Logistic Model Training

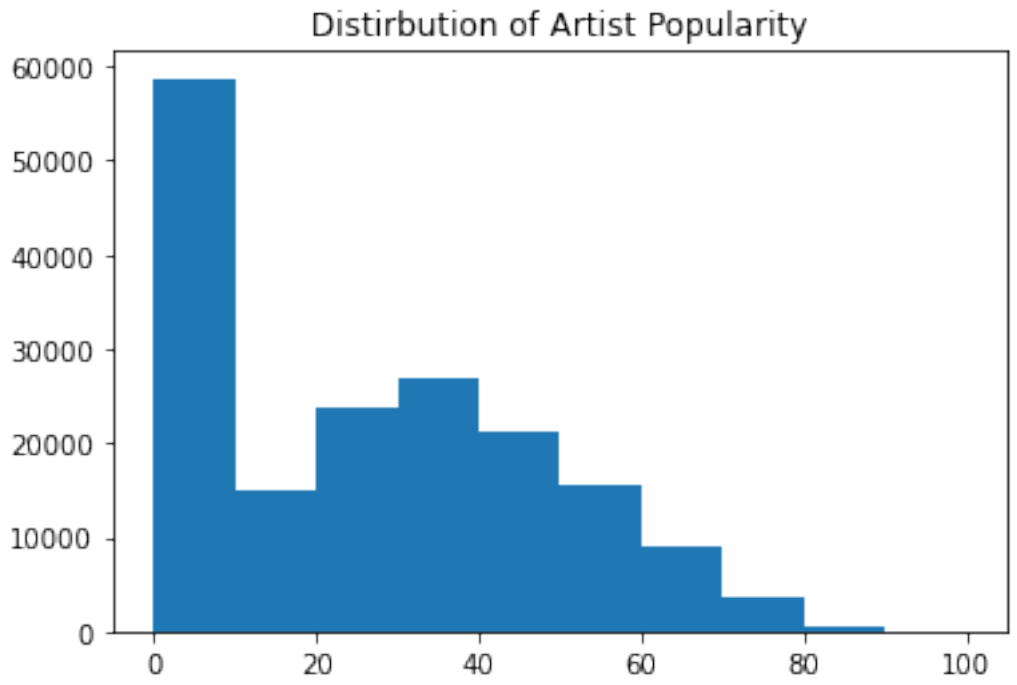
May 11, 2021

[39]:
```
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

[ ]:
```

```

[40]:
```
#load final data set as spotify
spotify = pd.read_csv("data.csv")
spotify.head(5)

#calculate the popularity mean so we can make popularity a binary variable in␣
 ↪following steps
popularity_mean = np.mean(spotify['popularity'])
popularity_mean

plt.hist(spotify['popularity'])
plt.title('Distirbution of Artist Popularity')
spotify['popularity'].describe()
```

[40]:
```
count    174389.000000
mean         25.693381
std          21.872740
min           0.000000
25%           1.000000
50%          25.000000
75%          42.000000
max         100.000000
Name: popularity, dtype: float64
```



[41]:
```
#Going to drop the name, releasase date, id since it is not significant to the␣
 ↪dependant variable, and we have season in lieu of release date
spotify = spotify.drop(['name', 'artists', 'release_date', 'id'], axis = 1)
#dummy variable for season
season_dum = pd.get_dummies(spotify['Season'])
spotify['Season'] = season_dum
```

[42]:
```
#see the scale of each parameter to see the need to scale them.
spotify.describe()
```

[42]:

	Unnamed: 0	acousticness	danceability	duration_ms	\
count	174389.000000	174389.000000	174389.000000	1.743890e+05	
mean	87194.000000	0.499228	0.536758	2.328100e+05	
std	50341.912384	0.379936	0.176025	1.483958e+05	
min	0.000000	0.000000	0.000000	4.937000e+03	
25%	43597.000000	0.087700	0.414000	1.661330e+05	
50%	87194.000000	0.517000	0.548000	2.057870e+05	
75%	130791.000000	0.895000	0.669000	2.657200e+05	
max	174388.000000	0.996000	0.988000	5.338302e+06	

	energy	explicit	instrumentalness	key	\
count	174389.000000	174389.000000	174389.000000	174389.000000	
mean	0.482721	0.068135	0.197252	5.205305	
std	0.272685	0.251978	0.334574	3.518292	

```
min         0.000000       0.000000          0.000000       0.000000
25%         0.249000       0.000000          0.000000       2.000000
50%         0.465000       0.000000          0.000524       5.000000
75%         0.711000       0.000000          0.252000       8.000000
max         1.000000       1.000000          1.000000      11.000000

           liveness       loudness  …         Season    Name Length  \
count  174389.000000  174389.000000  …  174389.000000  174389.000000
mean        0.211123     -11.750865  …       0.151403       4.673099
std         0.180493       5.691591  …       0.358442       3.301915
min         0.000000     -60.000000  …       0.000000       1.000000
25%         0.099200     -14.908000  …       0.000000       2.000000
50%         0.138000     -10.836000  …       0.000000       4.000000
75%         0.270000      -7.499000  …       0.000000       6.000000
max         1.000000       3.855000  …       1.000000      44.000000

               live           love            mix            no  \
count  174389.000000  174389.000000  174389.000000  174389.000000
mean        0.029870       0.036034       0.024766       0.027261
std         0.172105       0.190515       0.169057       0.174373
min         0.000000       0.000000       0.000000       0.000000
25%         0.000000       0.000000       0.000000       0.000000
50%         0.000000       0.000000       0.000000       0.000000
75%         0.000000       0.000000       0.000000       0.000000
max         2.000000       4.000000       3.000000       3.000000

                 op         remast        version         year_y
count  174389.000000  174389.000000  174389.000000  174389.000000
mean        0.024503       0.066145       0.024451       0.023688
std         0.155160       0.248583       0.155371       0.152942
min         0.000000       0.000000       0.000000       0.000000
25%         0.000000       0.000000       0.000000       0.000000
50%         0.000000       0.000000       0.000000       0.000000
75%         0.000000       0.000000       0.000000       0.000000
max         3.000000       2.000000       2.000000       2.000000

[8 rows x 27 columns]
```

[43]:
```
#need to drop first unname coloum since it is just an index. Need to Re-scale␣
↪features since it ranges from 0-1 to 10.
spotify = spotify.drop('Unnamed: 0', 1)

#import minmaxscaler
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
```

[44]:
```
spotify.dtypes
```

[44]:
```
acousticness        float64
danceability        float64
duration_ms           int64
energy              float64
explicit              int64
instrumentalness    float64
key                   int64
liveness            float64
loudness            float64
mode                  int64
popularity            int64
speechiness         float64
tempo               float64
valence             float64
year_x                int64
Collaboration         int64
Season                uint8
Name Length           int64
live                  int64
love                  int64
mix                   int64
no                    int64
op                    int64
remast                int64
version               int64
year_y                int64
dtype: object
```

[45]:
```
#all numerical variable that needs to be scaled
numerical_var = ['acousticness', 'danceability',␣
 ↪'duration_ms','energy','instrumentalness','key', 'liveness',␣
 ↪'loudness','speechiness','tempo','valence']

#rescale all numerical variables
spotify[numerical_var] = scaler.fit_transform(spotify[numerical_var])
spotify
```

[45]:

	acousticness	danceability	duration_ms	energy	explicit	\
0	0.994980	0.605263	0.030637	0.224	0	
1	0.645582	0.862348	0.027237	0.517	0	
2	0.996988	0.654858	0.029792	0.186	0	
3	0.000174	0.738866	0.078215	0.798	0	
4	0.296185	0.712551	0.030054	0.707	1	
…	…	…	…	…	…	
174384	0.009207	0.801619	0.026752	0.866	0	
174385	0.798193	0.434211	0.026209	0.211	0	
174386	0.809237	0.679150	0.039977	0.589	0	

```
174387      0.923695      0.467611     0.044824   0.240         1
174388      0.239960      0.685223     0.036145   0.460         0

        instrumentalness       key  liveness  loudness  mode  …  Season  \
0               0.000522  0.454545    0.3790  0.741868     0  …       0
1               0.026400  0.454545    0.0809  0.825918     0  …       0
2               0.000018  0.000000    0.5190  0.750168     1  …       0
3               0.801000  0.181818    0.1280  0.825135     1  …       0
4               0.000246  0.909091    0.4020  0.845102     0  …       1
…                    …         …         …         …   …  …     …
174384          0.000060  0.545455    0.1780  0.859933     0  …       0
174385          0.000000  0.363636    0.1960  0.756949     1  …       0
174386          0.920000  0.363636    0.1130  0.745549     0  …       0
174387          0.000000  0.000000    0.1130  0.750497     1  …       0
174388          0.891000  0.636364    0.2150  0.747992     1  …       0

        Name Length  live  love  mix  no  op  remast  version  year_y
0                 6     0     0    0   0   0       0        0       0
1                 6     0     0    0   0   0       0        0       0
2                 2     0     0    0   0   0       0        0       0
3                10     0     0    0   0   0       0        0       0
4                 1     0     0    0   0   0       0        0       0
…               …    …   …  …   ..  ..      …       …      …
174384            2     0     0    0   0   0       0        0       0
174385            3     0     0    0   0   0       0        0       0
174386            1     0     0    0   0   0       0        0       0
174387            2     0     0    0   0   0       0        0       0
174388            1     0     0    0   0   0       0        0       0

[174389 rows x 26 columns]
```

[46]:
```
#modify dataframe to treat popularity as a binary variable
def popularity(c):
    if c['popularity'] > popularity_mean:
        return 1
    else:
        return 0
```

[47]:
```
#set popularity as a binary column: 1 means popular 0 means unpopular
spotify['popularity'] = spotify.apply(popularity, axis = 1)
```

[48]:
```
#split test and training data 30% test, 70% training set

from sklearn.model_selection import train_test_split

spotify_train, spotify_test = train_test_split(spotify, test_size=0.3,␣
 ↪random_state=88)
```

```
spotify_train.shape, spotify_test.shape
```

[48]:
```
((122072, 26), (52317, 26))
```

[49]:
```
#compose the x and y corresponding data for train and test set
x_train = spotify_train.drop(['popularity'], axis =1)
y_train = spotify_train[['popularity']]

x_test = spotify_test.drop(['popularity'], axis =1)
y_test = spotify_test[['popularity']]
```

[50]:
```
import statsmodels.api as smf
#building model and fitting data
log_reg = smf.Logit(y_train, x_train).fit()

#visualize summary.
# define p value cut off as 0.05.
print(log_reg.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.523117
         Iterations 7
                           Logit Regression Results
==============================================================================
Dep. Variable:             popularity   No. Observations:               122072
Model:                          Logit   Df Residuals:                   122047
Method:                           MLE   Df Model:                           24
Date:                Thu, 06 May 2021   Pseudo R-squ.:                  0.2453
Time:                        12:32:52   Log-Likelihood:                -63858.
converged:                       True   LL-Null:                       -84613.
Covariance Type:            nonrobust   LLR p-value:                     0.000
==============================================================================
====
                         coef    std err          z      P>|z|      [0.025
0.975]
------------------------------------------------------------------------------
----
acousticness        -1.9143      0.030    -63.706      0.000      -1.973
-1.855
danceability         0.2130      0.052      4.132      0.000       0.112
0.314
duration_ms          3.2051      0.270     11.875      0.000       2.676
3.734
energy               1.0990      0.057     19.443      0.000       0.988
1.210
explicit             1.3494      0.039     34.382      0.000       1.272
1.426
```

```
instrumentalness    -1.6910      0.025    -68.284      0.000      -1.739
-1.642
key                 -0.0286      0.022     -1.318      0.188      -0.071
0.014
liveness            -0.7297      0.043    -16.839      0.000      -0.815
-0.645
loudness            -4.4954      0.142    -31.719      0.000      -4.773
-4.218
mode                 0.0784      0.015      5.095      0.000       0.048
0.109
speechiness         -4.4992      0.069    -64.848      0.000      -4.635
-4.363
tempo               -0.1107      0.058     -1.904      0.057      -0.225
0.003
valence             -0.3782      0.035    -10.793      0.000      -0.447
-0.310
year_x               0.0025   5.23e-05     48.004      0.000       0.002
0.003
Collaboration       -0.4504      0.019    -23.680      0.000      -0.488
-0.413
Season               0.4875      0.020     24.652      0.000       0.449
0.526
Name Length         -0.0529      0.003    -19.521      0.000      -0.058
-0.048
live                -0.1924      0.044     -4.340      0.000      -0.279
-0.106
love                 0.1965      0.035      5.622      0.000       0.128
0.265
mix                 -1.9295      0.053    -36.168      0.000      -2.034
-1.825
no                   0.2276      0.061      3.702      0.000       0.107
0.348
op                  -0.5054      0.075     -6.733      0.000      -0.653
-0.358
remast               0.0056      0.028      0.202      0.840      -0.049
0.060
version             -0.0153      0.043     -0.358      0.720      -0.099
0.068
year_y              -2.6546      0.065    -41.124      0.000      -2.781
-2.528
==============================================================================
====
```

[51]:
```
#p-value for most are lower than cutoff of 0.05. Only four are above: key,␣
 ↪tempo, remast, version
#retrain model without these four variables
x_train = x_train.drop(['key', 'tempo', 'remast', 'version'], 1)
```

```
x_train
```

[51]:

	acousticness	danceability	duration_ms	energy	explicit	\
67798	0.439759	0.750000	0.030334	0.5940	0	
170292	0.017470	0.797571	0.041807	0.4700	0	
57359	0.017269	0.813765	0.054199	0.5300	0	
21990	0.914659	0.655870	0.034064	0.1320	0	
13990	0.304217	0.673077	0.055217	0.8990	0	
…	…	…	…	…	…	
90474	0.104418	0.659919	0.052849	0.6300	1	
133553	0.135542	0.397773	0.036222	0.4190	0	
36815	0.231928	0.459514	0.065168	0.5340	0	
104736	0.158635	0.336032	0.057936	0.2790	0	
124888	0.953815	0.178138	0.033827	0.0302	0	

	instrumentalness	liveness	loudness	mode	speechiness	…	year_x	\
67798	0.771000	0.1020	0.803727	1	0.038929	…	1976	
170292	0.723000	0.1830	0.727194	1	0.046138	…	2020	
57359	0.010200	0.1110	0.705677	1	0.077240	…	2020	
21990	0.000000	0.2680	0.703735	1	0.043872	…	1933	
13990	0.000002	0.4450	0.818260	1	0.090319	…	1990	
…	…	…	…	…	…	…	…	
90474	0.000000	0.0299	0.819137	0	0.367662	…	2003	
133553	0.048100	0.0852	0.751343	1	0.029866	…	1973	
36815	0.818000	0.1370	0.788991	0	0.040268	…	2016	
104736	0.158000	0.0838	0.695310	1	0.030999	…	1992	
124888	0.748000	0.0943	0.546410	1	0.042122	…	2011	

	Collaboration	Season	Name Length	live	love	mix	no	op	year_y
67798	0	0	1	0	0	0	0	0	0
170292	0	0	3	0	0	0	0	0	0
57359	0	1	4	0	0	0	0	0	0
21990	0	0	7	0	0	0	0	0	0
13990	0	1	4	0	0	0	0	0	0
…	…	…	…	…	…	..	..	…	
90474	1	0	5	0	0	0	0	0	0
133553	0	0	9	0	0	0	0	0	0
36815	0	1	1	0	0	0	0	0	0
104736	0	0	1	0	0	0	0	0	0
124888	1	1	9	0	0	0	0	0	0

[122072 rows x 21 columns]

[52]:
```
#train model once again with updated x_train
log_reg = smf.Logit(y_train, x_train).fit()
```

