Detection and prediction of Alzheimer’s

BY Akanksha B, Niharika D, Manasi M, Ishita R

Research Paper Analysis

Research Paper	Author	Dataset	Algorithm used
1. Automatic Detection and Classification of Alzheimer's Disease from MRI	Ahmed F. Seddik Micheline Eman	OASIS dataset which is made available by Dr. Randy Buckner at Harvard University	Decision Tree
2. Diagnosis of Alzheimer's Disease Based on Structural MRI Images Using a Regularized Extreme Learning Machine and PCA Features	Ramesh Kumar Lama Jeonghwan Gwak Jeong-Seon Park Sang-Woong Lee	Data used in preparation of this paper were obtained from the Alzheimer's disease neuroimaging initiative database (ADNI) (http://adni.loni.usc.edu/)	SVM

Research Paper	Author	Dataset	Algorithm used
3. Kaggle	Hyunseok Choi	MRI related data that was generated by the Open Access Series of Imaging Studies (OASIS)	Decision Tree
4. Alzheimer Disease Prediction using Machine Learning Algorithms	J.Neelaveni M.S.Geetha Devasana	MRI related data that was generated by the Open Access Series of Imaging Studies (OASIS)	Decision Tree and SVM

Workflow of Project

01 **Dataset Preprocessing**

- Remove unwanted features
- Handle null values in dataset
- Standardise the values of dataset to make the input consistent throughout

02 **Useful Feature Extraction**

- Use correlation matrix to find highly correlated features w.r.t target output variable
- Use library functions to find top 10 important features

03 **Train the Model to the improvised dataset**

- Split input dataset to training and test data
- Apply the model via scikit learn machine learning library

04 **Check model's accuracy**

- Check the accuracy of model via confusion matrix and other statistical parameters

05 **Apply trained model to a test dataset**

- Finally apply the trained model to a set of test data to see its functionality

Overview of data set

- MRI related data that was generated by the Open Access Series of Imaging Studies (OASIS) project that is available both, on their website and kaggle that can be utilized for the purpose of training various machine learning models to identify patients with mild to moderate dementia.

- Description

  We will be using the longitudinal MRI data.

  The dataset consists of a longitudinal MRI data of 150 subjects aged 60 to 96.

  Each subject was scanned at least once.

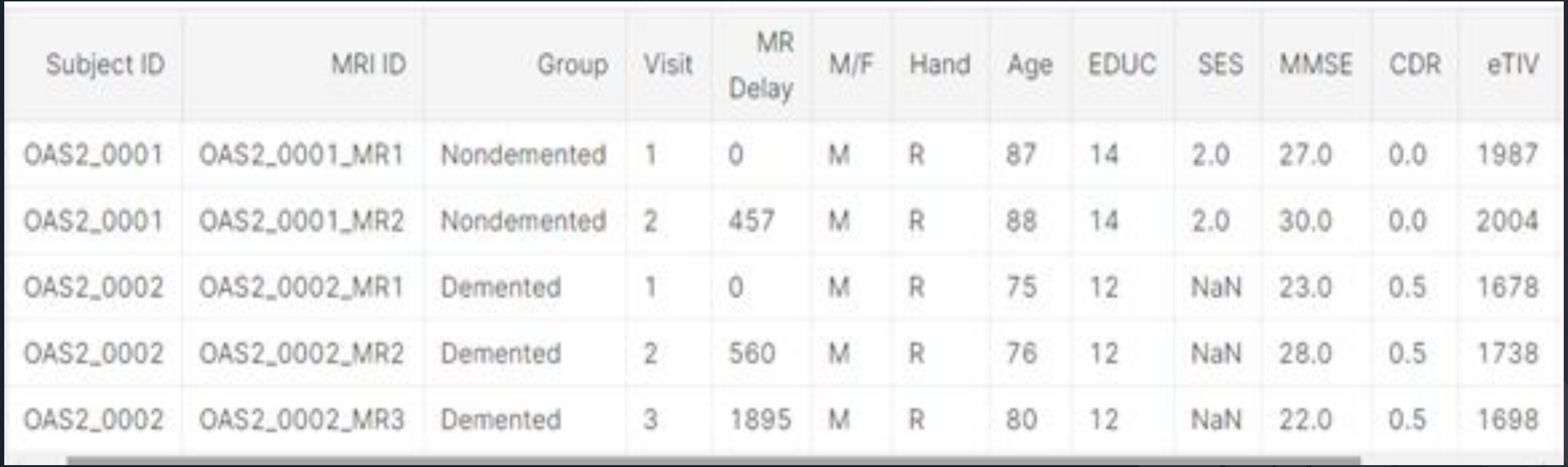
  Everyone is right-handed.

