National College of Ireland

Configuration Manual

MSc Research Project
MSc in Cloud Computing

Vipin Yadav
Student ID:19211791

School of Computing
National College of Ireland

Supervisor : Prof. Divyaa Manimaran Elango

National College of Ireland Project Submission Sheet School of Computing

Student Name:	Vipin Yadav
Student ID:	19211791
Program:	MSc in Cloud Computing
Year:	2021
Module:	MSc Research Project
Supervisor:	Divyaa Manimaran Elango www.ncirl.ie
Submission Due Date:	16-12-2021
Project Title:	Configuration Manual
Word Count:	767
Page Count:	11

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Vipin Yadav
Date:	31-01-2022

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Vipin Yadav
x19211791

1 Introduction

This configuration manual will document the implementation of the current research project. It will explain how to run the implemented scripts. This documentation will help to execute the code without any problem. There will be recommendations of hardware and software versions required to run this. The project's results can be replicated if these methods are followed properly.

2 System Configuration

2.1 Hardware and Software Configuration

For this project implementation, we have used Google Colab IDE and Microsoft Office tools are being used for formatting and writing

Hardware Configuration		**Software Configuration**
Operating System	Microsoft Windows 10	Microsoft Office suite
RAM	8 GB	IDE - Google collab
Hard Disk	500 GB	Python 3.6.9
Processor	Intel(R) Core(TM) i7-10510U CPU @ 1.80GHz, 2304 Mhz, 4 Core(s), 8 Logical Processor(s)	

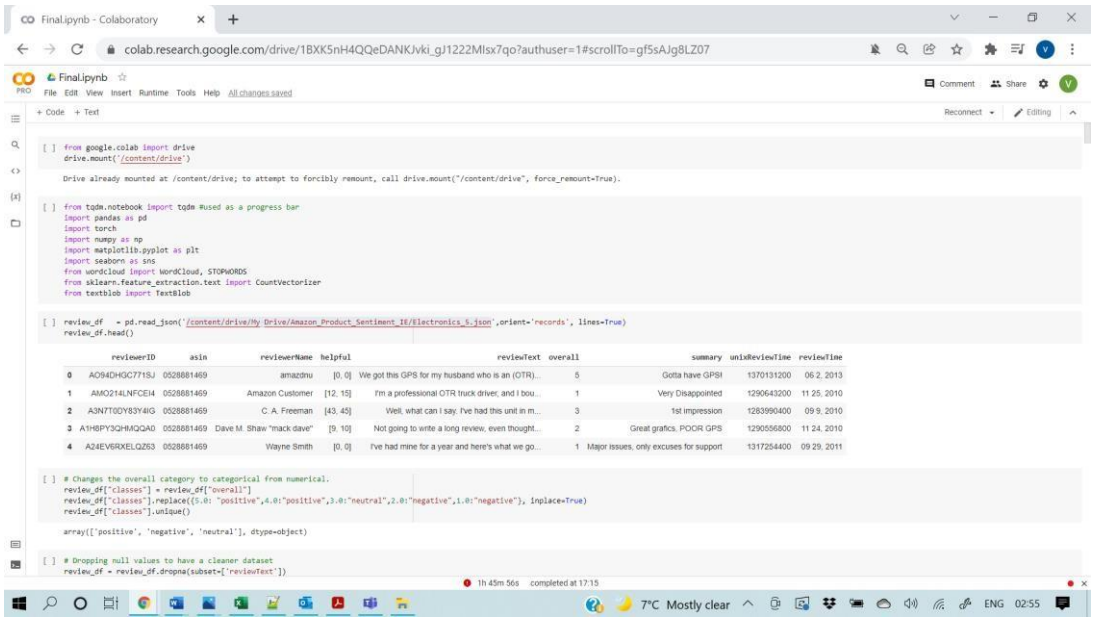
Figure 1: Software and Hardware configuration

2.2 Project Development

Amazon's product reviews dataset was created by Jianmo Ni and is publically presented on the below link:
http://deepyeti.ucsd.edu/jianmo/amazon/index.html

This Dataset of Electronic product needs to be uploaded on Google drive

Below figure 2 shows to mount the google collab with the drive so that the dataset can be easily available.




```
[ ] from google.colab import drive
    drive.mount('/content/drive')

    Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

Figure 2: Mounting drive with Google colab

```
from tqdm.notebook import tqdm #used as a progress bar
import pandas as pd
import torch
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from wordcloud import WordCloud, STOPWORDS
from sklearn.feature_extraction.text import CountVectorizer
from textblob import TextBlob
```

Figure 3: Import the import libraries

To run the code successfully required packages will be imported. Word cloud, Stopwords CountVectorisation (Pankaj et al. 2019) are important libraries for data cleaning. Figure 2a shows the list of required libraries that have been imported.

```
from tqdm.notebook import tqdm #used as a progress bar
import pandas as pd
import torch
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from wordcloud import WordCloud, STOPWORDS
from sklearn.feature_extraction.text import CountVectorizer
from textblob import TextBlob
```

Figure 2a: List of libraries imported

Data Read and Load: Dataset is in JSON format and uploaded in Drive location of /content/drive/My Drive/Amazon_Product_Sentiment_IE/Electronics_5.json. As shown in below figure 3, A data frame was created to read the dataset.

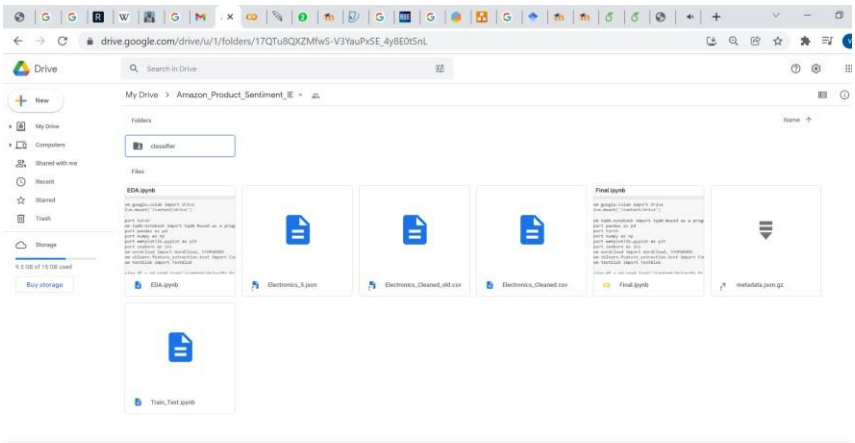


Fig A: Google Drive.

```
[ ] review_df   = pd.read_json('/content/drive/My Drive/Amazon_Product_Sentiment_IE/Electronics_5.json',orient='records', lines=True)
    review_df.head()
```

	reviewerID	asin	reviewerName	helpful	reviewText	overall	summary	unixReviewTime	reviewTime
0	AO94DHGC771SJ	0528881469	amazdnu	[0, 0]	We got this GPS for my husband who is an (OTR)...	5	Gotta have GPS!	1370131200	06 2, 2013
1	AMO214LNFCEI4	0528881469	Amazon Customer	[12, 15]	I'm a professional OTR truck driver, and I bou...	1	Very Disappointed	1290643200	11 25, 2010
2	A3N7T0DY83Y4IG	0528881469	C. A. Freeman	[43, 45]	Well, what can I say. I've had this unit in m...	3	1st impression	1283990400	09 9, 2010
3	A1H8PY3QHMQQA0	0528881469	Dave M. Shaw "mack dave"	[9, 10]	Not going to write a long review, even thought...	2	Great grafics, POOR GPS	1290556800	11 24, 2010
4	A24EV6RXELQZ63	0528881469	Wayne Smith	[0, 0]	I've had mine for a year and here's what we go...	1	Major issues, only excuses for support	1317254400	09 29, 2011

Figure 3

Figure 4 shows the categorization of data into positive, negative, and neutral categories with the help of the overall rating column. A positive review will be for 4 and 5 ratings, Negative (1 and 2) and Neutral review (3)

```
# Changes the overall category to categorical from numerical.
review_df["classes"] = review_df["overall"]
review_df["classes"].replace({5.0: "positive",4.0:"positive",3.0:"neutral",2.0:"negative",1.0:"negative"}, inplace=True)
review_df["classes"].unique()
```

```
array(['positive', 'negative', 'neutral'], dtype=object)
```

Figure 4

Data Cleaning:

Below figure 5 shows that null value rows are being dropped.

```
[ ]  # Dropping null values to have a cleaner dataset
     review_df = review_df.dropna(subset=['reviewText'])
```

Figure 5

Once data is divided into positive, negative, and natural ratings. A bar graph Figure 6 is plotted to visualize the ratio of Positive, negative, and neutral ratings

```
# Distribution of sentiment class
fig, axes = plt.subplots(1, figsize=(15,8))
fig.suptitle("Comparing Ratio of Neutral, Negative and Positive ratings", fontsize = 25)
plt.tight_layout(pad = 3.5)
sns.countplot(x = "classes", data = review_df)
axes.set_xlabel("Sentiment", fontsize = 20)
axes.set_ylabel("Count", fontsize = 20)
sns.despine()
```

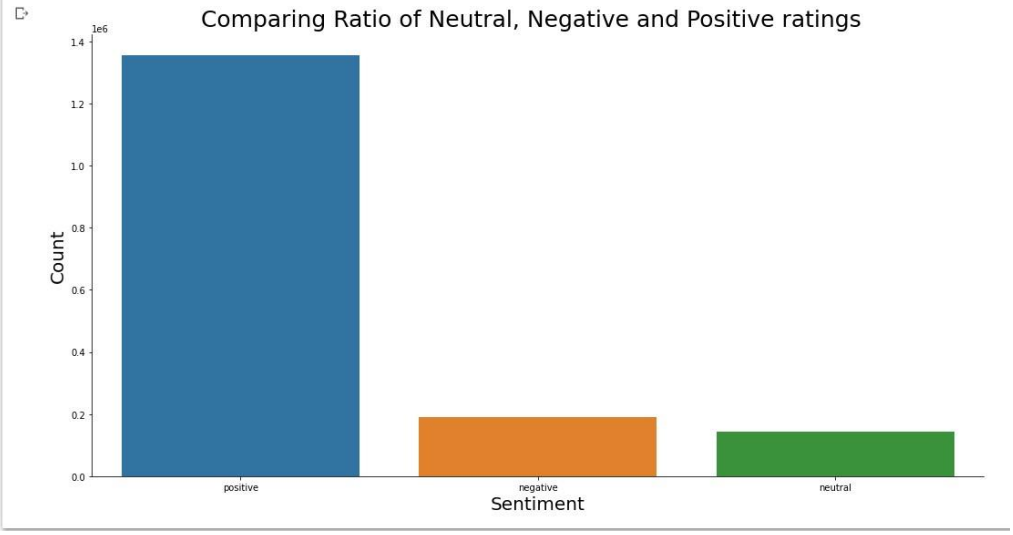


Figure: 6

```
# Percentage of neutral, negative, positive words in train and test data
def pert_count(data, category):
    return (len(data[data["classes"] == category])/len(data)) * 100
print(f"Percentage of neutral words in train --> {pert_count(review_df, 'neutral')} %")
print(f"Percentage of negative words in train --> {pert_count(review_df, 'negative')} %")
print(f"Percentage of positive words in train --> {pert_count(review_df, 'positive')} %")
```

```
Percentage of neutral words in train --> 8.421620328820712 %
Percentage of negative words in train --> 11.299156754606356 %
Percentage of positive words in train --> 80.27922291657293 %
```

Figure**: 7**

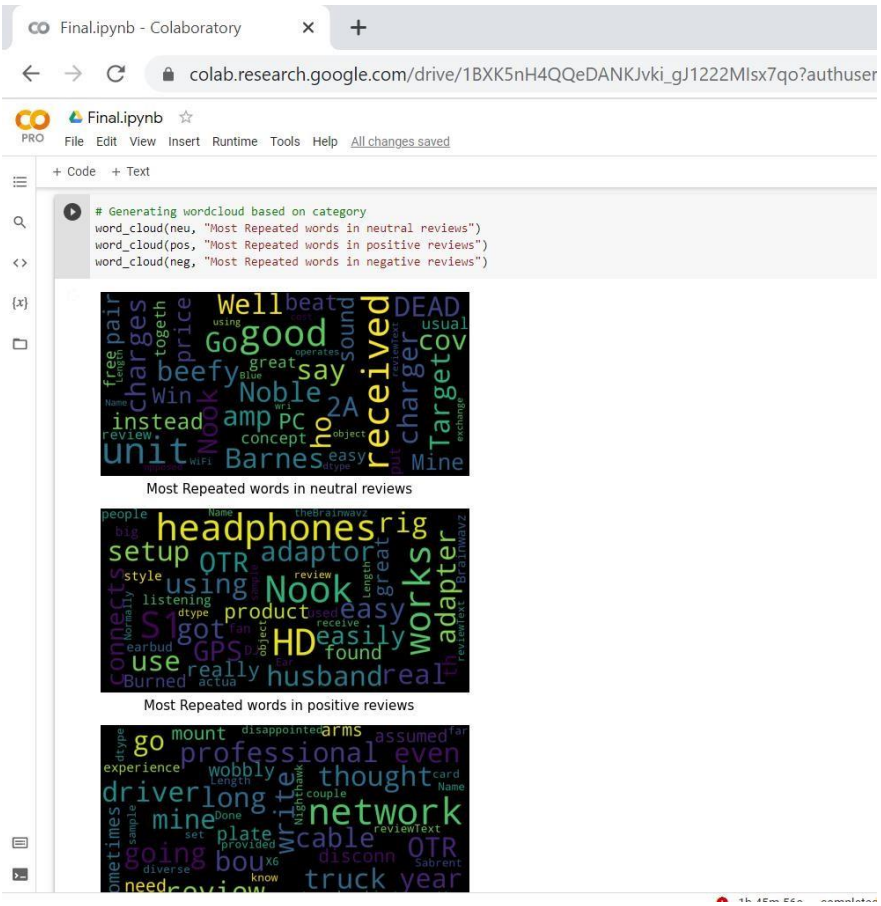


Figure 7a : Most repeated words

Below diagram 8 shows different data cleaning methods for removing punctuation

```
# removes all the punctuations for the strings of reviewText, ie '!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'
def remove_punctuation(text):
    no_punct=[words for words in text if words not in string.punctuation]
    words_wo_punct=''.join(no_punct)
    return words_wo_punct
```

```
[ ] def text_process(df):
    # Removing all the punctuations from the words, and changing the words to lower case to maintain uniformity
    df['reviewText']=df['reviewText'].apply(lambda x: remove_punctuation(x.lower()))
    # stemming
    stemmer = PorterStemmer()
    # stop words are the words like "the, I, our etc"
    words = stopwords.words("english")
    df['cleaned_reviews'] = df['reviewText'].apply(lambda x: " ".join([stemmer.stem(i) for i in re.sub("[^a-zA-Z]", " ", x).split() if i not in words]).lower())
    return df
```

marks, Changing text to lower case.

Figure 8

Tokenization of the text data needs to be down Figure 9.

```
[ ]  #Tokenization methods to break down text into tokens(words)
     #Using TF-IDF with ngram range of (1,1)
     tfidf_vectorizer = TfidfVectorizer(min_df= 3, sublinear_tf=True, norm='l2', ngram_range=(1,1))
     #Using Bag of Words with ngram range of (1,1)
     counter_vectorizer = CountVectorizer(ngram_range=(1,1))
```

```
     corpus = []   # X values
     labels = []   # Y values
     for text,label in zip(data["cleaned_reviews"],data["y"]):
       corpus.append(text)
       labels.append(label)
```

Figure 9

For implementing machine learning models below libraries needed to be imported as shown in Fig 10.

```
[ ]  import pandas as pd
     import numpy as np
     from sklearn.feature_extraction.text import CountVectorizer,TfidfTransformer,TfidfVectorizer
     from sklearn.naive_bayes import MultinomialNB
     from sklearn.naive_bayes import ComplementNB
     from sklearn.naive_bayes import BernoulliNB
     from sklearn.ensemble import RandomForestClassifier
     from sklearn.model_selection import GridSearchCV
     from sklearn.linear_model import SGDClassifier
     from sklearn.pipeline import Pipeline
     from sklearn.feature_selection import SelectKBest, chi2
     from nltk.corpus import stopwords
     from nltk.stem import PorterStemmer
     from nltk.tokenize import RegexpTokenizer
     from sklearn.metrics import confusion_matrix, classification_report
     import seaborn as sns
     from sklearn.metrics import confusion_matrix
     from sklearn.metrics import confusion_matrix,accuracy_score,roc_auc_score,roc_curve
     from sklearn.metrics import classification_report
     from yellowbrick.classifier import ClassificationReport
     from sklearn.linear_model import LogisticRegression
     from sklearn.naive_bayes import MultinomialNB
     from sklearn.ensemble import AdaBoostClassifier,RandomForestClassifier
     from sklearn.linear_model import SGDClassifier
     import matplotlib.pyplot as plt
     from sklearn import preprocessing
     import nltk
     import pickle
     nltk.download("stopwords")
```

```
     [nltk_data] Downloading package stopwords to /root/nltk_data...
     [nltk_data]   Unzipping corpora/stopwords.zip.
     True
```

Figure 10

Dataset is divided into test and train set in the ratio of 20: 80. This will help to analyze the ML algorithms easily. Below Fig 11 show the representation

```
[ ] #Splitting Dataset
    from sklearn.model_selection import train_test_split
    X_train_cVec, X_test_cVec, y_train_cVec, y_test_cVec = train_test_split(count_X, labels, test_size=0.2, random_state=42, shuffle= True)
    X_train_tVec, X_test_tVec, y_train_tVec, y_test_tVec = train_test_split(tfidf_X, labels, test_size=0.2, random_state=42, shuffle= True)
    print("\n### Splitting Complete ###\n")

    ### Splitting Complete ###
```

❖ Implementation of Machine Learning Models:

1- MultinomialNB model (MNB):

```
+ Code  + Text

#MultinomialNB
def MNaiveBayes_Count_Vec(X_train,y_train,X_test,y_test):
  global acc1
  clf = MultinomialNB()
  clf.fit(X_train,y_train)
  open('/content/drive/My Drive/Amazon_Product_Sentiment_IE/classifier/mnb_cvec.pkl', 'wb').write(pickle.dumps(clf))
  print("MultinomialNB :train set")
  y_pred = clf.predict(X_train)
  pred=clf.predict_proba(X_test)
  print("MultinomialNB using Count Vectorizer :Confusion Matrix: ", confusion_matrix(y_train, y_pred))
  print ("MultinomialNB using Count Vectorizer :Accuracy : ", accuracy_score(y_train,y_pred)*100)
  print("MultinomialNB :Test set")
  y_pred = clf.predict(X_test)
  print("MultinomialNB using Count Vectorizer :Confusion Matrix: ", confusion_matrix(y_test, y_pred))
  print ("MultinomialNB using Count Vectorizer :Accuracy : ", accuracy_score(y_test,y_pred)*100)
  acc1 =accuracy_score(y_test,y_pred)*100

  #confusion Matrix
  matrix =confusion_matrix(y_test, y_pred)
  class_names=['Negative', 'Neutral','Possitive']
  fig, ax = plt.subplots()
  tick_marks = np.arange(len(class_names))
  plt.xticks(tick_marks, class_names)
  plt.yticks(tick_marks, class_names)
  sns.heatmap(pd.DataFrame(matrix), annot=True, cmap="YlGnBu" ,fmt='g')
  ax.xaxis.set_label_position("top")
  plt.tight_layout()
  plt.title('Confusion matrix', y=1.1)
  plt.ylabel('Actual label')
  plt.xlabel('Predicted label')
  plt.show()


  #Classification Report
  target_names = ['Negative', 'Neutral','Possitive']
  prediction=clf.predict(X_test)
  print(classification_report(y_test, prediction, target_names=target_names))
  classes = ['Negative', 'Neutral','Possitive']
  visualizer = ClassificationReport(clf, classes=classes, support=True)
  visualizer.fit(X_train, y_train)
  visualizer.score(X_test, y_test)
  g = visualizer.poof()
```

Figure 12

2- MLP classifier (Multilayer Perceptron model)

```
#MLP
from sklearn.neural_network import MLPClassifier
import os
def MLP_Count_Vec(X_train,y_train,X_test,y_test):
  global acc2
  clf = MLPClassifier()
  #clf.fit(X_train,y_train)
  clf = pickle.loads(open(os.path.join('/content/drive/My Drive/Amazon_Product_Sentiment_IE/classifier/mlp_cvec.pkl'),'rb').read())
  #open('/content/drive/My Drive/Amazon_Product_Sentiment_IE/classifier/mlp_cvec.pkl', 'wb').write(pickle.dumps(clf))
  print("MLPClassifier :train set")
  y_pred = clf.predict(X_train)
  pred=clf.predict_proba(X_test)
  print("MLPClassifier using Count Vectorizer :Confusion Matrix: ", confusion_matrix(y_train, y_pred))
  print ("MLPClassifier using Count Vectorizer :Accuracy : ", accuracy_score(y_train,y_pred)*100)
  print("MLPClassifier :Test set")
  y_pred = clf.predict(X_test)
  print("MLPClassifier using Count Vectorizer :Confusion Matrix: ", confusion_matrix(y_test, y_pred))
  print ("MLPClassifier using Count Vectorizer :Accuracy : ", accuracy_score(y_test,y_pred)*100)
  acc2 =accuracy_score(y_test,y_pred)*100

  #confusion Matrix
  matrix =confusion_matrix(y_test, y_pred)
  class_names=['Negative', 'Neutral','Possitive']
  fig, ax = plt.subplots()
  tick_marks = np.arange(len(class_names))
  plt.xticks(tick_marks, class_names)
  plt.yticks(tick_marks, class_names)
  sns.heatmap(pd.DataFrame(matrix), annot=True, cmap="YlGnBu" ,fmt='g')
  ax.xaxis.set_label_position("top")
  plt.tight_layout()
  plt.title('Confusion matrix', y=1.1)
  plt.ylabel('Actual label')
  plt.xlabel('Predicted label')
  plt.show()


  #Classification Report
  target_names = ['Negative', 'Neutral','Possitive']
  prediction=clf.predict(X_test)
  print(classification_report(y_test, prediction, target_names=target_names))
  classes = ['Negative', 'Neutral','Possitive']
  visualizer = ClassificationReport(clf, classes=classes, support=True)
  visualizer.fit(X_train, y_train)
  visualizer.score(X_test, y_test)
  g = visualizer.poof()


MLP_Count_Vec(X_train_cVec,y_train_cVec,X_test_cVec,y_test_cVec)
```

3- SGD Classifier Model (sklearn.linear model):

```
from sklearn.linear_model import SGDClassifier

def SGD_Count_Vec(X_train,y_train,X_test,y_test):
  global acc3
  clf = SGDClassifier(alpha=0.00001)
  clf.fit(X_train,y_train)
  open('/content/drive/My Drive/Amazon_Product_Sentiment_IE/classifier/sgd_cvec.pkl', 'wb').write(pickle.dumps(clf))
  print("SGDClassifier :train set")
  y_pred = clf.predict(X_train)
  #pred=clf.predict_proba(X_test)
  print("SGDClassifier using Count Vectorizer :Confusion Matrix: ", confusion_matrix(y_train, y_pred))
  print ("SGDClassifier using Count Vectorizer :Accuracy : ", accuracy_score(y_train,y_pred)*100)
  print("SGDClassifier :Test set")
  y_pred = clf.predict(X_test)
  print("SGDClassifier using Count Vectorizer :Confusion Matrix: ", confusion_matrix(y_test, y_pred))
  print ("SGDClassifier using Count Vectorizer :Accuracy : ", accuracy_score(y_test,y_pred)*100)
  acc3 =accuracy_score(y_test,y_pred)*100

  #confusion Matrix
  matrix =confusion_matrix(y_test, y_pred)
  class_names=['Negative', 'Neutral','Possitive']
  fig, ax = plt.subplots()
  tick_marks = np.arange(len(class_names))
  plt.xticks(tick_marks, class_names)
  plt.yticks(tick_marks, class_names)
  sns.heatmap(pd.DataFrame(matrix), annot=True, cmap="YlGnBu" ,fmt='g')
  ax.xaxis.set_label_position("top")
  plt.tight_layout()
  plt.title('Confusion matrix', y=1.1)
  plt.ylabel('Actual label')
  plt.xlabel('Predicted label')
  plt.show()


  #Classification Report
  target_names = ['Negative', 'Neutral','Possitive']
  prediction=clf.predict(X_test)
  print(classification_report(y_test, prediction, target_names=target_names))
  classes = ['Negative', 'Neutral','Possitive']
  visualizer = ClassificationReport(clf, classes=classes, support=True)
  visualizer.fit(X_train, y_train)
  visualizer.score(X_test, y_test)
  g = visualizer.poof()


SGD_Count_Vec(X_train_cVec,y_train_cVec,X_test_cVec,y_test_cVec)
```

Results:

The visualization of output is done by confusion matrix (Fig 14) and classifier classification matrix (Fig 15)

```
MLPClassifier :train set
MLPClassifier using Count Vectorizer :Confusion Matrix:  [[ 90439  18281   5085]
 [ 13770  88128  11990]
 [  2710   7944 103069]]
MLPClassifier using Count Vectorizer :Accuracy :  82.49056869039529
MLPClassifier :Test set
MLPClassifier using Count Vectorizer :Confusion Matrix:  [[19047  7120  2285]
 [ 5771 17568  5030]
 [ 1347  3730 23457]]
MLPClassifier using Count Vectorizer :Accuracy :  70.37900533067776
```

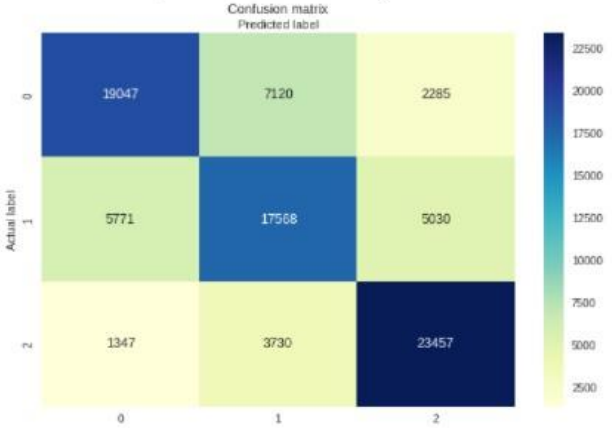


Figure 14: Confusion Matrix

Negative	0.73	0.67	0.70	28452
Neutral	0.62	0.62	0.62	28369
Possitive	0.76	0.82	0.79	28534
accuracy			0.70	85355
macro avg	0.70	0.70	0.70	85355
weighted avg	0.70	0.70	0.70	85355

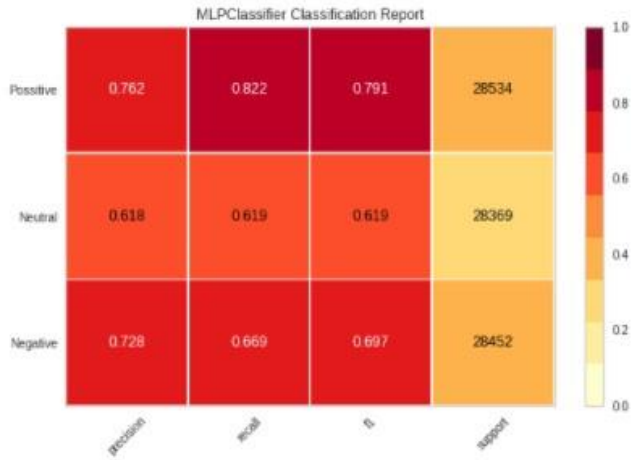


Figure: 15: Classification report

References:

Pankaj, Pandey, P., Muskan and Soni, N. (2019). Sentiment analysis on customer feedback data: Amazon product reviews, *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)*, pp. 320–322.