FINAL PROJECT CSCI-B659

Akshay Kamath (akkamath@iu.edu)
Erika Dsouza (ehdsouza@iu.edu)

INTRODUCTION

In this report, we discuss our analysis of various machine learning algorithms on the amazon review dataset. Our primary goal is to predict the sentiment of a review on books that are available on the website. All the machine learning algorithms used in this project were implemented from scratch in python and located in github: https://github.iu.edu/akkamath/aml-final-project.

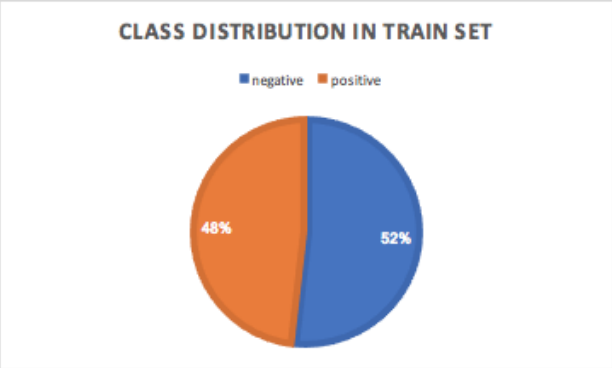
Algorithms Implemented

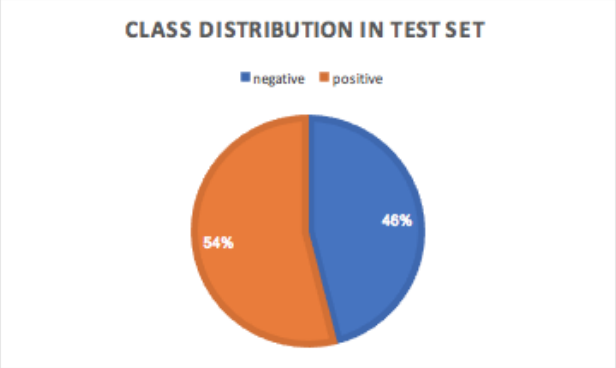
Following algorithms were implemented from scratch:

1. Naive Bayes with laplace smoothing
2. Logistic Regression
3. K Nearest Neighbours
4. Bagging with K- Nearest Neighbours
5. Soft Margin Support Vector Machines using CvxOpt QP solver
6. Linear Kernel, Polynomial Kernel and Gaussian Kernel for SVM
7. Term Frequency Inverse document frequency
8. K-Fold Cross validation on training data for tuning algorithm parameters
9. Feature Creation - word tokenization, writing to files, reading from files.

DATA

- For this task, we have selected the amazon review dataset for books from the Multi Domain Sentiment Dataset.[1]
- The dataset consisted of an xml file with book name, review text and review ratings as sub tags.
- The review ratings range from values 1 to 5 indicating the customer satisfaction for the book.
- The review text consists of review text provided by the same user that can be of length ranging from 1 - 1000 words on average.
- For our analysis, we have randomly sampled 1000 reviews from the xml file and put them into the csv files for further analysis.
- For the class labels, we have categorised review rating with below 3 being negative and all above 2 to be positive.
- Training set contains 699 reviews with corresponding sentiment and test set contains 301 reviews with corresponding sentiment.
- Class distribution for train and test data





FEATURES

Feature Creation

- For feature creation, we have used the bag of words model.
- We tokenize all the reviews and select all the unique words as features.
- We then remove all the stop words from the features.
- For balancing the impact of more frequent words in the dataset, which we believe do not provide a high impact on the sentiment, we convert the words to a numeric vector using our implementation of TF-IDF.
- We mark positive sentiment as +1 and negative sentiment as -1
- We then store all these vectors in a csv file for both, train set and test set.

EXPERIMENTS

Classifiers

We use the following classifiers for our experiments:

1. Naive Bayes
2. Support Vector Machine
3. Logistic Regression
4. K-Nearest Neighbors
5. Bagging with K-Nearest Neighbors

Approach:

Part 1 - Tuning

We tune different parameters for each classifier using K-Fold Cross Validation and identify the best parameters.

1. Try SVM with 5 different values of C (Soft Margin) for each kernel implementation and select the best one. C= 10, 20, 50, 100, 200
2. Try KNN with 5 different values of K and select the best one. K = 5,7,10,51,121
3. Try bagging KNN classifiers of K = 3 AND and five different number of bags. B = 2,3
4. Try logistic regression with 5 different values of step counts and select the best one. Alpha = 0.001, 0.1, 0.3, 1, 10

5. We test with default settings of Naive Bayes.

Part 2 - Testing

1. Using the best parameters obtained from the tuning set, we train the classifiers on our training set and predict the result on the test set.
2. We present our results in terms of accuracy, precision and AU-ROC

RESULTS:

Tuning

Tuning SVMs

Tuning Linear SVM Accuracies:

C	10	20	50	100	200
Accuracy(%)	75.67	77.39	76.39	76.53	76.82

Best Parameter: C = 20

Tuning Gaussian SVM Accuracies:

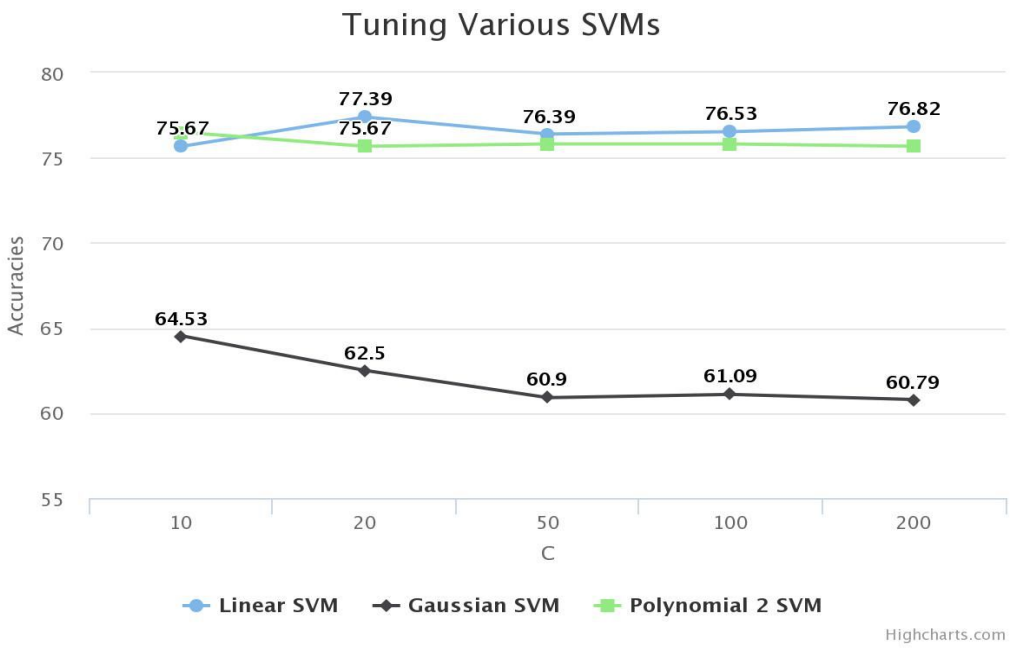
C	10	20	50	100	200
Accuracy(%)	64.53	62.5	60.9	61.09	60.79

Best Parameter: C = 10, Sigma = 1

Tuning Polynomial , P= 2, SVM Accuracies:

C	10	20	50	100	200
Accuracy(%)	76.5	75.67	75.81	75.81	75.67

Best Parameter: C = 10, P = 2



Tuning Logistic Regression:

Alpha	0.001	0.1	0.3	1	10
Accuracy(%)	48.2	482	48.2	48.2	48.2

Best Parameter: Choose alpha as 0.3
Note: Logistic regression using sklearn also gives the same values

Tuning K-Nearest Neighbours:

K	5	7	10	51	121
Accuracy(%)	52.21	51.78	51.78	51.78	51.78

Best Parameter: K=5

Tuning Bagged K-Nearest Neighbours:

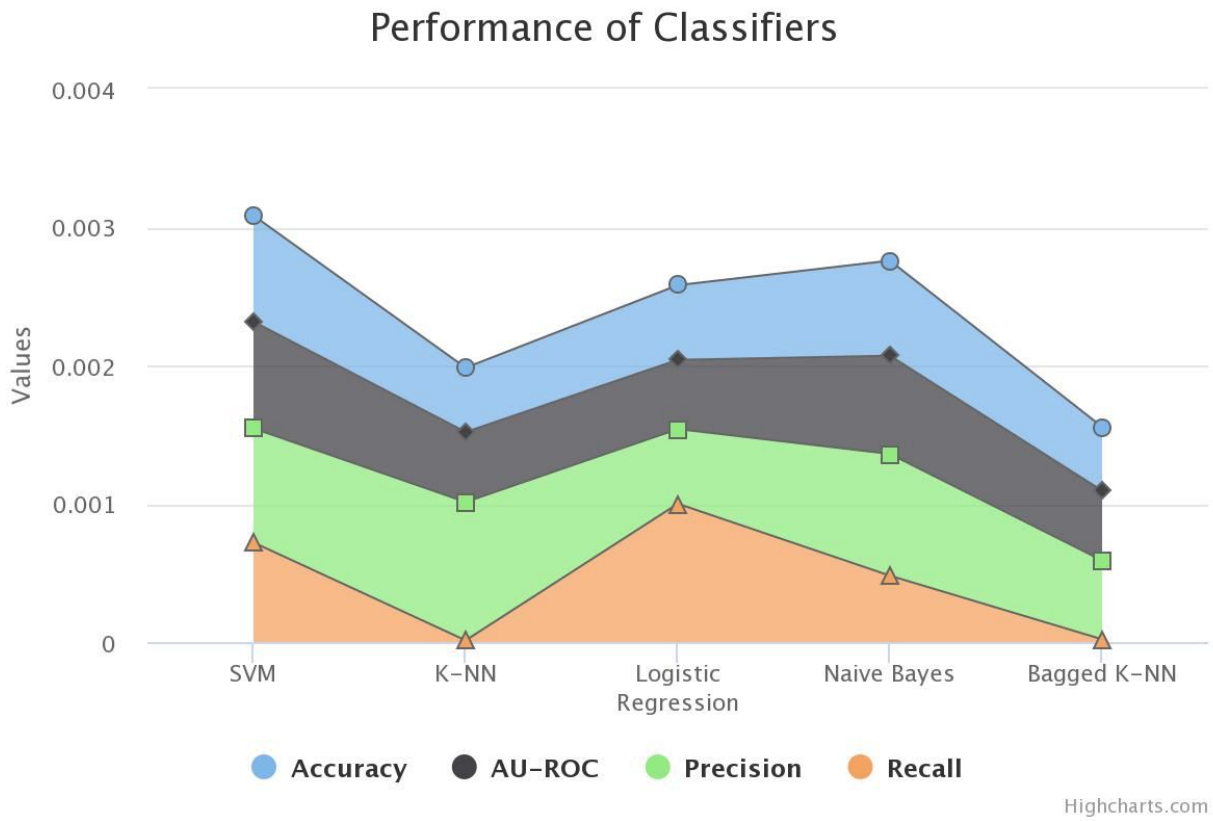
Keeping k =1 at all times

Bags	2	3
Accuracy(%)	88	88.42

Best Parameter: 3

Testing

Classifier	SVM	K-NN	Bagged K-NN	Logistic Regression	Naive Bayes
Accuracy(%)	76.74	46.51	46.17	54	68.77
AU-ROC	0.771383	0.5061	0.504	0.5	0.71
Precision	0.825	1.0	0.57	0.54	0.88
Recall	0.72	0.0122	0.02	1.0	0.48



Confusion Matrices

Naive bayes Confusion Matrix:

Total = 301	Predicted, 1	Predicted, -1	Total
Actual, 1	(TP) 79	(FN) 84	163

Actual, -1	(FP) 10	(TN) 128	138

SVM Confusion Matrix:

Total = 301	**Predicted, 1**	**Predicted, -1**	**Total**
Actual, 1	(TP) 118	(FN) 45	163
Actual, -1	(FP) 25	(TN) 113	138

KNN Confusion Matrix:

Total	**Predicted, 1**	**Predicted, -1**	**Total**
Actual, 1	(TP) 2	(FN) 161	163
Actual, -1	(FP) 0	(TN) 138	138

LR Confusion Matrix:

Total	**Predicted, 1**	**Predicted, -1**	**Total**
Actual, 1	(TP) 163	(FN) 0	163
Actual, -1	(FP) 138	(TN) 0	138

KNN Bagging Confusion Matrix:

Total	**Predicted, 1**	**Predicted, -1**	**Total**
Actual, 1	(TP) 4	(FN) 159	163
Actual, -1	(FP) 3	(TN) 135	138

AU-ROC

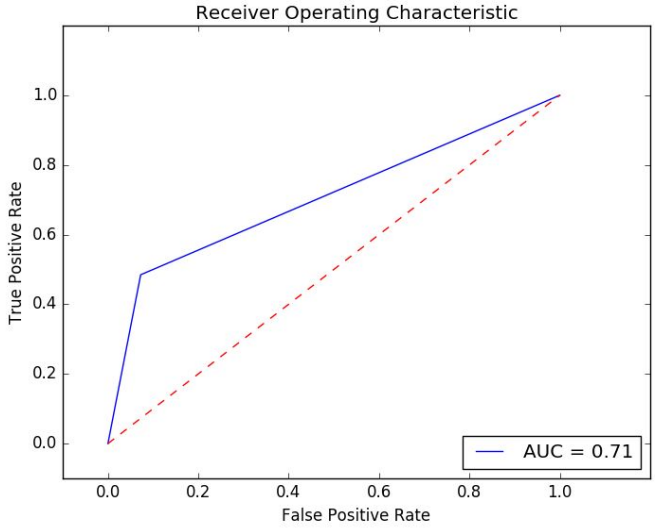


Fig 1. AU-ROC Naive Bayes

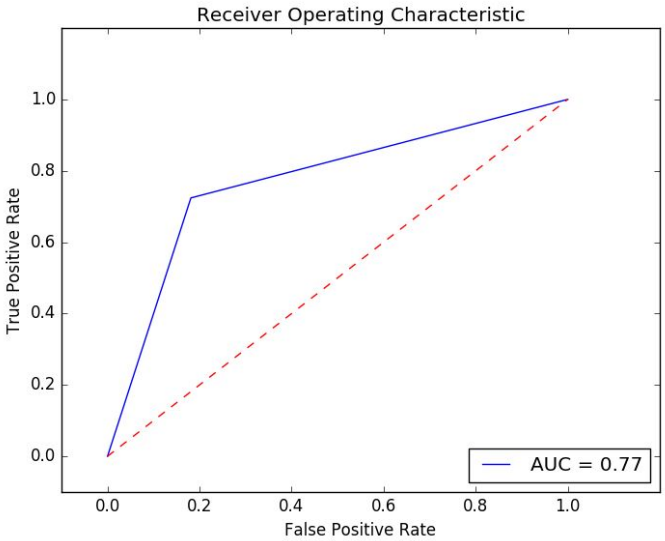


Fig 2. Au-ROC Linear SVM

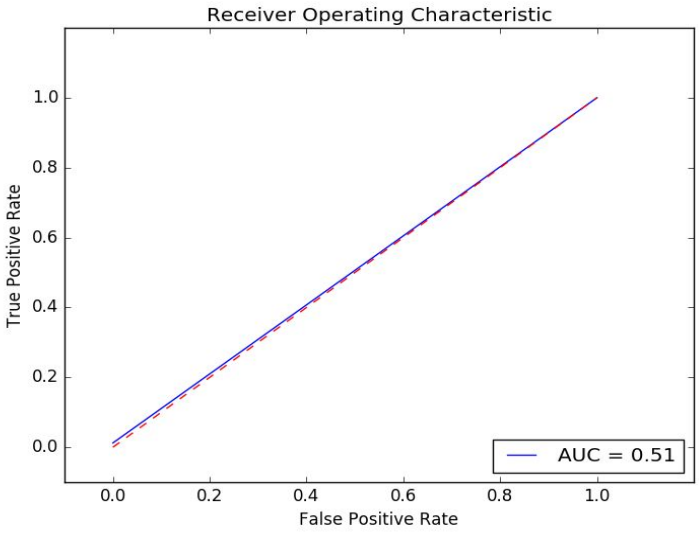


Fig 3. Au-ROC KNN

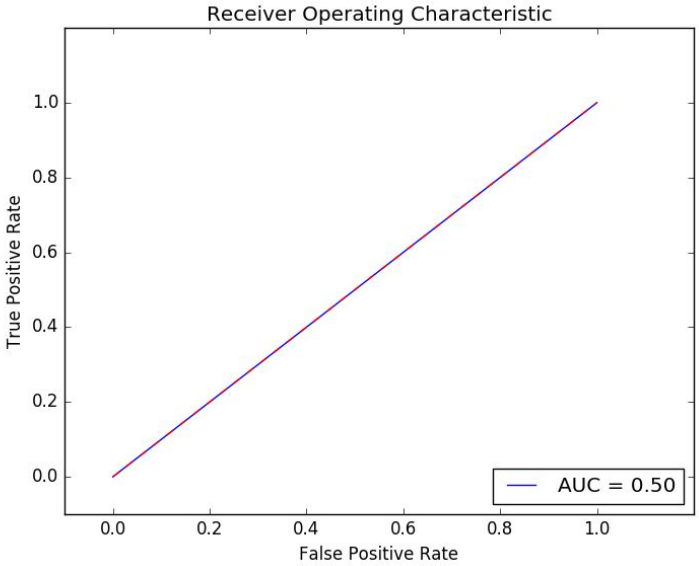


Fig 4. Au-ROC Logistic Regression

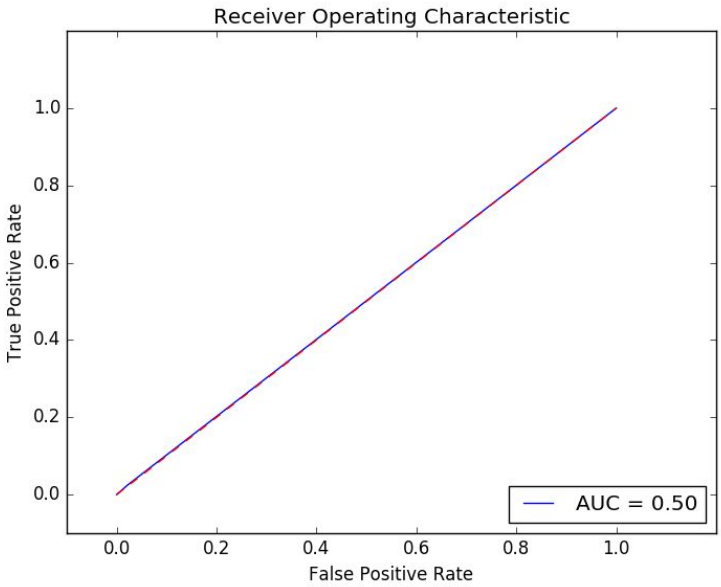


Fig 5. Au-ROC Bagged KNN

DISCUSSION

Our initial experiment involved, tuning the 5 classifiers that we used. Since this is a balanced dataset (+ve: 48%, -ve : 52%), we have used accuracy as our primary metric for analysis. We have also computed the corresponding AUC, Precision, recalls.

During tuning and testing we made the following observations,

- For **SVM**, the linear kernel performed better on almost all values, this could be due to the fact that the feature vector is large and has sparse values and indicative that the data is indeed linearly separable and does not require any implicit projection into a higher dimension.
- For **KNN** and **Bagged KNN**, the classifiers could not make better predictions even on tuning set, the accuracy remained almost the same through different values of k. This is because of high dimensionality and the sparse feature vector. Due to high dimensional data, almost all data points are uniformly distant from any point. Thus we could say that KNN breaks down in this high dimensional, sparse data[11]. Also, due to this, even the variance suffers at lower values of k. Hence bagging does not perform any better.
- Another side effect of high dimensional data was the speed of execution, both KNN and Bagging take a long time to compute the results.
- **Logistic regression** could not learn a better model and gave predictions as per majority class as the logistic regression classification was based on log-likelihood
- **Naive Bayes** gave a pretty good accuracy as it used the representation of bag of words well and was robust to irrelevant features. It seemed like a good dependable baseline for sentimental analysis using text classification

- Overall, we conclude that Naive Bayes and Svm are the best classifiers for this text classification task of sentiment, this can also be seen in the test accuracies and also in the ROC curve since the rise of TPR is faster than FPR.
- On the other hand, KNN, Logistic regression and Bagged KNN did not perform so well due to failing in high dimensional data and learning a clean separation between classes respectively. Consequently, their accuracies and corresponding AUC are very less and are indicative of poor classifiers.

CONCLUSION

We performed comprehensive experiments and analysis on the amazon book review dataset using five algorithms that were implemented by us from scratch. Due to balanced data, we chose accuracy as our choice of metric for comparison of classifiers. We noted that SVM and Naive Bayes were the best performers whereas logistic regression and KNN performed poorly. We also note that, the failure of these classifiers was mainly due to the high dimensionality and sparseness of the data. This leaves scope for some further improvement in performing feature reduction viz. Usage of principal component analysis or Chi-squared test. Overall, our results concur with the theory and also prove to a certain extent that SVM and Naive Bayes are better suited for text classification task such as sentiment analysis.

REFERENCES

[1] John Blitzer, Mark Dredze, Fernando Pereira. Biographies, Bollywood, Boom-boxes and Blenders: Domain Adaptation for Sentiment Classification

[2] **https://www.cs.jhu.edu/~mdredze/datasets/sentiment/index2.html**

[3] Scikit Learn

[4] Numpy

[5] Scipy

[6] CvxOpt

[7] stopwords

[8] http://machinelearningmastery.com/logistic-regression-tutorial-for-machine-learning/

[9] https://www.youtube.com/watch?v=EGKeC2S44Rs&t=780s

[10] https://pythonprogramming.net/soft-margin-kernel-cvxopt-svm-machine-learning-tutorial/

[11] Pedro Domingos: On the Surprising Behavior of Distance Metrics in High Dimensional Space