# IE 7275 - Sec 4 - Case Study Implementation - Group 5

April 23, 2023

[1]:
```python
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import math
import seaborn as sns
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
```

[2]:
```python
df = pd.read_csv('C:/Users/anjal/Documents/DataFiles/GTZAN/Data/features_3_sec.
 ↪csv')
```

[3]:
```python
df.shape
```

[3]:
```
(9990, 60)
```

[4]:
```python
df.head()
```

[4]:
```
            filename  length  chroma_stft_mean  chroma_stft_var  rms_mean  \
0  blues.00000.0.wav   66149          0.335406         0.091048  0.130405
1  blues.00000.1.wav   66149          0.343065         0.086147  0.112699
2  blues.00000.2.wav   66149          0.346815         0.092243  0.132003
3  blues.00000.3.wav   66149          0.363639         0.086856  0.132565
4  blues.00000.4.wav   66149          0.335579         0.088129  0.143289

    rms_var  spectral_centroid_mean  spectral_centroid_var  \
0  0.003521             1773.065032          167541.630869
1  0.001450             1816.693777           90525.690866
2  0.004620             1788.539719          111407.437613
3  0.002448             1655.289045          111952.284517
4  0.001701             1630.656199           79667.267654

   spectral_bandwidth_mean  spectral_bandwidth_var  …  mfcc16_var  \
0              1972.744388           117335.771563  …   39.687145
1              2010.051501            65671.875673  …   64.748276
2              2084.565132            75124.921716  …   67.336563
3              1960.039988            82913.639269  …   47.739452
4              1948.503884            60204.020268  …   30.336359
```

```
   mfcc17_mean  mfcc17_var  mfcc18_mean  mfcc18_var  mfcc19_mean  mfcc19_var  \
0    -3.241280   36.488243     0.722209   38.099152    -5.050335   33.618073
1    -6.055294   40.677654     0.159015   51.264091    -2.837699   97.030830
2    -1.768610   28.348579     2.378768   45.717648    -1.938424   53.050835
3    -3.841155   28.337118     1.218588   34.770935    -3.580352   50.836224
4     0.664582   45.880913     1.689446   51.363583    -3.392489   26.738789

   mfcc20_mean  mfcc20_var  label
0    -0.243027   43.771767  blues
1     5.784063   59.943081  blues
2     2.517375   33.105122  blues
3     3.630866   32.023678  blues
4     0.536961   29.146694  blues

[5 rows x 60 columns]
```

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

[5]:
```
        length  chroma_stft_mean  chroma_stft_var     rms_mean       rms_var  \
count   9990.0       9990.000000      9990.000000  9990.000000  9.990000e+03
mean   66149.0          0.379534         0.084876     0.130859  2.676388e-03
std        0.0          0.090466         0.009637     0.068545  3.585628e-03
min    66149.0          0.107108         0.015345     0.000953  4.379535e-08
25%    66149.0          0.315698         0.079833     0.083782  6.145900e-04
50%    66149.0          0.384741         0.085108     0.121253  1.491318e-03
75%    66149.0          0.442443         0.091092     0.176328  3.130862e-03
max    66149.0          0.749481         0.120964     0.442567  3.261522e-02

       spectral_centroid_mean  spectral_centroid_var  spectral_bandwidth_mean  \
count             9990.000000           9.990000e+03              9990.000000
mean              2199.219431           4.166727e+05              2241.385959
std                751.860611           4.349644e+05               543.854449
min                472.741636           8.118813e+02               499.162910
25%               1630.680158           1.231961e+05              1887.455790
50%               2208.628236           2.650692e+05              2230.575595
75%               2712.581884           5.624152e+05              2588.340505
max               5432.534406           4.794119e+06              3708.147554

       spectral_bandwidth_var  rolloff_mean  …  mfcc16_mean  mfcc16_var  \
count            9.990000e+03   9990.000000  …  9990.000000  9990.000000
mean             1.182711e+05   4566.076592  …     1.448240    49.988755
std              1.013505e+05   1642.065335  …     5.735149    34.442816
min              1.183520e+03    658.336276  …   -26.850016     1.325786
25%              4.876553e+04   3378.311110  …    -2.227478    29.584894
50%              8.996072e+04   4631.377892  …     1.461623    41.702393
75%              1.585674e+05   5591.634521  …     5.149752    59.274619
max              1.235143e+06   9487.446477  …    39.144405   683.932556
```

```
       mfcc17_mean   mfcc17_var  mfcc18_mean   mfcc18_var  mfcc19_mean  \
count  9990.000000  9990.000000  9990.000000  9990.000000  9990.000000
mean     -4.198706    51.962753     0.739943    52.488851    -2.497306
std       5.677379    36.400669     5.181313    38.177120     5.111799
min     -27.809795     1.624544   -20.733809     3.437439   -27.448456
25%      -7.951722    29.863448    -2.516638    29.636197    -5.734123
50%      -4.443021    42.393583     0.733772    41.831377    -2.702366
75%      -0.726945    61.676964     3.888734    62.033906     0.514246
max      34.048843   529.363342    36.970322   629.729797    31.365425

        mfcc19_var  mfcc20_mean   mfcc20_var
count  9990.000000  9990.000000  9990.000000
mean     54.973829    -0.917584    57.322614
std      41.585677     5.253243    46.444212
min       3.065302   -35.640659     0.282131
25%      30.496412    -4.004475    30.011365
50%      43.435253    -1.030939    44.332155
75%      65.328602     2.216603    68.210421
max    1143.230591    34.212101   910.473206

[8 rows x 58 columns]
```

[6]:
```python
missing_values_count = df.isnull().sum()
print(missing_values_count)
```

```
filename                   0
length                     0
chroma_stft_mean           0
chroma_stft_var            0
rms_mean                   0
rms_var                    0
spectral_centroid_mean     0
spectral_centroid_var      0
spectral_bandwidth_mean    0
spectral_bandwidth_var     0
rolloff_mean               0
rolloff_var                0
zero_crossing_rate_mean    0
zero_crossing_rate_var     0
harmony_mean               0
harmony_var                0
perceptr_mean              0
perceptr_var               0
tempo                      0
mfcc1_mean                 0
mfcc1_var                  0
mfcc2_mean                 0
```

```
mfcc2_var                  0
mfcc3_mean                 0
mfcc3_var                  0
mfcc4_mean                 0
mfcc4_var                  0
mfcc5_mean                 0
mfcc5_var                  0
mfcc6_mean                 0
mfcc6_var                  0
mfcc7_mean                 0
mfcc7_var                  0
mfcc8_mean                 0
mfcc8_var                  0
mfcc9_mean                 0
mfcc9_var                  0
mfcc10_mean                0
mfcc10_var                 0
mfcc11_mean                0
mfcc11_var                 0
mfcc12_mean                0
mfcc12_var                 0
mfcc13_mean                0
mfcc13_var                 0
mfcc14_mean                0
mfcc14_var                 0
mfcc15_mean                0
mfcc15_var                 0
mfcc16_mean                0
mfcc16_var                 0
mfcc17_mean                0
mfcc17_var                 0
mfcc18_mean                0
mfcc18_var                 0
mfcc19_mean                0
mfcc19_var                 0
mfcc20_mean                0
mfcc20_var                 0
label                      0
dtype: int64
```

[7]:
```
df = df.drop('filename', axis=1)
```

[8]:
```
df = df.drop('length', axis=1)
```

[9]:
```
df['label'].unique()
```

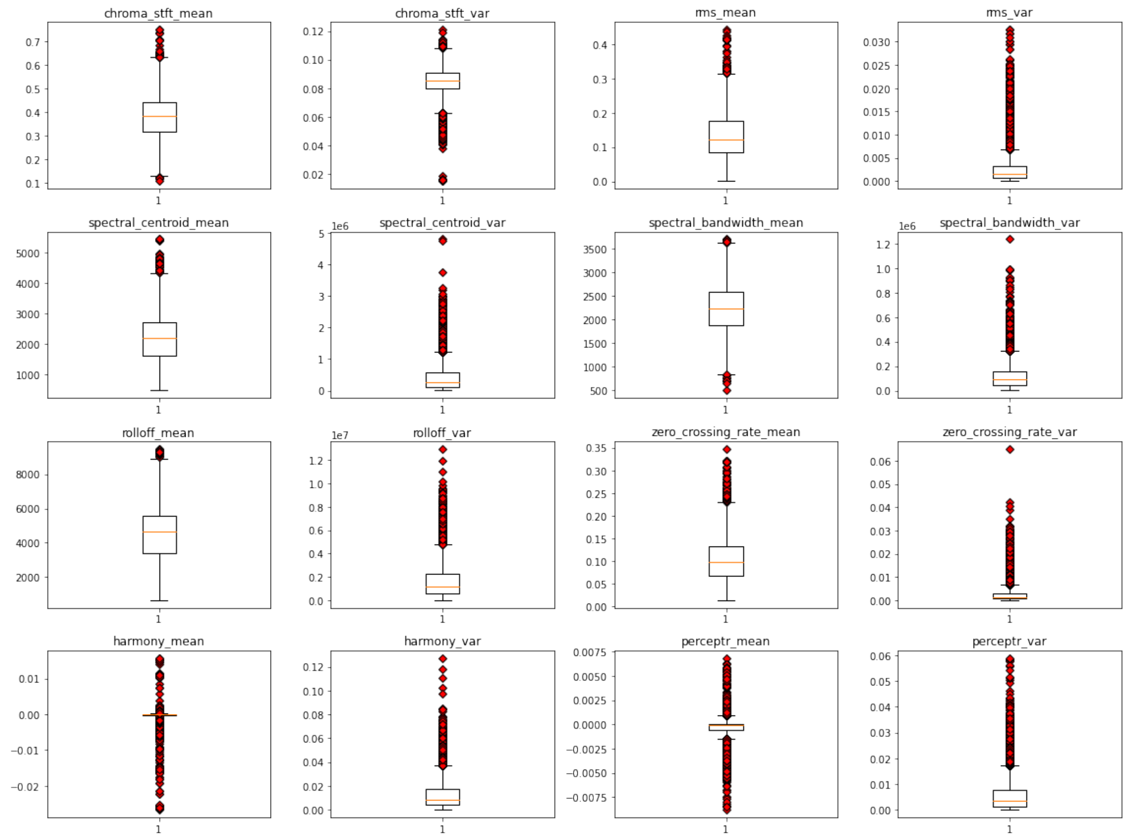
[9]:
```
array(['blues', 'classical', 'country', 'disco', 'hiphop', 'jazz',
       'metal', 'pop', 'reggae', 'rock'], dtype=object)
```

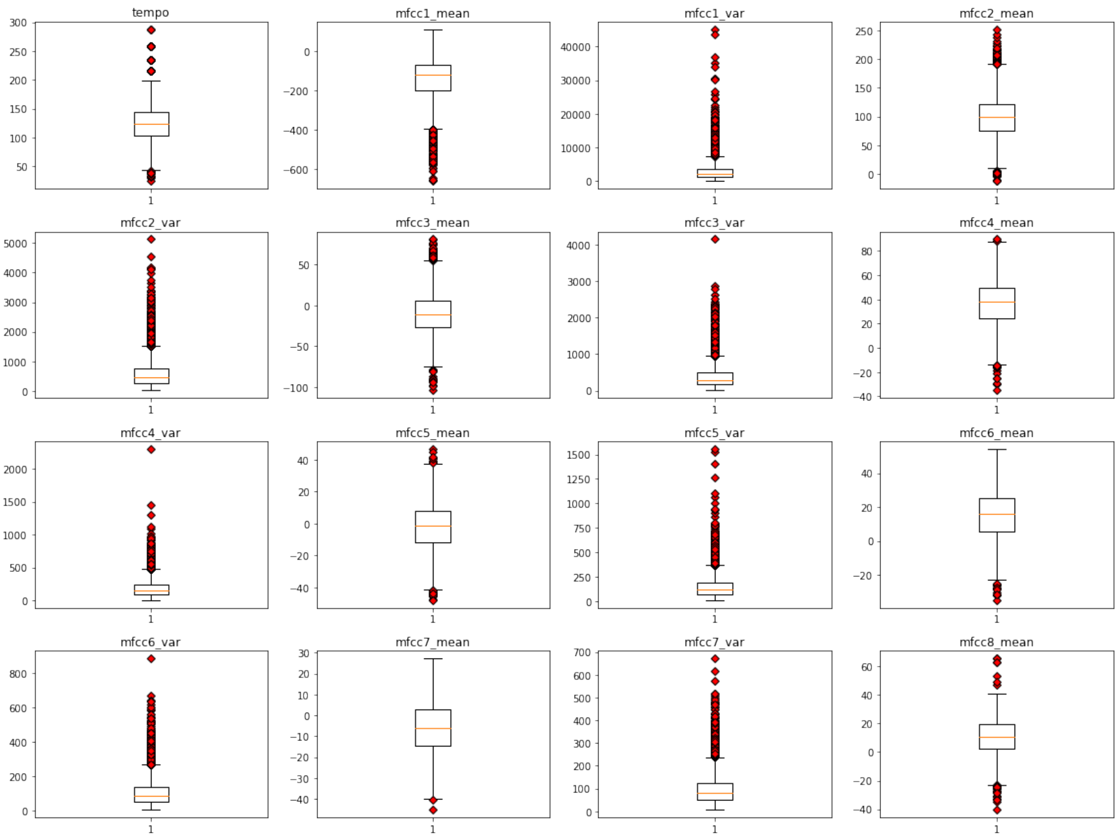
[10]:
```
label_encoding = {'blues': 1, 'classical': 2, 'country': 3, 'disco': 4,␣
↪'hiphop': 5, 'jazz': 6, 'metal': 7, 'pop': 8, 'reggae': 9, 'rock': 10}
df['label'] = df['label'].replace(label_encoding)
```

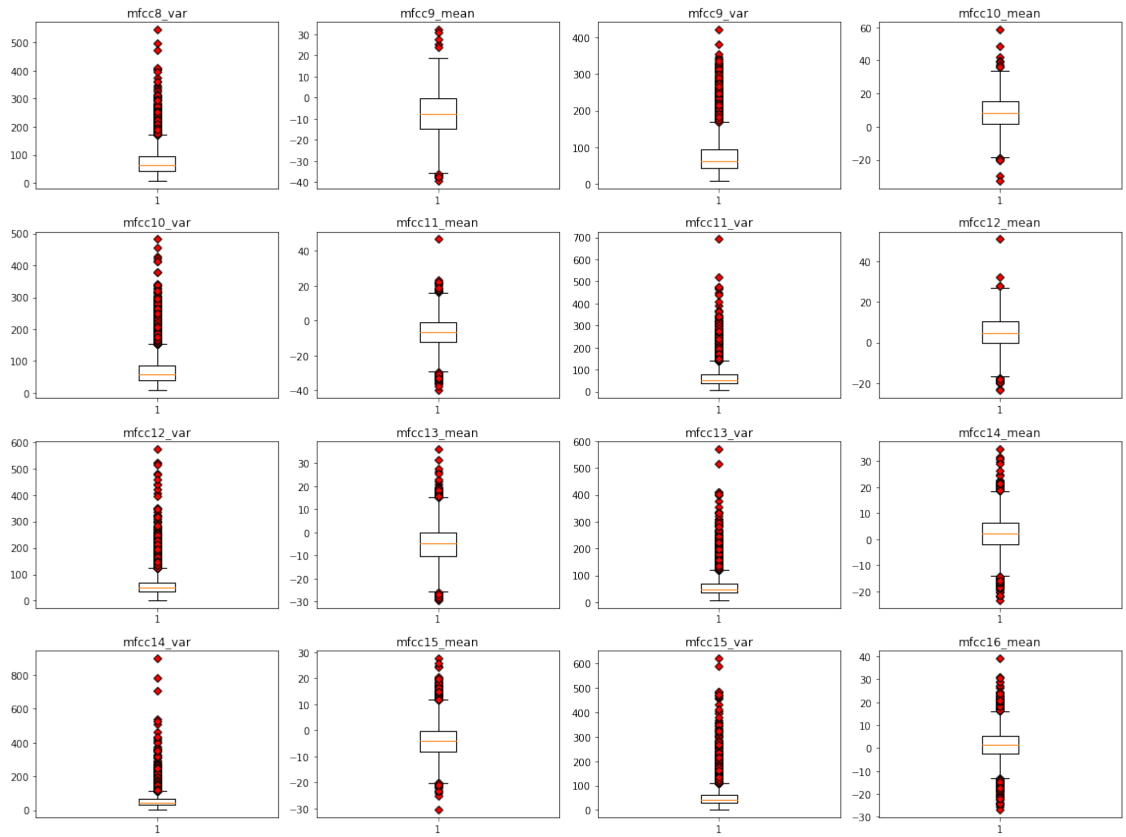
[11]:
```
df['label'].unique()
```

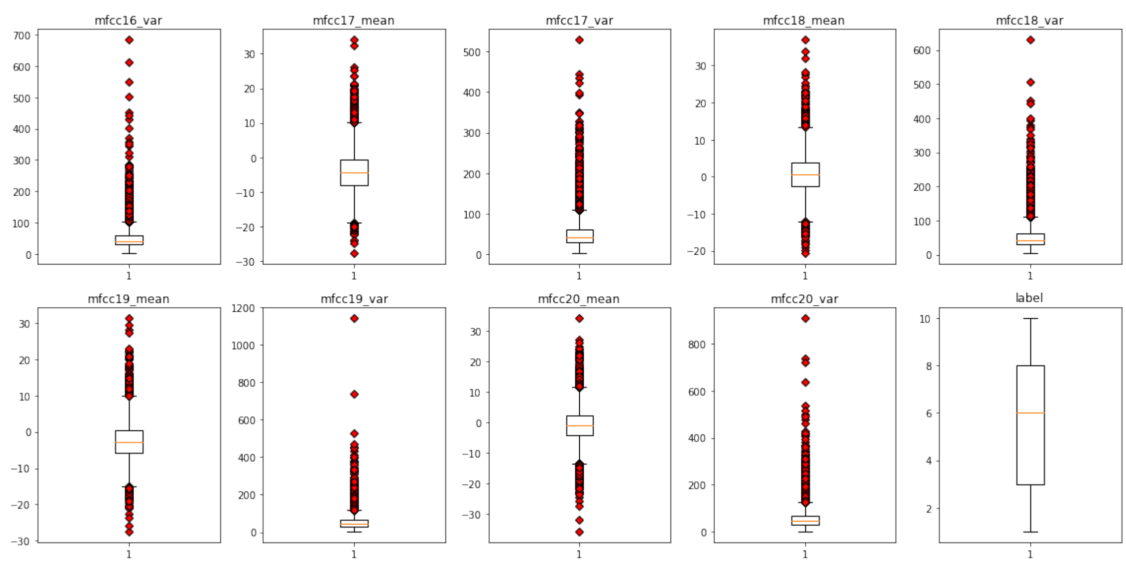
[11]:
```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10], dtype=int64)
```

[12]:
```
for i in range(0, 48, 16):
    fig, axs = plt.subplots(4, 4, figsize=(16, 12))
    for j, ax in enumerate(axs.flatten()):
        if i+j < 48:
            boxplot = ax.boxplot(df.iloc[:, i+j],␣
↪flierprops=dict(markerfacecolor='r', marker='D'))
            ax.set_title(df.columns[i+j])
    plt.tight_layout()

fig, axs = plt.subplots(2, 5, figsize=(16, 8))
for i, ax in enumerate(axs.flatten()):
    if i < 10:
        boxplot = ax.boxplot(df.iloc[:, i+48],␣
↪flierprops=dict(markerfacecolor='r', marker='D'))
        ax.set_title(df.columns[i+48])
plt.tight_layout()
```


chroma_stft_mean
chroma_stft_var
rms_mean
rms_var
spectral_centroid_mean
spectral_centroid_var
spectral_bandwidth_mean
spectral_bandwidth_var
rolloff_mean
rolloff_var
zero_crossing_rate_mean
zero_crossing_rate_var
harmony_mean
harmony_var
perceptr_mean
perceptr_var


tempo
mfcc1_mean
mfcc1_var
mfcc2_mean
mfcc2_var
mfcc3_mean
mfcc3_var
mfcc4_mean
mfcc4_var
mfcc5_mean
mfcc5_var
mfcc6_mean
mfcc6_var
mfcc7_mean
mfcc7_var
mfcc8_mean




[13]:
```
import numpy as np
Q1 = df.quantile(0.25)
```

```
Q3 = df.quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5*IQR
upper_bound = Q3 + 1.5*IQR


num_outliers = (df < lower_bound) | (df > upper_bound)
outlier_count = num_outliers.sum()
outlier_percent = (outlier_count / len(df)) * 100
outlier_percent_sorted = outlier_percent.sort_values(ascending=False)

print(outlier_percent_sorted)
```

```
harmony_mean             21.771772
perceptr_mean            14.624625
zero_crossing_rate_var    9.069069
rms_var                   9.049049
mfcc20_var                6.766767
mfcc19_var                6.556557
mfcc18_var                6.366366
perceptr_var              6.326326
spectral_centroid_var     5.935936
mfcc17_var                5.825826
mfcc16_var                5.585586
harmony_var               5.335335
mfcc15_var                5.245245
mfcc14_var                4.754755
mfcc13_var                4.704705
rolloff_var               4.454454
mfcc12_var                4.454454
spectral_bandwidth_var    4.434434
mfcc1_var                 4.424424
mfcc2_var                 4.414414
mfcc3_var                 4.294294
mfcc10_var                4.084084
mfcc7_var                 4.074074
mfcc5_var                 4.024024
mfcc6_var                 3.993994
mfcc4_var                 3.893894
mfcc11_var                3.843844
mfcc9_var                 3.813814
mfcc8_var                 3.733734
mfcc1_mean                3.193193
mfcc20_mean               2.802803
chroma_stft_var           2.792793
mfcc19_mean               2.212212
mfcc18_mean               2.082082
tempo                     1.751752
```

```
mfcc17_mean                1.681682
mfcc16_mean                1.671672
mfcc2_mean                 1.191191
mfcc14_mean                0.910911
mfcc15_mean                0.840841
rms_mean                   0.780781
zero_crossing_rate_mean    0.770771
mfcc3_mean                 0.640641
mfcc13_mean                0.490490
mfcc11_mean                0.470470
spectral_centroid_mean     0.400400
chroma_stft_mean           0.390390
mfcc5_mean                 0.260260
mfcc8_mean                 0.240240
rolloff_mean               0.200200
mfcc10_mean                0.200200
mfcc12_mean                0.190190
mfcc4_mean                 0.180180
spectral_bandwidth_mean    0.180180
mfcc9_mean                 0.130130
mfcc6_mean                 0.120120
mfcc7_mean                 0.020020
label                      0.000000
dtype: float64
```

[14]:
```
df = df.drop(['harmony_mean','perceptr_mean'], axis=1)
```

[15]:
```
len(df.columns)
```

[15]: 56

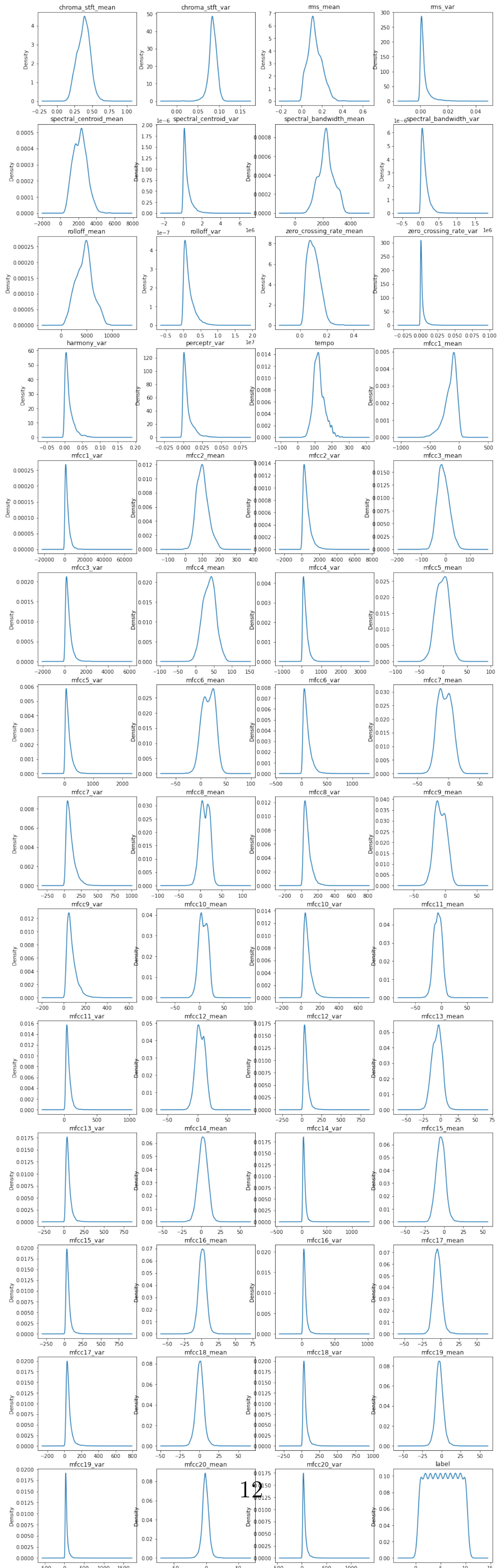
[16]:
```
df.describe()
```

[16]:
```
       chroma_stft_mean  chroma_stft_var     rms_mean       rms_var  \
count       9990.000000      9990.000000  9990.000000  9.990000e+03
mean           0.379534         0.084876     0.130859  2.676388e-03
std            0.090466         0.009637     0.068545  3.585628e-03
min            0.107108         0.015345     0.000953  4.379535e-08
25%            0.315698         0.079833     0.083782  6.145900e-04
50%            0.384741         0.085108     0.121253  1.491318e-03
75%            0.442443         0.091092     0.176328  3.130862e-03
max            0.749481         0.120964     0.442567  3.261522e-02

       spectral_centroid_mean  spectral_centroid_var  spectral_bandwidth_mean  \
count             9990.000000           9.990000e+03              9990.000000
mean              2199.219431           4.166727e+05              2241.385959
std                751.860611           4.349644e+05               543.854449
min                472.741636           8.118813e+02               499.162910
```

```
25%           1630.680158        1.231961e+05           1887.455790
50%           2208.628236        2.650692e+05           2230.575595
75%           2712.581884        5.624152e+05           2588.340505
max           5432.534406        4.794119e+06           3708.147554

       spectral_bandwidth_var  rolloff_mean   rolloff_var  …  mfcc16_var  \
count            9.990000e+03   9990.000000  9.990000e+03  …  9990.000000
mean             1.182711e+05   4566.076592  1.628790e+06  …    49.988755
std              1.013505e+05   1642.065335  1.489398e+06  …    34.442816
min              1.183520e+03    658.336276  1.145102e+03  …     1.325786
25%              4.876553e+04   3378.311110  5.595514e+05  …    29.584894
50%              8.996072e+04   4631.377892  1.160080e+06  …    41.702393
75%              1.585674e+05   5591.634521  2.262437e+06  …    59.274619
max              1.235143e+06   9487.446477  1.298320e+07  …   683.932556

       mfcc17_mean   mfcc17_var  mfcc18_mean   mfcc18_var  mfcc19_mean  \
count  9990.000000  9990.000000  9990.000000  9990.000000  9990.000000
mean     -4.198706    51.962753     0.739943    52.488851    -2.497306
std       5.677379    36.400669     5.181313    38.177120     5.111799
min     -27.809795     1.624544   -20.733809     3.437439   -27.448456
25%      -7.951722    29.863448    -2.516638    29.636197    -5.734123
50%      -4.443021    42.393583     0.733772    41.831377    -2.702366
75%      -0.726945    61.676964     3.888734    62.033906     0.514246
max      34.048843   529.363342    36.970322   629.729797    31.365425

        mfcc19_var  mfcc20_mean   mfcc20_var        label
count  9990.000000  9990.000000  9990.000000  9990.000000
mean     54.973829    -0.917584    57.322614     5.500801
std      41.585677     5.253243    46.444212     2.872355
min       3.065302   -35.640659     0.282131     1.000000
25%      30.496412    -4.004475    30.011365     3.000000
50%      43.435253    -1.030939    44.332155     6.000000
75%      65.328602     2.216603    68.210421     8.000000
max    1143.230591    34.212101   910.473206    10.000000

[8 rows x 56 columns]
```

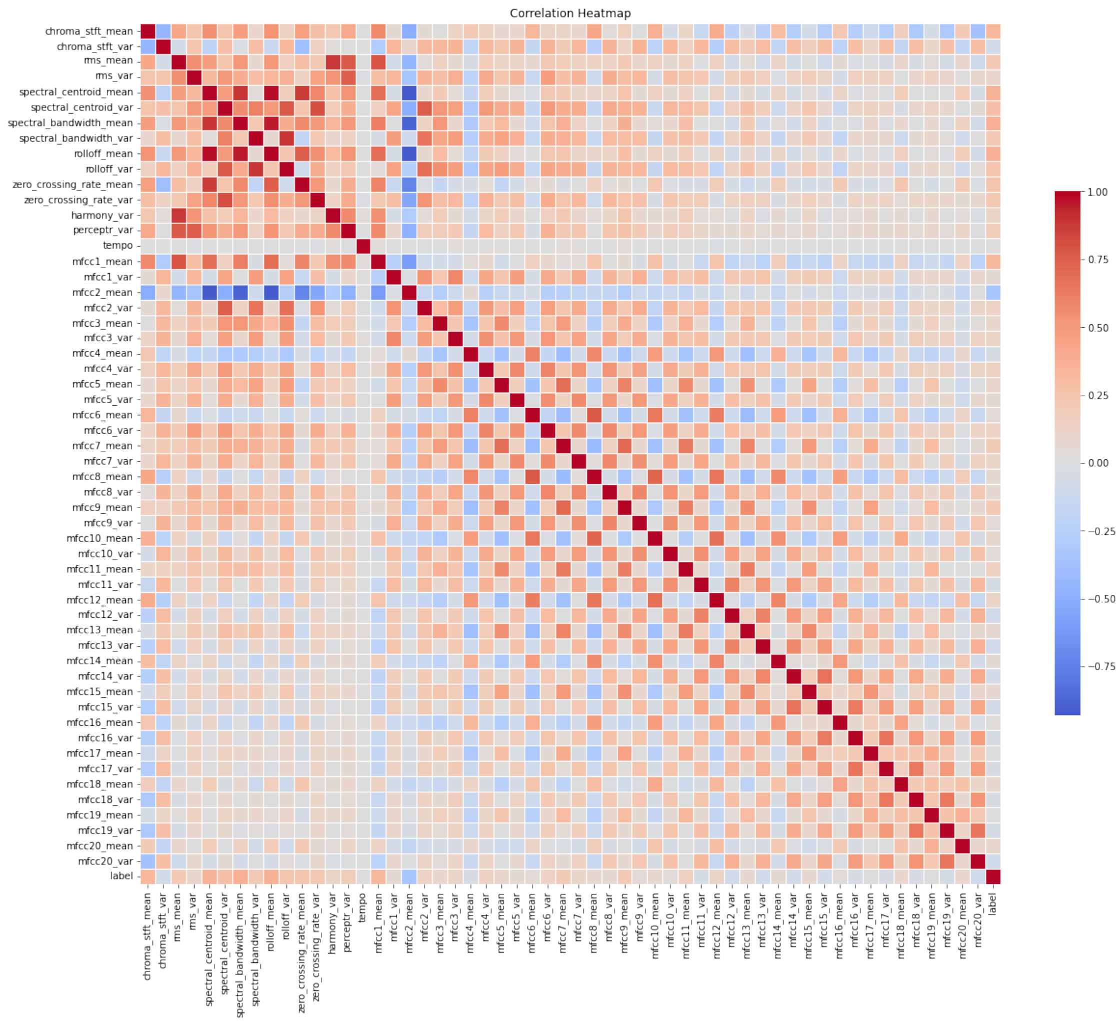
[17]:

```
cols = df.columns
num_cols = len(cols)
num_rows = int(math.ceil(num_cols / 4))

fig, axs = plt.subplots(num_rows, 4, figsize=(16, num_rows*4))
for i, col in enumerate(cols):
    row_idx = i // 4
    col_idx = i % 4
    axs[row_idx, col_idx].set_title(col)
    df[col].plot(kind='density', ax=axs[row_idx, col_idx])
```


chroma_stft_mean
chroma_stft_var
rms_mean
rms_var
spectral_centroid_mean
spectral_centroid_var
spectral_bandwidth_mean
spectral_bandwidth_var
rolloff_mean
rolloff_var
zero_crossing_rate_mean
zero_crossing_rate_var
harmony_var
perceptr_var
tempo
mfcc1_mean
mfcc1_var
mfcc2_mean
mfcc2_var
mfcc3_mean
mfcc3_var
mfcc4_mean
mfcc4_var
mfcc5_mean
mfcc5_var
mfcc6_mean
mfcc6_var
mfcc7_mean
mfcc7_var
mfcc8_mean
mfcc8_var
mfcc9_mean
mfcc9_var
mfcc10_mean
mfcc10_var
mfcc11_mean
mfcc11_var
mfcc12_mean
mfcc12_var
mfcc13_mean
mfcc13_var
mfcc14_mean
mfcc14_var
mfcc15_mean
mfcc15_var
mfcc16_mean
mfcc16_var
mfcc17_mean
mfcc17_var
mfcc18_mean
mfcc18_var
mfcc19_mean
mfcc19_var
mfcc20_mean
mfcc20_var
label
Density

[18]:

```
corr_matrix = df.corr()
fig, ax = plt.subplots(figsize=(20,20))
sns.heatmap(corr_matrix, cmap="coolwarm", center=0, square=True, linewidths=.5,
    cbar_kws={"shrink": .5})
ax.set_title("Correlation Heatmap")
plt.show()
```




[19]:

```
df.corr()
```

[19]:

```
                  chroma_stft_mean  chroma_stft_var  rms_mean  \
chroma_stft_mean          1.000000        -0.443757  0.424706
chroma_stft_var          -0.443757         1.000000 -0.078410
rms_mean                  0.424706        -0.078410  1.000000
```

| | | | |
|---|---|---|---|
| rms_var | 0.243647 | 0.251900 | 0.553770 |
| spectral_centroid_mean | 0.534670 | -0.208136 | 0.470781 |
| spectral_centroid_var | 0.251985 | 0.276964 | 0.241796 |
| spectral_bandwidth_mean | 0.464617 | -0.031197 | 0.495142 |
| spectral_bandwidth_var | 0.100909 | 0.291007 | 0.085760 |
| rolloff_mean | 0.526881 | -0.141792 | 0.500178 |
| rolloff_var | 0.138394 | 0.330930 | 0.157281 |
| zero_crossing_rate_mean | 0.457080 | -0.391281 | 0.293745 |
| zero_crossing_rate_var | 0.278017 | 0.138214 | 0.217606 |
| harmony_var | 0.216488 | 0.024139 | 0.884846 |
| perceptr_var | 0.417177 | 0.000478 | 0.766446 |
| tempo | 0.019084 | -0.004419 | 0.015668 |
| mfcc1_mean | 0.573974 | -0.296517 | 0.795000 |
| mfcc1_var | 0.054769 | 0.347460 | -0.080320 |
| mfcc2_mean | -0.513978 | 0.091757 | -0.453606 |
| mfcc2_var | 0.055816 | 0.326217 | 0.033635 |
| mfcc3_mean | 0.013981 | 0.323529 | 0.136936 |
| mfcc3_var | 0.121196 | 0.354501 | 0.069158 |
| mfcc4_mean | 0.213522 | -0.194552 | -0.034473 |
| mfcc4_var | 0.135822 | 0.333186 | 0.178296 |
| mfcc5_mean | 0.069027 | 0.248247 | 0.122941 |
| mfcc5_var | 0.090913 | 0.351826 | 0.119419 |
| mfcc6_mean | 0.342903 | -0.155817 | 0.072916 |
| mfcc6_var | 0.121105 | 0.363150 | 0.212010 |
| mfcc7_mean | 0.112848 | 0.192922 | 0.181207 |
| mfcc7_var | 0.101439 | 0.343599 | 0.140892 |
| mfcc8_mean | 0.423220 | -0.239743 | 0.134868 |
| mfcc8_var | 0.032791 | 0.352686 | 0.126329 |
| mfcc9_mean | 0.155357 | 0.126653 | 0.213195 |
| mfcc9_var | -0.016019 | 0.363827 | 0.087181 |
| mfcc10_mean | 0.376734 | -0.216207 | 0.144650 |
| mfcc10_var | -0.067534 | 0.337400 | 0.070977 |
| mfcc11_mean | 0.108459 | 0.149261 | 0.133593 |
| mfcc11_var | -0.147130 | 0.335556 | -0.012006 |
| mfcc12_mean | 0.400529 | -0.263772 | 0.134779 |
| mfcc12_var | -0.226017 | 0.302924 | -0.064247 |
| mfcc13_mean | -0.036171 | 0.209318 | 0.043769 |
| mfcc13_var | -0.242423 | 0.309993 | -0.088406 |
| mfcc14_mean | 0.297993 | -0.197412 | 0.113960 |
| mfcc14_var | -0.265843 | 0.283260 | -0.080352 |
| mfcc15_mean | -0.078784 | 0.162830 | 0.012184 |
| mfcc15_var | -0.270092 | 0.285569 | -0.088842 |
| mfcc16_mean | 0.228293 | -0.221620 | 0.055563 |
| mfcc16_var | -0.264815 | 0.296099 | -0.032953 |
| mfcc17_mean | -0.103968 | 0.111544 | -0.013952 |
| mfcc17_var | -0.273841 | 0.303262 | -0.041470 |
| mfcc18_mean | 0.168254 | -0.171744 | 0.087615 |

| | | | |
|---|---|---|---|
| mfcc18_var | -0.308342 | 0.295359 | -0.046180 |
| mfcc19_mean | -0.062985 | 0.107008 | 0.024552 |
| mfcc19_var | -0.313233 | 0.277295 | -0.062662 |
| mfcc20_mean | 0.193666 | -0.173398 | 0.082251 |
| mfcc20_var | -0.363354 | 0.287195 | -0.103519 |
| label | 0.330370 | -0.079683 | 0.205471 |

| | rms_var | spectral_centroid_mean | \ |
|---|---|---|---|
| chroma_stft_mean | 0.243647 | 0.534670 | |
| chroma_stft_var | 0.251900 | -0.208136 | |
| rms_mean | 0.553770 | 0.470781 | |
| rms_var | 1.000000 | 0.327809 | |
| spectral_centroid_mean | 0.327809 | 1.000000 | |
| spectral_centroid_var | 0.509235 | 0.476959 | |
| spectral_bandwidth_mean | 0.383329 | 0.890382 | |
| spectral_bandwidth_var | 0.285950 | 0.021120 | |
| rolloff_mean | 0.350716 | 0.974360 | |
| rolloff_var | 0.377474 | 0.172380 | |
| zero_crossing_rate_mean | 0.143768 | 0.865487 | |
| zero_crossing_rate_var | 0.453957 | 0.579997 | |
| harmony_var | 0.519717 | 0.274194 | |
| perceptr_var | 0.744850 | 0.531487 | |
| tempo | -0.020418 | 0.002111 | |
| mfcc1_mean | 0.296198 | 0.686196 | |
| mfcc1_var | 0.336492 | -0.061331 | |
| mfcc2_mean | -0.351508 | -0.931435 | |
| mfcc2_var | 0.355515 | 0.085022 | |
| mfcc3_mean | 0.277005 | 0.195977 | |
| mfcc3_var | 0.376544 | 0.042125 | |
| mfcc4_mean | -0.215820 | -0.165793 | |
| mfcc4_var | 0.412464 | 0.187926 | |
| mfcc5_mean | 0.248711 | 0.078506 | |
| mfcc5_var | 0.359285 | 0.027291 | |
| mfcc6_mean | -0.134974 | -0.027122 | |
| mfcc6_var | 0.479166 | 0.209412 | |
| mfcc7_mean | 0.290245 | 0.196388 | |
| mfcc7_var | 0.356301 | 0.106182 | |
| mfcc8_mean | -0.110396 | 0.088172 | |
| mfcc8_var | 0.381793 | 0.118458 | |
| mfcc9_mean | 0.301416 | 0.260035 | |
| mfcc9_var | 0.332587 | 0.057883 | |
| mfcc10_mean | -0.058046 | 0.146238 | |
| mfcc10_var | 0.297938 | 0.054746 | |
| mfcc11_mean | 0.251421 | 0.144027 | |
| mfcc11_var | 0.227644 | -0.009416 | |
| mfcc12_mean | -0.066374 | 0.118541 | |
| mfcc12_var | 0.164785 | -0.064614 | |

| | | |
|---|---|---|
| mfcc13_mean | 0.214514 | 0.058537 |
| mfcc13_var | 0.161973 | -0.055983 |
| mfcc14_mean | -0.026530 | 0.118186 |
| mfcc14_var | 0.130442 | -0.058378 |
| mfcc15_mean | 0.189566 | 0.035098 |
| mfcc15_var | 0.116201 | -0.064673 |
| mfcc16_mean | -0.064570 | 0.116866 |
| mfcc16_var | 0.159390 | -0.025006 |
| mfcc17_mean | 0.144238 | 0.026879 |
| mfcc17_var | 0.144163 | -0.018358 |
| mfcc18_mean | -0.004785 | 0.177934 |
| mfcc18_var | 0.102019 | -0.006376 |
| mfcc19_mean | 0.136798 | 0.018651 |
| mfcc19_var | 0.095054 | -0.011018 |
| mfcc20_mean | 0.015085 | 0.191512 |
| mfcc20_var | 0.073571 | -0.051205 |
| label | 0.101467 | 0.360175 |

| | spectral_centroid_var | spectral_bandwidth_mean \ |
|---|---|---|
| chroma_stft_mean | 0.251985 | 0.464617 |
| chroma_stft_var | 0.276964 | -0.031197 |
| rms_mean | 0.241796 | 0.495142 |
| rms_var | 0.509235 | 0.383329 |
| spectral_centroid_mean | 0.476959 | 0.890382 |
| spectral_centroid_var | 1.000000 | 0.556491 |
| spectral_bandwidth_mean | 0.556491 | 1.000000 |
| spectral_bandwidth_var | 0.614254 | 0.223836 |
| rolloff_mean | 0.492965 | 0.951000 |
| rolloff_var | 0.780308 | 0.406680 |
| zero_crossing_rate_mean | 0.242913 | 0.577015 |
| zero_crossing_rate_var | 0.818348 | 0.507718 |
| harmony_var | 0.192711 | 0.348828 |
| perceptr_var | 0.388760 | 0.507508 |
| tempo | -0.009407 | 0.011910 |
| mfcc1_mean | 0.159586 | 0.615946 |
| mfcc1_var | 0.429485 | 0.000893 |
| mfcc2_mean | -0.497158 | -0.887156 |
| mfcc2_var | 0.748612 | 0.137840 |
| mfcc3_mean | 0.513633 | 0.515133 |
| mfcc3_var | 0.461658 | 0.090634 |
| mfcc4_mean | -0.297699 | -0.298786 |
| mfcc4_var | 0.488902 | 0.230293 |
| mfcc5_mean | 0.406538 | 0.314642 |
| mfcc5_var | 0.390078 | 0.069709 |
| mfcc6_mean | -0.158945 | -0.094698 |
| mfcc6_var | 0.493938 | 0.260125 |
| mfcc7_mean | 0.391931 | 0.375831 |

| | | |
|---|---|---|
| mfcc7_var | 0.403324 | 0.131427 |
| mfcc8_mean | -0.158683 | -0.002155 |
| mfcc8_var | 0.405858 | 0.143645 |
| mfcc9_mean | 0.374794 | 0.385406 |
| mfcc9_var | 0.357289 | 0.086434 |
| mfcc10_mean | -0.135941 | 0.031015 |
| mfcc10_var | 0.326240 | 0.086466 |
| mfcc11_mean | 0.309186 | 0.277708 |
| mfcc11_var | 0.244125 | -0.005638 |
| mfcc12_mean | -0.139430 | 0.004485 |
| mfcc12_var | 0.160181 | -0.073405 |
| mfcc13_mean | 0.282591 | 0.199991 |
| mfcc13_var | 0.166777 | -0.068204 |
| mfcc14_mean | -0.113106 | 0.025717 |
| mfcc14_var | 0.111909 | -0.069594 |
| mfcc15_mean | 0.234941 | 0.125946 |
| mfcc15_var | 0.096053 | -0.076864 |
| mfcc16_mean | -0.134131 | 0.012459 |
| mfcc16_var | 0.125906 | -0.023060 |
| mfcc17_mean | 0.134869 | 0.089523 |
| mfcc17_var | 0.115752 | -0.021857 |
| mfcc18_mean | -0.045517 | 0.084331 |
| mfcc18_var | 0.093684 | -0.005564 |
| mfcc19_mean | 0.123857 | 0.074453 |
| mfcc19_var | 0.074366 | -0.020422 |
| mfcc20_mean | -0.012324 | 0.133895 |
| mfcc20_var | 0.052981 | -0.063954 |
| label | 0.281352 | 0.376621 |

| | spectral_bandwidth_var | rolloff_mean | rolloff_var | \ |
|---|---|---|---|---|
| chroma_stft_mean | 0.100909 | 0.526881 | 0.138394 | |
| chroma_stft_var | 0.291007 | -0.141792 | 0.330930 | |
| rms_mean | 0.085760 | 0.500178 | 0.157281 | |
| rms_var | 0.285950 | 0.350716 | 0.377474 | |
| spectral_centroid_mean | 0.021120 | 0.974360 | 0.172380 | |
| spectral_centroid_var | 0.614254 | 0.492965 | 0.780308 | |
| spectral_bandwidth_mean | 0.223836 | 0.951000 | 0.406680 | |
| spectral_bandwidth_var | 1.000000 | 0.070097 | 0.891339 | |
| rolloff_mean | 0.070097 | 1.000000 | 0.237905 | |
| rolloff_var | 0.891339 | 0.237905 | 1.000000 | |
| zero_crossing_rate_mean | -0.187738 | 0.755442 | -0.096437 | |
| zero_crossing_rate_var | 0.219781 | 0.542989 | 0.388534 | |
| harmony_var | 0.094987 | 0.316167 | 0.153447 | |
| perceptr_var | 0.133080 | 0.532023 | 0.218191 | |
| tempo | 0.003874 | 0.007359 | 0.000047 | |
| mfcc1_mean | -0.036112 | 0.688779 | 0.033406 | |
| mfcc1_var | 0.467327 | -0.049394 | 0.455660 | |

| | | | |
|---|---|---|---|
| mfcc2_mean | -0.112064 | -0.923652 | -0.266610 |
| mfcc2_var | 0.661289 | 0.076059 | 0.686924 |
| mfcc3_mean | 0.420372 | 0.308917 | 0.552629 |
| mfcc3_var | 0.472401 | 0.044365 | 0.468462 |
| mfcc4_mean | -0.272014 | -0.182640 | -0.315924 |
| mfcc4_var | 0.401072 | 0.195052 | 0.438535 |
| mfcc5_mean | 0.450533 | 0.137204 | 0.484348 |
| mfcc5_var | 0.375526 | 0.040068 | 0.391193 |
| mfcc6_mean | -0.142835 | -0.006269 | -0.152174 |
| mfcc6_var | 0.384290 | 0.222668 | 0.441628 |
| mfcc7_mean | 0.372212 | 0.234987 | 0.399362 |
| mfcc7_var | 0.337100 | 0.114703 | 0.381804 |
| mfcc8_mean | -0.153744 | 0.095158 | -0.160117 |
| mfcc8_var | 0.319851 | 0.122597 | 0.364339 |
| mfcc9_mean | 0.304600 | 0.287332 | 0.335188 |
| mfcc9_var | 0.304344 | 0.063764 | 0.340767 |
| mfcc10_mean | -0.161636 | 0.140193 | -0.160437 |
| mfcc10_var | 0.273149 | 0.061075 | 0.317133 |
| mfcc11_mean | 0.286844 | 0.167872 | 0.299959 |
| mfcc11_var | 0.205451 | -0.015225 | 0.225511 |
| mfcc12_mean | -0.167961 | 0.110553 | -0.182219 |
| mfcc12_var | 0.133148 | -0.077201 | 0.144860 |
| mfcc13_mean | 0.272876 | 0.087349 | 0.290017 |
| mfcc13_var | 0.121970 | -0.068515 | 0.140379 |
| mfcc14_mean | -0.158052 | 0.109308 | -0.160432 |
| mfcc14_var | 0.096696 | -0.070733 | 0.101009 |
| mfcc15_mean | 0.207737 | 0.042899 | 0.225537 |
| mfcc15_var | 0.089042 | -0.078149 | 0.088188 |
| mfcc16_mean | -0.180031 | 0.098611 | -0.188433 |
| mfcc16_var | 0.101519 | -0.032600 | 0.112760 |
| mfcc17_mean | 0.093049 | 0.035871 | 0.099636 |
| mfcc17_var | 0.078486 | -0.026397 | 0.095440 |
| mfcc18_mean | -0.122140 | 0.158802 | -0.110838 |
| mfcc18_var | 0.051146 | -0.012166 | 0.069654 |
| mfcc19_mean | 0.094140 | 0.033071 | 0.094853 |
| mfcc19_var | 0.022953 | -0.020990 | 0.044384 |
| mfcc20_mean | -0.091730 | 0.182626 | -0.090789 |
| mfcc20_var | 0.015888 | -0.065557 | 0.031120 |
| label | 0.210972 | 0.369515 | 0.260298 |

