dementia-svm-project

October 15, 2023

[1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

[2]:
```
df=pd.read_excel(r"C:
↪\Users\prera\OneDrive\Desktop\Imarticus\ML\datasets\dementia_dataset.xlsx")
```

[3]:
```
df
```

[3]:

	Subject ID	MRI ID	Group	Visit	MR Delay	M/F	Hand	Age	\
0	OAS2_0001	OAS2_0001_MR1	Nondemented	1.0	0.0	M	R	87.0	
1	OAS2_0001	OAS2_0001_MR2	Nondemented	2.0	457.0	M	R	88.0	
2	OAS2_0002	OAS2_0002_MR1	Demented	1.0	0.0	M	R	75.0	
3	OAS2_0002	OAS2_0002_MR2	Demented	2.0	560.0	M	R	76.0	
4	OAS2_0002	OAS2_0002_MR3	Demented	3.0	1895.0	M	R	80.0	
..	…	…	…	…	…	…	…	…	
368	OAS2_0185	OAS2_0185_MR2	Demented	2.0	842.0	M	R	82.0	
369	OAS2_0185	OAS2_0185_MR3	Demented	3.0	2297.0	NaN	R	86.0	
370	OAS2_0186	OAS2_0186_MR1	Nondemented	1.0	0.0	F	R	61.0	
371	OAS2_0186	OAS2_0186_MR2	Nondemented	NaN	763.0	F	R	63.0	
372	OAS2_0186	OAS2_0186_MR3	Nondemented	3.0	1608.0	F	R	NaN	

	EDUC	SES	MMSE	CDR	eTIV	nWBV	ASF
0	14.0	2.0	27.0	0.0	1987.0	0.696	0.883
1	14.0	2.0	30.0	0.0	2004.0	0.681	0.876
2	12.0	NaN	23.0	0.5	1678.0	0.736	1.046
3	12.0	NaN	28.0	0.5	1738.0	0.713	1.010
4	12.0	NaN	22.0	0.5	1698.0	0.701	1.034
..	…	…	…	…	…	…	…
368	16.0	1.0	NaN	0.5	1693.0	0.694	1.037
369	NaN	1.0	26.0	0.5	1688.0	0.675	NaN
370	13.0	2.0	30.0	0.0	1319.0	0.801	1.331
371	13.0	2.0	NaN	0.0	NaN	0.796	1.323
372	13.0	2.0	30.0	0.0	1333.0	0.801	1.317

[373 rows x 15 columns]

[4]:
```
df1=df.copy(deep=True)
df2=df.copy(deep=True)
df3=df.copy(deep=True)
```

[5]:
```
df.shape
```

[5]: (373, 15)

[6]:
```
df.head(10)
```

[6]:

	Subject ID	MRI ID	Group	Visit	MR Delay	M/F	Hand	Age
0	OAS2_0001	OAS2_0001_MR1	Nondemented	1.0	0.0	M	R	87.0
1	OAS2_0001	OAS2_0001_MR2	Nondemented	2.0	457.0	M	R	88.0
2	OAS2_0002	OAS2_0002_MR1	Demented	1.0	0.0	M	R	75.0
3	OAS2_0002	OAS2_0002_MR2	Demented	2.0	560.0	M	R	76.0
4	OAS2_0002	OAS2_0002_MR3	Demented	3.0	1895.0	M	R	80.0
5	OAS2_0004	OAS2_0004_MR1	Nondemented	1.0	0.0	NaN	R	88.0
6	OAS2_0004	OAS2_0004_MR2	Nondemented	2.0	538.0	F	NaN	90.0
7	OAS2_0005	OAS2_0005_MR1	Nondemented	1.0	0.0	M	R	80.0
8	OAS2_0005	OAS2_0005_MR2	Nondemented	2.0	1010.0	M	R	83.0
9	OAS2_0005	OAS2_0005_MR3	Nondemented	3.0	1603.0	NaN	R	85.0

	EDUC	SES	MMSE	CDR	eTIV	nWBV	ASF
0	14.0	2.0	27.0	0.0	1987.0	0.696	0.883
1	14.0	2.0	30.0	0.0	2004.0	0.681	0.876
2	12.0	NaN	23.0	0.5	1678.0	0.736	1.046
3	12.0	NaN	28.0	0.5	1738.0	0.713	1.010
4	12.0	NaN	22.0	0.5	1698.0	0.701	1.034
5	18.0	3.0	28.0	0.0	1215.0	0.710	1.444
6	18.0	3.0	27.0	0.0	1200.0	0.718	1.462
7	12.0	4.0	28.0	0.0	1689.0	0.712	1.039
8	12.0	4.0	29.0	0.5	1701.0	0.711	1.032
9	12.0	4.0	30.0	0.0	1699.0	0.705	1.033

[7]:
```
df.tail(10)
```

[7]:

	Subject ID	MRI ID	Group	Visit	MR Delay	M/F	Hand	Age
363	OAS2_0183	OAS2_0183_MR3	Nondemented	3.0	732.0	F	R	68.0
364	OAS2_0183	OAS2_0183_MR4	Nondemented	4.0	2107.0	F	R	72.0
365	OAS2_0184	OAS2_0184_MR1	Demented	1.0	0.0	F	R	72.0
366	OAS2_0184	OAS2_0184_MR2	Demented	2.0	553.0	F	NaN	73.0
367	OAS2_0185	OAS2_0185_MR1	Demented	1.0	0.0	M	R	80.0
368	OAS2_0185	OAS2_0185_MR2	Demented	2.0	842.0	M	R	82.0
369	OAS2_0185	OAS2_0185_MR3	Demented	3.0	2297.0	NaN	R	86.0
370	OAS2_0186	OAS2_0186_MR1	Nondemented	1.0	0.0	F	R	61.0

```
371  OAS2_0186  OAS2_0186_MR2  Nondemented    NaN     763.0    F    R  63.0
372  OAS2_0186  OAS2_0186_MR3  Nondemented    3.0    1608.0    F    R   NaN

     EDUC  SES  MMSE  CDR    eTIV   nWBV    ASF
363  13.0  2.0  30.0  0.0  1506.0  0.740    NaN
364  13.0  2.0  30.0  0.0  1510.0  0.723  1.162
365  16.0  3.0  24.0  0.5  1354.0  0.733  1.296
366  16.0  3.0  21.0  1.0     NaN  0.708  1.299
367  16.0  1.0  28.0  NaN  1704.0  0.711  1.030
368  16.0  1.0   NaN  0.5  1693.0  0.694  1.037
369   NaN  1.0  26.0  0.5  1688.0  0.675    NaN
370  13.0  2.0  30.0  0.0  1319.0  0.801  1.331
371  13.0  2.0   NaN  0.0     NaN  0.796  1.323
372  13.0  2.0  30.0  0.0  1333.0  0.801  1.317
```

[8]:
```
df.duplicated()
```

[8]:
```
0      False
1      False
2      False
3      False
4      False
       …
368    False
369    False
370    False
371    False
372    False
Length: 373, dtype: bool
```

[9]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 373 entries, 0 to 372
Data columns (total 15 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Subject ID  373 non-null    object
 1   MRI ID      373 non-null    object
 2   Group       373 non-null    object
 3   Visit       344 non-null    float64
 4   MR Delay    366 non-null    float64
 5   M/F         345 non-null    object
 6   Hand        343 non-null    object
 7   Age         355 non-null    float64
 8   EDUC        349 non-null    float64
 9   SES         337 non-null    float64
```

```
 10  MMSE        352 non-null    float64
 11  CDR         359 non-null    float64
 12  eTIV        340 non-null    float64
 13  nWBV        342 non-null    float64
 14  ASF         338 non-null    float64
dtypes: float64(10), object(5)
memory usage: 43.8+ KB
```

[10]:
```
df.describe(include='all')
```

[10]:

	Subject ID	MRI ID	Group	Visit	MR Delay	M/F
count	373	373	373	344.000000	366.000000	345
unique	150	373	3	NaN	NaN	2
top	OAS2_0070	OAS2_0001_MR1	Nondemented	NaN	NaN	F
freq	5	1	190	NaN	NaN	198
mean	NaN	NaN	NaN	1.877907	591.237705	NaN
std	NaN	NaN	NaN	0.933460	634.431780	NaN
min	NaN	NaN	NaN	1.000000	0.000000	NaN
25%	NaN	NaN	NaN	1.000000	0.000000	NaN
50%	NaN	NaN	NaN	2.000000	539.000000	NaN
75%	NaN	NaN	NaN	2.000000	871.250000	NaN
max	NaN	NaN	NaN	5.000000	2639.000000	NaN

	Hand	Age	EDUC	SES	MMSE	CDR
count	343	355.000000	349.000000	337.000000	352.000000	359.000000
unique	1	NaN	NaN	NaN	NaN	NaN
top	R	NaN	NaN	NaN	NaN	NaN
freq	343	NaN	NaN	NaN	NaN	NaN
mean	NaN	77.011268	14.521490	2.468843	27.369318	0.289694
std	NaN	7.692163	2.897229	1.136414	3.689513	0.376962
min	NaN	60.000000	6.000000	1.000000	4.000000	0.000000
25%	NaN	71.000000	12.000000	2.000000	27.000000	0.000000
50%	NaN	77.000000	14.000000	2.000000	29.000000	0.000000
75%	NaN	82.000000	16.000000	3.000000	30.000000	0.500000
max	NaN	98.000000	23.000000	5.000000	30.000000	2.000000

	eTIV	nWBV	ASF
count	340.000000	342.000000	338.000000
unique	NaN	NaN	NaN
top	NaN	NaN	NaN
freq	NaN	NaN	NaN
mean	1489.447059	0.729456	1.194393
std	174.925303	0.037290	0.134498
min	1123.000000	0.644000	0.876000
25%	1359.750000	0.700000	1.104000
50%	1475.000000	0.729000	1.195500
75%	1597.500000	0.756000	1.291000

```
max     2004.000000    0.837000    1.587000
```

[11]:
```
df.nunique()
```

[11]:
```
Subject ID    150
MRI ID        373
Group           3
Visit           5
MR Delay      197
M/F             2
Hand            1
Age            39
EDUC           12
SES             5
MMSE           18
CDR             4
eTIV          269
nWBV          132
ASF           246
dtype: int64
```

[12]:
```
df.isnull().sum()
```

[12]:
```
Subject ID     0
MRI ID         0
Group          0
Visit         29
MR Delay       7
M/F           28
Hand          30
Age           18
EDUC          24
SES           36
MMSE          21
CDR           14
eTIV          33
nWBV          31
ASF           35
dtype: int64
```

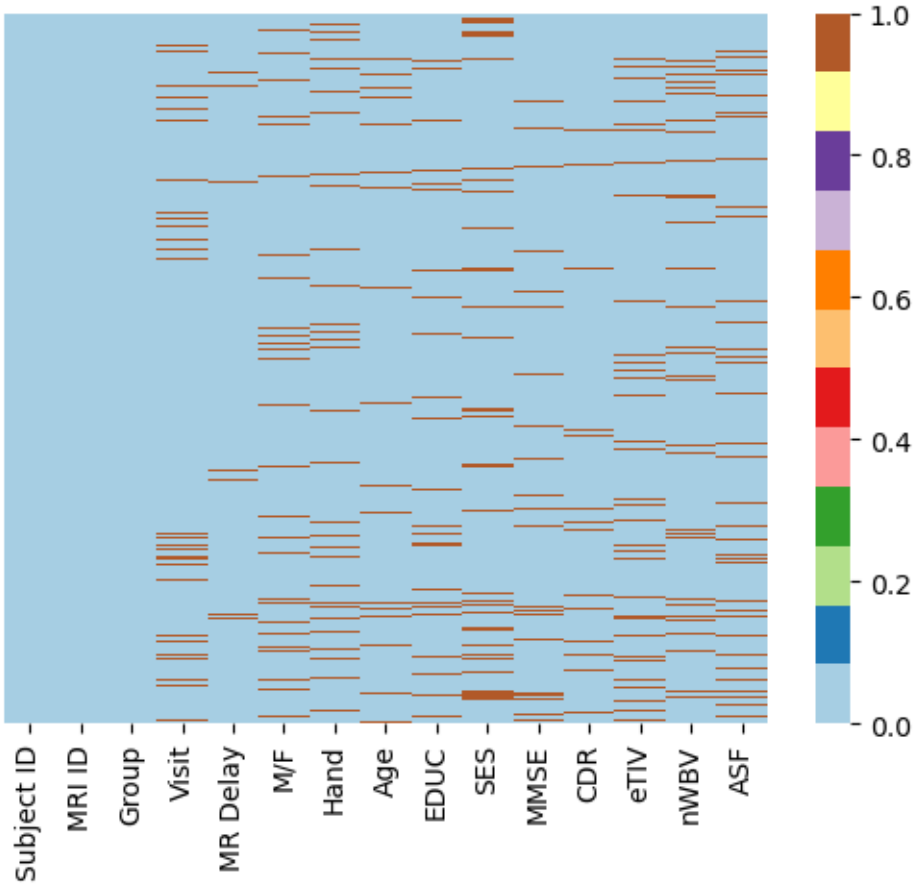
[13]:
```
(df.isnull().sum()/len(df))*100
```

[13]:
```
Subject ID    0.000000
MRI ID        0.000000
Group         0.000000
Visit         7.774799
MR Delay      1.876676
```

```
M/F             7.506702
Hand            8.042895
Age             4.825737
EDUC            6.434316
SES             9.651475
MMSE            5.630027
CDR             3.753351
eTIV            8.847185
nWBV            8.310992
ASF             9.383378
dtype: float64
```

[14]:
```
sns.heatmap(df.isnull(),yticklabels=False,cmap='Paired')
```

[14]: <Axes: >



[15]:
```
df.drop(['Subject ID','MRI ID'],axis=1,inplace=True)
```

[16]:
```
df.columns
```

[16]:
```
Index(['Group', 'Visit', 'MR Delay', 'M/F', 'Hand', 'Age', 'EDUC', 'SES',
       'MMSE', 'CDR', 'eTIV', 'nWBV', 'ASF'],
      dtype='object')
```

[17]:
```
df['Visit'].bfill(axis=0,inplace=True)
df['MR Delay'].fillna(df['MR Delay'].median(),inplace=True)
df['M/F'].ffill(axis=0,inplace=True)
df['Hand'].bfill(axis=0,inplace=True)
df['Age'].fillna(df['Age'].median(),inplace=True)
df['EDUC'].ffill(axis=0,inplace=True)
df['SES'].ffill(axis=0,inplace=True)
df['MMSE'].fillna(df['MMSE'].median(),inplace=True)
df['CDR'].ffill(axis=0,inplace=True)
df['eTIV'].fillna(df['eTIV'].median(),inplace=True)
df['nWBV'].fillna(df['nWBV'].median(),inplace=True)
df['ASF'].fillna(df['ASF'].median(),inplace=True)
```

[18]:
```
df.isnull().sum()
```

[18]:
```
Group       0
Visit       0
MR Delay    0
M/F         0
Hand        0
Age         0
EDUC        0
SES         0
MMSE        0
CDR         0
eTIV        0
nWBV        0
ASF         0
dtype: int64
```

[19]:
```
df['M/F'].value_counts()
```

[19]:
```
F    214
M    159
Name: M/F, dtype: int64
```

[20]:
```
df['Hand'].value_counts()
```

[20]:
```
R    373
Name: Hand, dtype: int64
```

[21]:
```
df['SES'].value_counts()
```

[21]:
```
2.0    108
1.0     94
3.0     83
4.0     81
5.0      7
Name: SES, dtype: int64
```

[22]:
```
df['EDUC'].value_counts()
```

[22]:
```
12.0    104
16.0     80
18.0     64
14.0     32
13.0     26
15.0     18
20.0     13
11.0     12
8.0      10
17.0      8
6.0       3
23.0      3
Name: EDUC, dtype: int64
```

[23]:
```
cat_data=df.select_dtypes(include=object)
num_data=df.select_dtypes(exclude=object)
```

[24]:
```
cat_data
```

[24]:
```
            Group M/F Hand
0    Nondemented   M    R
1    Nondemented   M    R
2       Demented   M    R
3       Demented   M    R
4       Demented   M    R
..           …  ..  …
368     Demented   M    R
369     Demented   M    R
370  Nondemented   F    R
371  Nondemented   F    R
372  Nondemented   F    R

[373 rows x 3 columns]
```

[25]:
```
num_data
```

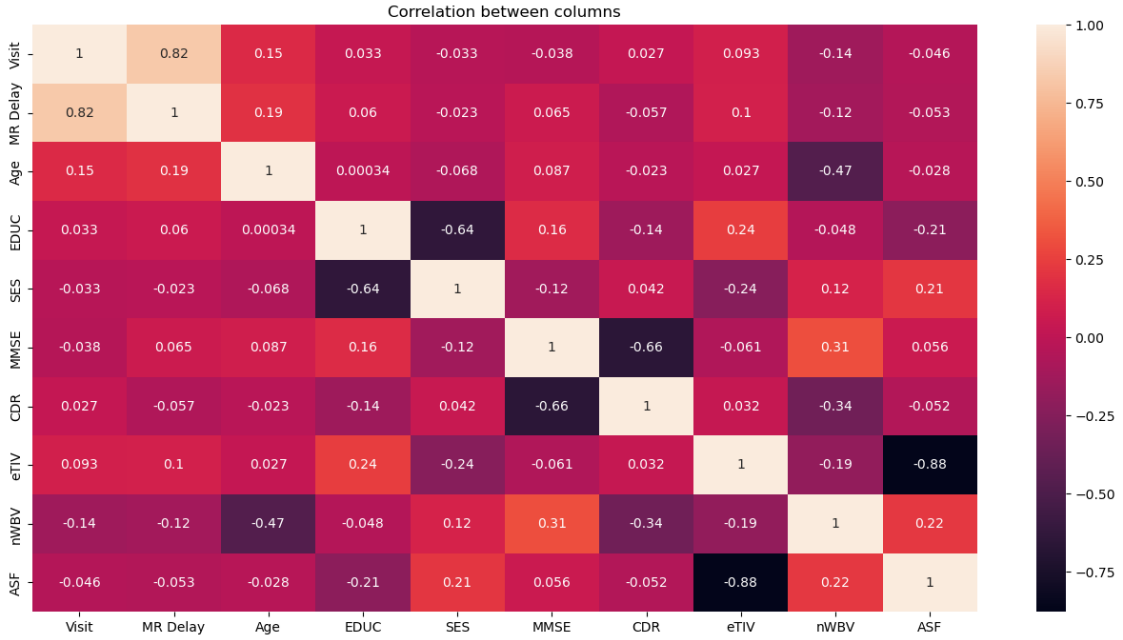
[25]:
```
     Visit  MR Delay   Age  EDUC  SES  MMSE  CDR    eTIV   nWBV     ASF
0      1.0       0.0  87.0  14.0  2.0  27.0  0.0  1987.0  0.696  0.8830
```

```
1      2.0     457.0  88.0  14.0  2.0  30.0  0.0  2004.0  0.681  0.8760
2      1.0       0.0  75.0  12.0  2.0  23.0  0.5  1678.0  0.736  1.0460
3      2.0     560.0  76.0  12.0  2.0  28.0  0.5  1738.0  0.713  1.0100
4      3.0    1895.0  80.0  12.0  2.0  22.0  0.5  1698.0  0.701  1.0340
..     …        …     …     …    …    …     …    …       …      …
368    2.0     842.0  82.0  16.0  1.0  29.0  0.5  1693.0  0.694  1.0370
369    3.0    2297.0  86.0  16.0  1.0  26.0  0.5  1688.0  0.675  1.1955
370    1.0       0.0  61.0  13.0  2.0  30.0  0.0  1319.0  0.801  1.3310
371    3.0     763.0  63.0  13.0  2.0  29.0  0.0  1475.0  0.796  1.3230
372    3.0    1608.0  77.0  13.0  2.0  30.0  0.0  1333.0  0.801  1.3170

[373 rows x 10 columns]
```

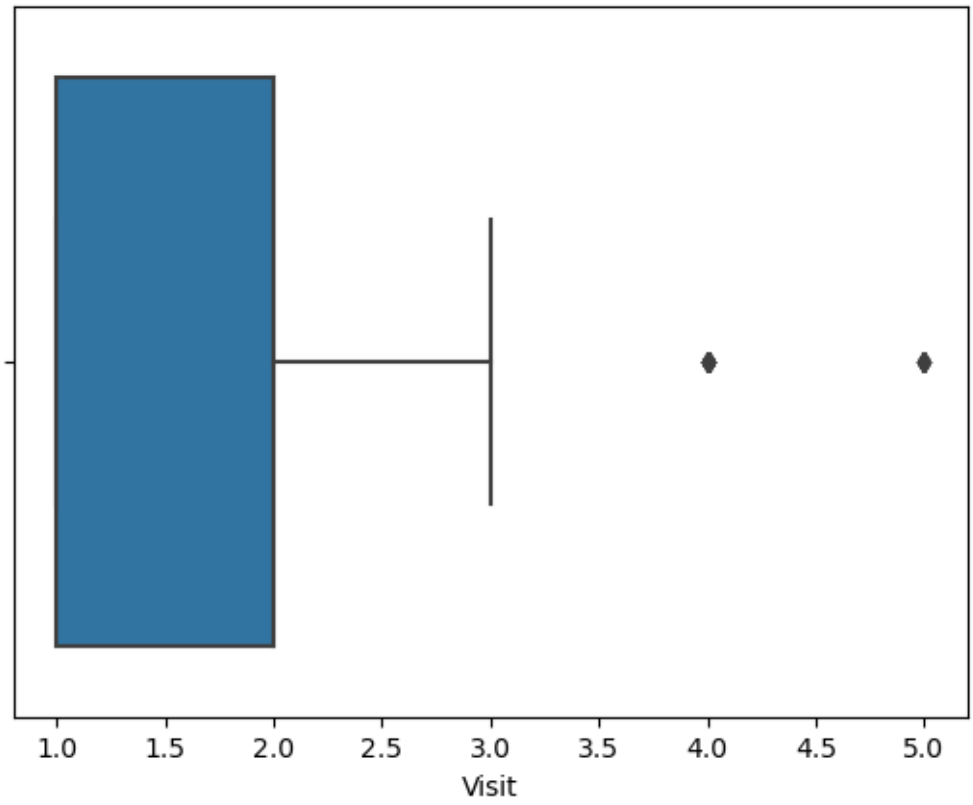
[26]:
```
plt.figure(figsize=(16,8))
sns.heatmap(df.corr(),annot=True)
plt.title('Correlation between columns')
plt.show()
```

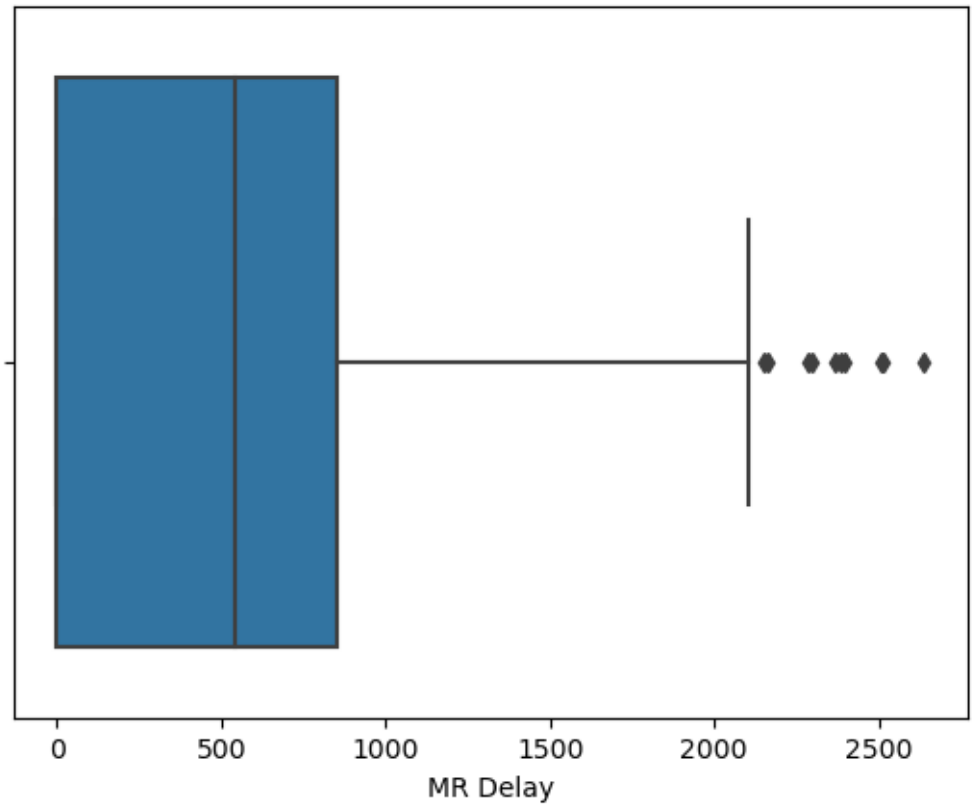
```
C:\Users\prera\AppData\Local\Temp\ipykernel_9388\1066296358.py:2: FutureWarning:
The default value of numeric_only in DataFrame.corr is deprecated. In a future
version, it will default to False. Select only valid columns or specify the
value of numeric_only to silence this warning.
  sns.heatmap(df.corr(),annot=True)
```

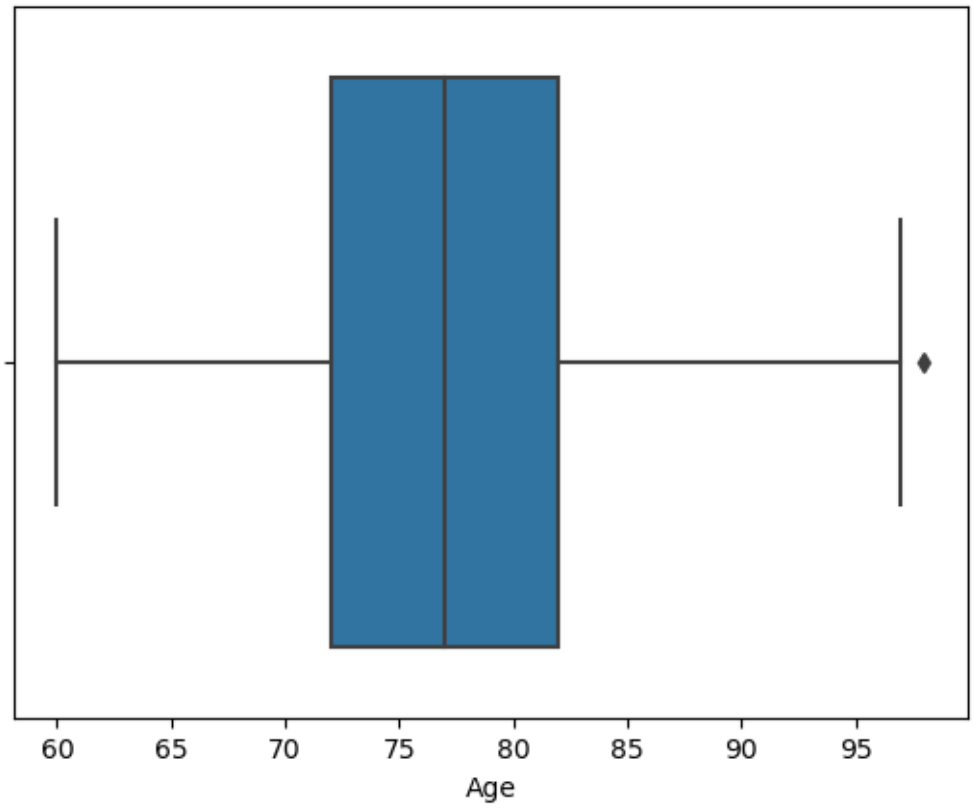


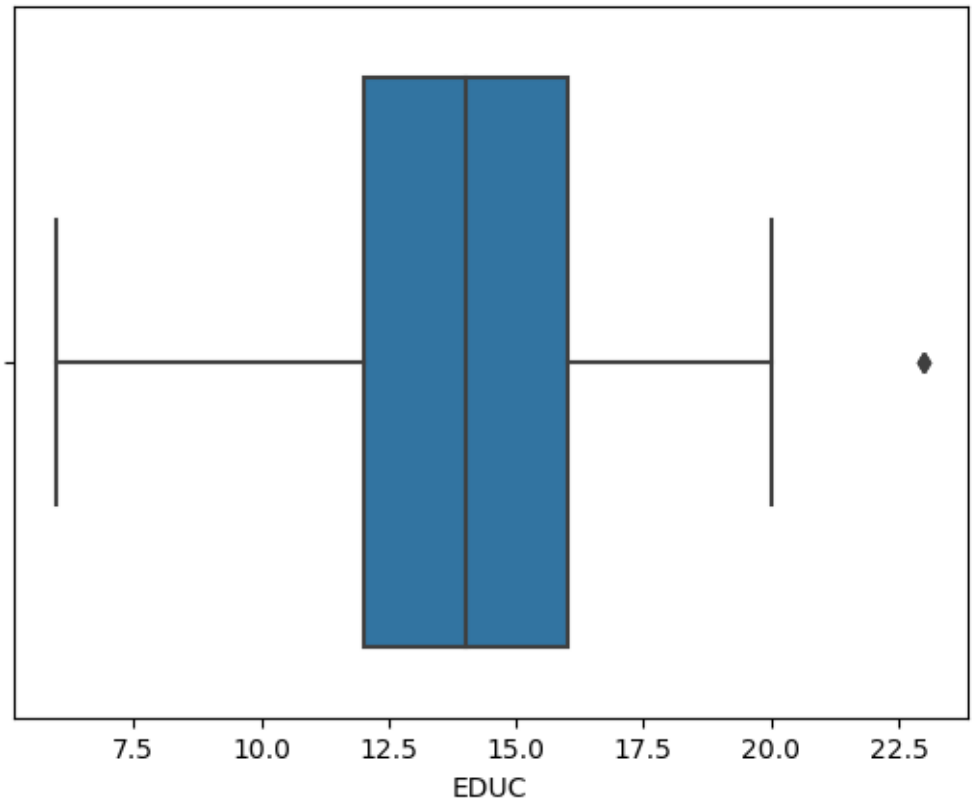
[27]:
```
for i in num_data:
    sns.boxplot(x=df[i])
```

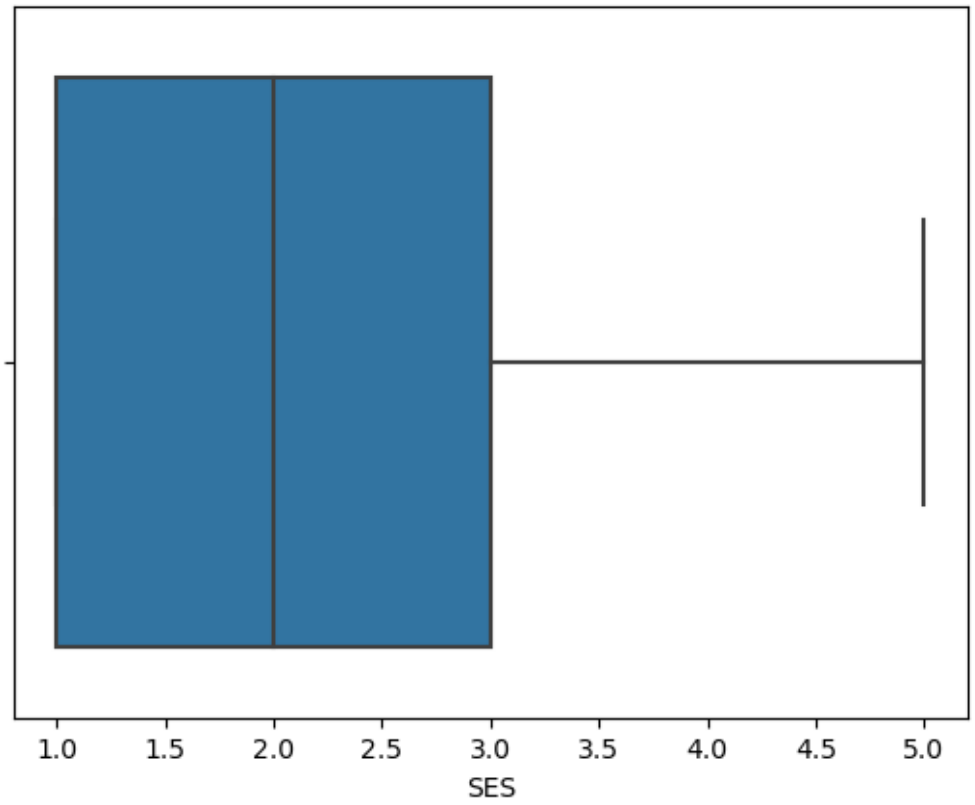
```
    plt.show()
```

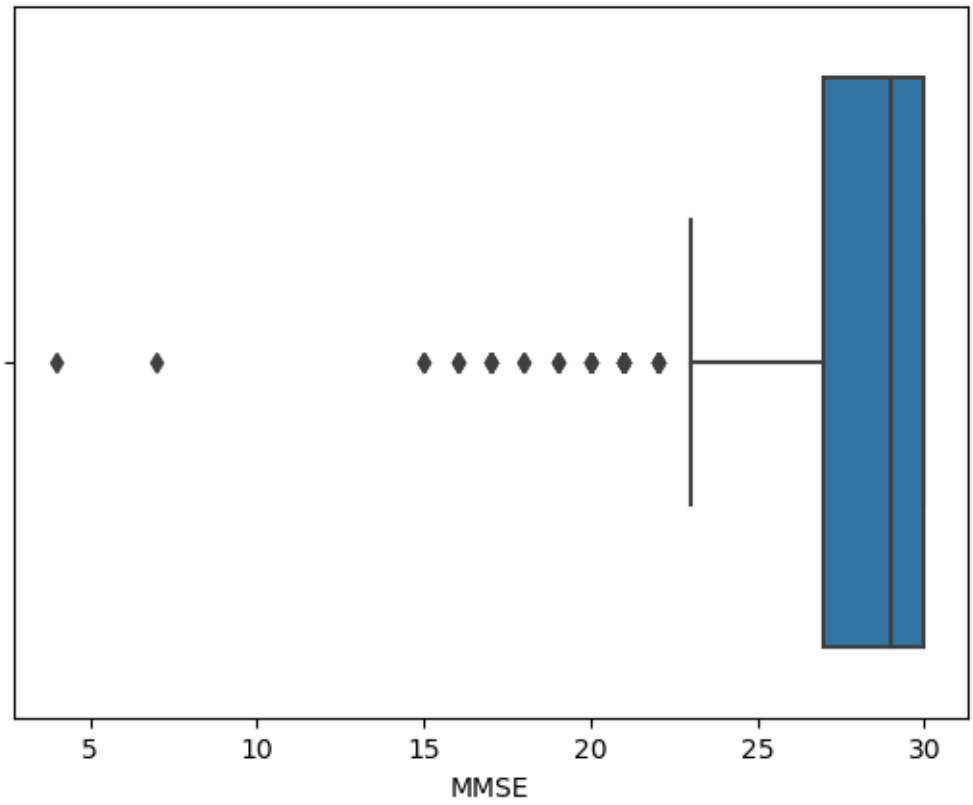


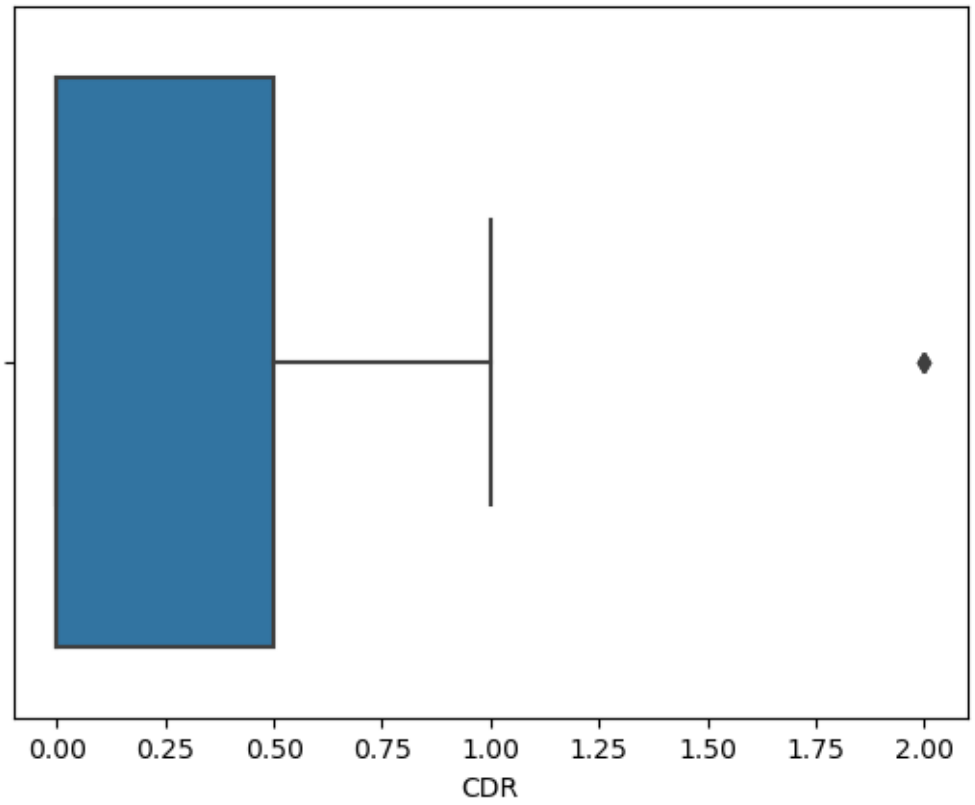

0
500
1000
1500
2000
2500
MR Delay

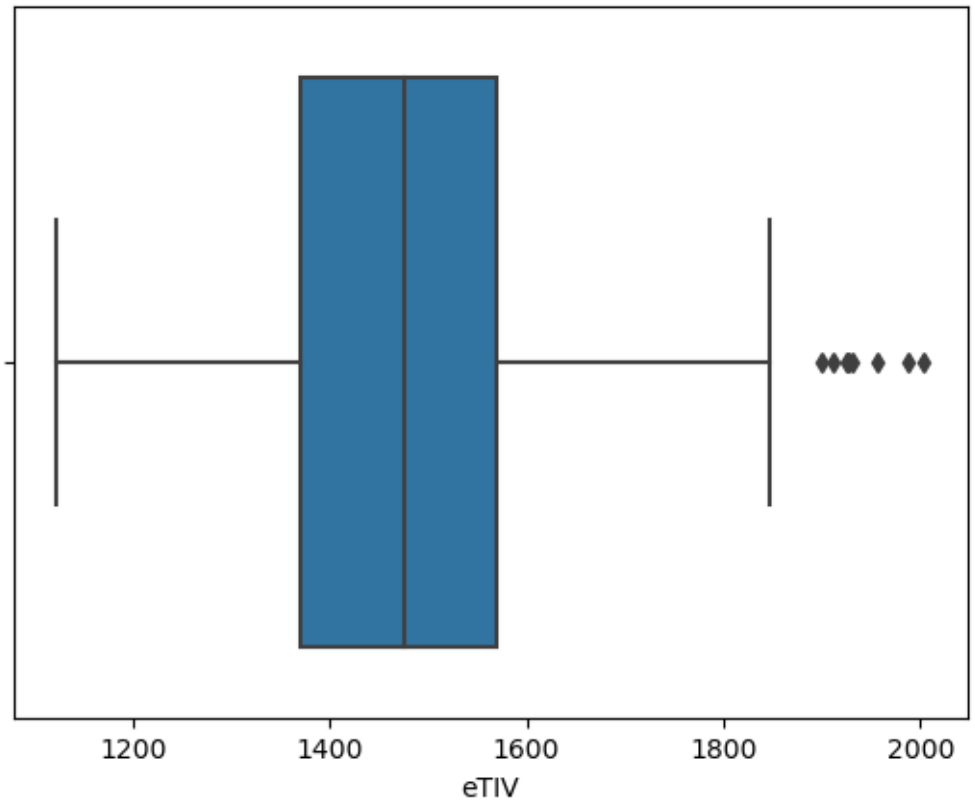

60
65
70
75
80
85
90
95
Age

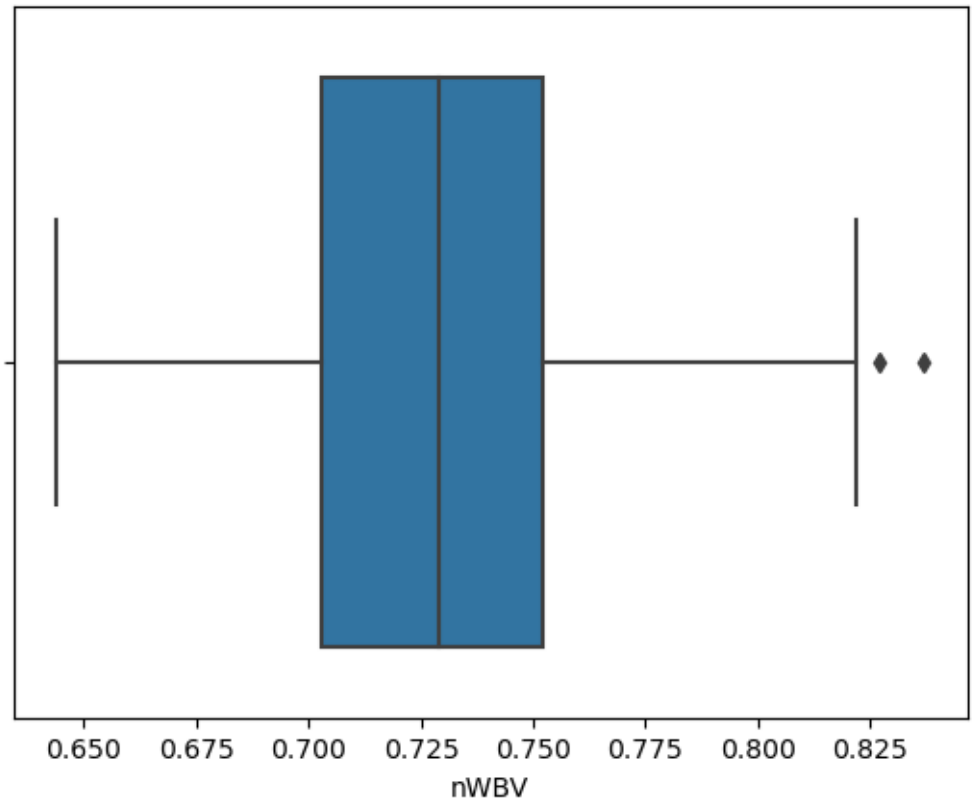

7.5
10.0
12.5
15.0
17.5
20.0
22.5
EDUC

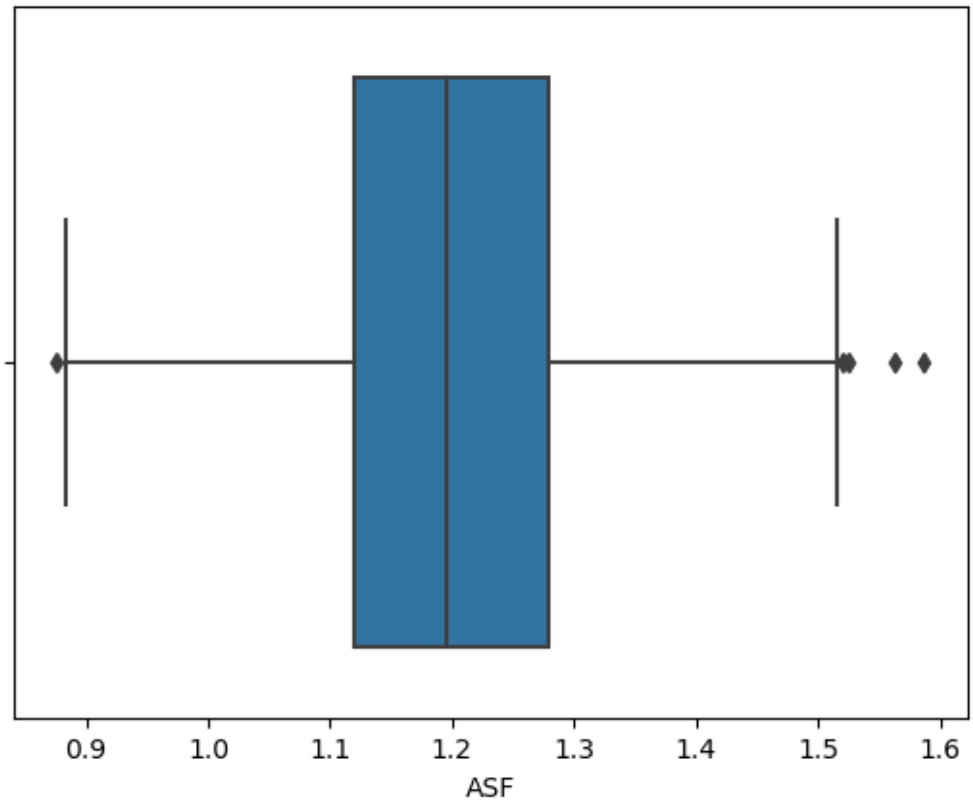

1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
SES


5
10
15
20
25
30
MMSE


0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
CDR

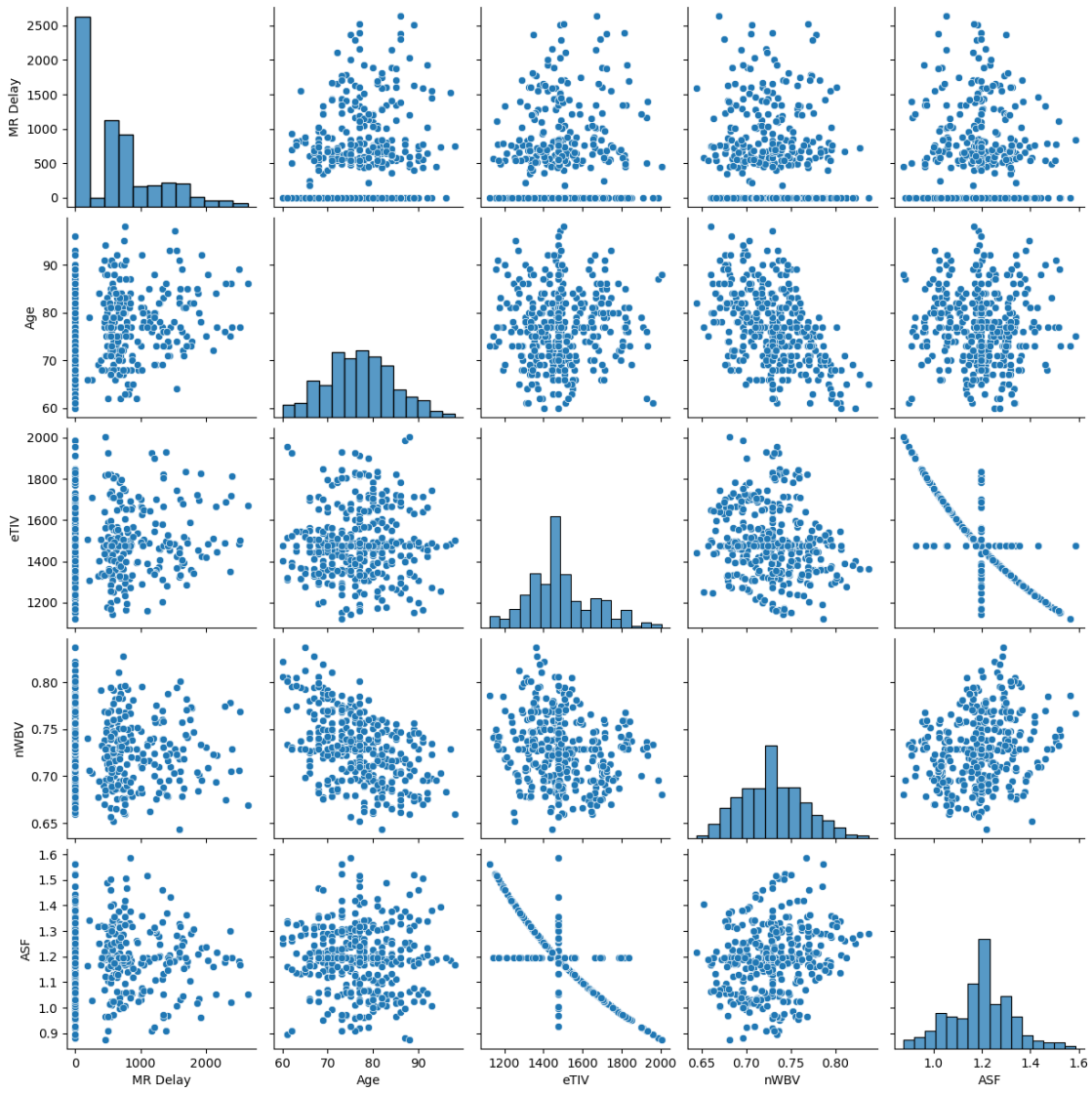

1200
1400
1600
1800
2000
eTIV


0.650
0.675
0.700
0.725
0.750
0.775
0.800
0.825
nWBV



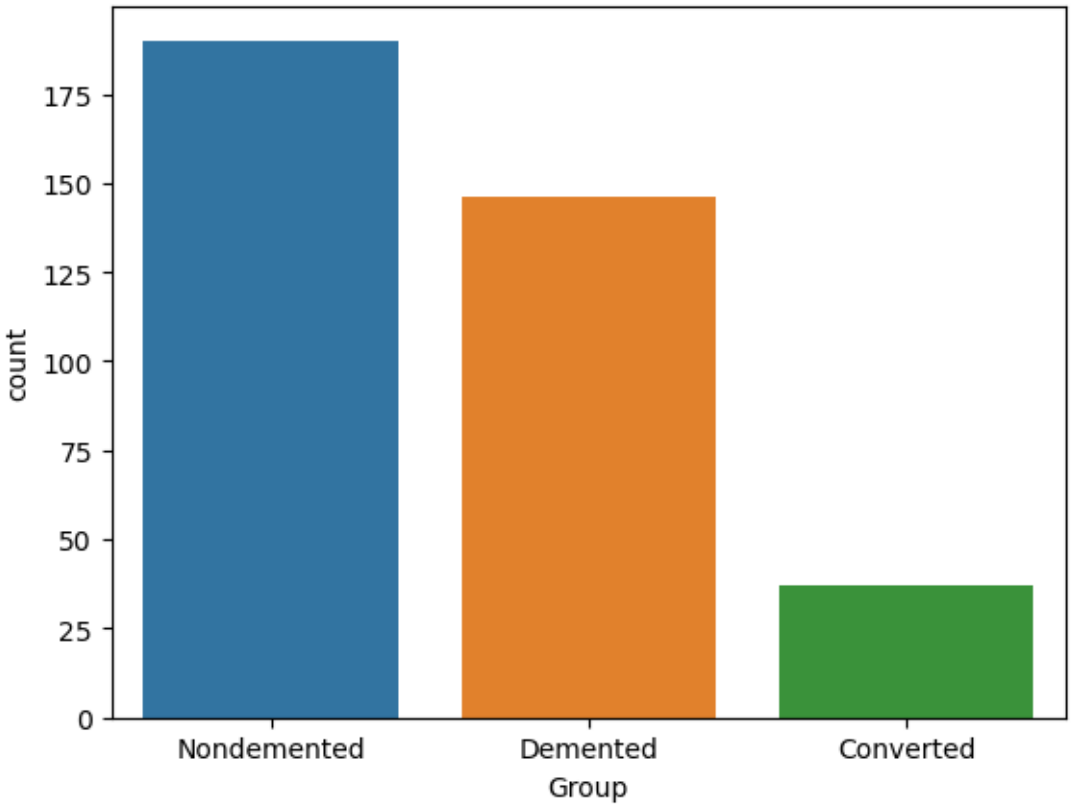
[28]:
```
pair_columns=['MR Delay','Age','eTIV','nWBV','ASF']
sns.pairplot(df[pair_columns])
```

[28]: <seaborn.axisgrid.PairGrid at 0x28285953610>



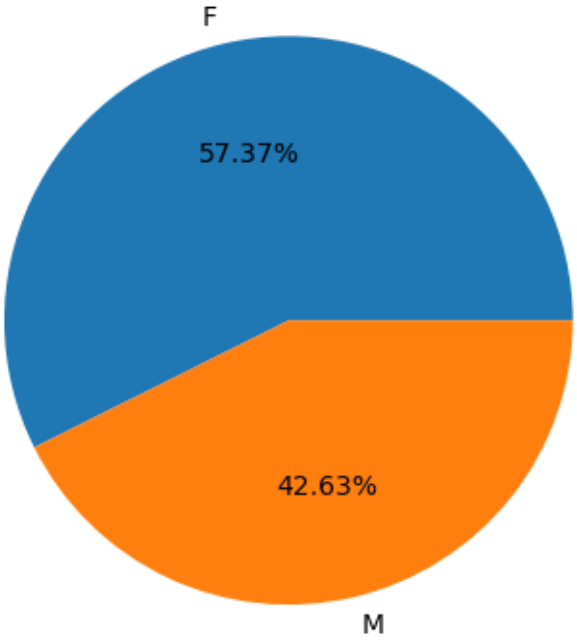
[29]:
```
sns.countplot(x='Group',data=df)
```

[29]: <Axes: xlabel='Group', ylabel='count'>



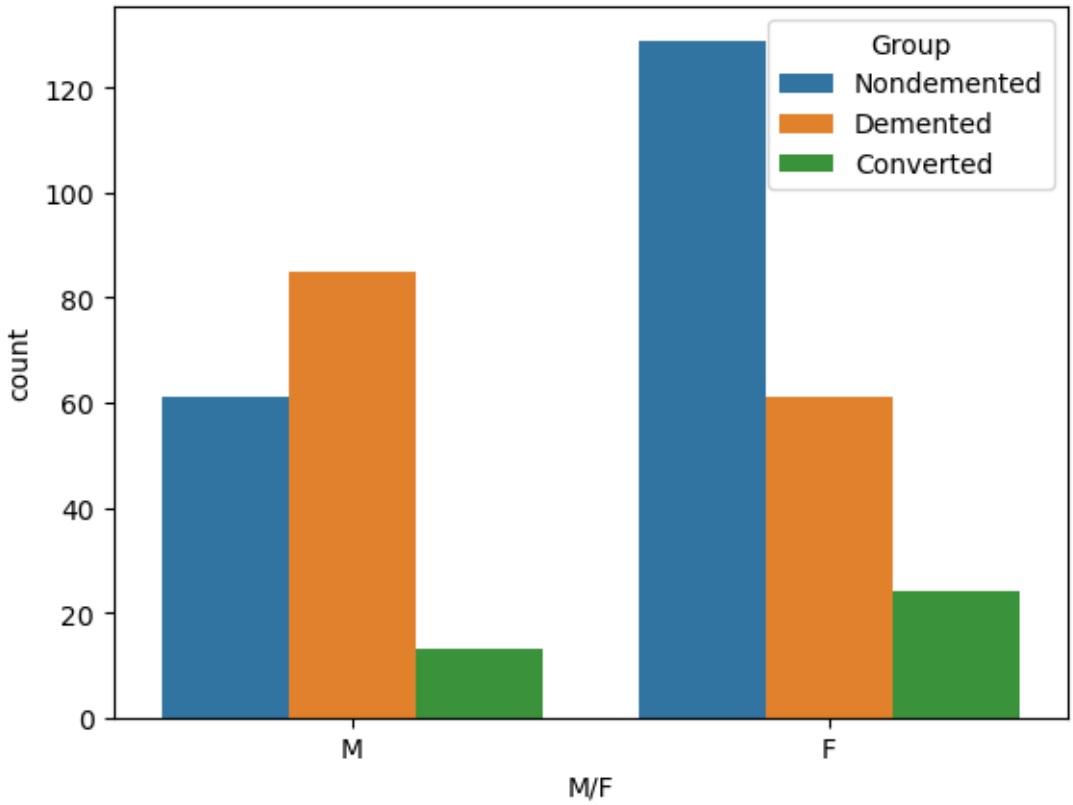
[30]:
```
a=df.groupby('M/F')['M/F'].count()
plt.pie(a,labels=a.index,autopct='%.2f%%')
```

[30]:
```
([<matplotlib.patches.Wedge at 0x28288353710>,
  <matplotlib.patches.Wedge at 0x28288364b10>],
 [Text(-0.25250867084593326, 1.0706256914289047, 'F'),
  Text(0.2525086708459333, -1.0706256914289047, 'M')],
 [Text(-0.13773200227959995, 0.5839776498703116, '57.37%'),
  Text(0.13773200227959997, -0.5839776498703116, '42.63%')])
```



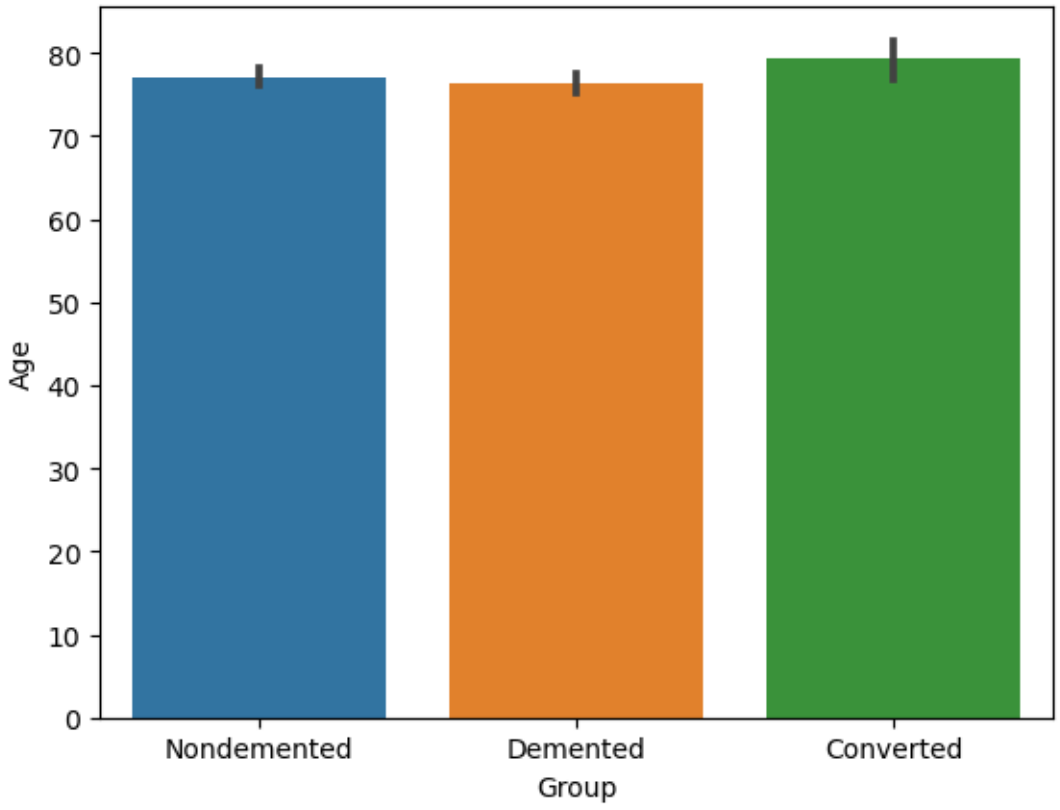
[31]:
```
sns.countplot(x='M/F',hue='Group',data=df)
```

[31]:
```
<Axes: xlabel='M/F', ylabel='count'>
```



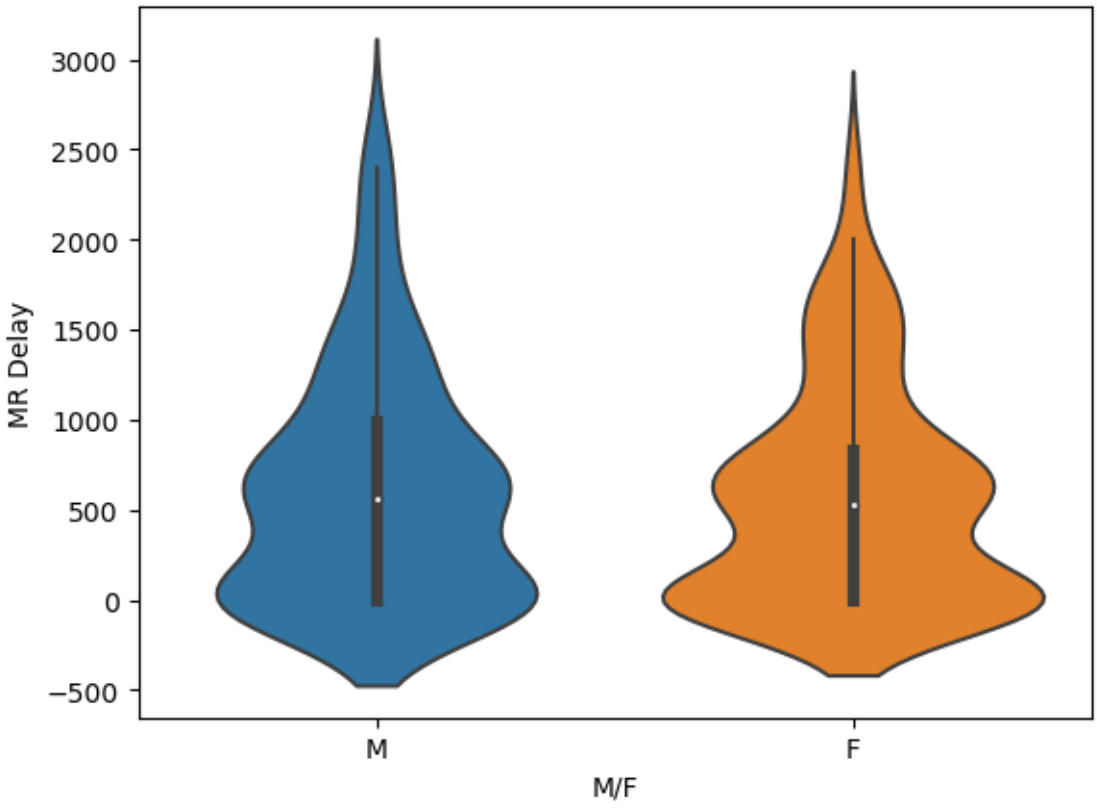
[32]:
```
sns.barplot(x='Group',y='Age',data=df)
```

[32]:
```
<Axes: xlabel='Group', ylabel='Age'>
```



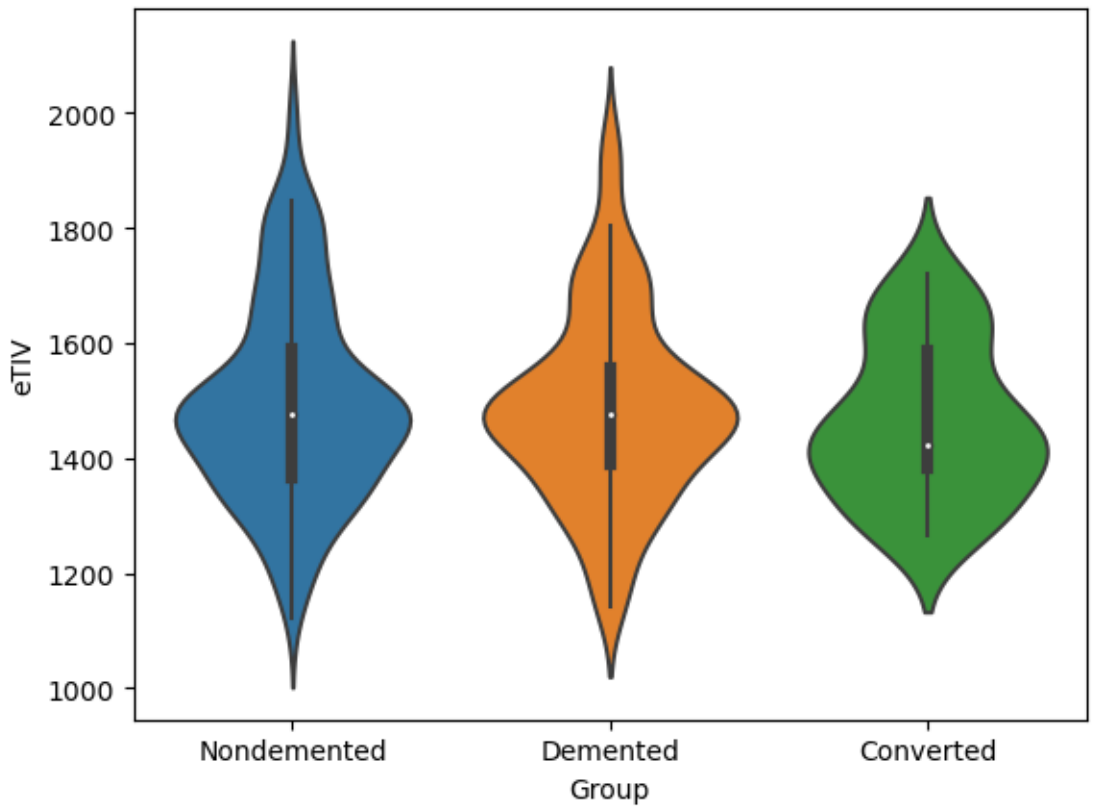
[33]:
```
sns.violinplot(x='M/F',y='MR Delay',data=df)
```

[33]:
```
<Axes: xlabel='M/F', ylabel='MR Delay'>
```



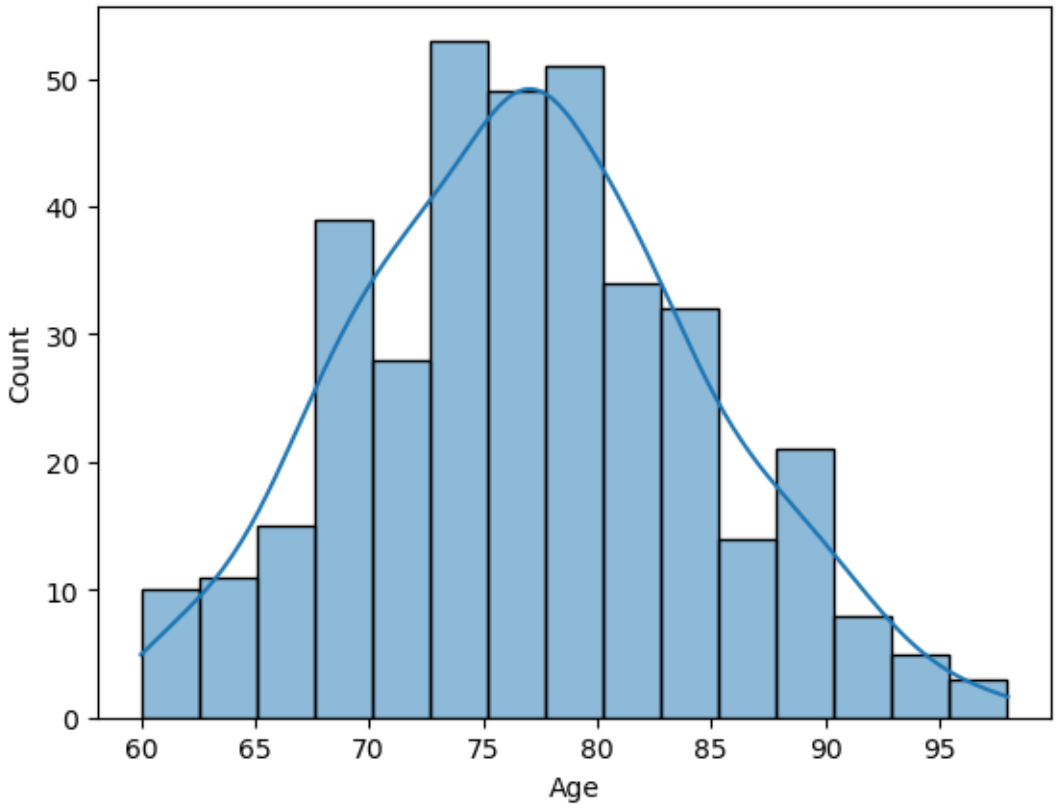
[34]:
```
sns.violinplot(x='Group',y='eTIV',data=df)
```

[34]:
```
<Axes: xlabel='Group', ylabel='eTIV'>
```



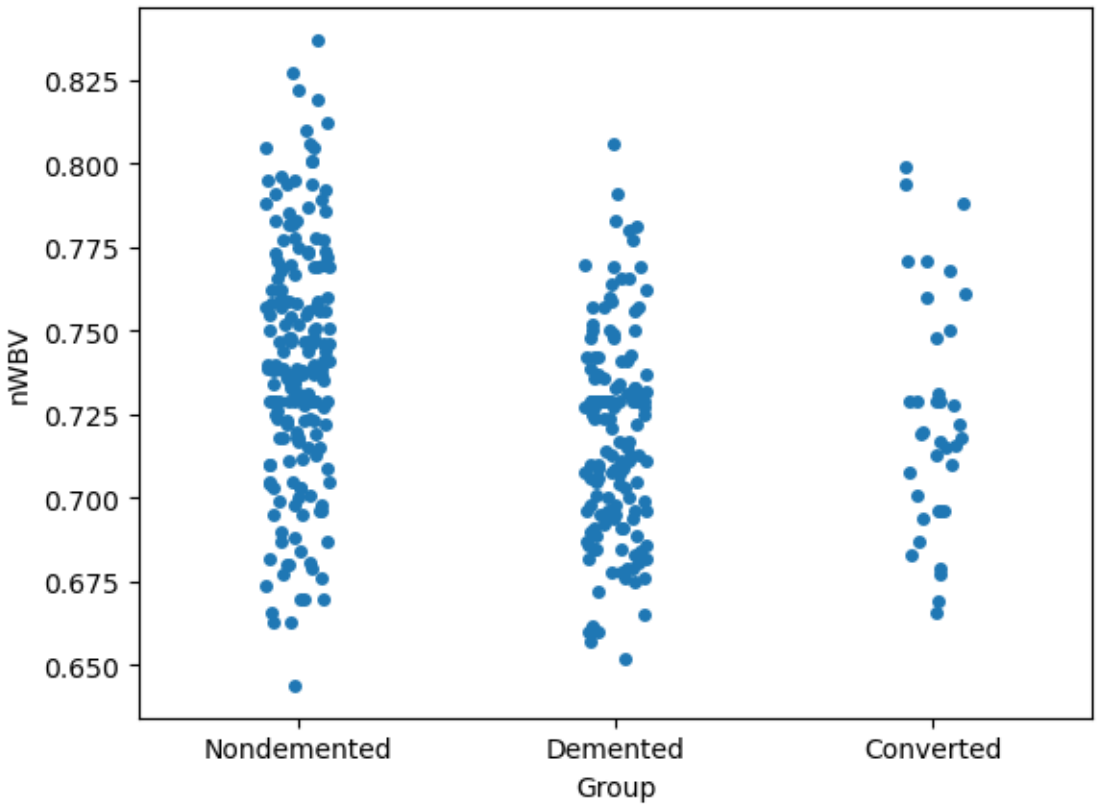
[35]:
```
sns.histplot(x='Age',data=df,bins=15,kde=True)
```

[35]:
```
<Axes: xlabel='Age', ylabel='Count'>
```



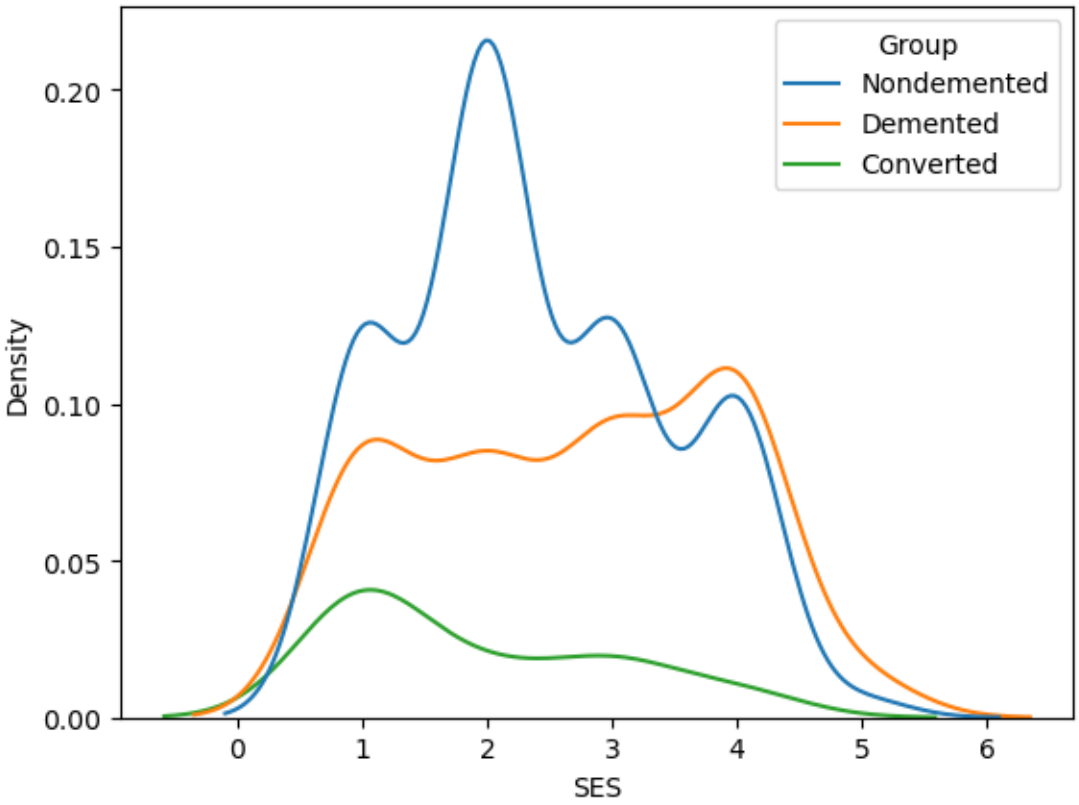
[36]:
```
sns.stripplot(x='Group',y='nWBV',data=df)
```

[36]:
```
<Axes: xlabel='Group', ylabel='nWBV'>
```



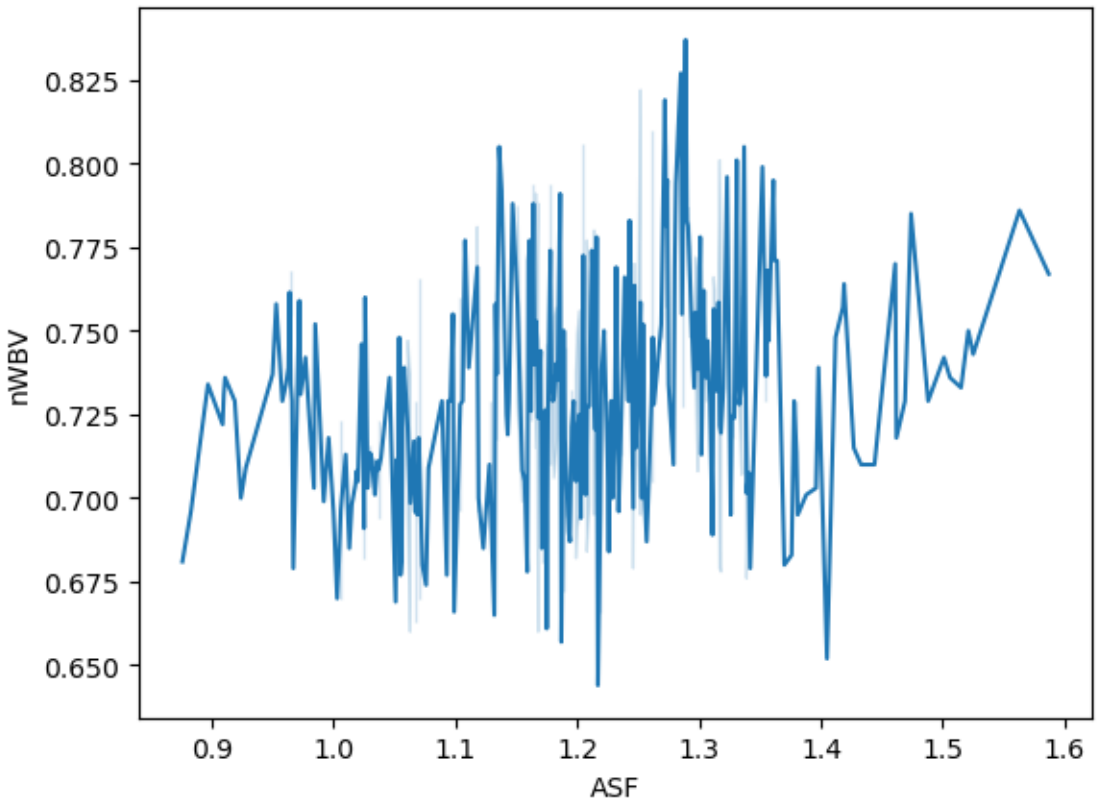
[37]:
```
sns.kdeplot(x='SES',hue='Group',data=df)
```

[37]:
```
<Axes: xlabel='SES', ylabel='Density'>
```



[38]:
```
sns.lineplot(x='ASF',y='nWBV',data=df)
```

[38]:
```
<Axes: xlabel='ASF', ylabel='nWBV'>
```



[39]:
```
df.replace({'Group':{'Demented':1,'Nondemented':0,'Converted':2}},inplace=True)
df.replace({'M/F':{'M':1,'F':0}},inplace=True)
df.replace({'Hand':{'R':1}},inplace=True)
```

[40]:
```
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report,␣
 ↪confusion_matrix, roc_curve, auc
```

[41]:
```
numerical_cols=['Visit','MR␣
 ↪Delay','Age','EDUC','SES','MMSE','CDR','eTIV','nWBV','ASF','M/F','Hand']
scaler=StandardScaler()
scaled_cols=pd.DataFrame(scaler.
 ↪fit_transform(df[numerical_cols]),columns=scaler.
 ↪get_feature_names_out(numerical_cols))
```

[42]:
```
scaled_cols
```

[42]:

	Visit	MR Delay	Age	EDUC	SES	MMSE	CDR	\
0	-0.935945	-0.940452	1.333025	-0.193431	-0.404523	-0.128136	-0.773771	

```
1    0.144883 -0.212318  1.466471 -0.193431 -0.404523  0.705490 -0.773771
2   -0.935945 -0.940452 -0.268322 -0.883862 -0.404523 -1.239637  0.562417
3    0.144883 -0.048209 -0.134877 -0.883862 -0.404523  0.149740  0.562417
4    1.225711  2.078836  0.398906 -0.883862 -0.404523 -1.517512  0.562417
..        …        …        …        …        …        …        …
368  0.144883  0.401099  0.665797  0.496999 -1.281775  0.427615  0.562417
369  1.225711  2.719339  1.199579  0.496999 -1.281775 -0.406011  0.562417
370 -0.935945 -0.940452 -2.136561 -0.538646 -0.404523  0.705490 -0.773771
371  1.225711  0.275229 -1.869670 -0.538646 -0.404523  0.427615 -0.773771
372  1.225711  1.621561 -0.001431 -0.538646 -0.404523  0.705490 -0.773771

          eTIV      nWBV       ASF       M/F  Hand
0    2.990364 -0.937282 -2.436553  1.160134   0.0
1    3.092275 -1.357988 -2.491307  1.160134   0.0
2    1.137989  0.184599 -1.161556  1.160134   0.0
3    1.497673 -0.460483 -1.443150  1.160134   0.0
4    1.257883 -0.797047 -1.255421  1.160134   0.0
..        …        …        …        …    …
368  1.227910 -0.993377 -1.231954  1.160134   0.0
369  1.197936 -1.526270  0.007843  1.160134   0.0
370 -1.014124  2.007657  1.067733 -0.861969   0.0
371 -0.078944  1.867422  1.005156 -0.861969   0.0
372 -0.930198  2.007657  0.958224 -0.861969   0.0

[373 rows x 12 columns]
```

[43]:
```
x=scaled_cols
y=df['Group']
```

[44]:
```
x.dtypes
```

[44]:
```
Visit       float64
MR Delay    float64
Age         float64
EDUC        float64
SES         float64
MMSE        float64
CDR         float64
eTIV        float64
nWBV        float64
ASF         float64
M/F         float64
Hand        float64
dtype: object
```

1 SVM

[45]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=5)
```

[46]:
```
svcm = SVC(kernel='linear')
svcm.fit(x_train,y_train)
```

[46]:
```
SVC(kernel='linear')
```

[47]:
```
y_pred=svcm.predict(x_test)
```

[48]:
```
acc=accuracy_score(y_test,y_pred)
acc
```

[48]:
```
0.8933333333333333
```

[49]:
```
print (classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.95      1.00      0.98        41
           1       0.87      0.93      0.90        28
           2       0.00      0.00      0.00         6

    accuracy                           0.89        75
   macro avg       0.61      0.64      0.62        75
weighted avg       0.84      0.89      0.87        75
```

[50]:
```
cm= confusion_matrix(y_test,y_pred)
print(cm)
```

```
[[41  0  0]
 [ 0 26  2]
 [ 2  4  0]]
```

[51]:
```
from sklearn.preprocessing import label_binarize
from sklearn.multiclass import OneVsRestClassifier
```

[52]:
```
yb=label_binarize(y, classes=[0,1,2])
```

[53]:
```
nc = yb.shape[1]
```

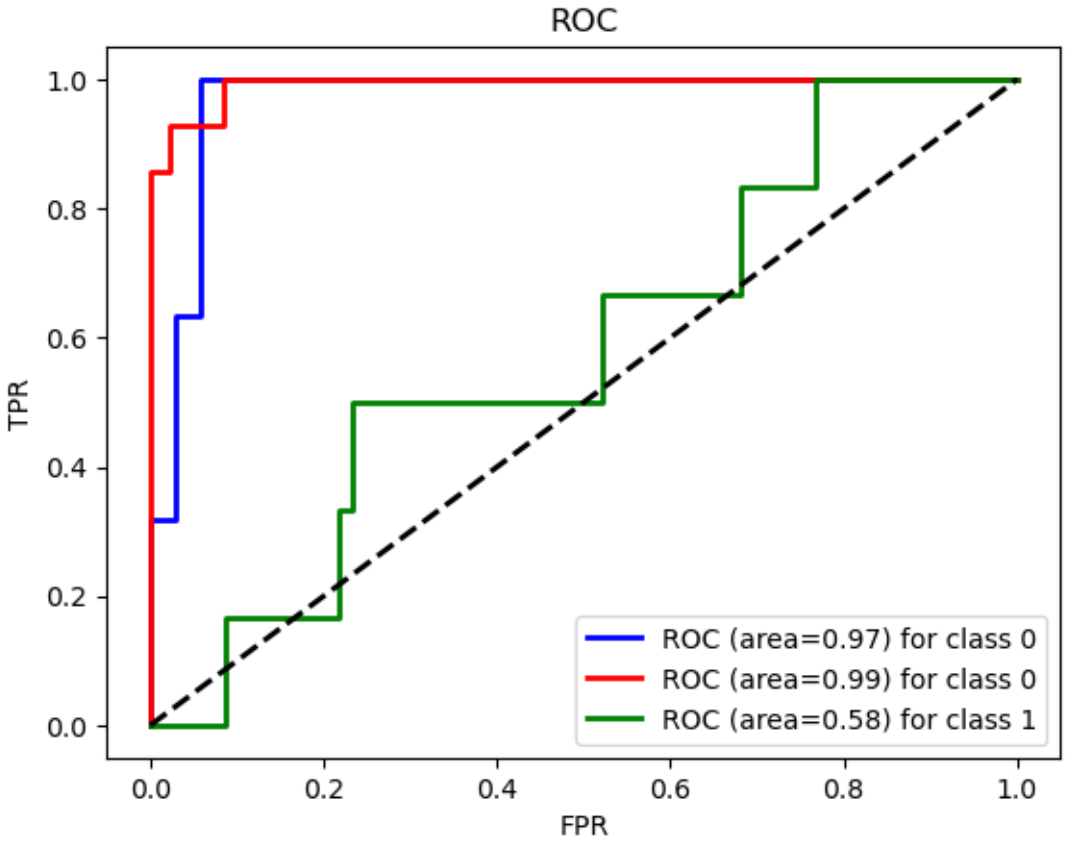
[54]:
```
classifier =␣
↪OneVsRestClassifier(SVC(kernel='linear',probability=True,random_state=42))
```

[55]:
```
y_score=classifier.fit(x_train,y_train).decision_function(x_test)
```

[56]:
```
fpr=dict()
tpr=dict()
roc_auc=dict()

for i in range (nc):
    fpr[i],tpr[i],_=roc_curve(y_test == i, y_score[:,i])
    roc_auc[i]=auc(fpr[i],tpr[i])
```

[57]:
```
plt.figure()
color=['blue','red','green']
for i, color in zip(range(nc),color):
    plt.plot(fpr[i],tpr[i],color=color, lw=2, label='ROC (area={:.2f}) for␣
 ↪class {}'.format(roc_auc[i],df['Group'][i]))
plt.plot([0,1],[0,1],'k--',lw=2)
plt.xlabel('FPR')
plt.ylabel('TPR')
plt.title('ROC')
plt.legend(loc='lower right')
plt.show()
```



[58]:
```
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
from scipy.stats import uniform
```

[59]:
```
param_grid={'C':[0.1,1,10,100,1000],
            'gamma':[1,0.1,0.01,0.001,0.0001],
            'kernel':['linear','rbf','poly','sigmoid']}
```

[60]:
```
svcm=SVC()
grid=GridSearchCV(svcm,param_grid,cv=5)
grid.fit(x_train,y_train)
```

[60]:
```
GridSearchCV(cv=5, estimator=SVC(),
             param_grid={'C': [0.1, 1, 10, 100, 1000],
                         'gamma': [1, 0.1, 0.01, 0.001, 0.0001],
                         'kernel': ['linear', 'rbf', 'poly', 'sigmoid']})
```

[125]:
```
best_param=grid.best_params_
best_model=grid.best_estimator_
y_pred_1=best_model.predict(x_test)
```

```
C:\Users\prera\anaconda3\Lib\site-packages\sklearn\base.py:439: UserWarning: X
does not have valid feature names, but SVC was fitted with feature names
  warnings.warn(
```

[126]:
```
print (classification_report(y_test,y_pred_1))
```

```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        47
           1       0.85      0.85      0.85        40
           2       0.09      0.71      0.16         7

    accuracy                           0.41        94
   macro avg       0.31      0.52      0.34        94
weighted avg       0.37      0.41      0.37        94


C:\Users\prera\anaconda3\Lib\site-
packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\prera\anaconda3\Lib\site-
packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\prera\anaconda3\Lib\site-
```

```
packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

[127]:
```
cm= confusion_matrix(y_test,y_pred_1)
print(cm)
```

```
[[ 0  4 43]
 [ 0 34  6]
 [ 0  2  5]]
```

[64]:
```
param_dist={'C':uniform(loc=0,scale=10),
            'kernel':['linear','rbf','poly','sigmoid']}
```

[65]:
```
n_iter_search=20
random_search = RandomizedSearchCV(svcm, param_distributions=param_dist,␣
 ↪n_iter=n_iter_search, cv=5, n_jobs=-1, random_state=42)
random_search.fit(x_train,y_train)
```

[65]:
```
RandomizedSearchCV(cv=5, estimator=SVC(), n_iter=20, n_jobs=-1,
                   param_distributions={'C':
<scipy.stats._distn_infrastructure.rv_continuous_frozen object at
0x000002828A679C50>,
                                        'kernel': ['linear', 'rbf', 'poly',
                                                   'sigmoid']},
                   random_state=42)
```

[128]:
```
best_param = random_search.best_params_
best_model = random_search.best_estimator_
y_pred_2=best_model.predict(x_test)
```

```
C:\Users\prera\anaconda3\Lib\site-packages\sklearn\base.py:439: UserWarning: X
does not have valid feature names, but SVC was fitted with feature names
  warnings.warn(
```

[129]:
```
print(classification_report(y_test,y_pred_2))
```

```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        47
           1       0.94      0.82      0.88        40
           2       0.12      1.00      0.21         7

    accuracy                           0.43        94
   macro avg       0.35      0.61      0.36        94
weighted avg       0.41      0.43      0.39        94
```

```
C:\Users\prera\anaconda3\Lib\site-
packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\prera\anaconda3\Lib\site-
packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\prera\anaconda3\Lib\site-
packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

[130]:
```
cm= confusion_matrix(y_test,y_pred_2)
print(cm)
```

```
[[ 0  2 45]
 [ 0 33  7]
 [ 0  0  7]]
```

2 Naive Bayes

[69]:
```
from sklearn import model_selection, naive_bayes, svm,␣
↪metrics,feature_extraction
```

[70]:
```
x=scaled_cols
y=df['Group']
```

[71]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.
↪25,random_state=32)
```

[72]:
```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
x_train = scaler.fit_transform(x_train)
x_test = scaler.transform(x_test)
```

[73]:
```
bayes = naive_bayes.MultinomialNB()
```

[109]:
```
bayes.fit(x_train,y_train)
```

[109]:
```
MultinomialNB()
```

[110]:
```
y_pred_nb=bayes.predict(x_test)
```

[76]:
```
accuracy=metrics.accuracy_score(y_test,y_pred_nb)
accuracy
```

[76]: 0.9042553191489362

[111]:
```
print(metrics.classification_report(y_test, y_pred_nb))
```

```
              precision    recall  f1-score   support

           0       0.90      1.00      0.95        47
           1       0.90      0.95      0.93        40
           2       0.00      0.00      0.00         7

    accuracy                           0.90        94
   macro avg       0.60      0.65      0.63        94
weighted avg       0.84      0.90      0.87        94
```

```
C:\Users\prera\anaconda3\Lib\site-
packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\prera\anaconda3\Lib\site-
packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\prera\anaconda3\Lib\site-
packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

[78]:
```
cm=confusion_matrix(y_test,y_pred_nb)
cm
```

[78]:
```
array([[47,  0,  0],
       [ 2, 38,  0],
       [ 3,  4,  0]], dtype=int64)
```

[79]:
```
yb=label_binarize(y, classes=[0,1,2])
nc = yb.shape[1]
```

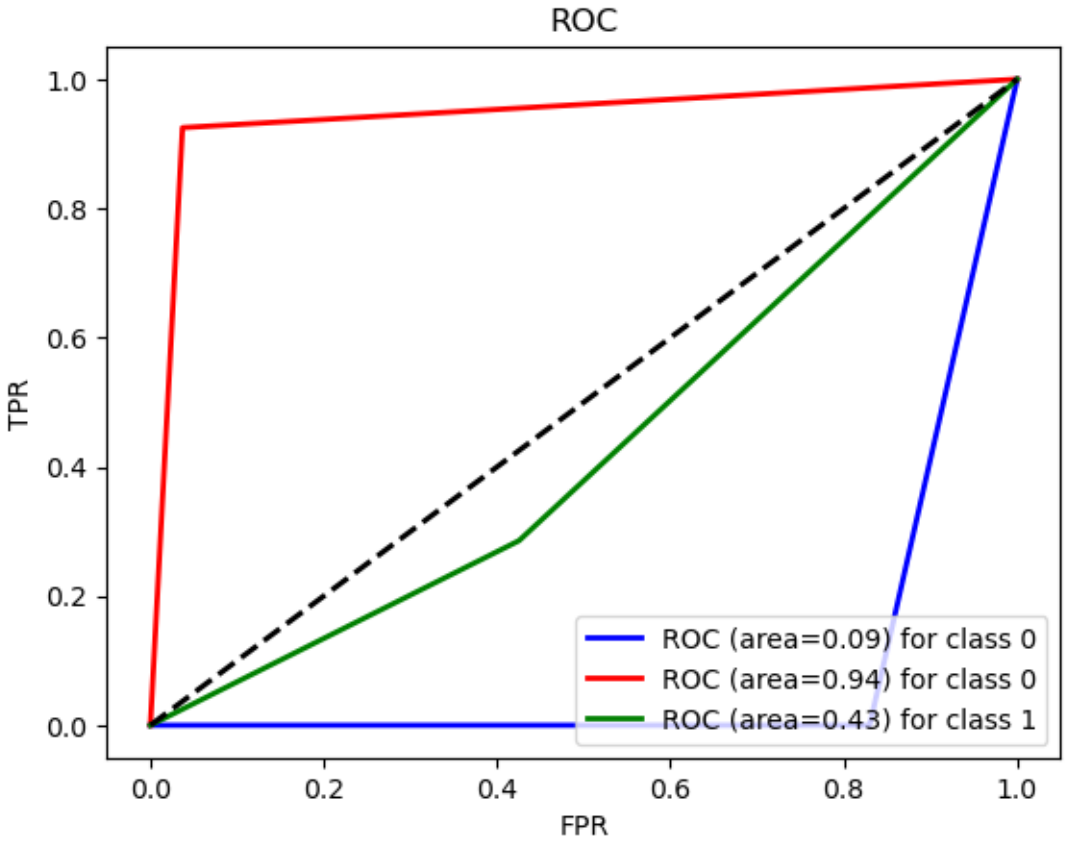
[80]:
```
classifier = OneVsRestClassifier(bayes)
```

[81]:
```
y_score=classifier.fit(x_train,y_train).predict(x_test)
```

[82]:
```
fpr=dict()
tpr=dict()
roc_auc=dict()

for i in range (nc):
    fpr[i],tpr[i],_=roc_curve(y_test == i, y_score)
    roc_auc[i]=auc(fpr[i],tpr[i])
```

[83]:
```
plt.figure()
color=['blue','red','green']
for i, color in zip(range(nc),color):
    plt.plot(fpr[i],tpr[i],color=color, lw=2, label='ROC (area={:.2f}) for␣
 ↪class {}'.format(roc_auc[i],df['Group'][i]))
plt.plot([0,1],[0,1],'k--',lw=2)
plt.xlabel('FPR')
plt.ylabel('TPR')
plt.title('ROC')
plt.legend(loc='lower right')
plt.show()
```



2.1 Grid Search & Randomized Search

[84]:
```
param_grid = {
'alpha': [0.1, 1, 10, 100],
'fit_prior': [True, False]
}
```

[85]:
```
bayes = naive_bayes.MultinomialNB()
grid_search = GridSearchCV(bayes, param_grid, cv=5)
grid_search.fit(x_train, y_train)
```

[85]:
```
GridSearchCV(cv=5, estimator=MultinomialNB(),
             param_grid={'alpha': [0.1, 1, 10, 100],
                         'fit_prior': [True, False]})
```

[112]:
```
best_param = grid_search.best_params_
best_nb = naive_bayes.MultinomialNB(alpha = best_param['alpha'], fit_prior =␣
 ↪best_param['fit_prior'])
best_nb.fit(x_train, y_train)
y_pred_1 = best_nb.predict(x_test)
```

[87]:
```
print("Best Hyperparameter : ", best_param)
```

```
Best Hyperparameter :  {'alpha': 0.1, 'fit_prior': True}
```

[113]:
```
acc = accuracy_score(y_test, y_pred_1)
acc
```

[113]:
```
0.925531914893617
```

[114]:
```
print (classification_report(y_test,y_pred_1))
```

```
              precision    recall  f1-score   support

           0       0.94      1.00      0.97        47
           1       0.91      1.00      0.95        40
           2       0.00      0.00      0.00         7

    accuracy                           0.93        94
   macro avg       0.62      0.67      0.64        94
weighted avg       0.86      0.93      0.89        94


C:\Users\prera\anaconda3\Lib\site-
packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\prera\anaconda3\Lib\site-
```

```
packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\prera\anaconda3\Lib\site-
packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

[120]:
```
cm=confusion_matrix(y_test,y_pred_1)
cm
```

[120]:
```
array([[47,  0,  0],
       [ 0, 40,  0],
       [ 3,  4,  0]], dtype=int64)
```

[91]:
```
#Randomized Search
param_dist = {
    'alpha': uniform(0.1, 2.0),  # Example: Uniform distribution for alpha
    'fit_prior':[True,False]
}
```

[92]:
```
bayes = naive_bayes.MultinomialNB()
```

[93]:
```
x=scaler.fit_transform(x)
```

[94]:
```
from sklearn.utils.validation import check_non_negative
check_non_negative(x, "MultinomialNB (input x)")
```

[95]:
```
randomized_search = RandomizedSearchCV(bayes, param_distributions=param_dist,␣
 ↪n_iter=10, scoring='accuracy', cv=5)
randomized_search.fit(x, y)  # X is your input data, y is your target labels
```

[95]:
```
RandomizedSearchCV(cv=5, estimator=MultinomialNB(),
                   param_distributions={'alpha':
<scipy.stats._distn_infrastructure.rv_continuous_frozen object at
0x000002828A670910>,
                                        'fit_prior': [True, False]},
                   scoring='accuracy')
```

[96]:
```
best_param = randomized_search.best_params_
print("Best Hyperparameter : ", best_param)
```

```
Best Hyperparameter :  {'alpha': 0.7407188763455518, 'fit_prior': True}
```

[122]:
```
best_nb = naive_bayes.MultinomialNB(alpha = best_param['alpha'], fit_prior =␣
 ↪best_param['fit_prior'])
best_nb.fit(x_train, y_train)
y_pred_2 = best_nb.predict(x_test)
```

[123]:
```
acc = accuracy_score(y_test, y_pred_2)
acc
```

[123]: 0.925531914893617

[124]:
```
print(classification_report(y_test, y_pred_2))
```

```
              precision    recall  f1-score   support

           0       0.94      1.00      0.97        47
           1       0.91      1.00      0.95        40
           2       0.00      0.00      0.00         7

    accuracy                           0.93        94
   macro avg       0.62      0.67      0.64        94
weighted avg       0.86      0.93      0.89        94
```

```
C:\Users\prera\anaconda3\Lib\site-
packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\prera\anaconda3\Lib\site-
packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\prera\anaconda3\Lib\site-
packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

[121]:
```
cm=confusion_matrix(y_test,y_pred_2)
cm
```

[121]:
```
array([[47,  0,  0],
       [ 0, 40,  0],
       [ 3,  4,  0]], dtype=int64)
```

[ ]: