MUSIC GENRE CLASSIFICATION USING DEEP LEARNING

A PROJECT REPORT

Submitted by

MOUNIKA K S (17BIT012)

DEYARADEVI S(17BIT026)

SWETHA K (17BIT059)

In partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

KUMARAGURU COLLEGE OF TECHNOLOGY

COIMBATORE-641 049

(An Autonomous Institution Affiliated to Anna University, Chennai)

ANNA UNIVERSITY: CHENNAI 600025

MAY 2021

KUMARAGURU COLLEGE OF TECHNOLOGY

COIMBATORE 641 049

(An Autonomous Institution Affiliated to Anna University, Chennai)

BONAFIDE CERTIFICATE

Certified that this project report **"Music Genre Classification using Deep Learning"** is the bonafide work of "Mounika K S (17BIT012), Deyaradevi S (17BIT026), Swetha K (17BIT059)**"** who carried out the project work under my supervision.

SIGNATURE
Dr.M.Alamelu
Head Of the Department
Associate Professor
Information Technology

SIGNATURE
Dr.V.Vanitha
Supervisor
Professor
Information Technology

The candidates with University register number 17BIT012, 17BIT026, 17BIT059 were examined in the Project Viva-Voce examination held on ……

Internal Examiner **External Examiner**

DECLARATION

We affirm that the project work titled **"Music Genre Classification using Deep Learning"** being submitted in partial fulfillment for the award of B.Tech Information Technology is the original work carried out by us. It has not formed the part of any other project work submitted for the award of any degree or diploma, either in this or any other University.

Mounika K S (17BIT012)

Deyaradevi S (17BIT026)

Swetha K (17BIT059)

I certify that the declaration made above by the candidates is true.

ACKNOWLEDGEMENT

We express our profound gratitude to the management of Kumaraguru College of Technology for providing the required infrastructure that enabled us to successfully complete the project.

We extend our gratitude to our Principal, **Dr. D. Saravanan,** for providing us the necessary facilities to pursue the project.

We would like to acknowledge **Dr. M. Alamelu**, Associate Professor and Head, Department of Information Technology, for her support and encouragement throughout this project.

We thank our project coordinator **Dr.P.C.Thirumal,** Associate Professor, Department of Information Technology and guide **Dr. V. Vanitha,** Professor, Department of Information Technology, for their constant and continuous effort, guidance and valuable time.

Our sincere and hearty thanks to staff members of the Department of Information Technology of Kumaraguru College of Technology for their well wishes, timely help and support rendered to us during our project. We are greatly indebted to our family, relatives and friends, without whom life would have not been shaped to this level.

Mounika K S

Deyaradevi S

Swetha K

ABSTRACT

In music genre classification, the application of convolutional neural networks and convolutional recurrent neural networks are discussed. In addition the well-known architecture transfer learning techniques are used. Genre classification will be valuable when there are some interesting facts or problems, finding the most specific song which has been listened to many times. Different approaches like fine-tuning, initializations and optimizers are implemented for the betterment of results. Multiframe approach is used to analyze the song in detail. A pre-trained model on a large dataset and fine-tune its weights to adapt it to perform a new task. In order to derive the output, frame acquisition and confusion matrix are used. The evaluation and performance of both handmade dataset and GTZAN dataset are used in a lot of works that compares the execution of this approach from one state. The most challenging task is to categorise the audio files as per their genre in the field of music information retrieval. The real world application in this field is automatic tagging of unspecified frames of music(apps like Saavn, Wynk, Spotify etc.,) Most companies use this field to satisfy the customer's. Finding the type of music genre is the first process for these applications.

Keywords: Deep learning, Convolutional neural network, Music genre classification,Transferlearning

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

People go crazy on hearing songs now-a-days. The songs they hear can differ according to their mood. So the genre of music such as jazz, classical, rock, hiphop, country,...etc plays an important role. People usually have the habit of sorting songs by different genres in their playlist. Indian people speak different languages, follow various cultures according to their geographical area. Hence, Indian music industry is a huge and vast area having numerous songs corresponding to their geographical area, finding their genre by human capabilities is quite a difficult task. Music is not just a collection of words, it conveys many useful information, reality of life. Moreover, it is one of the online platforms which connects thousands of minds together. Modern world has brought many improvements in the field of music. For instance, the classical songs that were composed sixty years ago are different from the classical songs that are composed today.

The extraction of acoustic features that are the best estimators in the classification of genres. This method follows a single or multi-label classification or in some stages, regression stage required for building that system. A music genre is a category of conventional which derives from the frames of music, feature extraction depends on signal processing and relevant features using time or frequency domain audio representation.This modified features are given as input. To know the most relevant feature to perform the task, deep neural network is approached.The word genre generally represents or relates the style or category of music, art and literature. The result of the music genre is obtained by predicting the music genre of audio signals in the form of a wav file.

The aim of this project is to identify the type of music genre. Millions of people like to hear songs in their day to day life. Most people of all ages listen to music in their leisure time. While analyzing, the features are identified and it is visualized for the clear review of the process. Here, a convolutional neural network algorithm is used to analyze the data set where the accuracy level is high. By implementing these methods, it gives a high accuracy rate while testing and training a lot of data.

CHAPTER 2

LITERATURE SURVEY

2.1 Music genre recognition with deep neural networks

Albert Jimenez, Ferran Jos ´e in 2018 published “Music genre recognition with deep neural networks”[1]. The methods they used are deep learning and convolutional neural networks, multiframe and transfer learning. The main advantage of this is the multiframe approach. Audio files are placed in a confusion matrix to find the mean of all frames. So the classification of genre is more accurate. They have used a two layer convolutional network with mel-spectrogram technique and also raw audio signals as their input features.

For training and fine-tuning the dataset optimizers like adaptive learning rate ADAM and SGD are used. Comparing the results ADAM is chosen to use. On the other hand, when it uses a large amount of data, it takes more time for processing, which is a drawback of this paper.

2.2 Survey on Transfer Learning

Sinno Jialin Pan and Qiang Yang Fellow in 2010 published “Survey on Transfer Learning”[2]. They have used transfer learning, machine learning and data mining techniques. Transfer learning is a technique in which people learn to apply the knowledge they have gained so far in every problem they face in their day to day life. The transfer learning technique is closely related to multi- task learning methods, in which multiple tasks are carried out simultaneously. Machine learning techniques always try to learn from scratch. On the other hand, transfer learning usually transfer knowledge from the training set to target data.

The advantage of this method is across multiple dispersals of the source and the task is improved with different features. It is of three categories; transductive transfer learning, inductive transfer learning and unsupervised transfer learning. There should be a clear difference between source and target domain. The drawback of this is identification and rectification of negative transfer is difficult. transfer learning technique is used in small applications with less number of datas in the field of image and text classification. As a future work it will be used for network analysis, video classification.

2.3 Transfer learning by supervised pre-training for audio-based music classification

Aaron van den Oord, Sander Dieleman, Benjamin Schrauwen in 2014 had published the paper "Transfer learning by supervised pre-training for audio-based music Classification"[3]. They have used transfer learning and neural networks.They have used the Million song dataset(MSD). The transfer learning method has been explored by inserting features of specified audio and labels from various datasets into an allocated space with linear transformations.

That transfer learning works good especially when the main task is tag prediction, i.e. when the target task and source are closely related. The main disadvantage is that the transfer learning method does the given job specifically good only when the source task is tag prediction, i.e. both source and target task are nearly connected.

2.4 Musical Genre Classification of Audio Signals

George Tzanetakis, Student Member, IEEE, and Perry Cook, Member in 2002 published “Musical Genre Classification of Audio Signals”[4]. They used audio classification and feature extraction technology. The different features used in this method are rhythmic content, pitch content and texture. This system mostly targets the signal classification having news, videos in various categories like sound, speech and music. A new method for non-speech song classification which is used to recognise sound effects and instrumental sounds separately. Using the dataset, features such as amplitude, frequency and bandwidth are found using the statistics of mean, autocorrelation and variance.

The main advantage is that to control incorrect higher rate of accuracy which occurs when similar features from the same file in which frames from those files could not be divided between training and testing data. Separate feature sets are designed in order to establish better results for rhythm and harmony. The disadvantage of this paper is that there always arises a discremination between speech, sound and music. Melody extraction and singer voice extraction is a hard problem in this paper.

2.5 Automatic Music Genres Classification using Machine Learning

Muhammad Asim Ali and Zain Ahmed Siddiqui in 2017 has published "Automatic Music Genres Classification using Machine Learning"[5]. They have used k-nearest neighbor (k-NN) and support vector machine algorithms(SVM). They have also used the Mel frequency cepstral coefficients (MFCC) for the extraction of information for the dataset.In addition, they had also performed the comparative analysis between the support vector machine and k-nearest neighbors with and without dimensionality reduction via Principal component analysis(PCA).

The accuracy of SVM is higher than k-NN. Although it also has misidentified disco as rock and reggae as hiphop. K-NN has difficulty in differentiating blues with other genres as it has a lower success rate of 49 percent.

2.6 Music Genre Classification

Rajeeva Shreedhara Bhat ,Rohit B. R. and Mamatha K. R. in 2020 have published "Music Genre classification"[6]. They have used convolutional neural networks. This obtained Mel spectrogram of each track in the GTZAN dataset.This is done using a python package called libROSA. The trained dataset has been preprocessed for each track and for the same the feature vector has been extracted .

The extracted feature vectors generate the feature vector database. The obtained feature vector database will train the neural network database. The accuracy is high as it uses the convolutional neural network. The main disadvantage of this is that it cannot be used for large datasets as it takes more time to give output. As a future work different formats of audio file like mp3, wav, au etc., will be tested and implemented.

2.7 Automatic Music Genre Classification for Indian Music

S. Jothilakshmi, N. Kathiresan in 2012 published the paper "Automatic Music Genre Classification for Indian Music"[7]. The methods they used are Gaussian mixture model(GMM) and k- nearest neighbour (kNN) classifier. Perceptual features, Temporal features, Energy feature and Spectral shape features methods are also introduced. The experiments are conducted for the analysis of different combinations of features for every classifier.

The advantage is that they used different classifiers and parameters to find the genre of the music. Comparing the result of both kNN and GMM they found that GMM gives higher accuracy rate than kNN algorithm. The main disadvantage is that they could analyze only using window features and not consider texture window features. As a future work genre classification for sub genres of each category will be implemented.

2.8 Music Genre Classification using Machine Learning Techniques

Hareesh Bahuleyan in 2018 published the "Music Genre Classification using Machine Learning Techniques"[8]. They have used the convolutional neural network model with VGG-16 which uses the spectrogram for the prediction of music genre. After the experiment they came to know that there is no significant difference between fine-tuning and transfer learning.

The unrolled pixel values used by the baseline feed-forward neural network from the spectrogram performs very poorly on the test set. CNNs were able to improve the scores significantly .The first step involves generating the spectrogram for the audio signal and treating that as an image. An CNN based image classifier, namely VGG-16 is trained on these images to predict the music genre solely based on this spectrogram. The second approach consists of extracting time domain and frequency domain features from the audio signals, followed by training traditional machine learning classifiers based on these features.

CHAPTER 3

METHODOLOGY

3.1 FRAMEWORK

- **Audio libraries: librosa**

The package of python for music and audio analysis is librosa. It gives the building blocks which are important for the creation of musical information retrieval systems. It uses sound files and audioread to load audio files. It should be considered that soundfile does not support MP3, it leads librosa to fail.

- **Deep learning framework: Keras and Theano**

Keras is an effective and open source Python library for the establishment and assessment of deep learning models. It wraps the efficient numerical computation libraries Theano and TensorFlow and allows you to define and train neural network models in just a few lines of code. Theano is a Python library that allows you to define, optimize, and efficiently evaluate mathematical expressions involving multi-dimensional

- **Music datasets:**
    - A million song dataset
    - GTZAN dataset
    - Handmade dataset

3.2 DATASET

The dataset created by Tzanetakis et al is GTZAN. It is composed of 1000 different music of 30 seconds duration with 100 examples in each of 10 different music genres like Classical, Country,Blues, Disco, Hip Hop, Metal,Popular,Jazz, Reggae and Rock. These files were collected from a variety of sources including personal microphone recordings, CDs, radio in order to represent a variety of recording conditions. They are all 22050 Hz Mono 16-bit audio files in .wav format. Our approach has one important limitation. The duration of each music excerpt (30s) makes that only one frame per song can be extracted and without discarding anything neither at the beginning nor in the end of the song.

3.3 Convolutional Neural Networks

In music genre classification,various tasks such as music tagging , genre classification, and user-item latent feature prediction are used for recommendation. To identify these tasks we use convolutional neural networks. CNNs always process the in different features hierarchy levels and the source can be extracted by convolutional kernels. The hierarchical sources acquire knowledge to complete the given functions that are learned during supervised training. To save the user's time in searching for songs CNN is used. It is mainly used for its high accuracy. Compared to K-NN and Support vector machines , CNN has more accuracy.

3.4 Classification Algorithm

Logistic Regression (LR)

Binary classification tasks are done by linear classifiers. A one-vs-rest method is executed for logistic regression in multi-class classification. Usually binary classifiers are trained in different methods separately. Here it undergoes 8 separate classifiers.After testing, the division with more probability among all the 8 classifiers is predicted as the final result.

Simple Artificial Neural Network (ANN)

ANN is not directly under the human control computing system which is planned to energize in the same way as the human brain works to support and process information . Processing unit built Artificial neural network composed of input and output.The input units get numerous forms and various structural information are built on an internal weight system and the neural network seeks to acquire knowledge about the content given to construct the output. Firstly, the Artificial Neural Network undergoes the phase of training which learns to understand the patterns of the dataset. Information that passes through the network transforms the structure of the ANN.

3.5 CNN brief explanation

CNNs learning part is going to work with these filters. As the learning weights which are in a MLP, Convolutional neural networks will also come to learn the most of the optimal filters for the recognition of specific objects and patterns. On the other hand CNN won't learn one filter,it learns multiple filters. Actually it learns multiple filters in each layer.

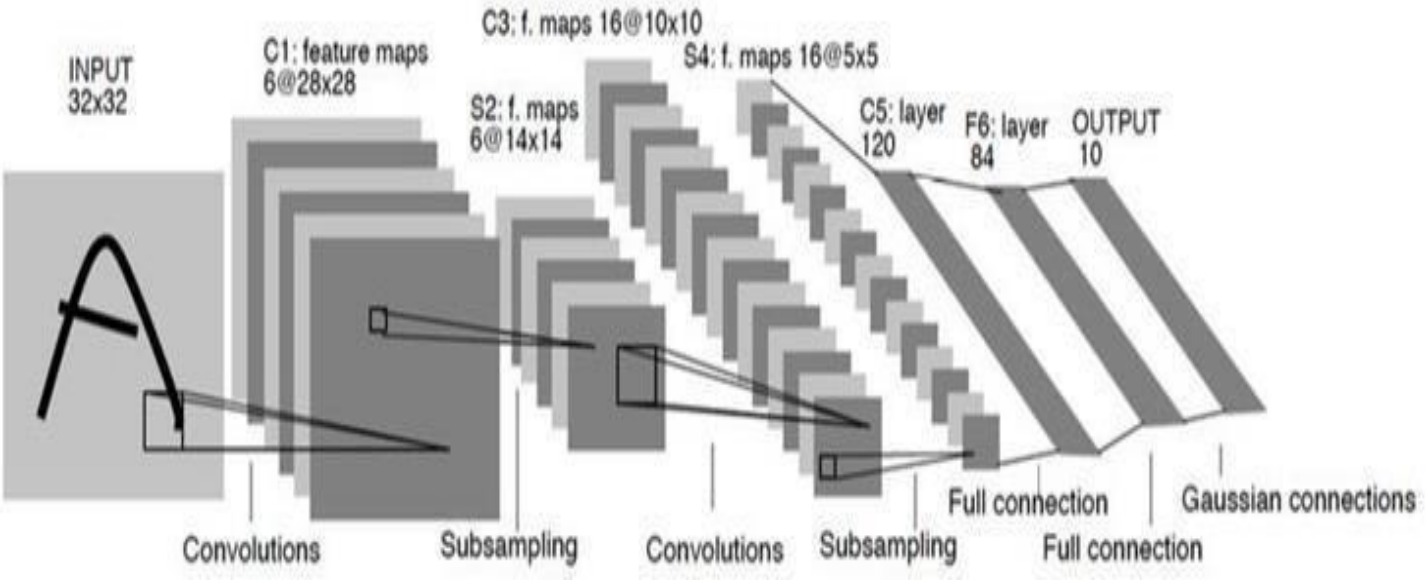
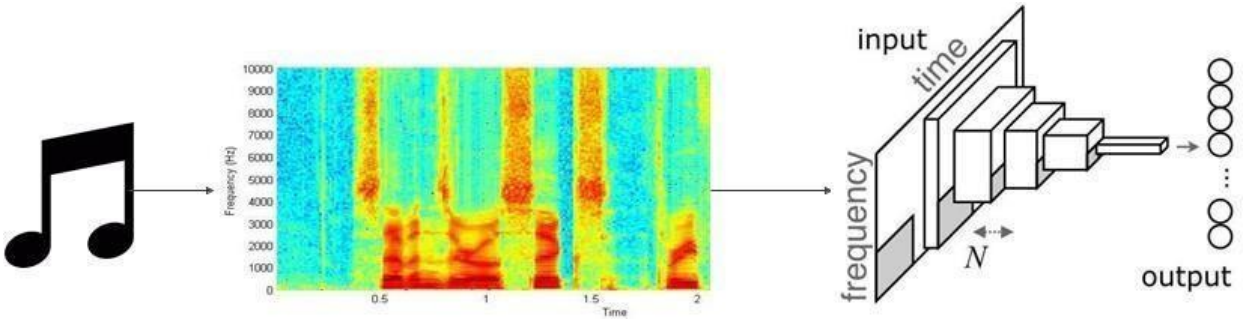


Figure 3.5.1 Diversity of filters within convolutional layer

In the figure 3.5.1, a pooling or subsampling layer often immediately accompanies a convolution layer in a convolutional neural network. Its work is to downsample the convolution layer's output along both the spatial dimensions of width and height. The fully connected layers act as classifiers.

CNN for Music classification

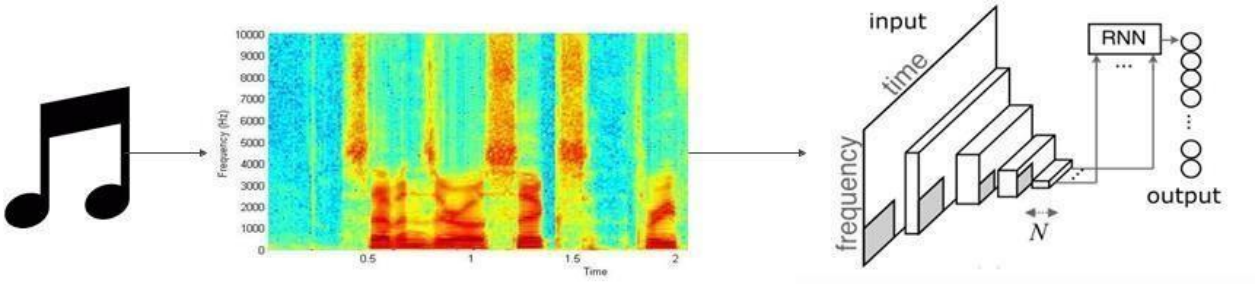


Song Audio file Mel-Spectrogram Feature extraction and classification

Figure 3.5.2. CNN for Music classification

In the figure 3.5.2, the song audio file which is in waveform is converted into mel-spectrogram. The mel-spectrogram compresses the audio file and then the feature extraction and classification takes place.

CRNNs for Music classification



Song Audio file Mel-Spectrogram Feature extraction and classification

Figure 3.5.3. CRNNs for Music classification

In the figure 3.5.3, the song, wav audio file after compresses and feature extraction from mel- spectrogram, it is pre-trained from low level features to high level features and it only changes the classifier which also fine tunes the weights of the given network.

CHAPTER 4

SYSTEM IMPLEMENTATION

CNN and CRNN are used to fine tune the audio signals initially. Firstly, the lowest layer is freezed and the consecutive top layers are fine tuned to see the variations.To increase adaptive different optimizers are used. In the case of CNN, we can analyze that the process of overfitting is very fast. To get the finite result, the classifier layer should be trained in the top of the network. On the other side, the behaviour of CRNN is more similar to CNN. Here, overfitting occurs when the last convolutional layer and recurrent layer are fine tuned.

To find the genre of music, the wav format audio file is selected, two different steps are carried out. Firstly, a short period is deleted both in the starting and in the ending of the song. This is for the filtration process, it is carried to remove vibrato singing, ringing ambience, vocal thirds and bass drum sounds. If these processes do not undergo, their overfitting to the classification process would give weaker results. The following step consists in dividing the remaining part of the song in frames of 29.12s. These frames should not intersect with one another and the final frame is also deleted if the duration is less than 29.12s. Finally, once obtained all the division of frames, the evaluation part is set at frame level or at song level.

Accuracy metric is used to measure the performance of our system. When evaluation is at song level, the average of the predicted tags for every frame of the song needs to be found. In order to do that, the mean among the tag scores of all the frames of the song is calculated and the tag with higher score is marked as the final result. Therefore, if a song has a small duration that can be grouped as a different genre, there will be no change in the final song classification. Another approach that would lead to a unique tag per song would be the nearest neighbors algorithm between the tags of all the frames of the song. Therefore, the most repeated tag among all the frames of the song would be selected as the tag of the song.

CHAPTER 5

RESULTS AND DISCUSSION

The model to evaluate the performance of our multiframe approach has been trained with the train partition of our handmade dataset and evaluated using the test partition. The genre prediction has been made using the recurrent layers fine-tuned CRNN model .

