**ITCS 6162 FINAL PROJECT REPORT**

**(KDD)**

*Group 8*

**Bhargavi Arcot Ramesh – 801079105**

**Likhith Kumar Reddy Suram – 801085629**

**Manisha Sathanur – 801069595**

**Venkata Sai Santosh Perumalla - 801029180**

*University of North Carolina at Charlotte*

**

**Abstract**

Here in this project we have hand-selected 500 articles for the HITs from five health topics:

Diabetes, Depression, Nutrition, Cancer, and Children’s Health. For each topic, there are 100 articles with the focus on everyday reading, avoiding those research-oriented articles with many chemistry symbols and medical terminologies. Foreach article, we have three different Turkers to rate on it to get reliable ratings from different individuals. therefore, we have 1,500 (500 x 3) HITs in total at MTurk. Here we are predicting the average surprise rating and Individual surprise rating for all the articles from the feature set of relevant attributes. We have evaluated our dataset by building models and finding out what needs to be done in a better way for a good outcome.

**Introduction**

Here we have hand-selected 500 articles for the HITs from five health topics. Each article is rated by 3 individuals r1, r2 and r3 on a scale of 5, for surprise factor of the article, familiarity with the category the article covers and how much they like the article.

**Problem Statement**

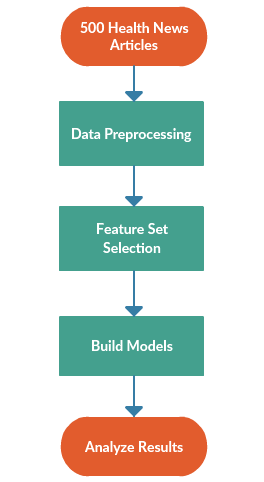
Based on the given corpus we need to find:

1. The average surprise rating from the three raters for each article using your proposed feature set.
2. Each individual rater surprise for an article based on your proposed features.

**Data Set**

1. *500MNTNews:* 500 pieces of health news in plain text format
2. *ground\_truth\_one\_row\_per\_article.csv*: The codebook for each field is:
   1. **article\_id**: a unique ID number assigned to each article which can be matched to *500MNTNews*
   2. **link:** the article’s URL link
   3. **title:** the article’s title
   4. **main\_category:** the article’s main topic
   5. **category\_k:** could be category\_1, category\_2, category\_3, or category\_4, representing the related health topic(s) of the article
   6. **surprise\_ri:** the rater ri’s rating on how surprising the article is on a 5-point Likert scale, 1 means very not surprising and 5 means very surprising
   7. **like\_ri:** the rater ri’s rating on how much she or he likes the article on a 5-point Likert scale, 1 means very little and 5 means very much
   8. **ck\_familarity\_ ri:** the rater ri’s rating on how familiar she or he is with the category\_k on a 5-point Likert scale, 1 means very little and 5 means very much

**Approach**



**Data Preprocessing**

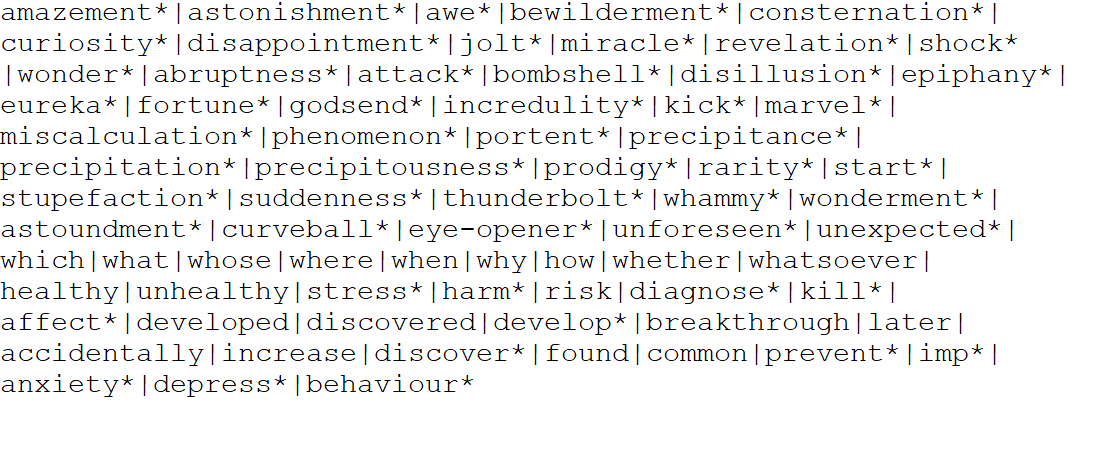
The first step for any analysis is pre-processing of the given data. The *500MNTNews* articles consists of various spaces, stop words such as I, we, him, he etc. and special characters. We used ***nltk*** package’s *stopWordRemoval* and *removeSpecialCharacters* to remove stop words from the given articles.

