Alzheimer Prediction

Organization of this document

1. Introduction

What is Dementia?

Dementia is a broad category of brain diseases that cause a long-term and often gradual decrease in the ability to think and remember that is severe enough to affect daily functioning. Other common symptoms include emotional problems, difficulties with language, and a decrease in motivation. Consciousness is usually not affected. A dementia diagnosis requires a change from a person's usual mental functioning and a greater decline than one would expect due to aging. These diseases have a significant effect on caregivers.

The most common type of dementia is Alzheimer's disease, which makes up 50% to 70% of cases. Other common types include vascular dementia (25%), dementia with Lewy bodies (15%), and frontotemporal dementia. Less common causes include normal pressure hydrocephalus, Parkinson's disease dementia, syphilis, HIV, and Creutzfeldt–Jakob disease. More than one type of dementia may exist in the same person. A small proportion of cases run in families. In the DSM-5, dementia was reclassified as a neurocognitive disorder, with degrees of severity. Diagnosis is usually based on history of the illness and cognitive testing with medical imaging and blood tests used to rule out other possible causes. The mini mental state examination is one commonly used cognitive test. Efforts to prevent dementia include trying to decrease risk factors such as high blood pressure, smoking, diabetes, and obesity. Screening the general population for the disorder is not recommended.

There is no known cure for dementia. Cholinesterase inhibitors such as donepezil are often used and may be beneficial in mild to moderate disorder. Overall benefit, however, may be minor. There are many measures that can improve the quality of life of people with dementia and their caregivers. Cognitive and behavioral interventions may be appropriate. Educating and providing emotional support to the caregiver is important. Exercise programs may be beneficial with respect to activities of daily living and potentially improve outcomes.Treatment of behavioral problems with antipsychotics is common but not usually recommended, due to the limited benefit and the side effects, including an increased risk of death.

Globally, dementia affected about 46 million people in 2015. About 10% of people develop the disorder at some point in their lives. It becomes more common with age. About 3% of people between the ages of 65–74 have dementia, 19% between 75 and 84, and nearly half of those over 85 years of age. In 2013 dementia resulted in about 1.7 million deaths, up from 0.8 million in 1990. As more people are living longer, dementia is becoming more common. For people of a specific age, however, it may be becoming less frequent, at least in the developed world, due to a decrease in risk factors. It is one of the most common causes of disability among the old. It is believed to result in economic costs of US$604 billion a year.[2] People with dementia are often physically or chemically restrained to a greater degree than necessary, raising issues of human rights. Social stigma against those affected is common.

Alzheimers disease

Alzheimer's disease (AD), also referred to simply as Alzheimer's, is a chronic neurodegenerative disease that usually starts slowly and gradually worsens over time. It is the cause of 60–70% of cases of dementia. The most common early symptom is difficulty in remembering recent events. As the disease advances, symptoms can include problems with language, disorientation (including easily getting lost), mood swings, loss of motivation, not managing self-care, and behavioural issues. As a person's condition declines, they often withdraw from family and society. Gradually, bodily functions are lost, ultimately leading to death. Although the speed of progression can vary, the typical life expectancy following diagnosis is three to nine years. The cause of Alzheimer's disease is poorly understood. About 70% of the risk is believed to be inherited from a person's parents with many genes usually involved. Other risk factors include a history of head injuries, depression, and hypertension. The disease process is associated with plaques and neurofibrillary tangles in the brain. A probable diagnosis is based on the history of the illness and cognitive testing with medical imaging and blood tests to rule out other possible causes. Initial symptoms are often mistaken for normal ageing. Examination of brain tissue is needed for a definite diagnosis. Mental and physical exercise, and avoiding obesity may decrease the risk of AD; however, evidence to support these recommendations is weak. There are no medications or supplements that have been shown to decrease risk.

No treatments stop or reverse its progression, though some may temporarily improve symptoms. Affected people increasingly rely on others for assistance, often placing a burden on the caregiver. The pressures can include social, psychological, physical, and economic elements. Exercise programs may be beneficial with respect to activities of daily living and can potentially improve outcomes. Behavioural problems or psychosis due to dementia are often treated with antipsychotics, but this is not usually recommended, as there is little benefit with an increased risk of early death.

In 2015, there were approximately 29.8 million people worldwide with AD. It most often begins in people over 65 years of age, although 4–5% of cases are early-onset Alzheimer's. It affects about 6% of people 65 years and older. In 2015, dementia resulted in about 1.9 million deaths. It was first described by, and later named after, German psychiatrist and pathologist Alois Alzheimer in 1906. In developed countries, AD is one of the most financially costly diseases.

2. Objective

> Based on the given data predicting the alzheimer`s disease

3. Understanding the data

Dataset was taken from here : https://www.kaggle.com/jboysen/mri-and-alzheimers (https://www.kaggle.com/jboysen/mri-and-alzheimers)

The dataset 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. The subjects are all right-handed and include both men and women. 72 of the subjects were characterized as nondemented 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 nondemented at the time of their initial visit and were subsequently characterized as demented at a later visit.

Feature	Description
Subject_id	The uniquie id given to a person.
MRI id	The unique id given to a sinlge MRI scan.
Group	Whether the patient is Demented/Non Demented or Converted
Visit	Contains the number of visits. (This feature contains the visit in order.)
MR Delay	The MR delauy time.
M.F	Denoted the gender of the patient. (M/F)
Hand	Contains whether the patient is right handed or left handed. (In this dataset all the patients are right handed.)
Age	Contains the age of the patient.
EDUC	Contains the years of education the patient went through.
SES	Contains the socio-economic status of the patient.
MMSE	Stands for "Mini mental state examination score". Ranges from 0(worst) to 30 (best).
CDR	Stands for clinical dementia rating. It`s a 5 point scale score. (0 = No dementia, 0.5 = Very Mild, 1 = Mild, 2 = Moderate, 3 = Severe)
eTIV	Stands for"estimated intracranial volume" measured in mm3.
nWBV	Stands for "Normalized Whole brain volume".
ASF	Stands for "Atlas scaling factor"

4. Importing the necessary packages

In [40]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
from sklearn.dummy import DummyClassifier
from sklearn.metrics import log_loss
from sklearn.metrics import roc_auc_score
from scipy.sparse import hstack
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
```

5. Importing the data

In [2]:

```
data = pd.read_csv("Data/oasis_longitudinal.csv")
data.head()
```

Out[2]:

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

6. Understanding the high level details about the data

In [3]:

```
data.describe().T
```

Out[3]:

	count	mean	std	min	25%	50%	75%	max
Visit	373.0	1.882038	0.922843	1.000	1.000	2.000	2.000	5.000
MR Delay	373.0	595.104558	635.485118	0.000	0.000	552.000	873.000	2639.000
Age	373.0	77.013405	7.640957	60.000	71.000	77.000	82.000	98.000
EDUC	373.0	14.597855	2.876339	6.000	12.000	15.000	16.000	23.000
SES	354.0	2.460452	1.134005	1.000	2.000	2.000	3.000	5.000
MMSE	371.0	27.342318	3.683244	4.000	27.000	29.000	30.000	30.000
CDR	373.0	0.290885	0.374557	0.000	0.000	0.000	0.500	2.000
eTIV	373.0	1488.128686	176.139286	1106.000	1357.000	1470.000	1597.000	2004.000
nWBV	373.0	0.729568	0.037135	0.644	0.700	0.729	0.756	0.837
ASF	373.0	1.195461	0.138092	0.876	1.099	1.194	1.293	1.587

In [4]:

```
data.isnull().sum()
```

Out[4]:

```
Subject ID     0
MRI ID         0
Group          0
Visit          0
MR Delay       0
M/F            0
Hand           0
Age            0
EDUC           0
SES           19
MMSE           2
CDR            0
eTIV           0
nWBV           0
ASF            0
dtype: int64
```

There are 19 missing values in the "SES" and 2 missing values in the "MMSE" features. The dataset contains 15 features and as well as 373 rows.

7. Exploring the data through visualizations

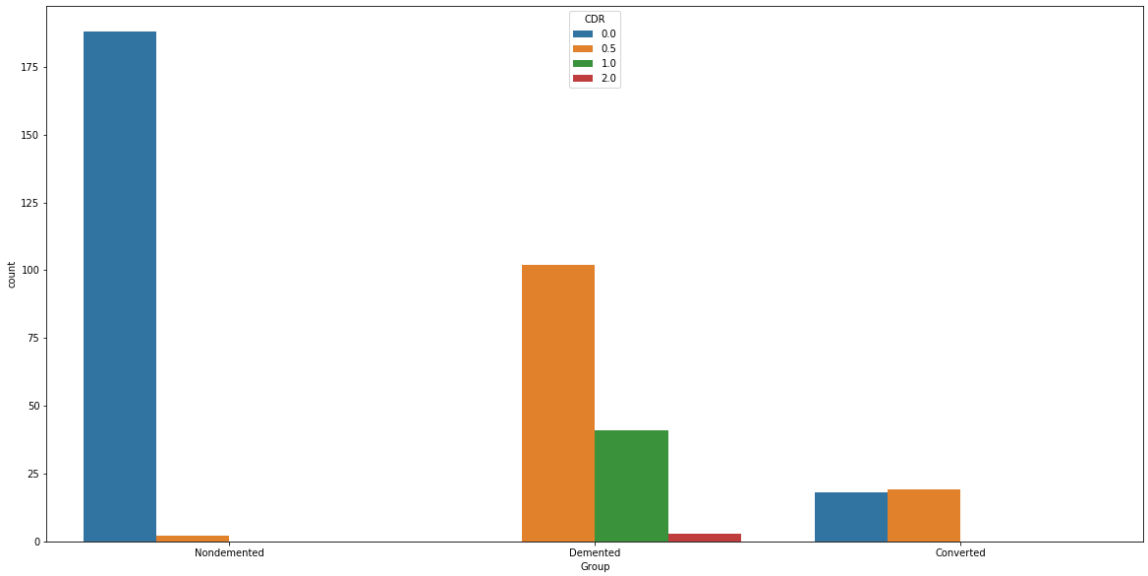
7.1 Exploring Group feature

In [5]:

```
plt.figure(figsize=(20,10))
sns.countplot(data=data,x="Group",hue="CDR")
```

Out[5]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f52b526fda0>
```



- Even though a subject is put under a group of "Non Demented", A few of them have a clinical dementia rating as 0.5(MILD).
- Majority of the people who got converted from mild cognitive impairment to Dementia have a CDR of 1.0 or 0.5.

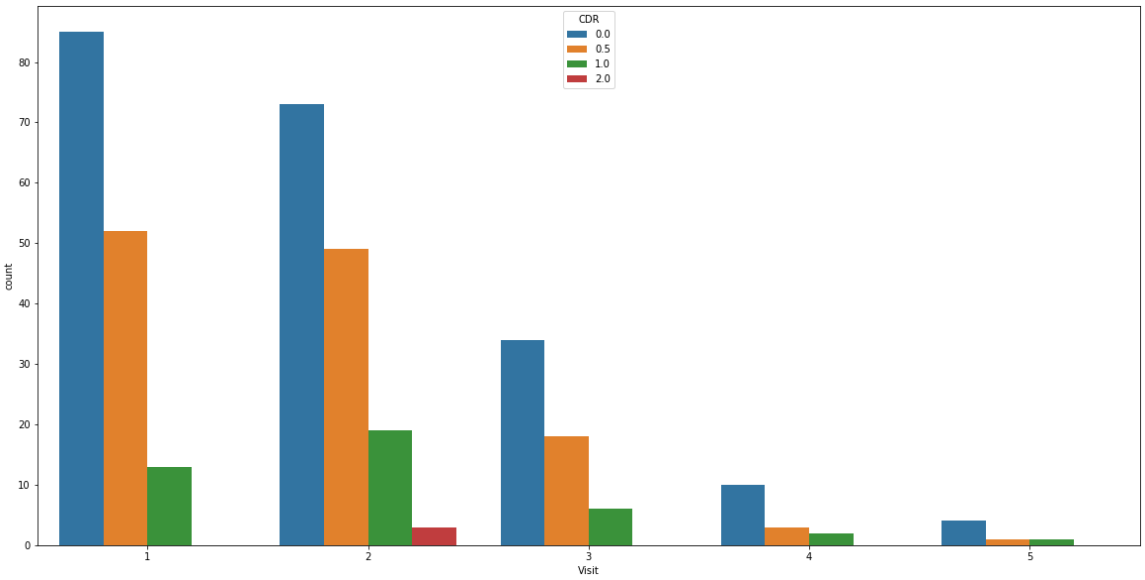
7.2 Exploring Visit feature

In [6]:

```
plt.figure(figsize=(20,10))
sns.countplot(data=data,x="Visit",hue="CDR")
```

