Course: DATS 6401 – Visualization of Complex Data

Instructor: Dr. Reza Jafari

Final Term Project

Topic: Analysis of Spotify Data

Name: Rehapriadarsini Manikandasamy

RM
Initials

05.04.2022
Date

TABLE OF CONTENTS

Abstract

The project work focuses on developing a web-based application using python dash package. The analysis involves usage of Spotify data containing top 100 songs for the year 1921 to 2020. The application has numerous tabs which helps the user to navigate from low level to high level interpretation of the dataset. The popularity of the song in the initial years turned out to be less due to the less interactions with users but over the year the popularity has increased based on the aesthetics of the song. The dashboard is an interactive user-friendly application to analyze the Spotify data.

INTRODUCTION

Dash apps give a point-&-click interface to models written in Python, vastly expanding the notion of what's possible in a traditional "dashboard." The dynamic dashboard created using dash helps the user to visualize the popularity of top 100 songs over the years from Spotify music platform.

The app layout has been designed in such a way the end user can navigate easily to desired part of analysis. The dataset is preprocessed by removing null values. Since data was cleaned the step of preprocessing was skipped in the project. One of the important tasks in analyzing a dataset is to detect and remove the outliers. The outliers can be identified using different methods and this project uses box plots for outlier identification and IQR method for outlier removal.

The cleaned dataset is then tested for normality and PCA is done to reduce the dimensions of the original feature space. PCA analysis helps in finding the best number of feature components for modelling as well finding the correlation between the variables. The heatmap shows the correlation matrix of the dataset which helps in finding the collinearity between the variables.

The plots like histograms, line, violin, rug, pie, scatter matrix and scatter plot with regression line are developed to study the dataset in wider view. The final dashboard has subplots of dependent variable to analyze the impact of other variables in its prediction.

A detailed description about these techniques is discussed in the next chapter.

METHOD, THEORY AND PROCEDURES

Dash:

Dash is an open-source framework for building analytical applications, with no Javascript required, and it is tightly integrated with the Plotly graphing library. Dash is a python framework mostly used for building data visualization apps.

Dash Tabs:

The dcc.Tabs and dcc.Tab components can be used to create tabbed sections in your app. The dcc.Tab component controls the style and value of the individual tab and the dcc.Tabs component hold a collection of dcc.Tab components.

Dash Callbacks:

functions that are automatically called by Dash whenever an input component's property changes, to update some property in another component (the output).

Dash Core Components:

The Dash Core Components module (dash.dcc) can be imported and used with from dash import dcc and gives you access to many interactive components, including, dropdowns, checklists, and sliders.

Dash HTML Components:

Dash is a web application framework that provides pure Python abstraction around HTML, CSS, and JavaScript. Instead of writing HTML or using an HTML templating engine, you compose your layout using Python with the Dash HTML Components module (dash.html).

Line Plot:

A line plot is a graph that displays data using a number line. To create a line plot, first create a number line that includes all the values in the data set. Next, place an X (or dot) above each data value on the number line. If a value occurs more than once in a data set, place an Xs over that number for each time it occurs.

Histogram:

A histogram is a graphical representation that organizes a group of data points into user-specified ranges. Similar in appearance to a bar graph, the histogram condenses a data series into an easily interpreted visual by taking many data points and grouping them into logical ranges or bins.

Pie Chart:

Pie charts can be used to show percentages of a whole and represents percentages at a set point in time. Unlike bar graphs and line graphs, pie charts do not show changes over time.

Dropdown:

To create a basic dropdown, provide options and a value to dcc.Dropdown in that order.

Graph:

The dcc.Graph component can be used to render any plotly-powered data visualization, passed as the figure argument.

Input:

Number type is now close to native HTML5 input behavior across browsers. We also apply a strict number casting in callbacks: valid number converts into corresponding number types, and invalid number converts into None.

Correlation Matrix:

A correlation matrix is a table showing correlation coefficients between sets of variables. Each random variable (Xi) in the table is correlated with each of the other values in the table (Xj). This allows you to see which pairs have the highest correlation.

SVD:

Singular Value Decomposition or SVD the popular technique for dimensionality reduction. This is a linear algebra technique to create a projection of a sparse dataset prior to fitting a model.

s, d, v = np.linalg.svd(H)

Principal Component Analysis:

Reducing the number of input variables for a predictive model is referred to as dimensionality reduction. Fewer input variables can result in a simpler predictive model that may have better performance when making predictions on new data. Perhaps the most popular technique for dimensionality reduction in machine learning is Principal Component Analysis, or PCA for short. This is a technique that comes from the field of linear algebra and can be used as a data preparation technique to create a projection of a dataset prior to fitting a model.

Outlier Detection and Removal:

Outliers are data points that are far from other data points. With outlier detection and treatment, anomalous observations are viewed as part of different populations to ensure stable findings for the population of interest.

Outlier Detection- Interquartile Range (IQR) - A commonly used rule says that a data point is an outlier if it is more than 1.5*IQR above the third quartile or below the first quartile. IQR is calculated as : Q3-Q1. Low outliers are below Q1 − 1.5 * IQR and High outliers are above Q3 + 1.5 * IQR where Q1 is the first quartile and Q3 is the third quartile.

Kolmogorov-Smirnov (K-S) Test:

The Kolmogorov-Smirnov (K-S) Test compares your data with a known distribution and lets you know if they have the same distribution. The K-S test is non-parametric test. It is commonly used as a test for normality to see if your data is normally distributed.

H0: The data are Normally distributed. p-value > alpha

H1: The data are not Normally distributed. p-value < alpha

Shapiro-Wilk Test:

The Shapiro-Wilk test is a way to tell if a random sample comes from normal distribution. In practice, the Shapiro-Wilk test is believed to be a reliable test of normality, although there is some suggestion that the test may be suitable for smaller samples of data.

H0: The data are Normally distributed. p-value > alpha

H1: The data are not Normally distributed. p-value < alpha

D’Agostino’s K2 test:

D’Agostino’s K2 test is a goodness-of-fit measure of departure from normality, that is the test aims to establish whether or not the given sample comes from a normally distributed population.

H0: The data are Normally distributed. p-value > alpha

H1: The data are not Normally distributed. p-value < alpha

Procedure:

1. Load the dataset and necessary libraries
2. Setup the dash app layout with necessary tabs
3. Initialize the layout for tab with necessary dash core components and html components.
4. Setup callback functions to make the app interactive with user's choice of operations
5. Repeat step 3 and 4 for all the tabs
6. Run the app locally to make sure everything is aligned
7. Create an account in google cloud console and add the python file and docker file in the editor
8. Follow the steps provided in the following article to deploy and publish the app in the internet.
   https://medium.com/kunder/deploying-dash-to-cloud-run-5-minutes-c026eeea46d4

EXPERIMENTAL SETUP

Required libraries:

The following python libraries are required to run the code file seamlessly.

```
import dash
import dash
import numpy as np
import plotly.express as px
from dash import dcc
from dash.dependencies import Input,Output
from dash import html
import pandas as pd
from statsmodels.graphics.gofplots import qqplot
from sklearn.preprocessing import StandardScaler
from numpy import linalg as la
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import kstest
from scipy.stats import shapiro
from scipy.stats import normaltest
import plotly.graph_objects as go
```

DESCRIPTION OF THE DATASET:

The dataset used in the project consists of more than 160,000 songs collected from Spotify Web API. The dataset is from Spotify and contains 169k songs from the year 1921 to year 2020. Each year got top 100 songs.

Attribute Information:

- id (Id of track generated by Spotify)

  Numerical:
- acousticness (Ranges from 0 to 1)
- danceability (Ranges from 0 to 1)
- energy (Ranges from 0 to 1)
- duration_ms (Integer typically ranging from 200k to 300k)
- instrumentalness (Ranges from 0 to 1)
- valence (Ranges from 0 to 1)
- popularity (Ranges from 0 to 100)
- tempo (Float typically ranging from 50 to 150)
- liveness (Ranges from 0 to 1)
- loudness (Float typically ranging from -60 to 0)
- speechiness (Ranges from 0 to 1)
- year (Ranges from 1921 to 2020)

  Dummy:
- mode (0 = Minor, 1 = Major)
- explicit (0 = No explicit content, 1 = Explicit content)

  Categorical:
- key (All keys on octave encoded as values ranging from 0 to 11, starting on C as 0, C# as 1 and so on...)
- artists (List of artists mentioned)
- release_date (Date of release mostly in yyyy-mm-dd format, however precision of date may vary)
- name (Name of the song)

Fig 1: Attribute Information

Dataset source: https://www.kaggle.com/datasets/ektanegi/spotifydata-19212020

Looking at the dataset information, it is clear the data has been recorded based on every year's top 100 songs based on the popularity. The dataset to be specific has 169909 observations and 19 attributes out of which 3 are categorical and 13 numerical variables with release and artist information variables. The attribute 'popularity' is chosen to be the independent variable and all the other variables which defines the genre and acoustic information are defined to be the dependent variables.

The analysis of this dataset helps to derive conclusions on popularity level of songs which will be published in future based on the various aspects of songs which has been released over the year. This helps in understanding the trend of analyzing the reach of a song to the customer community.

APP LAYOUT:

The app is designed in such a way the user can access from the very basic information of dataset to high level analysis at one web-based application. The app is user friendly it has well named tabs which replicates the content present in each tab.

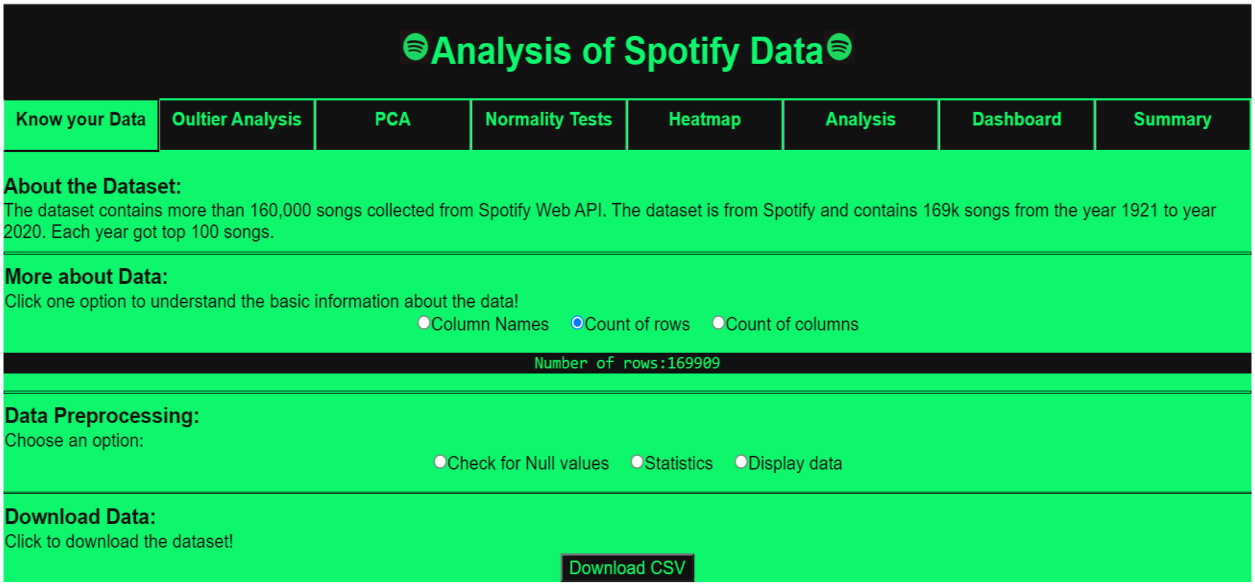


Fig2: App Layout

PREPROCESSING DATASET:

The dataset was having no missing values, hence the step of removing Nan's and missing entries was skipped. Looking at the below null analysis all the columns have zero null values

Fig3: Data preprocessing

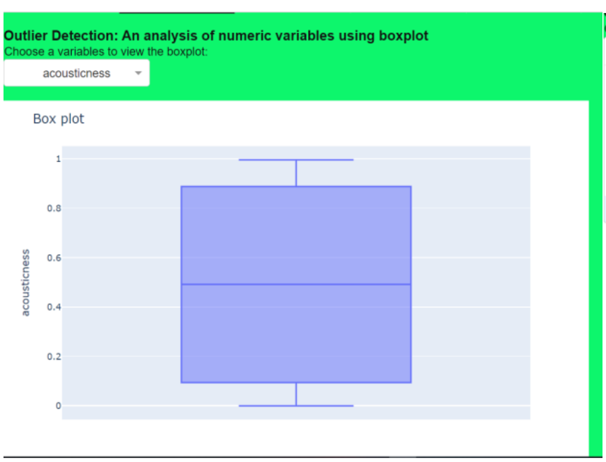
```
    acousticness                                   artists  danceability  ...  \
0          0.995                       ['Carl Woitschach']         0.708  ...
1          0.994  ['Robert Schumann', 'Vladimir Horowitz']         0.379  ...
2          0.604                   ['Seweryn Goszczyński']         0.749  ...
3          0.995                      ['Francisco Canaro']         0.781  ...
4          0.990  ['Frédéric Chopin', 'Vladimir Horowitz']         0.210  ...

              tempo  valence  year
           0  118.469   0.7790  1928
           1   83.972   0.0767  1928
           2  107.177   0.8800  1928
           3  108.003   0.7200  1928
           4   62.149   0.0693  1928

             [5 rows x 19 columns]
```

Fig4: First 5 observations of the dataset

The cleaned dataset can be downloaded to the user's local machine by clicking the 'Download CSV' button available on the end division of 'Know your Data' tab.

OUTLIER DETECTION & REMOVAL:

The second tab in the application helps in analyzing the outliers of the dataset. Outliers are data points that are far from other data points. Outliers are problematic for many statistical analyses because they can cause tests to either miss significant findings or distort real results. Hence, it is necessary to detect the outliers in the data and remove them for better analysis. This project uses Boxplots to detect the presence of outliers in the variables.



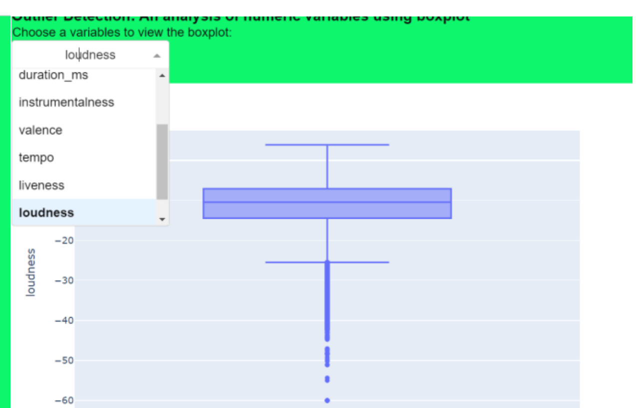


Fig5: Variable without outliers **Fig6: Variable with outliers**

'Outlier Analysis' tab allows the user to visualize boxplots for all numerical variables in the dataset. The dropdown menu in the layout helps user to select one variable at a time to visualize the graph. The Fig5 shows the box plot of variable ‘acousticness’ where there are no outlying data points after the whisker. But in Fig6 we can see outliers of the variable ‘loudness’.

Visualization of all the variables using box plot helped in finding the variables with outliers.

Outlier Removal:IQR method

The outliers from following variables were removed: danceability, duration_ms, instrumentalness, tempo, liveness, loudness, speechiness

Fig7: Variables with outliers

The outliers from these variables will be removed based on the detection of outlier range using IQR method discussed in previous chapter. The IQR analysis provided the following upper and lower bound of actual data points of the variables and any data point which doesn’t fit under the range are outliers and removed from the dataset.

```
Q1 and Q3 of the danceability is 0.42  & 0.67
 IQR for the danceability is 0.25
Any danceability < 0.04  and danceability > 1.04  is an outlier
Q1 and Q3 of the duration_ms is 171107.00  & 263000.00
 IQR for the duration_ms is 91893.00
Any duration_ms < 33267.50  and duration_ms > 400839.50  is an outlier
Q1 and Q3 of the instrumentalness is 0.00  & 0.05
 IQR for the instrumentalness is 0.05
Any instrumentalness < -0.08  and instrumentalness > 0.14  is an outlier
Q1 and Q3 of the tempo is 95.08  & 137.13
 IQR for the tempo is 42.05
Any tempo < 32.00  and tempo > 200.20  is an outlier
Q1 and Q3 of the liveness is 0.10  & 0.27
 IQR for the liveness is 0.17
Any liveness < -0.16  and liveness > 0.53  is an outlier
Q1 and Q3 of the loudness is -12.97  & -6.48
 IQR for the loudness is 6.49
Any loudness < -22.70  and loudness > 3.25  is an outlier
Q1 and Q3 of the speechiness is 0.03  & 0.08
 IQR for the speechiness is 0.04
Any speechiness < -0.03  and speechiness > 0.15  is an outlier
```

Fig8: IQR analysis values for the variables

The outliers from the variables are removed based on the IQR range provided. After the removal of outliers, we can see through the below graphs that the outlying points discussed on the previous figures have been significantly reduced.

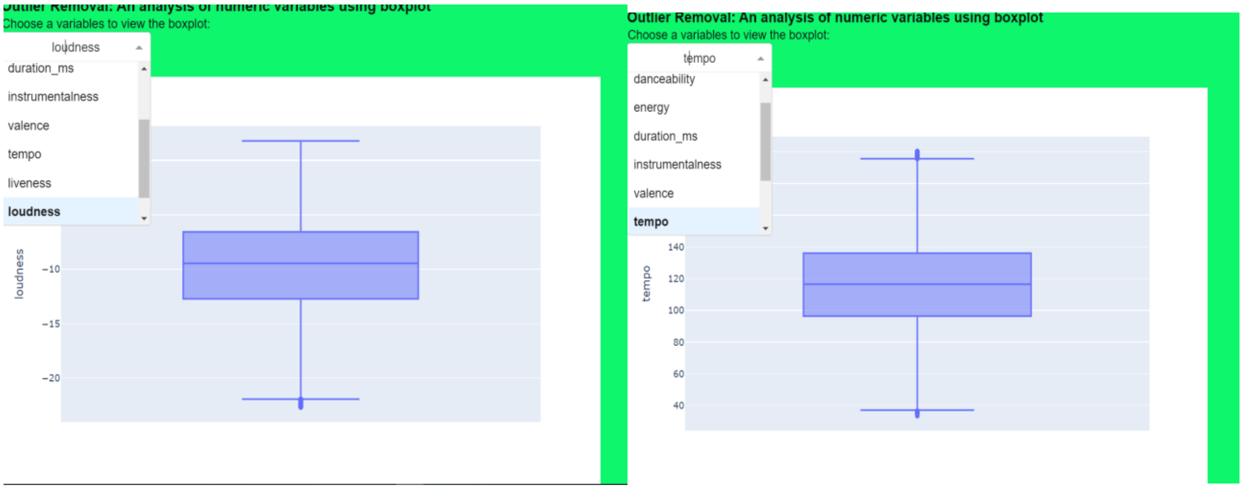


Fig9: Variables ‘loudness’ & ‘tempo’ after the removal of outliers

PRINCIPAL COMPONENT ANALYSIS (PCA):

As discussed on the previous chapter the PCA technique is helpful in reducing the feature dimensions. In other words, it helps us to find the best components/features for further analysis.

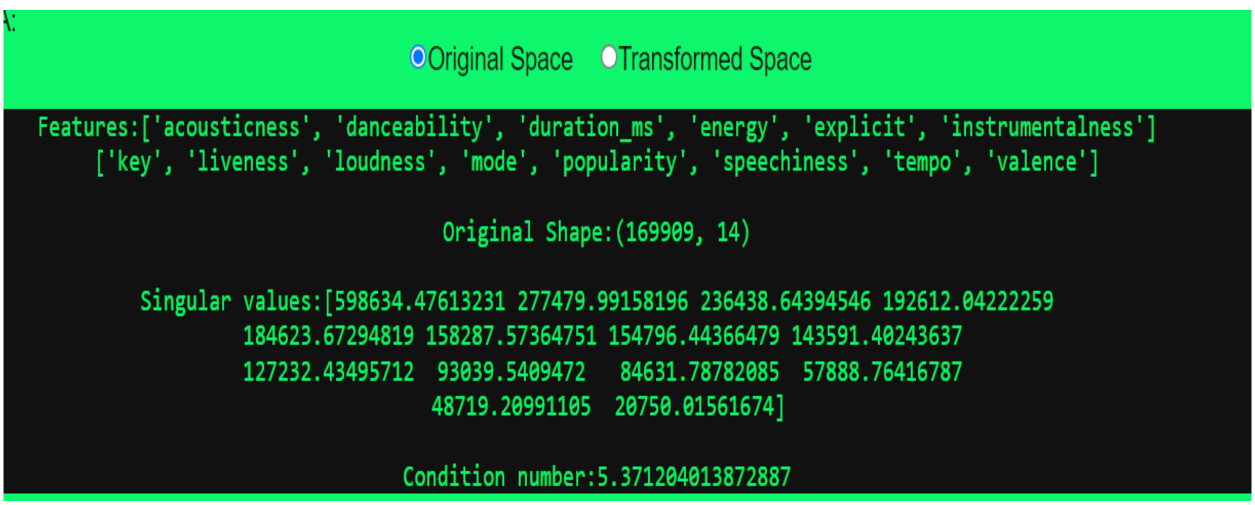


Fig10: Original Feature space

The 14 numerical columns are considered as features of the data and the original feature space has a dimension of (169909,14). The SVD analysis of the original feature space shows the condition number to be 5.37 which is not bigger number but still reducing the components through PCA would result in having much more smaller number. As we can see the singular values are also not converging to zero at the end.

