Twitter Sentiment Prediction

1.1 Introduction

Taking a look at twitter, there are a lot of tweets generated every single second by different users. In addition, some of these tweets might be positive while the others might be negative. It would be really useful if machine learning and deep learning is deployed in real-time to classify the texts and tweets as positive, negative or neutral respectively. As a result of this, a lot of time and money would be saved by the company without the need to use manpower for these simple tasks respectively.

In this machine learning project, we would be working with real-time twitter tweets and predicting the sentiment of the text whether it is positive, negative or neutral. With the help of deep neural networks along with hyperparameter tuning, we would be classifying the tweets, ensuring that there is a good business value for the company by analyzing the text and understanding the working of machine learning models.

1.2 Metrics

1. *Accuracy*
2. *Precision*
3. *Recall*

1.3 Source

The dataset is taken from Kaggle. It is available in the website below. Feel free to download the dataset.

https://www.kaggle.com/c/tweet-sentiment-extraction/data

Table of Contents

Let us now start the project by reading some useful libraries that we would be working with from the start.

In [109…

```
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.models import Sequential
from wordcloud import WordCloud, STOPWORDS
from sklearn.preprocessing import LabelBinarizer
from tensorflow.keras.utils import to_categorical
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer, TfidfVecto
!pip install wordcloud
```

```
Requirement already satisfied: wordcloud in /opt/conda/lib/python3.7/sit
e-packages (1.8.1)
Requirement already satisfied: pillow in /opt/conda/lib/python3.7/site-p
ackages (from wordcloud) (7.2.0)
Requirement already satisfied: numpy>=1.6.1 in /opt/conda/lib/python3.7/
site-packages (from wordcloud) (1.19.5)
Requirement already satisfied: matplotlib in /opt/conda/lib/python3.7/si
te-packages (from wordcloud) (3.4.1)
Requirement already satisfied: cycler>=0.10 in /opt/conda/lib/python3.7/
site-packages (from matplotlib->wordcloud) (0.10.0)
Requirement already satisfied: pyparsing>=2.2.1 in /opt/conda/lib/python
3.7/site-packages (from matplotlib->wordcloud) (2.4.7)
Requirement already satisfied: kiwisolver>=1.0.1 in /opt/conda/lib/pytho
n3.7/site-packages (from matplotlib->wordcloud) (1.3.1)
Requirement already satisfied: python-dateutil>=2.7 in /opt/conda/lib/py
thon3.7/site-packages (from matplotlib->wordcloud) (2.8.1)
Requirement already satisfied: six in /opt/conda/lib/python3.7/site-pack
ages (from cycler>=0.10->matplotlib->wordcloud) (1.15.0)
```

1. Reading the data

We would be first reading the data that is present in the kaggle repository to ensure that we work with it.

In [110…

```
df = pd.read_csv('../input/training-data/train.csv')
```

Reading the head of the dataframe, we see that there are different columns such as textID, text, selected_text and the output that we are going to be predict which is sentiment respectively.

In [111…

```
df.head()
```

Out[111…

	textID	text	selected_text	sentiment
0	cb774db0d1	I'd have responded, if I were going	I'd have responded, if I were going	neutral

1	549e992a42	Sooo SAD I will miss you here in San Diego!!!	Sooo SAD	negative
2	088c60f138	my boss is bullying me...	bullying me	negative
3	9642c003ef	what interview! leave me alone	leave me alone	negative
4	358bd9e861	Sons of ****, why couldn't they put them on t...	Sons of ****,	negative

We would be dropping the selected text as the test set does not contain those values. It would be good to leave only the text for the machine learning models for predictions respectively.

In [112…
```
df.drop(['selected_text'], axis = 1, inplace = True)
```

Since we have removed the selected text, we are only left with the actual text values along with the sentiment whether it is positive, negative or neutral output respectively.

