WE FEEL FINE

For 1K Dataset: (Accuracy)

Technique	BOW_ADJ	CNN features	Features Reduction
Logistic	0.9979	0.9909	0.9976
LDA	0.7724	0.9868	0.8556
SVM	0.9973	0.9912	0.9982

For 2K Dataset : (Accuracy)

Technique	Normal features	CNN features				Features Reduction
		Stride = 8	Stride = 16	Stride = 2	Stride = 4	1000 feats.
Logistic	0.774657172996	0.596650843882	0.496703586498	0.766745780591	0.699235232068	0.792457805907
LDA	0.606276371308	0.588212025316	0.525712025316	0.587025316456	0.651503164557	0.81236814346
SVM	0.769910337553	0.587816455696	0.506856540084	0.746967299578	0.699235232068	0.802478902954

For WFF 5K :

Neural Net

Hidden layer	Accuracy (Normal)	Feature Reduction	CNN
100		0.782436708861	0.738396624473
500	0.725079113924	0.774525316456	0.727320675105
500,200	0.746571729958	0.789820675105	0.739055907173
100,50		0.76516350211	0.730617088608

200		0.772020042194	0.720200421941
500,200,100	0.742220464135	0.763713080169	0.731671940928
500,200,100,50	0.74498945	0.746703586498	0.652689873418

For bigrams:

#features ~ 3000

For 1K Dataset: (Accuracy)	
Technique	**Normal features**
Logistic	0.547199170124
LDA	
SVM	0.637966804979 penalty: l2, c: 0.5, loss: squared_hinge

#features ~ 3000

For 2K Dataset: Accuracy	
Technique	**Normal features**
Logistic	0.510892116183
LDA	
SVM	0.68031496063 penalty: l2, c: 0.5, loss: squared_hinge

For uni+bi:

#features ~ 3000

For 1K Dataset: (Accuracy)	
Technique	**Normal features**
Logistic	0.713952282158
LDA	
Linear SVM	0.857883817427 penalty: l2, c: 0.5, loss: squared_hinge
SVM with RBF kernel	0.207468879668

#features ~ 3000

For 2K Dataset: Accuracy	
Technique	**Normal features**
Logistic	0.784748654679
LDA	
SVM	0.876640419948 penalty: l2, c: 0.5, loss: squared_hinge
SVM with RBF kernel	0.205276282977

#features~2857

For 5K Dataset: Accuracy	
Technique	**Normal features**
Logistic	0.870410601781
LDA	
SVM	0.902889419944

	penalty: l2, c: 0.5, loss: squared_hinge
SVM with RBF kernel	

#features ~ 28000

For 5K Dataset: Accuracy	
Technique	**Normal features**
Logistic	0.865196610906
LDA	
SVM	0.897892678688 penalty: l2, c: 0.5, loss: squared_hinge
SVM with RBF	0.422170323702

For 10k, total features = 9053 (7360 uni + 1693 bi)
TEXT_EMOTION

For Full Dataset: (Accuracy)

Technique	**Normal features**	**CNN features(4)**	**Features Reduction**
Logistic	0.358065921301	0.340786989981	0.380717293451
LDA	0.268476840424	0.277624509946	0.354145491506
SVM	0.329170901699	0.306374328445	0.351822273849

Neural Net

Hidden layer	**Accuracy (Normal)**	**Feature Reduction**		**CNN**
100	**0.329**			

500	**0.3216**	**0.348047045158**		
500,200	**0.3242**	**0.350660665021**		
100,50	**0.33483**	**0.3432**		
200	**0.3227**	**0.3522**		
500,200,100		**0.354435893713**		
500,200,100,50		**0.334252940322**		

Neural Net

Hidden layer	**Feature Reduction**			
	1200	**1000**	**800**	**500**
100	**0.3195876289**			
200	**0.3251052708**			
300	**0.3474662407**	**0.3294613039**		
400	**0.3400609844**	**0.3240888630**		
500	**0.3362857558**			
200,100	**0.3274284885**			
300,200	**0.3248148686**	**0.3280092928**		
400,300	**0.3310585160**	**0.3397705822**		
500,400	**0.3461594308**	**0.3383185712**		
400,300,200,100	**0.2320313634**	**0.2289821402**		

500,400,300,200	**0.3357049514**	**0.2313053579**		
4000		**0.3471758385**		
1000		**0.3394801800**		

How to preprocess tweets dataset:

Simple CountVect Approach

Features	**Accuracy**
10 for each class	0.4365
20	0.5410
30	0.5590
40	0.60945
50	0.624378
60	0.6113
70	0.62562
80	0.62562
90	0.61878
100	0.63184
200	0.6455
300	0.6455
400	0.6393

Simple TfIDFVectorizer Approach

Features	**Accuracy**
10 for each class	0.436567

20	0.541044
30	0.5590796
40	0.6094527
50	0.6243781
60	0.6113184
70	0.62562189
80	0.62562189
90	0.618781
100	0.63184
200	0.645522
300	0.645522
400	0.639303

Affective Text Dataset

Simple CountVect Approach

Features	**Accuracy**
100	0.434579
500	0.48598
1000	0.53738
1500	0.537383

Simple TFIDF Vectorizer Approach

Features	**Accuracy**

100	0.4345794
500	0.4859813
1000	0.537383
1500	0.537383

Precision:

```
from sklearn.metrics import average_precision_score
average_precision = average_precision_score(y_test, y_score)

print('Average precision-recall score: {0:0.2f}'.format(
      average_precision))
```

Recall

```
>>> from sklearn.metrics import recall_score
>>> y_true = [0, 1, 2, 0, 1, 2]
>>> y_pred = [0, 2, 1, 0, 0, 1]
>>> recall_score(y_true, y_pred, average='macro')
```

Precision-Recall curve:

```
from sklearn.metrics import precision_recall_curve
import matplotlib.pyplot as plt

precision, recall, _ = precision_recall_curve(y_test, y_score)

plt.step(recall, precision, color='b', alpha=0.2,
         where='post')
plt.fill_between(recall, precision, step='post', alpha=0.2,
                 color='b')

plt.xlabel('Recall')
plt.ylabel('Precision')
plt.ylim([0.0, 1.05])
plt.xlim([0.0, 1.0])
plt.title('2-class Precision-Recall curve: AP={0:0.2f}'.format(
          average_precision))
```

Confusion matrix:

```
>>> from sklearn.metrics import confusion_matrix
>>> y_true = [2, 0, 2, 2, 0, 1]
>>> y_pred = [0, 0, 2, 2, 0, 2]
>>> confusion_matrix(y_true, y_pred)
```

```
i=0
                    no_uniqueclasses = len(unique_classes)
                    labels= []
                    if no_uniqueclasses ==2:
                            labels.append(0)
                            labels.append(1)
                    elif no_uniqueclasses==3:
                            labels.append(0)
                            labels.append(1)
                            labels.append(2)
                    elif no_uniqueclasses==10:
                            labels.append(0)
                            labels.append(1)
                            labels.append(2)
                            labels.append(3)
                            labels.append(4)
                            labels.append(5)
                            labels.append(6)
                            labels.append(7)
                            labels.append(8)
                            labels.append(9)
                    confusion_matrix = np.zeros((no_uniqueclasses,no_uniqueclasses))
                    for i in range(no_uniqueclasses):
                            j=0
                            for j in range(no_uniqueclasses):
                                    k=0
                                    count_array = np.zeros(no_uniqueclasses)
                                    for k in range(len_list):
                                            #print "i:",i," j:",j
                                            #print "f:",test_data1y[k]," g:",test_y[k]
                                            if test_data1y[k]==i and (int(test_y[k]))==j:
                                                    #print "inside if:"
                                                    count_array[j] += 1
                                    confusion_matrix[i][j] = count_array[j]
                                    #print "i:",i," j:",j," value:",confusion_matrix[i][j]
```

```
fig, ax = plot_confusion_matrix(conf_mat = confusion_matrix)
plt.show()
```

Plots

2d tsne

3d tsne

Pie chart for data distribution (for every dataset)

Histogram of most important features (for every dataset)

Heatmap

Size = 1k (Using uni+bi)			
Hidden Nodes	Training Accuracy	Validation Accuracy	Test Accuracy
100(relu,alpha=0.001,iterations=770)	0.990548700446	0.882352941176	
Size = 2k			
50 (logistic, adam, alpha=0.01, iterations=)	0.993830401688	0.871148459384	
50, 100 (logistic, adam, alpha=0.01, iterations=)	0.992911525335	0.854341736695	0.863731656184

Test accuracy :

Accuracy on 5K: Bigrams

Neural Network:

One hidden layer:

Hidden Nodes	Training Accuracy	Validation Accuracy	Test Accuracy
activation: logistic solver: adam alpha: 0.001 hidden layers: 400	0.832437129977	0.66888760139	0.681146828844
activation: logistic solver: adam alpha: 0.001 hidden layers: 400	0.827059909837	0.670915411356	0.659426585578
activation: logistic solver: adam alpha: 0.01 hidden layers: 400	0.82010754440	0.68829663	0.682884448306
activation: logistic solver: adam alpha: 0.01 hidden layers: 600, 50	0.807615012764	0.652085747	0.6646394439
activation: logistic solver: adam alpha: 0.01 hidden layers: 600, 50	0.810276465157	0.676998841251	0.697654213727
hidden layers: 50 activation: logistic solver: adam alpha: 0.1	0.788876215306	0.689455388181	0.677671589922
hidden layers: 50 activation: logistic solver: adam alpha: 0.01	0.830427461843	0.679606025492	0.685490877498
hidden layers: 100 activation: logistic solver: adam alpha: 0.01	0.825213187768	0.690324449594	0.679409209383
hidden layers: 100 activation: logistic	0.836565096953	0.652375434531	0.656820156386

solver: adam alpha: 0.001			
hidden layers: 200 activation: logistic solver: adam alpha: 0.001	0.838411819021	0.67033603708	0.661164205039
hidden layers: 100 activation: logistic solver: adam alpha: 0.0001	0.84025854109	0.647740440324	0.655951346655

(WFF)

Accuracy on 5K:

Neural Network:

One hidden layer:

Hidden Nodes	Training Accuracy	Validation Accuracy
30(relu,alpha=0.1,max_irer=30)	0.9802	0.8699
50(relu,alpha=0.1)	0.9828	0.8713
100(relu,alpha=0.1)	0.9828	0.8670
200(relu,alpha=0.1)	0.9828	0.8717
200,100 (same)	0.9883	0.8650
300,100(same)	0.9872	0.8678

Test accuracy : 0.8711, precision :

Accuracy on 10K:

Neural Network:

One hidden layer:

Hidden Nodes	Training Accuracy	Validation Accuracy
30(logistic,alpha=0.1,max_irer=50)	0.9292	0.8974
50(logistic,alpha=0.1)	0.9279	0.8976
100(logistic,alpha=0.1)	0.9258	0.8958
200(logistic,alpha=0.1)	0.9249	0.8926
200,100 (same)	0.9308	0.8990
300,100(same)	0.9285	0.8596

Test Accuracy :

(TE)

Neural Network:

One hidden layer:

Hidden Nodes	Training Accuracy	Validation Accuracy
30(relu,alpha=0.1,max_irer=50)	0.7994	0.3728
30(relu,alpha=0.2,max_irer=60)	0.8308	0.3804
50(relu,alpha=0.1,max_irer=150)	0.9882	0.3525
50(relu,alpha=0.2,max_irer=100)	0.9619	0.3630
100(relu,alpha=0.1,max_irer=200)	0.9950	0.3577
100(relu,alpha=0.2,max_irer=200)	0.9896	0.3682

200(relu,alpha=0.1,max_irer=200)	0.9952	0.3600
200(relu,alpha=0.2,max_irer=200)	0.9828	0.3763
200,100(relu,alpha=0.1,max_irer=200)	0.9959	0.3989
200,100(relu,alpha=0.2,max_irer=200)	0.9961	0.3635
300,100(relu,alpha=0.1,max_irer=200)	0.9944	0.3525
300,100(relu,alpha=0.2,max_irer=200)	0.9959	0.3612

Test accuracy : 0.3600

With (200,100) relu,alpha=0.2, max_iter=200, varying beta_1

beta_1	Training Accuracy	Validation Accuracy
0.9	0.9957	0.3455
0.8	0.9957	0.3624
0.7	0.9932	0.3502
0.6	0.9961	0.3635
0.5	0.9957	0.3566
0.4	0.9961	0.3571
0.3	0.9754	0.3583
0.2	0.9961	0.3478
0.1	0.9949	0.3368

Test_accuracy : 0.3647

varying beta_2

beta_1	Training Accuracy	Validation Accuracy
0.9	0.9961	0.3659
0.8	0.9961	0.3693
0.7	0.9955	0.3635
0.6	0.9864	0.3600
0.5	0.9739	0.3757
0.4	0.8602	0.3595
0.3	0.8724	0.3275
0.2	0.7912	0.3124
0.1	0.8065	0.3577

Test Accuracy : 0.3647

Varying beta_1 and beta_2 :

beta_1	Training Accuracy	Validation Accuracy
0.9	0.9961	0.3496
0.8	0.9961	0.3490
0.7	0.9959	0.3438
0.6	0.9957	0.3322
0.5	0.9952	0.3426
0.4	0.9946	0.3426
0.3	0.9936	0.3264
0.2	0.9901	0.3182

0.1	0.9870	0.3252

Test accuracy : 0.3722