In [1]:
```
import pandas as pd
from sklearn import svm

data = pd.read_csv("/Users/user/documents/Class/BMEN415/GitHub/BMEN415FinalProject/Dat

#Converted the group to dummy variables
data.loc[data.Group=='Nondemented', 'Group'] = 0
data.loc[data.Group=='Demented', 'Group'] = 1
data.loc[data.Group=='Converted', 'Group'] = 2

#Converted the M/F to dummy variables
data.loc[data.Sex=='M', 'Sex'] = 0
data.loc[data.Sex=='F', 'Sex'] = 1

#ToDo
#Fill in SES and Mini Mental State missing sample values
#Use the mean of that category for the value
# Get rid of SES data

data
```

Out[1]:

| | Subject ID | MRI ID | Group | Visit | MR Delay | Sex | Hand | Age | EDUC | SES | Mini Mental State | Clinica Dementi Ratin |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | OAS2_0001 | OAS2_0001_MR1 | 0 | 1 | 0 | 0 | R | 87 | 14 | 2.0 | 27.0 | 0. |
| **1** | OAS2_0001 | OAS2_0001_MR2 | 0 | 2 | 457 | 0 | R | 88 | 14 | 2.0 | 30.0 | 0. |
| **2** | OAS2_0002 | OAS2_0002_MR1 | 1 | 1 | 0 | 0 | R | 75 | 12 | NaN | 23.0 | 0. |
| **3** | OAS2_0002 | OAS2_0002_MR2 | 1 | 2 | 560 | 0 | R | 76 | 12 | NaN | 28.0 | 0. |
| **4** | OAS2_0002 | OAS2_0002_MR3 | 1 | 3 | 1895 | 0 | R | 80 | 12 | NaN | 22.0 | 0. |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | . |
| **368** | OAS2_0185 | OAS2_0185_MR2 | 1 | 2 | 842 | 0 | R | 82 | 16 | 1.0 | 28.0 | 0. |
| **369** | OAS2_0185 | OAS2_0185_MR3 | 1 | 3 | 2297 | 0 | R | 86 | 16 | 1.0 | 26.0 | 0. |
| **370** | OAS2_0186 | OAS2_0186_MR1 | 0 | 1 | 0 | 1 | R | 61 | 13 | 2.0 | 30.0 | 0. |
| **371** | OAS2_0186 | OAS2_0186_MR2 | 0 | 2 | 763 | 1 | R | 63 | 13 | 2.0 | 30.0 | 0. |
| **372** | OAS2_0186 | OAS2_0186_MR3 | 0 | 3 | 1608 | 1 | R | 65 | 13 | 2.0 | 30.0 | 0. |

373 rows × 15 columns

In [2]:
```
import numpy as np
from sklearn.model_selection import train_test_split
```

In [3]:
```
#PCA might be a good technique to select predictors

#note that PCA performs best when data is normalized (range b/w 0 and 1)

#It is possible to use categorical and continuous predictors
#for a regression problem. My understanding is you need to make
#dummy variables for the binary predictors.

#Variables that we will need to deal with:
# Hand, Visit, Subject ID, MRI ID
```

In [4]:
```
#Attempting PCA on data
#Hand is completely useless as it is identical for all samples
data_drop = data.drop(['Hand','Visit','Subject ID','MRI ID'], axis = 1) #axis = 1 mea

#get rid of row 360 and 359 bc they are missing alot of data (both SES and MMS)
data_drop = data_drop.drop([360, 359])


#delete all data points that dont have SES in them (this is where they have NaN)
data_drop = data_drop.dropna()

#dementia status is what we want to predict - change this to single target
group = data_drop[['Group']]


data_drop = data_drop.drop(['Group'], axis = 1) #axis = 1 means to drop column not ro
```

In [5]:
```
# give it a label type
group = group.astype('int')
```

In [6]:
```
#get a list of columns in pandas object
names_of_data = data_drop.columns.tolist()

#shuffle = false prevents data split being different everytime
X_train, X_test, y_train, y_test = train_test_split(data_drop, group, test_size=0.2, s

#split test into validate and test, again making sure the data is always the same for
#X_test, X_val, y_test, y_val = train_test_split(X_train, y_train, test_size=0.25, shu

#Normalizing the data
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

#running the actual PCA
from sklearn.decomposition import PCA

pca = PCA()
X_train = pca.fit_transform(X_train)
X_test = pca.transform(X_test)

#relief f algorithm - sorting features
```

In [7]:
```
data_drop.loc[355]
```

Out[7]:
```
MR Delay                              652
Sex                                     0
Age                                    81
EDUC                                   20
SES                                     1
Mini Mental State                      26
Clinical Dementia Rating              0.5
Estimated total Intracranial Volume  1556
Normalize Whole Brain Volume        0.691
Atlas Scaling Factor                1.128
Name: 355, dtype: object
```

In [8]:
```
data_drop.loc[354]
```

Out[8]:
```
MR Delay                                0
Sex                                     0
Age                                    79
EDUC                                   20
SES                                     1
Mini Mental State                      26
Clinical Dementia Rating              0.5
Estimated total Intracranial Volume  1548
Normalize Whole Brain Volume        0.711
Atlas Scaling Factor                1.134
Name: 354, dtype: object
```

In [9]:
```
explained_variance = pca.explained_variance_ratio_
print(len(explained_variance))
print(explained_variance)
```

```
10
[0.2790578  0.21909196 0.14981908 0.12851434 0.08639602 0.05229841
 0.03160894 0.02928376 0.02278443 0.00114525]
```

## From here on out is SVM

In [10]:
```
#Create SVM
AlzSVM = svm.SVC(kernel='rbf')
#Train SVM
AlzSVM.fit(X_train,y_train.values.ravel())
#Predict SVM
predictSVM = AlzSVM.predict(X_test)
```

In [11]:
```
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
print(confusion_matrix(y_test, predictSVM))
print(classification_report(y_test, predictSVM))
results = classification_report(y_test, predictSVM)

accuracy_score(y_test, predictSVM)
```

```
[[38  0  0]
 [ 1 22  3]
 [ 4  1  2]]
              precision    recall  f1-score   support

           0       0.88      1.00      0.94        38
           1       0.96      0.85      0.90        26
           2       0.40      0.29      0.33         7

    accuracy                           0.87        71
   macro avg       0.75      0.71      0.72        71
weighted avg       0.86      0.87      0.86        71
```

Out[11]:
```
0.8732394366197183
```