National College of Ireland

Configuration Manual

MSc Research Project
MSC in Data Analytics

Kunal Chauhan
Student ID: 21139245

School of Computing
National College of Ireland

Supervisor: Qurrat Ul Ain

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: ……Kunal Chauhan. ……

Student ID: ……21139245…….. ……..……

Programme: …… **Year:** ……..

Module: ……..……

Lecturer: ……Qurrat Ul Ain……..……

Submission Due Date: 15/12/2022 ……..……

Project Title: Music Recommender System using Natural Language Processing Technique and Audio Feature Analysis ……..……

Word Count: …… **Page Count:** ……..……..……

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: ……

Date: 14/12/2022…… ……

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

Kunal Chauhan
Student ID: 21139245

1 Introduction

The purpose of this Configuration Manual is to provide the reader/user who is going through the coding documents and actually access and use the files smoothly without any hassle and also knows how things were performed. This part of the research study is divided into several sections as to how many steps are involved in the coding process and how they were executed.

2 System Specifics

Operating System	Windows 11
Hard Disk Storage	128 GB
Random Access Memory	16 GB
GPU	Fast enough to process ML Models
Processor	Intel 11

3 IDE (Integrated Development Environment)

- Anaconda Navigator
- Python Programming Language Version: 3.9.12

4 Python Libraries required to run the Models

- absl-py 1.3.0
- aiohttp 3.8.1
- aiosignal 1.2.0
- alabaster 0.7.12
- anaconda-client 1.9.0
- anaconda-navigator 2.1.4
- anaconda-project 0.10.2
- anyio 3.5.0
- appdirs 1.4.4
- argon2-cffi 21.3.0
- argon2-cffi-bindings 21.2.0
- arrow 1.2.2
- astroid 2.6.6
- astropy 5.0.4
- asttokens 2.0.5
- astunparse 1.6.3
- async-timeout 4.0.1
- atomicwrites 1.4.0

- attrs 21.4.0
- audioread 3.0.0
- Automat 20.2.0
- autopep8 1.6.0
- Babel 2.9.1
- backcall 0.2.0
- backports.functools-lru-cache 1.6.4
- backports.tempfile 1.0
- backports.weakref 1.0.post1
- bcrypt 3.2.0
- beautifulsoup4 4.11.1
- binaryornot 0.4.4
- bitarray 2.4.1
- bkcharts 0.2
- black 19.10b0
- bleach 4.1.0
- bokeh 2.4.2
- boto3 1.21.32
- botocore 1.24.32
- Bottleneck 1.3.4
- brotlipy 0.7.0
- cachetools 4.2.2
- certifi 2021.10.8
- cffi 1.15.0
- chardet 4.0.0
- charset-normalizer 2.0.4
- click 8.0.4
- cloudpickle 2.0.0
- clyent 1.2.2
- colorama 0.4.4
- colorcet 2.0.6
- comtypes 1.1.10
- conda 4.12.0
- conda-build 3.21.8
- conda-content-trust 0+unknown
- conda-pack 0.6.0
- conda-package-handling 1.8.1
- conda-repo-cli 1.0.4
- conda-token 0.3.0
- conda-verify 3.4.2
- constantly 15.1.0
- cookiecutter 1.7.3
- cryptography 3.4.8
- cssselect 1.1.0
- cycler 0.11.0
- Cython 0.29.28
- cytoolz 0.11.0
- daal4py 2021.5.0
- dask 2022.2.1
- datashader 0.13.0
- datashape 0.5.4
- debugpy 1.5.1
- decorator 5.1.1
- defusedxml 0.7.1

- diff-match-patch 20200713
- distributed 2022.2.1
- docutils 0.17.1
- entrypoints 0.4
- et-xmlfile 1.1.0
- executing 0.8.3
- fastjsonschema 2.15.1
- filelock 3.6.0
- flake8 3.9.2
- Flask 1.1.2
- flatbuffers 22.10.26
- fonttools 4.25.0
- frozenlist 1.2.0
- fsspec 2022.2.0
- future 0.18.2
- gast 0.4.0
- gensim 4.1.2
- glob2 0.7
- google-api-core 1.25.1
- google-auth 1.33.0
- google-auth-oauthlib 0.4.6
- google-cloud-core 1.7.1
- google-cloud-storage 1.31.0
- google-crc32c 1.1.2
- google-pasta 0.2.0
- google-resumable-media 1.3.1
- googleapis-common-protos 1.53.0
- greenlet 1.1.1
- grpcio 1.42.0
- h5py 3.6.0
- HeapDict 1.0.1
- holoviews 1.14.8
- hvplot 0.7.3
- hyperlink 21.0.0
- hyperopt 0.2.7
- idna 3.3
- imagecodecs 2021.8.26
- imageio 2.9.0
- imagesize 1.3.0
- imbalanced-learn 0.9.1
- importlib-metadata 4.11.3
- incremental 21.3.0
- inflection 0.5.1
- iniconfig 1.1.1
- intake 0.6.5
- intervaltree 3.1.0
- ipykernel 6.9.1
- ipython 8.2.0
- ipython-genutils 0.2.0
- ipywidgets 7.6.5
- isort 5.9.3
- itemadapter 0.3.0
- itemloaders 1.0.4
- itsdangerous 2.0.1

- jdcal 1.4.1
- jedi 0.18.1
- Jinja2 2.11.3
- jinja2-time 0.2.0
- jmespath 0.10.0
- joblib 1.1.0
- json5 0.9.6
- jsonschema 4.4.0
- jupyter 1.0.0
- jupyter-client 6.1.12
- jupyter-console 6.4.0
- jupyter-core 4.9.2
- jupyter-server 1.13.5
- jupyterlab 3.3.2
- jupyterlab-pygments 0.1.2
- jupyterlab-server 2.10.3
- jupyterlab-widgets 1.0.0
- keras 2.11.0
- keyring 23.4.0
- kiwisolver 1.3.2
- lazy-object-proxy 1.6.0
- libarchive-c 2.9
- libclang 14.0.6
- librosa 0.9.2
- llvmlite 0.38.0
- locket 0.2.1
- lxml 4.8.0
- Markdown 3.3.4
- MarkupSafe 2.0.1
- matplotlib 3.5.1
- matplotlib-inline 0.1.2
- mccabe 0.6.1
- menuinst 1.4.18
- mistune 0.8.4
- mkl-fft 1.3.1
- mkl-random 1.2.2
- mkl-service 2.4.0
- mock 4.0.3
- mpmath 1.2.1
- msgpack 1.0.2
- multidict 5.1.0
- multipledispatch 0.6.0
- multitasking 0.0.11
- munkres 1.1.4
- mypy-extensions 0.4.3
- navigator-updater 0.2.1
- nbclassic 0.3.5
- nbclient 0.5.13
- nbconvert 6.4.4
- nbformat 5.3.0
- nest-asyncio 1.5.5
- networkx 2.7.1
- nltk 3.7
- nose 1.3.7

- notebook 6.4.8
- numba 0.55.1
- numexpr 2.8.1
- numpy 1.21.5
- numpydoc 1.2
- oauthlib 3.2.2
- olefile 0.46
- openpyxl 3.0.9
- opt-einsum 3.3.0
- packaging 21.3
- pandas 1.4.2
- pandocfilters 1.5.0
- panel 0.13.0
- param 1.12.0
- paramiko 2.8.1
- parsel 1.6.0
- parso 0.8.3
- partd 1.2.0
- pathspec 0.7.0
- patsy 0.5.2
- pep8 1.7.1
- pexpect 4.8.0
- pickleshare 0.7.5
- Pillow 9.0.1
- pip 21.2.4
- pkginfo 1.8.2
- plotly 5.6.0
- pluggy 1.0.0
- pooch 1.6.0
- poyo 0.5.0
- prometheus-client 0.13.1
- prompt-toolkit 3.0.20
- Protego 0.1.16
- protobuf 3.19.1
- psutil 5.8.0
- ptyprocess 0.7.0
- pure-eval 0.2.2
- py 1.11.0
- py4j 0.10.9.7
- pyasn1 0.4.8
- pyasn1-modules 0.2.8
- pycodestyle 2.7.0
- pycosat 0.6.3
- pycparser 2.21
- pyct 0.4.6
- pycurl 7.44.1
- PyDispatcher 2.0.5
- pydocstyle 6.1.1
- pyerfa 2.0.0
- pyflakes 2.3.1
- Pygments 2.11.2
- PyHamcrest 2.0.2
- PyJWT 2.1.0
- pylint 2.9.6

- pyls-spyder 0.4.0
- PyNaCl 1.4.0
- pyodbc 4.0.32
- pyOpenSSL 21.0.0
- pyparsing 3.0.4
- pyreadline 2.1
- pyrsistent 0.18.0
- PySocks 1.7.1
- pytest 7.1.1
- python-dateutil 2.8.2
- python-lsp-black 1.0.0
- python-lsp-jsonrpc 1.0.0
- python-lsp-server 1.2.4
- python-slugify 5.0.2
- python-snappy 0.6.0
- pytz 2021.3
- pyviz-comms 2.0.2
- PyWavelets 1.3.0
- pywin32 302
- pywin32-ctypes 0.2.0
- pywinpty 2.0.2
- PyYAML 6.0
- pyzmq 22.3.0
- QDarkStyle 3.0.2
- qstylizer 0.1.10
- QtAwesome 1.0.3
- qtconsole 5.3.0
- QtPy 2.0.1
- queuelib 1.5.0
- regex 2022.3.15
- requests 2.27.1
- requests-file 1.5.1
- requests-oauthlib 1.3.1
- resampy 0.4.2
- rope 0.22.0
- rsa 4.7.2
- Rtree 0.9.7
- ruamel-yaml-conda 0.15.100
- s3transfer 0.5.0
- scikit-image 0.19.2
- scikit-learn 1.1.2
- scikit-learn-intelex 2021.20220215.102710
- scipy 1.7.3
- Scrapy 2.6.1
- seaborn 0.11.2
- Send2Trash 1.8.0
- service-identity 18.1.0
- setuptools 61.2.0
- sip 4.19.13
- six 1.16.0
- smart-open 5.1.0
- sniffio 1.2.0
- snowballstemmer 2.2.0
- sortedcollections 2.1.0

- sortedcontainers 2.4.0
- soundfile 0.11.0
- soupsieve 2.3.1
- Sphinx 4.4.0
- sphinxcontrib-applehelp 1.0.2
- sphinxcontrib-devhelp 1.0.2
- sphinxcontrib-htmlhelp 2.0.0
- sphinxcontrib-jsmath 1.0.1
- sphinxcontrib-qthelp 1.0.3
- sphinxcontrib-serializinghtml 1.1.5
- spyder 5.1.5
- spyder-kernels 2.1.3
- SQLAlchemy 1.4.32
- stack-data 0.2.0
- statsmodels 0.13.2
- sympy 1.10.1
- tables 3.6.1
- tabulate 0.8.9
- TBB 0.2
- tblib 1.7.0
- tenacity 8.0.1
- tensorboard 2.11.0
- tensorboard-data-server 0.6.1
- tensorboard-plugin-wit 1.8.1
- tensorflow 2.11.0
- tensorflow-estimator 2.11.0
- tensorflow-intel 2.11.0
- tensorflow-io-gcs-filesystem 0.28.0
- termcolor 2.1.1
- terminado 0.13.1
- testpath 0.5.0
- text-unidecode 1.3
- textdistance 4.2.1
- threadpoolctl 2.2.0
- three-merge 0.1.1
- tifffile 2021.7.2
- tinycss 0.4
- tldextract 3.2.0
- toml 0.10.2
- tomli 1.2.2
- toolz 0.11.2
- tornado 6.1
- tqdm 4.64.0
- traitlets 5.1.1
- Twisted 22.2.0
- twisted-iocpsupport 1.0.2
- typed-ast 1.4.3
- typing_extensions 4.1.1
- ujson 5.1.0
- Unidecode 1.2.0
- urllib3 1.26.9
- w3lib 1.21.0
- watchdog 2.1.6
- wcwidth 0.2.5

- webencodings 0.5.1
- websocket-client 0.58.0
- Werkzeug 2.0.3
- wheel 0.37.1
- widgetsnbextension 3.5.2
- win-inet-pton 1.1.0
- win-unicode-console 0.5
- wincertstore 0.2
- wordcloud 1.8.2.2
- wrapt 1.12.1
- xarray 0.20.1
- xgboost 1.7.1
- xlrd 2.0.1
- XlsxWriter 3.0.3
- xlwings 0.24.9
- yapf 0.31.0
- yarl 1.6.3
- yfinance 0.1.74
- zict 2.0.0
- zipp 3.7.0
- zope.interface 5.4.0

5 Dataset

Two datasets were used in this research. The links for both the datasets are mentioned below:

- Lyrics Dataset: https://www.kaggle.com/datasets/deepshah16/song-lyrics-dataset
- Audio Dataset: https://www.kaggle.com/datasets/andradaolteanu/gtzan-dataset-music-genre-classification

6 Code Files

- Kunal_Lyrics_NLP.pynb: This is for the lyrics dataset coding.
- Kunal_Audio_Feature_Analysis_Final.pynb: This is for the audio dataset coding

7 Lyrics Dataset Coding for NLP Sentiment Analysis and Recommendations

To create the lyrics NLP Model following steps were involved

1. EDA
    a. The files were read first

```
df_a=pd.read_csv('C:\\Users\\mouna\\Desktop\\DIVERS\\Kunal\\Project\\csv\\ArianaGrande.csv')
df_b=pd.read_csv('C:\\Users\\mouna\\Desktop\\DIVERS\\Kunal\\Project\\csv\\Beyonce.csv')
df_be=pd.read_csv('C:\\Users\\mouna\\Desktop\\DIVERS\\Kunal\\Project\\csv\\BillieEilish.csv')
df_ed=pd.read_csv('C:\\Users\\mouna\\Desktop\\DIVERS\\Kunal\\Project\\csv\\EdSheeran.csv')
df_j=pd.read_csv('C:\\Users\\mouna\\Desktop\\DIVERS\\Kunal\\Project\\csv\\JustinBieber.csv')
df_k=pd.read_csv('C:\\Users\\mouna\\Desktop\\DIVERS\\Kunal\\Project\\csv\\KatyPerry.csv')
df_l=pd.read_csv('C:\\Users\\mouna\\Desktop\\DIVERS\\Kunal\\Project\\csv\\LadyGaga.csv')
df_m5=pd.read_csv('C:\\Users\\mouna\\Desktop\\DIVERS\\Kunal\\Project\\csv\\Maroon5.csv')
df_p=pd.read_csv('C:\\Users\\mouna\\Desktop\\DIVERS\\Kunal\\Project\\csv\\PostMalone.csv')
df_r=pd.read_csv('C:\\Users\\mouna\\Desktop\\DIVERS\\Kunal\\Project\\csv\\Rihanna.csv')
df_s=pd.read_csv('C:\\Users\\mouna\\Desktop\\DIVERS\\Kunal\\Project\\csv\\SelenaGomez.csv')
df_all=[df_a,df_b,df_be,df_ed,df_j,df_k,df_l,df_m5,df_p,df_r,df_s]
```

Cleaning 📌 Note: Removing the nan values,unreleased albums and songs sung in any other language than english.

Lyrics to words -> To convert all the Lyrics to word To word -> Converts all the words and removes all the repetative words Count word -> This function can count all the unique and as well non unique words Word stats -> To give the statistics of words.

    b. Removal of stopwords and performing Lemmatization

```
def lyrics_to_words(document):
    """
    This function splits the text of lyrics to  single words, removing stopwords and doing the lemmatization to each word

    parameters:
    document: text to split to single words
    """
    stop_words = set(stopwords.words('english'))
    exclude = set(string.punctuation)
    lemma = WordNetLemmatizer()
    stopwordremoval = " ".join([i for i in document.lower().split() if i not in stop_words])
    punctuationremoval = ''.join(ch for ch in stopwordremoval if ch not in exclude)
    normalized = " ".join(lemma.lemmatize(word) for word in punctuationremoval.split())
    return normalized
```

c. Finding the Lexical Richness for all the artists:

Entrée [19]:
```
artists=['ArianaGrande','Beyonce','BillieEilish','EdSheeran','JustinBieber','KatyPerry','LadyGaga','Maroon5','PostMalone','Rihanr
df_info =pd.DataFrame({'name':artists,'before':before,'after':after,'words':wd,'unique words':ulength,'word count':length})
df_info['diff']=df_info['before']-df_info['after']
df_info['words per songs'] = round(df_info['word count'] / df_info['after'],0)
df_info['words per songs'] = df_info['words per songs'].astype('int')
df_info['lexicalrichness']=(df_info['unique words']/df_info['word count'])*100
df_info=df_info[['name','before','after','diff','words','words per songs','unique words','word count','lexicalrichness']]
df_info
```

Out[19]:

	name	before	after	diff	words	words per songs	unique words	word count	lexicalrichness
0	ArianaGrande	308	202	106	[thought, id, end, sean, match, wrote, song, r...	65	3394	13193	25.725764
1	Beyonce	406	224	182	[beyoncé, ive, drinkin, get, filthy, liquor, t...	85	4980	19041	26.154089
2	BillieEilish	145	73	72	[know, im, good, ive, learned, lose, cant, aff...	51	1519	3720	40.833333
3	EdSheeran	296	202	94	[club, best, place, find, lover, bar, go, frie...	92	4490	18650	24.075067
4	JustinBieber	348	268	80	[produced, benny, blanco, time, rained, parade...	70	3752	18659	20.108259
5	KatyPerry	325	191	134	[refrain, know, strut, fuck, katy, perry, tige...	69	3605	13202	27.306469
6	LadyGaga	402	236	166	[lady, gaga, r, kelly, yeah, oh, turn, mic, eh...	73	4299	17147	25.071441
7	Maroon5	197	125	72	[adam, levine, say, hey, baby, oh, mama, play,...	62	1841	7724	23.834801
8	PostMalone	148	128	20	[post, malone, hahahahaha, tank, god, ayy, ive...	85	3091	10823	28.559549
9	Rihanna	405	248	157	[rihanna, work, said, haffi, see, mi, dirt, pu...	66	2869	16391	17.503508
10	SelenaGomez	175	108	67	[promised, world, fell, put, first, adored, se...	53	1965	5676	34.619450

2. Sentiment Analysis

Entrée [26]:
```
#Create lists to store the different scores for each word
def setimentanalyzer(df):
    neg='Negative'
    neu='Neutral'
    pos='Positive'
    negative = []
    neutral = []
    positive = []
    dominant_sentiment=[]
    dominant_sentiment_score=[]
    #Initialize the model
    sid = SentimentIntensityAnalyzer()
    #Iterate for each row of lyrics and append the scores
    for i in df.index:

        scores = sid.polarity_scores(df['Lyric'].iloc[i])
        negative.append(scores['neg'])
        neutral.append(scores['neu'])
        positive.append(scores['pos'])
        if scores['neg']>scores['pos']:
            dominant_sentiment_score.append(scores['neg'])
            dominant_sentiment.append(neg)
        elif scores['neg']<scores['pos']:
            dominant_sentiment_score.append(scores['pos'])
            dominant_sentiment.append(pos)
        else:
            dominant_sentiment_score.append(scores['neu'])
```

a. Dominant score for sentiment analysis

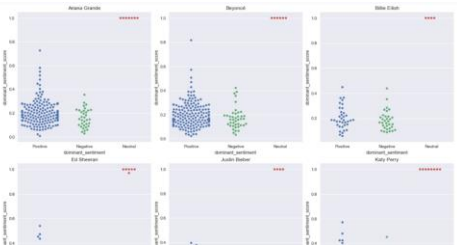
Entrée [27]:
```
df_sentiment=setimentanalyzer(df_main)
df_sentiment.head(5)
```

Out[27]:

	Artist	Title	Album	Date	Lyric	Year	words	negative	neutral	positive	dominant_sentiment	dominant_sentiment_score
0	Ariana Grande	thank u, next	thank u, next	2018-11-03	thought i'd end up with sean but he wasn't a m...	2018	[thought, id, end, sean, match, wrote, song, r...	0.062	0.503	0.435	Positive	0.435
1	Ariana Grande	7 rings	thank u, next	2019-01-18	yeah breakfast at tiffany's and bottles of bub...	2019	[yeah, breakfast, tiffany, bottle, bubble, gir...	0.070	0.650	0.280	Positive	0.280
2	Ariana Grande	God is a woman	Sweetener	2018-07-13	you you love it how i move you you love it how...	2018	[love, move, touch, one, said, done, believe, ...	0.000	0.733	0.267	Positive	0.267
3	Ariana Grande	Side To Side	Dangerous Woman	2016-05-20	ariana grande nicki minaj i've been here all ...	2016	[ariana, grande, nicki, minaj, ive, night, day...	0.062	0.865	0.073	Positive	0.073
4	Ariana Grande	no tears left to cry	Sweetener	2018-04-20	right now i'm in a state of mind i wanna be in...	2018	[right, im, state, mind, wanna, like, time, ai...	0.079	0.716	0.204	Positive	0.204

b. Swarm Plot for dominant score code and output

```
Entrée [28]: df_temp=[]
             name=df_sentiment['Artist'].unique()
             name=name[0:9]
             num_plots = 9
             total_cols = 3
             total_rows = 3
             #fig, axs = plt.subplots(nrows=total_rows, ncols=total_cols,
             fig, axs = plt.subplots(nrows=total_rows, ncols=total_cols,
                                     figsize=(5*total_cols, 5*total_rows), constrained_layout=True)
             for artist in df_main['Artist'].unique():
                 df_temp.append(df_sentiment[df_sentiment['Artist']==artist])
             for i, var in enumerate(name):
                 row = i//total_cols
                 pos = i % total_cols
                 plot = sns.swarmplot(data=df_temp[i], x="dominant_sentiment", y="dominant_sentiment_score",ax=axs[row][pos])
                 axs[row][pos].set_title(name[i])

             name=df_main['Artist'].unique()
             name=name[9:11]
             fig, axs = plt.subplots(nrows=1, ncols=2,
                                     figsize=(10, 5), constrained_layout=True)
             plot = sns.swarmplot(data=df_temp[9], x="dominant_sentiment", y="dominant_sentiment_score",ax=axs[0])
             axs[0].set_title(name[0])
             plot = sns.swarmplot(data=df_temp[10], x="dominant_sentiment", y="dominant_sentiment_score",ax=axs[1])
             axs[1].set_title(name[1])

Out[28]: Text(0.5, 1.0, 'Selena Gomez')
```



3. Cosine Similarity:

```
Entrée [38]:
lyrics3=lyrics1['Lyric']

tfidf_vectorizer=TfidfVectorizer()
tfidf_matrix=tfidf_vectorizer.fit_transform(lyrics3)
tfidf_matrix.shape #(5000,40019) #5000 songs (rows of the matrix), 40019 tf-idf terms (columns of the matrix)
cos_sim=cosine_similarity(tfidf_matrix[0:1],tfidf_matrix)
```

```
Entrée [39]: sim_score=list(cos_sim[0])
             sim_score
             0.1707935019444157,
             0.06677107554780017,
             0.10175411633750742,
             0.09855983285387643,
             0.15103383837796097,
             0.13133000019192892,
             0.16927840105314576,
             0.19800760136081136,
             0.1600581260107213 6,
             0.16681143879260324,
             0.22041871163442306,
             0.08655853971288821,
             0.10850945003900861,
             0.1970986955591244,
             0.17436335267937728,
             0.11764974696631764,
             0.08808697232117536,
             0.1397624524786392,
             0.15032673660849485,
             0.17266358934993414,
```

4. Recommendations:

```
Entrée [43]: #Songs with most similarity with first song
             df_sim = lyrics1[lyrics1["Similairty Score"]>0.25]
             df_sim
```

Out[43]:

