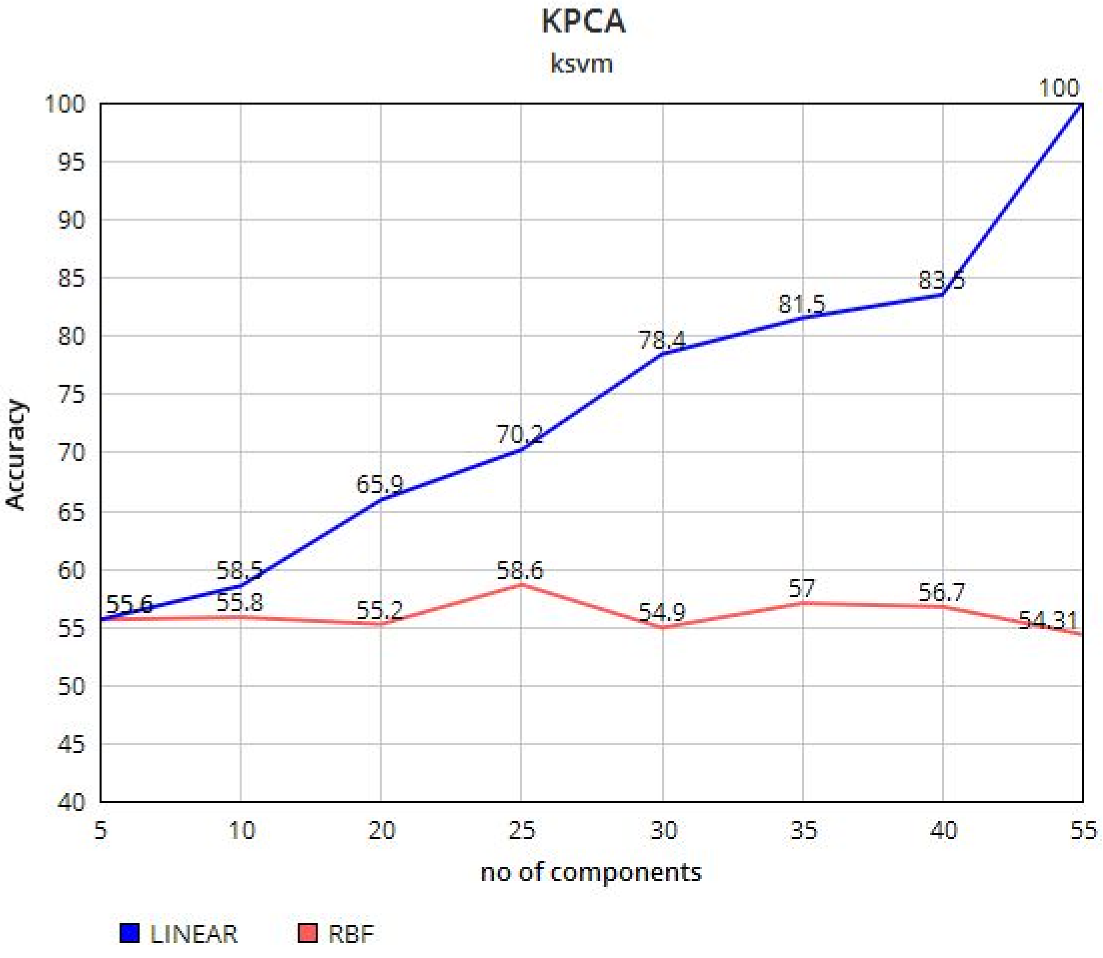
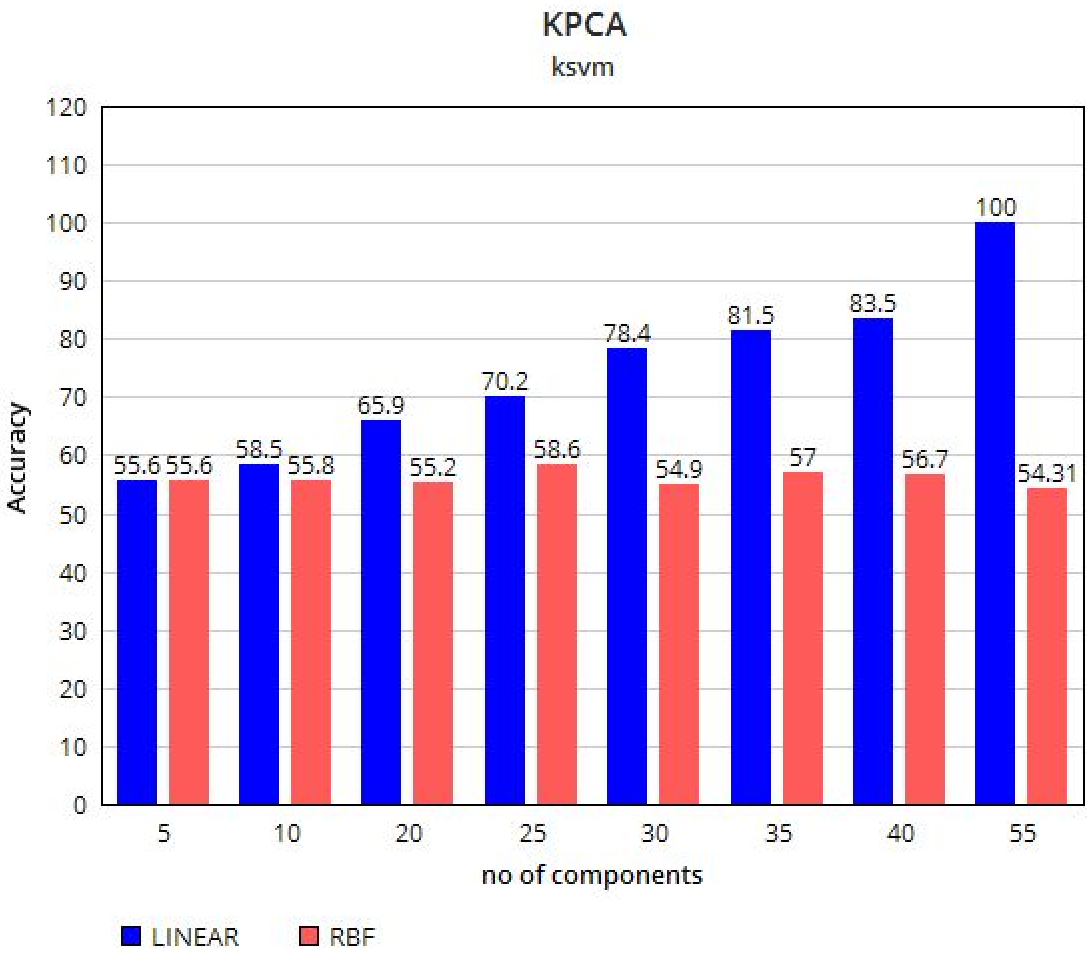
ASSIGNMENT 3

201402163

T YASHWANTH REDDY

2 a Kernel PCA





CODE FOR KERNEL PCA

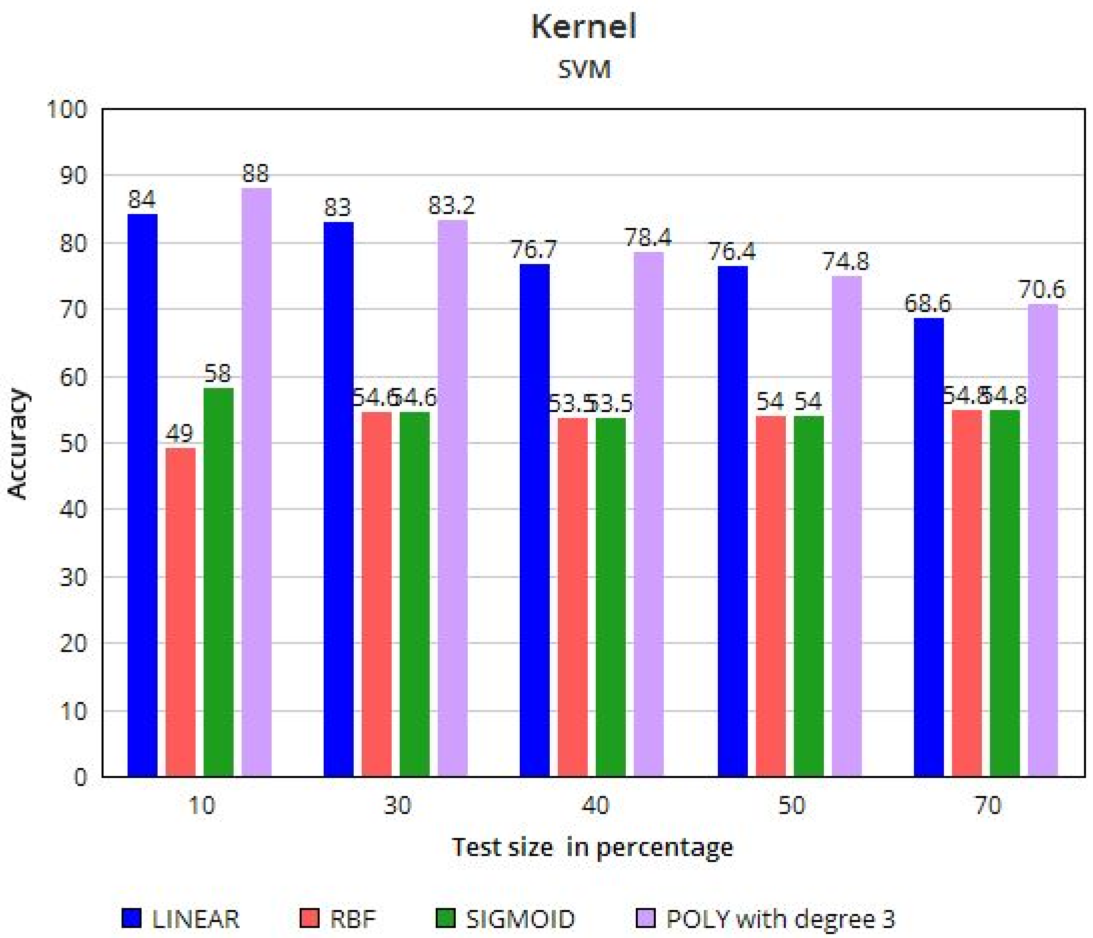
```
from sklearn import svm
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.decomposition import PCA, KernelPCA
import pandas as pd
import numpy as np
from sklearn.cross_validation import train_test_split
_file1=pd.read_csv('arcene_train.csv', delim_whitespace=True)
_file2=pd.read_csv('arcene_train.labels')
_file1['Class'] = (_file2['1']).astype(int)
#print _file1.isnull()
#print
_file1.info()#################################################################################
#cleaning missing values
_file1= _file1.fillna(lambda x: x.median())
#print _file1.info()
#print _file1
```

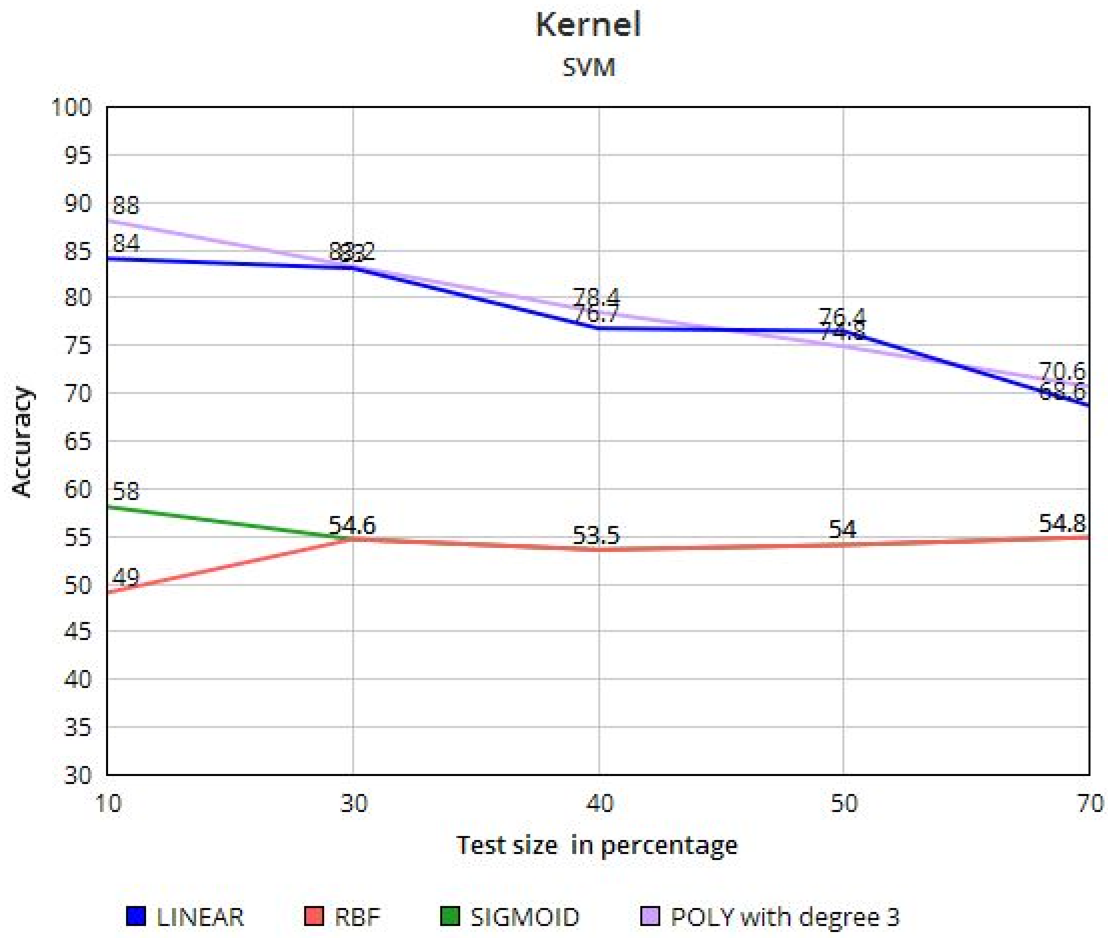
```
train, test = train_test_split(_file1, test_size = 0.4)
linear_svm=svm.SVC(kernel='linear')###########################################################
#############linear_SVM
rbf_svm=svm.SVC(kernel='rbf')#################################################################
#############rbf_SVM
kpca = KernelPCA(n_components=55,kernel="rbf", fit_inverse_transform=True,
gamma=10)#######################################KPCA

#print train,test
train=train.values.tolist()
test=test.values.tolist()
#print len(train[:][:]),len(test)
##############################################################################################
#############################data
x=[]
y=[]
for i in train:
        x.append(i[:-2])
        y.append(i[-1])
##############################################################################################
##########################end data

##############################################################################################
#kpca
X_kpca = kpca.fit_transform(x)
X_back = kpca.inverse_transform(X_kpca)
pca = PCA()
X_pca = pca.fit_transform(x)
#X_kpca=X_kpca.values.tolist()
#print X_kpca
linear_svm.fit(X_kpca,y)
rbf_svm.fit(X_kpca,y)
x=[]
for i in X_kpca:
        x.append(i[:])
P_l=linear_svm.predict(x)
P_r=rbf_svm.predict(x)
#print P_l[:],P_r[:],y
acc_l=0
acc_r=0
total=0
for i in y:
        if P_l[total]==i:
                acc_l=acc_l+1
        if P_r[total]==i:
                acc_r=acc_r+1
        total=total+1
A_l=acc_l/float(total)
A_r=acc_r/float(total)
print "linear  rbf with n_components"
print A_l*100,A_r*100
```

3 Kernel svm



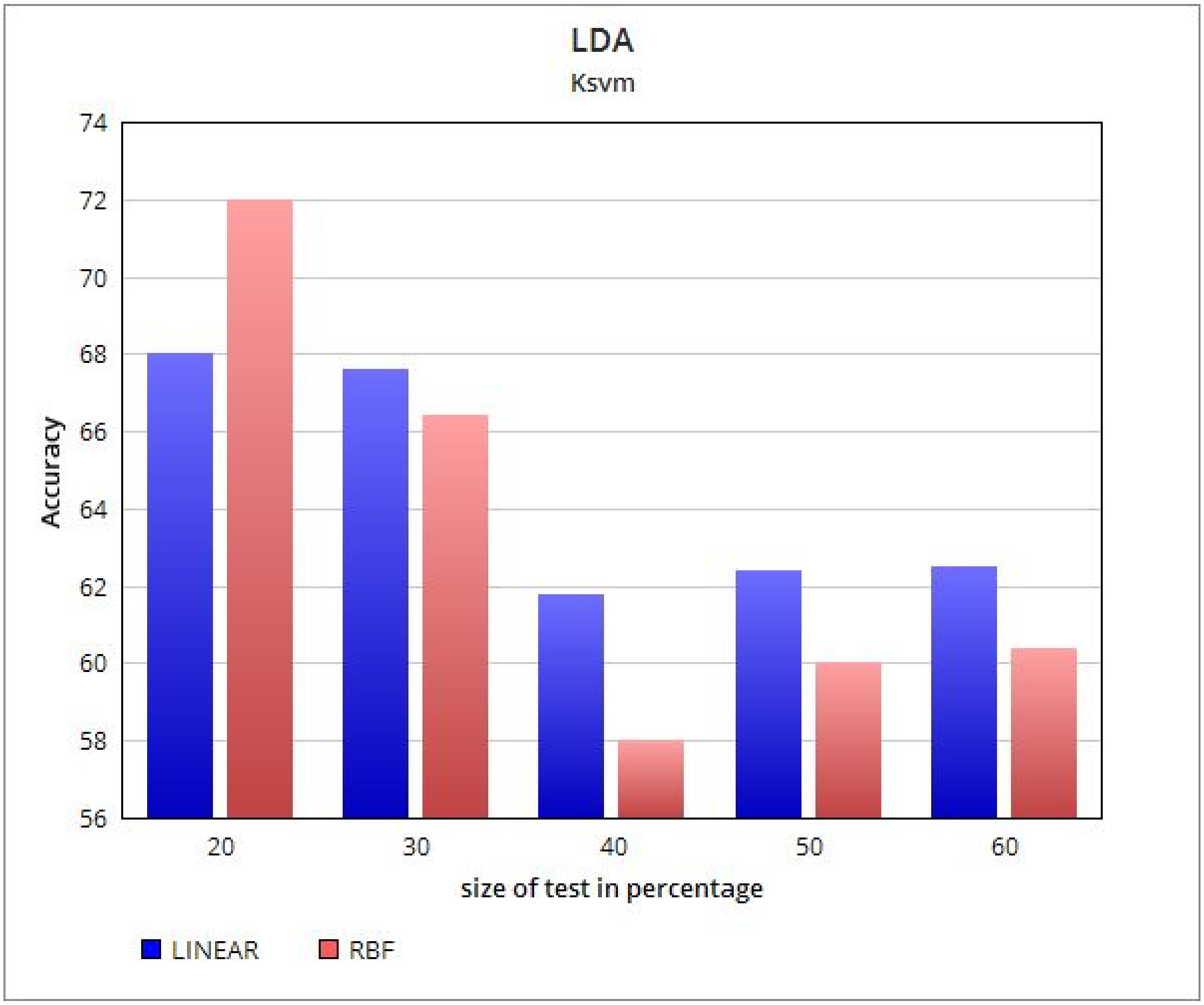


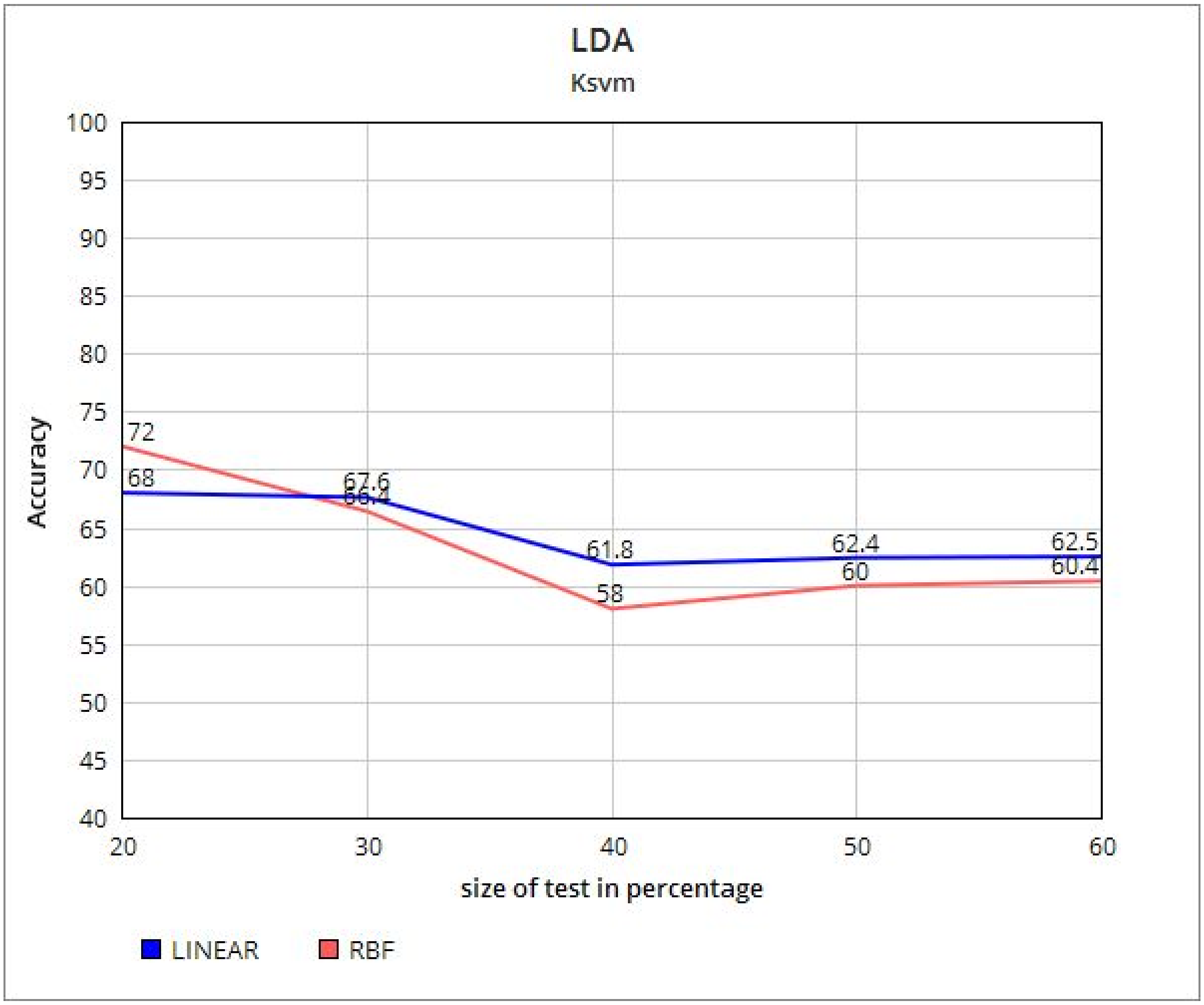
Code for Kernel svm with different kernel

```
from sklearn import svm
import pandas as pd
import numpy as np
from sklearn.cross_validation import train_test_split
_file1=pd.read_csv('arcene_train.csv', delim_whitespace=True)
_file2=pd.read_csv('arcene_train.labels')
_file1['Class'] = (_file2['1']).astype(int)
#print _file1
train, test = train_test_split(_file1, test_size = 0.4)
linear_svm=svm.SVC(kernel='linear')
rbf_svm=svm.SVC(kernel='rbf')
sigmoid_svm=svm.SVC(kernel='sigmoid')
poly_svm=svm.SVC(kernel='poly')
#print train,test
train=train.values.tolist()
test=test.values.tolist()
#print len(train[:][:]),len(test)
```

```
x=[]
y=[]
for i in train:
    x.append(i[:-2])
    y.append(i[-1])
#print len(x),y
linear_svm.fit(x,y)
rbf_svm.fit(x,y)
sigmoid_svm.fit(x,y)
poly_svm.fit(x,y)
x=[]
y=[]
for i in test:
    x.append(i[:-2])
    y.append(i[-1])
P_l=linear_svm.predict(x)
P_r=rbf_svm.predict(x)
P_s=sigmoid_svm.predict(x)
P_p=poly_svm.predict(x)
#print P_l[:],P_r[:],y
acc_l=0
acc_r=0
acc_s=0
acc_p=0
total=0
for i in y:
    if P_l[total]==i:
        acc_l=acc_l+1
    if P_r[total]==i:
        acc_r=acc_r+1
    if P_s[total]==i:
        acc_s=acc_s+1
    if P_p[total]==i:
        acc_p=acc_p+1
    total=total+1
A_l=acc_l/float(total)
A_r=acc_r/float(total)
A_s=acc_s/float(total)
A_p=acc_p/float(total)
print A_l*100,A_r*100,A_s*100,A_p*100
```

1-dimensional LDA


LDA
Ksvm
74
72
70
68
66
64
62
60
58
56
Accuracy
20
30
40
50
60
size of test in percentage
🟦 LINEAR
🟥 RBF



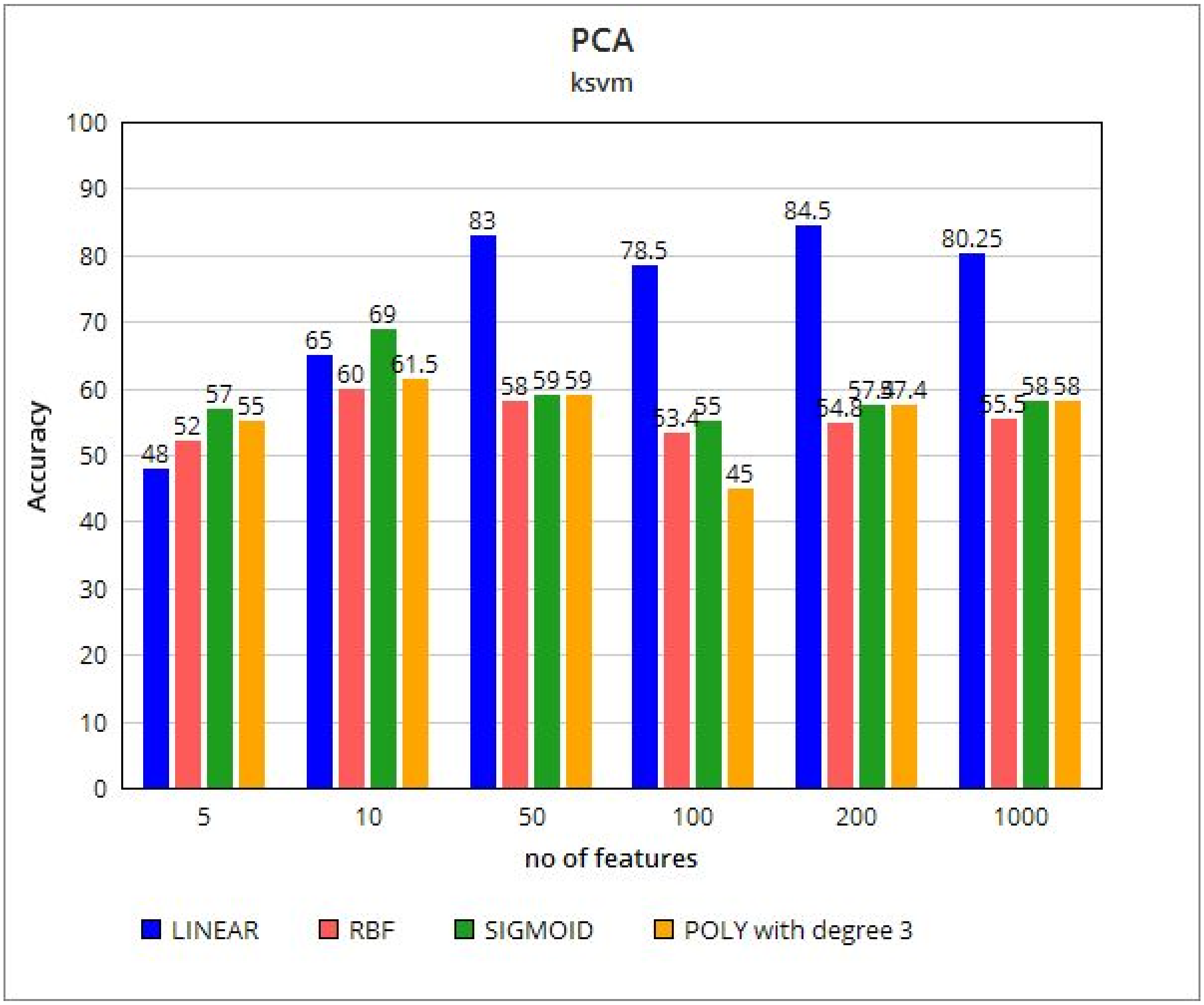
Code for LDA

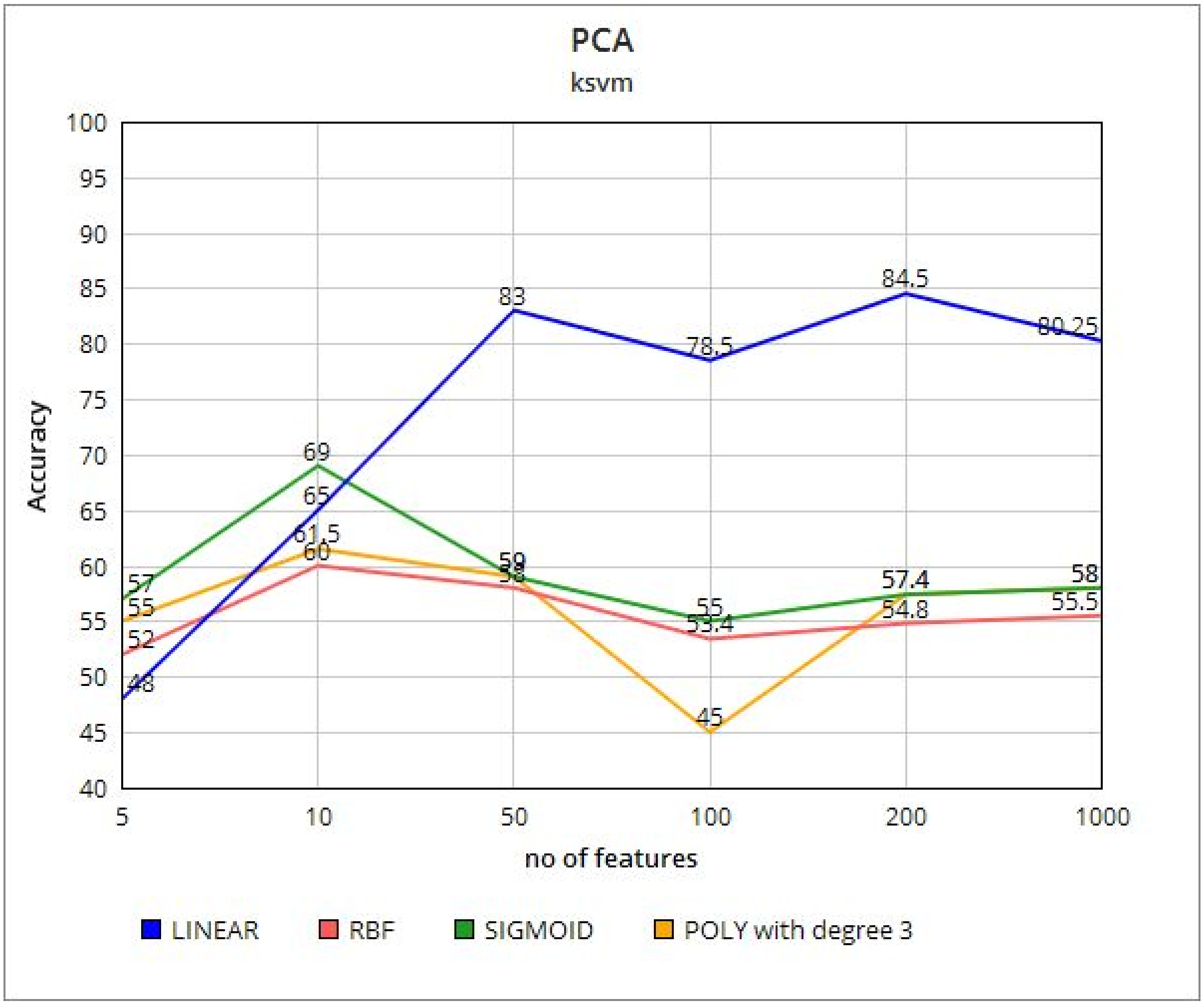
```
from sklearn import svm
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.decomposition import PCA, KernelPCA
import pandas as pd
import numpy as np
from sklearn.cross_validation import train_test_split
_file1=pd.read_csv('arcene_train.csv', delim_whitespace=True)
_file2=pd.read_csv('arcene_train.labels')
_file1['Class'] = (_file2['1']).astype(int)
##################################################################################cleaning
missing values
_file1= _file1.fillna(lambda x: x.median())
#print _file1
train, test = train_test_split(_file1, test_size = 0.2)
linear_svm=svm.SVC(kernel='linear')#################################################################
#############linear_SVM
```

```
rbf_svm=svm.SVC(kernel='rbf')#################################################################
#############rbf_SVM

#print train,test
train=train.values.tolist()
test=test.values.tolist()
#print len(train[:][:]),len(test)
x=[]
y=[]
for i in train:
        x.append(i[:-2])
        y.append(i[-1])
#print len(x),y
_lda =
LinearDiscriminantAnalysis(n_components=5)####################################################
#####################LDA
_lda.fit(x, y)
z=_lda.transform(x)
print len(z)
linear_svm.fit(z,y)
rbf_svm.fit(z,y)
x=[]
y=[]
for i in test:
        x.append(i[:-2])
        y.append(i[-1])
z=_lda.transform(x)
P_l=linear_svm.predict(z)
P_r=rbf_svm.predict(z)
#print P_l[:],P_r[:],y
acc_l=0
acc_r=0
total=0
for i in y:
        if P_l[total]==i:
                acc_l=acc_l+1
        if P_r[total]==i:
                acc_r=acc_r+1
        total=total+1
A_l=acc_l/float(total)
A_r=acc_r/float(total)
print A_l*100,A_r*100
```

PCA with svm


PCA
ksvm
Accuracy
100
90
80
70
60
50
40
30
20
10
0
48
52
57
55
65
60
69
61.5
83
58
59
59
78.5
53.4
55
45
84.5
54.8
57.5
57.4
80.25
55.5
58
58
5
10
50
100
200
1000
no of features
LINEAR
RBF
SIGMOID
POLY with degree 3



Code for PCA & SVM

```
from sklearn import svm
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.decomposition import PCA, KernelPCA
import pandas as pd
import numpy as np
from sklearn.cross_validation import train_test_split
_file1=pd.read_csv('arcene_train.csv', delim_whitespace=True)
_file2=pd.read_csv('arcene_train.labels')
_file1['Class'] = (_file2['1']).astype(int)
##################################################################################cleaning
missing values
_file1= _file1.fillna(lambda x: x.median())
#print _file1
train, test = train_test_split(_file1, test_size = 0.4)
linear_svm=svm.SVC(kernel='linear')###########################################################
#############linear_SVM
```

```
rbf_svm=svm.SVC(kernel='rbf')#################################################################
#############rbf_SVM
sigmoid_svm=svm.SVC(kernel='sigmoid')
poly_svm=svm.SVC(kernel='poly')
#print train,test
train=train.values.tolist()
test=test.values.tolist()
#print len(train[:][:]),len(test)
x=[]
y=[]
for i in train:
        x.append(i[:-2])
        y.append(i[-1])
#print len(x),y
_lda =
PCA(n_components=200)#########################################################################
LDA
_lda.fit(x, y)
z=_lda.transform(x)
#print z.shape
#print len(z)
linear_svm.fit(z,y)
rbf_svm.fit(z,y)
sigmoid_svm.fit(z,y)
poly_svm.fit(z,y)
x=[]
y=[]
for i in test:
        x.append(i[:-2])
        y.append(i[-1])
z=_lda.transform(x)
P_l=linear_svm.predict(z)
P_r=rbf_svm.predict(z)
P_s=sigmoid_svm.predict(z)
P_p=poly_svm.predict(z)
#print P_l[:],P_r[:],y
acc_l=0
acc_r=0
acc_s=0
acc_p=0
total=0
for i in y:
        if P_l[total]==i:
                acc_l=acc_l+1
        if P_r[total]==i:
                acc_r=acc_r+1
        if P_s[total]==i:
                acc_s=acc_r+1
        if P_p[total]==i:
                acc_p=acc_r+1
        total=total+1
A_l=acc_l/float(total)
A_r=acc_r/float(total)
A_s=acc_s/float(total)
A_p=acc_p/float(total)
print A_l*100,A_r*100,A_s*100,A_p*100
```