| | … | mfcc16_var | mfcc17_mean | mfcc17_var | \ |
|---|---|---|---|---|---|
| chroma_stft_mean | … | -0.264815 | -0.103968 | -0.273841 | |
| chroma_stft_var | … | 0.296099 | 0.111544 | 0.303262 | |
| rms_mean | … | -0.032953 | -0.013952 | -0.041470 | |
| rms_var | … | 0.159390 | 0.144238 | 0.144163 | |
| spectral_centroid_mean | … | -0.025006 | 0.026879 | -0.018358 | |
| spectral_centroid_var | … | 0.125906 | 0.134869 | 0.115752 | |

| | | | | |
|---|---|---|---|---|
| spectral_bandwidth_mean | … | -0.023060 | 0.089523 | -0.021857 |
| spectral_bandwidth_var | … | 0.101519 | 0.093049 | 0.078486 |
| rolloff_mean | … | -0.032600 | 0.035871 | -0.026397 |
| rolloff_var | … | 0.112760 | 0.099636 | 0.095440 |
| zero_crossing_rate_mean | … | -0.037794 | -0.053256 | -0.025369 |
| zero_crossing_rate_var | … | 0.119463 | 0.108177 | 0.123928 |
| harmony_var | … | 0.020852 | 0.069224 | 0.011693 |
| perceptr_var | … | 0.066420 | 0.091803 | 0.057017 |
| tempo | … | -0.007704 | -0.007811 | -0.008103 |
| mfcc1_mean | … | -0.152901 | -0.165295 | -0.156403 |
| mfcc1_var | … | 0.146094 | 0.109361 | 0.136629 |
| mfcc2_mean | … | -0.032249 | -0.039512 | -0.036995 |
| mfcc2_var | … | 0.193442 | 0.080209 | 0.172803 |
| mfcc3_mean | … | 0.053349 | 0.183993 | 0.052039 |
| mfcc3_var | … | 0.216863 | 0.080565 | 0.198165 |
| mfcc4_mean | … | -0.143569 | -0.313803 | -0.134879 |
| mfcc4_var | … | 0.202449 | 0.037338 | 0.201788 |
| mfcc5_mean | … | 0.021671 | 0.287429 | 0.019136 |
| mfcc5_var | … | 0.185922 | -0.026951 | 0.173399 |
| mfcc6_mean | … | -0.114969 | -0.316158 | -0.095829 |
| mfcc6_var | … | 0.242685 | 0.054646 | 0.238444 |
| mfcc7_mean | … | 0.042288 | 0.400902 | 0.037623 |
| mfcc7_var | … | 0.232891 | -0.042328 | 0.240745 |
| mfcc8_mean | … | -0.140601 | -0.317046 | -0.123043 |
| mfcc8_var | … | 0.262940 | 0.035981 | 0.244299 |
| mfcc9_mean | … | 0.023086 | 0.451653 | 0.020333 |
| mfcc9_var | … | 0.252087 | 0.016128 | 0.259623 |
| mfcc10_mean | … | -0.071358 | -0.229256 | -0.051245 |
| mfcc10_var | … | 0.300269 | 0.039474 | 0.258303 |
| mfcc11_mean | … | -0.008101 | 0.377836 | -0.016705 |
| mfcc11_var | … | 0.317634 | 0.071374 | 0.287789 |
| mfcc12_mean | … | -0.115042 | -0.263923 | -0.116400 |
| mfcc12_var | … | 0.366304 | 0.096531 | 0.323763 |
| mfcc13_mean | … | 0.092776 | 0.379630 | 0.056303 |
| mfcc13_var | … | 0.426287 | 0.105066 | 0.380623 |
| mfcc14_mean | … | 0.006934 | -0.109517 | -0.016905 |
| mfcc14_var | … | 0.507372 | 0.109390 | 0.440862 |
| mfcc15_mean | … | 0.179354 | 0.522975 | 0.155241 |
| mfcc15_var | … | 0.635182 | 0.149313 | 0.505812 |
| mfcc16_mean | … | 0.073514 | 0.184646 | 0.079352 |
| mfcc16_var | … | 1.000000 | 0.210599 | 0.667416 |
| mfcc17_mean | … | 0.210599 | 1.000000 | 0.225542 |
| mfcc17_var | … | 0.667416 | 0.225542 | 1.000000 |
| mfcc18_mean | … | 0.092274 | 0.267657 | 0.162034 |
| mfcc18_var | … | 0.513841 | 0.234316 | 0.652012 |
| mfcc19_mean | … | 0.108409 | 0.396497 | 0.165415 |
| mfcc19_var | … | 0.479247 | 0.222192 | 0.508315 |

| | | | | |
|---|---|---|---|---|
| mfcc20_mean | … | -0.039532 | 0.016829 | -0.006057 |
| mfcc20_var | … | 0.473173 | 0.201749 | 0.479436 |
| label | … | -0.044354 | 0.004921 | -0.040047 |

| | mfcc18_mean | mfcc18_var | mfcc19_mean | mfcc19_var | \ |
|---|---|---|---|---|---|
| chroma_stft_mean | 0.168254 | -0.308342 | -0.062985 | -0.313233 | |
| chroma_stft_var | -0.171744 | 0.295359 | 0.107008 | 0.277295 | |
| rms_mean | 0.087615 | -0.046180 | 0.024552 | -0.062662 | |
| rms_var | -0.004785 | 0.102019 | 0.136798 | 0.095054 | |
| spectral_centroid_mean | 0.177934 | -0.006376 | 0.018651 | -0.011018 | |
| spectral_centroid_var | -0.045517 | 0.093684 | 0.123857 | 0.074366 | |
| spectral_bandwidth_mean | 0.084331 | -0.005564 | 0.074453 | -0.020422 | |
| spectral_bandwidth_var | -0.122140 | 0.051146 | 0.094140 | 0.022953 | |
| rolloff_mean | 0.158802 | -0.012166 | 0.033071 | -0.020990 | |
| rolloff_var | -0.110838 | 0.069654 | 0.094853 | 0.044384 | |
| zero_crossing_rate_mean | 0.233676 | -0.013337 | -0.058103 | -0.005162 | |
| zero_crossing_rate_var | 0.035576 | 0.104703 | 0.103994 | 0.098745 | |
| harmony_var | 0.051217 | 0.006638 | 0.073491 | -0.011757 | |
| perceptr_var | 0.096911 | 0.042719 | 0.094491 | 0.029040 | |
| tempo | -0.010346 | -0.001752 | 0.002024 | -0.008792 | |
| mfcc1_mean | 0.145346 | -0.147427 | -0.106033 | -0.166703 | |
| mfcc1_var | -0.086184 | 0.099405 | 0.092047 | 0.090691 | |
| mfcc2_mean | -0.165099 | -0.045010 | -0.017996 | -0.041203 | |
| mfcc2_var | -0.101693 | 0.138819 | 0.103120 | 0.112521 | |
| mfcc3_mean | -0.150529 | 0.054158 | 0.160484 | 0.041994 | |
| mfcc3_var | -0.051560 | 0.167991 | 0.122434 | 0.142421 | |
| mfcc4_mean | 0.117687 | -0.150930 | -0.203544 | -0.131683 | |
| mfcc4_var | -0.046673 | 0.166600 | 0.105607 | 0.136637 | |
| mfcc5_mean | -0.238413 | 0.008021 | 0.246904 | -0.006957 | |
| mfcc5_var | -0.070219 | 0.128536 | 0.054416 | 0.094152 | |
| mfcc6_mean | 0.251641 | -0.102708 | -0.235553 | -0.098891 | |
| mfcc6_var | -0.034646 | 0.200909 | 0.101947 | 0.167559 | |
| mfcc7_mean | -0.204423 | 0.054778 | 0.312595 | 0.048932 | |
| mfcc7_var | 0.006264 | 0.195529 | 0.015234 | 0.164734 | |
| mfcc8_mean | 0.305576 | -0.126858 | -0.249339 | -0.118777 | |
| mfcc8_var | -0.025354 | 0.209749 | 0.079623 | 0.220721 | |
| mfcc9_mean | -0.088542 | 0.035755 | 0.313192 | 0.059584 | |
| mfcc9_var | -0.024732 | 0.227163 | 0.054775 | 0.213431 | |
| mfcc10_mean | 0.406386 | -0.049005 | -0.162905 | -0.026069 | |
| mfcc10_var | -0.015491 | 0.236696 | 0.054100 | 0.251245 | |
| mfcc11_mean | -0.112753 | -0.012119 | 0.340867 | -0.006021 | |
| mfcc11_var | -0.034051 | 0.243194 | 0.050443 | 0.282128 | |
| mfcc12_mean | 0.330363 | -0.138856 | -0.169898 | -0.127784 | |
| mfcc12_var | -0.004206 | 0.289262 | 0.039303 | 0.322797 | |
| mfcc13_mean | -0.214577 | 0.034250 | 0.307131 | 0.041056 | |
| mfcc13_var | -0.006597 | 0.332213 | 0.073547 | 0.327539 | |
| mfcc14_mean | 0.269913 | -0.047278 | -0.149703 | -0.063786 | |

| | | | | |
|---|---|---|---|---|
| mfcc14_var | -0.011016 | 0.415638 | 0.055541 | 0.395105 |
| mfcc15_mean | -0.061796 | 0.112609 | 0.241806 | 0.088215 |
| mfcc15_var | 0.025380 | 0.460656 | 0.066365 | 0.416319 |
| mfcc16_mean | 0.442218 | 0.045698 | -0.050022 | 0.013609 |
| mfcc16_var | 0.092274 | 0.513841 | 0.108409 | 0.479247 |
| mfcc17_mean | 0.267657 | 0.234316 | 0.396497 | 0.222192 |
| mfcc17_var | 0.162034 | 0.652012 | 0.165415 | 0.508315 |
| mfcc18_mean | 1.000000 | 0.197126 | 0.289336 | 0.167120 |
| mfcc18_var | 0.197126 | 1.000000 | 0.244033 | 0.648110 |
| mfcc19_mean | 0.289336 | 0.244033 | 1.000000 | 0.265922 |
| mfcc19_var | 0.167120 | 0.648110 | 0.265922 | 1.000000 |
| mfcc20_mean | 0.267654 | 0.071501 | 0.377956 | 0.126406 |
| mfcc20_var | 0.109515 | 0.526891 | 0.226714 | 0.658459 |
| label | 0.082595 | -0.039150 | 0.013143 | -0.059141 |

| | mfcc20_mean | mfcc20_var | label |
|---|---|---|---|
| chroma_stft_mean | 0.193666 | -0.363354 | 0.330370 |
| chroma_stft_var | -0.173398 | 0.287195 | -0.079683 |
| rms_mean | 0.082251 | -0.103519 | 0.205471 |
| rms_var | 0.015085 | 0.073571 | 0.101467 |
| spectral_centroid_mean | 0.191512 | -0.051205 | 0.360175 |
| spectral_centroid_var | -0.012324 | 0.052981 | 0.281352 |
| spectral_bandwidth_mean | 0.133895 | -0.063954 | 0.376621 |
| spectral_bandwidth_var | -0.091730 | 0.015888 | 0.210972 |
| rolloff_mean | 0.182626 | -0.065557 | 0.369515 |
| rolloff_var | -0.090789 | 0.031120 | 0.260298 |
| zero_crossing_rate_mean | 0.211582 | -0.031773 | 0.243590 |
| zero_crossing_rate_var | 0.055677 | 0.073218 | 0.215464 |
| harmony_var | 0.032613 | -0.032397 | 0.115452 |
| perceptr_var | 0.115535 | -0.011110 | 0.170666 |
| tempo | -0.013771 | -0.006341 | 0.012369 |
| mfcc1_mean | 0.131098 | -0.215759 | 0.326771 |
| mfcc1_var | -0.071271 | 0.082459 | 0.059184 |
| mfcc2_mean | -0.173734 | -0.001894 | -0.348035 |
| mfcc2_var | -0.082988 | 0.111730 | 0.127176 |
| mfcc3_mean | -0.057273 | 0.020956 | 0.089573 |
| mfcc3_var | -0.041873 | 0.132295 | 0.110916 |
| mfcc4_mean | 0.080553 | -0.145016 | -0.048384 |
| mfcc4_var | -0.039665 | 0.119339 | 0.144013 |
| mfcc5_mean | -0.101725 | -0.008134 | 0.100811 |
| mfcc5_var | -0.100357 | 0.075830 | 0.051668 |
| mfcc6_mean | 0.124554 | -0.136149 | 0.039947 |
| mfcc6_var | -0.017656 | 0.131261 | 0.131331 |
| mfcc7_mean | -0.031434 | 0.049845 | 0.198996 |
| mfcc7_var | -0.062794 | 0.144604 | 0.107591 |
| mfcc8_mean | 0.173534 | -0.144878 | 0.071334 |
| mfcc8_var | -0.038570 | 0.208753 | 0.070706 |

```
mfcc9_mean                0.016066   0.068832  0.207059
mfcc9_var                -0.081883   0.224179  0.041014
mfcc10_mean               0.248017  -0.041117  0.074355
mfcc10_var               -0.056297   0.266320  0.021395
mfcc11_mean              -0.000124   0.017518  0.152093
mfcc11_var               -0.036124   0.324765 -0.049489
mfcc12_mean               0.293125  -0.151380  0.111270
mfcc12_var               -0.069951   0.367465 -0.090699
mfcc13_mean              -0.031201   0.075620  0.066110
mfcc13_var               -0.040220   0.367158 -0.089063
mfcc14_mean               0.222342  -0.077124  0.051521
mfcc14_var               -0.023828   0.412246 -0.068083
mfcc15_mean              -0.091470   0.097467  0.065622
mfcc15_var               -0.026204   0.424161 -0.067919
mfcc16_mean               0.203262  -0.030900  0.026503
mfcc16_var               -0.039532   0.473173 -0.044354
mfcc17_mean               0.016829   0.201749  0.004921
mfcc17_var               -0.006057   0.479436 -0.040047
mfcc18_mean               0.267654   0.109515  0.082595
mfcc18_var                0.071501   0.526891 -0.039150
mfcc19_mean               0.377956   0.226714  0.013143
mfcc19_var                0.126406   0.658459 -0.059141
mfcc20_mean               1.000000   0.098934  0.083224
mfcc20_var                0.098934   1.000000 -0.099627
label                     0.083224  -0.099627  1.000000

[56 rows x 56 columns]
```

[20]:
```python
corr_matrix = df.corr()
pairwise_correlations = corr_matrix.unstack()
pairwise_correlations = pairwise_correlations.drop_duplicates().dropna()
sorted_correlations = pairwise_correlations.sort_values(ascending=False)
```

[21]:
```python
print(sorted_correlations.head(15))
```

```
chroma_stft_mean         chroma_stft_mean          1.000000
spectral_centroid_mean   rolloff_mean              0.974360
spectral_bandwidth_mean  rolloff_mean              0.951000
spectral_bandwidth_var   rolloff_var               0.891339
spectral_centroid_mean   spectral_bandwidth_mean   0.890382
rms_mean                 harmony_var               0.884846
spectral_centroid_mean   zero_crossing_rate_mean   0.865487
spectral_centroid_var    zero_crossing_rate_var    0.818348
rms_mean                 mfcc1_mean                0.795000
spectral_centroid_var    rolloff_var               0.780308
mfcc6_mean               mfcc8_mean                0.769248
rms_mean                 perceptr_var              0.766446
rolloff_mean             zero_crossing_rate_mean   0.755442
```

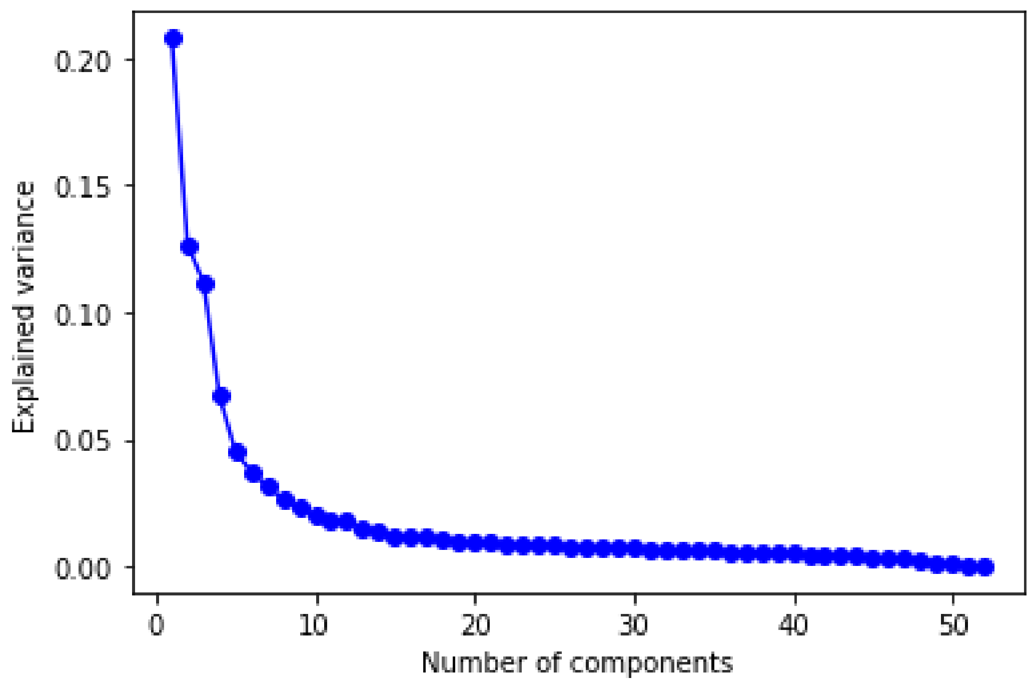
```
spectral_centroid_var    mfcc2_var               0.748612
rms_var                  perceptr_var            0.744850
dtype: float64
```

[22]:
```
df = df.
 ↪drop(['spectral_centroid_mean','spectral_bandwidth_mean','spectral_bandwidth_var'],axis=1)
```

[23]:
```
X = df.drop('label', axis=1)
y = df['label']
```

[24]:
```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

[25]:
```
pca = PCA()
pca.fit(X_scaled)
plt.plot(range(1, len(pca.explained_variance_ratio_) + 1), pca.
 ↪explained_variance_ratio_, 'bo-')
plt.xlabel('Number of components')
plt.ylabel('Explained variance')
plt.show()
```




[26]:
```
pca = PCA(n_components=12)
X_pca = pca.fit_transform(X_scaled)
```

[27]:
```python
df_pca = pd.DataFrame(data=X_pca)
df_pca['label'] = y
```

[28]:
```python
df_pca.shape
```

[28]:
```
(9990, 13)
```

[29]:
```python
df_pca.columns
```

[29]:
```
Index([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 'label'], dtype='object')
```

[30]:
```python
df_pca['label'].value_counts()
```

[30]:
```
1     1000
6     1000
7     1000
8     1000
9     1000
4      999
2      998
5      998
10     998
3      997
Name: label, dtype: int64
```

[31]:
```python
df_pca.head()
```

[31]:
```
          0         1         2         3         4         5         6  \
0 -1.731736  0.216238  0.749912 -0.921875 -0.016142 -1.004118 -0.024569
1 -2.484077  0.200474  1.767866  0.428246 -0.391701 -0.222257 -0.758078
2 -1.728930 -0.013076  0.548662  0.042710 -1.171010 -1.189234 -0.625472
3 -2.341590  0.039708  0.919030 -0.604440 -0.200098 -0.869820 -0.380123
4 -2.863800  0.140608  0.684801 -0.154782 -0.566477 -1.177621 -0.322459

          7         8         9        10        11  label
0 -0.281547  0.138578  0.187780  0.136367  0.174685      1
1 -0.113312  0.548457 -0.088799  0.451051 -0.428794      1
2  0.062313 -0.281826  0.306589 -0.020125  0.077267      1
3 -0.557823  0.309720  0.464806  0.545901  0.216463      1
4 -0.421278 -0.163608  0.640347 -0.113425  0.521318      1
```

[32]:
```python
from sklearn.model_selection import train_test_split


X_train, X_test, y_train, y_test = train_test_split(df_pca.drop('label',␣
↪axis=1), df_pca['label'], test_size=0.25, random_state=42)
```

```
print("X_train Shape:", X_train.shape)
print("X_test Shape:", X_test.shape)
print("y_train Shape:", y_train.shape)
print("y_test Shape:", y_test.shape)
```

```
X_train Shape: (7492, 12)
X_test Shape: (2498, 12)
y_train Shape: (7492,)
y_test Shape: (2498,)
```

[33]:

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.svm import SVC
from sklearn.metrics import classification_report


knn = KNeighborsClassifier()
nb = GaussianNB()
rf = RandomForestClassifier()
lr = LogisticRegression()
lda = LinearDiscriminantAnalysis()
svm = SVC()


knn.fit(X_train, y_train)
nb.fit(X_train, y_train)
rf.fit(X_train, y_train)
lr.fit(X_train, y_train)
lda.fit(X_train, y_train)
svm.fit(X_train, y_train)


knn_pred = knn.predict(X_test)
nb_pred = nb.predict(X_test)
rf_pred = rf.predict(X_test)
lr_pred = lr.predict(X_test)
lda_pred = lda.predict(X_test)
svm_pred = svm.predict(X_test)


print("KNN Classification Report:")
print(classification_report(y_test, knn_pred))

print("Naive Bayes Classification Report:")
```

```python
print(classification_report(y_test, nb_pred))

print("Random Forest Classification Report:")
print(classification_report(y_test, rf_pred))

print("Logistic Regression Classification Report:")
print(classification_report(y_test, lr_pred))

print("LDA Classification Report:")
print(classification_report(y_test, lda_pred))

print("SVM Classification Report:")
print(classification_report(y_test, svm_pred))
```

```
C:\Users\anjal\anaconda3\lib\site-
packages\sklearn\linear_model\_logistic.py:814: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

```
KNN Classification Report:
              precision    recall  f1-score   support

           1       0.74      0.77      0.75       257
           2       0.86      0.92      0.89       256
           3       0.62      0.74      0.68       232
           4       0.63      0.72      0.67       255
           5       0.81      0.75      0.78       270
           6       0.79      0.74      0.76       244
           7       0.88      0.82      0.85       261
           8       0.80      0.82      0.81       224
           9       0.75      0.75      0.75       254
          10       0.72      0.53      0.61       245

    accuracy                           0.76      2498
   macro avg       0.76      0.76      0.76      2498
weighted avg       0.76      0.76      0.76      2498

Naive Bayes Classification Report:
              precision    recall  f1-score   support

           1       0.53      0.25      0.34       257
```

```
           2       0.78      0.80      0.79       256
           3       0.33      0.44      0.38       232
           4       0.39      0.41      0.40       255
           5       0.61      0.41      0.49       270
           6       0.50      0.41      0.45       244
           7       0.50      0.87      0.64       261
           8       0.64      0.67      0.65       224
           9       0.55      0.56      0.55       254
          10       0.30      0.25      0.27       245

    accuracy                           0.51      2498
   macro avg       0.51      0.51      0.50      2498
weighted avg       0.51      0.51      0.50      2498

Random Forest Classification Report:
              precision    recall  f1-score   support

           1       0.75      0.67      0.70       257
           2       0.84      0.93      0.88       256
           3       0.64      0.65      0.65       232
           4       0.65      0.68      0.66       255
           5       0.83      0.75      0.79       270
           6       0.70      0.73      0.71       244
           7       0.79      0.88      0.83       261
           8       0.75      0.83      0.79       224
           9       0.72      0.72      0.72       254
          10       0.67      0.52      0.59       245

    accuracy                           0.74      2498
   macro avg       0.73      0.74      0.73      2498
weighted avg       0.74      0.74      0.73      2498

Logistic Regression Classification Report:
              precision    recall  f1-score   support

           1       0.47      0.51      0.49       257
           2       0.85      0.88      0.86       256
           3       0.37      0.33      0.35       232
           4       0.37      0.27      0.31       255
           5       0.63      0.47      0.54       270
           6       0.58      0.61      0.59       244
           7       0.62      0.82      0.71       261
           8       0.61      0.81      0.69       224
           9       0.54      0.58      0.56       254
          10       0.34      0.26      0.29       245

    accuracy                           0.55      2498
   macro avg       0.54      0.55      0.54      2498
```

```
weighted avg       0.54      0.55      0.54      2498

LDA Classification Report:
              precision    recall  f1-score   support

           1       0.45      0.41      0.43       257
           2       0.79      0.85      0.82       256
           3       0.35      0.34      0.35       232
           4       0.30      0.21      0.25       255
           5       0.68      0.40      0.50       270
           6       0.53      0.52      0.52       244
           7       0.55      0.84      0.67       261
           8       0.53      0.79      0.64       224
           9       0.51      0.57      0.54       254
          10       0.28      0.19      0.23       245

    accuracy                           0.51      2498
   macro avg       0.50      0.51      0.49      2498
weighted avg       0.50      0.51      0.50      2498

