Applied Machine Learning - Midterm 1

Name: Jingyuan Liu; **AndrewId**: jingyual; **email**: jingyual@andrew.cmu.edu

Part 1: Thought Questions (10 points each)

1. What is the difference between weights and per feature conditional probabilities in terms of how they function in a trained model (i.e., a weight based model vs a probabilistic model, respectively)? What is an argument for using probabilities instead of weights? What is an argument against using probabilities instead of weights?

Solution:

The major difference between weights and per feature conditional probabilities in terms of how they function in a trained model is the "scaling". Basically, weight based model would compose the model likelihood and try to optimize the weights to maximize the likelihood. On the other hand, the conditional probabilistic models are computed from likelihoods by scaling in such a way that the sum of probabilities for all possible events within a model will sum to 1

These two models are suitable for different scenarios. For the probabilistic models, it would give statistical models their nice forms because the sum of probabilities sum to 1.

The disadvantage of probabilistic statistical model is that they always depend on assumptions that are not in general true. Besides, in practice, they don't work better than "ad-hoc" methods.

2. Explain the smoothing method we discussed in class in connection with Naïve Bayes. What is the problem with using this form of smoothing with small datasets?

Solution:

Smoothing is to add a pseudo count to each case in the Naïve Bayes counting step. Naïve Bayes model is based on counting, therefore, smoothing is to add "1" to each counts. With this method, Naïve Bayes would be smoothed, which can also be known as adding a prior.

When smoothing with small datasets, it would cause some problems, especially when the prior probabilities are highly skewed. We would simply add 1 to those small datasets (0).

3. Review the slides from Week 5 Lecture 1 where we discussed the Tic Tac Toe dataset. Note that you can find this dataset on blackboard in the Lecture Slides folder near the slides for that lecture. Explain why you think linear models worked better than the other two algorithms.

Solution:

The decision tree model is trained based on splitting features. Basically, in this case, we would have too many "features" and "cases" to split. Therefore, the model is very much likely to be over-fitting. So the model classification performances would not be very high.

Another model is the probabilistic model, Naïve Bayes. This model assumes that the features are conditionally independent on labels. This assumption would not agree with the realistic case in the Tic Tac Toe Game. Because it is easily known that each step is highly related with other steps in a game,

The linear model, like SVM, would be perfect fit to this game. This is very reasonable to use SVM here. Basically, SVM is to maximize the margin of the classifier and is majorly influenced by those "support vectors". In this case, we would know that the "middle point" is the key "support vectors". Therefore, SVM could achieve good performances in this case.

4. Is it valid to do supervised feature selection over your whole data set prior to doing cross validation for evaluating the model building? Why or why not? What should you do instead?

Solution:

No. Feature selection is part of the model building. So doing supervised feature selection over the whole data set prior to doing cross validation would give the model building an unfair advantage by narrowing the space of model considered to those that are more likely to perform well on test data. To conclude, this would give us better performances on cross validation data, but not able to make sure that the test data would be better.

Instead, we should do feature selection over the training data on each fold separatelu.

5. In class we looked at a multi-layer (i.e., nonlinear) model for modeling **A XOR B**. We discussed why a linear model can not compute **A XOR B** if your feature space is two inputs, one of which represents A and the other of which represents B. What would be ***a different feature space representation*** that would allow you to model **A XOR B** with a linear model?

Solution:

We could use hidden layers and activation threshold to allow feature with more complexity. This would transfer those complex features, like "**A XOR B**", to a different feature space representation for a linear model.

Part 2: Test of Practical Competence

As you do your work with this data set, refer to the LightSide Manual and the lecture slides.

Part A: File setup.

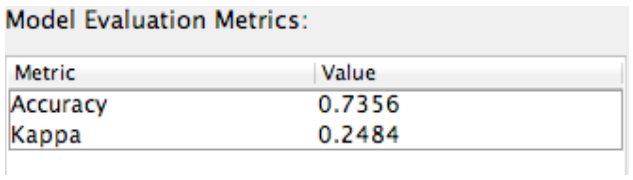
1. Start with the *midtermdata.csv* file that you downloaded with these instructions. Notice that the file has 1569 instances from 771 students. You should divide the data into a development set with 100 students and a cross validation set with the remaining students. Call the development set *midtermdata-dev.csv* and call the cross validation set *midtermdata-cv.csv*. **You should include these two csv files when you submit your midterm.**

Part B: Investigate the use of text features.

2. On the first ("Extract Features") panel, load *midtermdata-cv.csv* and configure the panel so that you are using only unigram features. Under "Basic Features", make sure "Unigrams" and "Track Feature Hit Location" are checked. Uncheck "Include Punctuation".
    - **Save the unigram feature table as an ARFF file called *baseline.arff*. You should include this file when you submit your midterm.**

3. Build a baseline model.
    - On the "Build Models" panel under "Learning Plugins", select the LibLINEAR SVM (under "Support Vector Machines") as your machine learning algorithm.
    - In the center pane, make sure you're set up for leave-subpopulations-out 10-fold cross-validation ("by annotation") where StudID is the subpopulation. (See lecture slides from Week 9 Lecture 1)
    - Press the "Train" button to start training and evaluating the model.
    - When training is done, you'll see the results at the bottom of the pane. **Record the baseline performance (Accuracy and Kappa), under "Model Evaluation Metrics".**

Solution:
The result is:



Model Evaluation Metrics:

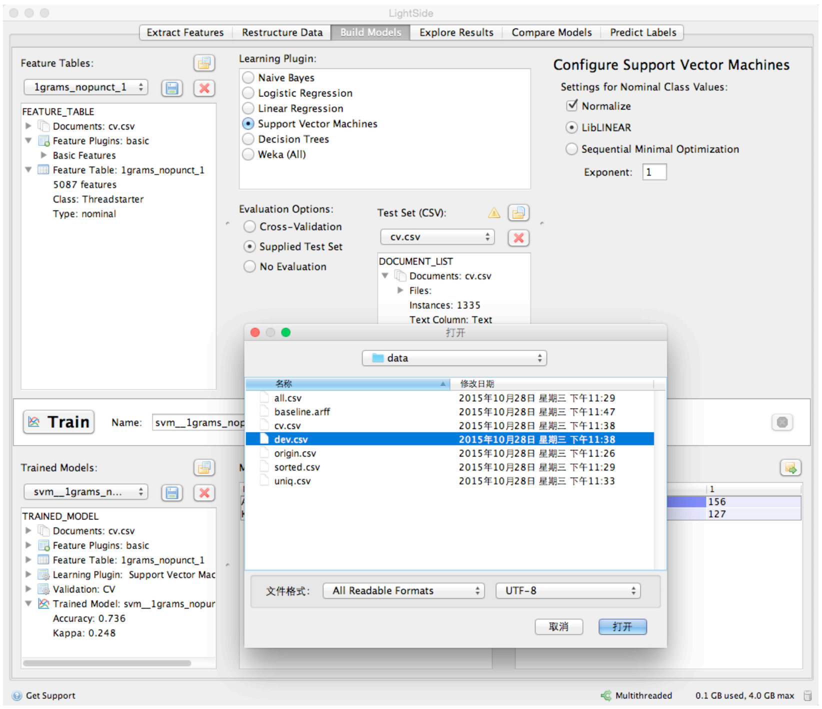
Metric	Value
Accuracy	0.7356
Kappa	0.2484

4. You're going to prepare to do an error analysis, to determine where the machine learning model is making mistakes. We want to avoid "cheating" by analyzing the training data directly, so you'll train on the cross-validation set, and test on the development set. You'll perform the error analysis on the test results from the development set. **Make sure you document your process as you observe in the answer key from assignment 6.**
    - Under the center pane of "Build Models", set up the evaluation to use the "Supplied Test Set" option. Load the development set as the test set.

- Press the “Train” button. The result should be similar to what you saw when you did the cross-validation, but it won’t be exactly the same.

Solution:

First open the dev.csv file, and load into LightSide.



Feature Tables: 1grams_nopunct_1

FEATURE_TABLE
- Documents: cv.csv
- Feature Plugins: basic
  - Basic Features
- Feature Table: 1grams_nopunct_1
  - 5087 features
  - Class: Threadstarter
  - Type: nominal

Learning Plugin:
- Naive Bayes
- Logistic Regression
- Linear Regression
- [✓ Support Vector Machines]
- Decision Trees
- Weka (All)

Configure Support Vector Machines

Settings for Nominal Class Values:
- [✓ Normalize]
- [✓ LibLINEAR]
- Sequential Minimal Optimization

Exponent: 1

Evaluation Options:
- Cross-Validation
- [✓ Supplied Test Set]
- No Evaluation

Test Set (CSV): cv.csv

DOCUMENT_LIST
- Documents: cv.csv
  - Files:
  - Instances: 1335
  - Text Column: Text

Name: svm__1grams_nop

Trained Models: svm__1grams_n...

TRAINED_MODEL
- Documents: cv.csv
- Feature Plugins: basic
- Feature Table: 1grams_nopunct_1
- Learning Plugin: Support Vector Mac
- Validation: CV
- Trained Model: svm__1grams_nopur
  - Accuracy: 0.736
  - Kappa: 0.248

打开

data

名称	修改日期
all.csv	2015年10月28日 星期三 下午11:29
baseline.arff	2015年10月28日 星期三 下午11:47
cv.csv	2015年10月28日 星期三 下午11:38
dev.csv	2015年10月28日 星期三 下午11:38
origin.csv	2015年10月28日 星期三 下午11:26
sorted.csv	2015年10月28日 星期三 下午11:29
uniq.csv	2015年10月28日 星期三 下午11:33

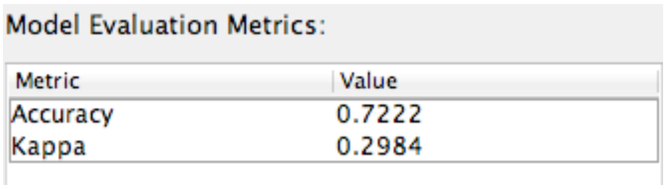
文件格式: All Readable Formats   UTF-8

取消   打开

First, under the center pane of “build models”, set up the “supplied test set” option. Load the development dataset. Note that the “dev” data contains different students from “cv”.

Then, press the “train” button to train the model.

The result is different on “cv” data. I got accuracy of 72.22 % and kappa of 0.2984.



Model Evaluation Metrics:

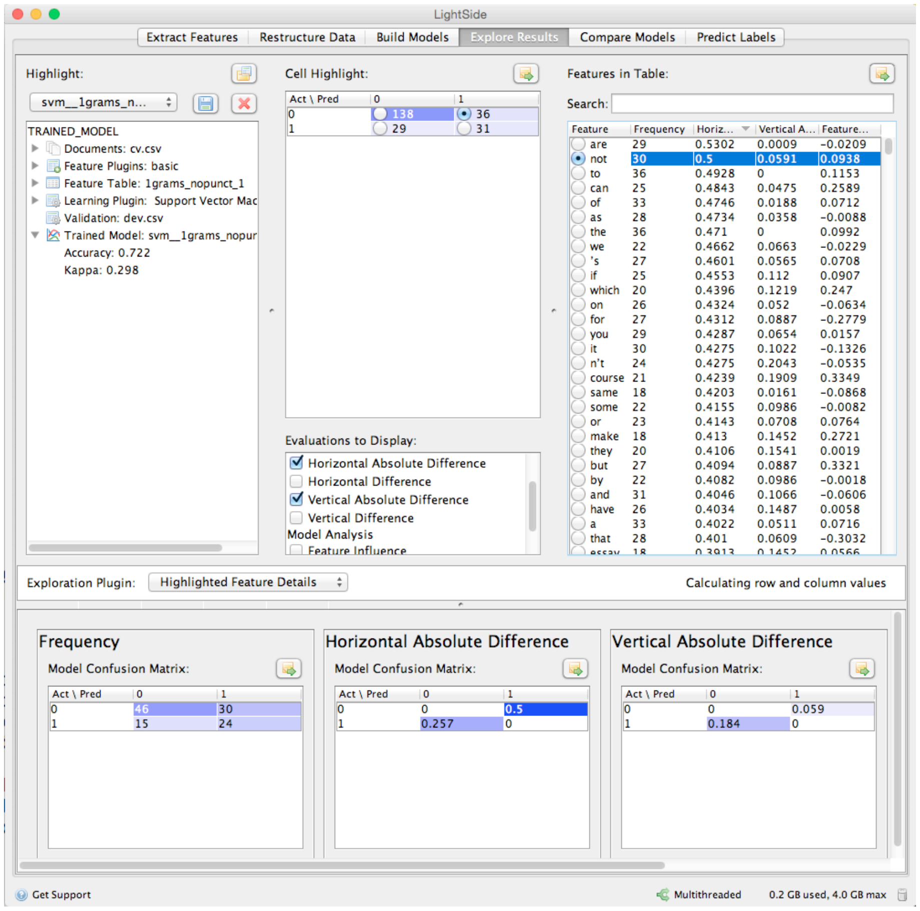
Metric	Value
Accuracy	0.7222
Kappa	0.2984

5. Switch to the “Explore Results” panel to do your error analysis on these test results.
   - Make sure the second trained model is selected in the top-left pane.
   - Use the error analysis methodology demonstrated in class and in the Assignment 6 answer key. This includes using horizontal and vertical comparisons to identify problematic features, then using additional tests to try to identify the reasons why these features are problematic.
   - The Explore Results pane (including all of the built-in error analysis metrics) is described in Chapter 7 of the LightSide manual.

- **Describe the problematic features you identified, and explain how you determined they were problematic. Describe also the qualitative analysis you did when you went back to the text to see how those features were potentially used differently between cells of the confusion matrix, or what other things were going on in those texts that might be the real problem that needs to be solved.**

Solution:

The panel is here:



Highlight: svm__1grams_n...

TRAINED_MODEL
- Documents: cv.csv
- Feature Plugins: basic
- Feature Table: 1grams_nopunct_1
- Learning Plugin: Support Vector Mac
- Validation: dev.csv
- Trained Model: svm__1grams_nopur
  - Accuracy: 0.722
  - Kappa: 0.298

Cell Highlight:

Act \ Pred	0	1
0	138	● 36
1	29	31

Evaluations to Display:
- [✓] Horizontal Absolute Difference
- [ ] Horizontal Difference
- [✓] Vertical Absolute Difference
- [ ] Vertical Difference

Model Analysis
- [ ] Feature Influence

Features in Table:

Search:

Feature	Frequency	Horiz...	Vertical A...	Feature...
are	29	0.5302	0.0009	-0.0209
● not	30	0.5	0.0591	0.0938
to	36	0.4928	0	0.1153
can	25	0.4843	0.0475	0.2589
of	33	0.4746	0.0188	0.0712
as	28	0.4734	0.0358	-0.0088
the	36	0.471	0	0.0992
we	22	0.4662	0.0663	-0.0229
's	27	0.4601	0.0565	0.0708
if	25	0.4553	0.112	0.0907
which	20	0.4396	0.1219	0.247
on	26	0.4324	0.052	-0.0634
for	27	0.4312	0.0887	-0.2779
you	29	0.4287	0.0654	0.0157
it	30	0.4275	0.1022	-0.1326
n't	24	0.4275	0.2043	-0.0535
course	21	0.4239	0.1909	0.3349
same	18	0.4203	0.0161	-0.0868
some	22	0.4155	0.0986	-0.0082
or	23	0.4143	0.0708	0.0764
make	18	0.413	0.1452	0.2721
they	20	0.4106	0.1541	0.0019
but	27	0.4094	0.0887	0.3321
by	22	0.4082	0.0986	-0.0018
and	31	0.4046	0.1066	-0.0606
have	26	0.4034	0.1487	0.0058
a	33	0.4022	0.0511	0.0716
that	28	0.401	0.0609	-0.3032
essay	18	0.3913	0.1452	0.0566

Exploration Plugin: Highlighted Feature Details

Calculating row and column values

Frequency

Model Confusion Matrix:

Act \ Pred	0	1
0	46	30
1	15	24

Horizontal Absolute Difference

Model Confusion Matrix:

Act \ Pred	0	1
0	0	0.5
1	0.257	0

Vertical Absolute Difference

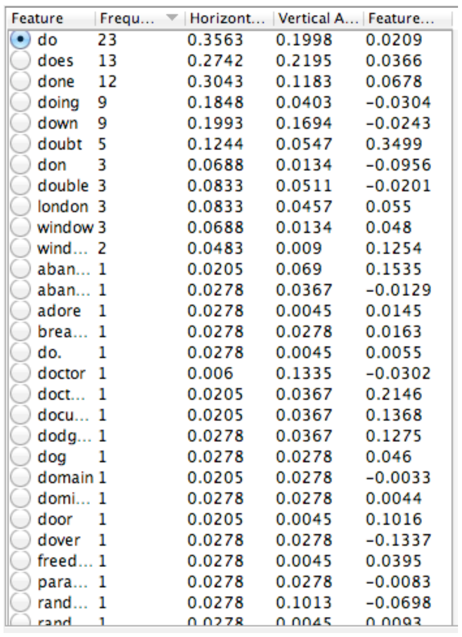
Model Confusion Matrix:

Act \ Pred	0	1
0	0	0.059
1	0.184	0

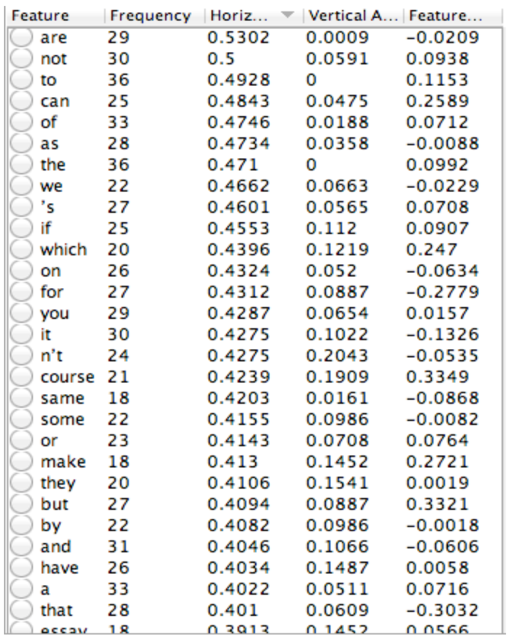
Get Support — Multithreaded — 0.2 GB used, 4.0 GB max

I went to the Explore Results panel to see the result. And selected the second created morel in the upper left. Looking the confusion matric in the "Cell Highlight" panel, I choose the bottom left error cell because it was the biggest error one. I selected that and selected a list of features came up in the Features in Table Panel.

I choose both the horizontal and vertical errors and analyze the horizontal at first. I selected Frequency, Horizontal Absolute Difference, Vertical Absolute Difference, and Feature Weight. I clicked to sort it. In the sorted result, we could find "n't", "can", are of high frequency and Horizontal errors. One thing we know about LightSide is that it would break "don't" into "do" and ''n't". This would cause confusion, since the "n't" could result from different combinations.



Feature	Frequ...	Horizont...	Vertical A...	Feature...
do	23	0.3563	0.1998	0.0209
does	13	0.2742	0.2195	0.0366
done	12	0.3043	0.1183	0.0678
doing	9	0.1848	0.0403	-0.0304
down	9	0.1993	0.1694	-0.0243
doubt	5	0.1244	0.0547	0.3499
don	3	0.0688	0.0134	-0.0956
double	3	0.0833	0.0511	-0.0201
london	3	0.0833	0.0457	0.055
window	3	0.0688	0.0134	0.048
wind...	2	0.0483	0.009	0.1254
aban...	1	0.0205	0.069	0.1535
aban...	1	0.0278	0.0367	-0.0129
adore	1	0.0278	0.0045	0.0145
brea...	1	0.0278	0.0278	0.0163
do.	1	0.0278	0.0045	0.0055
doctor	1	0.006	0.1335	-0.0302
doct...	1	0.0205	0.0367	0.2146
docu...	1	0.0205	0.0367	0.1368
dodg...	1	0.0278	0.0367	0.1275
dog	1	0.0278	0.0278	0.046
domain	1	0.0205	0.0278	-0.0033
domi...	1	0.0278	0.0278	0.0044
door	1	0.0205	0.0045	0.1016
dover	1	0.0278	0.0278	-0.1337
freed...	1	0.0278	0.0045	0.0395
para...	1	0.0278	0.0278	-0.0083
rand...	1	0.0278	0.1013	-0.0698
rand...	1	0.0278	0.0045	0.0093



Feature	Frequency	Horiz...	Vertical A...	Feature...
are	29	0.5302	0.0009	-0.0209
not	30	0.5	0.0591	0.0938
to	36	0.4928	0	0.1153
can	25	0.4843	0.0475	0.2589
of	33	0.4746	0.0188	0.0712
as	28	0.4734	0.0358	-0.0088
the	36	0.471	0	0.0992
we	22	0.4662	0.0663	-0.0229
's	27	0.4601	0.0565	0.0708
if	25	0.4553	0.112	0.0907
which	20	0.4396	0.1219	0.247
on	26	0.4324	0.052	-0.0634
for	27	0.4312	0.0887	-0.2779
you	29	0.4287	0.0654	0.0157
it	30	0.4275	0.1022	-0.1326
n't	24	0.4275	0.2043	-0.0535
course	21	0.4239	0.1909	0.3349
same	18	0.4203	0.0161	-0.0868
some	22	0.4155	0.0986	-0.0082
or	23	0.4143	0.0708	0.0764
make	18	0.413	0.1452	0.2721
they	20	0.4106	0.1541	0.0019
but	27	0.4094	0.0887	0.3321
by	22	0.4082	0.0986	-0.0018
and	31	0.4046	0.1066	-0.0606
have	26	0.4034	0.1487	0.0058
a	33	0.4022	0.0511	0.0716
that	28	0.401	0.0609	-0.3032
essay	18	0.3913	0.1452	0.0566

Then in the vertical part, I also clicked the button to sort the result. we would still found that "do" has a high frequency and Vertical Absolute Error. This means that the "don't" might indeed cause confusions for our classifiers.

Besides, another thing worth noting is that we could see many words without significant meaning, like "the", "of", and "a", which is called stop words, have very high frequency, we could try to filter the stop words for higher performances.

What's more, some word combinations are separated into several different words due to the unigram model, so we could try to use the "bigram" model to test if would lead to better performances.

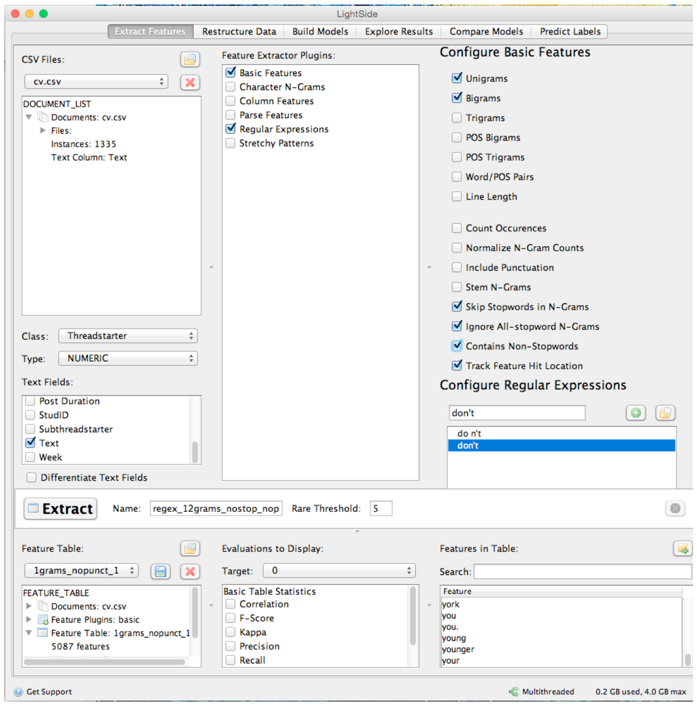
6. Based on what you learned from the error analysis, come up with some ideas for new features that you think might help the machine learning algorithm overcome the errors you found. For this section, use only text-based features – ignore the extra columns in the dataset for now. **Explain your proposed features and why you think they will solve the problems you noticed.**

Solution

Based on my found problems, I could use "do + n't" to introduce a new feature. This could easily be achieved by the regular expression. This would allow me to use the new feature "don't" rather than "do" + "n't" separately".

Besides, LightSide also offers choices to filter the stop words, which could help improve the performances.

What's more, the LightSide also offers choices to build bigram word features. We could test whether bigram could be a good idea.



CSV Files: cv.csv

DOCUMENT_LIST
Documents: cv.csv
Files:
Instances: 1335
Text Column: Text

Class: Threadstarter
Type: NUMERIC

Text Fields:
- [ ] Post Duration
- [ ] StudID
- [ ] Subthreadstarter
- [x] Text
- [ ] Week

- [ ] Differentiate Text Fields

Feature Extractor Plugins:
- [x] Basic Features
- [ ] Character N-Grams
- [ ] Column Features
- [ ] Parse Features
- [x] Regular Expressions
- [ ] Stretchy Patterns

Configure Basic Features
- [x] Unigrams
- [x] Bigrams
- [ ] Trigrams
- [ ] POS Bigrams
- [ ] POS Trigrams
- [ ] Word/POS Pairs
- [ ] Line Length
- [ ] Count Occurences
- [ ] Normalize N-Gram Counts
- [ ] Include Punctuation
- [ ] Stem N-Grams
- [x] Skip Stopwords in N-Grams
- [x] Ignore All-stopword N-Grams
- [x] Contains Non-Stopwords
- [x] Track Feature Hit Location

Configure Regular Expressions
don't
do n't
don't

Name: regex_12grams_nostop_nop Rare Threshold: 5

Feature Table: 1grams_nopunct_1

FEATURE_TABLE
Documents: cv.csv
Feature Plugins: basic
Feature Table: 1grams_nopunct_1
5087 features

Evaluations to Display:
Target: 0
Basic Table Statistics
- [ ] Correlation
- [ ] F-Score
- [ ] Kappa
- [ ] Precision
- [ ] Recall

Features in Table:
Search:

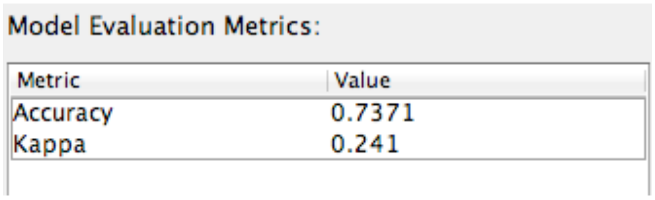
Feature
york
you
you.
young
younger
your

0.2 GB used, 4.0 GB max

7. Test whether your ideas are effective at addressing the problems you found. You will evaluate the effect on the cross-validation set.
   - Return to "Extract Features" panel and set up the feature extraction using the ideas you listed under #6. See Chapter 4 of the LightSide manual for more information on advanced feature extraction. In addition to the options under Basic Features, Regular Expressions may be especially useful.
   - **Extract the features and then save the new feature table as an ARFF file named *new.arff*. You will turn this file in with your midterm.**
   - Return to the Build Models pane. Reset the options in the center pane for leave-one-subpopulation-out 10-fold cross-validation with StudID as the subpopulation
   - Train and evaluate a cross-validated model to test the performance of the new feature space. **Record the new performance values (Accuracy and Kappa).**

Solution:

We can see the result:



Model Evaluation Metrics:

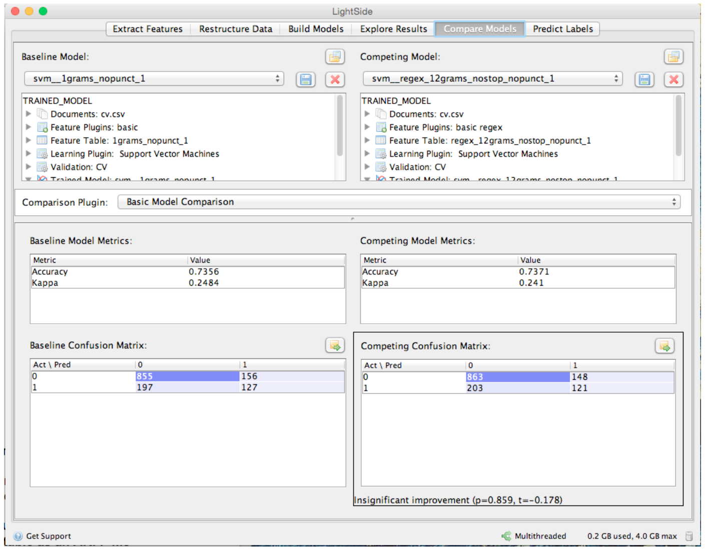
Metric	Value
Accuracy	0.7371
Kappa	0.241

The accuracy is 73.71% and the Kapp is 0.241. This is better than the previous result.

8. Compare the results.
    - Switch to the “Compare Models” pane. Make sure the baseline model is selected on the left and the final model is selected on the right. See Chapter 8 of the LightSide manual for more information.
    - **Test whether the performance difference is statistically significant.**
    - **Try to determine (and describe) the kinds of instances on which the two models perform differently.**

Solution:

The result is here:



Baseline Model: svm__1grams_nopunct_1

```
TRAINED_MODEL
  Documents: cv.csv
  Feature Plugins: basic
  Feature Table: 1grams_nopunct_1
  Learning Plugin:  Support Vector Machines
  Validation: CV
  Trained Model: svm__1grams_nopunct_1
```

Competing Model: svm__regex_12grams_nostop_nopunct_1

```
TRAINED_MODEL
  Documents: cv.csv
  Feature Plugins: basic regex
  Feature Table: regex_12grams_nostop_nopunct_1
  Learning Plugin:  Support Vector Machines
  Validation: CV
  Trained Model: svm__regex_12grams_nostop_nopunct_1
```

Comparison Plugin: Basic Model Comparison

Baseline Model Metrics:

Metric	Value
Accuracy	0.7356
Kappa	0.2484

Competing Model Metrics:

Metric	Value
Accuracy	0.7371
Kappa	0.241

Baseline Confusion Matrix:

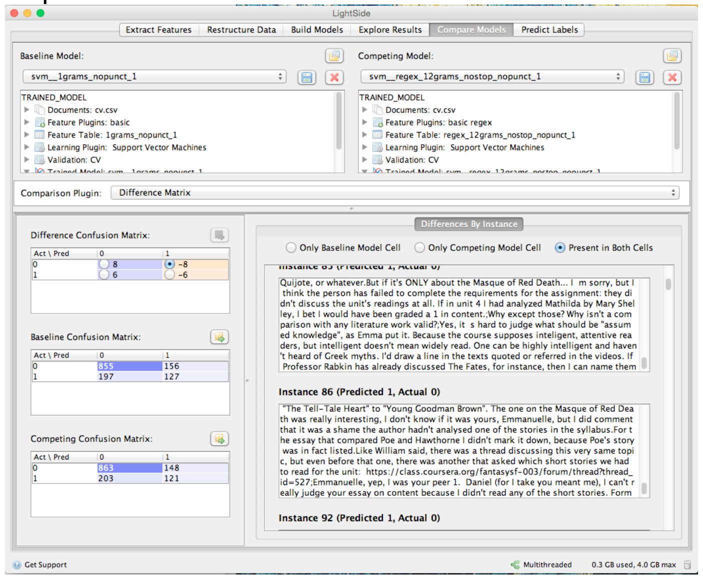
Act \ Pred	0	1
0	855	156
1	197	127

Competing Confusion Matrix:

Act \ Pred	0	1
0	863	148
1	203	121

Insignificant improvement (p=0.859, t=-0.178)

We could see that the new feature is not significant important. But we did have performances improvement.



Baseline Model: svm__1grams_nopunct_1

```
TRAINED_MODEL
  Documents: cv.csv
  Feature Plugins: basic
  Feature Table: 1grams_nopunct_1
  Learning Plugin:  Support Vector Machines
  Validation: CV
  Trained Model: svm__1grams_nopunct_1
```

Competing Model: svm__regex_12grams_nostop_nopunct_1

```
TRAINED_MODEL
  Documents: cv.csv
  Feature Plugins: basic regex
  Feature Table: regex_12grams_nostop_nopunct_1
  Learning Plugin:  Support Vector Machines
  Validation: CV
  Trained Model: svm__regex_12grams_nostop_nopunct_1
```

Comparison Plugin: Difference Matrix

Difference Confusion Matrix:

Act \ Pred	0	1
0	8	[✓ -8]
1	6	-6

Baseline Confusion Matrix:

Act \ Pred	0	1
0	855	156
1	197	127

Competing Confusion Matrix:

Act \ Pred	0	1
0	863	148
1	203	121

Only Baseline Model Cell   Only Competing Model Cell   [✓ Present in Both Cells]

Instance 85 (Predicted 1, Actual 0)

```
Quijote, or whatever.But if it's ONLY about the Masque of Red Death... I  m sorry, but I
 think the person has failed to complete the requirements for the assignment: they di
dn't discuss the unit's readings at all. If in unit 4 I had analyzed Mathilda by Mary Shel
ley, I bet I would have been graded a 1 in content.;Why except those? Why isn't a com
parison with any literature work valid?;Yes, it  s hard to judge what should be "assum
ed knowledge", as Emma put it. Because the course supposes inteligent, attentive rea
ders, but intelligent doesn't mean widely read. One can be highly intelligent and haven
't heard of Greek myths. I'd draw a line in the texts quoted or referred in the videos. If
 Professor Rabkin has already discussed The Fates, for instance, then I can name them
```

Instance 86 (Predicted 1, Actual 0)

```
 "The Tell-Tale Heart" to "Young Goodman Brown". The one on the Masque of Red Dea
th was really interesting, I don't know if it was yours, Emmanuelle, but I did comment
that it was a shame the author hadn't analysed one of the stories in the syllabus.For t
he essay that compared Poe and Hawthorne I didn't mark it down, because Poe's story
 was in fact listed.Like William said, there was a thread discussing this very same topi
c, but even before that one, there was another that asked which short stories we had
to read for the unit:  https://class.coursera.org/fantasysf-003/forum/thread?thread_
id=527;Emmanuelle, yep, I was your peer 1.  Daniel (for I take you meant me), I can't r
eally judge your essay on content because I didn't read any of the short stories. Form
```

Instance 92 (Predicted 1, Actual 0)

We could see that in the new model, those instances with many “don’t” words, and with many “stop words” would be correctly classified, while in the previous model, the result would be wrong.

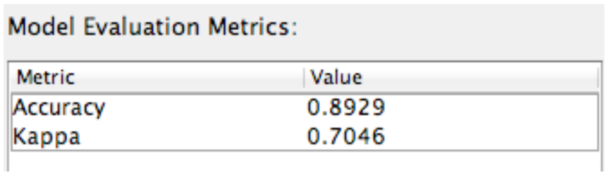
Part C: Investigate the use of meta-data features.

1. On the first ("Extract Features") panel, load *midtermdata-cv.csv* and configure the panel so that you are using only column features. Make sure that you do not include StudID as one of the column features you extract.
   - **Save the metadata-only feature table as an ARFF file called *baseline-meta.arff*. You should include this file when you submit your midterm.**

2. Build a baseline model just as you did in #2 in Part B.
   - **Record the baseline performance (Accuracy and Kappa), under "Model Evaluation Metrics".**

Solution:

The result is:



Model Evaluation Metrics:

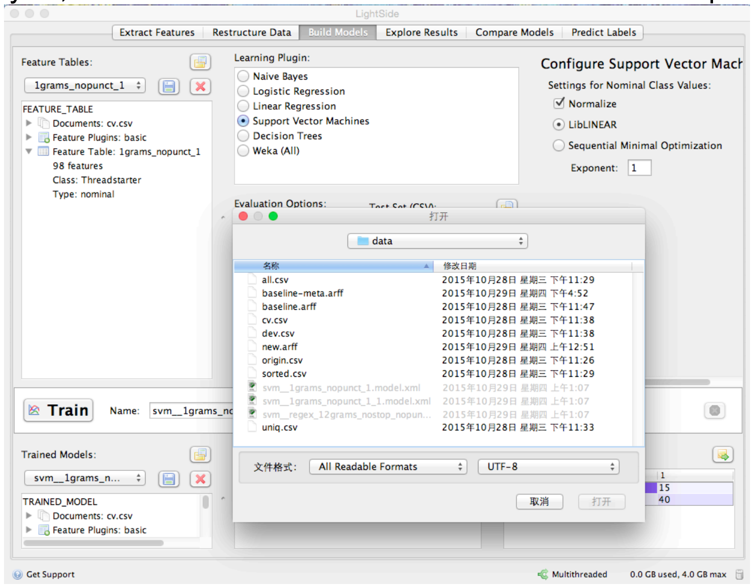
Metric	Value
Accuracy	0.8929
Kappa	0.7046

We can see that the accuracy is 0.89289 and the Kappa is 0.7046.

3. You're going to prepare to do an error analysis the same way you did in #3 in Part B. **Make sure you document your process as you observe in the answer key from assignment 6.**

Solution:

To do error analysis, we first should use the "dev" data test the model performances.



Feature Tables: 1grams_nopunct_1

FEATURE_TABLE
- Documents: cv.csv
- Feature Plugins: basic
- Feature Table: 1grams_nopunct_1
  - 98 features
  - Class: Threadstarter
  - Type: nominal

Learning Plugin: Naive Bayes / Logistic Regression / Linear Regression / [✓ Support Vector Machines] / Decision Trees / Weka (All)

Configure Support Vector Mach

Settings for Nominal Class Values:
- ✓ Normalize
- [✓ LibLINEAR]
- Sequential Minimal Optimization
- Exponent: 1

Evaluation Options: Test Set (CSV):

打开 — data

名称	修改日期
all.csv	2015年10月28日 星期三 下午11:29
baseline-meta.arff	2015年10月29日 星期四 下午4:52
baseline.arff	2015年10月28日 星期三 下午11:47
cv.csv	2015年10月28日 星期三 下午11:38
dev.csv	2015年10月28日 星期三 下午11:38
new.arff	2015年10月29日 星期四 上午12:51
origin.csv	2015年10月28日 星期三 下午11:26
sorted.csv	2015年10月28日 星期三 下午11:29
svm__1grams_nopunct_1.model.xml	2015年10月29日 星期四 上午1:07
svm__1grams_nopunct_1_1.model.xml	2015年10月29日 星期四 上午1:07
svm__regex_12grams_nostop_nopun...	2015年10月29日 星期四 上午1:07
uniq.csv	2015年10月28日 星期三 下午11:33

文件格式: All Readable Formats — UTF-8

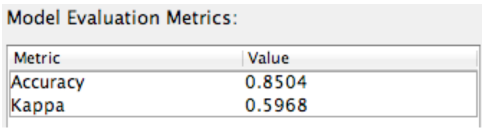
Name: svm__1grams_nc

Trained Models: svm__1grams_n...

TRAINED_MODEL
- Documents: cv.csv
- Feature Plugins: basic

1
15
40

And the result is:



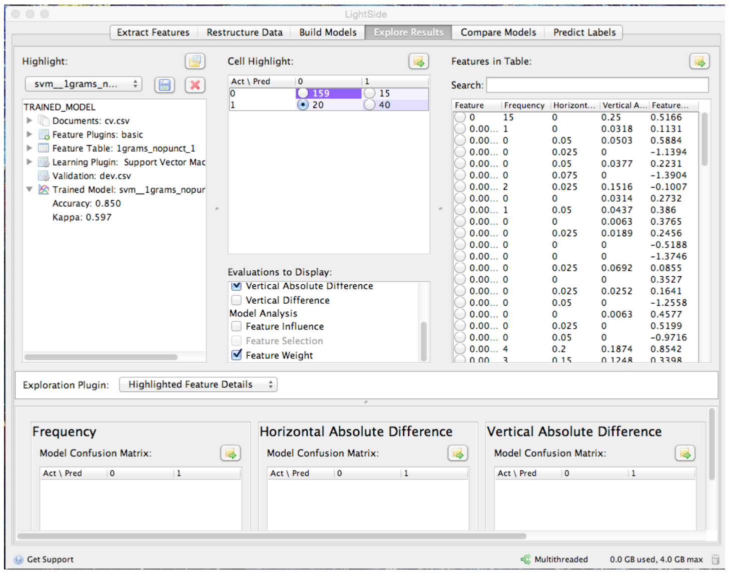
Model Evaluation Metrics:

Metric	Value
Accuracy	0.8504
Kappa	0.5968

4. Switch to the “Explore Results” panel to do your error analysis on these test results.
    - Make sure the last trained model is selected in the top-left pane.
    - Use the error analysis methodology demonstrated in class. This includes using horizontal and vertical comparisons to identify problematic features, then using additional tests to try to identify the reasons why these features are problematic. You may find it useful to export a csv from the error analysis panel with the extra columns that it provides.
    - The Explore Results pane (including all of the built-in error analysis metrics) is described in Chapter 7 of the LightSide manual. But recall what we discussed in class regarding doing an error analysis when you have not extracted any text features. You can refer to the slides for Week 9 Lecture 1.
    - **Describe the problematic features you identified, and explain how you determined they were problematic.**

Solution:

First, we need to go to the explore result table for next step.



svm__1grams_n...

TRAINED_MODEL
- Documents: cv.csv
- Feature Plugins: basic
- Feature Table: 1grams_nopunct_1
- Learning Plugin: Support Vector Mac
- Validation: dev.csv
- Trained Model: svm__1grams_nopur
    - Accuracy: 0.850
    - Kappa: 0.597

Act \ Pred	0	1
0	159	15
1	20	40

Evaluations to Display:
- [✓] Vertical Absolute Difference
- [ ] Vertical Difference

Model Analysis
- [ ] Feature Influence
- [ ] Feature Selection
- [✓] Feature Weight

Feature	Frequency	Horizont...	Vertical A...	Feature...
0	15	0	0.25	0.5166
0.00...	1	0	0.0318	0.1131
0.00...	0	0.05	0.0503	0.5884
0.00...	0	0.025	0	-1.1394
0.00...	0	0.05	0.0377	0.2231
0.00...	0	0.075	0	-1.3904
0.00...	2	0.025	0.1516	-0.1007
0.00...	0	0	0.0314	0.2732
0.00...	1	0.05	0.0437	0.386
0.00...	0	0	0.0063	0.3765
0.00...	0	0.025	0.0189	0.2456
0.00...	0	0	0	-0.5188
0.00...	0	0	0	-1.3746
0.00...	0	0.025	0.0692	0.0855
0.00...	0	0	0	0.3527
0.00...	0	0.025	0.0252	0.1641
0.00...	0	0.05	0	-1.2558
0.00...	0	0	0.0063	0.4577
0.00...	0	0.025	0	0.5199
0.00...	0	0.05	0	-0.9716
0.00...	4	0.2	0.1874	0.8542
0.00...	3	0.15	0.1248	0.3398

Exploration Plugin: Highlighted Feature Details

Frequency — Model Confusion Matrix:

Act \ Pred	0	1

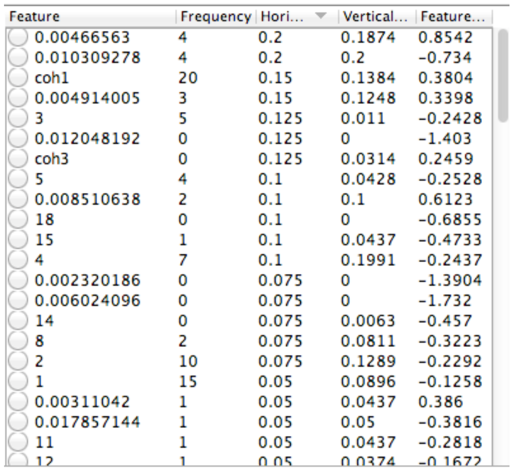
Horizontal Absolute Difference — Model Confusion Matrix:

Act \ Pred	0	1

Vertical Absolute Difference — Model Confusion Matrix:

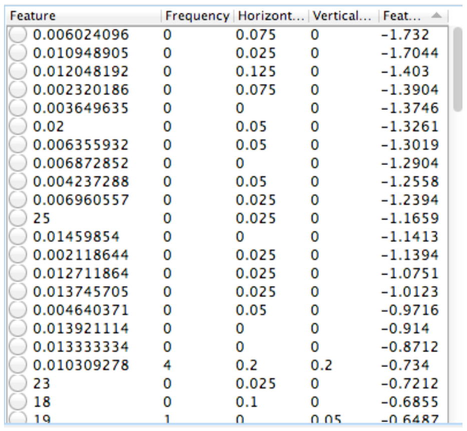
Act \ Pred	0	1

We choose the right bottom feature, because it has the most wrong cases. The following is the detailed result graph:



Feature	Frequency	Hori...	Vertical...	Feature...
0.00466563	4	0.2	0.1874	0.8542
0.010309278	4	0.2	0.2	-0.734
coh1	20	0.15	0.1384	0.3804
0.004914005	3	0.15	0.1248	0.3398
3	5	0.125	0.011	-0.2428
0.012048192	0	0.125	0	-1.403
coh3	0	0.125	0.0314	0.2459
5	4	0.1	0.0428	-0.2528
0.008510638	2	0.1	0.1	0.6123
18	0	0.1	0	-0.6855
15	1	0.1	0.0437	-0.4733
4	7	0.1	0.1991	-0.2437
0.002320186	0	0.075	0	-1.3904
0.006024096	0	0.075	0	-1.732
14	0	0.075	0.0063	-0.457
8	2	0.075	0.0811	-0.3223
2	10	0.075	0.1289	-0.2292
1	15	0.05	0.0896	-0.1258
0.00311042	1	0.05	0.0437	0.386
0.017857144	1	0.05	0.05	-0.3816
11	1	0.05	0.0437	-0.2818
12	1	0.05	0.0374	-0.1672

I click to sort the table and fond those with specific numbers are very much likely to cause the trouble.



Feature	Frequency	Horizont...	Vertical...	Feat...
0.006024096	0	0.075	0	-1.732
0.010948905	0	0.025	0	-1.7044
0.012048192	0	0.125	0	-1.403
0.002320186	0	0.075	0	-1.3904
0.003649635	0	0	0	-1.3746
0.02	0	0.05	0	-1.3261
0.006355932	0	0.05	0	-1.3019
0.006872852	0	0	0	-1.2904
0.004237288	0	0.05	0	-1.2558
0.006960557	0	0.025	0	-1.2394
25	0	0.025	0	-1.1659
0.01459854	0	0	0	-1.1413
0.002118644	0	0.025	0	-1.1394
0.012711864	0	0.025	0	-1.0751
0.013745705	0	0.025	0	-1.0123
0.004640371	0	0.05	0	-0.9716
0.013921114	0	0	0	-0.914
0.013333334	0	0	0	-0.8712
0.010309278	4	0.2	0.2	-0.734
23	0	0.025	0	-0.7212
18	0	0.1	0	-0.6855
19	1	0	0.05	-0.6487

5. Based on what you learned from the error analysis, come up with some ideas for new metadata-based features that you think might help the machine learning algorithm overcome the errors you found. In this case, you may need to create new features in your spreadsheet editor (Excel, etc) by computing formulas over the metadata attributes. Remember the discussions about transforming features we had early in the semester in connection with the Opinion Poll dataset. **Explain your proposed features and why you think they will solve the problems you noticed.**

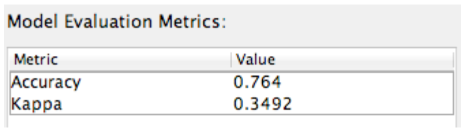
Solution:

I would delete the feature of Authority and Hub. Because this two feature are numeric features, we are using them as the nominal value. Besides, based on our error analysis, we should try to get rid of the bad effects from those "bad" features.

6. Test whether your ideas are effective at addressing the problems you found. You will evaluate the effect on the cross-validation set.
   - Add extra columns for your proposed features to your CSV (in your spreadsheet editor, etc). Make sure to add any new columns to both datasets. Select the column features you want in the Extract Features pane.
     (Alternatively, you can combine existing column features by using the Combine Features plugin in the Restructure Data pane – see Section 5.2 of the LightSide manual)
   - **Extract (or restructure) the features and save the new feature table as an ARFF file named *new-meta.arff*. You will turn this file in with your midterm.**
   - Return to the Build Models pane. Reset the options in the center pane for leave-subpopulations-out ("by annotation") 10-fold cross-validation with StudID as the subpopulation.
   - Train and evaluate a cross-validated model to test the performance of the new feature space. **Record the new performance values (Accuracy and Kappa).**

Solution:

The result is



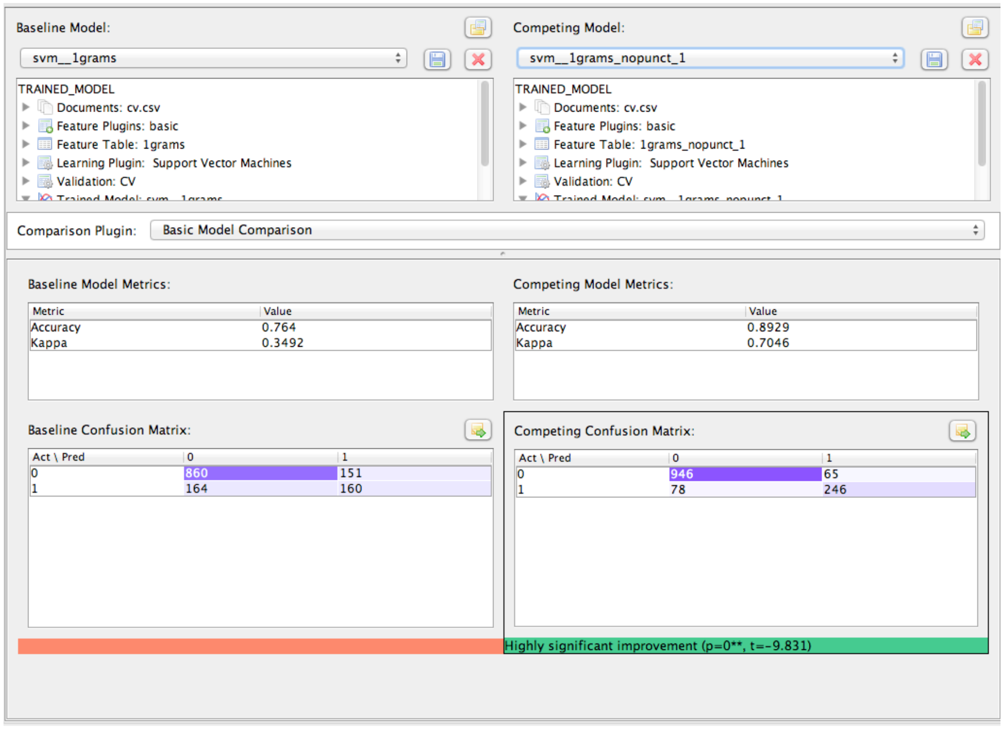
Model Evaluation Metrics:

Metric	Value
Accuracy	0.764
Kappa	0.3492

7. Compare the results.
   - Switch to the “Compare Models” pane. Make sure the baseline model is selected on the left and the final model is selected on the right. See Chapter 8 of the LightSide manual for more information.
   - **Test whether the performance difference is statistically significant.**
   - **Try to determine (and describe) the kinds of instances on which the two models perform differently.**

Solution:

We could it is statistically significant



Baseline Model: svm__1grams

TRAINED_MODEL
- Documents: cv.csv
- Feature Plugins: basic
- Feature Table: 1grams
- Learning Plugin: Support Vector Machines
- Validation: CV

Competing Model: svm__1grams_nopunct_1

TRAINED_MODEL
- Documents: cv.csv
- Feature Plugins: basic
- Feature Table: 1grams_nopunct_1
- Learning Plugin: Support Vector Machines
- Validation: CV

Comparison Plugin: Basic Model Comparison

Baseline Model Metrics:

Metric	Value
Accuracy	0.764
Kappa	0.3492

Competing Model Metrics:

Metric	Value
Accuracy	0.8929
Kappa	0.7046

Baseline Confusion Matrix:

Act \ Pred	0	1
0	860	151
1	164	160

Competing Confusion Matrix:

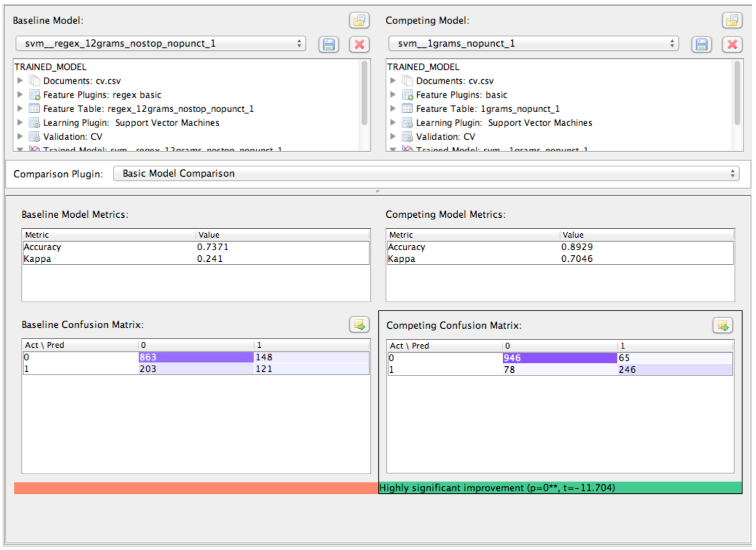
Act \ Pred	0	1
0	946	65
1	78	246

Highly significant improvement (p=0**, t=-9.831)

Part D: Final Model.

1. Go back to the “Compare Models” pane now and compare the performance of best model you built in Part B with the best model you built in Part C. Which was better?

Solution:



Baseline Model:

svm__regex_12grams_nostop_nopunct_1

TRAINED_MODEL
- Documents: cv.csv
- Feature Plugins: regex basic
- Feature Table: regex_12grams_nostop_nopunct_1
- Learning Plugin: Support Vector Machines
- Validation: CV

Competing Model:

svm__1grams_nopunct_1

TRAINED_MODEL
- Documents: cv.csv
- Feature Plugins: basic
- Feature Table: 1grams_nopunct_1
- Learning Plugin: Support Vector Machines
- Validation: CV

Comparison Plugin: Basic Model Comparison

Baseline Model Metrics:

Metric	Value
Accuracy	0.7371
Kappa	0.241

Competing Model Metrics:

Metric	Value
Accuracy	0.8929
Kappa	0.7046

Baseline Confusion Matrix:

Act \ Pred	0	1
0	863	148
1	203	121

Competing Confusion Matrix:

Act \ Pred	0	1
0	946	65
1	78	246

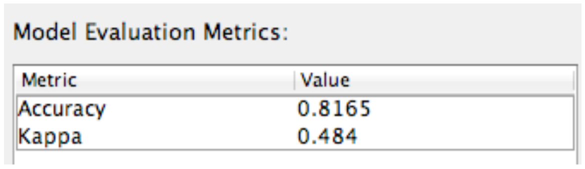
Highly significant improvement (p=0**, t=-11.704)

We could find that the model of Part C is better. Prediction with Metadata is better than predicting with text.

2. Create a new feature space that combines the features from the two models you compared in Part D #1. Now evaluate its performance. How does the performance of the combined model compare with each of the other two? Is any difference you found statistically significant in either case.

Solution:

We could find the result here:



Model Evaluation Metrics:

Metric	Value
Accuracy	0.8165
Kappa	0.484

We could see that it is much better than the Part B model, and a little bit worse than the Part C model.

We can see that the Part D model is significant better than the Part B model, and Part C model is significant better than Part D Model.

3. Extra Credit: For the model that worked best, why do you think its features worked better? Which ones were most important? What kinds of instances were classified differently by the three models, and why do you think they performed differently on those examples?

   I think the last model feature should work better, because it contains all the well designed features, both the metadata feature and the text features. We know that Probabilistic models are better at Word Features than Linear model like SVM. SVM always performs best in those common used models.