Out[6]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f52b4a4e208>
```



- There are only few people who visited >4 number of times.

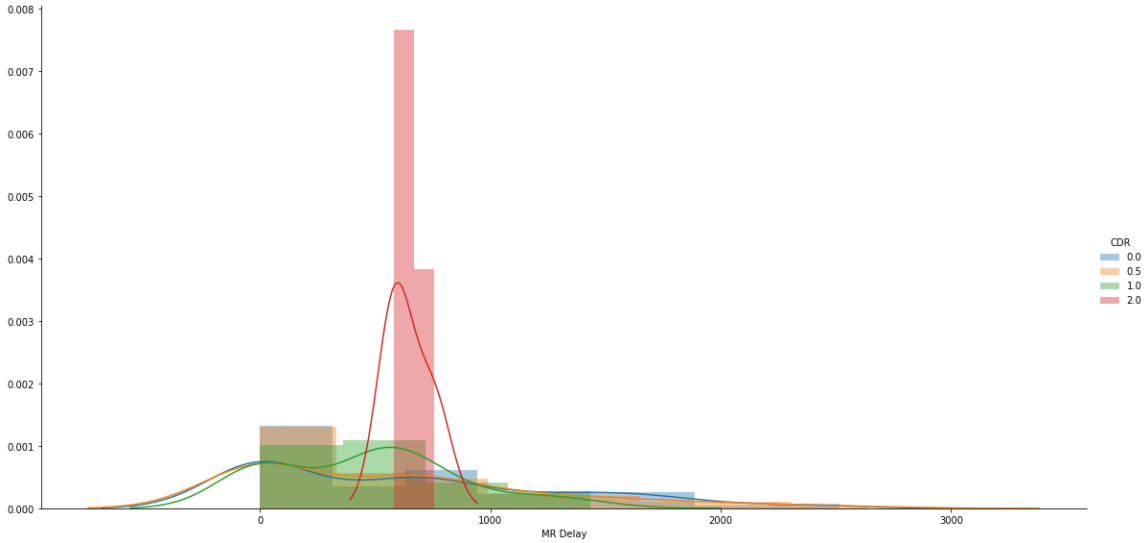
7.3 Exploring the MR Delay feature

In [7]:

```
sns.FacetGrid(data=data,hue="CDR",height=8,aspect=2).map(sns.distplot,"MR Delay"
).add_legend()
```

Out[7]:

```
<seaborn.axisgrid.FacetGrid at 0x7f52b499ac50>
```



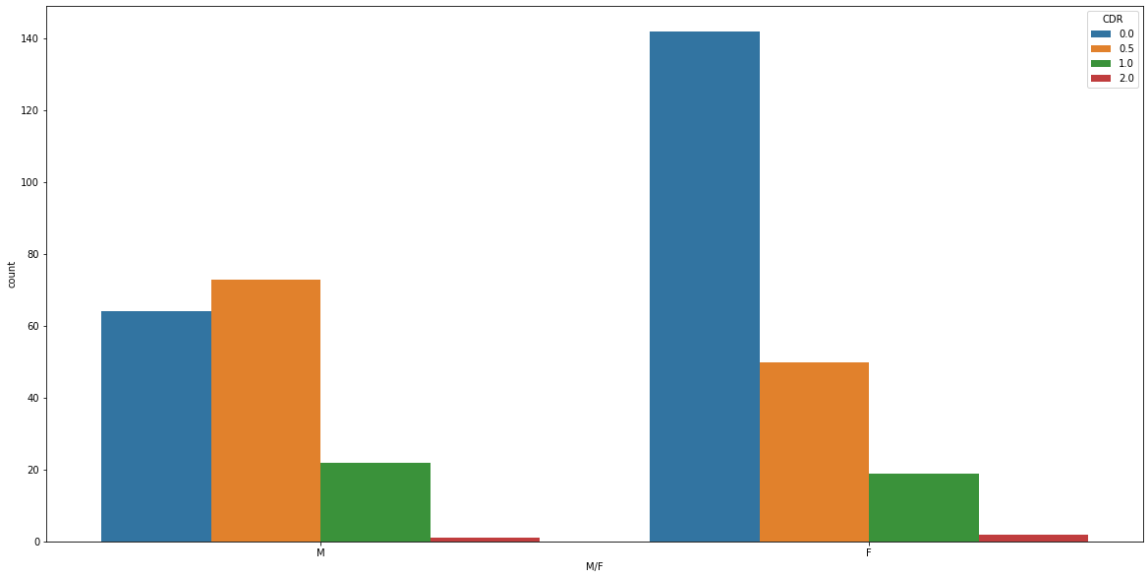
7.4 Exploring the Gender feature

In [8]:

```
plt.figure(figsize=(20,10))
sns.countplot(data=data,x="M/F",hue="CDR")
```

Out[8]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f52b4979048>
```



- There are slightly more number of males who are suffering from dementia rather than females.

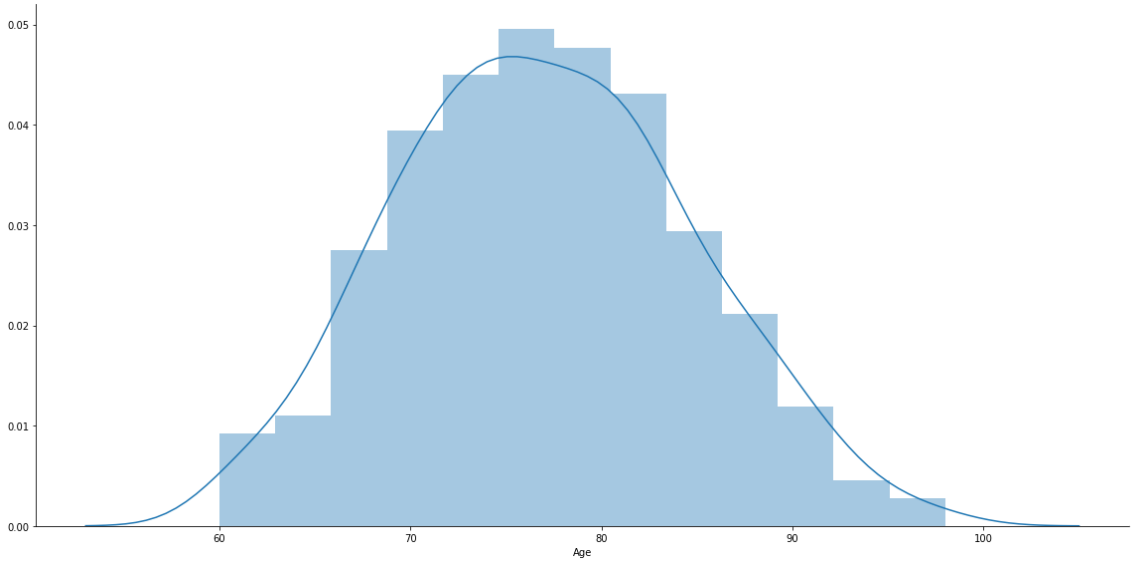
7.5 Exploring the Age feature

In [9]:

```
sns.FacetGrid(data=data,height=8,aspect=2).map(sns.distplot,"Age").add_legend()
```

Out[9]:

```
<seaborn.axisgrid.FacetGrid at 0x7f52b484af98>
```



- The age feature follows a gaussian distribution. We can further analyze this feature to gain more insights.

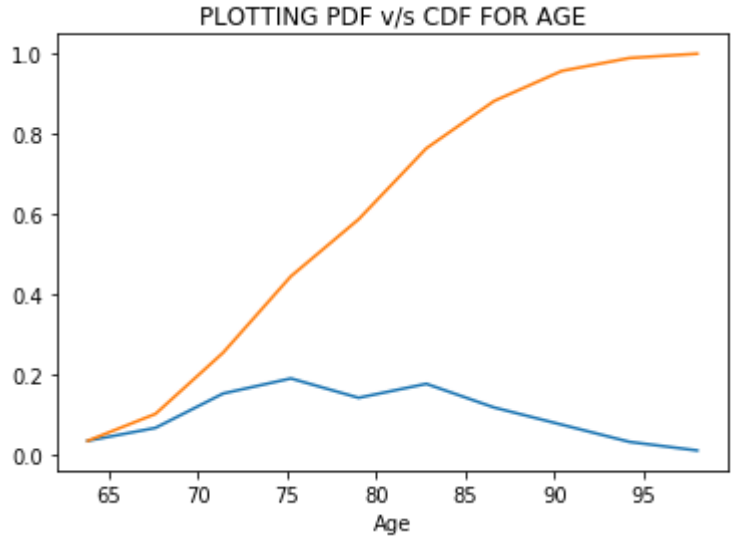
PDF v/s CDF for the Age attribute :

In [10]:

```
count1,binedges1 = np.histogram(data['Age'], bins=10, density = True)
pdf1=count1/sum(count1)
cdf1=np.cumsum(pdf1)
plt.plot(binedges1[1:],pdf1)
plt.plot(binedges1[1:], cdf1)
plt.xlabel("Age")
plt.title("PLOTTING PDF v/s CDF FOR AGE")
```

Out[10]:

```
Text(0.5, 1.0, 'PLOTTING PDF v/s CDF FOR AGE')
```



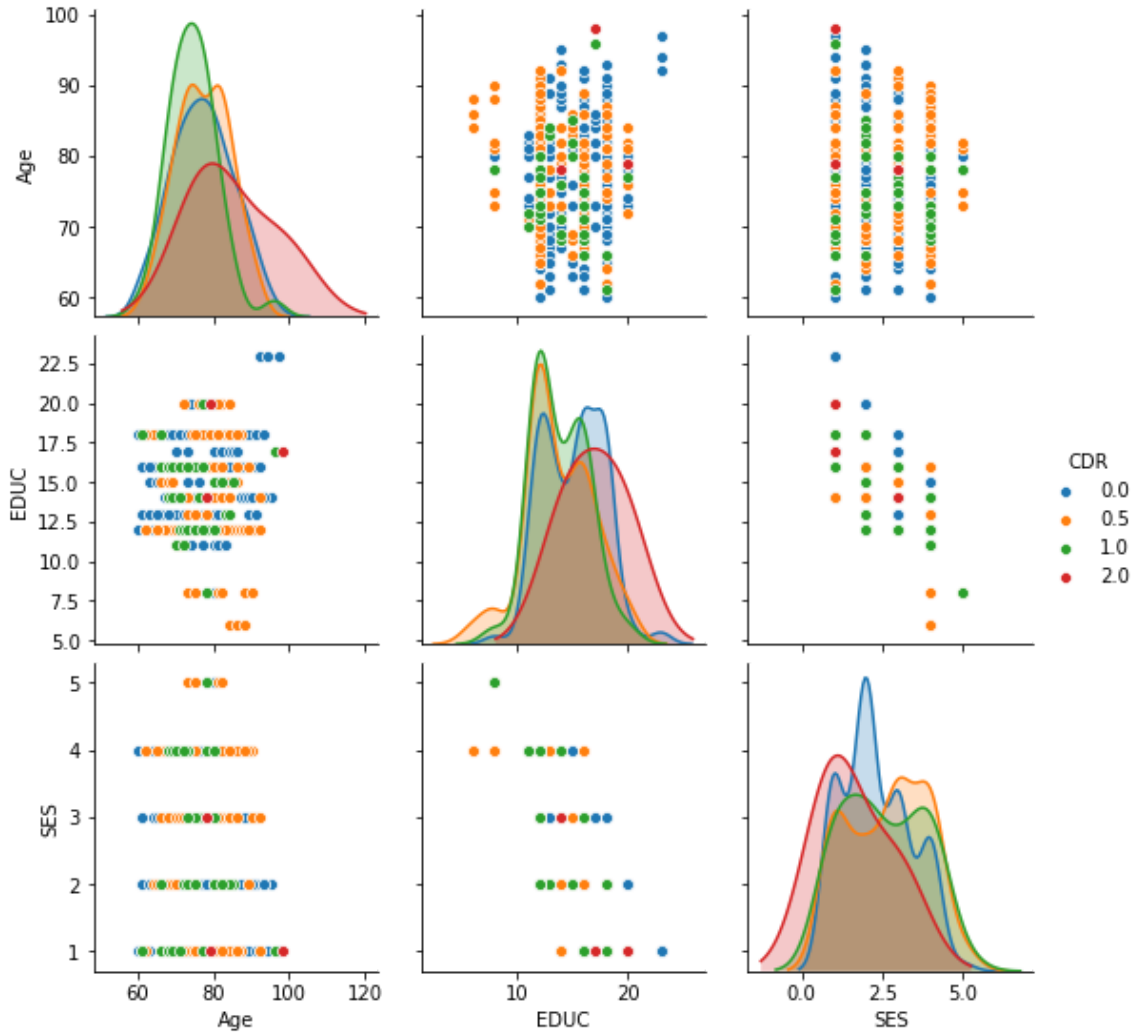
7.6 Exploring the Demographic features

In [11]:

```
Demographics = data[['Age','EDUC','SES','CDR']]
sns.pairplot(Demographics,hue="CDR")
```

Out[11]:

```
<seaborn.axisgrid.PairGrid at 0x7f52b476a4e0>
```



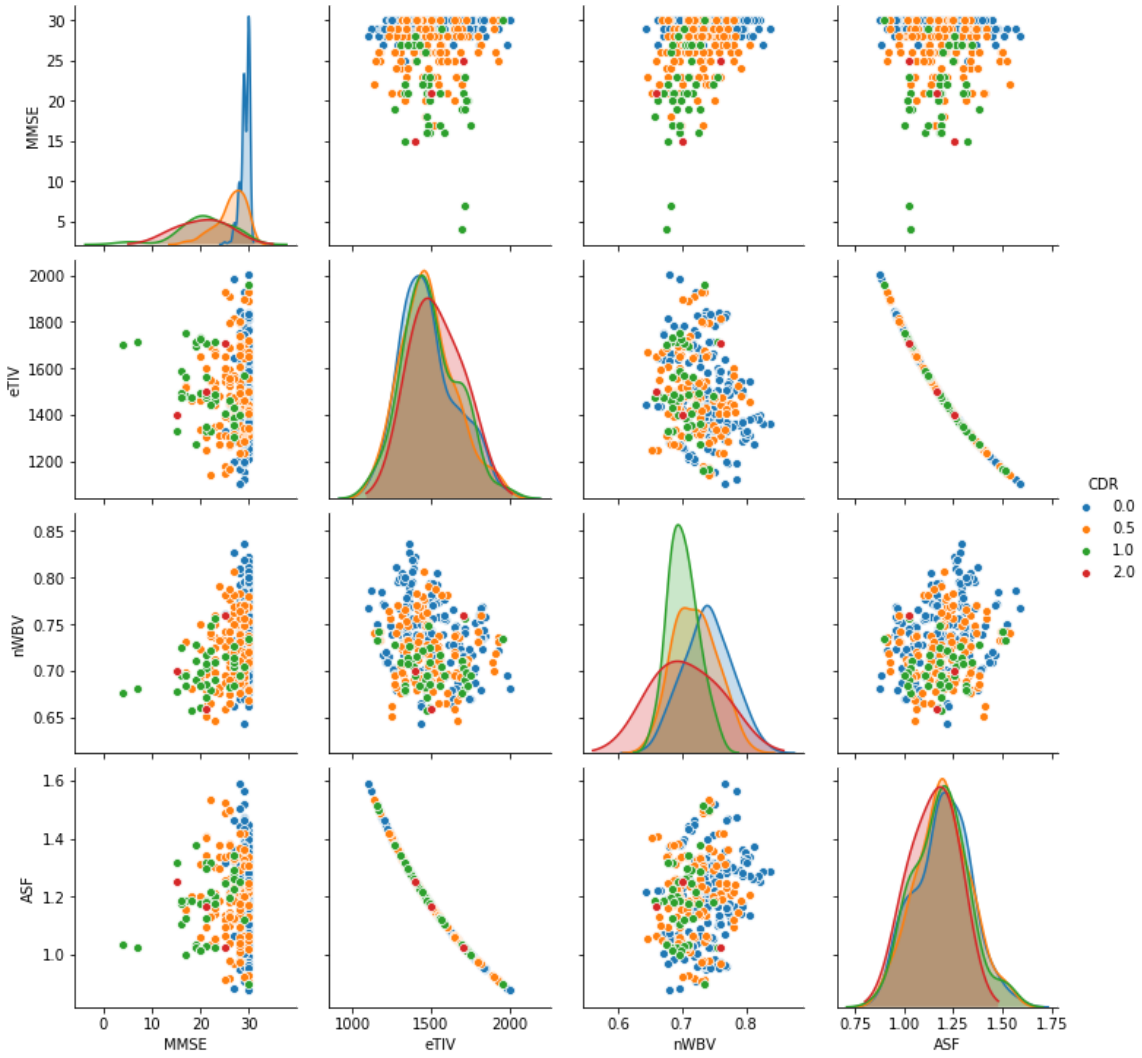
7.7 Exploring the Clincal features

In [12]:

```
Clinical = data[['MMSE','eTIV','nWBV','ASF','CDR']]
sns.pairplot(Clinical,hue="CDR")
```

Out[12]:

```
<seaborn.axisgrid.PairGrid at 0x7f52b3fc9c18>
```



- It is very clear that the eTIV and ASF features are negatively correlated. i.e As the eTIV value increases the ASF value decreases and vice versa.

