In [108]:

```
import pandas as pd
import numpy as np
from bs4 import BeautifulSoup as bs
import requests

import string
from collections import Counter
import seaborn as sns
import matplotlib.pyplot as plt
from nltk.corpus import stopwords
from nltk.sentiment.vader import SentimentIntensityAnalyzer
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize
import nltk
import warnings
warnings.filterwarnings('ignore')

#!wget http://nlp.stanford.edu/data/glove.6B.zip
#!unzip glove.6B.zip
```

In [73]:

```
!wget "https://s3.amazonaws.com/dl4j-distribution/GoogleNews-vectors-negative300.bin.gz"
```

```
--2022-03-30 18:31:58--  https://s3.amazonaws.com/dl4j-distribution/GoogleNews-vectors-negative300.bin.gz (htt
ps://s3.amazonaws.com/dl4j-distribution/GoogleNews-vectors-negative300.bin.gz)
Resolving s3.amazonaws.com (s3.amazonaws.com)... 52.216.184.101
Connecting to s3.amazonaws.com (s3.amazonaws.com)|52.216.184.101|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1647046227 (1.5G) [application/x-gzip]
Saving to: ‘GoogleNews-vectors-negative300.bin.gz’

GoogleNews-vectors- 100%[===================>]   1.53G  45.5MB/s    in 37s

2022-03-30 18:32:36 (42.3 MB/s) - ‘GoogleNews-vectors-negative300.bin.gz’ saved [1647046227/1647046227]
```

In [109]:

```
import tensorflow as tf
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np


import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
from nltk.stem import SnowballStemmer



from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder


import re
from keras.preprocessing.sequence import pad_sequences
from keras.models import Sequential
from keras.layers import Dense, Embedding, LSTM, SpatialDropout1D
from sklearn.model_selection import train_test_split
from keras.utils.np_utils import to_categorical
from keras.callbacks import EarlyStopping
from keras.layers import Dropout
import re
from nltk.corpus import stopwords
from nltk import word_tokenize
STOPWORDS = set(stopwords.words('english'))
from bs4 import BeautifulSoup
import plotly.graph_objs as go

#import cufflinks
from IPython.core.interactiveshell import InteractiveShell
import plotly.figure_factory as ff
InteractiveShell.ast_node_interactivity = 'all'

print("Tensorflow Version",tf.__version__)
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Out[109]:

```
True

Tensorflow Version 2.8.0
```

In [110]:

```
model_data = pd.read_csv("/content/model_data.csv")
model_data.head()
```

Out[110]:

	title	summary	anger	anticipation	disgust	fear	joy	negative	positive	sadness	surprise	trust	sentiment_class
0	The Hunger Games	Could you survive on your own in the wild, wit...	8	8	10	9	6	16	11	11	5	5	Negative
1	Harry Potter and the Order of the Phoenix	There is a door at the end of a silent corrido...	14	7	5	15	6	18	18	13	5	10	Neutral
2	To Kill a Mockingbird	The unforgettable novel of a childhood in a sl...	2	4	2	5	8	8	15	5	1	2	Positive
3	Pride and Prejudice	Alternate cover edition of ISBN 9780679783268S...	5	15	2	3	22	8	26	0	3	15	Positive
4	The Book Thief	Librarian's note: An alternate cover edition c...	3	5	3	7	3	9	11	4	4	11	Positive

In [29]:

```
model_data.describe()
```

Out[29]:

	anger	anticipation	disgust	fear	joy	negative	positive	sadness	surprise	tru
count	1084.000000	1084.000000	1084.000000	1084.000000	1084.000000	1084.000000	1084.000000	1084.000000	1084.000000	1084.00000
mean	4.802583	7.273063	3.440037	7.455720	6.680812	10.226937	15.158672	5.087638	3.691882	8.29612
std	4.180568	4.835920	3.160102	5.472577	4.949567	6.611929	8.652238	4.011373	2.938126	5.39637
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000
25%	2.000000	4.000000	1.000000	3.000000	3.000000	5.000000	9.000000	2.000000	2.000000	4.00000
50%	4.000000	7.000000	3.000000	6.000000	6.000000	9.000000	14.000000	4.000000	3.000000	8.00000
75%	7.000000	10.000000	5.000000	11.000000	9.000000	14.000000	20.000000	7.000000	5.000000	11.00000
max	25.000000	46.000000	21.000000	31.000000	35.000000	39.000000	86.000000	25.000000	20.000000	58.00000

In [111]:

