**COMP551 MINI PROJECT 2**

**Mohammad Hamed Azizi (260812541), Saeed Shoaraye Nejati (260890049), Seasy Huang (260678549)**

**1. Abstract**

The Internet Movie Database, abbreviated IMDb, is an online collection of information related to popular films, television shows, and other media.In this project, we train a selection of machine learning models to determine the overall sentiment of a given dataset of IMDb movie reviews. The reviews were individually scraped from the website beforehand, and the provided test data was categorized as either ‘positive’ or ‘negative’. Following text preprocessing, we implement Bernoulli Naive Bayes from scratch with Laplace smoothing. We use seven separate models from the SciKit learn package and implement two feature extraction pipelines for text processing from modelling natural language programs. Adjusting for best hyperparameters, we compare each model’s final prediction accuracy on the given test set to determine which is best. Using k-fold cross validation, we show that our best performing model is Naive Bayes SVM. Following submission to Kaggle, this model has 89.97% accuracy on 30% of the validation set.

**2. Introduction**

In the last few decades, interest in sentiment analysis has been increasing. With the development of more advanced text-mining and analysis techniques, it is becoming easier and easier to gauge public opinion by monitoring social media platforms such as Facebook, Twitter, and Reddit as well as review sites like IMDb. Among these techniques, Bernoulli Naive Bayes has been shown to be a surprisingly successful model for classifying text sentiment, with adaptations of this model producing accuracies up to 88.80% on similar dataset [1].

The goal of this project is to determine which of our models can best identify the sentiment of reviews from a popular IMDb dataset. As the dataset consists of real world information, it is essential to perform text preprocessing on each sample before running our models on the dataset. Reviews with basic text preprocessing are used as inputs for the Naive Bayes implementation. Additional text features are constructed using tf\*idf and N-grams methods and inputted to the additional SciKit learn models.

We compare the performances of each SciKit learn model via k-fold cross validation and determine the best hyperparameter combinations for each model. Compared to the other models, Naive Bayes SVM performs the best with an average accuracy of 89.97%.

**3. Related Work**

Sentiment analysis is a extensively researched field. SVMs [2] and Naive Bayes [3] are perhaps the most well established algorithms for tackling this task. Typically, due to the high dimensionality of the data, standard Decision Trees do not fare well in text sentiment analysis.

Logistic Regression, while not traditionally used to classify text sentiment, has been shown more recently to have comparable results to SVMs and Naive Bayes in terms of predictive performance [4].

**4. Dataset and setup**

Because the dataset is provided as folders of files, we first extracted the raw text for data cleaning. Initial text preprocessing consisted of the following procedure executed in Python:

1. Removal of extra white space and special characters such as *<br />*.
2. Removal of non-alphabetical characters.
   1. Specific removal of tags, links, and numbers.
3. Tokenization of the review.
4. Lemmatization of the reviews (using *NLTK WordNetLemmatizer*).
5. Rejoining the individual tokens (for saving purposes).

The above procedure was applied to training data and used as input to our Naive Bayes implementation from scratch. The training data was partitioned with an 80-20 split and used to determine accuracy. The dataset was additionally processed before being run on SciKit-learn based models:

1. Vectorization (using *CountVectorizer* from *sklearn*) with max features of 5000.
   1. Transformation of text to lowercase.
   2. Implementation of N-grams and TF\*IDF (from *sklearn*)
   3. Removal of English stopwords.
2. Transformation to numpy matrix form.
3. 83-17 split of train/test data.

**5. Proposed approach :**

We propose the implementation of Bag-of-Words and TF\*IDF and n-grams for feature design. The methods discussed in this section are implemented in *preprocessing.py*. We set the range of n-grams to (1, 2) in *countVectorzier*. Document frequency is set to the range of *min(cut-off) = 5, max = 0.7*. *countVectorizer* is assumed as input of *TfidfTransformer*, and we specify the regularization type as L2. As number of features increase from 1500 to 20000, accuracy increases at the expense of significant increases in computation time.

We implement the following SciKit models: Bernoulli Naive Bayes [5], NaiveBayesSVM [6], LogisticRegression [5], RigidClassifier [5], KNN [5], Decision Tree [5], and RandomForest [5]. We use K-fold Cross Validation with *cv=10* for all the models.

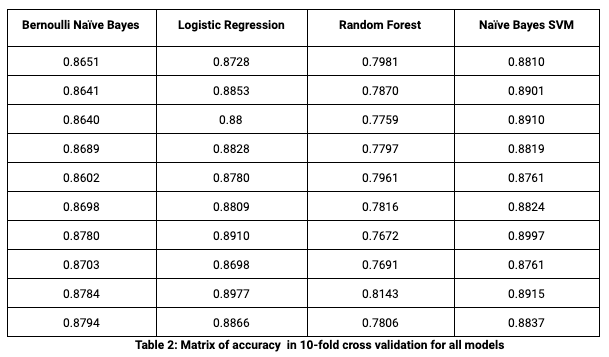
**6. Results** :

For our Naive Bayes implementation from scratch, we determined the final accuracy to be 82.8%. We determine this accuracy by partitioning the training set with a 80-20 split; due to the relatively long runtime (5 hours) of our implementation, it was difficult to test with more than 5000 training set-derived test points. This result is in agreement with the suggested accuracy of standard Naive Bayes mentioned by the professor in lecture. For our SciKit-learn based models, accuracy was determined with a 83-17 split.

