MINI PROJECT

Alzheimer's Disease Detection

Submitted By:

Satakshi Dubey
Vanshita Bansal
Mansi Gupta
Himani Gupta

Submitted to:

Prof. Jitendra Singh Thakur Sir

Early diagnosis of Alzheimer's disease using machine learning techniques.

Problem Statement -

- Alzheimer's Disease is a progressive and irreversible neurological disease and is the most common cause of Dementia in people of the age 65 years and above.

- Detection of Alzheimer's disease at initial stage is very important as it can prevent serious damage to the patient's brain.

Solution -

- Machine Learning Algorithms are excellent at learning patterns and subsequently making predictions on given data.

- This feature of machine learning algorithms makes them a perfect choice for “proactive diagnosis” or prognosis of diseases like Alzheimer's and helps in early stage detection of the disease.

Data and Feature Set -

- The dataset used in this study is sourced from Kaggle made available by the Washington University Alzheimer’s Disease Research Center.

- The features available in the dataset are - **['Subject ID', 'MRI ID', 'Group', 'Visit', 'MR Delay', 'M/F', 'Hand', 'Age', 'EDUC', 'SES', 'MMSE', 'CDR', 'eTIV', 'nWBV', 'ASF']**

Project Steps -

- Data pre-processing.
    - Label Encoding
- Feature Scaling.
- Feature Selection - Correlation Matrix.
- Fitting Machine Learning Algorithms.
    - *Logistic Regression*
    - *Decision Tree*
    - *KNN*
    - *SVM Regressor*

The Clinical Dementia Rating or CDR is a numeric scale used to quantify the severity of symptoms of dementia (i.e. its 'stage').

Composite Rating	Symptoms
0	None
0.5	Very Mild
1	Mild
2	Moderate
3	Severe

Import Necessary Libraries

In [1]:
```
import numpy as np
import pandas as pd
import sklearn
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder

from sklearn.preprocessing import StandardScaler
```

Import dataset

In [2]:
```
data_long = pd.read_csv('oasis_longitudinal.csv')
data_long = data_long.fillna(method='ffill')
```

In [3]:
```
print(data_long.columns)
```

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

Data preview -

In [6]:
```
data_long.head()
```

Out[6]:

	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	2.0	23.0
3	OAS2_0002	OAS2_0002_MR2	Demented	2	560	M	R	76	12	2.0	28.0
4	OAS2_0002	OAS2_0002_MR3	Demented	3	1895	M	R	80	12	2.0	22.0

Data Preprocessing -

Drop unnecessary columns - MRI Id and Visit columns do not help in predicting anything

```
data_long.drop(['MRI ID'], axis=1, inplace=True)
# data_long.drop(['Visit'], axis=1, inplace=True)
```

Converting different CDR values to categorical variable A.B,C,D

```
data_long['CDR'].replace(to_replace=0.0, value='A', inplace=True)
data_long['CDR'].replace(to_replace=0.5, value='B', inplace=True)
data_long['CDR'].replace(to_replace=1.0, value='C', inplace=True)
data_long['CDR'].replace(to_replace=2.0, value='D', inplace=True)
```

Applying Label Encoder

In [27]:
```
for x in data_long.columns:
    f = LabelEncoder()
    data_long[x] = f.fit_transform(data_long[x])
    print(str(x) + " is being transformed.")
```

```
Group is being transformed.
MR Delay is being transformed.
M/F is being transformed.
Hand is being transformed.
Age is being transformed.
EDUC is being transformed.
SES is being transformed.
MMSE is being transformed.
CDR is being transformed.
eTIV is being transformed.
nWBV is being transformed.
ASF is being transformed.
```

Notice that all columns (previously categorical such as M/F) changed to numerical columns (including CDR)

In [13]:
```
data_long.head()
```

Out[13]:

	Group	MR Delay	M/F	Hand	Age	EDUC	SES	MMSE	CDR	eTIV	nWBV	ASF
0	2	0	1	0	27	5	1	14	0	284	35	1
1	2	15	1	0	28	5	1	17	0	285	20	0
2	1	0	1	0	15	3	1	10	1	231	73	52
3	1	32	1	0	16	3	1	15	1	254	51	32
4	1	185	1	0	20	3	1	9	1	238	40	46

In [14]:
```
data_long.CDR.unique()
```

Out[14]:
```
array([0, 1, 2, 3], dtype=int64)
```

Feature Selection -

We have used correlation matrix to identify the features closely correlated to the target variable.

