Cogs109FinalCode

June 15, 2023

[2]:
```
#the usuals..
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import math

#for logistic regression and for kfold
from statsmodels.formula.api import logit
import statsmodels.formula.api as smf
from sklearn.model_selection import KFold
import sklearn.metrics as metrics
from sklearn.metrics import mean_squared_error
from sklearn.linear_model import LogisticRegression
from statsmodels.formula.api import ols

#this won't be necessary, but it is very useful for creating
#confusion matrix
from sklearn.metrics import confusion_matrix
```

[3]:
```
df = pd.read_csv('dataset.csv')
#df = df.loc[30002 : 30999]
df = df.drop(columns = ['track_id', 'artists', 'album_name', 'track_name',␣
 ↪'duration_ms', 'explicit', 'key', 'mode', 'time_signature', 'track_genre'])
df = df.loc[30002 : 30201]
print(df)
```

```
       Unnamed: 0  popularity  danceability  energy  loudness  speechiness  \
30002       30002          79         0.489   0.597    -6.633       0.0292
30003       30003          98         0.561   0.965    -3.673       0.0343
30004       30004          68         0.786   0.667    -8.272       0.0540
30005       30005          73         0.451   0.692    -4.741       0.0398
30006       30006          67         0.595   0.817    -6.187       0.0474
…             …          …             …       …         …            …
30197       30197           0         0.524   0.807    -4.160       0.0579
30198       30198           0         0.635   0.813    -5.098       0.0578
30199       30199           0         0.660   0.804    -6.133       0.0485
30200       30200          76         0.691   0.695    -5.600       0.0367
```

```
30201       30201          74          0.432   0.781    -4.038       0.0567

       acousticness  instrumentalness  liveness  valence    tempo
30002      0.270000          0.000000    0.1050    0.324   95.012
30003      0.003830          0.000007    0.3710    0.304  128.040
30004      0.011200          0.053000    0.0740    0.688  102.046
30005      0.287000          0.000000    0.1150    0.423  174.122
30006      0.000712          0.038600    0.1420    0.483  144.961
…               …                 …         …        …        …
30197      0.004150          0.000834    0.2120    0.620  124.127
30198      0.018700          0.000000    0.1630    0.641  123.065
30199      0.040200          0.000000    0.1290    0.359  109.000
30200      0.059200          0.000000    0.0647    0.514  106.064
30201      0.041000          0.000004    0.0789    0.197  139.432

[200 rows x 11 columns]
```

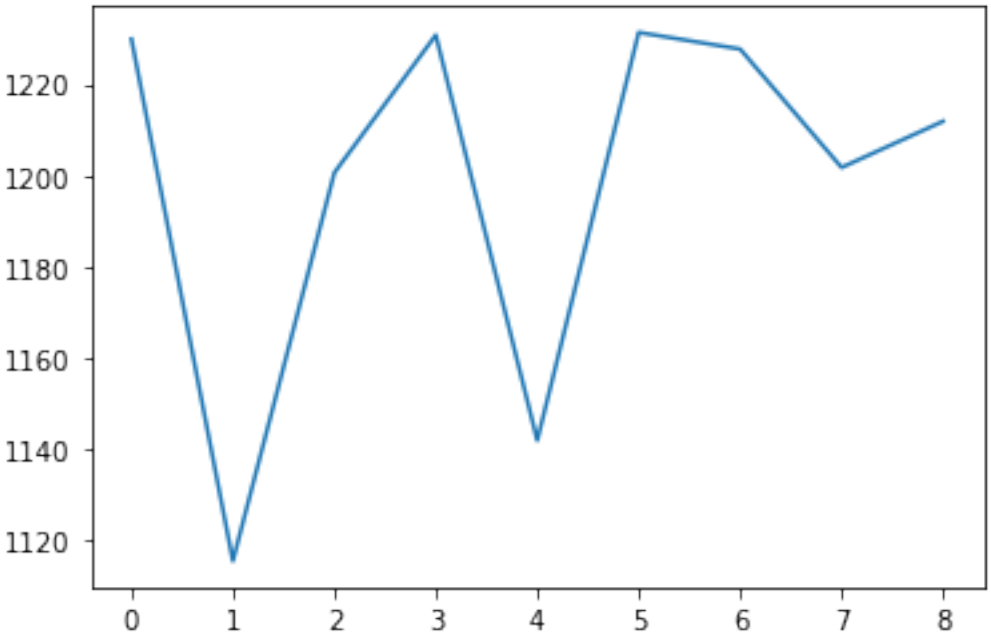
[ ]:

[4]:
```
mseArr = np.zeros(9)

#now lets loop through each column values
for n,i in enumerate(df.columns[2:].values):
    formulaThis = 'popularity~ 1+' + i
    #fit the model :D
    mdl = ols(formula =formulaThis ,data = df).fit()

    #make prediction :D
    predictor = mdl.predict(df)


    #find mse :D and put it into the mseArray
    mseArr[n] = mean_squared_error(df['popularity'], predictor)

print(mseArr)
```

```
[1229.86274593 1115.56185439 1200.56698142 1230.71628468 1141.9585713
 1231.29602833 1227.64898216 1201.71586513 1211.83805533]
```

[5]:
```
x = np.arange(9)
plt.plot(x,mseArr)
plotMin = np.argmin(mseArr)
print(plotMin)
#Thw best predictor is the first index
#therefore, energy is best predictor
```

```
1
```



[6]:
```
mdl1 = ols(formula = 'popularity ~ 1 + energy', data = df).fit()
mdl1.summary()
```

[6]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:             popularity   R-squared:                       0.095
Model:                            OLS   Adj. R-squared:                  0.090
Method:                 Least Squares   F-statistic:                     20.68
Date:                Thu, 15 Jun 2023   Prob (F-statistic):           9.46e-06
Time:                        02:46:56   Log-Likelihood:                -985.50
No. Observations:                 200   AIC:                             1975.
Df Residuals:                     198   BIC:                             1982.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept    100.2832     16.358      6.131      0.000      68.025     132.541
energy       -97.8815     21.526     -4.547      0.000    -140.331     -55.432
==============================================================================
Omnibus:                       43.679   Durbin-Watson:                   0.691
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               26.676
Skew:                           0.758   Prob(JB):                     1.61e-06
Kurtosis:                       2.051   Cond. No.                         14.2
```

```
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
"""
```

[7]:
```
mdl2 = ols(formula = 'popularity ~ 1 + danceability + energy + loudness +␣
↪liveness + speechiness + acousticness + instrumentalness + valence + tempo'␣
↪, data = df).fit()
mdl2.summary()
```

[7]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:             popularity   R-squared:                       0.161
Model:                            OLS   Adj. R-squared:                  0.121
Method:                 Least Squares   F-statistic:                     4.057
Date:                Thu, 15 Jun 2023   Prob (F-statistic):           8.94e-05
Time:                        02:46:56   Log-Likelihood:                -977.85
No. Observations:                 200   AIC:                             1976.
Df Residuals:                     190   BIC:                             2009.
Df Model:                           9
Covariance Type:            nonrobust
==============================================================================
====
                     coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
----
Intercept          124.4536     37.878      3.286      0.001      49.738
199.169
danceability         8.4369     25.301      0.333      0.739     -41.469
58.343
energy             -73.2496     29.552     -2.479      0.014    -131.542
-14.958
loudness             1.4675      1.879      0.781      0.436      -2.240
5.175
liveness            11.0032     23.443      0.469      0.639     -35.238
57.245
speechiness         -1.1714     42.309     -0.028      0.978     -84.626
82.284
acousticness        46.1068     18.695      2.466      0.015       9.230
82.984
instrumentalness   -10.1827     18.657     -0.546      0.586     -46.984
26.618
```

```
valence            -31.8265     14.126     -2.253      0.025     -59.690
-3.964
tempo               -0.2642      0.131     -2.011      0.046      -0.523
-0.005
==============================================================================
Omnibus:                       19.879   Durbin-Watson:                   0.721
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               18.511
Skew:                           0.675   Prob(JB):                     9.56e-05
Kurtosis:                       2.369   Cond. No.                     2.43e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 2.43e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
"""
```

