COMP 551 – Applied Machine Learning
Mini-Project 2

Omar A— – 2608—
David C— – 2607—
Mounir Hamdar – 2607—

October 18, 2019

Abstract

Reddit, a social media platform, is divided into subcategories called subreddits, each with a specific focus. Using a set of 70000 comments evenly selected from 20 subreddits, we reach a classification accuracy of 58.588% using a voting classifier, constituted from a multi-layer perceptron, a support-vector machine (SVM) with linear kernel and a multinomial Naive Bayes (MNB) classifier.

1 Introduction

The objective of this project was to classify comments coming from 20 subreddits on the popular social media website, reddit. The subreddits we had to classify among were varied in topic, some being related to gaming, sports, movies as well as more general ones pertaining to daily life. We were provided with 70000 comments evenly distributed on the subreddits. Using ensemble methods, we combined the strengths of a Neural Network, SVM and MultinomialNB, to reach an accuracy of 58.588%. Our most important findings include the superior prediction ability of ensemble methods, the importance of proper feature engineering as well as the limitations of PCA, a feature selection method presented in class.

2 Related Work

Previous work on Reddit content classification has been undertaken[2][3]. However, most existing research examines Reddit posts rather than comments, or have access to a far larger set of comments than what is available for this project and w. Nonetheless, there is a general conclusion that neural networks and support vector machines tend to perform best this classification task. Notably, Gutman and Nam[4] achieve an accuracy of 0.773 with a linear SVM under a sample size of 1.7 billion comments.

3 Dataset and setup

The dataset consists of 70 000 comments. These comments have been selected at an even proportion from 20 different subreddits. The comments have been selected with a minimum of 256 characters other than URLs and comments more than 1000 characters long have been truncated.

We first processed the comments by tokenizing, removing punctuation, removing stop words according to the English `NLKT` list, and then stemming using `NLTK`'s Porter stemmer. Subsequently, we removed words appearing only once or twice in the training set, reducing the total amount of distinct words to 14 000 as opposed to 50 000.

None of the comments in the training set came up empty after this treatment, but two of the comments in the testing set came up empty: they were replaced with the underscore symbol.

We then proceeded to vectorize the comments using the term frequency – inverse document frequency (TF-IDF) method, which gave us a sparse matrix with about 10 nonzero entries per row.

Attempts were made at using principal component analysis (PCA) in order to reduce the dimensionality of the data, and thus speed up training time for subsequent models. However, this approach was ineffective as it was transforming a sparse matrix into a dense matrix (thus increasing memory requirements). Moreover, we believe that certain rare keywords are very important in categorization, but their rarity makes it so that variance is low across that dimension. Hence, using PCA leads to ignoring low-variance features that are significant for classification purposes.

4 Proposed approaches

4.1 Validation Pipeline

Almost all of the models attempted below have meta-parameters to optimize, which means that we have to develop a validation pipeline. We used the same method for all of the following models: we used 5-fold cross-validation in order to obtain a coarse estimate of the desired parameters, then used 10-fold cross-validation in order to narrow down these values.

We used k-fold cross-validation as a base, since in our context a difference of 0.2 or 0.3% in accuracy matters. The dataset is not homogeneous enough to use a pre-determined validation set and to assume that the results generalize: some of the models below have an accuracy that can vary up to 1% between folds under 10-fold cross-validation.

However, using 10-fold cross-validation everywhere is very costly in terms of computation time, hence the compromise that we found above.

4.2 Naive Bayes approaches

Naive Bayes models are predicated on the assumption that the features used are all independent priors. As such, we can

The two approaches attempted here are Bernoulli Naive Bayes, which was implemented from first principles, and Multinomial Naive Bayes, for which we used the `scikit-learn` implementation.

We implemented a Bernoulli Naive Bayes model using laplace smoothing giving each of the classes a bayesian prior probability equal to

$$\frac{1}{|C|}$$

where C is the set of classes(subreddits). With:

$$P(x_j = 1|y = k) = \frac{\{X|x_j = 1 \wedge y = k\}| + \alpha}{|\{y|y = k\}| + \alpha \cdot |C|}$$

. For the vocabulary features we chose the most common words/2-grams in all of the comments and the most common ones of each subreddit.

For the Multinomial Naive Bayes mode, we used SKlearn's implementation with a TF-IDF matrix representing the features for each comment. We optimized the model along possible smoothing constants, α.

4.3 Support Vector Machines

Support vector machines (SVM), as introduced by Cortes and Vapnik [6], have as a goal to maximize the distance of the training samples to the decision boundary, which is an affine hyperplane in the feature space. Extensions with nonlinear boundaries also exist: by adding a kernel to the model, we can obtain nonlinear boundaries, such as with polynomial or radial basis function (RBF) kernels. However, the optimization process remains the same.

The optimizing decision boundary is not decided with respect to all vectors; rather, a limited set of support vectors is determined and is used to find the best separation boundary.

Since the distance of the points to the decision boundary is differentiable with respect to the decision hyperplane's parameters, we can use gradient descent in order to find the maximum. In scenarios where memory is constrained, such as in our case, stochastic gradient ascent is employed.

SVM is useful is our case since we are dealing with data that is sparse but highly dimensional. As SVM scales very well with the amount of dimensions, it is a method that is very usable in order to obtain linear classification boundaries.

4.4 Decision Tree and Random Forest

A decision tree is a nonlinear classifier that recursively selects a decision boundary to discriminate between the different classes of the data by computing simple decision boundaries. In the rest of this explanation, we are assuming that the decision in question is binary and is done along a specific axis (ie. feature $x_i > \alpha$), in order to reduce the search space for an optimizing decision tree.

The best decision boundary at any given step of the formation of the tree is decided in order to minimize the remaining entropy of the model, and the recursive decision process stops subdividing the search space into smaller decision regions either when the information gain becomes too small, when there are insufficiently many training points in a given region, or when the tree depth reaches a certain limit.

Random Forest classification, as developed by Tin Kam Ho [5], is an ensemble method where a certain amount of decision trees are trained over a subset of the *features* and then stacked in order to form a classifier.

This is advantageous since maximally-expanded decision trees have high variance, and thus providing an ensemble of decision trees allows to reduce the variance of the model, thus mitigating the effects of overfitting.

4.5 Multi-Layer Perceptron

A multi-layer perceptron, as described by Rumelhart *et al.* [7], is a directed acyclic graph which is organized in layers, and where every artificial neuron from a given layer derives its value from a linear combination of the value of the neurons in the previous layer, and then an *activation function* is applied. The optimization of the model occurs over the precise linear combination of the neurons in the previous layers. Moreover, most loss function are diffferentiable with respect to the weights, allowing the use of gradient methods, or stochastic gradient descent methods in cases where memory is limited.

Multi-layer perceptrons are advantageous in our scenario due to their scalability across large amounts of data, as well as the flexibility of their internal structure (number, size of the internal layers) allowing for complex nonlinear classification boundaries while still retaining reasonable training times.

5 Results

5.1 Multinomial Naive Bayes

In spite of their relatively elementary nature, Naive Bayes classifiers, and particularly Multinomial Naive Bayes, have performed surprisingly well on this dataset. By using 5-fold cross validation followed by 10-fold cross-validation, we were able to determine that $\alpha = 0.175$ was an optimal regularisation constant for the model, which grants us a mean accuracy of 56.879 % over 10 folds.

5.2 Bernoulli Naive Bayes

The Bernoulli Naive Bayes model performed worse than the closely related MultinoNB, only reaching an accuracy of 42% with 20-fold cross validation using the top 150 words/2-grams globally and top 200 words for each subreddit with variable smoothing constants. The lower performance could be attributed to the much more simplistic feature construction as well as the fact that the features for BernoulliNB are binary.

5.3 Linear Support Vector Machines (SVM)

Linear SVMs performed relatively well on this dataset. We used the `sklearn.LinearSVC` class to fit a model, and we used `GridSearchCV` to optimize using grid

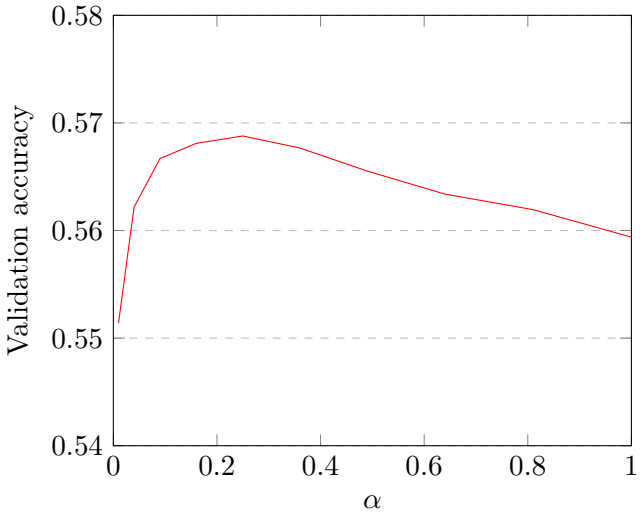


Figure 1: 5-fold cross-validation accuracy for Multinomial Naive Bayes (MNB) under varying regularization parameter α.

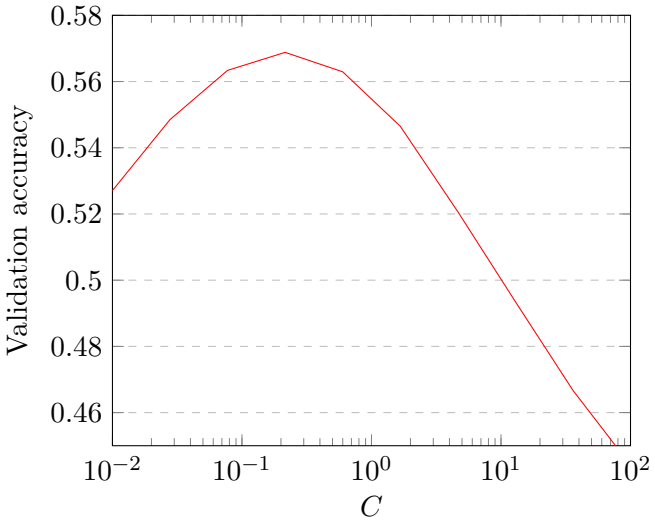


Figure 2: Mean accuracy of SVM under 10-fold cross-validation

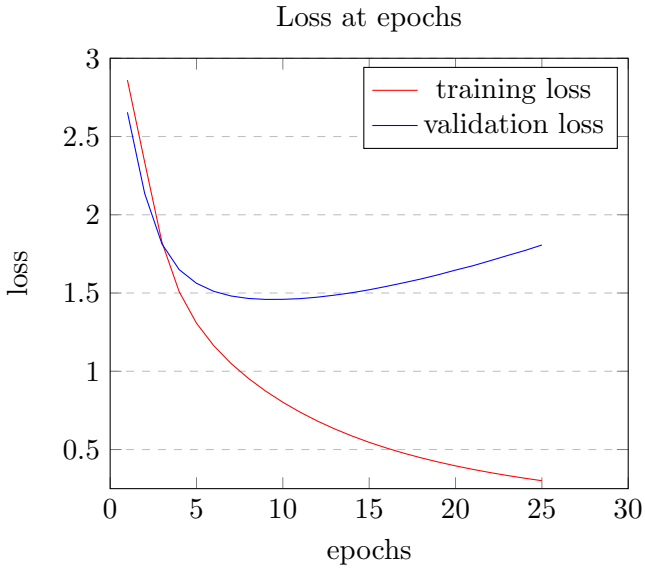
search. We attempted to optimize the C, *loss*, and *penalty* hyperparameters. We determined that the optimal parameters were $C = 0.25$, loss=`squared hinge`, penalty=`L2`. On 10-fold cross validation, this model achieved an accuracy of 0.57. Moreover, some research has indicated that using word n-gram representations of more than one word increases the performance of SVMs[1]. However, attempting this on our dataset marginally reduced accuracy.

5.4 Decision Tree

Decision trees did not perform exceptionally well on this dataset. Using `sklearn`'s `DecisionTreeClassifier` to build a tree, and optimizing over its maximum depth to prevent overfitting, we achieved a maximum accuracy of 0.40 on 10-fold cross-validation. We moreover attempted to use random forests, an ensemble of decision trees, but still achieved a comparable accuracy of 0.396.

5.5 Multi-Layer Perceptron (Neural Networks)

Using `tensorflow` and `keras`, we trained three-layer neural networks with varying parameters on this dataset. Broadly, they performed well. After optimizing over number of epochs, neurons, activation functions, loss functions, and learning rates, we achieved an average accuracy of 0.5855 over 20-fold cross validation. Importantly, to avoid overfitting, we picked an appropriate number of epochs to train by evaluating training and validation loss:



As shown in the graph, validation loss slowly starts increasing around the 8-epoch mark. Based on this observation, we decided to train our neural networks on 8 epochs in order to avoid overfitting.

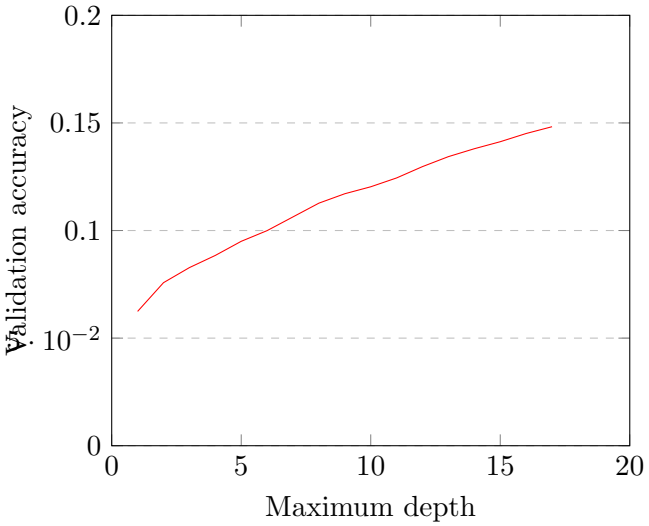


Figure 3: 10-fold cross-validation accuracy for decision tree, as we increase the allowed depth of the tree.

5.6 Ensemble methods

Ensemble methods were, broadly, the best performing classifiers on this dataset. We tried two main approaches: soft voting classifiers, in which the predicted probabilities from each model in the ensemble are summed, and hard voting classifiers, in which only the label predicted by the models is taken into account, with the final label being the most commonly predicted one. Soft voting classifiers performed better than hard voting classifier for any combination of models we tested. The following table shows our final Kaggle accuracy achieved by each soft voting classifier:

Models in voting classifier	Kaggle accuracy
Neural Network, SVM, MNB	0.58588
5 Neural Networks	0.58388
MNB, SVM	0.58311
MNB, Logistic	0.58055

6 Discussion and Conclusion

Broadly, we conclude that ensemble methods are what work best at classifying reddit comments by subreddit. SVM, MNB, and neural networks performed the best as individual models out of everything attempted, and a soft voting classifier of these three models achieved the highest accuracy. Given the overlap in comment content from different subreddits, it makes sense that an ensemble of models that approach classification with very different techniques would reduce variance and increase overall performance. In future research, it might be useful to explore feature engineering further to extract more subreddit-specific information. Specifically, features such comment length and link presence might be worth looking at.

7 Statement of Contribution

- Omar Abdel Baky: Manual implementation of the Naive Bayes model.
- David Carrier: Attempts at using PCA, cross-validation and optimization of parameters for SVM.
- Mounir Hamdar: Feature design, cross-validation and optimization of parameters for Multinomial Naive Bayes (MNB) and for the multi-layer perceptron (MLP).

References

[1] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4147615/

[2] https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1174/reports/2735436.pdf#cite.sr_cluster1

[3] https://pdfs.semanticscholar.org/79b1/72c55d37c0fee543fdbf2f9045b99b6a8a09.pdf

[4] https://jgutman.github.io/assets/SNLP_writeup_gutman_nam.pdf

[5] Available on the Wayback Archive at https://web.archive.org/web/20160417030218/http://ect.bell-labs.com/who/tkh/publications/papers/odt.pdf

[6] https://link.springer.com/content/pdf/10.1007%2FBF00994018.pdf

[7] https://apps.dtic.mil/dtic/tr/fulltext/u2/a164453.pdf