Naïve Bayes and Text Mining Report

- Text mining require certain libraries which are needed to be installed and such libraries are
  - **Spacy**
  - **NLTK**
  - **En_core_web_md**, this is the English word dictionary
- Importing the needful libraries such as **NumPy, Pandas, Matplotlib, String** (for string operations)
- Loaded the dataset and found the shape of the data
- Downloaded the NLTK data files both **punkt** and **stop words**

Data Pre-Processing:

- Imported the stop words and word tokenizer from the NLTK module
- Along with those we also imported tfidfVectorizer and **ENGLISH_STOP_WORDS** from feature extraction module of sklearn
- Defined a function to
  - *Normalise the text [convert them to lower case]*
  - *Remove punctuations*
  - *Tokenize the sentences to words [make sentences into smaller chunks called tokens]*
  - *Remove stop words*
- Applied the preprocessing function we defined to completely do the mentioned steps one by one
- As these are text data from blogs, when we are transforming them to their root form, we should be careful. So, to avoid any stop words to be formed we used **Lemmatization** to change the similar words to their root form
- Imported the **WordNetLemmatizer** module and lemmatized the data

Feature Extraction:

- We are using the **TF-IDF method** to do **Feature Extraction** and given the **max feature** as 1000 to maintain the dimensionality and don’t make the process more complex
- Plotted the distribution of the categories of the labels in the data

Naïve Bayes Model:

- Imported the **train-test-split, MultinomialNB, evaluation metrics, cross validation score** from sklearn
- Split the Lemmatized data into train and test with test size as 20% of the original data
- Built the model with the train data
- And stored the predictions with the test data in **y_pred**
- Calculated the evaluation metrics as **Accuracy, Precision, Recall, F1-Score and Classification report**

Sentiment Analysis:

- Imported **Text Blob** for the **Sentiment Analysis**
- Defined a function which calculates the **sentiment polarity** with the help of Text Blob module function
- and applied the function to the lemmatized data and found out the sentiment counts of the data

- plotted the **sentiment distribution** of the data on a **bar chart**
- grouped the data with labels and sentiment counts and plotted that data on to the bar chart (**sentiment distribution across the categories**)
- know we had combined the text data into a single string for the **word cloud** and generated a word cloud which shows the most used words in the text data
- defined a function that check for most used words in a category and display them if any
- developed a **Heatmap** between the sentiment distribution along the categories
- and checked with the cross-validation score with mean