spotify blended

May 12, 2021

[1]:
```python
import pandas as pd
import numpy as np
```

[2]:
```python
# import ml classifiers
from nltk.tokenize import sent_tokenize # tokenizes sentences
from nltk.stem import PorterStemmer     # parsing/stemmer
from nltk.tag import pos_tag            # parts-of-speech tagging
from nltk.corpus import wordnet         # sentiment scores
from nltk.stem import WordNetLemmatizer # stem and context
from nltk.corpus import stopwords       # stopwords
from nltk.util import ngrams
```

[3]:
```python
df = pd.read_csv('spotify_final.csv').iloc[:, 1:]
df.head()
```

[3]:
```
   acousticness                    artists  danceability  duration_ms  \
0      0.991000             ['Mamie Smith']         0.598       168333
1      0.643000  ['"Screamin Jay Hawkins"']         0.852       150200
2      0.993000             ['Mamie Smith']         0.647       163827
3      0.000173         ['Oscar Velazquez']         0.730       422087
4      0.295000                    ['Mixe']         0.704       165224

   energy  explicit                      id  instrumentalness  key  liveness  \
0   0.224         0  0cS0A1fUEUd1EW3FcF8AEI          0.000522    5    0.3790
1   0.517         0  0hbkKFIJm7Z05H8Zl9w30f          0.026400    5    0.0809
2   0.186         0  11m7laMUgmOKqI3oYzuhne          0.000018    0    0.5190
3   0.798         0  19Lc5SfJJ5O1oaxY0fpwfh          0.801000    2    0.1280
4   0.707         1  2hJjbsLCytGsnAHfdsLejp          0.000246   10    0.4020

   …  Season  Name Length live  love mix  no  op  remast  version  year_y
0  …     NaN            6    0     0   0   0   0       0        0       0
1  …  Winter            6    0     0   0   0   0       0        0       0
2  …     NaN            2    0     0   0   0   0       0        0       0
3  …  Winter           10    0     0   0   0   0       0        0       0
4  …    Fall            1    0     0   0   0   0       0        0       0

[5 rows x 30 columns]
```

[4]:
```
df.shape
```

[4]:
```
(174389, 30)
```

[5]:
```
df.describe()
```

[5]:
```
        acousticness   danceability   duration_ms         energy  \
count  174389.000000  174389.000000  1.743890e+05  174389.000000
mean        0.499228       0.536758  2.328100e+05       0.482721
std         0.379936       0.176025  1.483958e+05       0.272685
min         0.000000       0.000000  4.937000e+03       0.000000
25%         0.087700       0.414000  1.661330e+05       0.249000
50%         0.517000       0.548000  2.057870e+05       0.465000
75%         0.895000       0.669000  2.657200e+05       0.711000
max         0.996000       0.988000  5.338302e+06       1.000000

            explicit  instrumentalness            key       liveness  \
count  174389.000000     174389.000000  174389.000000  174389.000000
mean        0.068135          0.197252       5.205305       0.211123
std         0.251978          0.334574       3.518292       0.180493
min         0.000000          0.000000       0.000000       0.000000
25%         0.000000          0.000000       2.000000       0.099200
50%         0.000000          0.000524       5.000000       0.138000
75%         0.000000          0.252000       8.000000       0.270000
max         1.000000          1.000000      11.000000       1.000000

            loudness           mode  …  Collaboration    Name Length  \
count  174389.000000  174389.000000  …  174389.000000  174389.000000
mean      -11.750865       0.702384  …       0.215392       4.673099
std         5.691591       0.457211  …       0.411095       3.301915
min       -60.000000       0.000000  …       0.000000       1.000000
25%       -14.908000       0.000000  …       0.000000       2.000000
50%       -10.836000       1.000000  …       0.000000       4.000000
75%        -7.499000       1.000000  …       0.000000       6.000000
max         3.855000       1.000000  …       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 25 columns]
```

[6]:
```
df.columns
```

[6]:
```
Index(['acousticness', 'artists', 'danceability', 'duration_ms', 'energy',
       'explicit', 'id', 'instrumentalness', 'key', 'liveness', 'loudness',
       'mode', 'name', 'popularity', 'release_date', 'speechiness', 'tempo',
       'valence', 'year_x', 'Collaboration', 'Season', 'Name Length', 'live',
       'love', 'mix', 'no', 'op', 'remast', 'version', 'year_y'],
      dtype='object')
```

[7]:
```
# Delete some of the columns
del df['id']
del df['release_date']
del df['artists']
```

[8]:
```
# Change some of the column names
df = df.rename(columns={'Name Length': 'name_length'})
df.head()
```

[8]:
```
   acousticness  danceability  duration_ms  energy  explicit  \
0      0.991000         0.598       168333   0.224         0
1      0.643000         0.852       150200   0.517         0
2      0.993000         0.647       163827   0.186         0
3      0.000173         0.730       422087   0.798         0
4      0.295000         0.704       165224   0.707         1

   instrumentalness  key  liveness  loudness  mode  …  Season  name_length  \
0          0.000522    5    0.3790   -12.628     0  …     NaN            6
1          0.026400    5    0.0809    -7.261     0  …  Winter            6
2          0.000018    0    0.5190   -12.098     1  …     NaN            2
3          0.801000    2    0.1280    -7.311     1  …  Winter           10
4          0.000246   10    0.4020    -6.036     0  …    Fall            1

   live  love  mix  no  op remast  version  year_y
0     0     0    0   0   0      0        0       0
1     0     0    0   0   0      0        0       0
2     0     0    0   0   0      0        0       0
```

```
3      0      0     0    0    0       0          0        0
4      0      0     0    0    0       0          0        0

[5 rows x 27 columns]
```

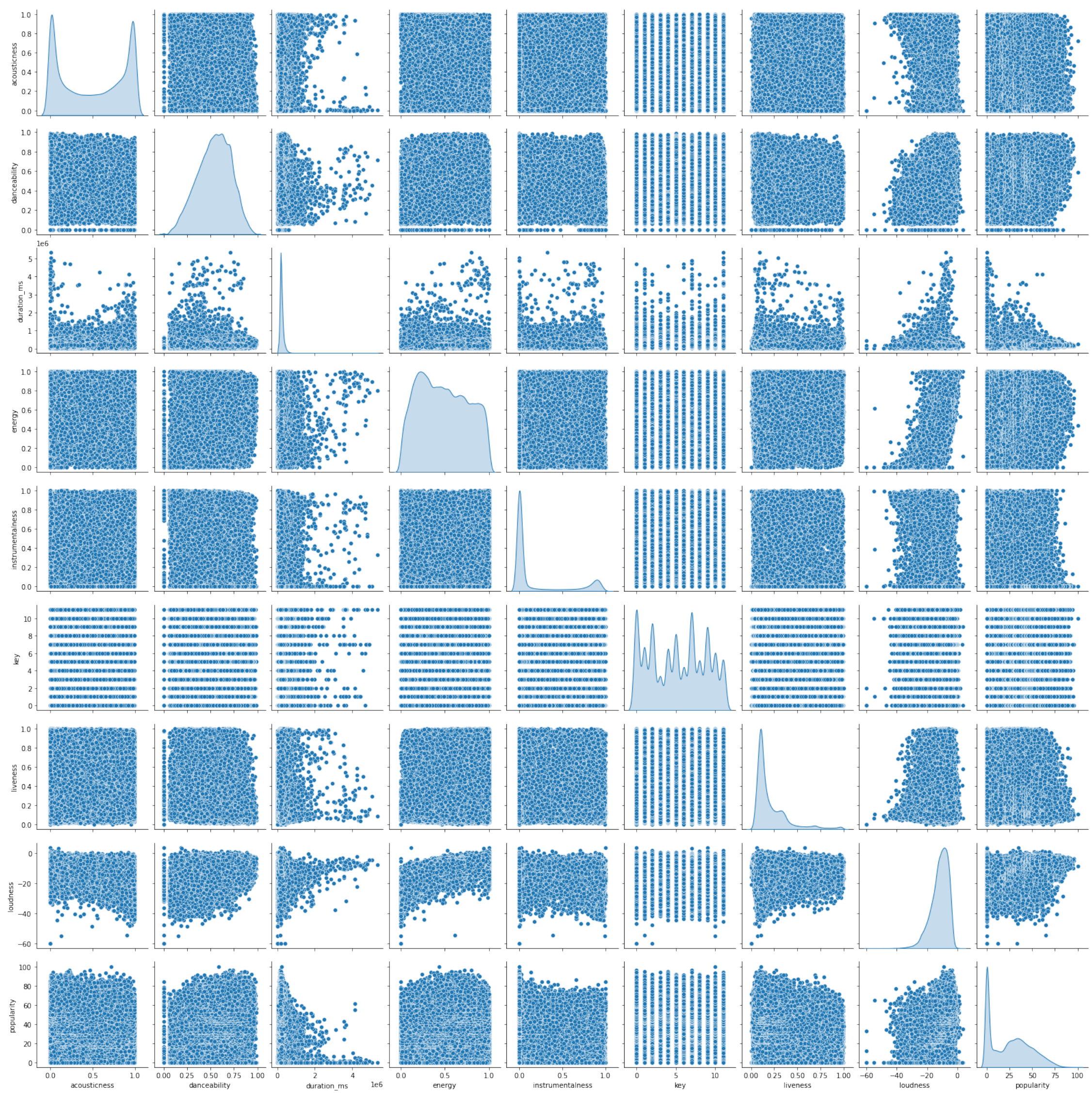
[9]:
```
# Get the dummy variables for season
#dummy variable for season
season_dum = pd.get_dummies(df['Season'])
df['Season'] = season_dum
```

[10]:
```
# Create the final dataset
df["Season"]
```

[10]:
```
0         0
1         0
2         0
3         0
4         1
         ..
174384    0
174385    0
174386    0
174387    0
174388    0
Name: Season, Length: 174389, dtype: uint8
```

[11]:
```
# Plot scatter matrix for each pair of variables off diagonal and the␣
 ↪histograms (or density plots) on the diagonal
# In ggplot2 in R, one can use ggscatmat, which also prints the correlation in␣
 ↪the upper triangle.
import seaborn as sns

cols = ['acousticness', 'danceability', 'duration_ms', 'energy',
       'instrumentalness', 'key', 'liveness', 'loudness','popularity']
sns.pairplot(df[cols],diag_kind='kde')
```

[11]:
```
<seaborn.axisgrid.PairGrid at 0x7f47ec865b50>
```



0.1 Building CART Model

0.1.1 Predict if the song is good

- A song is good if its popularity is greater than 25

[12]:
```python
from sklearn.model_selection import train_test_split
cols = ['acousticness', 'danceability', 'duration_ms', 'energy','explicit',
       'instrumentalness', 'key', 'liveness', 'loudness', 'mode', 'speechiness',
       'tempo', 'valence', 'year_x', 'Collaboration', 'name_length', 'live',
        'live','love', 'mix', 'no', 'op', 'remast', 'version', 'year_y',
       'Season', 'popularity']

# re-split the dataset into training and testing data
```

```
y = df['popularity']
X = df[cols]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33,␣
↪random_state=88)
blend_df = X_test.reset_index()[cols]
X_train = X_train.drop(columns=["popularity"])
X_test = X_test.drop(columns=["popularity"])

X_train.shape, X_test.shape
```

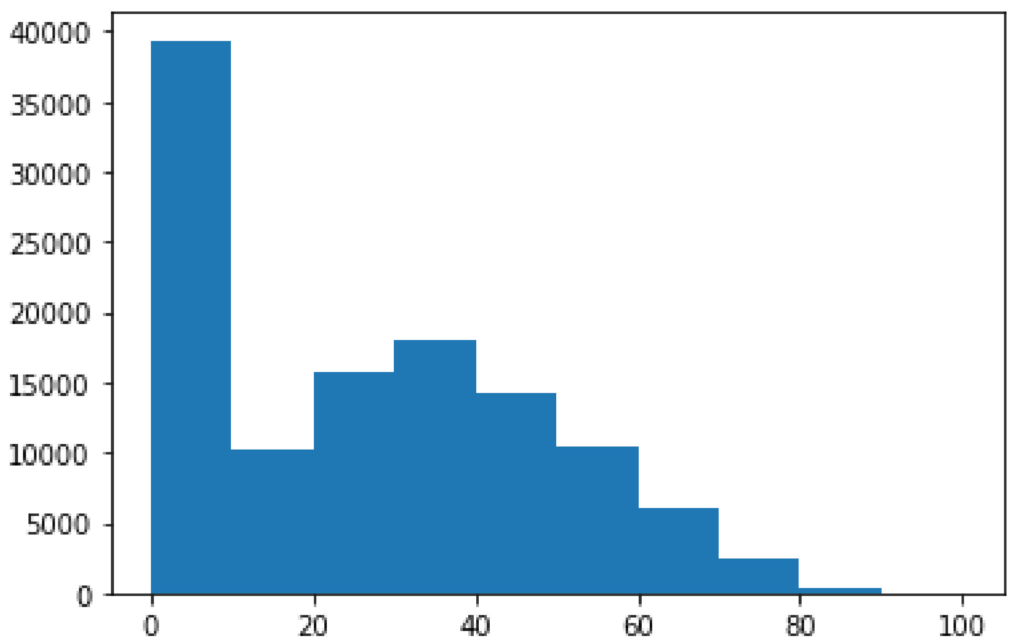
[12]: ((116840, 26), (57549, 26))

