Assignment 3

due November 18

CSMC 352: Machine Learning
Prof. Rose Sloan
Nathan Cho
nc5123@bard.edu

1. Algorithms

Implementation

The following algorithms were added from the scikit-learn package.

K-Nearest Neighbors	(using **`sklearn.neighbors.KNeighborsClassifier`**)
Decision Tree	(using **`sklearn.tree.DecisionTreeClassifier`**)
Logistic Regression	(using **`sklearn.linear_model.LogisticRegression`**)
Linear Support Vector	(using **`sklearn.svm.LinearSVC`**)
Support Vector	(using **`sklearn.svm.SVC`**)
Random Forest	(using **`sklearn.ensemble.RandomForestClassifier`**)

Results

Algorithm	**Accuracy**	**Precision**	**Recall**	**Training Time** (seconds)	**Validating Time** (seconds)
K-Nearest Neighbors	0.582	0.672	0.320	1.79	13.44
Decision Tree	0.720	0.719	0.723	54.36	0.08
Logistic Regression	0.845	0.843	0.848	18.02	0.14
Linear Support Vector	0.845	0.840	0.852	253.47	0.10
Support Vector	0.886	0.864	0.914	2135.00	362.17
Random Forest	0.843	0.840	0.847	49.21	0.23

Raw terminal outputs are available at ***problem1_results.txt***.

The **support vector algorithm** performed the best for this given task in terms of metrics, its time complexity shows that support vector algorithms are not scalable for large training datasets with high dimension features. (See line 120 of ***problem1.py*** for further analysis in time complexity)

As we can see logistic regression and linear support vector algorithms perform better than kNN and decision tree algorithms, we can infer that the data is **linearly separable to some degree.** With the support vector algorithm, as it performs better than linearly separating algorithms, we can infer that the data **can be separated better with non-linear decision boundaries.** But as the performance gain isn't substantial from linear separation and non-linear separation, we can infer that the data **contains outliers which can't be easily classified.**

2. Hyperparameters

Assumption

With the assumption that the current performance can be improved by classifying outliers correctly, we can modify the hyperparameter **gamma** to improve outlier classification. Increasing gamma will increase the support vector expansion, which means the decision boundary will be more fitted to the training data set. This will ultimately create a decision boundary which contains more outliers.

Relative to the default gamma value, multiples of 0.1, 0.2, 0.5, 1, 2, 5, and 10 were tested as gamma values.

Test 1: Test each gamma values

The following is the results of the algorithm testing with multiple gamma values.

Factor		default / 10	default / 5	default / 2	default	default * 2	default * 5	default * 10
Gamma		1.5E-05	3.0E-05	7.5E-05	1.5E-04	3.0E-04	7.5E-04	1.5E-03
Training	Accuracy	0.913	0.956	0.978	0.992	0.998	1.000	1.000
	Precision	0.969	0.946	0.971	0.989	0.996	1.000	1.000
	Recall	0.852	0.966	0.984	0.994	1.000	1.000	1.000
Validation	Accuracy	0.856	0.858	0.860	0.868	0.844	0.674	0.548
	Precision	0.923	0.844	0.844	0.859	0.820	0.854	0.815
	Recall	0.788	0.892	0.896	0.892	0.896	0.450	0.169

To increase learning performance, only the first 5,000 rows were used for training.

Test 2: Grid search all gamma values

Using **`GridSearchCV`** and **`StratifiedShuffleSplit`** provided by the scikit-learn package, the same gamma values were tested.

Factor	default / 10	default / 5	default / 2	default	default * 2	default * 5	default * 10
Gamma	1.5E-05	3.0E-05	7.5E-05	1.5E-04	3.0E-04	7.5E-04	1.5E-03
Accuracy	0.763	0.852	0.853	0.846	0.826	0.714	0.563

Results

Unlike the assumption, **decreasing the gamma value resulted in better performances**. The reason for this is because the model was already well fitted to the training dataset with the default gamma value. As we can see in the results of Test 1, the training set performance was very close to 1.00 on gamma value bigger than default. This means that the model will be too overfitted with values bigger than default.

From the test results, using the default gamma value (or a value slightly smaller than that) would make the model perform the best. Using this gamma, the accuracy, precision and recall scores were 0.868, 0.859 and 0.892. (Shown in Test 1)

3. Features

Ideas

The following modification of features were implemented for these reasons:

A: Remove all features except top 30~1000 words in frequency
→ improves accuracy by focusing on important words, better time complexity with less features

B: Remove all features except sentiment vocabulary features (from given word dataset)
→ improves accuracy by filtering out neutral words

C: Add new feature of number of words in a review (found in bag of word dataset)
→ *might* improve accuracy by adding another dimension of metadata

D: Add new feature of number of positive words in a review
→ improves accuracy as there will be a direct correlation between positive words used and the overall sentiment of the review

E: Add new feature of number of negative words in a review
→ vice versa of above

F: Add new feature of number of neutral (neither positive or negative) words in a review
→ *might* improve accuracy by adding another dimension of metadata

Results

Feature set		**Original**	**A**	**B**	**A + B**	**C**	**D + E**	**D + E + F**
# of features		6,666	970	1,339	2,169	1	2	3
Training	**Accuracy**	0.992	0.940	0.927	0.948	0.519	0.736	0.738
	Precision	0.989	0.934	0.914	0.944	0.535	0.739	0.745
	Recall	0.994	0.946	0.943	0.953	0.286	0.729	0.726
Validation	**Accuracy**	0.868	0.864	0.850	0.856	0.511	0.736	0.744
	Precision	0.859	0.856	0.832	0.838	0.509	0.745	0.742
	Recall	0.892	0.880	0.884	0.882	0.274	0.733	0.728

Unlike expectations, none of the modifications performed better than the original feature set.

Reason why the original feature set performs better than A, B or A + B is because learning with the entire bag of words trains the model to fit to less frequently used words. This can be seen in the training set metrics where the original feature set trains the model closest to the dataset.

Reason why C, D + E, D + E + F didn't yield any meaningful results is because the number of features were too small and there were no direct correlations between the number of sentimental words and the overall sentiment of the review.

Using the support vector algorithm with default gamma and original feature set, the accuracy, precision and recall scores for the test set were 0.887, 0.862 and 0.916. As all of the scores are over 0.85, we can say this model is pretty accurate and reliable. But due to its massive time complexity in both training and testing process, using faster algorithms such as random forest or logistic regression can be a more practical solution. In order to maximize the accuracy, I would use a model that can understand the relationship between words such as recurrent neural network.