Team Name: Faang

1. **Team Members and Email ID:**

Swetha Mamidiga (Smami3@unh.newhaven.edu)

Ajay Kumar (Avatt1@unh.newhaven.edu)

Sruthi Sundararajan (ssund3@unh.newhaven.edu)

2. **Research Question:**

MUSIC MAKES AN IMPACT

Using data mining techniques and Spotify to predict a person's mood.
Would it be possible of predict a person mood to play same type of songs if a person chooses one type of song? The data behind their music listening history would reveal any insight into their emotional states.?

Selected dataset:

The dataset which we have selected is the actual data based on the people's moods, it includes Danceability, Acousticness, Energy,Instrumentalness, Liveness, Valence, Loudness, Speechiness and Tempo because they have more influence to classify the tracks .

3. **List of Data Mining techniques:**

In our project we have used Regression because data in the dataset is numerical. Regression is a data science task of predicting the value of target (numerical variable) by building a model based on one or more predictors. In Regression we have used Frequency table and Covariance Matrix. In Frequency table we use classifier as Decision Tree and in Covariance Matrix we use classifier as Logistic Regression.

For Frequency Matrix –

Decision Tree Classifier

For Covariance Matrix –

Logistic Regression Classifier

For similar function:

Random forest

4. **Parameters and Hyper parameters:**

Logistic regression:

Hyper parameters:

C= 1e20

Decision Tree:

Hyper parameters:

We used all hyper parameters here

ccp_alpha=0.0, class_weight=None, criterion='gini',
max_depth=None, max_features=None, max_leaf_nodes=None,
min_impurity_decrease=0.0, min_impurity_split=None,
min_samples_leaf=1, min_samples_split=2,
min_weight_fraction_leaf=0.0, presort='deprecated',
random_state=None, splitter='best

Random Forest :

Hyper parameters:

We used all the in built parameters

5.Techniques used to optimize the data:

1.Logistic Regression:

In the logistic regression we have defined a grid which consist of all the hyper parameters which we can use to optimize the data. The defined grid is named to be param_grid.

```
param_grid = [
{'penalty' : ['l1', 'l2', 'elasticnet', 'none'],
'C' : np.logspace(-4, 4, 20),
'solver' : ['lbfgs','newton-cg','liblinear','sag','saga'],
'max_iter' : [100, 1000,2500, 5000]
}
```

Later we used GridSearchCV and best_estimator_. GridSearchCV runs through all predefined hyperparameters and fit your model on your training set. So, in the end, you can select the best parameters from the hyperparameters

Best_estimator results the best outcome from the all hyperparameters present in the grid.

```
clg = GridSearchCV(logRegModel, param_grid = param_grid, cv = 3,
verbose=T rue, n_jobs=-1)


best_clf.best_estimator_
```

At last we used : `'Accuracy - : {best_clf.score(X_train, y_train):.3f}'` to get the best accurate score for this classifier. The accuracy obtained is 0.67.

2.Decision Tree:

In the decisionTree we have defined a grid which consist of all the hyper parameters which we can use to optimize the data. The defined grid is named to be params.

```
params = [
{ 'max_depth': [2, 3, 5, 10, 20],
      'min_samples_leaf': [5, 10, 20, 50, 100],
      'criterion': ['gini','entropy']
}
]
```

Later we used GridSearchCV and best_estimator_. GridSearchCV runs through all predefined hyperparameters and fit your model on your training set. So, in the end, you can select the best parameters from the hyperparameters

Best_estimator results the best outcome from the all hyperparameters present in the grid.

```
grid_search = GridSearchCV(estimator=decisontree,
param_grid=params,
cv=4, n_jobs=-
1, verbose=1, scoring = "accuracy")
best_Desicionmodel.best_estimato
r_
```

At last we used : `Accuracy - : {best_Desicionmodel.score(X_train, y_train):.3f}`

to get the best accurate score for this classifier. The accuracy obtained is 0.78.

3.Random forest :

As we did hyper parameter tuning the best estimator we got is mentioned below:

```
RandomForestRegressor(max_depth=10, max_features=5, n_estimators=25, n_jobs=-1)
```

But we failed to estimate the accuracy for it. AS we tried we got the output as follows:

ACCURACY ==➔ -INF%

5. **Visualization technique used:**

We used dataframe to visualize the data in form of tables where all hyperparameters are considered.