8. Analyzing the imbalance in the data

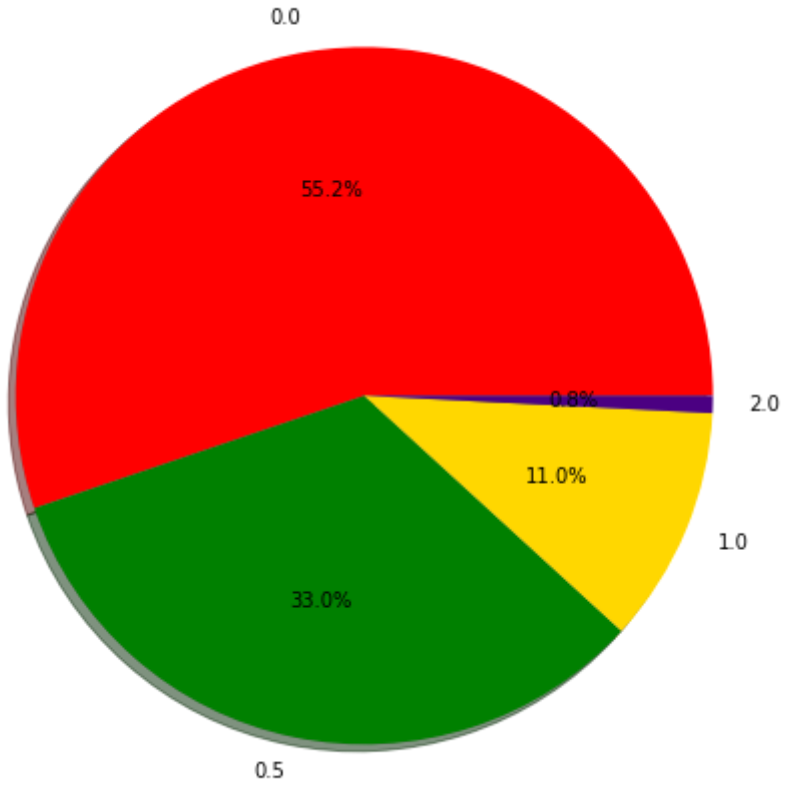
In [13]:

```
class_label = data['CDR'].value_counts()
total_points = len(data)
print("Points with CDR = 0.0  are = ",class_label.values[0]/total_points*100,"%"
)
print("Points with CDR = 0.5 are = ",class_label.values[1]/total_points*100,"%")
print("Points with CDR = 1.0 are = ",class_label.values[2]/total_points*100,"%")
print("Points with CDR = 2.0 are = ",class_label.values[3]/total_points*100,"%")
labels = ['0.0','0.5','1.0','2.0']
sizes = [55.22,32.97,10.99,0.80]
colors = ['red','green', 'gold','indigo']
plt.figure(figsize=(8,8))
plt.pie(sizes, labels=labels, colors=colors,autopct='%1.1f%%', shadow=True)
```

```
Points with CDR = 0.0  are =  55.22788203753352 %
Points with CDR = 0.5 are =  32.975871313672926 %
Points with CDR = 1.0 are =  10.991957104557642 %
Points with CDR = 2.0 are =  0.8042895442359249 %
```

Out[13]:

```
([<matplotlib.patches.Wedge at 0x7f52b207c5c0>,
  <matplotlib.patches.Wedge at 0x7f52b207cf60>,
  <matplotlib.patches.Wedge at 0x7f52b20868d0>,
  <matplotlib.patches.Wedge at 0x7f52b208f240>],
 [Text(-0.1799594079746407, 1.0851795296085422, '0.0'),
  Text(-0.2251403805952944, -1.0767134293884357, '0.5'),
  Text(1.0150396579339906, -0.4239038721470324, '1.0'),
  Text(1.0996524674345731, -0.027648704581864855, '2.0')],
 [Text(-0.09815967707707672, 0.5919161070592048, '55.2%'),
  Text(-0.12280384396106966, -0.587298234211874, '33.0%'),
  Text(0.5536579952367221, -0.23122029389838128, '11.0%'),
  Text(0.5998104367824943, -0.015081111590108099, '0.8%')])
```



- **The dataset is imbalanced**. As there are more number of points with CDR = 0.0. And there are very few points with CDR = 1.0 (11%) and CDR = 2.0 (0.8%).

9. Making necessary assumptions and changes

- As the dataset is imbalanced, we cannot use the accuracy score as a metric to determine the correctness of our model. We would use ROC curves and AUC scores to determine the model.
- Initially we are going to train a random model and use its accuracy as our baseline.
- As to improve the AUC and reduce the generalization error we are going to use hyper parameter tuning to tune the hyper parameters of our models.
- We are going to drop some columns which are not useful in building our model.
- We are going to drop Subject ID, MRI ID, Hand (because all are right handed only).
- From the plot of 7.7 we can understand that the features eTIV and ASF are negatively correlated. So We can either drop any one of the both features or keep them. Its better to analyze performance of the model with and without removing the collinear features.

Dropping the unnecessary columns

In [14]:

```
new_data = data.drop(['Subject ID', 'MRI ID','Hand'],axis=1)
```

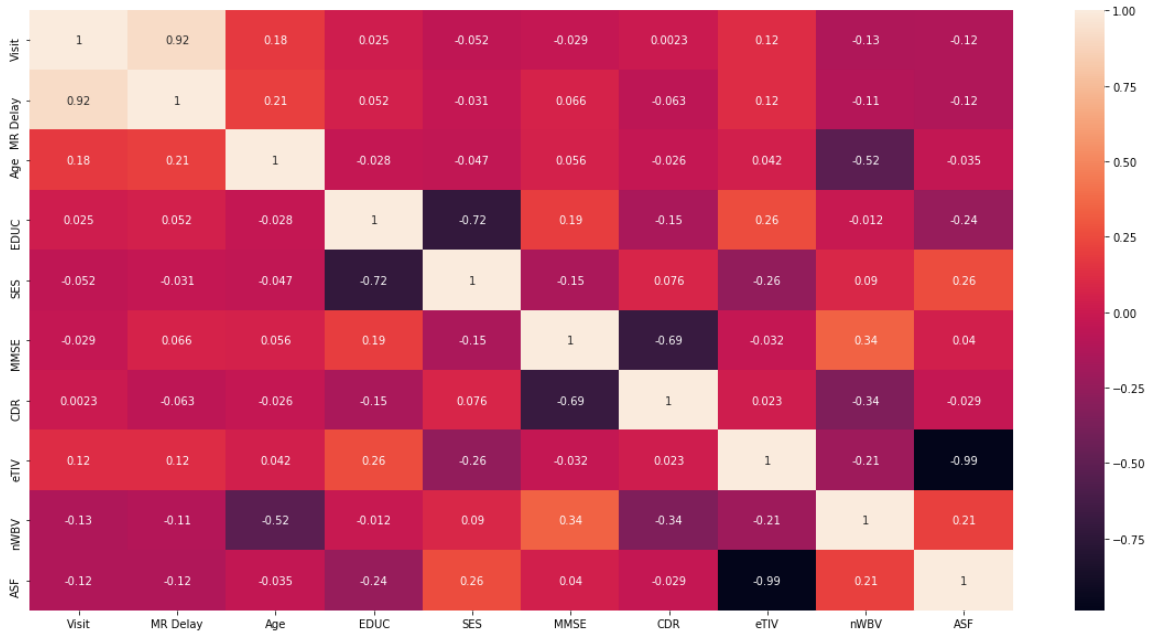
Checking the correlation matrix

In [15]:

```
cor = new_data.corr()
plt.figure(figsize=(20,10))
sns.heatmap(cor,annot=True)
```

Out[15]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f52b208fe48>
```



- **From the above correlation plot, we can understand that the eTIV and ASF are highly correlated so we can safely drop one feature. And similarly (EDUC [==] SES) are negatively correlated and as well as (MRDelay [==] Visit) are positively correlated. We cannot be sure to remove any feature, as we donot know the consequences of it.**

10. Dividing the data into train and test

Dividing data into X and Y

In [16]:

```
x = new_data.drop(['CDR'],axis=1)
y = new_data['CDR'].values
```

Dividing into train and test sets

In [17]:

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3,random_s
tate=0)
print("Shape of x_train : ",x_train.shape)
print("Shape of x_test : ",x_test.shape)
```

```
Shape of x_train :  (261, 11)
Shape of x_test :  (112, 11)
```

11. Feature Encoding

11.1 Encoding the Group feature

In [18]:

```
vectorizer = CountVectorizer()
vectorizer.fit(x_train['Group'])
x_train_group = vectorizer.transform(x_train['Group'])
x_test_group = vectorizer.transform(x_test['Group'])
print(x_train_group.shape)
print(x_test_group.shape)
```

```
(261, 3)
(112, 3)
```

11.2 Encoding the MR Delay feature

In [19]:

```
normalizer = preprocessing.Normalizer()
normalizer.fit(x_train['MR Delay'].values.reshape(1,-1))
x_train_mrdelay = normalizer.transform(x_train['MR Delay'].values.reshape(1,-1))
.T
x_test_mrdelay = normalizer.transform(x_test['MR Delay'].values.reshape(1,-1)).T
print(x_train_mrdelay.shape)
print(x_test_mrdelay.shape)
```

```
(261, 1)
(112, 1)
```

11.3 Encoding the Gender feature

In [20]:

```
le = preprocessing.LabelEncoder()
le.fit(x_train['M/F'])
x_train_gender = le.transform(x_train['M/F']).reshape(1,-1).T
x_test_gender = le.transform(x_test['M/F']).reshape(1,-1).T
print(x_train_gender.shape)
print(x_test_gender.shape)
```

```
(261, 1)
(112, 1)
```

11.4 Encoding the Age feature

In [21]:

```
normalizer = preprocessing.Normalizer()
normalizer.fit(x_train['Age'].values.reshape(1,-1))
x_train_age = normalizer.transform(x_train['Age'].values.reshape(1,-1)).T
x_test_age = normalizer.transform(x_test['Age'].values.reshape(1,-1)).T
print(x_train_age.shape)
print(x_test_age.shape)
```

```
(261, 1)
(112, 1)
```

11.5 Encoding the EDUC feature

In [22]:

```
normalizer = preprocessing.Normalizer()
normalizer.fit(x_train['EDUC'].values.reshape(1,-1))
x_train_educ = normalizer.transform(x_train['EDUC'].values.reshape(1,-1)).T
x_test_educ = normalizer.transform(x_test['EDUC'].values.reshape(1,-1)).T
print(x_train_educ.shape)
print(x_test_educ.shape)
```

```
(261, 1)
(112, 1)
```

11.6 Encoding the SES feature

Filling missing values with 0

In [23]:

```
x_train['SES'] = x_train['SES'].fillna(0.0)
x_test['SES'] = x_test['SES'].fillna(0.0)
```

```
/home/chiranjeevi_karthik/anaconda3/envs/karthik/lib/python3.6/site-
packages/ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pand
as-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-co
py
  """Entry point for launching an IPython kernel.
/home/chiranjeevi_karthik/anaconda3/envs/karthik/lib/python3.6/site-
packages/ipykernel_launcher.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pand
as-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-co
py
```

Converting the SES column from float to int

In [24]:

```
x_train.SES = x_train.SES.astype(int)
x_test.SES = x_test.SES.astype(int)
```

```
/home/chiranjeevi_karthik/anaconda3/envs/karthik/lib/python3.6/site-
packages/pandas/core/generic.py:5302: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pand
as-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-co
py
  self[name] = value
```

Note : When you use int in the above code.... the datatype 'int64' is shown in the x_train.info()

11.7 Encoding the MMSE feature

Imputing missing values with the 0

In [25]:

```
x_train['MMSE'] = x_train['MMSE'].fillna(0)
x_test['MMSE'] = x_test['MMSE'].fillna(0)
```

```
/home/chiranjeevi_karthik/anaconda3/envs/karthik/lib/python3.6/site-
packages/ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pand
as-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-co
py
  """Entry point for launching an IPython kernel.
/home/chiranjeevi_karthik/anaconda3/envs/karthik/lib/python3.6/site-
packages/ipykernel_launcher.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pand
as-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-co
py
```

In [26]:

```
normalizer = preprocessing.Normalizer()
normalizer.fit(x_train['MMSE'].values.reshape(1,-1))
x_train_mmse = normalizer.transform(x_train['MMSE'].values.reshape(1,-1)).T
x_test_mmse = normalizer.transform(x_test['MMSE'].values.reshape(1,-1)).T
print(x_train_mmse.shape)
print(x_test_mmse.shape)
```

```
(261, 1)
(112, 1)
```

11.8 Encoding the eTIV feature

In [27]:

```
normalizer = preprocessing.Normalizer()
normalizer.fit(x_train['eTIV'].values.reshape(1,-1))
x_train_etiv = normalizer.transform(x_train['eTIV'].values.reshape(1,-1)).T
x_test_etiv = normalizer.transform(x_test['eTIV'].values.reshape(1,-1)).T
print(x_train_etiv.shape)
print(x_test_etiv.shape)
```

```
(261, 1)
(112, 1)
```

11.9 Encoding the nWBV feature

In [28]:

```
normalizer = preprocessing.Normalizer()
normalizer.fit(x_train['nWBV'].values.reshape(1,-1))
x_train_nwbv = normalizer.transform(x_train['nWBV'].values.reshape(1,-1)).T
x_test_nwbv = normalizer.transform(x_test['nWBV'].values.reshape(1,-1)).T
print(x_train_nwbv.shape)
print(x_test_nwbv.shape)
```

```
(261, 1)
(112, 1)
```

11.10 Encoding the ASF feature

In [29]:

```
normalizer = preprocessing.Normalizer()
normalizer.fit(x_train['ASF'].values.reshape(1,-1))
x_train_asf = normalizer.transform(x_train['ASF'].values.reshape(1,-1)).T
x_test_asf = normalizer.transform(x_test['ASF'].values.reshape(1,-1)).T
print(x_train_asf.shape)
print(x_test_asf.shape)
```

```
(261, 1)
(112, 1)
```

11.11 Encoding the class label

In [30]:

```
#train class label
for i in range(len(y_train)):
    if y_train[i] == 0.0:
        y_train[i]=1
    if y_train[i] == 0.5:
        y_train[i]=2
    if y_train[i] == 1.0:
        y_train[i]=3
    if y_train[i] == 2.0:
        y_train[i]=4
#test class label
for i in range(len(y_test)):
    if y_test[i] == 0.0:
        y_test[i]=1
    if y_test[i] == 0.5:
        y_test[i]=2
    if y_test[i] == 1.0:
        y_test[i]=3
    if y_test[i] == 2.0:
        y_test[i]=4
```

12. Training a random model and assuming it as baseline

In [31]:

```
dummy_clf = DummyClassifier(strategy="uniform")
dummy_clf.fit(x_train, y_train)
print("Train Accuracy = ",dummy_clf.score(x_train,y_train))
print("Test Accuracy = ",dummy_clf.score(x_test,y_test))
```

```
Train Accuracy =  0.4827586206896552
Test Accuracy =  0.49107142857142855
```

13. Building the feature set

> Note: In here we are dropping the ASF feature.

In [32]:

```
x_tr = hstack((x_train_group,x_train['Visit'].values.reshape(-1,1),x_train_mrdel
ay,x_train_gender,x_train_age,x_train_educ,x_train['SES'].values.reshape(-1,1),x
_train_mmse,x_train_etiv,x_train_nwbv)).tocsr()
x_ts = hstack((x_test_group,x_test['Visit'].values.reshape(-1,1),x_test_mrdelay,
x_test_gender,x_test_age,x_test_educ,x_test['SES'].values.reshape(-1,1),x_test_m
mse,x_test_etiv,x_test_nwbv)).tocsr()
```

14. Training a random forest with best hyper parameters

In [33]:

```
estimator = RandomForestRegressor()
param_grid = {
"n_estimators"
: [10,20,30],
"max_features"
: ["auto", "sqrt", "log2"],
"min_samples_split" : [2,4,8],
"bootstrap": [True, False],
}
grid = GridSearchCV(estimator, param_grid, n_jobs=-1)
grid.fit(x_tr, y_train)
```

Out[33]:

```
GridSearchCV(cv=None, error_score=nan,
             estimator=RandomForestRegressor(bootstrap=True, ccp_alp
ha=0.0,
                                             criterion='mse', max_de
pth=None,
                                             max_features='auto',
                                             max_leaf_nodes=None,
                                             max_samples=None,
                                             min_impurity_decrease=
0.0,
                                             min_impurity_split=Non
e,
                                             min_samples_leaf=1,
                                             min_samples_split=2,
                                             min_weight_fraction_lea
f=0.0,
                                             n_estimators=100, n_job
s=None,
                                             oob_score=False, random
_state=None,
                                             verbose=0, warm_start=F
alse),
             iid='deprecated', n_jobs=-1,
             param_grid={'bootstrap': [True, False],
                         'max_features': ['auto', 'sqrt', 'log2'],
                         'min_samples_split': [2, 4, 8],
                         'n_estimators': [10, 20, 30]},
             pre_dispatch='2*n_jobs', refit=True, return_train_score
=False,
             scoring=None, verbose=0)
```

In [34]:

```
grid.best_params_
```

Out[34]:

```
{'bootstrap': True,
 'max_features': 'auto',
 'min_samples_split': 4,
 'n_estimators': 20}
```

In [35]:

```
rf = RandomForestRegressor(n_estimators=30,min_samples_split=8,max_features='aut
o',bootstrap=True)
rf.fit(x_tr,y_train)
```

Out[35]:

```
RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='ms
e',
                      max_depth=None, max_features='auto', max_leaf_
nodes=None,
                      max_samples=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=1,
                      min_samples_split=8, min_weight_fraction_leaf=
0.0,
                      n_estimators=30, n_jobs=None, oob_score=False,
                      random_state=None, verbose=0, warm_start=Fals
e)
```

Evaluating the performance of the model

In [36]:

```
print(rf.score(x_tr,y_train))
print(rf.score(x_ts,y_test))
```

```
0.8780825780970305
0.28883246008106944
```

15. Training a Logistic regression model

In [37]:

```
params={"C":np.logspace(-3,3,7)}
lg=LogisticRegression(max_iter=1000)#if max_iter=100 ...the model will not conve
rge
grid=GridSearchCV(lg,params,cv=2)
grid.fit(x_tr,y_train)
```

Out[37]:

```
GridSearchCV(cv=2, error_score=nan,
             estimator=LogisticRegression(C=1.0, class_weight=None,
dual=False,
                                          fit_intercept=True,
                                          intercept_scaling=1, l1_ra
tio=None,
                                          max_iter=1000, multi_class
='auto',
                                          n_jobs=None, penalty='l2',
                                          random_state=None, solver
='lbfgs',
                                          tol=0.0001, verbose=0,
                                          warm_start=False),
             iid='deprecated', n_jobs=None,
             param_grid={'C': array([1.e-03, 1.e-02, 1.e-01, 1.e+00,
1.e+01, 1.e+02, 1.e+03])},
             pre_dispatch='2*n_jobs', refit=True, return_train_score
=False,
             scoring=None, verbose=0)
```

In [38]:

```
grid.best_params_
```

Out[38]:

```
{'C': 1000.0}
```

In [39]:

```
lg=LogisticRegression(C=1000,max_iter=1000)
lg.fit(x_tr,y_train)
print(lg.score(x_tr,y_train))
print(lg.score(x_ts,y_test))
```

```
0.9080459770114943
0.6517857142857143
```

16. Training a Naive bayes Model

In [46]:

```
gnb = GaussianNB()
gnb.fit(x_tr.toarray(), y_train)
print("Train accuracy  = ",gnb.score(x_tr.toarray(),y_train))
print("Test accuracy = ",gnb.score(x_ts.toarray(),y_test))
```

```
Train accuracy  =  0.8697318007662835
Test accuracy =  0.7142857142857143
```