Configuration Manual

MSc Research Project
MS in Data Analytics

Manisha
Student ID: x21207194

School of Computing
National College of Ireland

Supervisor: Paul Stynes, William Clifford, Eugene McLaughlin

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Manisha
Student ID:	x21207194
Programme:	MS in Data Analytics
Year:	2023
Module:	MSc Research Project
Supervisor:	Paul Stynes, William Clifford, Eugene McLaughlin
Submission Due Date:	14/08/2023
Project Title:	Configuration Manual
Word Count:	534
Page Count:	5

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:	Manisha
Date:	18th September 2023

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

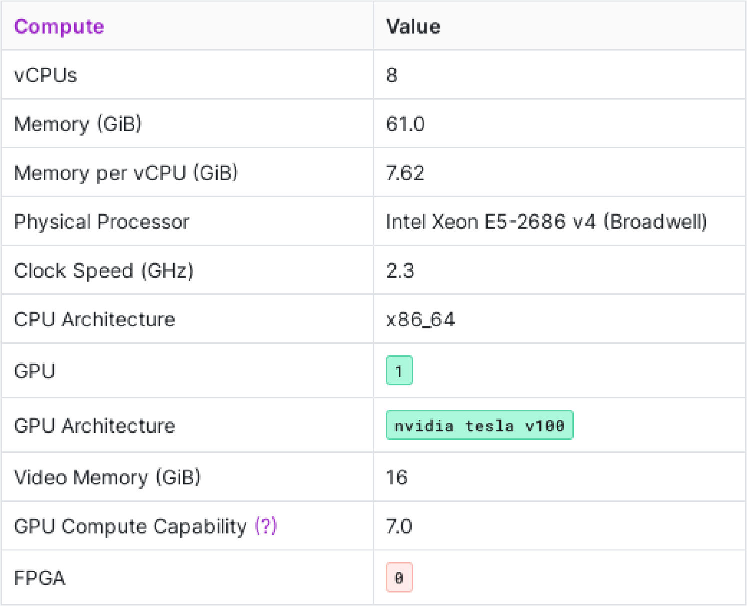
Manisha
x21207194

1 Introduction

This configuration manual lists all hardware and software requirements to replicate the results of the research. A step-by-step guide taken from data acquisition to model implementation are described in this manual.

2 Hardware and Software Configurations

Fine tuning pre-trained models is computationally intensive task and requires high execution time. Using a GPU (Graphics Processing Unit) is well suited for the task. For this research Deep Learning AMI GPU TensorFlow 2.13 (Ubuntu 20.04) 20230724 amazon machine image (AMI) with p3.2xlarge is used. Figure 1 shows the aws EC2 instance details required for the experiment.



Compute	Value
vCPUs	8
Memory (GiB)	61.0
Memory per vCPU (GiB)	7.62
Physical Processor	Intel Xeon E5-2686 v4 (Broadwell)
Clock Speed (GHz)	2.3
CPU Architecture	x86_64
GPU	1
GPU Architecture	nvidia tesla v100
Video Memory (GiB)	16
GPU Compute Capability (?)	7.0
FPGA	0

Figure 1: EC2 Instance Details

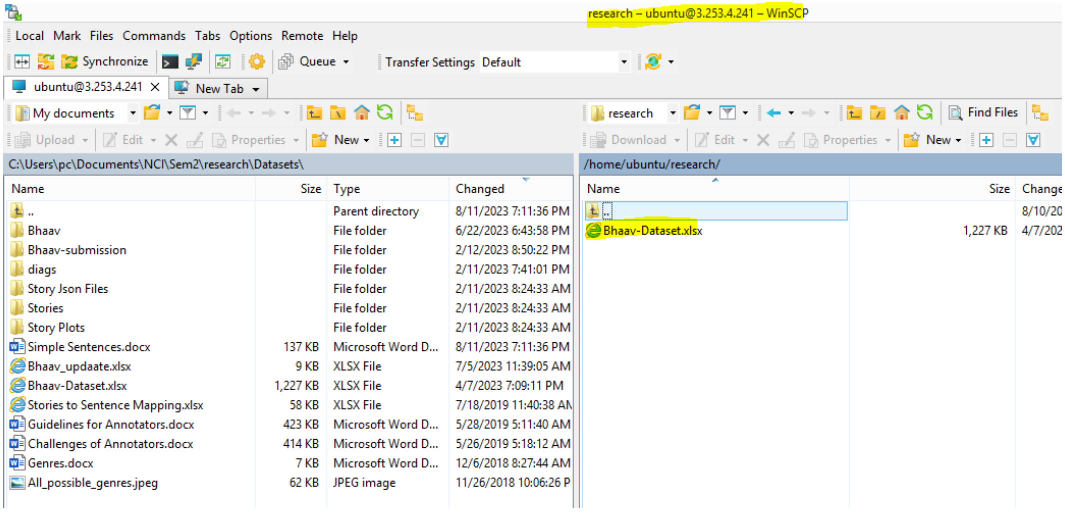
For the experiment 'Python' is used as the programming language. Python allows writing simple and readable code. It provides extensive set of libraries and frameworks for training machine learning (ML) and deep learning (DL) models. Table 1 details the libraries used and their respective versions.

Library	Version
python	3.10.12
transformers	4.31.0
tensorflow	2.13.0
pandas	2.0.3
openpyxl	3.1.2
indic-nlp-library	0.92
numpy	1.24.3
tensorflow	2.13.0
fasttext	0.9.2
scikit-learn	1.3.0

Table 1: Software specifications

3 Dataset acquisition

The BHAAV dataset used in the research was created by (Kumar et al.; 2019) and is publicly available as an open dataset. This dataset can be directly downloaded from https://zenodo.org/record/3457467. Post download, the dataset should be transferred to EC2 using Winscp by logging in the server details. Figure 2shows transfer of data from local to EC2 server. After placing the input excel file on server the datasets can be read in pandas dataframe using python openpyxl and pandas library for further processing.



research – ubuntu@3.253.4.241 – WinSCP

Transfer Settings Default

ubuntu@3.253.4.241

C:\Users\pc\Documents\NCI\Sem2\research\Datasets\

Name	Size	Type	Changed
..		Parent directory	8/11/2023 7:11:36 PM
Bhaav		File folder	6/22/2023 6:43:58 PM
Bhaav-submission		File folder	2/12/2023 8:50:22 PM
diags		File folder	2/11/2023 7:41:01 PM
Story Json Files		File folder	2/11/2023 8:24:33 AM
Stories		File folder	2/11/2023 8:24:33 AM
Story Plots		File folder	2/11/2023 8:24:33 AM
Simple Sentences.docx	137 KB	Microsoft Word D...	8/11/2023 7:11:36 PM
Bhaav_updaate.xlsx	9 KB	XLSX File	7/5/2023 11:39:05 AM
Bhaav-Dataset.xlsx	1,227 KB	XLSX File	4/7/2023 7:09:11 PM
Stories to Sentence Mapping.xlsx	58 KB	XLSX File	7/18/2019 11:40:38 AM
Guidelines for Annotators.docx	423 KB	Microsoft Word D...	5/28/2019 5:11:40 AM
Challenges of Annotators.docx	414 KB	Microsoft Word D...	5/26/2019 5:18:12 AM
Genres.docx	7 KB	Microsoft Word D...	12/6/2018 8:27:44 AM
All_possible_genres.jpeg	62 KB	JPEG image	11/26/2018 10:06:26 P

/home/ubuntu/research/

Name	Size	Change
..		8/10/20
Bhaav-Dataset.xlsx	1,227 KB	4/7/202

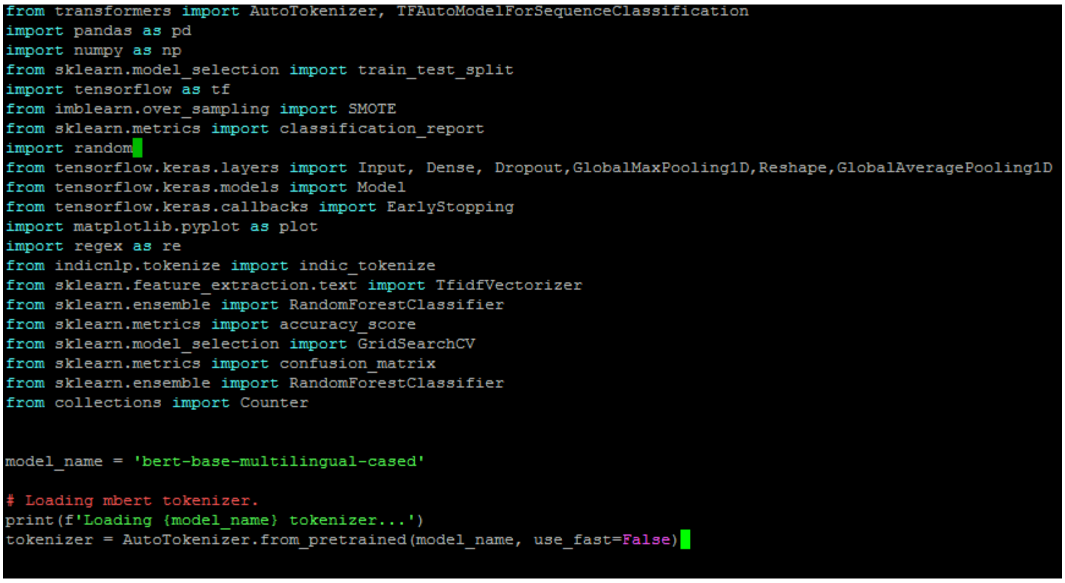
Figure 2: Importing the necessary python libraries

4 Project development

Only the most crucial, necessary steps have been discussed in this section.

4.1 Importing the required libraries

Figure 3 shows the number of packages and libraries imported for the emotion classification task.In addition, tokenizers from different pre-trained transformer models were loaded for the transfer based learning.



```
from transformers import AutoTokenizer, TFAutoModelForSequenceClassification
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
import tensorflow as tf
from imblearn.over_sampling import SMOTE
from sklearn.metrics import classification_report
import random
from tensorflow.keras.layers import Input, Dense, Dropout,GlobalMaxPooling1D,Reshape,GlobalAveragePooling1D
from tensorflow.keras.models import Model
from tensorflow.keras.callbacks import EarlyStopping
import matplotlib.pyplot as plot
import regex as re
from indicnlp.tokenize import indic_tokenize
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import confusion_matrix
from sklearn.ensemble import RandomForestClassifier
from collections import Counter


model_name = 'bert-base-multilingual-cased'

# Loading mbert tokenizer.
print(f'Loading {model_name} tokenizer...')
tokenizer = AutoTokenizer.from_pretrained(model_name, use_fast=False)
```

Figure 3: Importing the necessary python libraries

4.2 Downloading word embedding model for deep learning models

In the study, deep learning models use Fasttext word embedding for feature representation of texts. Fasttext embedding model trained on Hindi Wikipedia corpus should be downloaded from https://fasttext.cc/docs/en/crawl-vectors.html and placed on the server along with the input file before running the models.

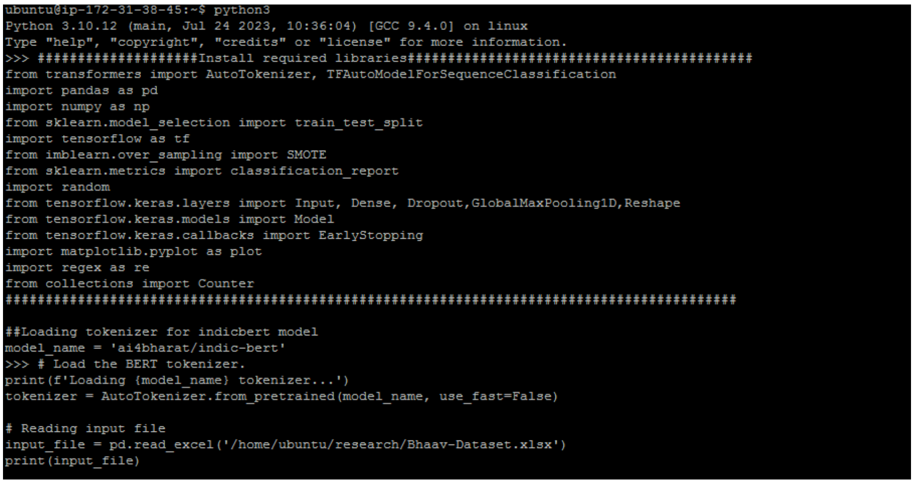
4.3 Code Execution

Separate scripts for each ML and DL models have been provided in the code artifact folder. Post installing the required libraries code can be run line-by-line after opening a python session on the server. Figure 4 shows how to open a python session on the server terminal and execute code.

4.4 Model Implementation

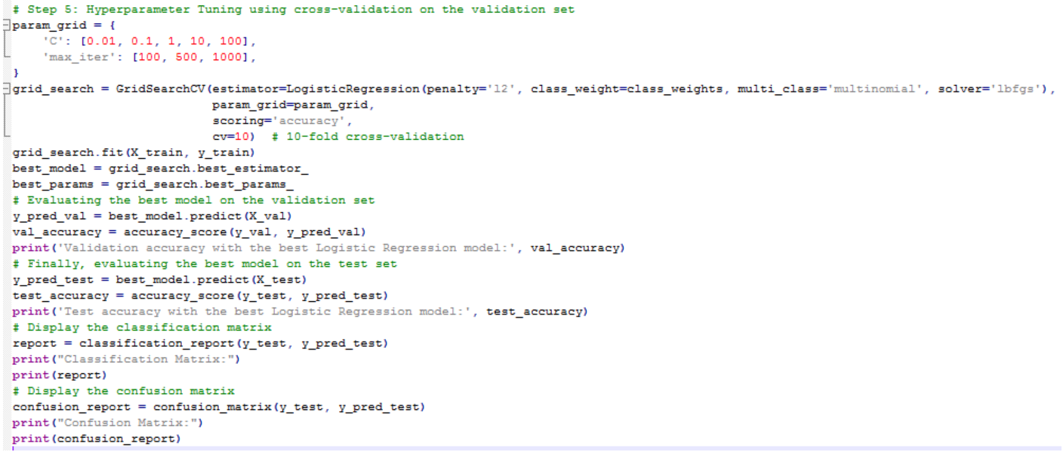
The research is conducted with three machine learning algorithms, 2 deep learning models and 3 pre-trained models on the single dataset. The algorithms implemented in the project are Random Forest, Logistic Regression, Support Vector Machine, CNN, BiLSTM, mBERT, IndicBERT and XLM-Roberta. Figure 5 and 6 shows example implementation for Logistic regression and CNN model.

For transformer models, the models have been loaded from HuggingFace transformers library. early_stopping_callback with patience =3 is applied which monitors the validation loss score and stops training when no improvement is observed. Figure 7 shows model configuration for XLM-Roberta model.



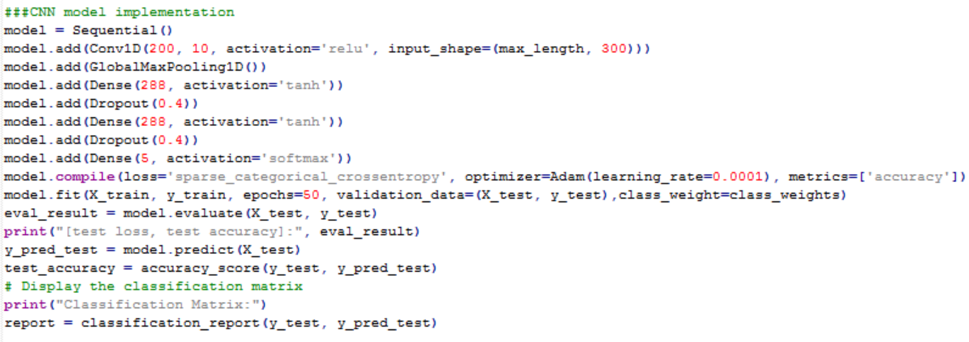
```
ubuntu@ip-172-31-38-45:~$ python3
Python 3.10.12 (main, Jul 24 2023, 10:36:04) [GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> ###################Install required libraries##########################################
from transformers import AutoTokenizer, TFAutoModelForSequenceClassification
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
import tensorflow as tf
from imblearn.over_sampling import SMOTE
from sklearn.metrics import classification_report
import random
from tensorflow.keras.layers import Input, Dense, Dropout,GlobalMaxPooling1D,Reshape
from tensorflow.keras.models import Model
from tensorflow.keras.callbacks import EarlyStopping
import matplotlib.pyplot as plot
import regex as re
from collections import Counter
###########################################################################################

##Loading tokenizer for indicbert model
model_name = 'ai4bharat/indic-bert'
>>> # Load the BERT tokenizer.
print(f'Loading {model_name} tokenizer...')
tokenizer = AutoTokenizer.from_pretrained(model_name, use_fast=False)

# Reading input file
input_file = pd.read_excel('/home/ubuntu/research/Bhaav-Dataset.xlsx')
print(input_file)
```

Figure 4: Execution step



```
# Step 5: Hyperparameter Tuning using cross-validation on the validation set
param_grid = {
    'C': [0.01, 0.1, 1, 10, 100],
    'max_iter': [100, 500, 1000],
}
grid_search = GridSearchCV(estimator=LogisticRegression(penalty='l2', class_weight=class_weights, multi_class='multinomial', solver='lbfgs'),
                           param_grid=param_grid,
                           scoring='accuracy',
                           cv=10)  # 10-fold cross-validation
grid_search.fit(X_train, y_train)
best_model = grid_search.best_estimator_
best_params = grid_search.best_params_
# Evaluating the best model on the validation set
y_pred_val = best_model.predict(X_val)
val_accuracy = accuracy_score(y_val, y_pred_val)
print('Validation accuracy with the best Logistic Regression model:', val_accuracy)
# Finally, evaluating the best model on the test set
y_pred_test = best_model.predict(X_test)
test_accuracy = accuracy_score(y_test, y_pred_test)
print('Test accuracy with the best Logistic Regression model:', test_accuracy)
# Display the classification matrix
report = classification_report(y_test, y_pred_test)
print("Classification Matrix:")
print(report)
# Display the confusion matrix
confusion_report = confusion_matrix(y_test, y_pred_test)
print("Confusion Matrix:")
print(confusion_report)
```

Figure 5: Logistic Regression Implementation

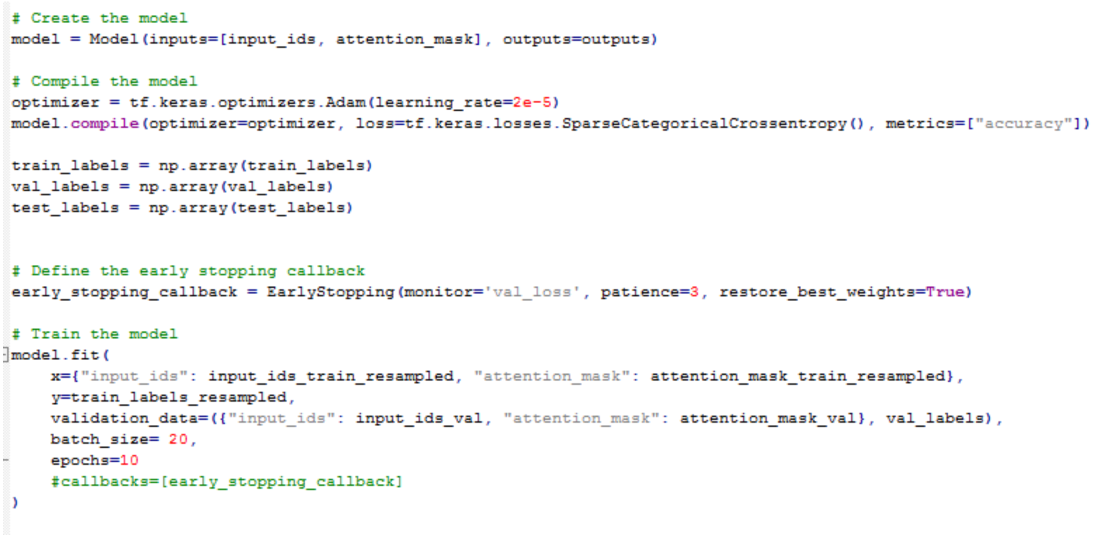


```
###CNN model implementation
model = Sequential()
model.add(Conv1D(200, 10, activation='relu', input_shape=(max_length, 300)))
model.add(GlobalMaxPooling1D())
model.add(Dense(288, activation='tanh'))
model.add(Dropout(0.4))
model.add(Dense(288, activation='tanh'))
model.add(Dropout(0.4))
model.add(Dense(5, activation='softmax'))
model.compile(loss='sparse_categorical_crossentropy', optimizer=Adam(learning_rate=0.0001), metrics=['accuracy'])
model.fit(X_train, y_train, epochs=50, validation_data=(X_test, y_test),class_weight=class_weights)
eval_result = model.evaluate(X_test, y_test)
print("[test loss, test accuracy]:", eval_result)
y_pred_test = model.predict(X_test)
test_accuracy = accuracy_score(y_test, y_pred_test)
# Display the classification matrix
print("Classification Matrix:")
report = classification_report(y_test, y_pred_test)
```

Figure 6: CNN Implementation

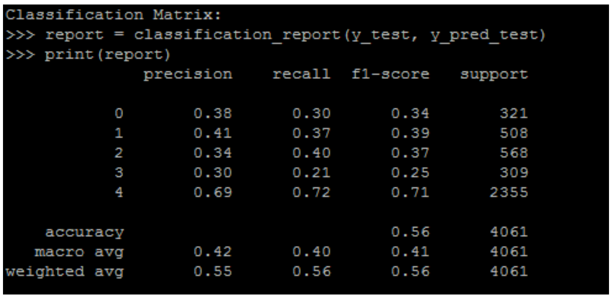
4.5 Model Evaluation

For evaluating model performance macro averaged precision, recall, F1 score and accuracy is inferred from the classification matrix. In addition, confusion matrix is used to assess the validity of the model. Figure 8 and 9 shows the classification and confusion matrix for Logistic Regression model.



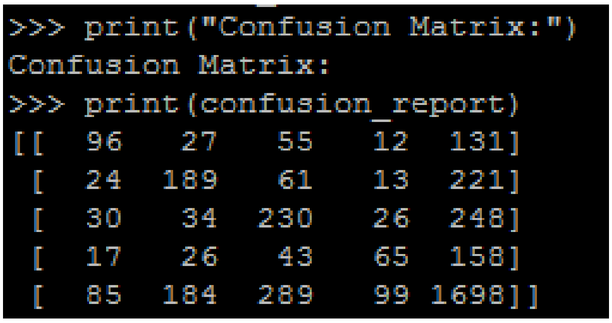
```
# Create the model
model = Model(inputs=[input_ids, attention_mask], outputs=outputs)

# Compile the model
optimizer = tf.keras.optimizers.Adam(learning_rate=2e-5)
model.compile(optimizer=optimizer, loss=tf.keras.losses.SparseCategoricalCrossentropy(), metrics=["accuracy"])

train_labels = np.array(train_labels)
val_labels = np.array(val_labels)
test_labels = np.array(test_labels)


# Define the early stopping callback
early_stopping_callback = EarlyStopping(monitor='val_loss', patience=3, restore_best_weights=True)

# Train the model
model.fit(
    x={"input_ids": input_ids_train_resampled, "attention_mask": attention_mask_train_resampled},
    y=train_labels_resampled,
    validation_data=({"input_ids": input_ids_val, "attention_mask": attention_mask_val}, val_labels),
    batch_size= 20,
    epochs=10
    #callbacks=[early_stopping_callback]
)
```

Figure 7: XLM-Roberta Implementation



```
Classification Matrix:
>>> report = classification_report(y_test, y_pred_test)
>>> print(report)
              precision    recall  f1-score   support

           0       0.38      0.30      0.34       321
           1       0.41      0.37      0.39       508
           2       0.34      0.40      0.37       568
           3       0.30      0.21      0.25       309
           4       0.69      0.72      0.71      2355

    accuracy                           0.56      4061
   macro avg       0.42      0.40      0.41      4061
weighted avg       0.55      0.56      0.56      4061
```

Figure 8: Classification Matrix for Logistic Regression



```
>>> print("Confusion Matrix:")
Confusion Matrix:
>>> print(confusion_report)
[[  96   27   55   12  131]
 [  24  189   61   13  221]
 [  30   34  230   26  248]
 [  17   26   43   65  158]
 [  85  184  289   99 1698]]
```

Figure 9: Confusion Matrix for Logistic Regression

References

Kumar, Y., Mahata, D., Aggarwal, S., Chugh, A., Maheshwari, R. and Shah, R. R. (2019). Bhaav-a text corpus for emotion analysis from hindi stories, *arXiv preprint arXiv:1910.04073* .