Project

May 16, 2018

In [1]:
```
import csv
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from collections import Counter
```

In [2]:
```
import seaborn as sns
from sklearn.utils import shuffle
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn import svm, metrics
from sklearn.linear_model import LogisticRegression, Ridge
from sklearn.ensemble import RandomForestClassifier
from sklearn.decomposition import PCA
from sklearn import preprocessing
from sklearn.utils import resample
import itertools
from sklearn.model_selection import GridSearchCV
%matplotlib inline
```

In [34]:
```
#import os, sys

#dir_tree = '/Users/chirag/Downloads/MillionSongSubset'

#for dir_path, dir_names, file_names in os.walk(dir_tree):
#    for file_name in file_names:
 #       try:
 #           os.rename(os.path.join(dir_path, file_name), os.path.join(dir_tree, file
 #       except OSError:
  #          print ("Could not move %s " % os.join(dir_path, file_name))
```

In [35]:
```
#def make_artist_table(path):

# Get file names

 #   files = [os.path.join(path,fn) for fn in os.listdir(path) if fn.endswith('.h5')]
 #   data = {'file':[], 'artist':[], 'title':[]}
#
```

```
            # Add artist and title data to dictionary
        #   for f in files:
         #      store = pd.HDFStore(f)
          #     title = store.root.metadata.songs.cols.title[0]
          #     artist = store.root.metadata.songs.cols.artist_name[0]
          #     data['file'].append(os.path.basename(f))
          #     data['title'].append(title.decode("utf-8"))
          #     data['artist'].append(artist.decode("utf-8"))
          #     store.close()

            # Convert dictionary to pandas DataFrame
          # df = pd.DataFrame.from_dict(data, orient='columns')
         #  df = df[['file', 'artist', 'title']]
          # return df
```

In [36]:
```
#path = '/Users/chirag/Downloads/MillionSongSubset'
#df = make_artist_table(path)

#df.tail()
```

In [80]:
```
mydata = pd.read_csv("year_prediction.csv", sep=",")
```

In [81]:
```
mydata.head()
```

Out[81]:

	label	TimbreAvg1	TimbreAvg2	TimbreAvg3	TimbreAvg4	TimbreAvg5	\
0	2001	49.94357	21.47114	73.07750	8.74861	-17.40628	
1	2001	48.73215	18.42930	70.32679	12.94636	-10.32437	
2	2001	50.95714	31.85602	55.81851	13.41693	-6.57898	
3	2001	48.24750	-1.89837	36.29772	2.58776	0.97170	
4	2001	50.97020	42.20998	67.09964	8.46791	-15.85279	

	TimbreAvg6	TimbreAvg7	TimbreAvg8	TimbreAvg9	...	\
0	-13.09905	-25.01202	-12.23257	7.83089	...	
1	-24.83777	8.76630	-0.92019	18.76548	...	
2	-18.54940	-3.27872	-2.35035	16.07017	...	
3	-26.21683	5.05097	-10.34124	3.55005	...	
4	-16.81409	-12.48207	-9.37636	12.63699	...	

	TimbreCovariance69	TimbreCovariance70	TimbreCovariance71	\
0	13.01620	-54.40548	58.99367	
1	5.66812	-19.68073	33.04964	
2	3.03800	26.05866	-50.92779	
3	34.57337	-171.70734	-16.96705	
4	9.92661	-55.95724	64.92712	

	TimbreCovariance72	TimbreCovariance73	TimbreCovariance74	\
0	15.37344	1.11144	-23.08793	
1	42.87836	-9.90378	-32.22788	
2	10.93792	-0.07568	43.20130	

```
3          -46.67617          -12.51516           82.58061
4          -17.72522           -1.49237           -7.50035

   TimbreCovariance75  TimbreCovariance76  TimbreCovariance77  \
0            68.40795            -1.82223           -27.46348
1            70.49388            12.04941            58.43453
2          -115.00698            -0.05859            39.67068
3           -72.08993             9.90558           199.62971
4            51.76631             7.88713            55.66926

   TimbreCovariance78
0             2.26327
1            26.92061
2            -0.66345
3            18.85382
4            28.74903

[5 rows x 91 columns]
```

```
In [5]: mydata.tail()
```

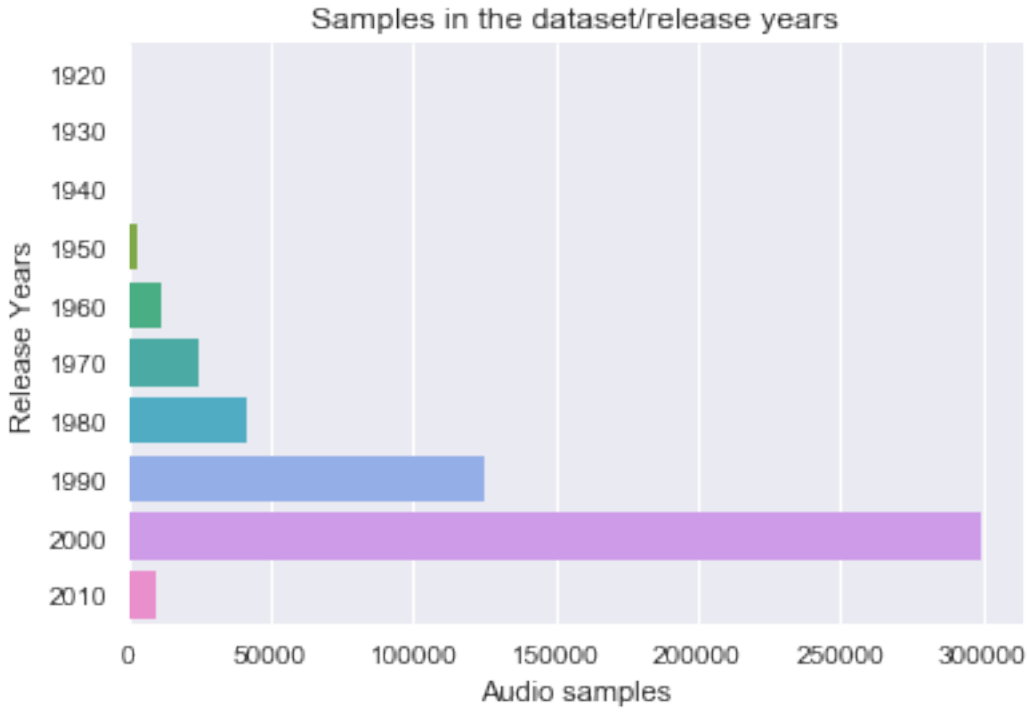
```
Out[5]:         label  TimbreAvg1  TimbreAvg2  TimbreAvg3  TimbreAvg4  TimbreAvg5  \
515340   2006    51.28467    45.88068    22.19582    -5.53319    -3.61835
515341   2006    49.87870    37.93125    18.65987    -3.63581   -27.75665
515342   2006    45.12852    12.65758   -38.72018     8.80882   -29.29985
515343   2006    44.16614    32.38368    -3.34971    -2.49165   -19.59278
515344   2005    51.85726    59.11655    26.39436    -5.46030   -20.69012

         TimbreAvg6  TimbreAvg7  TimbreAvg8  TimbreAvg9         ...          \
515340   -16.36914     2.12652     5.18160    -8.66890         ...
515341   -18.52988     7.76108     3.56109    -2.50351         ...
515342    -2.28706   -18.40424   -22.28726    -4.52429         ...
515343   -18.67098     8.78428     4.02039   -12.01230         ...
515344   -19.95528    -6.72771     2.29590    10.31018         ...

         TimbreCovariance69  TimbreCovariance70  TimbreCovariance71  \
515340             4.81440            -3.75991           -30.92584
515341            32.38589           -32.75535           -61.05473
515342           -18.73598           -71.15954          -123.98443
515343            67.16763           282.77624            -4.63677
515344           -11.50511           -69.18291            60.58456

         TimbreCovariance72  TimbreCovariance73  TimbreCovariance74  \
515340            26.33968            -5.03390            21.86037
515341            56.65182            15.29965            95.88193
515342           121.26989            10.89629            34.62409
515343           144.00125            21.62652           -29.72432
515344            28.64599            -4.39620           -64.56491
```

```
        TimbreCovariance75  TimbreCovariance76  TimbreCovariance77  \
515340           -142.29410             3.42901            -41.14721
515341            -10.63242            12.96552             92.11633
515342           -248.61020            -6.07171             53.96319
515343             71.47198            20.32240             14.83107
515344            -45.61012            -5.51512             32.35602

        TimbreCovariance78
515340            -15.46052
515341             10.88815
515342             -8.09364
515343             39.74909
515344             12.17352

[5 rows x 91 columns]
```

```
In [82]: mydata['label'] = mydata.label.apply(lambda year : year-(year%10))
```

```
In [41]: sns.countplot(y="label", data=mydata)
         plt.xlabel("Audio samples")
         plt.ylabel("Release Years")
         plt.title("Samples in the dataset/release years")
```

```
Out[41]: <matplotlib.text.Text at 0x1a0f4a90d0>
```



```
In [83]: print("(Samples, Features) {}".format(mydata.iloc[:,1:].shape))
         mydata.iloc[:,1:].describe()

(Samples, Features) (515345, 90)
```

Out[83]:

	TimbreAvg1	TimbreAvg2	TimbreAvg3	TimbreAvg4	\
count	515345.000000	515345.000000	515345.000000	515345.000000	
mean	43.387126	1.289554	8.658347	1.164124	
std	6.067558	51.580351	35.268585	16.322790	
min	1.749000	-337.092500	-301.005060	-154.183580	
25%	39.954690	-26.059520	-11.462710	-8.487500	
50%	44.258500	8.417850	10.476320	-0.652840	
75%	47.833890	36.124010	29.764820	8.787540	
max	61.970140	384.065730	322.851430	335.771820	

	TimbreAvg5	TimbreAvg6	TimbreAvg7	TimbreAvg8	\
count	515345.000000	515345.000000	515345.000000	515345.000000	
mean	-6.553601	-9.521975	-2.391089	-1.793236	
std	22.860785	12.857751	14.571873	7.963827	
min	-181.953370	-81.794290	-188.214000	-72.503850	
25%	-20.666450	-18.440990	-10.780600	-6.468420	
50%	-6.007770	-11.188390	-2.046670	-1.736450	
75%	7.741870	-2.388960	6.508580	2.913450	
max	262.068870	166.236890	172.402680	126.741270	

	TimbreAvg9	TimbreAvg10	...	TimbreCovariance69	\
count	515345.000000	515345.000000	...	515345.000000	
mean	3.727876	1.882385	...	15.755406	
std	10.582861	6.530232	...	32.099635	
min	-126.479040	-41.631660	...	-437.722030	
25%	-2.293660	-2.444850	...	-1.812650	
50%	3.822310	1.783520	...	9.171850	
75%	9.961820	6.147220	...	26.274480	
max	146.297950	60.345350	...	840.973380	

	TimbreCovariance70	TimbreCovariance71	TimbreCovariance72	\
count	515345.000000	515345.000000	515345.000000	
mean	-73.461500	41.542422	37.934119	
std	175.618889	122.228799	95.050631	
min	-4402.376440	-1810.689190	-3098.350310	
25%	-139.555160	-20.986900	-4.669540	
50%	-53.090060	28.791060	33.623630	
75%	13.478730	89.661770	77.785800	
max	4469.454870	3210.701700	1734.079690	

	TimbreCovariance73	TimbreCovariance74	TimbreCovariance75	\
count	515345.000000	515345.000000	515345.000000	

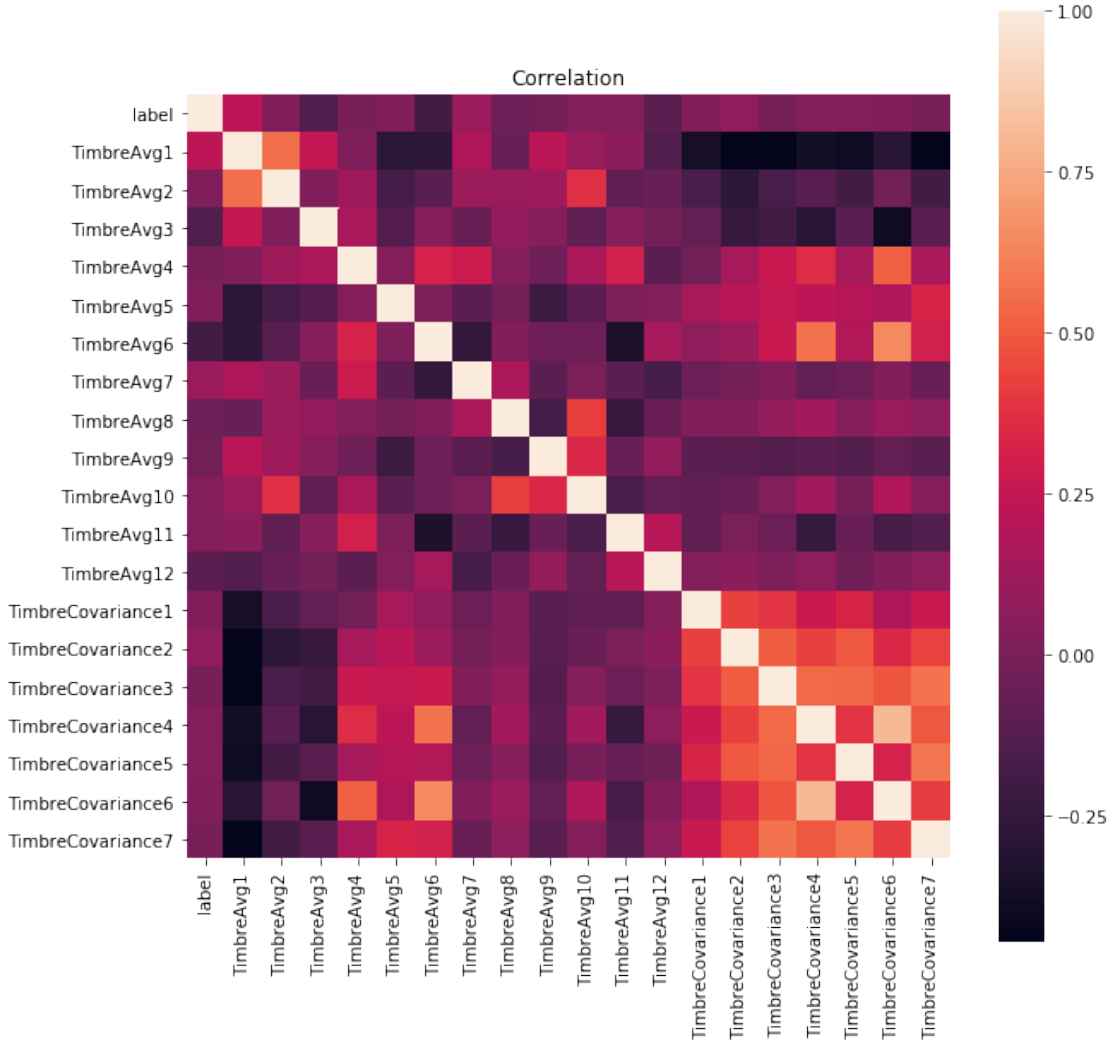
```
mean              0.315751          17.669213         -26.315336
std              16.161764         114.427905         173.977336
min            -341.789120       -3168.924570       -4319.992320
25%              -6.781590         -31.580610        -101.530300
50%               0.820840          15.598470         -21.204120
75%               8.470990          67.794960          52.389330
max             260.544900        3662.065650        2833.608950

       TimbreCovariance76  TimbreCovariance77  TimbreCovariance78
count       515345.000000       515345.000000       515345.000000
mean             4.458641           20.035136            1.329105
std             13.346557          185.558247           22.088576
min           -236.039260        -7458.378150         -381.424430
25%             -2.566090          -59.509270           -8.820210
50%              3.117640            7.759730            0.053050
75%              9.967740           86.351610            9.679520
max            463.419500         7393.398440          677.899630

[8 rows x 90 columns]
```

In [43]:
```
#Scaling Features

mydata.iloc[:,1:] = (mydata.iloc[:,1:]-mydata.iloc[:,1:].min())/(mydata.iloc[:,1:].max
mydata.iloc[:,1:].describe()
```

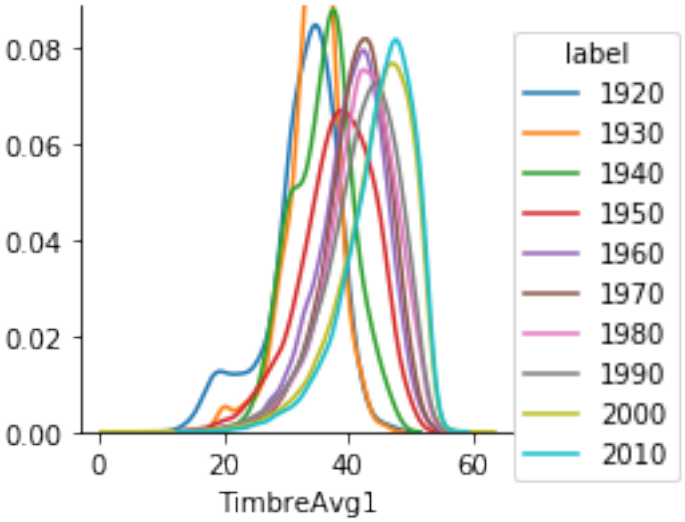
Out[43]:
```
          TimbreAvg1     TimbreAvg2     TimbreAvg3     TimbreAvg4  \
count  515345.000000  515345.000000  515345.000000  515345.000000
mean        0.691420       0.469220       0.496370       0.317065
std         0.100755       0.071524       0.056533       0.033315
min         0.000000       0.000000       0.000000       0.000000
25%         0.634423       0.431296       0.464117       0.297366
50%         0.705890       0.479105       0.499284       0.313357
75%         0.765261       0.517524       0.530202       0.332624
max         1.000000       1.000000       1.000000       1.000000

          TimbreAvg5     TimbreAvg6     TimbreAvg7     TimbreAvg8  \
count  515345.000000  515345.000000  515345.000000  515345.000000
mean        0.395025       0.291384       0.515292       0.354893
std         0.051486       0.051839       0.040408       0.039970
min         0.000000       0.000000       0.000000       0.000000
25%         0.363241       0.255425       0.492028       0.331428
50%         0.396254       0.284665       0.516247       0.355178
75%         0.427220       0.320143       0.539971       0.378515
max         1.000000       1.000000       1.000000       1.000000

          TimbreAvg9    TimbreAvg10         ...          TimbreCovariance69  \
count  515345.000000  515345.000000         ...               515345.000000
mean        0.477338       0.426704         ...                    0.354641
```

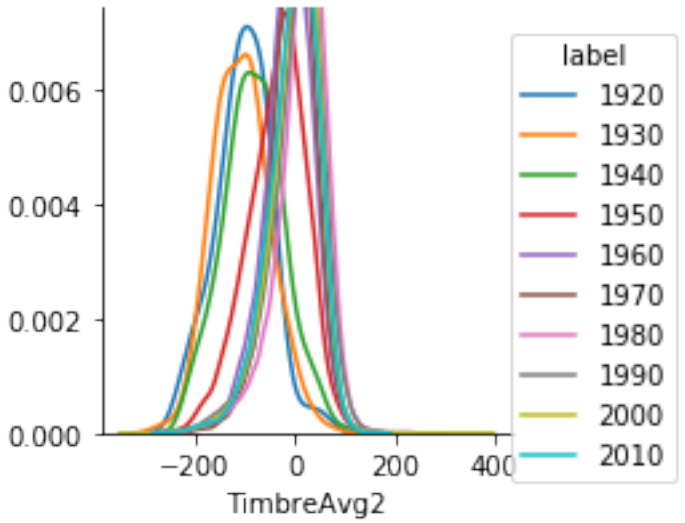
```
std         0.038797       0.064036         ...                    0.025103
min         0.000000       0.000000         ...                    0.000000
25%         0.455263       0.384271         ...                    0.340902
50%         0.477685       0.425735         ...                    0.349492
75%         0.500192       0.468526         ...                    0.362867
max         1.000000       1.000000         ...                    1.000000

       TimbreCovariance70  TimbreCovariance71  TimbreCovariance72  \
count       515345.000000       515345.000000       515345.000000
mean             0.487939            0.368868            0.649008
std              0.019795            0.024342            0.019669
min              0.000000            0.000000            0.000000
25%              0.480489            0.356416            0.640192
50%              0.490235            0.366329            0.648116
75%              0.497739            0.378451            0.657254
max              1.000000            1.000000            1.000000

       TimbreCovariance73  TimbreCovariance74  TimbreCovariance75  \
count       515345.000000       515345.000000       515345.000000
mean             0.567965            0.466491            0.600212
std              0.026832            0.016751            0.024320
min              0.000000            0.000000            0.000000
25%              0.556182            0.459281            0.589698
50%              0.568804            0.466188            0.600926
75%              0.581505            0.473829            0.611214
max              1.000000            1.000000            1.000000

       TimbreCovariance76  TimbreCovariance77  TimbreCovariance78
count       515345.000000       515345.000000       515345.000000
mean             0.343834            0.503537            0.361319
std              0.019081            0.012494            0.020852
min              0.000000            0.000000            0.000000
25%              0.333791            0.498181            0.351738
50%              0.341917            0.502710            0.360114
75%              0.351711            0.508002            0.369201
max              1.000000            1.000000            1.000000

[8 rows x 90 columns]
```

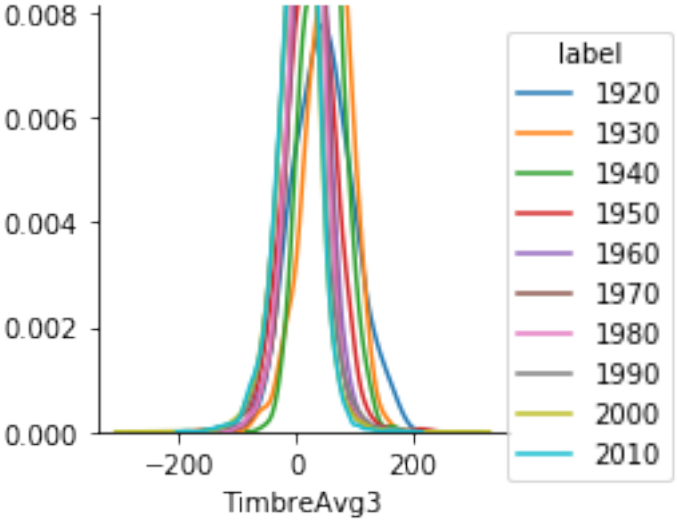
In [7]:
```
corr = mydata.iloc[:,:20].corr()
fig, ax = plt.subplots(figsize=(10,10))
plt.title("Correlation")
sns.heatmap(corr, square=True)
plt.show()
```

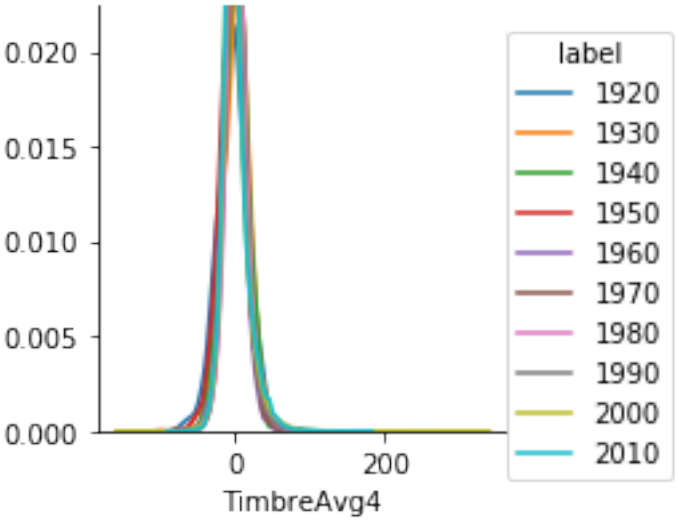


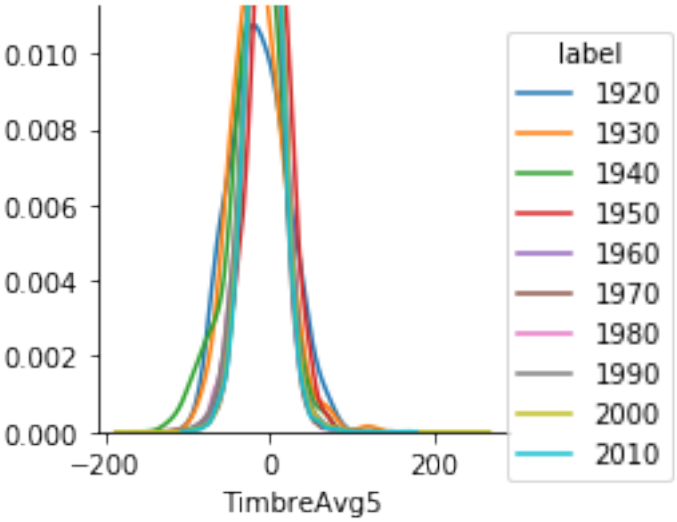

```
In [94]: df_t = mydata[mydata.label>1940]
         min_samples = df_t.label.value_counts().min()
         decades = df_t.label.unique()
         df_sampled = pd.DataFrame(columns=df_t.columns)
         for decade in decades:
             df_sampled = df_sampled.append(df_t[df_t.label==decade].sample(min_samples))
         df_sampled.label = df_sampled.label.astype(int)

In [95]: #Visualizing the impact of each feature on the target variable

         for component in mydata.columns[1:11]:
             sns.FacetGrid(mydata, hue="label", size=3) \
                .map(sns.kdeplot, component) \
                .add_legend()
             plt.show()
```

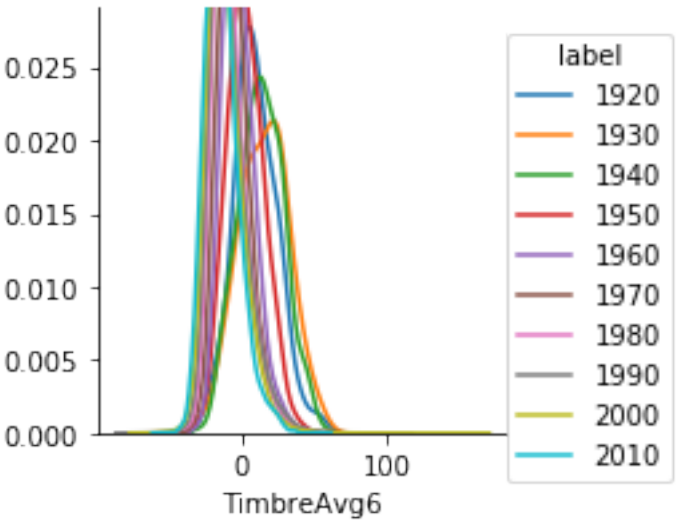

0.08
0.06
0.04
0.02
0.00
0
20
40
60
TimbreAvg1
label
1920
1930
1940
1950
1960
1970
1980
1990
2000
2010

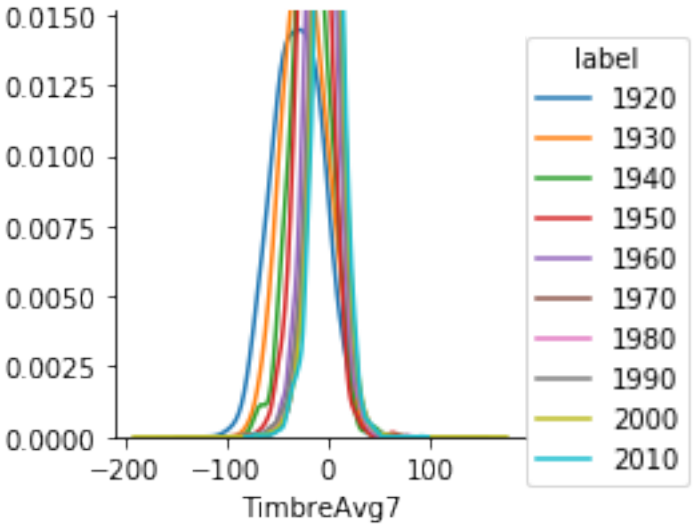

0.006
0.004
0.002
0.000
−200
0
200
400
TimbreAvg2
label
1920
1930
1940
1950
1960
1970
1980
1990
2000
2010

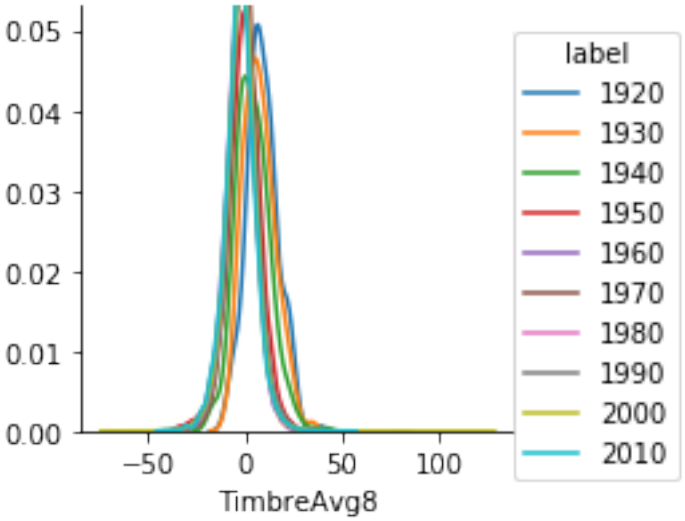

0.008
0.006
0.004
0.002
0.000
−200
0
200
TimbreAvg3
label
1920
1930
1940
1950
1960
1970
1980
1990
2000
2010

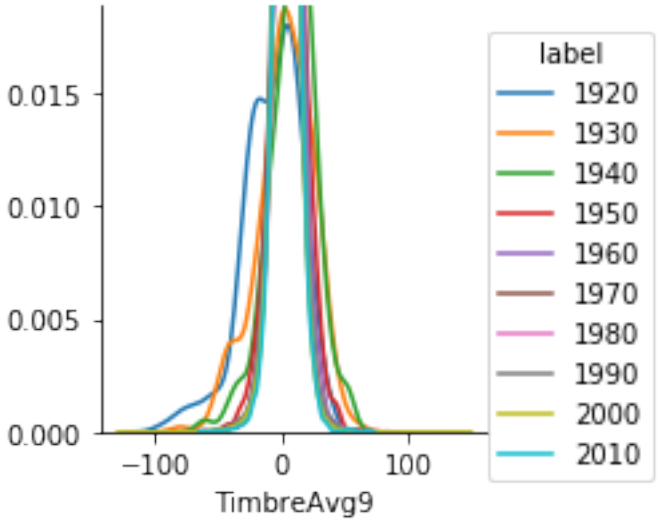

0.020
0.015
0.010
0.005
0.000
0
200
TimbreAvg4
label
1920
1930
1940
1950
1960
1970
1980
1990
2000
2010

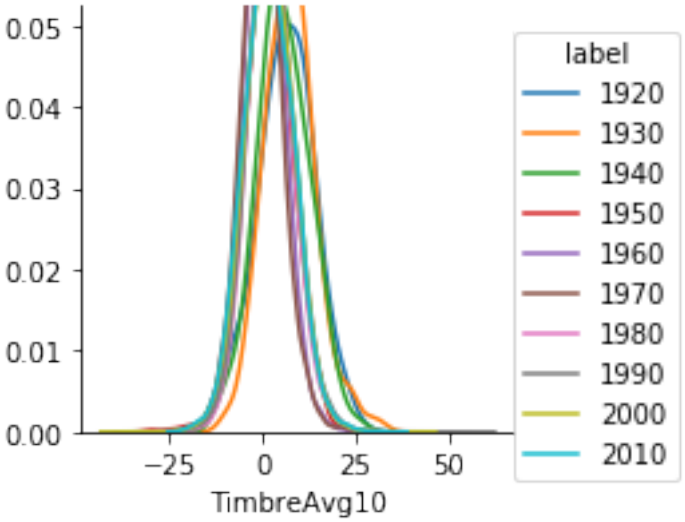

0.010
0.008
0.006
0.004
0.002
0.000
−200
0
200
TimbreAvg5
label
1920
1930
1940
1950
1960
1970
1980
1990
2000
2010


0.025
0.020
0.015
0.010
0.005
0.000
0
100
TimbreAvg6
label
1920
1930
1940
1950
1960
1970
1980
1990
2000
2010


0.0150
0.0125
0.0100
0.0075
0.0050
0.0025
0.0000
−200
−100
0
100
TimbreAvg7
label
1920
1930
1940
1950
1960
1970
1980
1990
2000
2010


0.05
0.04
0.03
0.02
0.01
0.00
−50
0
50
100
TimbreAvg8
label
1920
1930
1940
1950
1960
1970
1980
1990
2000
2010





In [96]:
```
#Dimensionality Reduction using PCA to reduce 90 features.
X = df_sampled.iloc[:,1:].values
y = df_sampled.iloc[:,0].values
print("X ", X.shape, ", y ", y.shape)

pca = PCA(n_components=20).fit(X)
X_pca = pca.transform(X)
```

```
('X ', (21714L, 91L), ', y ', (21714L,))
```

```
In [48]: print(sum(pca.explained_variance_ratio_))
```

```
0.791268887485
```

No use performing PCA as 20 principle components are unable to explain 95% of the variance explained.

```
In [49]: print len(Counter(mydata['label']))
```

```
10
```

1 Classification

```
In [97]: df_sampled = shuffle(df_sampled)
```

```
In [98]: mydata.data = df_sampled.iloc[:,1:].values
         mydata.target = df_sampled.iloc[:,0].values
         mydata.data.shape
         mydata.target.shape
```

```
Out[98]: (21714L,)
```

```
In [107]: X_train, X_test, y_train, y_test = train_test_split(mydata.data, mydata.target, test_
          clf = svm.SVC(kernel='rbf',C=100,gamma=0.001).fit(X_train,y_train)
          y_predict = clf.predict(X_test)
          scores_3 = cross_val_score(clf, X_train, y_train, cv=3)
          scores_3
```

```
Out[107]: array([ 0.14539357,  0.14528227,  0.14533965])
```

```
In [106]: print("Classification report for classifier %s:\n%s\n"
                % (clf, metrics.classification_report(y_test, y_predict)))
          cnf_matrix = metrics.confusion_matrix(y_test, y_predict)
```

```
Classification report for classifier SVC(C=100, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma=0.001, kernel='rbf',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False):
             precision    recall  f1-score   support

       1950       1.00      0.00      0.00       917
       1960       1.00      0.00      0.00       933
       1970       1.00      0.00      0.00       967
       1980       0.00      0.00      0.00       971
       1990       0.14      1.00      0.24       899
       2000       1.00      0.00      0.00       901
       2010       0.00      0.00      0.00       927
```

```
avg / total       0.59      0.14      0.03      6515
```

```
C:\Users\krish\Anaconda2\lib\site-packages\sklearn\metrics\classification.py:1135: UndefinedMet
  'precision', 'predicted', average, warn_for)
```

```
In [105]: cnf_matrix
```

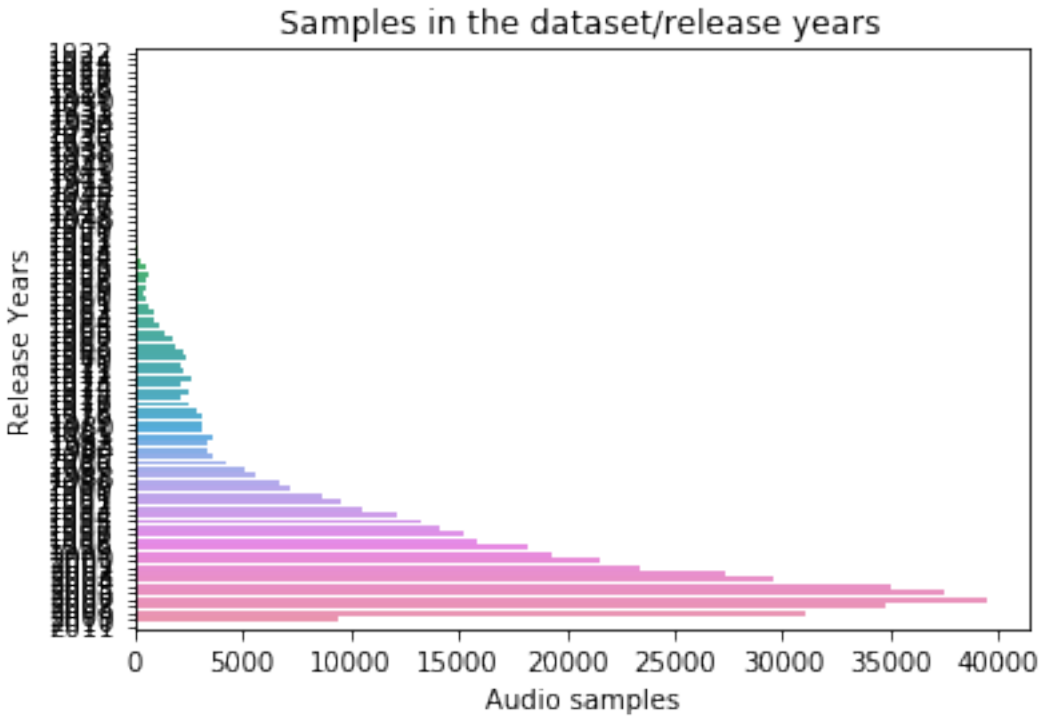
```
        ---------------------------------------------------------------------------

        NameError                                 Traceback (most recent call last)

        <ipython-input-105-454c1935cb46> in <module>()
    ----> 1 cnf_matrix


        NameError: name 'cnf_matrix' is not defined
```

```
In [102]: from sklearn.metrics import accuracy_score
          svm_acc = accuracy_score(y_test, y_predict)

In [103]: svm_acc

Out[103]: 0.13860322333077513

In [53]: logit = LogisticRegression().fit(X_train,y_train)
         y_pred = logit.predict(X_test)
         y_pred

Out[53]: array([1950, 1950, 1950, ..., 1980, 1950, 2010])

In [73]: accuracy = metrics.accuracy_score(y_test,y_pred)
         #print accuracy
         logit_score = cross_val_score(logit, X_train, y_train, cv=3)
         logit_score
```

```
0.406907137375
```

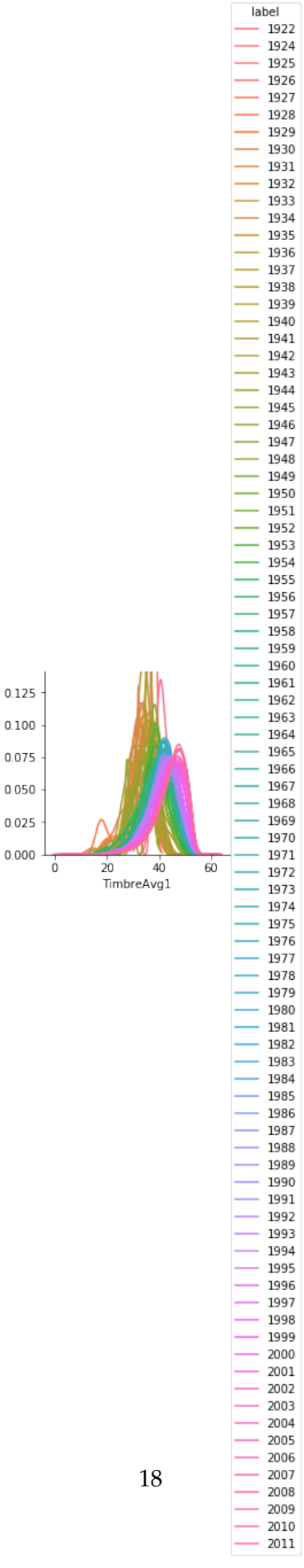
```
Out[73]: array([ 0.39238509,  0.39932886,  0.40027646])

In [77]: ridge = Ridge(alpha=100).fit(X_train,y_train)
         ridge_pred = ridge.predict(X_test)
         ridge_score = cross_val_score(ridge, X_train, y_train, cv=10)
         ridge_score
```

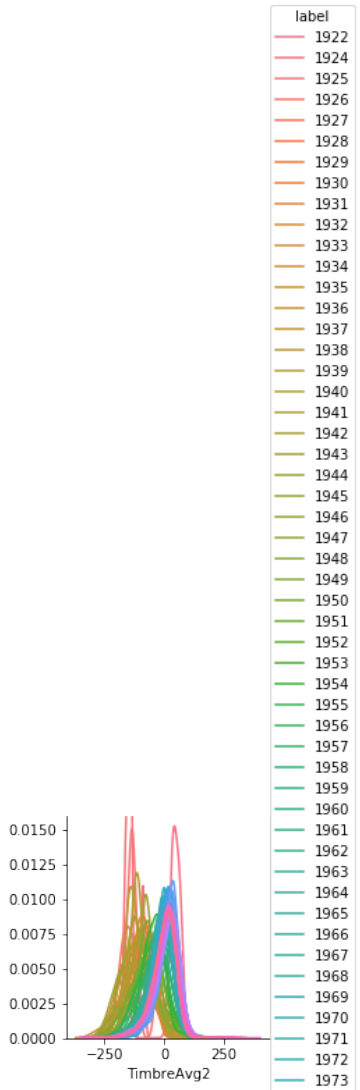
```
Out[77]: array([ 0.25683425,  0.22276448,  0.22525493,  0.23718342,  0.24175998,
                 0.23813866,  0.20670859,  0.23045411,  0.24134429,  0.21914523])

In [79]: rf = RandomForestClassifier(max_depth=2,).fit(X_train, y_train)
         rf_pred = rf.predict(X_test)
         rf_pred

Out[79]: array([1950, 1950, 1950, ..., 2010, 1950, 1970])

In [81]: rf_score = cross_val_score(rf, X_train,y_train,cv=3)
         rf_score

Out[81]: array([ 0.28151509,  0.28227398,  0.26875987])

In [82]: rf_accuracy = metrics.accuracy_score(y_test,rf_pred)
         rf_accuracy

Out[82]: 0.28027628549501149

In [8]: sns.countplot(y="label", data=mydata)
        plt.xlabel("Audio samples")
        plt.ylabel("Release Years")
        plt.title("Samples in the dataset/release years")

Out[8]: Text(0.5,1,u'Samples in the dataset/release years')
```

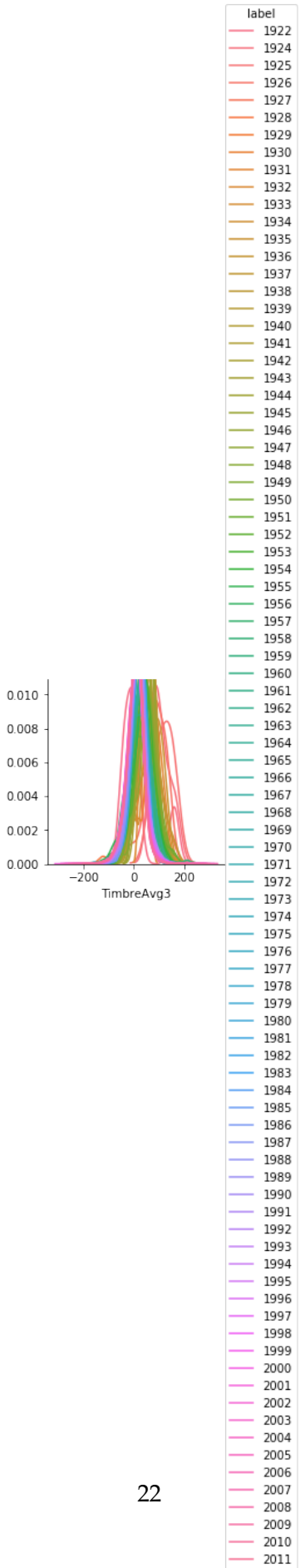


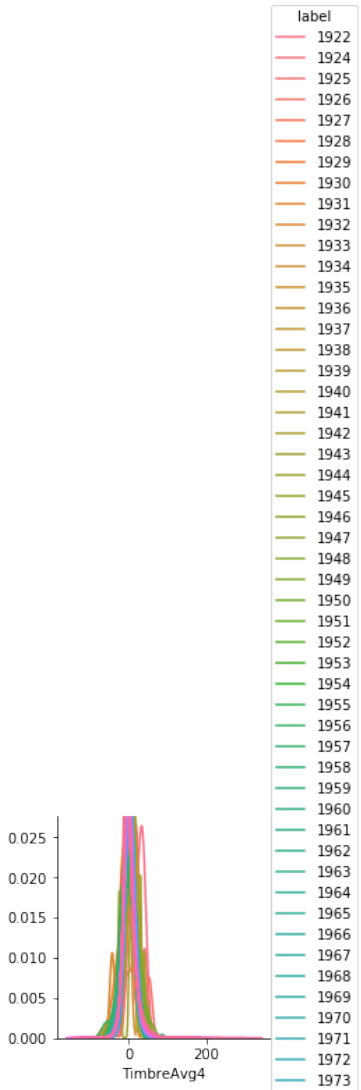
In [9]:
```
#Visualizing the impact of each feature on the target variable

        for component in mydata.columns[1:11]:
            sns.FacetGrid(mydata, hue="label", size=3) \
               .map(sns.kdeplot, component) \
               .add_legend()
            plt.show()
```

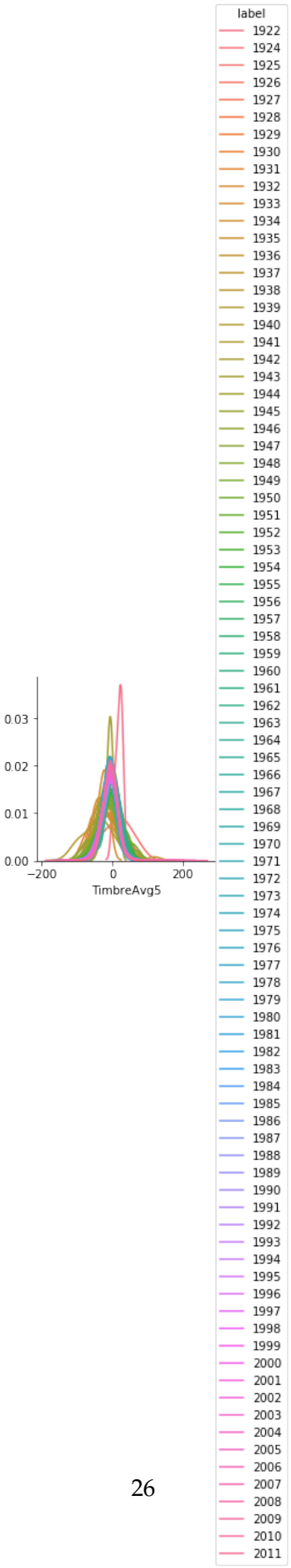
```
C:\Users\krish\Anaconda2\lib\site-packages\numpy\core\_methods.py:135: RuntimeWarning: Degrees
  keepdims=keepdims)
C:\Users\krish\Anaconda2\lib\site-packages\numpy\core\_methods.py:127: RuntimeWarning: invalid
  ret = ret.dtype.type(ret / rcount)
C:\Users\krish\Anaconda2\lib\site-packages\statsmodels\nonparametric\bandwidths.py:22: RuntimeW
  return np.minimum(np.std(X, axis=0, ddof=1), IQR)
C:\Users\krish\Anaconda2\lib\site-packages\statsmodels\nonparametric\kdetools.py:34: RuntimeWar
  FAC1 = 2*(np.pi*bw/RANGE)**2
```

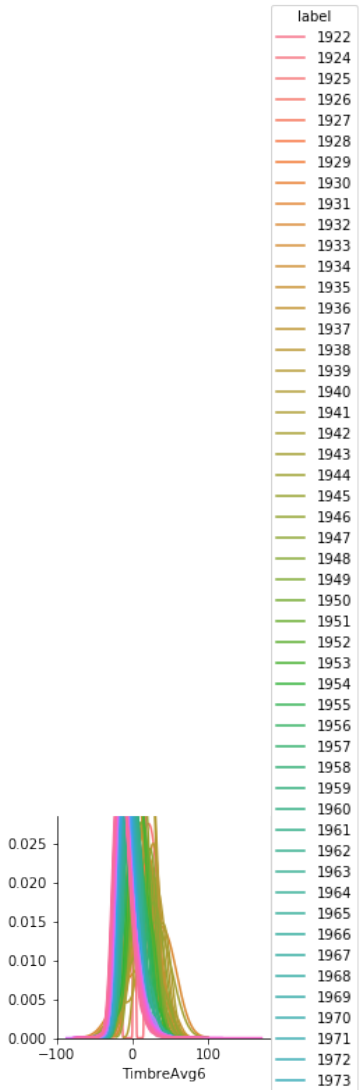

label
1922
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
0.125
0.100
0.075
0.050
0.025
0.000
0
20
40
60
TimbreAvg1

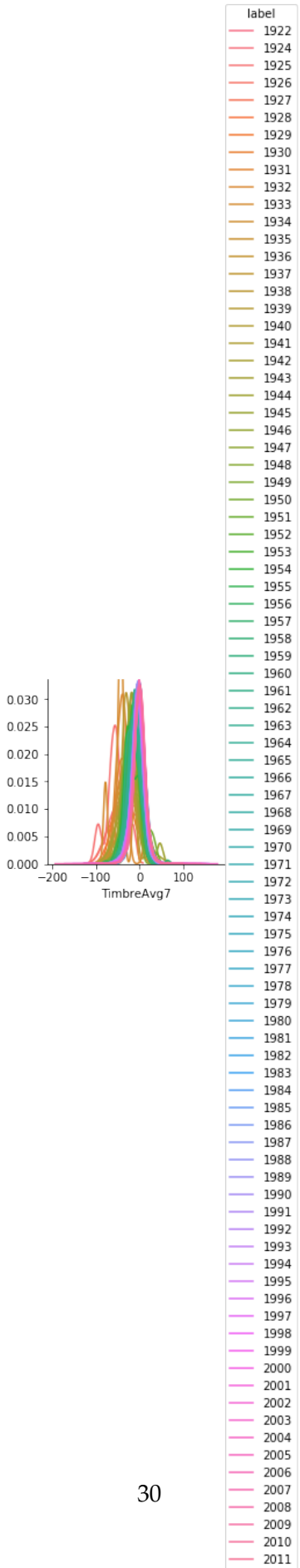

label
1922
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
0.0150
0.0125
0.0100
0.0075
0.0050
0.0025
0.0000
−250
0
250
TimbreAvg2

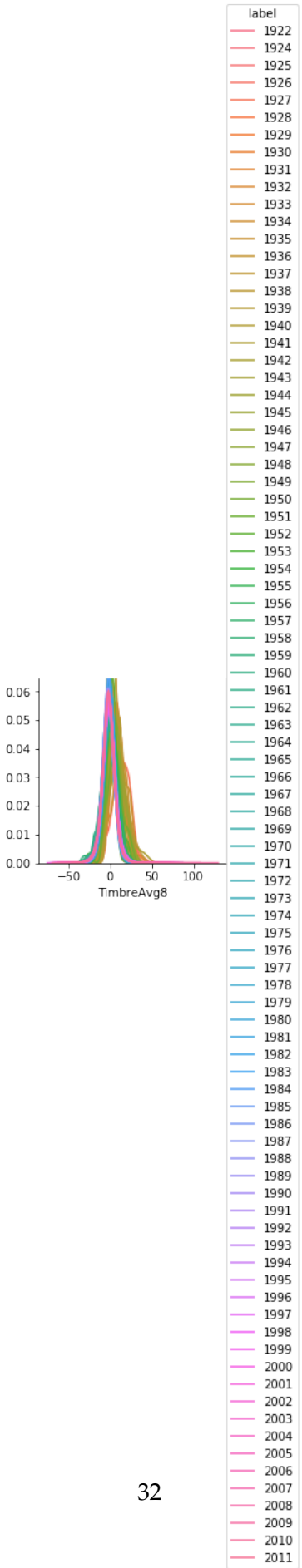

label
1922
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
0.010
0.008
0.006
0.004
0.002
0.000
−200
0
200
TimbreAvg3

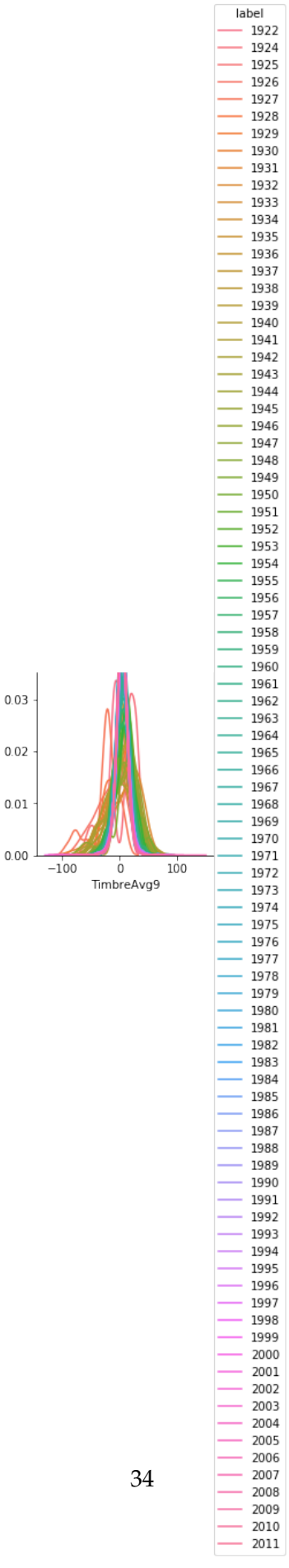

label
1922
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
0.025
0.020
0.015
0.010
0.005
0.000
0
200
TimbreAvg4

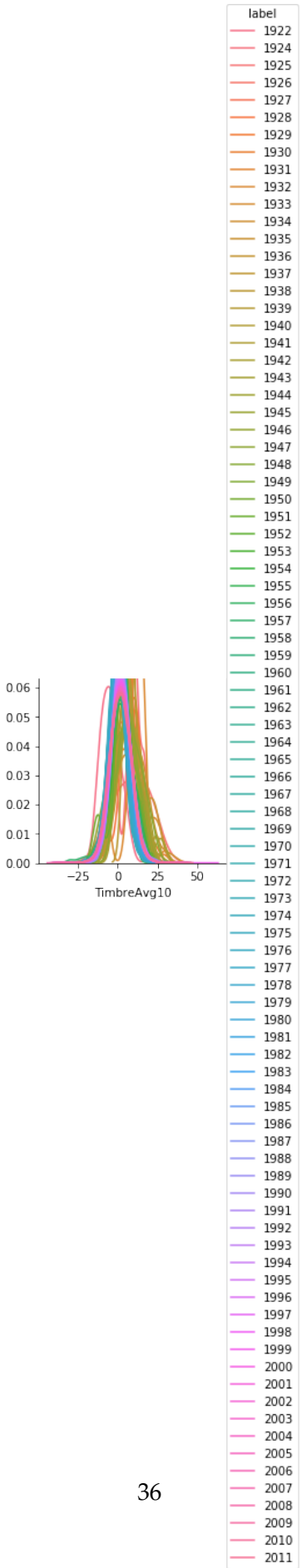

label
1922
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
0.03
0.02
0.01
0.00
−200
0
200
TimbreAvg5


label
1922
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
0.025
0.020
0.015
0.010
0.005
0.000
−100
0
100
TimbreAvg6


label
1922
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
0.030
0.025
0.020
0.015
0.010
0.005
0.000
−200
−100
0
100
TimbreAvg7


label
1922
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
0.06
0.05
0.04
0.03
0.02
0.01
0.00
−50
0
50
100
TimbreAvg8


label
1922
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
0.03
0.02
0.01
0.00
−100
0
100
TimbreAvg9


label
1922
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
0.06
0.05
0.04
0.03
0.02
0.01
0.00
−25
0
25
50
TimbreAvg10

```
In [10]: #Dimensionality Reduction using PCA to reduce 90 features.
         X = mydata.iloc[:,1:].values
         Y = mydata.iloc[:,0].values
         print("X ", X.shape, ", y ", Y.shape)

         pca = PCA(n_components=20).fit(X)
         X_pca = pca.transform(X)
```

```
('X ', (515345L, 90L), ', y ', (515345L,))
```

```
In [11]: print(sum(pca.explained_variance_ratio_))
```

```
0.941676129591
```

```
In [12]: principal_components =  pca.components_
```

```
In [13]: from sklearn.linear_model import LinearRegression
```

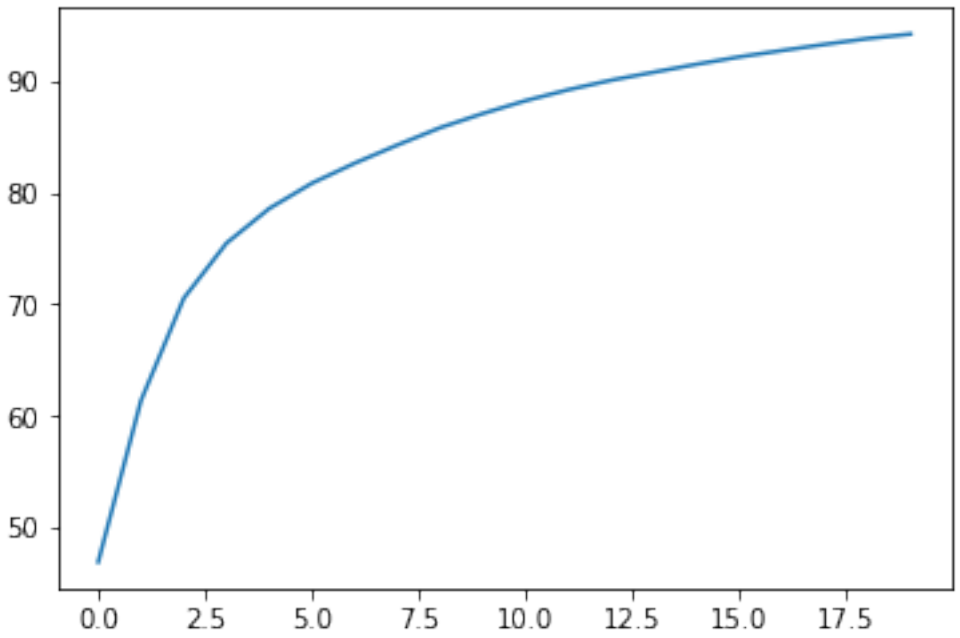
```
In [57]: model = LinearRegression()
         model.fit(X,Y)
```

```
Out[57]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

```
In [17]: var1=np.cumsum(np.round(pca.explained_variance_ratio_, decimals=4)*100)
```

```
In [18]: plt.plot(var1)
```

```
Out[18]: [<matplotlib.lines.Line2D at 0x268d23c8>]
```



```
In [21]: model.predict(X)[0:5]

Out[21]: array([ 1997.17782585,  1999.32323446,  1997.3085077 ,  2001.18228451,
                 1999.42810508])

In [23]: predicted = model.predict(X)

In [24]:

Out[24]: 515345

In [42]: X_train, X_test, y_train, y_test = train_test_split(mydata.as_matrix(), mydata['label

In [43]: model = LinearRegression()
         model.fit(X_train,y_train)

Out[43]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)

In [45]: predicted = model.predict(X_test)

In [46]: predicted

Out[46]: array([ 2007.,  1993.,  1993., ...,  2007.,  1999.,  2000.])

In [59]: print(model.coef_)

[  8.75418124e-01  -5.63271164e-02  -4.36490654e-02   3.35326082e-03
  -1.47468748e-02  -2.20071726e-01  -6.73919081e-03  -1.00897584e-01
  -7.04728814e-02   2.50691291e-02  -1.65703623e-01  -1.85480169e-03
   4.70139734e-02   3.55100627e-04  -4.22581025e-04   5.99188699e-04
   4.76557935e-04   1.46665847e-03   1.92445360e-03   2.12832389e-03
   7.69871550e-04  -4.02569165e-04   7.53934986e-03   2.81198502e-03
  -3.55560501e-03   7.11382813e-05   1.58941497e-03   5.29431505e-04
   8.74511415e-04  -3.04182481e-04  -1.40497004e-03  -1.40130314e-03
  -5.55968147e-03   2.47236496e-03   1.84963210e-03  -5.29413753e-03
  -2.77273205e-04   6.79201032e-04   1.36516401e-03  -1.71045701e-03
  -1.99137811e-03  -7.64154508e-04  -1.40252362e-03  -2.35913075e-03
  -3.17985554e-03   6.81262742e-03   4.56071335e-04  -2.07494336e-03
   2.75184183e-04   1.94121271e-03   2.20058312e-04  -1.60491529e-03
   1.97091583e-03   4.90779335e-04  -8.43754307e-05   1.62872728e-04
  -1.89762160e-03   1.94046249e-03  -1.30448733e-03   2.33234081e-04
  -3.03171007e-03  -1.87987844e-03  -7.76853586e-03   1.19021991e-03
  -2.02504103e-03   6.59671311e-04  -1.93391474e-04  -4.27537529e-04
  -4.25100115e-03  -5.08404783e-03  -1.06135702e-03   2.37762676e-04
   6.89131114e-04   3.98716357e-03   3.00119774e-03   1.52110273e-02
   1.99614222e-04  -4.42313018e-03  -4.24323177e-05  -1.51553391e-04
  -8.27829011e-04  -5.56825270e-04   1.37235285e-03   9.96491033e-04
   2.61366380e-02   1.07412669e-04   1.16490865e-03  -3.11970706e-02
  -1.38056076e-03  -1.61549972e-03]
```

```
In [51]: print model.intercept_

3.86535248253e-11


In [58]: print model.score(X,Y)

0.237000619844


In [60]: Y

Out[60]: array([2001, 2001, 2001, ..., 2006, 2006, 2005], dtype=int64)

In [61]: X

Out[61]: array([[  4.99435700e+01,   2.14711400e+01,   7.30775000e+01, ...,
                  -1.82223000e+00,  -2.74634800e+01,   2.26327000e+00],
                [  4.87321500e+01,   1.84293000e+01,   7.03267900e+01, ...,
                   1.20494100e+01,   5.84345300e+01,   2.69206100e+01],
                [  5.09571400e+01,   3.18560200e+01,   5.58185100e+01, ...,
                  -5.85900000e-02,   3.96706800e+01,  -6.63450000e-01],
                ...,
                [  4.51285200e+01,   1.26575800e+01,  -3.87201800e+01, ...,
                  -6.07171000e+00,   5.39631900e+01,  -8.09364000e+00],
                [  4.41661400e+01,   3.23836800e+01,  -3.34971000e+00, ...,
                   2.03224000e+01,   1.48310700e+01,   3.97490900e+01],
                [  5.18572600e+01,   5.91165500e+01,   2.63943600e+01, ...,
                  -5.51512000e+00,   3.23560200e+01,   1.21735200e+01]])

In [62]: import pandas as pd
         import numpy as np
         import itertools
         import time
         import statsmodels.api as sm
         import matplotlib.pyplot as plt

C:\Users\krish\Anaconda2\lib\site-packages\statsmodels\compat\pandas.py:56: FutureWarning: The
  from pandas.core import datetools


In [77]: def processSubset(feature_set):
             # Fit model on feature_set and calculate RSS
             model = sm.OLS(Y,X[list(feature_set)])
             regr = model.fit()
             RSS = ((regr.predict(X[list(feature_set)]) - Y) ** 2).sum()
             return {"model":regr, "RSS":RSS}
```

```
In [64]: def getBest(k):

             tic = time.time()

             results = []

             for combo in itertools.combinations(X.columns, k):
                 results.append(processSubset(combo))

             # Wrap everything up in a nice dataframe
             models = pd.DataFrame(results)

             # Choose the model with the highest RSS
             best_model = models.loc[models['RSS'].argmin()]

             toc = time.time()
             print("Processed", models.shape[0], "models on", k, "predictors in", (toc-tic), "s

             # Return the best model, along with some other useful information about the model
             return best_model

In [74]: def forward(predictors):

             # Pull out predictors we still need to process
             remaining_predictors = [p for p in df.columns if p not in predictors]

             tic = time.time()

             results = []

             for p in remaining_predictors:
                 results.append(processSubset(predictors+[p]))

             # Wrap everything up in a nice dataframe
             models = pd.DataFrame(results)

             # Choose the model with the highest RSS
             best_model = models.loc[models['RSS'].argmin()]

             toc = time.time()
             print("Processed ", models.shape[0], "models on", len(predictors)+1, "predictors i

             # Return the best model, along with some other useful information about the model
             return best_model

In [70]: df = mydata

In [72]: del df['label']
```

```
In [84]: mydata['predicted'] = 2000

In [87]: y_actual = mydata['label'].astype(int)
         y_pred = mydata['predicted'].astype(int)

In [90]: from sklearn.metrics import accuracy_score
         baseline_acc = accuracy_score(y_actual, y_pred)

In [92]: baseline_acc

Out[92]: 0.5801996720643452
```