```
#visualize summary.
# define p value cut off as 0.05.
print(log_reg.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.523140
         Iterations 7
                           Logit Regression Results
==============================================================================
Dep. Variable:             popularity   No. Observations:               122072
Model:                          Logit   Df Residuals:                   122051
Method:                           MLE   Df Model:                           20
Date:                Thu, 06 May 2021   Pseudo R-squ.:                  0.2453
Time:                        13:28:16   Log-Likelihood:                -63861.
converged:                       True   LL-Null:                       -84613.
Covariance Type:            nonrobust   LLR p-value:                     0.000
==============================================================================
====
                       coef    std err          z      P>|z|      [0.025
0.975]
------------------------------------------------------------------------------
----
acousticness        -1.9110      0.030    -63.901      0.000      -1.970
-1.852
danceability         0.2273      0.051      4.468      0.000       0.128
0.327
duration_ms          3.2110      0.270     11.898      0.000       2.682
3.740
energy               1.0932      0.056     19.368      0.000       0.983
1.204
explicit             1.3519      0.039     34.449      0.000       1.275
1.429
instrumentalness    -1.6910      0.025    -68.284      0.000      -1.739
-1.642
liveness            -0.7265      0.043    -16.783      0.000      -0.811
-0.642
loudness            -4.5016      0.142    -31.771      0.000      -4.779
-4.224
mode                 0.0805      0.015      5.270      0.000       0.051
0.110
speechiness         -4.5068      0.069    -65.126      0.000      -4.642
-4.371
valence             -0.3889      0.035    -11.233      0.000      -0.457
-0.321
year_x               0.0025   5.02e-05     49.352      0.000       0.002
0.003
Collaboration       -0.4499      0.019    -23.785      0.000      -0.487
```

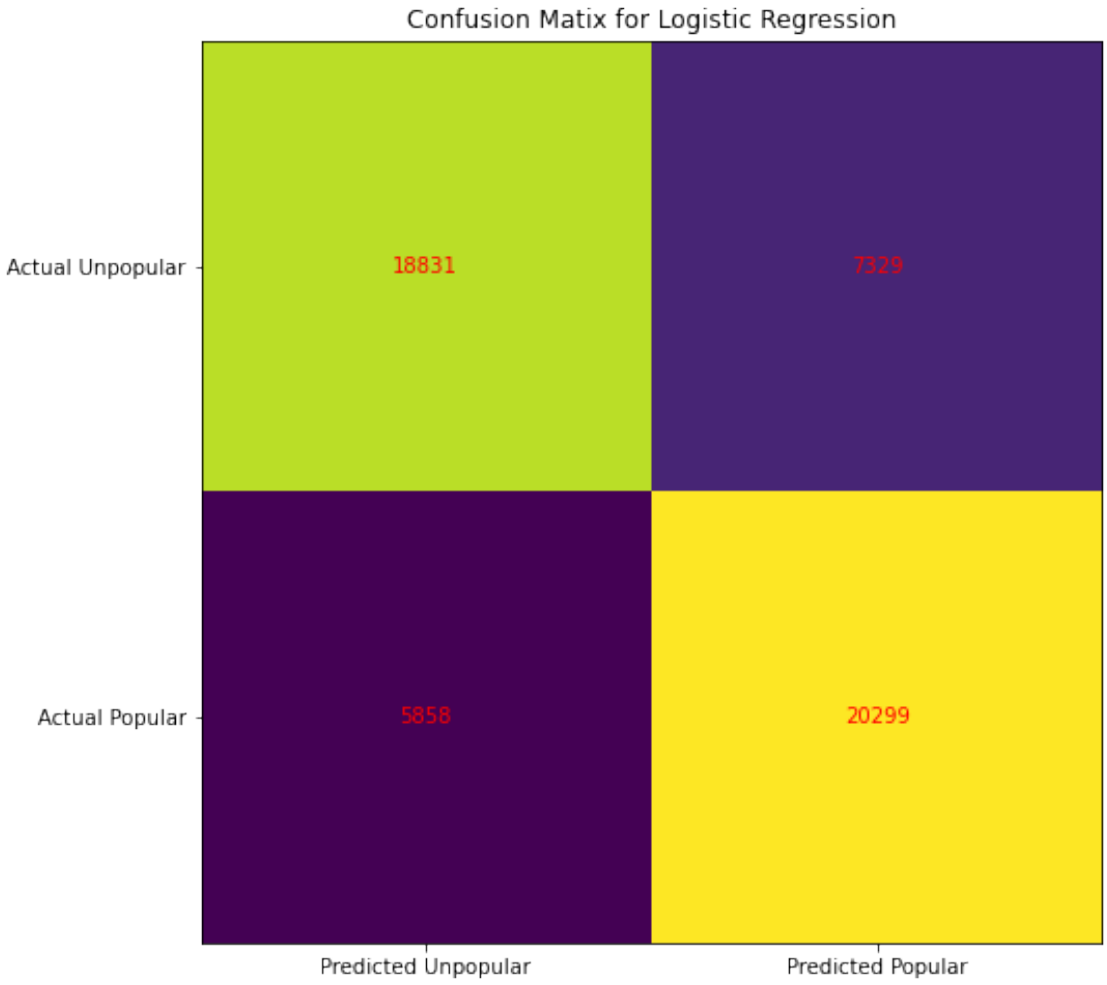
```
-0.413
Season                0.4872      0.020      24.663      0.000       0.448
0.526
Name Length          -0.0529      0.003     -20.559      0.000      -0.058
-0.048
live                 -0.1924      0.044      -4.355      0.000      -0.279
-0.106
love                  0.1964      0.035       5.620      0.000       0.128
0.265
mix                  -1.9321      0.053     -36.340      0.000      -2.036
-1.828
no                    0.2278      0.061       3.712      0.000       0.108
0.348
op                   -0.5038      0.075      -6.715      0.000      -0.651
-0.357
year_y               -2.6560      0.065     -41.162      0.000      -2.783
-2.530
==============================================================================
====
```

[53]:
```
#define y_hat as the predicted value of y for test dataset
#drop the four coloumns that we dropped for x_train on x_test
x_test = x_test.drop(['key', 'tempo', 'remast', 'version'], 1)
y_hat = log_reg.predict(x_test)
prediction = list(map(round, y_hat))

from sklearn.metrics import (confusion_matrix,
                             accuracy_score)

#confusion matrix
cm = confusion_matrix(y_test, prediction)
print("Confusion matrixx : \n", cm)
```

```
Confusion matrixx :
 [[18831  7329]
 [ 5858 20299]]
```

[54]:
```
fig, ax = plt.subplots(figsize=(8, 8))
ax.imshow(cm)
ax.grid(False)
ax.xaxis.set(ticks=(0, 1), ticklabels=('Predicted Unpopular', 'Predicted␣
 ↪Popular'))
ax.yaxis.set(ticks=(0, 1), ticklabels=('Actual Unpopular', 'Actual Popular'))
ax.set_ylim(1.5, -0.5)
for i in range(2):
    for j in range(2):
        ax.text(j, i, cm[i, j], ha='center', va='center', color='red')
```

```
plt.title('Confusion Matix for Logistic Regression')
plt.show()
```



[55]:
```
#accuracy of the model
print('Test Accuracy = ', accuracy_score(y_test, prediction))
```

```
Test Accuracy =  0.7479404400099394
```

[ ]:

[ ]: