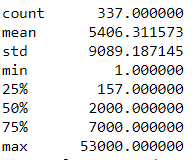


Figure 1

**2.3) Lemmatization**

The data was lemmatized using the wordnet lemmatizer from the nltk library. This ensures that words with same roots are treated as one feature as against considering all of them to be different features. This reduces the feature space without the loss of information. Additionally, part of speech tagging was used to correctly lemmatize the words as sometimes same words will have multiple lemmas based on meaning or context. The technique was borrowed from a blog [1].

**2.4) Vectorization**

The choice of vectorization really depends on the problem at hand. In this problem, since there are only 337 blog posts which are full length and hence rich in vocabulary, additional care must be taken to ensure that the feature space is not extremely high as it may lead to overfitting. Additionally, naïve Bayes classifier works best with raw term frequencies as against Tf-idf weighting or normalized term frequencies. Hence, Count Vectorizer from sklearn was chosen for the vectorization task. Other settings were, min\_df or minimum document frequency was set to 2% or roughly about 7, max\_df or maximum document frequency was set to 95% or roughly about 320, stop words were removed and all words were converted to lowercase. This ensures that only the words giving some information are retained as min\_df will take care of the rarely occurring words and max\_df will take care of high frequent words. Additionally, after the comparison of the two classifiers, Tf-idf vectorization was used only on the SVM classifier to see if an improvement in performance is noticed.

**2.5) Validation**

Hold out approach was used for validation. 70% of the data was used for training and 30% of the data was used for testing.

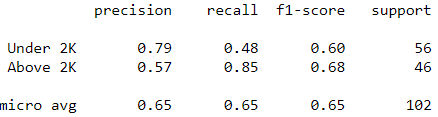
**3) Classification Models**

As mentioned above, naïve Bayes classifier and Support Vector Machine classifier was used for the classification task. Both these algorithms were used with the 2-class and 4-class discretization options and were compared.

**3.1) Naïve Bayes Classifier**

**3.1.1) 2-Class prediction**

When the algorithm is applied to a 2-class prediction problem, the random guess baseline is 50% as both the categories have 50% of the data. The Accuracy score achieved was 64.7% which is better than the random guess baseline by about 15 percentage points. The F1 score for the category ‘Under 2K’ was 60% and for the category ‘Above 2K’ was 68%.

 Figure 2

**3.1.2) 4-Class prediction**

When the algorithm is applied to a 4-class prediction problem, the random guess baseline is 25% as all the categories have 25% of the data. The Accuracy score achieved was 51.9% which is better than the random guess baseline by about 25 percentage points or almost twice as good as the random guess baseline. The F1 scores for the categories are as mentioned below in figure 3.

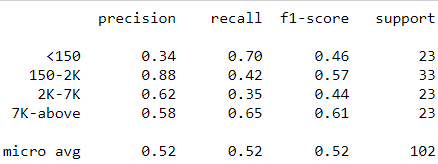


Figure 3

**3.2) Support Vector Machine**

The choice of the Cost parameter C which controls the amount of permissible misclassifications in order to generalize the model was decided based on empirical evidence. This value was set to 0.3. The range of C is from 0 to 1. When the value is set to 1, the algorithm does not allow any misclassification and hence does not generalize well as it tries to memorize the training data. The general direction of tuning this parameter is from high to low.

**3.2.1) 2-class prediction**

When the algorithm is applied to a 2-class prediction problem, the random guess baseline is 50% as both the categories have 50% of the data. The Accuracy score achieved was 77.5% which is better than the random guess baseline by about 27 percentage points. The F1 score for the category ‘Under 2K’ was 74% and for the category ‘Above 2K’ was 80%. These results are significantly better than the naïve Bayes 2 category model.

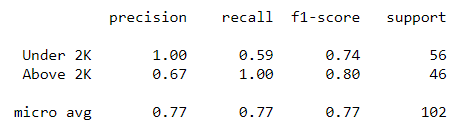


Figure 4

**3.3.2) 4-class prediction**

When the algorithm is applied to a 4-class prediction problem, the random guess baseline is 25% as all the categories have 25% of the data. The Accuracy score achieved was 61.76% which is better than the random guess baseline by about 37 percentage points or more than twice as good as the random guess baseline. The F1 scores for the categories are as mentioned below in figure 5. These results are much better than the naïve Bayes model with 4 categories.

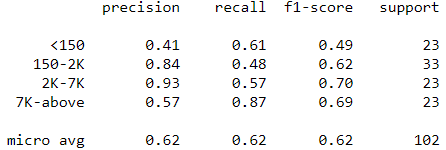


Figure 5

**3.3.3) 2-class prediction with Tf-idf weigthing**

When it was established that the Support Vector Machine algorithm was doing better than naïve Bayes algorithm on both the 2-class and 4-class problems, another svm model with tf-idf weighting was experimented with, to check if tf-idf weighting has any influence on the performace. This vectorization option was tested on the 2-class prediction problem. The accuracy score achieved was 79.41%. The F1 score for the category ‘Under 2K’ was 77% and for the category ‘Above 2K’ was 81%. This is better than the SVM 2-class model performance with count vectorizer.

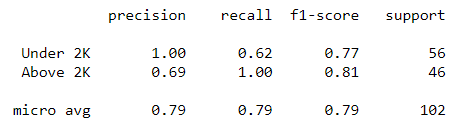
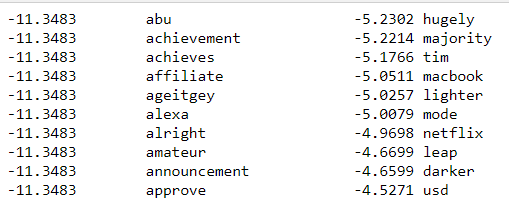
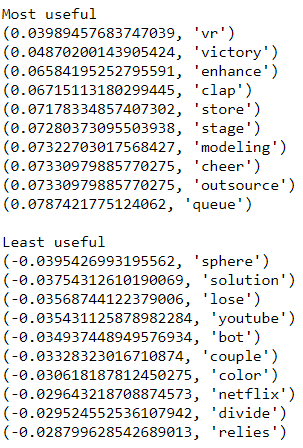


Figure 6

**3.4) Top words**

It sometimes helps to look at the top words that impact the predictions of classifiers just to have a sense of interpretation of the model. Below are the top words that influence the most and the least for naïve Bayes and SVM classifiers.



Naïve Bayes top words 

Support Vector Machines top words

**4) Interperations**

**4.1) Choice of categories**

Altough the choice of categories depends on the client requirements, just for a sense of simplication, 2-class prediction looks like a good choice. The intuition is that rather than having a stratified classification with 4 classes, if the platform gets a signal of whether the blog will cross the ‘2K’ mark of claps, it can easily engineer solutions to tweak their recommadation system to push these articles to the prospective consumers. Again, to reiterate, this choice will be based on the client needs.

**4.2) Precison and Recall**

Altough , F1 scores gives a general idea of model performance by combining the precison and recall, it sometimes helps to look at the individual precison and recall scores depending on the problem. In this problem, we are interested if the blog’s popularity in terms of claps will cross the ‘2K’ mark. We are more interested in recall, which measures the proportion of true positives to actual positives. We can be more lenient for precision which measures the proportion of true positives to predicted positives. This essentially means that we are more inclined to look whether all our actual postives whether predicted as such. An interesting trend is observed in Figure 4 and Figure 6 which are the classification reports for SVM 2-class models for Count Vectorization and Tf-idf vectorization options respectively. For these models, the recall value for the category ‘Above 2K’ was 1. This means that all the true ‘Above 2K’ values were predicted to be ‘Above 2K’. This is a significant result.

**5) Conclusion**

It was observed that the SVM algorithm performed better than the naïve Bayes alogorithm on both the 2-class and 4-class problem. 2-Class prediction model is intuitively easier to understand in terms of generating a popularity signal.SVM 2-class prediction problem performs better with Tf-idf vectorization. The model is successful in predicting the category ‘Above 2K’ with a very high recall score. These results, although significant, can be improved if more data is available for training purposes .This model can be used by the platform to drive their user engagement and provide a rich experience.

**Refrences:**

[1] <https://www.machinelearningplus.com/nlp/lemmatization-examples-python/>