

**Term Project**  
**Python – FSDM Term 2**

**Multi-Class Genre Music  
Classification using Machine  
Learning**

**Akash Manhas**  
**Rudhar Sharma**

## Data Sets –

We used Audio Database - [GTZAN Genre Collection](#). All the audio file is .wav , 16-bit, Mono Channel, 44100 Hz format. We used pyAudioAnalysis Python Library to extract the features. Here are the 34 audio features we extracted using this.

Feature ID	Feature Name	Description
1	Zero Crossing Rate	The rate of sign-changes of the signal during the duration of a particular frame.
2	Energy	The sum of squares of the signal values, normalized by the respective frame length.
3	Entropy of Energy	The entropy of sub-frames' normalized energies. It can be interpreted as a measure of abrupt changes.
4	Spectral Centroid	The center of gravity of the spectrum.
5	Spectral Spread	The second central moment of the spectrum.
6	Spectral Entropy	Entropy of the normalized spectral energies for a set of sub-frames.
7	Spectral Flux	The squared difference between the normalized magnitudes of the spectra of the two successive frames.
8	Spectral Rolloff	The frequency below which 90% of the magnitude distribution of the spectrum is concentrated.
9-21	MFCCs	Mel Frequency Cepstral Coefficients form a cepstral representation where the frequency bands are not linear but distributed according to the mel-scale.
22-33	Chroma Vector	A 12-element representation of the spectral energy where the bins represent the 12 equal-tempered pitch classes of western-type music (semitone spacing).
34	Chroma Deviation	The standard deviation of the 12 chroma coefficients.

These are the features we can extract.

## featureExtraction.py

```
import pandas as pd
import numpy as np
from pyAudioAnalysis import ShortTermFeatures
from pyAudioAnalysis import audioBasicIO
import pdb

for i in range(0, 100):
    cat = "blues"
    file_path = "/home/sky/Downloads/extracted
files/archive/Data/genres_original/" + cat + "/" + cat
    if (i<10):
        file_name = ".0000" + str(i) + ".wav"
    else:
        file_name = ".000" + str(i) + ".wav"

    file_path = file_path + file_name
    print("Extracting - " + file_path)
    features = pd.DataFrame()
    [Fs, x] = audioBasicIO.read_audio_file(file_path)
    F, f_names = ShortTermFeatures.feature_extraction(x, Fs, 1.0*Fs, 1.0*Fs,
False)
    C = np.array([])
    for index in range(0,34):
        A = F[index,:]
        if A.size <= 30:
            A = np.append(A, np.repeat(np.nan, 30-A.size))
        else:
            A = A[...,:-(A.size-30)]
        C = np.concatenate((C,A))

    features = features.append(pd.DataFrame([list(C)]))
    num+=1
features= features.fillna(features.mean())
features.to_csv("features.csv", mode='a', index=False)
```

## **Data Pre-processing –**

For any NAN value encountered during the features extraction, we replace that value with the average value.

## Training and Testing –

Now we have the labelled dataset and on that we are planning to do supervised learning using the following classification models.

### Logarithmic Regression –

This is a type of regression analysis used to model relation between a dependent variable and one or more independent variables. We can assume that, logarithmic regression model assumes changes in dependent variable are proportional to the logarithm of the independent variable. This is helpful in times where the relationship between variables is nonlinear and transformation could support it to become a linearize the relation. It is also being used in different fields such as finance, biology, economics, and social science. In machine learning it is often used for binary classification problems, where the objective is to predict whether the data belong to a particular class or not.

#### log\_reg\_train.py

```
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.model_selection import train_test_split
import pandas as pd
from sklearn.linear_model import LogisticRegression

all = pd.read_csv("features_1000ms.csv")
print (all.info())
rel_cols = list(all.columns)[1:-3]
features = all[rel_cols]
features = features.fillna(features.mean())
classes = all["1020"]
x_train, x_test, y_train, y_test = train_test_split(features, classes,
test_size = 0.10, random_state=42)

LogReg = LogisticRegression()
LogReg.fit(x_train, y_train)
y_pred = LogReg.predict(x_test)
print("\n\nClassification Report")
print(classification_report(y_test, y_pred))
print("\n\nConfusion Matrix")
print(confusion_matrix(y_test, y_pred))
print("\n\nAccuracy Score : ", end="")
print(accuracy_score(y_pred, y_test))
```

```
sky@sky: ~/Downloads/Lambton
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
n_iter_i = _check_optimize_result(

Classification Report
precision    recall  f1-score   support

   0       0.50      0.50      0.50        10
   1       0.50      0.71      0.59         7
   2       0.67      0.29      0.40        14
   3       0.55      0.60      0.57        10
   4       0.67      0.86      0.75         7
   5       0.33      0.62      0.43         8
   6       0.62      0.38      0.48        13

 accuracy          0.52        69
 macro avg          0.55        69
weighted avg          0.56        69

Confusion Matrix
[[5 1 0 1 0 2 1]
 [1 5 0 0 0 1 0]
 [2 3 4 1 1 3 0]
 [0 0 0 6 0 3 1]
 [0 0 0 0 6 0 1]
 [2 1 0 0 0 5 0]
 [0 0 2 3 2 1 5]]

Accuracy Score : 0.5217391304347826
sky@sky: ~/Downloads/Lambton$

sky@sky: ~/Downloads/Lambton$ python3 log_reg_train.py
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 686 entries, 0 to 685
Columns: 1024 entries, Unnamed: 0 to 1022
dtypes: float64(1020), int64(2), object(2)
memory usage: 5.4+ MB
None
/home/sky/.local/lib/python3.10/site-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
n_iter_i = _check_optimize_result(

Classification Report
precision    recall  f1-score   support

   0       0.50      0.50      0.50        10
   1       0.50      0.71      0.59         7
   2       0.67      0.29      0.40        14
   3       0.55      0.60      0.57        10
   4       0.67      0.86      0.75         7
   5       0.33      0.62      0.43         8
   6       0.62      0.38      0.48        13

 accuracy          0.52        69
 macro avg          0.55        69
weighted avg          0.56        69

Confusion Matrix
[[5 1 0 1 0 2 1]
 [1 5 0 0 0 1 0]
 [2 3 4 1 1 3 0]
 [0 0 0 6 0 3 1]
 [0 0 0 0 6 0 1]
 [2 1 0 0 0 5 0]
 [0 0 2 3 2 1 5]]

Accuracy Score : 0.5217391304347826
sky@sky: ~/Downloads/Lambton$
```

The accuracy achieved is 52.17%.

## Random Forest –

This is a widely used machine learning algorithm used for both classification and regression problems. It may combine multiple decision trees to improve the accuracy and stability of the model. The large number of decision trees are trained on random subsets of dataset and the final output is the average or majority vote of individual tree.

Some advantages of this algorithm are as follows –

- Robust to different types data noise.
- Handles high dimensional datasets.
- Can capture complex non-linear relationship between the features and target variables.

### random forest train.py

```
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.model_selection import train_test_split
import pandas as pd
from sklearn.ensemble import RandomForestClassifier

all = pd.read_csv("features_1000ms.csv")
print (all.info())
rel_cols = list(all.columns)[1:-3]
features = all[rel_cols]
features= features.fillna(features.mean())
classes = all["1020"]
x_train, x_test, y_train, y_test = train_test_split(features,classes,
test_size = 0.10,random_state=42)

model_rf = RandomForestClassifier(n_estimators=250,random_state=42)
model_rf.fit(x_train,y_train)
y_predict = model_rf.predict(x_test)
print("\n\nClassification Report")
print(classification_report(y_test,y_predict))
print("\n\nConfusion Matrix")
print(confusion_matrix(y_test, y_predict))
print("\n\nAccuracy Score : ", end="")
print(accuracy_score(y_predict, y_test))
```

```
sky@sky: ~/Downloads/Lambton$ python3 random_forest_train.py
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 686 entries, 0 to 685
Columns: 1024 entries, Unnamed: 0 to 1022
dtypes: float64(1020), int64(2), object(2)
memory usage: 5.4+ MB
None

Classification Report
      precision    recall  f1-score   support

0         0.45      0.50      0.48        10
1         0.75      0.86      0.80         7
2         0.91      0.71      0.80        14
3         0.78      0.70      0.74        10
4         0.86      0.86      0.86         7
5         0.50      0.75      0.60         8
6         0.91      0.77      0.83        13

 accuracy          0.72        69
 macro avg         0.74        69
weighted avg         0.76        69

Confusion Matrix
[[ 5  1  0  0  0  4  0]
 [ 1  6  0  0  0  0  0]
 [ 2  1 10  1  0  0  0]
 [ 1  0  0  7  0  2  0]
 [ 0  0  0  0  6  0  1]
 [ 1  0  0  1  0  6  0]
 [ 1  0  1  0  1  0 10]]

Accuracy Score : 0.7246376811594203
sky@sky: ~/Downloads/Lambton$
```