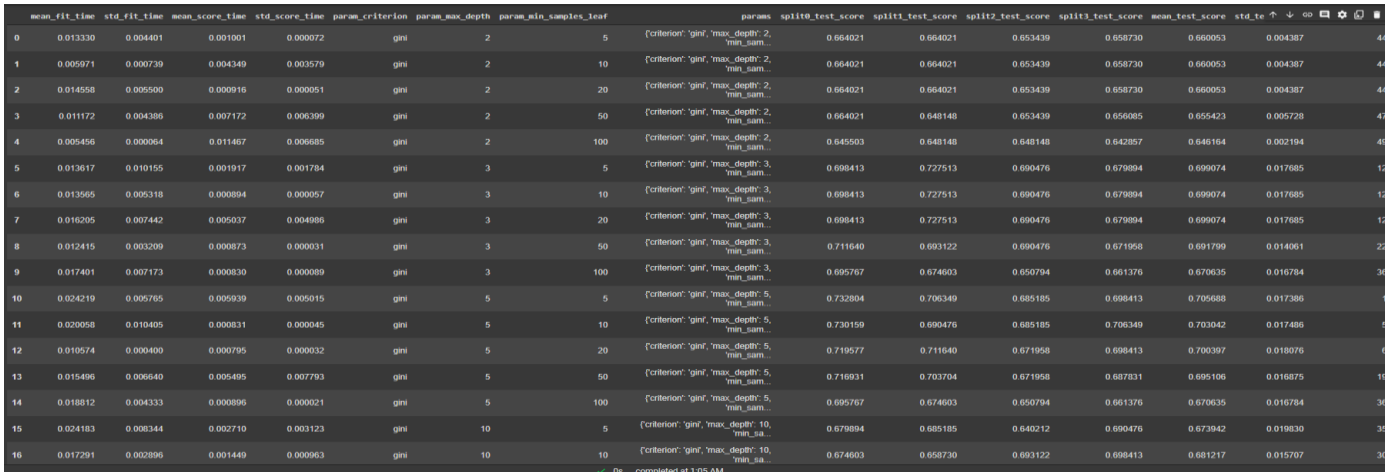
1.Logistic Regression:

`pd.DataFrame(clg.cv_results_) is used.`

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_C	param_max_iter	param_penalty	param_solver	params	split0_test_score	split1_test_score	split2_test_score	mean_test_score	std_test_score	rank_test_score
0	0.001527	0.001005	0.000000	0.000000	0.0001	100	l1	lbfgs	{'C': 0.0001, 'max_iter': 100, 'penalty': 'l1'...	NaN	NaN	NaN	NaN	NaN	1600
1	0.000794	0.000077	0.000000	0.000000	0.0001	100	l1	newton-cg	{'C': 0.0001, 'max_iter': 100, 'penalty': 'l1'...	NaN	NaN	NaN	NaN	NaN	1047
2	0.004465	0.001875	0.000869	0.000083	0.0001	100	l1	liblinear	{'C': 0.0001, 'max_iter': 100, 'penalty': 'l1'...	0.492063	0.492063	0.494048	0.492725	0.000935	861
3	0.000767	0.000096	0.000000	0.000000	0.0001	100	l1	sag	{'C': 0.0001, 'max_iter': 100, 'penalty': 'l1'...	NaN	NaN	NaN	NaN	NaN	1041
4	0.003017	0.000400	0.000733	0.000059	0.0001	100	l1	saga	{'C': 0.0001, 'max_iter': 100, 'penalty': 'l1'...	0.507937	0.507937	0.494048	0.503307	0.006547	850
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1595	0.010682	0.001222	0.000799	0.000033	10000.0	5000	none	lbfgs	{'C': 10000.0, 'max_iter': 5000, 'penalty': 'n...	0.682540	0.658730	0.642857	0.661376	0.016308	115
1596	0.016953	0.002581	0.000780	0.000007	10000.0	5000	none	newton-cg	{'C': 10000.0, 'max_iter': 5000, 'penalty': 'n...	0.682540	0.658730	0.642857	0.661376	0.016308	115
1597	0.000694	0.000055	0.000000	0.000000	10000.0	5000	none	liblinear	{'C': 10000.0, 'max_iter': 5000, 'penalty': 'n...	NaN	NaN	NaN	NaN	NaN	1048
1598	0.019762	0.008119	0.000904	0.000070	10000.0	5000	none	sag	{'C': 10000.0, 'max_iter': 5000, 'penalty': 'n...	0.682540	0.658730	0.642857	0.661376	0.016308	115
1599	0.020313	0.006930	0.000917	0.000064	10000.0	5000	none	saga	{'C': 10000.0, 'max_iter': 5000, 'penalty': 'n...	0.682540	0.658730	0.642857	0.661376	0.016308	115

1600 rows × 15 columns

2.Decision Tree:



	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_criterion	param_max_depth	param_min_samples_leaf	params	split0_test_score	split1_test_score	split2_test_score	split3_test_score	mean_test_score	std_te
0	0.013330	0.004401	0.001001	0.000072	gini	2	5	{'criterion': 'gini', 'max_depth': 2, 'min_sam...	0.664021	0.664021	0.653439	0.658730	0.660053	0.004387
1	0.005971	0.000739	0.004349	0.003579	gini	2	10	{'criterion': 'gini', 'max_depth': 2, 'min_sam...	0.664021	0.664021	0.653439	0.658730	0.660053	0.004387
2	0.014558	0.005500	0.000916	0.000051	gini	2	20	{'criterion': 'gini', 'max_depth': 2, 'min_sam...	0.664021	0.664021	0.653439	0.658730	0.660053	0.004387
3	0.011172	0.004386	0.007172	0.006399	gini	2	50	{'criterion': 'gini', 'max_depth': 2, 'min_sam...	0.664021	0.648148	0.653439	0.656085	0.655423	0.005728
4	0.005456	0.000064	0.011467	0.006685	gini	2	100	{'criterion': 'gini', 'max_depth': 2, 'min_sam...	0.645503	0.648148	0.648148	0.642857	0.646164	0.002194
5	0.013617	0.010155	0.001917	0.001784	gini	3	5	{'criterion': 'gini', 'max_depth': 3, 'min_sam...	0.698413	0.727513	0.690476	0.679894	0.699074	0.017685
6	0.013565	0.005318	0.000894	0.000057	gini	3	10	{'criterion': 'gini', 'max_depth': 3, 'min_sam...	0.698413	0.727513	0.690476	0.679894	0.699074	0.017685
7	0.016205	0.007442	0.005037	0.004986	gini	3	20	{'criterion': 'gini', 'max_depth': 3, 'min_sam...	0.698413	0.727513	0.690476	0.679894	0.699074	0.017685
8	0.012415	0.003209	0.000873	0.000031	gini	3	50	{'criterion': 'gini', 'max_depth': 3, 'min_sam...	0.711640	0.693122	0.690476	0.671958	0.691799	0.014061
9	0.017401	0.007173	0.000830	0.000089	gini	3	100	{'criterion': 'gini', 'max_depth': 3, 'min_sam...	0.695767	0.674603	0.650794	0.661376	0.670635	0.016784
10	0.024219	0.005765	0.005939	0.005015	gini	5	5	{'criterion': 'gini', 'max_depth': 5, 'min_sam...	0.732804	0.706349	0.685185	0.698413	0.705688	0.017386
11	0.020058	0.010405	0.000831	0.000045	gini	5	10	{'criterion': 'gini', 'max_depth': 5, 'min_sam...	0.730159	0.690476	0.685185	0.706349	0.703042	0.017486
12	0.010574	0.000400	0.000795	0.000032	gini	5	20	{'criterion': 'gini', 'max_depth': 5, 'min_sam...	0.719577	0.711640	0.671958	0.698413	0.700397	0.018076
13	0.015496	0.006640	0.005495	0.007793	gini	5	50	{'criterion': 'gini', 'max_depth': 5, 'min_sam...	0.716931	0.703704	0.671958	0.687831	0.695106	0.016875
14	0.018812	0.004333	0.000896	0.000021	gini	5	100	{'criterion': 'gini', 'max_depth': 5, 'min_sam...	0.695767	0.674603	0.650794	0.661376	0.670635	0.016784
15	0.024183	0.008344	0.002710	0.003123	gini	10	5	{'criterion': 'gini', 'max_depth': 10, 'min_sa...	0.679894	0.685185	0.640212	0.690476	0.673942	0.019830
16	0.017291	0.002896	0.001449	0.000963	gini	10	10	{'criterion': 'gini', 'max_depth': 10, 'min_sa...	0.674603	0.658730	0.693122	0.698413	0.681217	0.015707

7.Conclusion:

To check which model suits best we have evaluated dataset using three different modeling techniques like logistic regression , decision tree and random forest.

With higher accuracy of 0.78 decision tree is the best technique for our dataset, followed by logistic regression with 0.67. As a result, we declared decision tree as best to use.

Coming to Optimization:

We have used various hyper parameters for each data mining technique to improve our accuracy for random forest, Logistic regression and decision tree. As a result, we obtained the below accuracies for the techniques:

Decision tree with higher accuracy of 78 is the best performer on our dataset, followed by logistic regression with 67.

8. GitHub repository link:

https://github.com/Ajay-kumarv/music-makes-an-impact.git