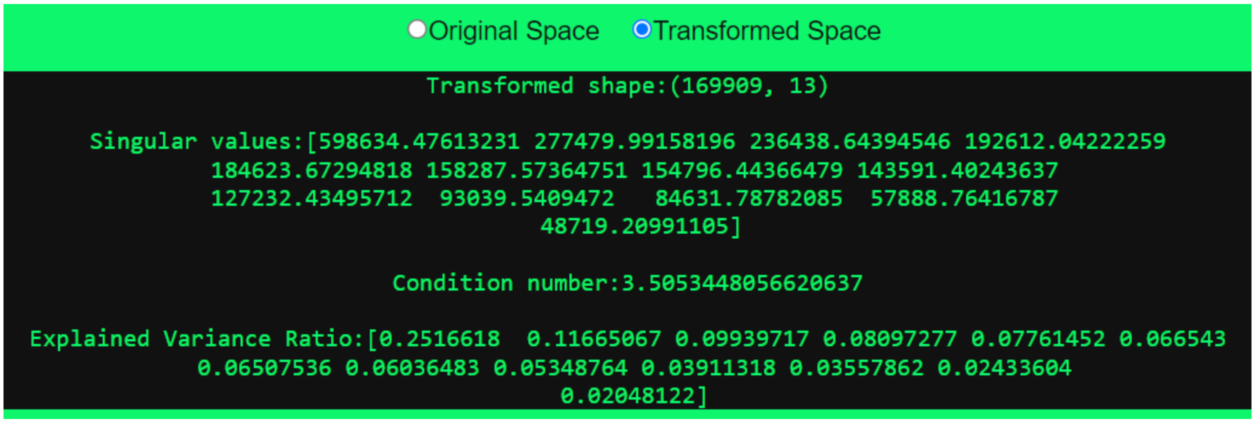


Fig11: Transformed Feature space

After performing the PCA the total components have been reduced from 14 to 13. It shows most of our features have much importance in the dataset. The explained variance ratio between original and transformed feature space has been displayed in the end of text area.

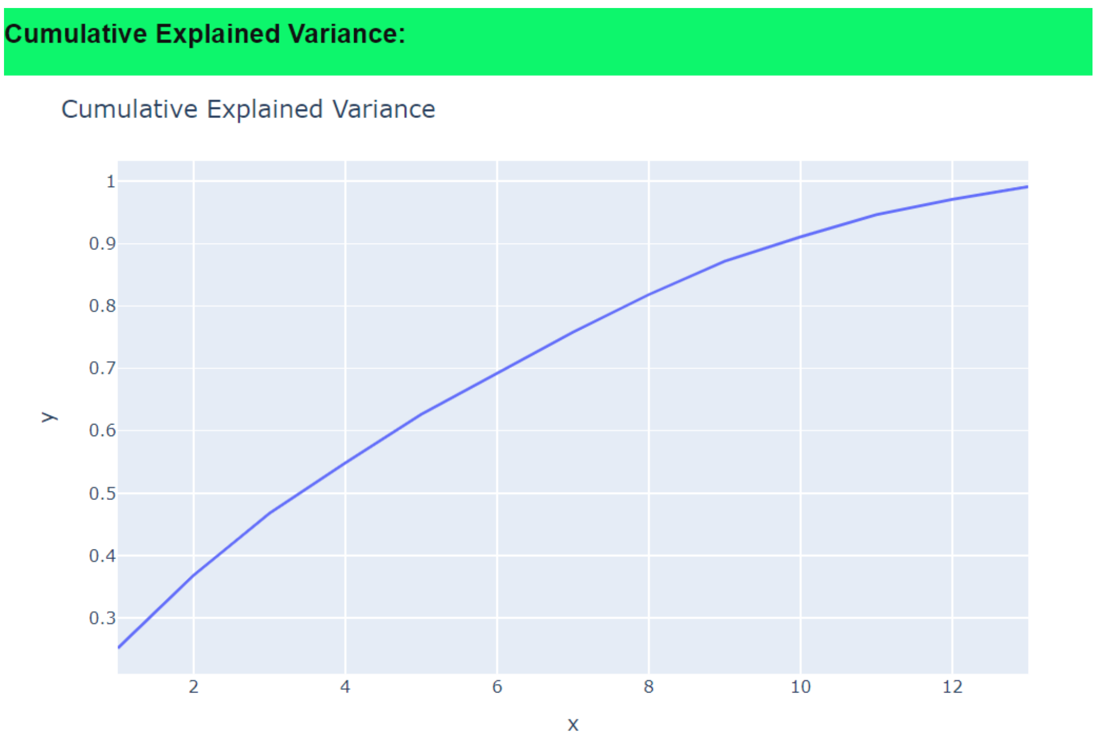


Fig12: Cumulative explained variance plot

The above graph shows the cumulative explained variance of reduced 13 PCA components. A heatmap representing the correlation between the components was generated. The graph generated on the dashboard doesn't have annotations due the deprecation in recent plotly package hence seaborn version of heatmap was produced on console.

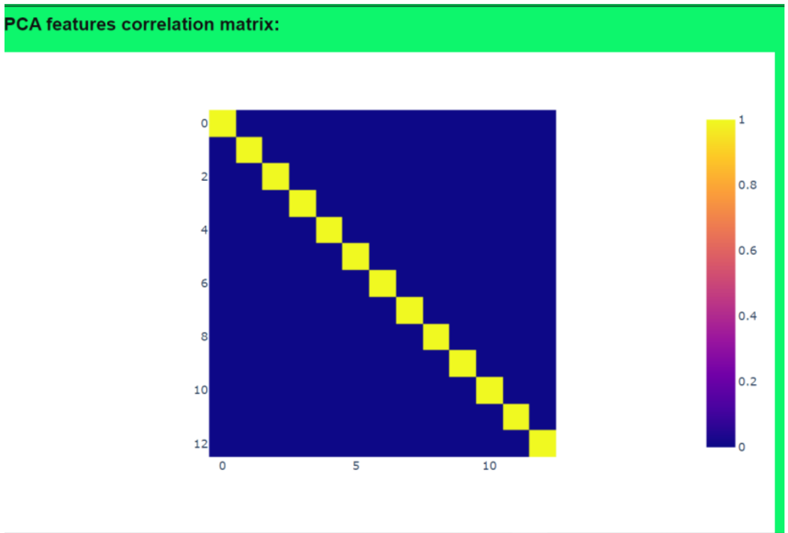


Fig13: PCA correlation matrix

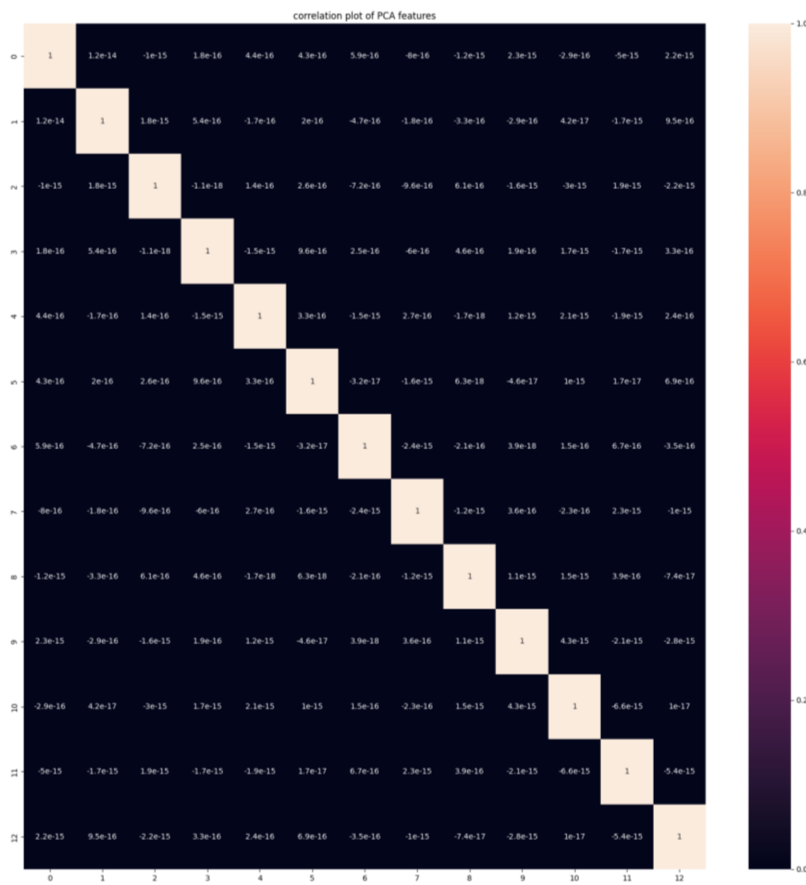


Fig14: PCA correlation matrix in console

Looking at the above heatmaps, though the graph's text values are not clear. We can see a pattern of light shade over the diagonal denoting the features with themselves have high correlation. But other part of plot has correlation color almost close to zero. It shows the features have very low collinearity amongst others.

NORMALITY TESTS:

The normality test tabs have the flexibility to select any numerical variable from the dataset and run the one amongst three normality tests discussed in the previous chapter.