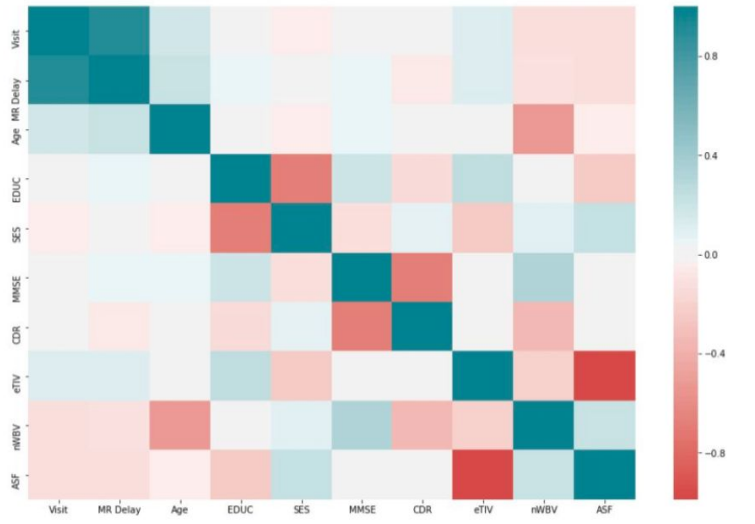
This gives us the most relevant features that can determine the target variable most closely.

MMSE and nWBV comes out to be the top correlated features.

Correlation between various factors

```
corr_matrix =data_long.corr()
rcParams['figure.figsize'] = 15, 10
sns.heatmap(corr_matrix, cmap=sns.diverging_palette(20, 220, n=100))
```

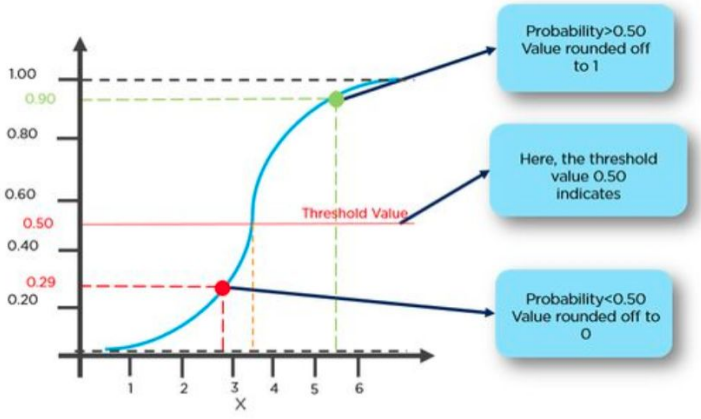
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f555b86f048>
```



Machine Learning Algorithms -

1. Logistic Regression

The logistic model is used to model the probability of a certain class or event existing such as pass/fail, win/lose, alive/dead or healthy/sick.



Logistic Regression - Code and Classification Accuracy

In [36]:
```
from sklearn.linear_model import LogisticRegression
logistic_classifier = LogisticRegression()
logistic_classifier.fit(X_train,y_train)
prediction = logistic_classifier.predict(X_test)
```

```
C:\Users\satak\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:43
2: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a
solver to silence this warning.
  FutureWarning)
C:\Users\satak\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:46
9: FutureWarning: Default multi_class will be changed to 'auto' in 0.22. Specif
y the multi_class option to silence this warning.
  "this warning.", FutureWarning)
```

In [37]:
```
print('Accuracy of prediction - ', logistic_classifier.score(X_test, y_test))
```

