

Assignment 3

Hemant Pandey

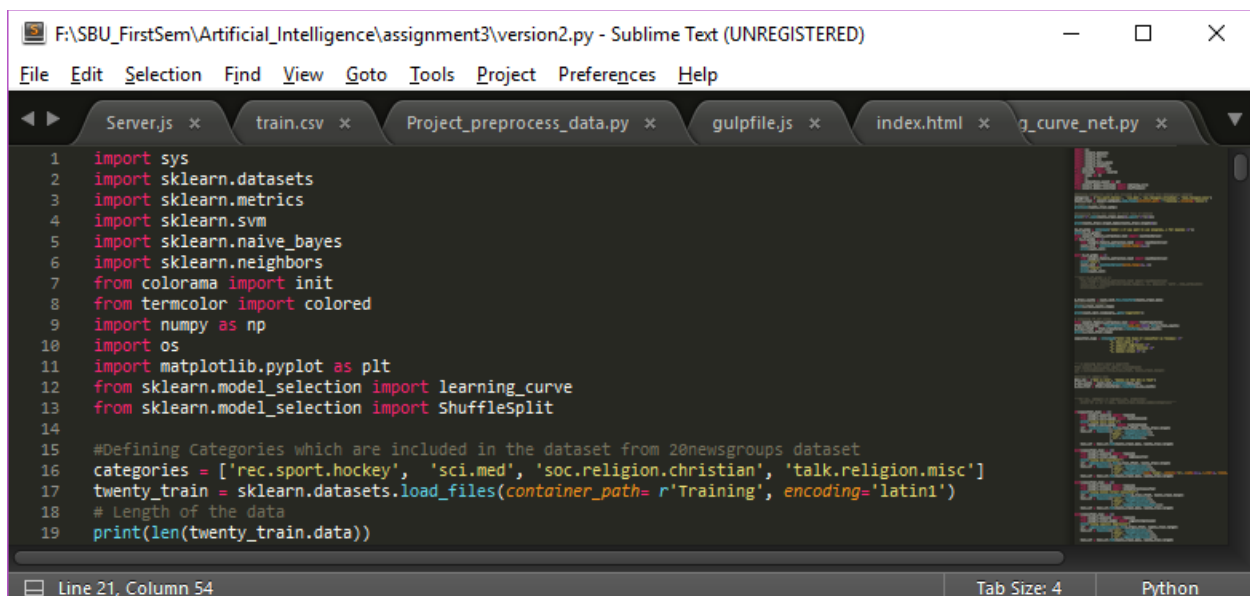
110828730

INTRODUCTION AND COMPARISON

The purpose of this assignment is to investigate the utility of different learning algorithms for a text classification task. We have used the implementations available in the scikit python library:

http://scikit-learn.org/stable/supervised_learning.html#supervised-learning

We are using the dataset from 20 newsgroup which contains up to 20 categories. Out of these 20, we have been using 4 categories which we will be using as classes. We are provided with training and test data of all the cases. Using the `load_files()` method of `sklearn.datasets`, we are loading the files.

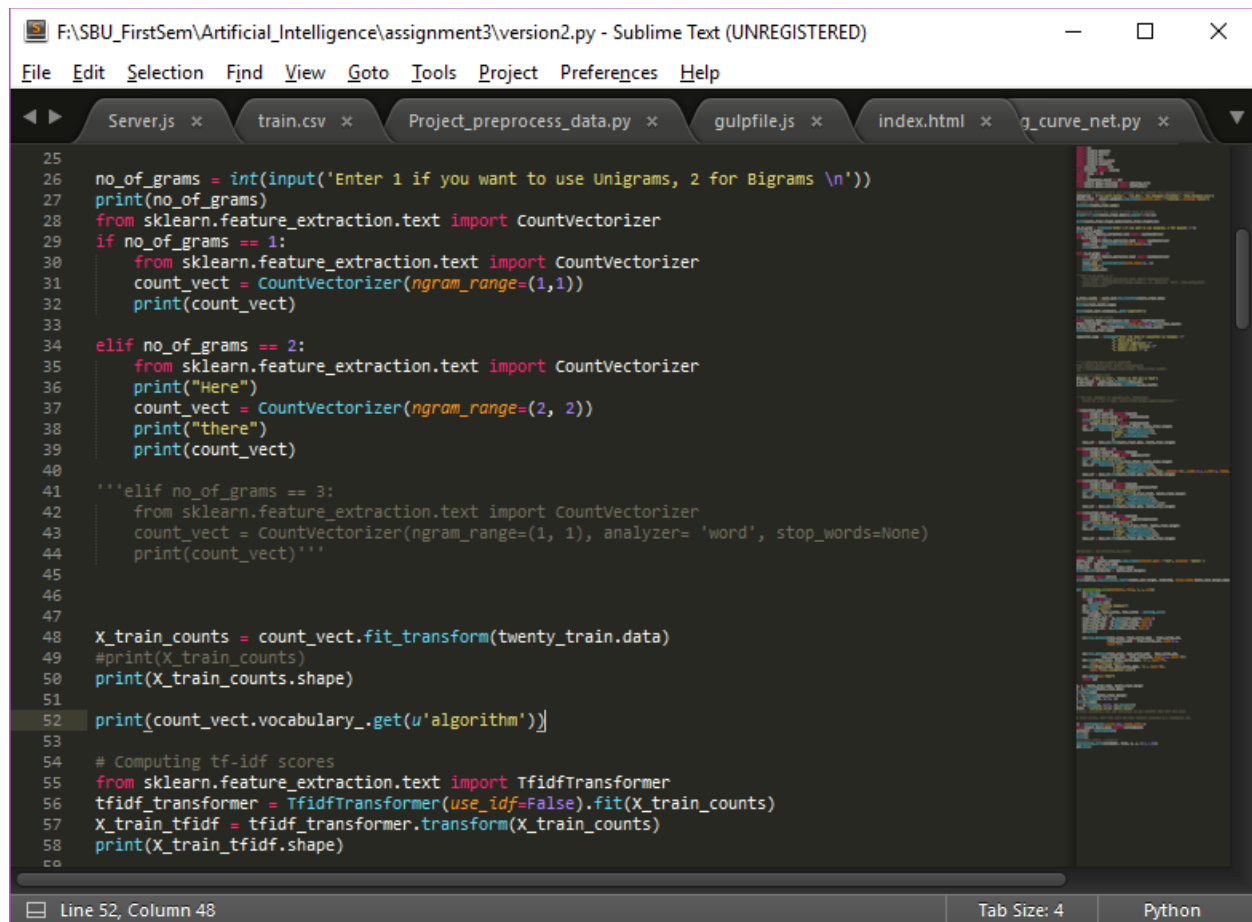


```
F:\SBU_FirstSem\Artificial_Intelligence\assignment3\version2.py - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
Server.js x train.csv x Project_preprocess_data.py x gulpfile.js x index.html x g_curve_net.py x
1 import sys
2 import sklearn.datasets
3 import sklearn.metrics
4 import sklearn.svm
5 import sklearn.naive_bayes
6 import sklearn.neighbors
7 from colorama import init
8 from termcolor import colored
9 import numpy as np
10 import os
11 import matplotlib.pyplot as plt
12 from sklearn.model_selection import learning_curve
13 from sklearn.model_selection import ShuffleSplit
14
15 #Defining Categories which are included in the dataset from 20newsgroups dataset
16 categories = ['rec.sport.hockey', 'sci.med', 'soc.religion.christian', 'talk.religion.misc']
17 twenty_train = sklearn.datasets.load_files(container_path= r'Training', encoding='latin1')
18 # Length of the data
19 print(len(twenty_train.data))
Line 21, Column 54 Tab Size: 4 Python
```

Now, I have used the `CountVectorizer()` method of `sklearn.feature_selection.text` and set the range of `n_grams` to deal with unigrams and bigrams.

Then, I have used the `TfidfTransformer()` method of `sklearn.feature_selection.text` and set the desired parameters.

The code can be seen in the below screenshot :



```
25 no_of_grams = int(input('Enter 1 if you want to use Unigrams, 2 for Bigrams \n'))
26 print(no_of_grams)
27 from sklearn.feature_extraction.text import CountVectorizer
28 if no_of_grams == 1:
29     from sklearn.feature_extraction.text import CountVectorizer
30     count_vect = CountVectorizer(ngram_range=(1,1))
31     print(count_vect)
32
33
34 elif no_of_grams == 2:
35     from sklearn.feature_extraction.text import CountVectorizer
36     print("Here")
37     count_vect = CountVectorizer(ngram_range=(2, 2))
38     print("there")
39     print(count_vect)
40
41 '''elif no_of_grams == 3:
42     from sklearn.feature_extraction.text import CountVectorizer
43     count_vect = CountVectorizer(ngram_range=(1, 1), analyzer= 'word', stop_words=None)
44     print(count_vect)'''
45
46
47
48 X_train_counts = count_vect.fit_transform(twenty_train.data)
49 #print(X_train_counts)
50 print(X_train_counts.shape)
51
52 print(count_vect.vocabulary_.get(u'algorithm'))
53
54 # Computing tf-idf scores
55 from sklearn.feature_extraction.text import TfidfTransformer
56 tfidf_transformer = TfidfTransformer(use_idf=False).fit(X_train_counts)
57 X_train_tfidf = tfidf_transformer.transform(X_train_counts)
58 print(X_train_tfidf.shape)
59
60
```

Pipeline() function from sklearn.pipeline is used for four different classifiers :

- 1.) Naïve Baye's
- 2.) Logistic Regression
- 3.) Stochastic Gradient Descent
- 4.) Random Forest

The code can be seen in the screenshot on the next page.

The test data is loaded in similar way and observations are made.

Then the precision recall metrics of each classifier using both unigram and bigram is made and following observations are made.

```

version2.py - [C:\Users\Hemant\AppData\Local\Temp\version2.py] - F:\SBU_FirstSem\Artificial_Intelligence\assignment3\version2.py - PyCharm Community Edition 2016.2.3
File Edit View Navigate Code Refactor Run Tools VCS Window Help
version2.py x My_best_conf.py x text.py x
95 if(classifier_type==1):
96     from sklearn.pipeline import Pipeline
97     from sklearn.naive_bayes import MultinomialNB
98     print("Using Naive Bayes")
99     from sklearn.naive_bayes import MultinomialNB
100     clf = MultinomialNB().fit(X_train_tfidf, twenty_train.target)
101     text_clf = Pipeline([('bag', CountVecorizer(ngram_range=(2,2))),
102                          ('tfidf', TfidfTransformer()),
103                          ('clf', MultinomialNB()),
104                          ])
105     text_clf = text_clf.fit(twenty_train.data, twenty_train.target)
106
107 elif(classifier_type==3):
108     from sklearn.pipeline import Pipeline
109     from sklearn.svm import LinearSVC
110     print("Using Linear SVC Classifier")
111     clf = LinearSVC().fit(X_train_tfidf, twenty_train.target)
112     text_clf = Pipeline([('vect', CountVecorizer(ngram_range=(1,1))),
113                          ('tfidf', TfidfTransformer()),
114                          ('clf', LinearSVC())])
115     text_clf = text_clf.fit(twenty_train.data, twenty_train.target)
116
Run version2
Using SGD Classifier
0.929460580913
precision    recall  f1-score   support
rec.sport.hockey      0.98      1.00      0.99         200
sci.med               0.95      0.96      0.96         198
soc.religion.christian 0.86      0.96      0.91         199
talk.religion.misc    0.95      0.71      0.81         126
avg / total          0.93      0.93      0.93         723
Platform and Plugin Updates: PyCharm Community Edition is ready to update. (yesterday 20:09) 108:42 CRLF+ UTF-8+

```

```

F:\SBU_FirstSem\Artificial_Intelligence\assignment3\version2.py - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
Server.js x train.csv x Project_preprocess_data.py x gulpfile.js x index.html x rve_net.py x
127 #predicted = clf.predict(X_new_tfidf)
128
129 import numpy as np
130 twenty_test = sklearn.datasets.load_files(container_path= r'Test', encoding= 'latin1' )
131 docs_test = twenty_test.data
132 predicted = text_clf.predict(docs_test)
133 print(np.mean(predicted == twenty_test.target))
134
135 from sklearn import metrics
136 print(metrics.classification_report(twenty_test.target, predicted, target_names=twenty_test.target_
137
Line 52, Column 48 Tab Size: 4 Python

```

For Unigrams
Naïve Bayes's
Mean=0.816044260028

Class	Precision	Recall	F1- Score
Rec.sport.hockey	0.98	1.00	0.99
Sci.med	0.97	0.86	0.91
Soc.religion.christian	0.61	0.99	0.76
Talk.religion.misc	1.00	0.17	0.29
Avg/Total	0.88	0.82	0.78

Logistic Regression
Mean=0.907330567082

Class	Precision	Recall	F1- Score
Rec.sport.hockey	0.98	1.00	0.99
Sci.med	0.90	0.98	0.94
Soc.religion.christian	0.83	0.97	0.89
Talk.religion.misc	0.99	0.54	0.70
Avg/Total	0.92	0.91	0.90

Support Vector
Mean=0.915629322268

Class	Precision	Recall	F1- Score
Rec.sport.hockey	0.98	1.00	0.99
Sci.med	0.95	0.96	0.96
Soc.religion.christian	0.86	0.96	0.91
Talk.religion.misc	0.95	0.71	0.81
Avg/Total	0.93	0.93	0.93

Random Forest
Mean=0.763485477178

Class	Precision	Recall	F1- Score
Rec.sport.hockey	0.83	0.94	0.88
Sci.med	0.74	0.84	0.78
Soc.religion.christian	0.73	0.78	0.75
Talk.religion.misc	0.72	0.35	0.47

Avg/Total	0.76	0.76	0.75
------------------	------	------	------

For Bigrams

Naïve Baye's
Mean=0.831044260028

Class	Precision	Recall	F1- Score
Rec.sport.hockey	0.96	0.98	0.97
Sci.med	0.94	0.81	0.87
Soc.religion.christian	0.66	0.99	0.79
Talk.religion.misc	0.98	0.37	0.53
Avg/Total	0.88	0.83	0.82

Logistic Regression
Mean=0.861687413555

Class	Precision	Recall	F1- Score
Rec.sport.hockey	0.92	0.98	0.95
Sci.med	0.90	0.87	0.88
Soc.religion.christian	0.76	0.97	0.85
Talk.religion.misc	0.98	0.47	0.63
Avg/Total	0.88	0.86	0.85

Support Vector
Mean=0.892116182573

Class	Precision	Recall	F1- Score
Rec.sport.hockey	0.95	0.98	0.97
Sci.med	0.92	0.89	0.91

Soc.religion.christian	0.86	0.98	0.91
Talk.religion.misc	0.95	0.71	0.81
Avg/Total	0.91	0.91	0.91

Random Forest
Mean=0.721991701245

Class	Precision	Recall	F1- Score
Rec.sport.hockey	0.71	0.88	0.78
Sci.med	0.71	0.68	0.69
Soc.religion.christian	0.71	0.87	0.78
Talk.religion.misc	0.97	0.31	0.47
Avg/Total	0.75	0.72	0.70

The plot for the four classifiers are as :

Red : Naïve Baye's

Blue : Logistic Regression

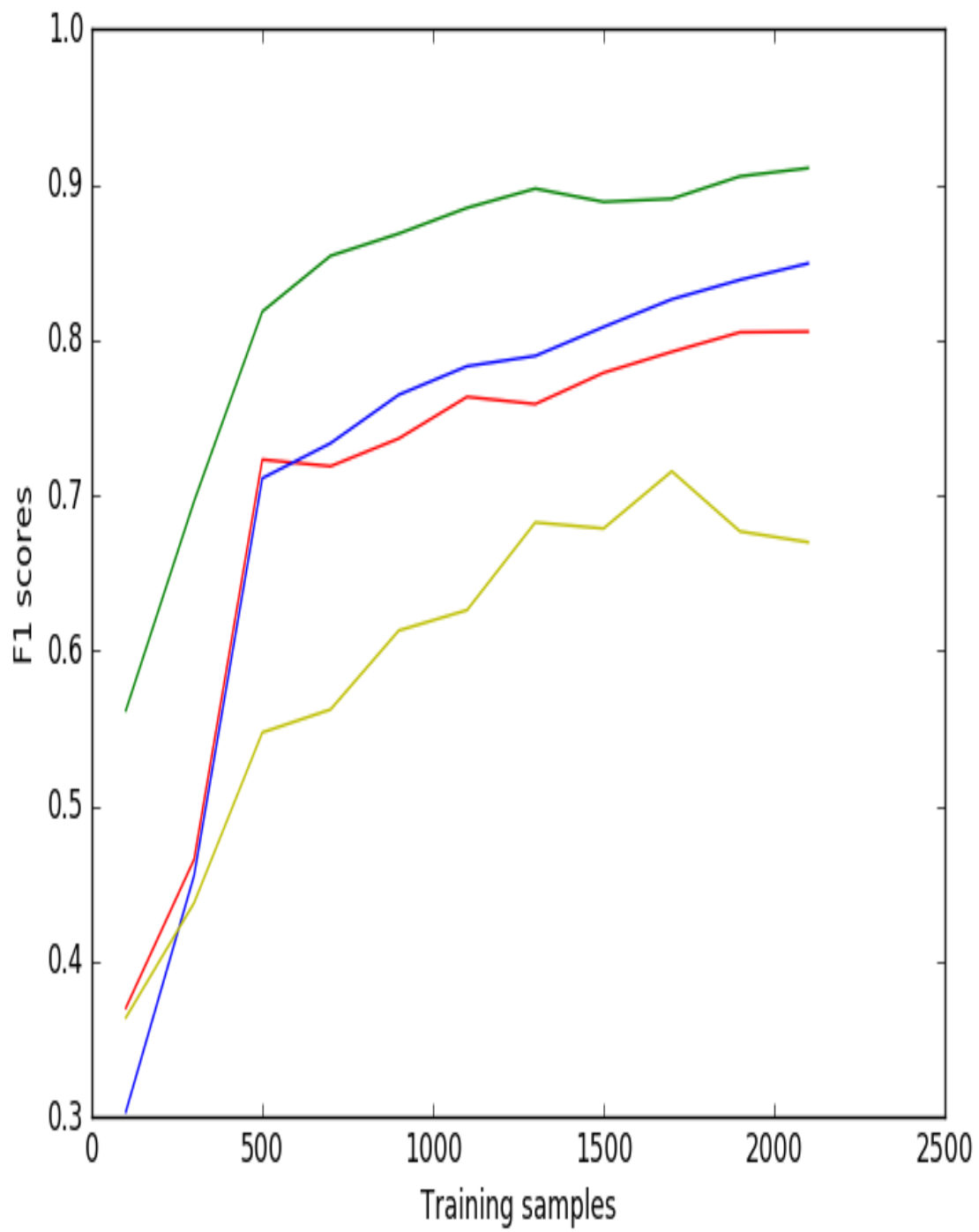
Green : Support Vector Machine

Yellow : Randomized Forest

SVM classifier is suitable in cases where there is a reasonable amount of data. Since SVMs use overfitting protection, which does not necessarily depend on the number of features, they have the potential to handle these large feature spaces.

Naive Bayes classifiers are a family of humble probabilistic classifiers based on applying Bayes' theorem with strong (naive) independence conventions between the features. Random classifier is an ensemble learning method for classification that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. **Logistic regression** is a statistical method for analyzing a dataset in which there are one or more independent variables that determine an outcome. The outcome is measured with a dichotomous variable (in which there are only two possible outcomes).

Since, we got better results in Linear SVC, I haven't extended it to radial dimensions. For a good amount of training dataset and a good number of feature spaces, SVM suits the given dataset much appropriate than the other classifier methods.



Since, the F1 score i.e. the harmonic mean of precision and recall is maximum i.e. 0.91 in case of Support Vector Machines with unigrams, I have used the LinearSVC classifier in my best configuration model.

I have implemented and tested the following 9 configurations

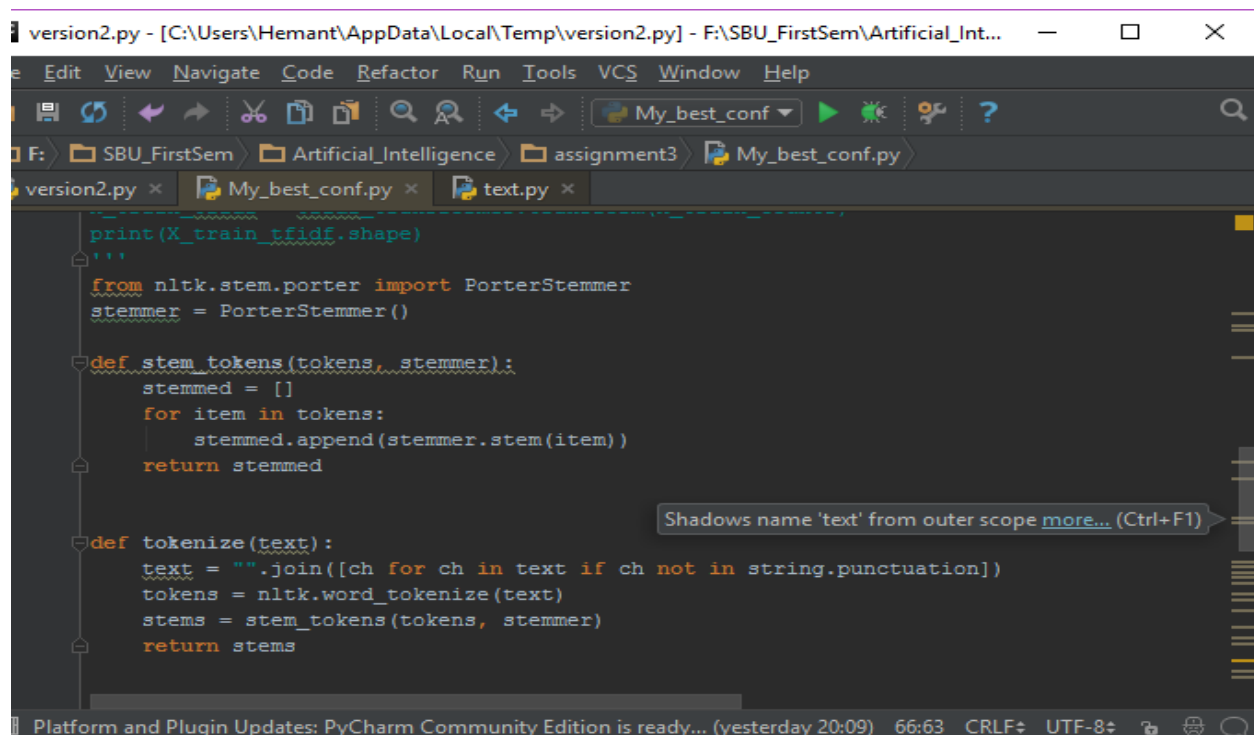
1. Removed Stop words
2. Porter Stemmed
3. Removed Stop words + Porter Stemmed
4. Neither stop words removed nor stemming done
5. L2 + Both Unigrams and Bigrams
6. Both unigrams and bigrams – L2 + Univariate Feature Selection (Chi2)
7. Porter Stemming + Stop Words + L2 + Univariate Feature Selection (Chi2)
8. Only stop words + L2

The last one has given the best results.

9. Adding penalty parameter + L2 + Stop Words

Following are the snapshots of code snippets for different features

I have used the following code which builds a callable analyzer into the Count Vectorizer functions and performs stemming of words and removal of punctuation marks.



```
version2.py - [C:\Users\Hemant\AppData\Local\Temp\version2.py] - F:\SBU_FirstSem\Artificial_Int...
File Edit View Navigate Code Refactor Run Tools VCS Window Help
F:\SBU_FirstSem\Artificial_Intelligence\assignment3\My_best_conf.py
version2.py x My_best_conf.py x text.py x
print(X_train_tfidf.shape)
'''
from nltk.stem.porter import PorterStemmer
stemmer = PorterStemmer()

def stem_tokens(tokens, stemmer):
    stemmed = []
    for item in tokens:
        stemmed.append(stemmer.stem(item))
    return stemmed

def tokenize(text):
    text = "".join([ch for ch in text if ch not in string.punctuation])
    tokens = nltk.word_tokenize(text)
    stems = stem_tokens(tokens, stemmer)
    return stems

Shadows name 'text' from outer scope more... (Ctrl+F1)
Platform and Plugin Updates: PyCharm Community Edition is ready... (yesterday 20:09) 66:63 CRLF UTF-8
```


The 'L1' and 'L2' regularization constant parameter can be modified by changing the penalty parameter in the LinearSVC classifier and adding norm parameter to tfidftransformer.

```

65 length = len(twenty_train.data)
66 value = []
67 from sklearn.pipeline import Pipeline
68 from sklearn.svm import LinearSVC
69 from sklearn.feature_extraction.text import TfidfTransformer
70 print("Using Linear SVC Classifier")
71 for n in range(100, length, 200):
72     text_clf = Pipeline([('vect', CountVectorizer(ngram_range=(1,2), lowercase=True)),
73                          ('tfidf', TfidfTransformer(norm='l2')),
74                          ('clf', LinearSVC(C=1, loss='squared_hinge', dual=False, fit_intercept=True))])
75     text_clf = text_clf.fit(twenty_train.data[1:n], twenty_train.target[1:n])
76     predicted = text_clf.predict(docs_test)
77     print(np.mean(predicted == twenty_test.target))
78     from sklearn import metrics
79     print(metrics.classification_report(twenty_test.target, predicted, target_names=twenty_test.target_names))
80     value.append(metrics.f1_score(twenty_test.target, predicted, average='weighted'))
81
82

```

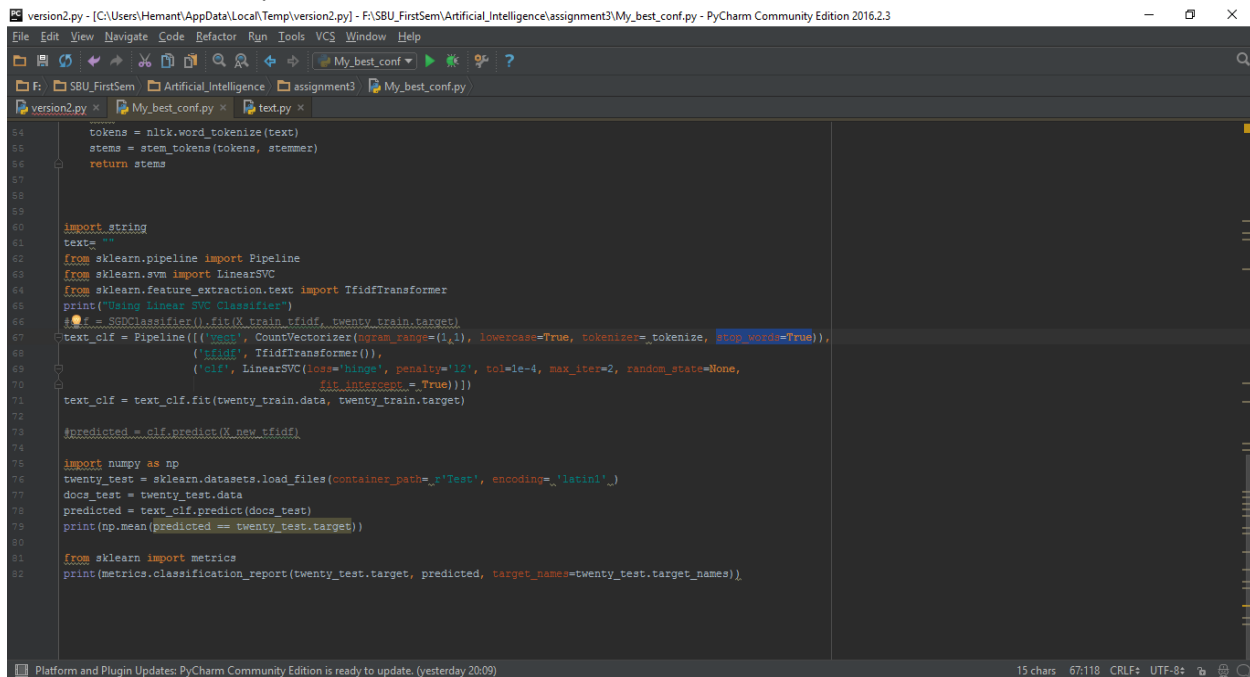
To make all the capital letters into smaller case, we make lowercase=True in CountVectorizer.

```

54 tokens = nltk.word_tokenize(text)
55 stems = stem_tokens(tokens, stemmer)
56 return stems
57
58
59
60 import string
61 text = ""
62 from sklearn.pipeline import Pipeline
63 from sklearn.svm import LinearSVC
64 from sklearn.feature_extraction.text import TfidfTransformer
65 print("Using Linear SVC Classifier")
66 # clf = SGDClassifier().fit(X_train_tfidf, twenty_train.target)
67 text_clf = Pipeline([('vect', CountVectorizer(ngram_range=(1,1), lowercase=True, tokenizer=tokenize, stop_words=True)),
68                     ('tfidf', TfidfTransformer()),
69                     ('clf', LinearSVC(loss='hinge', penalty='l2', tol=1e-4, max_iter=2, random_state=None, fit_intercept=True))])
70 text_clf = text_clf.fit(twenty_train.data, twenty_train.target)
71
72
73 # predicted = clf.predict(X_new_tfidf)
74
75 import numpy as np
76 twenty_test = sklearn.datasets.load_files(container_path='Test', encoding='latin1')
77 docs_test = twenty_test.data
78 predicted = text_clf.predict(docs_test)
79 print(np.mean(predicted == twenty_test.target))
80
81 from sklearn import metrics
82 print(metrics.classification_report(twenty_test.target, predicted, target_names=twenty_test.target_names))

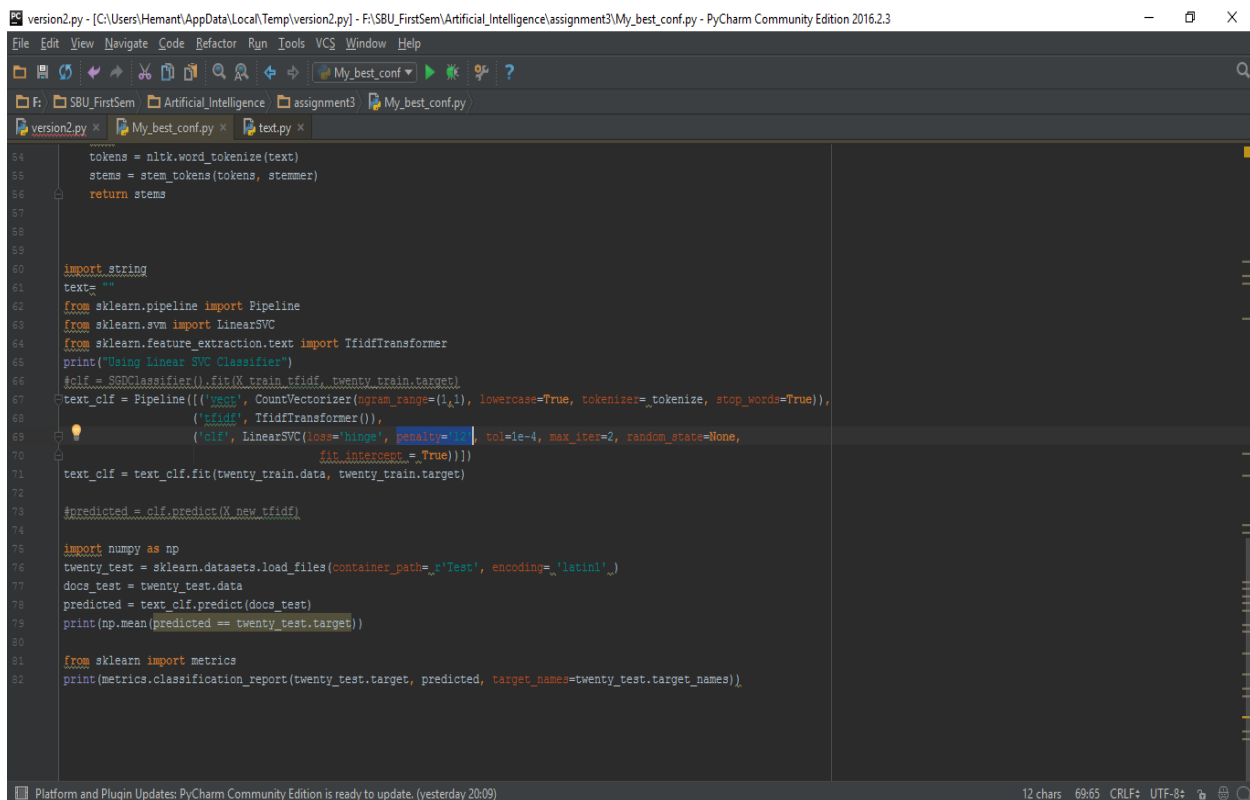
```

To remove all the stop words.



```
version2.py - [C:\Users\Hemant\AppData\Local\Temp\version2.py] - F:\SBU_FirstSem\Artificial_Intelligence\assignment3\My_best_conf.py - PyCharm Community Edition 2016.2.3
File Edit View Navigate Code Refactor Run Tools VCS Window Help
F: SBU_FirstSem Artificial_Intelligence assignment3 My_best_conf.py
version2.py x My_best_conf.py x text.py x
54 tokens = nltk.word_tokenize(text)
55 stems = stem_tokens(tokens, stemmer)
56 return stems
57
58
59
60 import string
61 text = ""
62 from sklearn.pipeline import Pipeline
63 from sklearn.svm import LinearSVC
64 from sklearn.feature_extraction.text import TfidfTransformer
65 print("Using Linear SVC Classifier")
66 #clf = SGDClassifier().fit(X_train_tfidf, twenty_train.target)
67 text_clf = Pipeline([('vect', CountVecorizer(ngram_range=(1,1), lowercase=True, tokenizer=tokenize, stop_word=True)),
68 ('tfidf', TfidfTransformer()),
69 ('clf', LinearSVC(loss='hinge', penalty='l2', tol=1e-4, max_iter=2, random_state=None, fit_intercept=True))])
70 text_clf = text_clf.fit(twenty_train.data, twenty_train.target)
71
72
73 #predicted = clf.predict(X_new_tfidf)
74
75
76 import numpy as np
77 twenty_test = sklearn.datasets.load_files(container_path=r'Test', encoding='latin1')
78 docs_test = twenty_test.data
79 predicted = text_clf.predict(docs_test)
80 print(np.mean(predicted == twenty_test.target))
81
82 from sklearn import metrics
83 print(metrics.classification_report(twenty_test.target, predicted, target_names=twenty_test.target_names))
Platform and Plugin Updates: PyCharm Community Edition is ready to update. (yesterday 20:09) 15 chars 67:118 CRLF+ UTF-8+
```

To change the regularization parameter,



```
version2.py - [C:\Users\Hemant\AppData\Local\Temp\version2.py] - F:\SBU_FirstSem\Artificial_Intelligence\assignment3\My_best_conf.py - PyCharm Community Edition 2016.2.3
File Edit View Navigate Code Refactor Run Tools VCS Window Help
F: SBU_FirstSem Artificial_Intelligence assignment3 My_best_conf.py
version2.py x My_best_conf.py x text.py x
54 tokens = nltk.word_tokenize(text)
55 stems = stem_tokens(tokens, stemmer)
56 return stems
57
58
59
60 import string
61 text = ""
62 from sklearn.pipeline import Pipeline
63 from sklearn.svm import LinearSVC
64 from sklearn.feature_extraction.text import TfidfTransformer
65 print("Using Linear SVC Classifier")
66 #clf = SGDClassifier().fit(X_train_tfidf, twenty_train.target)
67 text_clf = Pipeline([('vect', CountVecorizer(ngram_range=(1,1), lowercase=True, tokenizer=tokenize, stop_word=True)),
68 ('tfidf', TfidfTransformer()),
69 ('clf', LinearSVC(loss='hinge', C=1.0, tol=1e-4, max_iter=2, random_state=None, fit_intercept=True))])
70 text_clf = text_clf.fit(twenty_train.data, twenty_train.target)
71
72
73 #predicted = clf.predict(X_new_tfidf)
74
75
76 import numpy as np
77 twenty_test = sklearn.datasets.load_files(container_path=r'Test', encoding='latin1')
78 docs_test = twenty_test.data
79 predicted = text_clf.predict(docs_test)
80 print(np.mean(predicted == twenty_test.target))
81
82 from sklearn import metrics
83 print(metrics.classification_report(twenty_test.target, predicted, target_names=twenty_test.target_names))
Platform and Plugin Updates: PyCharm Community Edition is ready to update. (yesterday 20:09) 12 chars 69:65 CRLF+ UTF-8+
```

For univariate feature selection,

```

version2.py - [C:\Users\Hemant\AppData\Local\Temp\version2.py] - F:\SBU_FirstSem\Artificial_Intelligence\assignment3\mc.py - PyCharm Community Edition 2016.2.3
File Edit View Navigate Code Refactor Run Tools VCS Window Help
F:\SBU_FirstSem\Artificial_Intelligence\assignment3\mc.py
My_best_conf.py x mc.py x validation.py x version2.py x
Q docs_test
56 def stem_tokens(tokens, stemmer):
57     stemmed = []
58     for item in tokens:
59         stemmed.append(stemmer.stem(item))
60     return stemmed
61
62
63 def tokenize(text):
64     text = "".join([ch for ch in text if ch not in string.punctuation])
65     tokens = nltk.word_tokenize(text)
66     stems = stem_tokens(tokens, stemmer)
67     return stems
68
69 selector = SelectPercentile(chi2, 35)
70 X_new = selector.fit_transform(X, Y)
71 Test_vect = selector.transform(Test_vect)
72
73
74
75 import string
76 length = len(twenty_train.data)
77 value = []
78 from sklearn.pipeline import Pipeline
79 from sklearn.svm import LinearSVC
80 from sklearn.feature_extraction.text import TfidfTransformer
81 print("Using Linear SVC Classifier")
82 for n in range(100, length, 200):
83     text_clf = Pipeline([
84         ('vect', CountVecorizer(ngram_range=(1,2), lowercase=True, stop_words='english')),
85         ('tfidf', TfidfTransformer(norm='l2')),
86         ('clf', LinearSVC(C=1, loss='hinge', dual=False, penalty='l2', tol=1e-6, max_iter=1000, random_state=None,
87                             # is important = True!!!!

```

Now,

1.) Removed Stop words but Porter stemmed

Mean=0.918395573997

Class	Precision	Recall	F1- Score
Rec.sport.hockey	0.98	0.99	0.99
Sci.med	0.95	0.96	0.96
Soc.religion.christian	0.83	0.95	0.89
Talk.religion.misc	0.92	0.67	0.78
Avg/Total	0.92	0.92	0.92

2.) Both Porter Stemmed and Stop words

Mean=0.923928077455

Class	Precision	Recall	F1- Score
Rec.sport.hockey	0.98	0.99	0.98
Sci.med	0.96	0.97	0.96
Soc.religion.christian	0.85	0.95	0.90
Talk.religion.misc	0.93	0.71	0.80
Avg/Total	0.93	0.92	0.92

3.) Removed both stop words and porter stemmed

Mean=0.922544951591

Class	Precision	Recall	F1- Score
Rec.sport.hockey	0.98	1.00	0.99
Sci.med	0.95	0.96	0.96
Soc.religion.christian	0.84	0.96	0.89
Talk.religion.misc	0.94	0.67	0.79
Avg/Total	0.93	0.92	0.92

4.) No Stemming but stop words removed

Mean=0.929460580913

Class	Precision	Recall	F1- Score
Rec.sport.hockey	0.98	1.00	0.99
Sci.med	0.95	0.96	0.96
Soc.religion.christian	0.86	0.95	0.90

Talk.religion.misc	0.92	0.74	0.82
Avg/Total	0.93	0.93	0.93

So the best results are obtained in case 4 when stop words were removed but porter stemmer was not used. Now, I am using the same arguments for count vectorizer and will be manipulating the parameters of LinearSVC to obtain much better results.

5.) Taking both unigrams and bigrams and applying L2 Regularization

Mean=0.9377593361

Class	Precision	Recall	F1- Score
Rec.sport.hockey	0.99	1.00	0.99
Sci.med	0.95	0.97	0.96
Soc.religion.christian	0.88	0.96	0.92
Talk.religion.misc	0.95	0.75	0.84
Avg/Total	0.94	0.94	0.94

6.) Taking both unigrams and bigrams without L2 Regularization and using Univariate Feature Selection (Chi2)

Mean=0.914246196404

Class	Precision	Recall	F1- Score
Rec.sport.hockey	1.00	1.00	1.00
Sci.med	0.91	0.96	0.93
Soc.religion.christian	0.87	0.95	0.91
Talk.religion.misc	0.86	0.64	0.74
Avg/Total	0.91	0.91	0.91

7.) Porter Stemming + Stop Words removed + L2 + Univariate Feature Selection (Chi2)

Mean=0.934993084371

Class	Precision	Recall	F1- Score
Rec.sport.hockey	0.99	1.00	1.00
Sci.med	0.96	0.99	0.98
Soc.religion.christian	0.87	0.95	0.91
Talk.religion.misc	0.92	0.71	0.80
Avg/Total	0.94	0.93	0.93

8.) Only Stop words removed and L2 regularization applied

Mean=0.9377593361

Class	Precision	Recall	F1- Score
Rec.sport.hockey	0.99	1.00	0.99
Sci.med	0.95	0.97	0.96
Soc.religion.christian	0.88	0.96	0.92
Talk.religion.misc	0.95	0.75	0.84
Avg/Total	0.94	0.94	0.94

9.) Adding penalty parameter + L2 Regularization + Only Stop words

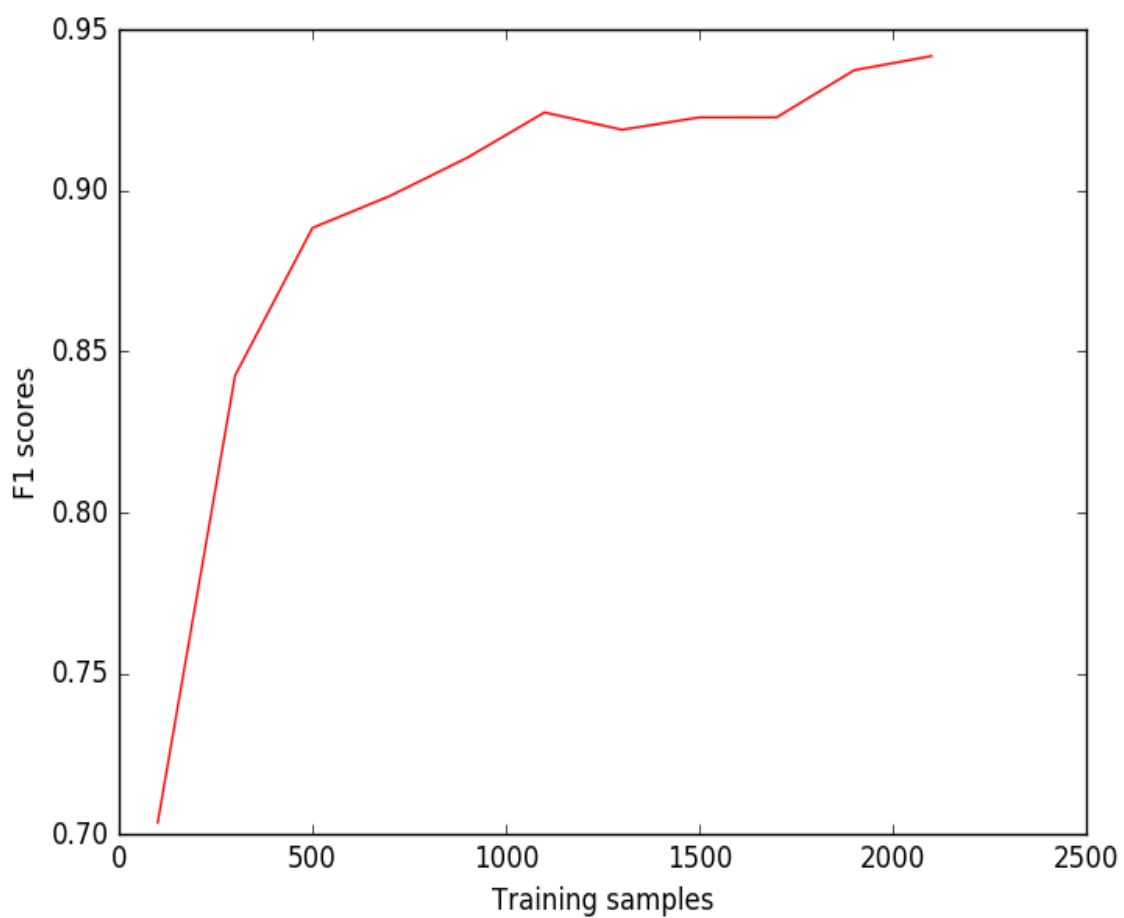
Mean=0.943291839557

Class	Precision	Recall	F1- Score
Rec.sport.hockey	0.98	1.00	0.99
Sci.med	0.96	0.97	0.96

Soc.religion.christian	0.89	0.97	0.93
Talk.religion.misc	0.96	0.77	0.85
Avg/Total	0.95	0.94	0.94

Thus, observing all the possible configurations made the best results were obtained with stop words removal and L2 regularization i.e. F1 score **0.94** and precision **0.95**.

The learning curve is as below:



The changes made to my best configuration are :

```
text_clf = Pipeline([('vect', CountVectorizer(ngram_range=(1,2), lowercase=True,
stop_words='english'))),
                    ('tfidf', TfidfTransformer(norm='l2')),
                    ('clf', LinearSVC(C=1, loss='squared_hinge', dual = False,
penalty='l2', tol=1e-6, max_iter=1000, random_state=None,
fit_intercept = True))])
```

Stop words have been removed.

Using PorterStemmer didn't improved my result.

Adding L2 Regularization Parameter to TfidfTransformer.

Adding penalty parameter improved my result well.

Changing the ngram_range to (1,2)

1.) Parameters of CountVectorizer()

ngram_range defines the range of grams:

(1,1) is used for Unigrams

(2,2) is used for Bigrams

(1,2) is used for both unigrams as well as bigrams

Lowercase = True:

This converts all capital letters to lower case letter.

Stop_words = 'english':

This removes all the stop words.

2.) Parameters for TfidfTransformer()

Norm used to normalize term vectors. None for no normalization.

We have used L2.

3.) Parameters for LinearSVC()

Loss : Specifies the loss function. 'hinge' is the standard SVM loss (used e.g. by the SVC class) while 'squared_hinge' is the square of the hinge loss. We have used L2.

Dual: Select the algorithm to either solve the dual or primal optimization problem. Prefer dual=False when n_samples > n_features.

C : Penalty parameter C of the error term

Tol: Tolerance for stopping criteria.

Max_iter: The maximum number of iterations to be run.

Random_state : The seed of the pseudo random number generator to use when shuffling the data.

My code will run for the datasets at particular intervals, show the report at every step and then finally displays the precision recall table for complete dataset and plot the learning curve.

References

- 1.) http://scikit-learn.org/stable/supervised_learning.html#supervised-learning Sklearn Documentation.
- 2.) http://scikit-learn.org/stable/tutorial/text_analytics/working_with_text_data.html Sklearn Text Documentation