Summer Internship
IIIT Surat

Internship Report on Machine Learning and Deep Learning (Natural Language Processing)

Submitted by

Udit Nath
B.Tech in Computer Science and Engineering
Central Institute of Technology Kokrajhar

Under the guidance of

Dr. Pradeep Kumar Roy
Computer Science and Engineering Assistant Professor (Grade-II,Level-10)
IIIT Surat

Department of Computer Science and Engineering
INDIAN INSTITUTE OF INFORMATION TECHNOLOGY, SURAT

Summer Internship 2022

Abstract

This is an internship report on Machine Learning and Deep learning introduction. This report contains the details of the various tasks performed during two months of internship and is compiled for recognition for the completion of the internship. Tasks such as data cleaning, data processing and featured engineering, Feature Engineering using Countvectorizer and Bag of Words approach, Feature Engineering using Tf-Idf, Applying Machine learning algorithms on the processed dataset, Implementation of CNN algorithm on the same dataset, Text Normalization and finally Applying ML and CNN models on Bi-lingual dataset and comparing their outcomes.

Contents

1 Introduction **1**

2 Work Done **2**

2.1 Experimental setup . . . 2

2.1.1 Softwares used . . . 2

2.1.2 Hardware used . . . 2

2.2 Detailed Description . . . 3

2.2.1 Task 1 . . . 3

2.2.2 Task 2 . . . 3

2.2.3 Task 3 . . . 4

2.2.4 Task 4 . . . 6

2.2.5 Task 5 . . . 6

2.2.6 Task 6 . . . 9

2.2.7 Task 7 . . . 10

2.2.8 Task 8 . . . 10

3 Conclusion **12**

Chapter 1

Introduction

This report contains the detailed analysis of the two months of internship in Machine Learning and Deep Learning in IIIT Surat.
It starts with the introductory data cleaning process. Omitting Null spaces and deleting stopwords are the basics which are to be mastered first. Later on the data cleaning process involves the noise reduction in the data. It includes, removing HTML tags, punctuation and unnecessary obstructions to data reading process.
Introduction of Bag-of-words, Count vectorizer, TF-IDF had been covered in the next part. Convolutional Neural Network is next covered for the 1D layers for text analysis. At the end, a simple Codemix Dataset of Bi-Lingual data is been process with the techniques that were used for the prior tasks.

Chapter 2

Work Done

This Chapter introduces the various methods used in the Machine Learning and Deep Learning processes and the detailed analysis of the same.

2.1 Experimental setup

2.1.1 Softwares used

Google Collab is used as a platform for the internship tasks and some words had been done on Jupyter NB which had been later shited in the Google Collab notebook.

Python Libraries	**Usage**
Pandas	Used for data analysis and machine learning tasks
NumPy	Provides support for multi-dimensional arrays
Re	A regular expression (RE) is a language for specifying text search strings
NLTK	Making natural human language usable by computer programs
keras	Creating deep models
sklearn	Provides tools for machine learning and statistical modeling

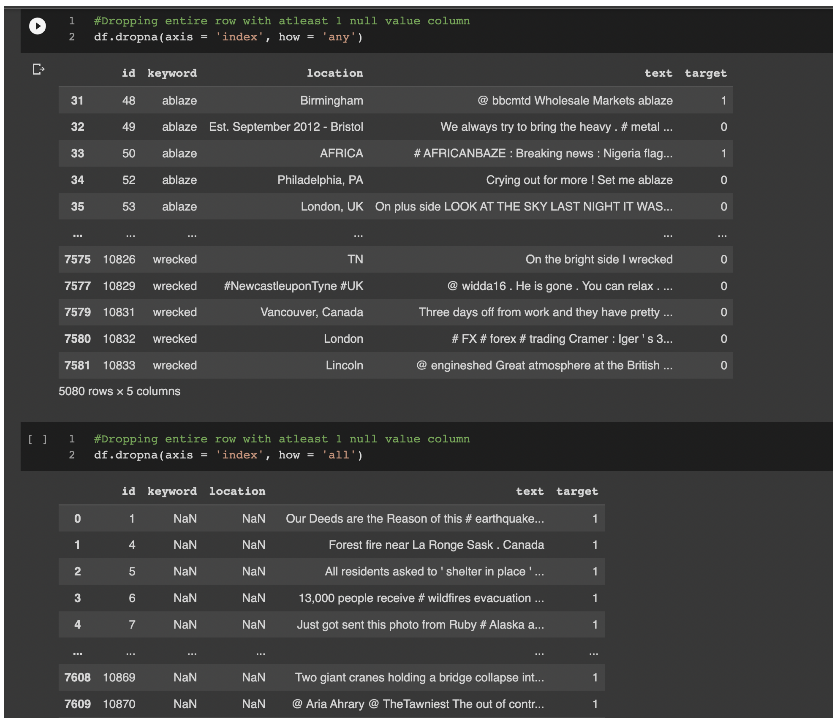
2.1.2 Hardware used

A portable system can be used. In this particular case Macbook Pro has been used containing Apple M1 chipset and 8-core CPU with 6 performance cores and 2 efficiency cores, 14-core GPU, 16-core Neural Engine, 200GB/s memory bandwidth.

2.2 Detailed Description

2.2.1 Task 1

The Task 1 contains the simple data cleaning process. For this task, a NPL disaster dataset has been used.
Firstly, the data is loaded with the help of Pandas library. Next, we check the null values present in the dataset. For this process, many iterations had been tried on the same dataset to experiment with the difference in results. Then the null value were filled with custom values which were user defined. inplace argument in the .dropna() function had been learnt as a bonus point for the further use. The use of frequent stopwords is repetitive in the dataset. So, it has to be removed in the cleaning process.



```
#Dropping entire row with atleast 1 null value column
df.dropna(axis = 'index', how = 'any')
```

	id	keyword	location	text	target
31	48	ablaze	Birmingham	@ bbcmtd Wholesale Markets ablaze	1
32	49	ablaze	Est. September 2012 - Bristol	We always try to bring the heavy . # metal ...	0
33	50	ablaze	AFRICA	# AFRICANBAZE : Breaking news : Nigeria flag...	1
34	52	ablaze	Philadelphia, PA	Crying out for more ! Set me ablaze	0
35	53	ablaze	London, UK	On plus side LOOK AT THE SKY LAST NIGHT IT WAS...	0
...	...	...	...	...	...
7575	10826	wrecked	TN	On the bright side I wrecked	0
7577	10829	wrecked	#NewcastleuponTyne #UK	@ widda16 . He is gone . You can relax	0
7579	10831	wrecked	Vancouver, Canada	Three days off from work and they have pretty ...	0
7580	10832	wrecked	London	# FX # forex # trading Cramer : Iger ' s 3...	0
7581	10833	wrecked	Lincoln	@ engineshed Great atmosphere at the British ...	0

5080 rows × 5 columns

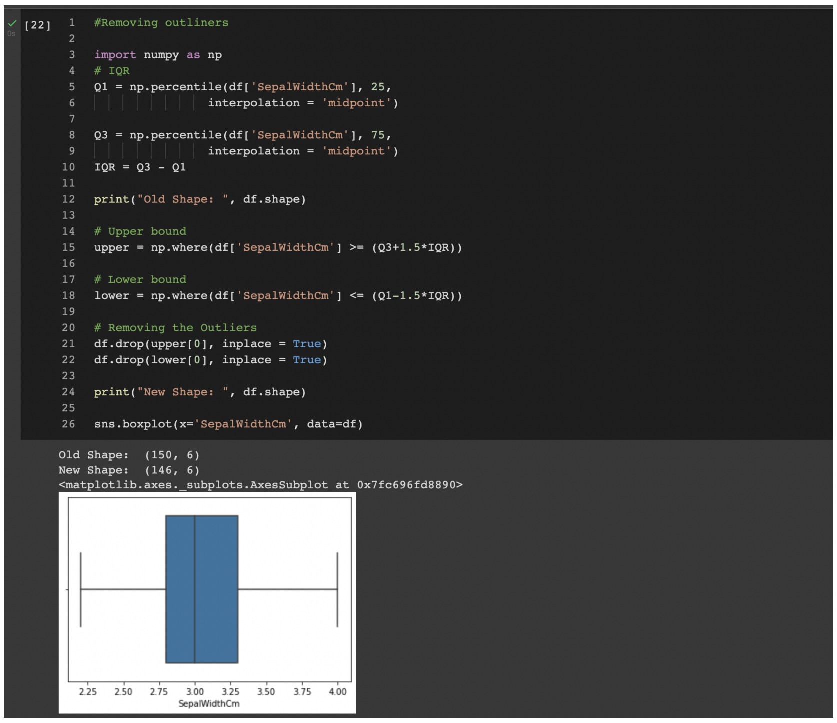
```
#Dropping entire row with atleast 1 null value column
df.dropna(axis = 'index', how = 'all')
```

	id	keyword	location	text	target
0	1	NaN	NaN	Our Deeds are the Reason of this # earthquake...	1
1	4	NaN	NaN	Forest fire near La Ronge Sask . Canada	1
2	5	NaN	NaN	All residents asked to ' shelter in place ' ...	1
3	6	NaN	NaN	13,000 people receive # wildfires evacuation ...	1
4	7	NaN	NaN	Just got sent this photo from Ruby # Alaska a...	1
...	...	...	...	...	...
7608	10869	NaN	NaN	Two giant cranes holding a bridge collapse int...	1
7609	10870	NaN	NaN	@ Aria Ahrary @ TheTawniest The out of contr...	1

Figure 2.1: Text Cleaning processes

2.2.2 Task 2

For performing Task 2, the same dataset has been used. In this task the basics of feature engineering are known. Some preprocessing techniques like cleaning the stop words had been implemented in this task. First of all we perform the basic data preprocessing tasks on the data. Then proceeded to data vsualization. We plot different graphs to see different outcome of the same dataset.Next we handle the co-relation in the dataset. At the end we detect the outliner using Interquartile range (IQR) and remove them.



```python
#Removing outliners

import numpy as np
# IQR
Q1 = np.percentile(df['SepalWidthCm'], 25,
                   interpolation = 'midpoint')

Q3 = np.percentile(df['SepalWidthCm'], 75,
                   interpolation = 'midpoint')
IQR = Q3 - Q1

print("Old Shape: ", df.shape)

# Upper bound
upper = np.where(df['SepalWidthCm'] >= (Q3+1.5*IQR))

# Lower bound
lower = np.where(df['SepalWidthCm'] <= (Q1-1.5*IQR))

# Removing the Outliers
df.drop(upper[0], inplace = True)
df.drop(lower[0], inplace = True)

print("New Shape: ", df.shape)

sns.boxplot(x='SepalWidthCm', data=df)
```

```
Old Shape:  (150, 6)
New Shape:  (146, 6)
<matplotlib.axes._subplots.AxesSubplot at 0x7fc696fd8890>
```

Figure 2.2: Preprocessing of the data

2.2.3 Task 3

In this part of feature extraction, the report deals with various methods and techniques that are used for feature extraction. The theories and usage of Bag-of-words, Count Vectorizer, TF-IDF had been learnet during the task 3.

A bag-of-words is a representation of text that describes the occurrence of words within a document. It involves two things:

A vocabulary of the known words.

A measure of the presence of known words.

Example:
This is my pen.
This is where I go to study.

Bag of Words for sentence 1:
”This” = 1
”is” = 1
”my” = 1
”pen” = 1
”where” = 0
”I” = 0
”go” = 0
”to” = 0
”school” = 0

Limitations:
Vocabulary, Sparsity and Meaning of sentences

CountVectorizer is a method to convert text to numerical data.

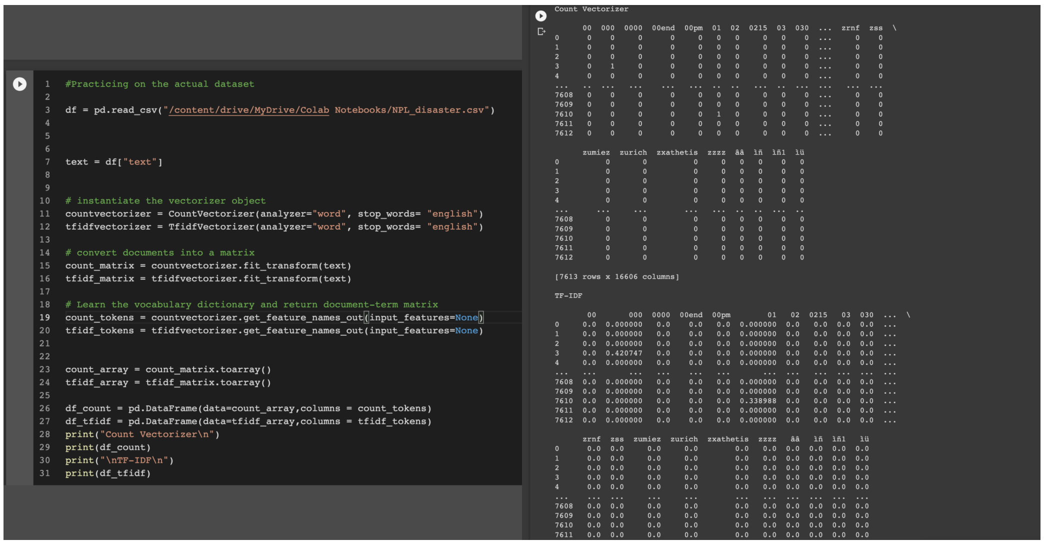
CountVectorizer(parameters):

- lowercase = True (default)
- stop_words: Words that are insignificant and can be avoided
- stop_words = ”english” — sklearn built in stop words list
- max_df=0.75 some words crosses the threshold of 75% and removed from the sparse matrix
- min_df=0.50 percentage of documents(0.50, ignore words appearing in 50% of documents)
- max_features = 3 it will select the 3 most common words in the data. binary = True the CountVectorizer no more takes into consideration the frequency of the term/word
- .fit_transform(df) = transforms the data to vector form
- .get_feature_names() = Learn the vocabulary dictionary and return document-term matrix
- .vocabulary = represents the position of the words in the matrix

2.2.4 Task 4

TF-IDF: Rescale the frequency of words by how often they appear in all documents, so that the scores for frequent words like “the” that are also frequent across all documents are penalized.

- Term Frequency: is a scoring of the frequency of the word in the current document
- Inverse Document Frequency: is a scoring of how rare the word is across documents



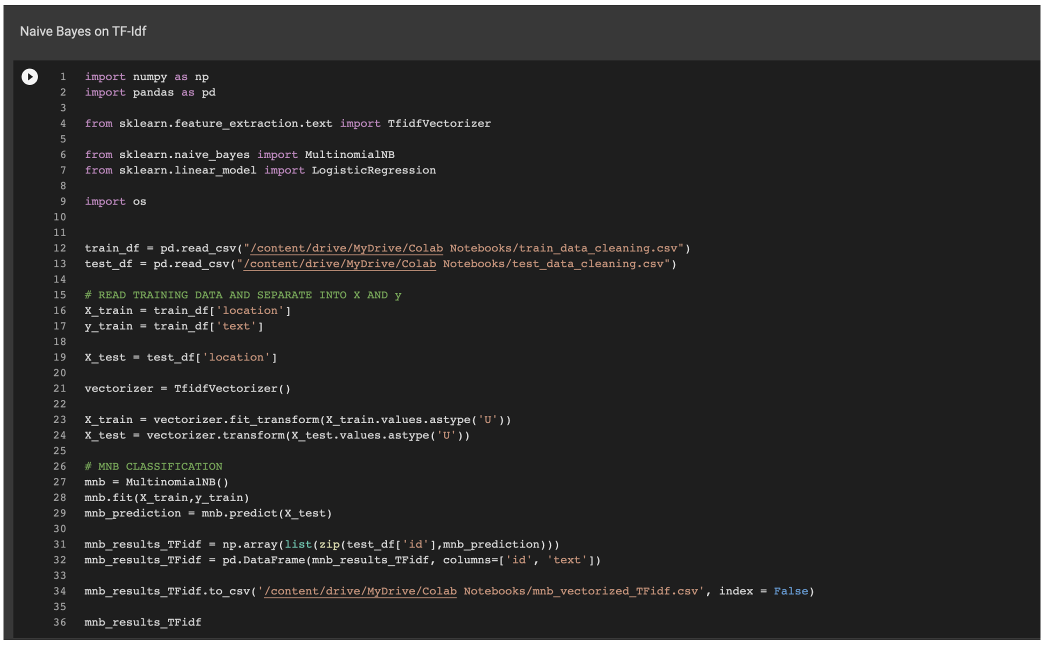
```python
#Practicing on the actual dataset

df = pd.read_csv("/content/drive/MyDrive/Colab Notebooks/NPL_disaster.csv")



text = df["text"]


# instantiate the vectorizer object
countvectorizer = CountVectorizer(analyzer="word", stop_words= "english")
tfidfvectorizer = TfidfVectorizer(analyzer="word", stop_words= "english")

# convert documents into a matrix
count_matrix = countvectorizer.fit_transform(text)
tfidf_matrix = tfidfvectorizer.fit_transform(text)

# Learn the vocabulary dictionary and return document-term matrix
count_tokens = countvectorizer.get_feature_names_out(input_features=None)
tfidf_tokens = tfidfvectorizer.get_feature_names_out(input_features=None)


count_array = count_matrix.toarray()
tfidf_array = tfidf_matrix.toarray()

df_count = pd.DataFrame(data=count_array,columns = count_tokens)
df_tfidf = pd.DataFrame(data=tfidf_array,columns = tfidf_tokens)
print("Count Vectorizer\n")
print(df_count)
print("\nTF-IDF\n")
print(df_tfidf)
```

```
Count Vectorizer

      00  000  0000  00end  00pm  01  02  0215  03  030  ...  zrnf  zss  \
0      0    0     0      0     0   0   0     0   0    0  ...     0    0
1      0    0     0      0     0   0   0     0   0    0  ...     0    0
2      0    0     0      0     0   0   0     0   0    0  ...     0    0
3      0    1     0      0     0   0   0     0   0    0  ...     0    0
4      0    0     0      0     0   0   0     0   0    0  ...     0    0
...   ..  ...   ...    ...   ...  ..  ..   ...  ..  ...  ...   ...  ...
7608   0    0     0      0     0   0   0     0   0    0  ...     0    0
7609   0    0     0      0     0   0   0     0   0    0  ...     0    0
7610   0    0     0      0     0   1   0     0   0    0  ...     0    0
7611   0    0     0      0     0   0   0     0   0    0  ...     0    0
7612   0    0     0      0     0   0   0     0   0    0  ...     0    0

      zumiez  zurich  zxathetis  zzzz  ââ  ìñ  ìñ1  ìü
0          0       0          0     0   0   0    0   0
1          0       0          0     0   0   0    0   0
2          0       0          0     0   0   0    0   0
3          0       0          0     0   0   0    0   0
4          0       0          0     0   0   0    0   0
...      ...     ...        ...   ...  ..  ..  ...  ..
7608       0       0          0     0   0   0    0   0
7609       0       0          0     0   0   0    0   0
7610       0       0          0     0   0   0    0   0
7611       0       0          0     0   0   0    0   0
7612       0       0          0     0   0   0    0   0

[7613 rows x 16606 columns]

TF-IDF

       00       000  0000  00end  00pm        01   02  0215   03  030  ...  \
0     0.0  0.000000   0.0    0.0   0.0  0.000000  0.0   0.0  0.0  0.0  ...
1     0.0  0.000000   0.0    0.0   0.0  0.000000  0.0   0.0  0.0  0.0  ...
2     0.0  0.000000   0.0    0.0   0.0  0.000000  0.0   0.0  0.0  0.0  ...
3     0.0  0.420747   0.0    0.0   0.0  0.000000  0.0   0.0  0.0  0.0  ...
4     0.0  0.000000   0.0    0.0   0.0  0.000000  0.0   0.0  0.0  0.0  ...
...   ...       ...   ...    ...   ...       ...  ...   ...  ...  ...  ...
7608  0.0  0.000000   0.0    0.0   0.0  0.000000  0.0   0.0  0.0  0.0  ...
7609  0.0  0.000000   0.0    0.0   0.0  0.000000  0.0   0.0  0.0  0.0  ...
7610  0.0  0.000000   0.0    0.0   0.0  0.338988  0.0   0.0  0.0  0.0  ...
7611  0.0  0.000000   0.0    0.0   0.0  0.000000  0.0   0.0  0.0  0.0  ...
7612  0.0  0.000000   0.0    0.0   0.0  0.000000  0.0   0.0  0.0  0.0  ...

      zrnf  zss  zumiez  zurich  zxathetis  zzzz   ââ   ìñ  ìñ1   ìü
0      0.0  0.0     0.0     0.0        0.0   0.0  0.0  0.0  0.0  0.0
1      0.0  0.0     0.0     0.0        0.0   0.0  0.0  0.0  0.0  0.0
2      0.0  0.0     0.0     0.0        0.0   0.0  0.0  0.0  0.0  0.0
3      0.0  0.0     0.0     0.0        0.0   0.0  0.0  0.0  0.0  0.0
4      0.0  0.0     0.0     0.0        0.0   0.0  0.0  0.0  0.0  0.0
...    ...  ...     ...     ...        ...   ...  ...  ...  ...  ...
7608   0.0  0.0     0.0     0.0        0.0   0.0  0.0  0.0  0.0  0.0
7609   0.0  0.0     0.0     0.0        0.0   0.0  0.0  0.0  0.0  0.0
7610   0.0  0.0     0.0     0.0        0.0   0.0  0.0  0.0  0.0  0.0
7611   0.0  0.0     0.0     0.0        0.0   0.0  0.0  0.0  0.0  0.0
```

Figure 2.3: CountVectorizer and TF-IDF

2.2.5 Task 5

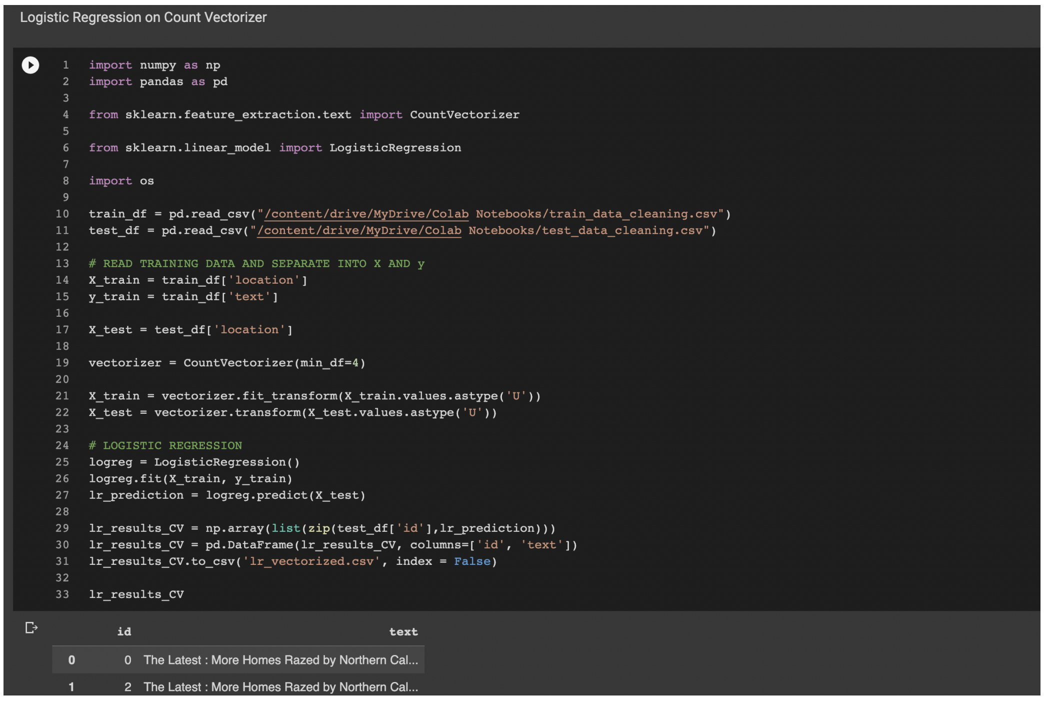
In this section, Machine Learning algorithms had been used for the same input of the previous tasks to compare the output of different approach.
Machine Learning algorithms like Naive Bayes, Logistic Regression and Random Forest had been practiced with the dataset. After getting the basic understanding of the ML algorithms, it had been implimented on the Iris Dataset, wherein the confusion matrix, accuracy, precision, recall and fi-score had been calculated and displayed.



Naive Bayes on TF-Idf

```
import numpy as np
import pandas as pd

from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression

import os


train_df = pd.read_csv("/content/drive/MyDrive/Colab Notebooks/train_data_cleaning.csv")
test_df = pd.read_csv("/content/drive/MyDrive/Colab Notebooks/test_data_cleaning.csv")

# READ TRAINING DATA AND SEPARATE INTO X AND y
X_train = train_df['location']
y_train = train_df['text']

X_test = test_df['location']

vectorizer = TfidfVectorizer()

X_train = vectorizer.fit_transform(X_train.values.astype('U'))
X_test = vectorizer.transform(X_test.values.astype('U'))

# MNB CLASSIFICATION
mnb = MultinomialNB()
mnb.fit(X_train,y_train)
mnb_prediction = mnb.predict(X_test)

mnb_results_TFidf = np.array(list(zip(test_df['id'],mnb_prediction)))
mnb_results_TFidf = pd.DataFrame(mnb_results_TFidf, columns=['id', 'text'])

mnb_results_TFidf.to_csv('/content/drive/MyDrive/Colab Notebooks/mnb_vectorized_TFidf.csv', index = False)

mnb_results_TFidf
```

Figure 2.4: Naive Bayes on TF-Idf

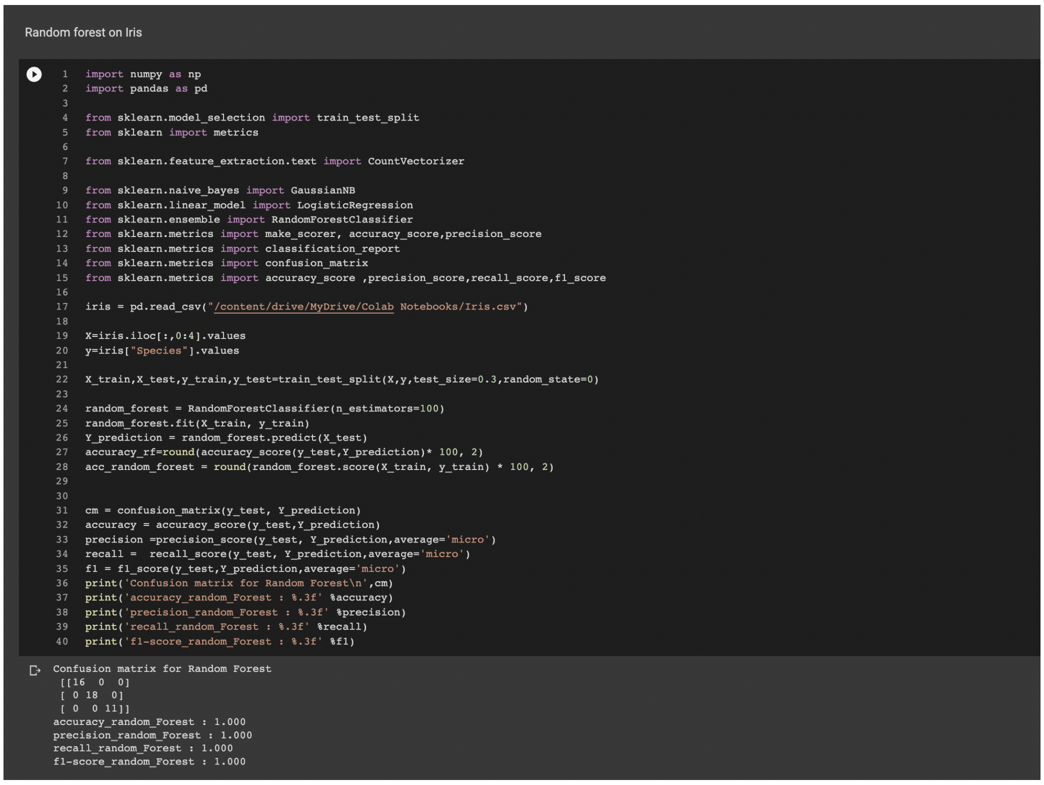


Logistic Regression on Count Vectorizer

```
import numpy as np
import pandas as pd

from sklearn.feature_extraction.text import CountVectorizer

from sklearn.linear_model import LogisticRegression

import os

train_df = pd.read_csv("/content/drive/MyDrive/Colab Notebooks/train_data_cleaning.csv")
test_df = pd.read_csv("/content/drive/MyDrive/Colab Notebooks/test_data_cleaning.csv")

# READ TRAINING DATA AND SEPARATE INTO X AND y
X_train = train_df['location']
y_train = train_df['text']

X_test = test_df['location']

vectorizer = CountVectorizer(min_df=4)

X_train = vectorizer.fit_transform(X_train.values.astype('U'))
X_test = vectorizer.transform(X_test.values.astype('U'))

# LOGISTIC REGRESSION
logreg = LogisticRegression()
logreg.fit(X_train, y_train)
lr_prediction = logreg.predict(X_test)

lr_results_CV = np.array(list(zip(test_df['id'],lr_prediction)))
lr_results_CV = pd.DataFrame(lr_results_CV, columns=['id', 'text'])
lr_results_CV.to_csv('lr_vectorized.csv', index = False)

lr_results_CV
```

	id	text
0	0	The Latest : More Homes Razed by Northern Cal...
1	2	The Latest : More Homes Razed by Northern Cal...

Figure 2.5: Logistic Regression on CountVectorizer


Random forest on Iris

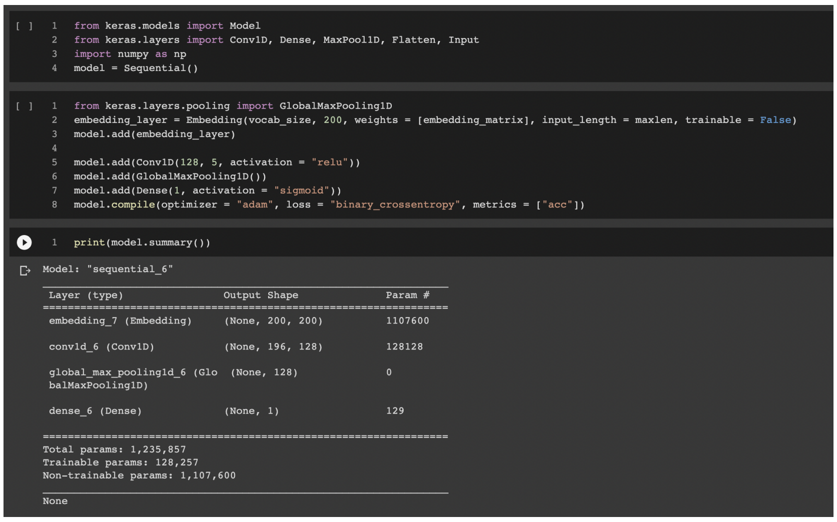
```python
import numpy as np
import pandas as pd

from sklearn.model_selection import train_test_split
from sklearn import metrics

from sklearn.feature_extraction.text import CountVectorizer

from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import make_scorer, accuracy_score,precision_score
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score ,precision_score,recall_score,f1_score

iris = pd.read_csv("/content/drive/MyDrive/Colab Notebooks/Iris.csv")

X=iris.iloc[:,0:4].values
y=iris["Species"].values

X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,random_state=0)

random_forest = RandomForestClassifier(n_estimators=100)
random_forest.fit(X_train, y_train)
Y_prediction = random_forest.predict(X_test)
accuracy_rf=round(accuracy_score(y_test,Y_prediction)* 100, 2)
acc_random_forest = round(random_forest.score(X_train, y_train) * 100, 2)


cm = confusion_matrix(y_test, Y_prediction)
accuracy = accuracy_score(y_test,Y_prediction)
precision =precision_score(y_test, Y_prediction,average='micro')
recall =  recall_score(y_test, Y_prediction,average='micro')
f1 = f1_score(y_test,Y_prediction,average='micro')
print('Confusion matrix for Random Forest\n',cm)
print('accuracy_random_Forest : %.3f' %accuracy)
print('precision_random_Forest : %.3f' %precision)
print('recall_random_Forest : %.3f' %recall)
print('f1-score_random_Forest : %.3f' %f1)
```

```
Confusion matrix for Random Forest
 [[16  0  0]
 [ 0 18  0]
 [ 0  0 11]]
accuracy_random_Forest : 1.000
precision_random_Forest : 1.000
recall_random_Forest : 1.000
f1-score_random_Forest : 1.000
```

Figure 2.6: Random forest on Iris Dataset

2.2.6 Task 6

This section deals with the implementation of Convolutional Neural Network on the dataset. For this task, the dataset used is taken from Codalab of the competition on Sentiment Analysis and Homophobia detection of YouTube comments. Out of many datasets to work on, the English dataset had been selected. Firstly, the dataset is imported, and preprocessed. HTML tags, special characters, punctuation are removed. Later, the multiple spaces in the data is cleaned and the entire dataset is rendered in lowercase. Next we removed the unnecessary tags that were in the dataset. We store the preprocessed data in a list. Finally we divide our dataset into train and test sets, followed by tokenizing the lists. Next data embedding is done which converts the textual data into numeric data that is needed for the CNN model. We will create an embedding matrix where each row number will correspond to the index of the word in the corpus the matrix will have two hundred columns where each column will contain the globe word embedding for the words in the corpus. A simple CNN is created with one convolution layer and one pooling layer. A sequential model is created followed by an embedding layer.Convolutional layer is generated with 128 features, the kernel size is 5 and the activation function used here is sigmoid. The feature size is reduced using the pooling layer. Next the model is evaluated and graph is plotted of the results



```
from keras.models import Model
from keras.layers import Conv1D, Dense, MaxPool1D, Flatten, Input
import numpy as np
model = Sequential()
```

```
from keras.layers.pooling import GlobalMaxPooling1D
embedding_layer = Embedding(vocab_size, 200, weights = [embedding_matrix], input_length = maxlen, trainable = False)
model.add(embedding_layer)

model.add(Conv1D(128, 5, activation = "relu"))
model.add(GlobalMaxPooling1D())
model.add(Dense(1, activation = "sigmoid"))
model.compile(optimizer = "adam", loss = "binary_crossentropy", metrics = ["acc"])
```

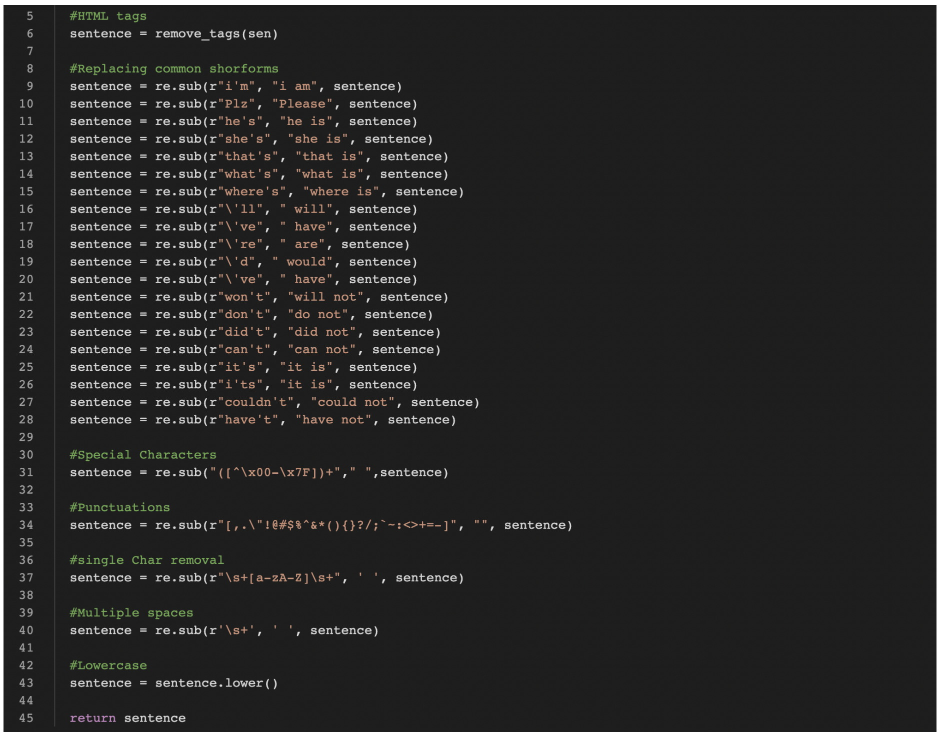
```
print(model.summary())
```

```
Model: "sequential_6"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_7 (Embedding)     (None, 200, 200)          1107600

 conv1d_6 (Conv1D)           (None, 196, 128)          128128

 global_max_pooling1d_6 (Glo  (None, 128)              0
 balMaxPooling1D)

 dense_6 (Dense)             (None, 1)                 129

=================================================================
Total params: 1,235,857
Trainable params: 128,257
Non-trainable params: 1,107,600
_________________________________________________________________
None
```

Figure 2.7: 1D Convolutional Neural Network

2.2.7 Task 7

For this task, a dataset containing codemix text had been used. In this the data is preprocessed as the previous tasks. Next in the data we can see there are common short forms that were misspelled or are incorrectly used. For this we create a small dictionary using re and we try to correct the words in our vicinity.



```python
#HTML tags
sentence = remove_tags(sen)

#Replacing common shorforms
sentence = re.sub(r"i'm", "i am", sentence)
sentence = re.sub(r"Plz", "Please", sentence)
sentence = re.sub(r"he's", "he is", sentence)
sentence = re.sub(r"she's", "she is", sentence)
sentence = re.sub(r"that's", "that is", sentence)
sentence = re.sub(r"what's", "what is", sentence)
sentence = re.sub(r"where's", "where is", sentence)
sentence = re.sub(r"\'ll", " will", sentence)
sentence = re.sub(r"\'ve", " have", sentence)
sentence = re.sub(r"\'re", " are", sentence)
sentence = re.sub(r"\'d", " would", sentence)
sentence = re.sub(r"\'ve", " have", sentence)
sentence = re.sub(r"won't", "will not", sentence)
sentence = re.sub(r"don't", "do not", sentence)
sentence = re.sub(r"did't", "did not", sentence)
sentence = re.sub(r"can't", "can not", sentence)
sentence = re.sub(r"it's", "it is", sentence)
sentence = re.sub(r"i'ts", "it is", sentence)
sentence = re.sub(r"couldn't", "could not", sentence)
sentence = re.sub(r"have't", "have not", sentence)

#Special Characters
sentence = re.sub("([^\x00-\x7F])+"," ",sentence)

#Punctuations
sentence = re.sub(r"[,.\"!@#$%^&*(){}?/;`~:<>+=-]", "", sentence)

#single Char removal
sentence = re.sub(r"\s+[a-zA-Z]\s+", ' ', sentence)

#Multiple spaces
sentence = re.sub(r'\s+', ' ', sentence)

#Lowercase
sentence = sentence.lower()

return sentence
```

Figure 2.8: 1D Convolutional Neural Network

2.2.8 Task 8

This task comprises of the work done in the internship. It introduces the data cleaning methods, the use of differnt machine leaning models to process the data and the NLP basics that are applied to the dataset.
First we install the required packages namely textblob, pyspellchecker, emoji, googletrans.
Then we start importing the required packages for the preprocessing of the dataset and also for the visualizing and translation. Next we import the dataset. It is a codemix dataset from kaggle containing tweet. Sentiment

analysis is to be done on the dataset.
After importing, we find that the lables are marked 0 fornegative and 4 for the positive reviews. So we create a new column stating the same. We then discard the columns that are not required for the particular NLP task.
After this we start the preprocessing task. First, we remove the emojis, next we compile a code with many contractions and their full forms and apply the same in the dataset. The emoticons, though not used widely nowadays are also taken care of next. After this, we remove the HTML tags, mentions and the stopwords.
We import the required NLTK libraries and then process the data with all the requied functions. The output is a cleaner dataset which we store in a separate file. After this, we correct grammar of the sentences and finally we get a clean text.
We then split the data into train and test lists. We then move forward to feed the data into different classifiers to see their outcomes.

Classifiers	**Accuracy**
Naive Bayes	63.32277953674388
Multinomail naive bayes	63.406098983502744
Bernouli naive bayes	63.27278786868855
Logistic regression	63.73937677053825

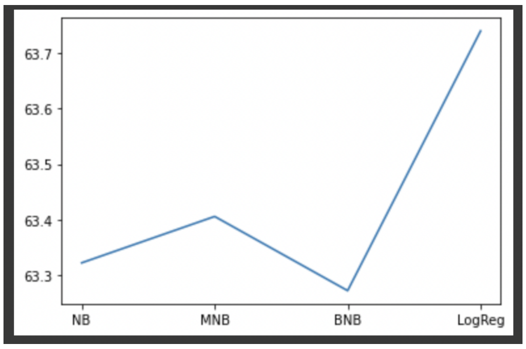


Figure 2.9: Classifiers accuracy plot

We notice that the Logistic Regression has the highest accuracy rate.
Next we create the confusion matrix of the data (positive and negative tweets). Next we use the google translate library to translate the data and make it monolingual.
Link: ***Google Colab Notebook***

Chapter 3

Conclusion

I conclude this report by sharing my personal experience with the 2 months internship on Machine Learning and Deep Learning. Firstly, I gained the basic understanding of the working of the python libraries and frameworks. Secondly, I got to work with different datasets and the process of how we can contribute to a competition. Thirdly, I got exposure about the different methods that are to be used in the process of Machine Learning and how those are different from that of Deep Learning. Lastly, I had been practicing the processes and eager to learn more in this domain.
Regarding the future scope of the internship, the models used are not by far perfectly used and there is much more to learn through the libraries used in this internship. The task 7 still contains just the English shortforms and much more data is needed in the real world situations. In task 8 there had been issues with the translation part and not all of the words get correctly translated due to similar words in Hindi and English like "Main", "To", etc.