K L UNIVERSITY

COMPUTER SCIENCE ENGINEERING DEPARTMENT

A Project Based Lab

Report On

PREDICTING THE COMMERCIAL SUCCESS OF SONGS BASED ON LYRICS AND OTHER METRICS

SUBMITTED BY:

ID NUMBER	NAME
180030519	BOSUKONDA TEJASWINI
180030528	ANGERI JASWANTH REDDY
180030543	CH.V.S.R.NAIDU

UNDER THE ESTEEMED GUIDANCE OF
DR. PRAGNYABAN MISHRA

KL UNIVERSITY
Greenfields,Vaddeswaram
522502 Guntur Dt., AP, India

DEPARTMENT OF BASIC ENGINEERING SCIENCES

CERTIFICATE

This is to certify that the project based laboratory report entitled "Predicting The Commercial Success Of Songs Based On Lyrics And Other Metrics" submitted by Ms. Bosukonda Tejaswini, Mr. Angeri Jaswanth Reddy and Mr. Ch.V.S.R.Naidu bearing Regd. Nos. 180030519, 180030528 and 180030543 to the Department of Basic Engineering Sciences, K L University in partial fulfillment of the requirements for the completion of a project based Laboratory in "Deep Learning" course in III B. Tech VI Semester, is a bonafide record of the work carried out by him/her under my supervision during the academic year 2020 –2021.

PROJECT SUPERVISOR	HEAD OF THEDEPARTMENT
DR. PRAGNYABAN MISHRA	DR. HARI KIRAN VEGE

ACKNOWLEDGEMENTS

It is a great pleasure for me to express my gratitude to our honorable President **Sri. Koneru Satyanarayana**, for giving the opportunity and platform with facilities in accomplishing the project-based laboratory report.

I express the sincere gratitude to our principal for his administration towards our academic growth.

I express my sincere gratitude to our Coordinator and HOD for his leadership and constant motivation provided in successful completion of our academic semester. I record it as my privilege to deeply thank for providing us the efficient faculty and facilities to make our ideas into reality.

I express my sincere thanks to our project supervisor for his/her novel association of ideas, encouragement, appreciation and intellectual zeal which motivated us to venture this project successfully.

Finally, it is pleased to acknowledge the indebtedness to all those who devoted themselves directly or indirectly to make this project report success.

Bosukonda Tejaswini(180030519)
Angeri Jaswanth Reddy(180030528)
Ch.V.S.R.Naidu(180030543)

ABSTRACT

The aim of the project is to determine the success of a song provided the quantifiable metrics such as the lyrics, genre, rhythm, beat, instrumentation, artist etcetera.

The success of a song can be of two types: prior and posterior of the release of the song. A song is said to be a success after the release of the song(posterior), if it is in trend or is a hot news. The parameters affecting the success depend upon the play count, user downloads, billboard ranking, news, reviews etc. This also depends upon the publicity of the song which depends upon the recommender system which suggest the song on various platforms. From the artists' perspective, the prior success depends upon the expected response of the listener based upon the audio features and sentiment of the lyrics.
Here, we design a model that predicts how likely a song will result in a hit (success rate), defined by making it the top trend list. This is done by training it with old songs' success rate given its metadata and then testing the model on new songs.

INDEX

INTRODUCTION

- Predicting the Commercial Success of Songs Based on Lyrics and Other Metrics algorithms is the project .

- Predicting the success of the songs using the Machine Learning Algorithms and Artificial Neural Network

- Deep Learning classification techniques can be used on these type of problem statements.

- Steps we implement to predict the success of the song are:

- Importing the dataset
- Data pre -processing

- Applying machine learning algorithms such as Logistic Regression

- Checking the performance of algorithm.

- Applying the Artificial Neural Network.

Now compare the accuracy of the above applied ml algorithms and ANN.

METHODOLOGY

- The million songs dataset is obtained from the Kaggle.

ATTRIBUTES:

1)The songs were divided into two types prior and posterior.

2)Success of the song predicted by bbhot (1 = Success, 0 = failure).

3)The song release time, song hotness, time signature confidence,release date ,time signature.

4)Mean absolute error and the accuracy are the main attributes for predicting whether the song is success or not.

5)Epoch is also used for the prediction of the success of the song.

❑ LOGISTIC REGRESSION:

- Importing dataset and performing data pre-processing
- Fitting logistic regression to the training set
- Predicting the test result
- Test accuracy of the result
- Visualizing the test set result

❑ ARTIFICIAL NEURAL NETWORK:

- Importing dataset and performing data pre-processing.
- Fitting the ANN to the training set
- Calculate the total no of epochs from the given dataset.
- Predicting the test result.
- Test accuracy of the result

IMPLEMENTATION

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

df = pd.read_csv('C:/Users/Umadevi/Desktop/ SpotifyFeatures.csv')
df.head()
df.info()

#removing features which do not affect the popularity of the song

drop_list = ['artist_id','artist_location', 'artist_latitude',
    'artist_longitude','artist_name', 'release', 'title' ,'song_hotttnesss']
x = df.drop(drop_list, axis=1)
x.info()

#data visualization

import seaborn as sns
sns.pairplot(df)
f, ax = plt.subplots(1, figsize=(10,8))
sns.heatmap(df.corr(), annot=True, ax=ax)

#handling null values

x.isnull().sum()
x["artist_familiarity"] =
    x["artist_familiarity"].fillna(x["artist_familiarity"].median())

#dependent variable
y = df.bbhot
#independent variable
x = x.drop("bbhot", axis=1)
x.head()
print(y.shape)
x.shape
```

```
#splitting into train and test set

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.33,
random_state=10)

#using logistic regression
from sklearn.linear_model import LogisticRegression
model = LogisticRegression(solver='liblinear')
model.fit(X_train, y_train)

# make predictions for test data
y_pred = model.predict(X_test)
accuracy = model.score(X_test, y_test)
print("Accuracy: %.2f%%" % (accuracy * 100.0))

#model evaluation
from sklearn.model_selection import cross_val_score
def testingModel(model, X_train, Y_train):
    scores = cross_val_score(model, X_train, Y_train, cv=10, scoring =
"roc_auc")
    print("Scores:", scores)
    print("Mean:", scores.mean())
    print("Standard Deviation:", scores.std())
    return scores.mean()
acc_log = testingModel(model, X_train, y_train)


#modelling using ANN`
#normalising the dataset

from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

```
print(X_train.shape)
print(X_test.shape)

from keras import Sequential
from keras.layers import Dense, Dropout
model = Sequential()
model.add(Dense(14, input_dim=14, activation='relu'))
model.add(Dropout(rate=0.2))
model.add(Dense(9, activation='relu'))
model.add(Dropout(rate=0.2))
model.add(Dense(4, activation='relu'))
model.add(Dropout(rate=0.2))
model.add(Dense(1, activation='linear'))
model.summary()

model.compile(loss='mse', optimizer='adam', metrics=['mse', 'mae',
'accuracy'])

m = model.fit(X_train, y_train, epochs=150, batch_size=50, verbose=1,
validation_split=0.2)
import matplotlib.pyplot as plt
print(m.history.keys())

# "Loss"
plt.plot(m.history['loss'])
plt.plot(m.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'validation'], loc='upper left')
plt.show()

from sklearn.metrics import mean_absolute_error, accuracy_score
print("MAE : ", mean_absolute_error(y_test, y_pred))
print("Accuracy score : ",accuracy_score(y_test,y_pred))
```

RESULTS

Importing dataset

In [1]:
```
import pandas as pd
df = pd.read_csv("project.csv")
df.head()
```

Out[1]:

	artist_familiarity	artist_hotttnesss	artist_id	artist_latitude	artist_location	artist_longitude	artist_name	duration	end_of_fade_in	key	...	m
0	0.780462	0.574275	ARMQHX71187B9890D3	NaN	Atlanta, GA	NaN	Mastodon	280.21506	0.238	5	...	
1	0.581794	0.401998	ARD7TVE1187B99BFB1	NaN	California - LA	NaN	Casual	218.93179	0.247	1	...	
2	0.630630	0.417500	ARMJAGH1187FB546F3	35.14968	Memphis, TN	-90.04892	The Box Tops	148.03546	0.148	6	...	
3	0.487357	0.343428	ARKRRTF1187B9984DA	NaN	NaN	NaN	Sonora Santanera	177.47546	0.282	8	...	
4	0.630382	0.454231	AR7G5I41187FB4CE6C	NaN	London, England	NaN	Adam Ant	233.40363	0.000	0	...	

5 rows × 23 columns

In [2]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10001 entries, 0 to 10000
Data columns (total 23 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   artist_familiarity        9997 non-null   float64
 1   artist_hotttnesss         10001 non-null  float64
 2   artist_id                 10001 non-null  object
 3   artist_latitude           3742 non-null   float64
 4   artist_location           5709 non-null   object
 5   artist_longitude          3742 non-null   float64
 6   artist_name               10001 non-null  object
 7   duration                  10001 non-null  float64
 8   end_of_fade_in            10001 non-null  float64
 9   key                       10001 non-null  int64
 10  key_confidence            10001 non-null  float64
 11  loudness                  10001 non-null  float64
 12  mode                      10001 non-null  int64
 13  mode_confidence           10001 non-null  float64
 14  release                   10001 non-null  object
 15  song_hotttnesss           5649 non-null   float64
 16  start_of_fade_out         10001 non-null  float64
 17  tempo                     10001 non-null  float64
 18  time_signature            10001 non-null  int64
 19  time_signature_confidence 10001 non-null  float64
 20  title                     10000 non-null  object
 21  year                      10001 non-null  int64
 22  bbhot                     10001 non-null  int64
dtypes: float64(13), int64(5), object(5)
memory usage: 1.8+ MB
```

Data preprocessing

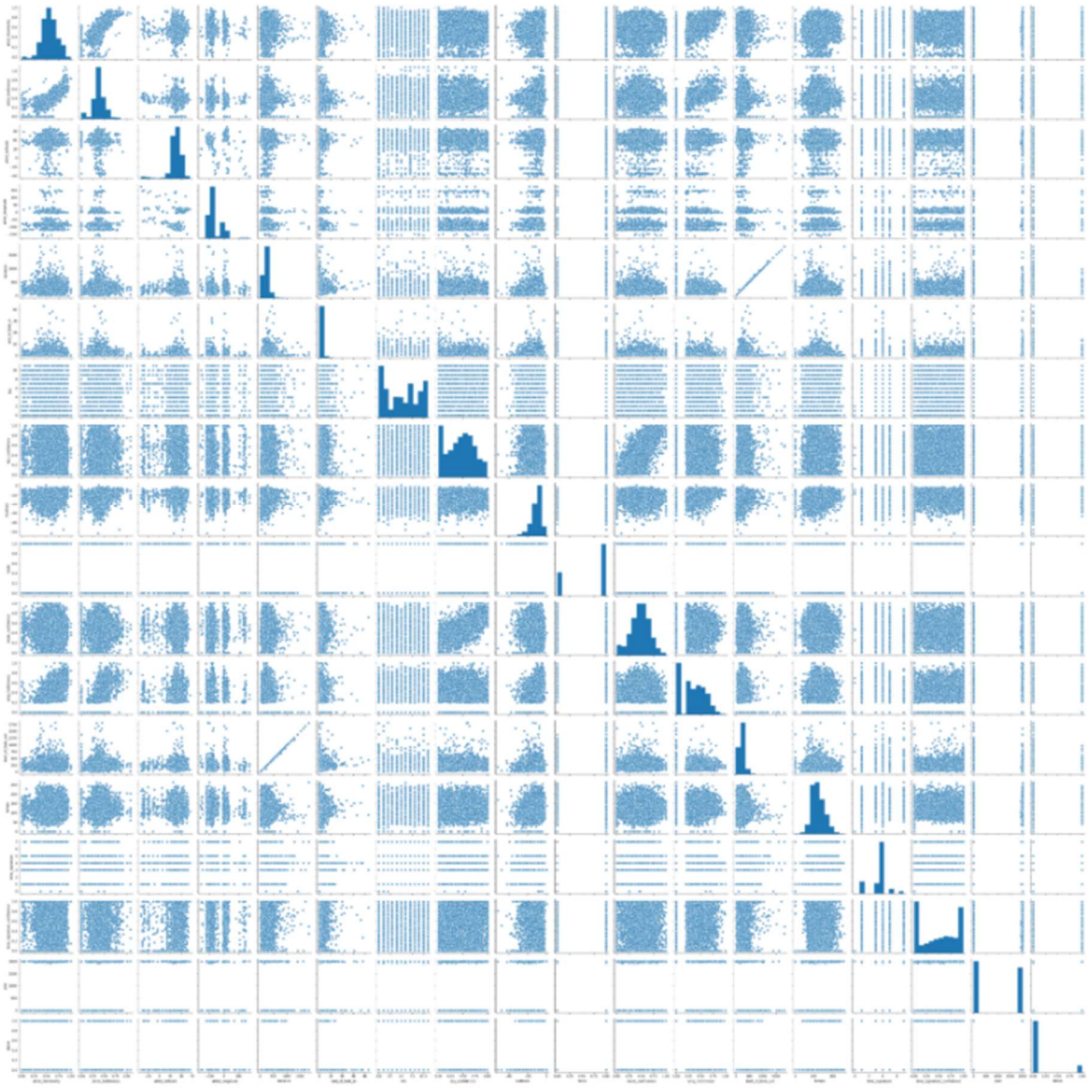
In [3]:
```
#removing features which do not affect the popularity of the song

drop_list = ['artist_id','artist_location', 'artist_latitude', 'artist_longitude','artist_name', 'release', 'title' ,'song_hotttr
x = df.drop(drop_list, axis=1)
```

In [4]:
```
x.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10001 entries, 0 to 10000
Data columns (total 15 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   artist_familiarity        9997 non-null   float64
 1   artist_hotttnesss         10001 non-null  float64
 2   duration                  10001 non-null  float64
 3   end_of_fade_in            10001 non-null  float64
 4   key                       10001 non-null  int64
 5   key_confidence            10001 non-null  float64
 6   loudness                  10001 non-null  float64
 7   mode                      10001 non-null  int64
 8   mode_confidence           10001 non-null  float64
 9   start_of_fade_out         10001 non-null  float64
 10  tempo                     10001 non-null  float64
 11  time_signature            10001 non-null  int64
 12  time_signature_confidence 10001 non-null  float64
 13  year                      10001 non-null  int64
 14  bbhot                     10001 non-null  int64
dtypes: float64(10), int64(5)
memory usage: 1.1 MB
```

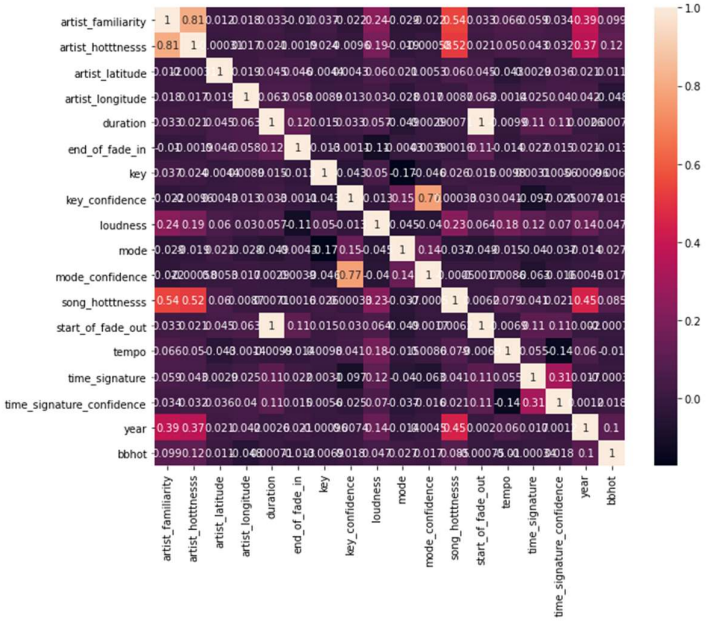
In [5]:
```
import seaborn as sns
sns.pairplot(df)
```

Out[5]:
```
<seaborn.axisgrid.PairGrid at 0x135c902e040>
```



In [8]:
```
f, ax = plt.subplots(1, figsize=(10,8))
sns.heatmap(df.corr(), annot=True, ax=ax)
```

Out[8]: <matplotlib.axes._subplots.AxesSubplot at 0x13d8cdc9820>



In [9]:
```
x.isnull().sum()
```

Out[9]:
```
artist_familiarity           4
artist_hotttnesss            0
duration                     0
end_of_fade_in               0
key                          0
key_confidence               0
loudness                     0
mode                         0
mode_confidence              0
start_of_fade_out            0
tempo                        0
time_signature               0
time_signature_confidence    0
year                         0
bbhot                        0
dtype: int64
```

In [10]:
```
#handling null values
x["artist_familiarity"] = x["artist_familiarity"].fillna(x["artist_familiarity"].median())
```

In [11]:
```
#dependent variable
y = df.bbhot
```

In [12]:
```
x = x.drop("bbhot", axis=1)
```

In [13]:
```
x.head()
```

Out[13]:

	artist_familiarity	artist_hotttnesss	duration	end_of_fade_in	key	key_confidence	loudness	mode	mode_confidence	start_of_fade_out	tempo	time_sigr
0	0.780462	0.574275	280.21506	0.238	5	0.555	-3.306	1	0.500	275.528	173.205	
1	0.581794	0.401998	218.93179	0.247	1	0.736	-11.197	0	0.636	218.932	92.198	
2	0.630630	0.417500	148.03546	0.148	6	0.169	-9.843	0	0.430	137.915	121.274	
3	0.487357	0.343428	177.47546	0.282	8	0.643	-9.689	1	0.565	172.304	100.070	
4	0.630382	0.454231	233.40363	0.000	0	0.751	-9.013	1	0.749	217.124	119.293	

In [14]:
```
print(y.shape)
```

```
(10001,)
```

In [15]:
```
x.shape
```

Out[15]:
```
(10001, 14)
```

Training, testing and splitting dataset

In [16]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.33, random_state=10)
```

Modelling using logistic regression

In [17]:
```
#using logistic regression
from sklearn.linear_model import LogisticRegression
model = LogisticRegression(solver='liblinear')
model.fit(X_train, y_train)
```

Out[17]:
```
LogisticRegression(solver='liblinear')
```

In [18]:
```
# make predictions for test data
y_pred = model.predict(X_test)
accuracy = model.score(X_test, y_test)
print("Accuracy: %.2f%%" % (accuracy * 100.0))
```

```
Accuracy: 88.31%
```

In [19]:
```
from sklearn.model_selection import cross_val_score
def testingModel(model, X_train, Y_train):
    scores = cross_val_score(model, X_train, Y_train, cv=10, scoring = "roc_auc")
    print("Scores:", scores)
    print("Mean:", scores.mean())
    print("Standard Deviation:", scores.std())
    return scores.mean()

acc_log = testingModel(model, X_train, y_train)
```

```
Scores: [0.59338983 0.64002119 0.58137712 0.6107839  0.58869396 0.617787
 0.59370349 0.68754323 0.64576914 0.61294515]
Mean: 0.6172014005014865
Standard Deviation: 0.03091871684488421
```

Modelling using Artificial Neural Network

In [20]:
```
#normalising the dataset
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
print(X_train.shape)
print(X_test.shape)
```

```
(6700, 14)
(3301, 14)
```

In [21]:
```
from keras import Sequential
from keras.layers import Dense, Dropout
model = Sequential()
model.add(Dense(14, input_dim=14, activation='relu'))
model.add(Dropout(rate=0.2))
model.add(Dense(9, activation='relu'))
model.add(Dropout(rate=0.2))
model.add(Dense(4, activation='relu'))
model.add(Dropout(rate=0.2))
model.add(Dense(1, activation='linear'))
```

In [22]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 14)                210
_________________________________________________________________
dropout (Dropout)            (None, 14)                0
_________________________________________________________________
dense_1 (Dense)              (None, 9)                 135
_________________________________________________________________
dropout_1 (Dropout)          (None, 9)                 0
_________________________________________________________________
dense_2 (Dense)              (None, 4)                 40
_________________________________________________________________
dropout_2 (Dropout)          (None, 4)                 0
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 5
=================================================================
Total params: 390
Trainable params: 390
Non-trainable params: 0
_________________________________________________________________
```

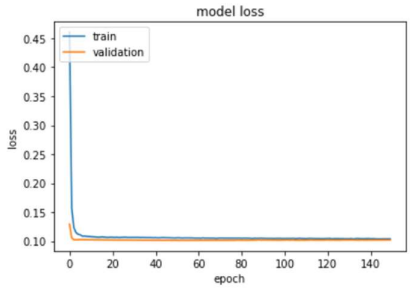
In [23]:
```
model.compile(loss='mse', optimizer='adam', metrics=['mse', 'mae', 'accuracy'])
```

In [24]:
```
m = model.fit(X_train, y_train, epochs=150, batch_size=50, verbose=1, validation_split=0.2)
```

```
108/108 [==============================] - 0s 4ms/step - loss: 0.1013 - mse: 0.1013 - mae: 0.2064 - accuracy: 0.8823 - val_lo
ss: 0.1021 - val_mse: 0.1021 - val_mae: 0.2189 - val_accuracy: 0.8843
Epoch 145/150
108/108 [==============================] - 0s 4ms/step - loss: 0.1028 - mse: 0.1028 - mae: 0.2128 - accuracy: 0.8801 - val_lo
ss: 0.1021 - val_mse: 0.1021 - val_mae: 0.2153 - val_accuracy: 0.8843
Epoch 146/150
108/108 [==============================] - 0s 4ms/step - loss: 0.1046 - mse: 0.1046 - mae: 0.2119 - accuracy: 0.8777 - val_lo
ss: 0.1019 - val_mse: 0.1019 - val_mae: 0.2107 - val_accuracy: 0.8843
Epoch 147/150
108/108 [==============================] - 0s 4ms/step - loss: 0.1028 - mse: 0.1028 - mae: 0.2074 - accuracy: 0.8802 - val_lo
ss: 0.1019 - val_mse: 0.1019 - val_mae: 0.2123 - val_accuracy: 0.8843
Epoch 148/150
108/108 [==============================] - 0s 3ms/step - loss: 0.1056 - mse: 0.1056 - mae: 0.2123 - accuracy: 0.8762 - val_lo
ss: 0.1020 - val_mse: 0.1020 - val_mae: 0.2125 - val_accuracy: 0.8843
Epoch 149/150
108/108 [==============================] - 0s 3ms/step - loss: 0.1014 - mse: 0.1014 - mae: 0.2039 - accuracy: 0.8821 - val_lo
ss: 0.1020 - val_mse: 0.1020 - val_mae: 0.2154 - val_accuracy: 0.8843
Epoch 150/150
108/108 [==============================] - 0s 3ms/step - loss: 0.0976 - mse: 0.0976 - mae: 0.2025 - accuracy: 0.8875 - val_lo
ss: 0.1021 - val_mse: 0.1021 - val_mae: 0.2177 - val_accuracy: 0.8843
```

In [25]:
```
import matplotlib.pyplot as plt
print(m.history.keys())
# "Loss"
plt.plot(m.history['loss'])
plt.plot(m.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'validation'], loc='upper left')
plt.show()
```

```
dict_keys(['loss', 'mse', 'mae', 'accuracy', 'val_loss', 'val_mse', 'val_mae', 'val_accuracy'])
```



In [31]:
```
from sklearn.metrics import mean_absolute_error, accuracy_score
print("MAE : ", mean_absolute_error(y_test, y_pred))
print("Accuracy score : ",accuracy_score(y_test,y_pred))
```

```
MAE :  0.11693426234474402
Accuracy score :  0.883065737655256
```

CONCLUSION

- Both machine learning methods and Artificial neural network models resulted with approximately same accuracy.
- Both the models successfully predicted the success or hotness of a song through both supervised and unsupervised methods.
- The analysis of the results signifies that the integration of multidimensional data along with different classification, feature selection and dimensionality reduction techniques can provide auspicious tools for inference in this domain.
- Further research in this field should be carried out for the better performance of the classification techniques so that it can predict on more variables.

REFERENCES

- EchoNest. EchoNest API. Oct. 2014 URL: http://developer.echonest.com/docs/v4

- Wang, D. Zhang and Y. H. Huang "Commercial Success of the song" Using Machine Learning" (2018), Vol. 66, NO. 7.

- Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. The Million Song Dataset. In Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR 2011), 2011