Another main objective of data preprocessing is to handle null values. The given *ground\_truth\_one\_row\_per\_article* file has several missing values for category\_2, category\_3 and for c2\_familiarity and c3\_familiarity for r1, r2 and r3 respectively. We have replaced the missing values with the mean of their respective column so that it would not affect the overall value.

**Feature Set Selection**

1. **Text Mining words frequency**

We are taking a bag of *surprise* words, *‘wh’* words and a set of *intuitive* words which we feel are important for surprise aspect after reading a set of files and passing it through the entire corpus for finding the word count and we are naming that attribute as ‘surprise\_count’ which can be one of the attributes in the feature set selected.



1. **Like Average**

As we need to find the average surprise rating for every rater we need to also consider their like towards the article which is helpful in finding the accuracy for the model.

1. **Familiarity related to article**

We also need to consider the familiarity because if the rater is familiar with the topic and if he finds anything new related to article there might be a chance for him to get surprised. So, this field is helpful for accuracy.

**Models Built**

**Support Vector Model**

Support Vector Model is a supervised machine learning algorithm which can be used for classification or regression analysis. For this model we used *sklearn.svm* package. In this algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a coordinate. Then, we perform classification by finding the hyperplane.

**Logistic Regression**

Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables. In the code we used *sk.learn\_linearmodel* package.

**K-Nearest Neighbours**

KNN algorithm will search through the training dataset for the k-most similar instances. In the code we used *sklearn.Neighbours*. The prediction attribute of the most similar instances is summarized and returned as the prediction for the unseen instance.

**Naive Bayes Classifier**

Naive Bayes classifiers are a collection of classification algorithms based on **Bayes Theorem**. Here, we use *sklearn.naive\_bayes* package. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other.

All the above models are built for each task on different feature sets respectively and are evaluated accordingly.

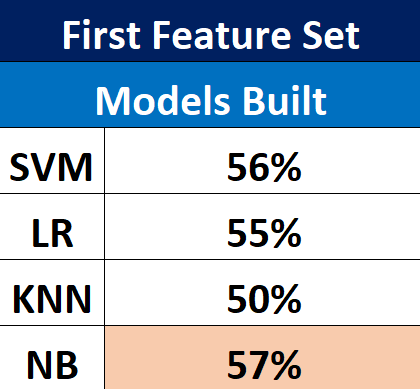
**Evaluation of results**

1. **Average Surprise rating**

1st Feature set used:

1. like\_avg – Average like for each article
2. c1\_familiarity – Category 1 familiarity
3. surprise\_count – Bag of words mined from articles.

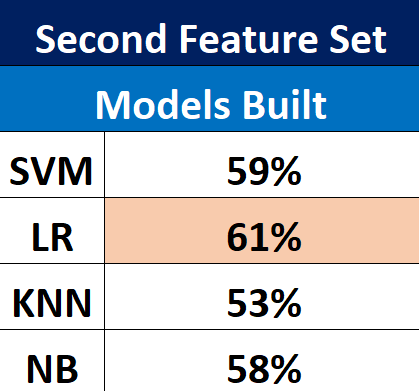
*'like\_avg','c1\_familiarity\_r1','c1\_familiarity\_r2','c1\_familiarity\_r3','surprise\_count’*



2nd Feature set used:

1. like\_avg – Average like for each article
2. c1\_familiarity – Category 1 familiarity
3. c2\_familiarity – Category 2 familiarity
4. surprise\_count – Bag of words mined from articles.

*'like\_avg','c1\_familiarity\_r1','c1\_familiarity\_r2','c1\_familiarity\_r3','c2\_familiarity\_r1','c2\_familiarity\_r2','c2\_familiarity\_r3','surprise\_count’*



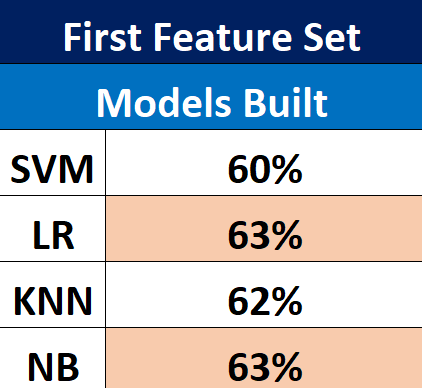
1. **Individual Surprise rating**

Here we have a corpus of 1500 rows (500\*3) for every single rater.

1st Feature set used

1. frequency – Contains the word frequency of 1500 rows
2. like – Contains the like rating of every rater
3. c1\_familiarity – Contains familiarity of every rater

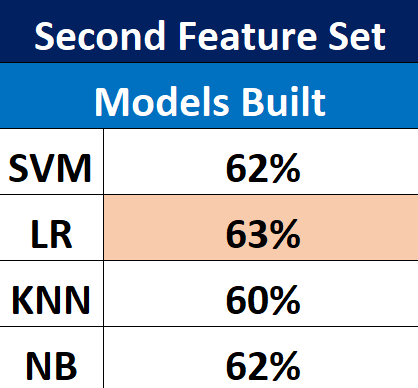
*'frequency','like','c1\_familiarity'*



2nd Feature set used:

1. frequency – Contains the word frequency of 1500 rows
2. like – Contains the like rating of every rater
3. c1\_familiarity – Contains familiarity of Category 1
4. c2\_familiarity – Contains familiarity of Category 2

*'frequency','like','c1\_familiarity', 'c2\_familiarity'*



**Challenges Faced**

Through trial and error method we considered different feature sets and got accuracy ranging from 20 moving up to 60%.

Initially we had trouble figuring out an apt bag of words for our analyzing the surprise ratings for the various articles.

We first tried to use synonyms of the word surprise from thesaurus such as awestruck, shock, eyeopener, jolt etc. We then build a linear regression model using the frequency count of these words along with the other features mentioned in the feature set above. This resulted in an accuracy as low as 20%. In order get a better accuracy we later tried adding the words starting with *‘wh’* such as who, what., wonder and so on. We also added the word *“How”.* We added these words because if an article asked questions such as who, what, how and answered them it would surprise the readers more.

This increased the accuracy from 20 to 40% but was still not satisfactory. We then tried intuitive data selection by reading articles with very high and low ratings and comparing the words in them. This helped us come up with our final list of words which consists of synonyms of the word surprise, words starting with ‘*wh’, ‘how’* and 13 words such as *accidentally, disease, depress, prevent* etc.

Finally, we ended up with an accuracy of 60%. This accuracy can further be improved by performing further intuitive analysis on the articles to get more words.

Another alternative could be to use topic modeling by creating a bag of words based on TF-IDF matrix and by getting some more relevant topics based on that method. This might result in a higher accuracy as the words selected for counting the frequency would be more accurate.

**Future Scope**

We can still do a lot better prediction if we can do further mining of data which can provide clear results. We can do a lot more by categorizing the data into several topics and by performing several models on them.

**References**

<https://www.thesaurus.com/browse/surprise>

<https://scikit-learn.org/stable/modules/naive_bayes.html>

<https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html>

<https://scikit-learn.org/stable/modules/svm.html>

<https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html>

**Code Repository**

<https://github.com/Likhith185/KDDFinalProject>