Python/Deep Learning Project Report

Classification of News into Categories Based on Headlines & Short Description

Team ID: 2

Member 1: Akhil Teja Kanugolu **Class ID: 11**

Member 2: Geetanjali Makineni **Class ID:13**

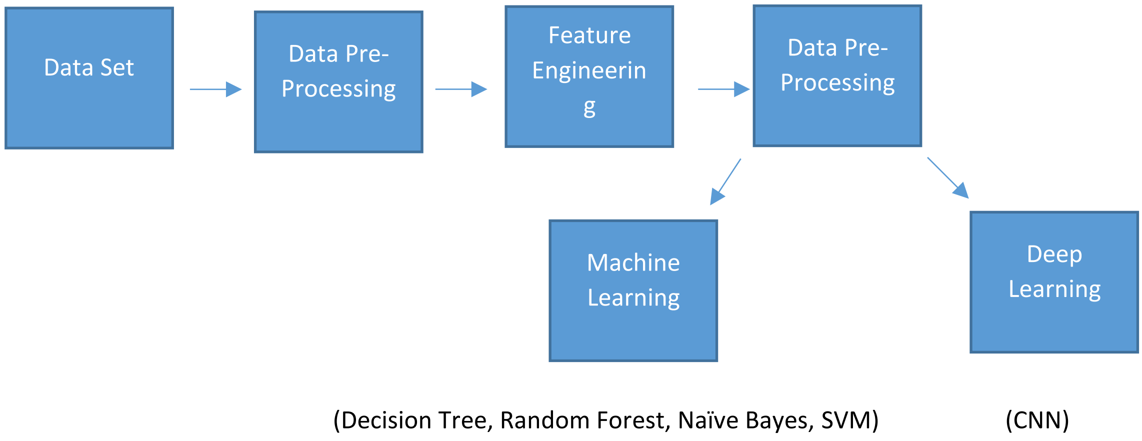
Github Link:

https://github.com/geetamakineni/PYTHON-PROJECT

Dataset link:

https://www.kaggle.com/rmisra/news-category-dataset

Overall Architecture:



Work:

Basically, we divided the whole project into 3 components as

- Dataset Preparation
- Feature Engineering
- Model Training

1) Dataset Preparation: The first step here is dataset preparation where we load the dataset and perform the basic preprocessing. The dataset is further split into training, validation, and test sets.

2) Feature Engineering – In this step, the raw dataset is transformed further into flat features. This mainly includes process in which we create new features from the existing features. We use the Count Vector matrix notation where we check variance and drop some of the features based on the threshold it handles.

3) Model Training – The final most step we use is the Model Building where a machine learning model is here trained on the dataset. We implement the models like Naïve Bayes Classifier, Convolution Neural Network and Decision Tree Model and find which model gives best accuracy.

Process:

Initially, we imported the required packages and loaded the dataset as below. And we also concatenated the headline as well as the short description into the single attribute named 'Combined_H&SD' for gaining more data to categorize the category of news type. And performed custom function process_data() where we applied stemmer which helped to remove duplicate words & removed the stop words, special characters. Based on the Data with 41 News categories using SMOTE function we splitted the Data with equal for increasing the efficiency. Thereafter, we classified the Data into Train, Development, Test Data: Train data is used for the training different Model, Development Data for tuning the Parameters, Test Data used for validating the different models trained. And visualized the Words which was processed with Word Cloud.

In the Data preprocessing, we did vectorization using BOW (Bag of Words). Later, we processed the data with tokenizer from nltk corpus library. This is classification model problem as the predicted value belongs to category. So, we use one hot label encoder to mask the predicted category.

Next, we will use feature Reduction based on the Variance Threshold=0.001 which removes the data that will not have the impact on the prediction. Data sampling was done because the categories are unequally distributed which may overfit or underfit some categories with more data or less data. This will help to train the data equally for every category using the SMOTE function.

Moving Next, Training of model using Machine Learning Models:

1. Decision Tree Model: Created the Decision Tree Model with dtc_model and trained the model and got the accuracy 31% and viewed the F1 score with help of classification report

2. Random Forest Model: Created the Random Forest Model with rf_model and trained the model and got the accuracy 34% and viewed the F1 score with help of classification report

3. Multinomial Naïve Bayes Classification: Created the Multinomial Naïve Bayes Model with nb_model and trained the model and got the accuracy 52% and viewed the F1 score with help of classification report.

   By Comparing other models, we got better accuracy for the Development Data. So, we predicted the data for test data & which resulted the accuracy of 54%. And when we process the data as we used Bag of words to vectorize it will not give the order of words, so we created custom function reverse vocabulary. Finally appended the words for each category helps for the prediction based on whole training data.

4. Support Vector Classification: Created the Decision Tree Model with svc_model and trained the model and got the accuracy 54% and viewed the F1 score with help of classification report.

As we got accuracy more than Multinomial Naïve Bayes, we performed same steps as Multinomial Naïve Bayes Model on test Data such as Reverse vocabulary & viewed the words for the prediction.

Using Deep learning Model:

Here we are trying to perform Convolution Neural Network. After importing the packages required for the Analysing the Dataset, we will read the JSON file and store to df. Apart from Machine learning model we tried different techniques to pre-process the Data like started with viewing the categories by using the group by. Then removing the empty data& short and later combined the headline and short description with the space. Calculated the max length of words for padding the Data. Later the category variable converted into ID. Thereafter glove embedding to remove the duplicates from getting the Stanford library words using inbuilt function. And splitted the Data into training and Test data.

Step 1:

For this we started with importing the packages:

```python
import pandas as pd
import numpy as np
import json
import copy
import string
import re
import nltk
import string
from nltk.stem import PorterStemmer
from nltk.corpus import stopwords
nltk.download('popular')
from wordcloud import WordCloud

from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.preprocessing import LabelEncoder
from sklearn.feature_selection import VarianceThreshold
from imblearn.over_sampling import SMOTE

from sklearn.tree import DecisionTreeClassifier

import matplotlib.pyplot as plt
np.random.seed(0)
```

+ Code + Text Connect Ec

```python
import matplotlib.pyplot as plt
np.random.seed(0)
```

```
[nltk_data] Downloading collection 'popular'
[nltk_data]    |
[nltk_data]    | Downloading package cmudict to /root/nltk_data...
[nltk_data]    |   Package cmudict is already up-to-date!
[nltk_data]    | Downloading package gazetteers to /root/nltk_data...
[nltk_data]    |   Package gazetteers is already up-to-date!
[nltk_data]    | Downloading package genesis to /root/nltk_data...
[nltk_data]    |   Package genesis is already up-to-date!
[nltk_data]    | Downloading package gutenberg to /root/nltk_data...
[nltk_data]    |   Package gutenberg is already up-to-date!
[nltk_data]    | Downloading package inaugural to /root/nltk_data...
[nltk_data]    |   Package inaugural is already up-to-date!
[nltk_data]    | Downloading package movie_reviews to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Package movie_reviews is already up-to-date!
[nltk_data]    | Downloading package names to /root/nltk_data...
[nltk_data]    |   Package names is already up-to-date!
[nltk_data]    | Downloading package shakespeare to /root/nltk_data...
[nltk_data]    |   Package shakespeare is already up-to-date!
[nltk_data]    | Downloading package stopwords to /root/nltk_data...
[nltk_data]    |   Package stopwords is already up-to-date!
[nltk_data]    | Downloading package treebank to /root/nltk_data...
[nltk_data]    |   Package treebank is already up-to-date!
[nltk_data]    | Downloading package twitter_samples to
[nltk_data]    |     /root/nltk_data...
```

Step 2:

Here, we are reading the news dataset from the json file we have and viewing the sample Data with head function

Reading News Dataset from News_Category_Dataset_v2.json

```
[ ] News_Dataset = pd.read_json('News_Category_Dataset_v2.json', lines=True)
```

```
[ ] News_Dataset.head(6)
```

	category	headline	authors	link	short_description	date
0	CRIME	There Were 2 Mass Shootings In Texas Last Week...	Melissa Jeltsen	https://www.huffingtonpost.com/entry/texas-ama...	She left her husband. He killed their children...	2018-05-26
1	ENTERTAINMENT	Will Smith Joins Diplo And Nicky Jam For The 2...	Andy McDonald	https://www.huffingtonpost.com/entry/will-smit...	Of course it has a song.	2018-05-26
2	ENTERTAINMENT	Hugh Grant Marries For The First Time At Age 57	Ron Dicker	https://www.huffingtonpost.com/entry/hugh-gran...	The actor and his longtime girlfriend Anna Ebe...	2018-05-26
3	ENTERTAINMENT	Jim Carrey Blasts 'Castrato' Adam Schiff And D...	Ron Dicker	https://www.huffingtonpost.com/entry/jim-carre...	The actor gives Dems an ass-kicking for not fi...	2018-05-26
4	ENTERTAINMENT	Julianna Margulies Uses Donald Trump Poop Bags...	Ron Dicker	https://www.huffingtonpost.com/entry/julianna-...	The "Dietland" actress said using the bags is ...	2018-05-26
5	ENTERTAINMENT	Morgan Freeman 'Devastated' That Sexual Harass...	Ron Dicker	https://www.huffingtonpost.com/entry/morgan-fr...	"It is not right to equate horrific	2018-05-26

Step 3:

Combining of column's headline's and short description into a single attribute which helps to get the sufficient data for prediction of Category.And the Combined headline is cleaned using stemmer and process_text() function

That Sexual Harass... Dicker fr... incidents ...

Combining column's Headline & Short Description into Combined_H&SD

```
[ ] News_Dataset['Combined_H&SD']=News_Dataset['headline']+News_Dataset['short_description']
```

```
[ ] stemmer = PorterStemmer()
```

```
[ ] def process_text(value):
        no_punc=[char for char in value if char not in string.punctuation]
        new1=''.join(no_punc)
        new2=[stemmer.stem(word) for word in new1]
        new3=''.join(new2)
        return[word for word in new3.split()if word.lower()not in stopwords.words('english') ]
```

```
[ ] News_Dataset['Combined_H&SD'].head()
```

```
0    There Were 2 Mass Shootings In Texas Last Week...
1    Will Smith Joins Diplo And Nicky Jam For The 2...
2    Hugh Grant Marries For The First Time At Age 5...
3    Jim Carrey Blasts 'Castrato' Adam Schiff And D...
4    Julianna Margulies Uses Donald Trump Poop Bags...
Name: Combined_H&SD, dtype: object
```

Step 4:

Splitting of the Dataset into train, test and development.

Training data is used for training out the model and Development data for tuning and checking the hyper parameters and test data to check how the model is performing.

After Processing of column- Combined_H&SD using Stemmer

```
[ ] News_Dataset['Combined_H&SD'].head(5).apply(process_text)
```

```
0    [2, Mass, Shootings, Texas, Last, Week, 1, TVS...
1    [Smith, Joins, Diplo, Nicky, Jam, 2018, World,...
2    [Hugh, Grant, Marries, First, Time, Age, 57The...
3    [Jim, Carrey, Blasts, Castrato, Adam, Schiff, ...
4    [Julianna, Margulies, Uses, Donald, Trump, Poo...
Name: Combined_H&SD, dtype: object
```

Splitting of News Data set Into Train, Test & Development

```
[ ] train_title, test_title, train_category, test_category = train_test_split(News_Dataset['Combined_H&SD'],News_Dataset['cate
    train_title, devp_title, train_category, devp_category = train_test_split(train_title,train_category)
```

Number of Records for train, Test & Development

```
[ ] print("Training Records    : ",len(train_title))
    print("Developement Records: ",len(devp_title))
    print("Testing Records     : ",len(test_title))
```

```
Training Records    :  112979
Developement Records:  37660
Testing Records     :  50214
```

Step 5:

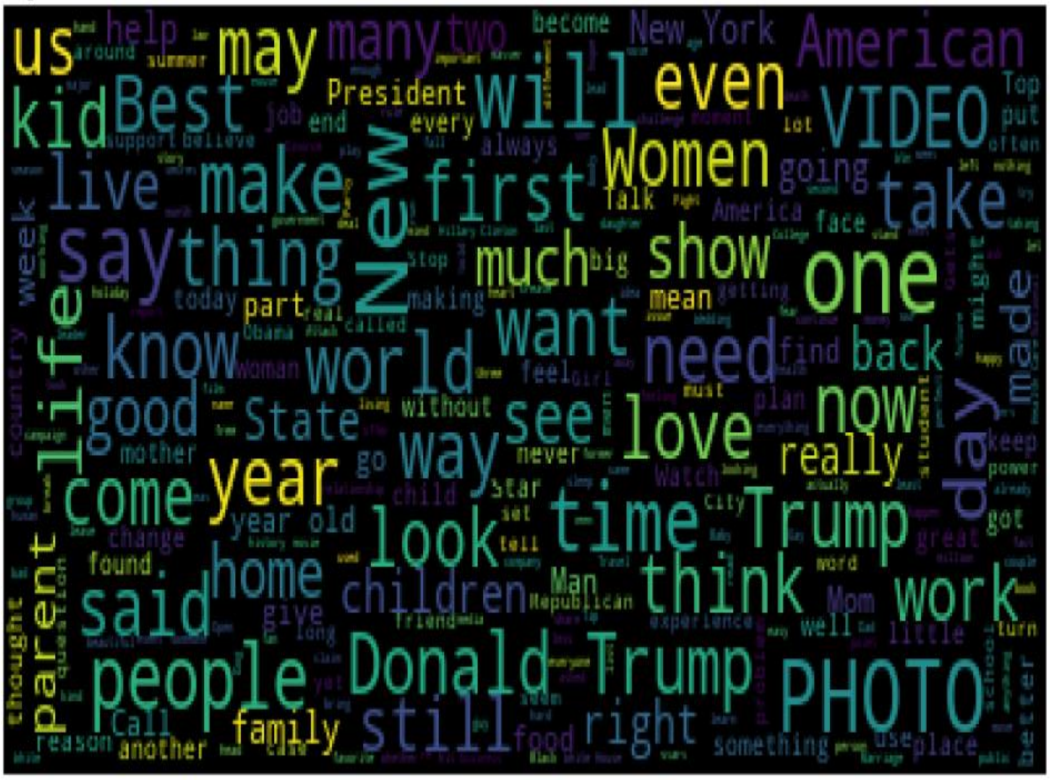
Visualized the data combined using WordCloud which gives the unique words of training Data which helps for the Prediction category

Visualized the Data of combined_H&SD using WordCloud

```
[ ] train_text = " ".join(train_title)
    wordcloud = WordCloud().generate(train_text)
    plt.figure()
    plt.subplots(figsize=(50,50))
    wordcloud = WordCloud(
        background_color="Black",
        max_words=len(train_text),
        max_font_size=30,
        relative_scaling=.5).generate(train_text)
    plt.imshow(wordcloud)
    plt.axis("off")
    plt.show()
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```



Step 6:

The next step is Data Preprocessing. Here, we are vectorizing the data using a Bag of Words (BOW) and we performs tokenizer on the train, test, Development.

Data Preprocessing

Vectorizing the data using Bag of words (BOW)

```
tokenizer = nltk.tokenize.RegexpTokenizer(r"\w+")
stop_words = nltk.corpus.stopwords.words("english")
c_vectorizer = CountVectorizer(tokenizer=tokenizer.tokenize, stop_words=stop_words)
```

```
c_vectorizer.fit(iter(train_title))
X_train = c_vectorizer.transform(iter(train_title))
X_devp = c_vectorizer.transform(iter(devp_title))
X_test = c_vectorizer.transform(iter(test_title))
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/feature_extraction/text.py:507: UserWarning: The parameter 'token_pattern'
  warnings.warn("The parameter 'token_pattern' will not be used"
```

Step 7:

Encoding the column categories are done using the label encoder for all the categories. After that the features are reduced.

If we clearly see the features before reduction are 126219 and after reduction are 3380 The threshold which we took is 0.001

Categorical Encoding of category Column using Label Encoder

```
[ ] encoder = LabelEncoder()
    encoder.fit(train_category)
    Y_train = encoder.transform(train_category)
    Y_devp = encoder.transform(devp_category)
    Y_test = encoder.transform(test_category)
```

Feature Reduction

```
[ ] print("Number of features before reduction : ", X_train.shape[1])
    selection = VarianceThreshold(threshold=0.001)
    X_train_whole = copy.deepcopy(X_train)
    Y_train_whole = copy.deepcopy(Y_train)
    selection.fit(X_train)
    X_train = selection.transform(X_train)
    X_devp = selection.transform(X_devp)
    X_test = selection.transform(X_test)
    print("Number of features after reduction : ", X_train.shape[1])
```

```
Number of features before reduction :  126219
Number of features after reduction :  3380
```

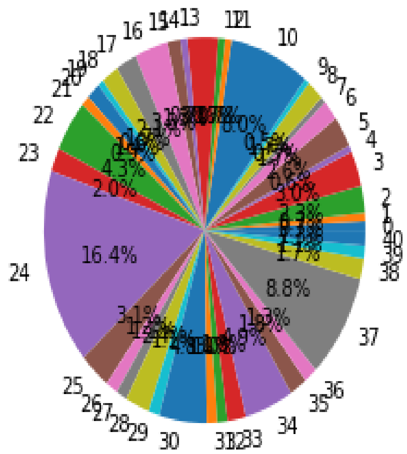
Step 8:

In Data Sampling,

We have counted the number of total labels and plotted them using a pie chart distribution model.

Sampling the data

```
[ ] labels = list(set(Ytr))
    counts = []
    for label in labels:
        counts.append(np.count_nonzero(Y_train == label))
    plt.pie(counts, labels=labels, autopct='%1.1f%%')
    plt.show()
```



Step 9:

We can clearly have a look that the class labels are here not distributed uniformly.

So, we had to use SMOT and then over sampled the classes which are lowest in the number. This is done because we can samples can be equally distributed helps for efficient prediction of category.

Step 10:

Model Training:

The following are the models we used to train our data:

Here we performed all Machine learn models Decision Tree, Random forest, SVC, Multinomial Naïve Bayes comparing all these accuracies we got High for SVC with 54 % and Multinomial Naïve Bayes with 52 %. So, we can select this two:

Decision Tree:

In decision tree model for every we can observe the every category the precision was less so we can say the accuracy was low with 31%.

Decision Tree Model

```
[ ] dtc_model = DecisionTreeClassifier()
    dtc_model.fit(X_train, Y_train)
    dtc_pred = dtc_model.predict(X_devp)
    print(classification_report(Y_devp, dtc_pred, target_names=encoder.classes_))
```

	precision	recall	f1-score	support
ARTS	0.08	0.11	0.09	297
ARTS & CULTURE	0.06	0.15	0.08	225
BLACK VOICES	0.23	0.22	0.23	803
BUSINESS	0.19	0.16	0.18	1097
COLLEGE	0.16	0.32	0.21	226
COMEDY	0.28	0.27	0.28	961
CRIME	0.22	0.34	0.26	647
CULTURE & ARTS	0.14	0.24	0.17	186
DIVORCE	0.35	0.52	0.42	640
EDUCATION	0.13	0.22	0.16	196
ENTERTAINMENT	0.39	0.23	0.29	3023
ENVIRONMENT	0.10	0.29	0.15	259
FIFTY	0.04	0.10	0.06	258
FOOD & DRINK	0.39	0.39	0.39	1182
GOOD NEWS	0.04	0.05	0.05	263
GREEN	0.15	0.16	0.16	506
HEALTHY LIVING	0.16	0.16	0.16	1252
HOME & LIVING	0.36	0.50	0.42	795

GOOD NEWS	0.04	0.05	0.05	263
GREEN	0.15	0.16	0.16	506
HEALTHY LIVING	0.16	0.16	0.16	1252
HOME & LIVING	0.36	0.50	0.42	795
IMPACT	0.08	0.12	0.10	621
LATINO VOICES	0.08	0.17	0.11	204
MEDIA	0.17	0.28	0.22	505
MONEY	0.11	0.22	0.15	326
PARENTING	0.32	0.30	0.31	1654
PARENTS	0.15	0.20	0.17	764
POLITICS	0.66	0.40	0.50	6066
QUEER VOICES	0.52	0.48	0.50	1200
RELIGION	0.26	0.26	0.26	506
SCIENCE	0.18	0.25	0.21	426
SPORTS	0.36	0.33	0.34	911
STYLE	0.16	0.20	0.18	409
STYLE & BEAUTY	0.60	0.54	0.57	1841
TASTE	0.13	0.17	0.15	381
TECH	0.18	0.24	0.21	410
THE WORLDPOST	0.17	0.25	0.20	676
TRAVEL	0.40	0.31	0.35	1817
WEDDINGS	0.50	0.62	0.55	709
WEIRD NEWS	0.09	0.09	0.09	476
WELLNESS	0.41	0.29	0.34	3374
WOMEN	0.16	0.17	0.17	653
WORLD NEWS	0.10	0.18	0.12	406
WORLDPOST	0.17	0.19	0.18	509
accuracy			0.31	37660
macro avg	0.23	0.26	0.24	37660
weighted avg	0.36	0.31	0.32	37660

Random Forest Model:

In Random forest model for every we can observe the every category the precision was less so we can say the accuracy was low with 39% so we reject this model.

Random Forest Model

```
rf_model = RandomForestClassifier(n_estimators=40)
rf_model.fit(X_train, Y_train)
rf_pred = rf_model.predict(X_devp)
print(classification_report(Y_devp, rf_pred, target_names=encoder.classes_))
```

```
                precision    recall  f1-score   support

          ARTS       0.08      0.12      0.10       297
 ARTS & CULTURE      0.07      0.15      0.09       225
   BLACK VOICES      0.31      0.26      0.28       803
       BUSINESS      0.30      0.21      0.25      1097
        COLLEGE      0.20      0.44      0.28       226
         COMEDY      0.39      0.30      0.34       961
          CRIME      0.27      0.47      0.35       647
 CULTURE & ARTS      0.16      0.25      0.19       186
        DIVORCE      0.41      0.63      0.50       640
      EDUCATION      0.15      0.32      0.20       196
  ENTERTAINMENT      0.56      0.29      0.38      3023
    ENVIRONMENT      0.12      0.32      0.17       259
          FIFTY      0.06      0.13      0.08       258
   FOOD & DRINK      0.44      0.45      0.45      1182
      GOOD NEWS      0.06      0.07      0.06       263
          GREEN      0.21      0.19      0.20       506
 HEALTHY LIVING      0.21      0.17      0.19      1252
```

```
         IMPACT       0.13      0.15      0.14       621
 LATINO VOICES        0.13      0.20      0.16       204
          MEDIA       0.23      0.36      0.28       505
          MONEY       0.15      0.35      0.21       326
      PARENTING       0.41      0.46      0.43      1654
        PARENTS       0.20      0.23      0.21       764
       POLITICS       0.78      0.52      0.63      6066
   QUEER VOICES       0.64      0.59      0.62      1200
       RELIGION       0.34      0.36      0.35       506
        SCIENCE       0.26      0.33      0.29       426
         SPORTS       0.43      0.39      0.41       911
          STYLE       0.18      0.21      0.19       409
 STYLE & BEAUTY       0.67      0.64      0.66      1841
          TASTE       0.15      0.19      0.17       381
           TECH       0.23      0.34      0.28       410
  THE WORLDPOST       0.24      0.34      0.28       676
         TRAVEL       0.54      0.43      0.48      1817
       WEDDINGS       0.54      0.73      0.62       709
     WEIRD NEWS       0.15      0.11      0.13       476
       WELLNESS       0.53      0.44      0.48      3374
          WOMEN       0.22      0.22      0.22       653
     WORLD NEWS       0.12      0.21      0.15       406
      WORLDPOST       0.19      0.20      0.20       509

       accuracy                           0.39     37660
      macro avg       0.29      0.33      0.30     37660
   weighted avg       0.45      0.39      0.41     37660
```

Multinomial Naïve Bayes Model:

In Multinomial Naïve Bayes Model we got accuracy for development data with 52%. So later we check with test data where we got accuracy with 54%. According to this we can say that this model is better than previously mentioned models.

Multinomial Naive Bayes Model

```
nb_model = MultinomialNB()
nb_model.fit(X_train, Y_train)
nb_pred = nb_model .predict(X_devp)
print(classification_report(Y_devp, nb_pred, target_names=encoder.classes_))
```

	precision	recall	f1-score	support
ARTS	0.24	0.19	0.21	297
ARTS & CULTURE	0.21	0.12	0.15	225
BLACK VOICES	0.37	0.27	0.31	803
BUSINESS	0.42	0.39	0.41	1097
COLLEGE	0.37	0.31	0.34	226
COMEDY	0.45	0.40	0.42	961
CRIME	0.39	0.67	0.49	647
CULTURE & ARTS	0.31	0.26	0.28	186
DIVORCE	0.61	0.63	0.62	640
EDUCATION	0.32	0.36	0.34	196
ENTERTAINMENT	0.57	0.60	0.58	3023
ENVIRONMENT	0.39	0.25	0.30	259
FIFTY	0.14	0.11	0.12	258
FOOD & DRINK	0.53	0.69	0.60	1182
GOOD NEWS	0.28	0.20	0.23	263
GREEN	0.31	0.31	0.31	506
HEALTHY LIVING	0.27	0.15	0.19	1252
HOME & LIVING	0.61	0.66	0.63	795
IMPACT	0.26	0.28	0.27	621
PARENTING	0.42	0.52	0.47	1654
PARENTS	0.29	0.24	0.26	764
POLITICS	0.72	0.71	0.72	6066
QUEER VOICES	0.68	0.55	0.61	1200
RELIGION	0.51	0.37	0.43	506
SCIENCE	0.49	0.44	0.47	426
SPORTS	0.60	0.56	0.58	911
STYLE	0.26	0.15	0.19	409
STYLE & BEAUTY	0.67	0.70	0.68	1841
TASTE	0.26	0.17	0.21	381
TECH	0.42	0.39	0.41	410
THE WORLDPOST	0.38	0.45	0.41	676
TRAVEL	0.59	0.69	0.64	1817
WEDDINGS	0.73	0.67	0.70	709
WEIRD NEWS	0.23	0.18	0.20	476
WELLNESS	0.52	0.65	0.58	3374
WOMEN	0.30	0.28	0.29	653
WORLD NEWS	0.27	0.17	0.21	406
WORLDPOST	0.28	0.30	0.29	509
accuracy			0.52	37660
macro avg	0.41	0.39	0.40	37660
weighted avg	0.51	0.52	0.51	37660

Support Vector Classification:

Comparing with Random Forest model this model also have better accuracy with 54% with development data. But, while we check with test data we got accuracy with 51%. So we finalised multi-nomial naïve bias model from the machine learning models.

Support Vector Classification

```
[56] from sklearn.svm import SVC
     svc_model = SVC()
     svc_model.fit(X_train, Y_train)
     svc_pred = svc_model.predict(X_devp)
     print(classification_report(Y_devp, svc_pred, target_names=encoder.classes_))
```

```
                    precision    recall  f1-score   support

             ARTS       0.24      0.11      0.15       297
    ARTS & CULTURE      0.28      0.07      0.11       225
      BLACK VOICES      0.47      0.24      0.32       803
          BUSINESS      0.46      0.33      0.39      1097
           COLLEGE      0.39      0.28      0.32       226
            COMEDY      0.59      0.31      0.41       961
             CRIME      0.53      0.51      0.52       647
    CULTURE & ARTS      0.74      0.17      0.27       186
           DIVORCE      0.81      0.56      0.66       640
         EDUCATION      0.34      0.11      0.16       196
     ENTERTAINMENT      0.41      0.69      0.52      3023
       ENVIRONMENT      0.89      0.15      0.26       259
             FIFTY      0.41      0.03      0.06       258
      FOOD & DRINK      0.57      0.67      0.62      1182
         GOOD NEWS      0.40      0.08      0.13       263
```

+ Code + Text

```
[56]    LATINO VOICES       0.57      0.10      0.17       204
                MEDIA       0.54      0.27      0.36       505
                MONEY       0.62      0.23      0.33       326
            PARENTING       0.50      0.65      0.57      1654
              PARENTS       0.43      0.22      0.29       764
             POLITICS       0.59      0.85      0.70      6066
         QUEER VOICES       0.80      0.55      0.65      1200
             RELIGION       0.62      0.27      0.38       506
              SCIENCE       0.63      0.34      0.44       426
               SPORTS       0.63      0.48      0.54       911
                STYLE       0.59      0.21      0.31       409
       STYLE & BEAUTY       0.76      0.75      0.75      1841
                TASTE       0.53      0.03      0.05       381
                 TECH       0.58      0.31      0.41       410
        THE WORLDPOST       0.51      0.37      0.43       676
               TRAVEL       0.64      0.67      0.65      1817
             WEDDINGS       0.81      0.68      0.74       709
           WEIRD NEWS       0.31      0.15      0.20       476
             WELLNESS       0.44      0.81      0.57      3374
                WOMEN       0.35      0.26      0.30       653
           WORLD NEWS       0.43      0.06      0.11       406
            WORLDPOST       0.45      0.11      0.18       509

             accuracy                           0.54     37660
            macro avg       0.53      0.34      0.38     37660
         weighted avg       0.54      0.54      0.50     37660
```

```
[57] #Predicting using Naive Bayes
     print("\n\nPredicting test data using Multinomial Naive Bayesian")
     pred_final = nb_model.predict(X_test)
     print(classification_report(Y_test, pred_final, target_names=encoder.classes_))
```

```
Predicting test data using Multinomial Naive Bayesian
                precision    recall  f1-score   support

          ARTS       0.27      0.22      0.24       367
ARTS & CULTURE       0.25      0.13      0.17       335
  BLACK VOICES       0.41      0.31      0.35      1170
      BUSINESS       0.45      0.42      0.44      1480
       COLLEGE       0.40      0.38      0.39       278
        COMEDY       0.43      0.41      0.42      1283
         CRIME       0.39      0.66      0.49       834
CULTURE & ARTS       0.30      0.29      0.29       238
       DIVORCE       0.62      0.63      0.62       847
     EDUCATION       0.35      0.44      0.39       250
 ENTERTAINMENT       0.57      0.60      0.58      3981
   ENVIRONMENT       0.39      0.28      0.32       318
         FIFTY       0.16      0.11      0.13       366
  FOOD & DRINK       0.51      0.66      0.58      1578
     GOOD NEWS       0.26      0.20      0.23       348
         GREEN       0.32      0.33      0.33       621
HEALTHY LIVING       0.26      0.14      0.19      1674
```

[57]

IMPACT	0.34	0.32	0.33	863
LATINO VOICES	0.33	0.09	0.15	289
MEDIA	0.45	0.39	0.41	722
MONEY	0.37	0.47	0.41	408
PARENTING	0.42	0.51	0.46	2137
PARENTS	0.27	0.24	0.25	963
POLITICS	0.72	0.72	0.72	8098
QUEER VOICES	0.66	0.54	0.60	1593
RELIGION	0.50	0.38	0.43	658
SCIENCE	0.44	0.38	0.40	520
SPORTS	0.58	0.54	0.56	1239
STYLE	0.31	0.17	0.22	574
STYLE & BEAUTY	0.66	0.71	0.69	2429
TASTE	0.25	0.13	0.17	534
TECH	0.42	0.42	0.42	521
THE WORLDPOST	0.40	0.45	0.42	905
TRAVEL	0.61	0.68	0.65	2538
WEDDINGS	0.71	0.66	0.68	945
WEIRD NEWS	0.26	0.20	0.23	670
WELLNESS	0.52	0.65	0.58	4557
WOMEN	0.34	0.29	0.32	870
WORLD NEWS	0.29	0.18	0.22	551
WORLDPOST	0.30	0.31	0.31	629
accuracy			0.52	50214
macro avg	0.42	0.40	0.40	50214
weighted avg	0.51	0.52	0.51	50214

While we use vectorizer from the BOW (bag of words) the order of words will be not saved so we use reverse vocabulary function to save the order.

[60]

```
reverse_vocabulary = {}
vocabulary = c_vectorizer.vocabulary_
for word in vocabulary:
    index = vocabulary[word]
    reverse_vocabulary[index] = word

vector = c_vectorizer.transform(iter(['Nasa scientists are good']))
indexes = vector.indices
for i in indexes:
    print (reverse_vocabulary[i])
```

```
good
nasa
scientists
```

Viewing the words using the model multi-nomial naive bias for help in predicting the categories.

```
[62] nb1_model=MultinomialNB()
     nb1_model.fit(X_train_whole, Y_train_whole)
     coefs = nb1_model.coef_
     target_names = encoder.classes_

     for i in range(len(target_names)):
         words = []
         for j in coefs[i].argsort()[-20:]:
             words.append(reverse_vocabulary[j])
         print (target_names[i], '-', words, "\n")
```

```
GOOD NEWS - ['little', 'new', 'watch', 'day', 'family', 'time', 'life', 'home', 'world', 'boy', 'love', 'old', 'woman', 'like',

GREEN - ['years', 'environmental', 'time', 'like', 'energy', 'u', 'year', 'could', 'trump', 'global', 'water', 'california', 'oi

HEALTHY LIVING - ['many', '5', 'help', 'things', 'know', 'day', 'ways', 'could', 'need', 'like', 'may', 'us', 'get', 'make', 'ne

HOME & LIVING - ['pinterest', '10', 'check', 'something', 'time', 'design', 'like', 'get', 'craft', 'diy', 'us', 'ideas', 'new',

IMPACT - ['change', 'years', 'many', 'health', 'social', 'make', 'homeless', 'life', 'need', 'us', 'year', 'new', 'children', 't

LATINO VOICES - ['american', 'immigrant', 'immigrants', 'said', 'us', 'rico', 'women', 'year', 'first', 'latina', 'u', 'mexican'

MEDIA - ['editor', 'white', 'bill', 'reporter', 'time', 'press', 'journalists', 'president', 'host', 'said', 'york', 'cnn', 'tir

MONEY - ['ways', 'could', '000', 'card', 'best', 'pay', 'like', 'debt', 'people', 'make', 'may', 'one', 'get', 'year', 'time', '

PARENTING - ['get', 'make', 'school', 'life', 'old', 'video', 'family', 'know', 'new', 'mom', 'like', 'day', 'year', 'time', 'ba
```

```
PARENTING - ['get', 'make', 'school', 'life', 'old', 'video', 'family', 'know', 'new', 'mom', 'like', 'day', 'year', 'time', 'b
PARENTS - ['moms', 'son', 'mother', 'know', 'parenting', 'daughter', 'life', 'new', 'things', 'year', 'baby', 'child', 'day', '
POLITICS - ['could', 'republicans', 'republican', 'white', 'state', 'would', 'people', 'hillary', 'one', 'u', 'house', 'says',
QUEER VOICES - ['man', 'year', 'time', 'men', 'like', 'first', 'sex', 'community', 'love', 'week', 'marriage', 'one', 'trans',
RELIGION - ['catholic', 'day', 'religion', 'jesus', 'spiritual', 'christian', 'life', 'new', 'faith', 'us', 'muslim', 'world',
SCIENCE - ['first', 'time', 'way', 'planet', 'mars', 'like', 'life', 'world', 'years', 'could', 'earth', 'one', 'nasa', 'scienc
SPORTS - ['bowl', 'u', 'players', 'year', 'win', 'nba', 'like', 'time', 'one', 'sports', 'olympic', 'player', 'world', 'new', '
STYLE - ['jenner', 'need', 'way', 'summer', 'makeup', 'time', 'make', 'red', 'looks', 'hair', 'one', 'look', 'beauty', 'style',
STYLE & BEAUTY - ['one', 'like', 'pinterest', 'huffpost', 'facebook', 'hair', 'photo', 'beauty', 'us', 'twitter', 'dress', 'wan
TASTE - ['way', 'need', '10', 'ice', 'cream', 'eat', 'day', 'good', 'delicious', 'summer', 'time', 'get', 'one', 'like', 'easy'
TECH - ['top', 'get', 'world', 'video', 'people', 'company', 'could', 'look', 'one', 'social', 'videos', 'twitter', 'youtube',
THE WORLDPOST - ['first', 'least', 'country', 'syria', 'korea', 'north', 'one', 'year', 'world', 'china', 'attack', 'state', 's
TRAVEL - ['make', 'places', 'vacation', 'trip', 'hotels', 'year', 'around', 'day', 'hotel', '10', 'get', 'time', 'city', 'like'
WEDDINGS - ['dress', 'brides', 'like', 'huffpost', 'planning', 'big', 'check', 'get', 'bride', 'couples', 'couple', 'love', 'vi
WEIRD NEWS - ['get', 'world', 'time', 'make', 'video', 'said', 'trump', 'year', 'know', 'news', 'dog', 'police', 'weird', 'watc
WELLNESS - ['healthy', 'good', 'know', 'way', 'cancer', 'could', 'sleep', 'day', 'may', 'like', 'make', 'get', 'study', 'us', '
```

Step 11:

Deep Learning: Convolution Neural Network

After completion of machine learning models we trained CNN model from the initial stage like data prepossessing.

Project_CNNipynb

File Edit View Insert Runtime Tools Help All changes saved

Files

Upload Refresh Mount Drive

- ..
- sample_data
- News_Category_Dataset_v2.json
- glove.6B.100d.txt
- glove.6B.200d.txt
- glove.6B.300d.txt
- glove.6B.50d.txt
- glove.6B.zip

\+ Code + Text

```
[19] import numpy as np # linear algebra
     import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

     import matplotlib.pyplot as plt
     plt.rcParams['figure.figsize'] = (6,6)

     from keras import backend as K
     from keras.engine.topology import Layer
     from keras import initializers, regularizers, constraints

     from keras.preprocessing import sequence
     from keras.preprocessing.text import Tokenizer, text_to_word_sequence
     from keras.utils import np_utils
     from keras.layers import Embedding
     from keras.initializers import Constant


     from sklearn.model_selection import train_test_split
     from sklearn.metrics import confusion_matrix


     import os


     import pandas as pd
```

```
[5] df = pd.read_json('News_Category_Dataset_v2.json', lines=True)
    df.head()
```

	category	headline	authors	link	short_description	date
0	CRIME	There Were 2 Mass Shootings In Texas Last Week...	Melissa Jeltsen	https://www.huffingtonpost.com/entry/texas-ama...	She left her husband. He killed their children...	2018-05-26
1	ENTERTAINMENT	Will Smith Joins Diplo And Nicky Jam For The 2...	Andy McDonald	https://www.huffingtonpost.com/entry/will-smit...	Of course it has a song.	2018-05-26
2	ENTERTAINMENT	Hugh Grant Marries For The First Time At Age 57	Ron Dicker	https://www.huffingtonpost.com/entry/hugh-gran...	The actor and his longtime girlfriend Anna Ebe...	2018-05-26
3	ENTERTAINMENT	Jim Carrey Blasts 'Castrato' Adam Schiff And D...	Ron Dicker	https://www.huffingtonpost.com/entry/jim-carre...	The actor gives Dems an ass-kicking for not fi...	2018-05-26
4	ENTERTAINMENT	Julianna Margulies Uses Donald Trump Poop Bags...	Ron Dicker	https://www.huffingtonpost.com/entry/julianna-...	The "Dietland" actress said using the bags is ...	2018-05-26

Checking the category types using groupby function

```
[6] cates = df.groupby('category')
    print("total categories:", cates.ngroups)
    print(cates.size())

    df.category = df.category.map(lambda x: "WORLDPOST" if x == "THE WORLDPOST" else x)
```

```
total categories: 41
category
ARTS               1509
ARTS & CULTURE     1339
BLACK VOICES       4528
BUSINESS           5937
COLLEGE            1144
COMEDY             5175
CRIME              3405
CULTURE & ARTS     1030
DIVORCE            3426
EDUCATION          1004
ENTERTAINMENT     16058
ENVIRONMENT        1323
FIFTY              1401
FOOD & DRINK       6226
GOOD NEWS          1398
GREEN              2622
HEALTHY LIVING     6694
HOME & LIVING      4195
IMPACT             3459
LATINO VOICES      1129
MEDIA              2815
MONEY              1707
```

```
[6]
IMPACT             3459
LATINO VOICES      1129
MEDIA              2815
MONEY              1707
PARENTING          8677
PARENTS            3955
POLITICS          32739
QUEER VOICES       6314
RELIGION           2556
SCIENCE            2178
SPORTS             4884
STYLE              2254
STYLE & BEAUTY     9649
TASTE              2096
TECH               2082
THE WORLDPOST      3664
TRAVEL             9887
WEDDINGS           3651
WEIRD NEWS         2670
WELLNESS          17827
WOMEN              3490
WORLD NEWS         2177
WORLDPOST          2579
dtype: int64
```

Combining the short description and headline as a single attribute and deleting the sentences with word length less than 5.

```python
[7]  # using headlines and short_description as input X

     df['text'] = df.headline + " " + df.short_description

     # tokenizing
     from keras.preprocessing.text import Tokenizer, text_to_word_sequence
     tokenizer = Tokenizer()
     tokenizer.fit_on_texts(df.text)
     X = tokenizer.texts_to_sequences(df.text)
     df['words'] = X
```

```python
[8]  # delete some empty and short data
     df['word_length'] = df.words.apply(lambda i: len(i))
     df = df[df.word_length >= 5]
     df.head()
```


```python
[8]  # delete some empty and short data
     df['word_length'] = df.words.apply(lambda i: len(i))
     df = df[df.word_length >= 5]
     df.head()
```


	category	headline	authors	link	short_description	date	text	words	word_
0	CRIME	There Were 2 Mass Shootings In Texas Last Week...	Melissa Jeltsen	https://www.huffingtonpost.com/entry/texas-ama...	She left her husband. He killed their children...	2018-05-26	There Were 2 Mass Shootings In Texas Last Week...	[74, 101, 257, 1331, 3001, 6, 698, 134, 96, 26...	
1	ENTERTAINMENT	Will Smith Joins Diplo And Nicky Jam For The 2...	Andy McDonald	https://www.huffingtonpost.com/entry/will-smit...	Of course it has a song.	2018-05-26	Will Smith Joins Diplo And Nicky Jam For The 2...	[42, 1604, 2960, 27762, 5, 25929, 5237, 8, 1, ...	
2	ENTERTAINMENT	Hugh Grant Marries For The First Time At Age 57	Ron Dicker	https://www.huffingtonpost.com/entry/hugh-gran...	The actor and his longtime girlfriend Anna Ebe...	2018-05-26	Hugh Grant Marries For The First Time At Age 5...	[5877, 5334, 8083, 8, 1, 76, 54, 21, 414, 8469...	
3	ENTERTAINMENT	Jim Carrey Blasts 'Castrato' Adam Schiff And D...	Ron Dicker	https://www.huffingtonpost.com/entry/jim-carre...	The actor gives Dems an ass-kicking for not fi...	2018-05-26	Jim Carrey Blasts 'Castrato' Adam Schiff And D...	[2710, 13374, 3596, 64143, 2295, 13055, 5, 569...	
4	ENTERTAINMENT	Julianna Margulies Uses Donald Trump Poop Bags...	Ron Dicker	https://www.huffingtonpost.com/entry/julianna-...	The "Dietland" actress said using the bags is ...	2018-05-26	Julianna Margulies Uses Donald Trump Poop Bags...	[41003, 36082, 1513, 97, 48, 7915, 3134, 2, 96...	

Padding was done with 50

Using 50 for padding length

```
[9]  maxlen = 50
     X = list(sequence.pad_sequences(df.words, maxlen=maxlen))
```

```
[10] # category to id

     categories = df.groupby('category').size().index.tolist()
     category_int = {}
     int_category = {}
     for i, k in enumerate(categories):
         category_int.update({k:i})
         int_category.update({i:k})

     df['c2id'] = df['category'].apply(lambda x: category_int[x])
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-vers
  if __name__ == '__main__':
```

Glove embedding was done from the below link we downloaded the zip file and unzip for the embedding.

```
[11] !wget http://nlp.stanford.edu/data/glove.6B.zip
```

```
--2020-04-24 20:50:42--  http://nlp.stanford.edu/data/glove.6B.zip
Resolving nlp.stanford.edu (nlp.stanford.edu)... 171.64.67.140
Connecting to nlp.stanford.edu (nlp.stanford.edu)|171.64.67.140|:80... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://nlp.stanford.edu/data/glove.6B.zip [following]
--2020-04-24 20:50:42--  https://nlp.stanford.edu/data/glove.6B.zip
Connecting to nlp.stanford.edu (nlp.stanford.edu)|171.64.67.140|:443... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: http://downloads.cs.stanford.edu/nlp/data/glove.6B.zip [following]
--2020-04-24 20:50:42--  http://downloads.cs.stanford.edu/nlp/data/glove.6B.zip
Resolving downloads.cs.stanford.edu (downloads.cs.stanford.edu)... 171.64.64.22
Connecting to downloads.cs.stanford.edu (downloads.cs.stanford.edu)|171.64.64.22|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 862182613 (822M) [application/zip]
Saving to: ‘glove.6B.zip’

glove.6B.zip        100%[===================>] 822.24M  2.01MB/s    in 6m 29s

2020-04-24 20:57:11 (2.12 MB/s) - ‘glove.6B.zip’ saved [862182613/862182613]
```

Glove Embedding was performed:

```
[12] !unzip glove*.zip
     word_index = tokenizer.word_index
```

```
Archive:  glove.6B.zip
  inflating: glove.6B.50d.txt
  inflating: glove.6B.100d.txt
  inflating: glove.6B.200d.txt
  inflating: glove.6B.300d.txt
```

```
[13] EMBEDDING_DIM = 100

     embeddings_index = {}
     f = open('glove.6B.100d.txt')
     for line in f:
         values = line.split()
         word = values[0]
         coefs = np.asarray(values[1:], dtype='float32')
         embeddings_index[word] = coefs
     f.close()

     print('Found %s unique tokens.' % len(word_index))
     print('Total %s word vectors.' % len(embeddings_index))
```

```
Found 116617 unique tokens.
Total 400000 word vectors.
```

We created the embedding layer with embedding_dim with length 100 and later we applied array to the x which is input and the category into id.

```
[20] embedding_matrix = np.zeros((len(word_index) + 1, EMBEDDING_DIM))
     for word, i in word_index.items():
         embedding_vector = embeddings_index.get(word)
         if embedding_vector is not None:
             embedding_matrix[i] = embedding_vector

     embedding_layer = Embedding(len(word_index)+1,
                                 EMBEDDING_DIM,
                                 embeddings_initializer=Constant(embedding_matrix),
                                 input_length=maxlen,
                                 trainable=False)

     X = np.array(X)
     Y = np_utils.to_categorical(list(df.c2id))
```

Split to training set and validation set

```
seed = 29
x_train, x_val, y_train, y_val = train_test_split(X, Y, test_size=0.2, random_state=seed)
```

Below we can observe the structure of the CNN model where we can see the embedding layer, dense layer and the output layer.

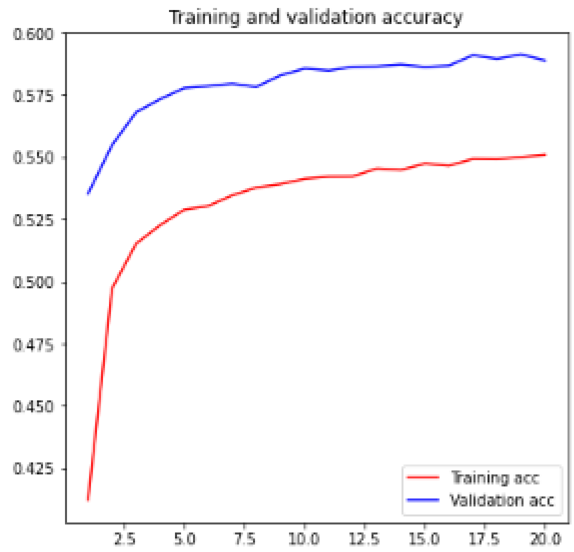
After we applied the clubbed the headline and short description as single attribute and Deleted some empty and short data with length less than 5. And padding length 50 was applied later converted Categories to ID. Later the tokenization was applied through glove embedding.

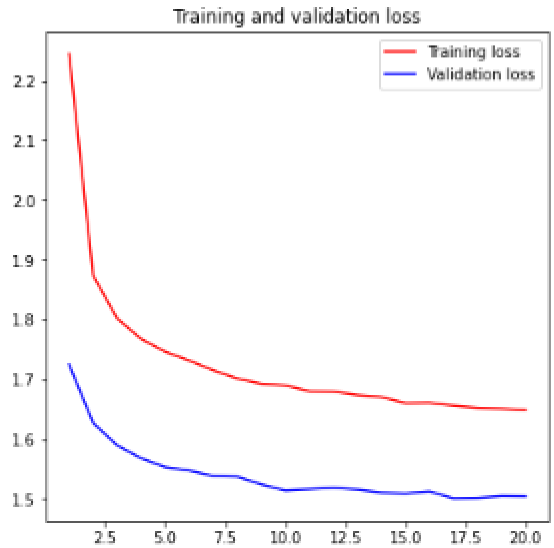
[33]

```
Model: "model_2"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_2 (InputLayer)            (None, 50)           0
__________________________________________________________________________________________________
embedding_2 (Embedding)         (None, 50, 100)      11661800    input_2[0][0]
__________________________________________________________________________________________________
conv1d_4 (Conv1D)               (None, 50, 64)       12864       embedding_2[0][0]
__________________________________________________________________________________________________
conv1d_5 (Conv1D)               (None, 50, 64)       19264       embedding_2[0][0]
__________________________________________________________________________________________________
conv1d_6 (Conv1D)               (None, 50, 64)       25664       embedding_2[0][0]
__________________________________________________________________________________________________
max_pooling1d_4 (MaxPooling1D)  (None, 16, 64)       0           conv1d_4[0][0]
__________________________________________________________________________________________________
max_pooling1d_5 (MaxPooling1D)  (None, 16, 64)       0           conv1d_5[0][0]
__________________________________________________________________________________________________
max_pooling1d_6 (MaxPooling1D)  (None, 16, 64)       0           conv1d_6[0][0]
__________________________________________________________________________________________________
dropout_5 (Dropout)             (None, 16, 64)       0           max_pooling1d_4[0][0]
__________________________________________________________________________________________________
dropout_6 (Dropout)             (None, 16, 64)       0           max_pooling1d_5[0][0]
__________________________________________________________________________________________________
dropout_7 (Dropout)             (None, 16, 64)       0           max_pooling1d_6[0][0]
__________________________________________________________________________________________________
concatenate_2 (Concatenate)     (None, 16, 192)      0           dropout_5[0][0]
                                                                 dropout_6[0][0]
                                                                 dropout_7[0][0]
__________________________________________________________________________________________________
flatten_2 (Flatten)             (None, 3072)         0           concatenate_2[0][0]
__________________________________________________________________________________________________
dropout_8 (Dropout)             (None, 3072)         0           flatten_2[0][0]
__________________________________________________________________________________________________
dense_2 (Dense)                 (None, 40)           122920      dropout_8[0][0]
==================================================================================================
Total params: 11,842,512
Trainable params: 180,712
Non-trainable params: 11,661,800
```

```
Train on 159931 samples, validate on 39983 samples
Epoch 1/20
159931/159931 [==============================] - 67s 417us/step - loss: 2.2455 - accuracy: 0.4123 - val_loss: 1.7249 - val_accuracy: 0.5353
Epoch 2/20
159931/159931 [==============================] - 66s 414us/step - loss: 1.8739 - accuracy: 0.4972 - val_loss: 1.6273 - val_accuracy: 0.5548
Epoch 3/20
159931/159931 [==============================] - 71s 443us/step - loss: 1.8019 - accuracy: 0.5152 - val_loss: 1.5897 - val_accuracy: 0.5680
Epoch 4/20
159931/159931 [==============================] - 68s 423us/step - loss: 1.7676 - accuracy: 0.5226 - val_loss: 1.5681 - val_accuracy: 0.5733
Epoch 5/20
159931/159931 [==============================] - 68s 425us/step - loss: 1.7463 - accuracy: 0.5288 - val_loss: 1.5533 - val_accuracy: 0.5777
Epoch 6/20
159931/159931 [==============================] - 65s 409us/step - loss: 1.7316 - accuracy: 0.5303 - val_loss: 1.5476 - val_accuracy: 0.5785
Epoch 7/20
159931/159931 [==============================] - 66s 410us/step - loss: 1.7152 - accuracy: 0.5346 - val_loss: 1.5385 - val_accuracy: 0.5795
Epoch 8/20
159931/159931 [==============================] - 67s 417us/step - loss: 1.7015 - accuracy: 0.5377 - val_loss: 1.5374 - val_accuracy: 0.5782
Epoch 9/20
159931/159931 [==============================] - 66s 411us/step - loss: 1.6927 - accuracy: 0.5391 - val_loss: 1.5244 - val_accuracy: 0.5828
Epoch 10/20
159931/159931 [==============================] - 66s 414us/step - loss: 1.6900 - accuracy: 0.5413 - val_loss: 1.5142 - val_accuracy: 0.5856
Epoch 11/20
159931/159931 [==============================] - 65s 408us/step - loss: 1.6803 - accuracy: 0.5422 - val_loss: 1.5166 - val_accuracy: 0.5847
Epoch 12/20
159931/159931 [==============================] - 68s 422us/step - loss: 1.6799 - accuracy: 0.5422 - val_loss: 1.5190 - val_accuracy: 0.5863
Epoch 13/20
159931/159931 [==============================] - 69s 435us/step - loss: 1.6739 - accuracy: 0.5454 - val_loss: 1.5159 - val_accuracy: 0.5864
Epoch 14/20
159931/159931 [==============================] - 70s 439us/step - loss: 1.6705 - accuracy: 0.5448 - val_loss: 1.5106 - val_accuracy: 0.5872
Epoch 15/20
159931/159931 [==============================] - 67s 421us/step - loss: 1.6604 - accuracy: 0.5474 - val_loss: 1.5094 - val_accuracy: 0.5862
Epoch 16/20
159931/159931 [==============================] - 65s 408us/step - loss: 1.6609 - accuracy: 0.5465 - val_loss: 1.5128 - val_accuracy: 0.5867
Epoch 17/20
159931/159931 [==============================] - 67s 419us/step - loss: 1.6562 - accuracy: 0.5492 - val_loss: 1.5010 - val_accuracy: 0.5910
Epoch 18/20
159931/159931 [==============================] - 69s 432us/step - loss: 1.6524 - accuracy: 0.5491 - val_loss: 1.5017 - val_accuracy: 0.5895
Epoch 19/20
159931/159931 [==============================] - 71s 442us/step - loss: 1.6509 - accuracy: 0.5500 - val_loss: 1.5050 - val_accuracy: 0.5913
Epoch 20/20
159931/159931 [==============================] - 73s 460us/step - loss: 1.6489 - accuracy: 0.5509 - val_loss: 1.5045 - val_accuracy: 0.5888
```

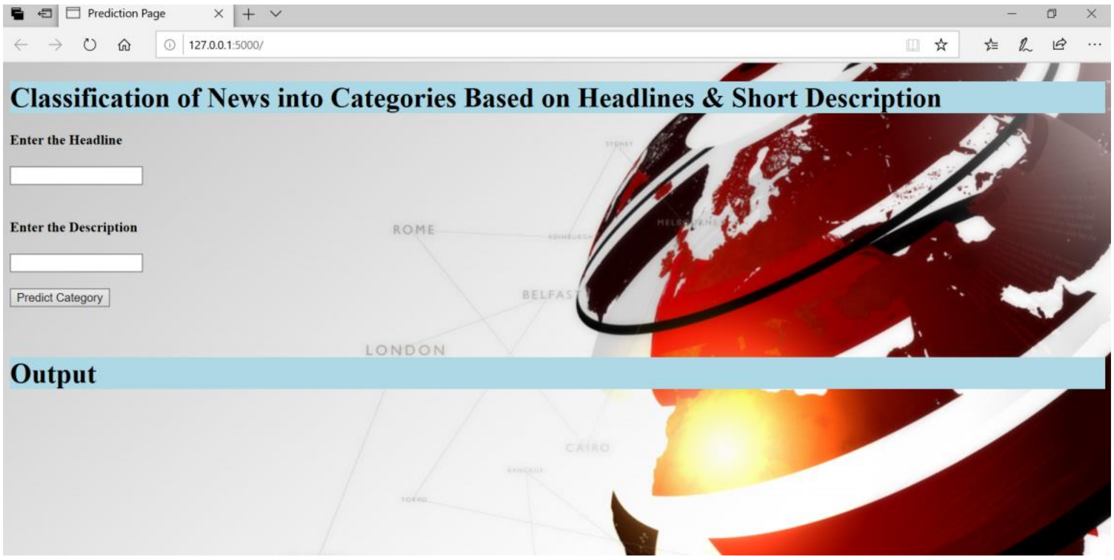
We can observe the epochs and the Validation accuracy & Validation Loss





From the above graphs we can observe the accuracy was increased for the test data compared to training data and same with validation loss we got better compared to the training data. But we can see our validation loss was greater than 1 which says our model was not upto the mark. For getting better low loss we try to change number of layers and change the activation layers but I can't see the better loss. I got fluctuations over the loss. Because the category data was not sufficient for the training data which created the under fitting. So, our loss was more. Basically we can say that our model was sensitive noise. For rectifying this we try to change data pre processing techniques other than bag of words and glove embedding.

Finally created the pickle file.



Team-work division:

- Geetanjali Makineni –
  - Dataset Preparation:
    - Combining Column Headline & Short Description
    - Stemming
  - Feature Engineering
    - Text Processing
    - Sampling
  - Model Selection
    - Decision Tree Model
    - Random Forest Model
  - Loss & Accuracy

- Akhil Teja Kanugolu –
  - Dataset Preparation:
    - Splitting Train, Test, Development
    - Visualization of Combined H&SD
  - Feature Engineering

- Vectorization
- Feature Reduction based on Threshold

- Model Selection
    - Multinomial Naïve Bayes Model
    - Support Vector Classification
    - CNN
- Hosting Static Webpage

Challenges Facing:

Since, we used a dataset with around 200K records, it can take more time to run the models.

More time consumed when pre-processing the data and cleaning it up.

Visualizing the data using Word Cloud is a little complex.

As the Data was Large the Run time was more while running the Machine Learning Models.

Validation loss was greater than 100%

Future Work:

- Here, we can also use some other machine learning other than Random fores, SVC, Decision Tree, Multinomial Naïve Bayes as well as deep learning algorithms other than CNN on our data set and we can see which model can give the better accuracy.

- We can here also induce some more other methods in feature engineering as well as parameters and can check how it might affect the accuracy of the model.