Report For ML Algorithm Selection

1. Preprocessing

- Read all the CSV files and concatenate them
- Select only necessary features for the classification problem (content and class)
- Drop duplicates
- Balance the dataset
- Clean the data

  - Remove punctuation

  - Remove HTML tags

  - Remove URLs

  - Remove emojis

  - Normalize words (repeated letters removal)

I have listed above all the step I have applied to the dataset. First I have merged all the data because training a different model for every cvs file is not a good choice for this dataset. The dataset is not large and a spam detector for YouTube comments shouldn't be affected by singer or song. These are the main reasons why I have decided the merge the data.

There wasn't a big unbalance issue but I have sampled ham comments using spam comments counts. (Class 0: 919, Class 1: 841)

Because the data is taken from a social media platform, there were lots of things to clean. I have used regex to clean the data. To normalize words I have used a naive approach. I have found the repeated letters in a word and delete all the repeated letters after the first one. This approach was affected words that have naturally repeated letters(e.g. happy, feel).

Ideas for further improvements for this step:

Preprocessing step can be improved by correcting typos, normalizing words properly, replacing internet abbreviations/slangs with their normal forms(e.g. plz -> please).

2. Model Selection

Firstly I have removed the stop words. I have applied lemmatization to the words but I have taken it back because it caused a bad model performance.

All models I have tested:

Linear Models	Probabilistic Models	Tree-based Models	Other
- Logistic Regression	- Multinomial NB	- Random Forest	- KNN
- SVM	- Complement NB	- Light GBM	

All models performed very well on the data except the KNN model. That means the dataset is an easy dataset to model. Selecting a model depends on the business goal. For example, if we want to classify all the spam comments so it doesn't bother other users we should look for a high TP(True Positive - model predicts the data as spam and the actual label is spam) value or high recall value for the class 1. The SVM model looks promising but if we care about training time, it would not be the first model to consider. An SVM maps training examples to points in space so as to maximise the width of the gap between the two categories and it is computainally expensive.I have tried classical ML models for the dataset but for a real life problem, I wouldn't choose any of these models.

Because of the dataset size vocabulary size is also low. All of the models would perform badly for out of vocabulary words. For this reason, using a word embedding results in a more robust system. Also, you can capture the semantic meaning of words in this way. There are libraries to convert words to vectors. Generally, they use a huge dataset to get these vectors hence their vocabulary size is also huge. I prefer using the spaCy library for English NLP tasks because it is fast and has lots of features. spaCy allocates some vectors for unknown words and assign randomly those words to one of these vectors. It allows getting information from unknown words. I have written a medium story about spaCy word embedding, actually, I have translated a post written by the founder of spaCy. You can found it here:

https://medium.com/@merve.din16/bloom-embeddings-54814483ce29

Collecting more data shouldn't be expensive because they just used an API to get the data. Collecting more data, using word embedding then training an LSTM or GRU model would be one of the best options for this task.