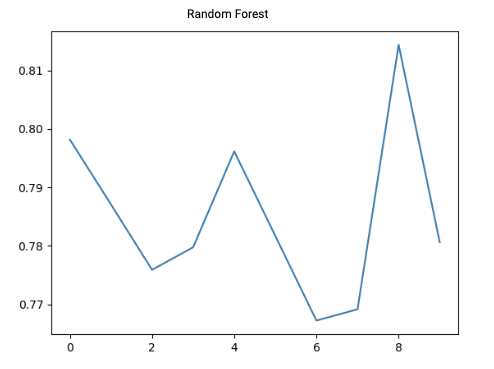
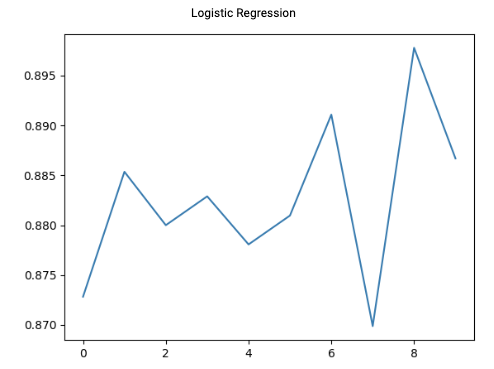
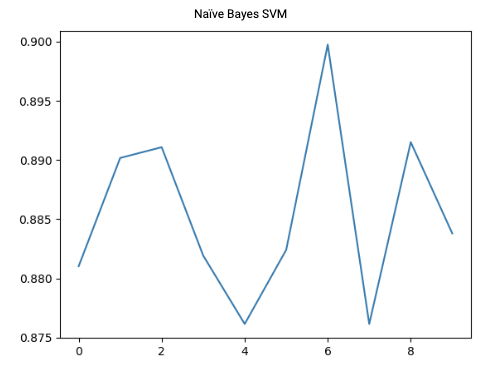
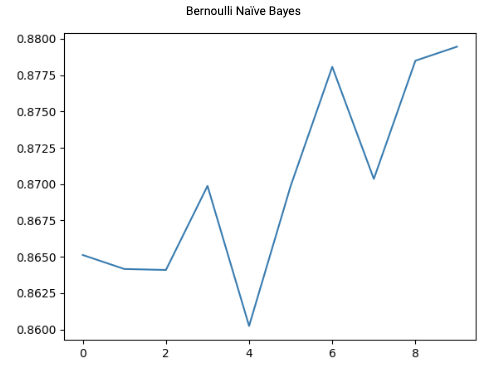
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Models** | **Bernoulli Naïve Bayes** | **Logistic Regression** | **Random Forest** | **Naïve Bayes SVM** |
| **Accuracy Score** | 0.8611 | 0.8778 | 0.7863 | 0.8722 |
| **CV Mean** | 0.8699 | 0.8825 | 0.7850 | 0.8853 |
| **CV STD** | 0.0064 | 0.0078 | 0.0136 | 0.0071 |

**Table 1: Comparison of Accuracy for Different Models**

For each model, we used k-fold cross validation as shown in the table below, and we set hyperparameters for logistic regression using a pipeline grid search on binary weighting and based TD\*IDF weighting. After k-fold cross validation, the model with the best accuracy score of 89.97% was determined to be SVM.



**Table 2: Matrix of accuracy in 10-fold cross validation for all models**



**Graph 1: Plot of accuracy vs fold cross validation for all models**

|  |  |  |  |
| --- | --- | --- | --- |
| Bernoulli Naïve Bayes | Logistic Regression | Random Forest | Naïve Bayes SVM |
|  |  |  |  |

**Graph 2: Confusion Matrix for all models**

|  |  |
| --- | --- |
| Parameters chosen for Grid search in pipeline | Parameters selected by Grid Search |
|  |  |

**Table 3: Pipeline parameters**

**7. Discussion and Conclusion**

In this paper, we investigated different combinations of machine learning models, features, and preprocessing levels for sentiment analysis of IMDb reviews. We found that preprocessing the data consistently increased the accuracy of some models, which was expected. Interestingly however, we found that in the Naive Bayes implementation, using preprocessing of the dataset gave the highest performance of 83%. For models using text features as input, we experimented with the use of different N-grams combination with feature construction techniques, and we found that the best features are bigram and tf\*idf based on parameters ‘C’ , ‘kernel’, and ‘gamma’. Our SVM combined with Naive Bayes model produced the highest accuracy score of 89.97%.

**8. Statement of Contributions**

All members contributed equally for implementation and report.

**9. References**

1. Narayanan, Vivek, Ishan Arora, and Arjun Bhatia. "Fast and accurate sentiment classification using an enhanced Naive Bayes model." *International Conference on Intelligent Data Engineering and Automated Learning*. Springer, Berlin, Heidelberg, 2013.
2. Joachims, Thorsten. "Text categorization with support vector machines: Learning with many relevant features." *European conference on machine learning*. Springer, Berlin, Heidelberg, 1998.
3. McCallum, Andrew, and Kamal Nigam. "A comparison of event models for naive bayes text classification." *AAAI-98 workshop on learning for text categorization*. Vol. 752. No. 1. 1998.
4. Komarek, Paul, and Andrew W. Moore. "Fast Robust Logistic Regression for Large Sparse Datasets with Binary Outputs." *AISTATS*. 2003.
5. Pedregosa, Fabian, et al. "Scikit-learn: Machine learning in Python." *Journal of machine learning research* 12.Oct (2011): 2825-2830.
6. Wang, Sida, and Christopher D. Manning. "Baselines and bigrams: Simple, good sentiment and topic classification." *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*. Association for Computational Linguistics, 2012.