	Artist	Title	Album	Date	Lyric	Year	Similairty Score
0	Beyoncé	Drunk in Love	BEYONCÉ	2013-12-17	beyoncé i've been drinkin' i've been drinkin' ...	2013.0	1.000000
2	Beyoncé	Partition	BEYONCÉ	2013-12-13	part yoncé let me hear you say hey ms carte...	2013.0	0.253938
4	Beyoncé	Hold Up	Lemonade	2016-04-23	hold up they don't love you like i love you sl...	2016.0	0.259436
9	Beyoncé	All Night	Lemonade	2016-04-23	i found the truth beneath your lies and true l...	2016.0	0.250239
13	Beyoncé	Don't Hurt Yourself	Lemonade	2016-04-23	beyoncé oh na na na oh na na na oh na na na do...	2016.0	0.255571
22	Beyoncé	Drunk in Love (Remix)	BEYONCÉ (Platinum Edition)	2014-11-24	danyèl waro beyoncé kinm in kalot oté mandela...	2014.0	0.877806
27	Beyoncé	Countdown	4	2011-10-04	boy oh killing me softly and i'm still falli...	2011.0	0.271587
170	Beyoncé	Drunk in Love (Homecoming Live)	HOMECOMING: THE LIVE ALBUM	2019-04-17	beyoncé i've been drinkin' i've been drinkin' ...	2019.0	0.518437
194	Beyoncé	Countdown (Homecoming Live)	HOMECOMING: THE LIVE ALBUM	2019-04-17	rah rah rah sing it y'all oh killing me soft...	2019.0	0.250446
199	Beyoncé	Naughty Girl (Remix)	Live at Wembley	2003-05-18	lil' kim this is ladies night once again it's ...	2003.0	0.283078
0	Ed Sheeran	Shape of You	÷ (Divide)	2017-01-06	the club isn't the best place to find a lover ...	2017.0	0.261189

8 Audio Dataset coding for Audio Feature Analysis

1. EDA
   a. File Input

```
path='C:\\Users\\mouna\\Downloads\\audiobook\\Data'
genres=os.listdir(os.path.join(path,'genres_original'))
print(genres)
```

```
['blues', 'classical', 'country', 'disco', 'hiphop', 'jazz', 'metal', 'pop', 'reggae', 'rock']
```

Sequence of vibrations with various pressure strengths constitutes sound (y) The number of audio samples conveyed each second, expressed in Hz or kHz, is known as the sample rate (sr).

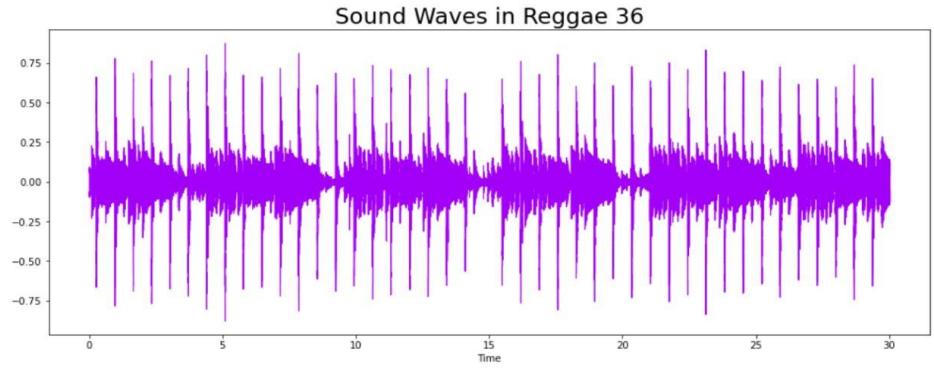
b. Sample Rate:

```
# Importing 1 file
y, sr = librosa.load(f'C:\\Users\\mouna\\Downloads\\audiobook\\Data\\genres_original\\reggae\\reggae.00036.wav')

print('y:', y, '\n')
print('y shape:', np.shape(y), '\n')
print('Sample Rate (KHz):', sr, '\n')

# Verify length of the audio
print('Check Len of Audio:', 661794/22050)
```

```
y: [0.02072144 0.04492188 0.05422974 ... 0.06912231 0.08303833 0.08572388]

y shape: (661794,)

Sample Rate (KHz): 22050

Check Len of Audio: 30.013333333333332
```

c. 2-D Representation of Sound Waves

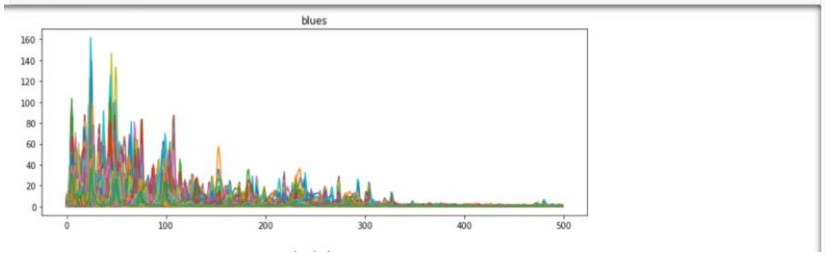
```
plt.figure(figsize = (16, 6))
librosa.display.waveshow(y = audio_file, sr = sr, color = "#A300F9");
plt.title("Sound Waves in Reggae 36", fontsize = 23);
```



2. Data Preparation

a. Fourier Transform:

```
n_fft=2048 #default value recommended, n_ftt represents the number of samples that will be converted at once.
hop_length=512 #understandable by name
win_length=2048 #window using which samples are converted.

for i in genres:
    aud,sr=librosa.load(os.path.join(path,'genres_original',i,f'{i}.00001.wav'))
    aud_ft= np.abs(librosa.stft(aud, n_fft = n_fft, hop_length = hop_length,win_length=win_length))
#     print(np.shape(aud_ft)) #(1025,1302)
    plt.figure(figsize=(12,4))
    plt.plot(aud_ft[:500,:])#viewing only upto 500 Hz
    plt.title(f'{i}')
```



b. Spectrogram

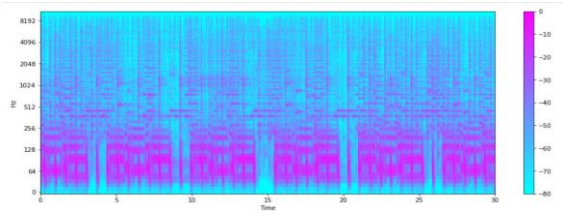
The Spectrogram

A spectrogram is a visual representation of a signal's frequency spectrum as it changes over time. Spectrograms are sometimes referred to as sonographs, voiceprints, or voicegrams when applied to an audio signal (wiki). In this step, we change the frequency axis to a logarithmic one.

```
# Default FFT window size
n_fft = 2048 # FFT window size
hop_length = 512 # number audio of frames between STFT columns (looks like a good default)

# Short-time Fourier transform (STFT)
D = np.abs(librosa.stft(audio_file, n_fft = n_fft, hop_length = hop_length))

print('Shape of D object:', np.shape(D))
```

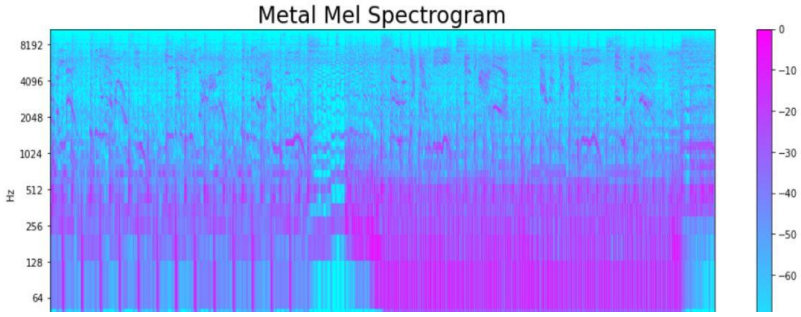
```
Shape of D object: (1025, 1293)
```

```
# Convert an amplitude spectrogram to Decibels-scaled spectrogram.
DB = librosa.amplitude_to_db(D, ref = np.max)

# Creating the Spectogram
plt.figure(figsize = (16, 6))
librosa.display.specshow(DB, sr = sr, hop_length = hop_length, x_axis = 'time', y_axis = 'log',cmap = 'cool')
plt.colorbar();
```



c. Mel Spectrogram:

```
y, sr = librosa.load(f'C:\\Users\\mouna\\Downloads\\audiobook\\Data\\genres_original\\metal\\metal.00036.wav')
y, _ = librosa.effects.trim(y)


S = librosa.feature.melspectrogram(y, sr=sr)
S_DB = librosa.amplitude_to_db(S, ref=np.max)
plt.figure(figsize = (16, 6))
librosa.display.specshow(S_DB, sr=sr, hop_length=hop_length, x_axis = 'time', y_axis = 'log',
                         cmap = 'cool');
plt.colorbar();
plt.title("Metal Mel Spectrogram", fontsize = 23);
```

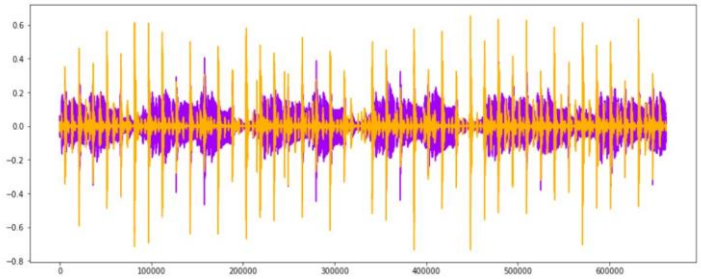


3. Feature Extraction

a. Harmonics

Harmonics are characteristics that human years cannot differentiate (represents the sound color).The sound rhythm and emotion are represented by the shock wave in perceptual understanding. ¶

```
y_harm, y_perc = librosa.effects.hpss(audio_file)

plt.figure(figsize = (16, 6))
plt.plot(y_harm, color = '#A300F9');
plt.plot(y_perc, color = '#FFB100');
```



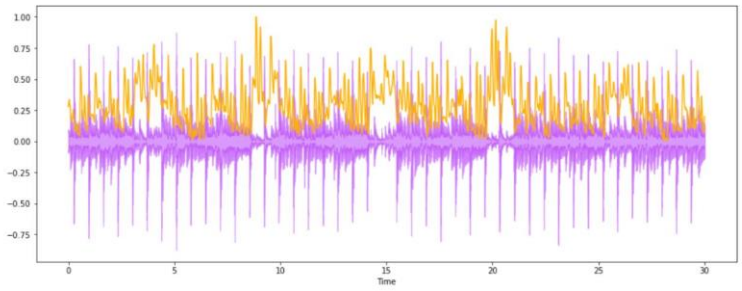
b. Spectral Centroid:

```
#frequencies present in the sound.
# Calculate the Spectral Centroids
spectral_centroids = librosa.feature.spectral_centroid(audio_file, sr=sr)[0]

# Shape is a vector
print('Centroids:', spectral_centroids, '\n')
print('Shape of Spectral Centroids:', spectral_centroids.shape, '\n')

# Computing the time variable for visualization
frames = range(len(spectral_centroids))

# Converts frame counts to time (seconds)
t = librosa.frames_to_time(frames)

print('frames:', frames, '\n')
print('t:', t)

# Function that normalizes the Sound Data
def normalize(x, axis=0):
    return sklearn.preprocessing.minmax_scale(x, axis=axis)
```

```
Centroids: [1758.29476432 1946.74243678 2038.8113414  ...  766.50416352 1041.07728901
 1391.05145642]

Shape of Spectral Centroids: (1293,)

frames: range(0, 1293)

t: [0.00000000e+00 2.32199546e-02 4.64399093e-02 ... 2.99537415e+01
 2.99769615e+01 3.00001814e+01]
```

```
#Plotting the Spectral Centroid along the waveform
plt.figure(figsize = (16, 6))
librosa.display.waveshow(audio_file, sr=sr, alpha=0.4, color = '#A300F9');
plt.plot(t, normalize(spectral_centroids), color='#FFB100')
```

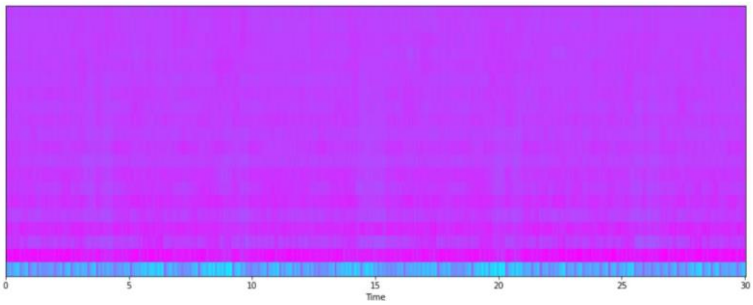
```
[<matplotlib.lines.Line2D at 0x20650d49e50>]
```



c. MFCC:

```
mfccs = librosa.feature.mfcc(audio_file, sr=sr)
print('mfccs shape:', mfccs.shape)

#Displaying  the MFCCs:
plt.figure(figsize = (16, 6))
librosa.display.specshow(mfccs, sr=sr, x_axis='time', cmap = 'cool');
```

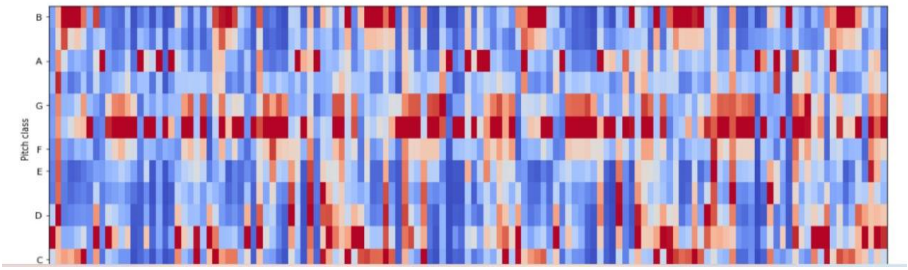
```
mfccs shape: (20, 1293)
```



d. Chroma Frequencies:

```
# Increase or decrease hop_length to change how granular you want your data to be
hop_length = 5000

# Chromogram
chromagram = librosa.feature.chroma_stft(audio_file, sr=sr, hop_length=hop_length)
print('Chromogram shape:', chromagram.shape)

plt.figure(figsize=(16, 6))
librosa.display.specshow(chromagram, x_axis='time', y_axis='chroma', hop_length=hop_length, cmap='coolwarm');
```

```
Chromogram shape: (12, 133)
```



e. Printing the data frame:

```
data = pd.read_csv(f'C:\\Users\\mouna\\Downloads\\audiobook\\Data\\features_30_sec.csv')
data.head()
```

	filename	length	chroma_stft_mean	chroma_stft_var	rms_mean	rms_var	spectral_centroid_mean	spectral_centroid_var	spectral_bandwidth_mean	s
0	blues.00000.wav	661794	0.350088	0.088757	0.130228	0.002827	1784.165850	129774.064525	2002.449060	
1	blues.00001.wav	661794	0.340914	0.094980	0.095948	0.002373	1530.176679	375850.073649	2039.036516	
2	blues.00002.wav	661794	0.363637	0.085275	0.175570	0.002746	1552.811865	156467.643368	1747.702312	
3	blues.00003.wav	661794	0.404785	0.093999	0.141093	0.006346	1070.106615	184355.942417	1596.412872	
4	blues.00004.wav	661794	0.308526	0.087841	0.091529	0.002303	1835.004266	343399.939274	1748.172116	

f. PCA and Normalization:

The explained variance ratio is the percentage of variance attributed to each of the chosen components. To avoid overfitting, you should choose the number of components to include in your model by adding the explained variance ratio of each component until you reach a total of around 0.8 or 80%.

```
from sklearn import preprocessing

data = data.iloc[0:, 1:]
y = data['label']
X = data.loc[:, data.columns != 'label']

#### NORMALIZE X ####
cols = X.columns
min_max_scaler = preprocessing.MinMaxScaler()
np_scaled = min_max_scaler.fit_transform(X)
X = pd.DataFrame(np_scaled, columns = cols)


#### PCA 2 COMPONENTS ####
from sklearn.decomposition import PCA

pca = PCA(n_components=2)
principalComponents = pca.fit_transform(X)
principalDf = pd.DataFrame(data = principalComponents, columns = ['principal component 1', 'principal component 2'])

# concatenate with target label
finalDf = pd.concat([principalDf, y], axis = 1)

pca.explained_variance_ratio_
```

```
array([0.2439355 , 0.21781804])
```

4. Implying the Machine Learning Models:

```
!pip install xgboost
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import SGDClassifier, LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier
from xgboost import XGBClassifier, XGBRFClassifier
from xgboost import plot_tree, plot_importance

from sklearn.metrics import confusion_matrix, accuracy_score, roc_auc_score, roc_curve
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.feature_selection import RFE
```

```
Collecting xgboost
  Downloading xgboost-1.7.1-py3-none-win_amd64.whl (89.1 MB)
Requirement already satisfied: numpy in c:\users\mouna\anaconda3\lib\site-packages (from xgboost) (1.21.5)
Requirement already satisfied: scipy in c:\users\mouna\anaconda3\lib\site-packages (from xgboost) (1.7.3)
Installing collected packages: xgboost
Successfully installed xgboost-1.7.1
```

a. Accuracies of ML Models:

```
# Naive Bayes
nb = GaussianNB()
model_assess(nb, "Naive Bayes")

# Stochastic Gradient Descent
sgd = SGDClassifier(max_iter=5000, random_state=0)
model_assess(sgd, "Stochastic Gradient Descent")

# KNN
knn = KNeighborsClassifier(n_neighbors=19)
model_assess(knn, "KNN")

# Decission trees
tree = DecisionTreeClassifier()
model_assess(tree, "Decission trees")

# Random Forest
rforest = RandomForestClassifier(n_estimators=1000, max_depth=10, random_state=0)
model_assess(rforest, "Random Forest")

# Support Vector Machine
svm = SVC(decision_function_shape="ovo")
model_assess(svm, "Support Vector Machine")

# Logistic Regression
lg = LogisticRegression(random_state=0, solver='lbfgs', multi_class='multinomial')
model_assess(lg, "Logistic Regression")
```

```
Accuracy Naive Bayes : 0.51952

Accuracy Stochastic Gradient Descent : 0.65532

Accuracy KNN : 0.80581

Accuracy Decission trees : 0.64264

Accuracy Random Forest : 0.81415

Accuracy Support Vector Machine : 0.75409

Accuracy Logistic Regression : 0.6977
```

b. Importing Tensor Flow:

```
!pip install hyperopt
from tensorflow import keras
from keras.models import Sequential
from IPython.display import Audio
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import normalize
from sklearn.feature_selection import RFECV,mutual_info_regression
from sklearn.metrics import confusion_matrix, accuracy_score,classification_report
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.decomposition import PCA
from xgboost import XGBClassifier
from hyperopt import STATUS_OK, Trials, fmin, hp, tpe
```

```
Collecting hyperopt
  Downloading hyperopt-0.2.7-py2.py3-none-any.whl (1.6 MB)
Requirement already satisfied: future in c:\users\mouna\anaconda3\lib\site-packages (from hyperopt) (0.18.2)
Requirement already satisfied: cloudpickle in c:\users\mouna\anaconda3\lib\site-packages (from hyperopt) (2.0.0)
Requirement already satisfied: tqdm in c:\users\mouna\anaconda3\lib\site-packages (from hyperopt) (4.64.0)
Requirement already satisfied: scipy in c:\users\mouna\anaconda3\lib\site-packages (from hyperopt) (1.7.3)
Requirement already satisfied: six in c:\users\mouna\anaconda3\lib\site-packages (from hyperopt) (1.16.0)
Requirement already satisfied: numpy in c:\users\mouna\anaconda3\lib\site-packages (from hyperopt) (1.21.5)
Collecting py4j
  Downloading py4j-0.10.9.7-py2.py3-none-any.whl (200 kB)
Requirement already satisfied: networkx>=2.2 in c:\users\mouna\anaconda3\lib\site-packages (from hyperopt) (2.7.1)
Requirement already satisfied: colorama in c:\users\mouna\anaconda3\lib\site-packages (from tqdm->hyperopt) (0.4.4)
Installing collected packages: py4j, hyperopt
Successfully installed hyperopt-0.2.7 py4j-0.10.9.7
```

c. Changing the categorical data to numeric data:

```
!pip install hyperopt
from tensorflow import keras
from keras.models import Sequential
from IPython.display import Audio
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import normalize
from sklearn.feature_selection import RFECV,mutual_info_regression
from sklearn.metrics import confusion_matrix, accuracy_score,classification_report
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.decomposition import PCA
from xgboost import XGBClassifier
from hyperopt import STATUS_OK, Trials, fmin, hp, tpe
```

```
Collecting hyperopt
  Downloading hyperopt-0.2.7-py2.py3-none-any.whl (1.6 MB)
Requirement already satisfied: future in c:\users\mouna\anaconda3\lib\site-packages (from hyperopt) (0.18.2)
Requirement already satisfied: cloudpickle in c:\users\mouna\anaconda3\lib\site-packages (from hyperopt) (2.0.0)
Requirement already satisfied: tqdm in c:\users\mouna\anaconda3\lib\site-packages (from hyperopt) (4.64.0)
Requirement already satisfied: scipy in c:\users\mouna\anaconda3\lib\site-packages (from hyperopt) (1.7.3)
Requirement already satisfied: six in c:\users\mouna\anaconda3\lib\site-packages (from hyperopt) (1.16.0)
Requirement already satisfied: numpy in c:\users\mouna\anaconda3\lib\site-packages (from hyperopt) (1.21.5)
Collecting py4j
  Downloading py4j-0.10.9.7-py2.py3-none-any.whl (200 kB)
Requirement already satisfied: networkx>=2.2 in c:\users\mouna\anaconda3\lib\site-packages (from hyperopt) (2.7.1)
Requirement already satisfied: colorama in c:\users\mouna\anaconda3\lib\site-packages (from tqdm->hyperopt) (0.4.4)
Installing collected packages: py4j, hyperopt
Successfully installed hyperopt-0.2.7 py4j-0.10.9.7
```

5. Applying CNN

a. Importing:

Convolutional Neural Network (CNN) Deep Learning

```
#Here, we use Adam optimizer to train the model
#All of the hidden layers are using RELU activation function
#Output layer uses softmax function


model = keras.models.Sequential([
    keras.layers.Dense(512, activation="relu", input_shape=(X_train.shape[1],)),
    keras.layers.Dropout(0.2),

    keras.layers.Dense(256,activation="relu"),
    keras.layers.Dropout(0.2),

    keras.layers.Dense(128,activation="relu"),
    keras.layers.Dropout(0.2),

    keras.layers.Dense(64,activation="relu"),
    keras.layers.Dropout(0.2),

    keras.layers.Dense(10, activation="softmax"),

])
print(model.summary())
model_history = trainModel(model=model, epochs=1500, optimizer='adam')
```

```
53/53 [==============================] - 1s 22ms/step - loss: 0.0241 - accuracy: 0.9943 - val_loss: 0.4161 - val_accuracy: 0.92
14
Epoch 186/1500
53/53 [==============================] - 1s 18ms/step - loss: 0.0176 - accuracy: 0.9943 - val_loss: 0.4365 - val_accuracy: 0.92
```

b. Test Loss and Accuracy:

```
test_loss, test_accuracy = model.evaluate(X_test, y_test, batch_size=128)
print("The test loss is :",test_loss)
print("\nThe test Accuracy is :",test_accuracy*100)
```

```
26/26 [==============================] - 0s 5ms/step - loss: 0.7292 - accuracy: 0.9327
The test loss is : 0.7291868329048157

The test Accuracy is : 93.26660633087158
```

c. Accuracy curve for CNN:

```
test_loss, test_accuracy = model.evaluate(X_test, y_test, batch_size=128)
print("The test loss is :",test_loss)
print("\nThe test Accuracy is :",test_accuracy*100)
```

```
26/26 [==============================] - 0s 5ms/step - loss: 0.7292 - accuracy: 0.9327
The test loss is : 0.7291868329048157

The test Accuracy is : 93.26660633087158
```