CM50265 Machine Learning 2: Coursework 2 Lab Report (Group 23)

1. Task 1: SVM Classification Task Analysis

1.1. Data Pre-processing

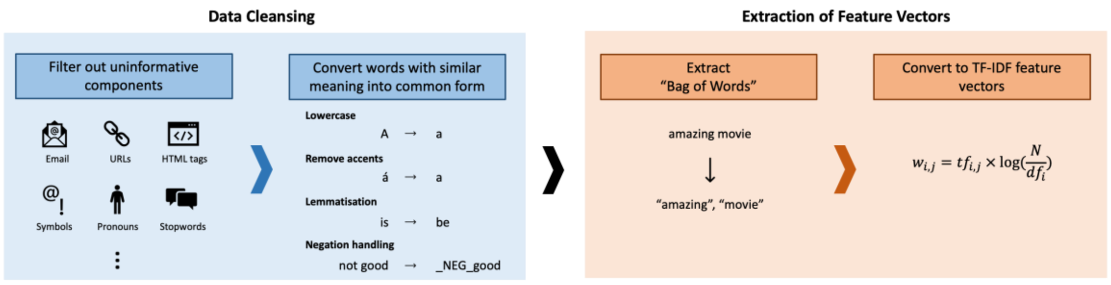


Figure 1.1-1 Steps of data pre-processing

1.1.1. Data Cleansing

A series of steps are taken to clean the raw movie review data and reduce the size of vocabulary in the dataset, so that the SVM model can focus on more important features during classification. Some of the steps taken include removal of symbols, conversion of accented characters and removal of stopwords. The reviews are then lemmatised by converting words with close lexical meaning into a common form. Negations in the review are handled by tagging the word immediately after a negating word such as "not", "no", and "none", which helps to interpret negated meanings.

1.1.2. Feature Vector Extraction

The lemmatised reviews are then converted into TF-IDF feature vectors, which represents the importance of a word or a sequence of word by considering its count within a review and among all reviews in the data set. A higher weight is assigned to the feature if it appears more frequently in the review but is rare among all reviews as they might be more discriminative and could be more valuable to the sentiment classification task. The settings of for the TF-IDF vectoriser is detailed in Table 1.1-1.

Table 1.1-1 Parameter settings for the TF-IDF vectoriser

Parameter	Value
min_df	2
max_df	0.7
ngram_range	(1,5)
max_features	30,000
smooth_idf	True
sublinear_tf	True

1.2. Evaluation on SVM Classifiers with Different Kernels

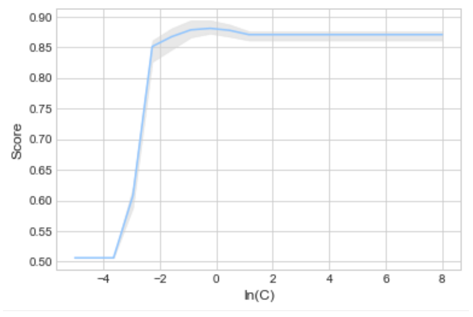
The feature vectors obtained after data processing are fed into the scikit-learn SVC classifier. Three default kernels, namely linear, RBF and polynomial, and a custom kernel are tested on the dataset.

1.2.1. Hyperparameter Tuning

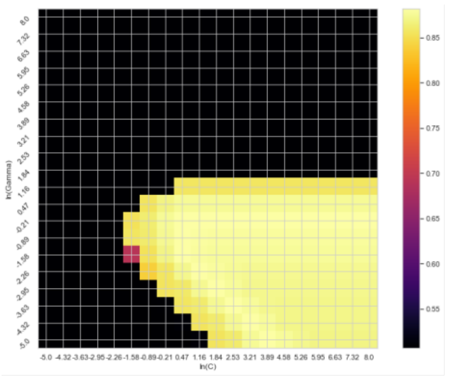
Stratified 5-fold cross-validation grid searches are performed to search for the best hyperparameter values for the four kernels. The range of values searched are listed in Table 1.2-1.

Table 1.2-1: (Task 1) Ranges of values searched during hyperparameter tuning

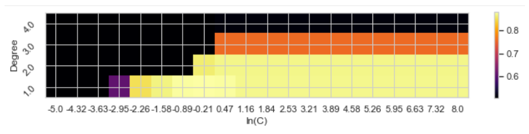
Hyperparameter	Range of values searched	Kernel (s)
C	$[e^{-3}, e^{8}]$, 20 steps	All
Gamma	$[e^{-3}, e^{8}]$, 20 steps	RBF kernel
Degree	{1, 2, 3, 4}	Polynomial kernel



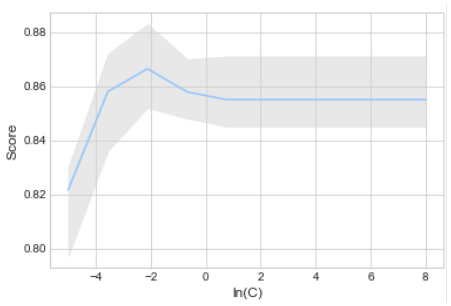
(a) Linear kernel



(b) RBF kernel



(c) Polynomial kernel



(d) Custom kernel

Figure 1.2-1: (Task 1) Performances of SVM kernels during hyperparameter tuning

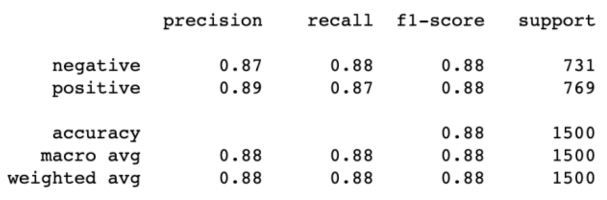
a. SVM with Linear Kernel
The linear SVM classifier involves only one hyperparameter C, which determines the penalty for the error term. A lower value of C implies that the classifier is more lenient to misclassifications, vice versa. It is found that values of C below $e^{-2.95}$ leads to a poor performance of the SVM classifier with accuracies lower than 60%. Besides, the variance of performance over folds is smaller with values of C greater than $e^{1.5}$. The best value of C found for the linear kernel is $e^{-0.21}$, which achieves an average accuracy of 88.1% over 5 folds.

b. SVM with RBF Kernel
The RBF SVM classifier has one additional hyperparameter, gamma. Gamma determines the curvature of the decision boundary. Figure 1.2-1 – (b) illustrates the performance of the RBF SVM classifier with different combinations of C and Gamma values. It can be observed that the classifier generally does not perform well with C values lower than $e^{-1.58}$ or with Gamma values above $e^{1.84}$. The best combination of parameters found for the kernel is $\{\, C = e^{0.47}; Gamma = e^{-0.21}\}$, which achieves an average accuracy of 88.2% over 5 folds.

c. SVM with Polynomial Kernel
We tuned the hyperparameters C and degree for the polynomial SVM classifier. From Figure 1.2-1 – (c), it is found that the classifier does not perform well with C smaller than $e^{2.95}$ or with degree greater than 2. The best combination of parameters found is $\{\, C = e^{0.47}; degree = 1\}$, which achieves an average accuracy of 88.0% over 5 folds. A best degree of 1 means that the kernel finds a linear separation performs the best for this dataset.

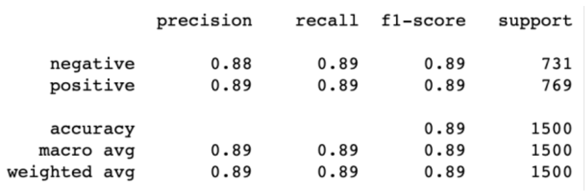
d. SVM with Custom Kernel
The custom kernel implemented is the histogram intersection kernel. As the kernel requires heavy computation, the number of features is reduced to 5,000. This SVM classifier involves only the C hyperparameter. The classifier performs slightly better with a C value at $e^{-2.1}$, which achieves an average accuracy of 86.6% over 5 folds.

1.3. Comparison of Kernel Performances



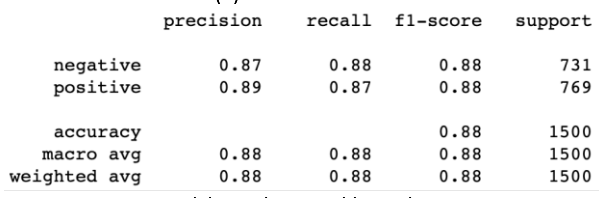
```
              precision    recall  f1-score   support

    negative       0.87      0.88      0.88       731
    positive       0.89      0.87      0.88       769

    accuracy                           0.88      1500
   macro avg       0.88      0.88      0.88      1500
weighted avg       0.88      0.88      0.88      1500
```

(a) Linear kernel



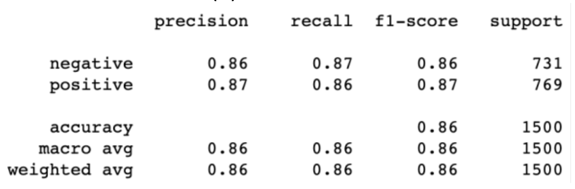
```
              precision    recall  f1-score   support

    negative       0.88      0.89      0.89       731
    positive       0.89      0.89      0.89       769

    accuracy                           0.89      1500
   macro avg       0.89      0.89      0.89      1500
weighted avg       0.89      0.89      0.89      1500
```

(b) RBF kernel



```
              precision    recall  f1-score   support

    negative       0.87      0.88      0.88       731
    positive       0.89      0.87      0.88       769

    accuracy                           0.88      1500
   macro avg       0.88      0.88      0.88      1500
weighted avg       0.88      0.88      0.88      1500
```

(c) Polynomial kernel



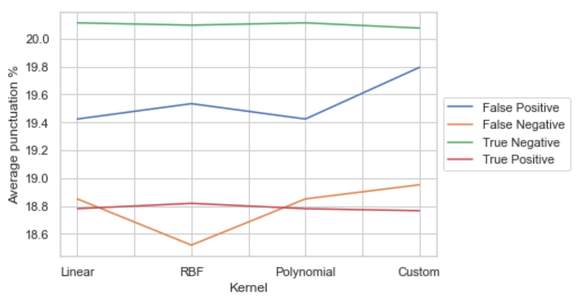
```
              precision    recall  f1-score   support

    negative       0.86      0.87      0.86       731
    positive       0.87      0.86      0.87       769

    accuracy                           0.86      1500
   macro avg       0.86      0.86      0.86      1500
weighted avg       0.86      0.86      0.86      1500
```

(d) Custom kernel

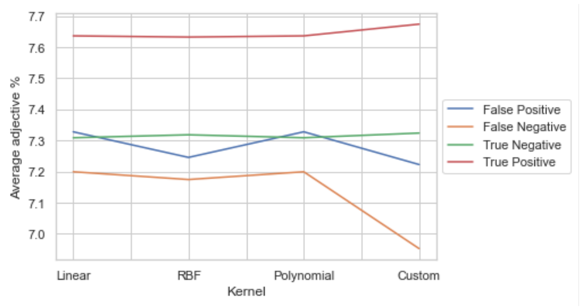
Figure 1.3-1: (Task 1) Classification reports of the SVM kernels

As shown in Figure 1.3-1, linear, RBF and polynomial kernels have similar performances. RBF kernel achieves the best accuracy at 89%. The custom kernel performs slightly worse at 86%, which could be caused by the reduction in feature vector size. All kernels have a slightly higher precision on positive reviews and slightly lower recall for positive reviews, this might indicate that the model is biased towards predicting reviews as negative. The similarity in performances of linear and polynomial kernels can be explained by the use of linear separator by both kernels.

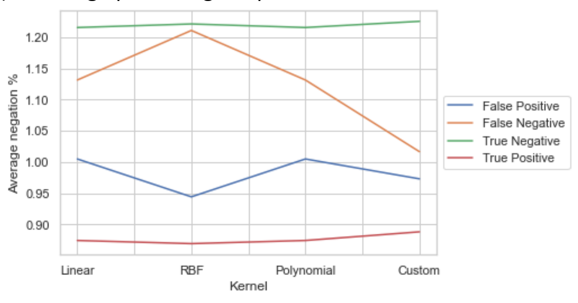
1.4. Analysis on Characteristics of Misclassified Samples



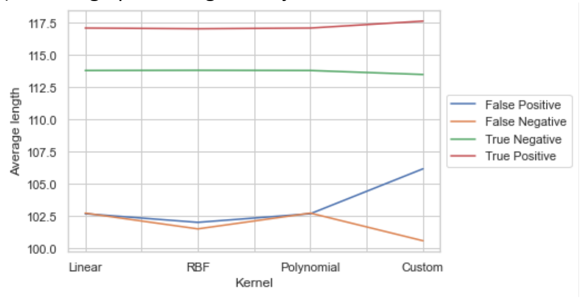
(a) Average percentage of punctuations in raw reviews



(b) Average percentage of adjectives in raw reviews



(c) Average percentage of negating words in raw reviews



(d) Average length of pre-processed reviews

Figure 1.4-1: (Task 1) Characteristics of different classification results

Characteristics of the classification results (true positive, true negative, false positive, false negative) made by SVM with each of the four kernels are analysed (Figure 1.4-1). The following patterns are observed among all kernels:

- **Misclassified samples have more punctuations than the correct classifications:** All punctuations in the reviews were removed during data pre-processing, which prevents the models from analysing the correlation between punctuations and sentiment, and result in misclassifications.
- **In general, true positive samples have significantly more adjectives than the other samples, while false negative samples have the least percentage of adjectives:** This might imply that the models perform well in distinguishing positive sentiments with lots of adjectives present. Difference between the average percentage of adjectives in false positive and true negative samples is minimal.
- **The range of average percentage of negating words between true positive and true negative samples is wide, and true negative and false negative samples have the highest percentage of negating words**: This might imply that the presence of negation might be a discriminative factor, and the classifiers tend to predict negative sentiment with the presence of negating words. However, this might also mislead the classifiers in correctly classifying positive sentiment reviews with negating words.
- **Correct classifications are generally lengthier than misclassified samples:** One possible interpretation is that with more words available in the review, more discriminative features might be extracted and the classifier can therefore performed more fine-grained classification for more accurate results.

2. Task 2: Boosting Task Analysis

2.1. Algorithm Outline

The AdaBoost algorithm is implemented. AdaBoost is employed to combine many weak classifiers into one strong classifier. By training and averaging many decision trees, a more robust classifier can be obtained. As each weak-learner is trained, data points are re-weighted to impose higher penalty on misclassified samples, so that the successive decision trees will focus on distinguishing the more difficult reviews. When combining trees into a strong classifier, the trees with low error receive high weighting whilst poorly performing trees have low contribution.

2.2. Data Pre-processing

The data pre-processing methodology for AdaBoost is the same as SVM. Again, the classifier is assisted by the removal of noisy words and the clarification of negatives in the data.

2.3. Evaluation on Algorithm Parameters

2.3.1. Testing on Variants of AdaBoost

Multiple implementations of AdaBoost were tested, mostly varying how the weight vectors are updated. Equations below are some of the formulae that we encountered for updating the weights of tree stumps, which effectively alters the learning rate of AdaBoost. Some dampen the update by 0.5 and some use a constant factor ϵ to stablise the update.

Through trial-and-error, we selected the best performing implementation, which is equation (1), to use in subsequent testing.

$$\alpha_m = \ln\frac{1-err_m}{err_m} \tag{1}$$

$$\alpha_m = \frac{1}{2}\ln\frac{1-err_m}{err_m} \tag{2}$$

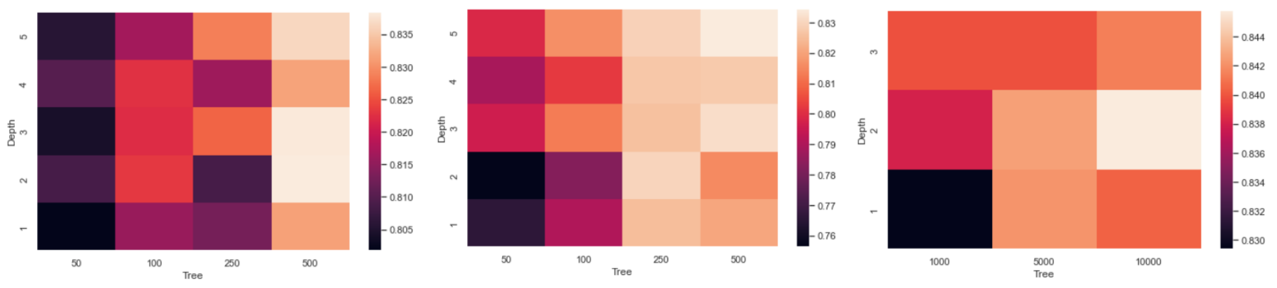
$$\alpha_m = \frac{1}{2}\ln\frac{1-err_m}{err_m+\epsilon} \tag{3}$$

2.3.2. Hyperparameter Tuning

Again, stratified 5-fold cross-validation grid searches are performed to search for the best hyperparameter values. We split the searches into two sub-grids to gain insight into the sensitivities while avoiding extreme runtime combinations. After the first search, it was found that splitting criterion has minimal effect on the overall accuracy, hence only Gini is tested in the second search. The range of values searched are listed in Table *2.3-1*. While 10,000 depth-2 trees produced the best accuracy, improvement was only marginal compared to the much quicker 5000 depth-1 model.

Table 2.3-1: (Task 2) Ranges of values searched during hyperparameter tuning

Grid	Tree depth	Number of Trees	Split Criterion
Search 1	{1, 2, 3, 4, 5}	{50, 100, 250, 500}	{gini, entropy}
Search 2	{1, 2, 3}	{1000, 5000, 10000}	{gini}



(a) Search 1 with Gini splits (b) Search 1 with Entropy splits (c) Search 2 with Gini splits

Figure 2.3-1: (Task 2) Performances with different hyperparameters

As shown in Figure 2.3-1, AdaBoost is trained with different number of trees and depths. It can be observed that lower number of trees and tree depth generally result in lower accuracies, which could be caused by underfitting. 10,000 trees and depth of 2 achieved the highest performance, however, as running 10,000 trees with depth of 2 would exceed the time limit, the final model selected for the purpose of this assessment is of 10,000 trees and depth 1.

2.4. <u>Analysis on Misclassified Samples</u>

2.4.1. Analysis on the Frequency of Features

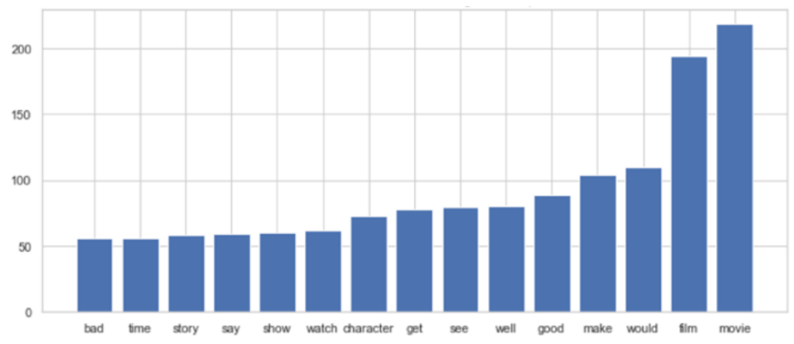


Figure 2.4-1: (Task 2) Most common words in false negative samples

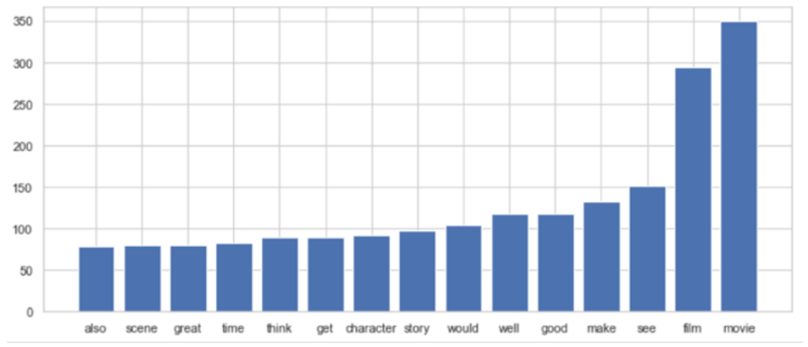
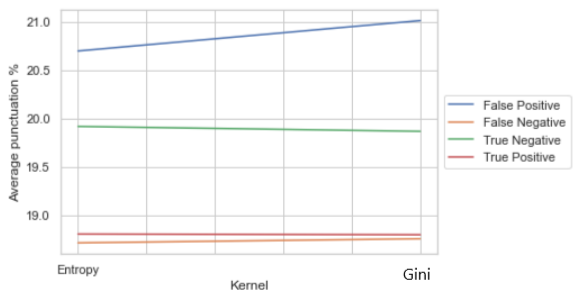


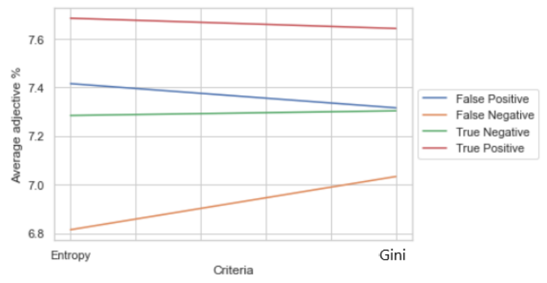
Figure 2.4-2: (Task 2) Most common words in false positive samples

By analysing the most common words in false negative and false positive samples predicted by AdaBoost (Figure 2.4-1 and Figure 2.4-2), it is found that sentimental adjectives may not truly reflect the sentiment of reviews. For instance, 'bad' is one of the most common word found in false negative samples, while 'good' and 'great' are some of the most common words in the false positive samples. This might cause misclassifications as it is likely that these adjectives are used as some of the strong discriminators by the majority of trees in our model.

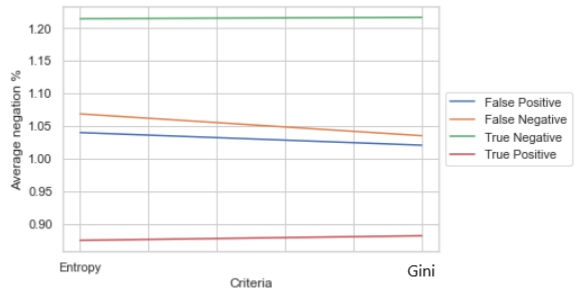
2.4.2. Analysis on Other Indicators



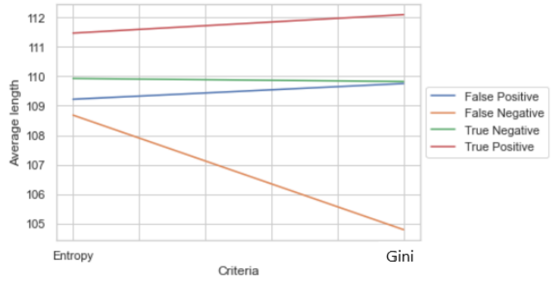
(a) Average percentage of punctuations in raw reviews



(b) Average percentage of adjectives in raw reviews



(c) Average percentage of negating words in raw reviews



(d) Average length of pre-processed reviews

Figure 2.4-3: (Task 2) Characteristics of different classification results

The following highlights some common patterns observed and the slight differences between the Gini and Entropy-split models:

- **Misclassified samples generally have higher percentage of punctuations:** This aligns with the finding in task 1. On the other hand, the average punctuation of false positive samples with the use of Gini is slightly higher than that of Entropy's. However, the overall differences between the two split criterion are minimal.
- **Positive classifications contains more adjectives than negative classifications:** This might indicate that both models tend to make positive prediction with the presence of more adjectives. On the other hand, the average percentage of adjectives in correct predictions in the Gini-split model is higher than that of Entropy-split's, which might imply that Gini performs slightly better than Entropy when there are more adjectives in the reviews.
- **The range between the percentage of negating words in true positive and negative reviews is wide, and true negative and false negative samples have the highest percentage of negating words:** This again aligns with the finding in task 1. In addition, it can be said that there is almost no difference between the Entropy and Gini values in terms of average percentage of negation in reviews.
- **True positive and false positive samples have a higher average length:** The average length of true positive and false positive samples is the highest. This fact is similar for both Gini and Entropy and might indicate that the classifier is biased towards making positive predictions with lengthier reviews.
- **The average length of false negative samples for the Gini model is shorter than that of Entropy model's:** With the patterns in other classification results being similar for both models, this could imply that the Gini model is weaker than the Entropy model in correctly classifying short positive reviews.