SVM Classification Report:
              precision    recall  f1-score   support

           1       0.68      0.63      0.65       257
           2       0.85      0.91      0.88       256
           3       0.59      0.62      0.61       232
           4       0.53      0.53      0.53       255
           5       0.79      0.68      0.73       270
           6       0.70      0.76      0.73       244
           7       0.77      0.86      0.81       261
           8       0.76      0.81      0.79       224
           9       0.70      0.68      0.69       254
          10       0.52      0.46      0.49       245

    accuracy                           0.69      2498
   macro avg       0.69      0.69      0.69      2498
weighted avg       0.69      0.69      0.69      2498
```

[34]:
```
from sklearn.metrics import classification_report, accuracy_score,␣
 ↪precision_score, recall_score, f1_score

print("KNN Metrics:")
print("Accuracy:", accuracy_score(y_test, knn_pred))
print("Precision:", precision_score(y_test, knn_pred,average='macro'))
print("Recall:", recall_score(y_test, knn_pred,average='macro'))
print("F1 Score:", f1_score(y_test, knn_pred,average='macro'))
```

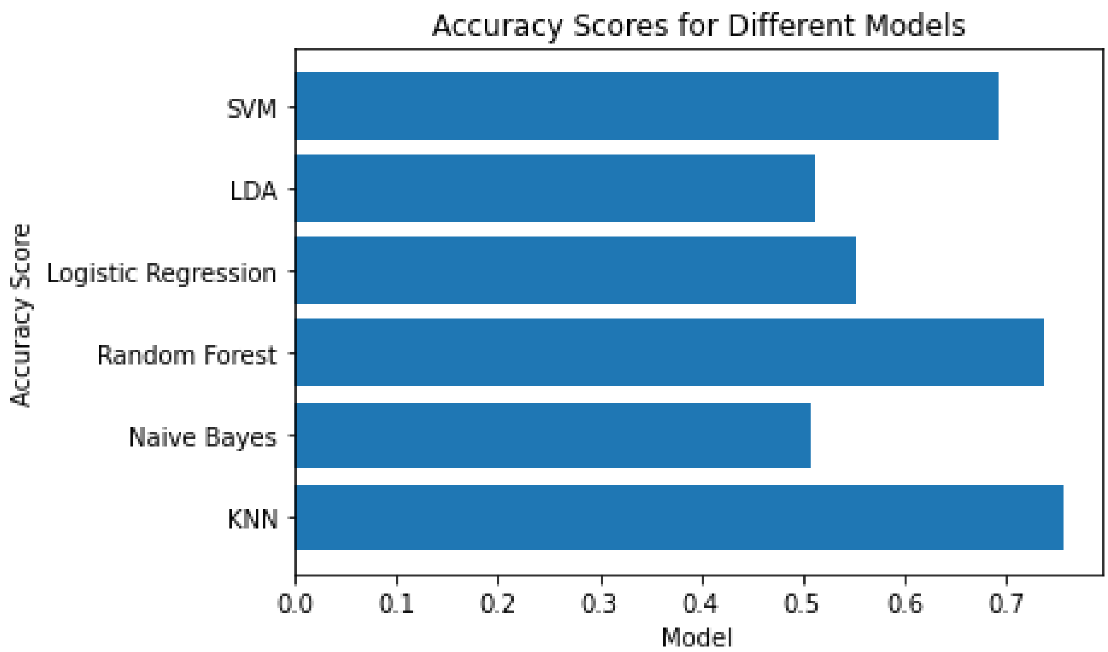
```
print('\n')
print("Naive Bayes Metrics:")
print("Accuracy:", accuracy_score(y_test, nb_pred))
print("Precision:", precision_score(y_test, nb_pred,average='macro'))
print("Recall:", recall_score(y_test, nb_pred,average='macro'))
print("F1 Score:", f1_score(y_test, nb_pred,average='macro'))
print('\n')
print("Random Forest Metrics:")
print("Accuracy:", accuracy_score(y_test, rf_pred))
print("Precision:", precision_score(y_test, rf_pred,average='macro'))
print("Recall:", recall_score(y_test, rf_pred,average='macro'))
print("F1 Score:", f1_score(y_test, rf_pred,average='macro'))
print('\n')
print("Logistic Regression Metrics:")
print("Accuracy:", accuracy_score(y_test, lr_pred))
print("Precision:", precision_score(y_test, lr_pred,average='macro'))
print("Recall:", recall_score(y_test, lr_pred,average='macro'))
print("F1 Score:", f1_score(y_test, lr_pred,average='macro'))
print('\n')
print("LDA Metrics:")
print("Accuracy:", accuracy_score(y_test, lda_pred))
print("Precision:", precision_score(y_test, lda_pred,average='macro'))
print("Recall:", recall_score(y_test, lda_pred,average='macro'))
print("F1 Score:", f1_score(y_test, lda_pred,average='macro'))
print('\n')
print("SVM Metrics:")
print("Accuracy:", accuracy_score(y_test, svm_pred))
print("Precision:", precision_score(y_test, svm_pred,average='macro'))
print("Recall:", recall_score(y_test, svm_pred,average='macro'))
print("F1 Score:", f1_score(y_test, svm_pred,average='macro'))
```

```
KNN Metrics:
Accuracy: 0.7566052842273819
Precision: 0.7591552790612084
Recall: 0.7561072420723634
F1 Score: 0.7550222155862784


Naive Bayes Metrics:
Accuracy: 0.5068054443554844
Precision: 0.5118378956995205
Recall: 0.5065222727321381
F1 Score: 0.4959297297818693


Random Forest Metrics:
Accuracy: 0.7373899119295436
Precision: 0.7342208041605509
```

```
Recall: 0.736517978984267
F1 Score: 0.7332081573205491


Logistic Regression Metrics:
Accuracy: 0.5536429143314652
Precision: 0.5375146597372649
Recall: 0.5535465618212347
F1 Score: 0.5400645858079806


LDA Metrics:
Accuracy: 0.5124099279423538
Precision: 0.49770742752733665
Recall: 0.5129413949669543
F1 Score: 0.494099763757428


SVM Metrics:
Accuracy: 0.6937550040032026
Precision: 0.6897122911535681
Recall: 0.6935901866195163
F1 Score: 0.6903376024342267
```

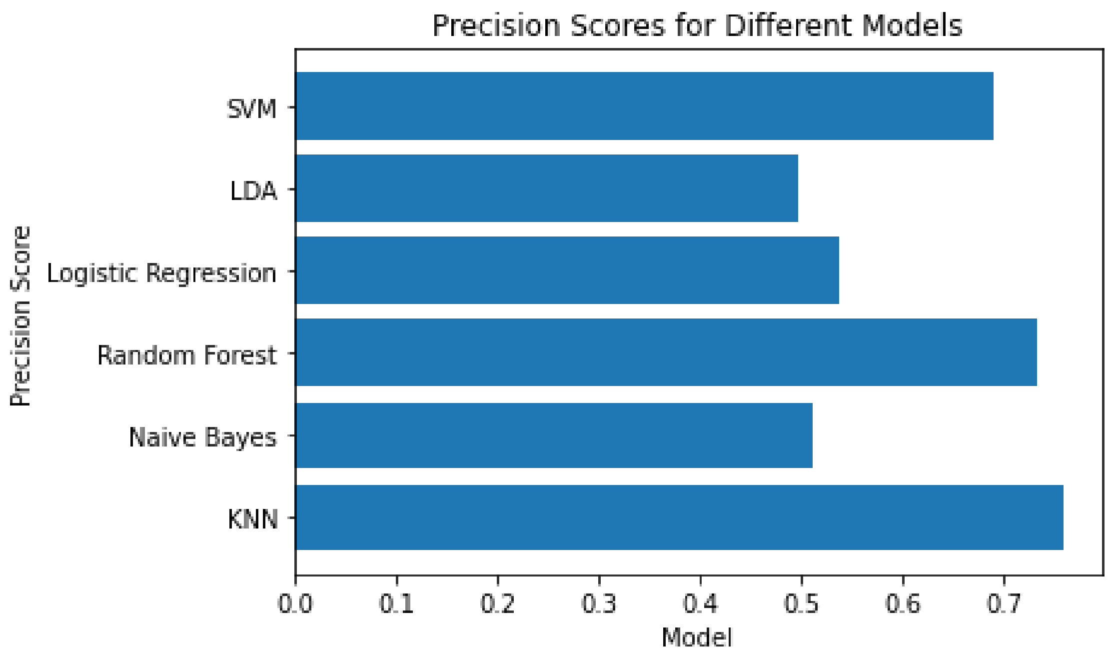
[35]:
```
models = ['KNN', 'Naive Bayes', 'Random Forest', 'Logistic Regression', 'LDA',␣
↪'SVM']
accuracy_scores = [accuracy_score(y_test, knn_pred), accuracy_score(y_test,␣
↪nb_pred), accuracy_score(y_test, rf_pred), accuracy_score(y_test, lr_pred),␣
↪accuracy_score(y_test, lda_pred), accuracy_score(y_test, svm_pred)]

for model, accuracy in zip(models, accuracy_scores):
    print(model, "Accuracy:", accuracy)

# Plot the accuracy scores for each model
plt.barh(models, accuracy_scores)
plt.title('Accuracy Scores for Different Models')
plt.xlabel('Model')
plt.ylabel('Accuracy Score')
plt.show()
```

```
KNN Accuracy: 0.7566052842273819
Naive Bayes Accuracy: 0.5068054443554844
Random Forest Accuracy: 0.7373899119295436
Logistic Regression Accuracy: 0.5536429143314652
LDA Accuracy: 0.5124099279423538
SVM Accuracy: 0.6937550040032026
```




[36]:
```
models = ['KNN', 'Naive Bayes', 'Random Forest', 'Logistic Regression', 'LDA',␣
 ↪'SVM']
precision_scores = [precision_score(y_test, knn_pred, average = 'macro'),␣
 ↪precision_score(y_test, nb_pred,average = 'macro'), precision_score(y_test,␣
 ↪rf_pred,average = 'macro'), precision_score(y_test, lr_pred,average =␣
 ↪'macro'), precision_score(y_test, lda_pred,average = 'macro'),␣
 ↪precision_score(y_test, svm_pred,average = 'macro')]

for model, precision in zip(models, precision_scores):
    print(model, "Precision:", precision)


plt.barh(models, precision_scores)
plt.title('Precision Scores for Different Models')
plt.xlabel('Model')
plt.ylabel('Precision Score')
plt.show()
```

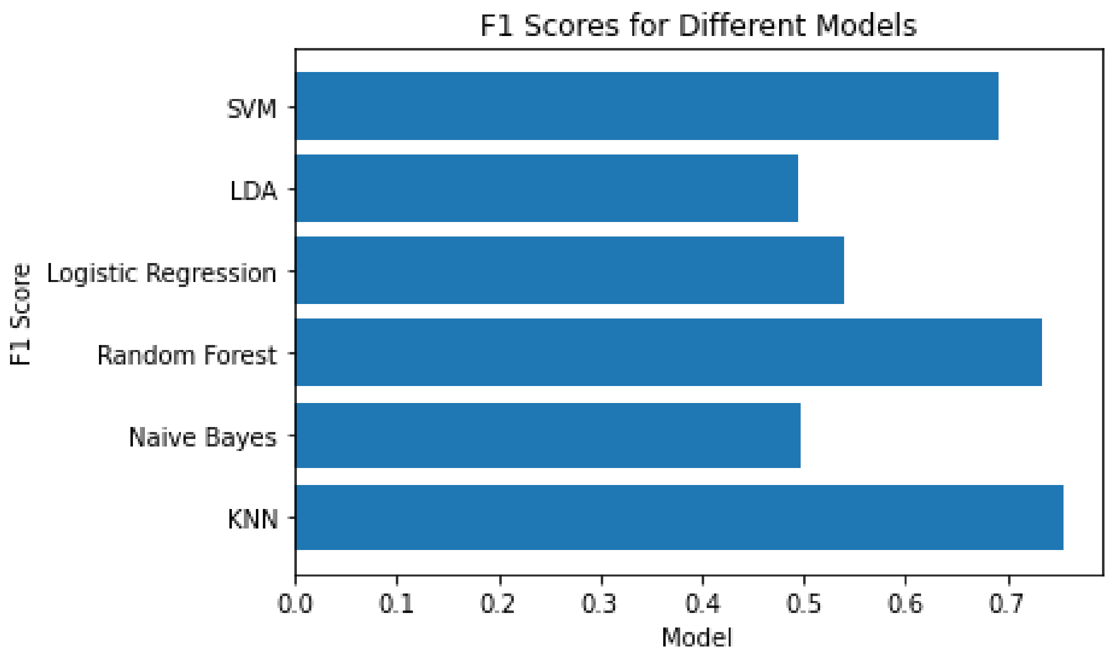
```
KNN Precision: 0.7591552790612084
Naive Bayes Precision: 0.5118378956995205
Random Forest Precision: 0.7342208041605509
Logistic Regression Precision: 0.5375146597372649
LDA Precision: 0.49770742752733665
SVM Precision: 0.6897122911535681
```




[37]:

```
models = ['KNN', 'Naive Bayes', 'Random Forest', 'Logistic Regression', 'LDA',␣
↪'SVM']
f1_scores = [f1_score(y_test, knn_pred, average = 'macro'), f1_score(y_test,␣
↪nb_pred,average = 'macro'), f1_score(y_test, rf_pred,average = 'macro'),␣
↪f1_score(y_test, lr_pred,average = 'macro'), f1_score(y_test,␣
↪lda_pred,average = 'macro'), f1_score(y_test, svm_pred,average = 'macro')]

for model, precision in zip(models, precision_scores):
    print(model, "F1 Score", precision)


plt.barh(models, f1_scores)
plt.title('F1 Scores for Different Models')
plt.xlabel('Model')
plt.ylabel('F1 Score')
plt.show()
```

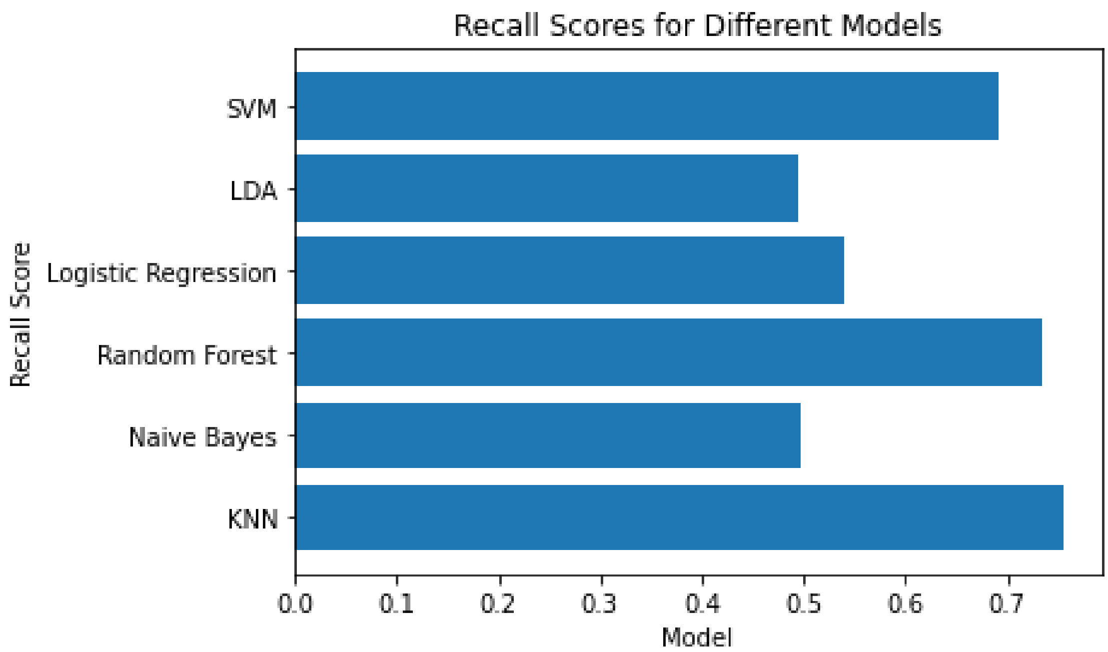
```
KNN F1 Score 0.7591552790612084
Naive Bayes F1 Score 0.5118378956995205
Random Forest F1 Score 0.7342208041605509
Logistic Regression F1 Score 0.5375146597372649
LDA F1 Score 0.49770742752733665
SVM F1 Score 0.6897122911535681
```




[38]:

```
models = ['KNN', 'Naive Bayes', 'Random Forest', 'Logistic Regression', 'LDA',␣
↪'SVM']
recall = [recall_score(y_test, knn_pred, average = 'macro'),␣
↪recall_score(y_test, nb_pred,average = 'macro'), recall_score(y_test,␣
↪rf_pred,average = 'macro'), recall_score(y_test, lr_pred,average = 'macro'),␣
↪recall_score(y_test, lda_pred,average = 'macro'), recall_score(y_test,␣
↪svm_pred,average = 'macro')]

for model, precision in zip(models, precision_scores):
    print(model, "Recall Score", precision)


plt.barh(models, f1_scores)
plt.title('Recall Scores for Different Models')
plt.xlabel('Model')
plt.ylabel('Recall Score')
plt.show()
```

```
KNN Recall Score 0.7591552790612084
Naive Bayes Recall Score 0.5118378956995205
Random Forest Recall Score 0.7342208041605509
Logistic Regression Recall Score 0.5375146597372649
LDA Recall Score 0.49770742752733665
SVM Recall Score 0.6897122911535681
```




[39]:
```
!pip install scikit-plot
```

```
Requirement already satisfied: scikit-plot in c:\users\anjal\anaconda3\lib\site-
packages (0.3.7)
Requirement already satisfied: matplotlib>=1.4.0 in
c:\users\anjal\anaconda3\lib\site-packages (from scikit-plot) (3.5.1)
Requirement already satisfied: scikit-learn>=0.18 in
c:\users\anjal\anaconda3\lib\site-packages (from scikit-plot) (1.0.2)
Requirement already satisfied: scipy>=0.9 in c:\users\anjal\anaconda3\lib\site-
packages (from scikit-plot) (1.7.3)
Requirement already satisfied: joblib>=0.10 in
c:\users\anjal\anaconda3\lib\site-packages (from scikit-plot) (1.1.0)
Requirement already satisfied: cycler>=0.10 in
c:\users\anjal\anaconda3\lib\site-packages (from matplotlib>=1.4.0->scikit-plot)
(0.11.0)
Requirement already satisfied: python-dateutil>=2.7 in
c:\users\anjal\anaconda3\lib\site-packages (from matplotlib>=1.4.0->scikit-plot)
(2.8.2)
Requirement already satisfied: kiwisolver>=1.0.1 in
c:\users\anjal\anaconda3\lib\site-packages (from matplotlib>=1.4.0->scikit-plot)
(1.3.2)
Requirement already satisfied: fonttools>=4.22.0 in
c:\users\anjal\anaconda3\lib\site-packages (from matplotlib>=1.4.0->scikit-plot)
(4.25.0)
Requirement already satisfied: numpy>=1.17 in c:\users\anjal\anaconda3\lib\site-
packages (from matplotlib>=1.4.0->scikit-plot) (1.21.0)
```

```
Requirement already satisfied: pyparsing>=2.2.1 in
c:\users\anjal\anaconda3\lib\site-packages (from matplotlib>=1.4.0->scikit-plot)
(3.0.4)
Requirement already satisfied: packaging>=20.0 in
c:\users\anjal\anaconda3\lib\site-packages (from matplotlib>=1.4.0->scikit-plot)
(21.3)
Requirement already satisfied: pillow>=6.2.0 in
c:\users\anjal\anaconda3\lib\site-packages (from matplotlib>=1.4.0->scikit-plot)
(9.0.1)
Requirement already satisfied: six>=1.5 in c:\users\anjal\anaconda3\lib\site-
packages (from python-dateutil>=2.7->matplotlib>=1.4.0->scikit-plot) (1.16.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in
c:\users\anjal\anaconda3\lib\site-packages (from scikit-learn>=0.18->scikit-
plot) (2.2.0)
```

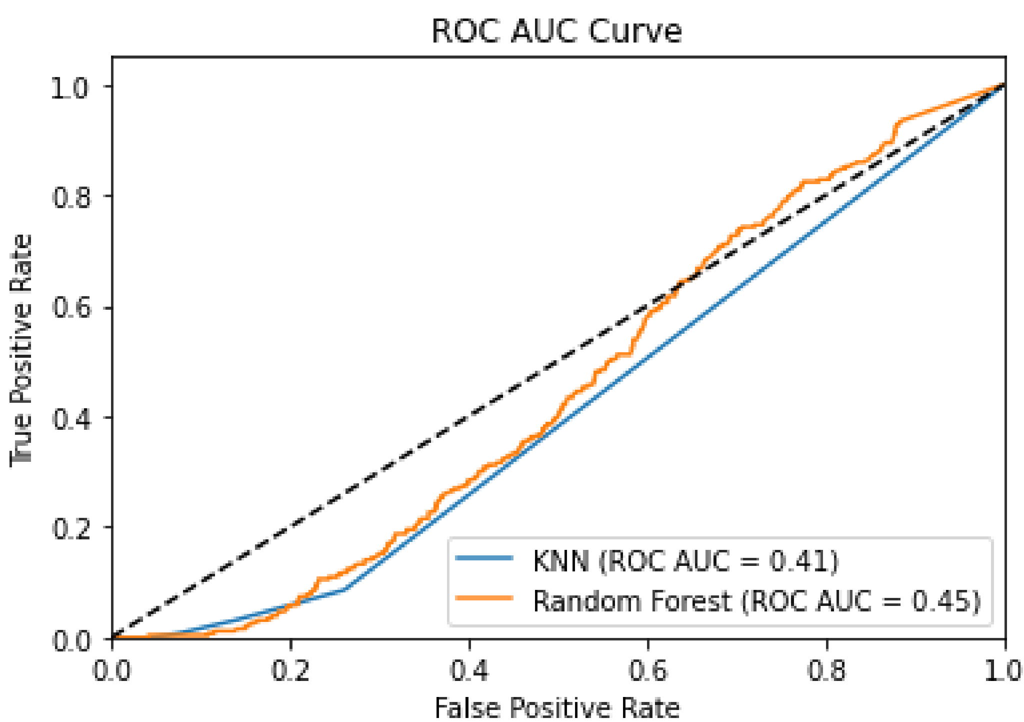
[40]:
```python
from sklearn.metrics import roc_curve, auc
```

[ ]:
```python

```

[41]:
```python
from sklearn.multiclass import OneVsRestClassifier
from sklearn.metrics import roc_curve, auc
from sklearn.preprocessing import label_binarize

knn = KNeighborsClassifier()
rf = RandomForestClassifier()
ovr_knn = OneVsRestClassifier(knn)
ovr_rf = OneVsRestClassifier(rf)
ovr_knn.fit(X_train, y_train)
ovr_rf.fit(X_train, y_train)
y_prob_knn = ovr_knn.predict_proba(X_test)
y_prob_rf = ovr_rf.predict_proba(X_test)

y_test_bin = label_binarize(y_test, classes=[0, 1, 2])


fpr_knn = dict()
tpr_knn = dict()
roc_auc_knn = dict()
for i in range(3):
    fpr_knn[i], tpr_knn[i], _ = roc_curve(y_test_bin[:, i], y_prob_knn[:, i])
    roc_auc_knn[i] = auc(fpr_knn[i], tpr_knn[i])

fpr_rf = dict()
tpr_rf = dict()
roc_auc_rf = dict()
for i in range(3):
    fpr_rf[i], tpr_rf[i], _ = roc_curve(y_test_bin[:, i], y_prob_rf[:, i])
```

```
    roc_auc_rf[i] = auc(fpr_rf[i], tpr_rf[i])


import matplotlib.pyplot as plt
plt.figure()
plt.plot(fpr_knn[2], tpr_knn[2], label='KNN (ROC AUC = %0.2f)' % roc_auc_knn[2])
plt.plot(fpr_rf[2], tpr_rf[2], label='Random Forest (ROC AUC = %0.2f)' %␣
 ↪roc_auc_rf[2])
plt.plot([0, 1], [0, 1], 'k--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC AUC Curve')
plt.legend(loc="lower right")
plt.show()
```

```
C:\Users\anjal\anaconda3\lib\site-packages\sklearn\metrics\_ranking.py:999:
UndefinedMetricWarning: No positive samples in y_true, true positive value
should be meaningless
  warnings.warn(
C:\Users\anjal\anaconda3\lib\site-packages\sklearn\metrics\_ranking.py:999:
UndefinedMetricWarning: No positive samples in y_true, true positive value
should be meaningless
  warnings.warn(
```



```
[ ]:
```

```
[ ]:
```

```
[ ]:
```