```
from sklearn.metrics import classification_report
from sklearn.preprocessing import LabelEncoder
df_summary = model_data[["title","summary", "sentiment_class"]]
```

Multiclass text classification

In [112]:

```
import logging
import pandas as pd
import numpy as np
from numpy import random
import gensim
import nltk
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.metrics import accuracy_score, confusion_matrix
import matplotlib.pyplot as plt
from nltk.corpus import stopwords
import re
from bs4 import BeautifulSoup
%matplotlib inline
```

In [49]:
```
from xgboost import XGBClassifier as XGBC
from sklearn.neighbors import KNeighborsClassifier as knn
from sklearn.naive_bayes import GaussianNB as GB
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier,GradientBoostingClassifier, VotingClassi
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from xgboost import XGBClassifier as XGBC

from matplotlib.gridspec import GridSpec
from sklearn.metrics import accuracy_score, confusion_matrix, recall_score, f1_score, precision_score, classific
def Classification_model(model,X_train_res,y_train_res,X_test,y_test): # here x is the variable which are used f


    model.fit(X_train_res,y_train_res.ravel())
    pred=model.predict(X_test)
    accuracy=accuracy_score(y_test,pred)
    recall = recall_score(y_test,pred, average='micro')
    precision = precision_score(y_test,pred, average='micro')
    F1_score = f1_score(y_test,pred, average='micro')
    return accuracy, recall, precision, F1_score
```

In [50]:
```
# Lets us make a list of models
models=["RandomForestClassifier","Gaussian Naive Bays","KNN","Logistic_Regression","Support_Vector"]
Classification_models = [RandomForestClassifier(n_estimators=100),GB(),knn(n_neighbors=7),LogisticRegression(),S
Model_Accuracy = []
Model_Recall = []
Model_Precision = []
Model_f1 = []
```

In [51]:
```
for model in Classification_models:
    Accuracy,Recall, Precision, F1_score = Classification_model(model,x_train,y_train,x_test,y_test)    #,Recall
    Model_Accuracy.append(Accuracy)
    Model_Recall.append(Recall)
    Model_Precision.append(Precision)
    Model_f1.append(F1_score)
```

In [52]:
```
Accuracy_with_Imp_features = pd.DataFrame(
    { "Classification Model" :models,
     "Accuracy with Imp features":Model_Accuracy, "Recall with Imp features":Model_Recall, "Precision with Imp f
     "F1 with Imp features":Model_f1})
```

In [53]:
```
Accuracy_with_Imp_features.sort_values(by="Accuracy with Imp features",ascending=False).reset_index(drop=True)
```

Out[53]:

	Classification Model	Accuracy with Imp features	Recall with Imp features	Precision with Imp features	F1 with Imp features
0	Logistic_Regression	0.690184	0.690184	0.690184	0.690184
1	RandomForestClassifier	0.662577	0.662577	0.662577	0.662577
2	Gaussian Naive Bays	0.656442	0.656442	0.656442	0.656442
3	Support_Vector	0.653374	0.653374	0.653374	0.653374
4	KNN	0.622699	0.622699	0.622699	0.622699

Naive Bayes Classifier for Multinomial Models

In [114]:

```
from sklearn.naive_bayes import MultinomialNB
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import TfidfTransformer
X = df_summary.summary
y = df_summary.sentiment_class
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state = 42)



nb = Pipeline([('vect', CountVectorizer()),
               ('tfidf', TfidfTransformer()),
               ('clf', MultinomialNB()),
              ])
nb.fit(X_train, y_train)


from sklearn.metrics import classification_report
y_pred = nb.predict(X_test)

print('accuracy %s' % accuracy_score(y_pred, y_test))
print(classification_report(y_test, y_pred,target_names=["Positive", "Negative", "Neutral"]))
```

Out[114]:

```
Pipeline(steps=[('vect', CountVectorizer()), ('tfidf', TfidfTransformer()),
                ('clf', MultinomialNB())])

accuracy 0.6748466257668712
              precision    recall  f1-score   support

