SPAM DETECTION

A Big Data Problem

Why a big data problem?

- Large Volume : Volume of mails is very large
- High Velocity: millions of mails every second.
- Variety of Data: Spam can take many different forms, including email, social media posts, and comments.
- Complexity of Algorithms: Modern spam detection algorithms are complex and require a lot of computational power to run. This can be a big data problem if the algorithms need to be run on very large data sets.
- False positives: False positives are a problem in spam detection, as they can result in legitimate messages being classified as spam. To reduce the number of false positives, spam detection algorithms need to be very accurate, which requires a lot of data and computational power for training and optimization.

Examples

Email spam detection: Email service providers use spam detection techniques to filter out unwanted emails from a user's inbox. These techniques may include rule-based filters or machine learning algorithms.

SMS spam detection: SMS service providers also use spam detection techniques to filter out unwanted text messages from a user's inbox. These techniques may include analyzing message content and sender information.

Comment spam detection: Websites that allow users to leave comments often use spam detection techniques to filter out unwanted comments that may contain spam or malicious links.

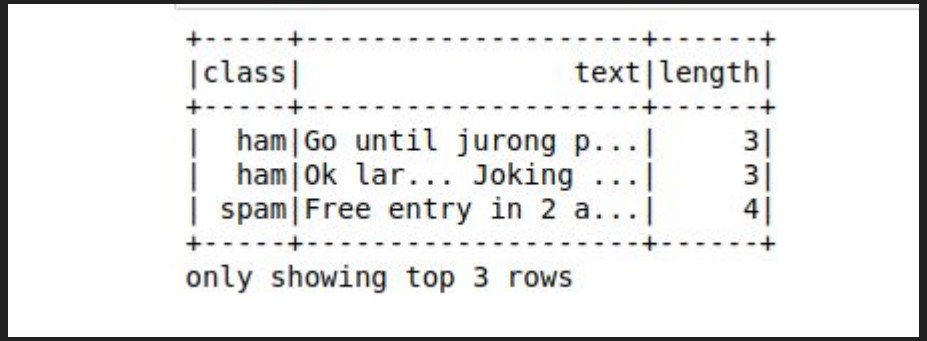
Social media spam detection: Social media platforms use spam detection techniques to identify and remove spam accounts and content.

Online advertising spam detection: Advertisers use spam detection techniques to identify and filter out fraudulent clicks or impressions on their ads.

-

Workflow

- Read csv using spark.read.csv()



```
+-----+--------------------+------+
|class|                text|length|
+-----+--------------------+------+
|  ham|Go until jurong p...|     3|
|  ham|Ok lar... Joking ...|     3|
| spam|Free entry in 2 a...|     4|
+-----+--------------------+------+
only showing top 3 rows
```

- Dropping null values
- Tokenizer: split paragraphs and sentences into smaller units that can be more easily assigned meaning.

Workflow

- Stop words removal: removing the words that occur commonly across all the documents in the corpus
- Count Vectorizer: breaking down a sentence or any text into words by performing preprocessing tasks like converting all words to lowercase, thus removing special characters
- TF-IDF: Term-Frequency; Inverse Document Frequency. uses the frequency of words to determine how relevant those words are to a given document
- Ham_spam_to_numeric: converting string literals to numeric: ham ⇒0, spam⇒1
- Using Naive Bayes algorithm to classify text to spam/ham

Pipeline

1. `ham_spam_to_numeric,`
2. `tokenizer,`
3. `stop_remove,`
4. `count_vec,`
5. `idf,`
6. `Clean_up`

This pipeline ends with Naive Bayes model.