  72 of the subjects were grouped as 'Nondemented' throughout the study.

64 of the subjects were grouped as 'Demented' at the time of their initial visits and remained so throughout the study.

14 subjects were grouped as 'Nondemented' at the time of their initial visit and were subsequently characterized as 'Demented' at a later visit. These fall under the 'Converted' category.

An snapshot of the dataset to be used.



Subject ID	MRI ID	Group	Visit	MR Delay	M/F	Hand	Age	EDUC	SES	MMSE	CDR	eTIV
OAS2_0001	OAS2_0001_MR1	Nondemented	1	0	M	R	87	14	2.0	27.0	0.0	1987
OAS2_0001	OAS2_0001_MR2	Nondemented	2	457	M	R	88	14	2.0	30.0	0.0	2004
OAS2_0002	OAS2_0002_MR1	Demented	1	0	M	R	75	12	NaN	23.0	0.5	1678
OAS2_0002	OAS2_0002_MR2	Demented	2	560	M	R	76	12	NaN	28.0	0.5	1738
OAS2_0002	OAS2_0002_MR3	Demented	3	1895	M	R	80	12	NaN	22.0	0.5	1698

We will drop unnecessary columns such as the MRI ID, Visit and Hand.

- Algorithm to be used

Decision Tree algorithm

- Simplifies the process of classification as it uses divide and conquer principle and therefore the load is distributed which makes the computation.
- According to the multiple research papers it has the highest accuracy in this case.

STEPS:

Step-1: Begin the tree with the root node, says S, which contains the complete dataset.

Step-2: Find the best attribute in the dataset using Attribute Selection Measure (ASM) using gini index

Gini index is a measure of impurity or purity used while creating a decision tree.

it only creates binary splits

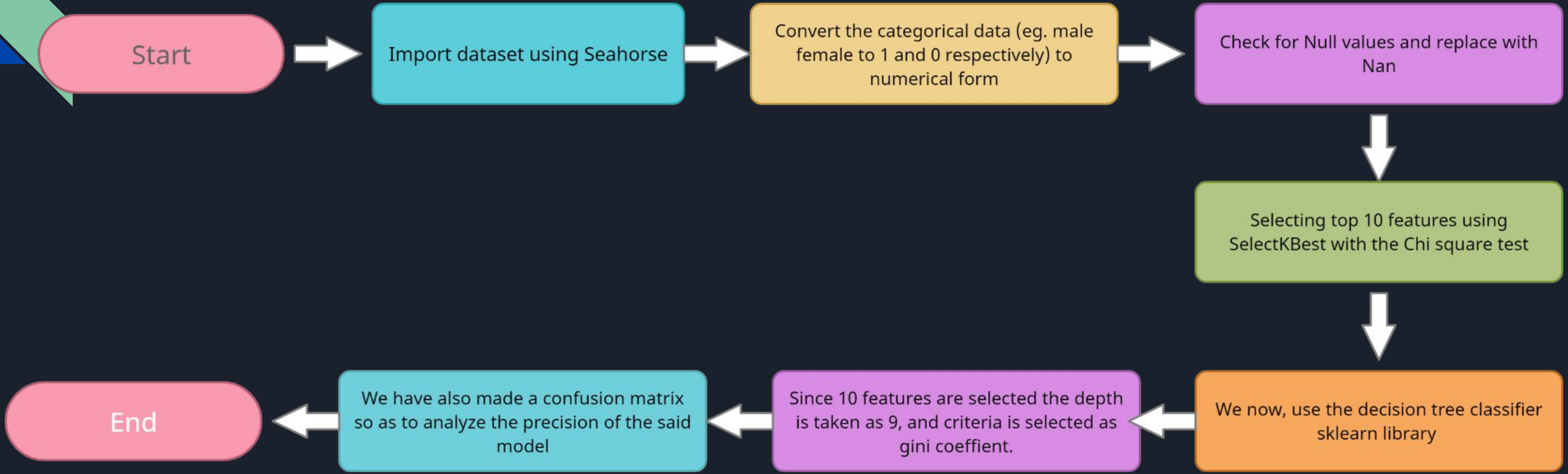
Step-3: Divide the S into subsets that contains possible values for the best attributes.

