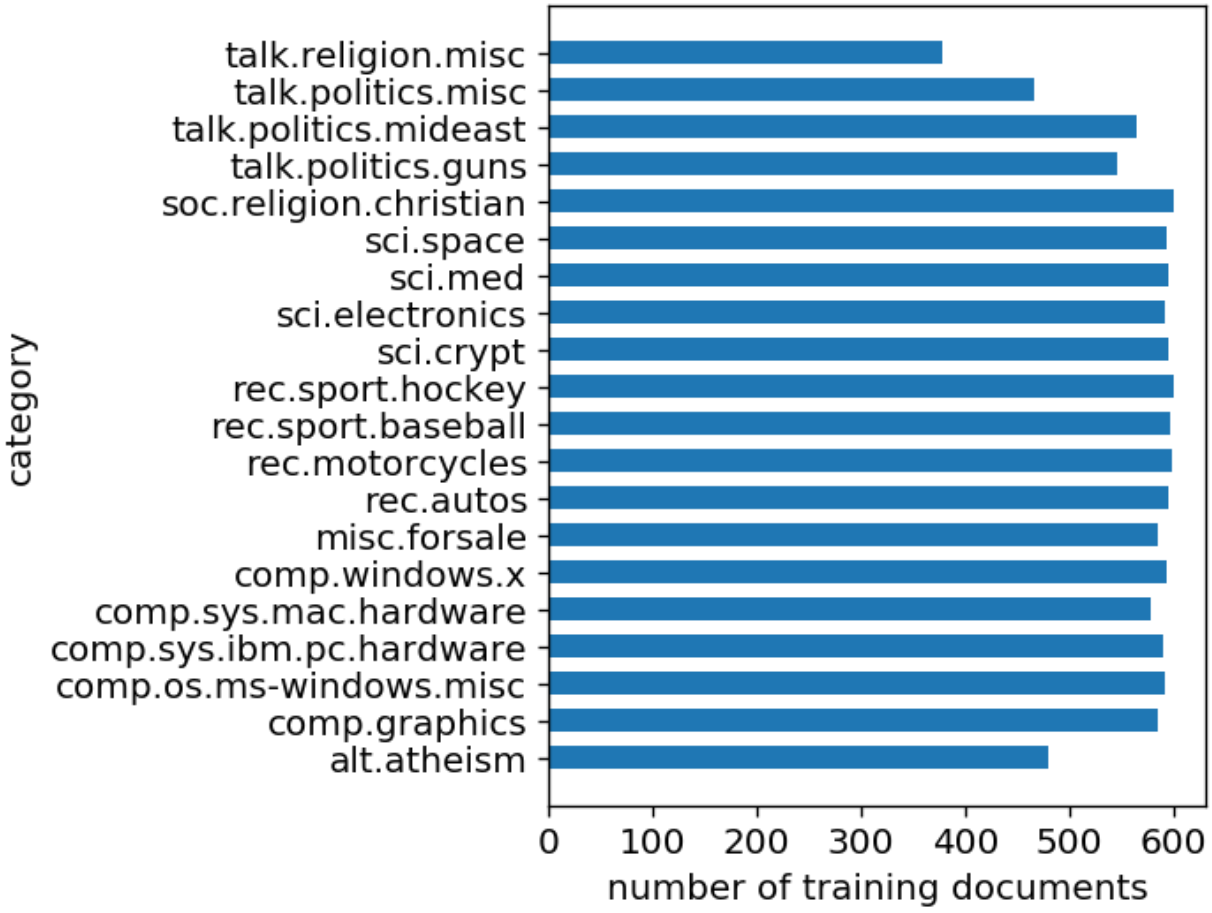
Large-Scale Data Mining: Models and Algorithms ECE 219 Winter 2019

Xiongfeng Zhu, UID: 404436256
Pei-Shan Chung, UID: 004685318

Project 1: Classification Analysis on Textual Data

QUESTION 1: Plot a histogram of the number of training documents for each of the 20 categories to check if they are evenly distributed.

The number if the training documents for each categories is quite evenly distributed. For number of document in category of ‘talk.religion.misc’’ is relatively lower than others.



