Predicting Financial Crisis from Machine Learning Approach

Sijia Dong, Yingrui (Ray) Chang *Team ID: SD_YC*

1 Overview

In this project, we have studied various learning algorithms to predict the appearance of the financial crisis era based on a collection of words extracted from thousands of quarterly and yearly reports of large financial service companies. Comparing the cross validation scores from different models, we choose the best performed Gradient Boosting Classifier after using regularized logistic regression for dimensionality reduction as our final learning algorithm. The overall best accuracy we achieved is 0.94275 on the public leaderboard, and the out-of-sample accuracy on the training data from 5-fold cross validation is 0.9375.

The features we have are words from a bag of words. These words are the top 500 most frequent word stems in the reports published in the U.S. Recession era with stop words filtered out. Our task is to reveal the correlation of the frequency of words and the appearance of the financial crisis. Since the dimensionality is high and our original models tend to overfit the data, we choose to eliminate unimportant features before applying the learning algorithm. In our study, we compared the performance of various models using two most popular dimension reduction approaches, regularized logistic regression and principle component analysis (PCA). Our results shows feature selection using logistic regression out performs PCA for this particular problem. We compared the performance of different learning algorithms and decide to report three interesting models, logistic regression, support vector machine (SVM) and gradient boosting classifier. We used cross validation to identify the optimal parameters for each model and choose the best performed as our final learning algorithm. The learning algorithms are implemented via open source packages *LIBSVM* [Chang and Lin, 2011] and *scikit-learn* [Pedregosa et al., 2011].

Due to the nature of this project, we focus on understanding the learning behavior of each model rather than the absolute score we achieve. The rest of the report is arranged as follows: In section 2, we discuss the possible strategies to preprocess data and possible modeling approach. In section 3, we summarize three interesting models and their learning behavior. Data preprocessing specific to the algorithm is also detailed in the respective subsection. In section 4, we compare the cross validation score of these model and choose the best performed model as our final algorithm.

Yingrui mainly worked on PCA, logistic regression and gradient boosting models, and Sijia mainly worked on support vector machine models. Yingrui also worked on decision tree and random forest. Sijia also attempted the naive linear regression model (assigning predictions < 0.5 as 0, values ≥ 0.5 as 1) as the first test, and was able to achieve an out-of-sample accuracy of 0.891. Both Yingrui and Sijia were involved in planning the project and conducting the initial tests of each other's models.

2 Data Manipulation and Preprocessing

The plain training data contains the frequency of each word in "top 500 most frequent word stems" from each financial report. However, in a large text corpus, the plain raw data is rarely used either because some words will be very present, hence carrying very little meaningful information about the actual contents of the document, or the less frequently happened words would provide less information on their importance. To address this issue, we use the provided "tf_idf" formatted data, which is a numerical statistics intending to reflect the importance of each word. For the purpose of this project, we do not study the effect of different variations of "tf_idf" weighting scheme on the total learning behavior.

2.1 Principle Component Analysis

Principle component analysis (PCA) [Pearson, 1901, Hotelling, 1933] is a common approach to extract information about the data and search the optimal lower dimension space which explains the majority of the variance. Fitting model in the principle space would effectively reduce the computational cost and possibly have better chance of generalization.

The original features represent 500 words. We select the first 100 principle components and plot the explained variances on the principle directions versus the number of principle direction shown in figure 1. We see the first 100 principle variances are able to capture over 95% of the total variance. In section 3, we will see how the model built on this lower dimensional space affects their behaviors.

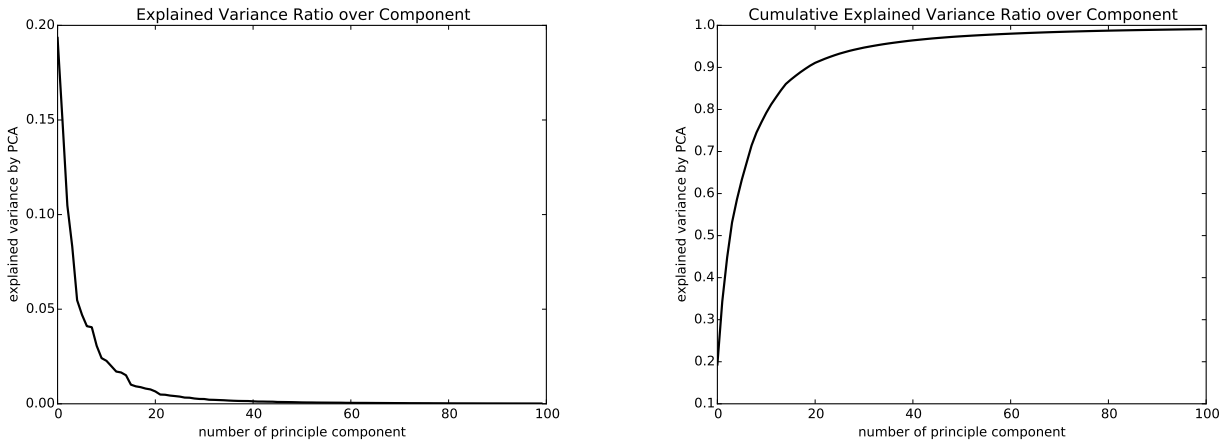

Figure 1: Explained variance (left) and accumulative explained variance via principle component decomposition

2.2 Data Preview and Modeling Strategy

A quick analysis of the training data shows there are more negative labels (82.325% predicting zero) than positive labels. A naive guess of all negative prediction would results in more than 82.325% of accuracy. In order to give a meaningful prediction, we seek learning model that could perform better than 0.82325 accuracy. We summarize the modeling approach as follows: first use PCA or logistic regression to select important features, based on which we rescale the input data making every feature in comparable range. We train different classification models on the resulting normalized data (including the data not subject to feature selection but subject to scaling) using cross validation scores as the criteria to evaluate the performance.

Since the training data are biased over the negative labels, to achieve cross validation score with low variance, we need to make sure the ratio of positive and negative labels in each fold be approximately the same. Thus we use the "StratifiedKFold" function in *scikit-learn* [Pedregosa et al., 2011] to make sure each fold contains the same number of positive labels. For the SVM model, since we use LIBSVM in which stratified cross validation is not an existing code, we turn to use different scoring metrics to evaluate the model performance and use a weighted parameter C to penalize the wrong classification of ones more. This is described in detail in section 3 below.

3 Learning Algorithm

In this section, we report the algorithm and behavior of three most significant learning models, logistic regression, support vector machine and gradient boosting classifier. We also report the model selection within each algorithm. Although logistic regression mainly served as a feature selection approach, its classification ability is still very interesting to discuss. Thus we investigate the feature selection of logistic regression and in later sections using its results to train more powerful models.

Since this problem is a binary classification problem, we report the following classical measure in later sections,

$$\text{accuracy} = \frac{\text{tp} + \text{tn}}{\text{tp} + \text{tn} + \text{fp} + \text{fn}}, \tag{1a}$$

$$\text{precision} = \frac{\text{tp}}{\text{tp} + \text{fp}}, \tag{1b}$$

$$\text{recall} = \frac{\text{tp}}{\text{tp} + \text{fn}}, \tag{1c}$$

$$\text{F-score} = 2\frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}, \tag{1d}$$

in which tp, tn, fp and fn represent "true positive", "true negative", "false positive" and "false negative" respectively. It is important to include precision, recall and F-score (f_1) in the evaluation of the models because the data set is unbalanced with much more "negative" (zeros) than "positive" (ones). These metrics perform better than accuracy in evaluating whether the model is doing well on predicting the ones.

3.1 Logistic Regression

We begin our discussion with simple linear model, logistic regression [Hastie et al., 2005]. The original dimensionality of the feature space is quite high, we choose to use l_1 regularization term to select important features. The logistic regression model in *scikit-learn* [Pedregosa et al., 2011] is defined as

$$\min_{\boldsymbol{w},c} \frac{1}{2}\boldsymbol{w}^\mathsf{T}\boldsymbol{w} + C\sum_{i=1}^{n} \log\left(\exp(-y_i(\boldsymbol{X}_i^\mathsf{T}\boldsymbol{w} + c) + 1\right), \tag{2}$$

where C is the regularization parameter, lower C gives more weights to the regularization term, resulting in sparser result, but might underfit the data. Higher C will include more features, but might expose to the danger of overfitting.

Ranging C from 0.01 to 10 equally in the log space, we use a five-fold stratified cross validation to ensure the number of positive labels approximately the same in each fold. We summarized the number of nonzero features, the mean training accuracy and mean cross validation accuracy in table 1. To have better understanding of the model on each label, we also summarized the learning behavior by confusion matrix in figure 2 for three representative regularization parameters. At the last, in figure 3, we summarize the training accuracy and testing accuracy as well as the +/- 2% standard deviation of training and cross validation accuracy as a function of regularization parameter and number of data points we use (learning curve).

From the summary, we see the learning behavior of logistic regression classifier under different regularization parameter: small C picks smaller amount of features, but the model tends to under fit the data (the left end of figure 3 shows both training and testing scores are low), while increasing C selects more features, but the model might overfit the data (the right end shows training score is quite high, but the testing score is low). An interesting observation from the confusion matrix shows more complex model tends to predict more positive labels. This might

regularizer C	10^{-2}	$10^{-1.67}$	$10^{-1.33}$	$10^{-1.0}$	$10^{-0.67}$	$10^{-0.33}$	10^{0}	$10^{0.33}$	$10^{0.67}$	10^{1}
# of selected features	28	65	151	236	327	395	434	464	487	493
training accuracy	0.866	0.878	0.909	0.935	0.955	0.969	0.980	0.990	0.998	1.000
validation accuracy	0.844	0.873	0.895	0.907	0.911	0.909	0.900	0.885	0.882	0.879

Table 1: Summary of logistic regression classifier, with total number of feature input 500

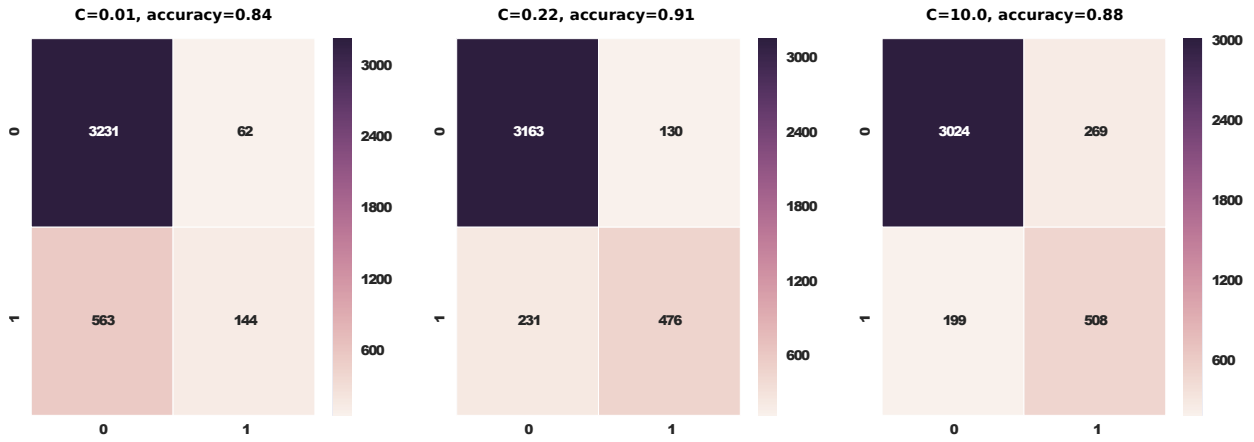

Figure 2: Confusion matrix of corss validation with different regularization parameter C: larger C will result in more positive prediction (predicting 1)

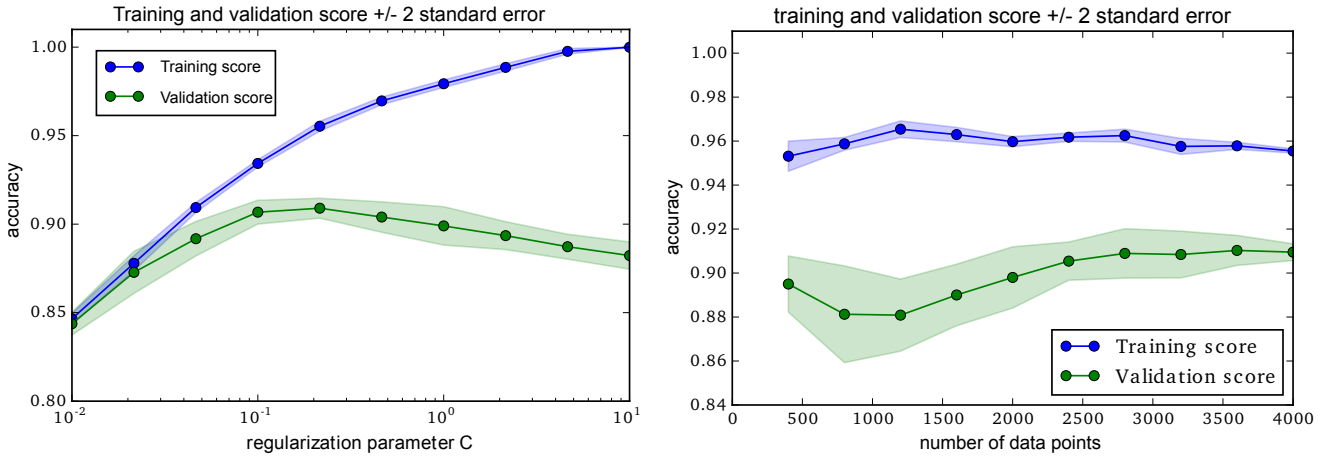

Figure 3: Training and cross validation score of logistic regression versus different regularization parameter C (left) and number of data points used(learning curve)

due to the fact that more negative label presented than positive label, when model is overfitting, it will create more complex boundary to get all the positive label right, but lack the ability to generalize.

Since logistic regression only decides a linear boundary in the feature space, the best validation score we can achieve is 91.1%, which might not be as attractive as the more complex models we will discuss later. However, logistic regression with l_1 regularization help us to eliminate unimportant features thus give us a cleaner space to work in, from which we might gain better chance of generalization. In later section, we will use the selected features and train more powerful models. In figure 3, we plotted the learning curve of logistic regression (score vs size of data set). Increasing the data set will generally decrease the training score but increase the performance of our model.

3.2 Support Vector Machine

We have tested support vector machine classification using the radial basis function (RBF) kernel. The RBF kernel is the most commonly used kernel, so it is tested first. We used the C-SVM classification module (C-SVC) in the *LIBSVM* [Chang and Lin, 2011]. For a general C-SVC, the following optimization problem is solved:

$$\begin{aligned} \min_{w,b,\zeta} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \zeta_i \\ \text{subject to } y_i(w^T \phi(x_i) + b) \geq 1 - \zeta_i, \\ \zeta_i \geq 0,\ i = 1, \ldots, n, \end{aligned} \tag{3}$$

where $x_i \in R^m$, $i = 1, \ldots, n$ are the n training vectors (data points) each of which having m features, and $y \in R^n$ is the vector that contains the prediction such that $y_i \in \{-1, 1\}$. The kernel function $K(x_i, x_j) \equiv \phi(x_i)^T \phi(x_j)$ is $K(x_i, x_j) = \exp(-\gamma \parallel x_i - x_j \parallel^2)$, $\gamma > 0$ for RBF. In LIBSVM, a scaled version of the dual problem of the above optimization problem is solved.

To allow our problem be solved by LIBSVM, we have replaced all 0 in y with -1, and have scaled the training data x and test data z based on

$$x_{\text{scale}ij} = \frac{x_{ij} - \min_i x_{ij}}{\max_i x_{ij} - \min_i x_{ij}} \tag{4}$$

and

$$z_{\text{scale}ij} = \frac{z_{ij} - \min_i x_{ij}}{\max_i x_{ij} - \min_i x_{ij}}, \tag{5}$$

where $i = 1, \ldots, n$, $j = 1, \ldots, m$. To train C-SVC, we need to determine the values of C and γ. We carried out a grid search of the two parameters and evaluated their performance in a 5-fold cross validation. We used the metric F-score in the grid search because a high F-score requires both precision and recall to be high. For convenience, we refer to the data with prediction -1 as class 0, and data with prediction 1 as class 1.

In addition, the unbalanced dataset may require a weighted C to be used in the model in order to penalize the false prediction of class 1 data (fn) more than class 0 (fp). The weighted C is defined as a vector C_{w} such that $C_{\text{w}i} = C \times w_C$ for all i that have $y_i = 1$, and $C_{\text{w}j} = C$ for all other data points j. The objective function of Eq (3) is now $\frac{1}{2} w^T w + C_{\text{w}}^T \zeta$. For an unbalanced dataset that has many more class 0 data points than class 1 data points, the common selection of w_C is n_0/n_1 where n_0 is the number of data points in class 0, n_1 is the number of data points in class 1. For our training set, $w_C = n_0/n_1 \approx 4.66$ but this number gives recall much larger than precision for the set of C and γ that gives high F-score during the grid search. This means this choice of w_C is too large. Therefore, we have also tried the smaller numbers 3.22, 2.33 and 1.50 and determined 2.33 is best to balance precision and recall at high F-score values.

The grid search results are plotted in Figure 4. From the coarse grid search which has a step size of 2 for both $\log_2 C$ and $\log_2 \gamma$ (Search1), the ranges of $\log_2 C$ and $\log_2 \gamma$ that give the highest F-score are around $[3, 7]$ and

$[-6, -3]$ respectively. Therefore, we did a finer grid search with a step size of 1 on the parameter space that includes this range (Search2), and further with a step size of 0.25 (Search3). Search3 does not greatly improve F-score comparing with Search2, and it requires higher C and γ values which are theoretically more likely to overfit than smaller C and γ values. Therefore, we plot the validation curves in Figure (5) based on the best C and γ values from Search2.

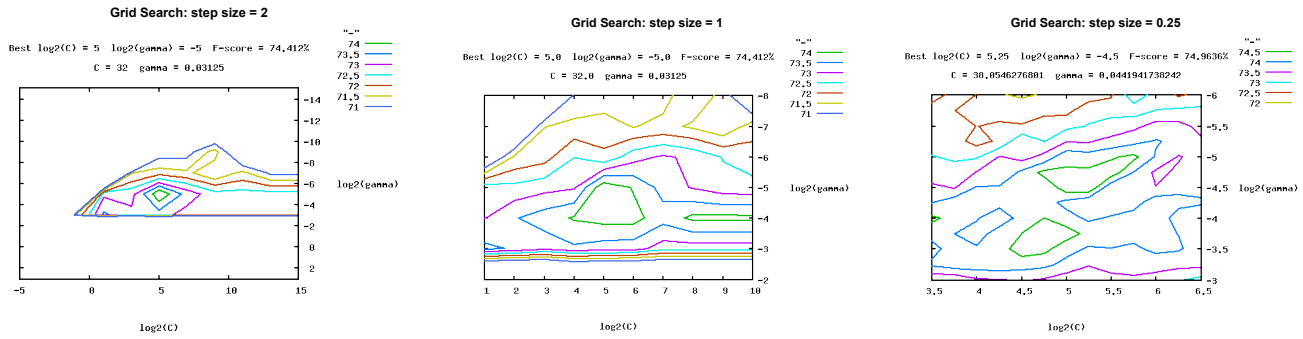

Figure 4: Grid search of C and γ scored by F-score. The step sizes are for both $\log_2 C$ and $\log_2 \gamma$. The x- and y- axis limits of the plots are the ranges of $\log_2 C$ and $\log_2 \gamma$ considered in the search.

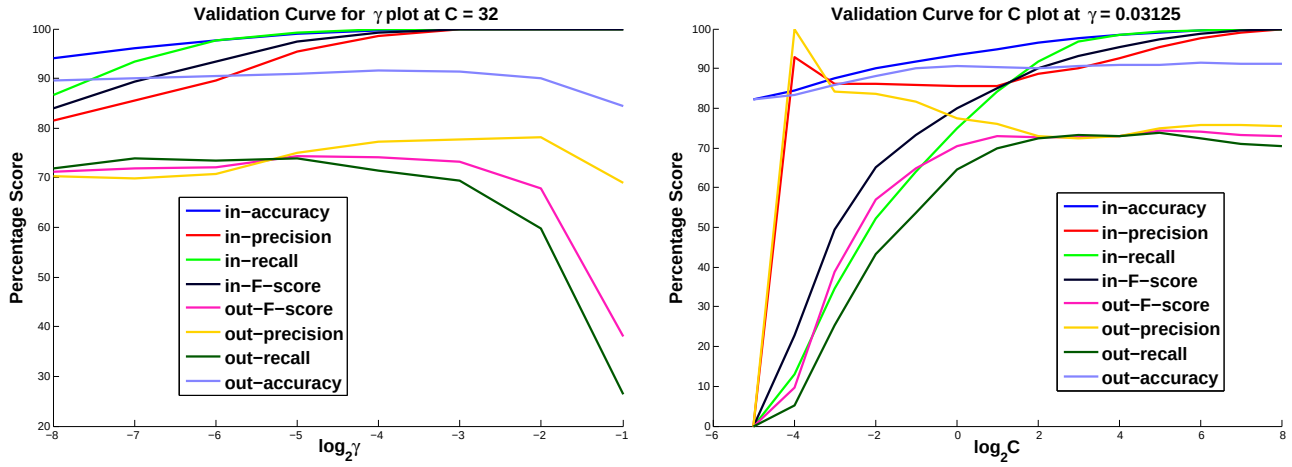

Figure 5: Validation curves for C and γ with different metrics. In the legend "in" means in-sample, "out" means results from 5-fold cross validation.

Figure 5 shows that, for a fixed C ($C = 2^5 = 32$ in this case), the difference between the in-sample and out-of-sample errors increases for all metrics as γ increases. This means the model becomes more overfitting as γ increases. Although the in-sample errors increase monotonically for the range of γ we plotted, the out-of-sample F-score peaks at around $\gamma = 2^{-5} = 0.03125$ where a balance of precision of recall is achieved. This means $\gamma = 0.03125$ should indeed be (step-size-allowed) optimal at $C = 32$. In addition, since the in-sample errors are always much greater than the out-of-sample errors for all metrics, there is a systematic overfitting for the range of (C, γ) pairs we choose in this plot.

For a fixed γ ($\gamma = 0.03125$ in this case), the in-sample and out-of-sample accuracy both increase as C increases until $C \approx 2^{-1} = 0.5$. In addition, the difference between them keeps being small until C becomes greater than 0.5. Therefore, we conclude that the model is underfitting when C is smaller than 0.5. After C becomes greater than 0.5, the out-of-sample accuracy starts to flatten out while the in-sample accuracy keeps increasing. This suggests that

the model becomes overfitting when C is greater than 0.5. A similar trend is found in F-score and recall, although they flatten out at around $C = 2^2 = 4$ to $2^5 = 32$. For precision, the out-of-sample score even decreases after C becomes greater than 2^{-2} while the in-sample score increases, which is a strong indication that the model becomes overfitting when C is big. One may suggest that we take $C = 0.5$ as the optimal C value when $\gamma = 0.03125$. However, as the out-of-sample F-score still increases until it peaks at $C = 32$ and the best balance between precision and recall is achieved around this value, we take $C = 32$ as the optimal.

In order to overcome the overfitting behavior of our model, we replace the original data set with those that only have subsets of features selected by logistic regression as described in the subsection above. We have scaled these data in the same way that we scale the original data. Then, we did a grid search for each of these data sets with a step size of 1 to find the best combination of C and γ for each of them. The in-sample and out-of-sample accuracy values are plotted against the number of selected features in Figure 12. The numbers are also tabulated in Table 2. As an example, we have picked the case with 327 selected features (labeled as "C4") to plot the validation curves on (Figure 6) in this report. We find that feature selection indeed decrease the gap between in-sample and out-of-sample errors by increasing the out-of-sample accuracy and F-score. We plot the validation curve at $C = 4$ and $\gamma = 0.125$ because they give the best F-score in grid search of the two parameters (Figure 7).

Summarizing the accuracy of the model trained with each of these data sets, the best we get is 92.4% from the data set with 236 features. However, the data set with 327 features may be more ideal as it has a similar performance in accuracy and in F-score, and has a smaller C value which means it is less likely to overfit the data. We should be cautious about this selection, however. This particular selection may not be the best in reality because there is randomness in cross validation scores, and the grid selection in the grid search of C and γ values cannot be infinitely small.

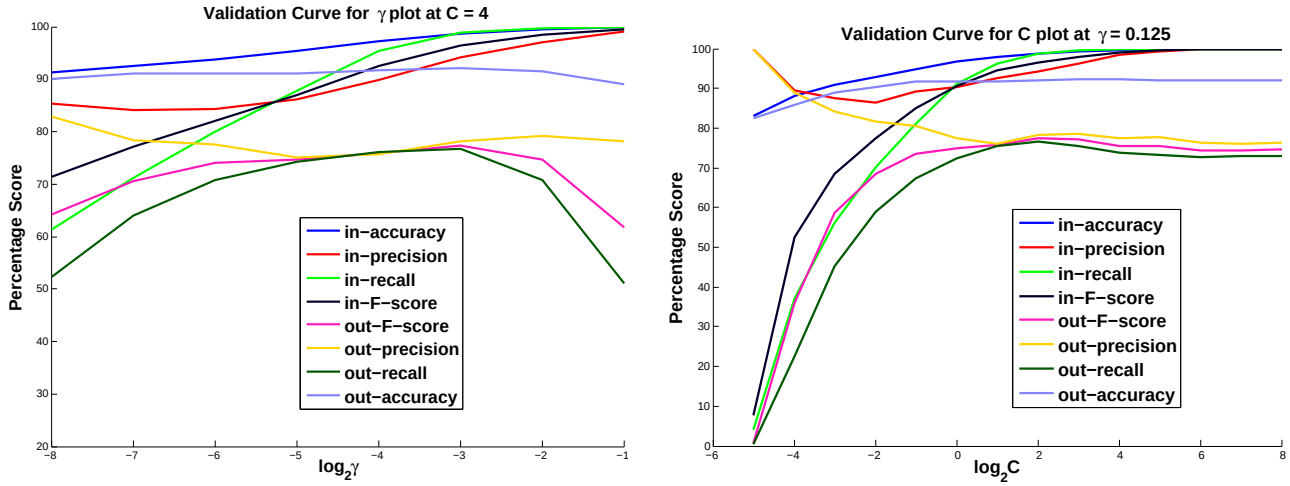

Figure 6: Validation curves for C and γ with different metrics. The data has been subject to feature selection C4. In the legend "in" means in-sample, "out" means results from 5-fold cross validation.

# of features	28	65	151	236	327	395	434	464	487	493	500
C	2	2	8	8	4	8	16	64	512	16	32
γ	2	0.5	0.125	0.125	0.125	0.125	0.125	0.0625	0.0625	0.0625	0.03125
out-F-score	0.7345	0.7510	0.7584	0.7760	0.7743	0.7584	0.7548	0.7518	0.7447	0.7424	0.7441
out-accuracy	0.9158	0.9115	0.9210	0.9240	0.9203	0.9175	0.9180	0.9190	0.9145	0.9160	0.9103

Table 2: Summary of SVM models trained with data with different number of selected features. "Out-F-score" is out-of-sample F-score, "out-accuracy" is out-of-sample accuracy. The scores are rounded to 4 decimal places.

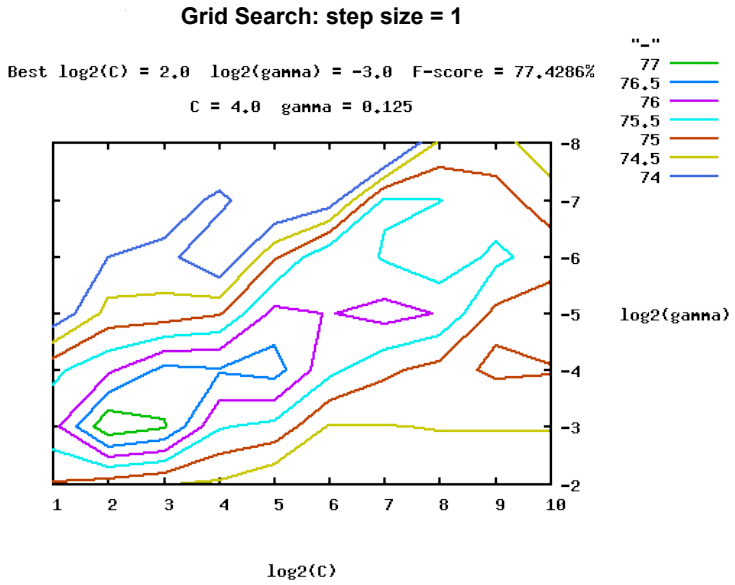

Figure 7: Grid search of C and γ on data subject to feature selection C4 scored by F-score. The step size is for both $\log_2 C$ and $\log_2 \gamma$. The x- and y- axis limits of the plot are the ranges of $\log_2 C$ and $\log_2 \gamma$ considered in the search.

3.3 Gradient Boosting Classifier

Gradient Boosting Trees Classifier [Schapire, 2003] is a generalization of boosting to arbitrary differentiable loss functions. It is an accurate and effective off-the-shelf procedure that can be used for both regression and classification problems and is robust to outliers. In general, gradient boosting classifier would outperform any other classical classifier. In the following discussion, we use the default built in loss function (0-1 loss) to train the classifier.

In section 3.1, we have selected the most important features using logistic regression, the best performed logistic regression model have selected 327 features. Here we use the selected data set to train the gradient boosting classifier.

We have investigated the effects of three parameters on gradient boosting classifier: the number of boosting stage to perform ("n_esimator"), the max depth of individual regression estimator ("max_depth") and the minimum number of samples required to split an internal node ("min_samples_leaf"). After performing grid search on those three parameters, the set of parameter resulting in best mean cross validation accuracy is n_estimator=1000, max_depth=4, min_samples_leaf=9, resulting in cross validation accuracy 0.9375, which is the best accuracy score we have gotten among our attempts. The effects of different parameters are plotted in figure 8. To get this plot, we vary principle parameters we are interested while fixing other two with the value getting the best performance. The most influential parameter is "n_estimator", the number of boosting stage to perform. Smaller number leads to underfitting. However, the model is fairly robust to overfitting, since we don't see any performance drop when the number of stages is large. On the other hand, since we fix the number of stages to be the optimal value (1000), the other two parameters are not very influential (on the plot we see a nearly horizontal line).

In figure 9, we study the effect of reduced dimensionality on the prediction. The features are selected using both l_1 norm regularized logistic regression and PCA. The left figure show the training and cross validation accuracies using different feature spaces selected by logistic regression. From the plot, we see there is not much improvement when we used higher dimensional feature space. Using logistic regression serves as an effective preprocessing approach to

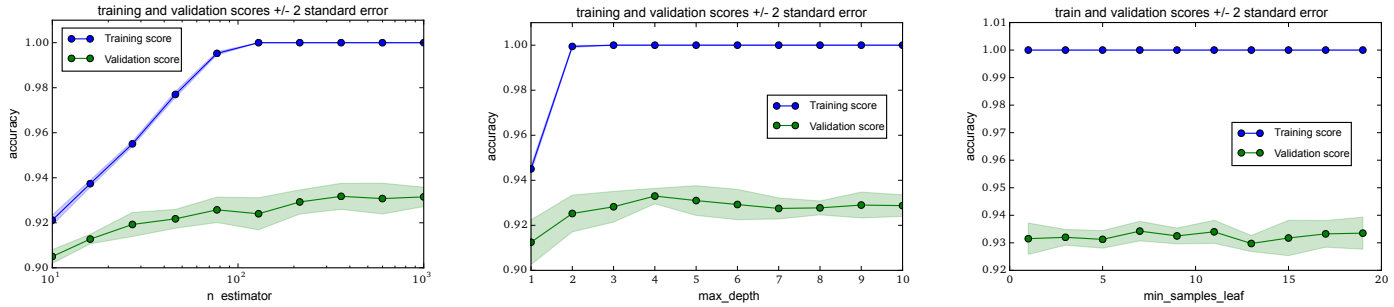

Figure 8: Learning behavior of gradient boosting classifier with respect to different parameters, trained using 327 features selected by logistic regression

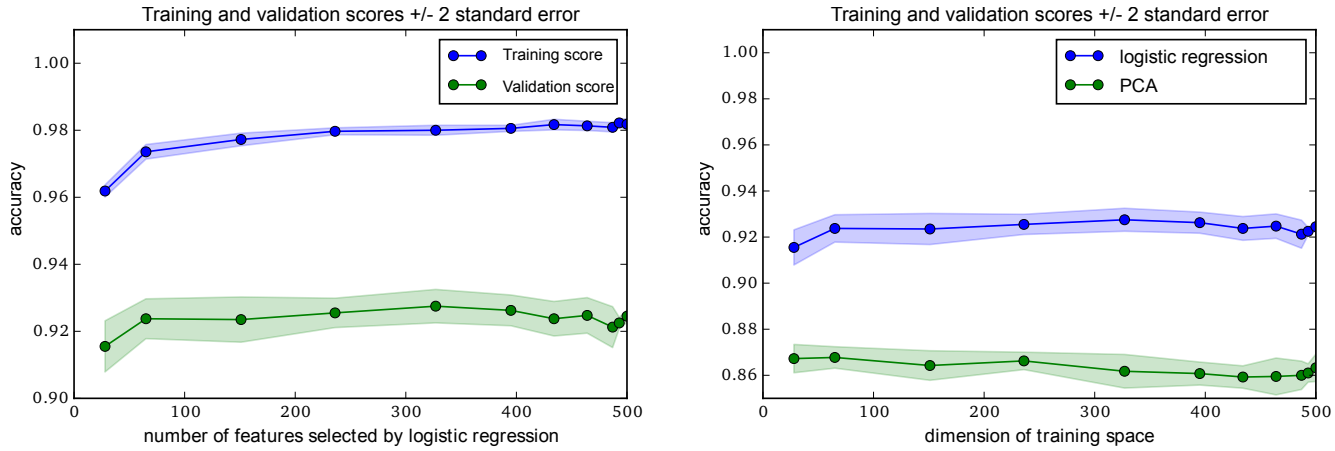

Figure 9: Summary of training error and validation error of gradient boosting classifier using different number of feature input selection by logistic regression (left) and comparing with training in the PCA selected feature space

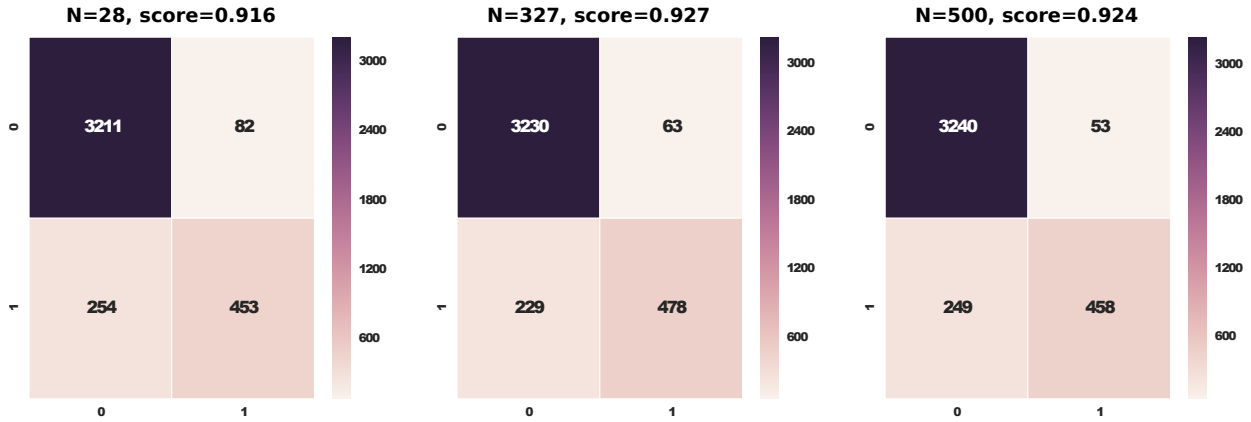

Figure 10: Confusion matrix with different number of feature inputs selected by logistic regression in section 3.1

# of selected features	28			327			500		
metric	precision	recall	f_1	precision	recall	f_1	precision	recall	f_1
negative	0.93	0.97	0.95	0.94	0.98	0.96	0.94	0.98	0.96
positive	0.82	0.67	0.74	0.89	0.72	0.80	0.89	0.71	0.79
average	0.91	0.92	0.91	0.93	0.93	0.93	0.93	0.93	0.93

Table 3: Summary of different scoring function under gradient boost classifier with different number of selected features by logistic regression in section 3.1

eliminate unnecessary features. The left figure shows the cross validation score using feature selected by both logistic regression and PCA, from where we see the model fitted in the space selected using PCA perform much worse. This observation is probably due to the nature of gradient boosting classifier, since every decision is made based on the original feature space. PCA pick the first few dimensions that have largest variance, which may be a combination of the original coordinates thus harming the performance the classifier. This can also be seen at the left end of the plot. When training with the whole feature space only rotated by PCA. The green line still remains very low.

Figure 10 and table 3 summarizes three representative cases using 28, 327 and 500 features by confusion matrix and different metrics. Increasing the feature space gains us more correct positive prediction. In this project, we treat positive and negative prediction equally, i.e. using 0-1 loss function, while in real practice, practical loss function might be defined, resulting in a model lean to make more false negative prediction than false positive prediction or vice versa. And the summary might also give a meaning direction on how to choose the effective feature subset to meet the real world demand.

The learning curve of gradient boosting classifier is shown in figure 11. We use the parameters and feature set resulting in best cross validation accuracy to train the model using different size of data set. From figure 11, we see the training accuracy stays at 1 since we use large number of stages ("n_estimator") in training. As in the logistic regression case, The cross validation score is positively related to the size of training data, since more data points generally leads to better prediction, agreeing with our observation.

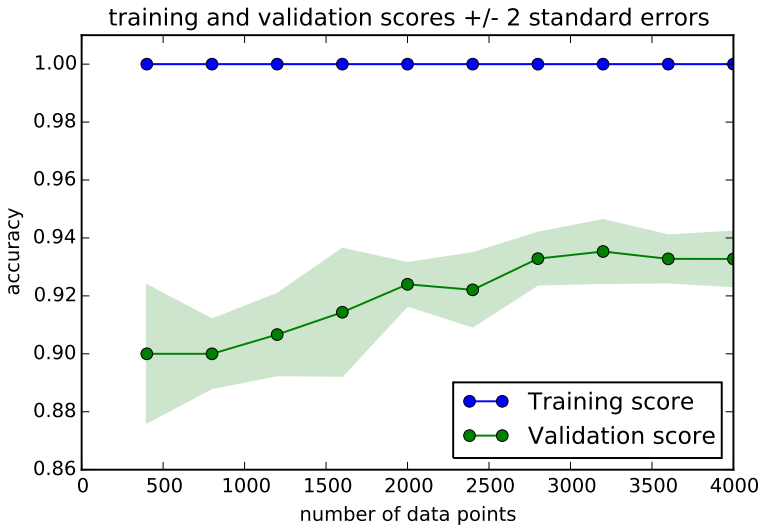

Figure 11: Learning curve of gradient boosting classifier with best parameters

4 Model Selection

So far we have discussed three popular models, logistic regression, SVM and gradient boosting classifier. The cross validation score as a function of input number of the features are plotted in figure 12 for each model. We also include other two models, decision tree and random forest. From the figure, we can see most models perform very robust under the reduced feature space. Using logistic regression preprocessing the data and for feature selection is an effective approach to reduce computational cost and deal with overftting.

Comparing the performance of each model, we choose to report the gradient boosting classifier as our final model. This is because gradient boosting classifier has the greatest out-of-sample accuracy. We should thus expect that the accuracy on the test data is similar to the out-of-sample accuracy reported here. It is important to note that our choice is based on classification accuracy because Kaggle calculates the score based on accuracy. However, if there are any certain practical concerns, we recommend re-evaluating the model using different scoring metrics and make choices correspondingly.

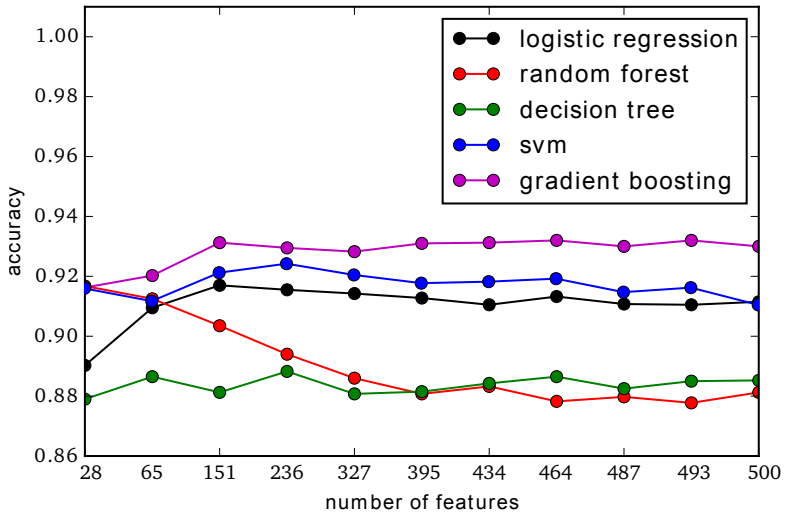

Figure 12: Summary of different learning models: shows the gradient boosting classifier perform best among all the model been considered

5 Conclusion

In conclusion, we have developed a model to predict financial crisis era based on a bag of words from quarterly report of financial institutes. The model we picked is able to predict with more than 94% accuracy in the test set on the public leaderboard, and has an out-of-sample accuracy around 93.75% from 5-fold cross validation.

Several possible improvements can be made for further study. For example, a larger data set is always helpful in increasing the accuracy of a model. Nearly all the models requires parameter search. Using finer grids could always help refine the model and achieve a better score, howevering exposing us very high computational cost. The SVM might be improved by adjusting the weight w_C, and gradient boosting classifier might be adjusted by changing the learning rate.

References

Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

Trevor Hastie, Robert Tibshirani, Jerome Friedman, and James Franklin. The elements of statistical learning: data mining, inference and prediction. *The Mathematical Intelligencer*, 27(2):83–85, 2005.

Harold Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6):417, 1933.

Karl Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

RobertE. Schapire. The boosting approach to machine learning: An overview. In DavidD. Denison, MarkH. Hansen, ChristopherC. Holmes, Bani Mallick, and Bin Yu, editors, *Nonlinear Estimation and Classification*, volume 171 of *Lecture Notes in Statistics*, pages 149–171. Springer New York, 2003. ISBN 978-0-387-95471-4. doi: 10.1007/978-0-387-21579-2_9. URL http://dx.doi.org/10.1007/978-0-387-21579-2_9.