Universität St.Gallen

Hochschule für Wirtschafts-,
Rechts- und Sozialwissenschaften
sowie Internationale Beziehungen

CODING PROJECT:

The Determinants of Popular Songs

FREDERIC BARZ
FRANCA BOEGLIN
EMILE CHAMPAGNE

Spring semester 2020
Programming with Advanced
Computer Languages

1. Project description

1.1. Goal

The focus of this group project is on determinants of popular songs. In particular, the project analyzes which sound characteristics explain the popularity of different songs. The project further investigates the dynamics of the factors across time (2010 - 2019). That is, the main focus is a study of multivariate correlations. In addition, the project provides an interactive part which enables the user of this project to interact with the data. Therefore, our group project addresses the following questions:

- Which determinants explain the popularity of songs?
- Have any dynamics changed over time?

1.2. Resources

The inputs used by the project is a dataset of 609 songs from the top rankings of the years 2010 to 2019. The data set contains the songs' titles, artist, year published, genre, popularity and multiple sound characteristics. The sound measures analyzed are the beats per minute, energy, danceability, loudness, liveness, duration, acousticness and speechiness. The coding project was written in Python with the help of the Jupyter Notebook and a GitHub repository file.

You can find the files here:

- Dataset Top10s.csv
- GitHub Repository
- group.project.ipynb (File contains code to load in Jupyter Notebook)

How to use the document:

1. To use the document, you should download the **skills_group_project repository** from GitHub via the link (GitHub Repository) above.
2. After downloading/ cloning the repository, you should launch Jupyter Notebook via Anaconda.
3. Then open the file group.project.ipynb in Juyppter Notebook to get access to the report and code. The group.project.ipynb contains the coded embedded in a report and a detailed documentation.
4. Run the code as described in the next chapter.

2. Code description

The group project is structured in the following parts:

1. Introduction
2. Set-Up and Data Understanding
3. Exploratory Analysis
4. Interactive part
5. Regression Analysis
6. Conclusion

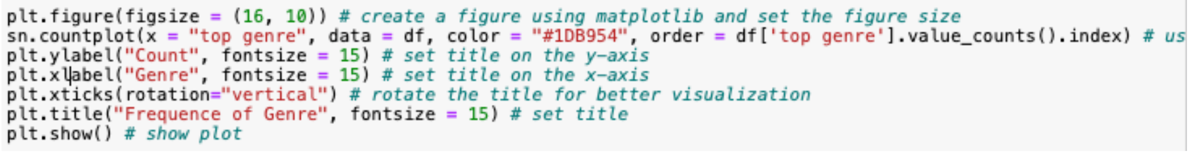
In the following, the code for the exploratory analysis, the interactive part as well as the regression analysis is described and underpinned with code extracts.

2.1. Exploratory Analysis

The exploratory analysis consists of a univariate and bivariate analysis. Since the variables are either categorical or numeric, barcharts respectively histograms are plotted regarding the univariate analysis. Both visualization techniques help the user to identify the distributions of the dataset.

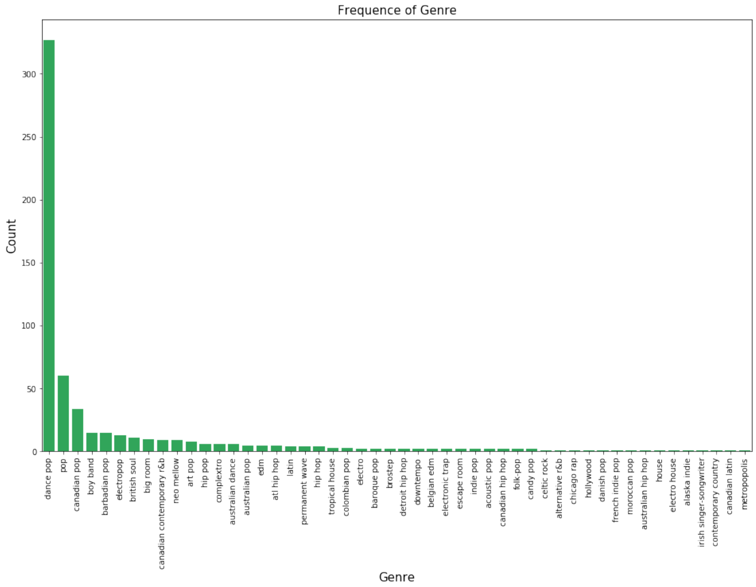
The following code describes how to plot a barchart for the genre. In particular, the plot shows how many songs belong to a certain genre.

CODE 1:



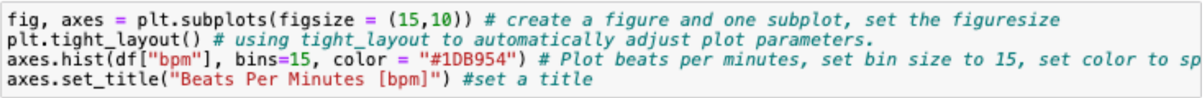
```
plt.figure(figsize = (16, 10)) # create a figure using matplotlib and set the figure size
sn.countplot(x = "top genre", data = df, color = "#1DB954", order = df['top genre'].value_counts().index) # us
plt.ylabel("Count", fontsize = 15) # set title on the y-axis
plt.xlabel("Genre", fontsize = 15) # set title on the x-axis
plt.xticks(rotation="vertical") # rotate the title for better visualization
plt.title("Frequence of Genre", fontsize = 15) # set title
plt.show() # show plot
```

OUTPUT 2:



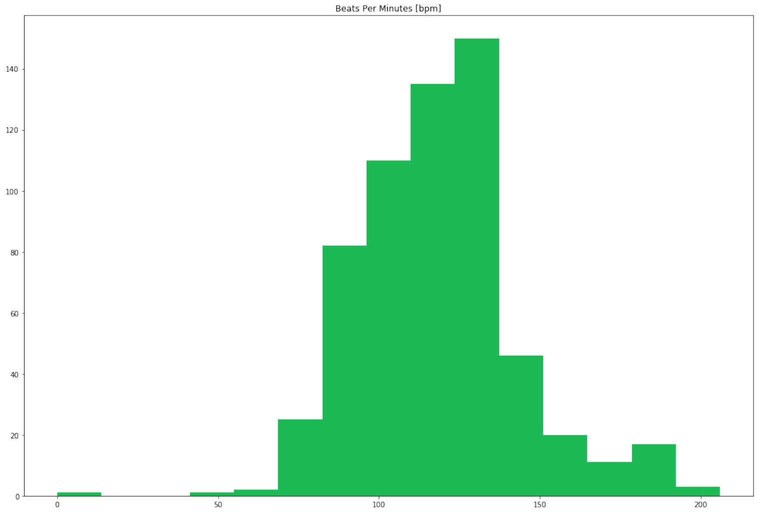
To understand the distribution of each numeric variable better, numerous histograms were plotted. The following histograms give an idea of how the values for the beats per minute variable are distributed.

CODE 2:



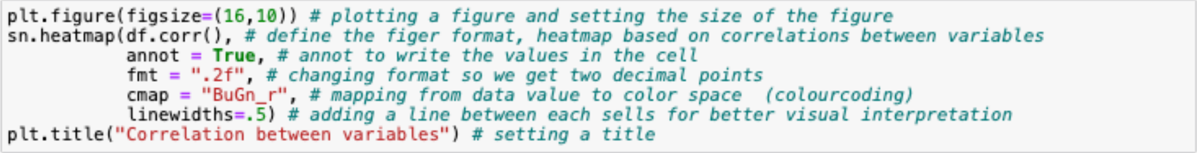
```
fig, axes = plt.subplots(figsize = (15,10)) # create a figure and one subplot, set the figuresize
plt.tight_layout() # using tight_layout to automatically adjust plot parameters.
axes.hist(df["bpm"], bins=15, color = "#1DB954") # Plot beats per minutes, set bin size to 15, set color to sp
axes.set_title("Beats Per Minutes [bpm]") #set a title
```

OUTPUT 2:



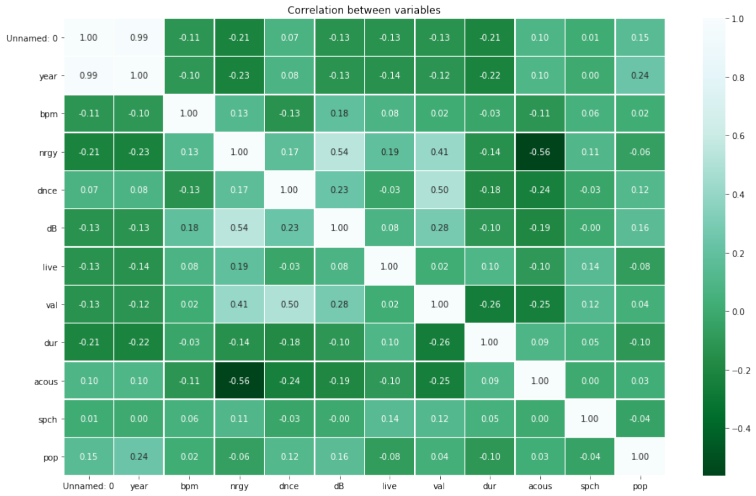
Regarding the univariate analysis, a correlation matrix and scatterplots were plotted. Color coding was used to identify relationships more easily. The following plot shows the correlations matrix.

CODE 3:



```
plt.figure(figsize=(16,10)) # plotting a figure and setting the size of the figure
sn.heatmap(df.corr(), # define the figer format, heatmap based on correlations between variables
           annot = True, # annot to write the values in the cell
           fmt = ".2f", # changing format so we get two decimal points
           cmap = "BuGn_r", # mapping from data value to color space  (colourcoding)
           linewidths=.5) # adding a line between each sells for better visual interpretation
plt.title("Correlation between variables") # setting a title
```

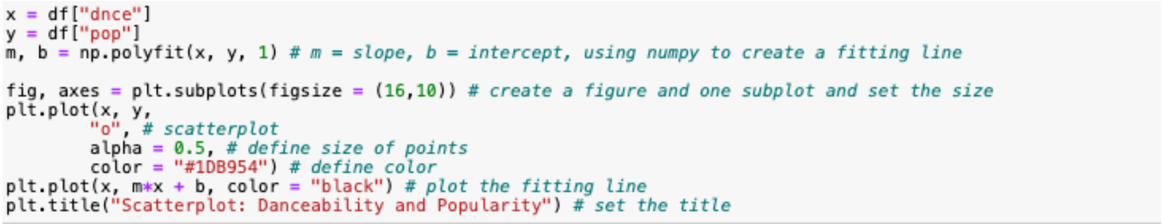
OUTPUT 3:



	Unnamed: 0	year	bpm	nrgy	dnce	dB	live	val	dur	acous	spch	pop
Unnamed: 0	1.00	0.99	-0.11	-0.21	0.07	-0.13	-0.13	-0.13	-0.21	0.10	0.01	0.15
year	0.99	1.00	-0.10	-0.23	0.08	-0.13	-0.14	-0.12	-0.22	0.10	0.00	0.24
bpm	-0.11	-0.10	1.00	0.13	-0.13	0.18	0.08	0.02	-0.03	-0.11	0.06	0.02
nrgy	-0.21	-0.23	0.13	1.00	0.17	0.54	0.19	0.41	-0.14	-0.56	0.11	-0.06
dnce	0.07	0.08	-0.13	0.17	1.00	0.23	-0.03	0.50	-0.18	-0.24	-0.03	0.12
dB	-0.13	-0.13	0.18	0.54	0.23	1.00	0.08	0.28	-0.10	-0.19	-0.00	0.16
live	-0.13	-0.14	0.08	0.19	-0.03	0.08	1.00	0.02	0.10	-0.10	0.14	-0.08
val	-0.13	-0.12	0.02	0.41	0.50	0.28	0.02	1.00	-0.26	-0.25	0.12	0.04
dur	-0.21	-0.22	-0.03	-0.14	-0.18	-0.10	0.10	-0.26	1.00	0.09	0.05	-0.10
acous	0.10	0.10	-0.11	-0.56	-0.24	-0.19	-0.10	-0.25	0.09	1.00	0.00	0.03
spch	0.01	0.00	0.06	0.11	-0.03	-0.00	0.14	0.12	0.05	0.00	1.00	-0.04
pop	0.15	0.24	0.02	-0.06	0.12	0.16	-0.08	0.04	-0.10	0.03	-0.04	1.00

