Blog Gender Classification

Detailed Project Report using CRISP-DM Methodology

Module Title: Natural Language Processing
Date Submitted: 4 July 2021
Student Name: Kevin Kumar Selva Kumar
Student Number: 10548674

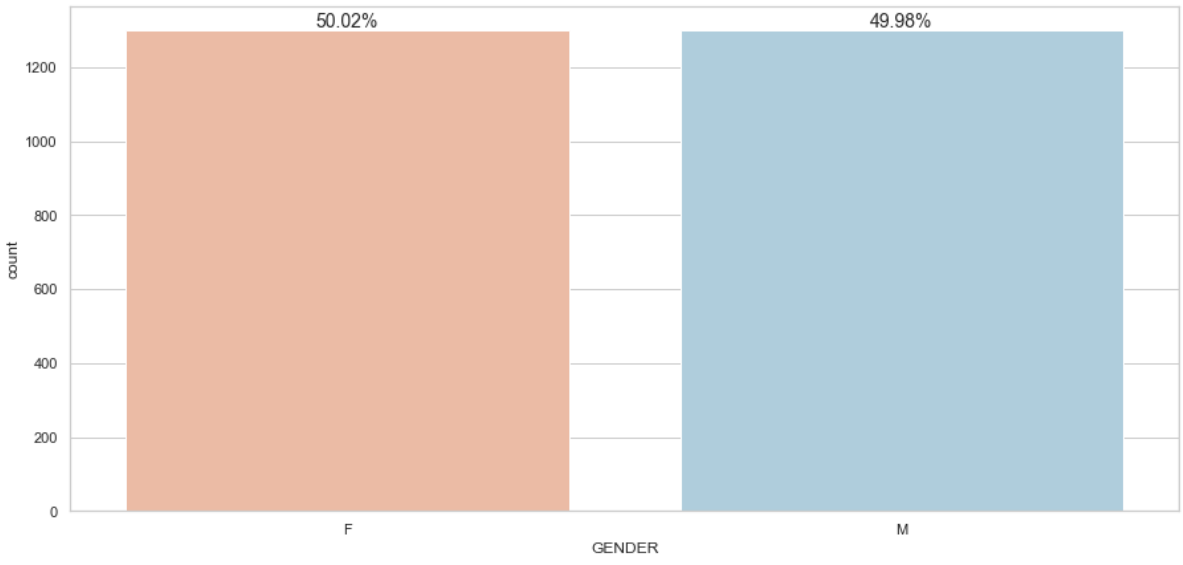
Table of Contents

1. Business Understanding

Weblogs, commonly known as blogs, refer to online personal diaries which generally contain informal writings. With the rapid growth of blogs, their value as an important source of information is increasing. A large amount of research work has been devoted to blogs in the natural language processing (NLP) and other communities. There are also many commercial companies that exploit information in blogs to provide value-added services, e.g., blog search, blog topic tracking, and sentiment analysis of people's opinions on products and services. Gender classification of blog authors is one such study, which also has many commercial applications. For example, it can help the user find what topics or products are most talked about by males and females, and what products and services are liked or disliked by men and women. Knowing this information is crucial for market intelligence because the information can be exploited in targeted advertising and also product development.

2. Data Understanding

The dataset consists of 2600 records among which there is one missing data that has been removed and the total records are 2599. Each record represents textual data which is basically a blog written by either male or female. The gender column acts as a target variable and the textual data are the features which would be converted into word embeddings. We are supposed to create a machine learning model which can identify whether a blog has been written by male or female. The dataset has a good balance between male and female variables with 1300 female blog posts and 1299 male blog posts.



The dataset includes several interesting features, namely emoticons, irregular punctuation / grammatical errors.

Sample blog post from this dataset:

"I had severe carpal tunnel syndrome and was headed straight under the knife, but truly didnt want surgewry... A friend told me about a 'platelet injection' made from your own blood.. i as sceptical & the pain doc in denton, tx told me it was purely experimental but it worked!!.. within a few weeks my pain deminished to the point i didnt notice it or didnt have it anymore. I cant say this is the answer for everyone.. but it surely worked for me :) Best of luck to you friend!"

3. Data Preparation

Before we start with any NLP project, we need to pre-process the data to get it all in a consistent format. We need to clean, tokenize and convert our data into a matrix. Some of the basic text pre-processing techniques includes:

- Make text all lowercase or uppercase and it is the most common and simplest text pre-processing technique. Applicable to most text mining and NLP problems. The main goal is to convert the text into the lower casing so that 'text', 'Text' and 'TEXT' are treated the same way. This is more helpful for text featurization techniques like frequency, tfidf as it helps to combine the same words together thereby reducing the duplication and get correct counts / tfidf values.
- Removing Noise i.e everything that isn't in a standard number or letter i.e Punctuation, Numerical values, common non-sensical text (/n)
- Tokenization: Tokenization is just the term used to describe the process of converting the normal text strings into a list of tokens i.e words that we actually want. Sentence tokenizer can be used to find the list of sentences and Word tokenizer can be used to find the list of words in strings.
- Stopword Removal: Sometimes, some extremely common words which would appear to be of little value in helping select documents matching a user need are excluded from the vocabulary entirely. These words are called stop words. I will be using my own custom built stopwords.

More data cleaning steps after tokenization:

- Stemming: Stemming is the process of reducing inflected (or sometimes derived) words to their stem, base or root form—generally a written word form. Example if we were to stem the following words: "Stems", "Stemming", "Stemmed", "and Stemtization", the result would be a single word "stem".
- Lemmatization: A slight variant of stemming is lemmatization. The major difference between these is, that, stemming can often create non-existent words, whereas lemmas are actual words. Examples of Lemmatization are that "run" is a base form for words like "running" or "ran" or that the word "better" and "good" are in the same lemma so they are considered the same.
- Parts of speech tagging

However, it is not necessary that we need to use all these steps. The usage depends on our problem statement at hand.

It has been found that for the Blog Gender classification task stemming was affecting the model performance, so I would not use stemming instead use only lemmatization. Also there were

some repetitive words and also some irregular words like hiiiiii, okkkkkk, meeeee which doesn't add any value to the analysis, so I have removed them.

Sample Text before cleaning:

'hiiiiiiiiiiiiiiiiiiiiiiiiiiiiii this is poo ok i m gfoinng to tell some thing about jaya gayathiri ok let me say sum lies abt her is a very good gal gal very bad in dancing then etc etc okkkkkkkkkkkkkkk okkkk i think u feel like slaping me nw to say abt jaya she is a very good dancer who always teach me the new dances which i dont knw and help me by clearing my doutsist nt ddddddddddddd she is a good friend for all of us but she like to talk a lot with my sister than meeeeeeeeeee she is one of the rowdy in my dancing class i just love her eyes a lot i like her attitude tat she is always frist and i admire her way of dancing i like her sooooooooooooo much when she fight with person i dont likei knw i m selfish she loves to take photos wen ever v go out last but nt the least she is a multi talented gallllllllllll'

Sample Text after cleaning:

'hi this is poo ok i m gfoinng to tell some thing about jaya gayathiri let me say sum lies abt he r a very good gal bad in dancing then etc ok ok think u feel like slaping nw she dancer who al ways teach the new dances which dont knw and help by clearing my doutsist nt d friend for al l of us but talk lot with sister than me one rowdy class just love eyes attitude tat frist admire w ay so much when fight person likei selfish loves take photos wen ever v go out last least multi talented gal'

Below table could be handy when dealing with text pre-processing, based on the problem statement , we can select the methods in order to clean the dataset.

Level of text preprocessing needed

	Domain Specific / Noisy Texts	General / Well Written Texts
Lots of data	- Moderate pre-processing - Text enrichment could be helpful	- Light pre-processing - Text enrichment could be helpful, but not critical
Sparse data	- Heavy pre-processing - Text enrichment is important	- Moderate pre-processing - Text enrichment could be helpful

The ultimate goal of the data preparation is to produce clean text so that machines can analyse them error free. After pre-processing the data, we end up with the clean dataset as mentioned below in CLEAN_TEXT column,

	BLOG	GENDER	TOKENS	CLEAN_TEXT
0	beyond getting there what travel days show ust...	F	[travel, day, ustoday, guest, post, gillian, o...	travel day ustoday guest post gillian onegiant...
1	remember much about the island large lighthous...	F	[remember, island, lighthouse, helped, fall, a...	remember island lighthouse helped fall asleep ...
2	have had asthma and allergies entire life whil...	M	[asthma, allergy, entire, life, bet, asthmatic...	asthma allergy entire life bet asthmatic disag...
3	the last few days have been emotional rollerco...	M	[day, emotional, rollercoaster, team, producer...	day emotional rollercoaster team producer rosh...
4	you lined all the teachers and staff school se...	F	[lined, teacher, staff, school, search, write,...	lined teacher staff school search write blog c...

Feature Engineering & EDA

In feature engineering, I have created three varieties of dataset namely:

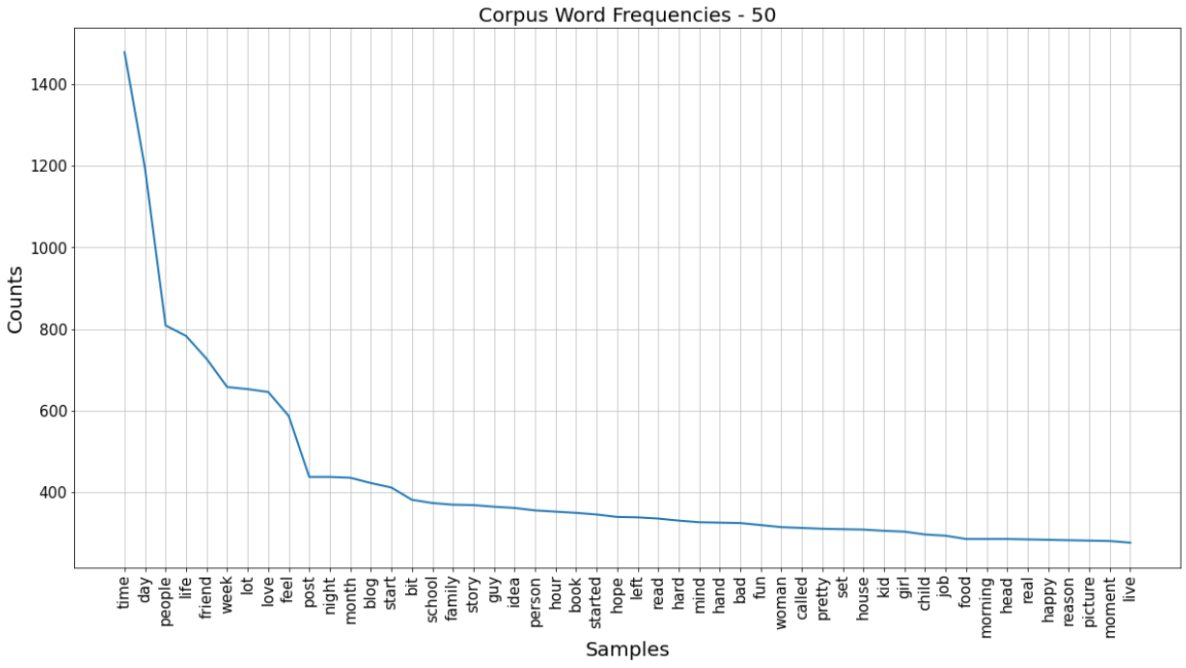
- Data with only text.
- Data with text and its features like word count, sentence count, polarities, etc.
- Data with POS tagging features

Then the model will be trained with all these datasets to find out which fits in perfectly and finally pick out the best performing model.

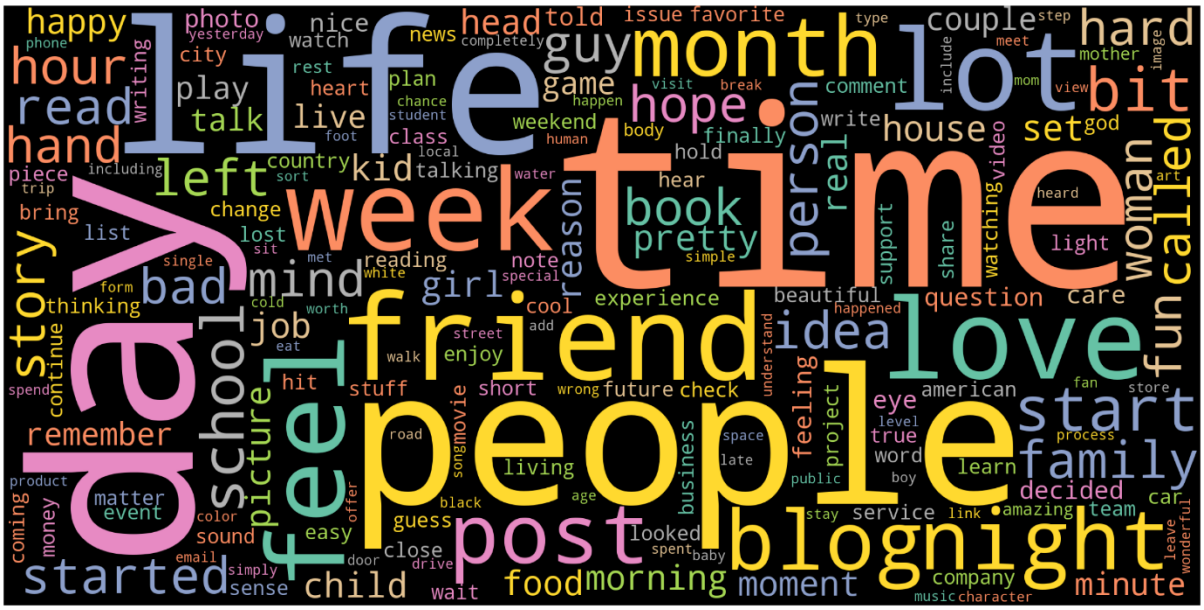
Text Analysis

Frequency Distribution

After getting the clean text, we find out the most frequent words that are occurring in the dataset. First we tokenize the text using RegexpTokenizer and then find the frequency distribution. As we can see, the most frequently occurring words are time, people, life, love, friend, week, blog, feel, book, post, school, day, night, game, story, month, family, read. We could also visualize them using word cloud as shown in the below figures.



A nice way to visualize the same information is with a word cloud where the frequency of each tag is shown with font size and color. A word cloud is a simple yet powerful visual representation object for text processing, which shows the most frequent word with bigger and bolder letters, and with different colors. The smaller the the size of the word the lesser it's important.



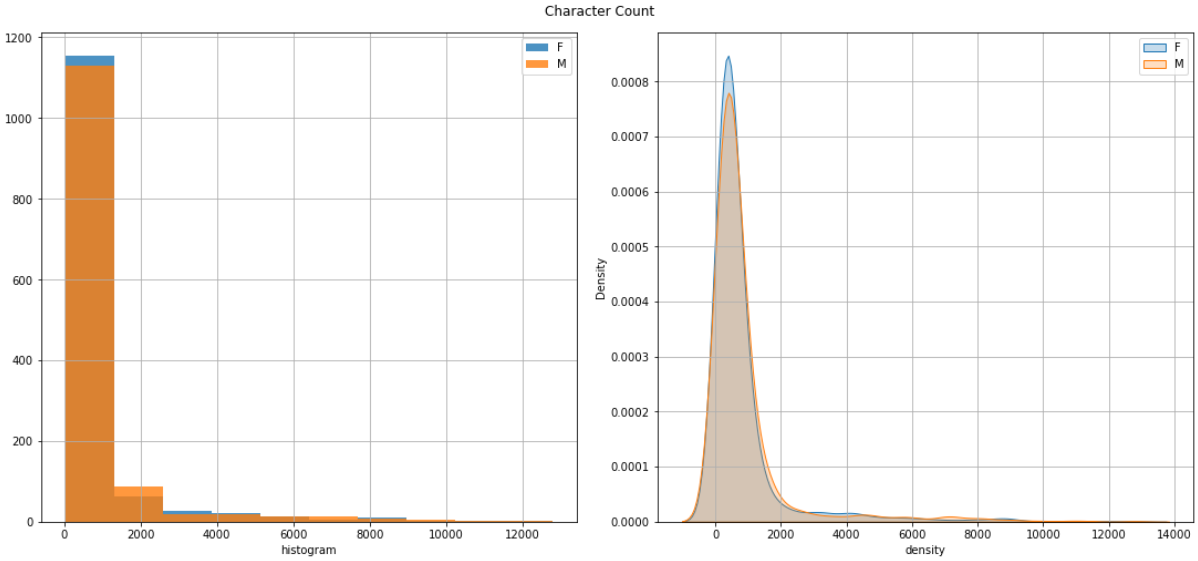
We could also create fancy word cloud. By fancier word cloud, I mean those word clouds in custom shapes. To create a fancy word cloud, we need to first find an image to use as a mask. It appears that the biggest challenge is to find the right image file. We may search for images with keywords: ‘masking images for word cloud’ on Google Images. I used the ‘star.png’ to generate the word cloud, the only difference is that we have imported the image to a numpy array then added mask=mask in the WordCloud.

Length Analysis

It is important to have a look at the length of the text because it's an easy calculation that can give a lot of insights.

There are several length measures for text data. Some of them are:

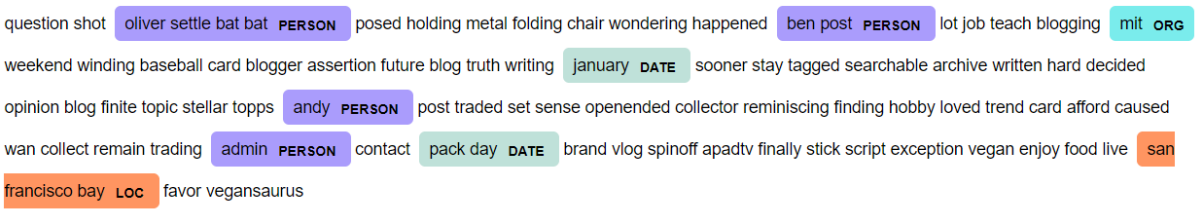
- **word count**: counts the number of tokens in the text (separated by a space)
- **character count**: sum the number of characters of each token
- **sentence count**: count the number of sentences (separated by a period)
- **average word length**: sum of words length divided by the number of words (character count/word count)
- **average sentence length**: sum of sentences length divided by the number of sentences (word count/sentence count)



Both male and female have a similar length distribution, the density plot is very useful because the samples have different size.

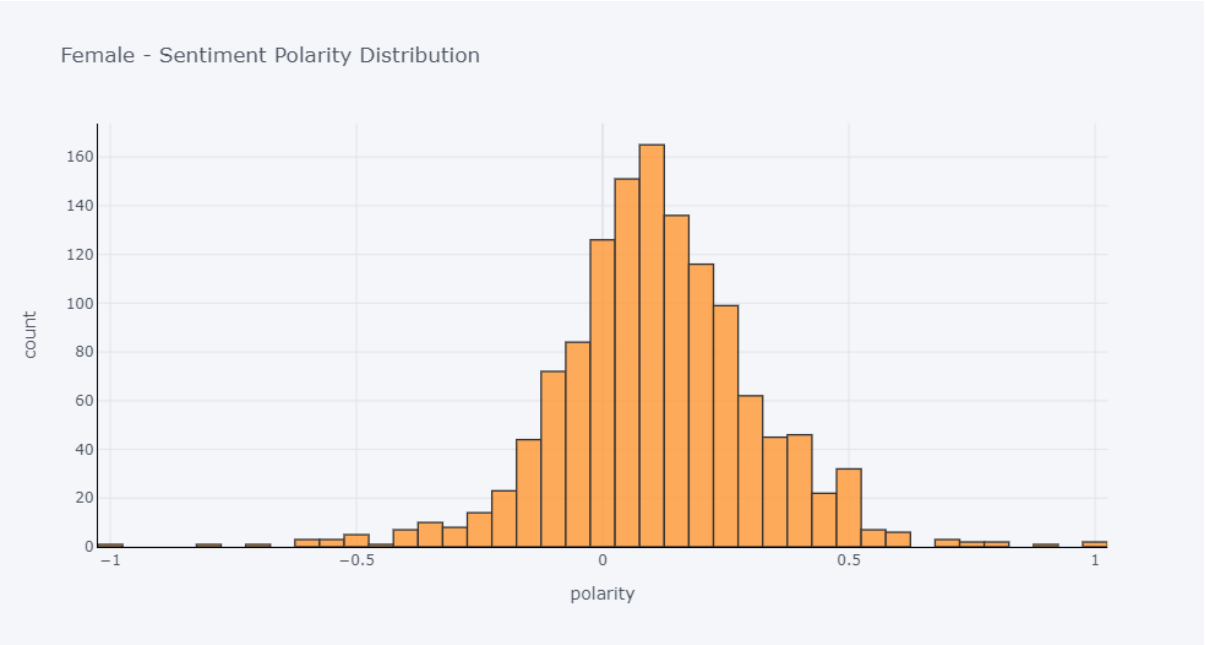
Named-Entity Recognition

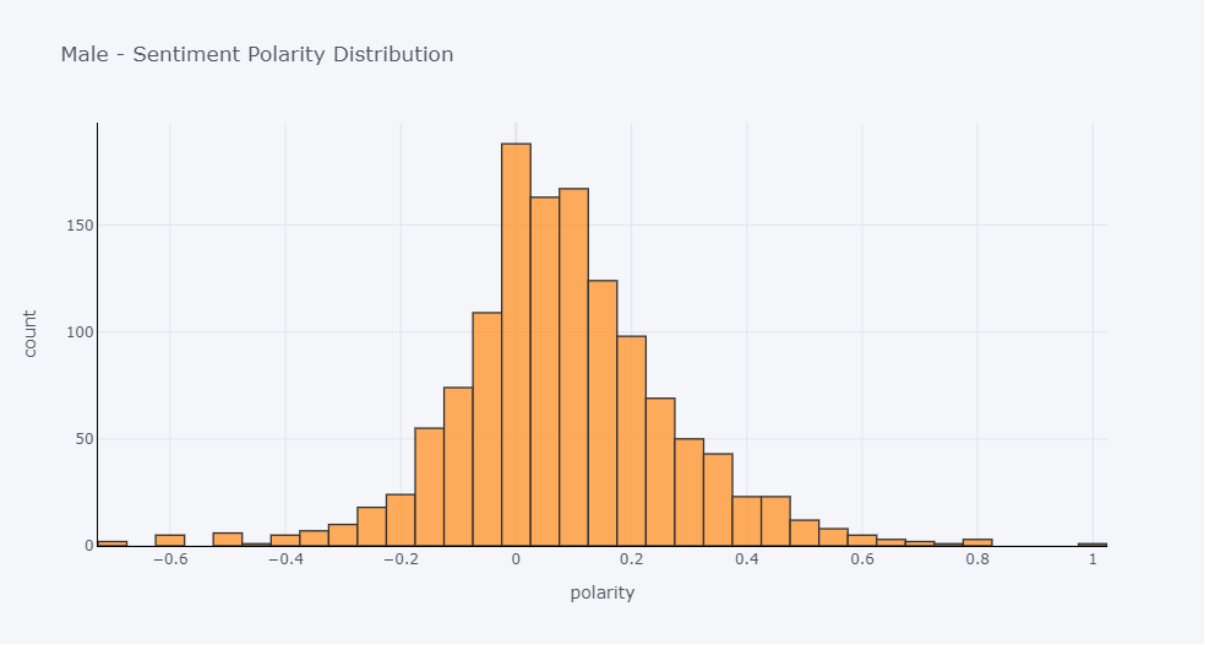
NER (Named-entity recognition) is the process to tag named entities mentioned in unstructured text with pre-defined categories such as person names, organizations, locations, time expressions, quantities, etc. Training a NER model is really time-consuming because it requires a pretty rich dataset. But our dataset is very small, so we are not going to consider it.



Polarity distribution among male and female, who is more prone to write positive/negative threads.

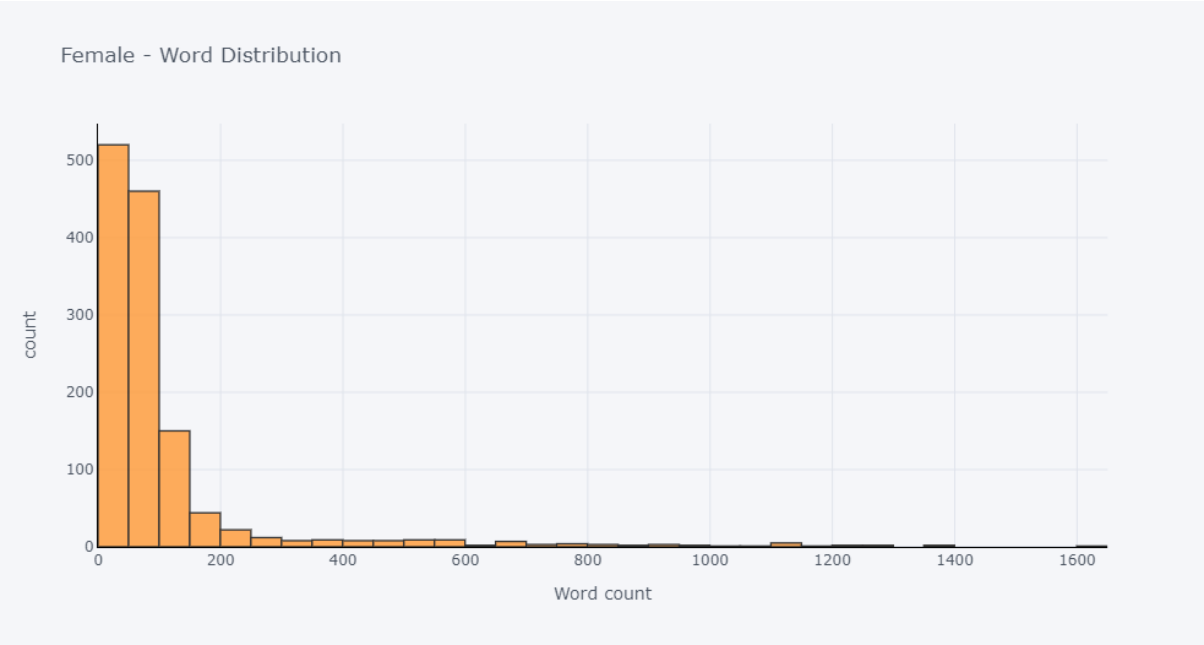
Understanding the polarity scores between male and female gives us a basic idea about whether the blog written is inclined towards positive or negative side. From below figures, it is evident that the blogs written by female are having more positive threads as compared to the bogs written by male.

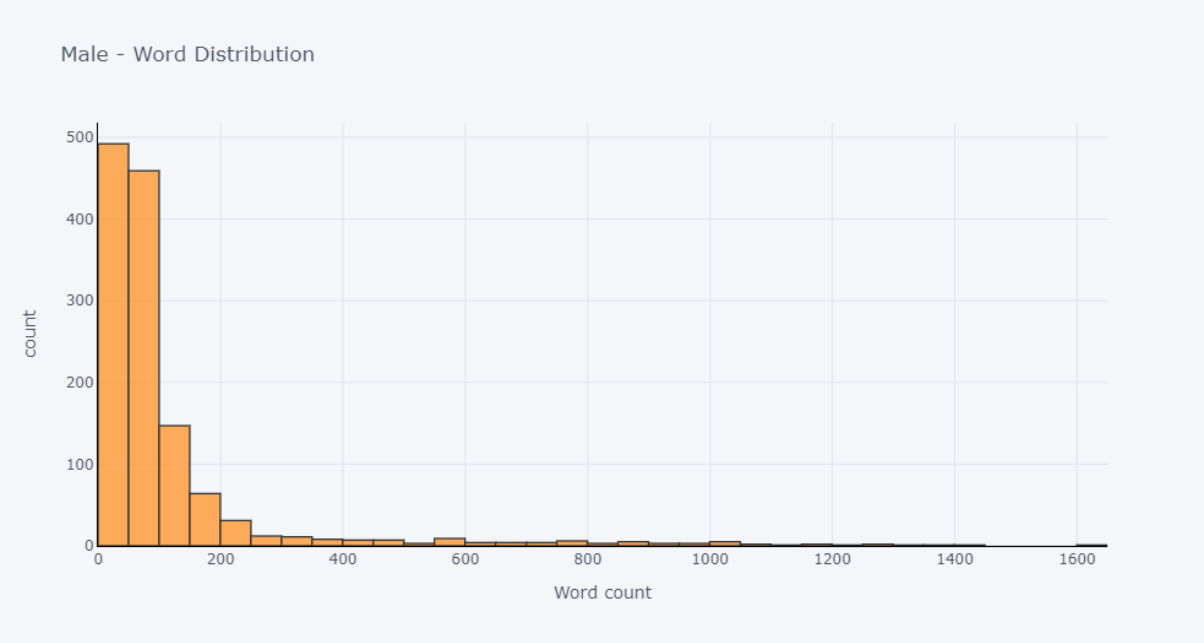




Word distribution between male and female

The word distribution is almost similar for both male and female and we can infer that from the below figures. It is a "distribution" because it tells us how the total number of word tokens in the text are distributed across the vocabulary items. Since we often need frequency distributions in language processing, NLTK provides built-in support for them.



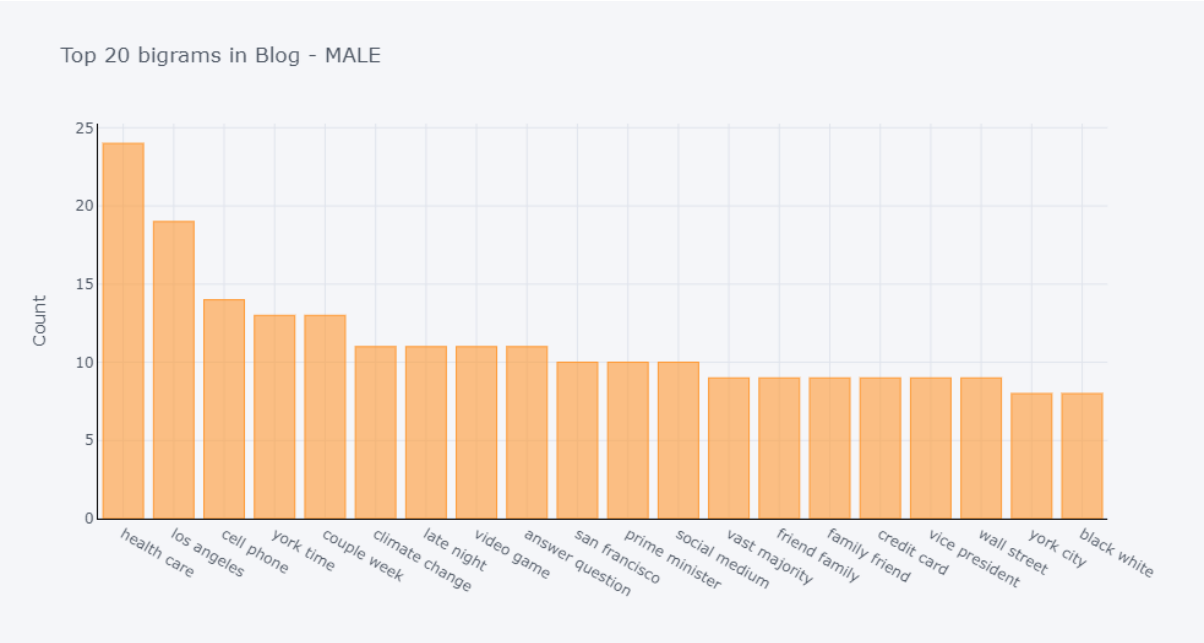


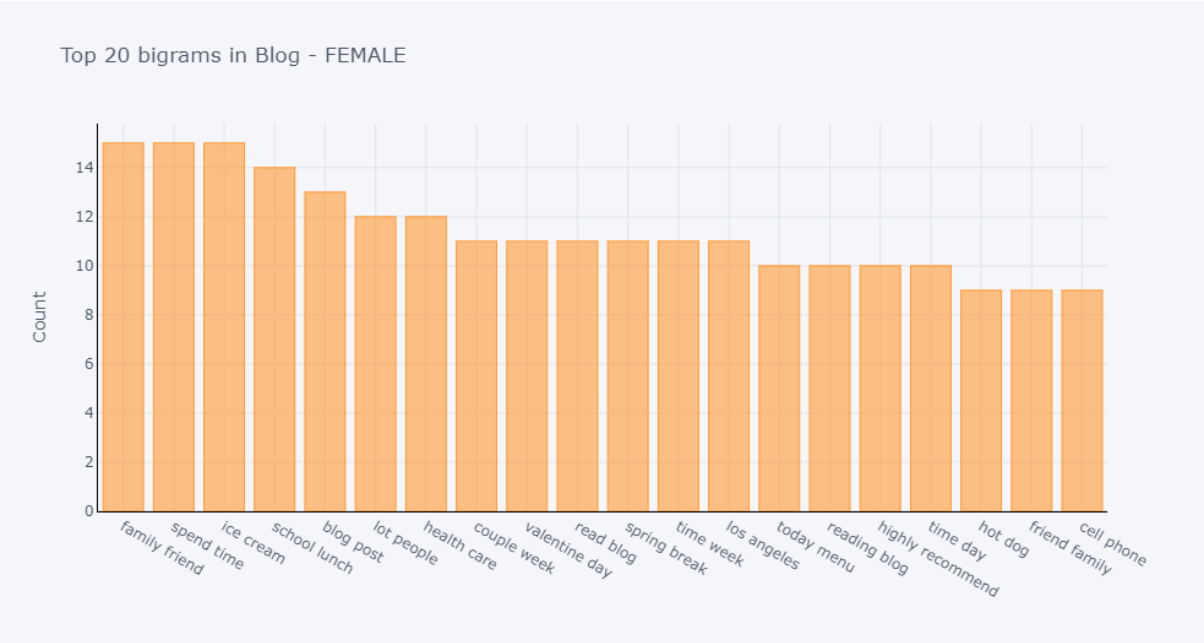
N-grams used by male & female

An n-gram is a sequence of *n* words where *n* is a discrete number that can range from 1 to infinity. For example, the word “cheese” is a 1-gram (unigram). The combination of the words “cheese flavored” is a 2-gram (bigram). Similarly, “cheese flavored snack” is a 3-gram (trigram). So on and so forth.

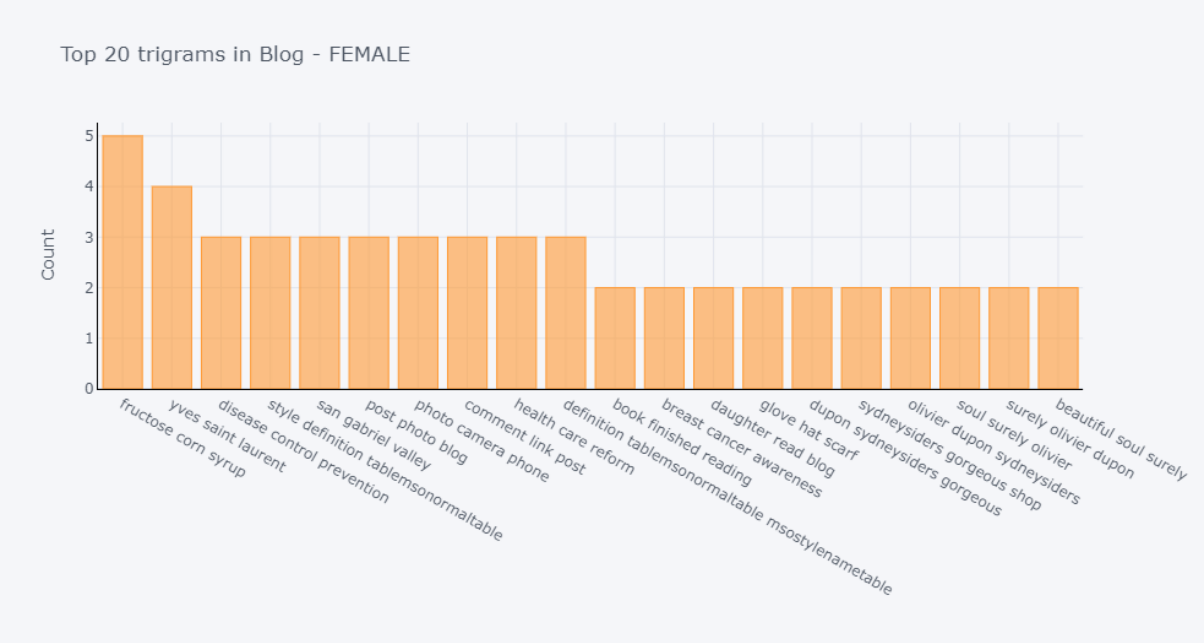
In n-gram ranking, we simply rank the n-grams according to how many times they appear in a body of text.

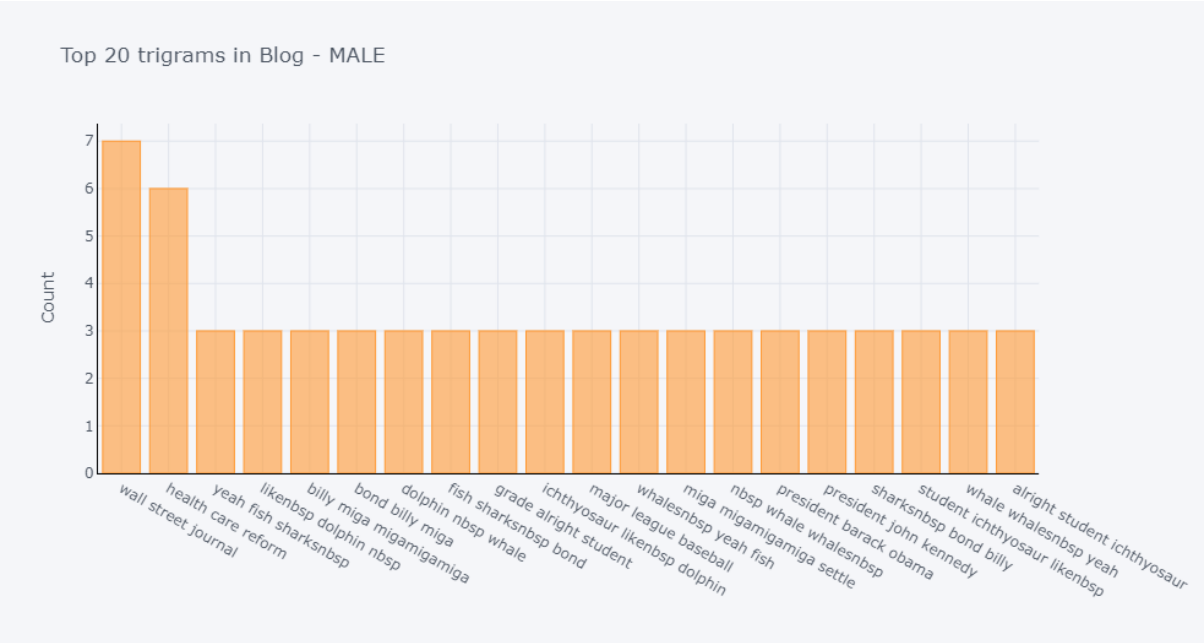
Top 20 Bi grams used by male & female





Top 20 Tri grams used by male & female



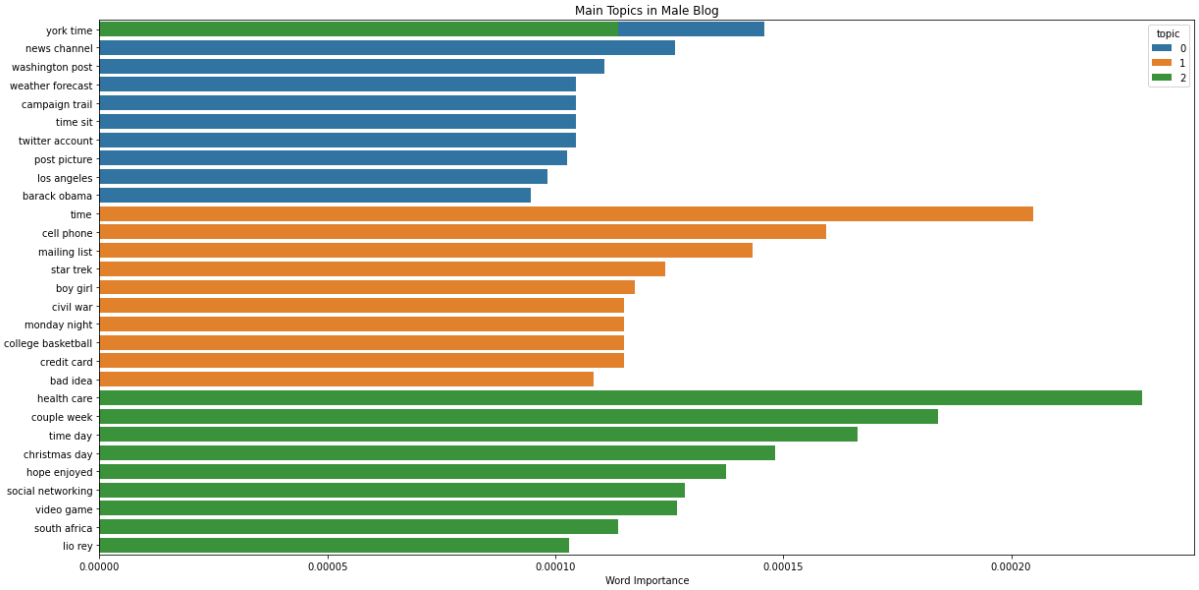


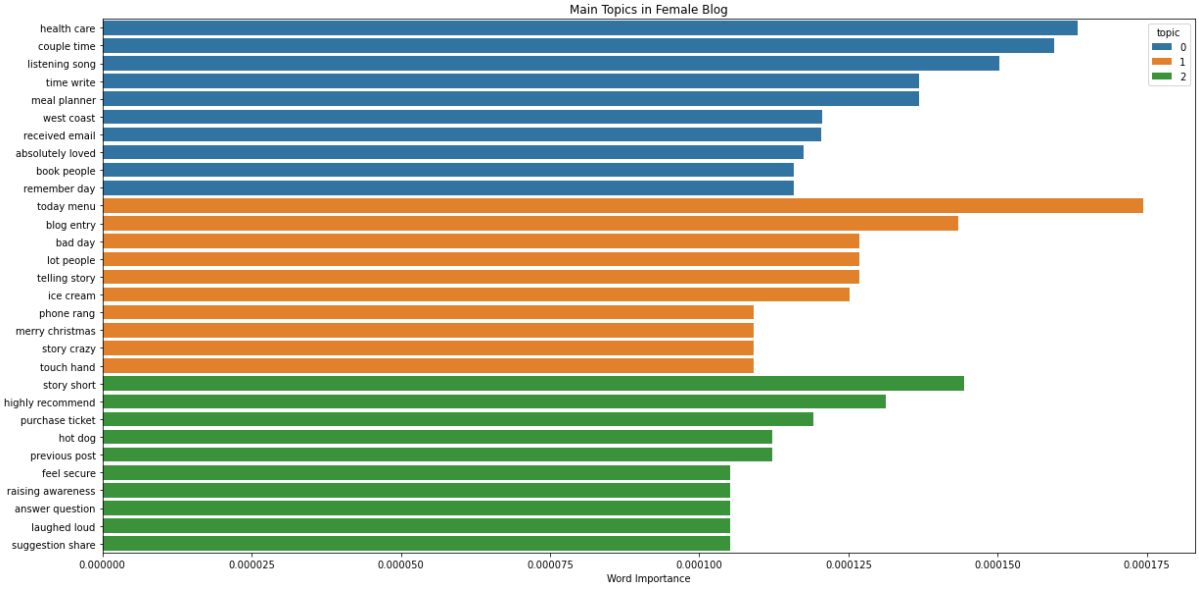
Topic Modelling

A topic model is a type of statistical model for discovering the abstract “topics” that occur in a collection of documents. The *Genism* package is specialized in topic modelling.

We have extracted topics using LDA (Latent Dirichlet Allocation): a generative statistical model that allows sets of observations to be explained by unobserved groups that explain why some parts of the data are similar. Basically, documents are represented as random mixtures over latent topics, where each topic is characterized by a distribution over words.

Let’s see what topics we can extract from male blogs and female blogs. We are supposed to specify the number of topics the model has to cluster, I have used 3.

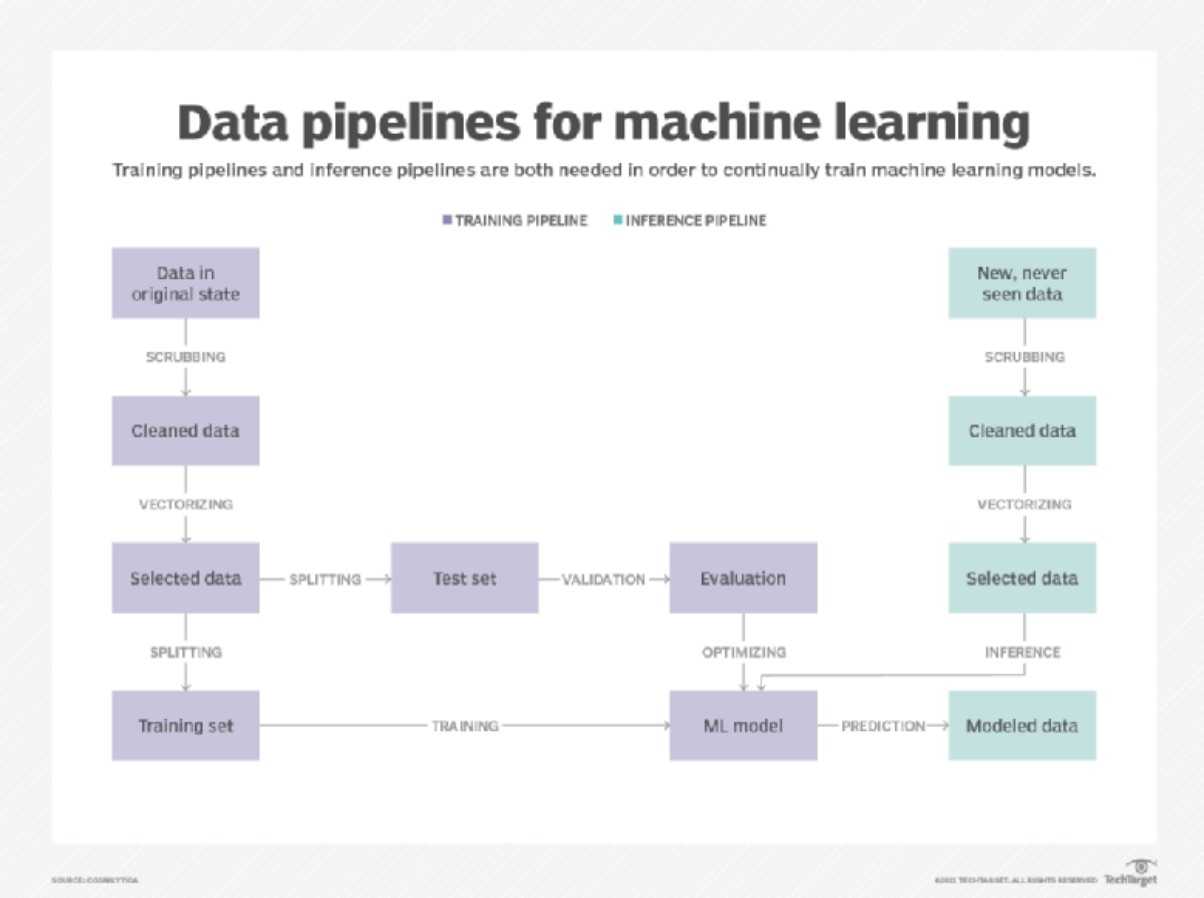




4. Modelling

This phase requires model technique selection and application, model training, model hyperparameter setting and adjustment, model validation, ensemble model development and testing, algorithm selection, and model optimization. To accomplish all that, the following actions are required:

- Select the right algorithm based on the learning objective and data requirements.
- Configure and tune hyperparameters for optimal performance and determine a method of iteration to attain the best hyperparameters.
- Identify the features that provide the best results.
- Determine whether model explainability or interpretability is required.
- Develop ensemble models for improved performance.
- Test different model versions for performance.
- Identify requirements for the model's operation and deployment.



Vectorization is the general process of turning a collection of text documents into numerical feature vectors. This specific strategy (tokenization, counting and normalization) is called the **Bag of Words** or "Bag of n-grams" representation. Documents are described by word occurrences while completely ignoring the relative position information of the words in the document. We have used the two most common techniques which are **CountVectorizer** and **TFIDFVectorizer** for vectorizing the text data

CountVectorizer: Counts the frequency of all words in our corpus, sorts them and grabs the most recurring features (using max_features hyperparameter). But these results are mostly biased and our model might loose out on some of the important less frequent features. These are all boolean values.

TFIDFVectorizer: TFIDF is a statistical measure said to have fixed the issues with CountVectorizer in some way. It consists of 2 parts, TF (Term Frequency) multiplied with IDF (Inverse Document Frequency). The main intuition being some words that appear frequently in 1 document and less frequently in other documents could be considered as providing extra insight for that 1 document and could help our model learn from this additional piece of information. In short, common words are penalized. These are relative frequencies identified as floating point numbers.

We have used four algorithms namely Multinomial Naïve Bayes, Logistic Regression, Random Forest and Adaboost. We will run each of our models in two ways: one using the CVEC (Count Vectorization) and another using TF-IDF. To run our models, we will set up a pipeline to run the vectorizer and the modeling one step. We will also be using GridSearchCV in order to tune the hyperparameters of both the vectorizer and the model. Finally, we will fit and score our models.

CVEC (Count Vectorization) Results

- Multinomial Naïve Bayes model gave us a training accuracy of .726 and a test accuracy of .686.
- Logistic Regression model gave us a training accuracy of .972 and a test accuracy of .640.
- Random Forest model gave us a training accuracy of .100 and a test accuracy of .663.
- AdaBoost model gave us a training accuracy of .100 and a test accuracy of .601.

TF-IDF Results

- Multinomial Naïve Bayes model gave us a training accuracy of .790 and a test accuracy of .688.
- Logistic Regression model gave us a training accuracy of .867 and a test accuracy of .684.
- Random Forest model gave us a training accuracy of .100 and a test accuracy of .655.
- AdaBoost model gave us a training accuracy of .999 and a test accuracy of .557.

Ensemble Method

Using voting classifier to vote between MultiNomial NB, Linear Regression, and RF (the three models that gave us the highest test accuracy). This will run the three models that we enter, compare the classifications, and vote on the winner. If two models indicate that a blog is written by male and one indicates that it is written by female, then the result will be that we classify the blog as written by male.

Because TF-IDF performed the best (highest test accuracy) for the three models that we are using in the voting classifier, we will only use TF-IDF here.
Voting classifier model gave us a training accuracy of .859 and a test accuracy of .680.

Deep Learning

While developing a DL model, we should keep in mind of key things like Model Architecture, Hyperparmeter Tuning and Performance of the model.

As we have seen in the word cloud, there are some words predominantly feature in both male and female blogs. This could be a problem if we are using a Machine Learning model like Naive Bayes, SVD, etc.. That's why we use Sequence Models.

Recurrent Neural Networks can handle a sequence of data and learn a pattern of input sequence to give either sequence or scalar value as output. In our case, the Neural Network outputs a scalar value prediction.

For model architecture, we use

1) **Embedding Layer** - Generates Embedding Vector for each input sequence.

2) **Conv1D Layer** - Its using to convolve data into smaller feature vectors.

3) **LSTM** - Long Short Term Memory, its a variant of RNN which has memory state cell to learn the context of words which are at further along the text to carry contextual meaning rather than just neighbouring words as in case of RNN.

4) **Dense** - Fully Connected Layers for classification

Optimization Algorithm

We use Adam, optimization algorithm for Gradient Descent.

Callbacks

Callbacks are special functions which are called at the end of an epoch. We can use any functions to perform specific operation after each epoch. I used two callbacks here,

- **LRScheduler** - It changes a Learning Rate at specific epoch to achieve more improved result. In this notebook, the learning rate exponentially decreases after remaining same for first 10 Epoch.
- **ModelCheckPoint** - It saves best model while training based on some metrics. Here, it saves the model with minimum Validity Loss.

5. Evaluation

Model evaluation can be considered the quality assurance of machine learning. Adequately evaluating model performance against metrics and requirements determines how the model will work in the real world.

For our Blog gender classification problem, we will consider the following,

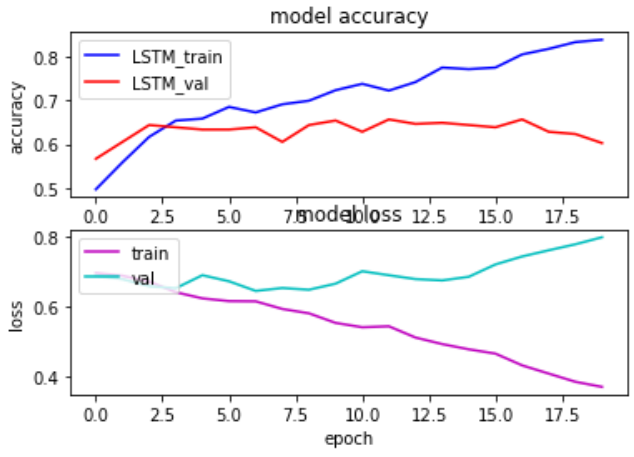
- Evaluate the models using test dataset
- Determine the confusion matrix values
- Hyper tuning for optimal performance

Now that we have run all of our models, we can formally compare them. We will create a function that will take in the models that we ran above and create the predictions for the model. Next, it will calculate all of the metrics needed for the confusion matrix (true negatives, false positives, false negatives, and false positives). Finally, it will calculate the accuracy and append this to a DataFrame.

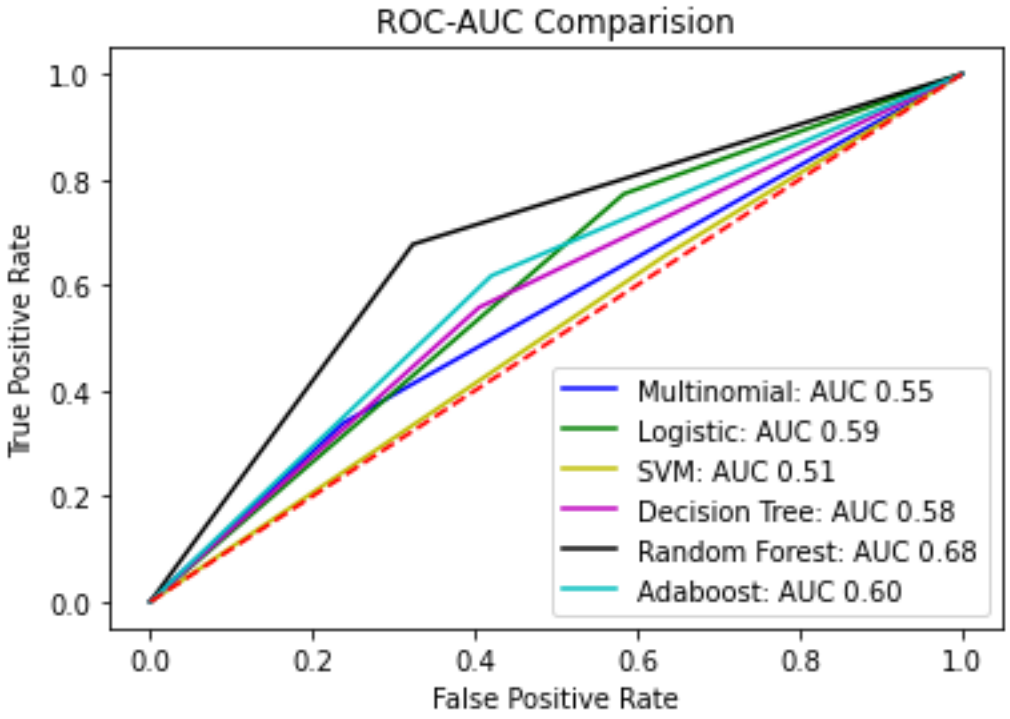
Below are the test accuracy scores of all models. The highest performing set was the multinomial naïve bayes with TF-IDF. In all cases, the TF-IDF version of each model outscored its CVEC counterpart. The results of this table are shown below,

	model	acc
1	MB TVEC	0.688462
0	MB CVEC	0.686538
3	LR TVEC	0.684615
8	Voting	0.682692
5	RF TVEC	0.655769
2	LR CVEC	0.640385
4	RF CVEC	0.640385
7	ADA TVEC	0.580769
6	ADA CVEC	0.555769

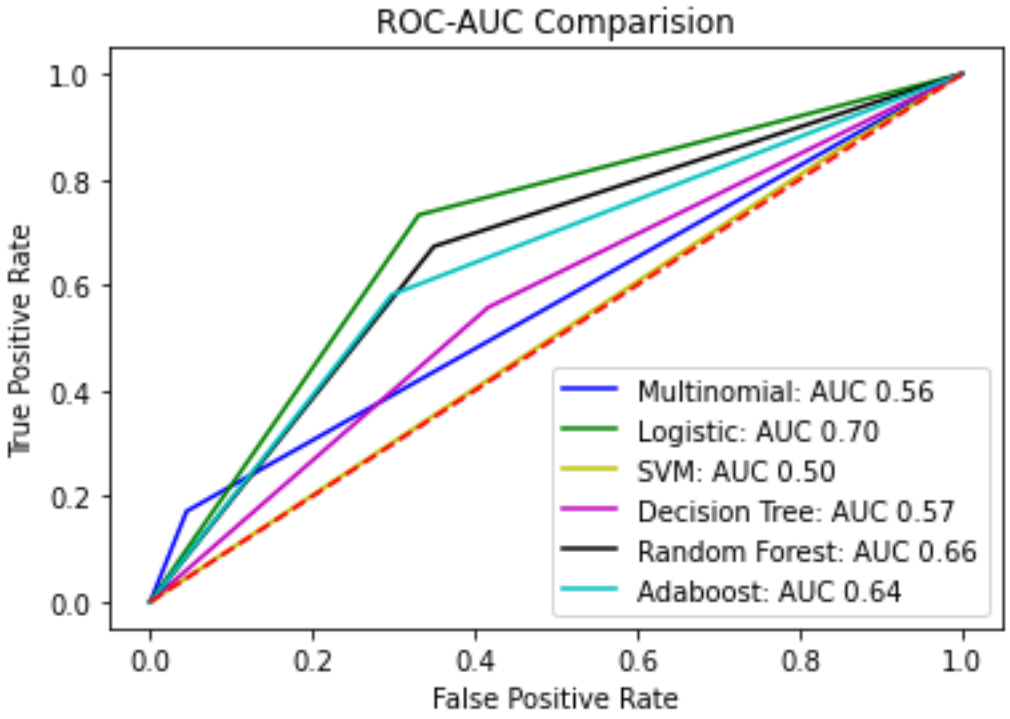
Deep Learning Result



Using Textual Features



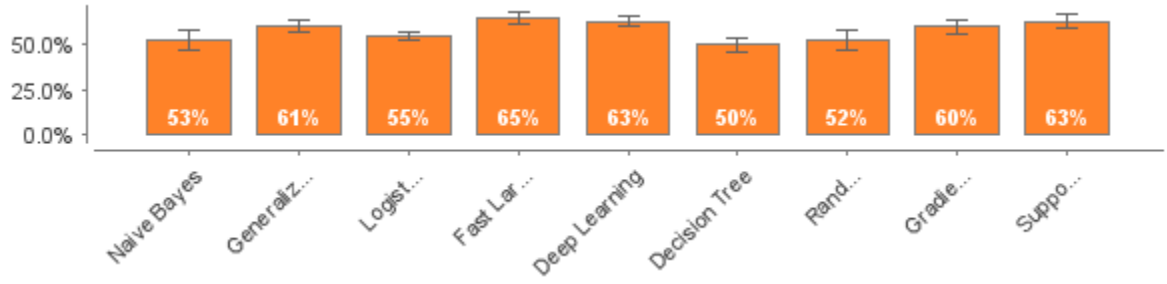
Using POS Tagging

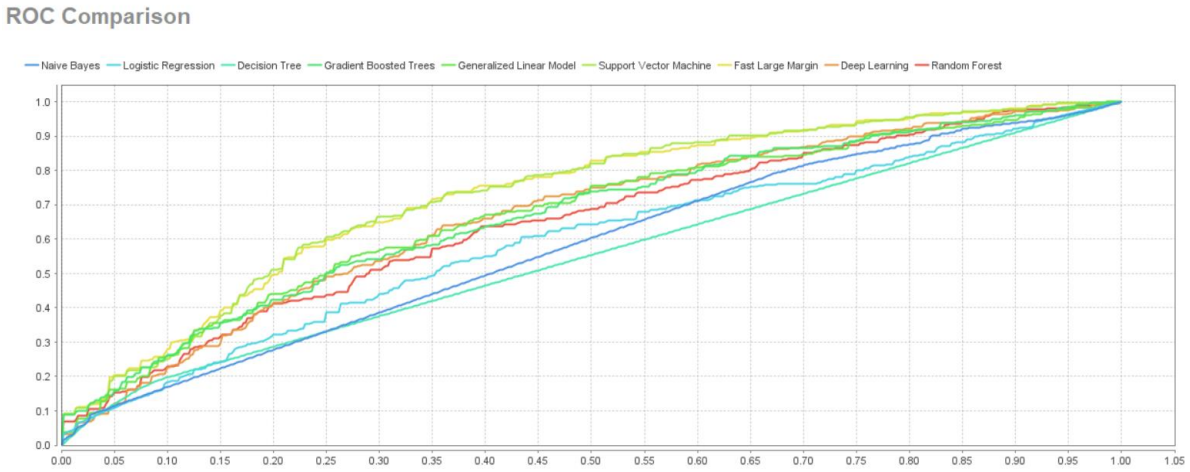


Result Summary

- Using only text data, it was found that Multinomial Naïve Bayes was performing good compared to other algorithms.
- Using text + textual features, it was found that Random Forest was performing good compared to other algorithms.
- Using text + POS features, it was found that logistic regression was performing good compared to other algorithms.

Model comparison with RapidMiner





6. Deployment

There are many platforms that allow us to deploy a web application. Two good ones are Dash Enterprise and Heroku. We can use Heroku since deploying an app is free (with some limitations), and for a small amount we can get a fully-operating server. I have used Heroku to deploy the app.

To deploy the application on Heroku we need to push all the files to the Github which will be linked to Heroku. Then upload following files to Github repository:

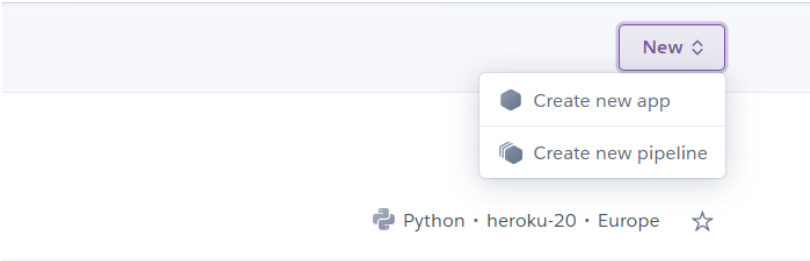
- NLP model pickle
- TFIDF Vectorizer pickle
- Flask app

