1. Dimensionality Reduction

i)

A **CUR decomposition** is a set of three matrices C, U, R that, when multiplied together, closely approximate a given matrix. This is similar to SVD. How the 3 matrices C,U,R is calculated is given below:

Let M be a matrix of m rows and n columns. Pick a target number of "concepts" r to be used in the decomposition. A CUR-decomposition of M is a randomly chosen set of r columns of M , which form the m x r matrix C , and a randomly chosen set of r rows of M , which form the r x n matrix R . There is also an r x r matrix U that is constructed from C and R as follows:

1. Let W be the r x r matrix that is the intersection of the chosen columns of C and the chosen rows of R . That is, the element in row i and column j of W is the element of M whose column is the j th column of C and whose row is the i th row of R .

2. Compute the SVD of W ; say W = X Σ T .

3. Compute Σ + , the Moore-Penrose pseudoinverse of the diagonal matrix . That is, if the i th diagonal element of is σ 6 = 0, then replace it by 1/σ . But if the i th element is 0, leave it as 0.

4. Let U = Y Σ (+)^2 XT .

In SVD, even if given Matrix M is sparse, U and V will be dense. Since is diagonal, it will be sparse, but is usually much smaller than U and V , so its sparseness does not help

The merit of CUR decomposition lies in the fact that if M is sparse, then the two large matrices (called C and R for "columns" and "rows") analogous to U and V in SVD are also sparse. Only the matrix in the middle (analogous to in SVD) is dense, but this matrix is small so the density does not hurt too much.

ii)

	Matrix	Alien	Star Wars	Casablanca	Titanic
Joe	1	1	1	0	0
Jim	3	3	3	0	0
John	4	4	4	0	0
Jack	5	5	5	0	0
Jill	0	0	0	4	4
Jenny	0	0	0	5	5
Jane	0	0	0	2	2

Figure 11.12: Matrix M, repeated from Fig. 11.6

The sum of squares of elements M=243.

(a) The columns for The Matrix and Alien and the rows for Jim and John.

Matrix and Alien:
Squared Frobenius norm: $1^2 + 3^2 + 4^2 + 5^2 = 51$.
Prob: 51/243 = 0.210.

So after the division by $\sqrt{(2 \times 0.210)} = 0.648$ on both Matrix and Alien, the matrix C is

$$\begin{pmatrix} 1.54 & 1.54 \\ 4.63 & 4.63 \\ 6.17 & 6.17 \\ 7.72 & 7.72 \\ 0.00 & 0.00 \\ 0.00 & 0.00 \\ 0.00 & 0.00 \end{pmatrix}$$

Row Jim's squared Frobenius norm and Probability: 27, 0.111.
Row John's squared Frobenius norm and Probability: 48, 0.198.

So after the division by $\sqrt{(2 \times 0.111)} = 0.471$ on Jim, and $\sqrt{(2 \times 0.198)} = 0.629$ on John, the matrix **R** is:

$$\begin{pmatrix} 6.37 & 6.37 & 6.37 & 0.00 & 0.00 \\ 6.36 & 6.36 & 6.36 & 0.00 & 0.00 \end{pmatrix}$$

W: $\begin{pmatrix} 3 & 3 \\ 4 & 4 \end{pmatrix}$ X: $\begin{pmatrix} -0.60 & -0.80 \\ -0.80 & 0.60 \end{pmatrix}$ Σ: $\begin{pmatrix} 7.07 & 0.00 \\ 0.00 & 0.00 \end{pmatrix}$ Y^T: $\begin{pmatrix} -0.71 & -0.71 \\ -0.71 & 0.71 \end{pmatrix}$

Σ^+: $\begin{pmatrix} 0.141 & 0.000 \\ 0.000 & 0.000 \end{pmatrix}$ Σ^{+2}: $\begin{pmatrix} 0.020 & 0.000 \\ 0.000 & 0.000 \end{pmatrix}$

$U = Y (\Sigma^+)^2 X^T =$

$$\begin{pmatrix} -0.71 & -0.71 \\ -0.71 & 0.71 \end{pmatrix} \begin{pmatrix} 0.020 & 0.000 \\ 0.000 & 0.000 \end{pmatrix} \begin{pmatrix} -0.60 & -0.80 \\ -0.80 & 0.60 \end{pmatrix} = \begin{pmatrix} 0.009 & 0.011 \\ 0.009 & 0.011 \end{pmatrix}$$

So, CUR = $\begin{pmatrix} 0.390 & 0.390 & 0.390 & 0.000 & 0.000 \\ 1.172 & 1.172 & 1.172 & 0.000 & 0.000 \\ 1.562 & 1.562 & 1.562 & 0.000 & 0.000 \\ 1.954 & 1.954 & 1.954 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \end{pmatrix}$

(b) The Columns: Alien and Star Wars; The rows: Jack and Jill.

Alien and Star Wars:
Squared Frobenius norm: $1^2 + 3^2 + 4^2 + 5^2 = 51$.
Their Probabilities: 51/243 = 0.210.

So after the division by $\sqrt{(2 \times 0.210)} = 0.648$ on both Alien and Star Wars, the matrix **C** is:

$\begin{pmatrix} 1.54 & 1.54 \\ 4.63 & 4.63 \\ 6.17 & 6.17 \\ 7.72 & 7.72 \\ 0.00 & 0.00 \\ 0.00 & 0.00 \\ 0.00 & 0.00 \end{pmatrix}$

Row Jack's squared Frobenius norm and Probability: 75, 0.309.
Row Jill's squared Frobenius norm and Probability: 32, 0.132.

So after the division by $\sqrt{(2 \times 0.309)} = 0.786$ on Jack, and $\sqrt{(2 \times 0.132)} = 0.514$ on Jill, the matrix **R** is:

$\begin{pmatrix} 6.36 & 6.36 & 6.36 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 7.78 & 7.78 \end{pmatrix}$

W: $\begin{pmatrix} 5 & 5 \\ 0 & 0 \end{pmatrix}$ X: $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ Σ: $\begin{pmatrix} 7.071 & 0.000 \\ 0.000 & 0.000 \end{pmatrix}$ Y^T: $\begin{pmatrix} 0.707 & 0.707 \\ -0.707 & 0.707 \end{pmatrix}$

Σ^+: $\begin{pmatrix} 0.141 & 0.000 \\ 0.000 & 0.000 \end{pmatrix}$ Σ^{+2}: $\begin{pmatrix} 0.020 & 0.000 \\ 0.000 & 0.000 \end{pmatrix}$

$U = Y (\Sigma^{+})^2 X^T = \begin{pmatrix} 0.707 & -0.707 \\ 0.707 & 0.707 \end{pmatrix} \begin{pmatrix} 0.020 & 0.000 \\ 0.000 & 0.000 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 0.014 & 0.000 \\ 0.014 & 0.000 \end{pmatrix}$

So, CUR=

$$\begin{pmatrix} 0.277 & 0.277 & 0.277 & 0.000 & 0.000 \\ 0.833 & 0.833 & 0.833 & 0.000 & 0.000 \\ 1.110 & 1.110 & 1.110 & 0.000 & 0.000 \\ 1.389 & 1.389 & 1.389 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \end{pmatrix}$$

(c) The Columns: Matrix and Titanic; The rows: Joe and Jane.

Matrix:
S.F. Norm: $1^2 + 3^2 + 4^2 + 5^2 = 51$.
Probability: 51/243 = 0.210.

Titanic:
S.F. Norm: $4^2 + 5^2 + 2^2 = 45$
Probability: 45/243 = 0.185.

So after the division by $\sqrt{(2 \times 0.210)} = 0.648$ on Matrix and by $\sqrt{(2 \times 0.185)} = 0.608$ on Titanic, the matrix **C** is:

$$\begin{pmatrix} 1.543 & 0.000 \\ 4.630 & 0.000 \\ 6.173 & 0.000 \\ 7.716 & 0.000 \\ 0.000 & 6.579 \\ 0.000 & 8.224 \\ 0.000 & 3.289 \end{pmatrix}$$

Row Joe's squared Frobenius norm and Probability: 3, 0.012.
Row Jane's squared Frobenius norm and Probability: 8, 0.033.

So after the division by $\sqrt{(2 \times 0.012)} = 0.155$ on Joe, and $\sqrt{(2 \times 0.033)} = 0.257$ on Jane, the matrix **R** is:

$$\begin{pmatrix} 6.452 & 6.452 & 6.452 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 7.782 & 7.782 \end{pmatrix}$$

W: $\begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix}$ X: $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ Σ: $\begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix}$ Y^T: $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$

Σ^+: $\begin{pmatrix} 0.50 & 0.00 \\ 0.00 & 1.00 \end{pmatrix}$ Σ^{+2}: $\begin{pmatrix} 0.25 & 0.00 \\ 0.00 & 1.00 \end{pmatrix}$

$U = Y (\Sigma^+)^2 X^T = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 0.25 & 0.00 \\ 0.00 & 1.00 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} 1.00 & 0.00 \\ 0.00 & 0.25 \end{pmatrix}$

So, CUR

$$\begin{pmatrix} 9.96 & 9.96 & 9.96 & 0.00 & 0.00 \\ 29.87 & 29.87 & 29.87 & 0.00 & 0.00 \\ 39.82 & 39.82 & 39.82 & 0.00 & 0.00 \\ 49.78 & 49.78 & 49.78 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 12.80 & 12.80 \\ 0.00 & 0.00 & 0.00 & 16.00 & 16.00 \\ 0.00 & 0.00 & 0.00 & 6.40 & 6.40 \end{pmatrix}$$

---

2. Item Based Collaborative Filtering (Total: 6 points)

	Movie A	Movie B	Movie C	Movie D
Ryan	1	5	3	5
Stavros			2	
Brahma		4	1	1
Brodie	4	3		2
Zosimus		5	1	

(i) Compute the item similarity matrix (denoted by R) using Pearson Correlation as a measure.

$$\text{corr}(i, j) = \frac{\sum_{\text{users } u} (R_{u,i} - \overline{R_i}) \sum_{\text{users } u} (R_{u,j} - \overline{R_j})}{\sqrt{\sum_{\text{users } u} (R_{u,i} - \overline{R_i})^2} \sqrt{\sum_{\text{users } u} (R_{u,j} - \overline{R_j})^2}}$$

corr(A,B) : avg(Movie A) = 2.5 , avg(Movie B) = 2 [considering ratings given by common users]

User	Movie A	Movie B
Ryan	-1.5	1
Brodie	1.5	-1

$$\frac{(-1.5 * 1) + (1.5 * -1)}{\sqrt{-1.5^2 + 1.5^2}\sqrt{1^2 + (-1)^2}}$$

which is equal to -1

Similarly correlation between all the movies are calculated and similarity Matrix R is generated.

R	Movie A	Movie B	Movie C	Movie D
Movie A		-1	0	-1
Movie B	-1		0.5	0.72
Movie C	0	0.5		1
Movie D	-1	0.72	1	

(ii) Compute the item similarity matrix (denoted by S) using Jaccard Coefficient as a measure.

J(A,B) = 2/4 J(A,C) = 1/5 J(A,D) = 2/3

J(B,C) = 3/ 5 J(B,D) = 3/4 J(C,D) = 2/5

S	Movie A	Movie B	Movie C	Movie D
Movie A		0.5	0.2	0.667

Movie B	0.5		0.6	0.75
Movie C	0.2	0.6		0.4
Movie D	0.667	0.75	0.4	

(iii) Use matrices R and S to compute the corresponding ratings that Zosimus would give to Movies A & D.

Predicted rating Zosimus would give to Movies A & D

Using Pearson Similarity Matrix

Rating(Zosimus, A) =((-1*5) + (0 * 1)) / (1+0) = -5

Rating(Zosimus, D) = (0.72*5) + (1 * 1) / (0.72+1) = 2.674

Using Jaccard Similarity Matrix

Rating(Zosimus, A) = ((0.5*5) + (0.2* 1)) / (0.5+0.2) = 3.857

Rating(Zosimus, D) = (0.75*5) + (0.4 * 1) / (0.75 + 0.4) = 3.608

Zosimus	Movie A	Movie D
Pearson coeff	-5	4.18
Jaccard coeff	3.857	3.608

(iv) Use matrices R and S to compute the corresponding ratings that Stavros would give to Movies A & B.

Using Pearson Similarity Matrix

Rating(Starvos, A) = (0 * 2) = 0

Rating(Starvos, B) = (0.5*2)/(0.5) = 2

Using Jaccard Similarity Matrix

Rating(Starvos, A) = (0.5)*2/0.5 = 2

Rating(Starvos, B) = (0.6*2)/0.6 = 2

Stavros	Movie A	Movie B
Pearson coeff	0	2
Jaccard coeff	2	2

3. Mapreduce (Total: 9 points)

i)

R		S	
A	B	B	C
0	1	0	1
1	2	1	2
2	3	2	3

1. Sol: **d) (3, [(R, 2)])**

   For each tuple (a, b) of R , produce the key-value pair (b, (R, a)) .
   (1, (R,0))
   (2, (R,1))
   (3, (R,2))

   For each tuple (b, c) of S , produce the key-value pair (b, (S, c)).
   (0,(S,1))
   (1,(S,2))
   (2,(S,3))

   After reconstruction(grouping)
   (0, [(S, 1)])
   (1, [(R, 0), (S, 2)])
   (2, [(R, 1),(S, 3)])
   (3, [(R, 2)])

   `

Hence the solution is (3, [(R, 2)])

(ii) Sol: **b) (3,90)**

In Map function, we output the prime factor of each integer

15 -> (3,15), (5,15)
21 -> (3,21), (7,21)
24 -> (2,24), (3,24)
30 -> (2,30), (3,30), (5,30)
49 -> (7,49)

In reduce function, we sum up the integers for each prime number as
(2, 24+30)
(3, 15+21+24+30)
(5, 15+30)
(7, 21+49)

Hence the solution is (3,90)

(iii) Sol: **c) (1,9)**

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

1
2
3
4

From each matrix element mij it produces the key-value pair (i, mijvj). Thus, all terms of the sum that make up the component xi of the matrix-vector product will get the same key, i .

The output of the map function will be

(1, 1) , (1,4), (1,9), (1, 16)
(2, 5), (2,12), (2, 21), (2, 32)
(3, 9), (3, 20), (3, 33), (3, 48)
(4,13), (4, 28), (4, 45), (4, 64)

Hence (1,9) is the solution

4. **Naive Bayes Classification**

i) Model:

Naive Bayes classifier models a joint distribution over the labels of the newsgroup and the data(which are the possible strings appearing in the news article). We calculate the probability of test data for each class and we assign it to the class with maximum probability.

$$argmax_y \log P(y|f_1 \dots f_m) = argmax_y \left\{ \log P(y) + \sum_{i=1}^{m} \log P(f_i|y) \right\}$$

Assumption:

- A uniform prior is assumed across all classes as given in the problem statement. Hence the prior term is ignored in the calculation.
- Features are assumed to be conditionally independent

Experiments:

Tested for different values of smoothing parameter from 0 to 5.
For smoothing parameter 1.0 maximum accuracy is obtained.

Smoothing Parameter	Accuracy
0	15.23
1	85.34
2	84.19
3	82.42
4	80.69
5	79.02

ii)

Step 1: Training

termcounts: counted by grouping first by labels and then by strings and then sum the number of occurance of each string. Output is Tuple3 <label, string, count>

```
DataSet<Tuple3<String, String, Long>> termCounts = labeledTerms.groupBy(0,1).sum(2);
```

termlabelcounts: which is total number of words in each label can be calculated by grouping by label and then sum the counts of all strings under that label and selecting/projecting 0th and 2nd field of the tuple which is nothing but label and count of all strings under that label.

```
DataSet<Tuple2<String, Long>> termLabelCounts = labeledTerms.groupBy(0).sum(2).project(0, 2);
```

```
// read input with df-cut
DataSet<Tuple3<String, String, Long>> labeledTerms = input.flatMap(new DataReader());

// conditional counter per word per label
DataSet<Tuple3<String, String, Long>> termCounts = labeledTerms.groupBy(0,1).sum(2);

termCounts.writeAsCsv(Config.pathToConditionals(), "\n", "\t", FileSystem.WriteMode.OVERWRITE);

// word counts per label
DataSet<Tuple2<String, Long>> termLabelCounts = labeledTerms.groupBy(0).sum(2).project(0, 2);
termLabelCounts.writeAsCsv(Config.pathToSums(), "\n", "\t", FileSystem.WriteMode.OVERWRITE);
```

Step 2: Classification

In the class classifier,

The broadcast variables conditionals and sums are received and transformed to a map from tuple. Also smoothing parameter is received as a variable closure.

```
public Classifier(double smoothingParameter) { this.smoothing = smoothingParameter; }
@Override
public void open(Configuration parameters) throws Exception {
    super.open(parameters);

    final List<Tuple3<String, String, Long>> conditionals = getRuntimeContext().getBroadcastVariable("conditionals");
    final List<Tuple2<String, Long>> sums = getRuntimeContext().getBroadcastVariable("sums");
    initCountPerCategory(sums);
    initConditionalWordCounts(conditionals);

}
```

```
private void initCountPerCategory(List<Tuple2<String, Long>> input) {
    for (Tuple2<String, Long> tuple : input) {
        wordSums.put(tuple.f0, tuple.f1);
    }
}

private void initConditionalWordCounts(List<Tuple3<String, String, Long>> input) {

    Set<String> distinctWords = new HashSet<>();

    for (Tuple3<String, String, Long> tuple : input) {
        distinctWords.add(tuple.f1);
        if(wordCounts.containsKey(tuple.f0))
        {
            Map<String, Long> valMap = wordCounts.get(tuple.f0);
            valMap.put(tuple.f1,tuple.f2);
            wordCounts.put(tuple.f0,valMap);
        }
        else
        {
            Map<String, Long> valMap = new HashMap<>();
            valMap.put(tuple.f1,tuple.f2);
            wordCounts.put(tuple.f0,valMap);
        }

    }

}
```

We apply the classifier on each datapoint from testdata.tab

```
DataSet<Tuple3<String, String, Double>> classifiedDataPoints = testData.map(new Classifier(smoothingParameter))
        .withBroadcastSet(conditionals, "conditionals")
        .withBroadcastSet(sums, "sums");

classifiedDataPoints.writeAsCsv(Config.pathToOutput(), "\n", "\t", FileSystem.WriteMode.OVERWRITE);
```

We can identify the label of the test set by calculating the probability and taking a log of it .

Conditional probability of a term is calculated as

$$\hat{P}(w|Y=y) = \frac{count(w,y)}{\sum_{w'} count(w',y)}$$

And then log is applied. Here Prior factor is removed as we consider all labels have equal prior probability.

$$argmax_y \log P(y|f_1 \dots f_m) = argmax_y \left\{ \log P(y) + \sum_{i=1}^{m} \log P(f_i|y) \right\}$$

This is implemented as below and results are updated in csv.

```
@Override
public Tuple3<String, String, Double> map(String line) throws Exception {

    String[] tokens = line.split("\t");
    String label = tokens[0];
    String[] terms = tokens[1].split(",");

    double maxProbability = Double.NEGATIVE_INFINITY;
    String predictionLabel = "";

    for (String labelVal : wordCounts.keySet()) {
        double logProb = calculateLogProbability(labelVal, terms);
        if (logProb > maxProbability) {
            maxProbability = logProb;
            predictionLabel = labelVal;
        }
    }


     return new Tuple3<String, String, Double>(label, predictionLabel, maxProbability);
}

private double calculateLogProbability(String label, String[] words) {
    double logProb = 0.0;//= priors.get(label);
    Map<String, Long> countsPerLabel = wordCounts.get(label);

    Double num;
    for (String word : words) {
        Long count_word = countsPerLabel.get(word);
        if (count_word != null) {
            num = count_word + this.smoothing;
        } else {
            num = this.smoothing;
        }
        Long den = wordSums.get(label);
        if (den == null)
            den = 0L;
        double count_wihout_word =  den - num + ((wordSums.size()-1)*this.smoothing);

        logProb += Math.log(num/count_wihout_word);
    }

    return logProb;
```

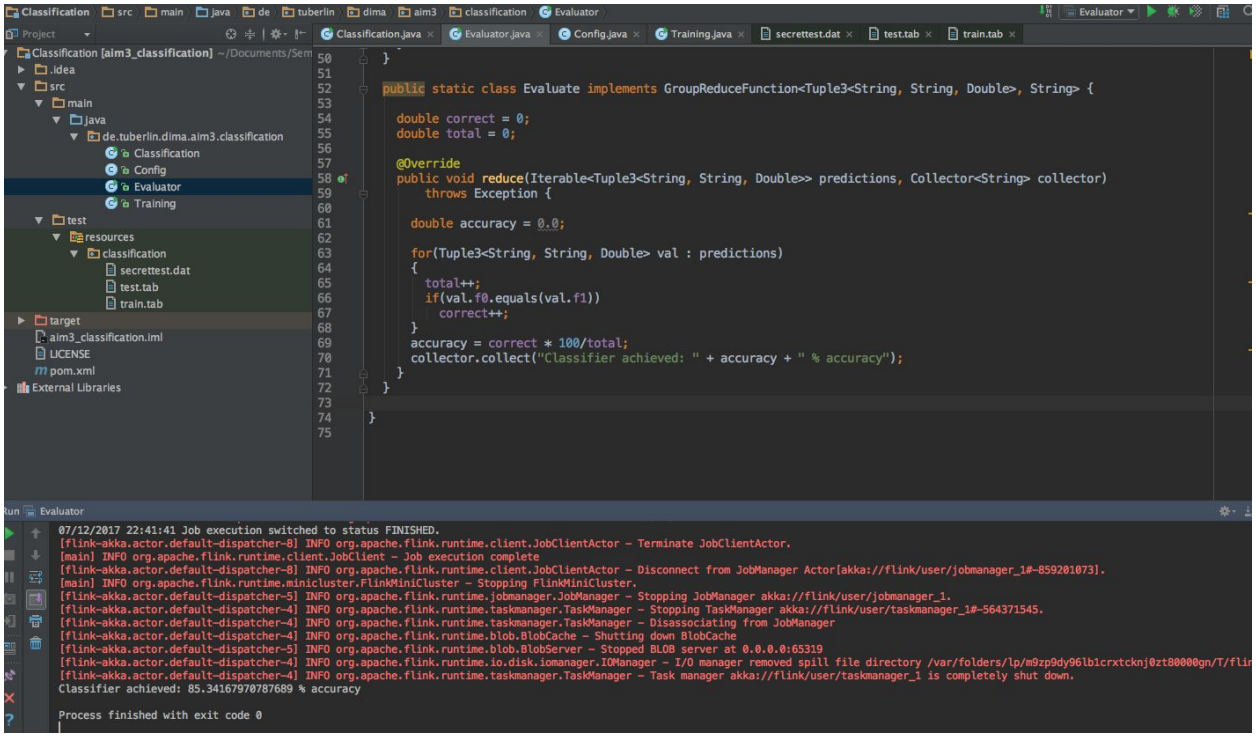
Step 3: Evaluation

The predicted label is evaluated with actual label and accuracy is calculated

```
public static class Evaluate implements GroupReduceFunction<Tuple3<String, String, Double>, String> {

  double correct = 0;
  double total = 0;

  @Override
  public void reduce(Iterable<Tuple3<String, String, Double>> predictions, Collector<String> collector)
      throws Exception {

    double accuracy = 0.0;

    for(Tuple3<String, String, Double> val : predictions)
    {
      total++;
      if(val.f0.equals(val.f1))
        correct++;
    }
    accuracy = correct * 100/total;
    collector.collect("Classifier achieved: " + accuracy + " % accuracy");
  }
}
```

iii)

85% accuracy obtained with the resource file test.tab



After testing the classifier with secrettest.dat, output of classifier looked like

[1] comp.sys.ibm.pc.hardware -490.6769926937943

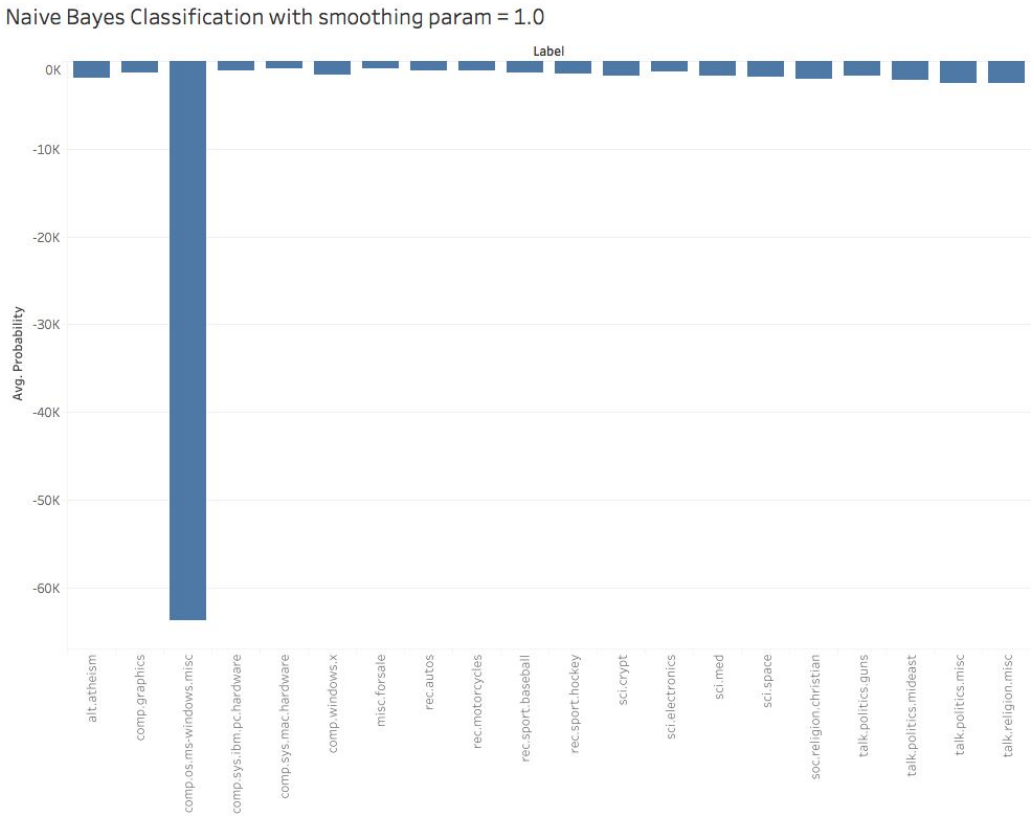
[2] comp.sys.ibm.pc.hardware -384.8255057891126

[3] rec.motorcycles -233.0300912679362

[4] talk.politics.mideast -329.5014480778917

[5] comp.windows.x -2359.6030387734395

iv)Plot of Naive Bayes Classifier with labels vs Avg Probability



Plot of Naive Bayes Classifier with labels vs Count