**01-12-2022**

**Predictive-Modelling-on-Hit-Songs-Using-Repeated-Chorus**

**About Project:**

We examined this myth by creating a data set of choruses from popular artists and applied supervised Machine Learning (ML) techniques to predict the popularity of their works purely based on the audio features extracted from the chorus.

**Data Collection:**

* We get top artists from year 2006 to 2021 using Bill Board API and Top-100-Songs.
* We get 80 artists from year 2006 to 2021.
* Then we collect popular songs data of those 80 artists which includes Title of songs, Artist name.
* We define Label=1 for all popular songs.
* We collect all other songs of those artist from above mentioned years and make Label = 0 for all unpopular songs.
* Final dataset has 3 columns named Artist, Title, Label.
* We save the data in csv file.

df.to\_csv('Asad\_Billboard.csv', index=False)

* Shape of the data frame is (809,3).

**Songs Downloading:**

* Using Youtube-dl API, we downloaded songs in mp3 format.
* First, we load dataset of song names we prepared above.
* We make lists of popular and unpopular songs as follows:
* We define a function to download popular and unpopular songs in mp3 format.
* We downloaded 403 popular songs and 406 unpopular songs.
* We add new column in our data frame named “Song\_path”, which have path address of downloaded songs in the system.
* We save table to csv file as:

df.to\_csv('Asad\_Billboard\_Data\_fullsongs.csv', index=False)

* Shape of the data frame is (809,4).

**Chorus Extraction:**

* Using Pychorus library, we extracted the repeated chorus of our downloaded songs.
* First, we load dataset of songs we prepared “Asad\_Billboard\_Data\_fullsongs.csv”.
* We check the null values and found we have 2 null values in song\_path column.
* We dropped the null values as

df = df.dropna (axis=0)

* Shape of data frame is now (807,4)
* We define a function to extract 15s choruses in .wav format from songs as required in paper.
* After applying function, it was found that chorus of 15 songs are not extracted.
* We add new column in our data frame named “extracted\_chorus\_path”, which have path address of downloaded Chorus in the system.
* We dropped the rows with un extracted values.
* Shape of data frame is now (792,5)
* We save table to csv file as:

data1.to\_csv('Asad\_extracted\_chorus.csv', index=False)

**Audio Features:**

* Using Librosa library, we extracted the audio features of our downloaded choruses.
* We load csv file named 'Asad\_extracted\_chorus.csv'.
* We define a function to get statistics of each dimension data for each audio raw feature.

def statistics(list, feature, columns\_name, data):

* We define a function to extract audio features with statistics.

def extract\_features(audio\_path, title):

* After applying functions we get 518 audio features.
* We create dataframe for new extracted audio features:

newData = pd.DataFrame(data, columns=cols)

* Shape of data frame is now (792,519).
* We merge 2 data sets as:

final\_data = df.join(newData)

* Shape of data frame is now (792,523).
* We save table to csv file as:

final\_data.to\_csv('Asad\_feature\_extract.csv', index=False)

**EDA:**

* First, we load dataset named 'Asad\_feature\_extract.csv'.
* We check first five rows of data.
* We checked shape of data frame: (792,523).
* We check data types. It has Integer, Object, Float data type.
* We check Summary Statistics using data.describe( )
* We remove un necessary columns:

data.drop(["Artist","Title","song\_path","extracted\_chorus\_path"],axis=1,inplace=True)

* We check null values and duplicated values.
* Shape of the data now: (792,519)
* We check data balance of our target variable. We have 50.4 % popular data (label=1) and 49.6 % unpopular data (label=0).
* We check correlation using corr\_df = data.corr().
* We check MI scores and get top corelated features.
* We perform PCA for feature selection.
* we can observe that to achieve 95% variance, the dimension was reduced to 173 principal components from the actual 519 dimensions.
* We make new data frame with best features as

final\_data.to\_csv('improved\_best\_features\_df.csv')Shape of the data frame is (809,4).

**Model Building:**

* We have made 4 models

Gradient Boosting Classifier.

Linear Discriminant Analysis.

ADA Boosting Classifier.

KNN Classifier.

**Gradient Boosting Classifier:**

* We import useful libraries.
* We split data in to test and train sets.

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.20, random\_state=42)

* We fit our model on train data sets.
* We check our model on different learning rates.
* We're mainly interested in the classifier's accuracy on the validation set, but it looks like a learning rate of 0.75 gives us the best performance on the validation set i.e. 0.98.
* Now we can evaluate the classifier by checking its accuracy and creating a confusion matrix. Let's create a new classifier and specify the best learning rate we discovered.
* We have got Confusion Matrix

Confusion Matrix:

[[76 0]

[ 0 83]]

* We get the accuracy of 100% on test data set.

**Linear Discriminant Analysis:**

* We import useful libraries.
* We split data in to test and train sets.

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.20, random\_state=42)

* We fit our model on train data sets.
* We test out model on test set.
* We get the accuracy score of 100% on test data set.

**ADA Boost Classifier:**

* We import useful libraries.
* We split data in to test and train sets.

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.20, random\_state=42)

* We fit our model on train data sets.
* We test out model on test set.
* We get the accuracy score of 100% on test data set.

**KNN Classifier:**

* We import useful libraries.
* We split data in to test and train sets.

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.20, random\_state=42)

* We fit our model on train data sets.
* We test out model on test set.
* We get the accuracy score of 100% on test data set.

**Cross Validation:**

* We check over fitting and under fitting of models.
* We use K-Fold Methods.
* For GB Classifier, we get average CV score of 100.
* For LDA Classifier, we get average CV score of 27.
* For ADA boost Classifier, we get average CV score of 100.
* For KNN Classifier, we get average CV score of 32.8.
* Only GB Classifier and ADA Boost are normally fitted.

**Model Deployment:**

* In deployment we used feature\_extract\_04.csv file (which is final data after extracting all the chorus features).
* After that we drop the "Artist","Title", "Label", "song\_path", and "extracted\_chorus\_path" columns and then apply pca on remaining data

{drop columns command ->

X = dataset1.drop('Artist', axis=1).drop('Title',axis=1).drop('Label', axis=1).drop('song\_path', axis=1).drop('extracted\_chorus\_path', axis=1)}

{pca commands->

* model = pca(normalize=True)out = model.fit\_transform(X)}
* Then we store all best features with respective values in "columns" variable

{

columns = list(df['feature'][df['type']=='best'])    X = dataset[columns]

}

* Then predict the output with following command

prediction = model.predict(X)

output = 'Unpopular' if prediction[0] == 0 else 'Popular'

* Return render\_template('index.html', prediction\_text = f'The song is {output}')
* Now in web app we give the path of our system where we previously store the extracted choruses in .wav format.(e.g. in my case it is chorus\_extract/song\_name.wav))