Step-4: Generate the decision tree node, which contains the best attribute.

Step-5: Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.

Steps to code

Block Diagram



Start

Import dataset using Seahorse

Convert the categorical data (eg. male female to 1 and 0 respectively) to numerical form

Check for Null values and replace with Nan

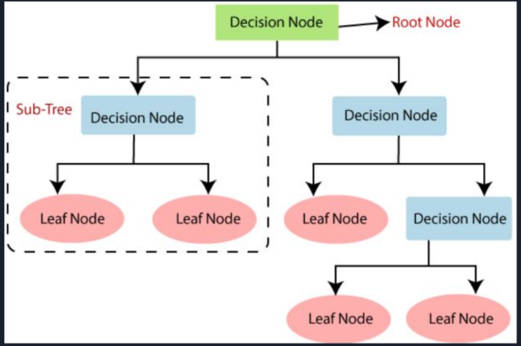
Selecting top 10 features using SelectKBest with the Chi square test

We now, use the decision tree classifier sklearn library

Since 10 features are selected the depth is taken as 9, and criteria is selected as gini coeffient.

We have also made a confusion matrix so as to analyze the precision of the said model

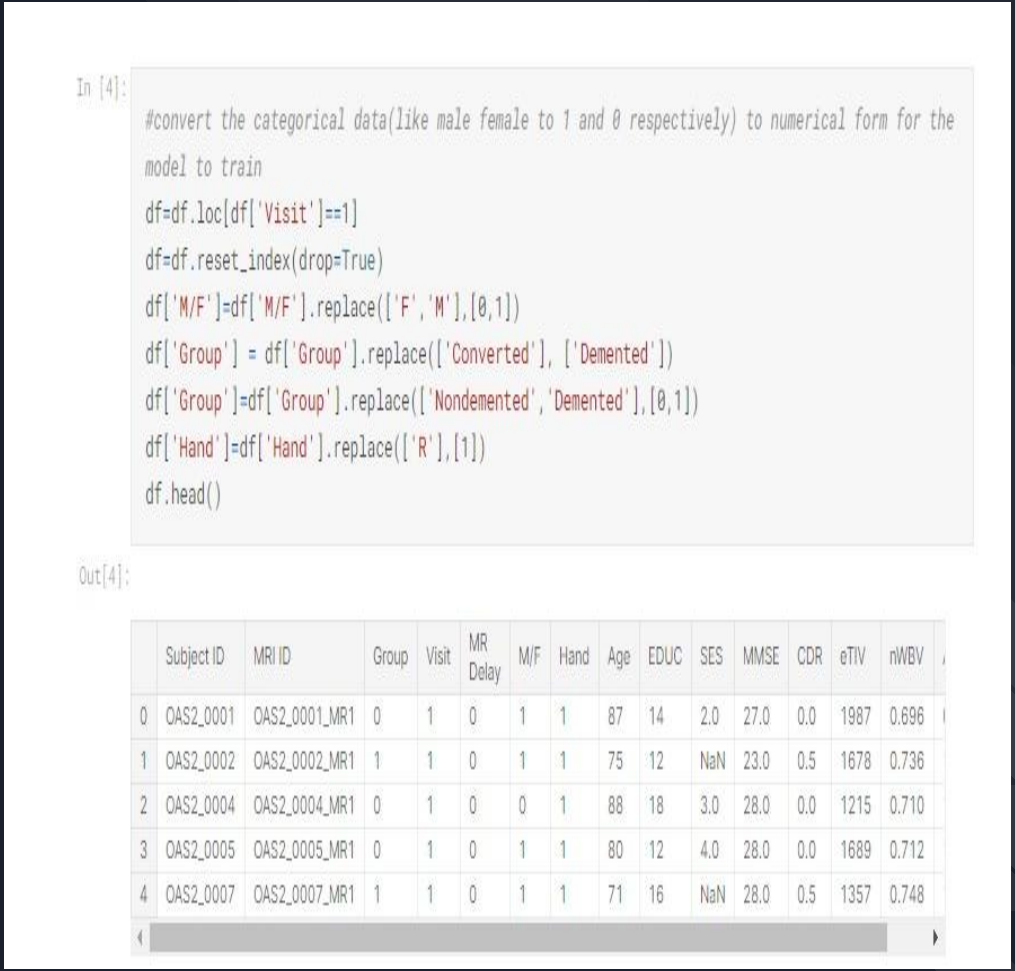
End



1. If ‘Visit’ column >1 , then drop
2. Replace categorical data with numbers

Sr. No.	Category	Replaced by
1	Male/Female	1/0
2	Non-demented/Demented	0/1
3	Hand - Right	1

The columns are replaced and index is updated



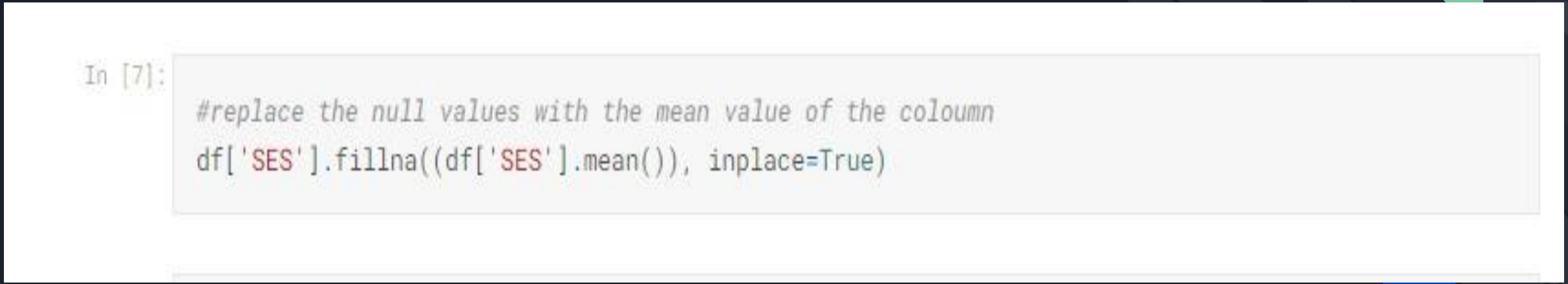
In [4]:

```
#convert the categorical data(like male female to 1 and 0 respectively) to numerical form for the
model to train
df=df.loc[df['Visit']==1]
df=df.reset_index(drop=True)
df['M/F']=df['M/F'].replace(['F','M'],[0,1])
df['Group'] = df['Group'].replace(['Converted'], ['Demented'])
df['Group']=df['Group'].replace(['Nondemented','Demented'],[0,1])
df['Hand']=df['Hand'].replace(['R'],[1])
df.head()
```

Out[4]:

	Subject ID	MRI ID	Group	Visit	MR Delay	M/F	Hand	Age	EDUC	SES	MMSE	CDR	eTIV	nWBV
0	OAS2_0001	OAS2_0001_MR1	0	1	0	1	1	87	14	2.0	27.0	0.0	1987	0.696
1	OAS2_0002	OAS2_0002_MR1	1	1	0	1	1	75	12	NaN	23.0	0.5	1678	0.736
2	OAS2_0004	OAS2_0004_MR1	0	1	0	0	1	88	18	3.0	28.0	0.0	1215	0.710
3	OAS2_0005	OAS2_0005_MR1	0	1	0	1	1	80	12	4.0	28.0	0.0	1689	0.712
4	OAS2_0007	OAS2_0007_MR1	1	1	0	1	1	71	16	NaN	28.0	0.5	1357	0.748

3. Check for any any null values in any columns.
 Since, SDE column has 8 null values.
 By following imputation we replace these by mean of the entire column.



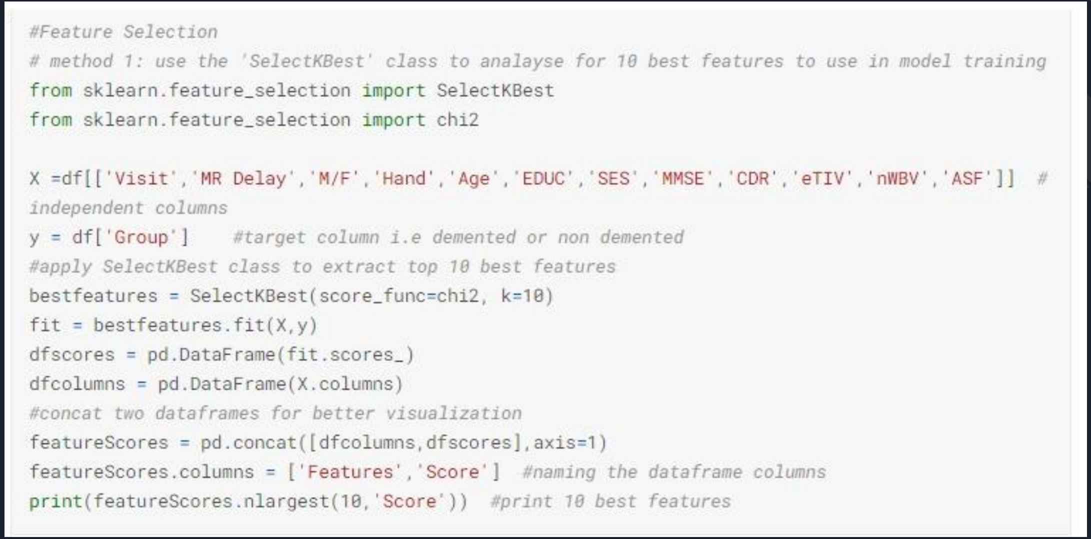
In [7]:
```
#replace the null values with the mean value of the coloumn
df['SES'].fillna((df['SES'].mean()), inplace=True)
```

4. Feature selection:

Segregate dataset into independent and target columns namely, X and Y.

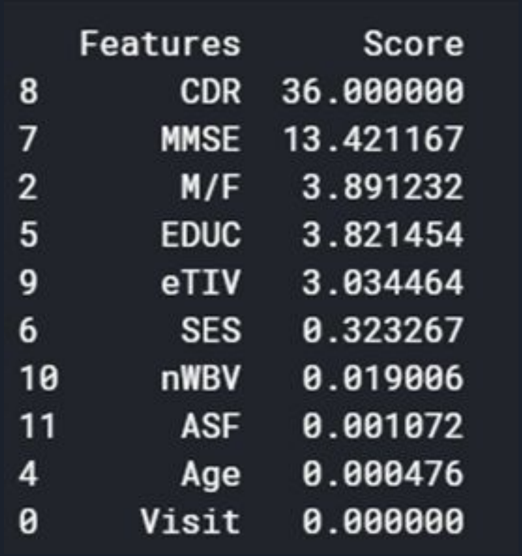
Applying SelectKBest class and using the Chi square test to select top 10 best features.

Fit the data set.



```
#Feature Selection
# method 1: use the 'SelectKBest' class to analayse for 10 best features to use in model training
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2

X =df[['Visit','MR Delay','M/F','Hand','Age','EDUC','SES','MMSE','CDR','eTIV','nWBV','ASF']]  #
independent columns
y = df['Group']    #target column i.e demented or non demented
#apply SelectKBest class to extract top 10 best features
bestfeatures = SelectKBest(score_func=chi2, k=10)
fit = bestfeatures.fit(X,y)
dfscores = pd.DataFrame(fit.scores_)
dfcolumns = pd.DataFrame(X.columns)
#concat two dataframes for better visualization
featureScores = pd.concat([dfcolumns,dfscores],axis=1)
featureScores.columns = ['Features','Score']  #naming the dataframe columns
print(featureScores.nlargest(10,'Score'))  #print 10 best features
```

We use the fit.scores to select the features we will be using. We neglect features having lower scores.

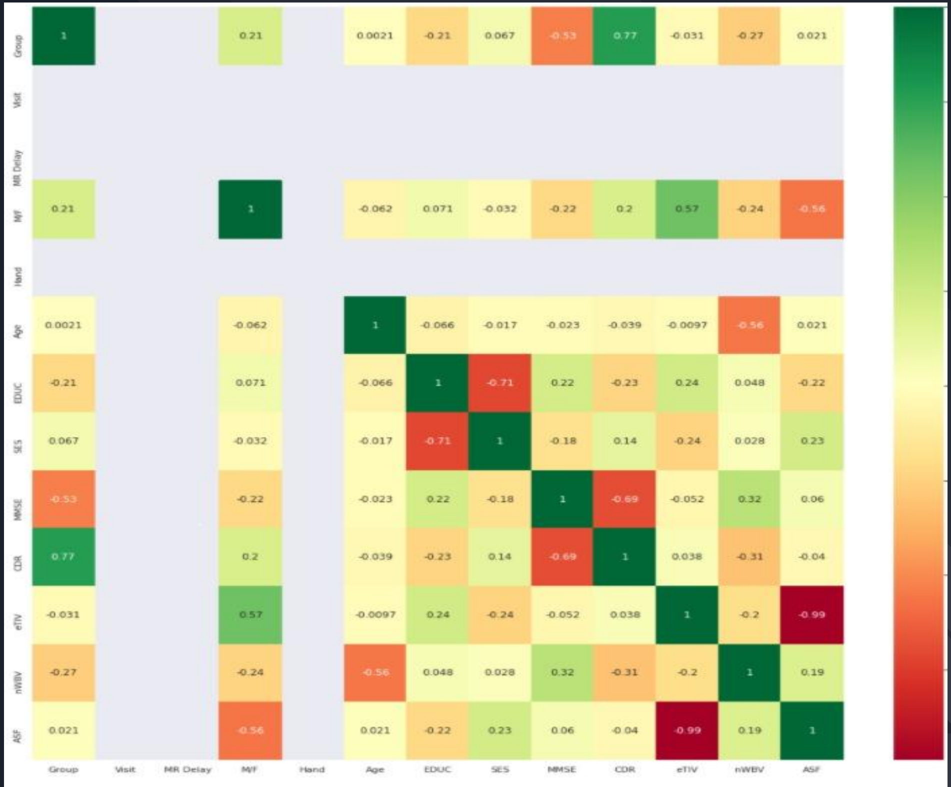


```
   Features      Score
8       CDR  36.000000
7      MMSE  13.421167
2       M/F   3.891232
5      EDUC   3.821454
9      eTIV   3.034464
6       SES   0.323267
10     nWBV   0.019006
11      ASF   0.001072
4       Age   0.000476
0     Visit   0.000000
```

This is the heatmap of independent columns which helps in finding out the correlation of each feature to target column.

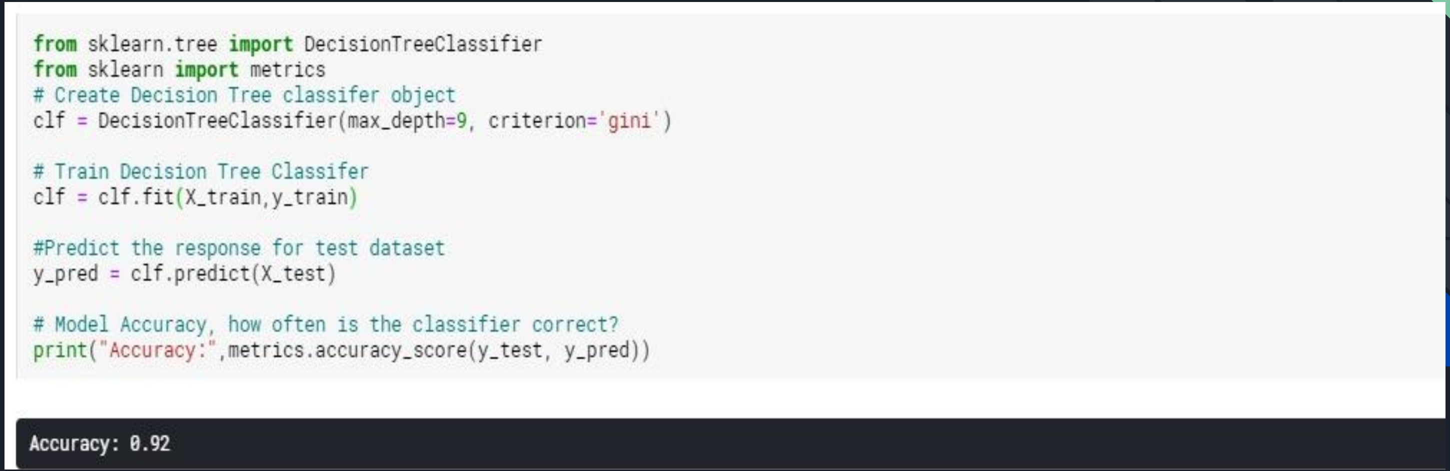
The features we are using are:

<u>'CDR', 'MMSE', 'nWBV', 'M/F', 'EDUC', 'SES', 'ASF', 'eTIV'</u>



```
useful_features=df[['CDR','MMSE','nWBV', 'ASF','M/F','EDUC','eTIV']] #'SES', 'nWBV' 'ASF'
```

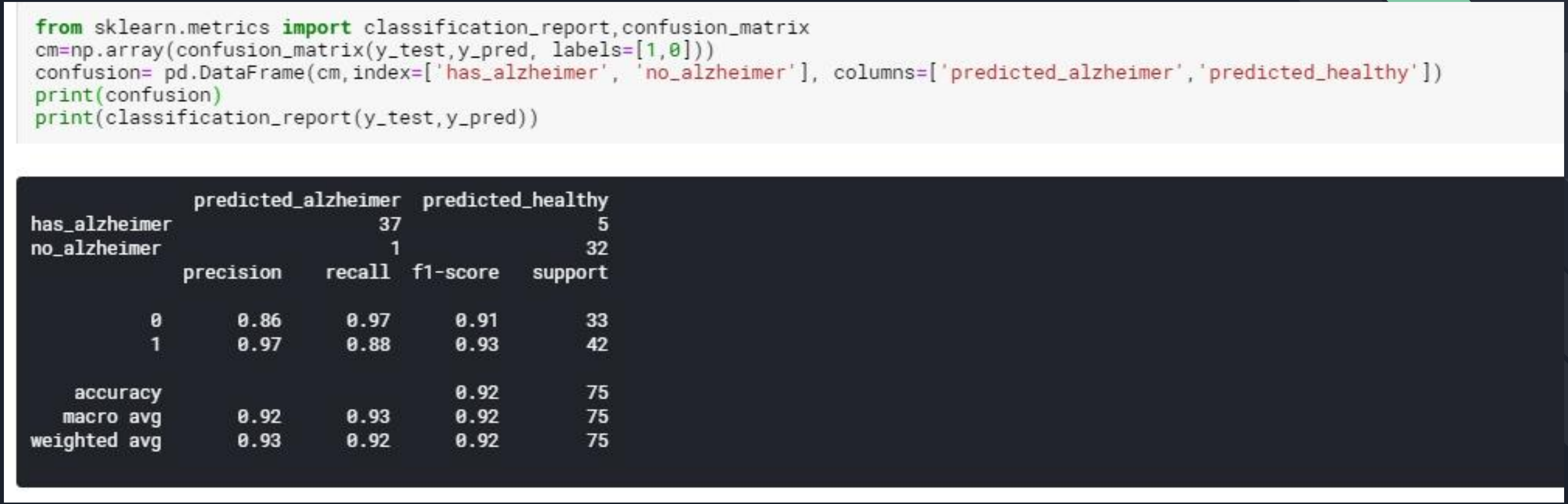
We have used DecisionTreeClassifier from sklearn library to create the classifier object and train the model. After training, the model is tested on test samples to get model accuracy. Our model has got an accuracy of **0.92**



```
from sklearn.tree import DecisionTreeClassifier
from sklearn import metrics
# Create Decision Tree classifer object
clf = DecisionTreeClassifier(max_depth=9, criterion='gini')

# Train Decision Tree Classifer
clf = clf.fit(X_train,y_train)

#Predict the response for test dataset
y_pred = clf.predict(X_test)

# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.92
```

To further understand the performance of the classification model, we have created a confusion matrix. Here we can analyze the precision using the concluded data.



```
from sklearn.metrics import classification_report,confusion_matrix
cm=np.array(confusion_matrix(y_test,y_pred, labels=[1,0]))
confusion= pd.DataFrame(cm,index=['has_alzheimer', 'no_alzheimer'], columns=['predicted_alzheimer','predicted_healthy'])
print(confusion)
print(classification_report(y_test,y_pred))
```

```
               predicted_alzheimer  predicted_healthy
has_alzheimer                   37                  5
no_alzheimer                     1                 32
              precision    recall  f1-score   support

           0       0.86      0.97      0.91        33
           1       0.97      0.88      0.93        42

    accuracy                           0.92        75
   macro avg       0.92      0.93      0.92        75
weighted avg       0.93      0.92      0.92        75
```

Thank you!

**This presentation is only for educational purposes and non commercial usage. The presentation is only meant for IEEE branch of EnTC Dept, Cummins College of Engineering for Women, Pune.*
*It should not be shared with anyone else**