PROJECT SPECIFICATION ¶

Machine Learning Capstone Project

This capstone project involves machine learning modeling and analysis of clinical, demographic, and brain related derived anatomic measures from human MRI (magnetic resonance imaging) tests (http://www.oasis-brains.org/ (http://www.oasis-brains.org/)). The objectives of these measurements are to diagnose the level of Dementia in the individuals and the probability that these individuals may have Alzheimer's Disease (AD).

In published studies, Machine Learning has been applied to Alzheimer's/Dementia identification from MRI scans and related data in the academic papers/theses in References 10 and 11 listed in the References Section below. Recently, a close relative of mine had to undergo a sequence of MRI tests for cognition difficulties.The motivation for choosing this topic for the Capstone project arose from the desire to understand and analyze potential for Dementia and AD from MRI related data. This Capstone project does not use the MRI "imaging" data and does not focus on AD, focusses only on Dementia.

Problem Statement

[The problem which needs to be solved is clearly defined. A strategy for solving the problem, including discussion of the expected solution, has been made.]

• Cross-Sectional and longitudinal OASIS MRI structural and demographic data (clinical, demographic, and brain related derived anatomic measures) from human MRI (magnetic resonance imaging) tests (http://www.oasis-brains.org/ (http://www.oasis-brains.org/)) will be used to train a set of linear and non-linear machine learning classification models.

• Clinical Dementia Rating (CDR) values provided in the data set will be used as "labels" for training the classification models. [Clinical Dementia Ratings (CDR values: 0=nondemented; 0.5 = very mild dementia; 1 = mild dementia; 2 = moderate dementia)].

• Pandas will be used for data loading and Python scikit-learn library for modeling.

• The goal is to train machine learning models to predict whether the individuals in the cross-validation set (test set) have dementia (CDR>0), and if they do, the severity level of dementia (CDR values of 0.5, 1, and 2). The problem will be formulated both as a binary classification problem (CDR=0, and CDR>0), and multiclass classification problem (CDR values in the dataset: 0, 0.5, 1, and 2). In the binary classification formulation, the CDR>0 the values in the sliced dataset will be relabeled as CDR=1.

• Classification Accuracy will be used as the primary metric. Additionally, for binary classification AUC/ROC values will be reported. For multi- class classification (multiple CDR labels) F-1 score will be reported. The results from the best model will be reported along with those from the other models.

• About 80% of the data in the dataset will be used for training the models. About 20% of data will used prediction of the CDR label for the k-fold cross-validation with k=10. Sensitivity studies with proportion other than 80:20, e.g. 70:30, will be used to test sensitivity of this split on the accuracy.

• The base case uses a dataset that combines the cross-sectional and the longitudinal MRI datasets. This has the benefit of having a larger dataset. The cross-sectional and the longitudinal datasets will also be trained/cross-validated separately, and classification accuracy will be reported.

• Data cleaning (e.g. removal of NaN values), data exploration, data preparation, data visualization, and data preprocessing will be described, as appropriate, and the impact of the latter on prediction metrics will discussed.

References

1. The Open Access Series of Imaging Studies (OASIS), http://www.oasis-brains.org/app/template/Index.vm;jsessionid=6926BBF18A3D5CD974E750FAC8ED01CE (http://www.oasis-brains.org/app/template/Index.vm;jsessionid=6926BBF18A3D5CD974E750FAC8ED01CE)
2. OASIS Fact Sheet (rev. 2007-8-20) Cross-Sectional Data Across the Adult Lifespan, Marcus et al., 2007, http://www.oasis-brains.org/pdf/oasis_cross-sectional_facts.pdf (http://www.oasis-brains.org/pdf/oasis_cross-sectional_facts.pdf)
3. MRI Reliability data across the adult lifespan, http://www.oasis- (http://www.oasis-) brains.org/app/action/BundleAction/bundle/OAS1_RELIABILITY
4. Buckner, RL, Head, D, Parker, J, Fotenos, AF, Marcus, D, Morris, JC, Snyder, AZ, 2004, "A unified approach for morphometric and functional data analysis in young, old, and demented adults using automated atlas-based head size normalization: reliability and validation against manual measurement of total intracranial volume", Neuroimage 23, 724-38.
5. Fotenos, AF, Snyder, AZ, Girton, LE, Morris, JC, and Buckner, RL, 2005, "Normative estimates of crosssectional and longitudinal brain volume decline in aging and AD", Neurology, 64: 1032-1039.
6. Marcus, DS, Wang, TH, Parker, J, M, Csernansky, JG, Morris, JC, Buckner, RL, 2007. "Open Access Series of Imaging Studies (OASIS): Cross-Sectional MRI Data in Young, Middle Aged, Nondemented and Demented Older Adults", Journal of Cognitive Neuroscience, 19, 1498-1507.
7. Morris, JC, 1993. "The Clinical Dementia Rating (CDR): current version and scoring rules", Neurology 43, 2412b-2414b.
8. Rubin, EH, Storandt, M, Miller, JP, Kinscherf, DA, Grant, EA, Morris, JC, Berg, L, "A prospective study of cognitive function and onset of dementia in cognitively healthy elders". Arch Neurol. 55, 395- 401.
9. Zhang, Y, Brady, M, Smith, S, "Segmentation of brain MR images through a hidden Markov random field model and the expectation maximization algorithm". IEEE Trans. on Medical Imaging, 20(1):45-57.
10. "Diagnosis of Alzheimer's Disease Based on Structural MRI Images Using a Regularized Extreme Learning Machine and PCA Feature", https://www.hindawi.com/journals/jhe/2017/5485080/ (https://www.hindawi.com/journals/jhe/2017/5485080/)

11. "Use of Machine Learning Technology in the Diagnosis of Alzheimer's Disease", http://doras.dcu.ie/21356/1/Noel_s_Master_s_thesis__Copy_(1).pdf (http://doras.dcu.ie/21356/1/Noel_s_Master_s_thesis__Copy_(1).pdf)
12. Scikit-learn, http://scikit-learn.org/stable/index.html (http://scikit-learn.org/stable/index.html)
13. Model evaluation: quantifying the quality of predictions, http://scikit-learn.org/stable/modules/model_evaluation.html (http://scikit-learn.org/stable/modules/model_evaluation.html)).
14. Precision and Recall, http://scikit-learn.org/stable/auto_examples/model_selection/plot_precision_recall.html#sphx-glr-auto-examples-model-selection-plot-precision-recall-py (http://scikit-learn.org/stable/auto_examples/model_selection/plot_precision_recall.html#sphx-glr-auto-examples-model-selection-plot-precision-recall-py)
15. Confusion Matrix, http://scikit-learn.org/stable/modules/model_evaluation.html#confusion-matrix (http://scikit-learn.org/stable/modules/model_evaluation.html#confusion-matrix)
16. Support, http://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_recall_fscore_support.html (http://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_recall_fscore_support.html)
17. F1-score, http://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html (http://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html)
18. "Conditional data slicing in a Pandas dataframe", https://stackoverflow.com/questions/17071871/select-rows-from-a-dataframe-based-on-values-in-a-column-in-pandas (https://stackoverflow.com/questions/17071871/select-rows-from-a-dataframe-based-on-values-in-a-column-in-pandas)
19. "Matplotlib colormap examples and color schemes for using in heatmap", http://pyhogs.github.io/colormap-examples.html (http://pyhogs.github.io/colormap-examples.html); https://matplotlib.org/examples/color/colormaps_reference.html (https://matplotlib.org/examples/color/colormaps_reference.html)
20. "Usefulness of data from magnetic resonance imaging to improve prediction of dementia: population based cohort study" http://www.bmj.com/content/350/bmj.h2863 (http://www.bmj.com/content/350/bmj.h2863)
21. C - Statistics: http://www.statisticshowto.com/c-statistic/ (http://www.statisticshowto.com/c-statistic/)
22. The Use of MRI and PET for Clinical Diagnosis of Dementia and Investigation of Cognitive Impairment: A Consensus Report https://www.alz.org/national/documents/imaging_consensus_report.pdf (https://www.alz.org/national/documents/imaging_consensus_report.pdf)
23. Knopman DS, DeKosky ST, Cummings JL, Chui H, Corey-Bloom J, Relkin N, et al. Practice parameter: Diagnosis of dementia (anevidence-based review). Report of the Quality Standards Subcommittee of the American Academy of Neurology. Neurology 2001;56(9):1143–1153.
24. Machine Learning Mastery, https://machinelearningmastery.com/ (https://machinelearningmastery.com/)
25. "The Role of Balanced Training and Testing Data Sets for Binary Classifiers in Bioinformatics", https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3706434/ (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3706434/)
26. "8 Proven Ways for improving the "Accuracy" of a Machine Learning Model", https://www.analyticsvidhya.com/blog/2015/12/improve-machine-learning-results/ (https://www.analyticsvidhya.com/blog/2015/12/improve-machine-learning-results/)

27. “A Gentle Introduction to the Gradient Boosting Algorithm for Machine Learning”, https://machinelearningmastery.com/gentle-introduction-gradient-boosting-algorithm-machine-learning/ (https://machinelearningmastery.com/gentle-introduction-gradient-boosting-algorithm-machine-learning/)
28. "Gradient Boosting Tree vs. Random Forest", https://stats.stackexchange.com/questions/173390/gradient-boosting-tree-vs-random-forest (https://stats.stackexchange.com/questions/173390/gradient-boosting-tree-vs-random-forest)

Metrics

*[Metrics used to measure performance of a model or result are clearly defined. Metrics are justified based on the characteristics of the problem.]

Classification Accuracy is used as the primary metric. This metric is applicable for binary classification where the number of records for the two labels are balanced. As shown later in this notebook, the ratio of the number of processed records with CDR=0 and CDR=1 is 60% to 40%, and is considered balanced. Classification accuracy is defined as the number of records correctly classified divided by the total number of records classified. Additionally, for binary classification here, AUC/ROC values are reported. Accuracy results are also be reported in sklearn Confusion Matrix format to evaluate classifier output quality, and in Classification Report format (provides, precision, recall, and f1-score) which are quite appropriate for the dataset used to train models for CDR classification. References 13 through 17 have details and discussion of these sklearn metrics.

Benchmark

[Student clearly defines a benchmark result or threshold for comparing performances of solutions obtained.\

My primandary benchmark is a neural network model (Appendix 1) based on the same data discussed above and as used in this problem: Keras is used as the frontend with tensorflow backend. My secondary benchmark will be results of the study in the two papers below:

** Paper title: Usefulness of data from magnetic resonance imaging to improve prediction of dementia: population based cohort study, Reference 20

"Results During 10 years of follow-up, there were 119 confirmed cases of dementia, 84 of which were Alzheimer’s disease. The conventional risk model incorporated age, sex, education, cognition, physical function, lifestyle (smoking, alcohol use), health (cardiovascular disease, diabetes, systolic blood pressure), and the apolipoprotein genotype (C statistic for discrimination performance was 0.77, 95% confidence interval 0.71 to 0.82). No significant differences were observed in the discrimination performance of the conventional risk model compared with models incorporating data from MRI including white matter lesion volume (C statistic 0.77, 95% confidence interval 0.72 to 0.82; P=0.48 for difference of C statistics, Reference 21), brain volume (0.77, 0.72 to 0.82; P=0.60), hippocampal volume (0.79, 0.74 to 0.84; P=0.07), or all three variables combined (0.79, 0.75 to 0.84; P=0.05). Inclusion of hippocampal volume or all three MRI variables combined in the conventional model did, however, lead to significant improvement in reclassification measured by

using the integrated discrimination improvement index (P=0.03 and P=0.04) and showed increased net benefit in decision curve analysis. Similar results were observed when the outcome was restricted to Alzheimer's disease."

