Linear Models

Project report (Multiple Linear Regression)

**Name & Prn:**

1. Akshata Satpute (22060641073)
2. Komal solkar (22060641055)

**model fitting:**

**Problem Statement:**

Predicting an artist’s popularity based on different independent variables like artists, popularity, Acousticness, Danceability, Duration\_ms, Energy, Instrumentalness, liveness, loudness, speechiness, temp, key, valence, mode, and count.

**Data:** At the end of each year, Spotify compiles a playlist of the songs streamed most often over the course of that year. This year's playlist (Top Tracks of 2018). The question is: What do these top songs have in common? Why do people like them?

Original Data Source: The audio features for each song were extracted using the Spotify Web API and the Spotify Python library. Credit goes to Spotify for calculating the audio feature values.

Data Description: There is one .csv file in the dataset. (top2018.csv) This file includes:

* Spotify URI for the song
* Name of the song
* Artist(s) of the song
* Audio features for the song (such as danceability, tempo, key, etc.)
* A more detailed explanation of the audio features can be found in the Metadata tab.

The data set consists of 27622 observations. Where it contains 13 columns and 27622 rows. We considered the artist’s popularity as our response variable and the other 12 as independent variables.

**Source of data:** Kaggle

**Software used:** Python

Python code with output:

**Installing libraries and importing data into Python**

**Command**:

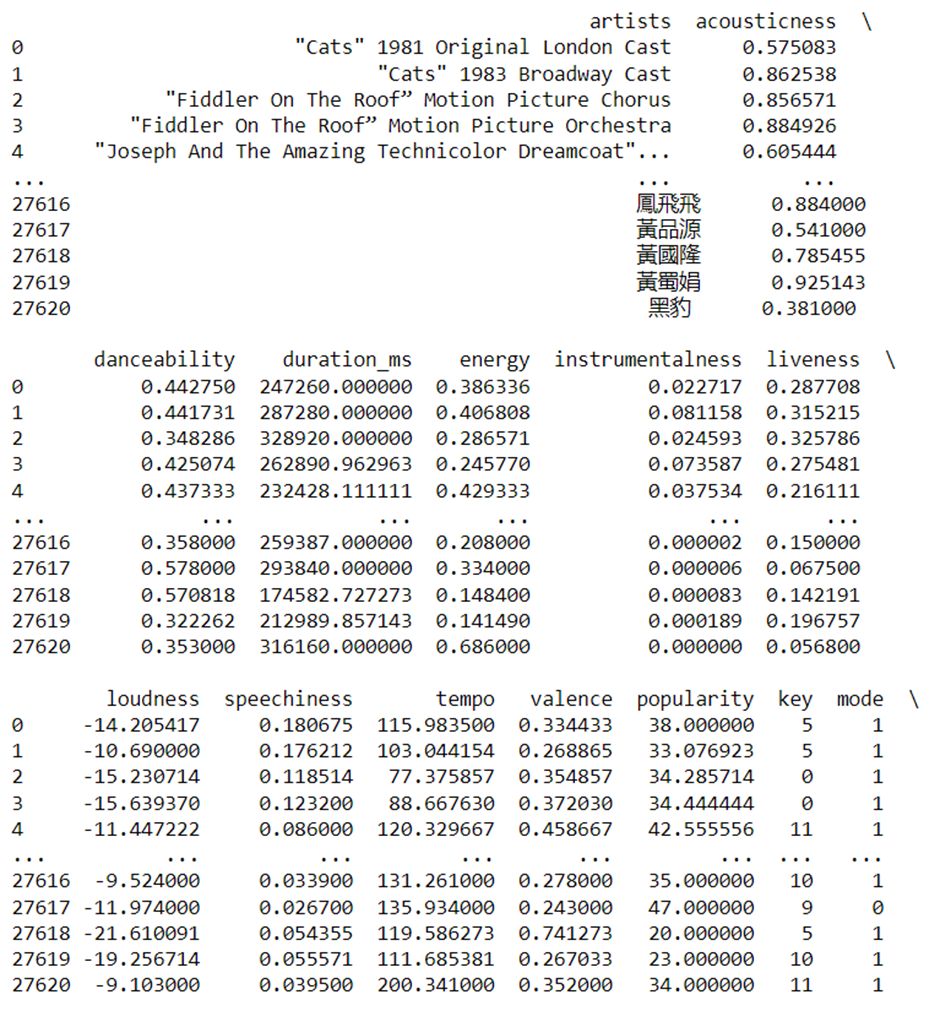
import pandas as pd

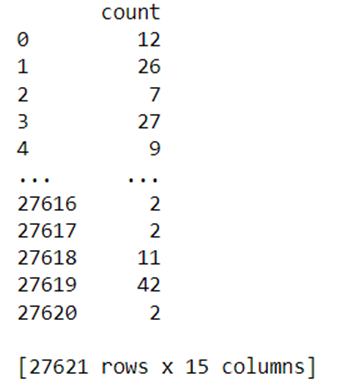
import numpy as np

data=pd.read\_csv("C:/Users/User/Downloads/artist.csv")

print(data)

**output:**



****

**Checking for data shape**

**Command:** data.shape

**Output:**

(27621, 15)

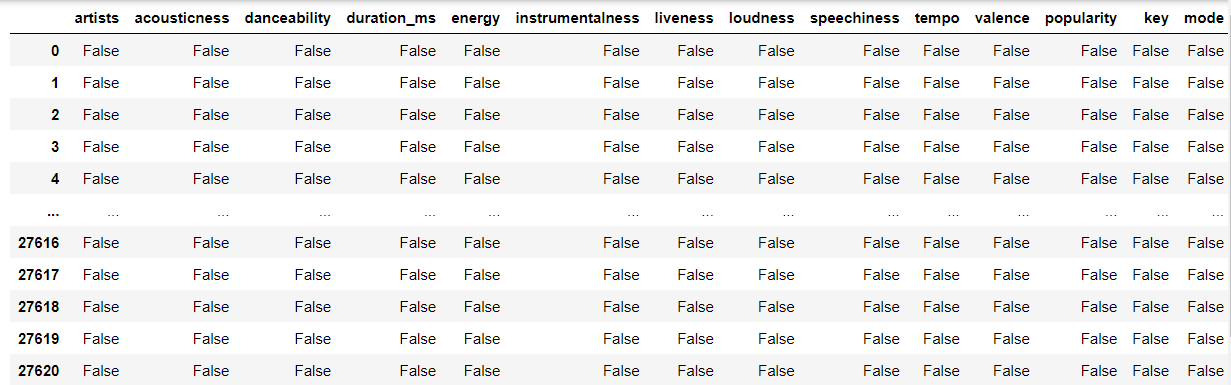
Conclusion: Data consists of 27621 rows and 15 columns in total.

**Checking for missing value**

**Command:**

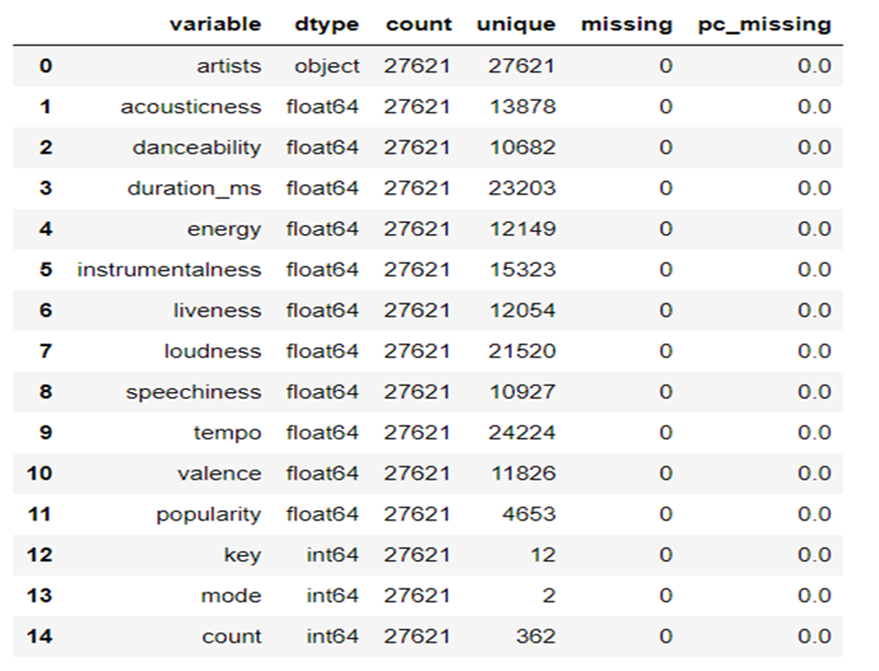
data.isna().sum()

**Output:**

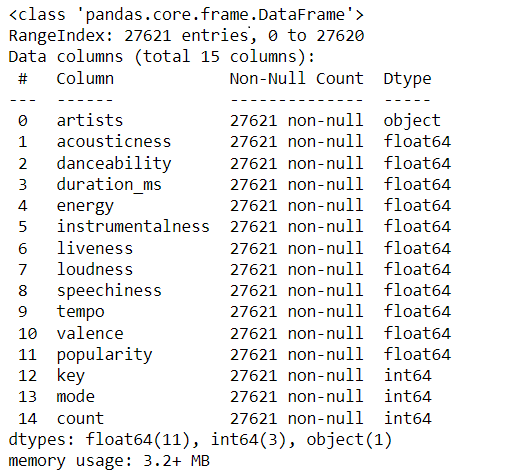
****

Conclusion: The Artist popularity Data set Contains no null/ Missing Values

**Checking total null Values count in the data:**

****Conclusion: Data contains no Null Values.

**Checking the data structure:**

****

Conclusion:

From above output, we conclude that this particular data set contains one variable artist as a datatype object. key mode and count as int64 data type and the remaining variables are float64 data type

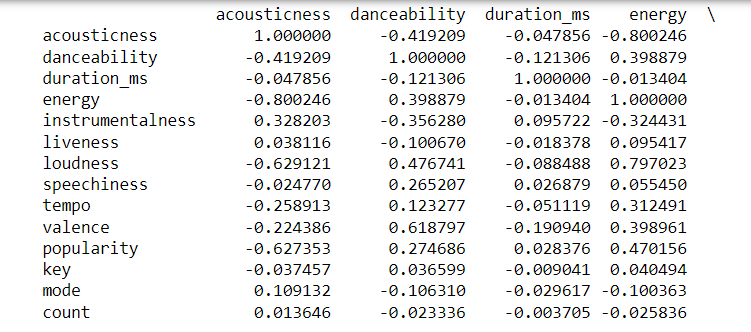
**Checking collinearity between the data variables;**

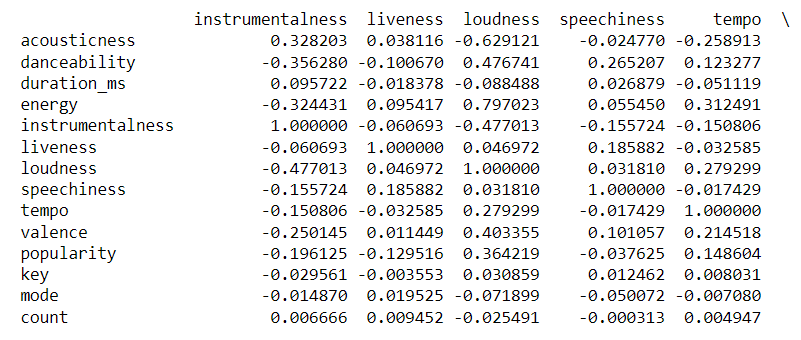
**Command:**

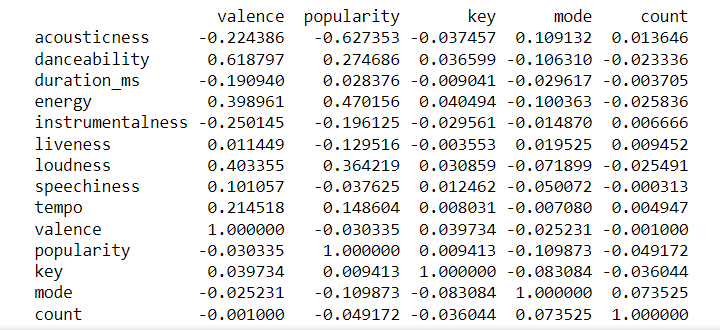
a=(data.corr())

print(a)

**Output:**

****

****

****

Conclusion:

The highest correlation between the dependent variable Popularity and other independent variable is seen to be maximum 0.4 whereas the independent variables show high correlation among themselves which implies that there exists a multicollinearity.

**Heatmap for checking correlation between dependent and independent variables:**

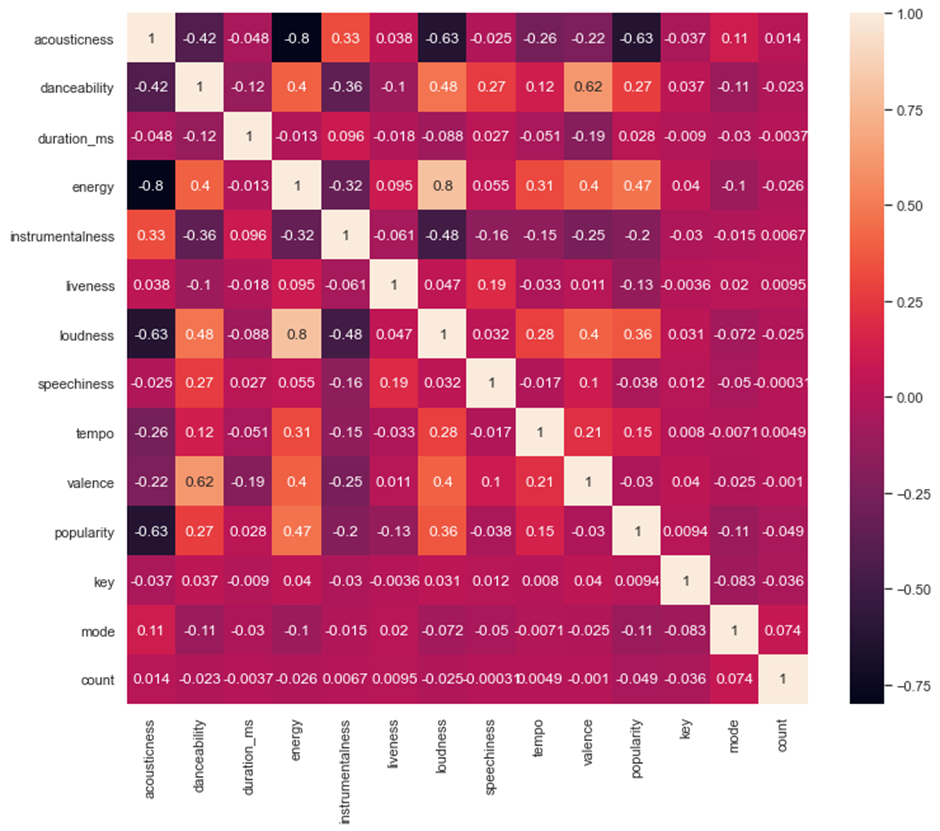
**Command:**

import seaborn

seaborn.set (rc = {'figure.figsize':(12, 10)})

seaborn.heatmap(a,annot=True)

**Output:**

****

**Defining X and Y**

**Command:**

from statsmodels.stats.outliers\_influence import variance\_inflation\_factor

x=data[['acousticness','danceability','duration\_ms','energy','instrumentalness','liveness','loudness','speechiness','tempo','valence','key','count']]

y=data[['popularity']]

**Determining multicollinearity between independent variables:**

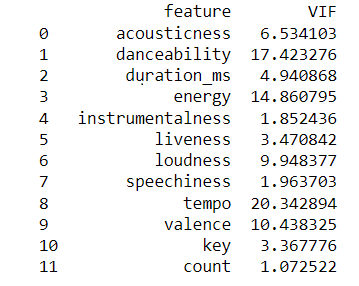
**Command:**

vif\_data["VIF"]=[variance\_inflation\_factor(x.values,i)

for i in range(len(x.columns))]

print(vif\_data)

**Output:**

****

Conclusion:

We found by VIF that the duration-MS, instrumentalness, liveness, speechiness, key, count are the variables with less VIF i.e. less than 5 which implies low multicollinearity whereas other variables with VIF value greater than 5 shows high multicollinearity.

**Applying the OLS i.e Ordinary Least square method to find the parameter estimates:**

# imports

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import statsmodels.api as sm

import statsmodels.formula.api as smf

import seaborn as sns

**# Fit a OLS regression variable**

y=data[['popularity']]

x=data[['acousticness','danceability','duration\_ms','energy','instrumentalness','liveness','loudness','speechiness','tempo','valence','key','count']]

x=sm.add\_constant(x)

results=sm.OLS(y,x).fit()

print(results.summary())

**# Get different Variables for diagnostic**

residuals = results.resid

fitted\_value = results.fittedvalues

stand\_resids = results.resid\_pearson

influence = results.get\_influence()

leverage = influence.hat\_matrix\_diag

**# PLot different diagnostic plots**

plt.rcParams["figure.figsize"] = (20,15)

fig, ax = plt.subplots(nrows=2, ncols=2)

plt.style.use('seaborn')

**# Residual vs Fitted Plot**

sns.scatterplot(x=fitted\_value, y=residuals, ax=ax[0, 0])

ax[0, 0].axhline(y=0, color='grey', linestyle='dashed')

ax[0, 0].set\_xlabel('Fitted Values')

ax[0, 0].set\_ylabel('Residuals')

ax[0, 0].set\_title('Residuals vs Fitted Fitted')

**# Normal Q-Q plot**

sm.qqplot(residuals, fit=True, line='45',ax=ax[0, 1], c='#4C72B0')

ax[0, 1].set\_title('Normal Q-Q')

**# Scale-Location Plot**

sns.scatterplot(x=fitted\_value, y=residuals, ax=ax[1, 0])

ax[1, 0].axhline(y=0, color='grey', linestyle='dashed')

ax[1, 0].set\_xlabel('Fitted values')

ax[1, 0].set\_ylabel('Sqrt(standardized residuals)')

ax[1, 0].set\_title('Scale-Location Plot')

**# Residual vs Leverage Plot**

sns.scatterplot(x=leverage, y=stand\_resids, ax=ax[1, 1])

ax[1, 1].axhline(y=0, color='grey', linestyle='dashed')

ax[1, 1].set\_xlabel('Leverage')

ax[1, 1].set\_ylabel('Sqrt(standardized residuals)')

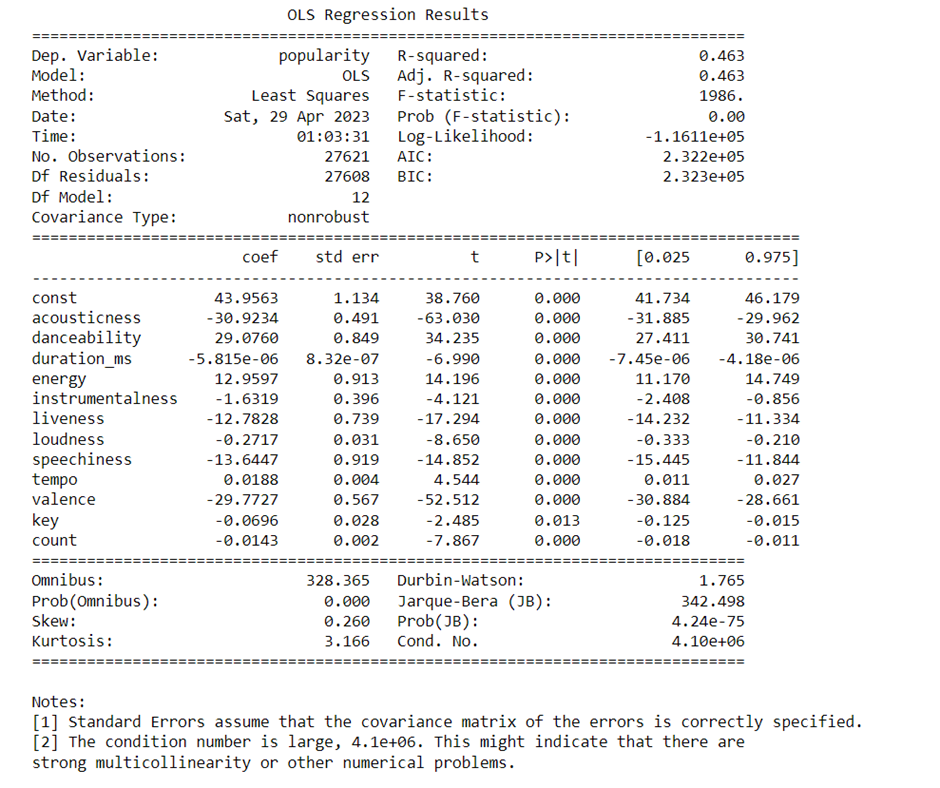
ax[1, 1].set\_title('Residuals vs Leverage Plot')

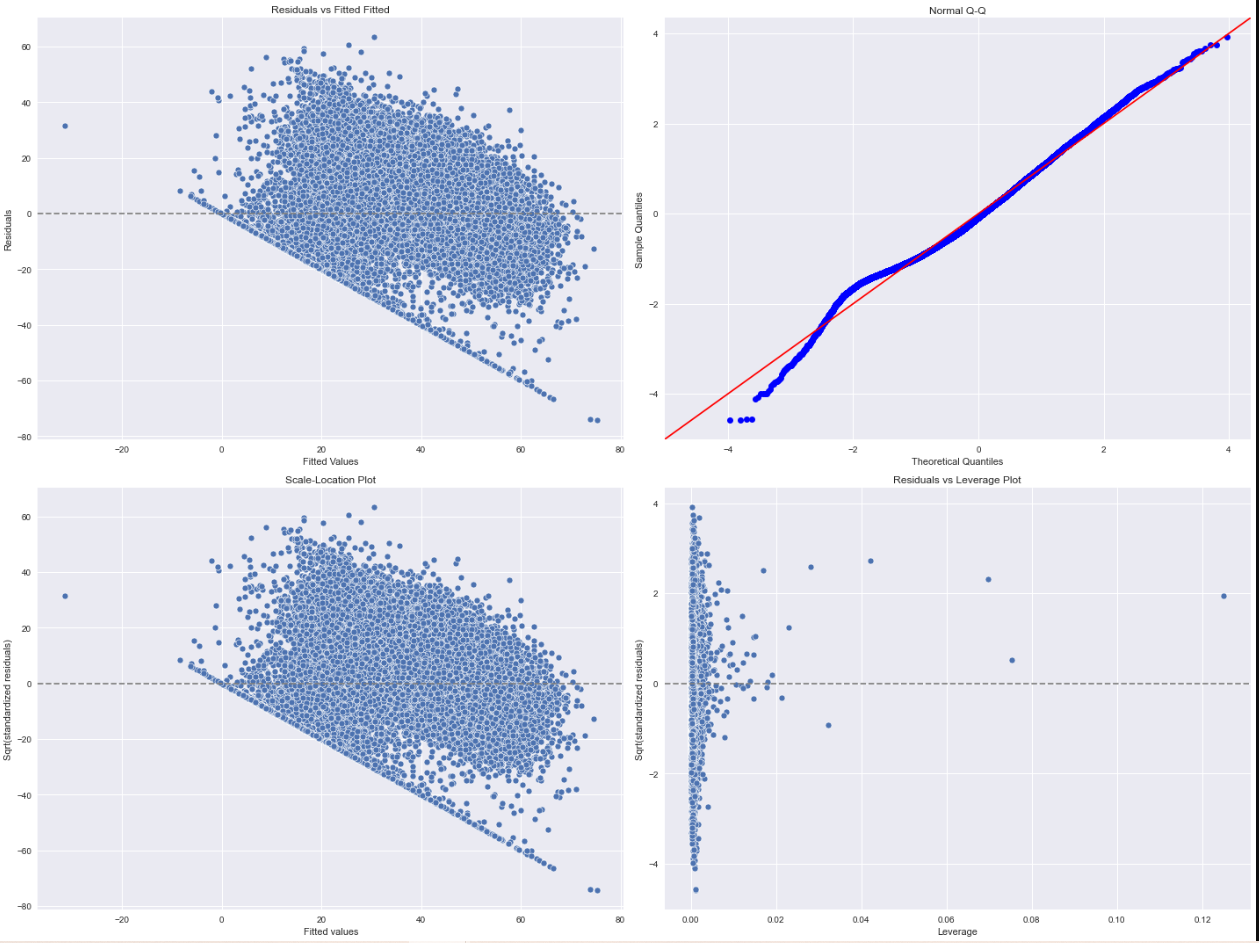
plt.tight\_layout()

plt.show()

**# PLot Cook's distance plot**

sm.graphics.influence\_plot(results, criterion="cooks")

****

****

Conclusion:

1. Residual versus fitted values:

The residuals vs. fitted plot appear to be relatively flat and homoscedastic. It has this odd cut-off in the bottom left.

1. Normal Q-Q plot:

The deviations from the straight line are minimal. This indicates normal distribution from the plot.

1. Scale location plot:

There is a clear pattern among the residuals. In other words, the residuals are not randomly scattered around the red line with roughly equal variability at all fitted values.

1. Residuals versus leverage plot:

If any point in this plot falls outside of Cook’s distance (the red dashed lines) then it is considered to be an influential observation. So this data set contains too many influential observations.

1. Here R-Squared value is equal to 0.463 but there exists high multicollinearity here hence excluding the variables with high multicollinearity and applying OLS to the excluded highly multi-correlated variable.

**Applying OLS to the excluded variable data:**

import statsmodels.api as sm

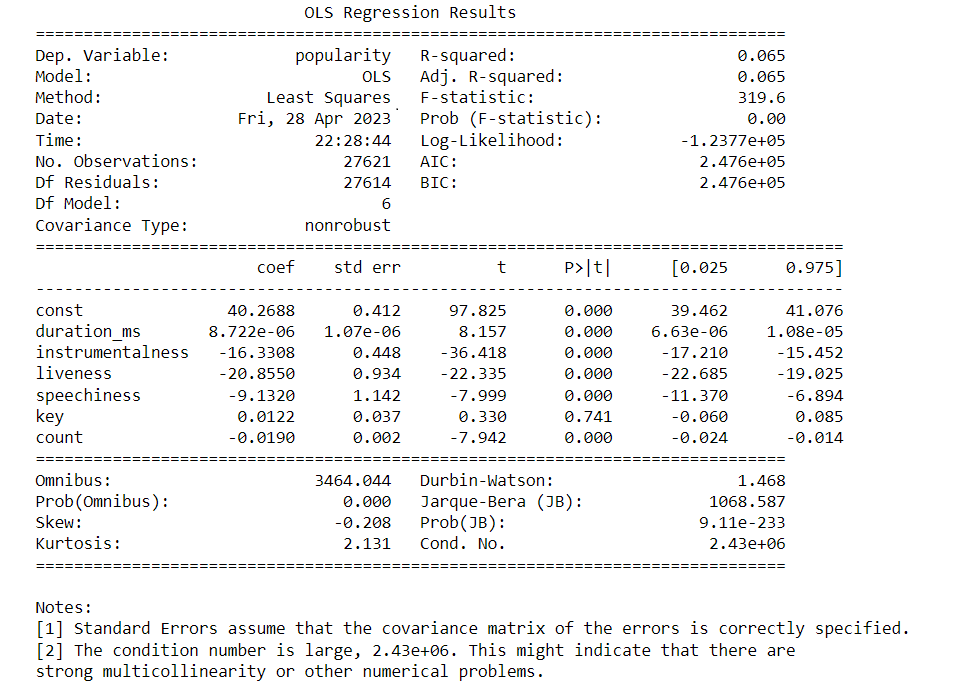
y1=data[['popularity']]

x1=data[['duration\_ms','instrumentalness','liveness','speechiness','key','count']]

x1=sm.add\_constant(x1)

model=sm.OLS(y1,x1).fit()

print(model.summary())

****

**Calculating the PACF for reduction of dimensionality:**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.preprocessing import scale

from sklearn import model\_selection

from sklearn.model\_selection import RepeatedKFold

from sklearn.model\_selection import train\_test\_split

from sklearn.decomposition import PCA

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error

#define predictor and res"mpg", "disp", "drat", "wt", "qsec"]]

X=data[['acousticness','danceability','duration\_ms','energy','instrumentalness','liveness','loudness','speechiness','tempo','valence','key','count']]

y=data[['popularity']]

**#scale predictor variables**

pca = PCA()

X\_reduced = pca.fit\_transform(scale(X))

#define cross validation method

cv = RepeatedKFold(n\_splits=10, n\_repeats=3, random\_state=1)

regr = LinearRegression()

mse = []

**# Calculate MSE with only the intercept**

score = -1\*model\_selection.cross\_val\_score(regr,

np.ones((len(X\_reduced),1)), y, cv=cv,

scoring='neg\_mean\_squared\_error').mean()

mse.append(score)

**# Calculate MSE using cross-validation, adding one component at a time**

for i in np.arange(1, 6):

score = -1\*model\_selection.cross\_val\_score(regr,

X\_reduced[:,:i], y, cv=cv, scoring='neg\_mean\_squared\_error').mean()

mse.append(score)

# Plot cross-validation results

plt.plot(mse)

plt.xlabel('Number of Principal Components')

plt.ylabel('MSE')

plt.title('popularity')

**#define predictor and res"mpg", "disp", "drat", "wt", "qsec"]]**

X=data[['acousticness','danceability','duration\_ms','energy','instrumentalness','liveness','loudness','speechiness','tempo','valence','key','count']]

y=data[['popularity']]

**#scale predictor variables**

pca = PCA()

X\_reduced = pca.fit\_transform(scale(X))

**#define cross validation method**

cv = RepeatedKFold(n\_splits=10, n\_repeats=3, random\_state=1)

regr = LinearRegression()

mse = []

**# Calculate MSE with only the intercept**

score = -1\*model\_selection.cross\_val\_score(regr,

np.ones((len(X\_reduced),1)), y, cv=cv,

scoring='neg\_mean\_squared\_error').mean()

mse.append(score**)**

**# Calculate MSE using cross-validation, adding one component at a time**

for i in np.arange(1, 6):

score = -1\*model\_selection.cross\_val\_score(regr,

X\_reduced[:,:i], y, cv=cv, scoring='neg\_mean\_squared\_error').mean()

mse.append(score)

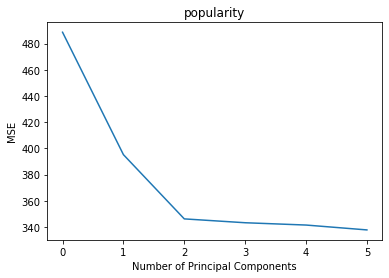
**# Plot cross-validation results**

plt.plot(mse)

plt.xlabel('Number of Principal Components')

plt.ylabel('MSE')

plt.title('popularity')

****

**np.cumsum(np.round(pca.explained\_variance\_ratio\_, decimals=4)\*100)**

array([29.36, 40.09, 49.84, 58.51, 66.79, 74.82, 81.89, 88.29, 93.81,

96.74, 99.13, 99.99])

**Splitting the data into train and test data**

#split the dataset into training (70%) and testing (30%) sets

X\_train,X\_test,y\_train,y\_test = train\_test\_split(X,y,test\_size=0.3,random\_state=0)

#scale the training and testing data

X\_reduced\_train = pca.fit\_transform(scale(X\_train))

X\_reduced\_test = pca.transform(scale(X\_test))[:,:1]

#train PCR model on training data

regr = LinearRegression()

regr.fit(X\_reduced\_train[:,:1], y\_train)

#calculate RMSE

pred = regr.predict(X\_reduced\_test)

np.sqrt(mean\_squared\_error(y\_test, pred))

19.98203346641746

Conclusion:

Based on a rule of thumb, it can be said that RMSE values between

0.2 and 0.5 show that the model can relatively predict the data

accurately. In addition, an Adjusted R-squared of more than 0.75 is a very good value for showing accuracy.

But here the fitted model is showing RMSE 19.982 which is greater than 1 and does not lie in the above range of specified RMSE. And also the R-squared for the model including the less multicorrelated variable is very low which is equal to 0.065.

Hence, the model is highly inadequate which implies the model which we have fitted is not that sufficient or able to describe the variation in the data set taken. 46% of variation is explained by the model fitted with the highly multicorrelated whereas the variation explained by the

Data with removed multi-correlated variables is 6.5%

**Steps follow while doing this model fitting and data analysis:**

* Importing data
* Checking for data shape
* Checking for missing value
* Checking total null Values count in the data
* Checking the data structure
* Checking collinearity between the data variables
* Heatmap for checking correlation between dependent and independent variables
* Determining multicollinearity between independent variables
* Applying the OLS i.e Ordinary Least square method to find the parameter estimates
* Applying OLS to the excluded variable data
* Calculating the PACF for reduction of dimensionality

**Summary:**

While fitting this model we observe from the heatmap that our data is having less correlation with the response variable. And correlation highly exists in independent variables. We calculated VIF to check multicollinearity and removed the variables having greater multicollinearity i.e. with a VIF greater than 5 and applied the PCA method to reduce the dimensionality. At last, we used the OLS to find the estimates of the parameters.