Billboard Decision Tree, Logistic Regression, Random Forest, SVM

May 19, 2019

[22]:
```
import pandas as pd
import numpy as np

from sklearn.model_selection import train_test_split
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier, export_graphviz
from sklearn.model_selection import train_test_split
from sklearn.tree import export_graphviz
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix

from matplotlib import pyplot as plt
import seaborn as sns

import graphviz
import pydotplus
import io
from scipy import misc
```

[23]:
```
hits = pd.read_csv("billboard100_with_songscores.csv", sep = ',')
non_hits = pd.read_csv("billboard_songscores.csv", sep = ',')
```

[24]:
```
hitsdf = hits.iloc[:,:]
pre_non_hitsdf = non_hits.iloc[:,:]
```

[25]:
```
pre_non_hitsdf2 = pre_non_hitsdf.loc[pre_non_hitsdf['billboard_hit'] == 0]
#print(non_hitsdf.dtypes)
#non_hitsdf['album_release_date'] = pd.
 ↪to_datetime(non_hitsdf['album_release_date'])
```

[26]:
```
hitstrain, hitstest, _, _ = train_test_split(hitsdf, hitsdf, test_size=0.1,␣
 ↪random_state=100)
```

[27]:
```
#get a sample of non-hits the same size as our hitsdf
non_hitsdf = pre_non_hitsdf2.sample(n=hitsdf.shape[0], random_state=100)
non_hitstrain, non_hitstest, _, _ = train_test_split(non_hitsdf, non_hitsdf,␣
 ↪test_size=0.1, random_state=100)
```

[28]:
```
#concatenate evenly distributed sets
full_train = pd.concat([hitstrain, non_hitstrain], axis=0).reset_index(drop=True)
full_test = pd.concat([hitstest, non_hitstest], axis=0).reset_index(drop=True)
```

[29]:
```
full_train['metric'].fillna(0, inplace=True)
full_test['metric'].fillna(0, inplace=True)
```

[30]:
```
def modeldf(df, subset_cols):

    df = df.dropna(subset=['explicit', 'mode', 'key',␣
 ↪'time_signature','duration_ms', 'acousticness', 'danceability',␣
 ↪'energy','instrumentalness', 'liveliness', 'loudness', 'speechiness',␣
 ↪'tempo','valence', 'billboard_hit'], how = 'any')
    explicit = pd.get_dummies(df['explicit'], prefix ='explicit')
    mode = pd.get_dummies(df['mode'], prefix='mode')
    key = pd.get_dummies(df['key'], prefix='key')
    time_signature = pd.get_dummies(df['time_signature'],␣
 ↪prefix='time_signature')
    df = pd.concat([df[subset_cols], explicit, mode, key, time_signature],␣
 ↪axis=1)
    return df
```

[31]:
```
subset_cols = ['track_title', 'duration_ms', 'acousticness', 'danceability',␣
 ↪'energy', 'instrumentalness', 'liveliness', 'loudness', 'speechiness',␣
 ↪'tempo', 'valence', 'billboard_hit']
full_train = modeldf(full_train, subset_cols=subset_cols)
full_test = modeldf(full_test, subset_cols=subset_cols)
```

[32]:
```
c = DecisionTreeClassifier(min_samples_split = 100)
features = full_train.columns.difference(['track_title',␣
 ↪'artist_title','billboard_hit'])
X_train = full_train[features]
Y_train = full_train['billboard_hit'].apply(np.int64)
X_test = full_test[features]
Y_test = full_test['billboard_hit'].apply(np.int64)
```

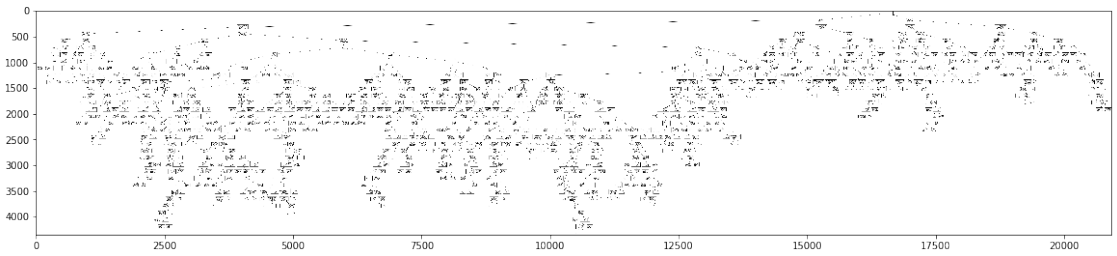
[33]:
```
dt = c.fit(X_train, Y_train)
```

[34]:
```
def show_tree(tree, features, path):
    f = io.StringIO()
    export_graphviz(tree, out_file=f, feature_names = features)
    pydotplus.graph_from_dot_data(f.getvalue()).write_png(path)
    img = misc.imread(path)
    plt.rcParams["figure.figsize"] = (20,20)
    plt.imshow(img)
```

[35]:
```
show_tree(dt, features, 'billboard_decision_tree.png')
```

```
/Users/angelobravo/anaconda3/lib/python3.6/site-
```

```
packages/ipykernel_launcher.py:5: DeprecationWarning: `imread` is deprecated!
`imread` is deprecated in SciPy 1.0.0, and will be removed in 1.2.0.
Use ``imageio.imread`` instead.
  """
/Users/angelobravo/anaconda3/lib/python3.6/site-packages/PIL/Image.py:2618:
DecompressionBombWarning: Image size (90915576 pixels) exceeds limit of 89478485
pixels, could be decompression bomb DOS attack.
  DecompressionBombWarning)
```



[36]:
```
y_pred = c.predict(X_test)
score = accuracy_score(Y_test, y_pred) * 100
print("Decision Tree Accuracy: ", score)
```

```
Decision Tree Accuracy:  76.37383177570094
```

[37]:
```
from sklearn.linear_model import LogisticRegression
```

[38]:
```
model = LogisticRegression()
model.fit(X_train, Y_train)
model.predict(X_test)
```

```
/Users/angelobravo/anaconda3/lib/python3.6/site-
packages/sklearn/linear_model/logistic.py:433: FutureWarning: Default solver
will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
```

[38]:
```
array([1, 0, 0, ..., 0, 0, 1])
```

[39]:
```
model.score(X_test, Y_test)
```

[39]:
```
0.6579439252336449
```

[40]:
```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(n_estimators = 100)
model.fit(X_train, Y_train)
```

[40]:
```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
```

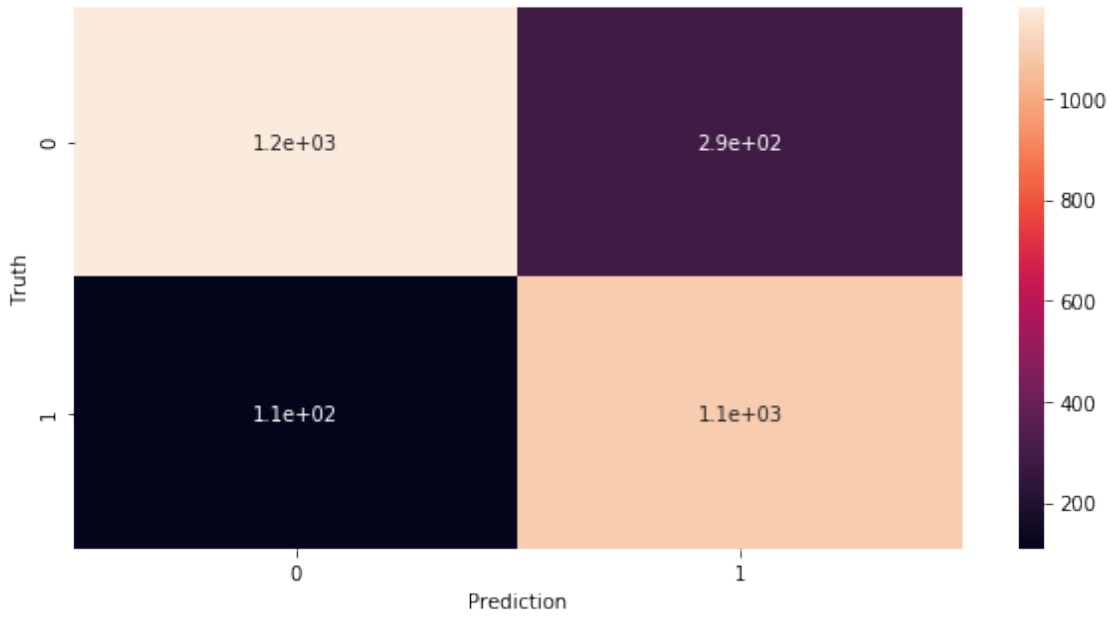
```
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=None,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
```

[41]:
```
model.score(X_test, Y_test)
Y_predicted = model.predict(X_test)
cm = confusion_matrix(Y_test, Y_predicted)
cm
```

[41]:
```
array([[1182,  293],
       [ 109, 1091]])
```

[42]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sn
plt.figure(figsize=(10,5))
sn.heatmap(cm, annot = True)
plt.xlabel("Prediction")
plt.ylabel("Truth")
```

[42]:
```
Text(69.0, 0.5, 'Truth')
```



[43]:
```
from sklearn.svm import SVC
model = SVC()
model.fit(X_train, Y_train)
model.score(X_test, Y_test)
```

```
/Users/angelobravo/anaconda3/lib/python3.6/site-
packages/sklearn/svm/base.py:196: FutureWarning: The default value of gamma will
change from 'auto' to 'scale' in version 0.22 to account better for unscaled
features. Set gamma explicitly to 'auto' or 'scale' to avoid this warning.
  "avoid this warning.", FutureWarning)
```

[43]:
```
0.8362616822429907
```