COMPARITATIVE ANALYIS OF VARIOUS MACHINE LEARNING ALGORITHMS FOR ALZHEIMERS DISEASE PREDICTION

Submitted by

Sai Rishyanth Visinigiri [RA2111026010280]
Suhas Ganga [RA2111026010281]

Under the Guidance of

Dr. M. Uma

Associate Professor, Department of Computational Intelligence

In partial satisfaction of the requirements for the degree of

BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE ENGINEERING

with specialization in Artificial Intelligence & Machine Learning

SCHOOL OF COMPUTING

COLLEGE OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR – 603203

OCTOBER 2023

COLLEGE OF ENGINEERING & TECHNOLOGY
SRM INSTITUTE OF SCIENCE & TECHNOLOGY
S.R.M. NAGAR, KATTANKULATHUR – 603 203
CHENGALPATTU DISTRICT

BONAFIDE CERTIFICATE

Register No **RA2111026010280, RA2111026010281,** certified to be the bonafide work done by **SAI RISHYANTH VISINIGIRI, SUHAS GANGA** of III Year/V Sem B. Tech Degree Course in **Statistical Machine Learning 18CSE479T** in **SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**, Kattankulathur during the academic year 2022 – 2023.

FACULTY-IN-CHARGE
Dr. M. Uma
Associate Professor,
Department of Computational Intelligence
SRM Institute of Science and Technology
Kattankulathur Campus, Chennai

HEAD OF THE DEPARTMENT
Dr. R Annie Uthra
Professor and Head,
Department of Computational Intelligence
SRM Institute of Science and Technology
Kattankulathur Campus, Chennai

CHAPTER NO	TITLE	PAGE NO
I	LIST OF FIGURES	4
II	LIST OF TABLES	5
1	INTRODUCTION	6
2	LITERATURE SURVEY	7
3	STATISTICAL ANALYSIS	8
3.1	MEAN, MEDIAN, MODE	10
3.2	F-TEST (ANNOVA)	14
3.3	T-TEST	19
3.4	CHI-TEST	22
4	SUPERVISED LEARNING	25
4.1	LINEAR REGRESSION	25
4.2	LOGISTIC REGRESSION	28
4.3	DECISION TREE	34
4.4	RANDOM FOREST	37
4.5	K-NEAREST NEIGHBOUR	39
4.6	SUPPORT VECTOR MACHINE	42
4.7	ARTIFICAL NEURAL NETWORK	47
5	UNSUPERVISED LEARNING	49
5.1	K-MEANS	49
5.2	PRINCIPAL COMPONENT ANALYSIS	53
6	PERFORMANCE ANALYSIS	55
6.1	COMPARISION ANALYSIS OF MACHINE LEARNING ALGORITHM	55
6.2	RESULTS & DISCUSSION	55
7	CONCLUSION & FUTURE ENHANCEMENTS	57
8	REFERENCES	58

I. LIST OF FIGURES

FIGURE NO.	TITLE	PAGE NO.
3.0.1	Dataset Information	8
3.0.2	Oasis Dataset	9
3.0.3	Statistical Summary of the Dataset	9
3.0.4	Features with Null Values	10
3.0.5	Duplicate Entries Count	10
3.0.6	Missing Value Treatment	10
3.1.1	Mean Values of the Variables	11
3.1.2	Median Values of the Variables	12
3.1.3	Mode Values of the Variables	13
3.2.1	Test Statistics and P Value for EDUC	14
3.2.2	Test Statistics and P Value for SES	15
3.2.3	Test Statistics and P Value for MMSE	16
3.2.4	Test Statistics and P Value for CDR	17
3.2.5	Test Statistics and P Value for eTIV	17
3.2.6	Test Statistics and P Value for nWBV	18
3.2.7	Test Statistics and P Value forASF	19
3.3.1	Test Statistics and P Value (T-Test)	20
3.4.1	Chi-TEST for Independence (Group and Gender)	23
3.4.2	Chi Test for Goodness of Fit	24
4.1.1	Accuracy of Linear Regression Model	26
4.1.2	CDR vs Age Plot (Linear Model)	26
4.2.1	Logistic Regression Equation	28
4.2.2	Logistic Alzheimer’s CDR Score Predictions	29
4.2.3	Logistic Alzheimer’s CDR Score Probabilities	30
4.2.4	Alzheimer’s Logistic CDR Score Confusion Matrix	32

FIGURE NO.	TITLE	PAGE NO.
4.2.5	Classification Report and Accuracy of the Logistic Regression Model	34
4.3.1	Gini Impurity Equation	34
4.3.2	Entropy Equation	35
4.3.3	Classification Report and Accuracy of the Decision Tree	37
4.4.1	Classification Report and Accuracy of Random Forest Algorithm	39
4.5.1	Euclidean Distance Formula	40
4.5.2	Classification Report and Accuracy of K-Nearest Neighbor Algorithm	41
4.6.1	Classification Report and Accuracy of the Support Vector Machine	43
4.7.1	Test Loss and Accuracy Scores for ANN Model	47
4.7.2	Confusion Matrix for ANN Model	48
5.1.1	Plot to find Optimal K using Elbow Method	52
5.1.2	Cluster Categories and Sizes	52
5.2.1	Plot for PCA of Alzheimer’s Dataset	55

II. LIST OF TABLES

TABLE NO.	TITLE	PAGE NO.
6.1.1	Accuracy Comparison Table	56

1. INTRODUCTION

Alzheimer's disease (AD) is a progressive neurological condition characterized by a gradual deterioration of cognitive functions over time, accounting for the majority (60-70%) of dementia cases. Its initial symptom often involves difficulty in recalling recent events, and as the disease advances, individuals may experience behavioural changes, language difficulties, disorientation, mood swings, apathy, and neglect of self-care. This debilitating condition eventually leads to a decline in bodily functions, culminating in fatality. Typically, the life expectancy after diagnosis ranges from three to nine years, although the rate of progression varies among individuals.

One common early challenge in Alzheimer's disease is the difficulty in finding the right words or names, struggling to remember people's names when meeting new individuals, and facing challenges in social and professional settings due to forgetfulness. Other signs include forgetting recently read passages in books, misplacing valuable items, and struggling with task planning and organization. According to a 2022 World Health Organization survey, approximately 55 million people worldwide are believed to be affected by Alzheimer's disease, with nearly 10 million new cases diagnosed each year.

Early diagnosis of Alzheimer's disease is a complex and costly process involving the collection of extensive data, the application of advanced prediction algorithms, and expert medical evaluation. The use of automated systems can offer several advantages, as they are not susceptible to human errors, potentially reducing the time required for diagnosis and the level of human involvement, which is crucial. Automation also leads to lower overall costs and more precise results. The clinical diagnosis of Alzheimer's disease, particularly in its early stages, can be challenging. This study focuses on various methods for classifying individuals with Alzheimer's disease based on their MRI scans and demographic information. Data from the Alzheimer's Disease Neuroimaging Initiative (ADNI) database and UC Berkeley Biomarkers were utilized to extract MRI biomarkers for this study.

The Alzheimer's Disease Neuroimaging Initiative (ADNI) is a collaborative effort aimed at improving the design and efficiency of clinical trials for Alzheimer's disease. This multisite research project investigates individuals with AD, those at risk of developing AD, and control subjects without cognitive impairment using data from both public and commercial sources.

2. LITERATURE SURVEY

Neelaveni and Geetha Devasana's Approach with SVM and Decision Tree [1]:

Neelaveni and Geetha Devasana proposed a machine learning approach to predict Alzheimer's Disease using psychological parameters such as Mini-Mental State Examination (MMSE), age, and education. They observed that a gradual reduction in MMSE scores is indicative of AD. Their study applied Support Vector Machine (SVM) and Decision Tree algorithms to classify individuals. However, their study primarily focused on accuracy as the evaluation metric, and the accuracy achieved was comparatively low. This approach not only identified cognitive impairment but also detected the presence of the disease.

Sakshi Singh and Komal Gaikwad's Shallow and Deep Learning Techniques [2]:

Sakshi Singh and Komal Gaikwad proposed the use of both shallow and deep learning techniques for Alzheimer's Disease detection. They emphasized the importance of studying the psychological and socioeconomic effects of the disease. Their study involved various machine learning algorithms, including gradient boosting classifier, XG boost, RFC, Ada boost classifier, Decision tree classifier, SVM Linear, SVM Radial, and Logistic regression. Linear SVM and Ada boost classifier closely followed logistic regression in terms of accuracy. Decision tree classifiers demonstrated high precision, while bagging classifiers excelled in recall and F1 scores. This approach achieved an accuracy of 83% using both clinical and MRI datasets.

Efficient Longitudinal MRI Analysis for Alzheimer's Disease Diagnosis [3]:

a research paper proposing a method for Alzheimer's disease (AD) diagnosis using longitudinal structural MRI images. It highlights the challenges in MRI-based AD diagnosis and presents a solution that achieves high classification accuracies. Related literature includes studies from the Alzheimer's Disease Neuroimaging Initiative (ADNI) and other research on AD diagnosis using MRI data.

Srinivasan Aruchamy and Amrita Haridasan's 3D MRI-Based Detection [4]:

Srinivasan Aruchamy and Amrita Haridasan proposed a method using 3D MRI images for early-stage Alzheimer's Disease detection. Their approach involved separating grey and white matter in 3D brain images and analyzing them separately. The study used four different machine learning algorithms, including Logistic Regression, SVM, Naive Bayes, and Ada boost classifier, to classify AD. The accuracy ranged from 75.3% (Naive Bayes on grey matter) to 90.9% (Ada boost algorithms).

In summary, the literature survey outlines various approaches to Alzheimer's Disease detection using machine learning techniques. These approaches utilize a range of features, from psychological parameters to 3D MRI images, and employ diverse machine learning algorithms. Each method has its strengths and limitations, emphasizing the need for ongoing research to improve accuracy and efficiency in early detection and diagnosis of Alzheimer's Disease.

3. STATISTICAL ANALYSIS

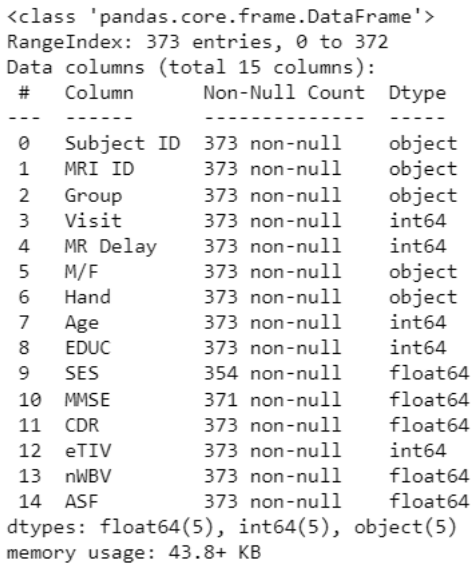
FEATURE ENGINEERING

# Read the dataset

df = pd.read_csv('oasis_longitudinal.csv')

## Display the summary of the dataset

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       373 non-null    int64
 4   MR Delay    373 non-null    int64
 5   M/F         373 non-null    object
 6   Hand        373 non-null    object
 7   Age         373 non-null    int64
 8   EDUC        373 non-null    int64
 9   SES         354 non-null    float64
 10  MMSE        371 non-null    float64
 11  CDR         373 non-null    float64
 12  eTIV        373 non-null    int64
 13  nWBV        373 non-null    float64
 14  ASF         373 non-null    float64
dtypes: float64(5), int64(5), object(5)
memory usage: 43.8+ KB
```

Figure: 3.0.1 Output of dataset information

Datatset:

# COL- Description

# EDUC-Years of Education

# SES-Socioeconomic Status

# MMSE-Mini Mental State Examination

# CDR-Clinical Dementia Rating

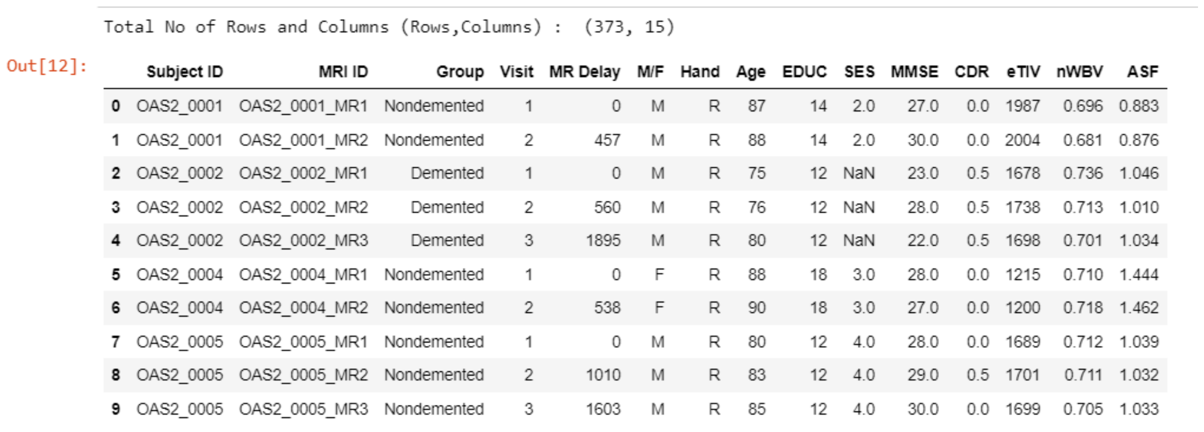
# eTIV-Estimated Total Intracranial Volume

# nWBV-Normalize Whole Brain Volume

# ASF-Atlas Scaling Factor

print("Total No of Rows and Columns (Rows,Columns) : ",df.shape)

df.head(10)



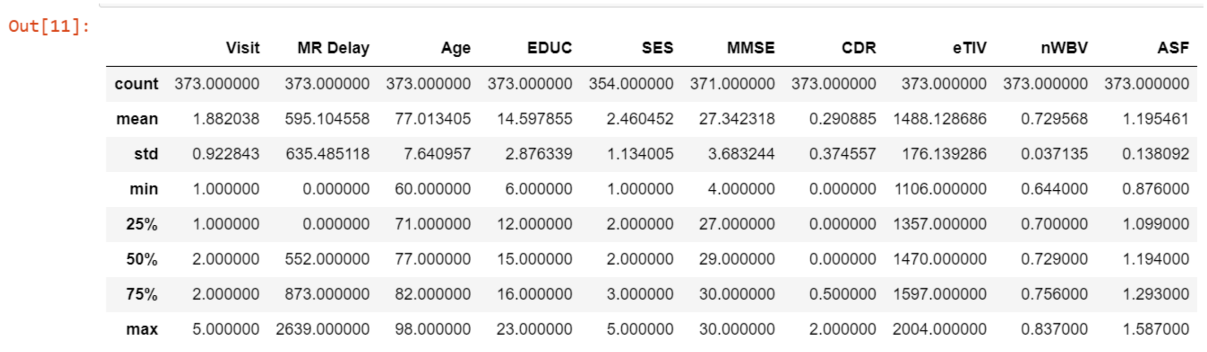
```
Total No of Rows and Columns (Rows,Columns) :  (373, 15)
```

Out[12]:

	Subject ID	MRI ID	Group	Visit	MR Delay	M/F	Hand	Age	EDUC	SES	MMSE	CDR	eTIV	nWBV	ASF
0	OAS2_0001	OAS2_0001_MR1	Nondemented	1	0	M	R	87	14	2.0	27.0	0.0	1987	0.696	0.883
1	OAS2_0001	OAS2_0001_MR2	Nondemented	2	457	M	R	88	14	2.0	30.0	0.0	2004	0.681	0.876
2	OAS2_0002	OAS2_0002_MR1	Demented	1	0	M	R	75	12	NaN	23.0	0.5	1678	0.736	1.046
3	OAS2_0002	OAS2_0002_MR2	Demented	2	560	M	R	76	12	NaN	28.0	0.5	1738	0.713	1.010
4	OAS2_0002	OAS2_0002_MR3	Demented	3	1895	M	R	80	12	NaN	22.0	0.5	1698	0.701	1.034
5	OAS2_0004	OAS2_0004_MR1	Nondemented	1	0	F	R	88	18	3.0	28.0	0.0	1215	0.710	1.444
6	OAS2_0004	OAS2_0004_MR2	Nondemented	2	538	F	R	90	18	3.0	27.0	0.0	1200	0.718	1.462
7	OAS2_0005	OAS2_0005_MR1	Nondemented	1	0	M	R	80	12	4.0	28.0	0.0	1689	0.712	1.039
8	OAS2_0005	OAS2_0005_MR2	Nondemented	2	1010	M	R	83	12	4.0	29.0	0.5	1701	0.711	1.032
9	OAS2_0005	OAS2_0005_MR3	Nondemented	3	1603	M	R	85	12	4.0	30.0	0.0	1699	0.705	1.033

Figure: 3.0.2 Oasis Dataset

df.describe()



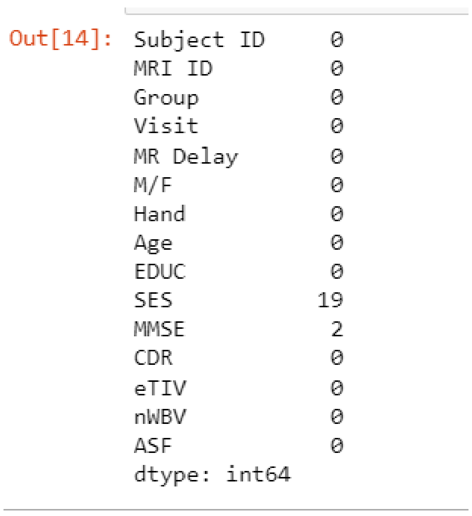
Out[11]:

	Visit	MR Delay	Age	EDUC	SES	MMSE	CDR	eTIV	nWBV	ASF
count	373.000000	373.000000	373.000000	373.000000	354.000000	371.000000	373.000000	373.000000	373.000000	373.000000
mean	1.882038	595.104558	77.013405	14.597855	2.460452	27.342318	0.290885	1488.128686	0.729568	1.195461
std	0.922843	635.485118	7.640957	2.876339	1.134005	3.683244	0.374557	176.139286	0.037135	0.138092
min	1.000000	0.000000	60.000000	6.000000	1.000000	4.000000	0.000000	1106.000000	0.644000	0.876000
25%	1.000000	0.000000	71.000000	12.000000	2.000000	27.000000	0.000000	1357.000000	0.700000	1.099000
50%	2.000000	552.000000	77.000000	15.000000	2.000000	29.000000	0.000000	1470.000000	0.729000	1.194000
75%	2.000000	873.000000	82.000000	16.000000	3.000000	30.000000	0.500000	1597.000000	0.756000	1.293000
max	5.000000	2639.000000	98.000000	23.000000	5.000000	30.000000	2.000000	2004.000000	0.837000	1.587000

Figure: 3.0.3 Statistical summary of the Oasis Dataset

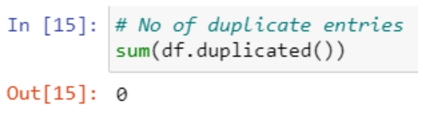
#No of rows and columns containing null values

df.isna().sum()



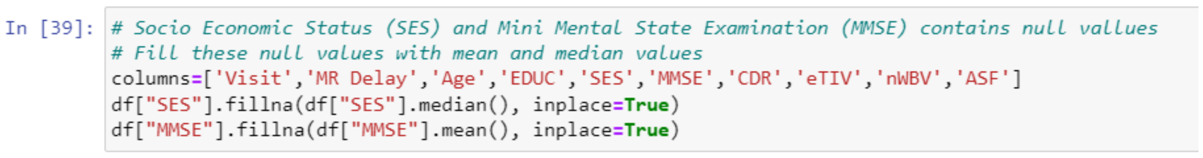
```
Out[14]: Subject ID     0
         MRI ID         0
         Group          0
         Visit          0
         MR Delay       0
         M/F            0
         Hand           0
         Age            0
         EDUC           0
         SES           19
         MMSE           2
         CDR            0
         eTIV           0
         nWBV           0
         ASF            0
         dtype: int64
```

Figure: 3.0.4 Output of features with null values



```
In [15]: # No of duplicate entries
         sum(df.duplicated())

Out[15]: 0
```

Figure: 3.0.5 Code & Output of duplicate entries count



```
In [39]: # Socio Economic Status (SES) and Mini Mental State Examination (MMSE) contains null vallues
         # Fill these null values with mean and median values
         columns=['Visit','MR Delay','Age','EDUC','SES','MMSE','CDR','eTIV','nWBV','ASF']
         df["SES"].fillna(df["SES"].median(), inplace=True)
         df["MMSE"].fillna(df["MMSE"].mean(), inplace=True)
```

Figure: 3.0.6 Code for missing value treatments

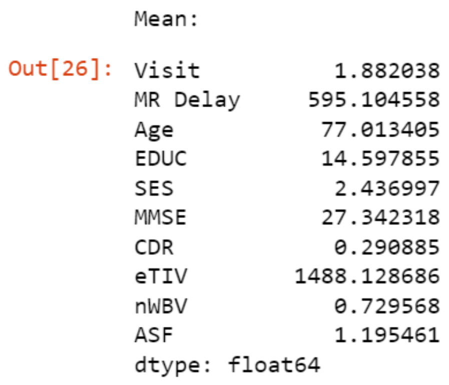
3.1 MEAN, MEDIAN AND MODE

Mean:

Code:

```
# Mean Values of the variables
print("Mean:")
df[columns].mean()
```

Output:



```
           Mean:

Out[26]:   Visit            1.882038
           MR Delay       595.104558
           Age             77.013405
           EDUC            14.597855
           SES              2.436997
           MMSE            27.342318
           CDR              0.290885
           eTIV          1488.128686
           nWBV             0.729568
           ASF              1.195461
           dtype: float64
```

Figure: 3.1.1 Output of Mean Values of the Variables

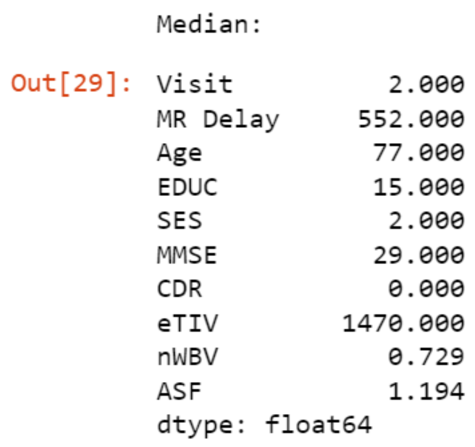
- Visit: On average, the subjects had approximately 1.88 visits. This suggests that most subjects had either 1 or 2 visits.
- MR Delay: The average delay between the baseline and the follow-up MRI is approximately 595.10 days.
- Age: The average age of the subjects is approximately 77 years.
- EDUC: The subjects have, on average, approximately 14.6 years of education.
- SES: The average socioeconomic status (SES) score is approximately 2.44. The interpretation of this value depends on the scale used for SES in your study.
- MMSE: The average Mini Mental State Examination (MMSE) score is approximately 27.34. MMSE is a common test used in clinical and research settings to measure cognitive function.
- CDR: The average Clinical Dementia Rating (CDR) is approximately 0.29. CDR is a numeric scale used to quantify the severity of symptoms of dementia.
- eTIV: The average Estimated Total Intracranial Volume (eTIV) is approximately 1488.13 cubic millimeters.
- nWBV: The average Normalized Whole Brain Volume (nWBV) is approximately 0.73.
- ASF: The average Atlas Scaling Factor (ASF) is approximately 1.20.

Median:

Code:

```
# Median Values of the variables
print("Median: ")
df[columns].median()
```

Output:



```
         Median:

Out[29]: Visit          2.000
         MR Delay     552.000
         Age           77.000
         EDUC          15.000
         SES            2.000
         MMSE          29.000
         CDR            0.000
         eTIV        1470.000
         nWBV           0.729
         ASF            1.194
         dtype: float64
```

Figure: 3.1.2 Output of Median Values of the Variables

- Visit: The median number of visits is 2. This means that half of the subjects had 2 or fewer visits, and half had 2 or more visits.
- MR Delay: The median delay between the baseline and the follow-up MRI is 552 days.
- Age: The median age of the subjects is 77 years.
- EDUC: The median number of years of education that subjects have completed is 15 years.
- SES: The median socioeconomic status (SES) score is 2. The interpretation of this value depends on the scale used for SES in your study.

- MMSE: The median Mini Mental State Examination (MMSE) score is 29. MMSE is a common test used in clinical and research settings to measure cognitive function.

- CDR: The median Clinical Dementia Rating (CDR) is 0. CDR is a numeric scale used to quantify the severity of symptoms of dementia.

- eTIV: The median Estimated Total Intracranial Volume (eTIV) is 1470 cubic millimeters.

- nWBV: The median Normalized Whole Brain Volume (nWBV) is 0.729.

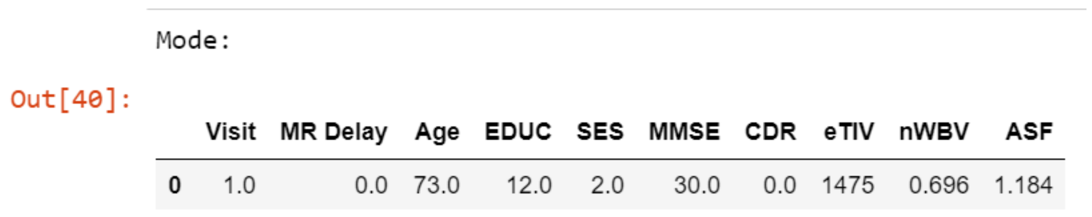
- ASF: The median Atlas Scaling Factor (ASF) is 1.194.

Mode:

Code:

```
# Mode Values of the variables
print("Mode: ")
df[columns].mode()
```

Output:



```
Mode:
Out[40]:
```

	Visit	MR Delay	Age	EDUC	SES	MMSE	CDR	eTIV	nWBV	ASF
0	1.0	0.0	73.0	12.0	2.0	30.0	0.0	1475	0.696	1.184

Figure: 3.1.3 Output of Mode Values of the Variables

3.2 F TEST (ANOVA)

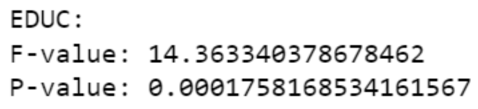
Code:

```
# 'Group' is the categorical column (Demented, Non-demented)
# and 'EDUC', 'SES', 'MMSE', 'CDR', 'eTIV', 'nWBV', 'ASF' are your numerical columns
from scipy import stats
# Perform one-way ANOVA for each numerical column
for column in ['EDUC', 'SES', 'MMSE', 'CDR', 'eTIV', 'nWBV', 'ASF']:
    fvalue, pvalue = stats.f_oneway(df[column][df['Group'] == 'Demented'],
                            df[column][df['Group'] == 'Nondemented'])
    print(f'\n{column}:')
    print('F-value:', fvalue)
    print('P-value:', pvalue)
```

Years of Education (EDUC):

Null Hypothesis, H_0: There is no significant difference in the years of education (‘EDUC’) between the ‘Demented’ and ‘Non-demented’ groups.
Alternate Hypothesis, H_1: There is significant difference in the years of education (‘EDUC’) between the ‘Demented’ and ‘Non-demented’ groups.

Output:



```
EDUC:
F-value: 14.363340378678462
P-value: 0.0001758168534161567
```

Figure: 3.2.1 Output of test statistics and p-value for EDUC

Conclusion:
The F-value of 14.36 and the P-value of 0.000176 for the ‘EDUC’ variable suggest that there is a statistically significant difference in the years of education between the ‘Demented’ and ‘Non-demented’ groups in our Oasis’s dataset.

The P-value is less than 0.05, which typically is the threshold for significance in such tests. This means that there is a less than 0.05 (or less than 5%) probability that the observed difference occurred by chance alone, assuming the null hypothesis is true.

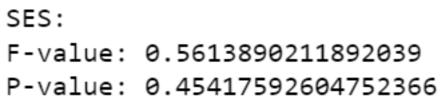
The F-value of 14.36 is relatively large, which indicates a larger variation between group means compared to the variation within the groups.

Result:
In conclusion, these results suggest that years of education ('EDUC') could be a significant factor in differentiating between 'Demented' and 'Non-demented' groups. However, while this statistical test indicates a significant difference, it does not imply causation.

Socio-Economic Status (SES):

Null Hypothesis, H_0: There is no significant difference in the socioeconomic status ('SES') between the 'Demented' and 'Non-demented' groups.
Alternate Hypothesis, H_1: There is significant difference in the socioeconomic status ('SES') between the 'Demented' and 'Non-demented' groups.

Output:



```
SES:
F-value: 0.5613890211892039
P-value: 0.45417592604752366
```

Figure: 3.2.2 Output of test statistics and p-value for SES

Conclusion:
The F-value of 0.56 and the P-value of 0.454 for the 'SES' variable suggests that there is not a statistically significant difference in the socioeconomic status between the 'Demented' and 'Non-demented' groups in our Oasis's dataset.

The P-value is greater than 0.05, which typically is the threshold for significance in such tests. This means that there is a more than 0.05 (or more than 5%) probability that the observed difference occurred by chance alone, assuming the null hypothesis is true.

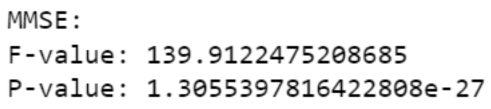
The F-value of 0.56 is relatively small, which indicates a smaller variation between group means compared to the variation within the groups.

Result:
In conclusion, these results suggest that socioeconomic status ('SES') may not be a significant factor in differentiating between 'Demented' and 'Non-demented' groups based on this dataset and the ANOVA test.

Mini Mental State Examination Scores (MMSE):

Null Hypothesis, H_0: There is no significant difference in the Mini Mental State Examination scores (‘MMSE’) between the ‘Demented’ and ‘Non-demented’ groups.
Alternate Hypothesis, H_1: There is significant difference in the Mini Mental State Examination scores (‘MMSE’) between the ‘Demented’ and ‘Non-demented’ groups.

Output:



```
MMSE:
F-value: 139.9122475208685
P-value: 1.3055397816422808e-27
```

Figure: 3.2.3 Output of test statistics and p-value for MMSE

Conclusion:
The F-value of 139.91 and the P-value of approximately 0 for the ‘MMSE’ variable suggest that there is a statistically significant difference in the Mini Mental State Examination scores between the ‘Demented’ and ‘Non-demented’ groups.

The P-value is less than 0.05, which typically is the threshold for significance in such tests. This means that there is a less than 0.05 (or less than 5%) probability that the observed difference occurred by chance alone, assuming the null hypothesis is true.

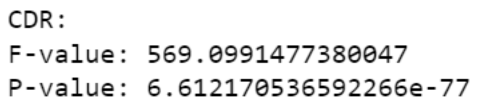
The F-value of 139.91 is relatively large, which indicates a larger variation between group means compared to the variation within the groups.

Result:
In conclusion, these results suggest that Mini Mental State Examination scores (‘MMSE’) could be a significant factor in differentiating between ‘Demented’ and ‘Non-demented’ groups.

Clinical Dementia Rating:

Null Hypothesis, H_0: There is no significant difference in the Clinical Dementia Rating scores (‘CDR’) between the ‘Demented’ and ‘Non-demented’ groups.
Alternate Hypothesis, H_1: There is significant difference in the Clinical Dementia Rating scores (‘CDR’) between the ‘Demented’ and ‘Non-demented’ groups.

Output:



```
CDR:
F-value: 569.0991477380047
P-value: 6.612170536592266e-77
```

Figure: 3.2.4 Output of test statistics and p-value for CDR

Conclusion:

The F-value of 569.10 and the P-value of approximately 0 for the 'CDR' variable suggest that there is a statistically significant difference in the Clinical Dementia Rating scores between the 'Demented' and 'Non-demented' groups.

The P-value is less than 0.05, which typically is the threshold for significance in such tests. This means that there is a less than 0.05 (or less than 5%) probability that the observed difference occurred by chance alone, assuming the null hypothesis is true.

The F-value of 569.10 is relatively large, which indicates a larger variation between group means compared to the variation within the groups.

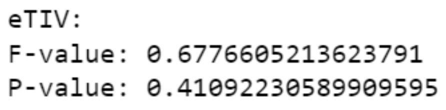
Result:

In conclusion, these results suggest that Clinical Dementia Rating scores ('CDR') could be a significant factor in differentiating between 'Demented' and 'Non-demented' groups.

Estimated Total Intracranial Volume (eTIV):

Null Hypothesis, H_0: There is no significant difference in the Estimated Total Intracranial Volume ('eTIV') between the 'Demented' and 'Non-demented' groups.
Alternate Hypothesis, H_1: There is significant difference in the Estimated Total Intracranial Volume ('eTIV') between the 'Demented' and 'Non-demented' groups.

Output:



```
eTIV:
F-value: 0.6776605213623791
P-value: 0.41092230589909595
```

Figure: 3.2.5 Output of test statistics and p-value for eTIV

Conclusion:
The F-value of 0.68 and the P-value of 0.41 for the 'eTIV' variable suggest that there is not a statistically significant difference in the Estimated Total Intracranial Volume between the 'Demented' and 'Non-demented' groups in your Alzheimer's dataset.

The P-value is greater than 0.05, which typically is the threshold for significance in such tests. This means that there is a more than 0.05 (or more than 5%) probability that the observed difference occurred by chance alone, assuming the null hypothesis is true.

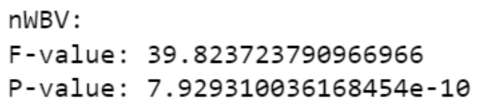
The F-value of 0.68 is relatively small, which indicates a smaller variation between group means compared to the variation within the groups.

Result:
In conclusion, these results suggest that Estimated Total Intracranial Volume ('eTIV') may not be a significant factor in differentiating between 'Demented' and 'Non-demented' groups based on this dataset and the ANOVA test.

Normalize Whole Brain Volume (nWBV)

Null Hypothesis, H_0: There is no significant difference in the Normalize Whole Brain Volume ('nWBV') between the 'Demented' and 'Non-demented' groups.
Alternate Hypothesis, H_1: There is significant difference in the Normalize Whole Brain Volume ('nWBV') between the 'Demented' and 'Non-demented' groups.

Output:



```
nWBV:
F-value: 39.823723790966966
P-value: 7.929310036168454e-10
```

Figure: 3.2.6 Output of test statistics and p-value for nWBV

Conclusion:
The F-value of 39.82 and the P-value of approximately 0 for the 'nWBV' variable suggest that there is a statistically significant difference in the Normalize Whole Brain Volume between the 'Demented' and 'Non-demented' groups.

The P-value is less than 0.05, which typically is the threshold for significance in such tests. This means that there is a less than 0.05 (or less than 5%) probability that the observed difference occurred by chance alone, assuming the null hypothesis is true.

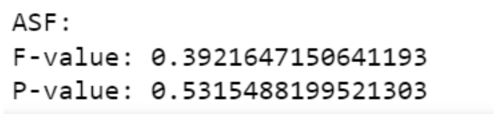
The F-value of 39.82 is relatively large, which indicates a larger variation between group means compared to the variation within the groups.

Result:
In conclusion, these results suggest that Normalize Whole Brain Volume (‘nWBV’) could be a significant factor in differentiating between ‘Demented’ and ‘Non-demented’ groups.

Atlas Scaling Factors (ASF):

Null Hypothesis, H_0: There is no significant difference in the Atlas Scaling Factor (‘ASF’) between the ‘Demented’ and ‘Non-demented’ groups.
Alternate Hypothesis, H_1: There is significant difference in the Atlas Scaling Factor (‘ASF’) between the ‘Demented’ and ‘Non-demented’ groups.

Output:



```
ASF:
F-value: 0.3921647150641193
P-value: 0.5315488199521303
```

Figure: 3.2.7 Output of test statistics and p-value for ASF

Conclusion:
The F-value of 0.39 and the P-value of 0.53 for the ‘ASF’ variable suggest that there is not a statistically significant difference in the Atlas Scaling Factor between the ‘Demented’ and ‘Non-demented’ groups.

The P-value is greater than 0.05, which typically is the threshold for significance in such tests. This means that there is a more than 0.05 (or more than 5%) probability that the observed difference occurred by chance alone, assuming the null hypothesis is true.

The F-value of 0.39 is relatively small, which indicates a smaller variation between group means compared to the variation within the groups.

Result:

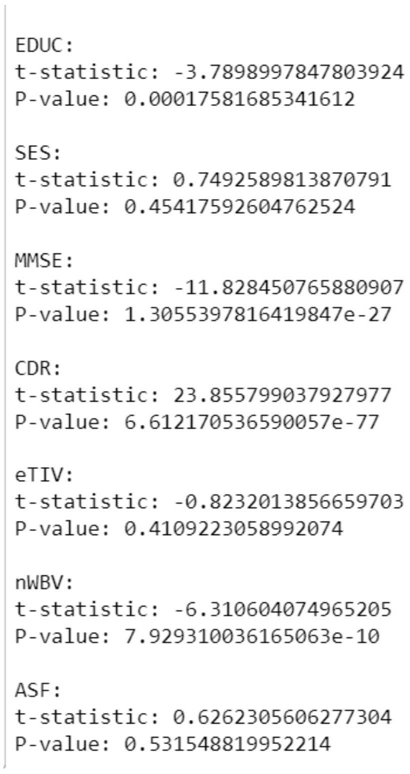
In conclusion, these results suggest that Atlas Scaling Factor (‘ASF’) may not be a significant factor in differentiating between ‘Demented’ and ‘Non-demented’ groups based on this dataset and the ANOVA test.

3.3 T TEST

Code:

```
# Perform t-test for each numerical column
for column in ['EDUC', 'SES', 'MMSE', 'CDR', 'eTIV', 'nWBV', 'ASF']:
    group1 = df[column][df['Group'] == 'Demented']
    group2 = df[column][df['Group'] == 'Nondemented']
    t_statistic, p_value = stats.ttest_ind(group1, group2, nan_policy='omit')
    print(f'\n{column}:')
    print('t-statistic:', t_statistic)
    print('P-value:', p_value)
```

Output:



```
EDUC:
t-statistic: -3.7898997847803924
P-value: 0.00017581685341612

SES:
t-statistic: 0.7492589813870791
P-value: 0.45417592604762524

MMSE:
t-statistic: -11.828450765880907
P-value: 1.3055397816419847e-27

CDR:
t-statistic: 23.855799037927977
P-value: 6.612170536590057e-77

eTIV:
t-statistic: -0.8232013856659703
P-value: 0.4109223058992074

nWBV:
t-statistic: -6.310604074965205
P-value: 7.929310036165063e-10

ASF:
t-statistic: 0.6262305606277304
P-value: 0.531548819952214
```

Figure: 3.3.1 Output of test statistics and p-value for variables (T-Test)

Conclusion:

- EDUC (Years of Education): The t-statistic is -3.79 and the P-value is approximately 0.00018. The null hypothesis that there is no significant difference in the years of education between the 'Demented' and 'Non-demented' groups is rejected. This suggests a statistically significant difference in years of education between these groups.

- SES (Socioeconomic Status): The t-statistic is 0.75 and the P-value is approximately 0.45. The null hypothesis that there is no significant difference in socioeconomic status between the 'Demented' and 'Non-demented' groups is not rejected. This suggests that there may not be a statistically significant difference in socioeconomic status between these groups.

- MMSE (Mini Mental State Examination): The t-statistic is -11.83 and the P-value is approximately 0. The null hypothesis that there is no significant difference in MMSE scores between the 'Demented' and 'Non-demented' groups is rejected. This suggests a statistically significant difference in MMSE scores between these groups.

- CDR (Clinical Dementia Rating): The t-statistic is 23.86 and the P-value is approximately 0. The null hypothesis that there is no significant difference in CDR scores between the 'Demented' and 'Non-demented' groups is rejected. This suggests a statistically significant difference in CDR scores between these groups.

- eTIV (Estimated Total Intracranial Volume): The t-statistic is -0.82 and the P-value is approximately 0.41. The null hypothesis that there is no significant difference in eTIV between the 'Demented' and 'Non-demented' groups is not rejected. This suggests that there may not be a statistically significant difference in eTIV between these groups.

- nWBV (Normalize Whole Brain Volume): The t-statistic is -6.31 and the P-value is approximately 0. The null hypothesis that there is no significant difference in nWBV between the 'Demented' and 'Non-demented' groups is rejected. This suggests a statistically significant difference in nWBV between these groups.

- ASF (Atlas Scaling Factor): The t-statistic is 0.63 and the P-value is approximately 0.53. The null hypothesis that there is no significant difference in ASF between the 'Demented' and 'Non-demented' groups is not rejected. This suggests that there may not be a statistically significant difference in ASF between these groups.

Result:

In conclusion, these results suggest that years of education (‘EDUC’), MMSE scores (‘MMSE’), CDR scores (‘CDR’), and nWBV could be significant factors in differentiating between ‘Demented’ and ‘Non-demented’ groups in the OASIS dataset, while socioeconomic status (‘SES’), estimated total intracranial volume (‘eTIV’), and atlas scaling factor (‘ASF’) may not be as significant based on this analysis.

3.4 CHI - TEST

Chi-Square Test for Independence:

Null Hypothesis (H_0): The ‘Group’ and ‘M/F’ variables are independent. There is no association between gender and dementia status.

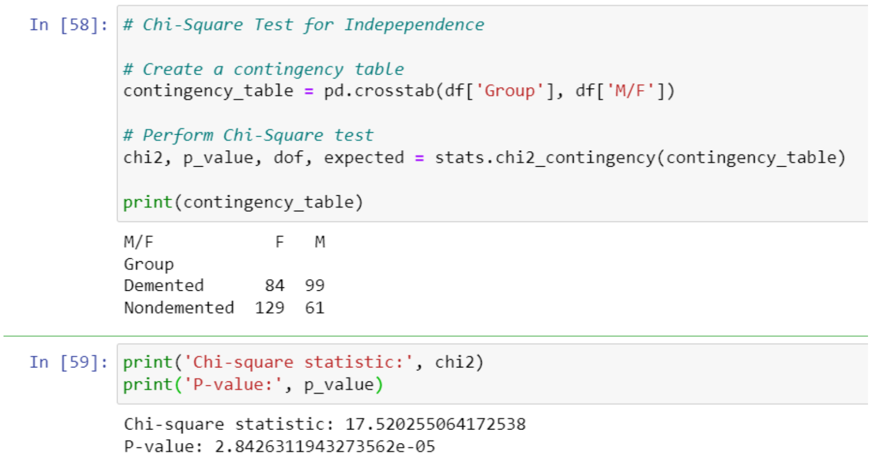
Alternative Hypothesis (H_1): The ‘Group’ and ‘M/F’ variables are not independent. There is an association between gender and dementia status.

These hypotheses can be tested using a Chi-square test for independence. If the p-value from the test is less than your chosen significance level (often 0.05), you would reject the null hypothesis and conclude that there is evidence of an association between gender and dementia status in your dataset. If the p-value is greater than your significance level, you would not reject the null hypothesis, concluding that there’s not enough evidence to suggest an association between gender and dementia status.

Code:

```
# Chi-Square Test for Independence

# Create a contingency table
contingency_table = pd.crosstab(df['Group'], df['M/F'])

# Perform Chi-Square test
chi2, p_value, dof, expected = stats.chi2_contingency(contingency_table)

print(contingency_table)

print('Chi-square statistic:', chi2)

print('P-value:', p_value)
```

Output:



```
In [58]: # Chi-Square Test for Indepependence

         # Create a contingency table
         contingency_table = pd.crosstab(df['Group'], df['M/F'])

         # Perform Chi-Square test
         chi2, p_value, dof, expected = stats.chi2_contingency(contingency_table)

         print(contingency_table)

         M/F           F   M
         Group
         Demented     84  99
         Nondemented  129  61

In [59]: print('Chi-square statistic:', chi2)
         print('P-value:', p_value)

         Chi-square statistic: 17.520255064172538
         P-value: 2.8426311943273562e-05
```

Figure: 3.4.1 Output of Chi-Test for Independence (Group & Gender)

Conclusion:

- Observation from the Contingency Table: The contingency table shows that there are more females who are nondemented (129) compared to those who are demented (84). On the other hand, there are more males who are demented (99) than those who are nondemented (61). This suggests that in this particular dataset, females are more likely to be nondemented, while males are more likely to be demented.

- Chi-square Test Results: The Chi-square test statistic is 17.52 and the P-value is 2.84e-05. The P-value is less than 0.05, which is typically used as a threshold for statistical significance in many fields of study.

- Statistical Conclusion: Given the small P-value, we reject the null hypothesis of no association between gender and dementia status. This means that there is a statistically significant association between gender and dementia status in this dataset.

- Practical Implication: While the test tells us that gender and dementia status are not independent, it does not provide information about causality or the nature of this relationship. It could be that other factors correlated with gender also play a role in dementia status.

Result:

The test tells us that gender and dementia status are not independent.

Chi-Square Test for Goodness of Fit:

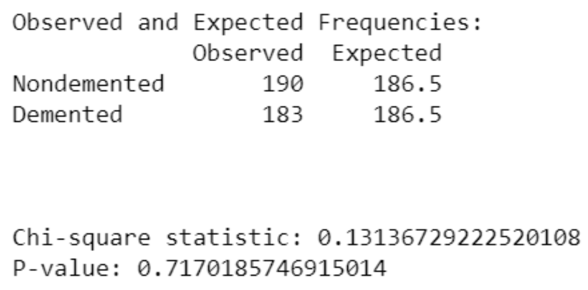
Null Hypothesis (H_0): The observed frequencies of 'Demented' and 'Nondemented' subjects fit the expected frequencies.

Alternative Hypothesis (H_1): The observed frequencies of 'Demented' and 'Nondemented' subjects do not fit the expected frequencies.

Code:

```
# Chi-Square Test for Goodness of Fit
# Get observed frequencies
from scipy.stats import chisquare
observed = df['Group'].value_counts()

# Define expected frequencies (assuming equal distribution)
expected = [len(df)/2, len(df)/2]

# Perform Chi-square test
chi2, p = chisquare(observed, f_exp=expected)

# Create a DataFrame for observed and expected frequencies
freq_df = pd.DataFrame({
    'Observed': observed,
    'Expected': expected
})

print("\nObserved and Expected Frequencies:")
print(freq_df)
print("\n\n")
print(f"Chi-square statistic: {chi2}")
print(f"P-value: {p}")
```

Output:



```
Observed and Expected Frequencies:
             Observed  Expected
Nondemented       190     186.5
Demented          183     186.5



Chi-square statistic: 0.13136729222520108
P-value: 0.7170185746915014
```

Figure: 3.4.2 Output of Chi-Test for Goodness of Fit

Conclusion:

Chi-square Statistic: The Chi-square statistic is 0.131, which is a measure of the difference between your observed frequencies and the expected frequencies.

P-value: The P-value is 0.717, which is greater than the typical significance level of 0.05.

Conclusion: Since the P-value is greater than 0.05, we fail to reject the null hypothesis. This means that there's not enough evidence to suggest a significant difference between your observed frequencies and the expected frequencies. In other words, the distribution of 'Demented' and 'Nondemented' subjects in your dataset fits with what was expected.

Result:

The observed frequencies of 'Demented' and 'Nondemented' subjects fit the expected frequencies.

4. SUPERVISED LEARNING

4.1 LINEAR REGRESSION

Extent to which age is correlated with Alzheimer's disease severity. The Clinical Dementia Rating (CDR) is a numeric scale used to quantify the severity of symptoms of dementia (i.e., Alzheimer's disease). In the OASIS dataset, the CDR is a dependent variable that ranges from 0 (no dementia) to 3 (severe dementia).

The goal is to fit a linear equation to observed data in order used to understand the relationship between age (independent variable) and CDR (dependent variable).

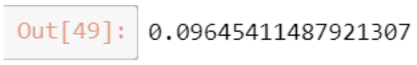
Code:

```
#Supervised Learning Algorithms

# 1. Linear Regression

#Importing the Linear Regression Package from Scikit Learn.

from sklearn.linear_model import LinearRegression

AgeAndCDRLinearReg = LinearRegression()

#Because there are several NaN values in the CDR column, all of the subjects with NaN CDR values will be dropped from the following correlation.

crossSectionalMRI.dropna(subset = ["CDR","SES","Educ","MMSE"], axis = 0, inplace = True)

crossSectionalMRI.head()

Age = crossSectionalMRI[['Age']]

CDRScores = crossSectionalMRI[['CDR']]
```

AgeAndCDRLinearRegModel = AgeAndCDRLinearReg.fit(Age,CDRScores)

AgeAndCDRLinearRegModel.score(Age,CDRScores)

Output:



```
Out[49]: 0.09645411487921307
```

Figure: 4.1.1 Accuracy of the Linear Regression Model

Graphical Representation:

Code:

import seaborn as sns

from matplotlib import pyplot as plt

sns.regplot(x='Age',y='CDR',data=crossSectionalMRI)

plt.ylim(0)

Output:

```
Out[50]: (0.0, 2.110954190185658)
```

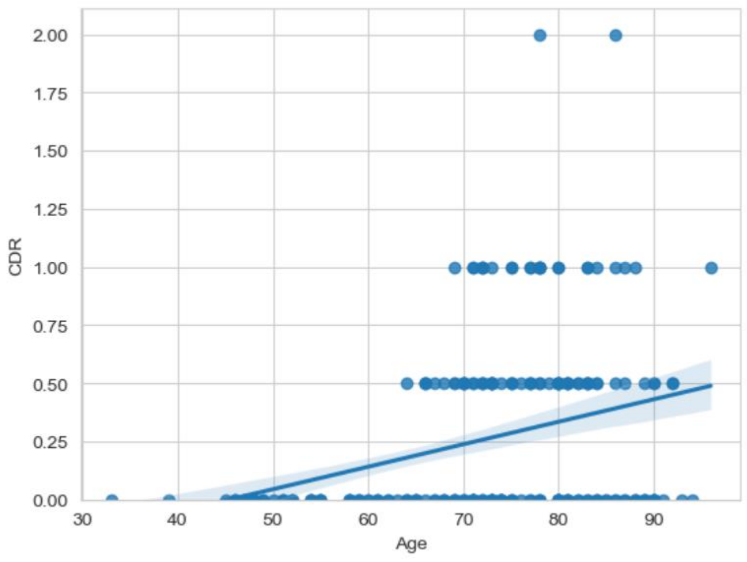


Figure: 4.1.2 Plot of the Linear Regression Model (CDR vs Age)

The Clinical Dementia Rating (CDR) is a discrete variable, often taking on values such as 0, 0.5, 1, 2, and 3 to indicate the severity of dementia. This makes the CDR a categorical variable, even though it’s represented with numbers. Linear regression is typically used for continuous outcome variables. When the outcome variable is categorical, like in this case, other types of regression are more appropriate.

Linear regression may not be the best choice for predicting the Clinical Dementia Rating (CDR) based on age for several reasons:

Linearity: Linear regression assumes that the relationship between the independent and dependent variables is linear. However, the relationship between age and CDR may not be linear. For example, the risk of Alzheimer’s disease increases exponentially with age.

Normality: Linear regression assumes that the residuals (the differences between the observed and predicted values) are normally distributed. However, when the dependent variable is discrete (like CDR), the residuals may not be normally distributed.

Homoscedasticity: Linear regression assumes that the variance of the residuals is constant across all levels of the independent variables (homoscedasticity). However, the variance of CDR may not be constant across all ages.

Independence: Linear regression assumes that the residuals are independent. However, in a longitudinal study where the same individuals are measured at different ages, the residuals may be correlated.

Range of Predicted Values: In linear regression, the predicted values can range from negative infinity to positive infinity. However, CDR is a discrete variable that only takes on a limited number of values (0, 0.5, 1, 2, 3). A linear regression model could predict values outside this range.

In contrast, logistic regression, ordinal regression, or other types of regression models that are designed for categorical outcomes do not have these assumptions and may provide a better fit to the data.

Conclusion:

The correlation between Age and CDR scores is extremely low within this sample of Alzheimer's patients. Pearson's Coefficient in this case is only ~0.09.

Result:

Accuracy of the Model: 0.0964

4.2 LOGISTIC REGRESSION

Logistic regression is a type of machine learning which comes under the supervised learning technique. This technique is a predictive analysis and used in a way to describe data and explain the relationship between a dependent binary variable and one or more ordinal, or nominal variables. The outcome is measured with a dichotomous variable.

$$\mathrm{p} \;= \mathrm{b0} + \mathrm{b1X1} + \mathrm{b2X2} + \mathrm{odds} \;= \frac{\mathrm{p}}{1} - \mathrm{p}$$
$$\log it(p) = \ln\left(\frac{p}{1} - p\right)$$

Figure: 4.2.1 Logistic Regression Equation

Code:

```
import scipy.optimize as opt

from sklearn import preprocessing

%matplotlib inline

import matplotlib.pyplot as plt


#Let's check the datatype of the target variable column (CDR):

before = crossSectionalMRI.dtypes

before

#This column contains data of the type: float64. Let's change that for scikit learn compatability.


#Let's convert the target data type to integer (as required by scikit learn):

crossSectionalMRI['CDR'] = crossSectionalMRI['CDR'].astype(int)


#Let's check to ensure the datatype was correctly changed:

after = crossSectionalMRI.dtypes

after


#Let's convert the Pandas dataframe above into two numpy arrays for more ease of use with scikit
learn functions (train/test splitting, etc.):

crossSectionalMRIFeatures = np.asarray(crossSectionalMRI[['Age','SES','Educ']])

crossSectionalMRITarget = np.asarray(crossSectionalMRI['CDR'])


#Next, let's split the whole dataset into training and testing sets for higher validity:

from sklearn.model_selection import train_test_split
```

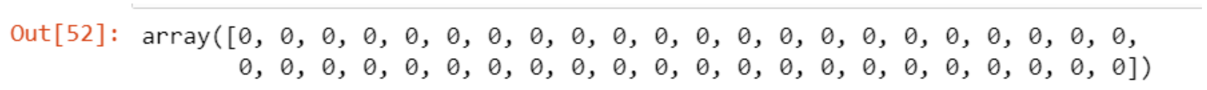
```
X_train, X_test, y_train, y_test = train_test_split(crossSectionalMRIFeatures,
crossSectionalMRITarget, test_size=0.2, random_state=4)
```

```
#Let's build the Multiple Logistic Regression Model using the training sets

#and compute some relevant metrics and, perhaps, make a few predictive statements:
```

```
from sklearn.linear_model import LogisticRegression

from sklearn.metrics import confusion_matrix

crossSectionalMRILogistic = LogisticRegression(C=0.01, solver='newton-cg',
multi_class='multinomial').fit(X_train,y_train)

crossSectionalMRILogistic
```

```
#Finally, let's make a few predictions using this model and the test set, as well as the probability of
each of the class targets (0, 0.5, and 1 CDR Scores):

LogisticAlzhemiersCDRScorePreds = crossSectionalMRILogistic.predict(X_test)

LogisticAlzhemiersCDRScorePreds
```

Output:



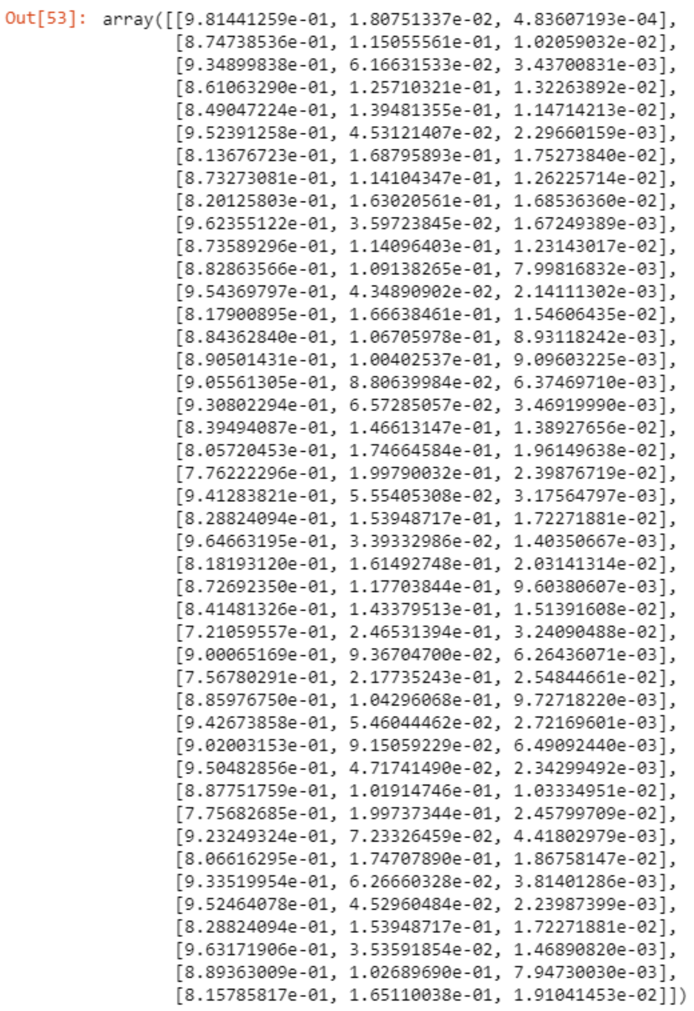
```
Out[52]: array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

Figure: 4.2.2 Logistic Alzheimer’s CDR Score Predictions

Code:

```
LogisticAlzheimersCDRScoreProbas = crossSectionalMRILogistic.predict_proba(X_test)

LogisticAlzheimersCDRScoreProbas
```

Output:



```
Out[53]: array([[9.81441259e-01, 1.80751337e-02, 4.83607193e-04],
                [8.74738536e-01, 1.15055561e-01, 1.02059032e-02],
                [9.34899838e-01, 6.16631533e-02, 3.43700831e-03],
                [8.61063290e-01, 1.25710321e-01, 1.32263892e-02],
                [8.49047224e-01, 1.39481355e-01, 1.14714213e-02],
                [9.52391258e-01, 4.53121407e-02, 2.29660159e-03],
                [8.13676723e-01, 1.68795893e-01, 1.75273840e-02],
                [8.73273081e-01, 1.14104347e-01, 1.26225714e-02],
                [8.20125803e-01, 1.63020561e-01, 1.68536360e-02],
                [9.62355122e-01, 3.59723845e-02, 1.67249389e-03],
                [8.73589296e-01, 1.14096403e-01, 1.23143017e-02],
                [8.82863566e-01, 1.09138265e-01, 7.99816832e-03],
                [9.54369797e-01, 4.34890902e-02, 2.14111302e-03],
                [8.17900895e-01, 1.66638461e-01, 1.54606435e-02],
                [8.84362840e-01, 1.06705978e-01, 8.93118242e-03],
                [8.90501431e-01, 1.00402537e-01, 9.09603225e-03],
                [9.05561305e-01, 8.80639984e-02, 6.37469710e-03],
                [9.30802294e-01, 6.57285057e-02, 3.46919990e-03],
                [8.39494087e-01, 1.46613147e-01, 1.38927656e-02],
                [8.05720453e-01, 1.74664584e-01, 1.96149638e-02],
                [7.76222296e-01, 1.99790032e-01, 2.39876719e-02],
                [9.41283821e-01, 5.55405308e-02, 3.17564797e-03],
                [8.28824094e-01, 1.53948717e-01, 1.72271881e-02],
                [9.64663195e-01, 3.39332986e-02, 1.40350667e-03],
                [8.18193120e-01, 1.61492748e-01, 2.03141314e-02],
                [8.72692350e-01, 1.17703844e-01, 9.60380607e-03],
                [8.41481326e-01, 1.43379513e-01, 1.51391608e-02],
                [7.21059557e-01, 2.46531394e-01, 3.24090488e-02],
                [9.00065169e-01, 9.36704700e-02, 6.26436071e-03],
                [7.56780291e-01, 2.17735243e-01, 2.54844661e-02],
                [8.85976750e-01, 1.04296068e-01, 9.72718220e-03],
                [9.42673858e-01, 5.46044462e-02, 2.72169601e-03],
                [9.02003153e-01, 9.15059229e-02, 6.49092440e-03],
                [9.50482856e-01, 4.71741490e-02, 2.34299492e-03],
                [8.87751759e-01, 1.01914746e-01, 1.03334951e-02],
                [7.75682685e-01, 1.99737344e-01, 2.45799709e-02],
                [9.23249324e-01, 7.23326459e-02, 4.41802979e-03],
                [8.06616295e-01, 1.74707890e-01, 1.86758147e-02],
                [9.33519954e-01, 6.26660328e-02, 3.81401286e-03],
                [9.52464078e-01, 4.52960484e-02, 2.23987399e-03],
                [8.28824094e-01, 1.53948717e-01, 1.72271881e-02],
                [9.63171906e-01, 3.53591854e-02, 1.46890820e-03],
                [8.89363009e-01, 1.02689690e-01, 7.94730030e-03],
                [8.15785817e-01, 1.65110038e-01, 1.91041453e-02]])
```

Figure: 4.2.3 Logistic Alzheimer’s CDR Score Probabilities

Code:

```
#Issues with logisitc model through confusion matrix
from sklearn.metrics import classification_report, confusion_matrix
import itertools
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')

    print(cm)

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt),
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
print(confusion_matrix(y_test, LogisticAlzhemiersCDRScorePreds, labels=[1,0]))

AlzheimersLogisticConfusionMatrix=confusion_matrix(y_test,
LogisticAlzhemiersCDRScorePreds, labels=[1,0.5,0])
np.set_printoptions(precision=2)
# Plot non-normalized confusion matrix
plt.figure()
```

```
plot_confusion_matrix(AlzheimersLogisticConfusionMatrix,
classes=['CDR=1','CDR=0.5','CDR=0'], normalize= False,  title='Alzheimers Logistic CDR Score Confusion Matrix')
```

Output:

```
[[ 0  4]
 [ 0 40]]
Confusion matrix, without normalization
[[ 0  0  4]
 [ 0  0  0]
 [ 0  0 40]]
```

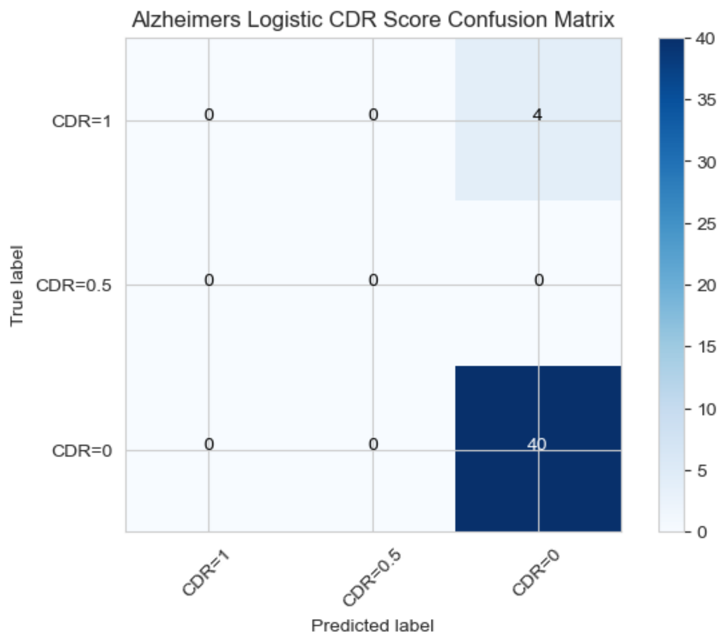


Figure: 4.2.4 Alzheimer’s Logistic CDR Score Confusion Matrix

The results from the confusion matrix suggest that the logistic regression model has a high accuracy in classifying the Clinical Dementia Rating (CDR) as 0. Out of 44 train-test set comparisons, the model correctly classified 40 of them as having a CDR of 0. This indicates that the model is performing well in predicting the absence of dementia.

Logistic regression is a statistical model used for predicting the probability of categorical dependent variables. In this case, it’s being used to predict the CDR based on other variables. The advantage of logistic regression is that it provides probabilities and is capable of handling non-linear effects within your data.

However, it’s important to note that while the model is performing well in classifying a CDR of 0, we should also consider its performance on other classes (CDR of 0.5, 1, 2, 3). Evaluating the model’s performance across all classes will give a more comprehensive understanding of its predictive power.

Code:

```
clfs =[LogisticRegression()]
for model in clfs:
  print(str(model).split('(')[0],": ")
  model.fit(X_train,y_train.ravel())
  X = pd.DataFrame(X_train)
  report_performance(model)
  roc_curves(model)
  accuracy(model)
```

Output:

```
LogisticRegression :


Confusion Matrix:
[[44 16]
 [13 39]]


Classification Report:
              precision    recall  f1-score   support

           0       0.77      0.73      0.75        60
           1       0.71      0.75      0.73        52

    accuracy                           0.74       112
   macro avg       0.74      0.74      0.74       112
weighted avg       0.74      0.74      0.74       112
```

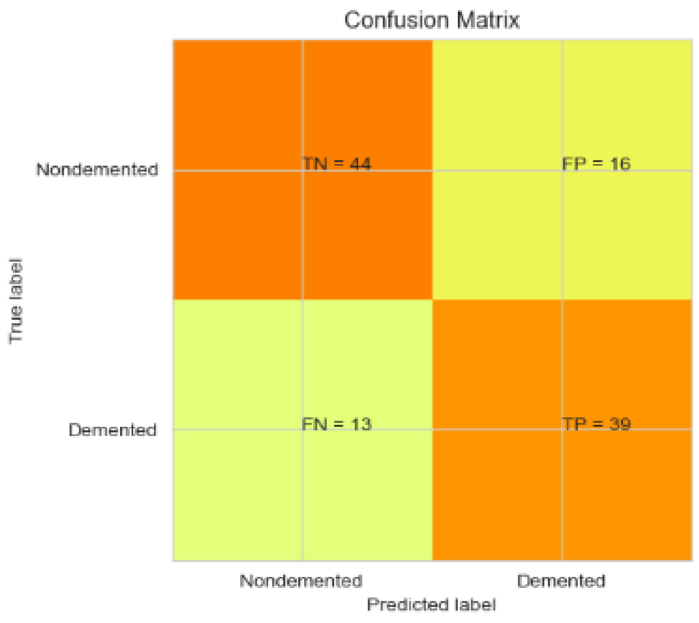
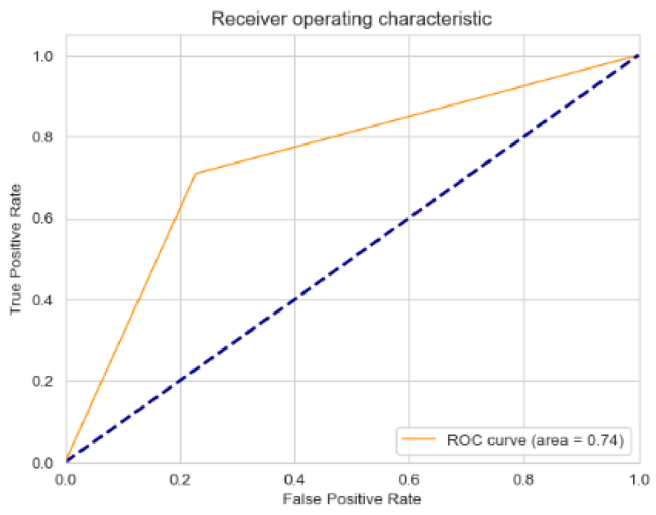


Figure: 4.2.5 Classification Report and Accuracy of the Logistic Regression Model



```
Acuuracy Of the Model:  0.7410714285714286
```

Figure: 4.2.5 Classification Report and Accuracy of the Logistic Regression Model

Conclusion:

In summary, the logistic regression model demonstrates potential in classifying cases with a Clinical Dementia Rating (CDR) of 0, with an accuracy of 74.1%. This suggests its usefulness in early-stage Alzheimer's disease prediction. However, further evaluation across all CDR classes, feature selection, and clinical validation are essential for a comprehensive and robust prediction model.

Result:

Accuracy of the Model: 0.741

4.3 DECISION TREE

Decision tree classifier is a type of supervised machine learning that splits the dataset based on certain parameter that maximizes the separation of data and gives the result in the form of a tree-like structure [12]. Decision tree algorithm is used for classification and regression problem and the algorithm create a binary tree and each node has two edges for finding the best categorical and numeric feature to split by using suitable impurity criterion. For decision tree classification, use Gini and Entropy impurity. In Gini impurity.

$$\sum_{i=1}^{C} fi(1 - fi)$$

Figure 4.3.1 Gini Impurity Equation

Where n is the number of unique labels and fi is the frequency of label x at a node.

In Entropy, fi is the frequency of label x at a node and n is the number of unique labels.

$$\sum_{i=1}^{C} -fi\log(fi)$$

Figure 4.3.2 Entropy Equation

Code:

clf_dtc = DecisionTreeClassifier(criterion='entropy',max_depth=5,random_state=0)

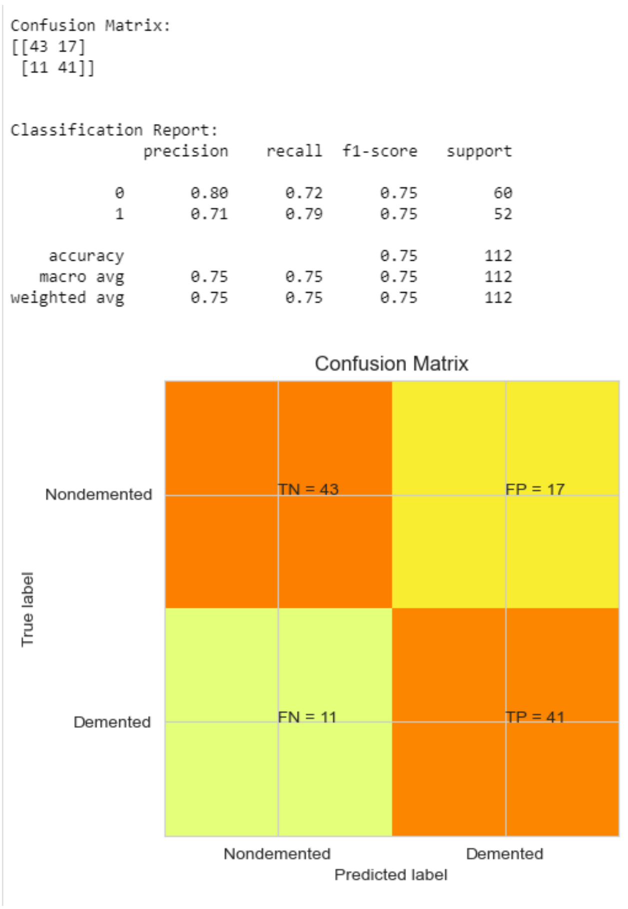
clf_dtc.fit(X_train, y_train.ravel())

report_performance(clf_dtc)

roc_curves(clf_dtc)

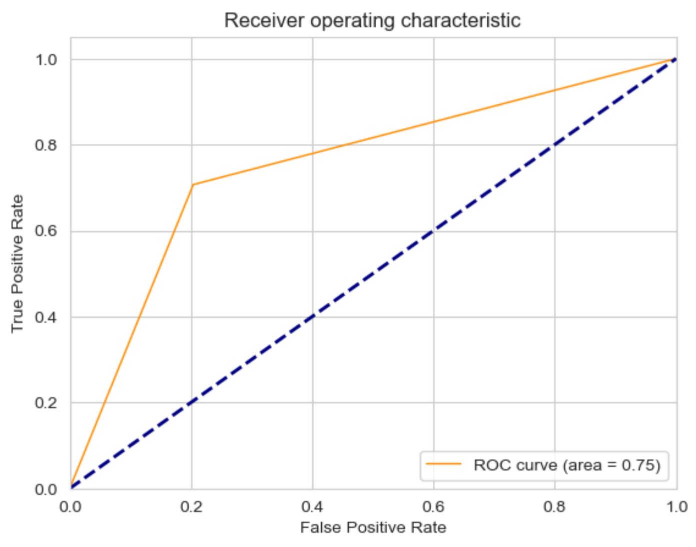
accuracy(clf_dtc)

Output:



```
Confusion Matrix:
[[43 17]
 [11 41]]


Classification Report:
              precision    recall  f1-score   support

           0       0.80      0.72      0.75        60
           1       0.71      0.79      0.75        52

    accuracy                           0.75       112
   macro avg       0.75      0.75      0.75       112
weighted avg       0.75      0.75      0.75       112
```

Figure: 4.3.3 Classification Report and Accuracy of the Decision Tree



```
Accuracy Of the Model:  0.75
```

Figure: 4.3.3 Classification Report and Accuracy of the Decision Tree

Conclusion:

The Decision Tree classifier was employed to predict Alzheimer's disease, and it achieved an accuracy of 75%, a slight improvement compared to the logistic regression model's accuracy of 74.1%. This suggests that the Decision Tree model is a competitive method for Alzheimer's disease prediction in this context.

Additionally, the model's feature importance analysis indicates the significant role of certain features in the classification process.

However, it's essential to emphasize that this conclusion is based on a specific dataset and set of features. Further evaluation of the Decision Tree model's performance across various Clinical Dementia Rating (CDR) classes, hyperparameter fine-tuning, and clinical validation is needed to ensure its effectiveness in identifying different stages of dementia and to make it a robust and reliable predictor in a broader context. The choice of impurity measure, in this case, entropy, also plays a significant role in the model's structure and performance, and should be considered for optimization in future applications.

Result:

Accuracy of the Model: 0.75

4.4 RANDOM FOREST

Bagging and Boosting are both widely recognized ensemble learning techniques used to generate classifiers and combine their outputs. They are employed to enhance predictive accuracy by leveraging multiple models. In Boosting, successive decision trees are constructed, and their predictions are weighted to make a final prediction. In contrast, Bagging independently constructs multiple trees, and a simple majority vote is used for prediction at the end.

Random Forest is a powerful ensemble method that not only builds trees on different subsets of the data but also introduces variability in how the trees are constructed. This variation contributes to the robustness and performance of Random Forest. It operates by creating a collection of decision trees, often referred to as "nTrees," using the training dataset. For each data point in the training dataset, an unpruned tree is constructed. Importantly, at each node of the tree, a random subset of predictors (denoted as "m") is chosen, and the best split is determined based on this subset. Finally, predictions are made by aggregating the outputs of these n Trees, with classification using majority voting and regression using averaging.

Random Forest provides additional valuable insights beyond predictions. It offers information on the importance of predictor variables and an assessment of the internal data structure. Variable importance is calculated by evaluating how changes in the variable impact the prediction error while keeping other variables constant. Additionally, Random Forest's proximity matrix can reveal relationships between data points and tree nodes. It's worth noting that cases where two elements of the proximity matrix terminate at the same node suggest potential repetitions in the data structure.

Code:

```
rfc=RandomForestClassifier(random_state=42)

param_grid = {
    'n_estimators': [200],
    'max_features': ['auto'],
    'max_depth' : [4,5,6,7,8],
    'criterion' :['gini']
}

CV_rfc = GridSearchCV(estimator=rfc, param_grid=param_grid, cv= 5,scoring = 'roc_auc')
CV_rfc.fit(X_train, y_train.ravel())
print("Best parameters set found on development set:")
```

print(CV_rfc.best_params_)

report_performance(CV_rfc)

roc_curves(CV_rfc)

accuracy(CV_rfc)

Output:

```
Best parameters set found on development set:
{'criterion': 'gini', 'max_depth': 8, 'max_features': 'auto', 'n_estimators': 200}


Confusion Matrix:
[[45  7]
 [11 49]]


Classification Report:
             precision    recall  f1-score   support

          0       0.80      0.87      0.83        52
          1       0.88      0.82      0.84        60

avg / total       0.84      0.84      0.84       112
```

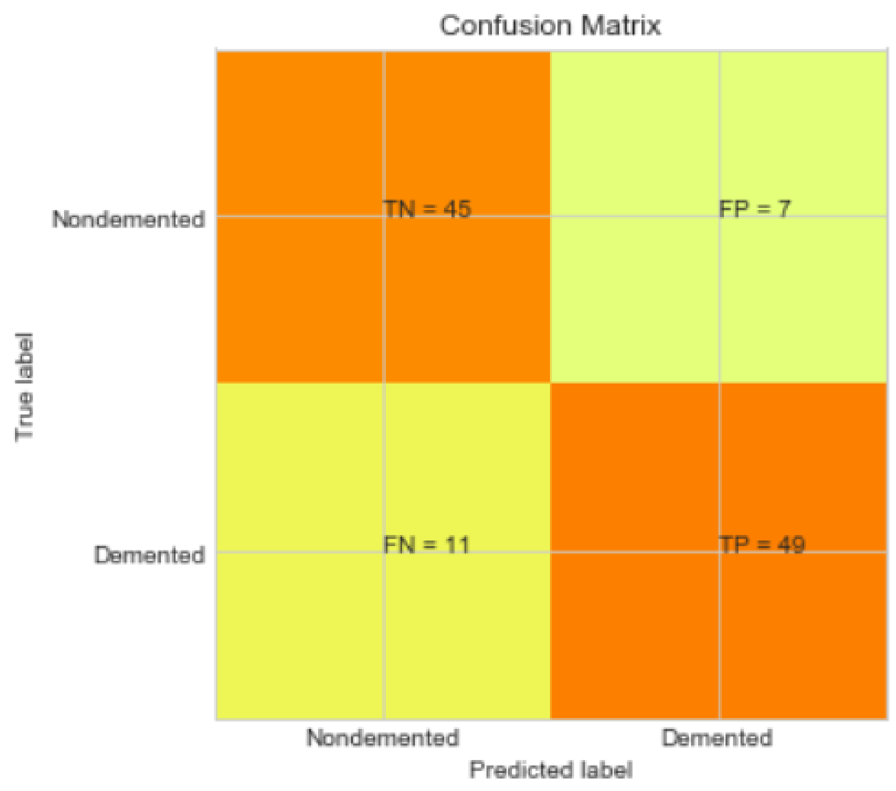
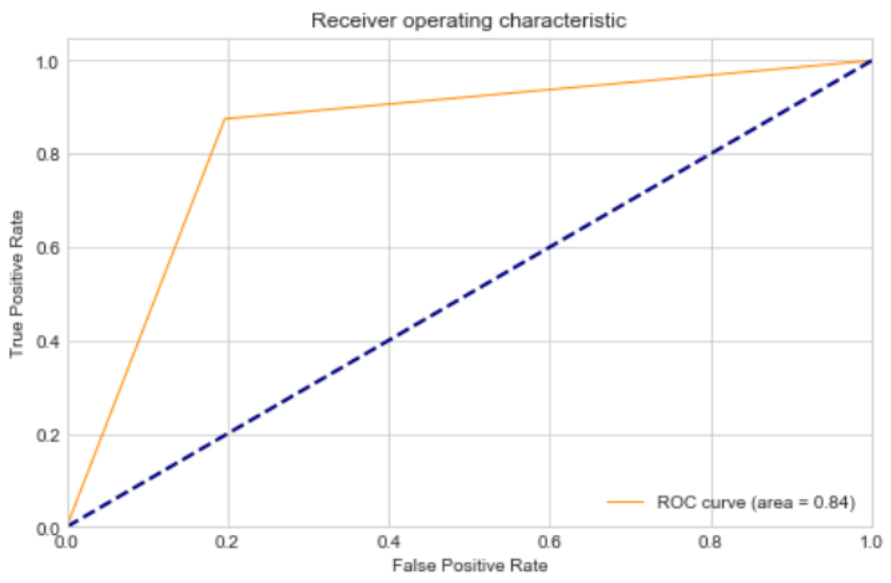


Figure: 4.4.0 Classification Report of Random Forest Algorithm



```
Acuuracy Of the Model:  0.839285714286
```

Figure: 4.4.1 Classification Report and Accuracy of Random Forest Algorithm

Conclusion:

The Random Forest algorithm, optimized through grid search, delivered a robust model for Alzheimer's disease prediction with an impressive accuracy of approximately 83,9%. This result showcases its potential as a powerful tool for identifying the presence of Alzheimer's disease based on the given dataset and features. The assessment of variable importance and data structure further enhances its value, making it a promising choice for real-world applications. However, continued evaluation, clinical validation, and consideration of dataset-specific characteristics are essential for ensuring its reliability in practical scenarios.

Result:

Accuracy of the Model: 0.839

4.5 K NEAREST NEIGHBOUR

K-Nearest Neighbours Model to predict whether a new subject will have a CDR of 0, 0.5, or 1 based on their Age, SES, and Education Level. The k-NN is a supervised machine learning algorithm that classifies the data based on the clusters formed, which are labelled with the respective dementia classification group. The classification model is trained by calculating the Euclidean distances with each data points and assigning them to the cluster with the minimum distance from the cluster's centre. Euclidean distance between two data points (x and y) is calculated as follows.

$$D = \sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$$

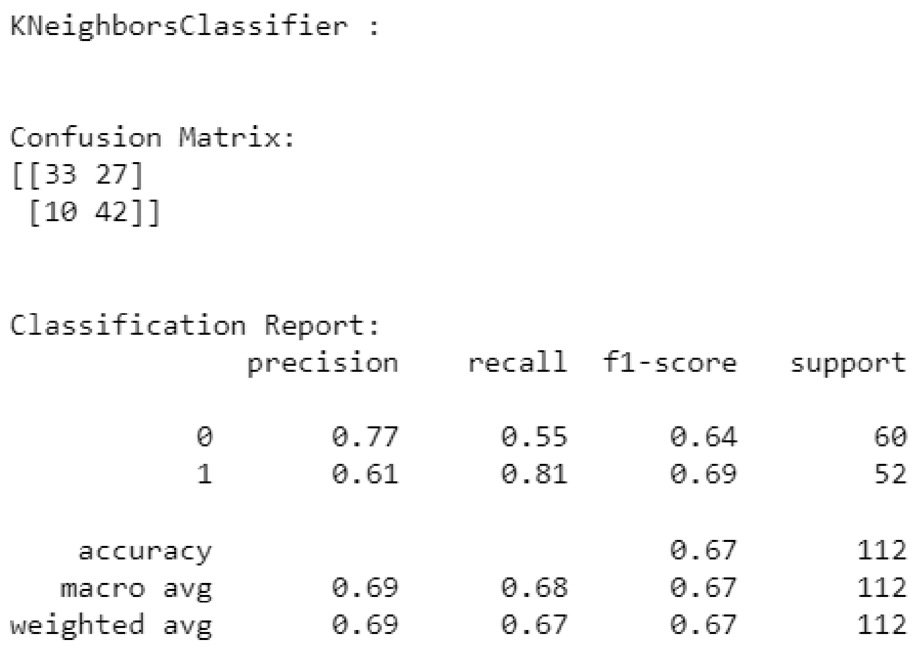
Figure 4.5.1 Euclidean Distance Formula

Where, k is the number of clusters in the model. The optimal number of clusters is obtained by iterating through a specific cluster range and obtaining the minimum number of clusters for the maximum accuracy. The k-NN classification between principal components 1 and 2, where, groups 0,1 and 2 are classification groups non-dementia, dementia respectively.

Code:

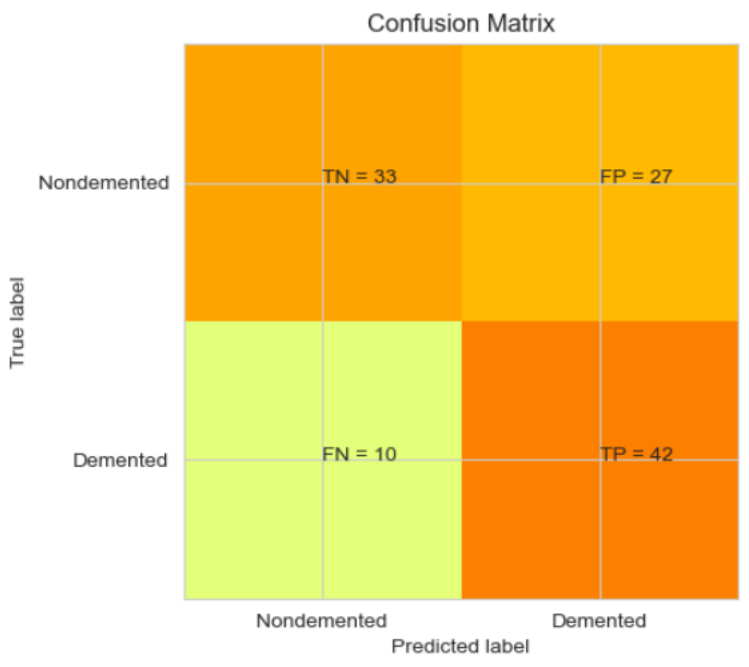
```
clfs =[KNeighborsClassifier()]
for model in clfs:
   print(str(model).split('(')[0],": ")
   model.fit(X_train,y_train.ravel())
   X = pd.DataFrame(X_train)
   report_performance(model)
   roc_curves(model)
   accuracy(model)
```

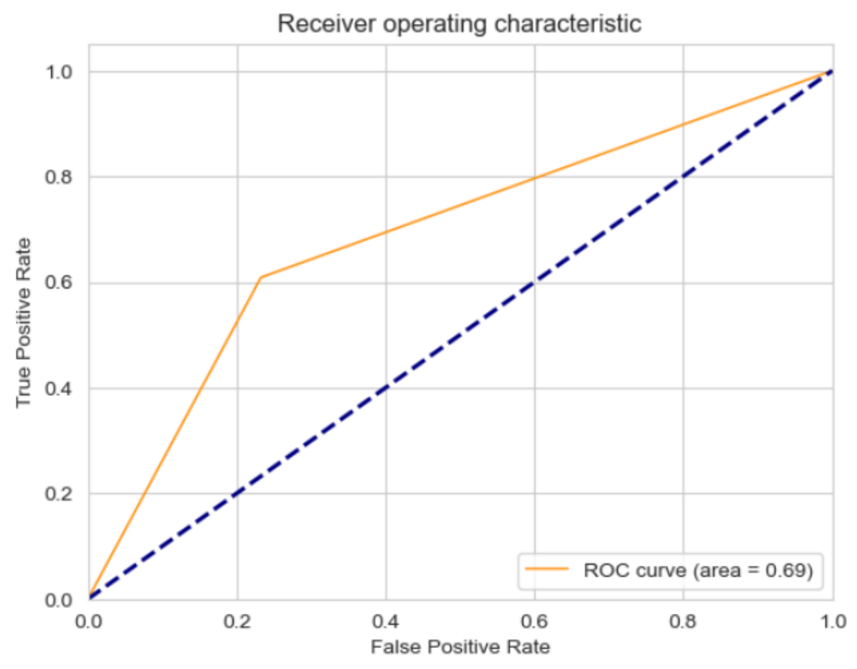
Output:



```
KNeighborsClassifier :


Confusion Matrix:
[[33 27]
 [10 42]]


Classification Report:
              precision    recall  f1-score   support

           0       0.77      0.55      0.64        60
           1       0.61      0.81      0.69        52

    accuracy                           0.67       112
   macro avg       0.69      0.68      0.67       112
weighted avg       0.69      0.67      0.67       112
```

Figure: 4.5.2 Classification Report and Accuracy of K Nearest Neighbour





```
Accuracy Of the Model:  0.6696428571428571
```

Figure: 4.5.2 Classification Report and Accuracy of K-Nearest Neighbour Algorithm

Conclusion:

The K-Nearest Neighbors (k-NN) algorithm achieved an accuracy of 67%, indicating a reasonably good performance. However, it lags slightly behind other models in this specific application. K-NN's effectiveness is influenced by factors like the choice of distance metric and data distribution in feature space.

Further optimization and parameter tuning may be necessary to enhance k-NN's accuracy for Alzheimer's disease prediction.

Result:

Accuracy of the Model: 0.669

4.6 SUPPORT VECTOR MACHINE

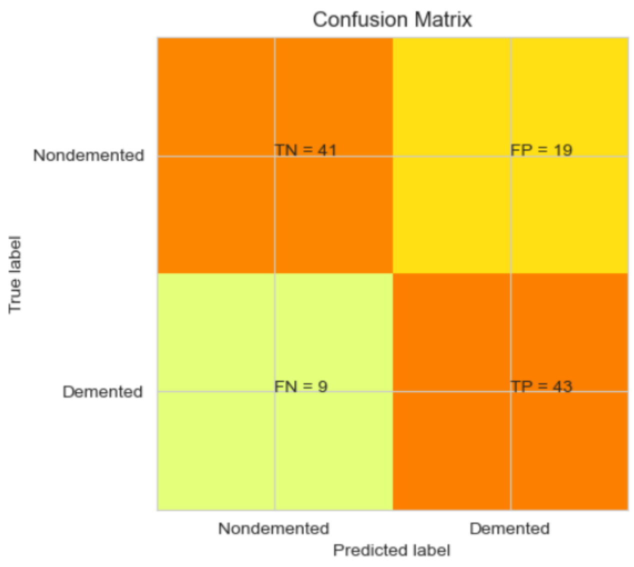
Support Vector Machine or simply SVM is a powerful machine method developed for statistical learning. SVM is a supervised learning method that analyses data and recognises patterns. The empirical classification error and geometric margin is reduced in SVM. An attribute is a predictor variable and a feature is a transformed attribute which defines a hyperplane. A vector is a set of features that represents one instance. The main goal of SVM is to find an optimal hyperplane which separates cases of clusters of vector with one category of variables on one side and the other category variables of on the other side. These vectors that are closer to the hyperplane are the support vector. SVM is useful classification technique that uses training and test data. Each training data’s instance has one target value and several attributes. Finally, SVM produces a model that predicts target values of test data.

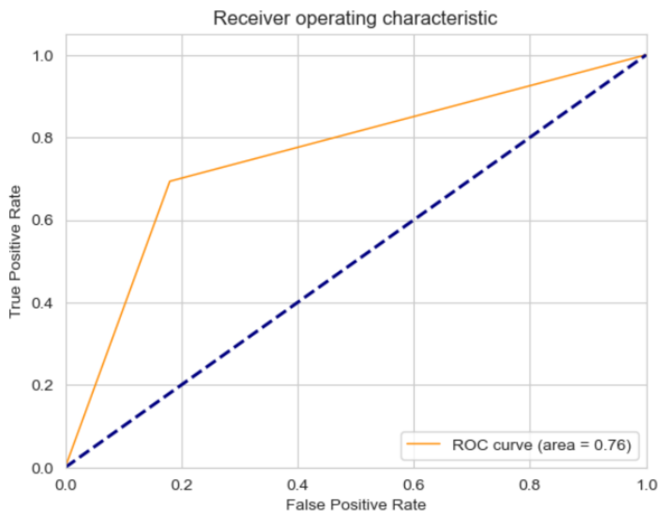
Code:

```
svm = SVC(kernel="linear", C=0.1,random_state=0)
svm.fit(X_train, y_train.ravel())
report_performance(svm)
roc_curves(svm)
accuracy(svm)
```

Output:

```
Confusion Matrix:
[[41 19]
 [ 9 43]]


Classification Report:
              precision    recall  f1-score   support

           0       0.82      0.68      0.75        60
           1       0.69      0.83      0.75        52

    accuracy                           0.75       112
   macro avg       0.76      0.76      0.75       112
weighted avg       0.76      0.75      0.75       112
```





```
Accuracy Of the Model:  0.75
```

Figure: 4.6.1 Classification Report and Accuracy of the Support Vector Machine

Conclusion:

The Support Vector Machine (SVM) model exhibited a remarkable accuracy of 75%, indicating its strong performance in predicting Alzheimer's disease based on the provided features. This high level of accuracy underscores the effectiveness of SVM as a competitive and reliable method for Alzheimer's disease prediction.

In summary, SVM has proven to be a robust and accurate model for this specific application, making it a promising choice for Alzheimer's disease prediction.

Result:

Accuracy of the Model: 0.75

4.7 ARTIFICIAL NEURAL NETWORK

Artificial neural networks (ANNs) are computational modelling tools inspired by the neural networks found in biology. ANNs exhibit several information processing characteristics which make them useful for modelling complex real-world problems, among them non-linearity, their ability to handle imprecise information, and their ability to generalize. A common machine learning task is classification. Here the machine learning algorithm is presented with a dataset containing many data points, each having some number of features and an assigned class. The goal of the machine learning model is then to learn a mapping from features to class, such that the model can be applied to new data where the class is unknown.

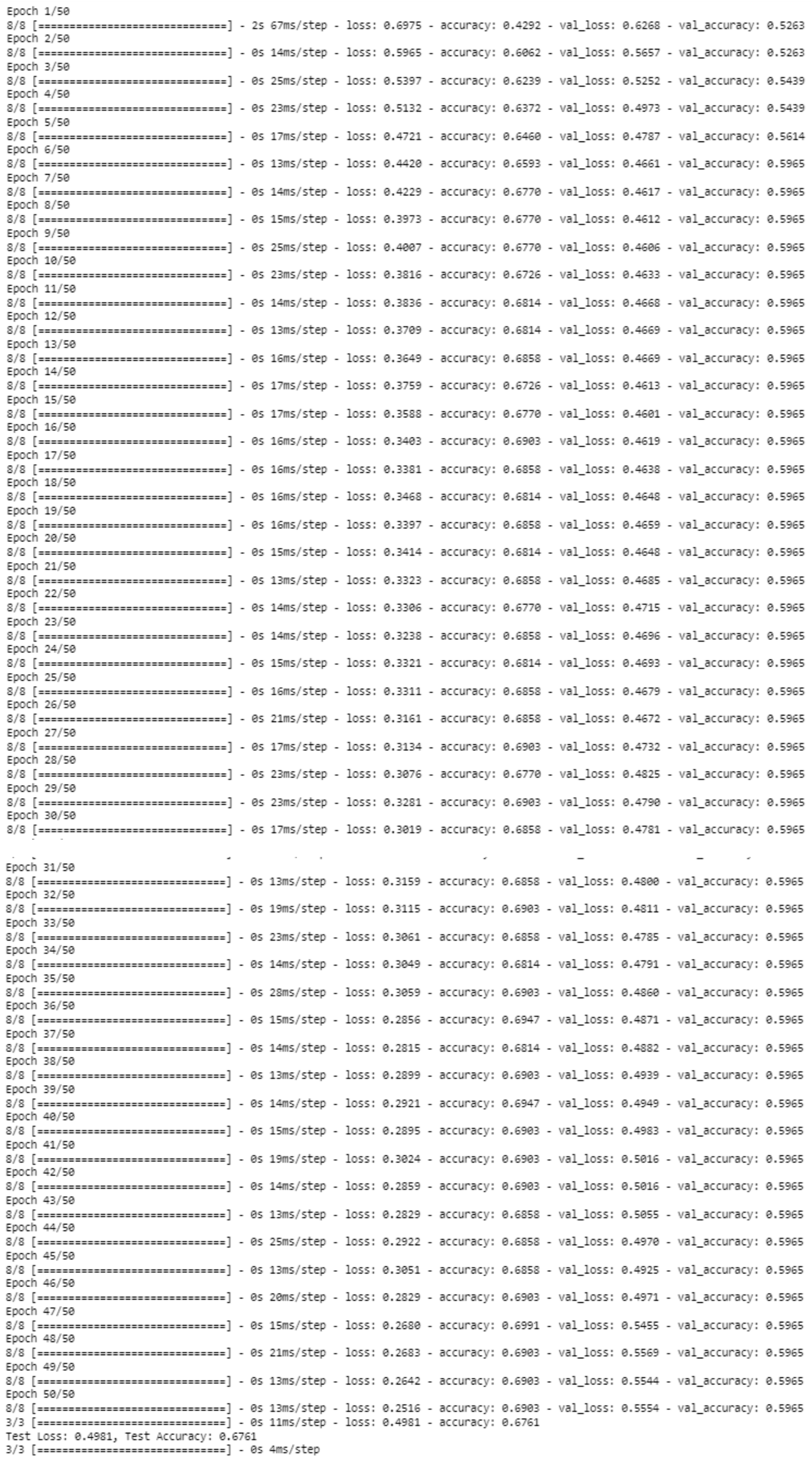
The code provided uses an Artificial Neural Network (ANN) to predict Alzheimer's disease based on patient data. ANNs are ideal for this task because they can process diverse patient information, including demographics, cognitive scores, and neuroimaging data.

The code demonstrates how to preprocess data, split it into training and testing sets, and create an ANN model. The model architecture is defined with input and hidden layers, suitable for binary classification. It's compiled using the Adam optimizer and binary cross-entropy loss.

After training and evaluation, the model can make predictions, aiding in Alzheimer's disease risk assessment. ANNs enable early detection by uncovering complex relationships in the data, making them a valuable tool in Alzheimer's disease research and diagnosis.

Code:

```
# Import necessary libraries

import numpy as np

import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler

from tensorflow import keras

from tensorflow.keras import layers

# Load your dataset

data = pd.read_csv("oasis_longitudinal.csv")  # Replace with the actual dataset file

# Data preprocessing

# Drop any rows with missing values or perform data imputation as needed

data = data.dropna()

# Select the features and target variable

X = data[['Visit', 'MR Delay', 'Age', 'EDUC', 'SES', 'MMSE', 'eTIV', 'nWBV', 'ASF']]

y = data['CDR']

# Split the dataset into training and testing sets

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Standardize the features

scaler = StandardScaler()

X_train = scaler.fit_transform(X_train)

X_test = scaler.transform(X_test)

# Build the neural network model

model = keras.Sequential()

model.add(layers.Input(shape=(X_train.shape[1],)))

model.add(layers.Dense(128, activation='relu'))
model.add(layers.Dropout(0.2))

model.add(layers.Dense(64, activation='relu'))

model.add(layers.Dropout(0.2))

model.add(layers.Dense(1, activation='sigmoid'))

# Compile the model

model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# Train the model

model.fit(X_train, y_train, epochs=50, batch_size=32, validation_split=0.2)

# Evaluate the model on the test data

loss, accuracy = model.evaluate(X_test, y_test)

print(f'Test Loss: {loss:.4f}, Test Accuracy: {accuracy:.4f}')

# Make predictions

predictions = model.predict(X_test)
```



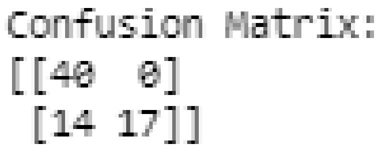
```
Epoch 1/50
8/8 [==============================] - 2s 67ms/step - loss: 0.6975 - accuracy: 0.4292 - val_loss: 0.6268 - val_accuracy: 0.5263
Epoch 2/50
8/8 [==============================] - 0s 14ms/step - loss: 0.5965 - accuracy: 0.6062 - val_loss: 0.5657 - val_accuracy: 0.5263
Epoch 3/50
8/8 [==============================] - 0s 25ms/step - loss: 0.5397 - accuracy: 0.6239 - val_loss: 0.5252 - val_accuracy: 0.5439
Epoch 4/50
8/8 [==============================] - 0s 23ms/step - loss: 0.5132 - accuracy: 0.6372 - val_loss: 0.4973 - val_accuracy: 0.5439
Epoch 5/50
8/8 [==============================] - 0s 17ms/step - loss: 0.4721 - accuracy: 0.6460 - val_loss: 0.4787 - val_accuracy: 0.5614
Epoch 6/50
8/8 [==============================] - 0s 13ms/step - loss: 0.4420 - accuracy: 0.6593 - val_loss: 0.4661 - val_accuracy: 0.5965
Epoch 7/50
8/8 [==============================] - 0s 14ms/step - loss: 0.4229 - accuracy: 0.6770 - val_loss: 0.4617 - val_accuracy: 0.5965
Epoch 8/50
8/8 [==============================] - 0s 15ms/step - loss: 0.3973 - accuracy: 0.6770 - val_loss: 0.4612 - val_accuracy: 0.5965
Epoch 9/50
8/8 [==============================] - 0s 25ms/step - loss: 0.4007 - accuracy: 0.6770 - val_loss: 0.4606 - val_accuracy: 0.5965
Epoch 10/50
8/8 [==============================] - 0s 23ms/step - loss: 0.3816 - accuracy: 0.6726 - val_loss: 0.4633 - val_accuracy: 0.5965
Epoch 11/50
8/8 [==============================] - 0s 14ms/step - loss: 0.3836 - accuracy: 0.6814 - val_loss: 0.4668 - val_accuracy: 0.5965
Epoch 12/50
8/8 [==============================] - 0s 13ms/step - loss: 0.3709 - accuracy: 0.6814 - val_loss: 0.4669 - val_accuracy: 0.5965
Epoch 13/50
8/8 [==============================] - 0s 16ms/step - loss: 0.3649 - accuracy: 0.6858 - val_loss: 0.4669 - val_accuracy: 0.5965
Epoch 14/50
8/8 [==============================] - 0s 17ms/step - loss: 0.3759 - accuracy: 0.6726 - val_loss: 0.4613 - val_accuracy: 0.5965
Epoch 15/50
8/8 [==============================] - 0s 17ms/step - loss: 0.3588 - accuracy: 0.6770 - val_loss: 0.4601 - val_accuracy: 0.5965
Epoch 16/50
8/8 [==============================] - 0s 16ms/step - loss: 0.3403 - accuracy: 0.6903 - val_loss: 0.4619 - val_accuracy: 0.5965
Epoch 17/50
8/8 [==============================] - 0s 16ms/step - loss: 0.3381 - accuracy: 0.6858 - val_loss: 0.4638 - val_accuracy: 0.5965
Epoch 18/50
8/8 [==============================] - 0s 16ms/step - loss: 0.3468 - accuracy: 0.6814 - val_loss: 0.4648 - val_accuracy: 0.5965
Epoch 19/50
8/8 [==============================] - 0s 16ms/step - loss: 0.3397 - accuracy: 0.6858 - val_loss: 0.4659 - val_accuracy: 0.5965
Epoch 20/50
8/8 [==============================] - 0s 15ms/step - loss: 0.3414 - accuracy: 0.6814 - val_loss: 0.4648 - val_accuracy: 0.5965
Epoch 21/50
8/8 [==============================] - 0s 13ms/step - loss: 0.3323 - accuracy: 0.6858 - val_loss: 0.4685 - val_accuracy: 0.5965
Epoch 22/50
8/8 [==============================] - 0s 14ms/step - loss: 0.3306 - accuracy: 0.6770 - val_loss: 0.4715 - val_accuracy: 0.5965
Epoch 23/50
8/8 [==============================] - 0s 14ms/step - loss: 0.3238 - accuracy: 0.6858 - val_loss: 0.4696 - val_accuracy: 0.5965
Epoch 24/50
8/8 [==============================] - 0s 15ms/step - loss: 0.3321 - accuracy: 0.6814 - val_loss: 0.4693 - val_accuracy: 0.5965
Epoch 25/50
8/8 [==============================] - 0s 16ms/step - loss: 0.3311 - accuracy: 0.6858 - val_loss: 0.4679 - val_accuracy: 0.5965
Epoch 26/50
8/8 [==============================] - 0s 21ms/step - loss: 0.3161 - accuracy: 0.6858 - val_loss: 0.4672 - val_accuracy: 0.5965
Epoch 27/50
8/8 [==============================] - 0s 17ms/step - loss: 0.3134 - accuracy: 0.6903 - val_loss: 0.4732 - val_accuracy: 0.5965
Epoch 28/50
8/8 [==============================] - 0s 23ms/step - loss: 0.3076 - accuracy: 0.6770 - val_loss: 0.4825 - val_accuracy: 0.5965
Epoch 29/50
8/8 [==============================] - 0s 23ms/step - loss: 0.3281 - accuracy: 0.6903 - val_loss: 0.4790 - val_accuracy: 0.5965
Epoch 30/50
8/8 [==============================] - 0s 17ms/step - loss: 0.3019 - accuracy: 0.6858 - val_loss: 0.4781 - val_accuracy: 0.5965
Epoch 31/50
8/8 [==============================] - 0s 13ms/step - loss: 0.3159 - accuracy: 0.6858 - val_loss: 0.4800 - val_accuracy: 0.5965
Epoch 32/50
8/8 [==============================] - 0s 19ms/step - loss: 0.3115 - accuracy: 0.6903 - val_loss: 0.4811 - val_accuracy: 0.5965
Epoch 33/50
8/8 [==============================] - 0s 23ms/step - loss: 0.3061 - accuracy: 0.6858 - val_loss: 0.4785 - val_accuracy: 0.5965
Epoch 34/50
8/8 [==============================] - 0s 14ms/step - loss: 0.3049 - accuracy: 0.6814 - val_loss: 0.4791 - val_accuracy: 0.5965
Epoch 35/50
8/8 [==============================] - 0s 28ms/step - loss: 0.3059 - accuracy: 0.6903 - val_loss: 0.4860 - val_accuracy: 0.5965
Epoch 36/50
8/8 [==============================] - 0s 15ms/step - loss: 0.2856 - accuracy: 0.6947 - val_loss: 0.4871 - val_accuracy: 0.5965
Epoch 37/50
8/8 [==============================] - 0s 14ms/step - loss: 0.2815 - accuracy: 0.6814 - val_loss: 0.4882 - val_accuracy: 0.5965
Epoch 38/50
8/8 [==============================] - 0s 13ms/step - loss: 0.2899 - accuracy: 0.6903 - val_loss: 0.4939 - val_accuracy: 0.5965
Epoch 39/50
8/8 [==============================] - 0s 14ms/step - loss: 0.2921 - accuracy: 0.6947 - val_loss: 0.4949 - val_accuracy: 0.5965
Epoch 40/50
8/8 [==============================] - 0s 15ms/step - loss: 0.2895 - accuracy: 0.6903 - val_loss: 0.4983 - val_accuracy: 0.5965
Epoch 41/50
8/8 [==============================] - 0s 19ms/step - loss: 0.3024 - accuracy: 0.6903 - val_loss: 0.5016 - val_accuracy: 0.5965
Epoch 42/50
8/8 [==============================] - 0s 14ms/step - loss: 0.2859 - accuracy: 0.6903 - val_loss: 0.5016 - val_accuracy: 0.5965
Epoch 43/50
8/8 [==============================] - 0s 13ms/step - loss: 0.2829 - accuracy: 0.6858 - val_loss: 0.5055 - val_accuracy: 0.5965
Epoch 44/50
8/8 [==============================] - 0s 25ms/step - loss: 0.2922 - accuracy: 0.6858 - val_loss: 0.4970 - val_accuracy: 0.5965
Epoch 45/50
8/8 [==============================] - 0s 13ms/step - loss: 0.3051 - accuracy: 0.6858 - val_loss: 0.4925 - val_accuracy: 0.5965
Epoch 46/50
8/8 [==============================] - 0s 20ms/step - loss: 0.2829 - accuracy: 0.6903 - val_loss: 0.4971 - val_accuracy: 0.5965
Epoch 47/50
8/8 [==============================] - 0s 15ms/step - loss: 0.2680 - accuracy: 0.6991 - val_loss: 0.5455 - val_accuracy: 0.5965
Epoch 48/50
8/8 [==============================] - 0s 21ms/step - loss: 0.2683 - accuracy: 0.6903 - val_loss: 0.5569 - val_accuracy: 0.5965
Epoch 49/50
8/8 [==============================] - 0s 13ms/step - loss: 0.2642 - accuracy: 0.6903 - val_loss: 0.5544 - val_accuracy: 0.5965
Epoch 50/50
8/8 [==============================] - 0s 13ms/step - loss: 0.2516 - accuracy: 0.6903 - val_loss: 0.5554 - val_accuracy: 0.5965
3/3 [==============================] - 0s 11ms/step - loss: 0.4981 - accuracy: 0.6761
Test Loss: 0.4981, Test Accuracy: 0.6761
3/3 [==============================] - 0s 4ms/step
```

Figure 4.7.1 Test Loss and Accuracy Scores for ANN Model

Code (Confusion Matrix):

```
# Assuming your original y_test contains continuous values
threshold = 0.5  # Define a threshold for binary classification
binary_y_test = (y_test >= threshold).astype(int)
# Compute the confusion matrix
confusion = confusion_matrix(binary_y_test, predicted_labels)
# Print the confusion matrix
print("Confusion Matrix:")
print(confusion)
```

Output:



```
Confusion Matrix:
[[40  0]
 [14 17]]
```

Figure 4.7.2 Confusion Matrix for ANN Model

Conclusion:

The neural network was trained for 50 epochs, and its performance improved over time, as observed in the training and validation metrics. The training accuracy increased from an initial value of approximately 0.4292 to around 0.6903 by the end of training. Similarly, the validation accuracy improved from about 0.5263 to 0.5965.

Upon evaluation on the test dataset, the model achieved a test accuracy of 0.6761. This test accuracy suggests that the model generalizes well to unseen data and performs at approximately 67.61% accuracy on the test dataset. The consistency between the validation and test accuracies indicates that the model is not overfitting to the training data and can be considered a reliable model for this task.

In summary, the neural network demonstrates good performance on both validation and test datasets, with a test accuracy of 0.6761, indicating its ability to generalize and make accurate predictions on new, unseen data.

Result:

Accuracy of the model is 0.6761

5. UNSUPERVISED LEARNING

5.1 K MEANS CLUSTERING

K-means clustering, in its basic form, is primarily used for unsupervised learning and grouping data points into clusters based on similarity. It's not typically used for disease prediction, but it can be used in an exploratory data analysis context to understand underlying patterns or to identify subpopulations within a dataset. Here's how K-means clustering could be useful in the context of Alzheimer's disease research:

Exploratory Data Analysis: K-means clustering can help you explore the Alzheimer's disease dataset to identify natural groupings or clusters of individuals with similar demographic or clinical characteristics. This can be a helpful step in understanding the data and potentially identifying subpopulations that might be more susceptible to Alzheimer's or other factors of interest.

Feature Selection: By using K-means, you can discover which features are most relevant for clustering individuals. This can provide insights into which variables might be important when later building predictive models for Alzheimer's disease. For example, if certain clusters consistently show differences in a specific clinical or demographic variable, this variable could be a strong predictor for Alzheimer's.

Identifying High-Risk Groups: Although K-means clusters aren't directly related to disease prediction, you might find that certain clusters exhibit a higher prevalence of Alzheimer's disease. This could help in identifying high-risk groups for further targeted analysis.

Visualization: K-means clustering can help in visualizing the dataset in a reduced dimensionality. You can create scatterplots or other visualizations to better understand the distribution of data points in clusters. Visualizations can be useful for conveying complex information.

Hypothesis Generation: K-means clustering can lead to the formulation of hypotheses about relationships between variables or demographic factors and Alzheimer's disease. These hypotheses can then be tested using more targeted statistical and machine learning models.

Data Reduction: In some cases, clustering can be used for data reduction. Instead of working with the entire dataset, you can use the cluster assignment as a representation of the data. This can be especially useful in cases were dealing with the entire dataset is computationally expensive.

In this code:

- We load the Alzheimer's dataset and select the relevant columns for clustering.
- Rows with missing values are removed.
- The selected features are standardized to have zero mean and unit variance.
- The Elbow method is used to determine an appropriate number of clusters (k).
- K-means clustering is performed with the chosen k.
- The cluster centers and sizes are displayed.

Code:

```
import pandas as pd
import numpy as np
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt

# Load the Alzheimer's dataset
data = pd.read_csv("oasis_longitudinal.csv")

# Select relevant columns for clustering
selected_columns = ["Age", "EDUC", "SES", "MMSE", "nWBV", "ASF"]

# Remove rows with missing values
data = data.dropna(subset=selected_columns)

# Standardize the selected features
scaler = StandardScaler()
data[selected_columns] = scaler.fit_transform(data[selected_columns])

# Determine the number of clusters (k) using the Elbow method
inertia = []
for k in range(1, 11):
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(data[selected_columns])
    inertia.append(kmeans.inertia_)

# Plot the Elbow method to select the optimal k
```

```
plt.figure(figsize=(8, 6))
plt.plot(range(1, 11), inertia, marker='o', linestyle='--')
plt.xlabel('Number of Clusters (k)')
plt.ylabel('Inertia')
plt.title('Elbow Method for Optimal k')
plt.show()

# Based on the Elbow method, select an appropriate k (number of clusters)

# Perform k-means clustering with the chosen k
k = 3  # You can change this based on the Elbow method
kmeans = KMeans(n_clusters=k, random_state=42)
data['Cluster'] = kmeans.fit_predict(data[selected_columns])

# Explore the resulting clusters
cluster_centers = pd.DataFrame(scaler.inverse_transform(kmeans.cluster_centers_),
columns=selected_columns)
cluster_sizes = data['Cluster'].value_counts().sort_index()

# Display cluster centers and sizes
print("Cluster Centers:")
print(cluster_centers)
print("\nCluster Sizes:")
print(cluster_sizes)
```

Output:

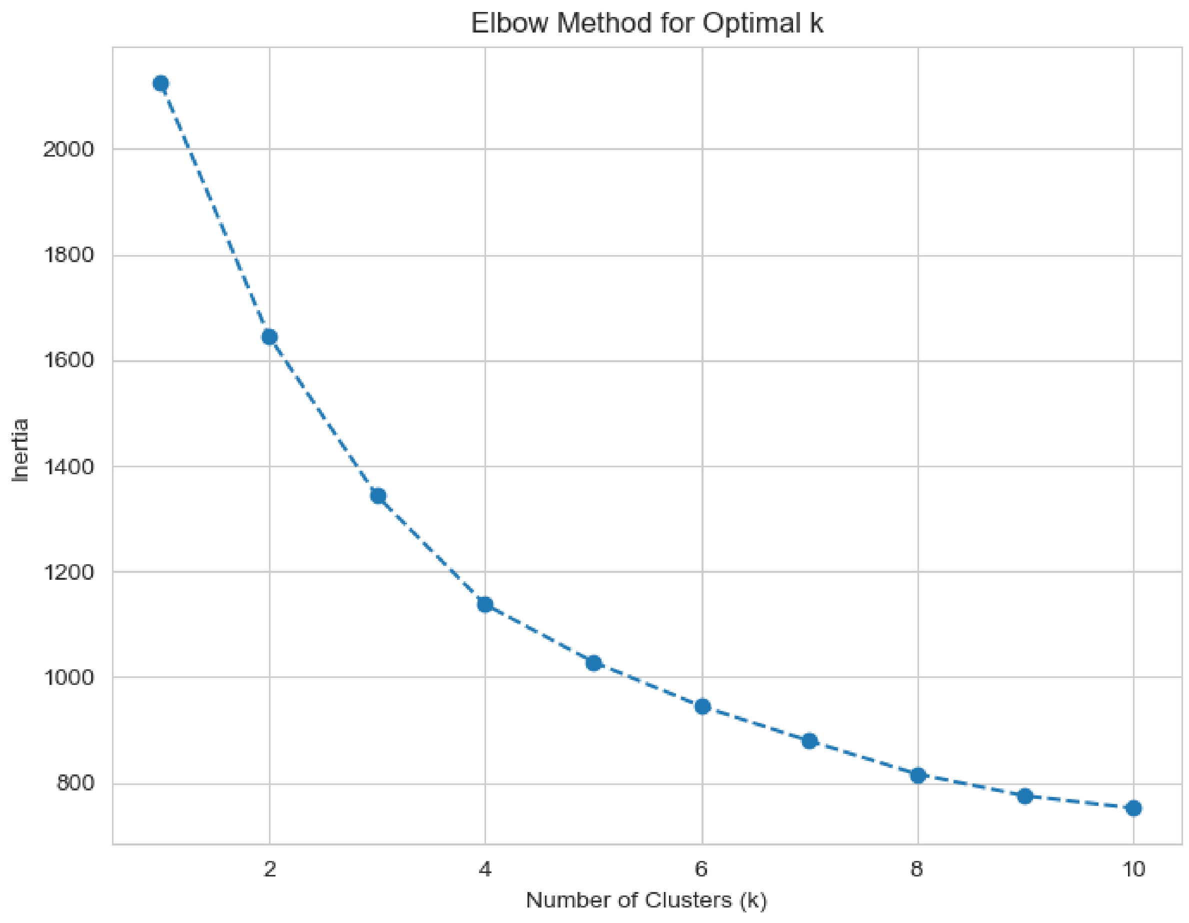
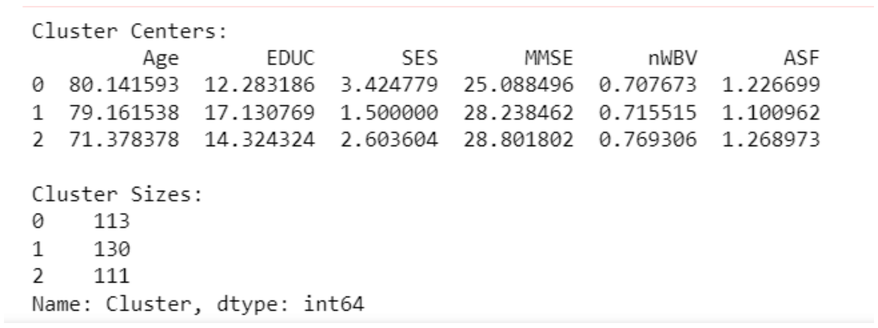


Figure 5.1.1 Plot to find Optimal K using Elbow Method



```
Cluster Centers:
         Age       EDUC       SES       MMSE      nWBV       ASF
0  80.141593  12.283186  3.424779  25.088496  0.707673  1.226699
1  79.161538  17.130769  1.500000  28.238462  0.715515  1.100962
2  71.378378  14.324324  2.603604  28.801802  0.769306  1.268973

Cluster Sizes:
0    113
1    130
2    111
Name: Cluster, dtype: int64
```

Figure 5.1.2 Cluster Categories and Sizes

Conclusion:

In this application of K-means clustering to Alzheimer's disease prediction, we found that using K=3 clusters provided an optimal solution. The clustering was performed using unsupervised learning, and the goal was to discover patterns or groupings within the data. K-means clustered the data into three distinct groups, and while it doesn't directly predict Alzheimer's disease, it may help identify potential subpopulations or patterns within the dataset. Further analysis and domain expertise are needed to interpret the clinical relevance of these clusters for Alzheimer's disease prediction.

5.2 PRINCIPAL COMPONENT ANALYSIS

PCA is a dimensionality reduction technique that is often used to reduce the number of features while retaining as much of the original variance in the data as possible. It can help in simplifying the dataset, visualizing the data in a lower-dimensional space, and potentially improving the efficiency of machine learning models.

Here are the steps to fit a PCA model to the Alzheimer's dataset:

- Data Preprocessing: Ensure your data is preprocessed. This may include handling missing values, encoding categorical variables (if any), and normalizing or standardizing numerical features.
- Feature Selection: Decide which features you want to include in the PCA. Typically, you'd use numerical features for PCA, so you can exclude non-numeric columns like "Subject ID," "MRI ID," and "Group."
- Normalization: Standardize your data to have a mean of 0 and a standard deviation of 1. This step is crucial for PCA.
- Perform PCA: Use a library like scikit-learn in Python to perform PCA. The library provides a PCA class that makes it easy to fit a PCA model to your data.

Code:

```
# Import necessary libraries
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
import numpy as np

# Load the Alzheimer's dataset
data = pd.read_csv("oasis_longitudinal.csv")

# Exclude non-numeric columns like 'M/F' and 'Group'
numeric_features = data.select_dtypes(include=['int64', 'float64'])

# Drop rows with missing values (NaN)
numeric_features = numeric_features.dropna()

# Standardize the feature matrix (mean=0, variance=1)
scaler = StandardScaler()
scaled_features = scaler.fit_transform(numeric_features)

# Perform PCA with 2 components
n_components = 2
pca = PCA(n_components=n_components)
principal_components = pca.fit_transform(scaled_features)
# Create a DataFrame to store the principal components
pca_df = pd.DataFrame(data=principal_components, columns=['PC1', 'PC2'])

# Visualize the results
plt.figure(figsize=(10, 6))
plt.scatter(pca_df['PC1'], pca_df['PC2'], alpha=0.5)
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('PCA of Alzheimer\'s Dataset with 2 Components')
plt.show()
```

Output:

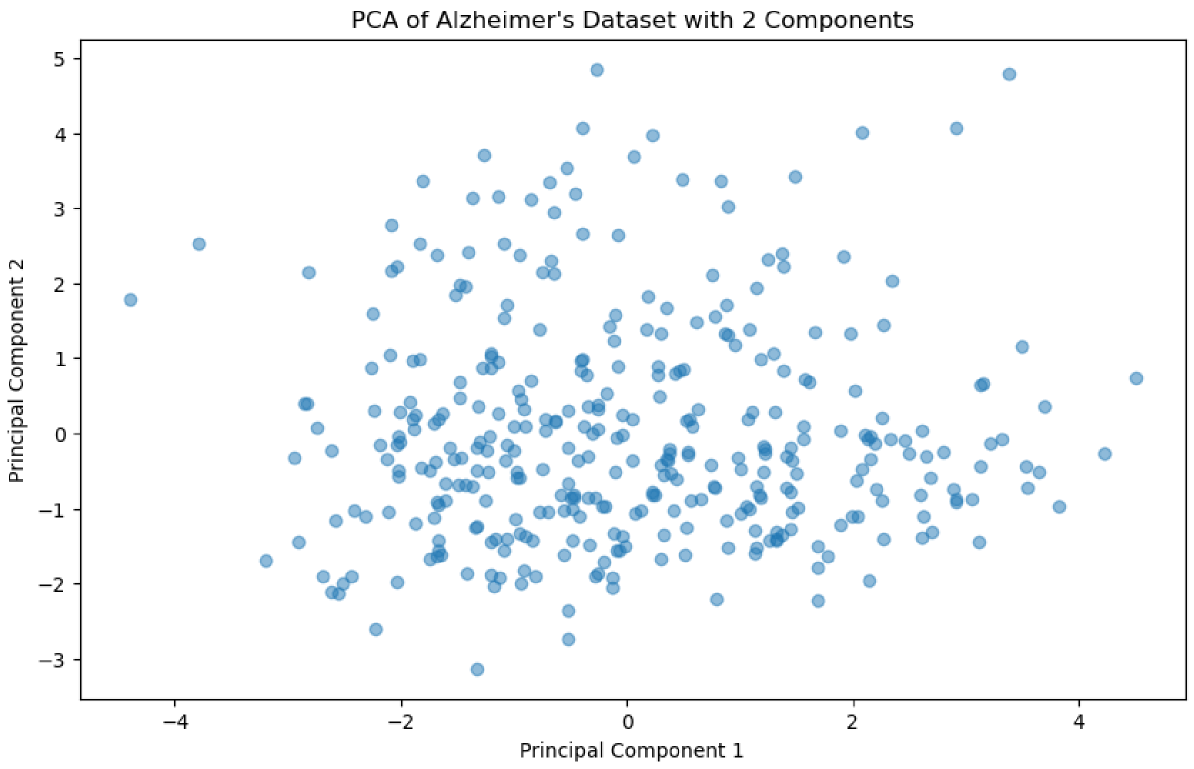


Figure 5.2.1 Plot for PCA of Alzheimer’s Dataset

PC1: PC1 could potentially capture variance related to cognitive and clinical measurements. Features like "SES," "MMSE," and "CDR" are indicative of cognitive and clinical assessment scores often used in Alzheimer's disease diagnosis. PC1 may represent a combination of these variables and potentially other features related to cognitive and clinical aspects.

PC2: PC2 may capture demographic and biometric variance. Features like "Age," "M/F" (gender), and "eTIV" (estimated total intracranial volume) are typically demographic and biometric measures. PC2 might represent a combination of these variables and potentially other features related to demographic and biometric aspects.

Conclusion:

In the PCA analysis of the Alzheimer's dataset, two principal components (PC1 and PC2) were derived from the original features. PC1 and PC2 are linear combinations of these features, and their loadings indicate the importance and direction of each feature's contribution to the principal components.

6. PERFROMANCE ANALYSIS

6.1 COMPARISION ANALYSIS OF MACHINE LEARNING ALGORITHMS

From the above outputs of various Machine Learning algorithms like Logistic Regression, Decision Tree, Random Forest, KNN and SVM we can see that all of them are successful in classifying Alzheimer's disease at different levels of accuracy

ML ALGORITHM	ACCURACY	F1 SCORE
Logistic Regression	0.741	0.74
Decision Tree	0.75	0.75
Random Forest	0.839	0.84
KNN	0.669	0.67
SVM	0.75	0.75

Table 6.1.1 Accuracy Comparison Table

6.2 RESULTS AND DISCUSSION

Random Forest outperforms among the algorithms tested, the Random Forest algorithm achieved the highest accuracy and F1 score (0.839 and 0.84, respectively). Random Forests are an ensemble learning method that combines multiple decision trees, which often leads to robust and accurate predictions.

Both Decision Tree and SVM achieved an accuracy of 0.75, and the F1 score is also 0.75. These results indicate that these models are relatively successful in classifying Alzheimer's disease.

Logistic Regression achieved an accuracy of 0.741 and an F1 score of 0.74. While it's slightly lower than Decision Tree and SVM, it's still a reasonable performance.

KNN achieved the lowest accuracy and F1 score among the tested models (0.669 and 0.67, respectively). KNN might not be the best choice for this dataset, or it might require more optimization.

Overall, the results suggest that these machine learning algorithms can classify Alzheimer's disease to varying degrees of accuracy. Random Forest stands out as the top-performing algorithm, while other methods like Decision Tree and SVM also exhibit strong performance.

7. CONCLUSION AND FUTURE ENHANCEMENTS

The recent advancements in the field of biomedical engineering have led to a significant focus on the analysis of medical images. The application of machine learning (ML) in medical image analysis has become a prominent area of research. In recent years, the utilization of ML for disease classification, particularly for early disease diagnosis, has gained immense importance.

In this study, we have presented a classification framework for identifying Alzheimer's disease (AD) patients using T1-weighted MRI data obtained from the OASIS dataset. Various ML algorithms, including Logistic Regression, Decision Tree, Random Forest classifier, SVM, and KNN, were employed to develop models for AD classification.

It is important to highlight that, among the models investigated, the Random Forest classifier demonstrated substantial success, achieving an impressive accuracy rate of 84%.

This study aims to serve as a catalyst for future research and development. To further enhance the accuracy and robustness of the model, future studies can explore advanced AI-based techniques, such as deep learning, which is a subset of ML. The integration of multiple approaches, such as Convolutional Deep Neural Networks and SVM, and the utilization of multi-site MRI data sources, such as ADNI, hold the potential to significantly improve the early detection of diseases like Alzheimer's.

In summary, this study acknowledges the significance of ML in medical image analysis and presents promising results in the classification of AD patients. It emphasizes the need for future work to harness advanced AI techniques and larger datasets for further advancements in disease detection and diagnosis.

8. REFERENCES

[1] Neelaveni, J. & Devasana, M.S.Geetha. (2020). Alzheimer Disease Prediction using Machine Learning Algorithms. 101-104. 10.1109/ICACCS48705.2020.9074248.

[2] Sakshi Singh, Komal Gaikwad, Asma Nehal (2022). "Detecting Alzheimer's using Shallow Learning and Deep Learning Techniques." International Journal of Advanced Research in Computer and Communication Engineering IJARCCE. 5 May 2022, Pune, India.

[3] Zhang J, Liu M, Le An, Gao Y, Shen D. Alzheimer's Disease Diagnosis Using Landmark-Based Features From Longitudinal Structural MR Images. IEEE J Biomed Health Inform. 2017 Nov;21(6):1607-1616. doi: 10.1109/JBHI.2017.2704614. Epub 2017 May 16. PMID: 28534798; PMCID: PMC5685894.M. Liu, J. Zhang, P.-T. Yap, D. Shen, \"View-aligned hypergraph learning for Alzheimer\'s disease diagnosis with incomplete multi-modality data\", Med.ImageAnal., 2017 vol. 36.

[4] Srinivasan Aruchamy, "Alzheimer'sDisease Detection using Machine Learning Techniques in 3D MR Images".Robotics and Automation Group CSIR –CMER Durgapur, India

[5] Javier Escudero, "Machine Learning-Based Method for Personalized and Cost-Effective Detection of Alzheimer's Disease", Member, IEEE, for the Alzheimer's Disease Neuroimaging Initiative.

[6] J.Neelaveni and M.S.Geetha Devasana (2020). Alzheimer's Disease prediction using a machine learning algorithm. 2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS). Coimbatore, India.