QUESTION 2: Use the following specs to extract features from the textual data

- Use the “english” stopwords of the CountVectorizer

The number of stopwords of the CountVectorizer in sklearn.feature_extraction.text is 318. We can also customize stopwords set from sets of nltk.corpus.stopwords and string.punctuation, which have number of words/punctuations of 179 and 32, respectively. Then, the union of the stopwords includes 410 words/punctuations.

- Exclude terms that are numbers (e.g. “123”, “-45”, “6.7” etc.)

Goal: Create a callable function ‘remove_num’ for parameter ‘analyzer’ in CountVectorizer.

Concept flow of the callable function:
Given a document (long string) → CountVectorizer().build_analyzer() → list of words → nltk.pos_tag() → word, tag → nltk.wordnet.WordNetLemmatizer().lemmatize() → lemmatized word → .isdigit() → exclude terms that are all numbers

To exclude words that are numbers, the function .isdigit() is used to identify if terms are numbers. Before excluding terms containing all numbers, we first lemmatize terms using the function of nltk.wordnet.WordNetLemmatizer().lemmatize(). The inputs of the function require word and tag of the word. In order to classifier words into their parts of speech and label them accordingly with tag, the function nltk.pos_tag() is used. The inputs of the pos_tag is a list of words and can be obtained by analyzing a document into list of words using a callable function CountVectorizer().build_analyzer().

- Perform lemmatization with nltk.wordnet.WordNetLemmatizer and pos_tag. Use min_df=3

Code:
```
vectorizer = CountVectorizer(analyzer=remove_num,min_df=3,stop_words='english')
```

Create a class attribute called ‘vectorizer’ of class CountVectorizer by replacing parameter ‘analyzer’ with the callable fuction ‘remove_num’, and use min_df of 3, and stop_words of ‘english’ or customized stopwords described above.

- Shape of TF-IDF matrices of train and test subsets respectively

Use fit() and transform() or fit_transform() function to fit and transform training datasets to CountVectorizer() and TfidfTransformer() in sequence. The terms in the TF-IDF is maintained for the test dataset. For test datasets, only function transform() is used to transform datasets to the established object of CountVectorizer and TfidfTransformer.

If stopwords 'english' is used:

Shape of TF-IDF matrix of train subset: (11314, 32942)

Shape of TF-IDF matrix of test subset: (7532, 32942)

If customized stopwords is used (including sklearn and nltk)

Shape of TF-IDF matrix of train subset: (11314, 32942)

Shape of TF-IDF matrix of test subset: (7532, 32942)

If customized stopwords is used (including sklearn, nltk, and punctuation)

Shape of TF-IDF matrix of train subset: (11314, 32942)

Shape of TF-IDF matrix of test subset: (7532, 32942)

The CountVectorizer.build_analyzer() ignore punctuation already. A customized stopwords (including sklearn and nltk) doesn't decrease the total terms in train datasets although it includes 378 words instead of 318 words from sklearn.

QUESTION 3: Reduce dimensionality of data using methods of LSI and NMF

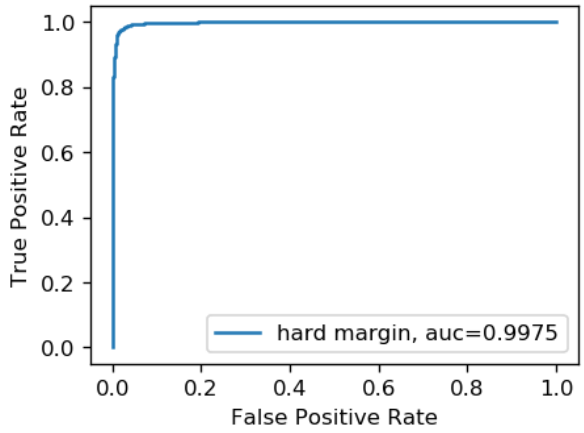
The dimensionality of the above TF-IDF vectors ranges in the order of thousands. However, learning algorithms may perform poorly in high-dimensional data. Here, the purpose of Latent Semantic Indexing (LSI) and Non-negative Matrix Factorization (NMF) is to reduce dimensionality and can reconstruct original data from its low-dimensional approximation. So each document is mapped to a 50-dimensional vector (k=50).

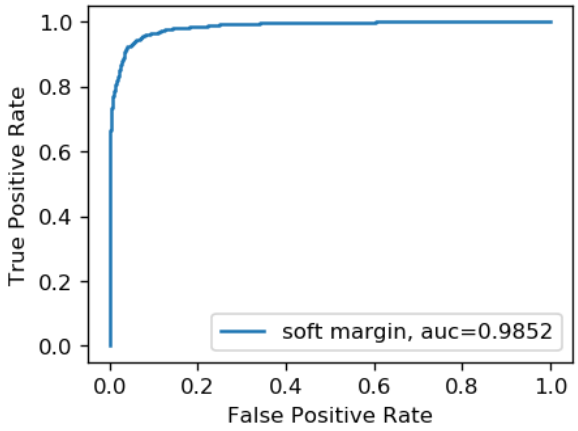
The mean squared residual of NMF and LSI are calculated separately. NMF has larger mean squared residual because it has a stronger non-negative constraint which sacrifices reconstruction accuracy for more interpretability.

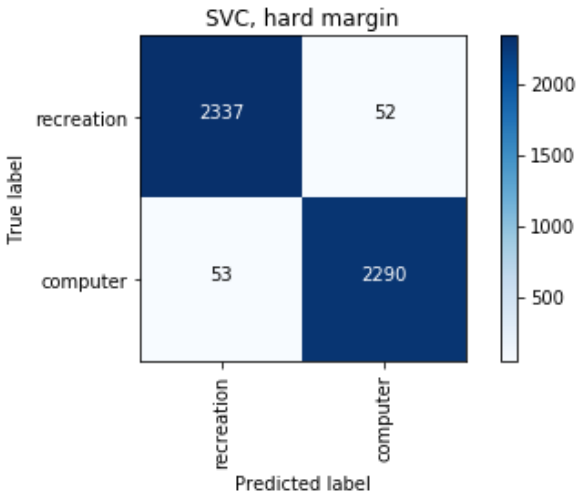
QUESTION 4: Hard margin and soft margin linear SVMs

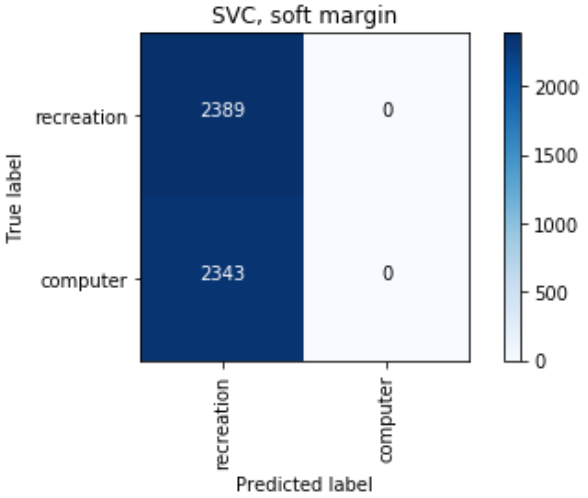
Margin determines how much error SVM can tolerate during the training. With too small a margin SVM does not take penalty on wrongly classified sample, thus can not effectively tell the two classes apart.

SVM only takes 0 as the decision boundary because the decision function is the distance from the learnt hyperplane. So the ROC curve does not really make sense for soft margin SVM because essentially it can not tell them apart with the learnt hyperplane.

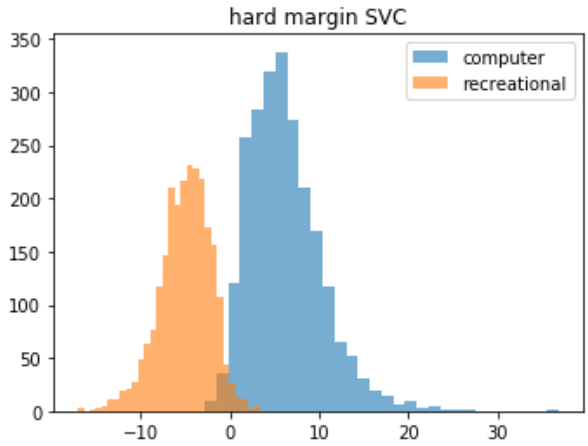


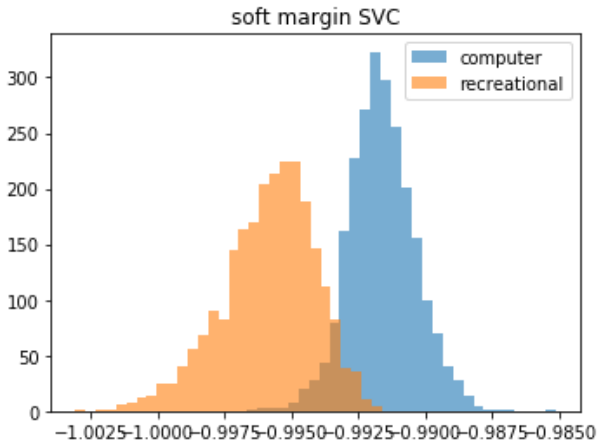




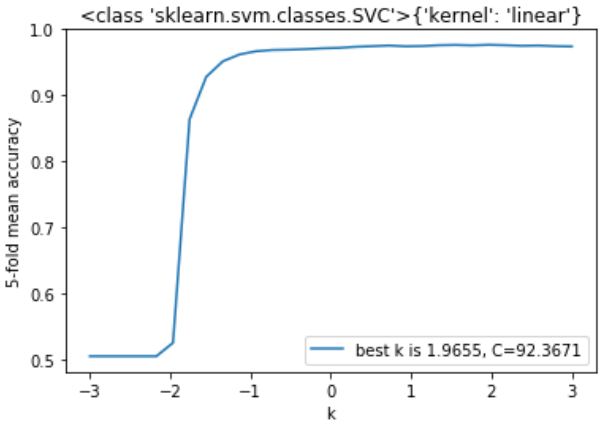


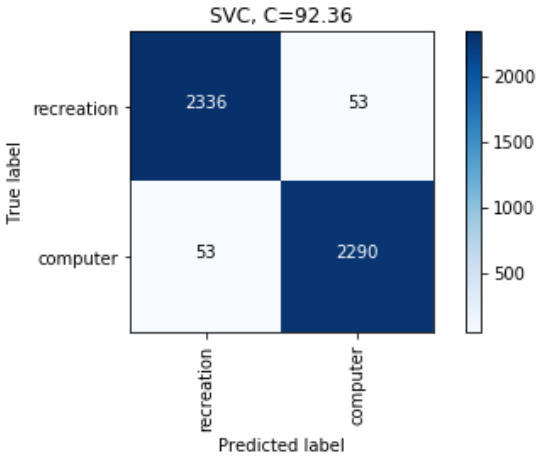
We can see the data distribution after projection by plotting out the decision function. But for soft margin SVM they are all on the same side of the hyperplane.





Using cross-validation we can find the best value for C. The best C is found to be 92.36 in our case.

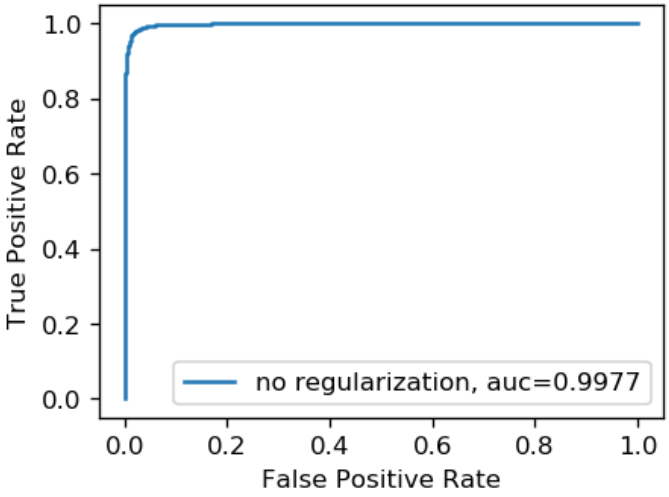




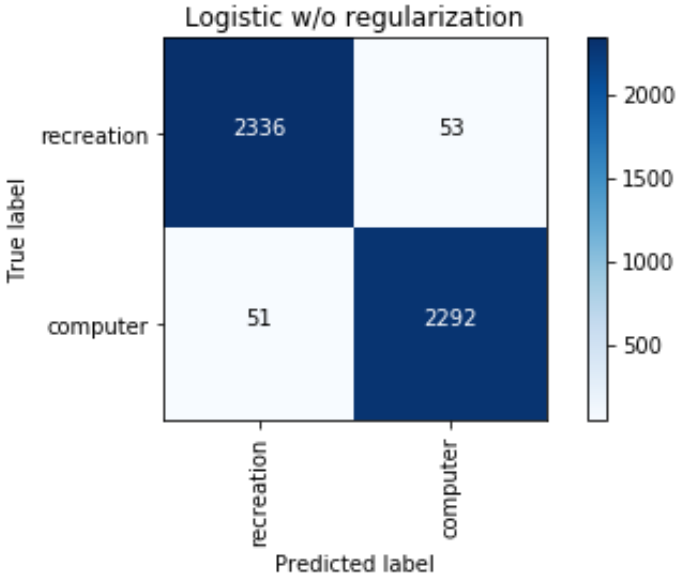
	accuracy	recall	precision	F1-score
Best SVC	0.9776	0.9774	0.9774	0.9774

QUESTION 5: Logistic classifier:

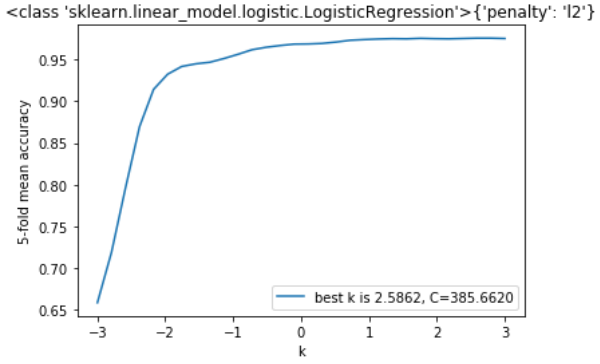
In the logistic regression module from sklearn, the parameter C is the inverse of regularization strength. That means if we want to approximate no regularization (zero strength), then C needs to be a large number, either with L1 or L2 penalty. So we set C to be 1e10. After training on the whole train set we can plot the following ROC curve.

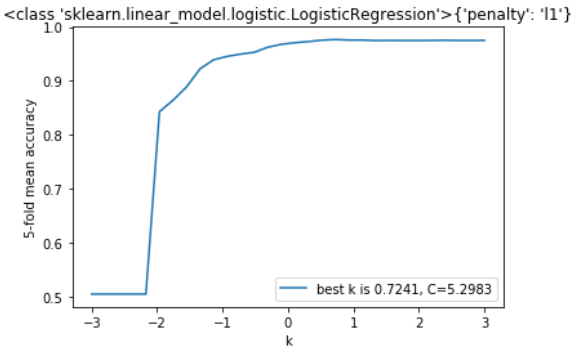


Then we use the trained model to predict. On the train dataset we have accuracy=0.9780, recall=0.9782, precision=0.9774, F_1 score=0.9778.



Using 5-fold cross-validation we can scan through the parameter range for C and find the best one with highest mean test accuracy. We have the following results on test dataset using 3 logistic classifiers.





classifier	accuracy	precision	recall	F-1 score
w/o	0.6787	0.6248	0.8795	0.7306
L1	0.6940	0.6326	0.9115	0.7468
L2	0.6825	0.6269	0.8865	0.7345

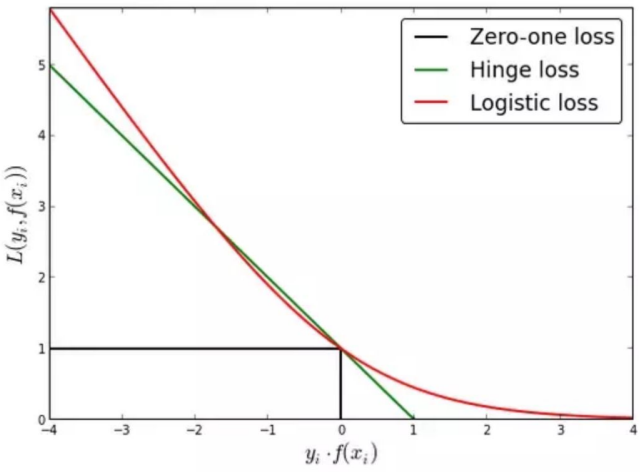
Generally with a very small regularization, classifier tends to overfit on the training dataset, so the error will be high. When the regularization increases, it will limit the magnitude of the coefficients to a smaller range. When regularization gets bigger (C smaller), in the case of L1 regularization it leads to sparse coefficient vectors with a few higher values, in other words it shrinks the less important feature's coefficient to zero thus, removing some feature altogether, which can be used for feature selection. While for L2 when C is smaller, it gives us smaller coefficients in general to avoid overfitting. But when C gets really small, it can lead to under-fitting.

For logistic regression and SVM, there are 2 differences to note:

Logistic loss diverges faster than hinge loss. So, in general, it will be more sensitive to outliers, while SVM can ignore those outliers by simply only looking at support vectors.

Logistic loss does not go to zero even if the point is classified sufficiently confidently. This might lead to minor degradation in accuracy.

> That is, they only differ in the loss function—**SVM minimizes hinge loss while logistic regression minimizes logistic loss.**

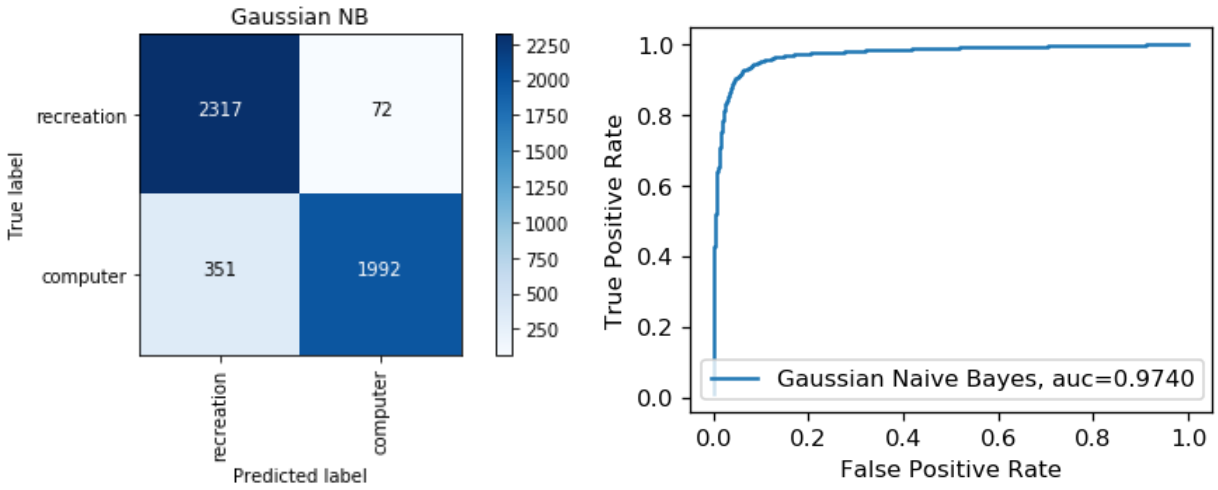


Loss functions

We can typically expect SVM to perform marginally better than logistic regression. And with a dataset favoring overfitting, the regularized models perform much better.

QUESTION 6: Naïve Bayes classifier

We used Gaussian Naive Bayes classifier model from sklearn. It assumes a Gaussian distribution of the data. The results are as follows



	accuracy	recall	precision	F-1 score
Train	0.9106	0.8502	0.9651	0.9040
Test	0.6473	0.7519	0.6183	0.6786

QUESTION 7: Grid search of parameters:

The purpose of the pipeline is to assemble several steps that can be cross-validated together while setting different parameters. It sequentially applies a list of transforms and a final estimator on the dataset. The steps we have include

1) CountVectorizer, with different min_df settings and different analyzers
2) The same tf-idf transformer shared by all pipelines
3) Dimension reduction, either using LSI(truncatedSVD) or NMF with 50 components
4) Classifier, options are linear SVC, logistic regression with different regularization, and Gaussian Naive Bayes model

The different options are set as the param_grid of a GridSearchCV for our pipeline. We use shuffled 5-fold cross validation with the seed of 42 to be consistent. This whole process is repeated twice for 1) loading data and remove header and footer and 2)

loading data without removing headers and footers respectively. Each run takes ~100min on my laptop with n_jobs=-1 enabled.

	param_clf	param_reduce_dim	param_vect__analyzer	param_vect__min_df	mean_fit_time	mean_test_score
0	SVC(C=92.36, cache_size=200, class_weight=None...	TruncatedSVD(algorithm='randomized', n_compone...	<function remove_num at 0x000001FBE904DD90>	3	281.953489	0.976543
1	LogisticRegression(C=5.3, class_weight=None, d...	TruncatedSVD(algorithm='randomized', n_compone...	<function remove_num at 0x000001FBE904DD90>	3	119.215585	0.976543
2	LogisticRegression(C=385.66, class_weight=None...	TruncatedSVD(algorithm='randomized', n_compone...	<function remove_num at 0x000001FBE904DD90>	3	110.951114	0.976120
3	SVC(C=92.36, cache_size=200, class_weight=None...	TruncatedSVD(algorithm='randomized', n_compone...	<function remove_num at 0x000001FBE904DD90>	5	263.688606	0.975909
4	LogisticRegression(C=5.3, class_weight=None, d...	TruncatedSVD(algorithm='randomized', n_compone...	<function remove_num at 0x000001FBE904DD90>	5	109.872530	0.975909
5	LogisticRegression(C=385.66, class_weight=None...	TruncatedSVD(algorithm='randomized', n_compone...	<function remove_num at 0x000001FBE904DD90>	5	109.470761	0.975063
6	LogisticRegression(C=5.3, class_weight=None, d...	TruncatedSVD(algorithm='randomized', n_compone...	<function remove_num_no_lemma at 0x000001FBE90...	5	8.152935	0.973795

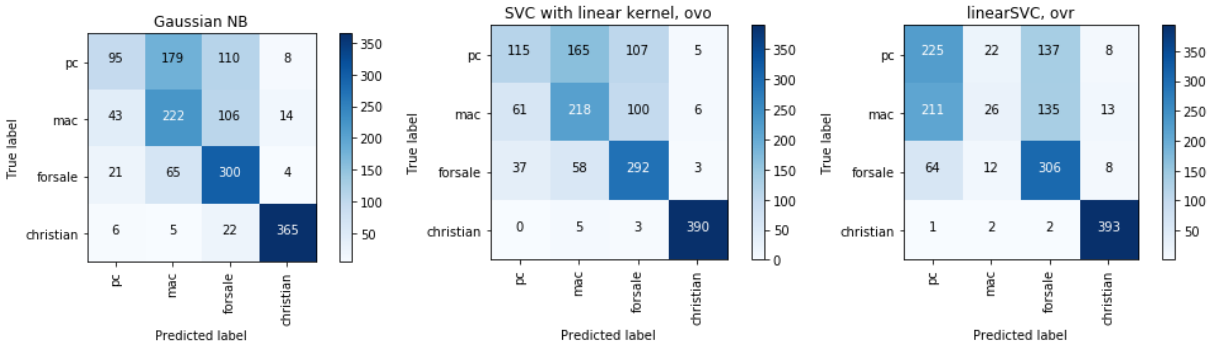
Fig. top 6 pipelines with highest mean test score and their parameters

After training, the best estimator is determined by picking the one with highest mean_test_accuracy during cross-validation. Then the best pipeline is fit on the whole train set again, then used to predict on the test set. The predicted results are compared with ground truth of the test set to get the final accuracy. We achieved 96.9% accuracy on the test set with the following combination

1) Min_df = 3, analyzer = remove numbers, and with lemmatization
2) Dimensionality reduction with LSI (truncatedSVD) of 50 components
3) SVM classifier with C=92.36 and linear kernel

QUESTION 8: Multi-class prediction

In terms of multiclass classification, Naive Bayes model can handle it natively just by calculating the probability for each class then predict the one with largest probability. SVM can have two strategies which are One vs One or One vs the rest. Here we compare their performance. The transformation steps are the same as the ones found in the best combination for binary classification, we only vary the estimator in the last step. Metrics are set to ‘macro’ which calculates metrics for each label, and find their unweighted mean. This does not take label imbalance into account.



classifier	accuracy	recall	precision	F1-score
GNB	0.6275	0.6263	0.6346	0.6079
SVC, ovo	0.6486	0.6471	0.6439	0.6330
linearSVC, ovr	0.6070	0.6025	0.5877	0.5374