In [1]:

```
import pandas as pd
import numpy as np
#-----------------------------------------Scikit-----------------------------
from sklearn.model_selection import train_test_split
from sklearn.decomposition import PCA
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import balanced_accuracy_score
from sklearn.preprocessing import LabelEncoder

#------------------------------------import feature selection---------------
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2          #score_func, need to no
from sklearn.feature_selection import f_classif
#-----------------------------------------Seaborn----------------------------
import matplotlib.pyplot as plt
import ssl
import seaborn as sns
ssl._create_default_https_context = ssl._create_unverified_context
sns.set(style='darkgrid')
```

In [2]:

```
df = pd.read_csv('./Data/features_30_sec.csv')    #read file
```

In [3]:

```
df.head()                                         #display first 5 rows of dat
```

Out[3]:

	filename	length	chroma_stft_mean	chroma_stft_var	rms_mean	rms_var	spectral_cent
0	blues.00000.wav	661794	0.350088	0.088757	0.130228	0.002827	17
1	blues.00001.wav	661794	0.340914	0.094980	0.095948	0.002373	15
2	blues.00002.wav	661794	0.363637	0.085275	0.175570	0.002746	15
3	blues.00003.wav	661794	0.404785	0.093999	0.141093	0.006346	10
4	blues.00004.wav	661794	0.308526	0.087841	0.091529	0.002303	18

5 rows × 60 columns

In [4]:

```
df = df.drop(['length','filename'],axis=1)          #remove the length and
df = df.sample(frac=1)                              #randomize rows of dat
df
```

Out[4]:

	chroma_stft_mean	chroma_stft_var	rms_mean	rms_var	spectral_centroid_mean	spectral_cei
223	0.305372	0.098811	0.038920	0.000472	1494.665090	6262
981	0.335662	0.086840	0.093668	0.001466	2553.527051	3597
527	0.325149	0.095212	0.040618	0.000251	1769.392505	1062

	chroma_stft_mean	chroma_stft_var	rms_mean	rms_var	spectral_centroid_mean	spectral_cei
957	0.390014	0.083890	0.122645	0.001005	3488.554943	5599
325	0.358008	0.093773	0.107915	0.003411	2410.182747	5931
...	...	...	...	...	...	
19	0.257325	0.095963	0.097660	0.002575	1195.470376	2495
780	0.433511	0.091234	0.215830	0.008250	3151.267308	8293
944	0.311694	0.083247	0.144356	0.001230	1651.421086	2524
730	0.357656	0.084707	0.203244	0.006338	3070.608038	6090
41	0.386868	0.085420	0.129166	0.000870	2390.390100	3066

In [5]:

```
#fig,axs = plt.subplots(1,2,figsize=(16,8),gridspec_kw=dict(width_ratios=[10,
#sns.scatterplot(data = df, x ='chroma_stft_var',y='rolloff_mean',hue='label'
```

In [6]:

```
#-----------------------------------------Encode The Genre Into Numbers------
labelEncoder = LabelEncoder()                #store encoded labels into variabl
le = labelEncoder.fit(df['label'])                             #fit label into
df['label'] = le.transform(df['label'])          #transform label values into n
Y_genre = df['label']
X_features = df.drop('label',axis=1)
```

In [7]:

```
X_features.head()
```

Out[7]:

	chroma_stft_mean	chroma_stft_var	rms_mean	rms_var	spectral_centroid_mean	spectral_cei
223	0.305372	0.098811	0.038920	0.000472	1494.665090	6262
981	0.335662	0.086840	0.093668	0.001466	2553.527051	3597
527	0.325149	0.095212	0.040618	0.000251	1769.392505	1062
957	0.390014	0.083890	0.122645	0.001005	3488.554943	5599
325	0.358008	0.093773	0.107915	0.003411	2410.182747	5931

5 rows × 57 columns

In [8]:

```
Y_genre.tail()
```

Out[8]:

```
19     0
780    7
944    9
730    7
41     0
Name: label, dtype: int64
```

In [9]:

```
#------------------------------Scatterplot of Data---------------------------
#sns.scatterplot(data = df, x ='chroma_stft_var',y='rolloff_mean',hue='label'
```

In [10]:

```
best_feat = SelectKBest(score_func= f_classif, k=20)
fit = best_feat.fit(X_features,Y_genre)
```

In [11]:

```
feat_scores = pd.DataFrame(fit.scores_)
feat_columns = pd.DataFrame(X_features.columns)
```

In [12]:

```
sel_scores = pd.concat([feat_columns,feat_scores],axis=1)
sel_scores.columns = ['Features','Scores']
```

In [13]:

```
sel_scores.sort_values(by=['Scores'],ascending=False)
sel_largest = sel_scores.nlargest(20,'Scores')
sel_largest
```

Out[13]:

	Features	Scores
0	chroma_stft_mean	176.453282
17	mfcc1_mean	130.371835
6	spectral_bandwidth_mean	116.601879
8	rolloff_mean	110.871317
4	spectral_centroid_mean	97.484924
23	mfcc4_mean	83.868555
19	mfcc2_mean	83.189909
5	spectral_centroid_var	82.134648
15	perceptr_var	80.388712
2	rms_mean	74.194652
3	rms_var	68.276438
31	mfcc8_mean	67.496117
27	mfcc6_mean	66.038123
1	chroma_stft_var	64.103833
11	zero_crossing_rate_var	62.333004
24	mfcc4_var	61.487770
49	mfcc17_mean	59.083651
10	zero_crossing_rate_mean	58.716380
28	mfcc6_var	56.717082
33	mfcc9_mean	55.340635

In [14]:

```
X_features = X_features[sel_largest['Features'].T]
X_features
```

Out[14]:

	chroma_stft_mean	mfcc1_mean	spectral_bandwidth_mean	rolloff_mean	spectral_centroid_mea
223	0.305372	-353.216125	2330.919047	3050.528783	1494.6650
981	0.335662	-121.429237	2219.053282	5123.411840	2553.5270
527	0.325149	-262.060669	2113.399859	3966.339409	1769.3925
957	0.390014	-58.556953	3241.605654	7660.024833	3488.5549
325	0.358008	-124.138123	2575.801114	5080.486990	2410.1827
...	...	...	...	...	
19	0.257325	-236.656754	1481.318880	2235.264725	1195.4703
780	0.433511	-44.953270	3220.605672	7294.301780	3151.2673
944	0.311694	-118.388779	2103.471128	3455.029920	1651.4210
730	0.357656	-18.115849	2975.765840	6653.027004	3070.6080
41	0.386868	-107.170265	2463.308269	5403.435076	2390.3901

1000 rows × 20 columns

In [15]:

```
#sns.scatterplot(data = pd.concat([X_features,Y_genre],axis=1), x ='chroma_st
```

In [16]:

```
#-------------------------------------Split data into train and test---------
#X_train,X_test,y_train,y_test = train_test_split(X_features,Y_genre,test_siz
```

In [17]:

```
# #---------------------------Use ML to reduce # of data columns------------
# pca = PCA(n_components=50)                         #reduce to 30 columns
# pca.fit(train_data)                                #fit data into the pc
# train_data_pca = pca.transform(train_data)
# test_data_pca = pca.transform(test_data)
```

In [18]:

```
#----------------------------------------------K Nearest Neighbor------------
# knn = KNeighborsClassifier(n_neighbors=8)
# knn.fit(train_data_pca,train_label.values.ravel())
# pred = knn.predict(test_data_pca)
```

In [19]:

```
df = df.sample(frac=1)                                 #randomize rows of data
row,col = pd.concat([X_features,Y_genre],axis=1).shape  #extract size of data
split = 0.75                                           #3/4 training split
X_train = df.iloc[:int(row*split),:-1]              #obtain 75% of test data
y_train = df.iloc[:int(row*split),-1:]             #obtain 75% of genre data
X_test = df.iloc[int(row*split):,:-1]               #obtain 25% of test data
y_test = df.iloc[int(row*split):,-1:]              #obtain 25% of genre data
```

In [23]:

```
clf = RandomForestClassifier(n_estimators=50, max_depth=None,min_samples_spli
clf.fit(X_train,y_train.values.ravel())
```

```
Out[23]: RandomForestClassifier(n_estimators=50, random_state=0)
```

```
In [24]: clf.predict(X_test)
```

```
Out[24]: array([8, 9, 1, 3, 1, 6, 5, 0, 6, 1, 7, 9, 3, 6, 1, 1, 3, 2, 0, 5, 5, 2,
                9, 5, 3, 2, 6, 7, 3, 2, 1, 7, 7, 3, 3, 1, 1, 4, 9, 1, 8, 2, 7, 3,
                6, 5, 9, 1, 6, 5, 7, 5, 8, 6, 9, 6, 7, 1, 6, 3, 9, 4, 4, 4, 0, 3,
                5, 9, 0, 6, 8, 9, 9, 7, 6, 3, 9, 5, 8, 1, 5, 8, 6, 3, 8, 2, 3, 4,
                6, 7, 5, 2, 1, 2, 6, 9, 1, 8, 3, 4, 9, 9, 8, 1, 2, 4, 0, 5, 5, 7,
                3, 9, 5, 3, 2, 6, 5, 5, 2, 8, 2, 4, 3, 6, 0, 9, 6, 3, 2, 4, 0, 1,
                9, 3, 2, 4, 8, 7, 8, 8, 4, 6, 8, 7, 7, 3, 9, 0, 3, 3, 6, 5, 6, 7,
                3, 4, 8, 9, 9, 5, 3, 0, 3, 6, 9, 4, 2, 7, 7, 1, 0, 6, 8, 2, 8, 2,
                1, 8, 6, 8, 2, 7, 7, 8, 0, 0, 7, 9, 6, 5, 0, 8, 9, 6, 4, 1, 3, 7,
                3, 9, 5, 6, 8, 7, 0, 0, 1, 4, 1, 2, 0, 6, 2, 4, 8, 7, 6, 8, 1, 0,
                7, 7, 6, 1, 6, 7, 6, 1, 8, 8, 3, 0, 6, 2, 8, 1, 2, 5, 2, 6, 1, 8,
                6, 4, 7, 8, 9, 8, 1, 9])
```

```
In [28]: clf.score(X_test,y_test)
```

```
Out[28]: 0.708
```

```
In [ ]:
```

```
In [ ]:
```