In [113…
```
df.head()
```

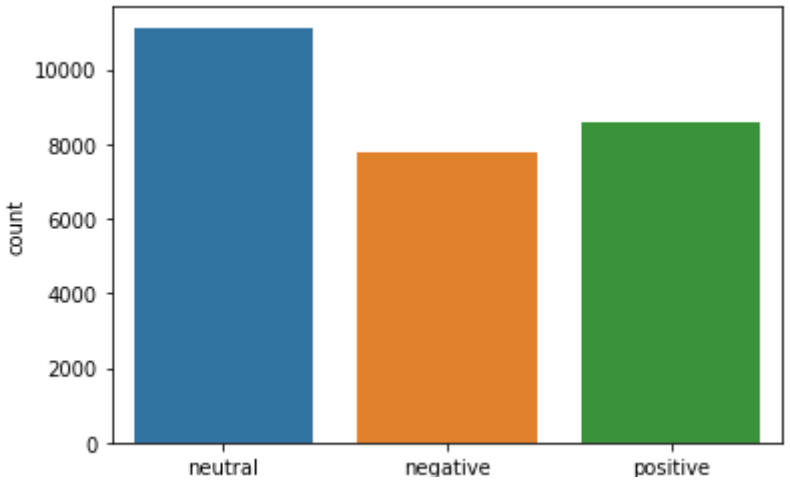
Out[113…

	textID	text	sentiment
0	cb774db0d1	I'd have responded, if I were going	neutral
1	549e992a42	Sooo SAD I will miss you here in San Diego!!!	negative
2	088c60f138	my boss is bullying me...	negative
3	9642c003ef	what interview! leave me alone	negative
4	358bd9e861	Sons of ****, why couldn't they put them on t...	negative

2. Countplot of the Sentiment

As can be seen, there seems to be more neutral sentiment in the text compared to positive or negative tweets. That is true in real life as most of the tweets would be quite neutral without them being either too positive or too negative respectively.

In [114…
```
sns.countplot(x = 'sentiment', data = df)
plt.show()
```



sentiment

Taking a look at the shape of the data, we see that there are about 27k data points that we are going to be working and applying our knowledge.

In [115…
```
df.shape
```

Out[115…
```
(27481, 3)
```

Reading a few lines in a particular row of the text, we see that it is quite a simple tweet without being too lengthy.

In [116…
```
df['text'].iloc[0]
```

Out[116…
```
' I`d have responded, if I were going'
```

Taking a look at the total number of characters from a single text, we see that there are about 36 characters respectively.

In [117…
```
len(df['text'].iloc[0])
```

Out[117…
```
36
```

Let us now read the head of the text and see the values that are present to understand the working of the machine learning models respectively.

In [118…
```
df['text'].head()
```

Out[118…
```
0                I`d have responded, if I were going
1       Sooo SAD I will miss you here in San Diego!!!
2                           my boss is bullying me...
3                      what interview! leave me alone
4    Sons of ****, why couldn`t they put them on t...
Name: text, dtype: object
```

Looking at the information present in our dataframe, we can see that almost all the values are not-null except a few that we are going to be clearing to ensure that those values don't interfere in the machine learning models and predictions respectively.

In [119…
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 27481 entries, 0 to 27480
Data columns (total 3 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   textID     27481 non-null  object
 1   text       27480 non-null  object
 2   sentiment  27481 non-null  object
dtypes: object(3)
memory usage: 644.2+ KB
```

In [120…
```
df['text'].iloc[0]
```

Out[120… ' I`d have responded, if I were going'

In [121…
```
text_length_list = []
for i in range(len(df)):
    if isinstance(df['text'].iloc[i], float) == True:
        print(df['text'].iloc[i])
```

nan

In [122…
```
isinstance("suhas", float)
```

Out[122… False

In [123…
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 27481 entries, 0 to 27480
Data columns (total 3 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   textID     27481 non-null  object
 1   text       27480 non-null  object
 2   sentiment  27481 non-null  object
dtypes: object(3)
memory usage: 644.2+ KB
```

In [124…
```
df.dropna(inplace = True)
```

In [125…
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 27480 entries, 0 to 27480
Data columns (total 3 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   textID     27480 non-null  object
 1   text       27480 non-null  object
 2   sentiment  27480 non-null  object
dtypes: object(3)
memory usage: 858.8+ KB
```

In [126…
```
df['text_length'] = df['text'].apply(lambda x: len(x))
```

In [127…
```
df.head()
```

Out[127…

	textID	text	sentiment	text_length
0	cb774db0d1	I'd have responded, if I were going	neutral	36
1	549e992a42	Sooo SAD I will miss you here in San Diego!!!	negative	46
2	088c60f138	my boss is bullying me...	negative	25
3	9642c003ef	what interview! leave me alone	negative	31
4	358bd9e861	Sons of ****, why couldn't they put them on t...	negative	75

In [128…
```
df['text_words'] = df['text'].apply(lambda x: len(x.split()))
```

In [129…
```
df.head()
```

Out[129…

	textID	text	sentiment	text_length	text_words
0	cb774db0d1	I'd have responded, if I were going	neutral	36	7
1	549e992a42	Sooo SAD I will miss you here in San Diego!!!	negative	46	10
2	088c60f138	my boss is bullying me...	negative	25	5
3	9642c003ef	what interview! leave me alone	negative	31	5
4	358bd9e861	Sons of ****, why couldn't they put them on t...	negative	75	14

In [130…
```
df.head()
```

Out[130…

	textID	text	sentiment	text_length	text_words
0	cb774db0d1	I'd have responded, if I were going	neutral	36	7
1	549e992a42	Sooo SAD I will miss you here in San Diego!!!	negative	46	10
2	088c60f138	my boss is bullying me...	negative	25	5
3	9642c003ef	what interview! leave me alone	negative	31	5
4	358bd9e861	Sons of ****, why couldn't they put them on t...	negative	75	14

In [131…
```
## Taking separate dataframes for different values such as positive, ne

positive_df = df[df['sentiment'] == 'positive']
negative_df = df[df['sentiment'] == 'negative']
neutral_df = df[df['sentiment'] == 'neutral']
```

In [132…
```
print("The shape of the dataframe that contains only the positive revie
print("The shape of the dataframe that contains only the negative revie
print("The shape of the dataframe that contains only the neutral review
```

```
The shape of the dataframe that contains only the positive reviews is:
(8582, 5)
The shape of the dataframe that contains only the negative reviews is:
(7781, 5)
The shape of the dataframe that contains only the neutral reviews is:
(11117, 5)
```

In [133…
```
wordcloud = WordCloud(width = 500, height = 500)
```

In [134…
```
df.head()
```

Out[134…

	textID	text	sentiment	text_length	text_words

0	cb774db0d1	I'd have responded, if I were going	neutral	36	7
1	549e992a42	Sooo SAD I will miss you here in San Diego!!!	negative	46	10
2	088c60f138	my boss is bullying me...	negative	25	5
3	9642c003ef	what interview! leave me alone	negative	31	5
4	358bd9e861	Sons of ****, why couldn't they put them on t...	negative	75	14

In [135…
```
positive_df.head()
```

Out[135…

	textID	text	sentiment	text_length	text_words
6	6e0c6d75b1	2am feedings for the baby are fun when he is a...	positive	64	14
9	fc2cbefa9d	Journey!? Wow... u just became cooler. hehe....	positive	69	10
11	16fab9f95b	I really really like the song Love Story by Ta...	positive	56	11
21	e48b0b8a23	Playing Ghost Online is really interesting. Th...	positive	135	24
25	e00c6ef376	the free fillin`app on my ipod is fun, im add...	positive	51	11

In [136…
```
positive_text = []
for i in range(len(positive_df)):
    positive_text.append(positive_df['text'].iloc[i])
```

In [137…
```
positive_text[:5]
```

Out[137…
```
['2am feedings for the baby are fun when he is all smiles and coos',
 ' Journey!? Wow... u just became cooler.  hehe... (is that possibl
e!?)',
 'I really really like the song Love Story by Taylor Swift',
 'Playing Ghost Online is really interesting. The new updates are Kirin
pet and Metamorph for third job.  Can`t wait to have a dragon pet',
 'the free fillin` app on my ipod is fun, im addicted']
```

3. Positive Text WordCloud

Worcloud gives us a good idea about the number of repeating words by the size of them. We see that there are some positive words such as "thank" and "good "which make up most of the positive reviews. There are also some words that occur quite rare such as "awesome" but these words also make a mark in the decisions respectively. In addition to this, we also find that there are very few words that are negative in the positive text. Therefore, we can conclude that words are a good indication of the polarity and sentiment of the text respectively.

In [138…
```
wordcloud = WordCloud(stopwords = stopwords)
wordcloud.generate(' '.join(positive_text))
```

```
plt.figure(figsize = (10, 10))
plt.imshow(wordcloud)
plt.show()
```

In [139…
```
negative_text = []
for i in range(len(negative_df)):
    negative_text.append(negative_df['text'].iloc[i])
```

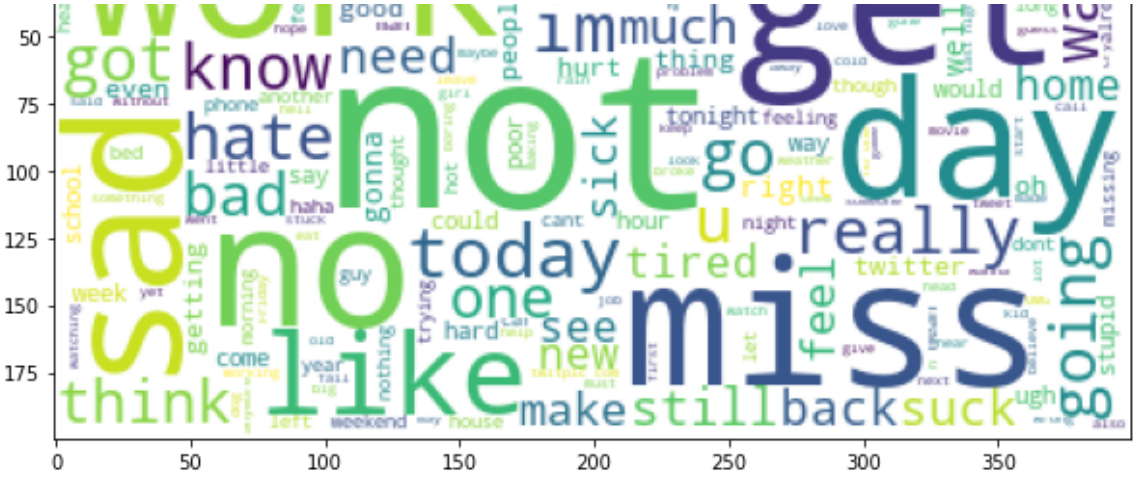
In [140…
```
negative_text[0: 5]
```

Out[140…
```
[' Sooo SAD I will miss you here in San Diego!!!',
 'my boss is bullying me...',
 ' what interview! leave me alone',
 ' Sons of ****, why couldn`t they put them on the releases we already
bought',
 'My Sharpie is running DANGERously low on ink']
```

4. Negative Text WordCloud

As can be seen in the wordcloud, there are some words such as "miss" and "no" which are considered to be negative respectively. In addition to this, we see that there are some words such as "work" and "sorry" that also tend to be negative as this is quite true in real-life where we say "sorry" for some negative things done and so on. Therefore, getting the wordcloud would ensure that we get to know the words present in the wordcloud dictionary and ensure that we understand the full context of the review respectively.

In [141…
```
wordcloud = WordCloud(stopwords = stopwords, background_color = 'white'
wordcloud.generate(''.join(negative_text))
plt.figure(figsize = (10, 10))
plt.imshow(wordcloud)
plt.show()
```



In [142…

```
df.head()
```

Out[142…

	textID	text	sentiment	text_length	text_words
0	cb774db0d1	I'd have responded, if I were going	neutral	36	7
1	549e992a42	Sooo SAD I will miss you here in San Diego!!!	negative	46	10
2	088c60f138	my boss is bullying me...	negative	25	5
3	9642c003ef	what interview! leave me alone	negative	31	5
4	358bd9e861	Sons of ****, why couldn't they put them on t...	negative	75	14

In [143…

```
negative_df.head()
```

Out[143…

	textID	text	sentiment	text_length	text_words
1	549e992a42	Sooo SAD I will miss you here in San Diego!!!	negative	46	10
2	088c60f138	my boss is bullying me...	negative	25	5
3	9642c003ef	what interview! leave me alone	negative	31	5
4	358bd9e861	Sons of ****, why couldn't they put them on t...	negative	75	14
12	74a76f6e0a	My Sharpie is running DANGERously low on ink	negative	44	8

In [144…

```
positive_df.head()
```

Out[144…

	textID	text	sentiment	text_length	text_words
6	6e0c6d75b1	2am feedings for the baby are fun when he is a...	positive	64	14
9	fc2cbefa9d	Journey!? Wow... u just became cooler. hehe....	positive	69	10
11	16fab9f95b	I really really like the song Love Story by Ta...	positive	56	11
21	e48b0b8a23	Playing Ghost Online is really interesting. Th...	positive	135	24

25	e00c6ef376	the free fillin`app on my ipod is fun, im add...	positive	51	11

5. List of Stopwords

Let us now make a list of all the stopwords that we are going to be using for our machine learning purposes. We see that there are some good list of stopwords that I have taken from the link that is provided in the below cell. These would be used for the stopwords and ensure that we are getting the best results respectively.

In [150…

```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'not
stopwords= ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves'
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'it
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'th
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'h
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or',
            'at', 'by', 'for', 'with', 'about', 'against', 'between', '
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out',
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'h
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should',
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", '
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn',
            'won', "won't", 'wouldn', "wouldn't"]
```

In [151…

```
df.head()
```

Out[151…

	textID	text	sentiment	text_length	text_words
0	cb774db0d1	I'd have responded, if I were going	neutral	36	7
1	549e992a42	Sooo SAD I will miss you here in San Diego!!!	negative	46	10
2	088c60f138	my boss is bullying me...	negative	25	5
3	9642c003ef	what interview! leave me alone	negative	31	5
4	358bd9e861	Sons of ****, why couldn't they put them on t...	negative	75	14

In [152…

```
df.drop(['textID'], axis = 1, inplace = True)
```

In [153…

```
df.head()
```

Out[153…

	text	sentiment	text_length	text_words
0	I'd have responded, if I were going	neutral	36	7
1	Sooo SAD I will miss you here in San Diego!!!	negative	46	10
2	my boss is bullying me...	negative	25	5

3	what interview! leave me alone	negative	31	5
4	Sons of ****, why couldnt they put them on t...	negative	75	14

In [154…
```
X = df.drop(['sentiment'], axis = 1)
y = df['sentiment']
```

6. Dividing the Data into Training and Cross Validation Data

Now we are going to be dividing the data into training and cross validation data and ensure that we understand the machine learning model well. We are going to be dividing the data into 2 parts where 70 percent of the data is taken as the training data and the 30 percent of the data would be taken as the test data. There is a random state assigned to it and split to ensure that we get a good accuracy.

In [155…
```
X_train, X_cv, y_train, y_cv = train_test_split(X, y, test_size = 0.3,
```

In [156…
```
X_train.shape
```

Out[156… (19236, 3)

In [157…
```
X_cv.shape
```

Out[157… (8244, 3)

7. Function for Replacing Words

There is a separate function that is being used to replace the words and substitute them with the other words to ensure that we get the best machine learning results respectively.

In [159…
```
# https://stackoverflow.com/a/47091490/4084039
import re

def decontracted(phrase):
    # specific
    phrase = re.sub(r"won't", "will not", phrase)
    phrase = re.sub(r"can\'t", "can not", phrase)

    # general
    phrase = re.sub(r"n\'t", " not", phrase)
    phrase = re.sub(r"\'re", " are", phrase)
    phrase = re.sub(r"\'s", " is", phrase)
    phrase = re.sub(r"\'d", " would", phrase)
    phrase = re.sub(r"\'ll", " will", phrase)
    phrase = re.sub(r"\'t", " not", phrase)
    phrase = re.sub(r"\'ve", " have", phrase)
    phrase = re.sub(r"\'m", " am", phrase)
    return phrase
```

In [160…
```
!pip install tqdm
```

```
Requirement already satisfied: tqdm in /opt/conda/lib/python3.7/site-pac
kages (4.59.0)
```

8. Preprocessing the Text

It is now time to preprocess the text and understand the output. We are going to be using the above functions and also estimate the time it takes to complete the entire preprocessing respectively. Taking into account the different preprocessing text values, we are going to be appending those values and understanding the output respectively.

In [161…
```
# Combining all the above stundents
from tqdm import tqdm
preprocessed_text = []
# tqdm is for printing the status bar
for sentence in tqdm(X_train['text'].values):
    sent = decontracted(sentence)
    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e.lower() not in stopwor
    preprocessed_text.append(sent.lower().strip())
```

```
100%|██████████| 19236/19236 [00:00<00:00, 24420.95it/s]
```

In [162…
```
preprocessed_text[0: 5]
```

Out[162…
```
['24 hours since dog put sleep rip old friend',
 'not feeling comfortable today',
 'hurray twin girls born beautiful may day',
 'bah h8 waking',
 'textmate crashed first time 3 months not bad actually textmate']
```

In [163…
```
for i in range(len(X_train)):
    X_train['text'].iloc[i] = preprocessed_text[i]
```

In [164…
```
X_train.head()
```

Out[164…

	text	text_length	text_words
19832	24 hours since dog put sleep rip old friend	78	18
10340	not feeling comfortable today	41	6
11044	hurray twin girls born beautiful may day	48	9
14088	bah h8 waking	19	5
20333	textmate crashed first time 3 months not bad a...	80	14

In [165…
```
# Combining all the above stundents
from tqdm import tqdm
preprocessed_text = []
# tqdm is for printing the status bar
for sentence in tqdm(X_cv['text'].values):
    sent = decontracted(sentence)
```

```
    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e.lower() not in stopwor
    preprocessed_text.append(sent.lower().strip())
```

```
100%|██████████| 8244/8244 [00:00<00:00, 23929.39it/s]
```

In [166…
```
preprocessed_text[0: 5]
```

Out[166…
```
['mom happy present yayy',
 'no surprise probably know',
 'nope difference 5 omr c class anyone notice flight number wy flight l
hr wy 911',
 'smells badly garlic',
 'friday whole different meaning work saturday sunday']
```

In [167…
```
for i in range(len(X_cv)):
    X_cv['text'].iloc[i] = preprocessed_text[i]
```

In [168…
```
X_cv.head()
```

Out[168…

	text	text_length	text_words
7214	mom happy present yayy	38	8
19840	no surprise probably know	67	11
19409	nope difference 5 omr c class anyone notice fl...	102	19
6600	smells badly garlic	25	5
7807	friday whole different meaning work saturday s...	78	14

In [169…
```
binarizer = LabelBinarizer()
y_train_converted = binarizer.fit_transform(y_train)
binarizer = LabelBinarizer()
y_cv_converted = binarizer.fit_transform(y_cv)
```

In [170…
```
y_cv_converted
```

Out[170…
```
array([[0, 0, 1],
       [0, 1, 0],
       [0, 1, 0],
       ...,
       [1, 0, 0],
       [0, 0, 1],
       [1, 0, 0]])
```

9. Tfidf Vectorizer

With the help of Tfidf Vectorizer, it is easy to convert the text into the form of vector based on the word frequency and the inverse document frequency and get those values which could be fed to the machine learning models for prediction respectively. Having a look at those values, we are going to be taking them and predicting using the machine learning approach.

In [171…
```
vectorizer = TfidfVectorizer()
X_train_text = vectorizer.fit_transform(X_train['text'])
X_cv_text = vectorizer.transform(X_cv['text'])
```

In [172…
```
X_train_text.shape
```

Out[172…
```
(19236, 20619)
```

In [173…
```
X_train_text[0: 5]
```

Out[173…
```
<5x20619 sparse matrix of type '<class 'numpy.float64'>'
        with 31 stored elements in Compressed Sparse Row format>
```

10. Neural Network Model for Prediction

We are going to be defining the neural network model and predicting the chances of a person suffering from a heart disease. Taking a look at the data, we see that there are different activation units that we are going to be working with and then, we are going to use the categorical cross entropy as this is a multiclass classification problem. There can be more metrics that we might take but let us deal with accuracy for now and use the adam optimizer respectively.

In [174…
```
model = Sequential()
model.add(Dense(100, activation = 'relu', input_shape = (20619,)))
model.add(Dense(50, activation = 'relu'))
model.add(Dense(25, activation = 'relu'))
model.add(Dense(10, activation = 'relu'))
model.add(Dense(1, activation = 'sigmoid'))

model.compile(loss = 'categorical_crossentropy', metrics = ['accuracy']
```

In [175…
```
to_categorical(np.arange(1, 10))
```

Out[175…
```
array([[0., 1., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 1., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 1., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 1., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 1., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 1., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 1., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 1., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 1.]], dtype=float32)
```

In [176…
```
y_train
```

Out[176…
```
19832    negative
10340    negative
11044    positive
14088    negative
20333     neutral
           ...
15650    positive
22638    positive
```

```
10124    positive
5601     positive
14001    negative
Name: sentiment, Length: 19236, dtype: object
```

We are now going to be converting the output values in a series format so that we can give these values to the neural network models that we would be working respectively.

In [177…
```
encoder = LabelEncoder()
y_train_converted = encoder.fit_transform(y_train)
```

In [178…
```
y_train_converted
```

Out[178…
```
array([0, 0, 2, ..., 2, 2, 0])
```

In [179…
```
y_train_final = to_categorical(y_train_converted)
```

In [180…
```
X_train.head()
```

Out[180…

	text	text_length	text_words
19832	24 hours since dog put sleep rip old friend	78	18
10340	not feeling comfortable today	41	6
11044	hurray twin girls born beautiful may day	48	9
14088	bah h8 waking	19	5
20333	textmate crashed first time 3 months not bad a...	80	14

In [181…
```
X_train_dropped = X_train.drop(['text'], axis = 1)
```

In [182…
```
X_train.head()
```

Out[182…

	text	text_length	text_words
19832	24 hours since dog put sleep rip old friend	78	18
10340	not feeling comfortable today	41	6
11044	hurray twin girls born beautiful may day	48	9
14088	bah h8 waking	19	5
20333	textmate crashed first time 3 months not bad a...	80	14

In [183…
```
X_train['text'].head()
```

Out[183…
```
19832          24 hours since dog put sleep rip old friend
10340                        not feeling comfortable today
11044           hurray twin girls born beautiful may day
14088                                        bah h8 waking
20333    textmate crashed first time 3 months not bad a...
Name: text, dtype: object
```

In [184…
```
X_train_dropped = X_train.drop(['text'], axis = 1)
```

In [185…
```
X_train_dropped.head()
```

Out[185…

	text_length	text_words
19832	78	18
10340	41	6
11044	48	9
14088	19	5
20333	80	14

In [186…
```
X_cv.head()
```

Out[186…

	text	text_length	text_words
7214	mom happy present yayy	38	8
19840	no surprise probably know	67	11
19409	nope difference 5 omr c class anyone notice fl...	102	19
6600	smells badly garlic	25	5
7807	friday whole different meaning work saturday s...	78	14

In [187…
```
X_cv_dropped = X_cv.drop(['text'], axis = 1)
```

In [188…
```
X_cv_dropped.head()
```

Out[188…

	text_length	text_words
7214	38	8
19840	67	11
19409	102	19
6600	25	5
7807	78	14

We are going to be converting the values from 0 to 1 respectively. We would have to be converting those values so that it becomes easy to perform machine learning analysis and this ensures that there is no gradient shifting and having different range of weights when performing the machine learning analysis respectively.

In [189…
```
scaler = MinMaxScaler()
X_train_final = scaler.fit_transform(X_train_dropped)
X_cv_final = scaler.transform(X_cv_dropped)
```

In [190…

In [190…
```
X_train_final[0: 5]
```

Out[190…
```
array([[0.54347826, 0.5483871 ],
       [0.27536232, 0.16129032],
       [0.32608696, 0.25806452],
       [0.11594203, 0.12903226],
       [0.55797101, 0.41935484]])
```

In [191…
```
X_cv_final[0: 5]
```

Out[191…
```
array([[0.25362319, 0.22580645],
       [0.46376812, 0.32258065],
       [0.7173913 , 0.58064516],
       [0.15942029, 0.12903226],
       [0.54347826, 0.41935484]])
```

In [192…
```
encoder = LabelEncoder()
y_train_encoded = encoder.fit_transform(y_train)