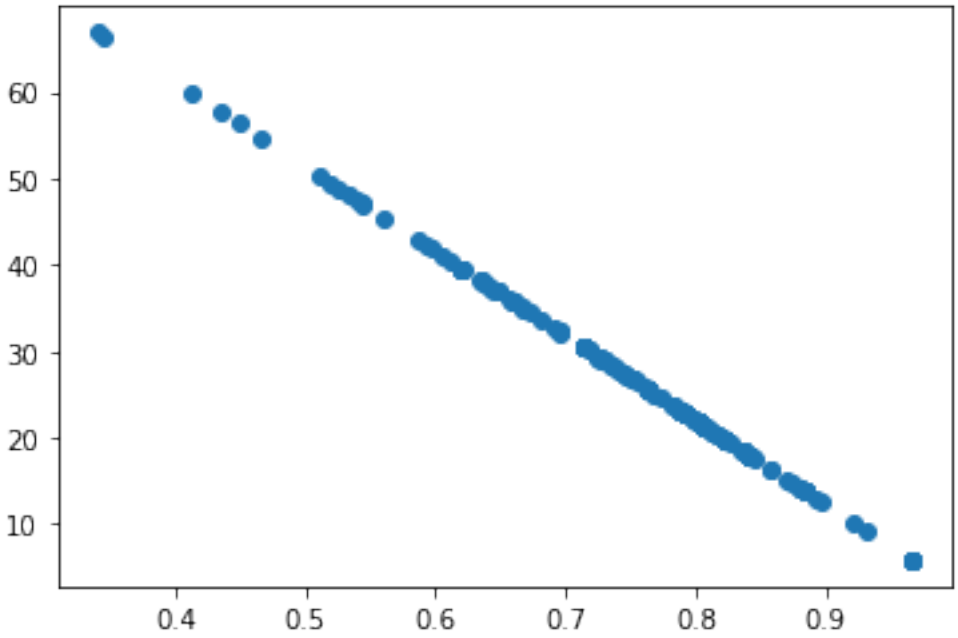
[8]:
```
data2 = {'Actual popularity': df['popularity'], 'Predicted Popularity 1␣
 ↪predictor': mdl1.predict(),
         'Predicted Popularity 9 predictors': mdl2.predict()}
df3 = pd.DataFrame(data=data2)
print(df3)
```

```
       Actual popularity  Predicted Popularity 1 predictor  \
30002                 79                          41.847930
30003                 98                           5.827537
30004                 68                          34.996224
30005                 73                          32.549187
30006                 67                          20.313999
…                    …                                …
30197                  0                          21.292814
30198                  0                          20.705525
30199                  0                          21.586458
30200                 76                          32.255542
30201                 74                          23.837733

       Predicted Popularity 9 predictors
30002                          53.273446
30003                          13.828621
30004                          21.961180
30005                          25.602064
30006                           8.028225
…                                  …
30197                          13.580959
30198                          12.453963
30199                          25.124109
```

```
30200                          30.176896
30201                          24.552007

[200 rows x 3 columns]
```