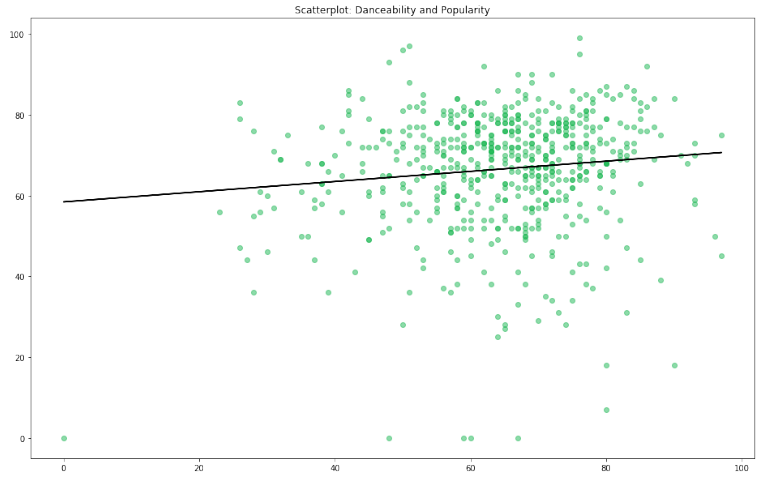
To see if there could be a potential relationship between an independent and the dependent variables “popularity” scatterplots were plotted. The following example shows the relationship between the independent variable “danceability” and the dependent variable “popularity”.

CODE 4:



```
x = df["dnce"]
y = df["pop"]
m, b = np.polyfit(x, y, 1) # m = slope, b = intercept, using numpy to create a fitting line

fig, axes = plt.subplots(figsize = (16,10)) # create a figure and one subplot and set the size
plt.plot(x, y,
        "o", # scatterplot
        alpha = 0.5, # define size of points
        color = "#1DB954") # define color
plt.plot(x, m*x + b, color = "black") # plot the fitting line
plt.title("Scatterplot: Danceability and Popularity") # set the title
```

OUTPUT 4:



2.2. Interactive Part

The second part of the code provides the user with three search functions to interact with the dataset and extract information:

- Search for information about an artist
- Search for the top ranking of a year
- Search for information about a song

The user is first prompted by the following text and can enter the numbers 1, 2 and 3 to access the different functions:

```
Here you can perform multiple types of searches within the database.
ATTENTION: CAPITALIZE THE FIRST LETTER OF EVERY WORD AND NAME

What do you want to search for?
    1: Search for information about an artist
    2: Search for the Top Song ranking of a year
    3: Search for information about a song
    Please enter the corresponding number:
```

Search function #1:

```
Input: 1
Output:
ARTIST SEARCH FUNCTION
Enter an artist's name (or part of a name) to see their songs:


INPUT: Lady Gaga
OUTPUT:

        artist                title  year  top genre
3    Lady Gaga          Bad Romance  2010  dance pop
13   Lady Gaga            Telephone  2010  dance pop
18   Lady Gaga            Alejandro  2010  dance pop
62   Lady Gaga        Born This Way  2011  dance pop
76   Lady Gaga     The Edge Of Glory  2011  dance pop
81   Lady Gaga            You And I  2011  dance pop
82   Lady Gaga               Judas  2011  dance pop
86   Lady Gaga       Marry The Night  2011  dance pop
195  Lady Gaga             Applause  2013  dance pop
243  Lady Gaga               G.U.Y.  2014  dance pop
370  Lady Gaga       Million Reasons  2016  dance pop
405  Lady Gaga      Perfect Illusion  2016  dance pop
475  Lady Gaga             The Cure  2017  dance pop
524  Lady Gaga  Shallow - Radio Edit  2018  dance pop
```

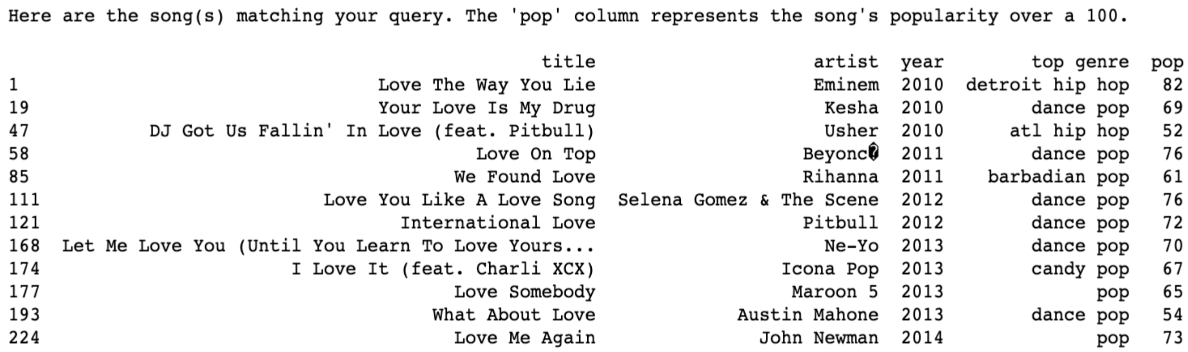
Search function #2:

```
INPUT: 2
OUTPUT:
TOP SONG RANKING DISPLAY SEARCH FUNCTION (2010-2019)
Which year's ranking do you want to display?


INPUT: Love
OUTPUT:
                        artist                                           title  year
0                        Train                                Hey, Soul Sister  2010
1                       Eminem                           Love The Way You Lie  2010
2                        Kesha                                         TiK ToK  2010
3                    Lady Gaga                                     Bad Romance  2010
4                   Bruno Mars                           Just the Way You Are  2010
5                Justin Bieber                                            Baby  2010
6                    Taio Cruz                                        Dynamite  2010
7                  OneRepublic                                         Secrets  2010
8                  Alicia Keys       Empire State of Mind (Part II) Broken Down  2010
9                      Rihanna                        Only Girl (In The World)  2010
10                    Flo Rida      Club Can't Handle Me (feat. David Guetta)  2010
11                  Bruno Mars                                       Marry You  2010
12                 Mike Posner                        Cooler Than Me - Single Mix  2010
13                   Lady Gaga                                       Telephone  2010
14           Far East Movement                                        Like A G6  2010
15                       Usher                               OMG (feat. will.i.am)  2010
16               Sean Kingston                                     Eenie Meenie  2010
17         The Black Eyed Peas                           The Time (Dirty Bit)  2010
18                   Lady Gaga                                       Alejandro  2010
```

Search function #3:

```
INPUT:  3
OUTPUT:
SONG SEARCH FUNCTION
Enter a song's name (or part of a name) to see information about it:
```

```
INPUT:  Love
OUTPUT:
```



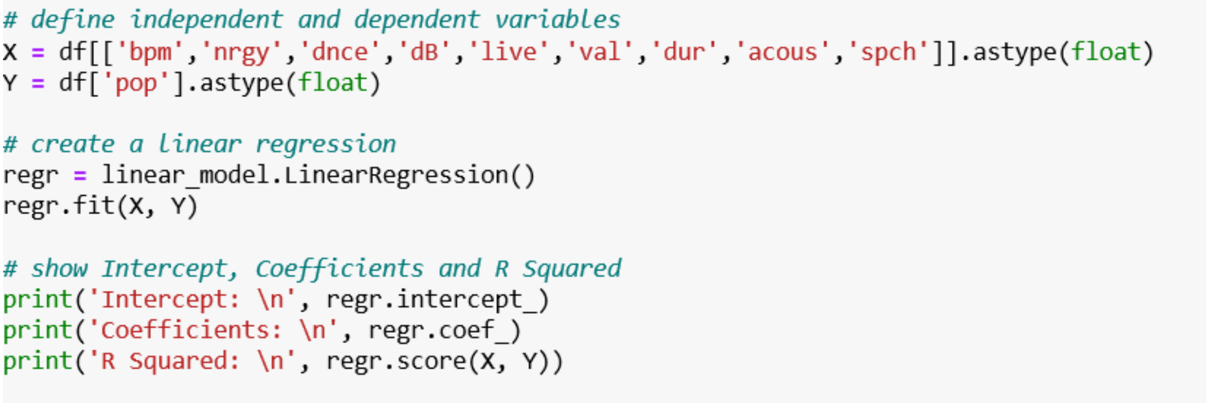
```
Here are the song(s) matching your query. The 'pop' column represents the song's popularity over a 100.

                                                 title                 artist  year        top genre  pop
1                                 Love The Way You Lie                 Eminem  2010  detroit hip hop   82
19                                Your Love Is My Drug                  Kesha  2010        dance pop   69
47          DJ Got Us Fallin' In Love (feat. Pitbull)                  Usher  2010      atl hip hop   52
58                                         Love On Top                Beyonc�  2011        dance pop   76
85                                       We Found Love                Rihanna  2011    barbadian pop   61
111                           Love You Like A Love Song  Selena Gomez & The Scene  2012        dance pop   76
121                                 International Love                Pitbull  2012        dance pop   72
168  Let Me Love You (Until You Learn To Love Yours...                  Ne-Yo  2013        dance pop   70
174                        I Love It (feat. Charli XCX)              Icona Pop  2013        candy pop   67
177                                      Love Somebody               Maroon 5  2013              pop   65
193                                    What About Love          Austin Mahone  2013        dance pop   54
224                                      Love Me Again            John Newman  2014              pop   73
```

2.3. Regression Analysis

In this section, the code creates a linear regression based on the dataset to predict the dependent variable popularity (“pop”) by using the song attributes as independent variables. Additionally, the program will provide a Graphical User Interface that allows the user to enter values to calculate a predicted popularity.

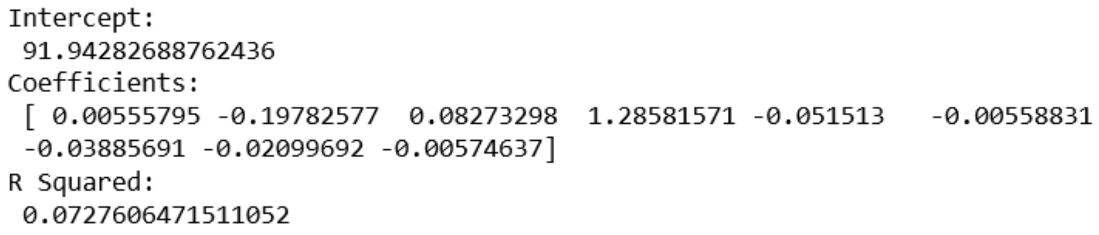
The First lines of code create a regression model based on the dataset. Further it shows the intercept, the coefficients and the R Squared in the outpour after running the code.

CODE 1: Creation of the linear regression and printing values



```
# define independent and dependent variables
X = df[['bpm','nrgy','dnce','dB','live','val','dur','acous','spch']].astype(float)
Y = df['pop'].astype(float)

# create a linear regression
regr = linear_model.LinearRegression()
regr.fit(X, Y)

# show Intercept, Coefficients and R Squared
print('Intercept: \n', regr.intercept_)
print('Coefficients: \n', regr.coef_)
print('R Squared: \n', regr.score(X, Y))
```

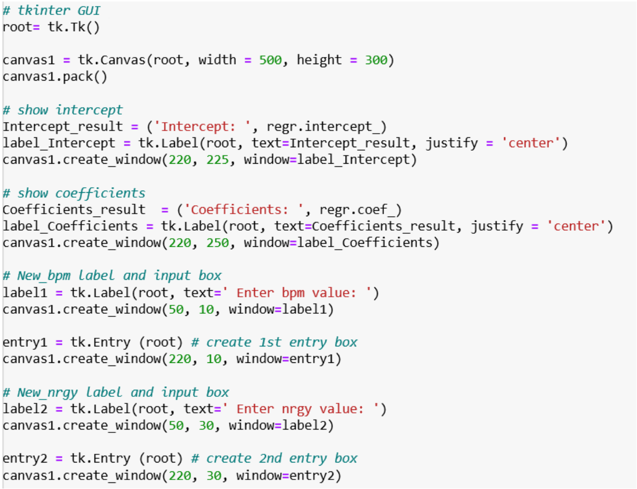
OUTPUT 1: Intercept, Coefficients and R Squared



```
Intercept:
 91.94282688762436
Coefficients:
 [ 0.00555795 -0.19782577  0.08273298  1.28581571 -0.051513   -0.00558831
 -0.03885691 -0.02099692 -0.00574637]
R Squared:
 0.0727606471511052
```

The code that follows is used to create a Graphical User Interface to enable the user to enter values for all independent variables to calculate the popularity. Additionally, it will contain an overview of all the scatter plots from the descriptive statistics section to assist the user in finding values for the specific factor. To achieve this the tkinter is first imported. The window manager is initialized with the tk.Tk() method. There it is assigned to the variable root. This creates a blank window with minimize, maximize, and close buttons. Afterwards the window size is defined. Now that the framework of the GUI has been set the contents are determined. We create a position where the user will see the regression

Code 2 creating the window, variables, and input boxes



```
# tkinter GUI
root= tk.Tk()

canvas1 = tk.Canvas(root, width = 500, height = 300)
canvas1.pack()

# show intercept
Intercept_result = ('Intercept: ', regr.intercept_)
label_Intercept = tk.Label(root, text=Intercept_result, justify = 'center')
canvas1.create_window(220, 225, window=label_Intercept)

# show coefficients
Coefficients_result  = ('Coefficients: ', regr.coef_)
label_Coefficients = tk.Label(root, text=Coefficients_result, justify = 'center')
canvas1.create_window(220, 250, window=label_Coefficients)

# New_bpm label and input box
label1 = tk.Label(root, text=' Enter bpm value: ')
canvas1.create_window(50, 10, window=label1)

entry1 = tk.Entry (root) # create 1st entry box
canvas1.create_window(220, 10, window=entry1)

# New_nrgy label and input box
label2 = tk.Label(root, text=' Enter nrgy value: ')
canvas1.create_window(50, 30, window=label2)

entry2 = tk.Entry (root) # create 2nd entry box
canvas1.create_window(220, 30, window=entry2)
```

function below the input boxes, we created for the calculation (Code 3). Then we define the new variables (“values”) as the new entries, which then will be used for the calculation of the predicted popularity (Code 3). Here we customize the position and the color of our newly created window to highlight the result. The following line of code creates a button that triggers the calculation. Here we also choose a pink coloring to highlight the button. Additionally, we placed the button directly under the input boxes. To finish the code, we then set the characteristics of the scatterplot figures below (Code 4). The mainloop() method creates an infinite loop and thus displays the window until its closed by the user.

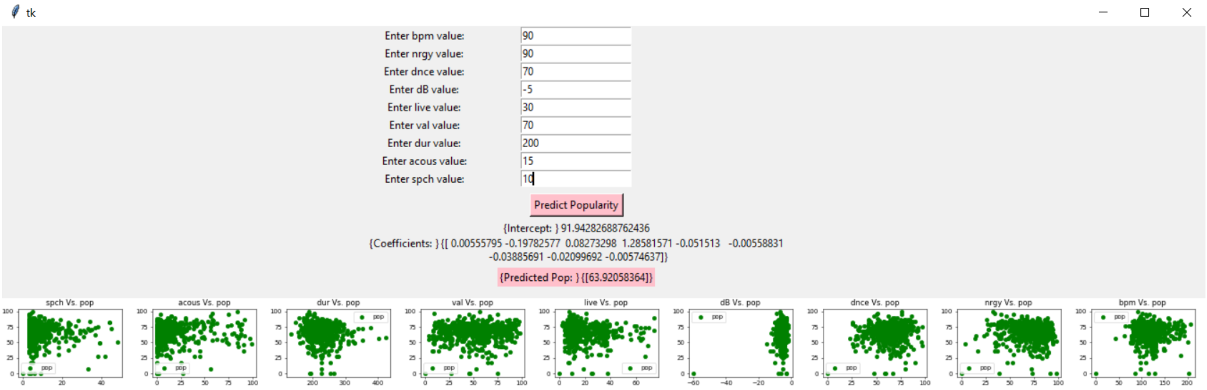
Code 3 new variables ("values")

```
# define new variables
def values():
    global New_bpm # 1st input variable
    New_bpm = float(entry1.get())

    global New_nrgy # 2nd input variable
    New_nrgy = float(entry2.get())

    global New_dnce # 3rd input variable
    New_dnce = float(entry3.get())

    global New_dB #our 4th input variable
    New_dB = float(entry4.get())
```

Code 4: Prediction result, button, and scatter plot

```
    # use new variable inputs to predict popularity
    Prediction_result  = ('Predicted Pop: ', regr.predict([[New_bpm ,New_nrgy
    label_Prediction = tk.Label(root, text= Prediction_result, bg='pink')
    canvas1.create_window(220, 280, window=label_Prediction)

    # button to call the 'values' command above
button1 = tk.Button (root, text='Predict Popularity',command=values, bg='pink'
canvas1.create_window(220, 200, window=button1)

#plot 1st scatter
figure1 = plt.Figure(figsize=(3,2), dpi=50)
ax1 = figure1.add_subplot(111)
ax1.scatter(df['bpm'].astype(float),df['pop'].astype(float), color = 'g')
scatter1 = FigureCanvasTkAgg(figure1, root)
scatter1.get_tk_widget().pack(side=tk.RIGHT, fill=tk.BOTH)
ax1.legend(['pop'])
ax1.set_xlabel('bpm')
ax1.set_title('bpm Vs. pop')
```

OUTPUT 2: Graphical User Interface

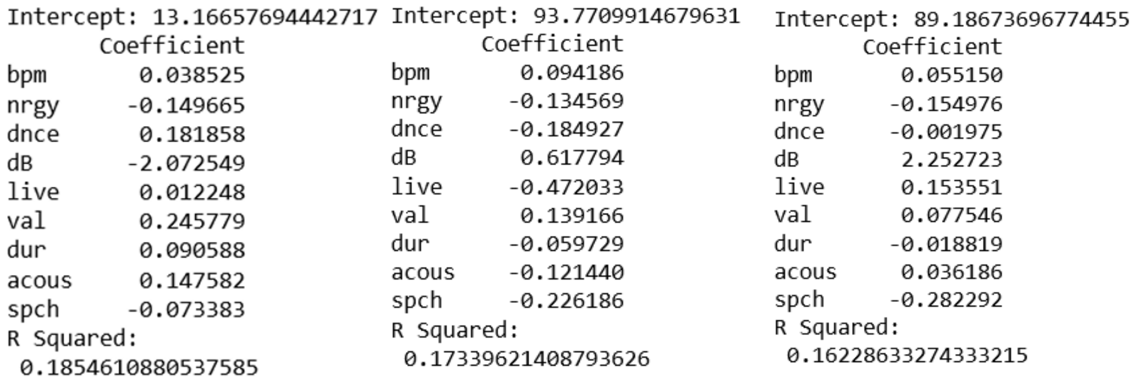


Enter bpm value: 90
Enter nrgy value: 90
Enter dnce value: 70
Enter dB value: -5
Enter live value: 30
Enter val value: 70
Enter dur value: 200
Enter acous value: 15
Enter spch value: 10

```
{Intercept: } 91.94282688762436
{Coefficients: }{[ 0.00555795 -0.19782577  0.08273298  1.28581571 -0.051513   -0.00558831
 -0.03885691 -0.02099692 -0.00574637]}
{Predicted Pop: }{[63.92058364]}
```

Now it could be considered that a time effect influences the accuracy of the model as it is only able to explain roughly 7% of the variance. Therefore, further regression models will be created to observe data from specific years separately to investigate if the coefficients are comparable in different years.

OUTPUT 3: Outputs of the 3 separate regressions



```
Intercept: 13.16657694442717
          Coefficient
bpm          0.038525
nrgy        -0.149665
dnce         0.181858
dB          -2.072549
live         0.012248
val          0.245779
dur          0.090588
acous        0.147582
spch        -0.073383
R Squared:
 0.1854610880537585
```

```
Intercept: 93.7709914679631
          Coefficient
bpm          0.094186
nrgy        -0.134569
dnce        -0.184927
dB           0.617794
live        -0.472033
val          0.139166
dur         -0.059729
acous       -0.121440
spch        -0.226186
R Squared:
 0.17339621408793626
```

```
Intercept: 89.18673696774455
          Coefficient
bpm          0.055150
nrgy        -0.154976
dnce        -0.001975
dB           2.252723
live         0.153551
val          0.077546
dur         -0.018819
acous        0.036186
spch        -0.282292
R Squared:
 0.16228633274333215
```

Output 3 shows that the coefficients differ significantly in different years. Thus, it can be assumed that a time effect distorts our previous regression, which could explain the low R-Squared value. However, it can be observed that the individual R-Squared values are not high enough for an accurate model. Therefore, we have to question if it makes sense to predict a songs popularity by using its characteristics.