    Positive       0.00      0.00      0.00        96
    Negative       0.00      0.00      0.00        10
     Neutral       0.67      1.00      0.81       220

    accuracy                           0.67       326
   macro avg       0.22      0.33      0.27       326
weighted avg       0.46      0.67      0.54       326
```

In [115]:

```
my_tags = ["Positive", "Negative", "Neutral"]
```

Linear Support Vector Machine

In [116]:

```

from sklearn.linear_model import SGDClassifier

sgd = Pipeline([('vect', CountVectorizer()),
                ('tfidf', TfidfTransformer()),
                ('clf', SGDClassifier(loss='hinge', penalty='l2',alpha=1e-3, random_state=42, max_iter=5, tol=No
               ])
sgd.fit(X_train, y_train)


y_pred = sgd.predict(X_test)

print('accuracy %s' % accuracy_score(y_pred, y_test))
print(classification_report(y_test, y_pred,target_names=my_tags))
```

Out[116]:

```
Pipeline(steps=[('vect', CountVectorizer()), ('tfidf', TfidfTransformer()),
                ('clf',
                 SGDClassifier(alpha=0.001, max_iter=5, random_state=42,
                               tol=None))])

accuracy 0.7269938650306749
              precision    recall  f1-score   support

    Positive       0.68      0.29      0.41        96
    Negative       0.00      0.00      0.00        10
     Neutral       0.73      0.95      0.83       220

    accuracy                           0.73       326
   macro avg       0.47      0.41      0.41       326
weighted avg       0.70      0.73      0.68       326
```

Logistic Regression

In [117]:

```
from sklearn.linear_model import LogisticRegression

logreg = Pipeline([('vect', CountVectorizer()),
                ('tfidf', TfidfTransformer()),
                ('clf', LogisticRegression(n_jobs=1, C=1e5)),
               ])
logreg.fit(X_train, y_train)



y_pred = logreg.predict(X_test)

print('accuracy %s' % accuracy_score(y_pred, y_test))
print(classification_report(y_test, y_pred,target_names=my_tags))
```

Out[117]:

```
Pipeline(steps=[('vect', CountVectorizer()), ('tfidf', TfidfTransformer()),
                ('clf', LogisticRegression(C=100000.0, n_jobs=1))])

accuracy 0.7300613496932515
              precision    recall  f1-score   support

    Positive       0.67      0.31      0.43        96
    Negative       0.00      0.00      0.00        10
     Neutral       0.74      0.95      0.83       220

    accuracy                           0.73       326
   macro avg       0.47      0.42      0.42       326
weighted avg       0.70      0.73      0.69       326
```

Deep learning

In [10]:

```
import itertools
import os

%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import tensorflow as tf
from tensorflow.keras.utils import to_categorical

from sklearn.preprocessing import LabelBinarizer, LabelEncoder
from sklearn.metrics import confusion_matrix

from tensorflow import keras
from keras.models import Sequential
from keras.layers import Dense, Activation, Dropout
from keras.preprocessing import text, sequence
from keras import utils

train_size = int(len(df_summary) * .7)
train_posts = df_summary['summary'][:train_size]
train_tags = df_summary['sentiment_class'][:train_size]

test_posts = df_summary['summary'][train_size:]
test_tags = df_summary['sentiment_class'][train_size:]

max_words = 15000
tokenize = text.Tokenizer(num_words=max_words, char_level=False)
tokenize.fit_on_texts(train_posts) # only fit on train

x_train = tokenize.texts_to_matrix(train_posts)
x_test = tokenize.texts_to_matrix(test_posts)

encoder = LabelEncoder()
encoder.fit(train_tags)
y_train = encoder.transform(train_tags)
y_test = encoder.transform(test_tags)
```

Out[10]:

```
LabelEncoder()
```

In [11]:

```
num_classes = np.max(y_train) + 1
num_classes
```

Out[11]:

```
3
```

In [12]:

```
y_train = to_categorical(y_train, num_classes)
y_test = to_categorical(y_test, num_classes)

batch_size = 32
```

In [14]:

```
# Build the model
model = Sequential()
model.add(Dense(128, input_shape=(max_words,)))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes))
model.add(Activation('sigmoid'))

model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])

history = model.fit(x_train, y_train,
                    batch_size=batch_size,
                    epochs=11,
                    verbose=1)#,validation_split=0.1)
```

```
Epoch 1/11
24/24 [==============================] - 2s 27ms/step - loss: 0.8562 - accuracy: 0.6478
Epoch 2/11
24/24 [==============================] - 1s 30ms/step - loss: 0.6575 - accuracy: 0.7190
Epoch 3/11
24/24 [==============================] - 1s 30ms/step - loss: 0.5022 - accuracy: 0.7770
Epoch 4/11
24/24 [==============================] - 1s 30ms/step - loss: 0.3162 - accuracy: 0.8852
Epoch 5/11
24/24 [==============================] - 1s 28ms/step - loss: 0.1742 - accuracy: 0.9433
Epoch 6/11
24/24 [==============================] - 1s 25ms/step - loss: 0.1065 - accuracy: 0.9617
Epoch 7/11
24/24 [==============================] - 1s 26ms/step - loss: 0.0706 - accuracy: 0.9789
Epoch 8/11
24/24 [==============================] - 1s 24ms/step - loss: 0.0469 - accuracy: 0.9921
Epoch 9/11
24/24 [==============================] - 1s 27ms/step - loss: 0.0336 - accuracy: 0.9947
Epoch 10/11
24/24 [==============================] - 1s 27ms/step - loss: 0.0241 - accuracy: 0.9934
Epoch 11/11
24/24 [==============================] - 1s 24ms/step - loss: 0.0160 - accuracy: 1.0000
```

In [100]:

```
score = model.evaluate(x_test, y_test,
                       batch_size=batch_size, verbose=1)
print('Test accuracy:', score[1])
```

```
11/11 [==============================] - 1s 9ms/step - loss: 1.0158 - accuracy: 0.7717
Test accuracy: 0.7716564512252808
```

Sentiment Classification (Prediction)

In [101]:

```
test_text = df_summary[1:2].summary.values
test_text[0]
```

Out[101]:

```
'There is a door at the end of a silent corridor. And itâ\x80\x99s haunting Harry Pottterâ\x80\x99s dreams. Wh
y else would he be waking in the middle of the night, screaming in terror?Harry has a lot on his mind for thi
s, his fifth year at Hogwarts: a Defense Against the Dark Arts teacher with a personality like poisoned honey;
a big surprise on the Gryffindor Quidditch team; and the loomiThere is a door at the end of a silent corridor.
And itâ\x80\x99s haunting Harry Pottterâ\x80\x99s dreams. Why else would he be waking in the middle of the nig
ht, screaming in terror?Harry has a lot on his mind for this, his fifth year at Hogwarts: a Defense Against th
e Dark Arts teacher with a personality like poisoned honey; a big surprise on the Gryffindor Quidditch team; a
nd the looming terror of the Ordinary Wizarding Level exams. But all these things pale next to the growing thr
eat of He-Who-Must-Not-Be-Named - a threat that neither the magical government nor the authorities at Hogwarts
can stop.As the grasp of darkness tightens, Harry must discover the true depth and strength of his friends, th
e importance of boundless loyalty, and the shocking price of unbearable sacrifice.His fate depends on them al
l.'
```

Saving the Model

In [121]:

```
model.save("model")
```

```
INFO:tensorflow:Assets written to: model/assets
```

In [125]:

```
def toknize_input(test_text):
  tokenize.fit_on_texts(test_text) # only fit on train
  x_train2 = tokenize.texts_to_matrix(test_text)
  return x_train2

def decode_sentiment(num_):
  if num_ == 0:
    return "Negative"
  elif num_==1:
    return "Neutral"
  else:
    return "Positive"

model_dl = keras.models.load_model('model')
```

In [126]:

```
tokenized_text = toknize_input(test_text)
scores = model_dl.predict(tokenized_text, verbose=1, batch_size=32)
y_pred = decode_sentiment(np.where(scores[0] == max(scores[0]))[0][0])
y_pred
```

```
1/1 [==============================] - 0s 65ms/step
```

Out[126]:

```
'Positive'
```