**Paper Title: The Use of MRI and PET for Clinical Diagnosis of Dementia and Investigation of Cognitive Impairment: A Consensus Report, Reference 22.

"Once the presence of dementia has been established, the role of imaging in the diagnosis of dementia subtypes is very much a function of the clinical diagnosis. The accuracy of the clinical diagnosis of Alzheimer's disease (AD) is quite good. Pathological AD has a prevalence of about 70% (range 50% to above 80% depending upon whether the AD occurs in isolation or with other entities) among all dementias (see evidence Table 1 in Reference 23); thus, even clinicians with limited neurological expertise should have a diagnostic accuracy, for AD at least, at about that level. A review of 13 published studies gave average values for sensitivity and specificity of the clinical diagnosis of AD of 81% and 70%, respectively(Reference 23). The overall accuracy of the clinical diagnosis of AD versus not-AD compared with the neuropathological standard based on those values for prevalence, sensitivity, and specificity, is 78%. "

Datasets and Inputs

Reference 1 provides the downloadable MRI related data in csv format. Reference 2 provides metadata and additional facts about the cross-sectional MRI. **OASIS Cross-sectional MRI Data in Young, Middle Aged, Non-demented and Demented Older Adults**

• This dataset consists of a cross-sectional collection for 416 persons aged 18 to 96

• For each person, 3 to 4 T1-weighted MRI scans that were obtained in single scan sessions are included.

• The persons include both men and women, and are all right-handed.

• In this dataset, one hundred persons over the age of 60 have been clinically diagnosed with very mild to moderate Alzheimer's disease (AD).

• Also, a reliability data set , Reference 3, is included which contains 20 non-demented subjects imaged on a subsequent visit within 90 days of their initial session.

• Dementia related Additional Data below for the cross-sectional MRI cases used this project. Features based on these Additional Data will be used to train classification models to predict the labels for the outcome (CDR).

OASIS: Longitudinal MRI Data in Non-demented and Demented Older Adults

This set consists of a longitudinal collection of 150 subjects aged 60 to 96. Each subject was scanned on two or more visits, separated by at least one year for a total of 373 imaging sessions. For each subject, 3 or 4 individual T1-weighted MRI scans obtained in single scan sessions are included.

Note: MRI image pixel data are NOT used in this problem, only related features prefixed with @ sign (below) will be used.

• The subjects are all right-handed and include both men and women.

• 72 of the subjects were characterized as non-demented throughout the study.

• 64 of the included subjects were characterized as demented at the time of their initial visits and remained so for subsequent scans, including 51 individuals with mild to moderate Alzheimer's disease.

• Another 14 subjects were characterized as non-demented at the time of their initial visit and were subsequently characterized as demented at a later visit.

• Dementia related **Additional Data** below for the longitudinal MRI cases are used this project.

Features based on the **Additional Data** are relevant to finding machine learning solutions to the problem defined above, and will be used to train classification models to predict the labels for the outcome (Critical Dementia Rating, CDR).

Additional Data: Specific References in parentheses below covering features are from Reference 2. These features include Demographic, clinical, and derived anatomic measures related to brain that are located in the file oasis_crosssectional.csv. Features prefixed with @ will be used for the problem.

Demographic data:

```
- @Gender (M/F), categorical data
- Handedness (Right or Left Handed), categorical data, all of which are r
ight handed in the dataset.
- @Age (numeric),
- @Education (Educ, categorical). Education codes correspond to the follo
wing levels of education:
    1=Less than high school graduate.
    2=High school graduate.
    3=Some college education
    4=College graduate.
    5=Beyond college.
```

Clinical data:

```
- @Mini-Mental State Examination (MMSE, Reference 8),

- @Clinical Dementia Rating (CDR, Reference 7)
     0 = non-demented (341 data points)
     0.5 = very mild dementia (193 data points)
     1 = mild dementia (69 data points)
     2 = moderate dementia (5 data points)
```

There are some records with NaN values in one or more fields; these records will be removed from datasets prior to analysis. All participants with dementia (CDR >0) were diagnosed with probable Alzheimer's Disease.

Derived anatomic volumes data:

- @Estimated total intracranial volume (eTIV, mm3), Reference 4
- @Atlas scaling factor (ASF), Reference 4
- @Normalized whole brain volume (nWBV, mm3), Reference 5

Load Libraries

In [56]:
```
# Load libraries
import numpy as np
from pandas import read_csv
from pandas.tools.plotting import scatter_matrix
from matplotlib import pyplot
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import RandomForestClassifier
```

Load CSV data to Pandas Dataframes

In [57]:
```
# Read a local csvfile in Pandas for the OASIS cross-sectional MRI dataset do
import pandas as pd
dfoasx=pd.read_csv('oasis_cross-sectional.csv')
dfoasx.shape
```

Out[57]: (436, 12)

436 rows and 12 columns of data

In [58]:
```
# Read a csvfile in Pandas for the OASIS longitudinal MRI dataset downloaded
import pandas as pd
dfoasl=pd.read_csv('oasis_longitudinal.csv')
dfoasl.shape
```

Out[58]: (373, 15)

373 rows and 15 columns of data.

Analysis

Data Exploration

[If a dataset is present, features and calculated statistics relevant to the problem have been reported and discussed, along with a sampling of the data. In lieu of a dataset, a thorough description of the input space or input data has been made. Abnormalities or characteristics about the data or input that need to be addressed have been identified.]

- Dataset shape (rows, columns)
- Column headings
- Data Summary and statistics
- Data Types
-

OASIS Cross-sectional Data Exploration

In [59]:
```
1 list(dfoasx.columns.values)
```

Out[59]:
```
['ID',
 'M/F',
 'Hand',
 'Age',
 'Educ',
 'SES',
 'MMSE',
 'CDR',
 'eTIV',
 'nWBV',
 'ASF',
 'Delay']
```

In [60]:
```
1 dfoasx.describe()
```

Out[60]:

	Age	Educ	SES	MMSE	CDR	eTIV	nWBV	
count	436.000000	235.000000	216.000000	235.00000	235.000000	436.000000	436.000000	436.00
mean	51.357798	3.178723	2.490741	27.06383	0.285106	1481.919725	0.791670	1.19
std	25.269862	1.311510	1.120593	3.69687	0.383405	158.740866	0.059937	0.12
min	18.000000	1.000000	1.000000	14.00000	0.000000	1123.000000	0.644000	0.88
25%	23.000000	2.000000	2.000000	26.00000	0.000000	1367.750000	0.742750	1.11
50%	54.000000	3.000000	2.000000	29.00000	0.000000	1475.500000	0.809000	1.19
75%	74.000000	4.000000	3.000000	30.00000	0.500000	1579.250000	0.842000	1.28
max	96.000000	5.000000	5.000000	30.00000	2.000000	1992.000000	0.893000	1.56

In [61]:
```
1 dfoasx.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 436 entries, 0 to 435
Data columns (total 12 columns):
ID       436 non-null object
M/F      436 non-null object
Hand     436 non-null object
Age      436 non-null int64
Educ     235 non-null float64
SES      216 non-null float64
MMSE     235 non-null float64
CDR      235 non-null float64
eTIV     436 non-null int64
nWBV     436 non-null float64
ASF      436 non-null float64
Delay    20 non-null float64
dtypes: float64(7), int64(2), object(3)
memory usage: 41.0+ KB
```

In [62]:
```
1 dfoasx.head(5)
```

Out[62]:

	ID	M/F	Hand	Age	Educ	SES	MMSE	CDR	eTIV	nWBV	ASF	Delay
0	OAS1_0001_MR1	F	R	74	2.0	3.0	29.0	0.0	1344	0.743	1.306	NaN
1	OAS1_0002_MR1	F	R	55	4.0	1.0	29.0	0.0	1147	0.810	1.531	NaN
2	OAS1_0003_MR1	F	R	73	4.0	3.0	27.0	0.5	1454	0.708	1.207	NaN
3	OAS1_0004_MR1	M	R	28	NaN	NaN	NaN	NaN	1588	0.803	1.105	NaN
4	OAS1_0005_MR1	M	R	18	NaN	NaN	NaN	NaN	1737	0.848	1.010	NaN

Cross-sectional MRI dataset has many rows with NaN values in multiple columns. These NaN values will be removed after merging the dataset with the Longitudinal MRI dataset.

In [63]:
```
1 typesx=dfoasx.dtypes
2 print(typesx)
```

```
ID        object
M/F       object
Hand      object
Age        int64
Educ     float64
SES      float64
MMSE     float64
CDR      float64
eTIV       int64
nWBV     float64
ASF      float64
Delay    float64
dtype: object
```

OASIS Longitudinal Data Exploration

In [64]:
```
# Determine the count of various values for the label CDR
print (dfoasx.CDR[(dfoasx.CDR == 0.0)].count(),",", dfoasx.CDR[(dfoasx.CDR ==
print (dfoasx.CDR[(dfoasx.CDR == 1.0)].count(),",", dfoasx.CDR[(dfoasx.CDR ==

```

```
135 , 70
28 , 2
```

In [ ]:
```

```

In [65]:
```
import pandas as pd
## dfoasx cross-sectional MRI data frame
## CDR is name of column for which you want to calculate the NaN values
sum(pd.isnull(dfoasx['CDR']))
```

Out[65]: 201

OASIS Longitudinal Data Exploration¶

In [66]:
```
list(dfoasl.columns.values)
```

Out[66]:
```
['Subject ID',
 'MRI ID',
 'Group',
 'Visit',
 'MR Delay',
 'M/F',
 'Hand',
 'Age',
 'EDUC',
 'SES',
 'MMSE',
 'CDR',
 'eTIV',
 'nWBV',
 'ASF']
```

In [67]:
```
dfoasl.describe()
```

Out[67]:

	Visit	MR Delay	Age	EDUC	SES	MMSE	CDR	
count	373.000000	373.000000	373.000000	373.000000	354.000000	371.000000	373.000000	373.
mean	1.882038	595.104558	77.013405	14.597855	2.460452	27.342318	0.290885	1488.
std	0.922843	635.485118	7.640957	2.876339	1.134005	3.683244	0.374557	176.
min	1.000000	0.000000	60.000000	6.000000	1.000000	4.000000	0.000000	1106.
25%	1.000000	0.000000	71.000000	12.000000	2.000000	27.000000	0.000000	1357.
50%	2.000000	552.000000	77.000000	15.000000	2.000000	29.000000	0.000000	1470.
75%	2.000000	873.000000	82.000000	16.000000	3.000000	30.000000	0.500000	1597.
max	5.000000	2639.000000	98.000000	23.000000	5.000000	30.000000	2.000000	2004.

In [68]:
```
1 dfoasl.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 373 entries, 0 to 372
Data columns (total 15 columns):
Subject ID    373 non-null object
MRI ID        373 non-null object
Group         373 non-null object
Visit         373 non-null int64
MR Delay      373 non-null int64
M/F           373 non-null object
Hand          373 non-null object
Age           373 non-null int64
EDUC          373 non-null int64
SES           354 non-null float64
MMSE          371 non-null float64
CDR           373 non-null float64
eTIV          373 non-null int64
nWBV          373 non-null float64
ASF           373 non-null float64
dtypes: float64(5), int64(5), object(5)
memory usage: 43.8+ KB
```

The three additional columns (Subject ID, Group, and Visit) in dfoasl dataset are not in the dfoasx dataset. These three columns are not meaningful for the cDR classification, and will be dropped before merging the dloasx and dfoasl datasets. The MR Delay and Delay columns in the two datasets have same meaning and the MR Delay column will be repositioned to be in the same column order as the Delay column.

In [69]:
```
1 dfoasl.head(5)
```

Out[69]:

	Subject ID	MRI ID	Group	Visit	MR Delay	M/F	Hand	Age	EDUC	SES	MMSE
0	OAS2_0001	OAS2_0001_MR1	Nondemented	1	0	M	R	87	14	2.0	27.0
1	OAS2_0001	OAS2_0001_MR2	Nondemented	2	457	M	R	88	14	2.0	30.0
2	OAS2_0002	OAS2_0002_MR1	Demented	1	0	M	R	75	12	NaN	23.0
3	OAS2_0002	OAS2_0002_MR2	Demented	2	560	M	R	76	12	NaN	28.0
4	OAS2_0002	OAS2_0002_MR3	Demented	3	1895	M	R	80	12	NaN	22.0

Longitudinal MRI dataset has many rows with NaN values in multiple columns. These NaN values will be removed after merging the dataset with the Cross-sectional MRI dataset.

In [70]:
```
typesl=dfoasx.dtypes
print(typesl)
```

```
ID        object
M/F       object
Hand      object
Age        int64
Educ     float64
SES      float64
MMSE     float64
CDR      float64
eTIV       int64
nWBV     float64
ASF      float64
Delay    float64
dtype: object
```

In [71]:
```
# Determine the count of various values for the label CDR
print (dfoasl.CDR[(dfoasl.CDR == 0.0)].count(),",", dfoasl.CDR[(dfoasl.CDR ==
print (dfoasl.CDR[(dfoasl.CDR == 1.0)].count(),",", dfoasl.CDR[(dfoasl.CDR ==
```

```
206 , 123
41 , 3
```

In [72]:
```

import pandas as pd
## dfoasl cross-sectional MRI data frame
## CDR is name of column for which you want to calculate the NaN values
sum(pd.isnull(dfoasl['CDR']))
```

Out[72]: 0

- CDR = 0, 0.5, 1.0, 2, NaN
- dfoasx= 135, 70, 28, 2, 201
- dfoasl= 206, 123, 41, 3, 0
- Total = 341, 193, 69, 5, 201

Methodology

Data Preprocessing

All preprocessing steps have been clearly documented. Abnormalities or characteristics about the data or input that needed to be addressed have been corrected. If no data preprocessing is necessary, it has been clearly justified.

Wikipedia: "If there is much irrelevant and redundant information present or noisy and unreliable data, then knowledge discovery during the training phase is more difficult. Data pre-processing includes cleaning, instance selection, normalization, transformation, feature extraction and selection, etc. The product of data pre-processing is the final training set."

In [73]:
```
# Cross-sectional MRI data preparation and preprocessing
# Replace gender data "M" and "F" with numerical inputs 0, and 1
dfoasx['M/F'] = dfoasx['M/F'].replace('F', 1)
dfoasx['M/F'] = dfoasx['M/F'].replace('M', 0)
dfoasx.head(10)
```

Out[73]:

	ID	M/F	Hand	Age	Educ	SES	MMSE	CDR	eTIV	nWBV	ASF	Delay
0	OAS1_0001_MR1	1	R	74	2.0	3.0	29.0	0.0	1344	0.743	1.306	NaN
1	OAS1_0002_MR1	1	R	55	4.0	1.0	29.0	0.0	1147	0.810	1.531	NaN
2	OAS1_0003_MR1	1	R	73	4.0	3.0	27.0	0.5	1454	0.708	1.207	NaN
3	OAS1_0004_MR1	0	R	28	NaN	NaN	NaN	NaN	1588	0.803	1.105	NaN
4	OAS1_0005_MR1	0	R	18	NaN	NaN	NaN	NaN	1737	0.848	1.010	NaN
5	OAS1_0006_MR1	1	R	24	NaN	NaN	NaN	NaN	1131	0.862	1.551	NaN
6	OAS1_0007_MR1	0	R	21	NaN	NaN	NaN	NaN	1516	0.830	1.157	NaN
7	OAS1_0009_MR1	1	R	20	NaN	NaN	NaN	NaN	1505	0.843	1.166	NaN
8	OAS1_0010_MR1	0	R	74	5.0	2.0	30.0	0.0	1636	0.689	1.073	NaN
9	OAS1_0011_MR1	1	R	52	3.0	2.0	30.0	0.0	1321	0.827	1.329	NaN

In [74]:
```
# Determine the count of various values for the label CDR
print (dfoasx.CDR[(dfoasx.CDR == 0.0)].count(),",", dfoasx.CDR[(dfoasx.CDR ==
print (dfoasx.CDR[(dfoasx.CDR == 1.0)].count(),",", dfoasx.CDR[(dfoasx.CDR ==
```

```
135 , 70
28 , 2
```

In [75]:
```
# Longitudinal  MRI data preparation
# Replace gender data "M" and "F" with numerical inputs 0, and 1
dfoasl['M/F'] = dfoasl['M/F'].replace('F', 1)
dfoasl['M/F'] = dfoasl['M/F'].replace('M', 0)
dfoasl.head(10)
```

Out[75]:

	Subject ID	MRI ID	Group	Visit	MR Delay	M/F	Hand	Age	EDUC	SES	MMSE
0	OAS2_0001	OAS2_0001_MR1	Nondemented	1	0	0	R	87	14	2.0	27.0
1	OAS2_0001	OAS2_0001_MR2	Nondemented	2	457	0	R	88	14	2.0	30.0
2	OAS2_0002	OAS2_0002_MR1	Demented	1	0	0	R	75	12	NaN	23.0
3	OAS2_0002	OAS2_0002_MR2	Demented	2	560	0	R	76	12	NaN	28.0
4	OAS2_0002	OAS2_0002_MR3	Demented	3	1895	0	R	80	12	NaN	22.0
5	OAS2_0004	OAS2_0004_MR1	Nondemented	1	0	1	R	88	18	3.0	28.0
6	OAS2_0004	OAS2_0004_MR2	Nondemented	2	538	1	R	90	18	3.0	27.0
7	OAS2_0005	OAS2_0005_MR1	Nondemented	1	0	0	R	80	12	4.0	28.0
8	OAS2_0005	OAS2_0005_MR2	Nondemented	2	1010	0	R	83	12	4.0	29.0
9	OAS2_0005	OAS2_0005_MR3	Nondemented	3	1603	0	R	85	12	4.0	30.0

In [76]:
```
# Determine the count of various values for the label CDR
print (dfoasl.CDR[(dfoasl.CDR == 0.0)].count(),",", dfoasl.CDR[(dfoasl.CDR ==
print (dfoasl.CDR[(dfoasl.CDR == 1.0)].count(),",", dfoasl.CDR[(dfoasl.CDR ==
```

```
206 , 123
41 , 3
```

In [77]:
```
# Select longitudinal dataset columns that are similar to the dataset for cro
dfoasl2=dfoasl[['MRI ID','M/F','Hand','Age',
 'EDUC','SES','MMSE','CDR','eTIV','nWBV','ASF', 'MR Delay']]
dfoasl2.head(6)
```

Out[77]:

	MRI ID	M/F	Hand	Age	EDUC	SES	MMSE	CDR	eTIV	nWBV	ASF	MR Delay
0	OAS2_0001_MR1	0	R	87	14	2.0	27.0	0.0	1987	0.696	0.883	0
1	OAS2_0001_MR2	0	R	88	14	2.0	30.0	0.0	2004	0.681	0.876	457
2	OAS2_0002_MR1	0	R	75	12	NaN	23.0	0.5	1678	0.736	1.046	0
3	OAS2_0002_MR2	0	R	76	12	NaN	28.0	0.5	1738	0.713	1.010	560
4	OAS2_0002_MR3	0	R	80	12	NaN	22.0	0.5	1698	0.701	1.034	1895
5	OAS2_0004_MR1	1	R	88	18	3.0	28.0	0.0	1215	0.710	1.444	0

In [78]:
```
# Rename columns:MRI ID to ID, EDUC to Educ, MR Delay to Delay similar to cro
dfoasl2=dfoasl2.rename(columns={'EDUC':'Educ', 'MRI ID':'ID', 'MR Delay':'Del
dfoasl2.head(6)
```

Out[78]:

	ID	M/F	Hand	Age	Educ	SES	MMSE	CDR	eTIV	nWBV	ASF	Delay
0	OAS2_0001_MR1	0	R	87	14	2.0	27.0	0.0	1987	0.696	0.883	0
1	OAS2_0001_MR2	0	R	88	14	2.0	30.0	0.0	2004	0.681	0.876	457
2	OAS2_0002_MR1	0	R	75	12	NaN	23.0	0.5	1678	0.736	1.046	0
3	OAS2_0002_MR2	0	R	76	12	NaN	28.0	0.5	1738	0.713	1.010	560
4	OAS2_0002_MR3	0	R	80	12	NaN	22.0	0.5	1698	0.701	1.034	1895
5	OAS2_0004_MR1	1	R	88	18	3.0	28.0	0.0	1215	0.710	1.444	0

In [79]:
```
dfoasx.head(6)
```

Out[79]:

	ID	M/F	Hand	Age	Educ	SES	MMSE	CDR	eTIV	nWBV	ASF	Delay
0	OAS1_0001_MR1	1	R	74	2.0	3.0	29.0	0.0	1344	0.743	1.306	NaN
1	OAS1_0002_MR1	1	R	55	4.0	1.0	29.0	0.0	1147	0.810	1.531	NaN
2	OAS1_0003_MR1	1	R	73	4.0	3.0	27.0	0.5	1454	0.708	1.207	NaN
3	OAS1_0004_MR1	0	R	28	NaN	NaN	NaN	NaN	1588	0.803	1.105	NaN
4	OAS1_0005_MR1	0	R	18	NaN	NaN	NaN	NaN	1737	0.848	1.010	NaN
5	OAS1_0006_MR1	1	R	24	NaN	NaN	NaN	NaN	1131	0.862	1.551	NaN

Merge the cross-sectional and the Longitudinal MRI datasets

In [80]:
```
# Merge the cross-sectional and the longitudinal MRI datasets
dfoas_merge = dfoasx.append(dfoasl2, ignore_index=True)
dfoas_merge.shape
```

Out[80]: (809, 12)

In [81]:
```
dfoas_merge_with_NaN= dfoas_merge
```

The merged dataset dfoas_merge has 809 rows and 12 columns

Explore the merged dataset

In [82]:
```
dfoas_merge.describe()
```

Out[82]:

	M/F	Age	Educ	SES	MMSE	CDR	eTIV	nl
count	809.000000	809.000000	608.000000	570.00000	606.000000	608.000000	809.000000	809.00
mean	0.594561	63.186650	10.184211	2.47193	27.234323	0.288651	1484.782447	0.76
std	0.491280	23.117511	6.058388	1.12805	3.687980	0.377697	166.911689	0.05
min	0.000000	18.000000	1.000000	1.00000	4.000000	0.000000	1106.000000	0.64
25%	0.000000	49.000000	4.000000	2.00000	26.000000	0.000000	1361.000000	0.71
50%	1.000000	72.000000	12.000000	2.00000	29.000000	0.000000	1475.000000	0.75
75%	1.000000	80.000000	16.000000	3.00000	30.000000	0.500000	1583.000000	0.81
max	1.000000	98.000000	23.000000	5.00000	30.000000	2.000000	2004.000000	0.89

In [83]:
```
# Replace NaN values for the Delay column to dataset average of 555 to facili
dfoas_merge['Delay'] = dfoas_merge['Delay'].replace(np.NaN, 555)
```

In [84]:
```
dfoas_merge.head(10)
```

Out[84]:

	ID	M/F	Hand	Age	Educ	SES	MMSE	CDR	eTIV	nWBV	ASF	Delay
0	OAS1_0001_MR1	1	R	74	2.0	3.0	29.0	0.0	1344	0.743	1.306	555.0
1	OAS1_0002_MR1	1	R	55	4.0	1.0	29.0	0.0	1147	0.810	1.531	555.0
2	OAS1_0003_MR1	1	R	73	4.0	3.0	27.0	0.5	1454	0.708	1.207	555.0
3	OAS1_0004_MR1	0	R	28	NaN	NaN	NaN	NaN	1588	0.803	1.105	555.0
4	OAS1_0005_MR1	0	R	18	NaN	NaN	NaN	NaN	1737	0.848	1.010	555.0
5	OAS1_0006_MR1	1	R	24	NaN	NaN	NaN	NaN	1131	0.862	1.551	555.0
6	OAS1_0007_MR1	0	R	21	NaN	NaN	NaN	NaN	1516	0.830	1.157	555.0
7	OAS1_0009_MR1	1	R	20	NaN	NaN	NaN	NaN	1505	0.843	1.166	555.0
8	OAS1_0010_MR1	0	R	74	5.0	2.0	30.0	0.0	1636	0.689	1.073	555.0
9	OAS1_0011_MR1	1	R	52	3.0	2.0	30.0	0.0	1321	0.827	1.329	555.0

In [85]:
```
1 dfoas_merge.dtypes
```

Out[85]:
```
ID        object
M/F        int64
Hand      object
Age        int64
Educ     float64
SES      float64
MMSE     float64
CDR      float64
eTIV       int64
nWBV     float64
ASF      float64
Delay    float64
dtype: object
```

Drop rows in the merged dataset with NaN in any column. We see that there are 809-570=239 rows with NaN values in any column.

In [86]:
```
1 # Drop rows with NaN
2 dfoas_merge=dfoas_merge.dropna(how='any')
```

In [87]:
```
1 dfoas_merge.shape
```

Out[87]:
```
(570, 12)
```

In [88]:
```
1 dfoas_merge.dtypes
```

Out[88]:
```
ID        object
M/F        int64
Hand      object
Age        int64
Educ     float64
SES      float64
MMSE     float64
CDR      float64
eTIV       int64
nWBV     float64
ASF      float64
Delay    float64
dtype: object
```

In [89]:
```
1 dfoas_merge.head(6)
```

Out[89]:

	ID	M/F	Hand	Age	Educ	SES	MMSE	CDR	eTIV	nWBV	ASF	Delay
0	OAS1_0001_MR1	1	R	74	2.0	3.0	29.0	0.0	1344	0.743	1.306	555.0
1	OAS1_0002_MR1	1	R	55	4.0	1.0	29.0	0.0	1147	0.810	1.531	555.0
2	OAS1_0003_MR1	1	R	73	4.0	3.0	27.0	0.5	1454	0.708	1.207	555.0
8	OAS1_0010_MR1	0	R	74	5.0	2.0	30.0	0.0	1636	0.689	1.073	555.0
9	OAS1_0011_MR1	1	R	52	3.0	2.0	30.0	0.0	1321	0.827	1.329	555.0
11	OAS1_0013_MR1	1	R	81	5.0	2.0	30.0	0.0	1664	0.679	1.055	555.0

In [90]:
```
1 dfoas_merge.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 570 entries, 0 to 808
Data columns (total 12 columns):
ID       570 non-null object
M/F      570 non-null int64
Hand     570 non-null object
Age      570 non-null int64
Educ     570 non-null float64
SES      570 non-null float64
MMSE     570 non-null float64
CDR      570 non-null float64
eTIV     570 non-null int64
nWBV     570 non-null float64
ASF      570 non-null float64
Delay    570 non-null float64
dtypes: float64(7), int64(3), object(2)
memory usage: 57.9+ KB
```

In [91]:
```
1 print (dfoas_merge.CDR[(dfoas_merge.CDR == 0.0)].count(),",", dfoas_merge.CDR
2 print (dfoas_merge.CDR[(dfoas_merge.CDR == 1.0)].count(),",", dfoas_merge.CDR
```

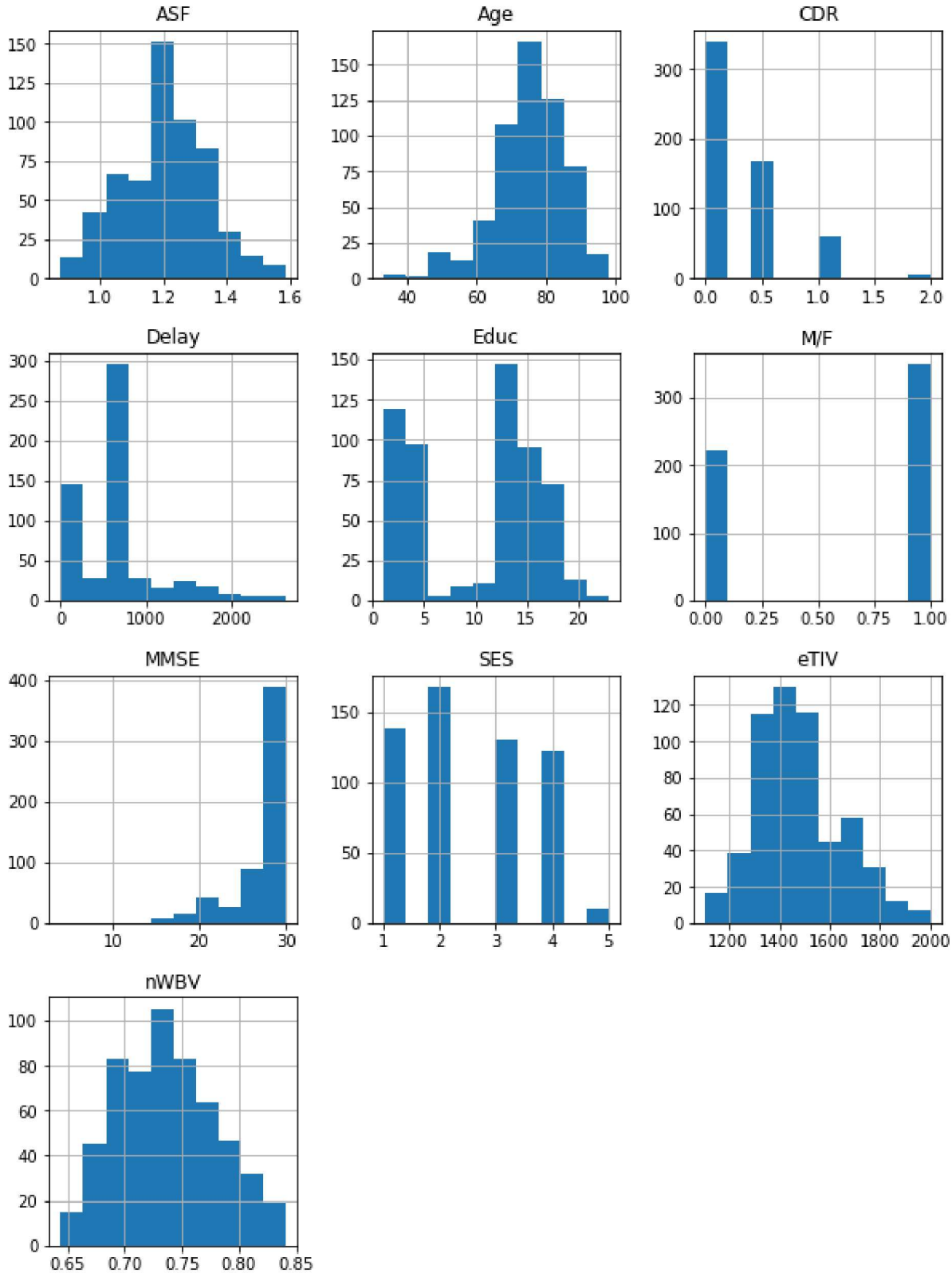
```
339 , 167
59 , 5
```

We see from above exploratory results that the combined cross-sectional and longitudinal datasets have 339 records for CDR=0.0, 167 records for CDR=0.5, 59 records for CDR=1.0, and 5 records for CDR=2. Thus the labels are not balanced. thus the metric reported will be Confusion Matrix (sensitivity, specificity, recall, and support) from sklearn. Classification accuracy will also be included as a reference value.

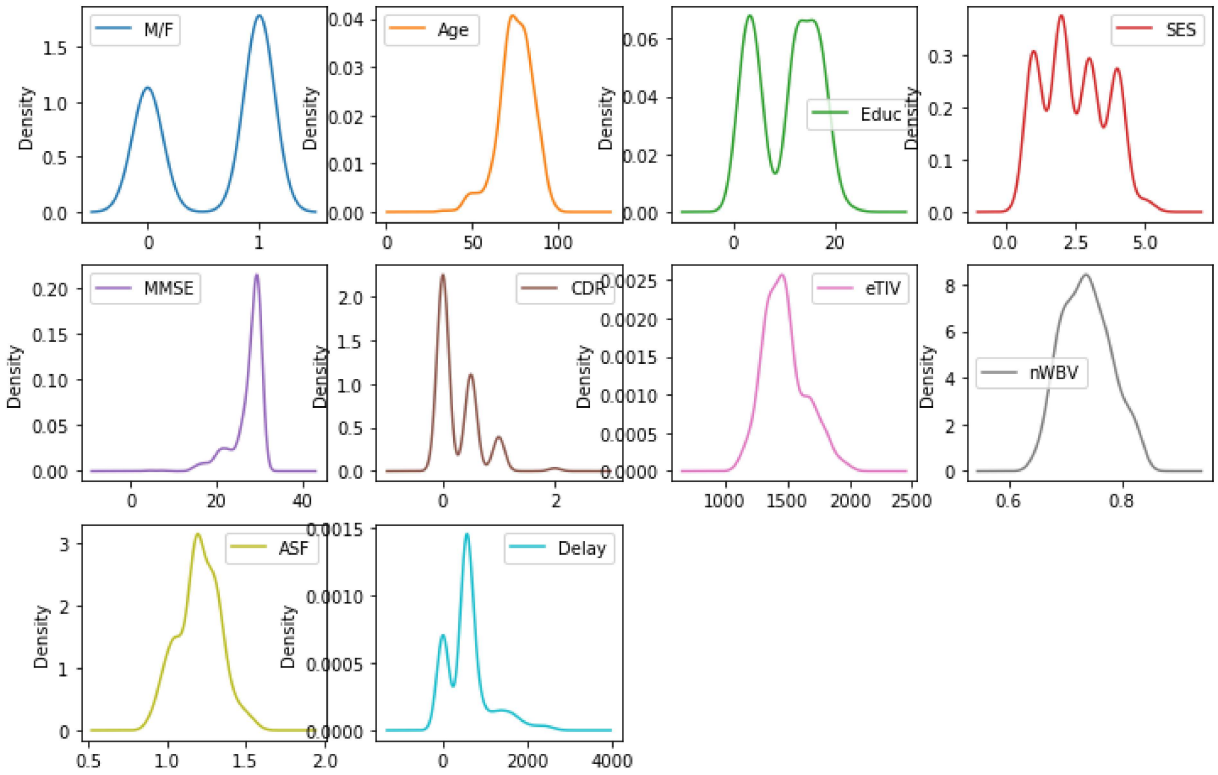
Exploratory Visualization

A visualization has been provided that summarizes or extracts a relevant characteristic or feature about the dataset or input data with thorough discussion. Visual cues are clearly defined.

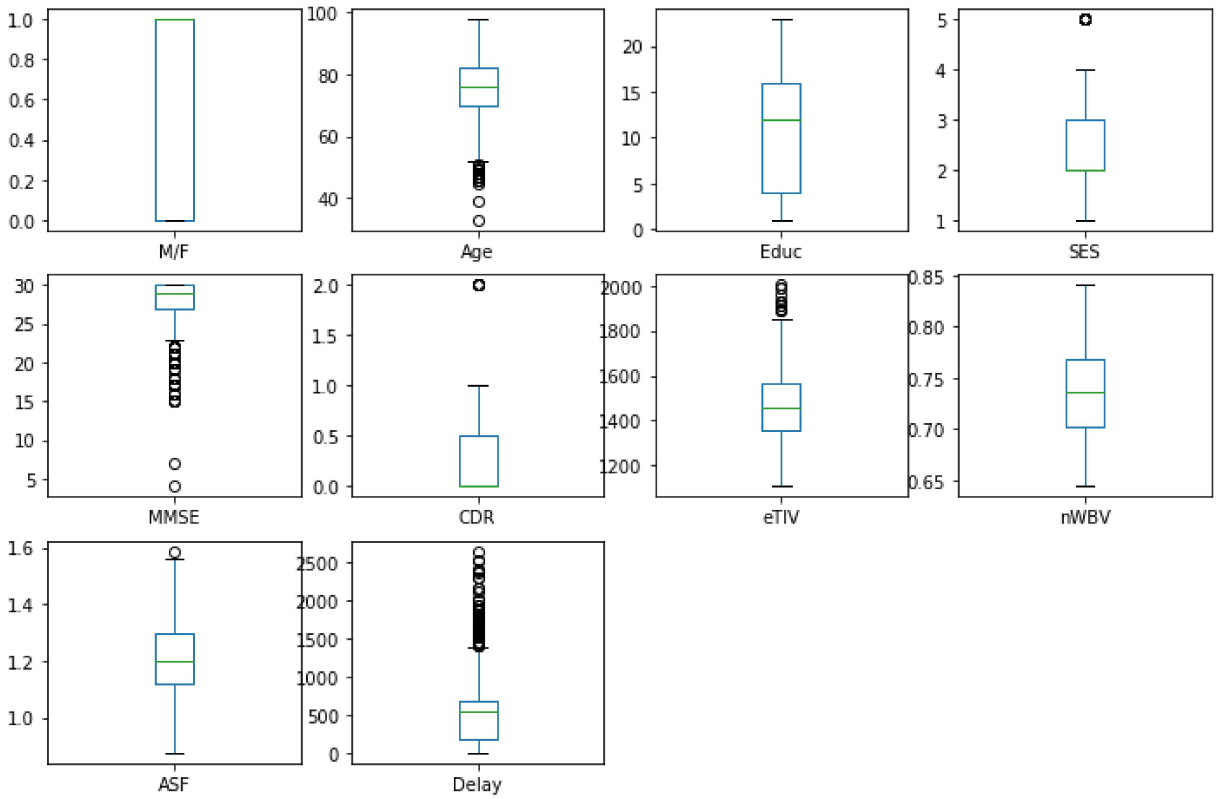
In [92]:
```
from matplotlib import pyplot
dfoas_merge.hist()
fig_size = pyplot.rcParams["figure.figsize"]
fig_size[0] = 10
fig_size[1] = 14
pyplot.rcParams["figure.figsize"] = fig_size
pyplot.show()
```



In [93]:
```
# http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.plot
# Univariate Density Plots
from matplotlib import pyplot
dfoas_merge.plot(kind='density', subplots=True, layout=(3, 4), figsize=(12, 8
pyplot.show();
```

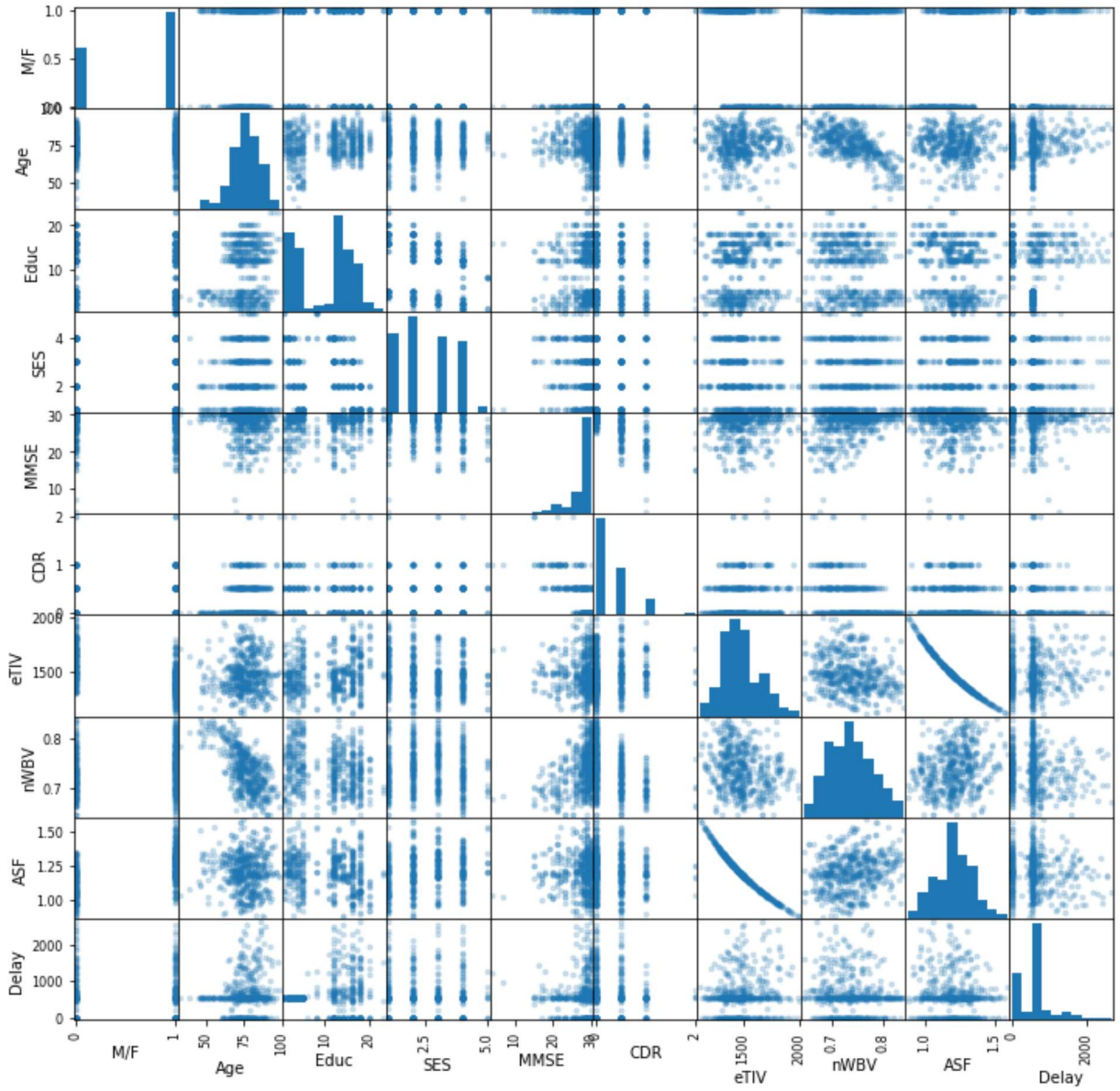


In [94]:
```
dfoas_merge.plot(kind='box', subplots=True, layout=(3, 4), figsize=(12, 8), s
pyplot.show();
```



From the histogram. density plots, and box plots for the feature variables, we note the following: Age, eTIV, nWBV, and ASF have approximately normal distribution. The feature variables M/F, Educ, and SES have bi-modal or multimodal distribution.

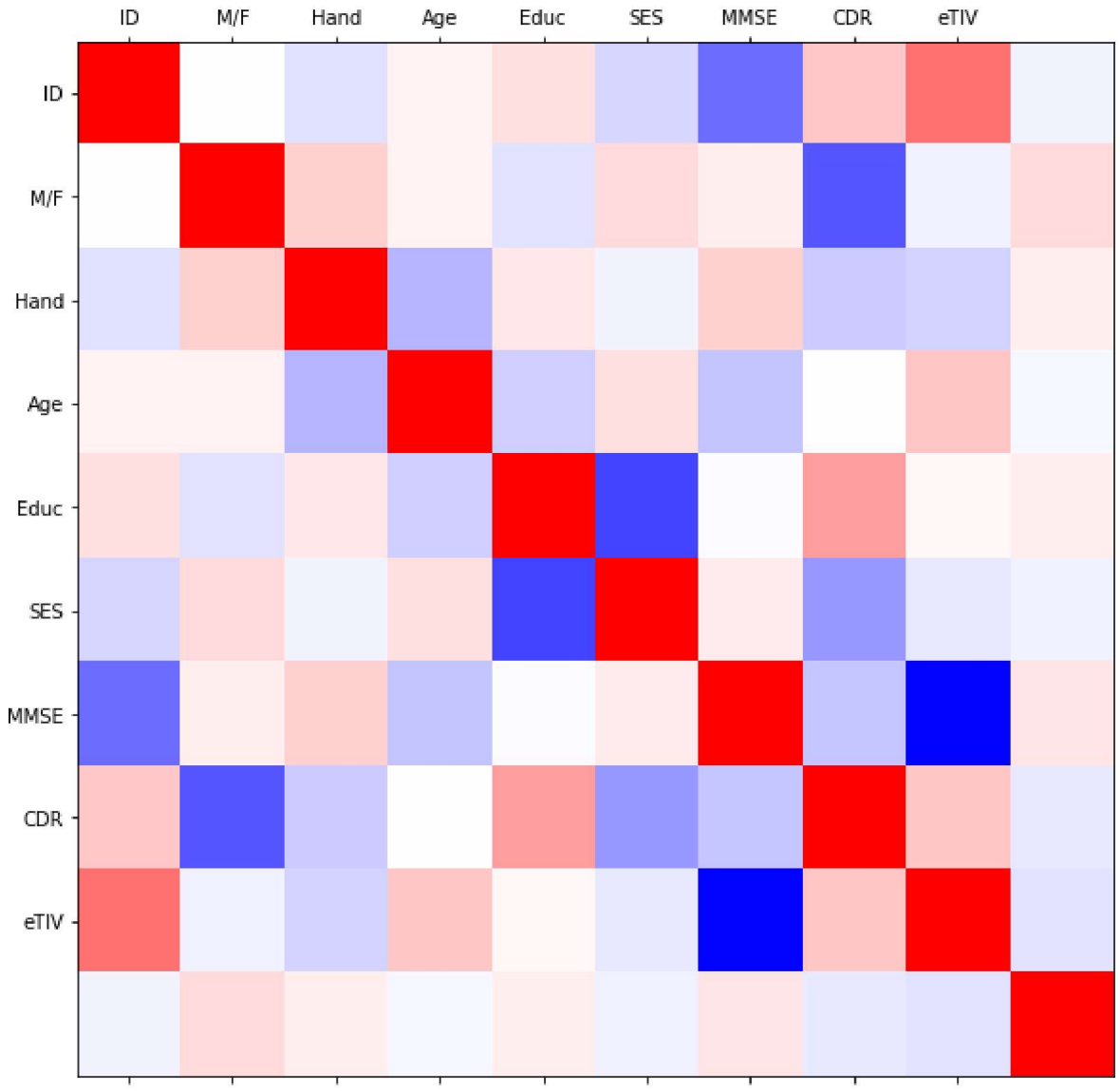
In [95]:
```
# Scatterplot Matrix
from matplotlib import pyplot
from pandas import read_csv
from pandas.tools.plotting import scatter_matrix
scatter_matrix(dfoas_merge, alpha=0.2, figsize=(12, 12))
pyplot.show();
```



Feature Selection Methodology

In [96]:
```
# Correlation Matrix Plot
from matplotlib import pyplot
from matplotlib import cm
import numpy as np
correlations = dfoas_merge.corr()
# plot correlation matrix
figoasx = pyplot.figure()
cm = pyplot.cm.bwr
fig = pyplot.figure()
ax = fig.add_subplot(111);
cax = ax.matshow(correlations, cmap=cm, vmin=-1, vmax=1);
ticks = np.arange(0,9,1);
names= list(dfoas_merge.columns.values)
ax.set_xticks(ticks)
ax.set_yticks(ticks)
ax.set_xticklabels(names)
ax.set_yticklabels(names)
fig.set_figheight(10)
fig.set_figwidth(10)
pyplot.show();
```

```
<matplotlib.figure.Figure at 0x207aa0c64a8>
```



Methodology - Select Training and Test/Validation Sets

In [97]:
```
# Split the dataset into a training set and a validation set
#Create an array from the dataset column values
array = dfoas_merge.values
#X = array[:,[1, 3, 4, 5, 6, 8, 9, 10]] Backup/extra
# Array indices 1 through 11 refer to the following features:
# [M/F  Hand  Age  Educ  SES  MMSE  CDR  eTIV  nWBV  ASF  Delay]
# Exclude Hand (all records are for right handed people), and Delay(more than

X = array[:,[1, 3, 4, 5, 6, 8, 9, 10]]
#X = array[:,[3, 6, 8, 9]] #Feature Extraction with RFE
#Y_float64 = array[:,12] # all rows and CDR column used for AUC ROC calculati
Y = array[:,7] # all rows and CDR_Class column
Y_float64 = array[:,7] # all rows and CDR column used for AUC ROC calculation
Y=Y.astype(np.str)
validation_size = 0.20
seed = 123456
X_train, X_validation, Y_train, Y_validation = train_test_split(X, Y,
test_size=validation_size, random_state=seed)
print (X)
print (Y)
```

```
[[1 74 2.0 ..., 1344 0.743 1.306]
 [1 55 4.0 ..., 1147 0.81 1.531]
 [1 73 4.0 ..., 1454 0.708 1.207]
 ...,
 [1 61 13.0 ..., 1319 0.8009999999999999 1.331]
 [1 63 13.0 ..., 1327 0.7959999999999999 1.3230000000000002]
 [1 65 13.0 ..., 1333 0.8009999999999999 1.317]]
['0.0' '0.0' '0.5' '0.0' '0.0' '0.0' '0.5' '0.0' '0.0' '0.0' '0.5' '0.5'
 '0.5' '0.0' '1.0' '0.0' '1.0' '0.0' '0.0' '0.0' '1.0' '0.5' '0.5' '0.0'
 '1.0' '1.0' '1.0' '0.0' '0.0' '0.0' '0.0' '0.5' '1.0' '0.0' '0.0' '0.0'
 '0.0' '0.0' '1.0' '0.0' '0.0' '0.5' '0.0' '0.0' '0.0' '0.5' '0.0' '0.0'
 '0.0' '0.0' '0.0' '0.0' '0.0' '0.5' '0.0' '0.5' '1.0' '0.5' '0.0' '0.0'
 '1.0' '0.0' '1.0' '0.0' '0.0' '0.5' '0.5' '0.0' '0.0' '0.5' '0.5' '0.5'
 '0.0' '0.0' '0.0' '0.0' '0.0' '0.5' '0.0' '0.0' '1.0' '0.0' '0.0' '0.0'
 '0.0' '0.0' '0.0' '0.0' '0.0' '0.0' '0.5' '0.0' '0.0' '0.0' '0.5' '0.0'
 '0.5' '0.0' '0.0' '0.5' '0.0' '0.0' '0.5' '0.0' '0.0' '0.5' '0.0' '0.5'
 '0.0' '0.5' '0.0' '0.0' '0.0' '0.0' '0.0' '0.0' '0.5' '0.0' '0.5' '1.0'
 '1.0' '0.0' '0.0' '0.5' '0.5' '0.0' '1.0' '0.0' '0.0' '0.0' '0.5' '0.5'
 '0.5' '0.0' '0.5' '1.0' '0.0' '0.0' '0.5' '0.0' '0.5' '0.0' '0.0' '0.5'
 '0.5' '2.0' '0.0' '0.5' '1.0' '0.0' '0.0' '0.0' '0.0' '0.5' '0.0' '0.0'
 '0.5' '0.0' '0.0' '0.0' '0.0' '0.0' '0.0' '2.0' '0.5' '0.0' '0.0' '0.0'
 '0.0' '0.0' '0.0' '0.0' '0.0' '0.0' '0.0' '0.0' '0.0' '1.0' '0.5' '0.0'
 '0.5' '0.0' '1.0' '0.5' '0.0' '0.5' '0.0' '0.5' '0.0' '1.0' '0.5' '0.5'
 '0.0' '0.0' '1.0' '1.0' '0.0' '0.0' '1.0' '0.5' '0.0' '0.0' '0.5' '0.5'
 '0.0' '0.0' '0.0' '0.5' '0.0' '0.5' '1.0' '0.5' '0.5' '0.0' '0.0' '0.0'
 '0.0' '0.0' '0.0' '0.0' '0.0' '0.5' '0.0' '0.0' '0.0' '0.5' '0.5' '0.5'
 '0.5' '0.0' '0.0' '0.0' '0.0' '0.0' '0.0' '0.5' '1.0' '0.5' '0.5' '0.0'
 '0.5' '0.0' '0.0' '0.0' '0.0' '0.5' '0.0' '0.5' '0.5' '0.5' '0.5' '0.0'
 '0.0' '0.5' '0.5' '0.5' '0.5' '0.0' '0.0' '0.0' '0.0' '0.5' '1.0' '0.0'
 '0.0' '0.0' '0.0' '0.0' '0.0' '0.5' '0.5' '0.5' '0.0' '0.0' '0.0' '0.0'
 '0.0' '0.0' '0.0' '0.0' '0.0' '0.0' '0.5' '0.5' '0.5' '0.5' '0.5' '0.5'
 '0.5' '0.5' '0.5' '0.0' '0.0' '0.5' '0.0' '0.0' '0.5' '0.5' '1.0' '1.0'
 '1.0' '0.0' '0.0' '0.5' '1.0' '0.0' '0.0' '1.0' '1.0' '1.0' '1.0' '1.0'
 '0.0' '0.0' '0.0' '0.5' '1.0' '0.0' '0.0' '0.0' '0.0' '0.0' '0.0' '0.0'
 '0.0' '0.5' '0.0' '0.0' '0.0' '0.0' '0.0' '0.0' '0.0' '0.5' '0.5' '0.5'
 '0.5' '0.5' '0.0' '0.0' '0.0' '0.0' '0.0' '0.0' '1.0' '0.5' '0.5' '1.0'
```

```
'0.5' '0.0' '0.0' '0.0' '0.0' '0.0' '0.0' '0.0' '0.0' '0.0' '0.0' '0.0'
'0.0' '0.0' '1.0' '1.0' '0.0' '0.0' '0.0' '0.0' '0.0' '0.5' '0.5' '0.0'
'0.0' '0.0' '0.0' '0.0' '0.0' '0.0' '0.0' '0.5' '1.0' '1.0' '0.5' '0.5'
'0.5' '0.5' '0.5' '0.0' '0.0' '0.0' '0.0' '1.0' '2.0' '1.0' '1.0' '0.5'
'1.0' '0.0' '0.0' '0.0' '0.0' '0.0' '0.0' '0.5' '0.0' '0.0' '0.0' '0.0'
'0.0' '0.0' '0.0' '0.0' '0.0' '0.5' '0.5' '0.0' '0.0' '0.0' '0.0' '0.0'
'0.0' '0.5' '0.5' '0.5' '0.0' '0.5' '0.5' '0.5' '1.0' '0.0' '0.0' '1.0'
'1.0' '0.5' '0.5' '0.0' '0.0' '0.5' '0.5' '0.5' '0.5' '0.5' '0.5' '0.5'
'0.5' '0.0' '0.0' '0.0' '0.0' '0.0' '0.5' '0.0' '0.0' '0.0' '1.0' '2.0'
'0.0' '0.0' '0.0' '0.0' '0.5' '0.5' '0.0' '0.0' '0.0' '0.0' '0.5' '0.5'
'0.5' '0.5' '0.0' '0.0' '0.0' '0.0' '0.0' '0.5' '0.0' '0.0' '0.5' '0.5'
'0.5' '0.0' '0.0' '0.5' '0.5' '0.0' '0.0' '0.5' '0.5' '0.5' '0.5' '0.5'
'0.0' '0.0' '0.0' '0.0' '0.0' '0.0' '0.0' '0.0' '0.5' '0.0' '0.5' '1.0'
'1.0' '0.0' '0.0' '0.0' '0.0' '0.0' '0.0' '0.5' '1.0' '0.0' '0.0' '0.0'
'0.0' '0.0' '0.0' '0.0' '1.0' '1.0' '0.0' '0.0' '0.5' '0.5' '0.0' '0.0'
'0.0' '0.5' '0.5' '1.0' '2.0' '1.0' '1.0' '0.0' '0.0' '0.0' '0.0' '0.0'
'0.5' '0.5' '0.0' '0.0' '0.0' '0.5' '0.5' '0.5' '0.0' '0.0' '0.5' '0.0'
'0.0' '0.0' '0.0' '0.0' '0.5' '0.5' '0.0' '0.0' '0.0' '0.0' '0.5' '1.0'
'0.5' '0.5' '0.5' '0.0' '0.0' '0.0']
```

Algorithms and Techniques

(Reference 24)

Spot Check Algorithms

We will evaluate eight different algorithms:

- Logistic Regression (LR).
- k-Nearest Neighbors (KNN).
- Classification and Regression Trees (CART).
- Gaussian Naive Bayes (NB).
- Support Vector Machines (SVM).
- Gradient Boosting Machine (GBM).
- Random Forest (RF).

This list is a good mixture of simple linear (LR), and nonlinear (KNN, CART, NB and SVM) algorithms. We reset the random number seed before each run to ensure that the evaluation of each algorithm is performed using exactly the same data splits. It ensures the results are directly comparable. Let's build and evaluate our six models:

In [98]:
```
# Spot-Check Algorithms
models = []
models.append(('LR', LogisticRegression()))
#models.append(('LDA', LinearDiscriminantAnalysis()))
models.append(('KNN', KNeighborsClassifier()))
models.append(('CART', DecisionTreeClassifier()))
models.append(('NB', GaussianNB()))
models.append(('SVM', SVC()))
models.append(('GBM', GradientBoostingClassifier()))
models.append(('RFC', RandomForestClassifier()))
# evaluate each model in turn
results = []
names = []
print('(', 'Model, ', 'Cross-Validation Accuracy: Mean, Stdev',')')
for name, model in models:
    kfold = KFold(n_splits=10, random_state=seed)
    cv_results = cross_val_score(model, X_train, Y_train, cv=kfold, scoring='
    results.append(cv_results)
    names.append(name)
    msg = (name, format(cv_results.mean(), '.2f'), format(cv_results.std(), '
    print(msg)
```

```
( Model,  Cross-Validation Accuracy: Mean, Stdev )
('LR', '0.74', '0.07')
('KNN', '0.60', '0.05')
('CART', '0.74', '0.06')
('NB', '0.76', '0.08')
('SVM', '0.59', '0.07')
('GBM', '0.81', '0.08')
('RFC', '0.78', '0.08')
```

In [99]:
```
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(max_depth=4)
classifier.fit(X_train, Y_train)
X_test = X_validation
Y_test = Y_validation
prediction = classifier.predict(X_test)
print (classifier.score(X_train, Y_train))
print (classifier.score(X_test, Y_test))
```

```
0.80701754386
0.719298245614
```

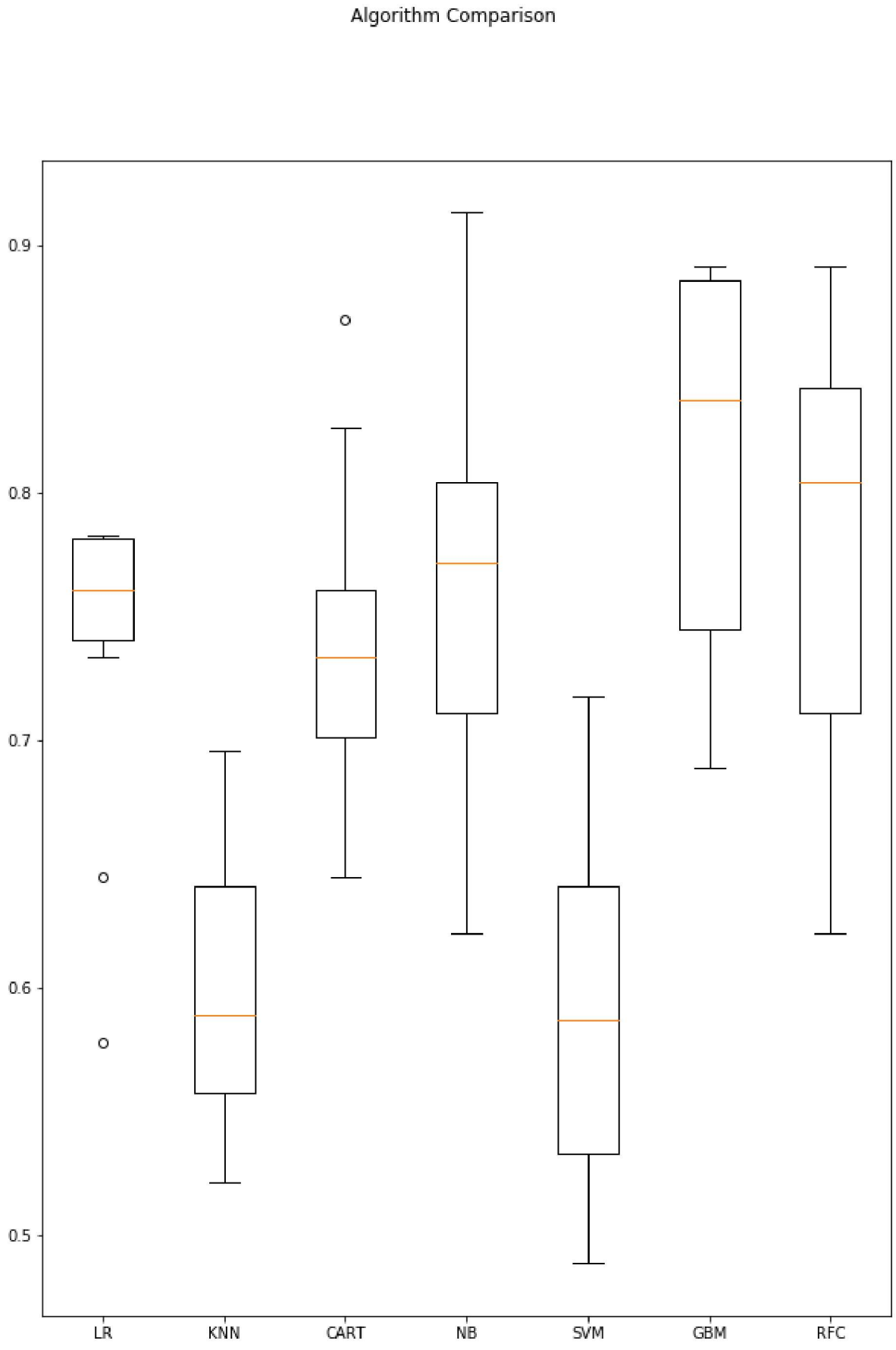
In [100]:
```
from sklearn.ensemble import GradientBoostingClassifier
classifier = GradientBoostingClassifier(max_depth=3)
classifier.fit(X_train, Y_train)
X_test = X_validation
Y_test = Y_validation
prediction = classifier.predict(X_test)
print (classifier.score(X_train, Y_train))
print (classifier.score(X_test, Y_test))
```

```
0.991228070175
0.745614035088
```

(from Brownlee Book) Using a 10-fold cross-validation to estimate accuracy. This splits the dataset

into 10 parts. Train on 9 parts and test on 1 part. Repeat for all combinations of train-test splits. Using the metric of accuracy to evaluate models. This is a ratio of the number of correctly predicted instances divided by the total number of instances in the dataset multiplied by 100 to give a percentage (e.g. 95% accurate). Using the scoring variable when we run build and evaluate each model next.

In [101]:
```
# Compare Algorithms
fig = pyplot.figure()
fig.suptitle('Algorithm Comparison')
ax = fig.add_subplot(111)
pyplot.boxplot(results)
ax.set_xticklabels(names)
pyplot.show();
```



We see fromthe above plots see that GBM and RFC have higher accuracies than other algorithms, followed by LR, NB, and CART. These model evaluation results compare the spread and the mean accuracy of each model. There is a population of accuracy measures for each algorithm because

each algorithm was evaluated 10 times (10 fold cross-validation).

In [102]:
```
# Make predictions on validation dataset using the RandomForestClassifier

from sklearn.ensemble import GradientBoostingClassifier
GBM=GradientBoostingClassifier()
GBM.fit(X_train, Y_train)
predictions = GBM.predict(X_validation)
print("*Accuracy score: ", accuracy_score(Y_validation, predictions), "\n")
print("*Confusion Matrix: ")
print(confusion_matrix(Y_validation, predictions))
print("*Classification Report: ")
print(classification_report(Y_validation, predictions))
```

```
*Accuracy score:  0.745614035088

*Confusion Matrix:
[[58 10  0  0]
 [11 20  4  0]
 [ 1  2  7  0]
 [ 0  0  1  0]]
*Classification Report:
             precision    recall  f1-score   support

        0.0       0.83      0.85      0.84        68
        0.5       0.62      0.57      0.60        35
        1.0       0.58      0.70      0.64        10
        2.0       0.00      0.00      0.00         1

avg / total       0.74      0.75      0.74       114


C:\Users\MD\Anaconda3\envs\dlnd\lib\site-packages\sklearn\metrics\classificatio
n.py:1135: UndefinedMetricWarning: Precision and F-score are ill-defined and be
ing set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
```

In [103]:
```
# Make predictions on validation dataset using GradientBoostingClassifier
from sklearn.ensemble import GradientBoostingClassifier
GBM = GradientBoostingClassifier()
GBM.fit(X_train, Y_train)
predictions = GBM.predict(X_validation)
print("*Accuracy score for GBM: ", accuracy_score(Y_validation, predictions),
print("*Confusion Matrix for GBM: ")
print(confusion_matrix(Y_validation, predictions))
print("*Classification Report for GBM: ")
print(classification_report(Y_validation, predictions))
```

```
*Accuracy score for GBM:  0.745614035088

*Confusion Matrix for GBM:
[[58 10  0  0]
 [11 20  4  0]
 [ 1  2  7  0]
 [ 0  0  1  0]]
*Classification Report for GBM:
             precision    recall  f1-score   support

        0.0       0.83      0.85      0.84        68
        0.5       0.62      0.57      0.60        35
        1.0       0.58      0.70      0.64        10
        2.0       0.00      0.00      0.00         1

avg / total       0.74      0.75      0.74       114


C:\Users\MD\Anaconda3\envs\dlnd\lib\site-packages\sklearn\metrics\classificatio
n.py:1135: UndefinedMetricWarning: Precision and F-score are ill-defined and be
ing set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
```

Accuracy score is (number of true positives + number of true negatives)/total number of data points in the dataset.

By definition a confusion matrix C is such that element C_{i, j} is equal to the number of observations known to be in group i but predicted to be in group j.

Thus in binary classification, the count of true negatives is C*{0,0}, false negatives is C*{1,0}, true positives is C*{1,1} and false positives is C*{0,1}.

The f1-score the harmonic mean of precision and recall. The scores corresponding to every class will tell you the accuracy of the classifier in classifying the data points in that particular class compared to all other classes.

The support is the number of samples of the true response that lie in that class. The total support should equal the number of datapoints in the test set.

Because there are very few records with CDR=2, the validation set picked up only one record (see support=1 value for the CDR=2 case in the Classification Report), and there was no predicted sample for this label . Hence the metrics(precision, recall, F-Score are all zero for CDR=2 label.

In [104]:
```
# Create a PIPELINE that extracts features from the data then creates a model
# Uses GradientBoostingClassifier
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.pipeline import Pipeline
from sklearn.pipeline import FeatureUnion
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.decomposition import PCA
from sklearn.feature_selection import SelectKBest
# X, Y as defined/calculated previously
# create feature union
features = []
features.append(('pca', PCA(n_components=3)))
features.append(('select_best', SelectKBest(k=4)))
feature_union = FeatureUnion(features)
# create pipeline
estimators = []
estimators.append(('feature_union', feature_union))
estimators.append(('GBC', GradientBoostingClassifier()))
model = Pipeline(estimators)
# evaluate pipeline
kfold = KFold(n_splits=10, random_state=899)
results = cross_val_score(model, X, Y, cv=kfold)
print(results.mean())
```

```
0.773684210526
```

Recursive Feature Elimination

(Reference 24) The Recursive Feature Elimination (or RFE) works by recursively removing attributes and building a model on those attributes that remain. It uses the model accuracy to identify which http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html#sklearn.feature_selec (http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html#sklearn.feature_selec

attributes (and combination of attributes) contribute the most to predicting the target attribute. You can learn more about the RFE class in the scikit-learn documentation. The example below uses RFE with the logistic regression algorithm to select the top 3 features. The choice of algorithm does not matter too much as long as it is skillful and consistent. We see columns Age, SES, MMSE are the top three featutes

In [105]:
```
# Feature Extraction with RFE
from sklearn.feature_selection import RFE
from sklearn.linear_model import LogisticRegression
# feature extraction
model = DecisionTreeClassifier()
rfe = RFE(model, 3)
fit = rfe.fit(X, Y)
print("Num Features: %d" % fit.n_features_)
print("Selected Features: %s" % fit.support_)
print("Feature Ranking: %s" % fit.ranking_)
```

```
Num Features: 3
Selected Features: [False False False False  True False  True  True]
Feature Ranking: [5 2 4 6 1 3 1 1]
```

Principal Component Analysis

Principal Component Analysis (or PCA) uses linear algebra to transform the dataset into a compressed form. Generally this is called a data reduction technique. A property of PCA is that we can choose the number of dimensions or principal components in the transformed result. In the example below, we use PCA and select 3 principal components.

In [106]:
```
# Feature Extraction with PCA
from sklearn.decomposition import PCA
# feature extraction
pca = PCA(n_components=3)
fit = pca.fit(X)
# summarize components
print("Explained Variance: %s" % fit.explained_variance_ratio_)
print(fit.components_)
```

```
Explained Variance: [ 0.99488306  0.00346391  0.00118106]
[[ -1.63525903e-03   3.33704766e-03   6.28458741e-03  -1.54868537e-03
   -3.21063218e-04   9.99971792e-01  -5.60520444e-05  -7.77833362e-04]
 [ -1.52943466e-03  -9.87638014e-01  -1.51568583e-01  -4.47623170e-03
    3.93677660e-02   4.25175111e-03   2.79895794e-03  -9.39644552e-05]
 [  1.51149687e-03   1.45435084e-01  -9.80433825e-01   5.54117703e-02
   -1.20384981e-01   5.72592463e-03  -1.70377294e-04  -2.07631309e-04]]
```

Appendix 2: Append CDR=2 data from the dataset to increase CDR=2 count

We saw previously that the number of records in the combined cross-sectional and longitudinal MRI datasets is only 5. Therefore, in the 20% of the data set aside for validation, the probability of having records with label CDR=2 is quite small. Correspondingly, in the metrics for the Gradient Boosting (GBM) and Random Forest (RFC) classsifiers, the accuracy, precision, recall, and f-score are very low for the label CDR=2. In this sensitivity study we append to the combined dataset, the records for CDR=2 two more times to artificially increase CDR=2 record count to 15 from 5. As we see below, the classification metrics for this multi-label sensitivity case have improved significantly.

In [107]:
```
1 dfoas_merge_CDR2 = dfoas_merge[(dfoas_merge.CDR == 2)]
2 dfoas_merge_CDR22=dfoas_merge.append(dfoas_merge_CDR2, ignore_index=True)
3 dfoas_merge_CDR23=dfoas_merge_CDR22.append(dfoas_merge_CDR2, ignore_index=Tru
```

In [108]:
```
1 dfoas_merge_CDR23.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 580 entries, 0 to 579
Data columns (total 12 columns):
ID       580 non-null object
M/F      580 non-null int64
Hand     580 non-null object
Age      580 non-null int64
Educ     580 non-null float64
SES      580 non-null float64
MMSE     580 non-null float64
CDR      580 non-null float64
eTIV     580 non-null int64
nWBV     580 non-null float64
ASF      580 non-null float64
Delay    580 non-null float64
dtypes: float64(7), int64(3), object(2)
memory usage: 54.5+ KB
```

In [109]:
```
# Split the dataset into a training set and a validation set
#Create an array from the dataset column values
array = dfoas_merge_CDR23.values
#X = array[:,[1, 3, 4, 5, 6, 8, 9, 10]] Backup/extra
# Array indices 1 through 11 refer to the following features:
# [M/F  Hand  Age  Educ  SES  MMSE  CDR  eTIV  nWBV  ASF  Delay]
# Exclude Hand (all records are for right handed people), and Delay(more than

X = array[:,[1, 3, 4, 5, 6, 8, 9, 10]]
#X = array[:,[3, 6, 8, 9]] #Feature Extraction with RFE
#Y_float64 = array[:,12] # all rows and CDR column used for AUC ROC calculati
Y = array[:,7] # all rows and CDR_Class column
Y_float64 = array[:,7] # all rows and CDR column used for AUC ROC calculation
Y=Y.astype(np.str)
validation_size = 0.20
seed = 123456
X_train, X_validation, Y_train, Y_validation = train_test_split(X, Y,
test_size=validation_size, random_state=seed)
print (X)
print (Y)
```

```
[[1 74 2.0 ..., 1344 0.743 1.306]
 [1 55 4.0 ..., 1147 0.81 1.531]
 [1 73 4.0 ..., 1454 0.708 1.207]
 ...,
 [1 98 17.0 ..., 1503 0.66 1.168]
 [1 78 14.0 ..., 1401 0.7 1.253]
 [0 79 20.0 ..., 1710 0.76 1.026]]
['0.0' '0.0' '0.5' '0.0' '0.0' '0.0' '0.5' '0.0' '0.0' '0.0' '0.5' '0.5'
 '0.5' '0.0' '1.0' '0.0' '1.0' '0.0' '0.0' '0.0' '1.0' '0.5' '0.5' '0.0'
 '1.0' '1.0' '1.0' '0.0' '0.0' '0.0' '0.0' '0.5' '1.0' '0.0' '0.0' '0.0'
 '0.0' '0.0' '1.0' '0.0' '0.0' '0.5' '0.0' '0.0' '0.0' '0.5' '0.0' '0.0'
 '0.0' '0.0' '0.0' '0.0' '0.0' '0.5' '0.0' '0.5' '1.0' '0.5' '0.0' '0.0'
 '1.0' '0.0' '1.0' '0.0' '0.0' '0.5' '0.5' '0.0' '0.0' '0.5' '0.5' '0.5'
 '0.0' '0.0' '0.0' '0.0' '0.0' '0.5' '0.0' '0.0' '1.0' '0.0' '0.0' '0.0'
 '0.0' '0.0' '0.0' '0.0' '0.0' '0.0' '0.5' '0.0' '0.0' '0.0' '0.5' '0.0'
 '0.5' '0.0' '0.0' '0.5' '0.0' '0.0' '0.5' '0.0' '0.0' '0.5' '0.0' '0.5'
 '0.0' '0.5' '0.0' '0.0' '0.0' '0.0' '0.0' '0.0' '0.5' '0.0' '0.5' '1.0'
 '1.0' '0.0' '0.0' '0.5' '0.5' '0.0' '1.0' '0.0' '0.0' '0.0' '0.5' '0.5'
 '0.5' '0.0' '0.5' '1.0' '0.0' '0.0' '0.5' '0.0' '0.5' '0.0' '0.0' '0.5'
 '0.5' '2.0' '0.0' '0.5' '1.0' '0.0' '0.0' '0.0' '0.0' '0.5' '0.0' '0.0'
 '0.5' '0.0' '0.0' '0.0' '0.0' '0.0' '0.0' '2.0' '0.5' '0.0' '0.0' '0.0'
 '0.0' '0.0' '0.0' '0.0' '0.0' '0.0' '0.0' '0.0' '0.0' '1.0' '0.5' '0.0'
 '0.5' '0.0' '1.0' '0.5' '0.0' '0.5' '0.0' '0.5' '0.0' '1.0' '0.5' '0.5'
 '0.0' '0.0' '1.0' '1.0' '0.0' '0.0' '1.0' '0.5' '0.0' '0.0' '0.5' '0.5'
 '0.0' '0.0' '0.0' '0.5' '0.0' '0.5' '1.0' '0.5' '0.5' '0.0' '0.0' '0.0'
 '0.0' '0.0' '0.0' '0.0' '0.0' '0.5' '0.0' '0.0' '0.0' '0.5' '0.5' '0.5'
 '0.5' '0.0' '0.0' '0.0' '0.0' '0.0' '0.0' '0.5' '1.0' '0.5' '0.5' '0.0'
 '0.5' '0.0' '0.0' '0.0' '0.0' '0.5' '0.0' '0.5' '0.5' '0.5' '0.5' '0.0'
 '0.0' '0.5' '0.5' '0.5' '0.5' '0.0' '0.0' '0.0' '0.0' '0.5' '1.0' '0.0'
 '0.0' '0.0' '0.0' '0.0' '0.0' '0.5' '0.5' '0.5' '0.0' '0.0' '0.0' '0.0'
 '0.0' '0.0' '0.0' '0.0' '0.0' '0.0' '0.5' '0.5' '0.5' '0.5' '0.5' '0.5'
 '0.5' '0.5' '0.5' '0.0' '0.0' '0.5' '0.0' '0.0' '0.5' '0.5' '1.0' '1.0'
 '1.0' '0.0' '0.0' '0.5' '1.0' '0.0' '0.0' '1.0' '1.0' '1.0' '1.0' '1.0'
 '0.0' '0.0' '0.0' '0.5' '1.0' '0.0' '0.0' '0.0' '0.0' '0.0' '0.0' '0.0'
 '0.0' '0.5' '0.0' '0.0' '0.0' '0.0' '0.0' '0.0' '0.0' '0.5' '0.5' '0.5'
 '0.5' '0.5' '0.0' '0.0' '0.0' '0.0' '0.0' '0.0' '1.0' '0.5' '0.5' '1.0'
```

```
'0.5' '0.0' '0.0' '0.0' '0.0' '0.0' '0.0' '0.0' '0.0' '0.0' '0.0' '0.0'
'0.0' '0.0' '1.0' '1.0' '0.0' '0.0' '0.0' '0.0' '0.0' '0.5' '0.5' '0.0'
'0.0' '0.0' '0.0' '0.0' '0.0' '0.0' '0.0' '0.5' '1.0' '1.0' '0.5' '0.5'
'0.5' '0.5' '0.5' '0.0' '0.0' '0.0' '0.0' '1.0' '2.0' '1.0' '1.0' '0.5'
'1.0' '0.0' '0.0' '0.0' '0.0' '0.0' '0.0' '0.5' '0.0' '0.0' '0.0' '0.0'
'0.0' '0.0' '0.0' '0.0' '0.0' '0.5' '0.5' '0.0' '0.0' '0.0' '0.0' '0.0'
'0.0' '0.5' '0.5' '0.5' '0.0' '0.5' '0.5' '0.5' '1.0' '0.0' '0.0' '1.0'
'1.0' '0.5' '0.5' '0.0' '0.0' '0.5' '0.5' '0.5' '0.5' '0.5' '0.5' '0.5'
'0.5' '0.0' '0.0' '0.0' '0.0' '0.0' '0.5' '0.0' '0.0' '0.0' '1.0' '2.0'
'0.0' '0.0' '0.0' '0.0' '0.5' '0.5' '0.0' '0.0' '0.0' '0.0' '0.5' '0.5'
'0.5' '0.5' '0.0' '0.0' '0.0' '0.0' '0.0' '0.5' '0.0' '0.0' '0.5' '0.5'
'0.5' '0.0' '0.0' '0.5' '0.5' '0.0' '0.0' '0.5' '0.5' '0.5' '0.5' '0.5'
'0.0' '0.0' '0.0' '0.0' '0.0' '0.0' '0.0' '0.0' '0.5' '0.0' '0.5' '1.0'
'1.0' '0.0' '0.0' '0.0' '0.0' '0.0' '0.0' '0.5' '1.0' '0.0' '0.0' '0.0'
'0.0' '0.0' '0.0' '0.0' '1.0' '1.0' '0.0' '0.0' '0.5' '0.5' '0.0' '0.0'
'0.0' '0.5' '0.5' '1.0' '2.0' '1.0' '1.0' '0.0' '0.0' '0.0' '0.0' '0.0'
'0.5' '0.5' '0.0' '0.0' '0.0' '0.5' '0.5' '0.5' '0.0' '0.0' '0.5' '0.0'
'0.0' '0.0' '0.0' '0.0' '0.5' '0.5' '0.0' '0.0' '0.0' '0.0' '0.5' '1.0'
'0.5' '0.5' '0.5' '0.0' '0.0' '0.0' '2.0' '2.0' '2.0' '2.0' '2.0' '2.0'
'2.0' '2.0' '2.0' '2.0']
```

In [110]:

```
# Make predictions on validation dataset using GradientBoostingClassifier
from sklearn.ensemble import GradientBoostingClassifier
GBM = GradientBoostingClassifier()
GBM.fit(X_train, Y_train)
predictions = GBM.predict(X_validation)
print("*Accuracy score for GBM: ", accuracy_score(Y_validation, predictions),
print("*Confusion Matrix for GBM: ")
print(confusion_matrix(Y_validation, predictions))
print("*Classification Report for GBM: ")
print(classification_report(Y_validation, predictions))
```

```
*Accuracy score for GBM:  0.810344827586

*Confusion Matrix for GBM:
[[62  3  0  0]
 [14 22  1  0]
 [ 0  2  8  2]
 [ 0  0  0  2]]
*Classification Report for GBM:
             precision    recall  f1-score   support

        0.0       0.82      0.95      0.88        65
        0.5       0.81      0.59      0.69        37
        1.0       0.89      0.67      0.76        12
        2.0       0.50      1.00      0.67         2

avg / total       0.82      0.81      0.80       116
```

In [ ]:

```
1
```