The accuracy achieved is 52.17%.



## XGBoost –

This is an open-source machine learning library used for supervised learning problems, like classification and regression. In this model the technique is to combine multiple weak learners to create strong learners.

Based on gradient boosting algorithm, it will add decision trees to the model to minimize a cost function. The final model get from this is a sum of all the trees.

### xgboost\_train.py

```
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.model_selection import train_test_split
import pandas as pd
from xgboost.sklearn import XGBClassifier

all = pd.read_csv("features_1000ms.csv")
print (all.info())
rel_cols = list(all.columns)[1:-3]
features = all[rel_cols]
features= features.fillna(features.mean())
classes = all["1020"]
x_train, x_test, y_train, y_test = train_test_split(features,classes,
test_size = 0.10,random_state=42)

model = XGBClassifier(max_depth=3,n_estimators=400,learning_rate=0.02)
model.fit(x_train,y_train)
y_predict=model.predict(x_test)
print("\nClassification Report")
print(classification_report(y_test,y_predict))
print("\nConfusion Matrix")
print(confusion_matrix(y_test, y_predict))
print("\nAccuracy Score : ", end="")
print(accuracy_score(y_predict, y_test))
```

```
sky@sky: ~/Downloads/Lambton
sky@sky: ~/Downloads/Lambton$ python3 xgboost_train.py
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 686 entries, 0 to 685
Columns: 1024 entries, Unnamed: 0 to 1022
dtypes: float64(1020), int64(2), object(2)
memory usage: 5.4+ MB
None

Classification Report
      precision    recall  f1-score   support

     0       0.46       0.60       0.52        10
     1       0.75       0.86       0.80         7
     2       0.83       0.71       0.77        14
     3       0.88       0.70       0.78        10
     4       0.88       1.00       0.93         7
     5       0.60       0.75       0.67         8
     6       0.90       0.69       0.78        13

 accuracy          0.74         69
 macro avg          0.76         69
 weighted avg       0.77         69

Confusion Matrix
[[ 6  1  0  1  0  2  0]
 [ 1  6  0  0  0  0  0]
 [ 2  1 10  0  0  0  1]
 [ 1  0  0  7  0  2  0]
 [ 0  0  0  0  7  0  0]
 [ 2  0  0  0  0  6  0]
 [ 1  0  2  0  1  0  9]]

Accuracy Score : 0.7391304347826086
sky@sky: ~/Downloads/Lambton$
```

The accuracy achieved is 73.91%.

## Web Application –

Since XGBoost trained model gave us the highest accuracy we are using that model to build a web app where users can upload a music file and our trained model will classify the file in a specific genre.

Now we created a web app on this XGBoost Trained Model to Classify any audio file which is of specific type (Mentioned above) of any length

### main.py

```
import os
from flask import Flask, request, redirect, url_for
from werkzeug.utils import secure_filename
#from sklearn.externals import joblib
import joblib
from sklearn.metrics import accuracy_score
import pandas as pd
import numpy as np

features = pd.DataFrame()

UPLOAD_FOLDER = 'music'
ALLOWED_EXTENSIONS = set(['wav'])

app = Flask(__name__)
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

def allowed_file(filename):
    return '.' in filename and filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS

@app.route('/', methods=['GET', 'POST'])

def upload_file():
    from pyAudioAnalysis import ShortTermFeatures
    from pyAudioAnalysis import audioBasicIO
    features = pd.DataFrame()
    if request.method == 'POST':
        # check if the post request has the file part
        if 'file' not in request.files:
            flash('No file part')
```

```

        return redirect(request.url)
    file = request.files['file']
    if file.filename == '':
        flash('No selected file')
        return redirect(request.url)
    if file and allowed_file(file.filename):
        filename = secure_filename(file.filename)
        file.save(os.path.join(app.config['UPLOAD_FOLDER'], filename))

    [Fs, x] =
audioBasicIO.read_audio_file(os.path.join(app.config['UPLOAD_FOLDER'],
filename))
    F, f_names = ShortTermFeatures.feature_extraction(x, Fs, 1.0*Fs,
1.0*Fs, False)
    C = np.array([])
    for index in range(0,34):
        A = F[index,:]
        if A.size <= 30:
            A = np.append(A, np.repeat(np.nan, 30-A.size))
        else:
            A = A[:,:(A.size-30)]
        C = np.concatenate((C,A))

    features = features.append(pd.DataFrame([list(C)]))
    num+=1
    features= features.fillna(features.mean())
    model = joblib.load("best_model_lambton.pkl")
    y_predict=model.predict(features.values)
    he = str(y_predict[0])
    if he == "0":
        res= "Disco"
    elif he == "1":
        res= "Metal"
    elif he=="2":
        res="Blues"
    elif he=="3":
        res= "Reggae"
    elif he=="4":
        res= "Classical"
    elif he=="5":
        res= "Pop"
    else:
        res= "Jazz"

    return ''

<!-- doctype html -->
<head>

```

```
<title>Multi-Class Music Genre Classification Using Machine
Learning</title>
</head>
<style>
body{
    width:1200px;
    margin:0 auto;
    font-family:arial;
    padding:10px;
    color:#193646;
}
h1{
    text-align:center;
    color:#193646;
    margin-top:50px;

}
h2{
    text-align:center;
    color:#193646;

}
h3{
    text-align:center;
    color:#193646;
}
form{
    width:560px;
    margin:0 auto;
}
#large{
    width:400px;
}
.top{
    height:100px;
    width:825px;
    margin:0 auto;
    border-bottom:5px solid #193646;
    padding-bottom:5px;
}
.image{
    width:100px;
    height:100px;

    float:left;
}
.image img{
    width:100%;
```

```

        height:100%;

    }
    .head{
        padding-top:30px;
        font-size:36px;
        float:left;
        width:720px;
color:#193646;
        text-align:center;
font-weight:bold;
    }
    .box{
        width:1000px;
padding:10px;
margin:40px auto;
padding-bottom:40px;
    }
    .box2{
width:470px;
padding:5px;
border-left:5px solid #193646;
margin-left:90px;
float:left;
    }
    .box3{
width:250px;
padding:5px;
border-left:5px solid #193646;
float:left;
margin-left:250px;
    }

</style>
<body>
<div class=top>
    <div class=image></div>
    <div class=head>Lambton College in Mississauga</div>
</div>
<h2> Full Stack Software Development </h2>
<div class=box>
    <h1>Multi-Class Music Genre Classification Using Machine Learning</h1>
<h3> Select File To Classify </h3>
    <form method=post enctype=multipart/form-data>
        <input type=file name=file id=large>
        <input type=submit value=Classify>
    </form>
<h3>Current File is Classified As - '''+res+'''+</h3>
</div>