encoder = LabelEncoder()
y_cv_encoded = encoder.fit_transform(y_cv)
```

In [193…
```
y_train_final = to_categorical(y_train_encoded)
y_cv_final = to_categorical(y_cv_encoded)
```

In [194…
```
y_train_final[0: 5]
```

Out[194…
```
array([[1., 0., 0.],
       [1., 0., 0.],
       [0., 0., 1.],
       [1., 0., 0.],
       [0., 1., 0.]], dtype=float32)
```

In [195…
```
y_cv_final[0: 5]
```

Out[195…
```
array([[0., 0., 1.],
       [0., 1., 0.],
       [0., 1., 0.],
       [1., 0., 0.],
       [0., 1., 0.]], dtype=float32)
```

In [196…
```
X_train_final[0: 5]
```

Out[196…
```
array([[0.54347826, 0.5483871 ],
       [0.27536232, 0.16129032],
       [0.32608696, 0.25806452],
       [0.11594203, 0.12903226],
       [0.55797101, 0.41935484]])
```

In [197…
```
X_train.head()
```

Out[197…

	text	text_length	text_words
19832	24 hours since dog put sleep rip old friend	78	18
10340	not feeling comfortable today	41	6

11044	hurray twin girls born beautiful may day	48	9
14088	bah h8 waking	19	5
20333	textmate crashed first time 3 months not bad a...	80	14

11. Count Vectorizer

Now we are going to be using the bag of words to understand the text and get a good knowledge about it. Since the data that is given to the machine learning models need to be in the form of vectors, it would be good to convert the text values into different vectors so that it becomes easy for the machine learning models to perform the computations respectively.

In [198…
```
vectorizer = CountVectorizer()
X_train_vectorized = vectorizer.fit_transform(X_train['text'])
X_cv_vectorized = vectorizer.transform(X_cv['text'])
```

In [199…
```
X_train_vectorized
```

Out[199…
```
<19236x20619 sparse matrix of type '<class 'numpy.int64'>'
        with 135844 stored elements in Compressed Sparse Row format>
```

In [200…
```
X_train_final[0: 5]
```

Out[200…
```
array([[0.54347826, 0.5483871 ],
       [0.27536232, 0.16129032],
       [0.32608696, 0.25806452],
       [0.11594203, 0.12903226],
       [0.55797101, 0.41935484]])
```

Since the values that we have are not in the form of arrays and in the form of sparse matrices, it would be a good idea to convert the values into the form of arrays so that it becomes easy for the machine learning models to make the predictions and ensure that they are getting the best predictions.

In [201…
```
X_train_bow_toarray = X_train_vectorized.toarray()
X_cv_bow_toarray = X_cv_vectorized.toarray()
```

We have formed the arrays that we would be using and now it is time to understand the matrices and concatenate them so that we are going to perform the machine learning analysis respectively. We are now going to be using the deep neural networks and this would ensure that we are getting the best results respectively.

In [202…
```
X_train_new = np.concatenate((X_train_bow_toarray, X_train_final), axis
X_cv_new = np.concatenate((X_cv_bow_toarray, X_cv_final), axis = 1)
```

12. Neutral Network Model

We would now be using the deep neural networks that we are going to be learning and

ensure that we are getting the best predictions respectively. We would start with 100 neutrons from the first layer and followed by 25 neutrons in the second layer and 10 units in the third layer followed by 3 final layers which we are going to be using the softmax classifier for predictions respectively.

In [203…

```
model = Sequential()
model.add(Dense(100, activation = 'relu'))
model.add(Dropout(0.3))
model.add(Dense(25, activation = 'relu'))
model.add(Dropout(0.5))
model.add(Dense(10, activation = 'relu'))
model.add(Dropout(0.3))
model.add(Dense(3, activation = 'softmax'))
model.compile(loss = 'categorical_crossentropy', metrics = ['accuracy']
```

We would be running the deep neutral network model for 10 epochs to ensure that we are going to be getting the best results in the test set respectively. We are also going to give the cross validation data and see how our model would be performing with the cross validation data, taking into account different parameters such as accuracy and cross validation loss respectively.

In [204…

```
model.fit(X_train_new, y_train_final, epochs = 10, validation_data = (X
```

```
Epoch 1/10
602/602 [==============================] - 4s 6ms/step - loss: 1.0098 -
accuracy: 0.4828 - val_loss: 0.7593 - val_accuracy: 0.6862
Epoch 2/10
602/602 [==============================] - 3s 4ms/step - loss: 0.6913 -
accuracy: 0.7126 - val_loss: 0.7227 - val_accuracy: 0.6972
Epoch 3/10
602/602 [==============================] - 3s 5ms/step - loss: 0.4995 -
accuracy: 0.8079 - val_loss: 0.7762 - val_accuracy: 0.6896
Epoch 4/10
602/602 [==============================] - 3s 4ms/step - loss: 0.3655 -
accuracy: 0.8646 - val_loss: 0.8712 - val_accuracy: 0.6855
Epoch 5/10
602/602 [==============================] - 3s 5ms/step - loss: 0.2798 -
accuracy: 0.8988 - val_loss: 1.0168 - val_accuracy: 0.6788
Epoch 6/10
602/602 [==============================] - 3s 4ms/step - loss: 0.2369 -
accuracy: 0.9169 - val_loss: 1.1245 - val_accuracy: 0.6769
Epoch 7/10
602/602 [==============================] - 3s 4ms/step - loss: 0.1777 -
accuracy: 0.9383 - val_loss: 1.2873 - val_accuracy: 0.6753
Epoch 8/10
602/602 [==============================] - 3s 4ms/step - loss: 0.1470 -
accuracy: 0.9479 - val_loss: 1.3884 - val_accuracy: 0.6758
Epoch 9/10
602/602 [==============================] - 3s 5ms/step - loss: 0.1332 -
accuracy: 0.9543 - val_loss: 1.5644 - val_accuracy: 0.6722
Epoch 10/10
602/602 [==============================] - 3s 5ms/step - loss: 0.1151 -
accuracy: 0.9610 - val_loss: 1.6192 - val_accuracy: 0.6718
```

Out[204… `<tensorflow.python.keras.callbacks.History at 0x7fe9a84b5cd0>`

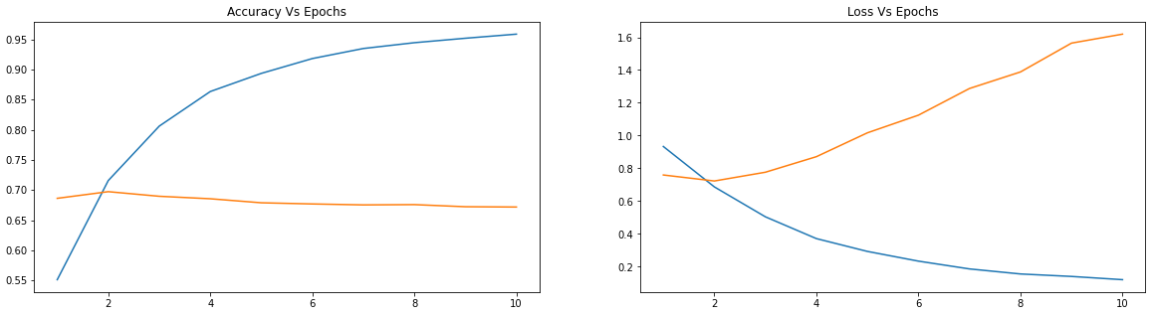
13. Plots of the Results

We would now be looking at the deep neural network plots and see how the values of accuracy and loss change with respect to the number of epochs that we are running. We see that as the number of epochs increase, there seems to be overfitting where the cross validation loss tends to go higher though there is a decrease in the training loss respectively. This gives us a clear indication that the model is overfitting after certain number of epochs are met. Therefore, we would be taking the right number of epochs when we are performing the machine learning analysis.

In [205…

```
accuracy = model.history.history['accuracy']
val_accuracy = model.history.history['val_accuracy']
loss = model.history.history['loss']
val_loss = model.history.history['val_loss']
epochs = np.arange(1, 11)
fig, ax = plt.subplots(1, 2, figsize = (20, 5))

sns.lineplot(x = epochs, y = accuracy, ax = ax[0])
sns.lineplot(x = epochs, y = val_accuracy, ax = ax[0])
ax[0].set_title('Accuracy Vs Epochs')
sns.lineplot(x = epochs, y = loss, ax = ax[1])
sns.lineplot(x = epochs, y = val_loss, ax = ax[1])
ax[1].set_title('Loss Vs Epochs')
plt.show()
```



14. Conclusions

1. *It would be a good idea to use some tools such as wordcloud when we are doing Natural Language Processing (NLP) to ensure that we are getting the best results for predictions respectively. We would be able to understand the frequently occurring words from the less frequently occurring words by the size of the words that are plotted in the wordcloud respectively.*
2. *Steps should be taken to ensure that the model does not overfit or underfit. This ensures that the best predictions are being generated and therefore, we are going to get the best outputs respectively.*
3. *Standarizing the text and ensuring that the values lie between 0 and 1 would be good as this would allow the machine learning models to generate weights that are quite small rather than having different weight range values.*