[13]:
```
y_train.mean(), y_train.max(), y_train.min()
```

[13]: (25.67584731256419, 100, 0)

[14]:
```
import matplotlib.pyplot as plt
plt.hist(y_train)
```

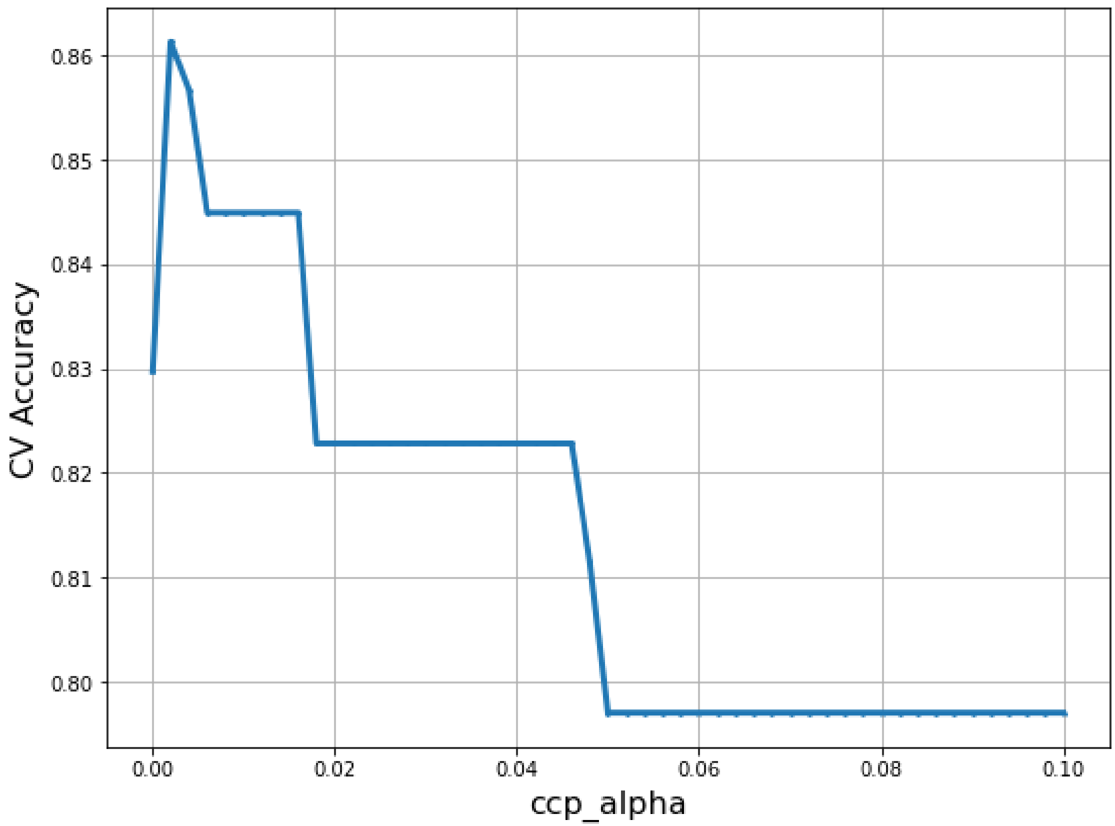
[14]:
```
(array([3.9388e+04, 1.0138e+04, 1.5827e+04, 1.8036e+04, 1.4227e+04,
        1.0394e+04, 6.0360e+03, 2.3960e+03, 3.7000e+02, 2.8000e+01]),
 array([  0.,  10.,  20.,  30.,  40.,  50.,  60.,  70.,  80.,  90., 100.]),
 <BarContainer object of 10 artists>)
```



[15]:
```
# convert the popularity to be 0 or 1
# 0 if the score is less than 85, else 1
y_train=pd.Series([1 if y_train.iloc[i]>=25 else 0 for i in␣
 ↪range(len(y_train))], index=y_train.index)
y_test=pd.Series([1 if y_test.iloc[i]>=25 else 0 for i in range(len(y_test))],␣
 ↪index=y_test.index)
blend_df['popularity']=pd.Series([1 if blend_df['popularity'].iloc[i]>=25 else␣
 ↪0 for i in range(len(blend_df['popularity']))], index=blend_df['popularity'].
 ↪index)
```

[16]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import plot_tree
from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeClassifier

grid_values = {'ccp_alpha': np.linspace(0, 0.1, 51)}

dtc = DecisionTreeClassifier(random_state=88)
dtc_cv = GridSearchCV(dtc, param_grid=grid_values, cv=5).fit(X_train, y_train)
```

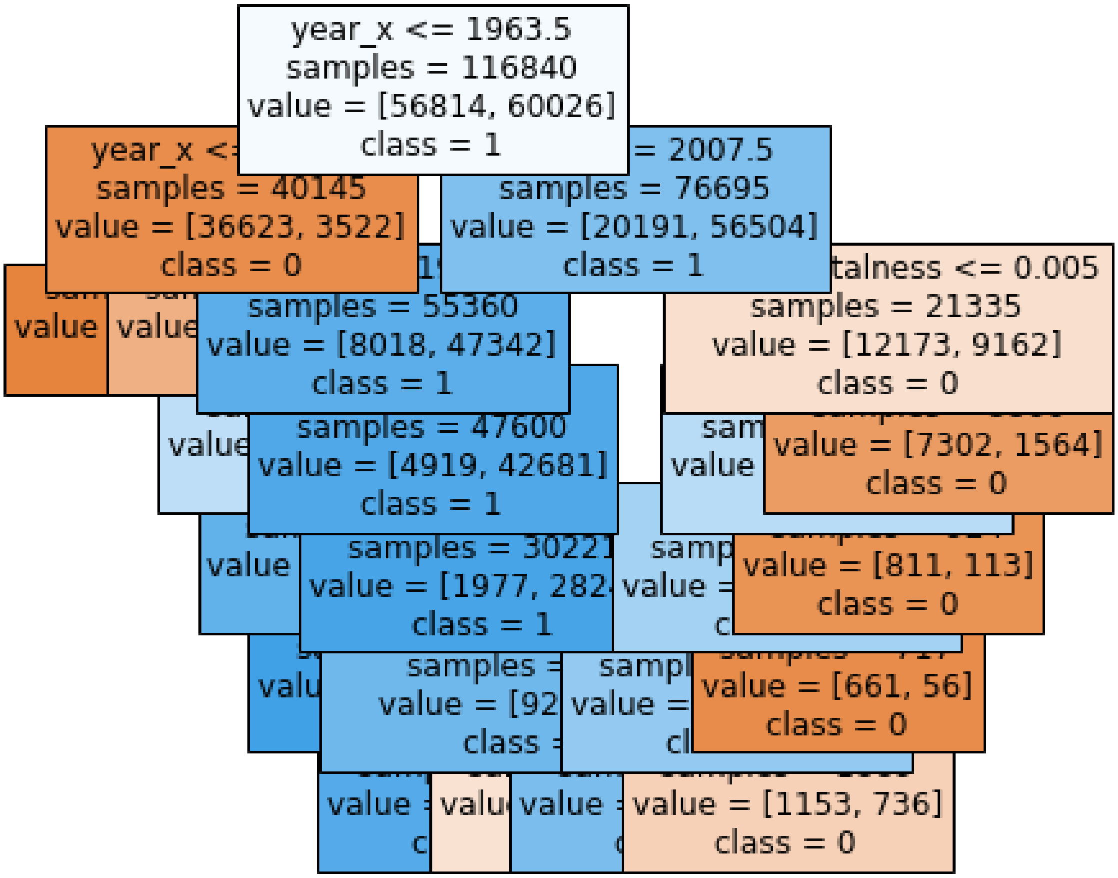
[17]:
```
ccp_alpha = dtc_cv.cv_results_['param_ccp_alpha'].data
ACC_scores = dtc_cv.cv_results_['mean_test_score']

plt.figure(figsize=(8, 6))
plt.xlabel('ccp_alpha', fontsize=16)
plt.ylabel('CV Accuracy', fontsize=16)
plt.scatter(ccp_alpha, ACC_scores, s=3)
plt.plot(ccp_alpha, ACC_scores, linewidth=3)
plt.grid(True, which='both')

plt.tight_layout()
plt.show()

print('Best ccp_alpha', dtc_cv.best_params_)
```




```
Best ccp_alpha {'ccp_alpha': 0.002}
```

[18]:
```python
from sklearn.tree import plot_tree

print('Node count =', dtc_cv.best_estimator_.tree_.node_count)
plt.figure(figsize=(6,6))
plot_tree(dtc_cv.best_estimator_,
          feature_names=X_train.columns,
          class_names=['0','1'],
          filled=True,
          impurity=False,
          fontsize=12)
plt.show()
```

```
Node count = 23
```



[19]:
```python
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score

# Compute the performance of the training set
y_pred_cart = dtc_cv.predict(X_test)

cm = confusion_matrix(y_test, y_pred_cart)

print ("Confusion Matrix: \n", cm)
print ("\nAccuracy:", accuracy_score(y_test, y_pred_cart))
print ("\nPrecision:", precision_score(y_test, y_pred_cart))
```

```
Confusion Matrix:
 [[23145  4872]
 [ 3228 26304]]

Accuracy: 0.8592503779388

Precision: 0.8437259430331023
```

[20]:
```
# The performance of the test set
y_pred = dtc_cv.predict(X_test)
cm = confusion_matrix(y_test, y_pred)
print ("Confusion Matrix: \n", cm)
print ("\nAccuracy:", accuracy_score(y_test, y_pred))
```

```
Confusion Matrix:
 [[23145  4872]
 [ 3228 26304]]

Accuracy: 0.8592503779388
```

0.2 Random Forest

[21]:
```
from sklearn.ensemble import RandomForestRegressor
import statsmodels.api as sm
from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import KFold

rf = RandomForestRegressor(max_features=5, min_samples_leaf=5,
                           n_estimators = 500, random_state=88, verbose=2)
rf.fit(X_train, y_train)
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.

building tree 1 of 500

[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.3s remaining:    0.0s

building tree 2 of 500
building tree 3 of 500
building tree 4 of 500
building tree 5 of 500
building tree 6 of 500
building tree 7 of 500
building tree 8 of 500
building tree 9 of 500
building tree 10 of 500
building tree 11 of 500
building tree 12 of 500
building tree 13 of 500
building tree 14 of 500
building tree 15 of 500
building tree 16 of 500
building tree 17 of 500
building tree 18 of 500
building tree 19 of 500
building tree 20 of 500
```

```
building tree 21 of 500
building tree 22 of 500
building tree 23 of 500
building tree 24 of 500
building tree 25 of 500
building tree 26 of 500
building tree 27 of 500
building tree 28 of 500
building tree 29 of 500
building tree 30 of 500
building tree 31 of 500
building tree 32 of 500
building tree 33 of 500
building tree 34 of 500
building tree 35 of 500
building tree 36 of 500
building tree 37 of 500
building tree 38 of 500
building tree 39 of 500
building tree 40 of 500
building tree 41 of 500
building tree 42 of 500
building tree 43 of 500
building tree 44 of 500
building tree 45 of 500
building tree 46 of 500
building tree 47 of 500
building tree 48 of 500
building tree 49 of 500
building tree 50 of 500
building tree 51 of 500
building tree 52 of 500
building tree 53 of 500
building tree 54 of 500
building tree 55 of 500
building tree 56 of 500
building tree 57 of 500
building tree 58 of 500
building tree 59 of 500
building tree 60 of 500
building tree 61 of 500
building tree 62 of 500
building tree 63 of 500
building tree 64 of 500
building tree 65 of 500
building tree 66 of 500
building tree 67 of 500
building tree 68 of 500
```

```
building tree 69 of 500
building tree 70 of 500
building tree 71 of 500
building tree 72 of 500
building tree 73 of 500
building tree 74 of 500
building tree 75 of 500
building tree 76 of 500
building tree 77 of 500
building tree 78 of 500
building tree 79 of 500
building tree 80 of 500
building tree 81 of 500
building tree 82 of 500
building tree 83 of 500
building tree 84 of 500
building tree 85 of 500
building tree 86 of 500
building tree 87 of 500
building tree 88 of 500
building tree 89 of 500
building tree 90 of 500
building tree 91 of 500
building tree 92 of 500
building tree 93 of 500
building tree 94 of 500
building tree 95 of 500
building tree 96 of 500
building tree 97 of 500
building tree 98 of 500
building tree 99 of 500
building tree 100 of 500
building tree 101 of 500
building tree 102 of 500
building tree 103 of 500
building tree 104 of 500
building tree 105 of 500
building tree 106 of 500
building tree 107 of 500
building tree 108 of 500
building tree 109 of 500
building tree 110 of 500
building tree 111 of 500
building tree 112 of 500
building tree 113 of 500
building tree 114 of 500
building tree 115 of 500
building tree 116 of 500
```

```
building tree 117 of 500
building tree 118 of 500
building tree 119 of 500
building tree 120 of 500
building tree 121 of 500
building tree 122 of 500
building tree 123 of 500
building tree 124 of 500
building tree 125 of 500
building tree 126 of 500
building tree 127 of 500
building tree 128 of 500
building tree 129 of 500
building tree 130 of 500
building tree 131 of 500
building tree 132 of 500
building tree 133 of 500
building tree 134 of 500
building tree 135 of 500
building tree 136 of 500
building tree 137 of 500
building tree 138 of 500
building tree 139 of 500
building tree 140 of 500
building tree 141 of 500
building tree 142 of 500
building tree 143 of 500
building tree 144 of 500
building tree 145 of 500
building tree 146 of 500
building tree 147 of 500
building tree 148 of 500
building tree 149 of 500
building tree 150 of 500
building tree 151 of 500
building tree 152 of 500
building tree 153 of 500
building tree 154 of 500
building tree 155 of 500
building tree 156 of 500
building tree 157 of 500
building tree 158 of 500
building tree 159 of 500
building tree 160 of 500
building tree 161 of 500
building tree 162 of 500
building tree 163 of 500
building tree 164 of 500
```

```
building tree 165 of 500
building tree 166 of 500
building tree 167 of 500
building tree 168 of 500
building tree 169 of 500
building tree 170 of 500
building tree 171 of 500
building tree 172 of 500
building tree 173 of 500
building tree 174 of 500
building tree 175 of 500
building tree 176 of 500
building tree 177 of 500
building tree 178 of 500
building tree 179 of 500
building tree 180 of 500
building tree 181 of 500
building tree 182 of 500
building tree 183 of 500
building tree 184 of 500
building tree 185 of 500
building tree 186 of 500
building tree 187 of 500
building tree 188 of 500
building tree 189 of 500
building tree 190 of 500
building tree 191 of 500
building tree 192 of 500
building tree 193 of 500
building tree 194 of 500
building tree 195 of 500
building tree 196 of 500
building tree 197 of 500
building tree 198 of 500
building tree 199 of 500
building tree 200 of 500
building tree 201 of 500
building tree 202 of 500
building tree 203 of 500
building tree 204 of 500
building tree 205 of 500
building tree 206 of 500
building tree 207 of 500
building tree 208 of 500
building tree 209 of 500
building tree 210 of 500
building tree 211 of 500
building tree 212 of 500
```

```
building tree 213 of 500
building tree 214 of 500
building tree 215 of 500
building tree 216 of 500
building tree 217 of 500
building tree 218 of 500
building tree 219 of 500
building tree 220 of 500
building tree 221 of 500
building tree 222 of 500
building tree 223 of 500
building tree 224 of 500
building tree 225 of 500
building tree 226 of 500
building tree 227 of 500
building tree 228 of 500
building tree 229 of 500
building tree 230 of 500
building tree 231 of 500
building tree 232 of 500
building tree 233 of 500
building tree 234 of 500
building tree 235 of 500
building tree 236 of 500
building tree 237 of 500
building tree 238 of 500
building tree 239 of 500
building tree 240 of 500
building tree 241 of 500
building tree 242 of 500
building tree 243 of 500
building tree 244 of 500
building tree 245 of 500
building tree 246 of 500
building tree 247 of 500
building tree 248 of 500
building tree 249 of 500
building tree 250 of 500
building tree 251 of 500
building tree 252 of 500
building tree 253 of 500
building tree 254 of 500
building tree 255 of 500
building tree 256 of 500
building tree 257 of 500
building tree 258 of 500
building tree 259 of 500
building tree 260 of 500
```

```
building tree 261 of 500
building tree 262 of 500
building tree 263 of 500
building tree 264 of 500
building tree 265 of 500
building tree 266 of 500
building tree 267 of 500
building tree 268 of 500
building tree 269 of 500
building tree 270 of 500
building tree 271 of 500
building tree 272 of 500
building tree 273 of 500
building tree 274 of 500
building tree 275 of 500
building tree 276 of 500
building tree 277 of 500
building tree 278 of 500
building tree 279 of 500
building tree 280 of 500
building tree 281 of 500
building tree 282 of 500
building tree 283 of 500
building tree 284 of 500
building tree 285 of 500
building tree 286 of 500
building tree 287 of 500
building tree 288 of 500
building tree 289 of 500
building tree 290 of 500
building tree 291 of 500
building tree 292 of 500
building tree 293 of 500
building tree 294 of 500
building tree 295 of 500
building tree 296 of 500
building tree 297 of 500
building tree 298 of 500
building tree 299 of 500
building tree 300 of 500
building tree 301 of 500
building tree 302 of 500
building tree 303 of 500
building tree 304 of 500
building tree 305 of 500
building tree 306 of 500
building tree 307 of 500
building tree 308 of 500
```

```
building tree 309 of 500
building tree 310 of 500
building tree 311 of 500
building tree 312 of 500
building tree 313 of 500
building tree 314 of 500
building tree 315 of 500
building tree 316 of 500
building tree 317 of 500
building tree 318 of 500
building tree 319 of 500
building tree 320 of 500
building tree 321 of 500
building tree 322 of 500
building tree 323 of 500
building tree 324 of 500
building tree 325 of 500
building tree 326 of 500
building tree 327 of 500
building tree 328 of 500
building tree 329 of 500
building tree 330 of 500
building tree 331 of 500
building tree 332 of 500
building tree 333 of 500
building tree 334 of 500
building tree 335 of 500
building tree 336 of 500
building tree 337 of 500
building tree 338 of 500
building tree 339 of 500
building tree 340 of 500
building tree 341 of 500
building tree 342 of 500
building tree 343 of 500
building tree 344 of 500
building tree 345 of 500
building tree 346 of 500
building tree 347 of 500
building tree 348 of 500
building tree 349 of 500
building tree 350 of 500
building tree 351 of 500
building tree 352 of 500
building tree 353 of 500
building tree 354 of 500
building tree 355 of 500
building tree 356 of 500
```

```
building tree 357 of 500
building tree 358 of 500
building tree 359 of 500
building tree 360 of 500
building tree 361 of 500
building tree 362 of 500
building tree 363 of 500
building tree 364 of 500
building tree 365 of 500
building tree 366 of 500
building tree 367 of 500
building tree 368 of 500
building tree 369 of 500
building tree 370 of 500
building tree 371 of 500
building tree 372 of 500
building tree 373 of 500
building tree 374 of 500
building tree 375 of 500
building tree 376 of 500
building tree 377 of 500
building tree 378 of 500
building tree 379 of 500
building tree 380 of 500
building tree 381 of 500
building tree 382 of 500
building tree 383 of 500
building tree 384 of 500
building tree 385 of 500
building tree 386 of 500
building tree 387 of 500
building tree 388 of 500
building tree 389 of 500
building tree 390 of 500
building tree 391 of 500
building tree 392 of 500
building tree 393 of 500
building tree 394 of 500
building tree 395 of 500
building tree 396 of 500
building tree 397 of 500
building tree 398 of 500
building tree 399 of 500
building tree 400 of 500
building tree 401 of 500
building tree 402 of 500
building tree 403 of 500
building tree 404 of 500
```

```
building tree 405 of 500
building tree 406 of 500
building tree 407 of 500
building tree 408 of 500
building tree 409 of 500
building tree 410 of 500
building tree 411 of 500
building tree 412 of 500
building tree 413 of 500
building tree 414 of 500
building tree 415 of 500
building tree 416 of 500
building tree 417 of 500
building tree 418 of 500
building tree 419 of 500
building tree 420 of 500
building tree 421 of 500
building tree 422 of 500
building tree 423 of 500
building tree 424 of 500
building tree 425 of 500
building tree 426 of 500
building tree 427 of 500
building tree 428 of 500
building tree 429 of 500
building tree 430 of 500
building tree 431 of 500
building tree 432 of 500
building tree 433 of 500
building tree 434 of 500
building tree 435 of 500
building tree 436 of 500
building tree 437 of 500
building tree 438 of 500
building tree 439 of 500
building tree 440 of 500
building tree 441 of 500
building tree 442 of 500
building tree 443 of 500
building tree 444 of 500
building tree 445 of 500
building tree 446 of 500
building tree 447 of 500
building tree 448 of 500
building tree 449 of 500
building tree 450 of 500
building tree 451 of 500
building tree 452 of 500
```

```
building tree 453 of 500
building tree 454 of 500
building tree 455 of 500
building tree 456 of 500
building tree 457 of 500
building tree 458 of 500
building tree 459 of 500
building tree 460 of 500
building tree 461 of 500
building tree 462 of 500
building tree 463 of 500
building tree 464 of 500
building tree 465 of 500
building tree 466 of 500
building tree 467 of 500
building tree 468 of 500
building tree 469 of 500
building tree 470 of 500
building tree 471 of 500
building tree 472 of 500
building tree 473 of 500
building tree 474 of 500
building tree 475 of 500
building tree 476 of 500
building tree 477 of 500
building tree 478 of 500
building tree 479 of 500
building tree 480 of 500
building tree 481 of 500
building tree 482 of 500
building tree 483 of 500
building tree 484 of 500
building tree 485 of 500
building tree 486 of 500
building tree 487 of 500
building tree 488 of 500
building tree 489 of 500
building tree 490 of 500
building tree 491 of 500
building tree 492 of 500
building tree 493 of 500
building tree 494 of 500
building tree 495 of 500
building tree 496 of 500
building tree 497 of 500
building tree 498 of 500
building tree 499 of 500
building tree 500 of 500
```

```
[Parallel(n_jobs=1)]: Done 500 out of 500 | elapsed:  2.8min finished
```

[21]:
```
RandomForestRegressor(max_features=5, min_samples_leaf=5, n_estimators=500,
                      random_state=88, verbose=2)
```

[22]:
```
# Evaluate the model performance on the testing set
y_prob_rf = rf.predict(X_test)
y_pred_rf = pd.Series([1 if x >= 0.5 else 0 for x in y_prob_rf])
cm = confusion_matrix(y_test, y_pred)
print ("Confusion Matrix : \n", cm)
print ("\nAccuracy:", accuracy_score(y_test, y_pred))
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
```

```
Confusion Matrix :
 [[23145  4872]
 [ 3228 26304]]

Accuracy: 0.8592503779388
```

```
[Parallel(n_jobs=1)]: Done 500 out of 500 | elapsed:    7.6s finished
```

0.3 Boosting

[23]:
```
from sklearn.ensemble import GradientBoostingClassifier
import time

tic = time.time()

gbc = GradientBoostingClassifier()
gbc.fit(X_train, y_train)

toc = time.time()

print('time:', round(toc-tic, 2),'s')
```

```
time: 42.63 s
```

[24]:
```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score

y_pred_boost = gbc.predict(X_test)

cm = confusion_matrix(y_test, y_pred)
print ("Confusion Matrix: \n", cm)

print ("\nAccuracy:", accuracy_score(y_test, y_pred))
tn, fp, fn, tp = cm.ravel()
```

```
rf_cv_tpr = tp / (tp + fn)
rf_cv_fpr = fp / (fp + tn)
rf_cv_acc = (tp + tn) / (tp + tn + fn + fp)
print(rf_cv_tpr)
print(rf_cv_acc)
```

```
Confusion Matrix:
 [[23145  4872]
 [ 3228 26304]]

Accuracy: 0.8592503779388
0.8906948394961398
0.8592503779388
```

0.4 Logistic Model

[25]:
```
X_train.corr
```

[25]:
```
<bound method DataFrame.corr of          acousticness  danceability  duration_ms
energy  explicit  \
43437          0.984         0.404       194505  0.1640         0
160745         0.994         0.301       180827  0.1160         0
123806         0.703         0.511       193440  0.6320         0
126495         0.114         0.721       339345  0.9030         1
59850          0.993         0.647       171360  0.1620         0
…              …            …           …          …       …
90474          0.104         0.652       286800  0.6300         1
133553         0.135         0.393       198120  0.4190         0
36815          0.231         0.454       352500  0.5340         0
104736         0.158         0.332       313933  0.2790         0
124888         0.950         0.176       185347  0.0302         0

        instrumentalness  key  liveness  loudness  mode  …  live  live  \
43437           0.073400    7    0.2110   -12.411     0  …     0     0
160745          0.000217    2    0.0830   -17.337     1  …     0     0
123806          0.000000    8    0.0991    -2.370     1  …     0     0
126495          0.000000    1    0.3370    -2.968     1  …     0     0
59850           0.776000   10    0.3090   -16.582     1  …     0     0
…                    …   …        …         …     …   …    …    …
90474           0.000000   10    0.0299    -7.694     0  …     0     0
133553          0.048100   10    0.0852   -12.023     1  …     0     0
36815           0.818000    6    0.1370    -9.619     0  …     0     0
104736          0.158000    0    0.0838   -15.601     1  …     0     0
124888          0.748000    8    0.0943   -25.109     1  …     0     0

        love  mix  no  op  remast  version  year_y  Season
43437      0    0   0   0       0        0       0       0
```

```
160745      0    0   0   0       1        1       0       0
123806      0    0   0   0       0        0       0       0
126495      0    0   0   0       0        0       0       0
59850       0    0   0   0       0        0       0       0
…       …  …  . .  . .      …      …      …      …
90474       0    0   0   0       0        0       0       0
133553      0    0   0   0       1        0       0       0
36815       0    0   0   0       0        0       0       1
104736      0    0   0   0       0        0       0       0
124888      0    0   0   0       0        0       0       1

