Configuration Manual

MSc Research Project
MSc. Data Analytics

Raj Ravindra Kupekar
Student ID: X18186432

School of Computing
National College of Ireland

Supervisor: Prof. Christian Horn

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: Raj Ravindra Kupekar

Student ID: X18186432

Programme: MSc. Data Analytics **Year:** 2020

Module: Research Project

Lecturer: Prof. Christian Horn

Submission Due Date: 28/09/2020

Project Title: Conversational Emotion Recognition using Text and Audio Modalities

Word Count: 448 **Page Count:** 8

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: Raj Ravindra Kupekar

Date: 28/09/2020

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.	□

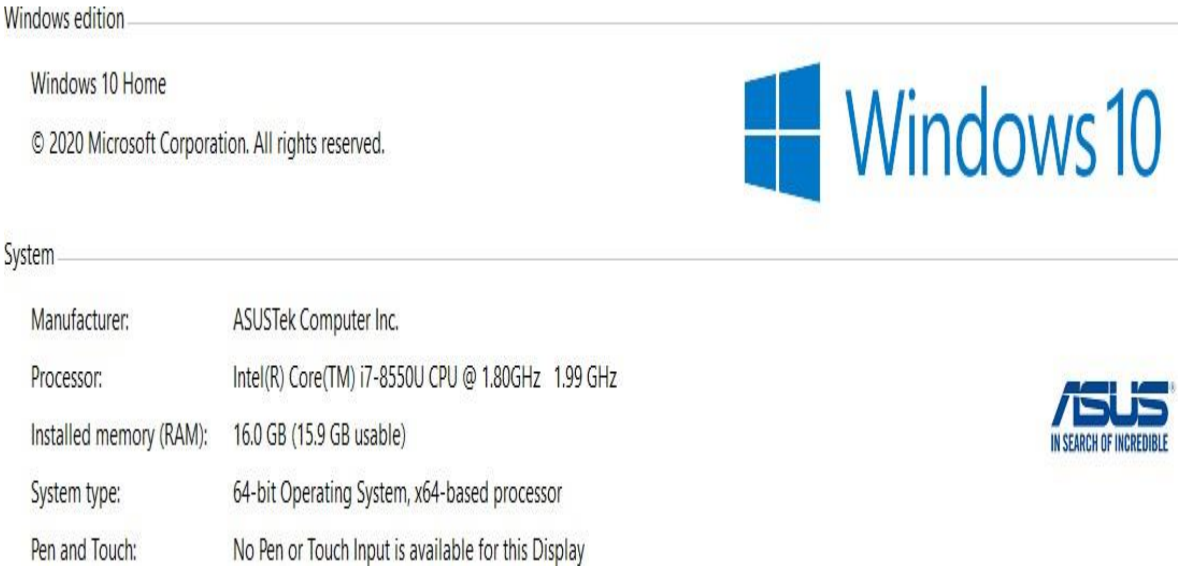
Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

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

Configuration Manual

Raj Ravindra Kupekar
X18186432

1 Hardware Setup



Windows edition

Windows 10 Home

© 2020 Microsoft Corporation. All rights reserved.

System

Manufacturer: ASUSTek Computer Inc.

Processor: Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz 1.99 GHz

Installed memory (RAM): 16.0 GB (15.9 GB usable)

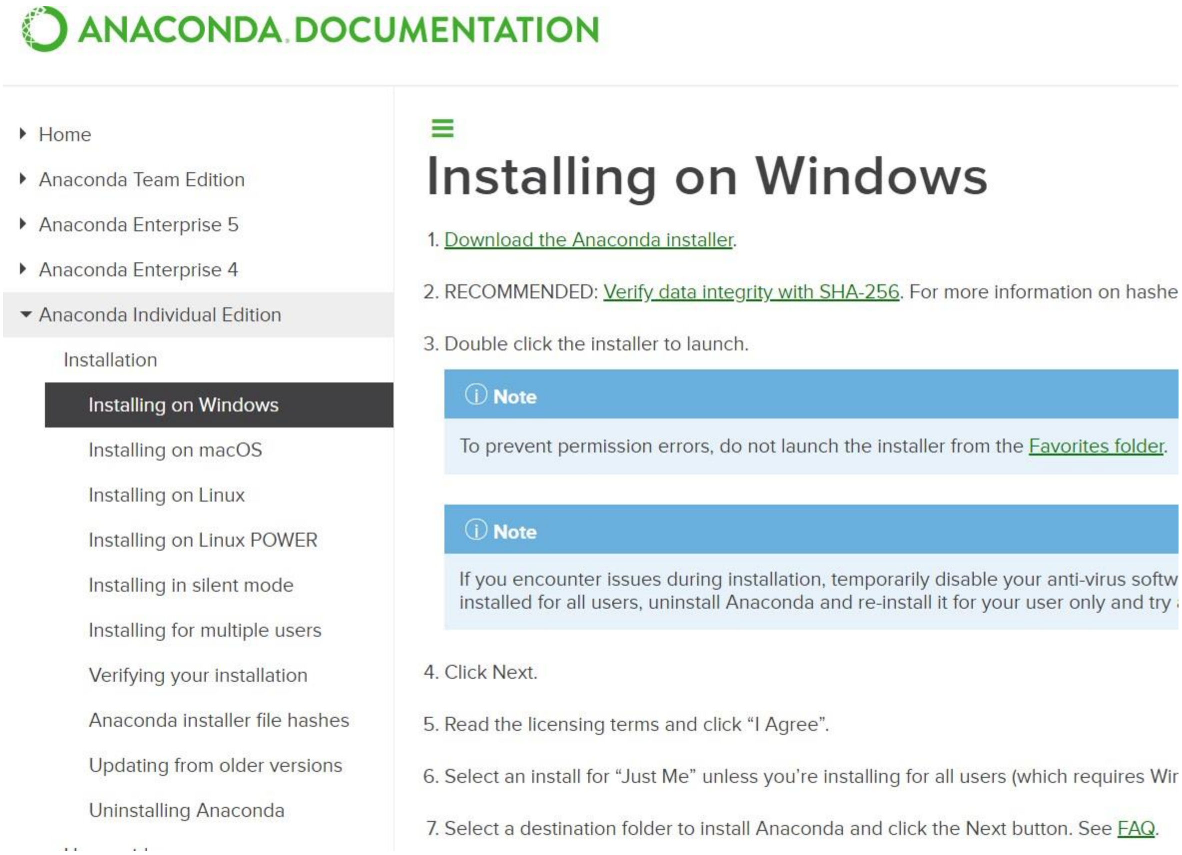
System type: 64-bit Operating System, x64-based processor

Pen and Touch: No Pen or Touch Input is available for this Display

Fig. 1 Computer Specification

The specification of the machine used in this research project is illustrated in above figure. It is having an installed RAM of 16 GB with 64-bit Operating System and an installed Windows 10 configuration. The processor of the machine is Intel® Core ™ i7-8550U CPU with Intel ® UHD Graphics 620.

2 Software Information


ANACONDA.DOCUMENTATION
Home
Anaconda Team Edition
Anaconda Enterprise 5
Anaconda Enterprise 4
Anaconda Individual Edition
Installation
Installing on Windows
Installing on macOS
Installing on Linux
Installing on Linux POWER
Installing in silent mode
Installing for multiple users
Verifying your installation
Anaconda installer file hashes
Updating from older versions
Uninstalling Anaconda

Installing on Windows

1. Download the Anaconda installer.

2. RECOMMENDED: Verify data integrity with SHA-256. For more information on hashe

3. Double click the installer to launch.

ⓘ Note

To prevent permission errors, do not launch the installer from the Favorites folder.

ⓘ Note

If you encounter issues during installation, temporarily disable your anti-virus softw
installed for all users, uninstall Anaconda and re-install it for your user only and try

4. Click Next.

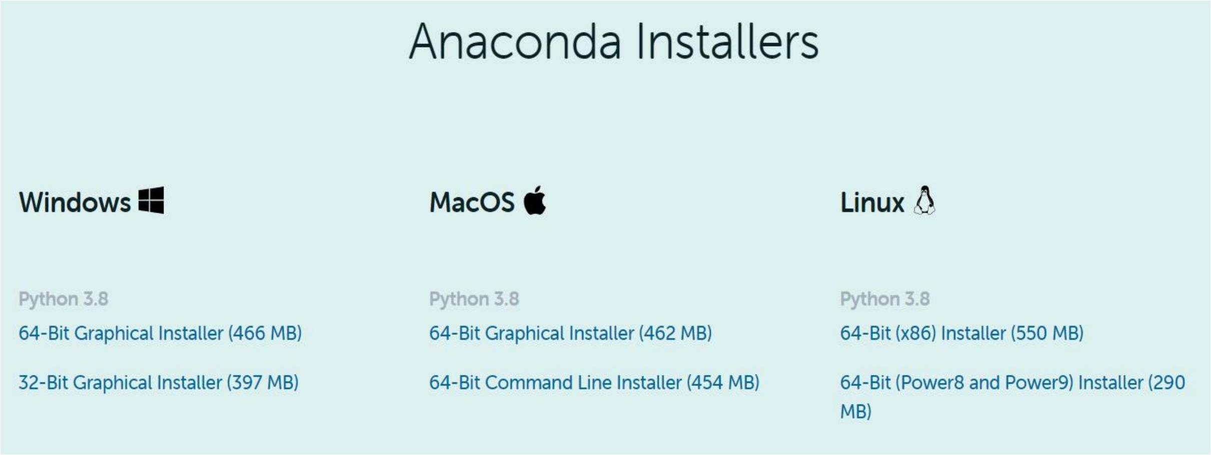
5. Read the licensing terms and click "I Agree".

6. Select an install for "Just Me" unless you're installing for all users (which requires Wir

7. Select a destination folder to install Anaconda and click the Next button. See FAQ.

Fig 2.1 Anaconda Software for Python

In this research project, an Anaconda software [2] is installed on machine for accessing the Jupyter environment.



Anaconda Installers

Windows	MacOS	Linux
Python 3.8	Python 3.8	Python 3.8
64-Bit Graphical Installer (466 MB)	64-Bit Graphical Installer (462 MB)	64-Bit (x86) Installer (550 MB)
32-Bit Graphical Installer (397 MB)	64-Bit Command Line Installer (454 MB)	64-Bit (Power8 and Power9) Installer (290 MB)

Fig 2.2 Anaconda Installer

Accordingly, the anaconda can be installed for each operating system with the proper operating bit. For this work, a windows 64-Bit anaconda installed

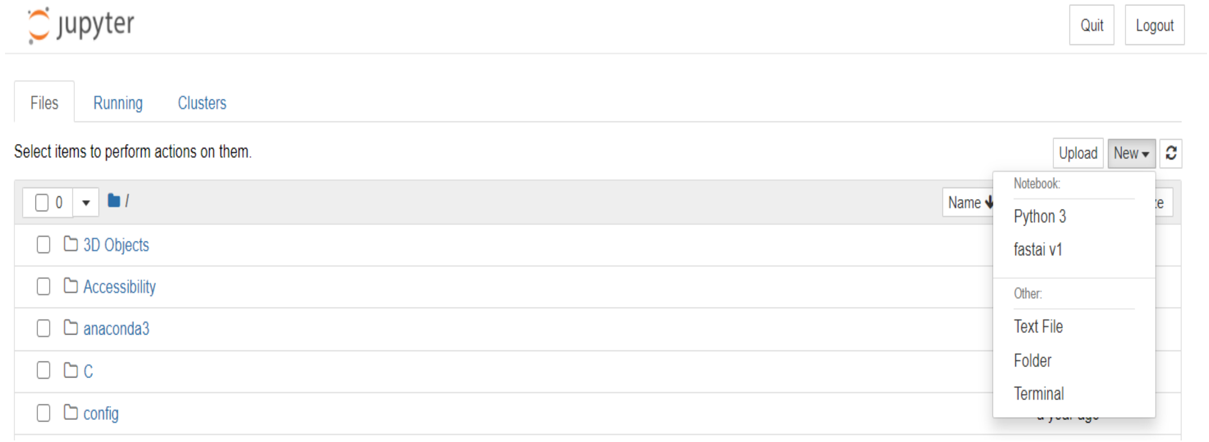
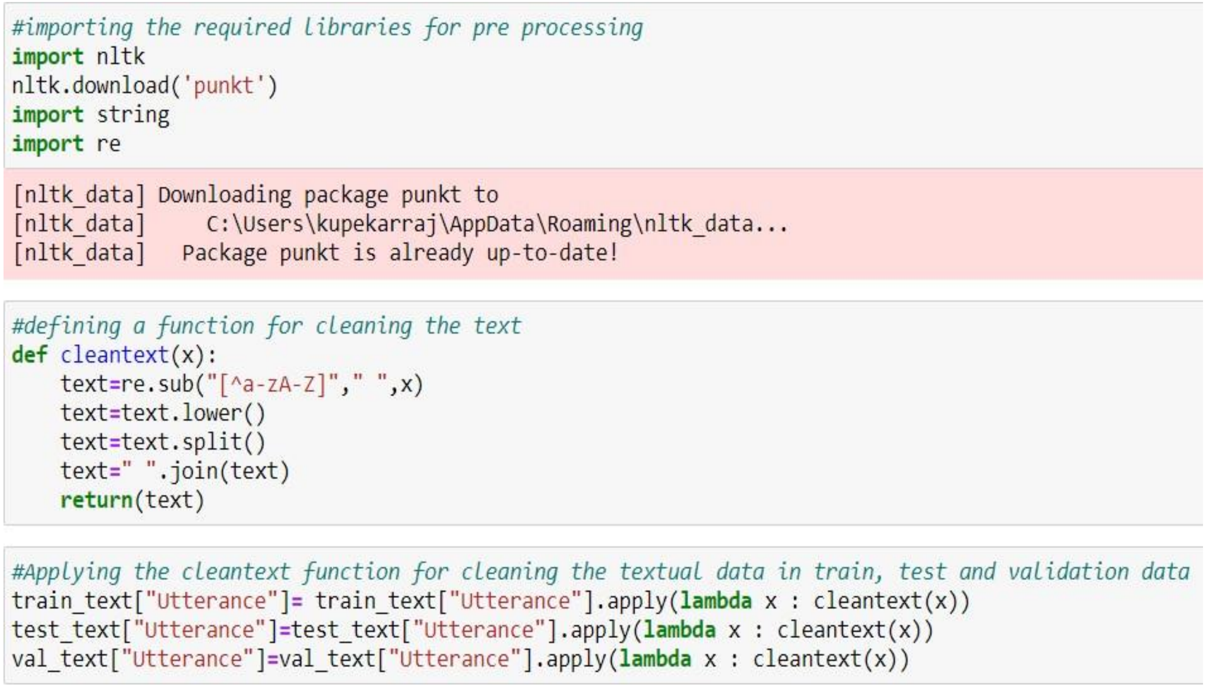


Fig 2.3 Jupyter Environment

A Jupyter notebook can be created by clicking the Python 3 option from new. Accordingly, in this work two separate notebooks are created for text and audio.

3 Data Preparation

Here the data preparation of text and audio modality are been carried out using the appropriate libraries.



```
#importing the required libraries for pre processing
import nltk
nltk.download('punkt')
import string
import re
```

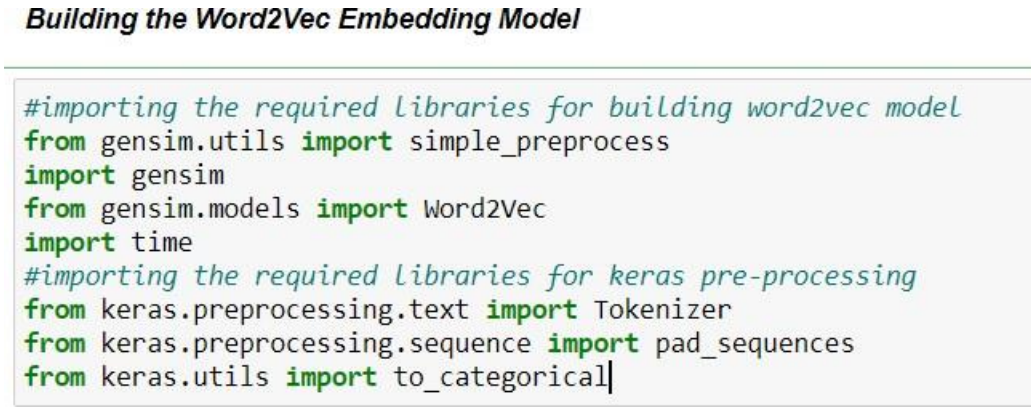
```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\kupekarraj\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

```
#defining a function for cleaning the text
def cleantext(x):
    text=re.sub("[^a-zA-Z]"," ",x)
    text=text.lower()
    text=text.split()
    text=" ".join(text)
    return(text)
```

```
#Applying the cleantext function for cleaning the textual data in train, test and validation data
train_text["Utterance"]= train_text["Utterance"].apply(lambda x : cleantext(x))
test_text["Utterance"]=test_text["Utterance"].apply(lambda x : cleantext(x))
val_text["Utterance"]=val_text["Utterance"].apply(lambda x : cleantext(x))
```

Fig 3.1 Pre-processing libraries for text

The above figure shows the pre-processing libraries for text modalities. Accordingly, the nltk libraries are used for pre-processing. The regular expression (re) libraries are used to remove the non-alphabetical words from the raw text.

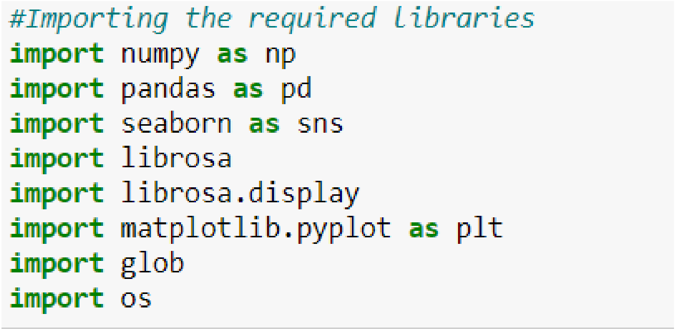


Building the Word2Vec Embedding Model

```
#importing the required libraries for building word2vec model
from gensim.utils import simple_preprocess
import gensim
from gensim.models import Word2Vec
import time
#importing the required libraries for keras pre-processing
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.utils import to_categorical
```

Fig 3.2 Feature extraction libraries for text

For text, the features are extracted using the above displayed libraries. The genism library is used for implemented the Word2Vec vector representation of the pre-processed text. The keras libraries are also used for the additional preprocessing of the of the processed text. Accordingly, the to_categorical library is used for converting the dependent variable into a single array matrix.



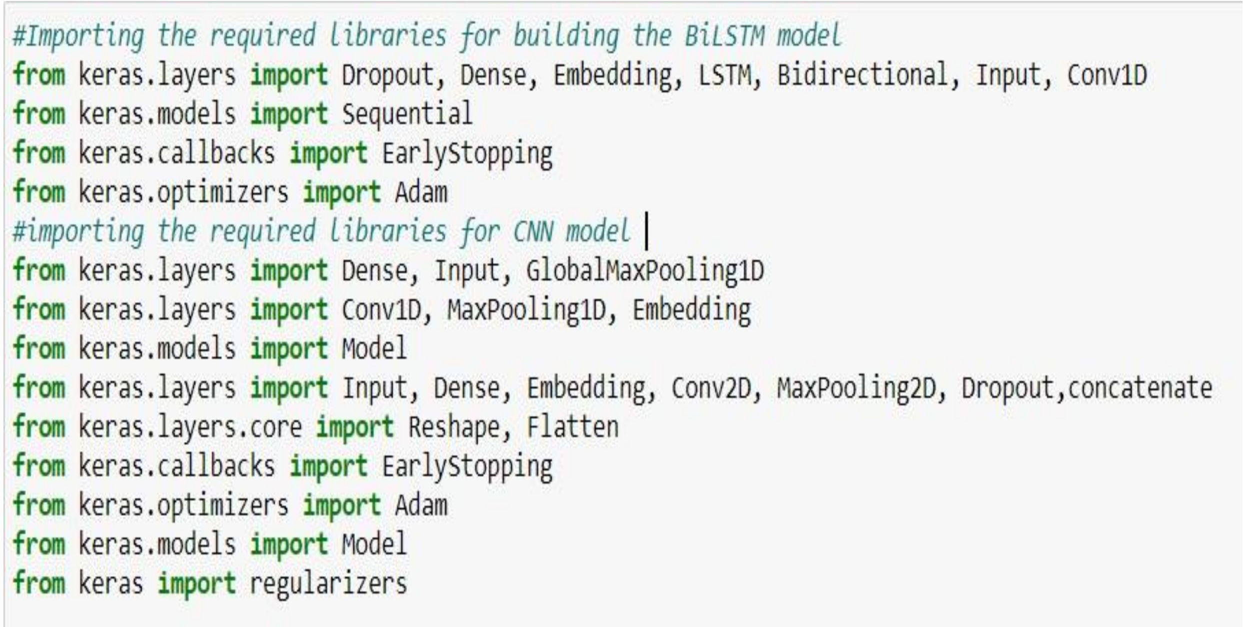
```
#Importing the required libraries
import numpy as np
import pandas as pd
import seaborn as sns
import librosa
import librosa.display
import matplotlib.pyplot as plt
import glob
import os
```

Fig 3.3 Pre-processing and feature extraction libraries for audio

Similarly, the audio files are pre-processed, and feature extracted using a Librosa library. The Librosa library comes with an MFCC function for extracting the MFCC coefficients and Librosa display for plotting the image representation of MFCCs.

4 Implementation

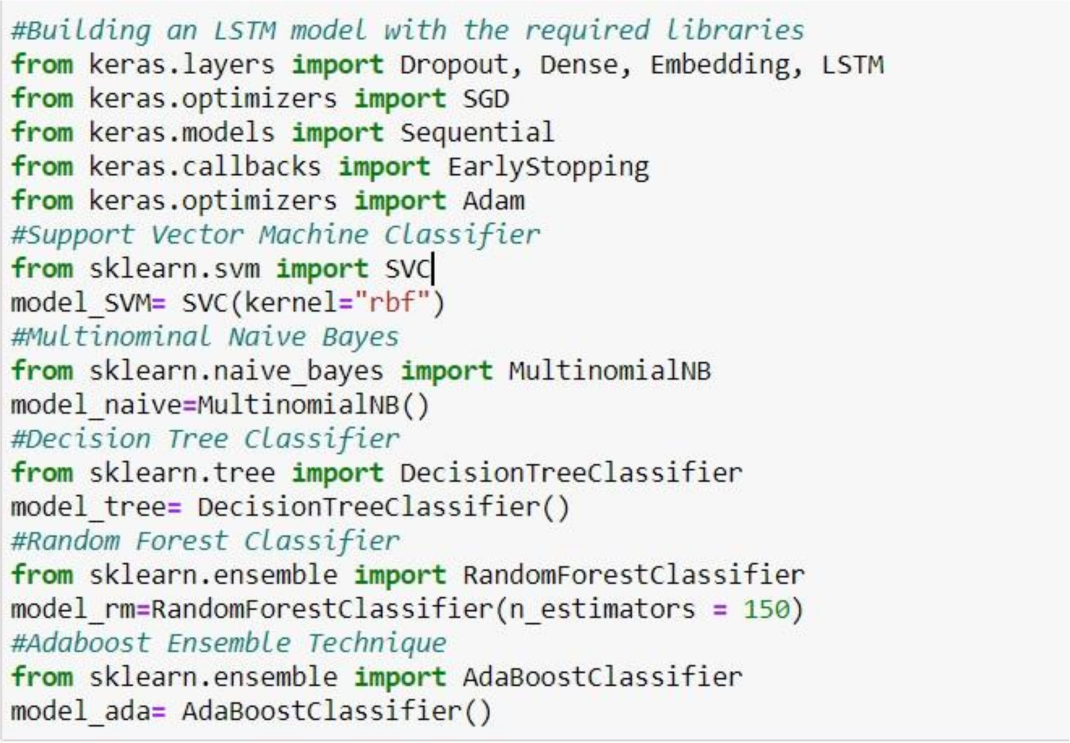
In this section, the required libraries for the model implementation are been displayed. For model building of text, a Bi-LSTM and a CNN model are built using the keras libraries displayed in Fig 4.1.



```
#Importing the required libraries for building the BiLSTM model
from keras.layers import Dropout, Dense, Embedding, LSTM, Bidirectional, Input, Conv1D
from keras.models import Sequential
from keras.callbacks import EarlyStopping
from keras.optimizers import Adam
#importing the required libraries for CNN model
from keras.layers import Dense, Input, GlobalMaxPooling1D
from keras.layers import Conv1D, MaxPooling1D, Embedding
from keras.models import Model
from keras.layers import Input, Dense, Embedding, Conv2D, MaxPooling2D, Dropout,concatenate
from keras.layers.core import Reshape, Flatten
from keras.callbacks import EarlyStopping
from keras.optimizers import Adam
from keras.models import Model
from keras import regularizers
```

Fig 4.1 Required libraries for model building of text

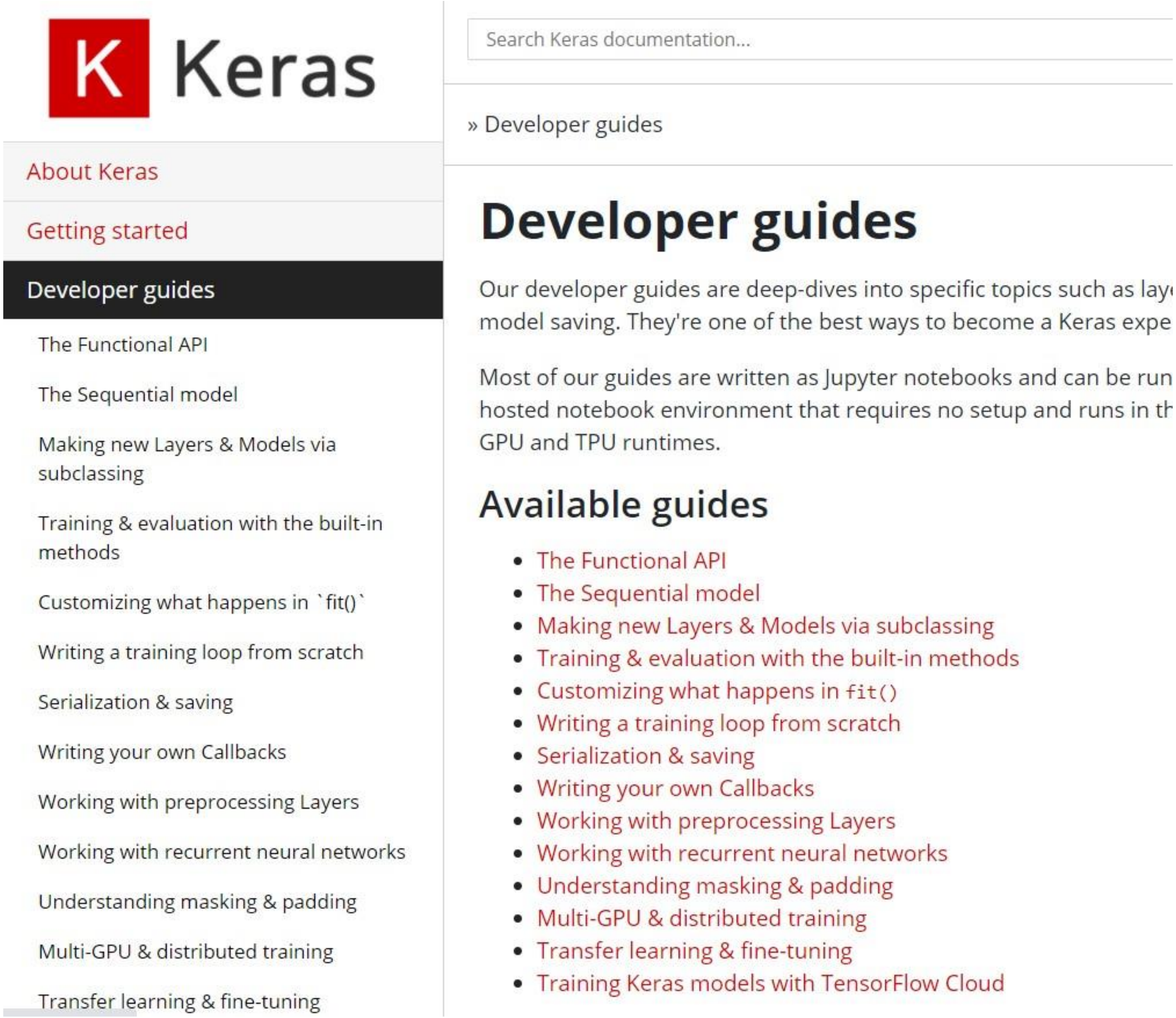
Accordingly, the LSTM and baseline machine learning models are implemented for audio emotion recognition. The following figure shows the required libraries for model building of audio analysis.



```
#Building an LSTM model with the required libraries
from keras.layers import Dropout, Dense, Embedding, LSTM
from keras.optimizers import SGD
from keras.models import Sequential
from keras.callbacks import EarlyStopping
from keras.optimizers import Adam
#Support Vector Machine Classifier
from sklearn.svm import SVC
model_SVM= SVC(kernel="rbf")
#Multinominal Naive Bayes
from sklearn.naive_bayes import MultinomialNB
model_naive=MultinomialNB()
#Decision Tree Classifier
from sklearn.tree import DecisionTreeClassifier
model_tree= DecisionTreeClassifier()
#Random Forest Classifier
from sklearn.ensemble import RandomForestClassifier
model_rm=RandomForestClassifier(n_estimators = 150)
#Adaboost Ensemble Technique
from sklearn.ensemble import AdaBoostClassifier
model_ada= AdaBoostClassifier()
```

Fig 4.2 Required libraries for model building of audio

5 Keras for deep learning



Keras
Search Keras documentation...
» Developer guides
About Keras
Getting started
Developer guides
The Functional API
The Sequential model
Making new Layers & Models via subclassing
Training & evaluation with the built-in methods
Customizing what happens in `fit()`
Writing a training loop from scratch
Serialization & saving
Writing your own Callbacks
Working with preprocessing Layers
Working with recurrent neural networks
Understanding masking & padding
Multi-GPU & distributed training
Transfer learning & fine-tuning

Developer guides

Our developer guides are deep-dives into specific topics such as lay
model saving. They're one of the best ways to become a Keras expe

Most of our guides are written as Jupyter notebooks and can be run
hosted notebook environment that requires no setup and runs in th
GPU and TPU runtimes.

Available guides

- The Functional API
- The Sequential model
- Making new Layers & Models via subclassing
- Training & evaluation with the built-in methods
- Customizing what happens in `fit()`
- Writing a training loop from scratch
- Serialization & saving
- Writing your own Callbacks
- Working with preprocessing Layers
- Working with recurrent neural networks
- Understanding masking & padding
- Multi-GPU & distributed training
- Transfer learning & fine-tuning
- Training Keras models with TensorFlow Cloud

Fig 5. Keras Documentation

The keras model are built by following the guidelines from the Keras documentation [1] on the given official site. The documentation covers the model building for both Bi-LSTM, CNN, and LSTM models along with the hyperparameters and optimizer information.

References

[1] https://keras.io/guides/

[2] https://docs.anaconda.com/anaconda/install/windows/