In [1]:
```
import os
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.svm import LinearSVC
from sklearn import preprocessing
from sklearn.model_selection import cross_val_score
from sklearn import svm
from sklearn.feature_selection import RFECV
from sklearn.feature_selection import RFE
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import GridSearchCV
from sklearn.feature_selection import SelectFromModel
from sklearn.metrics import accuracy_score
from sklearn.model_selection import RepeatedStratifiedKFold
```

# Saving Features as pandas dataframe

In [2]:
```
df = pd.read_csv('extracted_feature_values_midi.csv')
```

In [3]:
```
cluster_mood_filename = df.cluster_mood_filename
temp = []
cluster = []
mood = []
filename = []

for i in cluster_mood_filename:
    temp.append(i)
temp = [i.split('\\') for i in temp]

for path in temp:
    cluster.append(path[-3])
    mood.append(path[-2])
    filename.append(path[-1].split('.')[0])

df['cluster'] = cluster
df['mood'] = mood
df['filename'] = filename

cols = [col for col in df.columns]
xcols = cols[1:-3]
ycol = cols[-3]
print(ycol)
```

```
cluster
```

# Applying label encoder and splitting the dataset into train and test datasets

In [4]:
```
le = preprocessing.LabelEncoder()
X = df[xcols]
X = X.fillna(0)
Y = df[ycol]
Y = pd.DataFrame(Y)
Y = Y.apply(lambda col: le.fit_transform(col.astype(str)), axis=0, result_type
files = df['filename']
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2)
# print(type(df))
# print(type(X_train))
# print(type(Y_train))
# print(type(xcols))
# print(type(ycol))
# print(X.isnull().sum())
# print(Y.isnull().sum())
# print(X.shape)
# X_nonan = X.fillna(0)
# print(X_nonan.isnull().values.any())
# print(X_nonan.shape)
print(X_train.shape)
print(Y_train.shape)
X_train.head()
```

```
(156, 1495)
(156, 1)
```

Out[4]:

| | Basic_Pitch_Histogram_0 | Basic_Pitch_Histogram_1 | Basic_Pitch_Histogram_2 | Basic_Pitch_Histogram |
|---|---|---|---|---|
| **89** | 0.0 | 0 | 0 | |
| **149** | 0.0 | 0 | 0 | |
| **12** | 0.0 | 0 | 0 | |
| **162** | 0.0 | 0 | 0 | |
| **86** | 0.0 | 0 | 0 | |

5 rows × 1495 columns

# Normalizing the dataset for improved svm performance

In [5]:
```
#Normalizing using standard scalor
normalized_X = X.values
std_scalor = preprocessing.StandardScaler()
X_scaled = std_scalor.fit_transform(normalized_X)
normalized_X = pd.DataFrame(X_scaled, columns=xcols)
X_train3, X_test3, Y_train3, Y_test3 = train_test_split(normalized_X, Y, test_
normalized_X.head()
```

Out[5]:

| | Basic_Pitch_Histogram_0 | Basic_Pitch_Histogram_1 | Basic_Pitch_Histogram_2 | Basic_Pitch_Histogram_3 |
|---|---|---|---|---|
| **0** | -0.071611 | 0.0 | 0.0 | 0.0 |

| | Basic_Pitch_Histogram_0 | Basic_Pitch_Histogram_1 | Basic_Pitch_Histogram_2 | Basic_Pitch_Histogram_3 |
|---|---|---|---|---|
| **1** | -0.071611 | 0.0 | 0.0 | 0.0 |
| **2** | -0.071611 | 0.0 | 0.0 | 0.0 |
| **3** | -0.071611 | 0.0 | 0.0 | 0.0 |
| **4** | -0.071611 | 0.0 | 0.0 | 0.0 |

# SVM classification on original dataset without feature selection

In [6]:

```
# %%time
# #clf = LinearSVC(max_iter = 10000000, verbose = 1)
# clf = svm.SVC(C=1, kernel='linear', verbose = True)
# clf.fit(X_train, Y_train.values.ravel())
# cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=20)
# #cv = StratifiedKFold(5)
# #cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=20, random_state=1)
# train_accuracy_svm = cross_val_score(clf, X_train, Y_train.values.ravel(), 
# test_accuracy_svm = clf.score(X_test, Y_test.values.ravel())

# print("Training accuracy " + str(train_accuracy_svm))
# print("Test accuracy " + str(test_accuracy_svm))
```

# SVM classification on normalized dataset without feature selection

In [7]:

```
%%time
clf = svm.SVC(C=0.1, kernel='linear', verbose = True)
clf.fit(X_train3, Y_train3.values.ravel())
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=20)
#cv = StratifiedKFold(5)
#cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=20, random_state=1)
train_accuracy_svm_norm = cross_val_score(clf, X_train3, Y_train3.values.ravel
test_accuracy_svm_norm = clf.score(X_test3, Y_test3.values.ravel())

print("Training accuracy " + str(train_accuracy_svm_norm))
print("Test accuracy " + str(test_accuracy_svm_norm))
```

```
[LibSVM]Training accuracy 0.3192857142857143
Test accuracy 0.3389830508474576
Wall time: 8.09 s
```

# Feature Selection on Midi features

Methods to employ -

1) Tree based feature selection(Random Forests)

2) Recursive feature elimination with cross validation using SVM estimators

3) Anova F-measures

4) Chi-squared

5) Principal component analysis

We will apply each of them individually, evaluate performance, and choose a combination

Principal component analysis should be performed as the dimensions of the feature vectore(1495) is more than the number of samples(196)

**study showed svm classification based on midi features had a test accuracy of 35%** https://www.kaggle.com/arrohit/feature-selection-and-hypertuning-svm **tree based feature selection**

In [8]:
```
%%time
rf_clf = RandomForestClassifier()
rf_clf = rf_clf.fit(X_train3, Y_train3.values.ravel())
model = SelectFromModel(rf_clf, prefit=True)

rf_X_train = X_train3.loc[:, model.get_support()]
rf_X_test = X_test3.loc[:, model.get_support()]
print(type(rf_X_train))
print(rf_X_train.shape)
rf_X_train.head()
```

```
<class 'pandas.core.frame.DataFrame'>
(137, 536)
Wall time: 201 ms
```

Out[8]:

| | Basic_Pitch_Histogram_25 | Basic_Pitch_Histogram_28 | Basic_Pitch_Histogram_31 | Basic_Pitch_Histog |
|---|---|---|---|---|
| **25** | -0.007813 | 0.422440 | 0.023239 | 0 |
| **137** | -0.093576 | -0.356986 | -0.416183 | -0 |
| **183** | -0.093576 | -0.356986 | 0.431855 | 0 |
| **61** | -0.093576 | -0.356986 | -0.440096 | -0 |
| **95** | -0.093576 | -0.290034 | -0.420554 | -0 |

5 rows × 536 columns

In [9]:
```
%%time
clf = svm.SVC(C=0.1, kernel='linear', verbose = True)
clf.fit(rf_X_train, Y_train3.values.ravel())
#cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=20)
#cv = StratifiedKFold(5)
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=20, random_state=1)
train_accuracy_svm_rf = cross_val_score(clf, rf_X_train, Y_train3.values.ravel
test_accuracy_svm_rf = clf.score(rf_X_test, Y_test3.values.ravel())

print("Training accuracy " + str(train_accuracy_svm_rf))
print("Test accuracy " + str(test_accuracy_svm_rf))
```

```
[LibSVM]Training accuracy 0.336043956043956
Test accuracy 0.4067796610169492
Wall time: 955 ms
```

https://machinelearningmastery.com/rfe-feature-selection-in-python/ **RFECV**

In [10]:
```
%%time
clf = svm.SVC(C=0.1, kernel='linear')
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=20, random_state = 1)
#cv = StratifiedKFold(5)
rfecv = RFECV(estimator = clf, step = 1, cv = cv, scoring='accuracy', n_jobs=
rfetrain=rfecv.fit(X_train3, Y_train3.values.ravel())
print('Optimal number of features :', rfecv.n_features_)
```

```
Optimal number of features : 116
Wall time: 7min 25s
```

In [11]:
```
%%time
rfe = RFE(estimator=clf, n_features_to_select=rfecv.n_features_, step=1)
rfe = rfe.fit(X_train3, Y_train3.values.ravel())

rfe_X_train = X_train3.loc[:, rfe.get_support()]
rfe_X_test = X_test3.loc[:, rfe.get_support()]
```

```
Wall time: 12.8 s
```

In [12]:
```
rfe_X_train.head()
```

Out[12]:

| | Basic_Pitch_Histogram_43 | Basic_Pitch_Histogram_58 | Basic_Pitch_Histogram_62 | Basic_Pitch_Histog |
|---|---|---|---|---|
| **25** | -0.371752 | -0.579763 | 0.271071 | -0 |
| **137** | 1.708119 | -0.559762 | 1.419969 | 0 |
| **183** | 0.545021 | 1.098289 | 2.232147 | -0 |
| **61** | -0.475267 | 1.707781 | -0.454457 | 0 |
| **95** | -0.306289 | -0.518372 | -0.424093 | -0 |

5 rows × 116 columns

In [13]:
```
%%time
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=20)
#cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=20, random_state = 1)
#cv = StratifiedKFold(5)
train_accuracy_svm_rfe = cross_val_score(estimator=clf,X=rfe_X_train,y=Y_trai
print("Training accuracy " + str(train_accuracy_svm_rfe))
clf = clf.fit(rfe_X_train, Y_train3.values.ravel())
test_accuracy_svm_rfe = clf.score(rfe_X_test, Y_test3.values.ravel())
print("Test accuracy " + str(test_accuracy_svm_rfe))
```

```
Training accuracy 0.8030494505494506
Test accuracy 0.423728813559322
Wall time: 354 ms
```

https://machinelearningmastery.com/feature-selection-with-real-and-categorical-data/ **ANOVA F measure**

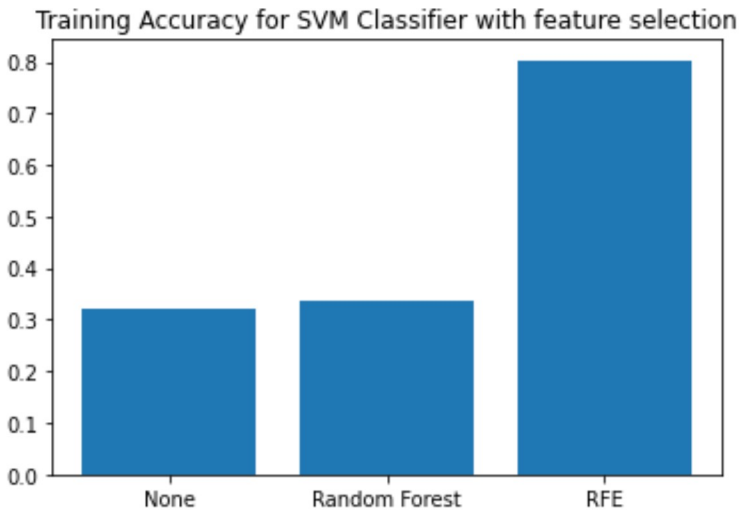
In [ ]:
```
```

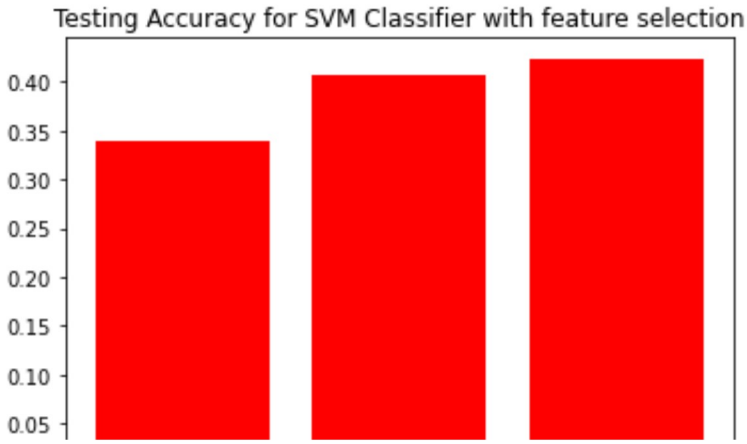
# Graph Maker

In [18]:
```
train_accuracy_list = [train_accuracy_svm_norm, train_accuracy_svm_rf, train_a
test_accuracy_list = [test_accuracy_svm_norm, test_accuracy_svm_rf, test_accu
labels = ['None', 'Random Forest', 'RFE']

fig1,ax1 = plt.subplots()
ax1.set_title("Training Accuracy for SVM Classifier with feature selection")
ax1.bar(labels, train_accuracy_list)
plt.savefig("Train_Accuracy.png")

fig2,ax2 = plt.subplots()
ax2.set_title("Testing Accuracy for SVM Classifier with feature selection")
ax2.bar(labels, test_accuracy_list, color='r')
plt.savefig("Test_Accuracy.png")
```




Testing Accuracy for SVM Classifier with feature selection
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05

In [ ]: