COMP 551 - Miniproject 2

Brendon McGuinness - Shubhika Ranjan - Pauline Riviere
Group 63

February 28, 2021

Abstract

The project required us to work on two benchmark text datasets, 20newsgroups and IMDB, to investigate the performance of two machine learning models. After data cleaning and exploration, training datasets were used to train the Naive Bayes and Logistic Regression models and tune hyperparameters using 5-fold cross-validation. After selection of the best hyperparameters, the models were trained again on the entire training dataset and accuracy was evaluated on test datasets. Performance was also evaluated according to the size of the training dataset. On top of that, a linear regression model was used to predict ratings on the IMDB dataset. We found that Logistic Regression performed better than Naive Bayes on the IMDB dataset whereas Naive Bayes did better than Logistic Regression on the 20newsgroups dataset. Text embedding methods and hyperparameters settings had a great influence on the performance of the models.

1 Introduction

The purpose of this project was to understand the implementation of Naive Bayes and k-fold cross-validation from scratch for the two provided datasets of text (20newsgroups dataset and IMDB movie reviews). The 20newsgroups dataset is a collection of around 18,000 newsgroups posts on 20 topics (for example, motorcycles, baseball, religion, etc.). The objective of the classification is to be able to predict the topic of the post. Using a Naive Bayes classifier, previous authors were able to achieve a 89% performance on this dataset [1]. The IMDB movie reviews dataset is a collection of 50,000 movie reviews with their rating and their sentiment: rating 1 to 4 corresponds to a negative review and 7 to 10 to a positive review. The objective of the classification is to predict the sentiment of the review based on the review text as done by [2]. Previous authors achieve an accuracy of 88% with a Neaive Bayes classifier on this dataset [3].
Overall, the best model to predict the topic of a post for the 20newsgroups dataset was Multinomial Naive Bayes with Laplace smoothing and log-sum-exp trick with a 86% accuracy. The best performance achieved regarding sentiment analysis on the IMDB dataset was 89% with Logistic Regression. With a reduced size of training dataset, Naive Bayes model performed better on the two datasets [5] as compared to Logistic regression.

2 Datasets

The two data-sets analyzed in this project, obtained from scikitlearn library and Standford University are 20newsgroups dataset and IMDB movie reviews. The 20newsgroups dataset contains around 18,000 newsgroups posts labelled with their topic. There are 20 topics related to sports, politics or computers for example as shown in Appendix Fig A.1. The dataset is divided into a train set of 11314 files and a test test of 7532 files. As mentioned in the repository, the IMDB movie reviews dataset contains 50,000 movie reviews along with their associated sentiment label, 25,000 positive and 25,000 negative as shown in Appendix Fig A.2, uniformly separated into training and test sets. Reviews rated 1 to 4 are labelled as negative and 7 to 10 as positive. To facilitate loading on Colab, all the files were gathered in a csv file with 4 columns: review, dataset, label, rating.

The review part was cleaned at this stage (converted to lower case, removing punctuation and replacing special characters and html tags by spaces).

Next, the text data needed to be converted to numerical features to train the models. Before converting, we removed the non-informative words, which are commonly referred as stopwords, such as 'in', 'of', 'at', 'a', 'the' from our textual datasets. We tried different conversion methods:

- using different methods to denote the absence/presence of a word:
  - using the absence or presence of the word as a feature (CountVectorizer function from ScikitLearn, with binary=True)
  - using the word count as a feature - how many times the word appears in the text (CountVectorizer with binary=False)
  - using the term frequency-inverse document frequency (tf-idf), i.e. the number of occurrences of each word in a document is divided by the total number of documents in the corpus that the world appears in (TfIdfVectorizer function from ScikitLearn)
- counting single words or two word sequences (ngrams parameter included in CountVectorizer and TfIdfVectorizer)

3 Results

3.1 Model selection

3.1.1 Features engineering

We used cross-validation to evaluate the performance of the models on a validation set using the 4 different types of features: 1.binary counts, 2.binary counts using bigrams, 3.word counts using bigrams, 4.tf-idf using bigrams. We tried to use Gaussian Naive Bayes with tf-idf features because these are continuous data. However, the computation of the likelihood transformed the sparse matrix into a dense one (values were not 0 anymore) leading to a crash in memory. Results are presented in Table 1 below for the two datasets. The features preprocessing for each dataset and model giving the best performance was used for the rest of the experiment.

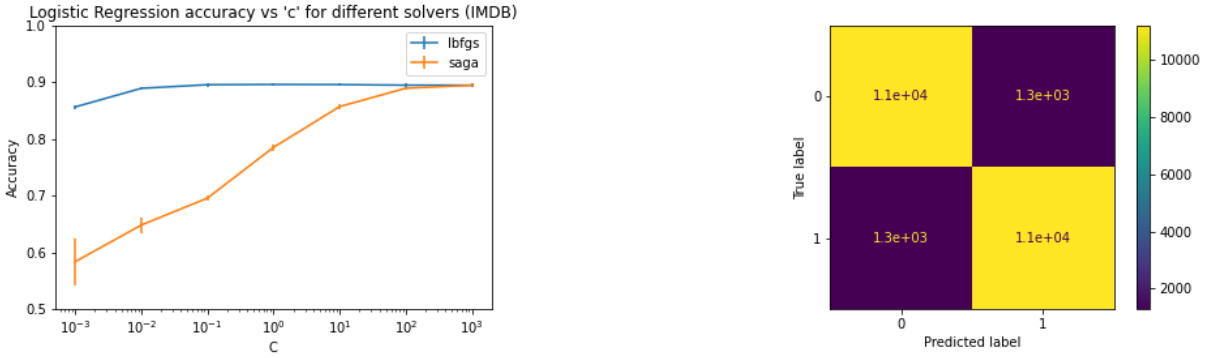
Model	Features	Performance IMBD	Performance 20NG
Naive Bayes			
	Binary counts	65%	78.71%
	Binary counts using bigrams	79.3%	83%
	Word counts	85.56%	86.1%
	Tf-idf*	na	na
Logistic regression			
	Binary counts	89.96%	59.23%
	Binary counts using bigrams	89.98%	57.18%
	Word counts	89.62 %	58.17%
	Tf-idf	88.46 %	68.41%
Linear regression			
	Binary counts	4.45	na
	Binary counts using bigrams	2.60	na
	Word counts	2.85	na
	Tf-idf	2.28	na

Table 1: Compared performance of the models according to preprocessing for the two datasets (performance expressed as accuracy in % for Naive Bayes and Logistic Regression and Root mean square error for Linear regression). NB Tf-idf*: The large sparse matrix crashed colab while loading in our gaussian NB model as the code could not return a sparse matrix so the values won't be 0.

3.1.2 Tuning of hyper-parameters with cross-validation

IMDB dataset - Logistic regression

We implemented and used 5-fold cross-validation. For hyper-parameter tuning of the logistic classifier, we specifically looked at the hyper-parameter $C = 1/\lambda$ which is the inverse of the regularization strength λ. In an attempt to reduce potential overfitting in our model, We varied C over a range of six orders of magnitude, with small values of C applying a larger penalty to large parameter value parameters while large values of C applies smaller to essentially no penalty for $C >> 1$. Additionally, we tried this for two different solvers LBFGS and SAGA (saga was given normalized data). The optimal solver and C value were LBFGS and 1 respectively which gave and average accuracy of 89.55% during cross-validation and an accuracy of 89.62% on the testing set. This similarity between performance on K-folded training data and testing data suggests that our model is not overfit.



(a) (b)

Figure 1: IMDB data. (a) 5-fold cross validation demonstrating the accuracy of logistic regression classifier over a range of C values with two different solvers. (b) Confusion Matrix of IMDB data classifier when run on testing data.

IMDB dataset - Linear regression

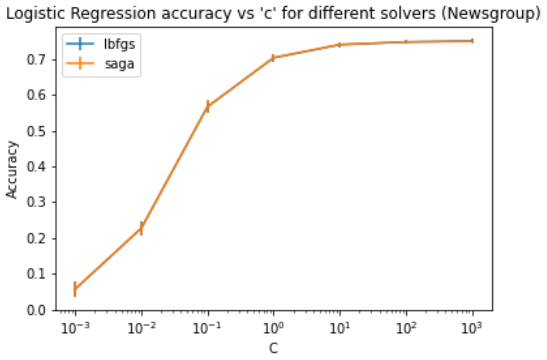
We used a linear regression model to predict the ratings of IMDB reviews with tf-idf features using Scikit Learn functions. We tested several algorithms:

- ordinary least squares (LinearRegression function)
- L1 regularization (Lasso function): we used this one given the high number of features in the dataset, we wanted to select the best ones via L1 regularization
- Ridge regularization (Ridge function)
- an algorithm mixing L1 and L2 regularization (ElasticNet function)

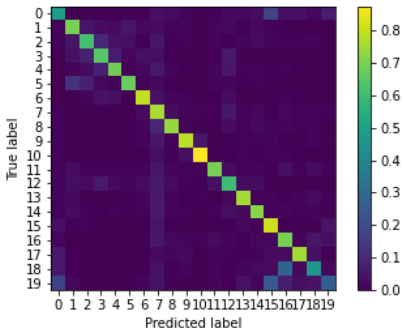
Regularization was used to prevent overfitting the training data. For the regularization algorithms, data was scaled using the MaxAbsScaler method of Scikit Learn, dedicated for sparse matrices. For each model, using cross-validation, we tested several values of the regularization term. For the ElasticNet model, we tested also several values of L1 versus L2-regularization ratio. For Ridge regression, we tested two solvers: stochastic average gradient descent ('saga') and least-square ('lsqr'). The root mean square error in the different settings are displayed in Appendix Table A.1.The RMSE according to the solver and regularization term is displayed in Appendix Figure A.3. Overall, the lowest RMSE was obtained using ordinary least square algorithm. The second best result was obtained with Ridge regression, using saga solver and regularization = 10 as hyperparameters. These two models were selected to report performance on the test dataset. The Ridge model was specifically chosen because we wanted to avoid overfitting related to the use of ordinary least square algorithm.

Newsgroups dataset

Our 5-fold cross-validation on our Newsgroup dataset gave us the optimal hyperparameters $C = 1000$ and the LBFGS solver for the newsgroup dataset. The variation in performance between the LBFGS and SAGA solvers were in the margin of error. The optimal hyperparameters gave the solver and an average accuracy of 75.00% during cross-validation and an average of 68.74% on the test set.



(a) Hyperparameter tuning with logistic regression on newsgroups dataset



(b) Confusion matrix of logistic regression on newsgroups dataset

3.2 Compared accuracy of the models on the two datasets

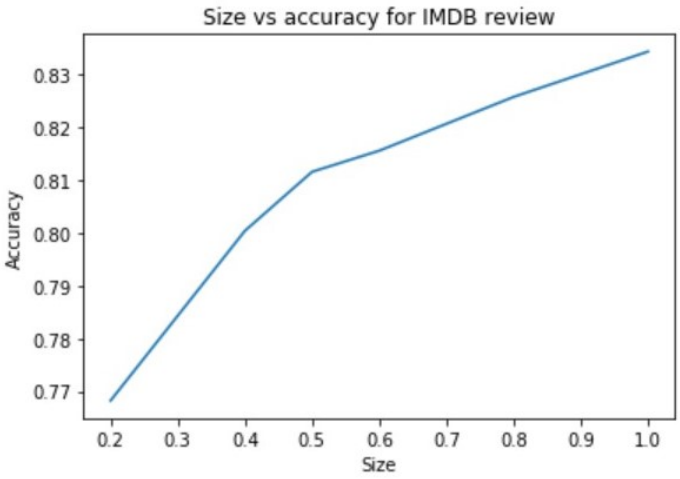
As displayed in Table 2, the Naive Bayes model performed better on the 20newsgroups dataset than the Logistic Regression model whereas it was the opposite for the IMDB dataset.

Model	Performance achieved	Recall (%)	Precision (%)
IMDB dataset			
Naive Bayes algorithm	85.56%	86%	86%
Logistic Regression	89.62%	89.49%	89.71%
Linear Regression			
Ordinary Least Square	2.08	na	na
Ridge regularization	2.62	na	na
Newsgroups dataset			
Naive Bayes algorithm	86.1%	83%	84%
Logistic regression	68.41%	67.15%	68.61%

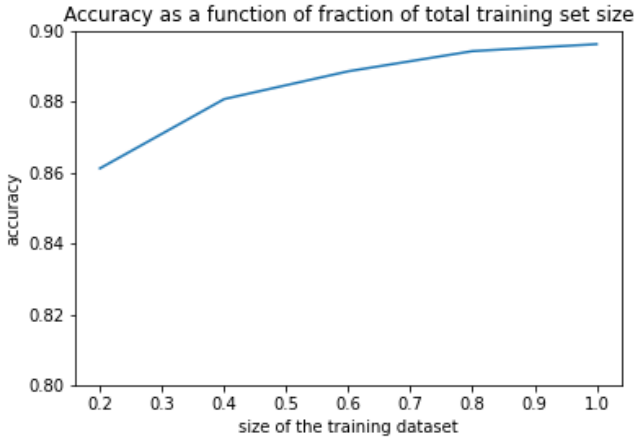
Table 2: Compared performance of the algorithms on the two datasets (performance expressed as accuracy in "%" for Naive Bayes and Logistic Regression and Root mean square error for Linear regression)

3.3 Accuracy of the models as a function of the size of the dataset

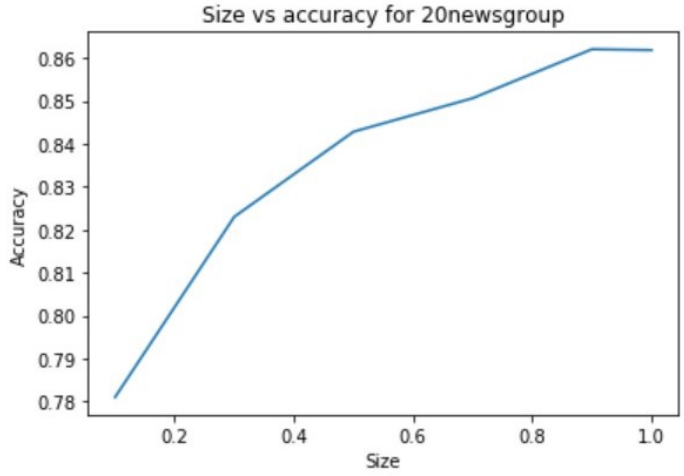
As shown in below figure 3, with a decreasing training size the accuracy for our two models falls considerably. The training size for this comparison ranges from 20% upto 100% of our dataset. In case of Naive Bayes model the original size of training dataset was derived based on a threshold of most frequent word which reduced the number of features in order to make our model efficient. The 2 experiments with threshold of 2000 and 5000 frequent words gave almost similar accuracy, indicating that by selecting the first n most frequent word, we our reducing the complexity of our model without affecting its efficiency. But by further decreasing the size, we noticed slight fall in accuracy.



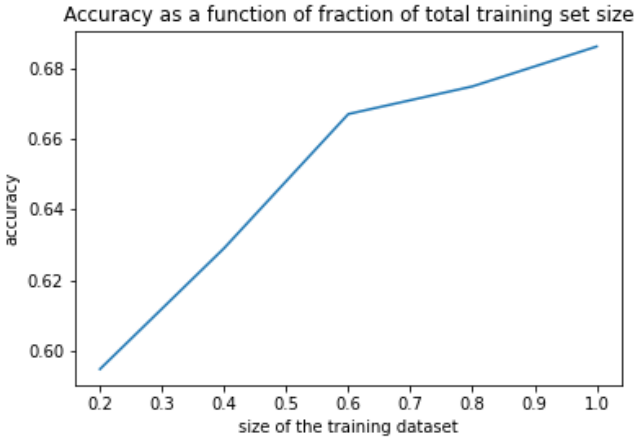
(a) Naive Bayes classifier on IMDB dataset



(b) Logistic Regression classifier on IMDB dataset



(c) Naive Bayes classifier on Newsgroup dataset



(d) Logistic Regression Classifier on Newsgroup dataset

Figure 3: Performance of classifiers in terms of their accuracy plotted against fraction of total training set used to fit model.

4 Discussion and Conclusion

Overall, for the newsgroups dataset, the Naive Bayes model performed better than Logistic Regression and for the IMDB, the Logistic Regression did better than Naive Bayes. The performance of the predictions for the two datasets was good: we achieved almost 90% of accuracy with Logistic Regresion on IMDB dataset and around 85% with Naive Bayes on the newsgroups dataset. Moreover, the linear regression model was able to predict the rating for the IMDB with an error of 2. Given that ratings from 1 to 4 are considered as negative and 7 to 10 as positive, linear regression would be a good surrogate to logistic regression in that case. The performance of linear and logistic regression increased almost linearly with the size of the training set. For Naive Bayes, the size of the training dataset seems to make less difference. This was also the case in the paper by Poyraz et al.: they found that from a 50% size of the 20newsgroups dataset, the accuracy was the same than with the entire training dataset [4].

The naive bayes algorithm works better with smaller dataset [5]. The problem we faced were majorly due to the sparse matrix constructed by vectorization of our 2 text datas. For example in 20 newsgroup the training data is of size 11314X101631 with 20 classes. And the large feature set led to more computational time and complexity. The same dataset when trained using the sklearn.multinomialNB, gave good prediction. But since the matrix in our model needed to access all the elements, it led to higher computational complexity displaying warning "the sparse matrix is ambiguous". With this we made an assumption of using the most frequently used

words to construct our vocabulary, which reduced the features by using the first 2000 to 5000 most frequent words based on the word count. This led to smaller and compressed sparse matrix which made the computation and efficiency much better of our self-implemented naive bayes model.

For the logistic regression classifier, 5-fold cross-validation deemed the hyperparameter $C = 1000$ to be the most optimal for the Newsgroup dataset with an average accuracy of 75%. This means that there was almost no regularization occuring in the model. This seemed to lead to our model overfitting the training data due to the fact that the accuracy on the testing data was 68%, quite a bit lower than that for the training data.

Regarding performance of the linear regression, the test error was lower than train error and regularization didn't help to increase performance compare to ordinary least square suggesting a possible underfitting/high bias. This could be related to the non-linear distribution of the features and transformation of the data using non-linear bases could have helped. However, given the already huge number of features, this would have lead to difficulties with the memory.

We performed various experiments to tune our naive bayes model to reach to a better conclusion. For both the dataset, if we did not consider Laplace smoothening and log-sum-exp trick, it gave an accuracy of 0% condering the sparse matrix construction for large datasets. And when only considering the smoothening for our data, it led to number underflow, as the count for most of the features will be far less as compared to the total number of instances. The probability being product of these small numbers, would be even smaller. Hence, our predictions in the above mentioned tables are from incorperating both the parameters, i.e.,log-sum-exp trick and laplace smoothening with alpha=beta=1.

5 Statement of contributions

Pauline performed Task 1 with features engineering and linear regression. Shubhika implemented Naive Bayes algorithm and perform experiments for this algorithm. Brendon implemented cross-validation and performed experiments for logistic regression. All of us contributed to the writing of the report and approved the final version of the manuscript.

References

[1] Thorsten Joachims. A probabilistic analysis of the rocchio algorithm with tfidf for text categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning*, ICML '97, page 143–151, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

[2] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[3] Vivek Narayanan, Ishan Arora, and Arjun Bhatia. Fast and accurate sentiment classification using an enhanced naive bayes model. *Lecture Notes in Computer Science*, page 194–201, 2013.

[4] M. Poyraz, Z. H. Kilimci, and M. C. Ganiz. A novel semantic smoothing method based on higher order paths for text classification. In *2012 IEEE 12th International Conference on Data Mining*, pages 615–624, 2012.

[5] Vijay and Bala. Chapter 4 - classification models in data science(second edition by morgan kaufmann. *Data Science (Second Edition) by Morgan Kaufmann*, pages 65–163, 2019.

6 Appendix

6.1 Figures

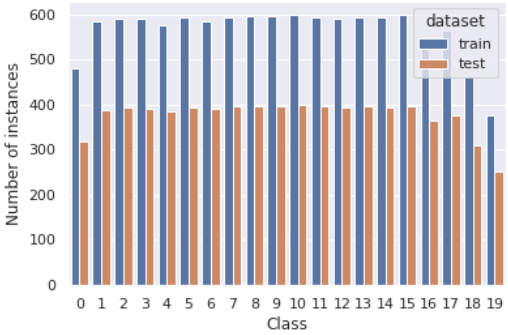


Figure A.1: Distribution of classes over the newsgroups dataset

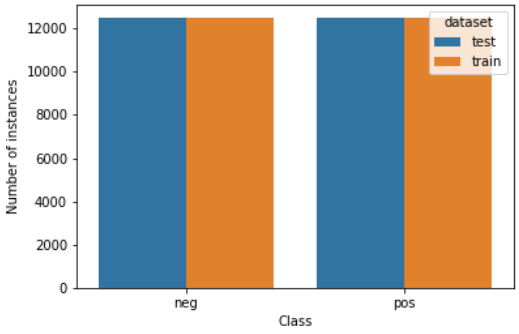


Figure A.2: Distribution of classes over the IMDB dataset

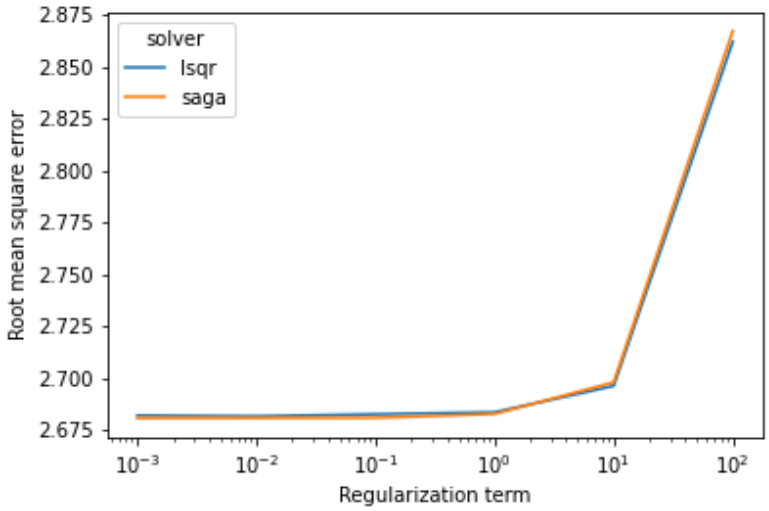


Figure A.3: Root mean square error with the Ridge regularization linear model according to solver and regularization term

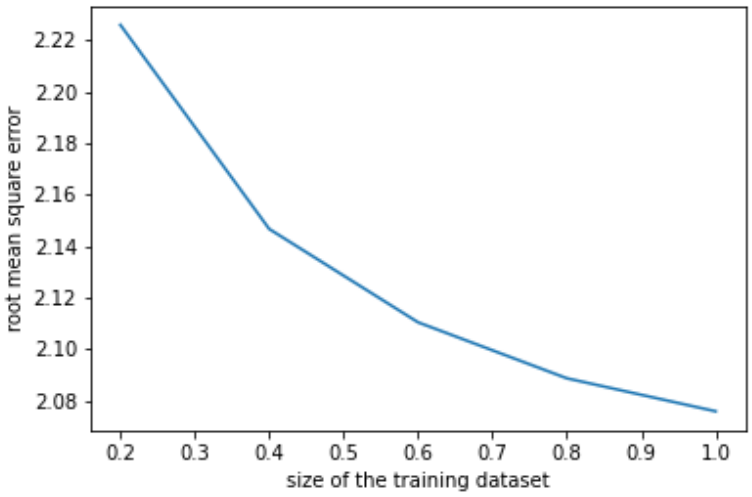


Figure A.4: Performance of Linear Regression on the IDMB dataset according to the size of training set

6.2 Tables

Model	Solver/Regularization ratio	Regularization term value	RMSE
Ordinary Least Square	na	na	2.28
L1 regularization			
	na	0.01	3.16
	na	0.1	4.09
	na	1	4.09
	na	10	4.09
Ridge regularization			
	lsqr	0.001	2.68
	lsqr	0.01	2.68
	lsqr	0.1	2.68
	lsqr	1	2.68
	lsqr	10	2.70
	lsqr	100	2.86
	saga	0.001	2.68
	saga	0.01	2.68
	saga	0.1	2.68
	saga	1	2.68
	saga	10	2.70
	saga	100	2.87
ElasticNet			
	0.25	0.01	2.97
	0.25	0.1	4.01
	0.25	1	4.09
	0.5	0.01	3.09
	0.5	0.1	4.08
	0.5	1	4.09
	0.75	0.01	3.15
	0.75	0.1	4.09
	0.75	1	4.09

Table A.1: Root mean square error with linear regression on the IDMB dataset according to different hyperparameters combination.