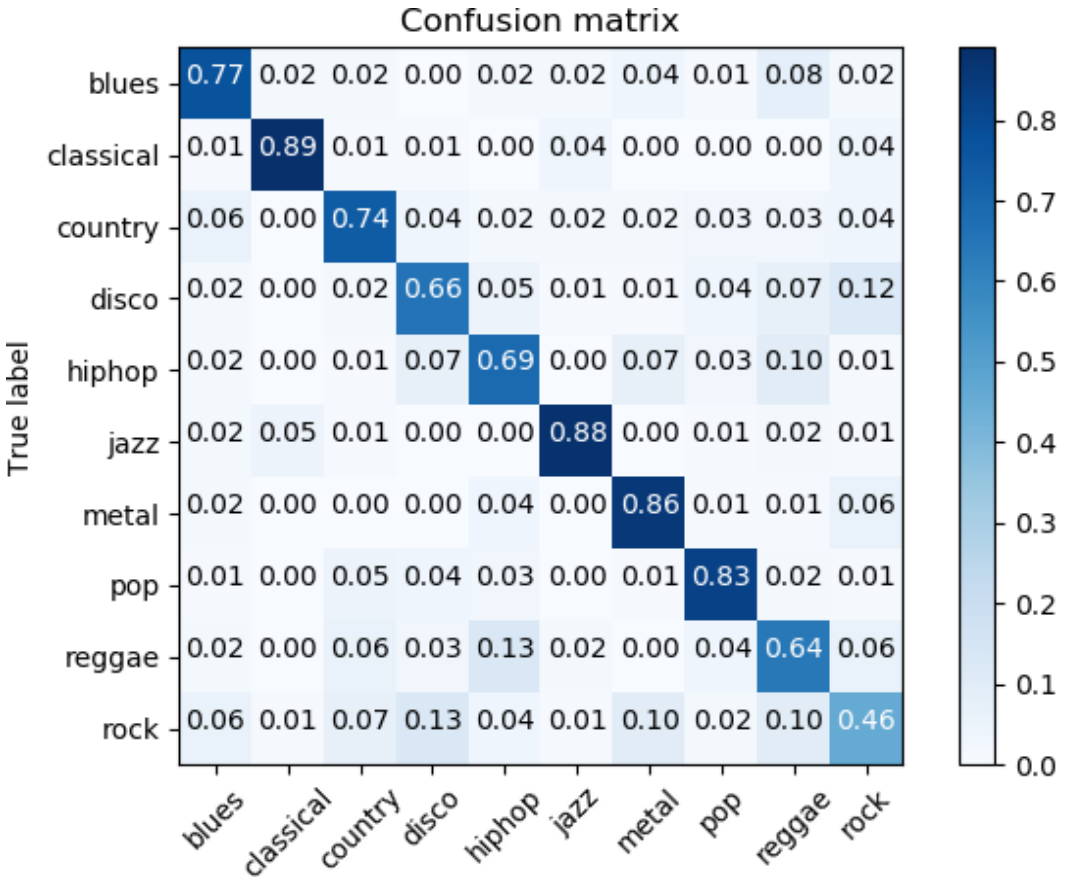


Figure 5.1. Confusion Matrix

In figure 5.1 the confusion matrix between the real genre and the predicted genre has been built for two different scenarios: using the predicted tag of each frame of the test database, and using the predicted tag of the average score obtained for each song of the test database are shown.

The dataset for testing consists of audio samples of 250 songs from each of 10 genres. The results are shown as follows:

Genre	Classical	Pop	Rock	Jazz	Hiphop	Country	Blues	Metal	Reggae	Disco	**Recall**
Classical	167	17	12	9	2	15	13	5	10	0	66.9%
Pop	3	138	0	10	13	35	6	18	19	8	53.5%
Rock	4	0	234	4	0	0	8	0	0	0	93.8%
Jazz	0	5	0	200	10	12	15	0	8	0	80.0%
Hiphop	0	20	1	25	142	0	20	12	10	18	56.9%
Country	0	0	3	6	5	235	0	0	1	0	94.3%
Blues	8	12	0	8	9	0	209	3	0	1	83.8%
Metal	5	12	4	30	58	0	0	124	8	9	49.8%
Reggae	8	9	0	35	30	0	0	0	168	0	67.2%
Disco	0	5	0	5	12	0	8	0	6	214	85.8%
Precision	83.5%	63.3 %	92.12 %	60.2 4%	50.53 %	64.3%	74.9 %	76.5 %	73%	85.6%	

Overall accuracy rate: 73.2%

The model which was implemented has the higher rate and accurately recognizes genre Rock, Disco and Blues with recall value of 93.8%, 85.8% and 83.8% respectively. However, it is not good and much less in recognizing music of Metal and Pop with lower recall value of 49.8% and 53.5% respectively. Blues and Metal roughly have a similar precision rate near 75%, while Hiphop has the lower precision rate of 50.53%.

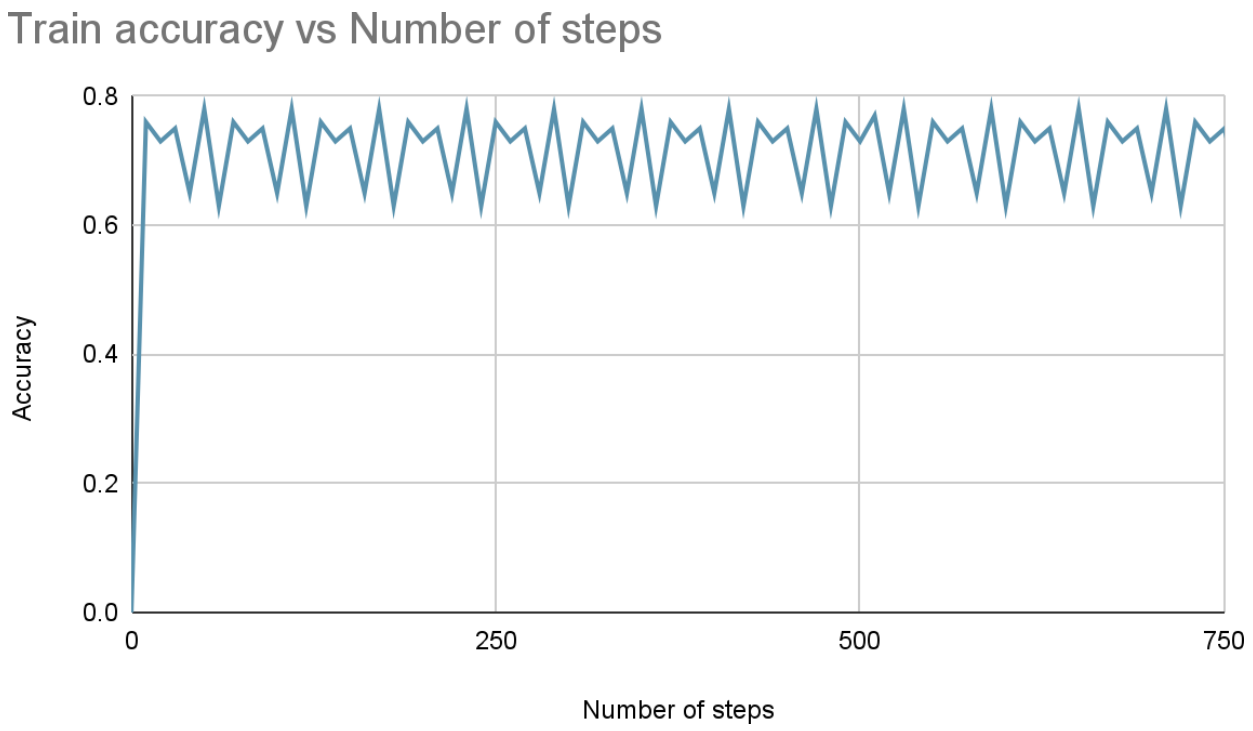


Figure 5.2. Train accuracy vs Number of steps

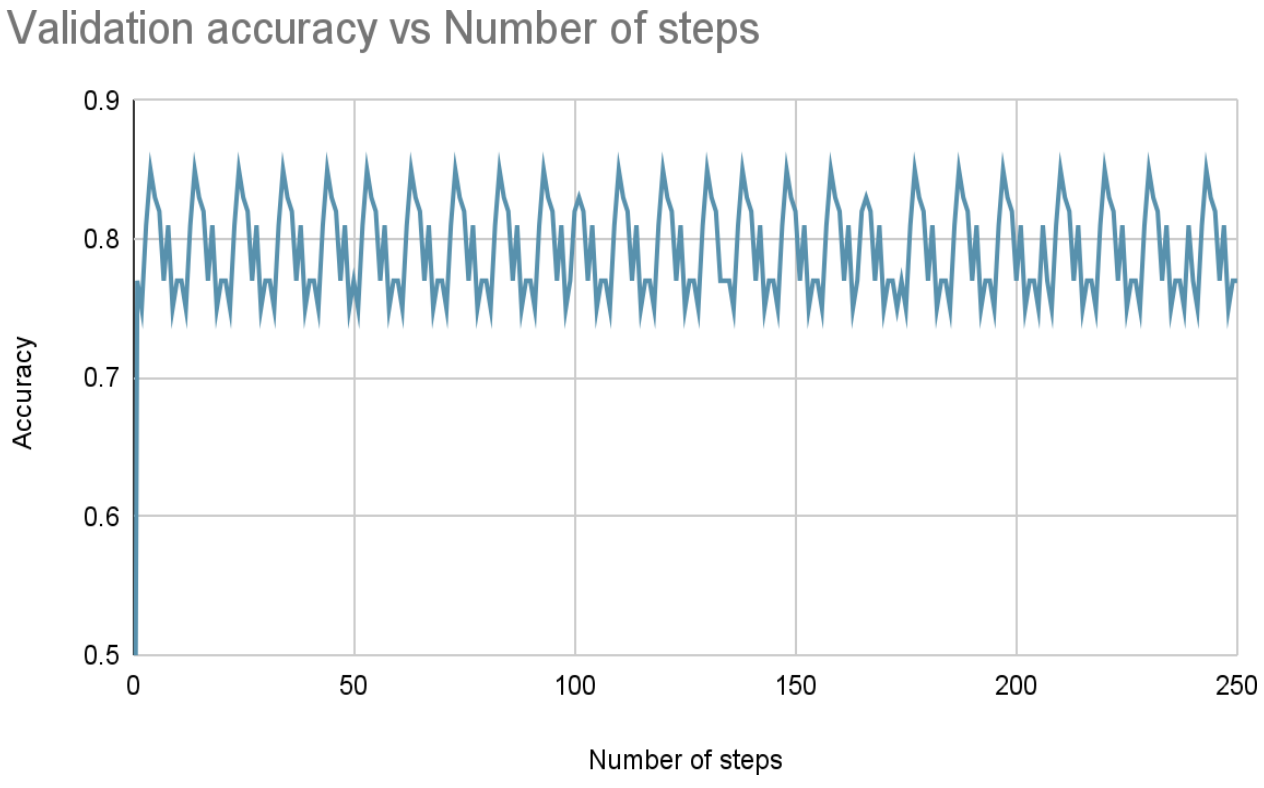


Figure 5.3. Validation accuracy vs number of steps

Train accuracy rate is approximately 12% lower than the validation accuracy that it may occur due to overfitting. the song which which we chosen for training will not be included in the test samples so that the differences may appear

Performances of each genre have small relative differences between them. Rock and disco songs have different vocal styling, sounds and tunes so that it is easier to differentiate. On the other hand Country and Jazz music are more confusing to differentiate as they use similar types of instruments, tunes. Here, comes our partial output; however a genre is misclassified as another, there may be some reason for its specific performances.

Finally, a web page is hosted using Laragon. The web page is of three sections: Home; Check genre; Statistics. Once the audio file is uploaded the output will be shown. In this project, 1000 sets of samples are taken, in which 750 samples are trained and 250 samples are tested which gives the accuracy of 73.2%

CHAPTER 6

CONCLUSION AND FUTURE WORK

The application of CNN and CRNN in the case of music genre classification are explored. In the scenario of having a small dataset and a task to perform, transfer learning can be used to fine-tune models that have been trained on large datasets and for other different purposes. Therefore, using the average stage non-representative frames dependence can be removed. This method requires a large amount of dataset which needs to be trained from scratch. In case of having small data for transfer learning, the accuracy will be lower.

For each track of the GTZAN dataset Mel Spectrum is obtained. This can be done by the python package library libROSA. A software system is implemented to perform the process of classifying the music according to their genre. Multiframe approach is used to improve the single- frame songs with an average state. The collection of 1000 songs of 10 different genres are experimented in which 7 of these 10 meet with the highest accuracy rate.

Future Work

As a future work, some other techniques to to find the subgenre in all these types of genre. By implementing this the complete package of each and every music has been found. Another extension in this work would be a methodology for analysing all types of input formats such as wav, mp3, au, etc., which will be tested on a common platform.

REFERENCES

[1] Albert Jimenez, Ferran Jos´e, “Music genre recognition with deep neural networks”, Universitat Politècnica de Catalunya,2018.

[2] Sinno Jialin Pan and Qiang Yang Fellow, “A Survey on Transfer Learning”, 2010.

[3] Aaron van den Oord, Sander Dieleman, Benjamin Schrauwen, "Transfer learning by supervised pre-training for audio-based music Classification", 2014.

[4] George Tzanetakis, Student Member, IEEE, and Perry Cook, “Musical Genre Classification of Audio Signals”, 2002.

[5] Muhammad Asim Ali, Zain Ahmed Siddiqui, “Automatic Music Genres Classification using Machine Learning”, 2017.

[6] Rajeeva Shreedhara Bhat, Rohit B. R., Mamatha K. R., “Music Genre Classification”, 2020.

[7] S. Jothilakshmi, N. Kathiresan, “Automatic Music Genre Classification for Indian Music” in 2012.

[8] Hareesh Bahuleyan, “Music Genre Classification using Machine Learning Techniques” in 2018.

APPENDIX 1

SOURCE CODE:

PREPROCESSING:

```
from django.contrib import admin
from .models import Document
# Register your models here.
admin.site.register(Document)
from django.apps import AppConfig
from django.conf import settings
import os
import pickle
class PredictorConfig(AppConfig):
    # create path to models
    pass
from django import forms
from .models import Document
class DocumentForm(forms.Form):
    def is_valid(*args, **kwargs):
        return True
    file = forms.FileField(help_text='Valid .mp3 file')
def getmetadata(filename):
    import librosa
    import numpy as np
    y, sr = librosa.load(filename)
    #fetching tempo
    onset_env = librosa.onset.onset_strength(y, sr)
    tempo = librosa.beat.tempo(onset_envelope=onset_env, sr=sr)
```

```
    #fetching beats
    y_harmonic, y_percussive = librosa.effects.hpss(y)
    tempo, beat_frames = librosa.beat.beat_track(y=y_percussive,sr=sr)

    #chroma_stft
    chroma_stft = librosa.feature.chroma_stft(y=y, sr=sr)
    #rmse
    rmse = librosa.feature.rms(y=y)
    #fetching spectral centroid
    spec_centroid = librosa.feature.spectral_centroid(y, sr=sr)[0]
    #spectral bandwidth
    spec_bw = librosa.feature.spectral_bandwidth(y=y, sr=sr)
    #fetching spectral rolloff
    spec_rolloff = librosa.feature.spectral_rolloff(y+0.01, sr=sr)[0]
    #zero crossing rate
    zero_crossing = librosa.feature.zero_crossing_rate(y)
    #mfcc
    mfcc = librosa.feature.mfcc(y=y, sr=sr)
    #metadata dictionary
    metadata_dict = {'tempo':tempo,'chroma_stft':np.mean(chroma_stft),'rmse':np.mean(rmse),
                     'spectral_centroid':np.mean(spec_centroid),'spectral_bandwidth':np.mean(spec_bw),
                     'rolloff':np.mean(spec_rolloff), 'zero_crossing_rates':np.mean(zero_crossing)}
    for i in range(1,21):
        metadata_dict.update({'mfcc'+str(i):np.mean(mfcc[i-1])})

    return list(metadata_dict.values())
from django.db import models

# Create your models here.
class Document(models.Model):
    file = models.FileField(upload_to='file/')
```

```
def predict_gen(meta1):
    import pickle
    import os
    from django.conf import settings
    path = os.path.join(settings.MODELS, 'models.p')
    with open(path, 'rb') as pickled:
        data = pickle.load(pickled)
    svmp = data['svmp']
    norma = data['norma']
    lgn = data['lgn']
    x = norma.transform([meta1])
    pred = svmp.predict(x)
    return(lgn[pred[0]])
from django.urls import path
from . import view
app_name = 'predictor
urlpatterns = [
    path('', views.IndexView.as_view(), name='index'),
    path('result/',views.model_form_upload,name = 'result'),
]
from django.shortcuts import render, redirect
from django.http import HttpResponse, HttpResponseRedirect
from django.urls import reverse
from django.views.generic import ListView
from .apps import PredictorConfig
from .forms import DocumentForm
from .models import Document
from .Metadata import getmetadata
import warnings
from .predict import predict_gen
from django.contrib import messages
warnings.simplefilter('ignore')
```

```
class IndexView(ListView):
    template_name= 'music/index.html'
    def get_queryset(self):
        return True

def model_form_upload(request):

    documents = Document.objects.all()
    if request.method == 'POST':
        if len(request.FILES) == 0:
            messages.error(request,'Upload a file')
            return redirect("predictor:index")

        form = DocumentForm(request.POST, request.FILES)
        if form.is_valid():
            uploadfile = request.FILES['document']
            print(uploadfile.name)
            print(uploadfile.size)
            if not uploadfile.name.endswith('.wav'):
                messages.error(request,'Only .wav file type is allowed')
                return redirect("predictor:index")
            meta = getmetadata(uploadfile)

            genre = predict_gen(meta)
            print(genre)

            context = {'genre':genre}
            return render(request,'music/result.html',context)

    else:
        form = DocumentForm()
```

```
    return render(request,'music/result.html',{'documents':documents,'form':form}
{

{
 "cells": [{
   "cell_type": "code",
   "execution_count": 1,
   "metadata": {},
   "outputs": [],
   "source": [
    "import pandas as pd\n",
   "import numpy as np"]},{
   "cell_type": "code",
   "execution_count": 2,
   "metadata": {},
   "outputs": [],
   "source": [
    "df = pd.read_csv('data.csv')"]},{
   "cell_type": "code",
   "execution_count":  3,
   "metadata": {},
   "outputs": [],
   "source": [
    "df = df.drop(['beats'], axis=1)"]},{
   "cell_type": "code",
```

```
"execution_count": 4,
"metadata": {
"scrolled": true},
"outputs": [{
  "data": {
   "text/html": [
        "<div>\n",
        "<style scoped>\n",
        "    .dataframe tbody tr th:only-of-type {\n",
        "    vertical-align: middle;\n","          }\n","\n",
        "    .dataframe tbody tr th {\n",
        "    vertical-align: top;\n","          }\n",  "\n",
        "    .dataframe thead th {\n",
        "          text-align: right;\n",    "    }\n",],
        "[5 rows x 29 columns]"]},
  "execution_count": 4,
  "metadata": {},
  "output_type": "execute_result"}],
"source": [
 "df.head()"]},{
"cell_type": "markdown",
"metadata": {},
"source": [
 "# Preprocessing"]},{
"cell_type": "code",
```

"execution_count": 5,

"metadata": {},

"outputs": [{

"data": {

"text/plain": [

"array(['blues', 'classical', 'country', 'disco', 'hiphop', 'jazz',\n",

"   'metal', 'pop', 'reggae', 'rock'], dtype=object)"]},

"execution_count": 5,

"metadata": {},

"output_type": "execute_result"}],

"source": [

"df['class_name'].unique()"]},{

"cell_type": "code",

"execution_count": 6,

"metadata": {},

"outputs": [],

"source": [

"df['class_name'] = df['class_name'].astype('category')\n",

"df['class_label'] = df['class_name'].cat.codes"]},{

"cell_type": "code",

"execution_count": 7,

"metadata": {

"scrolled": true},

"execution_count": 7,

"metadata": {},

```
  "output_type": "execute_result"}],
"source": [
 "lookup_genre_name = dict(zip(df.class_label.unique(), df.class_name.unique())) \n",
 "lookup_genre_name"] },{
"cell_type": "code",
"execution_count": 8,
"metadata": {},
"outputs": [{
  "data": {
   "text/plain": [
        "[blues, classical, country, disco, hiphop, jazz, metal, pop, reggae, rock]\n",
        "Categories (10, object): [blues, classical, country, disco, ..., metal, pop, reggae, rock]" ]},
  "execution_count": 8,
  "metadata": {},
  "output_type": "execute_result"}],
"source": [
"df['class_name'].unique()"]},{
"cell_type": "code",
"execution_count": 9,
"metadata": {},
"outputs": [],
"source": [
 "cols = list(df.columns)\n",
 "cols.remove('label')\n",
 "cols.remove('class_label')\n",
```

```
 "cols.remove('class_name')\n",
 "#df[cols]"]},{
"cell_type": "markdown",
"metadata": {},
"source": [
 "# Data Splitting for Training and Testing" ]},{
"cell_type": "code",
"execution_count": 10,
"metadata": {},
"outputs": [],
"source": [
 "%matplotlib notebook\n",
 "from sklearn.model_selection import train_test_split\n",
 "import matplotlib.pyplot as plt\n",
 "X = df.iloc[:,1:28]\n",
 "y = df['class_label']\n",
 "X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=3)" ]},{
"cell_type": "markdown",
"metadata": {},
"source": [
 "# Min-Max Normalization"]},{
"cell_type": "code",
"execution_count": 11,
"metadata": {},
"outputs": [],
```

"source": [
 "from sklearn.preprocessing import MinMaxScaler\n",
 "scaler = MinMaxScaler()\n",
 "X_train_scaled = scaler.fit_transform(X_train)\n",
 "# we must apply the scaling to the test set that we computed for the training set\n",
 "X_test_scaled = scaler.transform(X_test)"]},{
"cell_type": "markdown",
"metadata": {},
"source": [
 "# Feature importance using Random Forest"]},{
"cell_type": "code",
"execution_count": 12,
"metadata": {},
"outputs": [{
  "name": "stderr",
  "output_type": "stream",},{
  "data": {
  "application/javascript": [
        "/* Put everything inside the global mpl namespace */\n",
        "window.mpl = {};\n","\n","\n"
        "mpl.get_websocket_type = function() {\n",
        "   if (typeof(WebSocket) !== 'undefined') {\n",
        "   return WebSocket;\n",
        "   } else if (typeof(MozWebSocket) !== 'undefined') {\n",
        "   return MozWebSocket;\n",

```
"    } else {\n",
"        alert('Your browser does not have WebSocket support. ' +\n",
"         'Please try Chrome, Safari or Firefox ≥ 6. ' +\n",
"         'Firefox 4 and 5 are also supported but you ' +\n",
"        'have to enable WebSockets in about:config.');\n",
"  };\n","}\n","\n",
"mpl.figure = function(figure_id, websocket, ondownload, parent_element) {\n",
" this.id = figure_id;\n","\n",
"    this.ws = websocket;\n","\n",
"    this.supports_binary = (this.ws.binaryType != undefined);\n","\n",
"    if (!this.supports_binary) {\n",
"        var warnings = document.getElementById(\"mpl-warnings\");\n",
"        if (warnings) {\n",
"         warnings.style.display = 'block';\n",
"    warnings.textContent = (\n","}\n","  }\n","\n", "
    this.imageObj = new Image();\n","\n",
"        this.root = $('<div/>');\n",
"    this._root_extra_style(this.root)\n",
"    this.root.attr('style', 'display: inline-block');\n","\n",
"    $(parent_element).append(this.root);\n","\n",
"    this._init_header(this);\n",
"    this._init_canvas(this);\n",
"    this._init_toolbar(this);\n","\n",
"    var fig = this;\n","\n",
"   this.waiting = false;\n","\n",
```

```
"    this.ws.onopen = function () {\n",
"            fig.send_message(\"supports_binary\", {value: fig.supports_binary});\n",
"            fig.send_message(\"send_image_mode\", {});\n",
"        if (mpl.ratio != 1) {\n",
"            fig.send_message(\"set_dpi_ratio\", {'dpi_ratio': mpl.ratio});\n","    }\n",
"            fig.send_message(\"refresh\", {});\n","    }\n","\n",
"    this.imageObj.onload = function() {\n",
"    if (fig.image_mode == 'full') {\n"
"            fig.context.clearRect(0, 0, fig.canvas.width, fig.canvas.height);\n","    }\n", "
            fig.context.drawImage(fig.imageObj, 0, 0);\n","    };\n","\n",
"    this.imageObj.onunload = function() {\n",
"    fig.ws.close();\n","    }\n","\n",
"    this.ws.onmessage = this._make_on_message_function(this);\n","\n",
"    this.ondownload = ondownload;\n","}\n","\n",
"mpl.figure.prototype._init_header = function() {\n",
"    var titlebar = $(\n",
"        '<div class=\"ui-dialog-titlebar ui-widget-header ui-corner-all ' +\n",
"        'ui-helper-clearfix\"/>');\n",
"    var titletext = $(\n",
"            '<div class=\"ui-dialog-title\" style=\"width: 100%; ' +\n",
"            'text-align: center; padding: 3px;\"/>');\n",
"    titlebar.append(titletext)\n",
"    this.root.append(titlebar);\n",
"    this.header = titletext[0];\n","}\n","\n","\n","\n",
"mpl.figure.prototype._canvas_extra_style = function(canvas_div) {\n","\n","}\n","\n","\n",
```

```
"mpl.figure.prototype._root_extra_style = function(canvas_div) {\n","\n","}\n","\n",
"mpl.figure.prototype._init_canvas = function() {\n",
"    var fig = this;\n","\n",
"    var canvas_div = $('<div/>');\n","\n",
"    canvas_div.attr('style', 'position: relative; clear: both; outline: 0');\n","\n",
"    function canvas_keyboard_event(event) {\n",
"        return fig.key_event(event, event['data']);\n","  }\n","\n",
"    canvas_div.keydown('key_press', canvas_keyboard_event);\n",
"    canvas_div.keyup('key_release', canvas_keyboard_event);\n",
"    this.canvas_div = canvas_div\n",
"    this._canvas_extra_style(canvas_div)\n",
"    this.root.append(canvas_div);\n","\n",
"    var canvas = $('<canvas/>');\n",
"    canvas.addClass('mpl-canvas');\n",
"    canvas.attr('style', \"left: 0; top: 0; z-index: 0; outline: 0\")\n","\n",
"    this.canvas = canvas[0];\n",
"    this.context = canvas[0].getContext(\"2d\");\n","\n",
"    mpl.ratio = (window.devicePixelRatio || 1) / backingStore;\n","\n", "
    var rubberband = $('<canvas/>');\n",
"    rubberband.attr('style', \"position: absolute; left: 0; top: 0; z-index: 1;\")\n","\n",
"    var pass_mouse_events = true;\n","\n",
"    canvas_div.resizable({\n",
"        start: function(event, ui) {\n",
"        pass_mouse_events = false;\n","  },\n",
"        resize: function(event, ui) {\n",
```

```
"         fig.request_resize(ui.size.width, ui.size.height);\n"," },\n",
"         stop: function(event, ui) {\n",
"          pass_mouse_events = true;\n",
"    fig.request_resize(ui.size.width, ui.size.height);\n"," },\n"," });\n","\n", "
    function mouse_event_fn(event) {\n",
"         if (pass_mouse_events)\n",
"    return fig.mouse_event(event, event['data']);\n","}\n","\n", "
    rubberband.mousedown('button_press', mouse_event_fn);\n",
"    rubberband.mouseup('button_release', mouse_event_fn);\n",
"    rubberband.mousemove('motion_notify', mouse_event_fn);\n","\n",
"    rubberband.mouseenter('figure_enter', mouse_event_fn);\n",
"    rubberband.mouseleave('figure_leave', mouse_event_fn);\n","\n",
"    canvas_div.on(\"wheel\", function (event) {\n",
"         event = event.originalEvent;\n",
"         event['data'] = 'scroll'\n",
"         if (event.deltaY < 0) {\n",
"         event.step = 1;\n",
"         } else {\n",
"          event.step = -1;\n","}\n",
"    mouse_event_fn(event);\n","});\n","\n", "
    canvas_div.append(canvas);\n",
"    canvas_div.append(rubberband);\n","\n",
"    this.rubberband = rubberband;\n",
"    this.rubberband_canvas = rubberband[0];\n",
"    this.rubberband_context = rubberband[0].getContext(\"2d\");\n",
```

```
"    this.rubberband_context.strokeStyle = \"#000000\";\n","\n",

"    this._resize_canvas = function(width, height) {\n",

"        canvas_div.css('width', width)\n",

"        canvas_div.css('height', height)\n","\n",

"        canvas.attr('width', width * mpl.ratio);\n",

"        canvas.attr('height', height * mpl.ratio);\n",

"        canvas.attr('style', 'width: ' + width + 'px; height: ' + height + 'px;');\n","\n",

"        rubberband.attr('width', width);\n",

"    rubberband.attr('height', height);\n","}\n","\n", "

    this._resize_canvas(600, 600);\n","\n",

"    // Disable right mouse context menu.\n",

"    $(this.rubberband_canvas).bind(\"contextmenu\",function(e){\n",

"    return false;\n","});\n"\n",

"    function set_focus () {\n",

"    canvas.focus();\n",

"        canvas_div.focus();\n","}\n","\n",

"    window.setTimeout(set_focus, 100);\n","}\n","\n",

"mpl.figure.prototype._init_toolbar = function() {\n",

"    var fig = this;\n","\n",

"    var nav_element = $('<div/>');\n",

"    nav_element.attr('style', 'width: 100%');\n",

"    this.root.append(nav_element);\n","\n",

"    function toolbar_event(event) {\n",

"    return fig.toolbar_button_onclick(event['data']);\n","}\n", "

    function toolbar_mouse_event(event) {\n",
```

```
"    return fig.toolbar_button_onmouseover(event['data']);\n","}\n","\n", "
    for(var toolbar_ind in mpl.toolbar_items) {\n",
"        var name = mpl.toolbar_items[toolbar_ind][0];\n",
"        var tooltip = mpl.toolbar_items[toolbar_ind][1];\n",
"        var image = mpl.toolbar_items[toolbar_ind][2];\n",
"        var method_name = mpl.toolbar_items[toolbar_ind][3];\n","\n",
"        if (!name) {\n",
"         continue;\n","}\n",
"        var button = $('<button/>');\n",
"        button.addClass('ui-button ui-widget ui-state-default ui-corner-all ' +\n",
"        'ui-button-icon-only');\n",
"        button.attr('ole', 'button');\n",
"        button.attr('aria-disabled', 'false');\n",
"        button.click(method_name, toolbar_event);\n",
"        button.mouseover(tooltip, toolbar_mouse_event);\n","\n",
"        var icon_img = $('<span/>');\n",
"      icon_img.addClass('ui-button-icon-primary ui-icon');\n",
"      icon_img.addClass(image);\n",
"      icon_img.addClass('ui-corner-all');\n","\n",
"      var tooltip_span = $('<span/>');\n",
"      tooltip_span.addClass('ui-button-text');\n",
"      tooltip_span.html(tooltip);\n","\n",
"        button.append(icon_img);\n",
"        button.append(tooltip_span);\n","\n",
"        nav_element.append(button);\n","}\n","\n",
```

```
"    var fmt_picker_span = $('<span/>');\n","\n",
"    var fmt_picker = $('<select/>');\n",
"    fmt_picker.addClass('mpl-toolbar-option ui-widget ui-widget-content');\n",
"    fmt_picker_span.append(fmt_picker);\n",
"    nav_element.append(fmt_picker_span);\n",
"    this.format_dropdown = fmt_picker[0];\n","\n",
"    for (var ind in mpl.extensions) {\n",
"        var fmt = mpl.extensions[ind];\n",
"        var option = $(\n",
"        '<option/>', {selected: fmt === mpl.default_extension}).html(fmt);\n",
"        fmt_picker.append(option);\n","}\n","\n",
"    $( \".ui-button\" ).hover(\n",
"        function() { $(this).addClass(\"ui-state-hover\");},\n",
"    function() { $(this).removeClass(\"ui-state-hover\");}\n"," );\n","\n", "
    var status_bar = $('<span class=\"mpl-message\"/>');\n",
"    nav_element.append(status_bar);\n",
"    this.message = status_bar[0];\n","}\n","\n",
"mpl.figure.prototype.request_resize = function(x_pixels, y_pixels) {\n",
"        this.send_message('resize', {'width': x_pixels, 'height': y_pixels});\n","}\n","\n",
"mpl.figure.prototype.send_message = function(type, properties) {\n",
"    properties['type'] = type;\n",
"    properties['figure_id'] = this.id;\n",
"    this.ws.send(JSON.stringify(properties));\n","}\n","\n",
"mpl.figure.prototype.send_draw_message = function() {\n",
"    if (!this.waiting) {\n",
```

```
"          this.waiting = true;\n",

"          this.ws.send(JSON.stringify({type: \"draw\", figure_id:
this.id}));\n","}\n","}\n","\n","\n",

"mpl.figure.prototype.handle_save = function(fig, msg) {\n",

"    var format_dropdown = fig.format_dropdown;\n",

"    var format =format_dropdown.options[format_dropdown.selectedIndex].value;\n",

"    fig.ondownload(fig, format);\n","}\n","\n","\n",

"mpl.figure.prototype.handle_resize = function(fig, msg) {\n",

"    var size = msg['size'];\n",

"    if (size[0] != fig.canvas.width || size[1] != fig.canvas.height) {\n",

"    fig._resize_canvas(size[0], size[1]);\n",

"          fig.send_message(\"refresh\", {});\n","};\n","}\n","\n",

"          fig.rubberband_context.clearRect(\n",

"          0, 0, fig.canvas.width / mpl.ratio, fig.canvas.height / mpl.ratio);\n","\n",

"    fig.rubberband_context.strokeRect(min_x, min_y, width, height);\n","}\n","\n",

"mpl.figure.prototype.handle_figure_label = function(fig, msg) {\n",

"    fig.header.textContent = msg['label'];\n","}\n","\n",

"mpl.figure.prototype.handle_cursor = function(fig, msg) {\n",

"    var cursor = msg['cursor'];\n",

"    switch(cursor)\n","{\n",

"    case 0:\n",

"          cursor = 'pointer';\n",

"          break;\n",

"    case 1:\n",

"          cursor = 'default';\n",
```

```
"            break;\n",
"    case 2:\n",
"            cursor = 'crosshair';\n",
"            break;\n",
"    case 3:\n",
"            cursor = 'move';\n",
"            break;\n","}\n",
"    fig.rubberband_canvas.style.cursor = cursor;\n","}\n","\n",
"mpl.figure.prototype.handle_message = function(fig, msg) {\n",
"    fig.message.textContent = msg['message'];\n","}\n","\n",
"mpl.figure.prototype.handle_draw = function(fig, msg) {\n",
"    fig.send_draw_message();\n","}\n","\n",
"mpl.figure.prototype.handle_image_mode = function(fig, msg) {\n",
"    fig.image_mode = msg['mode'];\n","}\n","\n",
"mpl.figure.prototype.updated_canvas_event = function() {\n",
"    this.send_message(\"ack\", {});\n","}\n","\n",
"mpl.figure.prototype._make_on_message_function = function(fig) {\n",
"    return function socket_on_message(evt) {\n",
"            if (evt.data instanceof Blob) {\n",
"            /* FIXME: We get \"Resource interpreted as Image but\n",
"            * transferred with MIME type text/plain:\" errors on\n",
"            * Chrome. But how to set the MIME type? It doesn't seem\n",
"            * to be part of the websocket stream */\n",
"            evt.data.type = \"image/png\";\n","\n",
"            if (fig.imageObj.src) {\n",
```

```
"                    (window.URL || window.webkitURL).revokeObjectURL(\n",
"                    fig.imageObj.src);\n","}\n","\n",
"          fig.imageObj.src = (window.URL || window.webkitURL).createObjectURL(\n",
"          evt.data);\n",
"          fig.updated_canvas_event();\n",
"          fig.waiting = false;\n",
"          return;\n","}\n",
"          else if (typeof evt.data === 'string' && evt.data.slice(0, 21) ==
\"data:image/png;base64\") {\n",
"          fig.imageObj.src = evt.data;\n",
"          fig.updated_canvas_event();\n",
"          fig.waiting = false;\n",
"          return;\n","}\n","\n",
"          var msg = JSON.parse(evt.data);\n",
"          var msg_type = msg['type'];\n","\n",
" try{\n",
"          var callback = fig[\"handle_\" + msg_type];\n",
"          } catch (e) {\n",
"          console.log(\"No handler for the '\" + msg_type + \"' message type: \"msg);\n",
"          return;\n","}\n","\n",
"          if (callback) {\n",
"          try {\n",
"                    callback(fig, msg);\n",
"          } catch (e) {\n",
"                    console.log(\"Exception inside the 'handler_\" + msg_type + \"' callback:\",
e, e.stack, msg);\n","}\n","}\n","};\n","}\n","\n",
```

```
"mpl.findpos = function(e) {\n",
"          var targ;\n",
"    if (!e)\n",
"    e = window.event;\n", "
     if (e.target)\n",
"          targ = e.target;\n",
"    else if (e.srcElement)\n",
"          targ = e.srcElement;\n",
"   if (targ.nodeType == 3) // defeat Safari bug\n",
" targ = targ.parentNode;\n","\n",
"          var x = e.pageX - $(targ).offset().left;\n",
"     var y = e.pageY - $(targ).offset().top;\n","\n",
" return {\"x\": x, \"y\": y};\n","};\n","\n","/*\n",
" * return a copy of an object with only non-object keys\n",
" * we need this to avoid circular references\n",
" * http://stackoverflow.com/a/24161582/3208463\n",
" */\n",
"function simpleKeys (original) {\n",
" return Object.keys(original).reduce(function (obj, key) {\n",
"  if (typeof original[key] !== 'object')\n",
"          obj[key] = original[key]\n",
"    return obj;\n","}, {});\n","}\n","\n",
"mpl.figure.prototype.mouse_event = function(event, name) {\n",
"    var canvas_pos = mpl.findpos(event)\n","\n",
"    if (name === 'button_press')\n","{\n",
```

```
"          this.canvas.focus();\n",
"    this.canvas_div.focus();\n","}\n","\n", "
     var x = canvas_pos.x * mpl.ratio;\n",
"    var y = canvas_pos.y * mpl.ratio;\n","\n",
"    this.send_message(name, {x: x, y: y, button: event.button,\n",
"    step: event.step,\n",
"                          guiEvent: simpleKeys(event)});\n","\n",
   "mpl.figure.prototype._key_event_extra = function(event, name) {\n","}\n","\n",
"mpl.figure.prototype.key_event = function(event, name) {\n","\n",
"    // Prevent repeat events\n",
"    if (name == 'key_press')\n","{\n",
"          if (event.which === this._key)\n",
"          return;\n",
"          else\n",
"    this._key = event.which;\n","}\n", "
     if (name == 'key_release')\n",
"          this._key = null;\n","\n",
"    var value = '';\n",
"    if (event.ctrlKey && event.which != 17)\n",
"    value += \"ctrl+\";\n",
"    if (event.altKey && event.which != 18)\n",
"    value += \"alt+\";\n",
"    if (event.shiftKey && event.which != 16)\n",
"    value += \"shift+\";\n","\n",
"    value += 'k';\n",
```

```
"    value += event.which.toString();\n","\n",
"    this._key_event_extra(event, name);\n","\n",
"    this.send_message(name, {key: value,\n",
"    guiEvent: simpleKeys(event)});\n", "
    return false;\n","}\n","\n",
"mpl.figure.prototype.toolbar_button_onclick = function(name) {\n",
"    if (name == 'download') {\n",
"    this.handle_save(this, null);\n", "
    } else {\n",
"        this.send_message(\"toolbar_button\", {name: name});\n","}\n","};\n","\n",
"mpl.figure.prototype.toolbar_button_onmouseover = function(tooltip) {\n",
"    this.message.textContent = tooltip;\n","};\n","\n",
"mpl.extensions = [\"eps\", \"jpeg\", \"pdf\", \"png\", \"ps\", \"raw\", \"svg\",
\"tif\"];\n","\n",
"mpl.default_extension = \"png\";var comm_websocket_adapter = function(comm){\n",
"        var ws = {};\n","\n",
"    ws.close = function() {\n",
"    comm.close()\n","};\n", "
    ws.send = function(m) {\n",
"        comm.send(m);\n","};\n",
"    comm.on_msg(function(msg) {\n",
"    ws.onmessage(msg['content']['data'])\n","});\n", "
    return ws;\n","}\n","\n",
"mpl.mpl_figure_comm = function(comm, msg) {\n",       "\n",
"    var id = msg.content.data.id;\n",
```

```
"    var element = $(\"#\" + id);\n",
"    var ws_proxy = comm_websocket_adapter(comm)\n","\n",
"    function ondownload(figure, format) {\n",
"    window.open(figure.imageObj.src);\n","}\n","\n", "
    var fig = new mpl.figure(id, ws_proxy,\n",
"        ondownload,\n",
"        element.get(0));\n","\n",
"  fig.parent_element = element.get(0);\n",
"    fig.cell_info = mpl.find_output_cell(\"<div id='\" + id + \"'></div>\");\n",
"    if (!fig.cell_info) {\n",
"        console.error(\"Failed to find cell for figure\", id, fig);\n",
"        return;\n","}\n","\n",
"    var output_index = fig.cell_info[2]\n",
"    var cell = fig.cell_info[0];\n","\n","};\n","\n",
"mpl.figure.prototype.handle_close = function(fig, msg) {\n",
"    var width = fig.canvas.width/mpl.ratio\n",
"    fig.root.unbind('remove')\n", "\n",
"mpl.figure.prototype.close_ws = function(fig, msg){\n",
"    fig.send_message('closing', msg);\n",
"    // fig.ws.close()\n","}\n","\n",
 "mpl.figure.prototype.push_to_output = function(remove_interactive) {\n",
"    var width = this.canvas.width/mpl.ratio\n",
"    var dataURL = this.canvas.toDataURL();\n",
"    this.cell_info[1]['text/html'] = '<img src=\"' + dataURL + '\" width=\"' + width +
'\">';\n","}\n","\n",
```

```
"mpl.figure.prototype.updated_canvas_event = function() {\n",
"    IPython.notebook.set_dirty(true);\n",
"    this.send_message(\"ack\", {});\n",
"    var fig = this;\n",
"    setTimeout(function () { fig.push_to_output() }, 1000);\n","}\n","\n",
"mpl.figure.prototype._init_toolbar = function() {\n",
"    var fig = this;\n","\n",
"    var nav_element = $('<div/>');\n",
"    nav_element.attr('style', 'width: 100%');\n",
"     this.root.append(nav_element);\n","\n",
" function toolbar_event(event) {\n",
"    return fig.toolbar_button_onclick(event['data']);\n","}\n", "
    function toolbar_mouse_event(event) {\n",
"    return fig.toolbar_button_onmouseover(event['data']);\n","}\n","\n", "
    for(var toolbar_ind in mpl.toolbar_items){\n",
"        var name = mpl.toolbar_items[toolbar_ind][0];\n",
"        var tooltip = mpl.toolbar_items[toolbar_ind][1];\n",
"        var image = mpl.toolbar_items[toolbar_ind][2];\n",
"       var method_name = mpl.toolbar_items[toolbar_ind][3];\n","\n",
"       if (!name) { continue; };\n","\n",
"        var button = $('<button class=\"btn btn-default\" href=\"#\" title=\"' + name +
'\"><i class=\"fa ' + image + ' fa-lg\"></i></button>');\n",
"        button.click(method_name, toolbar_event);\n",
"        button.mouseover(tooltip, toolbar_mouse_event);\n",
"        nav_element.append(button);\n","}\n","\n",
```

```
"    var status_bar = $('<span class=\"mpl-message\" style=\"text-align:right; float:
right;\"/>');\n",
"    nav_element.append(status_bar);\n",
"    this.message = status_bar[0];\n","\n",
"mpl.figure.prototype._root_extra_style = function(el){\n",
"    var fig = this\n",
"    el.on(\"remove\", function(){\n",
"\tfig.close_ws(fig, {});\n","});\n","}\n","\n",
"mpl.figure.prototype._canvas_extra_style = function(el){\n",
"    el.attr('tabindex', 0)\n","\n",
"    if (IPython.notebook.keyboard_manager) {\n",
"   IPython.notebook.keyboard_manager.register_events(el);\n","}\n",   "
        else {\n",
"        IPython.keyboard_manager.register_events(el);\n","}\n","\n","}\n","\n",
"mpl.figure.prototype._key_event_extra = function(event, name) {\n",
"    var manager = IPython.notebook.keyboard_manager;\n",
"  if (!manager)\n",
"   manager = IPython.keyboard_manager;\n","\n",  "
        if (event.shiftKey && event.which == 13) {\n",
"          this.canvas_div.blur();\n",
"          event.shiftKey = false;\n",
" event.which =74;\n",
"         event.keyCode = 74;\n",
"          manager.command_mode();\n",
"          manager.handle_keydown(event);\n","}\n","}\n","\n",
```

```
        "mpl.figure.prototype.handle_save = function(fig, msg) {\n",
        "    fig.ondownload(fig, null);\n","}\n","\n","\n",
        "mpl.find_output_cell = function(html_output) {\n",
        "    var cells = IPython.notebook.get_cells();\n",
        "    var ncells = cells.length;\n",
        "    for (var i=0; i<ncells; i++) {\n",
        "    var cell = cells[i];\n",
        "          if (cell.cell_type === 'code'){\n",
        "           for (var j=0; j<cell.output_area.outputs.length; j++) {\n",
        "           var data = cell.output_area.outputs[j];\n",
        "                    if (data.data) {\n",
        "                    data = data.data;\n","}\n",
        "                    if (data['text/html'] == html_output) {\n",
        "                    return [cell, data, j];\n","}\n","}\n","}\n","}\n","}\n","\n",
        "if (IPython.notebook.kernel != null) {\n",
        "    IPython.notebook.kernel.comm_manager.register_target('matplotlib',
mpl.mpl_figure_comm);\n","}\n"],
      "text/plain": ["<IPython.core.display.Javascript object>"]},
     "metadata": {},
     "output_type": "display_data"},],
      "text/plain": ["<IPython.core.display.HTML object>"]},
     "metadata": {},
     "output_type": "display_data"}],
   "source": [
    "from sklearn.ensemble import RandomForestClassifier\n",
```

```
"%matplotlib notebook\n",
"clf = RandomForestClassifier(random_state=0, n_jobs=-1).fit(X_train_scaled, y_train)\n",
"importances = clf.feature_importances_\n",
"indices = np.argsort(importances)[::-1]\n",
"names = [X.columns.values[i] for i in indices]\n",
"plt.figure()\n",
"plt.title(\"Feature Importance\")\n",
"plt.bar(range(X.shape[1]), importances[indices])\n",
"plt.xticks(range(X.shape[1]), names, rotation=90)\n",
"plt.show()"]},{
"cell_type": "markdown",
"metadata": {},
"source": [
"# Feature importance using Decision Tree"]},{
  "  function canvas_keyboard_event(event) {\n",
    "        return fig.key_event(event, event['data']);\n","}\n","\n",
    "  canvas_div.keydown('key_press', canvas_keyboard_event);\n",
    "  canvas_div.keyup('key_release',  canvas_keyboard_event);\n",
    "  this.canvas_div = canvas_div\n",
    "  this._canvas_extra_style(canvas_div)\n",
    "  this.root.append(canvas_div);\n","\n",
    "  var canvas = $('<canvas/>');\n",
    "  canvas.addClass('mpl-canvas');\n",
    "  canvas.attr('style', \"left: 0; top: 0; z-index: 0; outline: 0\")\n","\n",
    "  this.canvas = canvas[0];\n",
```

```
"    this.context = canvas[0].getContext(\"2d\");\n","\n","\n",
"    mpl.ratio = (window.devicePixelRatio || 1) / backingStore;\n","\n",
"    var rubberband = $('<canvas/>');\n",
"    rubberband.attr('style', \"position: absolute; left: 0; top: 0; z-index: 1;\")\n","\n",
"    var pass_mouse_events = true;\n","\n",
"    canvas_div.resizable({\n",
"        start: function(event, ui) {\n",
"        pass_mouse_events = false;\n","    },\n",
"        resize: function(event, ui) {\n",
"        fig.request_resize(ui.size.width, ui.size.height);\n","},\n",
"        stop: function(event, ui) {\n",
"         pass_mouse_events = true;\n",
"    fig.request_resize(ui.size.width, ui.size.height);\n","    },\n","});\n","\n", "
    function mouse_event_fn(event) {\n",
"        if (pass_mouse_events)\n",
"    return fig.mouse_event(event, event['data']);\n","}\n","\n", "
    rubberband.mousedown('button_press', mouse_event_fn);\n",
"    rubberband.mouseup('button_release', mouse_event_fn);\n",
"    rubberband.mousemove('motion_notify', mouse_event_fn);\n","\n",
"    rubberband.mouseenter('figure_enter', mouse_event_fn);\n",
"    rubberband.mouseleave('figure_leave', mouse_event_fn);\n","\n",
"    canvas_div.on(\"wheel\", function (event) {\n",
"        event = event.originalEvent;\n",
"        event['data'] = 'scroll'\n",
"        if (event.deltaY < 0) {\n",
```

```
"            event.step = 1;\n",
"            } else {\n",
"            event.step = -1;\n","}\n",
"    mouse_event_fn(event);\n","});\n","\n", "
    canvas_div.append(canvas);\n",
"    canvas_div.append(rubberband);\n","\n",
"    this.rubberband = rubberband;\n",
"    this.rubberband_canvas = rubberband[0];\n",
"    this.rubberband_context = rubberband[0].getContext(\"2d\");\n",
"    this.rubberband_context.strokeStyle = \"#000000\";\n","\n",
"    this._resize_canvas = function(width, height) {\n",
"    canvas_div.css('width', width)\n",
"            canvas_div.css('height', height)\n","\n",
"            canvas.attr('width', width * mpl.ratio);\n",
"            canvas.attr('height', height * mpl.ratio);\n",
"            canvas.attr('style', 'width: ' + width + 'px; height: ' + height + 'px;');\n","\n",
"            rubberband.attr('width', width);\n",
"    rubberband.attr('height', height);\n","}\n","\n", "
    this._resize_canvas(600, 600);\n","\n",
"    $(this.rubberband_canvas).bind(\"contextmenu\",function(e){\n",
"    return false;\n","});\n","\n",
"    var status_bar = $('<span class=\"mpl-message\"/>');\n",
"    nav_element.append(status_bar);\n",
"    this.message = status_bar[0];\n","}\n","\n",
"mpl.figure.prototype.request_resize = function(x_pixels, y_pixels) {\n",
```

```
"    this.send_message('resize', {'width': x_pixels, 'height': y_pixels});\n","}\n","\n",

"mpl.figure.prototype.send_message = function(type, properties) {\n",

"    properties['type'] = type;\n",

"    properties['figure_id'] = this.id;\n",

"    this.ws.send(JSON.stringify(properties));\n","}\n","\n",

"mpl.figure.prototype.send_draw_message = function() {\n",

"    if (!this.waiting) {\n",

"        this.waiting = true;\n",

"        this.ws.send(JSON.stringify({type: \"draw\", figure_id:
this.id}));\n","}\n","}\n","\n","\n",

"mpl.figure.prototype.handle_save = function(fig, msg) {\n",

"    var format_dropdown = fig.format_dropdown;\n",

"    var format = format_dropdown.options[format_dropdown.selectedIndex].value;\n",

"    fig.ondownload(fig,  format);\n","}\n","\n","\n",

"mpl.figure.prototype.handle_resize = function(fig, msg) {\n",

"    var size = msg['size'];\n",

"    if (size[0] != fig.canvas.width || size[1] != fig.canvas.height) {\n",

"    fig._resize_canvas(size[0], size[1]);\n",

"        fig.send_message(\"refresh\", {});\n","};\n","}\n","\n",

"    fig.rubberband_context.clearRect(\n",

"        0, 0, fig.canvas.width / mpl.ratio, fig.canvas.height / mpl.ratio);\n","\n",

"    fig.rubberband_context.strokeRect(min_x, min_y, width, height);\n","}\n","\n",

"mpl.figure.prototype.handle_figure_label = function(fig, msg) {\n",

"    fig.header.textContent = msg['label'];\n","}\n", "\n",

"mpl.figure.prototype.handle_cursor = function(fig, msg) {\n",
```

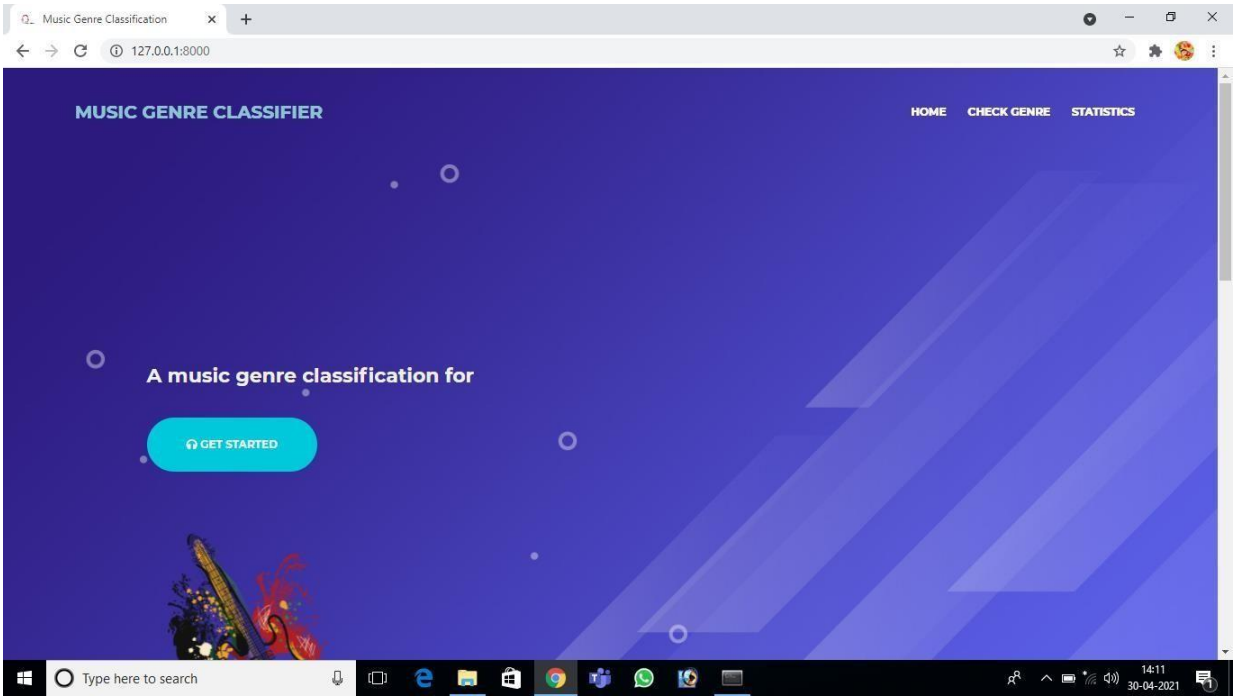
```
"    var cursor = msg['cursor'];\n",
"function simpleKeys (original) {\n",
" return Object.keys(original).reduce(function (obj, key) {\n",
"  if (typeof original[key] !== 'object')\n",
"          obj[key] = original[key]\n",
"    return obj;\n","}, {});\n","}\n","\n",
"mpl.figure.prototype.mouse_event = function(event, name) {\n",
"    var canvas_pos = mpl.findpos(event)\n","\n",
"    if (name === 'button_press')\n","{\n",
"          this.canvas.focus();\n",
"    this.canvas_div.focus();\n","}\n","\n", "
     var x = canvas_pos.x * mpl.ratio;\n",
"    var y = canvas_pos.y * mpl.ratio;\n","\n",
"    this.send_message(name, {x: x, y: y, button: event.button,\n",
"    step: event.step,\n",
"                              guiEvent: simpleKeys(event)});\n","\n",
"    /* This prevents the web browser from automatically changing to\n",
"          * the text insertion cursor when the button is pressed. We want\n",
"          * to control all of the cursor setting manually through the\n",
"    * 'cursor' event from matplotlib */\n", "
     event.preventDefault();\n",
"    return false;\n","}\n","\n",
"mpl.figure.prototype._key_event_extra = function(event, name) {\n","}\n","\n",
"mpl.figure.prototype.key_event = function(event, name) {\n","\n",
"    if (name == 'key_press')\n","{\n",
```

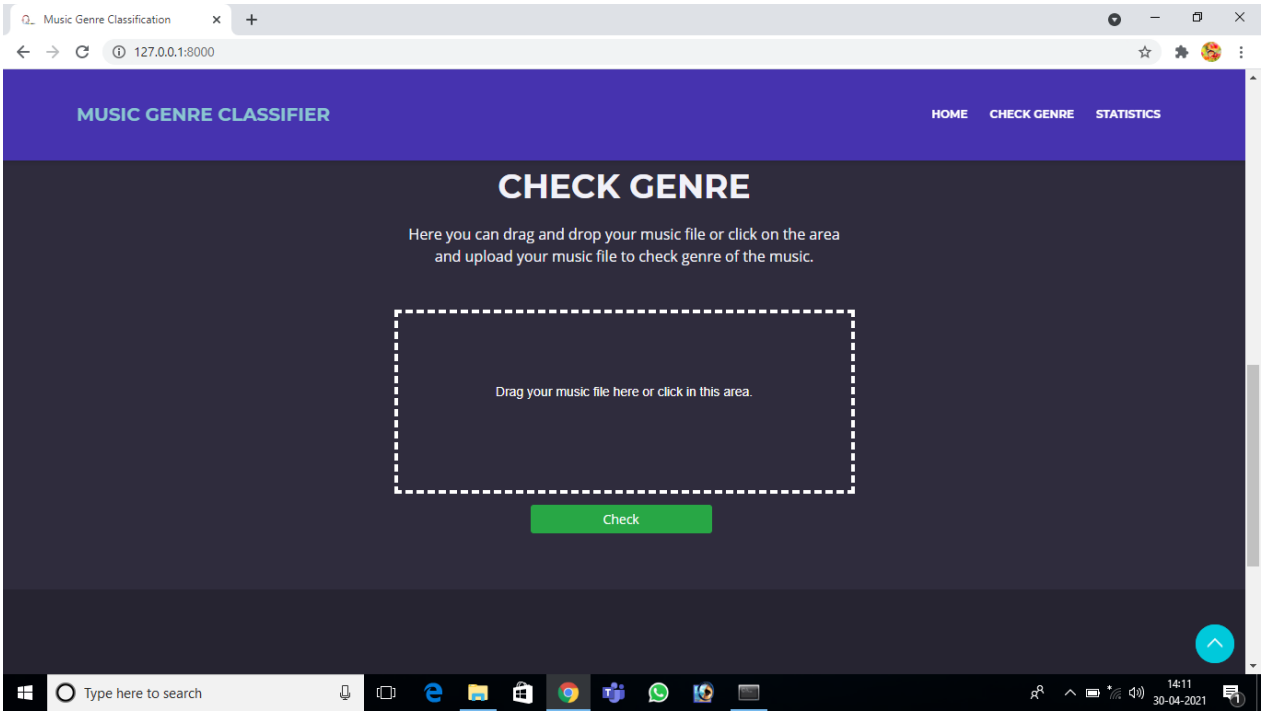
```
"          if (event.which === this._key)\n",
"          return;\n",
"          else\n",
"    this._key = event.which;\n","}\n", "
     if (name == 'key_release')\n",
"          this._key = null;\n","\n",
"    var value = ';\n",
"    if (event.ctrlKey && event.which != 17)\n",
"    value += \"ctrl+\";\n",
"    if (event.altKey && event.which != 18)\n",
"    value += \"alt+\";\n",
"    if (event.shiftKey && event.which != 16)\n",
"    value += \"shift+\";\n","\n",
"    value += 'k';\n",
"    value += event.which.toString();\n","\n",
"    this._key_event_extra(event, name);\n","\n",
"    this.send_message(name, {key: value,\n",
"    guiEvent: simpleKeys(event)});\n", "return
   false;\n","}\n","\n",
   "mpl.figure.prototype.toolbar_button_onclick = function(name) {\n",
"    if (name == 'download') {\n",
"    this.handle_save(this, null);\n", "
     } else {\n",
"        this.send_message(\"toolbar_button\", {name: name});\n","}\n","};\n","\n",
"mpl.figure.prototype.toolbar_button_onmouseover = function(tooltip) {\n",
```

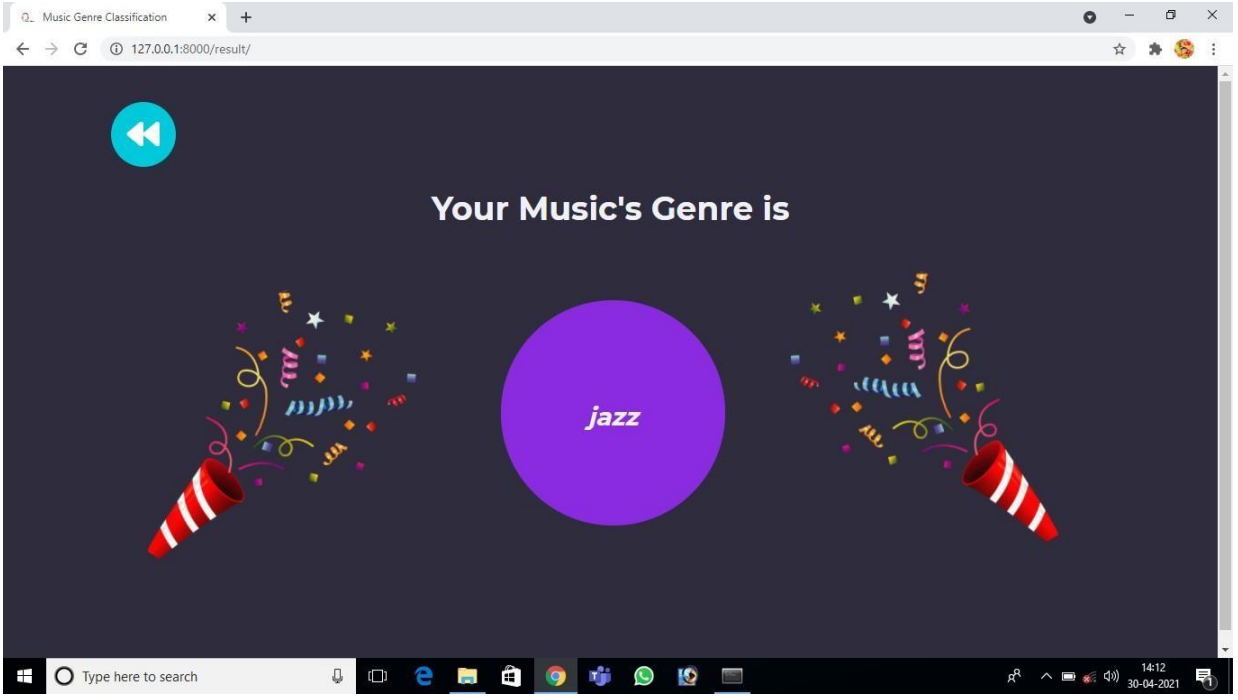
```
"    this.message.textContent = tooltip;\n","};\n",

"    ws.close = function() {\n",

"        comm.close()\n","};\n",

"    ws.send = function(m) {\n",

"if (IPython.notebook.kernel != null) {\n",

"    IPython.notebook.kernel.comm_manager.register_target('matplotlib',
mpl.mpl_figure_comm);\n","}\n"],

 "text/plain": [ "<IPython.core.display.Javascript object>"]},

 "metadata": {},

 "output_type": "display_data"},

 "text/plain": [

    "<IPython.core.display.HTML object>"]},

 "metadata": {},

 "output_type": "display_data"}],

"source": [

"from sklearn.tree import DecisionTreeClassifier\n",

"clf = DecisionTreeClassifier(random_state=0).fit(X_train_scaled, y_train)\n",

"importances = clf.feature_importances_\n",

"indices = np.argsort(importances)[::-1]\n",

"names = [X.columns.values[i] for i in indices]\n",

"plt.figure()\n",

"plt.title(\"Feature Importance\")\n",

"plt.bar(range(X.shape[1]), importances[indices])\n",

"plt.xticks(range(X.shape[1]), names, rotation=90)\n",

"plt.show()"]}
```

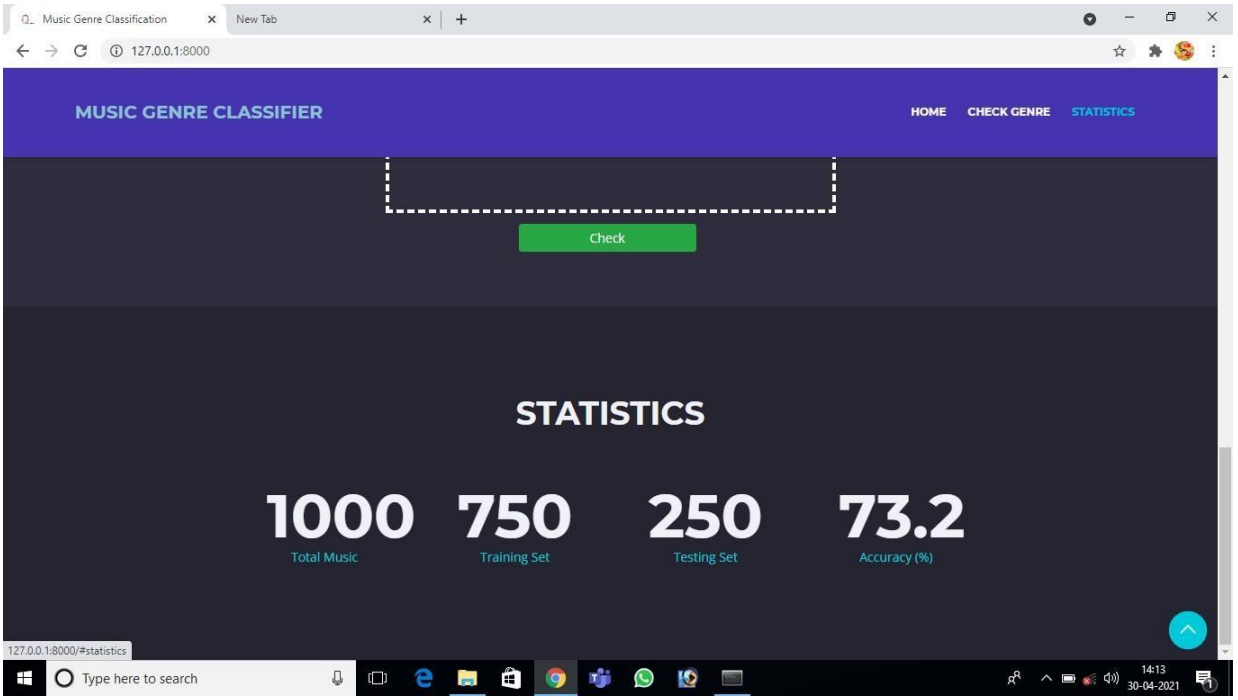
APPENDIX 2

SNAPSHOTS:




Music Genre Classification
127.0.0.1:8000
MUSIC GENRE CLASSIFIER
HOME
CHECK GENRE
STATISTICS
CHECK GENRE
Here you can drag and drop your music file or click on the area
and upload your music file to check genre of the music.
Drag your music file here or click in this area.
Check
Type here to search
14:11
30-04-2021


Music Genre Classification
127.0.0.1:8000/result/
Your Music's Genre is
jazz
Type here to search
14:12
30-04-2021


Music Genre Classification
New Tab
127.0.0.1:8000
MUSIC GENRE CLASSIFIER
HOME
CHECK GENRE
STATISTICS
Check
STATISTICS
1000
Total Music
750
Training Set
250
Testing Set
73.2
Accuracy (%)
127.0.0.1:8000/#statistics
Type here to search
14:13
30-04-2021