[116840 rows x 26 columns]>
```

[26]:

```python
import statsmodels.api as smf
#building model and fitting data
log_reg = smf.Logit(np.asarray(y_train), np.asarray(X_train)).fit()

#visualize summary. Can drop columns x1, x4, x15 since coef are minimal
print(log_reg.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.521484
         Iterations 7
                           Logit Regression Results
==============================================================================
Dep. Variable:                      y   No. Observations:               116840
Model:                          Logit   Df Residuals:                   116815
Method:                           MLE   Df Model:                           24
Date:                Wed, 12 May 2021   Pseudo R-squ.:                  0.2472
Time:                        13:56:51   Log-Likelihood:                -60930.
converged:                       True   LL-Null:                       -80943.
Covariance Type:            nonrobust   LLR p-value:                     0.000
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
x1            -2.0930        nan        nan        nan         nan         nan
x2             0.0358      0.047      0.761      0.447      -0.056       0.128
x3          6.006e-07        nan        nan        nan         nan         nan
x4             0.6169      0.073      8.456      0.000       0.474       0.760
x5             1.2667        nan        nan        nan         nan         nan
x6            -1.6017        nan        nan        nan         nan         nan
x7            -0.0026      0.002     -1.298      0.194      -0.006       0.001
x8            -0.7320      0.238     -3.079      0.002      -1.198      -0.266
x9            -0.0417      0.003    -15.173      0.000      -0.047      -0.036
x10            0.0851      0.016      5.383      0.000       0.054       0.116
x11           -4.5632      0.072    -63.302      0.000      -4.705      -4.422
x12           -0.0006      0.000     -2.586      0.010      -0.001      -0.000
```

```
x13            -0.3131      0.050     -6.315      0.000      -0.410      -0.216
x14             0.0008   3.69e-05     20.840      0.000       0.001       0.001
x15            -0.4561      0.040    -11.507      0.000      -0.534      -0.378
x16            -0.0513      0.006     -8.456      0.000      -0.063      -0.039
x17            -0.0474        nan        nan        nan         nan         nan
x18            -0.0474        nan        nan        nan         nan         nan
x19             0.2044      0.036      5.683      0.000       0.134       0.275
x20            -1.9074      0.053    -36.129      0.000      -2.011      -1.804
x21             0.2158        nan        nan        nan         nan         nan
x22            -0.4334      0.067     -6.481      0.000      -0.564      -0.302
x23             0.0307      0.030      1.027      0.304      -0.028       0.089
x24             0.0029      0.042      0.069      0.945      -0.079       0.085
x25            -2.6491      0.065    -40.925      0.000      -2.776      -2.522
x26             0.5055      0.021     24.640      0.000       0.465       0.546
==============================================================================

/opt/conda/lib/python3.8/site-packages/statsmodels/base/model.py:1354:
RuntimeWarning: invalid value encountered in sqrt
  bse_ = np.sqrt(np.diag(self.cov_params()))
```

[27]:
```
#define y_hat as the predicted value of y for test dataset
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 :
 [[19947  8070]
 [ 6158 23374]]
```

[28]:
```
#accuracy of the model
print('Test Accuracy = ', accuracy_score(y_test, prediction))
```

```
Test Accuracy =  0.7527672070757094
```

0.5 LDA

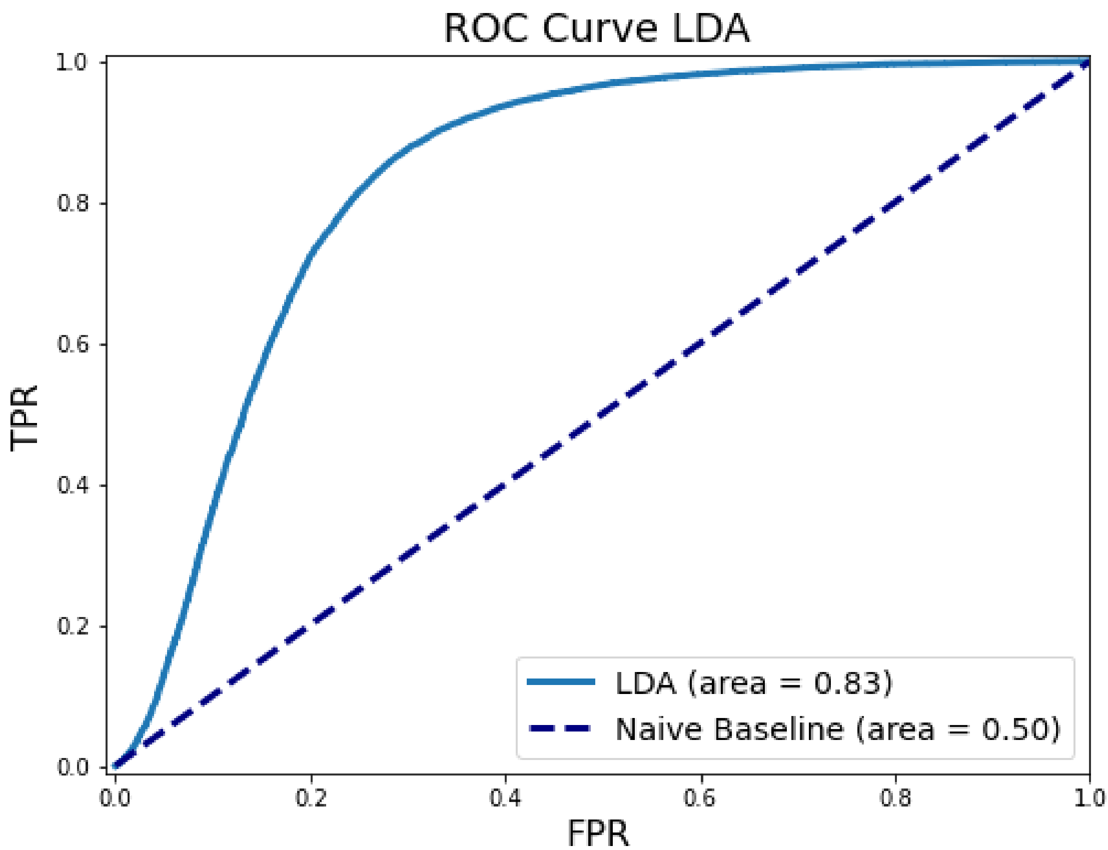
[29]:
```
#lda model
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.metrics import roc_curve
from sklearn.metrics import auc

lda = LinearDiscriminantAnalysis()
```

```
lda.fit(X_train, y_train)

y_prob_lda = lda.predict_proba(X_test)
y_pred_lda = pd.Series([1 if x > .5 else 0 for x in y_prob_lda[:,1]])


cm = confusion_matrix(y_test, y_pred_lda)
print ("Confusion Matrix: \n", cm)
print ("\nAccuracy:", accuracy_score(y_test, y_pred_lda))
```

```
Confusion Matrix:
 [[20406  7611]
 [ 4585 24947]]

Accuracy: 0.7880762480668647
```

[30]:

```
fpr_lda, tpr_lda, _ = roc_curve(y_test, y_prob_lda[:,1])
roc_auc_lda = auc(fpr_lda, tpr_lda)

plt.figure(figsize=(8, 6))
plt.title('ROC Curve LDA', fontsize=18)
plt.xlabel('FPR', fontsize=16)
plt.ylabel('TPR', fontsize=16)
plt.xlim([-0.01, 1.00])
plt.ylim([-0.01, 1.01])
plt.plot(fpr_lda, tpr_lda, lw=3, label='LDA (area = {:0.2f})'.
 ↪format(roc_auc_lda))
plt.plot([0, 1], [0, 1], color='navy', lw=3, linestyle='--', label='Naive␣
 ↪Baseline (area = 0.50)')
plt.legend(loc='lower right', fontsize=14)
plt.savefig("roc_lda.png", bbox_inches='tight', dpi=600)
plt.show()
```



0.6 Blending

[31]:
```
blend_df["val_pred_log"]=prediction
blend_df["val_pred_rf"]=y_pred_rf
blend_df["val_pred_boost"]=y_pred_boost
blend_df["val_pred_lda"]=y_pred_lda
blend_df["val_pred_cart"]=y_pred_cart
```

[32]:
```
import statsmodels.formula.api as smf

blending_ols = smf.ols(formula='popularity ~␣
↪val_pred_log+val_pred_lda+val_pred_rf+val_pred_boost', data=blend_df)
blending_res = blending_ols.fit()
print(blending_res.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:             popularity   R-squared:                       0.599
Model:                            OLS   Adj. R-squared:                  0.599
Method:                 Least Squares   F-statistic:                 2.150e+04
```

```
Date:                Wed, 12 May 2021   Prob (F-statistic):                0.00
Time:                        13:56:55   Log-Likelihood:                 -15444.
No. Observations:               57549   AIC:                         3.090e+04
Df Residuals:                   57544   BIC:                         3.094e+04
Df Model:                           4
Covariance Type:            nonrobust
================================================================================
==
                     coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
--
Intercept          0.0734      0.002     34.872      0.000       0.069
0.078
val_pred_log       0.0430      0.004      9.678      0.000       0.034
0.052
val_pred_lda       0.0538      0.005     10.784      0.000       0.044
0.064
val_pred_rf        0.5743      0.009     67.549      0.000       0.558
0.591
val_pred_boost     0.1423      0.009     16.733      0.000       0.126
0.159
==============================================================================
Omnibus:                     6403.283   Durbin-Watson:                   2.013
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            33924.693
Skew:                          -0.414   Prob(JB):                         0.00
Kurtosis:                       6.669   Cond. No.                         14.6
==============================================================================

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

[33]:

```
val_pred_blended =blending_res.predict(blend_df)
blend_df['pred_blended']= val_pred_blended
blend_df
```

[33]:

```
       acousticness  danceability  duration_ms  energy  explicit  \
0          0.000326         0.306       376360   0.268         0
1          0.000008         0.445       257160   0.876         0
2          0.198000         0.497        46811   0.691         0
3          0.464000         0.749       232800   0.546         0
4          0.008820         0.844       405707   0.442         0
…              …             …           …        …        …
57544      0.164000         0.674       186747   0.578         0
57545      0.337000         0.549       184667   0.460         0
57546      0.404000         0.401       424000   0.943         0
```

```
57547      0.900000          0.181       266973   0.241          0
57548      0.579000          0.752       207212   0.507          0

       instrumentalness  key  liveness  loudness  mode  …  version  year_y  \
0              0.829000    7    0.1060   -18.097     1  …        0       0
1              0.715000   11    0.3810    -6.549     1  …        0       0
2              0.935000    7    0.1030   -10.812     0  …        0       1
3              0.001910    8    0.1260    -7.217     0  …        1       0
4              0.000498    9    0.0879   -16.540     1  …        0       0
…                   …   …       …        …     …   …      …
57544          0.000000    4    0.1030   -10.919     1  …        0       0
57545          0.000003    1    0.1100   -11.343     1  …        0       0
57546          0.949000    0    0.0933    -4.441     1  …        0       0
57547          0.000002   10    0.0808   -11.626     1  …        0       1
57548          0.850000   10    0.0732   -11.581     1  …        0       1

       Season  popularity  val_pred_log  val_pred_rf  val_pred_boost  \
0           0           1             1            1               1
1           0           0             1            0               0
2           0           0             0            0               0
3           0           1             1            1               1
4           0           1             1            1               1
…         …          …            …           …              …
57544       0           1             1            1               1
57545       0           1             1            1               1
57546       0           1             0            1               1
57547       0           1             0            0               0
57548       0           0             0            0               0

       val_pred_lda  val_pred_cart  pred_blended
0                 1              1      0.886818
1                 1              0      0.170229
2                 0              0      0.073431
3                 1              1      0.886818
4                 1              1      0.886818
…              …             …            …
57544             1              1      0.886818
57545             1              1      0.886818
57546             1              1      0.843781
57547             0              0      0.073431
57548             0              0      0.073431

[57549 rows x 35 columns]
```

[34]:
```python
def masked_mae(X_true, X_pred, mask):
    masked_diff = X_true[mask] - X_pred[mask]
    return np.mean(np.abs(masked_diff))
```

```
def masked_mse(X_true, X_pred, mask):
    masked_diff = X_true[mask] - X_pred[mask]
    return np.mean(masked_diff ** 2)

def OSR2(mse_model, mse_baseline):
    return 1 - mse_model/mse_baseline
```

[36]:
```
val_mae_blended = np.mean(np.abs(blend_df["popularity"] - val_pred_blended))
print("Normalized MAE %s " % (val_mae_blended/100))

val_mae_blended = np.mean(np.abs(blend_df["popularity"] - val_pred_blended))
print("MAE %s " % (val_mae_blended))

val_mse_blended = np.mean((blend_df["popularity"] - val_pred_blended)**2)
print("Normalized RMSE %s " % (np.sqrt(val_mse_blended/100)))

val_mse_blended = np.mean((blend_df["popularity"] - val_pred_blended)**2)
print("RMSE %s " % (np.sqrt(val_mse_blended)))
```

```
Normalized MAE 0.002002855493548226
MAE 0.2002855493548226
Normalized RMSE 0.03164534320834753
RMSE 0.3164534320834753
```

[ ]:

[ ]: