**COMP 551**

**ASSIGNMENT 3**

**AHMET AKGUL – 260 624 068**

# 2. Yelp Dataset

## Random Classifier

**Training F1-Measure:** 0.180419612228

Confusion Matrix:

[[104 102 114 97 105]

[115 134 136 124 132]

[225 197 183 195 197]

[490 496 481 487 514]

[459 486 461 479 487]]

**Validation F1-Measure:** 0.169737750897

Confusion Matrix:

[[15 16 15 22 16]

[12 21 22 21 20]

[47 33 24 28 32]

[71 74 78 67 66]

[69 72 47 55 57]]

**Test F1-Measure:** 0.186163692511

Confusion Matrix:

[[ 38 31 28 25 21]

[ 41 36 34 34 45]

[ 60 48 58 76 58]

[148 157 134 124 139]

[119 149 124 125 148]]

## Most Frequent Classifier Score

**Training F1-Measure:** 0.104267004647

Confusion Matrix:

[[ 0 0 0 522 0]

[ 0 0 0 641 0]

[ 0 0 0 997 0]

[ 0 0 0 2468 0]

[ 0 0 0 2372 0]]

**Validation F1-Measure:** 0.105014749263

Confusion Matrix:

[[ 0 0 0 84 0]

[ 0 0 0 96 0]

[ 0 0 0 164 0]

[ 0 0 0 356 0]

[ 0 0 0 300 0]]

**Test F1-Measure:** 0.103923019985

Confusion Matrix:

[[ 0 0 0 143 0]

[ 0 0 0 190 0]

[ 0 0 0 300 0]

[ 0 0 0 702 0]

[ 0 0 0 665 0]]

## Bernoulli Naive Bayes

**Training F1-Measure:** 0.577269725617

Confusion Matrix:

[[ 275 8 15 22 202]

[ 10 323 23 85 200]

[ 9 25 471 153 339]

[ 36 155 86 1211 980]

[ 53 124 73 248 1874]]

**Validation F1-Measure:** 0.330091223067

Confusion Matrix:

[[ 29 9 5 7 34]

[ 9 16 12 26 33]

[ 2 23 28 54 57]

[ 8 33 27 123 165]

[ 13 25 9 69 184]]

**Test F1-Measure:** 0.341137874213

Confusion Matrix:

[[ 42 24 5 18 54]

[ 22 46 32 31 59]

[ 9 36 44 99 112]

[ 10 58 61 246 327]

[ 10 37 27 150 441]]

## Decision Tree

**Training F1-Measure:** 1.0

Confusion Matrix:

[[ 522 0 0 0 0]

[ 0 641 0 0 0]

[ 0 0 997 0 0]

[ 0 0 0 2468 0]

[ 0 0 0 0 2372]]

**Validation F1-Measure**: 0.261495450896

Confusion Matrix:

[[ 16 15 15 14 24]

[ 14 10 24 22 26]

[ 12 18 40 57 37]

[ 19 35 56 135 111]

[ 20 15 33 115 117]]

**Test F1-Measure:** 0.279169541182

Confusion Matrix:

[[ 29 21 31 34 28]

[ 28 23 45 59 35]

[ 17 30 59 127 67]

[ 35 39 97 298 233]

[ 22 38 60 251 294]]

## Linear SVC

**Training F1-Measure**: 0.997870801179

Confusion Matrix:

[[ 520 0 0 0 2]

[ 0 640 0 0 1]

[ 0 0 996 1 0]

[ 0 0 0 2454 14]

[ 0 0 0 2 2370]]

**Validation F1-Measure:** 0.413976371409

Confusion Matrix:

[[ 36 23 8 9 8]

[ 14 25 16 27 14]

[ 7 23 50 65 19]

[ 10 11 31 171 133]

[ 4 9 18 102 167]]

**Test F1-Measure:** 0.400687403274

Confusion Matrix:

[[ 55 33 15 20 20]

[ 33 58 41 36 22]

[ 12 42 77 118 51]

[ 12 31 83 341 235]

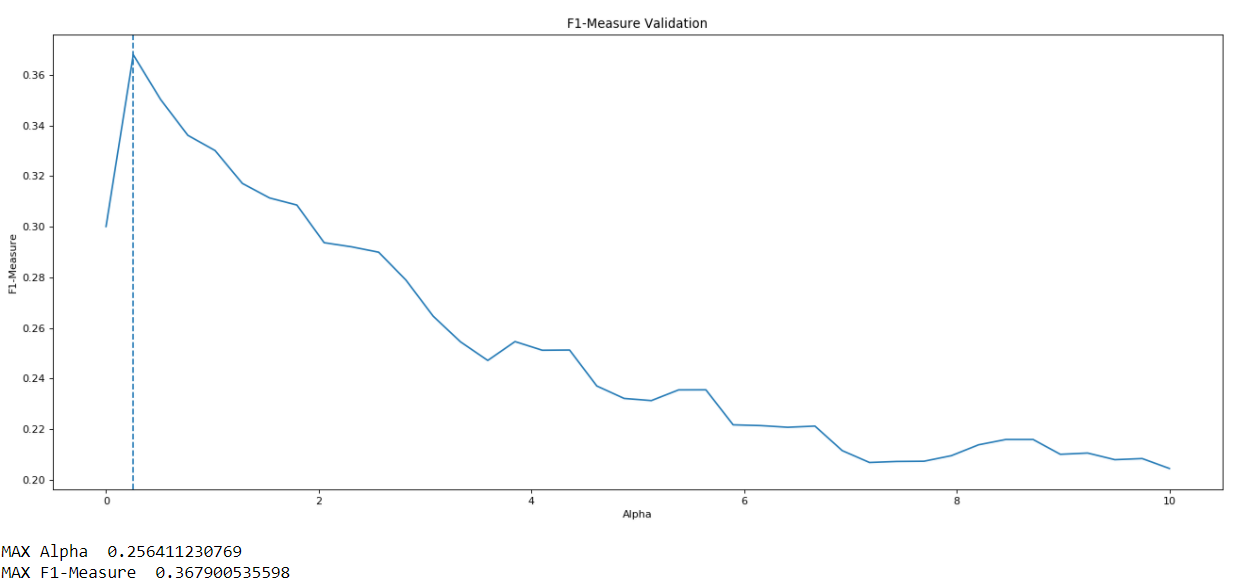
[ 15 12 41 236 361]]

# Yelp

## Bernoulli Naive Bayes Hyper-Parameter Tuning

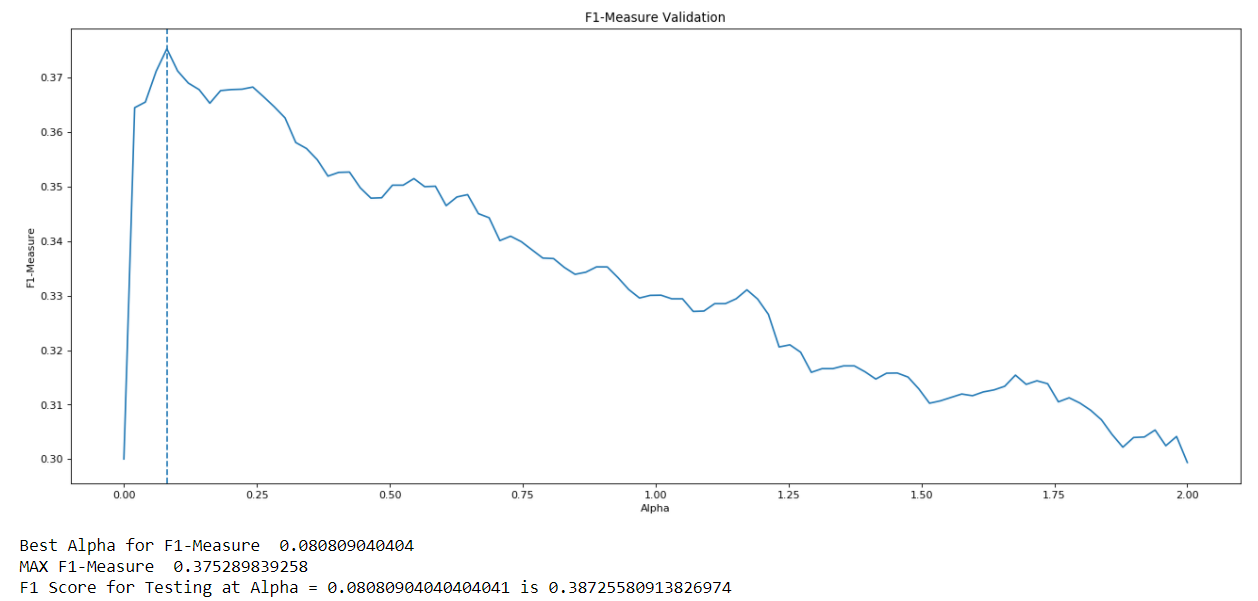
For the Bernoulli Naïve Bayes training model, there is only one parameter to tune, alpha. This alpha is the additive smoothing parameter. A value of 0 means no smoothing. We will first vary this between 0 and 20.

### Iteration 1



The higher F1-measures occur on the validation set for alpha values between 0 and 2. Now we update the iteration loop and re-run the program for values of alpha between 0 and 2.

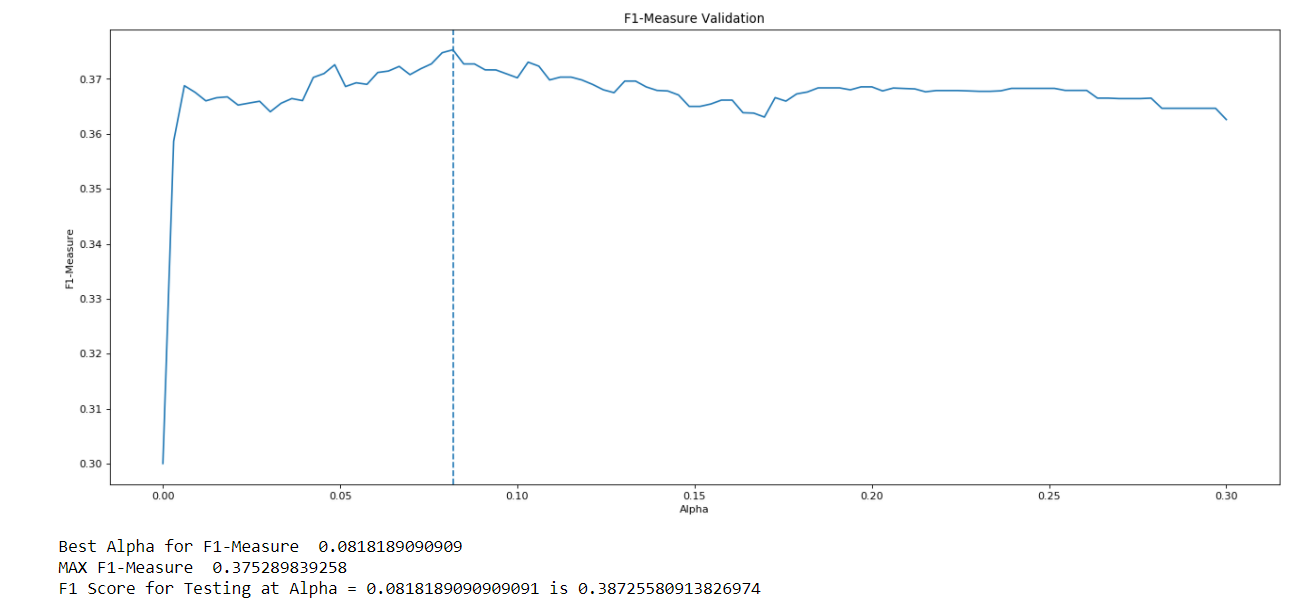
### Iteration 2



**Testing F1-Measure:** 0.3873

The testing F1-measure at alpha = 0.081 is 0.3873. We run through once more for values of alpha between 0 and 0.3.

### Iteration 3



**Testing F1-Measure:** 0.3872

At the value of alpha = 0.0818, the validation F1-measure is the highest—0.3753. At this value, the testing F1-Measure is 0.3872. This is the best tuning we can do for this model given the yelp dataset and a Bag of Words representation.

// TODO: REPORT TRAINING VALIDATION AND TESTING F1\_MEASURE FOR MODEL WITH THIS ALPHA

# Yelp

## Linear SVC Hyper-Parameter Tuning

For Linear SVC, there were more parameters to consider when tuning the model. These parameters were C, the penalty parameter of the error term, Dual vs Primal, which specifies which optimization problem to solve, loss, which specifies the loss function, max iterations, which are the maximum number of iterations the model will run while training the data, and tol, which is the tolerance for stopping criteria.

For C, I considered values from the array [0.01, 0.1, 1.0, 10, 100, 1000].

For tolerance, I considered values from the array [1e-6, 1e-5, 1e-4, 1e-3,1e-2,1e-1, 1, 10]

Because in this dataset the number of training samples (7000) is less than the number of features (10,000), I am setting Dual to true and solving the dual problem.

For maximum iterations I am considering values between the range 1000 and 10,000. The default value was 1000, this will essentially show whether increasing the iterations allows for accuracy improvements.

For loss, I am considering squared-hinge and hinged loss.

### Iteration 1

Model with rank: 1

Mean validation score: 0.486 (std: 0.003)

Parameters: {'C': 0.1, 'dual': True, 'loss': 'squared\_hinge', 'max\_iter': 2466, 'tol': 0.001}

Model with rank: 2

Mean validation score: 0.485 (std: 0.006)

Parameters: {'C': 0.1, 'dual': True, 'loss': 'hinge', 'max\_iter': 5709, 'tol': 0.001}

Model with rank: 3

Mean validation score: 0.456 (std: 0.002)

Parameters: {'C': 1.0, 'dual': True, 'loss': 'squared\_hinge', 'max\_iter': 9370, 'tol': 0.01}

This first iteration shows that the model prefers iterations that are above 1000. For the next iteration I will decrease the range. The model also seems to prefer lower tolerance values, hence we can remove some of the larger tolerance values, such as 1 and 10.

The best F1-Measure for validation is 0.486.

### Iteration 2

Model with rank: 1

Mean validation score: 0.514 (std: 0.005)

Parameters: {'C': 0.01, 'dual': True, 'loss': 'squared\_hinge', 'max\_iter': 4710, 'tol': 0.001}

Model with rank: 1

Mean validation score: 0.514 (std: 0.005)

Parameters: {'C': 0.01, 'dual': True, 'loss': 'squared\_hinge', 'max\_iter': 4066, 'tol': 0.0001}

Model with rank: 1

Mean validation score: 0.514 (std: 0.005)

Parameters: {'C': 0.01, 'dual': True, 'loss': 'squared\_hinge', 'max\_iter': 9689, 'tol': 0.001}

This iteration shows that the model generally prefers Squared Hinge loss. Hence, I will only consider that loss for the next iteration. Again, the model prefers lower tolerance values, so I will decrease the array some more.

This iteration resulted in a validation F1-Measure of 0.514.

### Iteration 3

Model with rank: 1

Mean validation score: 0.514 (std: 0.005)

Parameters: {'C': 0.01, 'max\_iter': 7449, 'tol': 1e-06}

Model with rank: 1

Mean validation score: 0.514 (std: 0.005)

Parameters: {'C': 0.01, 'max\_iter': 6358, 'tol': 0.0001}

Model with rank: 1

Mean validation score: 0.514 (std: 0.005)

Parameters: {'C': 0.01, 'max\_iter': 3775, 'tol': 0.0001}

//TODO: GET PREDICTIONS OF THE MODEL FOR WHEN THESE ARE THE PARAMETERS BEING USE

## Decision Tree Hyper Parameter Tuning

There are many parameters to tune for the decision tree. Not all these values were considered. The first parameter considered is max depth, which is the maximum depth of the tree. The default for this is infinity. I first considered values between 0 and 50.

The second parameter I considered is max features, which is the maximum number of features to consider when looking for the best split. The default for this is the number of features in the model, which is 10,000. I considered the options None, which considers n\_features, ‘sqrt’, which considers sqrt(n\_features), and ‘log2’, which considers log2(n\_features).

The third parameter I considered is max leaf nodes, which defines the maximum number of leaf nodes the model will have. The default for this is an unlimited number of leaf nodes. I first considered between 0 and 8,000 leaf nodes. This is because 8000 is essentially the maximum number of leaf nodes I could get with the given yelp training and validation data (used K-folds cross validation).

, min impurity decrease,

### Iteration 1

Model with rank: 1

Mean validation score: 0.308 (std: 0.002)

Parameters: {'max\_depth': 23, 'max\_features': None, 'max\_leaf\_nodes': 10000, 'min\_impurity\_decrease': 0, 'min\_samples\_leaf': 4, 'min\_samples\_split': 6}

Model with rank: 2

Mean validation score: 0.303 (std: 0.007)

Parameters: {'max\_depth': 23, 'max\_features': None, 'max\_leaf\_nodes': None, 'min\_impurity\_decrease': 0, 'min\_samples\_leaf': 5, 'min\_samples\_split': 8}

Model with rank: 3

Mean validation score: 0.301 (std: 0.001)

Parameters: {'max\_depth': 26, 'max\_features': None, 'max\_leaf\_nodes': None, 'min\_impurity\_decrease': 0, 'min\_samples\_leaf': 1, 'min\_samples\_split': 9}

### Iteration 2

Model with rank: 1

Mean validation score: 0.239 (std: 0.014)

Parameters: {'max\_depth': 32, 'max\_features': None, 'max\_leaf\_nodes': 1000, 'min\_impurity\_decrease': 0.0017847551550062902, 'min\_samples\_leaf': 8, 'min\_samples\_split': 4}

Model with rank: 2

Mean validation score: 0.223 (std: 0.026)

Parameters: {'max\_depth': 17, 'max\_features': None, 'max\_leaf\_nodes': 15000, 'min\_impurity\_decrease': 0.0021257207973072734, 'min\_samples\_leaf': 1, 'min\_samples\_split': 2}

Model with rank: 3

Mean validation score: 0.191 (std: 0.024)

Parameters: {'max\_depth': 10, 'max\_features': 'sqrt', 'max\_leaf\_nodes': 10000, 'min\_impurity\_decrease': 0.00048700757764565952, 'min\_samples\_leaf': 9, 'min\_samples\_split': 7}

It is clear that minimum impurity should stay at as close as possible to 0.

Iteration 3

Model with rank: 1

Mean validation score: 0.313 (std: 0.005)

Parameters: {'max\_depth': 24, 'max\_features': None, 'max\_leaf\_nodes': None, 'min\_impurity\_decrease': 0.000438451445691912, 'min\_samples\_leaf': 6, 'min\_samples\_split': 2}

Model with rank: 2

Mean validation score: 0.305 (std: 0.004)

Parameters: {'max\_depth': 25, 'max\_features': None, 'max\_leaf\_nodes': 15000, 'min\_impurity\_decrease': 0.00024863477331978133, 'min\_samples\_leaf': 3, 'min\_samples\_split': 9}

Model with rank: 3

Mean validation score: 0.301 (std: 0.008)

Parameters: {'max\_depth': 39, 'max\_features': None, 'max\_leaf\_nodes': None, 'min\_impurity\_decrease': 9.8586025656947074e-05, 'min\_samples\_leaf': 6, 'min\_samples\_split': 5}

Model with rank: 4

Mean validation score: 0.301 (std: 0.006)

Parameters: {'max\_depth': 36, 'max\_features': None, 'max\_leaf\_nodes': 5000, 'min\_impurity\_decrease': 0.00013896519546904517, 'min\_samples\_leaf': 7, 'min\_samples\_split': 2}

Model with rank: 5

Mean validation score: 0.284 (std: 0.007)

Parameters: {'max\_depth': 10, 'max\_features': None, 'max\_leaf\_nodes': 1000, 'min\_impurity\_decrease': 0.0006226190757785999, 'min\_samples\_leaf': 7, 'min\_samples\_split': 9}

Iteration 4

Model with rank: 1

Mean validation score: 0.320 (std: 0.010)

Parameters: {'max\_depth': 25, 'max\_features': None, 'max\_leaf\_nodes': 26406, 'min\_impurity\_decrease': 0.00051830527754645494, 'min\_samples\_leaf': 3, 'min\_samples\_split': 2}

Model with rank: 2

Mean validation score: 0.320 (std: 0.001)

Parameters: {'max\_depth': 28, 'max\_features': None, 'max\_leaf\_nodes': 61954, 'min\_impurity\_decrease': 0.00052115656459021263, 'min\_samples\_leaf': 7, 'min\_samples\_split': 16}

Model with rank: 3

Mean validation score: 0.319 (std: 0.001)

Parameters: {'max\_depth': 28, 'max\_features': None, 'max\_leaf\_nodes': 117155, 'min\_impurity\_decrease': 0.00054116549571744141, 'min\_samples\_leaf': 7, 'min\_samples\_split': 9}

Model with rank: 4

Mean validation score: 0.318 (std: 0.008)

Parameters: {'max\_depth': 26, 'max\_features': None, 'max\_leaf\_nodes': 95891, 'min\_impurity\_decrease': 0.00055145950091656931, 'min\_samples\_leaf': 2, 'min\_samples\_split': 7}

Model with rank: 5

Mean validation score: 0.318 (std: 0.007)

Parameters: {'max\_depth': 27, 'max\_features': None, 'max\_leaf\_nodes': 117142, 'min\_impurity\_decrease': 0.0006420976750314486, 'min\_samples\_leaf': 2, 'min\_samples\_split': 7}

Model with rank: 6

Mean validation score: 0.318 (std: 0.002)

Parameters: {'max\_depth': 22, 'max\_features': None, 'max\_leaf\_nodes': 27185, 'min\_impurity\_decrease': 0.00053376101873472536, 'min\_samples\_leaf': 8, 'min\_samples\_split': 6}

Model with rank: 7

Mean validation score: 0.316 (std: 0.003)

Parameters: {'max\_depth': 20, 'max\_features': None, 'max\_leaf\_nodes': 136732, 'min\_impurity\_decrease': 0.00050856640124573055, 'min\_samples\_leaf': 7, 'min\_samples\_split': 12}

Model with rank: 8

Mean validation score: 0.315 (std: 0.008)

Parameters: {'max\_depth': 23, 'max\_features': None, 'max\_leaf\_nodes': 143230, 'min\_impurity\_decrease': 0.00050131332373554984, 'min\_samples\_leaf': 4, 'min\_samples\_split': 16}

Model with rank: 9

Mean validation score: 0.315 (std: 0.005)

Parameters: {'max\_depth': 20, 'max\_features': None, 'max\_leaf\_nodes': 180962, 'min\_impurity\_decrease': 0.00056953429034298892, 'min\_samples\_leaf': 3, 'min\_samples\_split': 2}

Model with rank: 10

Mean validation score: 0.315 (std: 0.003)

Parameters: {'max\_depth': 27, 'max\_features': None, 'max\_leaf\_nodes': 134700, 'min\_impurity\_decrease': 0.00056086421290564526, 'min\_samples\_leaf': 8, 'min\_samples\_split': 19}

# Yelp

## Linear SVC

Model with rank: 1

Mean validation score: 0.447 (std: 0.008)

Parameters: {'C': 100, 'max\_iter': 2450, 'tol': 1e-06}

Model with rank: 2

Mean validation score: 0.447 (std: 0.006)

Parameters: {'C': 100, 'max\_iter': 9022, 'tol': 0.01}

Model with rank: 3

Mean validation score: 0.446 (std: 0.007)

Parameters: {'C': 100, 'max\_iter': 2290, 'tol': 0.01}