Identifying and Categorizing Offensive Language in Social Media

Kawthar Al Hinai, Lama Alsharif, Ibtisam Alshammari, Abdullateef Shittu, Joel Deladem Klo
Faculty of Engineering and Physical Sciences, School of Computing
University of Leeds, Leeds, United Kingdom

mm19kaah@leeds.ac.uk | sc19laa@leeds.ac.uk | ml18ikfa@leeds.ac.uk
sc19aas@leeds.ac.uk | sc19jdk@leeds.ac.uk

Abstract

With a lot of high-profile sackings of top CEOs because of past tweets and some celebrities feeling the heat from cancel culture, these cases emphasize the difficulties that employers, agents etc have to deal with in the social media age. Being able to classify tweets with the help of natural language processing techniques fills a void. Many entities would invest a lot to add this skill to a background check on a potential employee or protect their social or private network from getting exposed to offensive tweets. Differentiating millions of tweets is definitely a challenge but with the help of NLP techniques leveraging bag of words, tokenizers and WEKA classifiers, This project will showcase tested and evaluated models that automatically classifies offensive tweets in a given dataset into offensive or not offensive, targeted or untargeted and if targeted, classifies them into groups or individuals. After comparative analysis and fine tuning our best model, an accuracy of 80% was achieved upon applying the sequential minimal optimization classifier.

1 Introduction

Data mining is known as the process of identifying the undefined relationship in a data set to solve a problem and achieve certain results. This is done through analysing the previous data patterns, which in return helps analysing the current situation and predict future behaviour. Moreover, identifying future behaviour will help us tackle the issue before it occurs.

Social media is one of the places that people use to express their feelings and emotions and sometimes take the advantage that they can post anonymously. This allows them to use not appropriate and abusive language which may have a lot of undesired consequences.

SemEval 2019 is an international workshop on Semantic evaluation, and it aims in solving several tasks related to solving computational semantic issues. In this paper, we aim at solving task 6 of the SemEval using computational methods which consists of the following sub-tasks:

- Identifying abusive tweets
- Categorizing the offensive types automatically
- Identifying the offense target

To solve this, the CRISP-DM model data mining process was used.

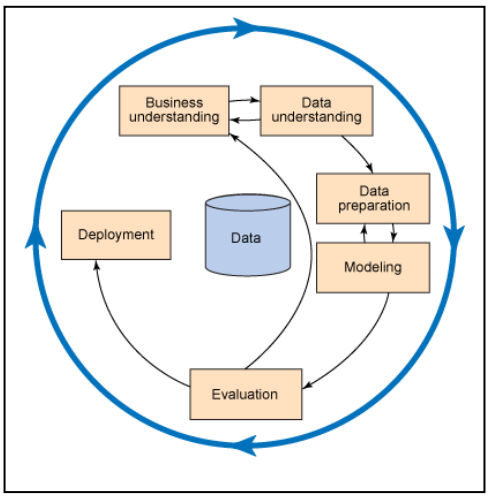

Figure 1: CRISP Data Mining Process

2 Methodology

The methods for achieving this will be through exploring different types of classifiers using a specific training set and training a number of models on the prepared training data. After that, evaluate the classifying models on a given test data to identify which will more successful in identifying the following criteria.

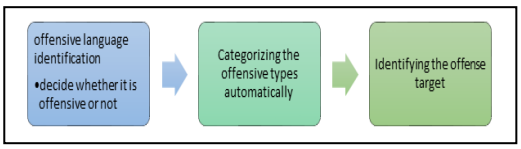

Figure 2: Tasks

The tools that will be used to solve the classification tasks are:

- WEKA
- Tableau
- Python (Pandas, Numpy, keras, sci-kit learn)

2.1 Business Understanding

In this section, the paper aims to identify the problem and objective of our task. Also, classify whether it is a supervised task or unsupervised. If the issue is supervised, then we need to verify the following:

- Target variable and whether its defined in an appropriate way
- What values the target can take
- If modelling the target variable will help to solve our business problem.

However, if we decide that the problem is unsupervised then we need define where the analysis is going. Answering the above questions will help identify the right tools and the roadmap for data mining process. [Provost and Fawcett, 2013]

In this paper, we aim to identify the offensive and abusive language from various types of tweets and determine whether they are targeted to an individual or group of people. Realizing that abusive words can cause serious issues to the society, makes it an important matter to address.

2.2 Data Understanding

In the stage of understanding the data, we get to know the existing data to help us move to the next step. In which problems that may affect the results are solved, as the data provided to us are offensive tweets located on Twitter and are in the form of TSV and.CSV files with UTF-8 encoding. This data is divided into two files, the training file, and a test file, and we find that the two files have the same format, which is TSV.

File name	File size	No. of records	No. of attributes
olid-training-v1.0.tsv	1929 KB	13240	5
testset-levela.tsv	132 KB	860	2
testset-levelb.tsv	35 KB	240	2
testset-levelc.tsv	33 KB	213	2
labels-levela.csv	10 KB	860	2
labels-levelb.csv	3 KB	240	2
labels-levelc.csv	3 KB	213	2

Figure 3: Profile of given data

The table in Figure 3 shows a short profile of the given data; the file types, number of records and number of attributes). The given datasets were all UTF-8 encoded.

In addition, tweets are classified into three categories subtask_a class, subtask_b class, and subtask_c class. Besides, tweets are classified into three categories subtask_a, subtask_b, and subtask_c. The purpose is to know the classification of offensive and non-offensive tweets, as they are denoted by OFF (offensive) and, NOT (not offensive). To import data into Weka, we need to convert it to a valid format. To get it, we have to delete characters and symbols that are not required, such as URL and @USER.

```
train_data=pd.read_csv('olid-training-v1.0.tsv', delimiter='\t', encoding='utf-8')

print(train_data)
```

```
          id                                              tweet subtask_a subtask_b subtask_c
0      86426  @USER She should ask a few native Americans wh...       OFF       UNT       NaN
1      90194  @USER @USER Go home you're drunk!!! @USER #MAG...       OFF       TIN       IND
2      16820  Amazon is investigating Chinese employees who ...       NOT       NaN       NaN
3      62688  @USER Someone should've taken" this piece of sh...       OFF       UNT       NaN
4      43605  @USER @USER Obama wanted liberals &amp;amp; illega...       NOT       NaN       NaN
...      ...                                                ...       ...       ...       ...
13235  95338  @USER Sometimes I get strong vibes from people...       OFF       TIN       IND
13236  67210  Benidorm ✅  Creamfields ✅  Maga ✅  Not too sh...       NOT       NaN       NaN
13237  82921  @USER And why report this garbage.  We don't g...       OFF       TIN       OTH
13238  27429                                @USER Pussy       OFF       UNT       NaN
13239  46552  #Spanishrevenge vs. #justice #HumanRights and ...       NOT       NaN       NaN

[13240 rows x 5 columns]
```

Figure 4: Sample of training data

Exploring a sample of the data shows that it is quite noisy and will, therefore, need to undergo some data preparation to make it suitable for modeling and evaluation (Figure 4).

Further exploration of the given training data using Tableau results in the displayed visual which shows the proportion of the different classes of tweets for each of the three tasks. Looking at the proportion of non-offensive tweets present in the training data, an educated guess can be made that our trained models might be better suited to identify non-offensive tweets than offensive ones (Figure 5).

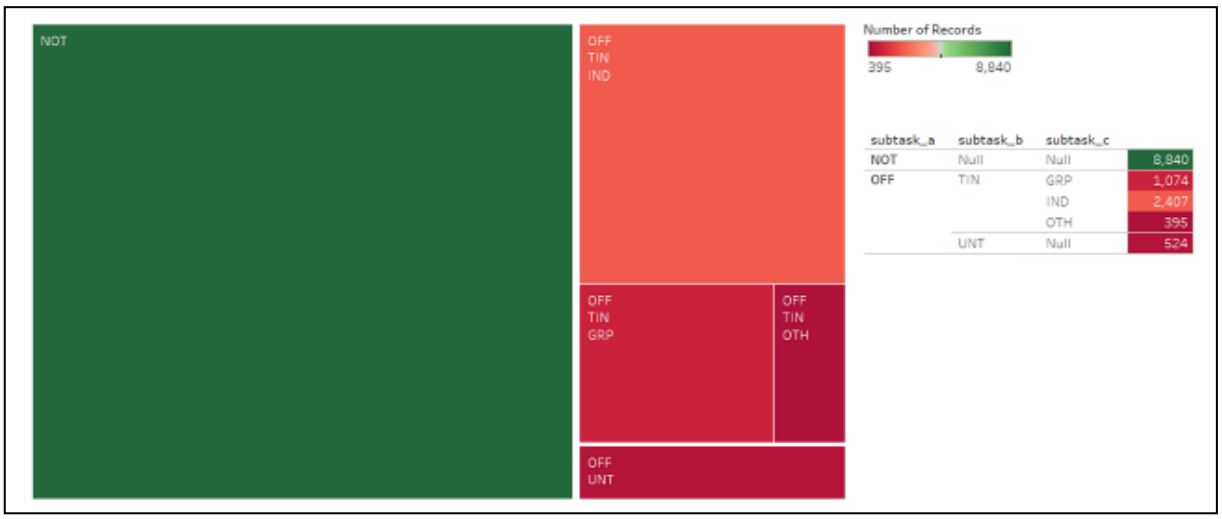

Figure 3: Visualization of training data

2.3 Data Preparation

This phase aims to prepare data, such as cleaning and applying necessary modifications, for use in WEKA. Data Preparation stage contains three main tasks which are clean data, select features and transform data. These tasks are applied on both training and testing data.

Training Data

The data preparation was carried out on the training data before the testing data.

i. Data cleaning

In this task was used Python scripts to convert TSV files to CSV files, also separated values by comma. Then, in data cleaning, it will be removing any white spaces and tabs from the tweet column, to ensure prevents issues when converting the TSV files to CSV files.

When converting issues have been resolved, it will clean data to be more useful. The phrases as “@USER”, "URL", emojis and code strings will remove from all tweets for each subtask which leads to reduce the size of data and optimize the classification process. Figure 6 shows a snipped of Python code used in data cleaning, depending on the mentioned procedures.

ii. Feature Selection

In this stage, the columns "id", “subtask_a”, “subtask_b” and “subtask_c” were removed. However, the important required features were kept in the classification step.

iii. Transformation

Transformation is the last step in data preparation. When the data is cleaned by importing the “CSV”, “panda”, and “numpy” libraries, it would convert the CSV file to arff file to make the data suitable for modeling and evaluation.

Test Data

The feature selection was not applied to testing data, and the steps that carried out are:

i. Data Cleaning

The structure of the data is the same; therefore, the same procedure implemented in cleaning training data was applied here.

ii. Transformations

Exactly like in the training data, Python was used to convert TSV file to CSV file, then merge test set tweets with respective labels, and export cleaned and merged test data to CSV.

Finally, the tweets were converted in both the training and testing dataset which are string to word vectors by applying the StringToWordVector filter in WEKA.

2.4 Modeling

To analyse the tweets, we built models using the classifiers shown below, we then did comparative analysis to decide the best performing classifier. The classifiers used to build the models were as follows:

- Sequential Minimum Optimization
- J48
- Random Forrest

```python
#Function to clean tweets in a data frame's tweet column
def clean_tweets(df):

    punctuations = string.punctuation

    df.loc[:, 'tweet'] = df.tweet.str.replace('@USER', '') #Remove mentions (@USER)
    df.loc[:, 'tweet'] = df.tweet.str.replace('URL', '') #Remove URLs
    df.loc[:, 'tweet'] = df.tweet.str.replace('&amp', 'and') #Replace ampersand (&) with and
    df.loc[:, 'tweet'] = df.tweet.str.replace('&lt','') #Remove &lt
    df.loc[:, 'tweet'] = df.tweet.str.replace('&gt','') #Remove &gt
    df.loc[:, 'tweet'] = df.tweet.str.replace('\d+','') #Remove numbers

    #Remove punctuations
    for punctuation in punctuations:
        df.loc[:, 'tweet'] = df.tweet.str.replace(punctuation, '')

    df.loc[:, 'tweet'] = df.astype(str).apply(
        lambda x: x.str.encode('ascii', 'ignore').str.decode('ascii')
    ) #Remove emojis
    df.loc[:, 'tweet'] = df.tweet.str.strip() #Trim leading and trailing whitespaces
```

Figure 4: Python function for cleaning tweets

- Naïve Bayes
- Naïve Bayes Multinomial
- Simple logistics

Sequential Minimum Optimization.
This classifier implements John Platt's sequential minimal Optimization algorithm for training a support vector classifier. This implementation transforms nominal attributes to binary ones. Coefficients are based on the normalized data. [Eecs.yorku.ca. 2020]

J48
This is a decision tree generating classifier that based on the C4.5 algorithm. It generates pruned and unpruned C4. C4.5 is an extension of quinlan's earlier ID3 algorithm. J48 builds decision trees from labelled training data, using the information entropy. The logic used here is that each data attribute can be used to make a decision by the data splitting further into smaller subsets. [Samal et. al, 2020]

Naive Bayes classifiers
Naïve bayes classifiers are a family of simple probabilistic classifiers based on applying bayes theorem with strong (naïve) independence assumptions between the features. [Chedella, 2020]
Two naïve bayes classifiers were experimented with, the **Naïve Bayes Classifer** and the **Multinomial Naïve bayes classifer.** the naïve bayes classifier refers to the conditional independence of each of the features in the model while the multinomial Naive Bayes classifier is a specific instance of a Naive Bayes classifier which uses a multinomial distribution for each of the features. [Russell and Norvig, 2003]

Random Forests
This is the generalization of random decision forests that are an ensemble learning technique for classification. The classifier constructs a forest of random trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual tree. [Samal et. al, 2020]

Simple Logistic
This is a classifier for building linear logistic regression models

These classifiers were selected because they have a good reputation in weka. Each classifier was run on data that was passed through a filter using Weka.classifiers.meta.FilteredClassifier. We then used the features to optimize the best model for each subtask. These features were tokenizers and confidence factor.

Logistic Regression (New Model)
A logistic regression model was built with the Python sci-kit learn library. The tweets were transformed with the CountVectorizer which coverts a collection of text documents to a matrix of token or word counts.

2.4.1 Subtask A

```
Correctly Classified Instances         700               81.3953 %
Incorrectly Classified Instances       160               18.6047 %
Kappa statistic                          0.4715
Mean absolute error                      0.186
Root mean squared error                  0.4313
Relative absolute error                 43.6808 %
Root relative squared error             95.4916 %
Total Number of Instances              860

=== Detailed Accuracy By Class ===

                 TP Rate   FP Rate   Precision   Recall  F-Measure   MCC      ROC Area  PRC Area  Class
                 0.463     0.050     0.782       0.463   0.581       0.498    0.706     0.512     OFF
                 0.950     0.538     0.820       0.950   0.880       0.498    0.706     0.815     NOT
Weighted Avg.    0.814     0.401     0.810       0.814   0.797       0.498    0.706     0.731

=== Confusion Matrix ===

   a   b   <-- classified as
 111 129 |   a = OFF
  31 589 |   b = NOT
```

Figure 7: Modeling output

Class	classifier	Correctly classified %	TP rate	FP	Precision	Recall	F1 score
OFF	SMO	81.40	0.46	0.05	0.78	0.46	0.58
	Simple Logistics	80.12	0.44	0.06	0.75	0.44	0.55
	Naïve bayes multino..	79.65	0.45	0.07	0.72	0.45	0.55
	Random Forrest	79.53	0.35	0.03	0.80	0.35	0.49
	J48	74.65	0.48	0.15	0.55	0.48	0.51
	Naïve Bayes	57.79	0.44	0.37	0.32	0.44	0.37
NOT	SMO	81.40	0.95	0.54	0.82	0.95	0.88
	Simple Logistics	80.12	0.94	0.56	0.81	0.94	0.87
	Naïve bayes multino..	79.65	0.93	0.55	0.81	0.93	0.87
	Random Forrest	79.53	0.97	0.65	0.79	0.97	0.87
	J48	74.65	0.85	0.53	0.81	0.85	0.83
	Naïve Bayes	57.79	0.63	0.56	0.74	0.63	0.68

Figure 8: Results for Subtask A

Models based on different classifiers were built using the supplied training and test set. The best performing model based on accuracy was then singled out and optimized further, investigating different parameters like tokenizers and confidence factor. Training data was uploaded via the pre-processing panel in weka.

The *meta.FliteredClassifier* was selected so that classification and filters were done at the same time. The filter used was string to word vector along with the 6 different classifiers

Before we ran the model we chose the supplied test set option and the provided test dataset was used.

2.4.2 Subtasks B & C

The same methodology that was used in the preprocessing and classification for subtask A was used in Subtask B and C.

The model performance was saved for every classifier and comparative analysis was done using their results.

Figure 7 shows the results of our best performing classifier for subtask A

2.5 Evaluation

The key metrics used to evaluate the performance of the models were:-

- Accuracy which is the measure of correctly classified tweets

$$Accuracy = \frac{tp+tn}{tp+tn+fp+fn} \quad (1)$$

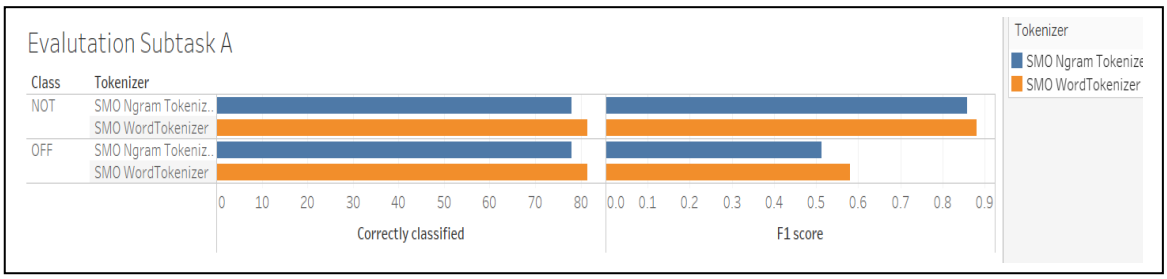

Figure 9: Tokenizer Evaluation

Class	classifier	Correctly classified & F	Incorrectly classified %	F1 score	TP rate	FP Rate	Precision	Recall
TIN	SMO	88.75	11.25	0.94	0.98	0.85	0.90	0.98
	Simple L..	88.75	11.25	0.94	1.00	1.00	0.89	1.00
	Random ..	88.33	11.67	0.94	1.00	1.00	0.89	1.00
	J48	88.33	11.67	0.94	0.99	0.96	0.89	0.99
	Naïve Ba..	87.92	12.08	0.94	0.99	0.96	0.89	0.99
	Naïve Ba..	83.33	16.67	0.90	0.88	0.56	0.93	0.88
UNT	SMO	88.75	11.25	0.23	0.15	0.02	0.50	0.15
	Simple L..	88.75	11.25		0.00	0.00		0.00
	Random ..	88.33	11.67	0.00	0.00	0.01	0.00	0.00
	J48	88.33	11.67	0.07	0.04	0.01	0.33	0.04
	Naïve Ba..	87.92	12.08	0.07	0.04	0.01	0.25	0.00
	Naïve Ba..	83.33	16.67	0.38	0.44	0.12	0.32	0.44

Figure 10: Results for Subtask B

class	Classifier	Correctly classified	Incorrectly classifi..	F1 score	Tp rate	Fp rate	precision	recal
GRP	J48	65.26	34.74	0.64	0.58	0.13	0.71	0.58
	Naïve Bayes	63.85	36.15	0.64	0.67	0.24	0.61	0.67
	Naïve bayes multino..	63.85	36.15	0.62	0.59	0.18	0.66	0.59
	Simple Logistics	63.38	36.62	0.63	0.59	0.16	0.69	0.59
	SMO	59.15	40.85	0.58	0.55	0.21	0.61	0.55
IND	J48	65.26	34.74	0.75	0.92	0.49	0.63	0.92
	Naïve Bayes	63.85	36.15	0.75	0.84	0.35	0.68	0.84
	Naïve bayes multino..	63.85	36.15	0.75	0.90	0.45	0.64	0.90
	Simple Logistics	63.38	36.62	0.74	0.89	0.47	0.63	0.89
	SMO	59.15	40.85	0.70	0.82	0.46	0.61	0.82
OTH	J48	65.26	34.74	0.11	0.06	0.01	0.67	0.06
	Naïve Bayes	63.85	36.15	0.00	0.00	0.03	0.00	0.00
	Naïve bayes multino..	63.85	36.15	0.00	0.00	0.01	0.00	0.00
	Simple Logistics	63.38	36.62	0.00	0.00	0.02	0.00	0.00
	SMO	59.15	40.85	0.05	0.03	0.04	0.13	0.03

Figure 11: Results for Subtask C

- F1 score which is the measure that combines precision and recall. It is the harmonic mean of precision and recall.

$$F = 2 . \frac{precision . recall}{precision + recall} \quad (2)$$

Where:

- **tp** is true positive which is the result where a model correctly predicts that a tweet is offensive.
- **tn** is true negative which is the result where a model correctly predicts that a tweet is not offensive.
- **fp** is false positive which is the result where the model incorrectly predicts an offensive tweet
- **fn** is false negative which is the result where the model incorrectly predicts a non-offensive tweet.

2.5.1 Subtask A

Figure 8 shows the results for subtask A. From the table it is observed that the best performing classifier is SMO, which had the highest F1 score and accuracy.

Following the comparative analysis, we singled out the best classifier for optimization. Different tokenizers were used and the word

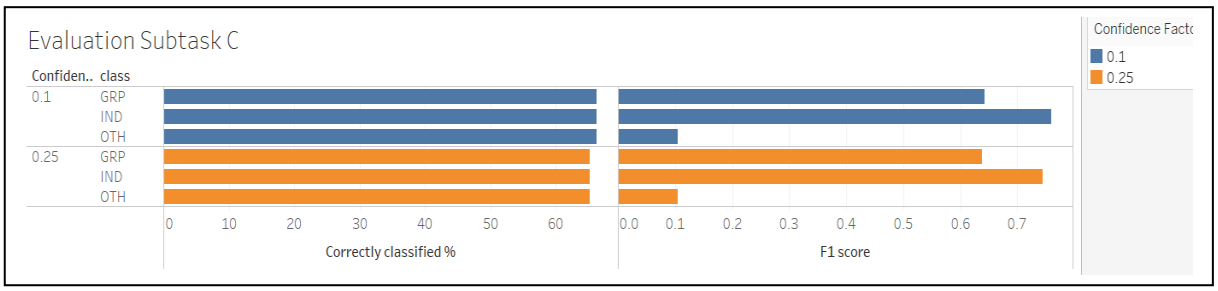

Figure 12: Confidence factor evaluation

tokenizer performed better than the Ngram tokenizer.

The new logistic regression model built was able to achieve an accuracy of 79.6%, thus, making the SMO the best classifier for subtask A.

2.5.2 Subtask B

Figure 10 shows the comparative analysis for subtask B. A lot of classifiers performed well in this task, but SMO again came out on top. We singled it out for optimization, using different tokenizers to optimize our model like in subtask A. The best performing tokenizer was the Word tokenizer.

2.5.3 Subtask C

Figure 11 shows the comparative analysis for subtask C. The best performing classifier was J48. This was singled out for further optimization.

We used the confidence factor to optimize. The model performs better when the confidence factor is 0.1 (Figure 12).

References

Foster Provost and Tom Fawcett. 2013. *Data Science for Business.* O'Reilly Media, 1005 Gravenstein Highway North, Sebastopol, CA 95472.

Alexander V. Mamishev and Sean D. Williams. 2010. *Technical Writing for Teams: The STREAM Tools Handbook*. Wiley-IEEE Press, Hoboken, NJ.

Eecs.yorku.ca. 2020. *Smoreg*. [online] Available at: <https://www.eecs.yorku.ca/tdb/_doc.php/userg/sw/weka/doc/weka/classifiers/functions/SMOreg.html> [Accessed 12 May 2020].

Samal, A., Pani, S. and Pramanik, J., 2016. *Comparative Study Of J48, AD Tree, REP Tree And BF Tree Data Mining Algorithms Through Colon Tumour Dataset*. [online] Ijsrd.com. Available at: <http://www.ijsrd.com/articles/IJSRDV4I31613.pdf> [Accessed 12 May 2020].

Chedella, S., 2020. *Intuition Behind Naive Bayes Algorithm & Laplace(Additive)Smoothing*. [online] Medium. Available at: <https://medium.com/analytics-vidhya/intuition-behind-naive-bayes-algorithm-laplace-additive-smoothing-e2cb43a82901> [Accessed 12 May 2020].

Stuart J. Russell and Peter Norvig. 2003. Artificial Intelligence: A Modern Approach (2 ed.). Pearson Education. *See p. 499 for reference to "idiot Bayes" as well as the general definition of the Naive Bayes model and its independence assumptions*