```
Accuracy of prediction -  0.8214285714285714
```

Linear Regression

In [21]:
```
from sklearn.linear_model import LinearRegression
classifier = LinearRegression()
classifier.fit(X_train, y_train)
prediction = classifier.predict(X_test)
```

In [22]:
```
print('Accuracy of prediction - ', classifier.score(X_test, y_test))
```

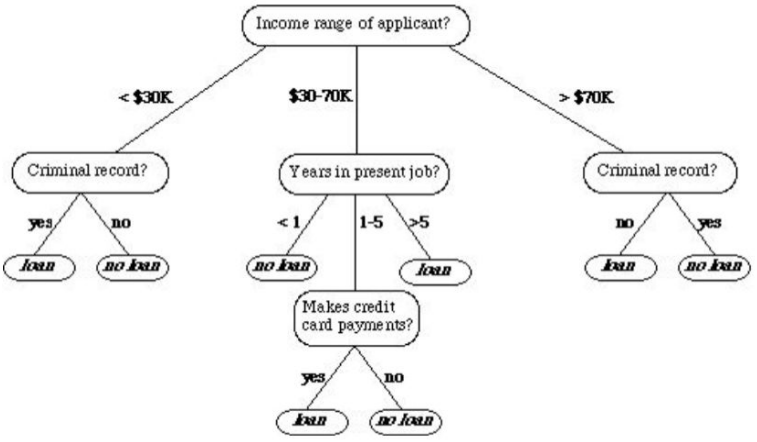
```
Accuracy of prediction -  0.6203928485391891
```

Logistic Regression performs fairly well but Linear regression has a very poor accuracy.

Machine Learning Algorithms -

2. Decision Tree

A decision tree is a decision tool that uses a tree-like model to perform tasks of classification.



Decision Tree - Code and Classification Accuracy

Decision Tree

In [24]:
```
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(max_depth=30)
classifier.fit(X_train, y_train)
prediction = classifier.predict(X_test)
print('Accuracy of prediction - ',classifier.score(X_test, y_test))
```

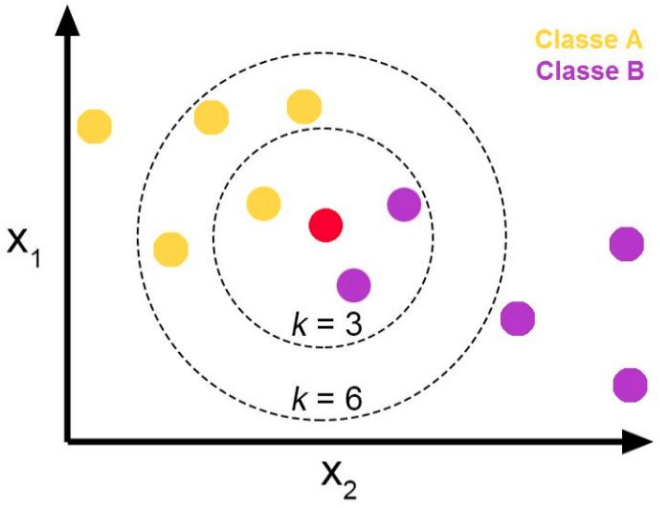
```
Accuracy of prediction -  0.8928571428571429
```

Decision Tree almost has an accuracy of 90%.

Machine Learning Algorithms -

3. KNN

The K-Nearest Neighbor classifier is a classifier algorithm where the learning is based “how similar” is a data (a vector) from other closely lying data points.



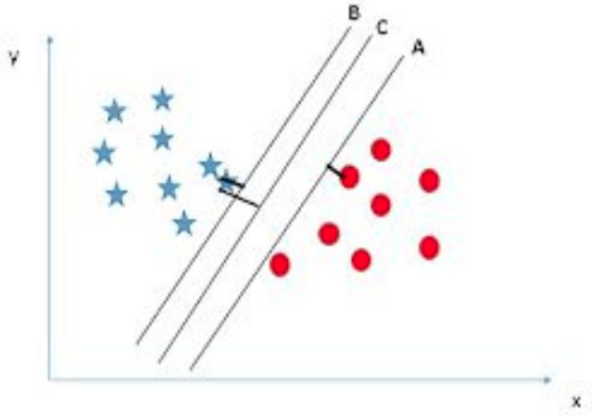
KNN- Code and Classification Accuracy

In [30]:
```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, y_train)
# print(knn.score(X_train, y_train))
prediction = knn.predict(X_test)
print('Accuracy of prediction - ',knn.score(X_test, y_test))
```

```
Accuracy of prediction -  0.8571428571428571
```

4. SVM Classifier

Support-vector machines are supervised learning models that draws a hyperplane to distribute data points into respective classes.



SVM Regressor

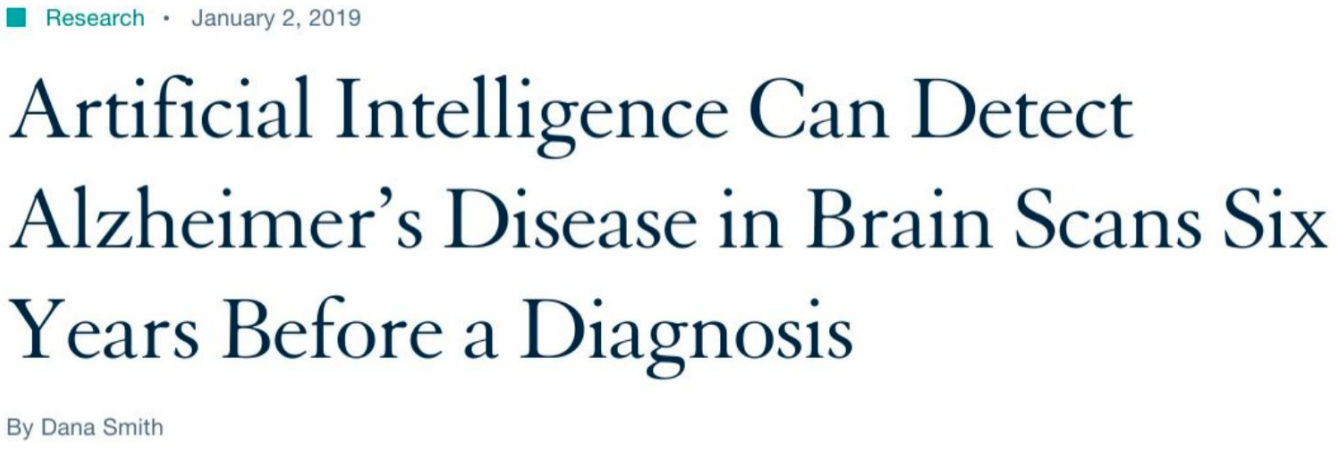
In [26]:
```
from sklearn.svm import SVC
svc=SVC(kernel="linear", C=0.3)
svc.fit(X_train, y_train)
prediction = svc.predict(X_test)
print('Accuracy of prediction - ', svc.score(X_test, y_test))
```

```
Accuracy of prediction -  0.8482142857142857
```

Summary -

Algorithm	Accuracy
Logistic Regression	82.2%
Decision Tree	89.3%
KNN	85.7%
SVM	84.8%

Conclusion -



Research • January 2, 2019

Artificial Intelligence Can Detect Alzheimer's Disease in Brain Scans Six Years Before a Diagnosis

By Dana Smith

- In this project, we have used four different machine learning algorithms to identify which one of them performs the best in learning the MRI features and subsequently predicting the state of Alzheimer disease factor called **CDR** or **Clinical Dementia Rating.**

- The high accuracy of these algorithms empowers the fact that ML algorithms can learn complex data and help in early diagnosis of disease thus saving lives and also helping medical professionals to intervene in clinical treatment in due time.