- HTML & CSS files
- Requirements file
- Procfile

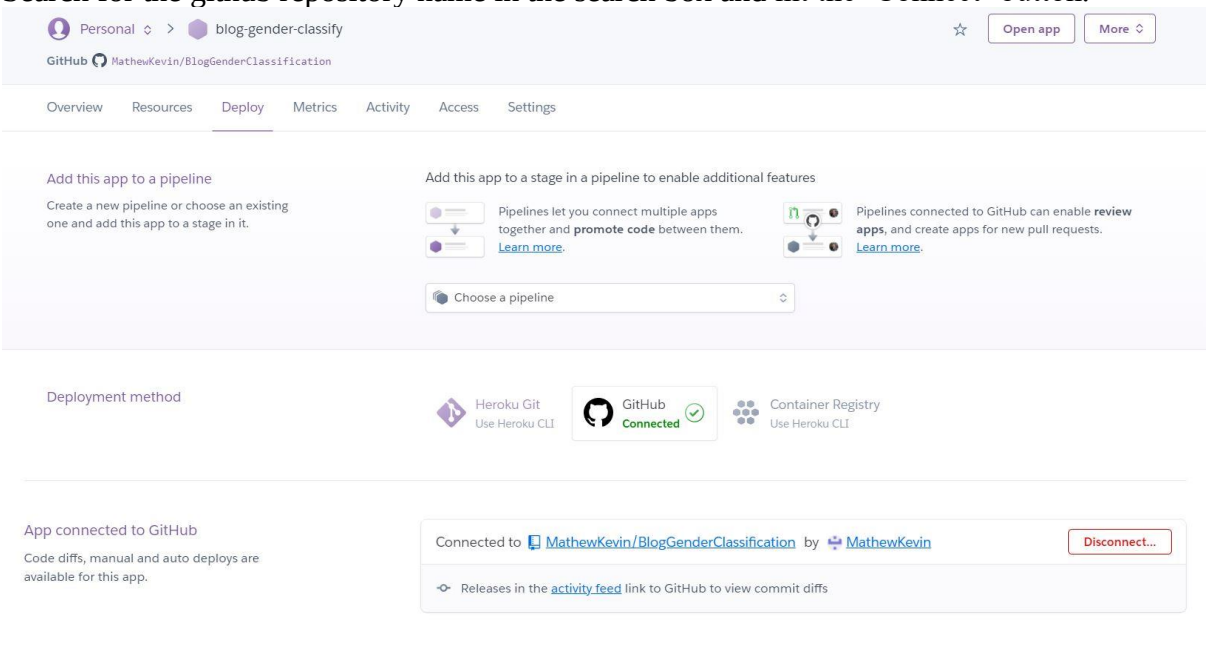
The requirements is a text file which contains all the dependencies (libraries like pandas, flask etc.) and their versions required for the app to run as expected. Heroku will install all these dependencies prior to deploying the app. Procfile is a configuration file which looks something like below:

```
web: gunicorn app1:app
```

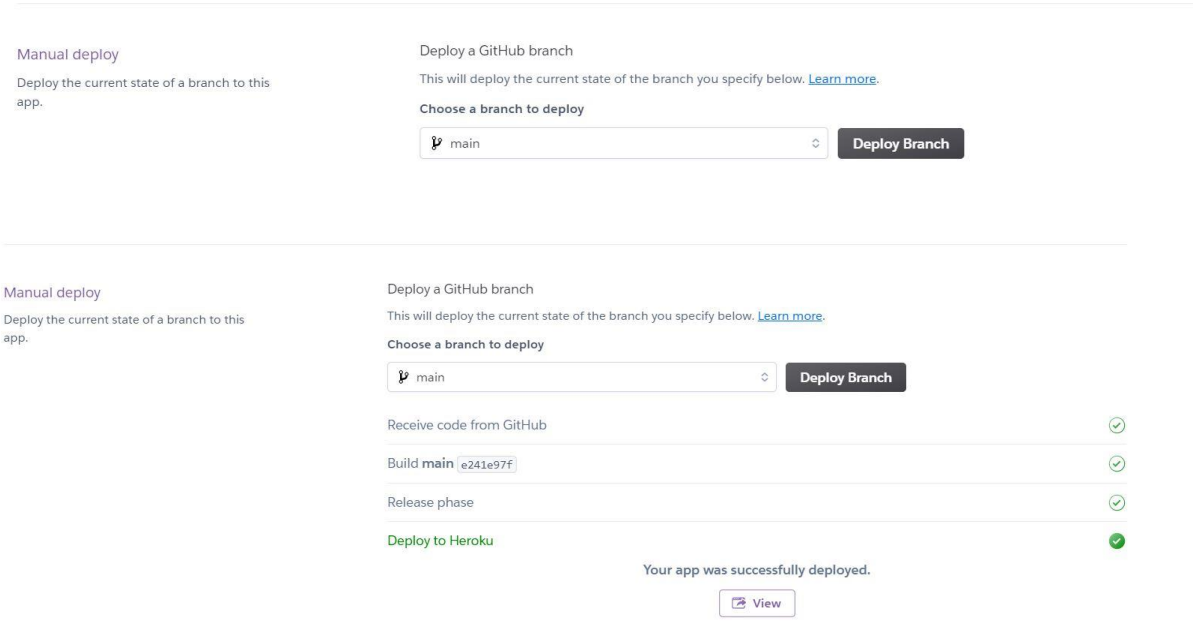
Here ‘app1’ is my flask file name which is ‘app1.py’ and ‘app’ is the flask app we defined earlier in the code(refer ipynb file). This just tells Heroku to look for the flask file named app1 and in it the flask app defined as ‘app’. After pushing all the files to a new github repository, we need to create an app on Heroku and connect to our github repository to it. To do this, go to Heroku. Create an account if you haven’t already and go to ‘create new app’.



In the newly created app, we need to connect to the github repository which has the code. Search for the github repository name in the search box and hit the ‘Connect’ button.



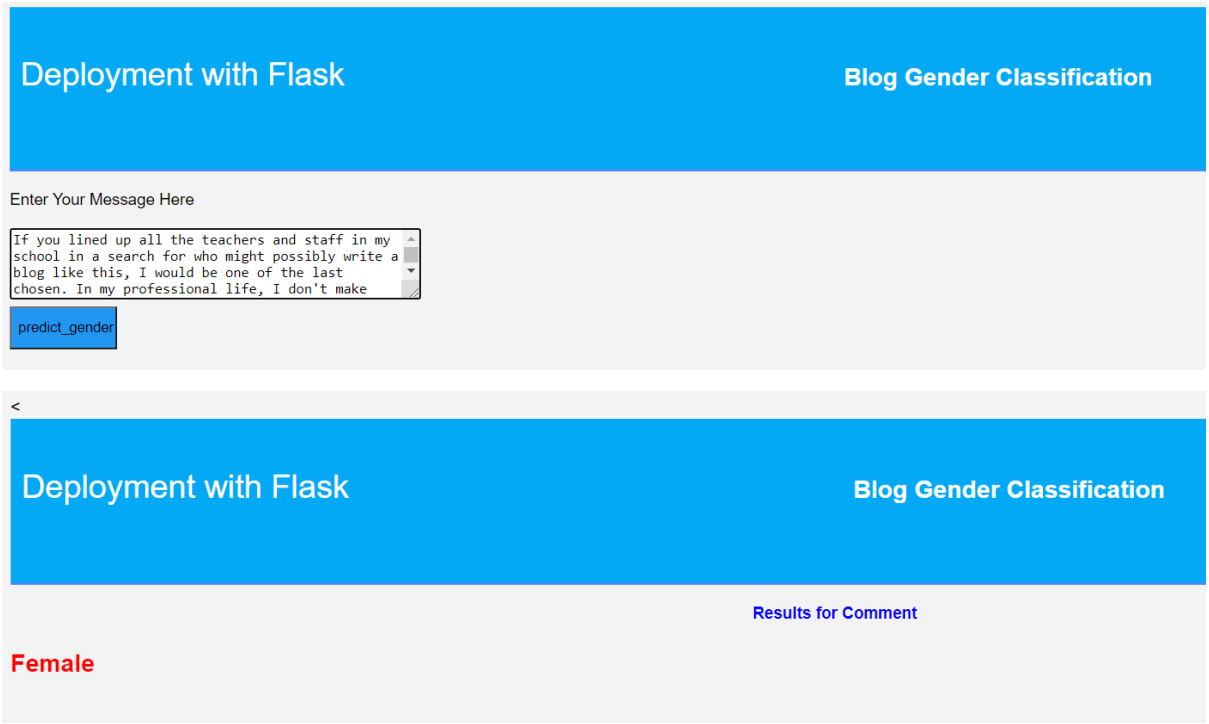
Once connected we just need to hit the ‘Deploy’ button and we’re done. Heroku will first check for the requirements file and install all the dependencies and then deploy the flask app to public host. A URL will be generated which is the URL of our application. Just copy it and open it in the browser.



Let’s check the app by running it, the URL is: https://blog-gender-classify.herokuapp.com/

BLOG	GENDER
Beyond Getting There: What Travel Days Show UsToday’s guest post is by Gillian at On	F
I remember so much about the island; the large Lighthouse that helped me fall asleep s	F
I have had asthma and allergies my entire life. While I bet many asthmatics might disag	M
The last few days have been an emotional rollercoaster for the team behind the show.	M
If you lined up all the teachers and staff in my school in a search for who might possibl	F
Welcome to everyone to ACL! It's pretty rare for me to end up conferencing in a count	M
This flat iron is great! I've had mine for 3.5 years now and it hasn't disappointed me ye	F
Here's a question: How many shots did it take for Al Oliver to settle on this bat? Also, if	M
I didn't attend this year's Photo Marketing Association (PMA) show, but as I read the cc	M

We will use the above highlighted text which has been written by a female and see if our model is predicting correctly.



Our model has correctly predicted that it was written by a female.