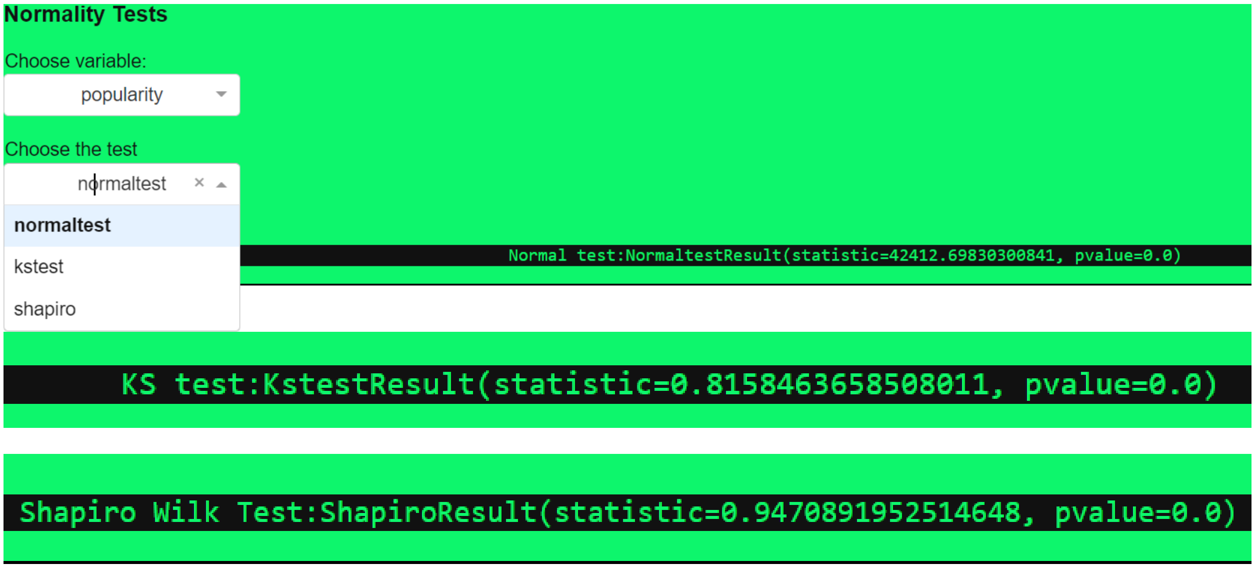


Fig15: Normality tests

Looking at the above figure, it is clear that the variable 'popularity' doesn't come from a normal distribution. Same way visualizing the other variable the p-value of all the test statistics were zero which shows the data doesn't come from normal distribution. The screenshots of other variables weren't included considering the redundancy and length of the report. The other features can be tested while using the application.

HEATMAP & PEARSON CORRELATION COEFFICIENT MATRIX:

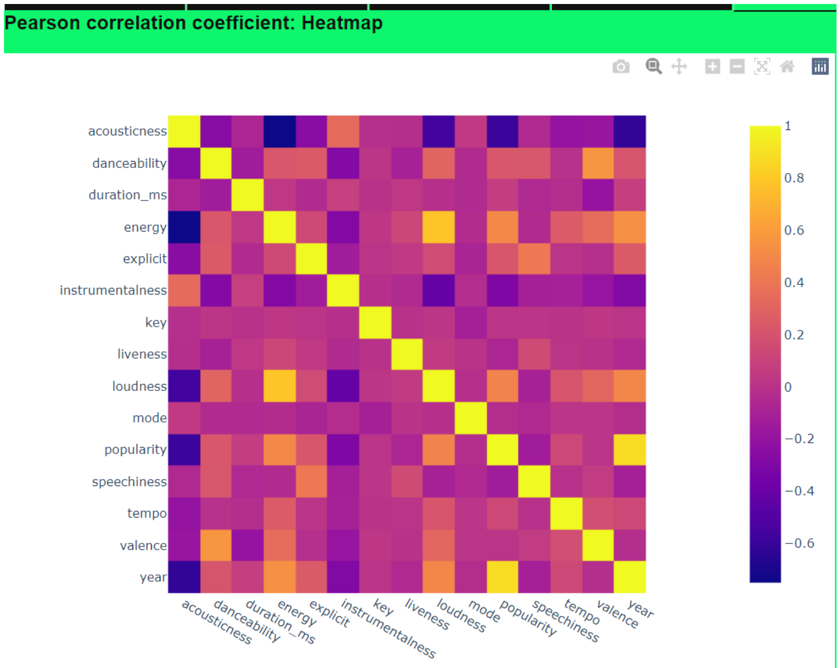


Fig16: Heatmap

The generated heatmap provides the correlation amongst the different variables of the dataset. The cleaned dataset is used to produce the correlation of all the data variables and then fed into the plot for heatmap. The given figure displays a checker board pattern denoting there is some amount of collinearity in the dataset.

STATISTICS:

Check for Null values   Statistics   Display data

	acousticness	danceability	duration_ms	energy
count	169909.000000	169909.000000	1.699090e+05	169909.000000
mean	0.493214	0.538150	2.314062e+05	0.488593
std	0.376627	0.175346	1.213219e+05	0.267390
min	0.000000	0.000000	5.108000e+03	0.000000
25%	0.094500	0.417000	1.710400e+05	0.263000
50%	0.492000	0.548000	2.086000e+05	0.481000
75%	0.888000	0.667000	2.629600e+05	0.710000
max	0.996000	0.988000	5.403500e+06	1.000000

	explicit	instrumentalness	key	liveness
count	169909.000000	169909.000000	169909.000000	169909.000000
mean	0.084863	0.161937	5.200519	0.206690
std	0.278679	0.309329	3.515257	0.176796
min	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	2.000000	0.098400
50%	0.000000	0.000204	5.000000	0.135000
75%	0.000000	0.086800	8.000000	0.263000
max	1.000000	1.000000	11.000000	1.000000

	loudness	mode	popularity	speechiness
count	169909.000000	169909.000000	169909.000000	169909.000000
mean	-11.370289	0.708556	31.556610	0.094058
std	5.666765	0.454429	21.582614	0.149937
min	-60.000000	0.000000	0.000000	0.000000
25%	-14.470000	0.000000	12.000000	0.034900
50%	-10.474000	1.000000	33.000000	0.045000
75%	-7.118000	1.000000	48.000000	0.075400
max	3.855000	1.000000	100.000000	0.969000

	tempo	valence	year
count	169909.000000	169909.000000	169909.000000
mean	116.948017	0.532095	1977.223231
std	30.726937	0.262408	25.593168
min	0.000000	0.000000	1921.000000
25%	93.516000	0.322000	1957.000000
50%	114.778000	0.544000	1978.000000
75%	135.712000	0.749000	1999.000000
max	244.091000	1.000000	2020.000000

Fig17: Descriptive Statistics

The above figure shows the total count of values in each column of the dataset. The mean, median, minimum and maximum values of the most columns are values less than or equal to zero. This is because most of the variables in the dataset has value range between 0 to 1 since they represent the aspect of song. For example, value 0.1 in liveness depicts there is less live audio clips in a song whereas the value 1 represents song to be having a highly vibrant live acoustics.

DATA VISUALIZATION:

The ‘Analysis’ tab helps in visualizing the dataset using different plots. Since most of the variables in the dataset are numerical there was limitations in applying more different varieties of plots for visualization.

Line plot:

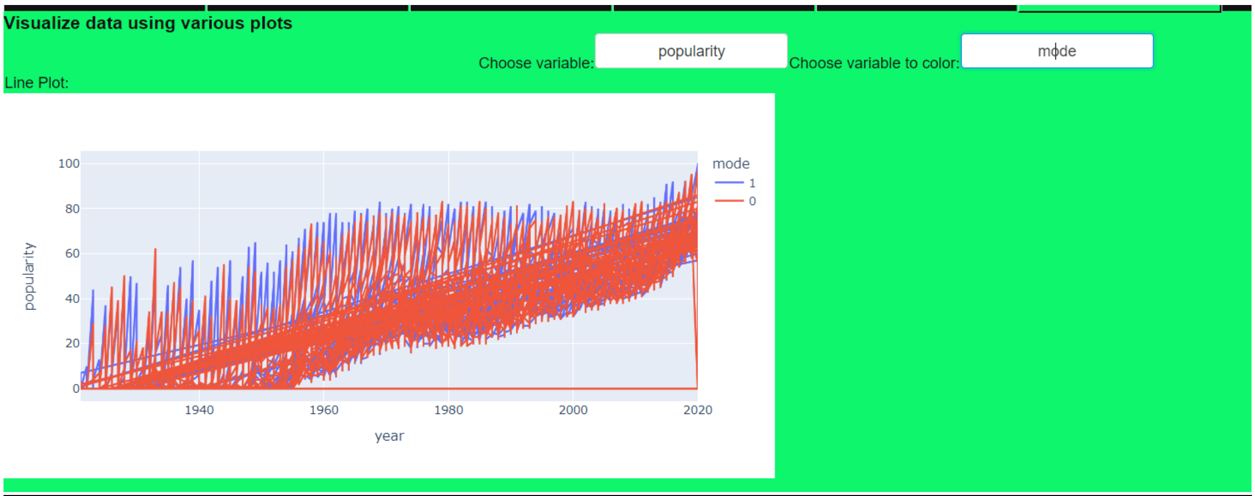


Fig18: Line plot

This layout helps the user to select a variable (numeric) of their choice to visualize the trend of variable over the period of time. This plot uses ‘year’ in x-axis and the y-axis variable is decided by the user’s choice. The hue for the plot can be selected based on the three categorical columns of the dataset.

Histograms:

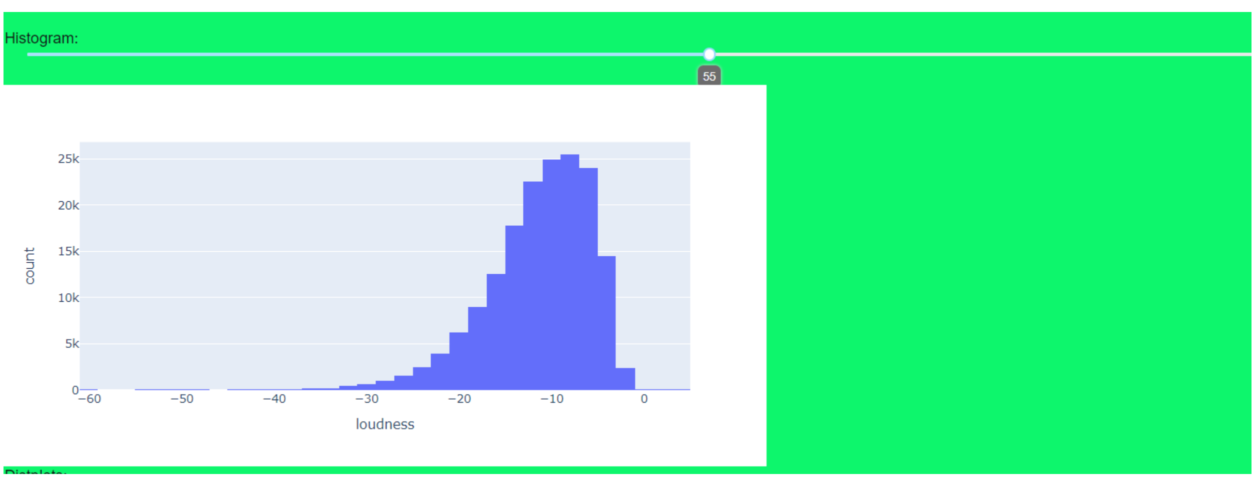


Fig19: Histograms

The number of bins for the histogram can be selected using the slider option available in the layout. The slider has a tooltip which denotes the bins value highlighted for the user's view. The histogram uses the same variable user selected for the line plot in above division.

<u>Distplots:</u>

The distplots helps in understanding a variable distribution using different types of graph combinations. The distplot can be composed of all or any combination of the following 3 components: (1) histogram, (2) curve: (a) kernel density estimation or (b) normal curve, and (3) rug plot. Additionally, multiple distplots (from multiple datasets) can be created in the same plot.

The application provides a histogram distribution categorized based on the hue and y-axis variable selected in the layout. And it also has option for the user to select a distribution option to visualize the category.

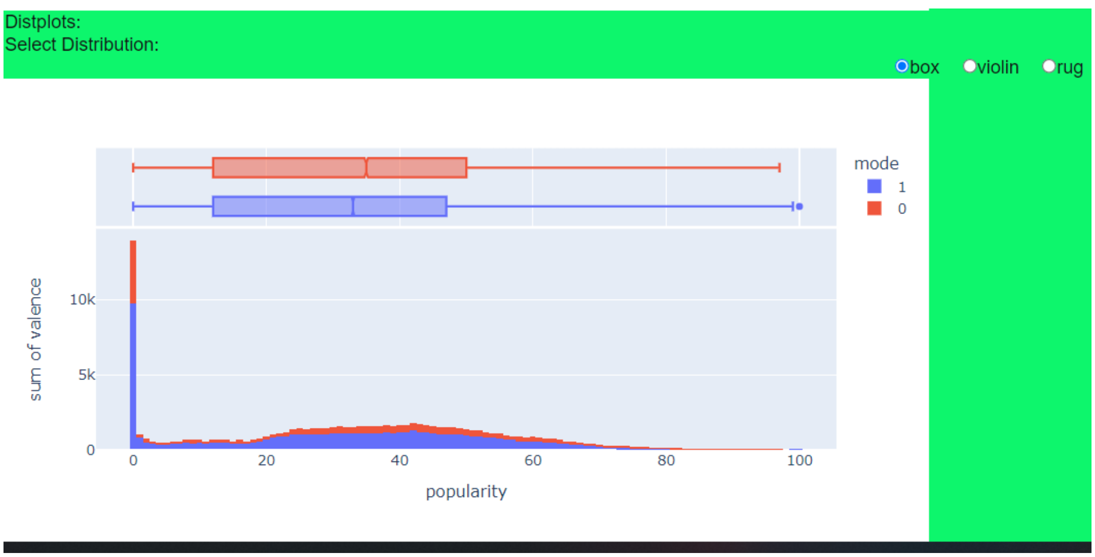


Fig20: Distplot with box distribution

The above distplot shows the histograms of 'popularity' in the major axis categorized based on 'mode' of the song. The option of box distribution has selected to visualize the mode categories in box plot.

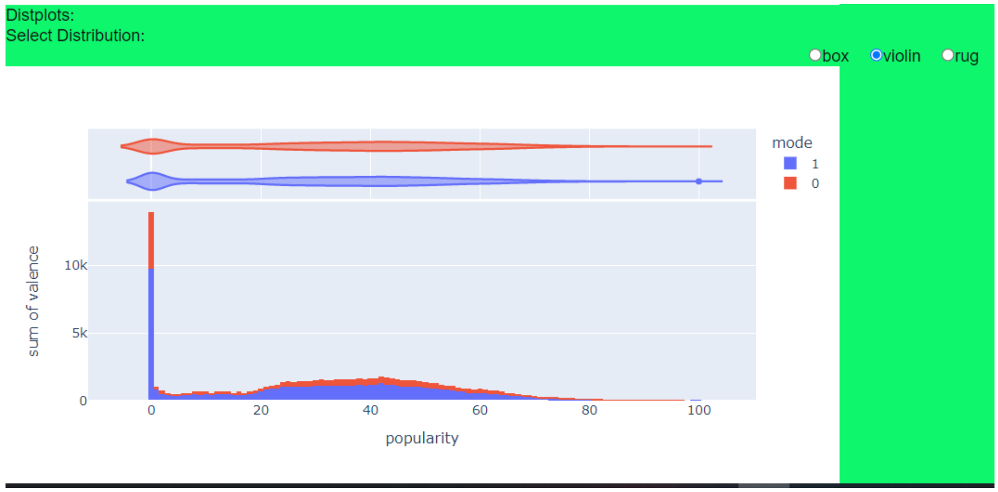


Fig21: Distplot with violin distribution

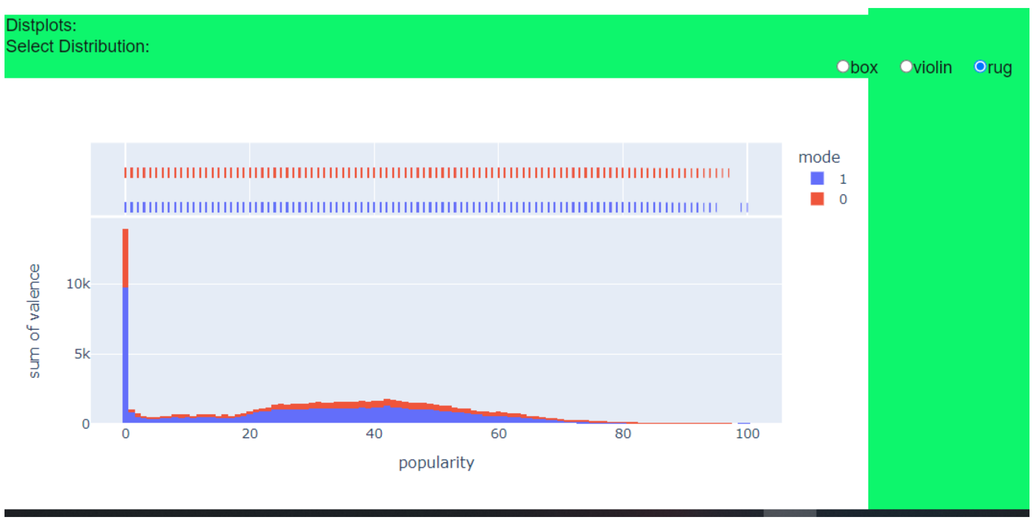


Fig22: Distplot with rug distribution

DASHBOARD:

The 'Dashboard' tab has combination of different plots in one window to understand the analysis of dependent variable 'popularity' better. The following figure shows the final dashboard created to deliver the user an overall story about the dataset.

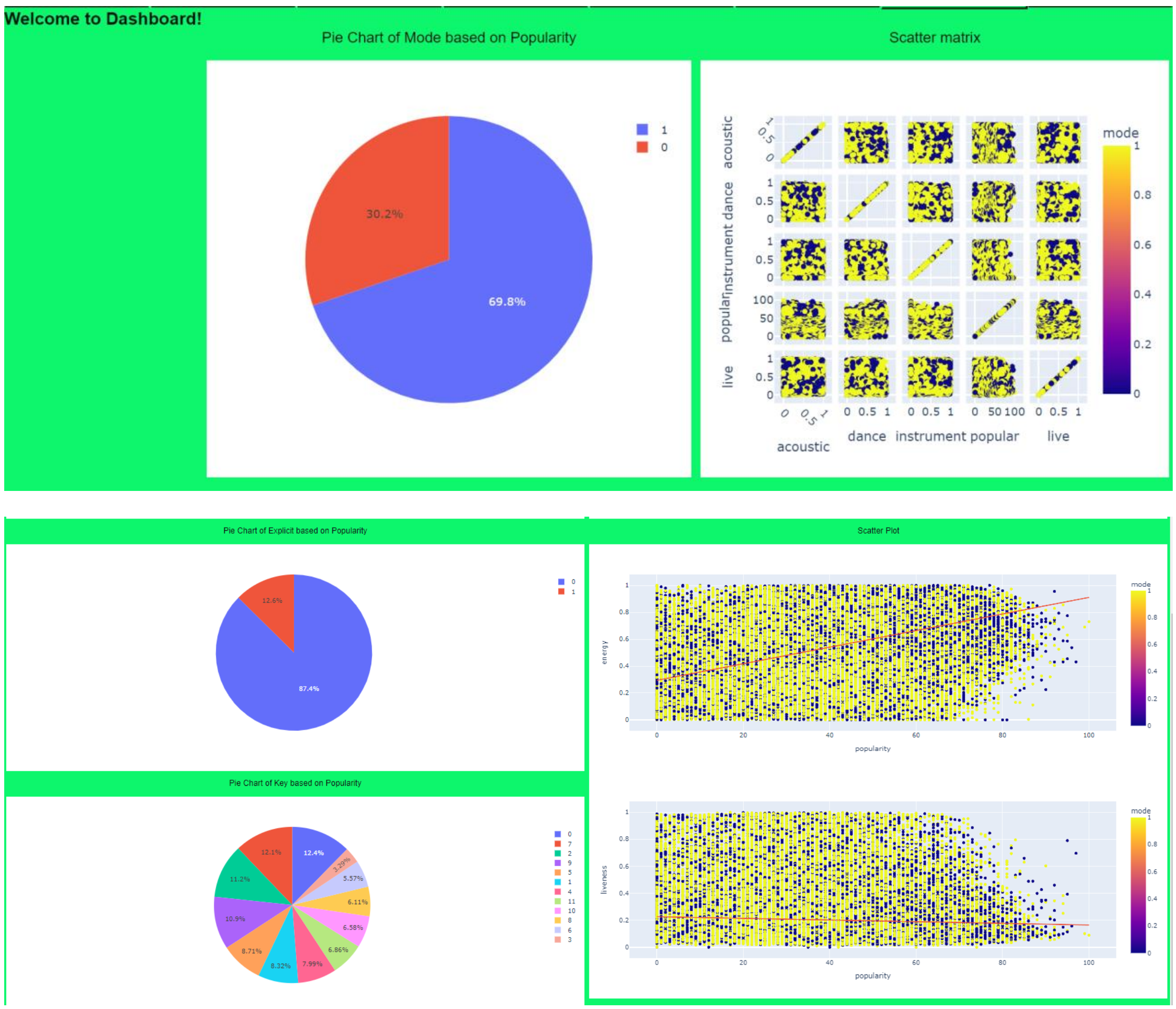


Fig23: Dashboard

The pie charts for representing the categorical columns with popularity values were created. Looking at the pie charts, mode '1' has more percentage of values in 'popularity' than '0'. While with respect to explicit the value of '0' is more in percent than the value of '1'.

The scatter matrix with hue of mode shows the correlation between variables: liveness, popularity, instrumentalness, acousticness, danceability. The plot is clumsier than expected since the dataset has more data points and they are widely spread over the same range of 0 to1. The other variables were not included due to the time taken dash app to load the layout and the plot was becoming messier than the existing one.

Last two subplots denote the scatter plot analysis using a regression line. The x-axis of the scatter plot is 'popularity' our dependent variable. The y-axis is 'liveness' and 'energy' with hue of 'mode'. The other variables were not included because same graph loading issue. The two subplots show how regression line differs. The scatter plot with energy, the line is increasing as it depicts the popularity of the song increases with increase in energy value. While the liveness increases the popularity of the song is decreasing which shows this variable doesn't support much in incrementing the popularity of the song.

<u>Dash Core Components:</u>

The following set of dash components were used in the development of the given project.

- Graph
- Tabs
- Multiple Divisions
- Range slider
- Radioitems
- Download component
- TextArea
- Dropdown
- Output

The following plots were used in the successful development of the project.

- Line plot
- Histograms
- Violin distribution
- Rug distribution
- Pie charts
- Scatter matrix
- Scatter plot with regression line
- Heatmaps
- Box plots

RECOMMENDATIONS:

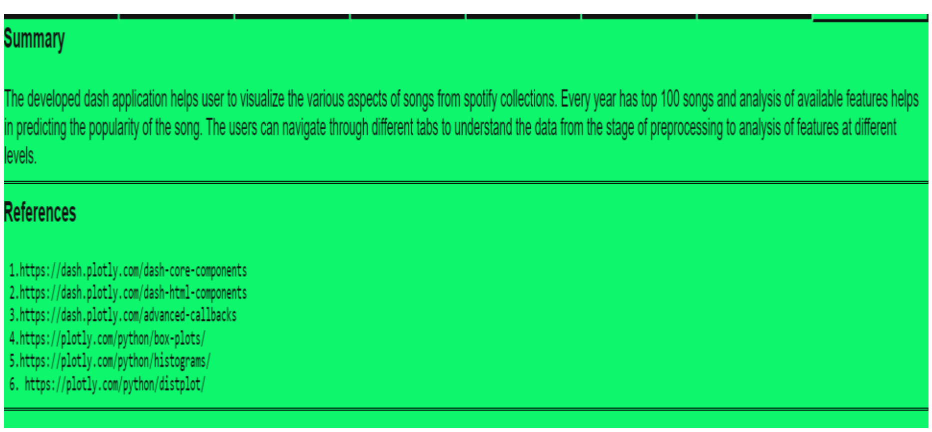


Fig24: Summary tab

The summary tab has overall idea about the analysis. The analysis of the Spotify data helped in understanding what song aspects are necessary to predict the popularity of the song. The dataset was not normal and had more outliers. After removing the outliers we were able to understand the data in a better way. The plots explained that the popularity of the songs in initial decades were less despite of having good liveness and energy. But over the period of time the songs with more energy and tempo had been popular than the songs with more liveness and instrumentalness. It shows the user's choice of song has always differed based on the decade they were living.

The app has been designed in a way where the user can find the very basic statistics of the dataset to advanced dashboard analysis. The application has been designed in a way that the user can navigate through the tabs easily due to the well named tabs. The layouts under each tab have been designed in a way that the user can interact by selecting the variables of their choice and type of statistical and test information.

The summary section has author and dataset information in order to receive feedbacks from users further improvement.

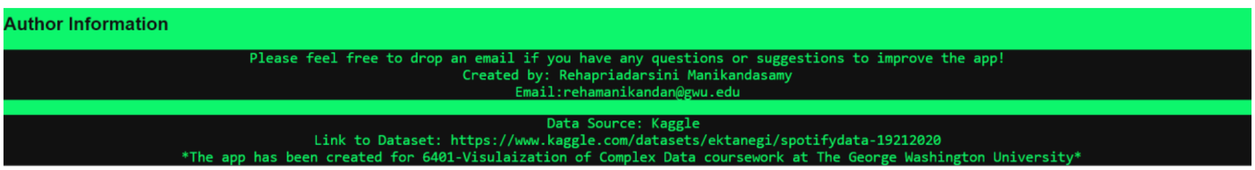


Fig25: Author and dataset information

CONCLUSION

The project focused on analyzing the Spotify data to find the popularity of the song based on the various aspects of the song. The data didn't require but it had more outliers to be removed. The cleaned dataset was useful in visualizing the data variables in such a way more information about the feature variables and their influence on finding the popularity of the song. The scatter plot analysis helped in understanding which variables increasing/decreasing property increased/ decreased the popularity of the song.

The GCP implementation of the app would be better version of publishing the application online. Attempt of adding to Google cloud was continued for two days but due to service unavailability and billing issues the app didn't get published. As the link https://dashapp-qfzuow47ha-ue.a.run.app/ , shows service unavailable due to the issue in the billing account. The future scope would involve finding an optimized way to publish the app on GCP.

APPENDIX

Steps to run the project file:

Run the Project_RM.py file to implement the project end-to-end.

Project_RM.py:

```
import dash
import dash
import numpy as np
import plotly.express as px
from dash import dcc
from dash.dependencies import Input,Output
from dash import html
import pandas as pd
from statsmodels.graphics.gofplots import qqplot
from sklearn.preprocessing import StandardScaler
from numpy import linalg as la
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import kstest
from scipy.stats import shapiro
from scipy.stats import normaltest
import plotly.graph_objects as go

#Load Dataset
pd.set_option('max_columns', 19)
data=pd.read_csv('data.csv')

#Load external stylesheets and assign tab styles
external_stylesheets=["https://unpkg.com/purecss@2.1.0/build/pure-min.css" ]
my_app=dash.Dash('My App',external_stylesheets=external_stylesheets)
tabs_styles = {
    'height': '45px'
}
tab_style = {
    'borderBottom': '2px solid #0DF66C',
    'borderTop': '2px solid #0DF66C',
    'borderRight': '2px solid #0DF66C',
    'fontWeight': 'bold',
    'backgroundColor': '#111111',
    'textAlign': 'center',
    'color':'#0DF66C',
    'padding': '6px'

}

tab_selected_style = {
    'borderTop': '2px solid #111111',
    'borderBottom': '2px solid #111111',
```

```
    'borderRight':'2px solid #111111',
    'padding': '6px',
    'fontWeight': 'bold',
    'backgroundColor': '#0DF66C',
    'textAlign':'center',
    'color': '#111111'
}

#Design the tab layout
my_app.layout=html.Div(style={'backgroundColor':'#111111',
                              'textAlign':'center','color':'#0DF66C'},

children=[html.Img(src='https://wallpaperaccess.com/full/1373294.jpg',
style={'display':'inline-block','height':'2.5%', 'width':'2.5%'}),
                                 html.H1(' Analysis of Spotify Data
',style={'display':'inline-block'}),

html.Img(src='https://wallpaperaccess.com/full/1373294.jpg',
style={'display':'inline-block','height':'2.5%', 'width':'2.5%'}),
                                 dcc.Tabs(id='tabs1',children=[
                                     dcc.Tab(label='Know your
Data',value='Know your Data',style=tab_style,
selected_style=tab_selected_style),
                                     dcc.Tab(label='Oultier
Analysis',value='Outlier Analysis',style=tab_style,
selected_style=tab_selected_style),

dcc.Tab(label='PCA',value='PCA',style=tab_style,
selected_style=tab_selected_style),
                                     dcc.Tab(label='Normality
Tests',value='Normality Tests',style=tab_style,
selected_style=tab_selected_style),

dcc.Tab(label='Heatmap',value='Heatmap',style=tab_style,
selected_style=tab_selected_style),

#dcc.Tab(label='Statistics',value='Statistics',style=tab_style,
selected_style=tab_selected_style),

dcc.Tab(label='Analysis',value='Analysis',style=tab_style,
selected_style=tab_selected_style),

dcc.Tab(label='Dashboard',value='Dashboard',style=tab_style,
selected_style=tab_selected_style),

dcc.Tab(label='Summary',value='Summary',style=tab_style,
selected_style=tab_selected_style)],style=tabs_styles,
                                          value='Know your Data'),

html.Div(id='layout',style={'backgroundColor':'#0DF66C','color':'#111111'})

                                 ])

#Layput for first tab
tab1_layout=html.Div(style={'backgroundColor':'#0DF66C','color':'#111111'},
                     children=[html.Br()
```

```
                          ,html.H3('About the Dataset:',style={'margin':
'0','textAlign':'left'
}),
                                html.P('The dataset contains more than 160,000
songs collected from Spotify Web API. The dataset is from Spotify and
contains 169k songs from the year 1921 to year 2020. Each year got top 100
songs.',style={'display':'inline-block','margin': '0','textAlign':'left'}),
                                html.Hr(style={'border': '1px solid black'}),
                                html.H3('More about Data:',style={'margin':
'1px','textAlign':'left'}),
                                html.P('Click one option to understand the
basic information about the data!',style={'margin':
'1px','textAlign':'left'}),
                                dcc.RadioItems(id='infos',options=[
                                    {'label':'Column Names','value':'Column'},
                                    {'label':'Count of rows','value':'rows'},
                                    {'label':'Count of
columns','value':'columns'
                                }],
                                value='Column',inputStyle={"margin-left":
"20px"}),
                                html.Plaintext(id='datainfo',style =
{'backgroundColor':'#111111','color':'#0DF66C','font-size':'15px'}),
                                html.Hr(style={'border': '1px solid black'}),
                                html.H3('Data Preprocessing:',style={'margin':
'1px','textAlign':'left'}),
                                html.P('Choose an option:',style={'margin':
'1px','textAlign':'left'}),
                                dcc.RadioItems(id='cleans', options=[
                                    {'label': 'Check for Null values',
'value': 'nulls'},

                                    {'label': 'Statistics', 'value': 'stats'},
                                    {'label':'Display data','value':'head'
                                     }],value='Column',inputStyle={"margin-
left": "20px"}),
                                html.Plaintext(id='preprocess',style =
{'backgroundColor':'#111111','color':'#0DF66C','font-size':'15px'}),
                                html.Hr(style={'border': '1px solid black'}),
                                html.H3('Download Data:', style={'margin':
'1px', 'textAlign': 'left'}),
                                html.P('Click to download the dataset!',
style={'margin': '1px', 'textAlign': 'left'}),
                                html.Button("Download CSV",
id="btn_csv",style={'background-color':'#111111','color':'#0DF66C'}),
                                dcc.Download(id="download-dataframe-csv")

])

#Outlier detection and removal
data1 = data.copy()
cols_out =
['danceability','duration_ms','instrumentalness','tempo','liveness','loudness
','speechiness']
for i in cols_out:

    q1_h, q2_h, q3_h = data1[i].quantile([0.25, 0.5, 0.75])
```

```
    IQR_h = q3_h - q1_h
    lower1 = q1_h - 1.5 * IQR_h
    upper1 = q3_h + 1.5 * IQR_h
    data1 = data1[(data1[i] > lower1) & (data1[i] < upper1)]
    print(f'Q1 and Q3 of the {i} is {q1_h:.2f}  & {q3_h:.2f} \n IQR for the
{i} is {IQR_h:.2f} \nAny {i} < {lower1:.2f}  and {i} > {upper1:.2f}  is an
outlier')


#Design for second tab layout
tab2_layout=html.Div(style={'backgroundColor':'#0DF66C','color':'#111111'},
                     children=[html.Br(),
                        html.H3('Outlier Detection: An analysis of numeric
variables using boxplot',style={'margin': '0','textAlign':'left'}),
                        html.P('Choose a variables to view the
boxplot:',style={'margin': '1px','textAlign':'left'}),
                        dcc.Dropdown(id='drop1',
                                            options=[
                                                {'label': 'acousticness',
'value': 'acousticness'},
                                                {'label': 'danceability',
'value': 'danceability'},
                                                {'label': 'energy', 'value':
'energy'},
                                                {'label': 'duration_ms',
'value': 'duration_ms'},
                                                {'label': 'instrumentalness',
'value': 'instrumentalness'},
                                                {'label': 'valence', 'value':
'valence'},
                                                {'label': 'tempo', 'value':
'tempo'},
                                                {'label': 'liveness',
'value': 'liveness'},
                                                {'label': 'loudness',
'value': 'loudness'},
                                                {'label': 'speechiness',
'value': 'speechiness'},

                                            ], value='acousticness',
clearable=False,style={'width': '200px'}),
                               html.Br(),

dcc.Graph(id='graphbox1',style={'width':'800px','height':'500px'}),
                               html.Hr(style={'border': '1px solid black'}),
                               html.H3('Outlier Removal:IQR method',
style={'margin': '1px', 'textAlign': 'left'}),
                               html.P('The outliers from following variables
were removed: danceability, duration_ms, instrumentalness, tempo, liveness,
loudness, speechiness',style =
{'backgroundColor':'#111111','color':'#0DF66C','font-size':'15px'}),
                               html.Hr(style={'border': '1px solid black'}),
                        html.H3('Outlier Removal: An analysis of numeric
variables using boxplot',style={'margin': '0','textAlign':'left'}),
                        html.P('Choose a variables to view the
boxplot:',style={'margin': '1px','textAlign':'left'}),
```

```
                        dcc.Dropdown(id='drop2',
                                            options=[
                                                {'label': 'acousticness',
'value': 'acousticness'},
                                                {'label': 'danceability',
'value': 'danceability'},
                                                {'label': 'energy', 'value':
'energy'},
                                                {'label': 'duration_ms',
'value': 'duration_ms'},
                                                {'label': 'instrumentalness',
'value': 'instrumentalness'},
                                                {'label': 'valence', 'value':
'valence'},
                                                {'label': 'tempo', 'value':
'tempo'},
                                                {'label': 'liveness',
'value': 'liveness'},
                                                {'label': 'loudness',
'value': 'loudness'},
                                                {'label': 'speechiness',
'value': 'speechiness'},

                                            ], value='acousticness',
clearable=False,style={'width': '200px'}),
                               html.Br(),

dcc.Graph(id='graphbox2',style={'width':'800px','height':'500px'}),
])


#PCA
Features=data._get_numeric_data().columns.to_list()[:-1]
x=data[data._get_numeric_data().columns.to_list()[:-1]]

x=x.values
x=StandardScaler().fit_transform(x)

pca=PCA(n_components='mle',svd_solver='full')
pca.fit(x)
x_pca=pca.transform(x)

#plot of cumsum
number_of_components=np.arange(1,len(np.cumsum(pca.explained_variance_ratio_)
)+1)
fig=px.line(x=number_of_components,y=np.cumsum(pca.explained_variance_ratio_)
)
fig.update_layout(title='Cumulative Explained Variance')

#svd and condition number
H=np.matmul(x.T,x)
_,d,_=np.linalg.svd(H)


#svd and condition number-tranformed
H_pca=np.matmul(x_pca.T,x_pca)
_,d_pca,_=np.linalg.svd(H_pca)
```

```
#PCA correlation matrix
fig1=px.imshow(pd.DataFrame(x_pca).corr())
#Better visuals
plt.figure(figsize=(20,20))
sns.heatmap(pd.DataFrame(x_pca).corr(), annot=True)
plt.title('correlation plot of PCA features')
plt.show()

#Design for third tab
tab3_layout=html.Div(style={'backgroundColor':'#0DF66C','color':'#111111'},
                     children=[html.Br(),
                               html.H3('Principal Component Analysis',
                                       style={'margin': '0', 'textAlign':
'left'}),
                               html.P('Choose options to view outputs of
PCA:',
                                      style={'margin': '1px', 'textAlign':
'left'}),
                               dcc.RadioItems(id='checkpca',options=[
                                   {'label':'Original
Space','value':'Original'},
                                   {'label':'Transformed
Space','value':'tranformed'}],value='Original',inputStyle={"margin-left":
"20px"}),
                               html.Plaintext(id='pcaout',style =
{'backgroundColor':'#111111','color':'#0DF66C','font-size':'15px'}),
                               html.Hr(style={'border': '1px solid black'}),
                               html.H3('Cumulative Explained Variance:',
                                       style={'margin': '0', 'textAlign':
'left'}),
                               html.Br(),

dcc.Graph(figure=fig,style={'width':'800px','height':'500px'}),
                               html.Hr(style={'border': '1px solid black'}),
                               html.H3('PCA features correlation matrix:',
                                       style={'margin': '0', 'textAlign':
'left'}),
                               html.Br(),
                               dcc.Graph(figure=fig1, style={'width':
'800px', 'height': '500px'})
                               ])

#Design for tab4
tab4_layout=html.Div(style={'backgroundColor':'#0DF66C','color':'#111111'},
                     children=[html.H3('Normality Tests',style={'margin':
'0', 'textAlign': 'left'}),
                               html.Br(),
                        html.P('Choose variable:',style={'margin': '1px',
'textAlign': 'left'}),
                        dcc.Dropdown(id='dropvar',
                            options = [
                                          {'label': 'acousticness', 'value':
'acousticness'},
                                          {'label': 'danceability', 'value':
'danceability'},
```

```
                                        {'label': 'energy', 'value':
'energy'},
                                        {'label': 'duration_ms', 'value':
'duration_ms'},
                                        {'label': 'instrumentalness',
'value': 'instrumentalness'},
                                        {'label': 'valence', 'value':
'valence'},
                                        {'label': 'tempo', 'value':
'tempo'},
                                        {'label': 'liveness', 'value':
'liveness'},
                                        {'label': 'loudness', 'value':
'loudness'},
                                        {'label': 'speechiness', 'value':
'speechiness'},
                                        {'label': 'popularity', 'value':
'popularity'},

                                    ], value =
'acousticness',style={'width': '200px'},clearable=False),
                        html.Br(),
                        html.P('Choose the test',style={'margin': '1px',
'textAlign': 'left'}),
                        dcc.Dropdown(id='droptest',options=[
                            {'label':'normaltest','value':'normaltest'},
                            {'label': 'kstest', 'value': 'kstest'},
                            {'label': 'shapiro', 'value': 'shapiro'}
                        ],value='normaltest',style={'width': '200px'}),
                               html.Br(),
                        html.Plaintext(id='ntout',style =
{'backgroundColor':'#111111','color':'#0DF66C','font-size':'15px'}),
                        html.Hr(style={'border': '1px solid black'}),

])

#Correlation coefficient matrices
fig2=px.imshow(data.corr())

#Design tab5 layout
tab5_layout=html.Div(style={'backgroundColor':'#0DF66C','color':'#111111'},
                     children=[html.H3('Pearson correlation coefficient:
Heatmap',style={'margin': '0', 'textAlign': 'left'}),
                               html.Br(),
                               dcc.Graph(figure=fig2,style={'width': '800px',
'height': '600px'})])

#Design for tab6
tab6_layout=html.Div(style={'backgroundColor':'#0DF66C','color':'#111111'},
                     children=[html.H3('Visualize data using various
plots',style={'margin': '0', 'textAlign': 'left'}),
                               html.P('Choose variable:',style={'margin':
'1px', 'textAlign': 'left','display':'inline-block'}),
                            dcc.Dropdown(id='dropline',
                            options = [
                                        {'label': 'acousticness', 'value':
'acousticness'},
```

```
                                             {'label': 'danceability', 'value': 
'danceability'},
                                             {'label': 'energy', 'value': 
'energy'},
                                             {'label': 'duration_ms', 'value': 
'duration_ms'},
                                             {'label': 'instrumentalness', 
'value': 'instrumentalness'},
                                             {'label': 'valence', 'value': 
'valence'},
                                             {'label': 'tempo', 'value': 
'tempo'},
                                             {'label': 'liveness', 'value': 
'liveness'},
                                             {'label': 'loudness', 'value': 
'loudness'},
                                             {'label': 'speechiness', 'value': 
'speechiness'},
                                             {'label': 'popularity', 'value': 
'popularity'},

                                         ], value = 
'acousticness',style={'width': '200px','display':'inline-
block'},clearable=False),
                        html.P('Choose variable to 
color:',style={'display':'inline-block','margin': '1px', 'textAlign': 
'left'}),
                        dcc.Dropdown(id='dropcolor',
                            options = [
                                             {'label': 'mode', 'value': 'mode'},
                                             {'label': 'explicit', 'value': 
'explicit'},
                                             {'label': 'key', 'value': 'key'},

                                         ], value = 
'mode',style={'display':'inline-block','width': '200px'},clearable=False),
                        html.Br(),

                        html.P('Line Plot:',style={'margin': '1px', 
'textAlign': 'left'}),
                        dcc.Graph(id='line', style={'width': '800px', 
'height': '400px'}),
                        html.Br(),
                        html.P('Histogram:',style={'margin': '1px', 
'textAlign': 'left'}),

dcc.Slider(id='bins',min=20,max=100,value=50,tooltip={"placement": "bottom", 
"always_visible": True}),
                        dcc.Graph(id='bar', style={'width': '800px', 
'height': '400px'}),
                        html.P('Distplots:',style={'margin': '1px', 
'textAlign': 'left'}),
                        html.P("Select Distribution:",style={'margin': '1px', 
'textAlign': 'left'}),
                        dcc.RadioItems(
                        id='distribution',
                        options=[
```

```
                                {'label':'box','value':'box'},
                               {'label':'violin','value':'violin'},
                        {'label':'rug','value':'rug'}
                    ],

                    value='box',inputStyle={"margin-left": "20px"}),
                    dcc.Graph(id="graphd",style={'width': '800px',
'height': '400px'}),
                    ])

#Design for Tab7

#First column
figmode=px.pie(data,names='mode',values='popularity')
figexp=px.pie(data,names='explicit',values='popularity')
figkey=px.pie(data,names='key',values='popularity')

#Second column
f=['acousticness','danceability','instrumentalness','popularity','liveness']
figscatter=px.scatter_matrix(data,
                      dimensions=f,
                      color='mode',
                             labels={
'acousticness':'acoustic','danceability':'dance','instrumentalness':'instrume
nt','popularity':'popular','liveness':'live'
                             })
figscatter.update_layout(yaxis=dict(tickangle = 45),xaxis=dict(tickangle =
45))

#scatterplot
figscatter2 =
px.scatter(data,y='energy',x='popularity',color='mode',trendline='ols')

figscatter3 =
px.scatter(data,y='liveness',x='popularity',color='mode',trendline='ols')


tab7_layout=html.Div([html.H3('Welcome to Dashboard!',style={'margin': '0',
'textAlign': 'left'}),
                      html.Br(),

html.Div(style={'backgroundColor':'#0DF66C','color':'#111111','padding':5,
'flex': 1},
        children=[
            html.P('Pie Chart of Mode based on Popularity'),
            dcc.Graph(figure=figmode),
            html.P('Pie Chart of Explicit based on Popularity'),
            dcc.Graph(figure=figexp),
            html.P('Pie Chart of Key based on Popularity'),
            dcc.Graph(figure=figkey),

        ],
    ),
html.Div(style={'backgroundColor':'#0DF66C','color':'#111111','padding':5,
'flex': 1},
        children=[
            html.P('Scatter matrix'),
```

```
            dcc.Graph(figure=figscatter),
            html.P('Scatter Plot'),
            dcc.Graph(figure=figscatter2),
            dcc.Graph(figure=figscatter3),
        ]
    ),
],style={'display': 'flex', 'flex-direction':
'row','backgroundColor':'#0DF66C','color':'#111111'})


#Design for tab8
tab8_layout=html.Div(style={'backgroundColor':'#0DF66C','color':'#111111'},
                     children=[html.H3('Summary',style={'margin': '0',
'textAlign': 'left'}),
                               html.Br(),
                               html.P('The developed dash application helps
user to visualize the various aspects of songs from spotify collections.\n
Every year has top 100 songs and analysis of available features helps in
predicting the popularity of the song. The users can navigate through
different tabs to understand the data from the stage of preprocessing to
analysis of features at different levels. ',style={'margin': '1px',
'textAlign': 'left'}),
                               html.Hr(style={'border': '1px solid black'}),
                               html.H3('References',style={'margin': '0',
'textAlign': 'left'}),
                               html.Br(),
                               html.Plaintext('
1.https://dash.plotly.com/dash-core-components \n
2.https://dash.plotly.com/dash-html-components \n
3.https://dash.plotly.com/advanced-callbacks \n
4.https://plotly.com/python/box-plots/ \n
5.https://plotly.com/python/histograms/ \n 6.
https://plotly.com/python/distplot/',style={'margin': '1px', 'textAlign':
'left'}),
                               html.Hr(style={'border': '1px solid black'}),
                               html.H3('Author Information',style={'margin':
'0', 'textAlign': 'left'}),

                               html.Plaintext('Please feel free to drop an
email if you have any questions or suggestions to improve the app!\nCreated
by: Rehapriadarsini Manikandasamy\nEmail:rehamanikandan@gwu.edu',style =
{'backgroundColor':'#111111','color':'#0DF66C','font-size':'15px'}),
                               html.Plaintext('Data Source: Kaggle \n Link to
Dataset: https://www.kaggle.com/datasets/ektanegi/spotifydata-19212020 \n
*The app has been created for 6401-Visulaization of Complex Data coursework
at The George Washington University*',style =
{'backgroundColor':'#111111','color':'#0DF66C','font-size':'15px'})
                               ]
                     )

#Main callback for the main layout
@my_app.callback(Output(component_id='layout',component_property='children'),
                 [Input(component_id='tabs1',component_property='value')
                ])

def update_layout(tabselect):
    if tabselect=='Know your Data':
```

```
        return tab1_layout
    elif tabselect=='Outlier Analysis':
         return tab2_layout
    elif tabselect=='PCA':
         return tab3_layout
    elif tabselect=='Normality Tests':
         return tab4_layout
    elif tabselect=='Heatmap':
         return tab5_layout
    elif tabselect=='Analysis':
         return tab6_layout
    elif tabselect=='Dashboard':
        return tab7_layout
    elif tabselect=='Summary':
        return tab8_layout

#Callback for tab1
@my_app.callback(Output(component_id='datainfo',component_property='children'
),
                 [Input(component_id='infos',component_property='value')])
def update_graph(input):
    if input=='Column':
        cols=data.columns
        return ['\n'+j for j in cols]
    elif input=='rows':
         i=len(data)
         return f'Number of rows:{i}'
    elif input=='columns':
         cols=data.columns
         return f'Number of columns:{len(cols)}'

@my_app.callback(Output(component_id='preprocess',component_property='childre
n'),
                 [Input(component_id='cleans',component_property='value')])
def update_graph(input):
    if input=='nulls':
        d=data.isnull().sum()
        return f'{d}\nDataset is cleaned already!'
    if input=='stats':
        return f'{data.describe()}'
    elif input=='head':
        pd.set_option('max_columns',6)
        return f'{data.head()}'

@my_app.callback(
    Output("download-dataframe-csv", "data"),
    Input("btn_csv", "n_clicks"),
    prevent_initial_call=True,
)
def func(n_clicks):
    return dcc.send_data_frame(data.to_csv, "Spotify_data.csv")


#Callbacks fro tab2 components
@my_app.callback(Output(component_id='graphbox1',component_property='figure')
,
                 [Input(component_id='drop1',component_property='value')])
```

```
def update_graph(input):
    fig = px.box(data,y=input)
    fig.update_layout(title='Box plot')
    return fig

@my_app.callback(Output(component_id='graphbox2',component_property='figure')
,
                 [Input(component_id='drop2',component_property='value')])
def update_graph(input):
    fig = px.box(data1,y=input)
    fig.update_layout(title='Box plot')
    return fig

#PCA callbacks
@my_app.callback(Output(component_id='pcaout',component_property='children'),
                 [Input(component_id='checkpca',component_property='value')])
def update_graph(input):
    if input=='Original':
        return f'Features:{Features[:6]}\n{Features[6:]}\n\nOriginal 
Shape:{x.shape}\n\nSingular values:{d}\n\nCondition number:{la.cond(x)}'
    elif input=='tranformed':
        return f'Transformed shape:{x_pca.shape}\n\nSingular 
values:{d_pca}\n\nCondition number:{la.cond(x_pca)}\n\nExplained Variance 
Ratio:{pca.explained_variance_ratio_}'

#Normality callbacks
@my_app.callback(
    Output(component_id='ntout',component_property='children'),
    [Input(component_id='dropvar',component_property='value'),
     Input(component_id='droptest',component_property='value')]
)
def tests(inp,inp2):
    f1=data[inp]
    if inp2=='normaltest':
        return f'Normal test:{normaltest(f1)}'
    elif inp2=='kstest':
        res=kstest(f1,'norm')
        return f'KS test:{res}'
    else:
        return f'Shapiro Wilk Test:{shapiro(f1)}'
#Analysis callbacks
@my_app.callback(Output(component_id='line',component_property='figure'),
                Output(component_id='bar',component_property='figure'),
                Output(component_id='graphd',component_property='figure'),
                 [Input(component_id='dropline',component_property='value'),
                  Input(component_id='dropcolor',component_property='value'),
                  Input(component_id='bins',component_property='value'),

Input(component_id='distribution',component_property='value')])
def update_graph(input,inp2,inp3,inp4):
    fig = px.line(data,x='year',y=input,color=inp2)
    fig1=px.histogram(data,x=input,nbins=inp3)
    fig3 = px.histogram(data, x='popularity', y=input, color=inp2,
        marginal=inp4)
    return fig,fig1,fig3
```

```
my_app.run_server(port=8100, host='0.0.0.0')
```

REFERENCES

- https://dash.plotly.com/dash-core-components
- https://dash.plotly.com/dash-html-components
- https://dash.plotly.com/advanced-callbacks
- https://plotly.com/python/box-plots/
- https://plotly.com/python/histograms/
- https://plotly.com/python/distplot/
- https://plotly.com/python/pie-charts/
- All Lecture notes and codes of assignments, in-class activity