Final Project Paper

Comment Toxicity Classification Models: A Comparative Analysis

Department of Computer Science

The George Washington University

Machine Learning CSCI 6364

Prof. David Trott

Skandana Gowda

G27230881

1. Introduction:

Online communication has transformed how we communicate and share ideas, but it is not without its limitations. Online conversations are under a cloud of negativity due to the increase in toxic remarks, which are vile, disrespectful, and nasty. Numerous perspectives are silenced and important conversations are hindered as platforms fight to uphold civility.

In response to this pressing need, I built a multi-headed model capable of identifying various types of toxicity including threats, obscenity, insults, and hate based on identity. Using comments from Wikipedia talk page updates as a dataset, I built a model to increase toxicity detection's precision and granularity.

By creating a more sophisticated and reliable toxicity detection methods, I hope to help build more secure and welcoming online spaces where people of different backgrounds may express themselves without worrying about being harassed or mistreated.

2. Objective:

The goal is to evaluate the effectiveness of two distinct toxicity classification methods or models. In particular, the goal is to compare and assess the efficacy of:

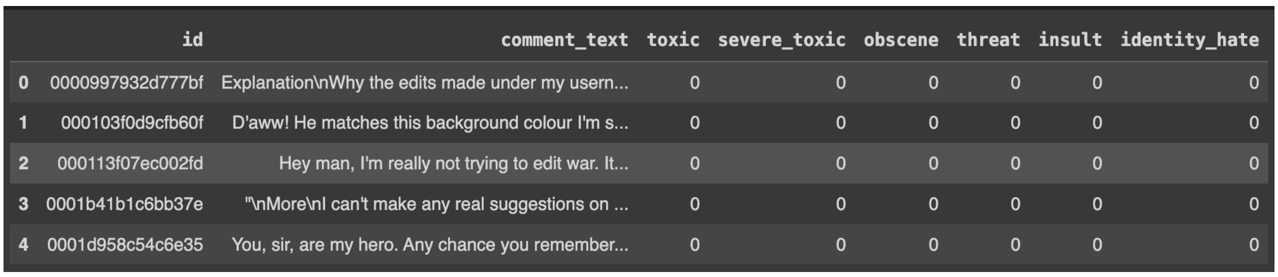
Model 1: TensorFlow/Keras is used to create a deep learning model that uses an embedding layer, bidirectional LSTM layers, and dense layers to classify toxicity.

Model 2: a scikit-learn-implemented Naive Bayes model with TF-IDF vectorization that uses binary relevance for multi-label harmful comment classification.

I compare the effectiveness of the two models' evaluation metrics (precision, recall, and F1-score) to see which method is better at identifying different kinds of toxicity in online comments, such as threats, obscenity, insults, and identity hate.

3. Dataset Used:

The dataset utilized is from Kaggle's Toxic Comment Classification Challenge, which was hosted by Jigsaw. The dataset comprises an enormous amount of Wikipedia comments that have been categorized for various types of toxic behavior by human raters. These behaviors include insults, toxicity, severe toxicity, obscenity, identity hate, and threats.



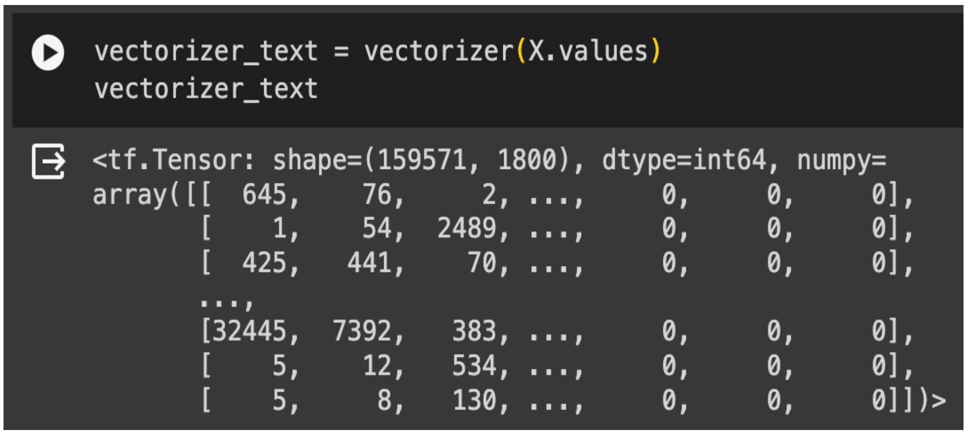
	id	comment_text	toxic	severe_toxic	obscene	threat	insult	identity_hate
0	0000997932d777bf	Explanation\nWhy the edits made under my usern...	0	0	0	0	0	0
1	000103f0d9cfb60f	D'aww! He matches this background colour I'm s...	0	0	0	0	0	0
2	000113f07ec002fd	Hey man, I'm really not trying to edit war. It...	0	0	0	0	0	0
3	0001b41b1c6bb37e	"\nMore\nI can't make any real suggestions on ...	0	0	0	0	0	0
4	0001d958c54c6e35	You, sir, are my hero. Any chance you remember...	0	0	0	0	0	0