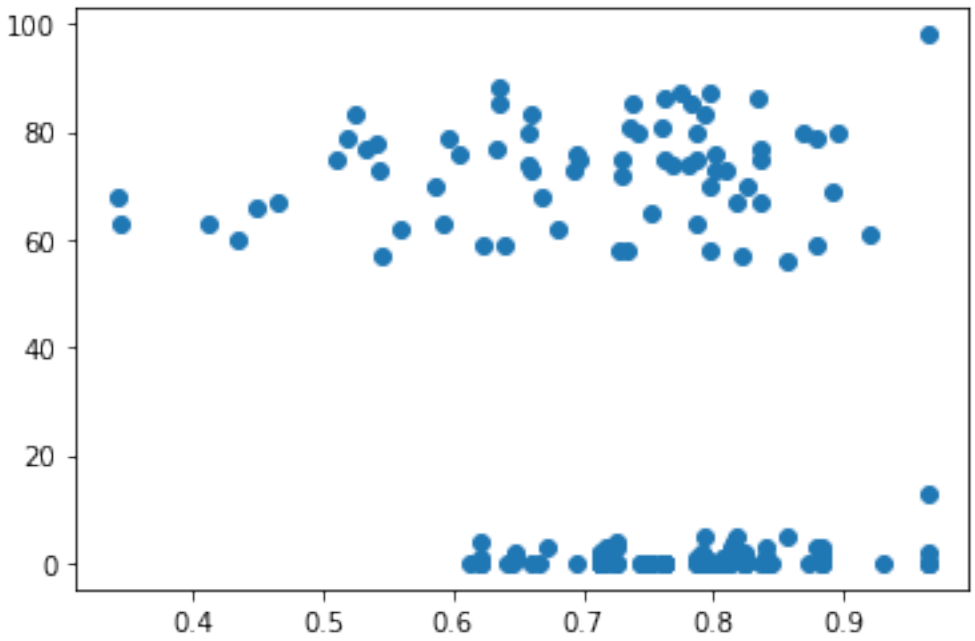
[9]:
```
#Plotting model 1 (plotting predicted probability as a function of energy)
plt.plot(df['energy'], mdl1.predict(), 'o')
```

[9]: [<matplotlib.lines.Line2D at 0x7f50c2eebf40>]



[10]:
```
#Plotting actual probability as a function of energy
plt.plot(df['energy'], df['popularity'], 'o')
```

[10]: [<matplotlib.lines.Line2D at 0x7f50c3123130>]



[19]:
```
kf = KFold(n_splits=5, random_state=109, shuffle=True)
trainMse1 = []
testMse1 = []
trainMse2 = []
testMse2 = []
formula1 = 'popularity ~ 1 + energy'
formula2 = 'popularity ~ 1 + danceability + energy + loudness + liveness +␣
 ↪speechiness + acousticness + instrumentalness + valence + tempo'
for n,(trainInd,testInd) in enumerate(kf.split(df)):
    trainDf = df.iloc[trainInd,]
    testDf = df.iloc[testInd,]
    #fit a model with your best predictor
    mdl_1Pred = ols(formula = formula1,data = df).fit()
    #fit a model with your best predictor + your second best predictor
    mdl_2Pred = ols(formula = formula2,data = df).fit()


    # ok.. now lets calculate the training and test mse seperately for these
    trainMse1.append(mean_squared_error(trainDf['popularity'],mdl_1Pred.
 ↪predict(trainDf)))
    trainMse2.append(mean_squared_error(trainDf['popularity'],mdl_2Pred.
 ↪predict(trainDf)))
```

```
    testMse1.append(mean_squared_error(testDf['popularity'],mdl_1Pred.
↪predict(testDf)))
    testMse2.append(mean_squared_error(testDf['popularity'],mdl_2Pred.
↪predict(testDf)))
```

[ ]:

[20]:
```
print("Testing MSE for best 1 predictor model : " + str(np.mean(testMse1)))
print('Testing MSE for 9 predictor model : ' +  str(np.mean(testMse2)))
```

```
Testing MSE for best 1 predictor model : 1115.5618543907894
Testing MSE for 9 predictor model : 1033.4628208998722
```

[34]:
```
xx = np.arange(200)


plt.plot(xx, df['popularity'], label = 'actual popularity', color = 'blue')
plt.plot(xx, mdl2.predict(), label = 'predicted popularity',  color = 'red')

plt.show()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/tmp/ipykernel_182/2352024282.py in <module>
      1 xx = np.arange(200)
      2
----> 3 plt.xlabel("hel")
      4 plt.ylabel('popularity')
      5 plt.plot(xx, df['popularity'], label = 'actual popularity', color =␣
↪'blue')

TypeError: 'str' object is not callable
```

[ ]:

[ ]:

[ ]:

[ ]: