COMP30027 Machine Learning - Project 2 Report

1. Introduction

In this report, we implemented and analysed four Machine Learning systems for identifying the age of author from short text (Schler et al. 2006), using a variety of supervised machine learning techniques. The choice of data representation, classifier and hyperparameters of classifier was described in the experiment design and the behaviour of different models associated with the evaluation of development dataset was justified based on the theoretical properties mentioned in the lectures and relevant documents.

2. Preprocessing

Although the top10 CSV files of training, developing and testing have been provided, they only 30 words as features from 627,632 different alphabetical word in the document, which may remove plenty of key knowledge to predict accurately. Therefore, we decided to process the raw document to gain a better feature representation though pre-implemented packages (such as scikit-learn).

2.1 Feature extraction

In order to extract the features from the raw text documents, the two main approaches are the "Bag of Words" and "Bag of n-grams" representations whose steps are both to tokenize strings, count the occurrences of each token in each text in the documents and normalize the significant tokens. The difference between them are the distinct tokenized way to truncate text. The prior one separates the text by each word while it may ignore the potential misspellings or word deviations. In order to avoid the above limitation, the latter one ignores the word but restricts the char length range of tokenized features (including whitespace) (Schonlau, Guenther, & Sucholutsky, 2017). However, although the "Bag of n-grams" produces better accuracy than "Bag of Words", the dimensions of the "Bag of n-grams" will be too heavy to store and compute in the limited time and memory, especially when our raw data includes 627,632 distinct words. Therefore, in spite of slight lower accuracy, one implementation of "Bag of Words", HashingVecorizer(), was used to select 30,000 features from 627,632 features for the purpose of saving processing time and preventing memory overflow.

2.2 Feature selection

When the total of 30,000 features were included during the experiment, there was a large slowdown in the training time probably because of the heavy computation. The feature selection method can be implemented to transform the high-dimensional datasets and improve the training performance. The univariate feature selection technique was utilized in the preprocessing step to pick the best features based on the univariate statistical tests. The transform method was SelectKBest to remove all but the k features with highest score. We used chi-square as the scoring function which is a common way to evaluate the predictability of classes from attributes with sparse data, then 5,000 attributes were picked out as the training data along with the target set.

3. Experiment Design

An experiment was designed to identify the classifiers and hyperparameters with the best performance for classifying the development data.

Two steps were contained in this experiment, selecting and tuning base classifiers and the ensemble classier, then tuning the probability threshold for each classifier to capture the “?” instances which is not labeled as either in the defined ranges in the development and test dataset.

3.1 Select and Tune base classifiers

Considering the need to identify the probability of each class to generate “?” classification, we only focused on the classifiers with probabilistic predictions in this case. We used grid search method to tune the hyperparameters of each base classier, then adopting the optimal hyperparameters for the next step. The classifier candidates with explored hyperparameter values are described as following.

3.1.1 Naive Bayes

Compared with other classifiers, the Naive Bayes classifier is usually considered as a baseline due to its simplicity and high efficiency. There are three styles of Naive Bayes Classifier, the Multinomial NB (features with occurrence counts) and Bernoulli NB (boolean features) are more suitable for discrete features than Gaussian NB (continuous features) (Pedregosa et al., 2012). In addition to this, the smoothing approaches have effect on the accuracy. Therefore, we compared these two Naive Bayes classifier with distinct smoothing methods.

Hyperparameters of Naive Bayes:

- The choice of smoothing approaches (alpha) : np.arange(0, 1.1, 0.1)
- The choice of the distribution to model the features: [Multinomial Naive Bayes, Bernoulli Naive Bayes]

3.1.2 Stochastic Gradient Descent Classifier (SGD)

Stochastic gradient descent (SGD) learning is a regularized linear model which is suitable for the data represented as sparse arrays of floating point values for the features. It was recommended in this problem because there were thousands of features generated but each blog only contain a few of them. The SGD model can be controlled with the loss parameter, it fits the linear support vector machine (SVM) by default. The regularizer is a penalty added to the loss function that reduces the parameters towards a simpler model to avoid overfitting. The epsilon defines the epsilon-insensitive loss function to ignore errors which are situated within certain distance of the correct label. Therefore, the hyperparameter of loss function, penalty and epsilon were chosen to test in this case.

Hyper-parameters of SGD:

- The loss function that shrinks model to zero vector (loss): [‘hinge’, ‘log’, ‘modified_huber’, ‘squared_hinge’, ‘perception’]
- The regularization term (penalty): [‘l1’, ‘l2’, ‘elasticnet’]
- Epsilon in the epsilon-insensitive loss function (epsilon): np.arange(0, 0.6, 0.1)

3.1.3 K-nearest neighbors (KNN)

KNN method is relatively simple and can handle arbitrarily multiple classes. The typical implementation of KNN method is less popular in problem like authorship age identification with high-dimensional datasets to make many predictions. Conversely, we used it here for the completeness. The number of nearest training instances k and weighting strategies have an effect on the classifier performance (Tan, Kumar, & Steinbach, 2005). Following hyperparameter values of k and distance metric was used to trial and error over the training data.

Hyperparameters of KNN:

- The choice of number of neighbours (k): [1, 20]
- The choice of weight strategy: [uniform, distance]

3.1.4 Random forest

Due to the huge number of attributes in our project, the process of building the deterministic decision tree will be low-efficient with model variance. In order to minimise the overall model variance and enhance the efficiency, random forest is suitable for our project, by only considering a fixed part of the attributes each time and then simply voting the mode of the combined classification of a number of random trees (Ghosal, & Hooker, 2018). The number of random tree and the fixed proportion of all the attributes can influence the performance of the model.

Hyperparameters of Random Forest:

- The number of random tree (n_estimators): [50, 200]
- The maximum fixed proportion of all the attributes (max_features): auto, int

3.1.5 Logistic Regression

Linear and Logistic Regression have significant improvement than Naive Bayes and are suitable for the frequency features. Moreover, our problem sets range of continuous age into corresponding discrete label, '14-16', '24-26', '34-36', '44-46' and '?', so our project adopted the logistic regression.

Hyperparameters of Logistic Regression:

- The penalization norm (penalty): ['l1', 'l2']
- The inverse of regularization strength: [1, 10]
- The weights related to classes (class_weight): ['balanced', None]
- Solver: ['newton-cg', 'lbfgs', 'liblinear', 'sag', saga']

3.1.6 Ensemble System (Stacking)

The stacking method was applied to train a meta-classifier over the outputs of the other base classifiers and try to improve the prediction results. We used logistic regression model as the meta-classier, and the three best performed systems above were selected as the hyperparameters to train the given training dataset.

Hyperparameters of Stacking:

- Base classifiers: SGD, randomForest, MNB
- Meta-classifier: Logistic regression

3.2 Tune thresholds

In the development and test sets, there are some blogs by authors whose ages fall out of either of defined categories in the training set. To identify these unknown classifications, one approach is to find sufficient amount of blog instances that are not represented in the age ranges of training set, while it was too difficult to implement. Therefore, we decided to adopt another more practical solution through using predict probability function with the minimum thresholds for each classifier. If the highest predict probability among the four categories is lower than the threshold, the instance will be predicted as the "?" class. The optimal probability threshold is different for each classifier, hence it was also tuned as a hyperparameter during the experiment. The performance of each classifier on the development dataset was evaluated for the thresholds ranging from 0.01 to 0.99 with 0.01 interval. The probability threshold which contributes to the highest accuracy was selected for that classifier at last.

4. Results and Discussion

Due to the extremely long time of implementation and limitation of computer memory, we decided to exclude the KNN and random forest model when analysing the experiment results. The best hyperparameters and thresholds that leading to the highest accuracy of each classifier were presented in Table 1. The bar chart of the best accuracies for the development dataset was shown in Figure 1.

Table 1. The best hyperparameters, thresholds and accuracy of each classifier

Classifier	Best hyperparameters	Thres holds	Accuracy
NB	alpha=0.5, MNB	0	0.49548
SGD	loss='modified_huber', penalty='elasticnet' epsilon=0.3	0.43	0.50933
LR	penalty='l2', C=5, class_weight=None, Solver='sag'	0.46	0.51004
STK	Classifier=[SGD, randomForest, MNB] Meta-classifier=LR	0.45	0.51035

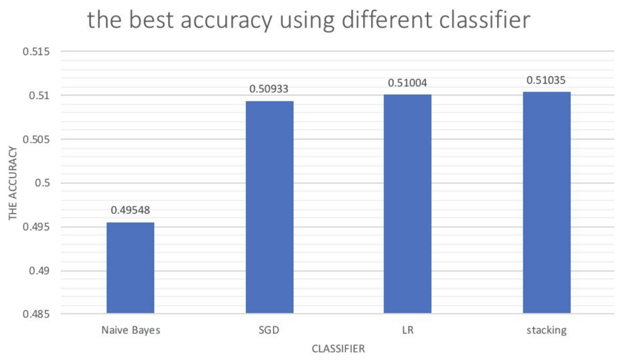

Figure 1. The best accuracy using different classifier

4.1 Result Analysis

From the results of the experiment (Figure 1), the SGD, logistic regression and stacking all have the better performance than MNB. Naive Bayes is usually suitable for small datasets, since such structure can avoid overfitting. Yet, as the size of datasets increasing, the performance of Naive Bayes tends to diminish. Moreover, Naive Bayes assumes that the features are independent with each other, whereas the features in this problem are "word" types that are mutually correlated and contain frequency values. Therefore, linear regression which is particularly suited to frequency-based features performs well with the given datasets. For SGD model, the "modified_huber" loss function gives rise to the relatively better performance among the other values of loss function hyperparameter given that it can bring tolerance to outliers and probability estimates (Pedregosa et al., 2012), although the improvement due to parameter tuning is very small compared to choice of learner and feature representation.

The result of stacking classifier which combines SGD, randomForest and MNB classifiers is slightly better than the best of the base classifiers, it is theoretically because the meta-classifier could be able to use the results of base classifiers to better cope with the bias-variance trade-off and provide predictions with higher accuracy. However, there is just little difference on the accuracies achieved by different base classifiers as well as stacking model. The most possible explanation for this result is that if there are huge amount of data available in dataset, the patterns displayed in the data will be clear, so the selection of classifier probably has little impact on the prediction results and the best choice could be vague (Banko, & Brill, 2001).

4.2 Error Analysis

In order to have better observation of the distribution of predicted classes and real labels for error analysis, Figure 2, 3 and 4 illustrate the confusion matrix of MNB, SGD and LR respectively. As seen in these figures, there are two main reasons to leading to misclassifications. The one is that although the correctness of the

‘14-16’ and “24-26” are high, the considerable amount of prediction is still confused between “14-16” and “24-26”, and between “24-26” and “34-36”. The other one is that most of “?” label still cannot be predicted correctly. Although the correctness of predicting “?” has an ascension after increasing the threshold, other classifications will be predicted as “?” as well, which may decrease the accuracy.

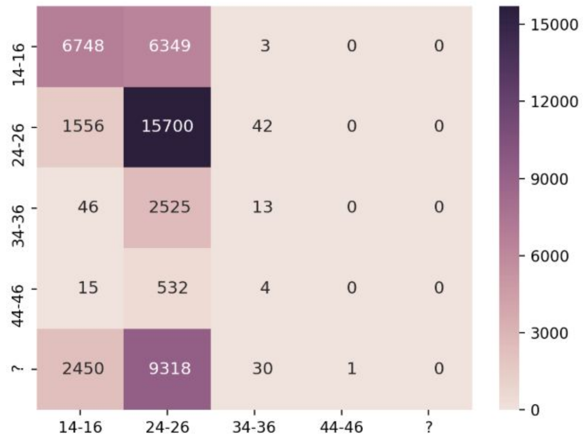

Figure 2. Confusion Matrix of MNB

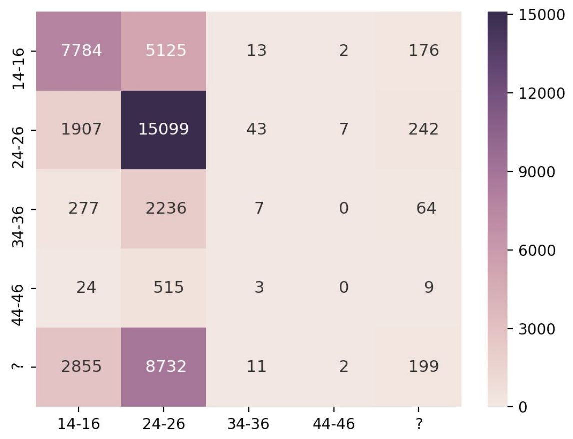

Figure 3. Confusion Matrix of SGD

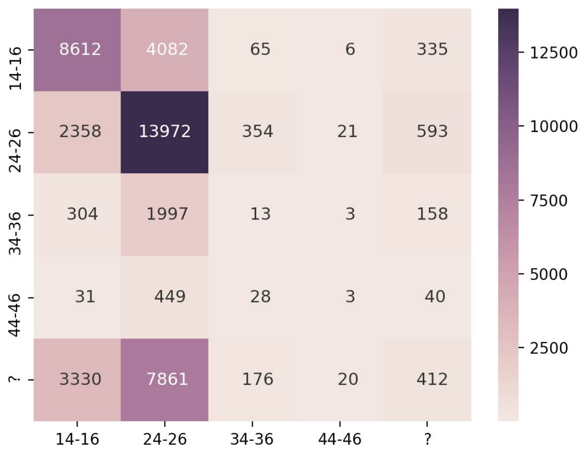

Figure 4. Confusion Matrix of LR

The reasons may contribute to the errors are as following:

- **Training set**:
  - ❖ **Lack of knowledge of “34-36”, “44-46”**: in the training set, there are a quite small proportion of instances labelled as “34-36” and “44-46” while large number of “14-16” and “24-26” instances, it causes our model biased towards the majority classes and the classifications of “34-36” and “44-46” are easy to be treated as outliers when predicting. As shown in confusion matrices above, most of the age ranges of “34-36” and “44-46” are wrongly predicted by each classifier .

  - ❖ **No knowledge of “?”**: the training set has no instance of “?” classification so that the only practical solution is to set the probability threshold of defined classification. Yet, when probability of every class is relatively even, this only means that the property of this instance is unclear but not be labelled as “?”. Therefore, as the threshold rising, the distribution of prediction also proportionally moves to “?”.

- ❖ **Inconsistent combination of features**: some combinations of the development set are absent in the training set and the class distribution in the training set is quite different to that in the development set, so that when using some classifiers like random forest and Naive Bayes, the results may be adversely influenced.

● **Non-optimal feature representation**:

- ❖ **“Bag of Words”**: due to the time and memory constraint, our project abandoned the better solutions, “Bag of n-grams” and multiple words, so that our project ignored the relationship between phrases and the probability of misspelling and deviation. This will lose some information of raw documents and hardly find the patterns from inadequate features.

- ❖ **“SelectKBest(n_feature=5000)”**: when using the TfidfVectorizer() function, it will return 399,108 features, while most of these features are useless or contains little knowledge. Therefore, we decided to select 5000 best features, which means some knowledge was removed, in spite of reducing quite little accuracy.

- ❖ **Not use perl**: because my teammate and I are not familiar with perl and perl is not the core of this project, we chose the feature extraction function in sklearn. It is undeniable that using perl will get better features than using HashingVectorizer(), since it can filter and cleanse the data based on the characteristic of the raw document, such as removing numeric characters, extracting emoji and so on.

● **Tune hyperparameter:**
it is hard to select the best combination of the hyperparameter due to its time-consuming and these hyperparameter may adversely impact on the accuracy. For example, the base classifiers and meta-classifier in stacking should be complementary rather than those classifiers with the highest accuracy individually.

Moreover, the content of the blog is not only related to author’s age but also other factors, such as the degree of his or her education, career and so on.

Based on the error analysis listed above, there are some testable hypotheses about how we could improve our model:
● Use perl: use the provided scripts but select more than top 10 features, such as top 30 features. Consider the relationship between words or the use the emoji.

● Add more instances in the training set: collect more instances in “34-36”, “44-46” and other age out of classifications.

● Consider more factors related to age rather than only a short blog.

5. Conclusion

In this report, we outlined our methodology to generate several supervised age classifiers based on short blog text documents of the authors. According to the characteristics of classifiers, our final learners were MNB, SGD, LR and STK which reached the maximum accuracy of 0.51035, but the error rate is still high. For future studies, we will further adopt the possible solutions and deeply analyse the experiment results to enhance the performance of model.

References

Banko, M., & Brill, E. (2001). Scaling to Very Very Large Corpora for Natural Language Disambiguation. *ANNUAL MEETING-ASSOCIATION FOR COMPUTATIONAL LINGUISTICS*, 26.

Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., & ... Varoquaux, G. (2013). API design for machine learning software: experiences from the scikit-learn project.

Ghosal, I., & Hooker, G. (2018). Boosting Random Forests to Reduce Bias; One-Step Boosted Forest and its Variance Estimate.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., & ... Duchesnay, É. (2012). Scikit-learn: Machine Learning in Python.

Schler, J., Koppel, M., Argamon, S., & Pennebaker, J. (2006) . Effects of Age and Gender on Blogging. *In Proceedings of 2006 AAAI Spring Symposium on Computational Approaches for Analyzing Weblogs*. Stanford, USA.

Schonlau, M., Guenther, N., & Sucholutsky, I. (2017). Text mining with n-gram variables. *Stata Journal*, 17(4), 866-881.

Tan, P., Kumar, V., & Steinbach, M. (2005). *Introduction to data mining*. Boston : Pearson Addison Wesley, c2006, [i.e. 2005].