4. Data Pre-processing:

For both models, the dataset goes through multiple pre-processing procedures to prepare the text data for training and classification. With the help of these procedures, the text data is cleaned, standardized, and formatted so that machine learning algorithms may use it. The pre-processing steps for both models' datasets are listed below:

i. **Text Cleaning:**
   - Taking out unnecessary characters, symbols, and noise from text data is known as text cleaning. This involves removing any HTML tags (if any), punctuation, and special characters from the comments.
   - Every remark in the dataset undergoes text cleaning using the `clean_text` function, which removes non-alphanumeric characters and converts text to lowercase.

ii. **Tokenization:**
   - The division of the text into discrete words, or tokens, is known as tokenization. By dividing the text data into smaller pieces, this stage facilitates processing and analysis.
   - Tokenizing each cleaned comment means dividing it into a list of individual words, or tokens, using the process_text function.



```
vectorizer_text = vectorizer(X.values)
vectorizer_text
```

```
<tf.Tensor: shape=(159571, 1800), dtype=int64, numpy=
array([[  645,    76,     2, ...,     0,     0,     0],
       [    1,    54,  2489, ...,     0,     0,     0],
       [  425,   441,    70, ...,     0,     0,     0],
       ...,
       [32445,  7392,   383, ...,     0,     0,     0],
       [    5,    12,   534, ...,     0,     0,     0],
       [    5,     8,   130, ...,     0,     0,     0]])>
```

iii. **Elimination of Stop Words:**
   - Stop words are common words that have no important meaning in the context of natural language processing. In order to lower noise and increase text analysis performance, these terms are usually eliminated.
   - To ensure that only pertinent words are kept for analysis, stop words are eliminated from the tokenized text data using the NLTK toolkit.

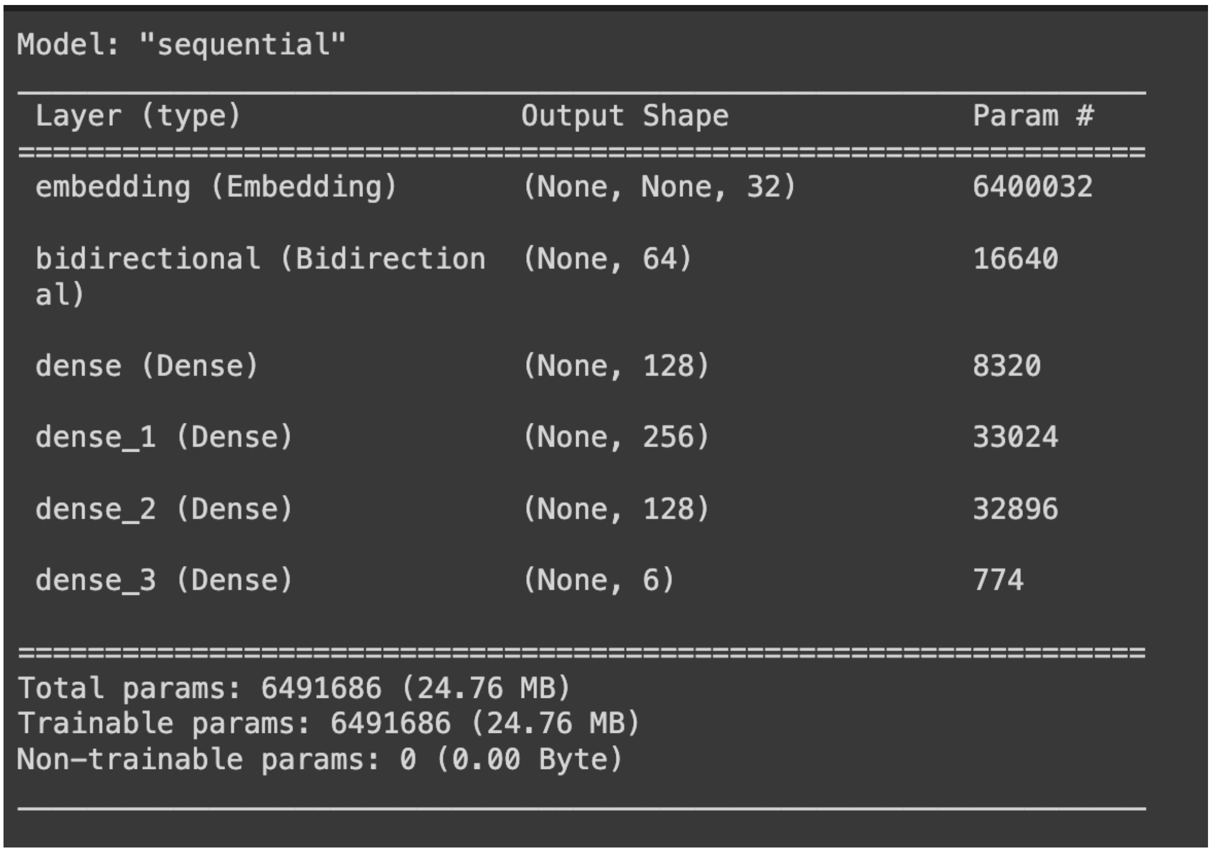
iv. **Lemmatization:**
   - Lemmatization is the process of breaking down words into their lemma, or basic or root form. In order to help standardize the text data, this stage converts various word forms (such plural and past tense) into its canonical form.
   - The tokenized text data is lemmatized using the WordNet Lemmatizer from the NLTK toolkit to guarantee consistency in word representation.

5. Model Description:

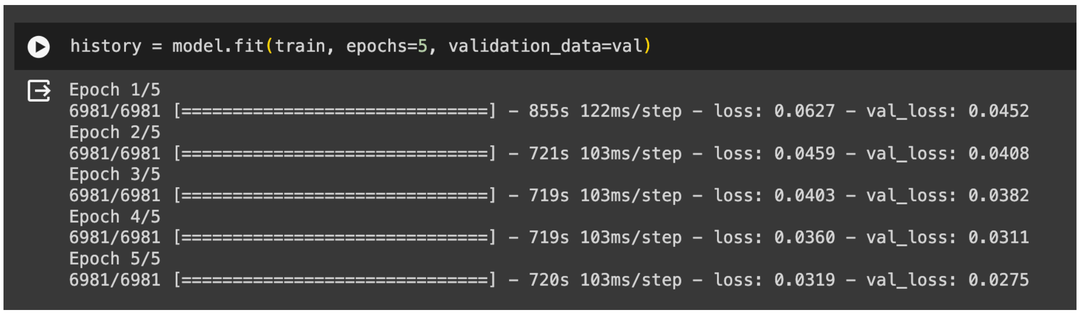
Model 1 (Deep Learning Model):
The deep learning model implemented using TensorFlow/Keras consists of several layers designed to process and classify toxic comments effectively.
The architecture includes:

- *Embedding Layer:* This layer converts each word in the input sequence into a dense vector representation. The embedding layer learns to map words with similar meanings to nearby points in the embedding space.
- *Bidirectional LSTM (Long Short-Term Memory) Layer:* Bidirectional LSTM layers process input sequences in both forward and backward directions, capturing contextual information effectively. LSTM units help the model retain information over long sequences, making them suitable for text classification tasks.
- *Dense Layers:* Fully connected dense layers are added after the LSTM layer to perform feature extraction and classification. These layers utilize activation functions such as ReLU (Rectified Linear Unit) to introduce non-linearity into the model and improve its expressive power.

The Binary Crossentropy loss function, which works well for binary classification problems like toxicity classification, is used to train the model. During training, the Adam optimizer is used to minimize the loss function and modify the model's parameters for better performance. A part of the training data set aside for validation is used to validate the model in order to prevent overfitting.



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, None, 32)          6400032

 bidirectional (Bidirection  (None, 64)                16640
 al)

 dense (Dense)               (None, 128)               8320

 dense_1 (Dense)             (None, 256)               33024

 dense_2 (Dense)             (None, 128)               32896

 dense_3 (Dense)             (None, 6)                 774

=================================================================
Total params: 6491686 (24.76 MB)
Trainable params: 6491686 (24.76 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```


```
history = model.fit(train, epochs=5, validation_data=val)

Epoch 1/5
6981/6981 [==============================] - 855s 122ms/step - loss: 0.0627 - val_loss: 0.0452
Epoch 2/5
6981/6981 [==============================] - 721s 103ms/step - loss: 0.0459 - val_loss: 0.0408
Epoch 3/5
6981/6981 [==============================] - 719s 103ms/step - loss: 0.0403 - val_loss: 0.0382
Epoch 4/5
6981/6981 [==============================] - 719s 103ms/step - loss: 0.0360 - val_loss: 0.0311
Epoch 5/5
6981/6981 [==============================] - 720s 103ms/step - loss: 0.0319 - val_loss: 0.0275
```

Model 2 (Naive Bayes with TF-IDF):

- Model 2 employs a Naive Bayes classifier in conjunction with TF-IDF (Term Frequency-Inverse Document Frequency) vectorization for toxicity classification.
- TF-IDF vectorization transforms the text data into numerical feature vectors, where each feature represents the importance of a word in the document relative to the entire corpus. This process captures the significance of words in distinguishing toxic comments from non-toxic ones.
- The Naive Bayes classifier is trained separately for each toxicity label using the Binary Relevance approach. In this approach, a separate classifier is trained for each label, treating the classification of each label as an independent binary classification task. This allows the model to capture correlations between different types of toxicity and make predictions for each label independently.
- The model is evaluated using appropriate metrics such as precision, recall, and F1-score to assess its performance in predicting the presence of different types of toxicity in comments.

6. Evaluation:

Several common metrics are used to evaluate both models in order to determine how well they perform in terms of toxicity classification. These measures include precision, recall, and F1-score, which provide information about the model's capacity to correctly identify toxic comments while minimizing false positives and negatives.

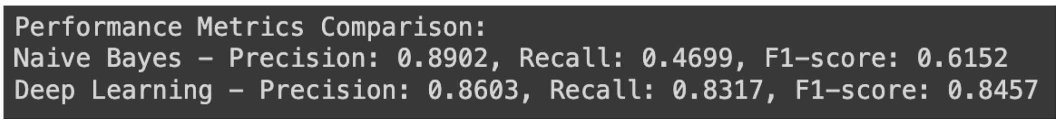
Evaluation Results:

Model 1 (Deep Learning):

- *Precision:* 0.8603
- *Recall:* 0.8317
- *F1-score:* 0.8457

Model 2 (Naive Bayes with TF-IDF):

- *Precision:* 0.8902
- *Recall:* 0.4699
- *F1-score:* 0.6152


```
Performance Metrics Comparison:
Naive Bayes - Precision: 0.8902, Recall: 0.4699, F1-score: 0.6152
Deep Learning - Precision: 0.8603, Recall: 0.8317, F1-score: 0.8457
```

7. Insights :

Model 1 (Deep Learning):
Strengths:

- Text classification tasks are well-suited for deep learning models, such as LSTM-based architectures, since they can effectively capture intricate patterns and dependencies in sequential data.
- It might be particularly good at picking up on subtle word relationships and harmful language in comments.

Weaknesses:

- Training deep learning models usually requires a lot of data and computer power, which can be costly and time-consuming.
- When trained on small amounts of data, it may be prone to overfitting, which could result in less than ideal performance on new data.

Model 2 (Naive Bayes with TF-IDF):
Strengths:

- Naive Bayes classifiers are simple and efficient, making them ideal for huge datasets with complex feature spaces.
- Given its lower computing overhead as compared to deep learning models, it might perform well in situations when computational resources are few.

Weaknesses:

- Naive Bayes classifiers presume feature independence, which may not be true for text data with associated features, like as words in a sentence.
- It's possible that this model will perform worse than deep learning models since it will have trouble grasping intricate word associations and spotting small instances of comments' toxicity.

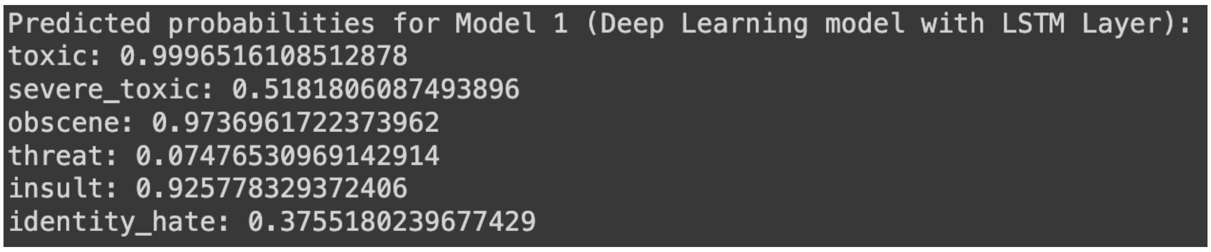
In summary, the choice between Models 1 and 2 is determined by considerations such as dataset size, processing resources, and desired level of performance. Deep learning models have the potential to be more accurate, but they also come with a larger training resource requirement and a higher risk of overfitting. Conversely, naive Bayes models are easier to understand and use less computing power, but they may have trouble identifying complex correlations in the data.

8. Prediction Example:

Input Sentence: "You freaking suck! I am going to hit you."

Predicted probabilities for Model 1 (Deep Learning model with LSTM Layer):

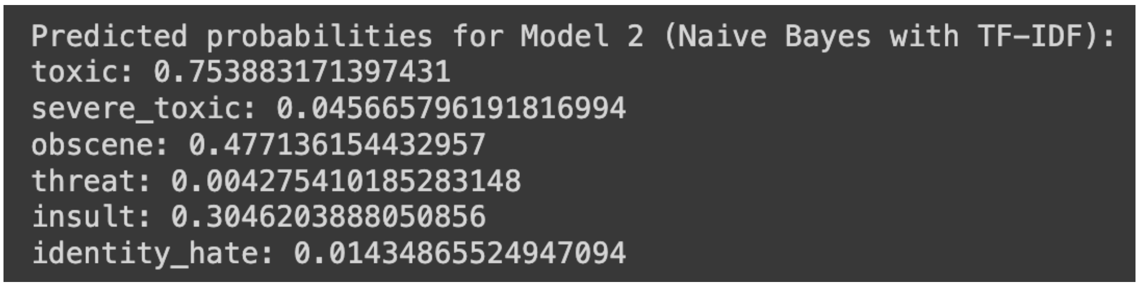
- toxic: 0.9997
- severe_toxic: 0.5182
- obscene: 0.9737
- threat: 0.0748
- insult: 0.9258
- identity_hate: 0.3755



```
Predicted probabilities for Model 1 (Deep Learning model with LSTM Layer):
toxic: 0.9996516108512878
severe_toxic: 0.5181806087493896
obscene: 0.9736961722373962
threat: 0.07476530969142914
insult: 0.925778329372406
identity_hate: 0.3755180239677429
```

Predicted probabilities for Model 2 (Naive Bayes with TF-IDF):

- toxic: 0.7539
- severe_toxic: 0.0457
- obscene: 0.4771
- threat: 0.0043
- insult: 0.3046
- identity_hate: 0.0143



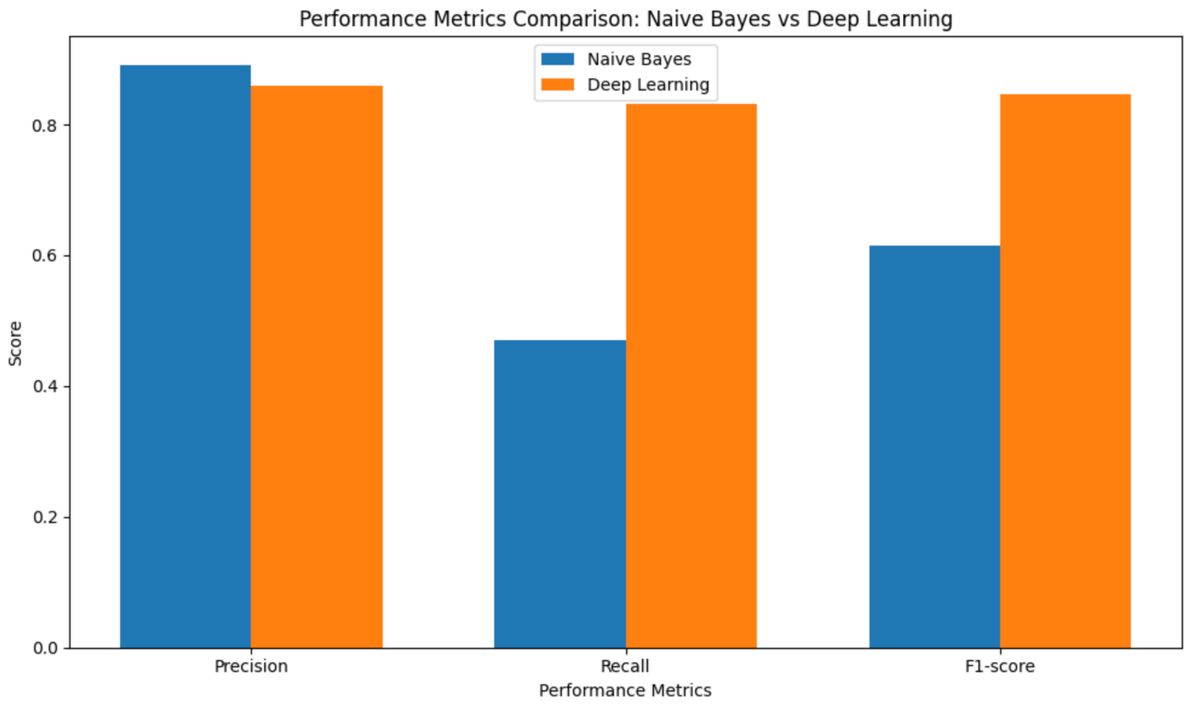
```
Predicted probabilities for Model 2 (Naive Bayes with TF-IDF):
toxic: 0.753883171397431
severe_toxic: 0.045665796191816994
obscene: 0.477136154432957
threat: 0.004275410185283148
insult: 0.3046203888050856
identity_hate: 0.01434865524947094
```

In this example, the input sentence "You freaking suck! I am going to hit you." was passed through both models for toxicity classification. The Deep Learning model predicted high probabilities for toxic, obscene, and insult labels, indicating a high likelihood of toxicity in the comment. On the other hand, the Naive Bayes model predicted lower probabilities across all labels, suggesting a less confident classification compared to the Deep Learning model.

9. Conclusion:

In conclusion, this paper examined how well two different toxicity classification models performed: a Naive Bayes model that used TF-IDF vectorization and a Deep Learning model that used LSTM layers. The Deep Learning model demonstrated a greater recall and F1-score across extensive evaluation criteria, highlighting its potential to capture real positive cases of toxicity. On the other hand, the Naive Bayes model showed better accuracy, demonstrating how effective it is in reducing false positive classifications. The selection between the two models is dependent on several needs, like the size of the dataset, the required ratio of precision to recall, and processing capabilities.

In practice, these models can be used in online content moderation to efficiently detect and remove harmful remarks. To improve the models' ability to identify subtle types of toxicity, more research and development are advised. This includes deep learning architecture optimization, ensemble technique investigation, and domain-specific information incorporation. This analysis provides useful insights for practitioners and researchers looking to design more effective content management systems for online debates.



10. References:

1) Google Jigsaw. (n.d.). Perspective API. Retrieved from https://www.perspectiveapi.com/
2) Kaggle. (2018). Toxic Comment Classification Challenge. Retrieved from https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge
3) Brownlee, J. (2020). How to Develop a Deep Learning Bag-of-Words Model for Sentiment Analysis (Text Classification). Retrieved from https://machinelearningmastery.com/deep-learning-bag-of-words-model-sentiment-analysis/
4) Scikit-learn. (n.d.). Naive Bayes. Retrieved from https://scikit-learn.org/stable/modules/naive_bayes.html