```

```

<div class=box2>
<table>
<tr><td> </td><td></td></tr>
<tr><td style=font-weight:bold>SUBMITTED BY :- </td><td></td></tr>
<tr><td></td><td>AKASH MANHAS ( C0873600 )</td></tr>
<tr><td></td><td>Rudhar Sharma ( C0871405 )</td></tr>
</table>
</div>
<div class=box3>
<table>
<tr><td style=font-weight:bold>MODEL USED </td><td>XG Boost</td></tr>
<tr><td style=font-weight:bold>LIBRARIES </td><td>Scikit.Learn</td></tr>
<tr><td> </td><td>pyAudioAnalysis</td></tr>
<tr><td style=font-weight:bold>ACCURACY </td><td>73.91 %</td></tr>
</table>
</div>

</body>
'''

return '''
    <!doctype html>
    <head>
    <title>Multi-Class Music Genre Classification Using Machine
Learning</title>
    </head>
    <style>
    body{
        width:1200px;
        margin:0 auto;
        font-family:arial;
        padding:10px;
        color:#193646;
    }
    h1{
        text-align:center;
        color:#193646;
        margin-top:50px;
    }
    h2{
        text-align:center;
        color:#193646;
    }
    h3{
        text-align:center;

```

```
        color:#193646;
    }
    form{
        width:560px;
        margin:0 auto;
    }
    #large{
        width:400px;
    }
    .top{
        height:100px;
        width:825px;
        margin:0 auto;
        border-bottom:5px solid #193646;
        padding-bottom:5px;
    }
    .image{
        width:100px;
        height:100px;

        float:left;
    }
    .image img{
        width:100%;
        height:100%;

    }
    .head{
        padding-top:30px;
        font-size:36px;
        float:left;
        width:720px;
color:#193646;
        text-align:center;
font-weight:bold;
    }
    .box{
        width:1000px;
padding:10px;
margin:40px auto;
padding-bottom:80px;
    }
    .box2{
width:470px;
padding:5px;
border-left:5px solid #193646;
margin-left:90px;
float:left;
```



```

}
.box3{
width:250px;
padding:5px;
border-left:5px solid #193646;
float:left;
margin-left:250px;
}

</style>
<body>
<div class=top>
    <div class=image></div>
    <div class=head>Lambton College in Mississauga</div>
</div>
<h2> Full Stack Software Development </h2>
<div class=box>
    <h1>Multi-Class Music Genre Classification Using Machine Learning</h1>
<h3> Select File To Classify </h3>
    <form method=post enctype=multipart/form-data>
        <input type=file name=file id=large>
        <input type=submit value=Classify>
    </form>
</div>

<div class=box2>
<table>
<tr><td> </td><td></td></tr>
<tr><td style=font-weight:bold>SUBMITTED BY :- </td><td></td></tr>
<tr><td></td><td>AKASH MANHAS ( C0873600 )</td></tr>
<tr><td></td><td>Rudhar Sharma ( C0871405 )</td></tr>
</table>
</div>
<div class=box3>
<table>
<tr><td style=font-weight:bold>MODEL USED </td><td>XG Boost</td></tr>
<tr><td style=font-weight:bold>LIBRARIES </td><td>Scikit.Learn</td></tr>
<tr><td> </td><td>pyAudioAnalysis</td></tr>
<tr><td style=font-weight:bold>ACCURACY </td><td>73.91 %</td></tr>
</table>
</div>


</body>
'''

if __name__ == "__main__":
    app.run()

```

Multi-Class Music Genre C x +

127.0.0.1:5000



# Lambton College in Mississauga

---

## Full Stack Software Development

### Multi-Class Music Genre Classification Using Machine Learning

Select File To Classify

Choose file

Galibaat-JimsherDJPunjab-Com1 (1).wav

Classify


SUBMITTED BY :-

AKASH MANHAS ( C0873600 )  
Rudhar Sharma ( C0871405 )

MODEL USED XG Boost  
LIBRARIES Scikit.Learn  
pyAudioAnalysis  
ACCURACY 73.91 %

Multi-Class Music Genre C x +

127.0.0.1:5000



# Lambton College in Mississauga

---

## Full Stack Software Development

### Multi-Class Music Genre Classification Using Machine Learning

Select File To Classify

Choose file

No file chosen

Classify

SUBMITTED BY :-

AKASH MANHAS ( C0873600 )  
Rudhar Sharma ( C0871405 )

MODEL USED XG Boost  
LIBRARIES Scikit.Learn  
pyAudioAnalysis  
ACCURACY 73.91 %

