

**AMRITA VISHWA VIDYAPEETHAM**

**MACHINE LEARNING - 22AIE213**

**PROJECT REPORT**

**A Novel Approach Utilizing Machine Learning for the Early Diagnosis of Alzheimer's Disease**

**Paper:** Uddin, K.M.M., Alam, M.J., Jannat-E-Anawar et al. A Novel Approach Utilizing Machine Learning for the Early Diagnosis of Alzheimer's Disease. Biomedical Materials & Devices 1, 882–898 (2023). [doi: 10.1007/s44174-023-00078-9](https://doi.org/10.1007/s44174-023-00078-9)

Submitted By:

P Bhuvaneshwari AV.EN.U4AIE22030

School of Computing,

Amrita Vishwa Vidyapeetham, Amaravati

Course Instructor: Dr Madhusudana Rao N.

**Introduction**

The increase in life expectancy and the prevalence of age-related cognitive disorders have led to great interest in studying normal and pathological aging with the aim to individuate early predictors of degenerative disorders, differential diagnosis, and efficacies of pharmacological and cognitive approaches in the treatment of these disorders. Indeed, considering the great burden of degenerative diseases on national healthcare systems in terms of cost and therapies, research aimed at improving the early and differential diagnosis of these pathologies is mandatory.[1]

Alzheimer’s disease (AD) is the most common form of dementia, which is a progressive brain disorder mostly occurring in the late life [2]. Comparing with the patient’s previous functions, a decline in memory and other cognitive functions is noted as a primary dementia syndrome. In 2006, the worldwide prevalence of AD was 26.6 million, and this number is expected to double in every 20 years. By 2046, 1.2% of the global population will be affected by AD [3]. The early diagnosis of AD is primarily associated to the detection of Mild Cognitive Impairment (MCI), a prodromal stage of AD. Though the memory complaints and deficits of MCI do not notably affect the patients’ daily activities, it has been reported that MCI has a high risk of progression to AD or other forms of dementia [4]. The accurate early diagnosis AD, especially identifying the risk of progression of MCI to AD, affords the AD patients awareness of the severity and allows them to take prevention measures, e.g., lifestyle changing and medications [5].

Many machine learning methods have been proposed to aid the diagnosis of AD based on high dimensional features extracted from various neuroimaging biomarkers, subclass MRI and PET. These machine learning methods not only need to identify the AD subjects from the normal control (NC) subjects automatically, but also predict the risk of MCI subjects evolving to AD, thus MCI instances can be labeled as MCI non-converters (ncMCI) or MCI converters (cMCI), depending on the risk of progression. Therefore, the early diagnosis of AD can be naturally modeled to be a multiclass classification problem

Some previous studies simplified the problem into a binary classification task [6-8]. The workflow in [6] combined features from multiple biomedical modalities using a multi-kernel SVM classifier. However it is difficult for SVM to classify subjects with more than two classes in one setting. Some methods embedded the prior knowledge in the model designing [9-11]. For example, an optimized graph cut algorithm was proposed in [9] with parameters adjusted corresponding to the distribution of particular classes in the training dataset. The dependence of prior knowledge may be also sensitive to the changes of the dataset and hard to configure.

Some of the common challenges in the early stage of Alzheimer's disease include:

• Diffculty recalling recent conversations or events.

• Frequently misplacing items.

• Struggling to remember the names of places and things.

• Having trouble fnding the right words.

• Making poor judgments or struggling to make decisions.

• Becoming less adaptable and more resistant to change.

• Experiencing memory issues that interfere with daily activities.

• Finding it difcult to solve problems or plan ahead.

• Having trouble completing routine tasks.

• Confusion about time or location.

• Losing track of items and the ability to recall past events

**Literature Survey**

|  |  |  |  |
| --- | --- | --- | --- |
| Paper | Research Gaps | Efficient Model Employed | Accuracy |
| Helaly et al. [12] | Need for validation on larger and more diverse populations. | CNN with VGG19 pre-trained model. | 97% |
| Kavitha et al. [13] | Limited exploration of real-world applicability and generalizability of the model. | Gradient Boosting, SVM, Decision Tree, and Voting classifiers. | ~83% |
| Ghazal et al. [14] | Need for enhancing model robustness across different MRI datasets. | Transfer learning. | 91.70%. |
| Gaudiuso et al. [15] | Limited sample size and need for validation with larger datasets. | LIBS with machine learning | 80% |
| Nawaz et al. [16] | Need for further exploration of deep learning models' interpretability | Deep feature-based CNN using pre-trained AlexNet. | 99.21% |
| Basheer et al. [17] | Need for improved generalizability to different populations and settings. | Classification model with feature correlation | 92.39% |
| Prajapati et al. [18] | Need for addressing overfitting and enhancing model generalization. | DNN | ~78.27% |
| Lucas et al. [19] | Need for validation on larger datasets and across different populations. | qEEG processing technique. | 87.0% |
| KN Rao et al. [20] | Limited integration of diverse neuroimaging modalities and machine learning techniques for early and accurate Alzheimer's disease detection. | SVM | 97.47% |
| R Franciotti et al. [21] | Lack of consensus on the stability and interpretability of machine learning algorithms across diverse biomarkers and clinical conditions for predicting Mild Cognitive Impairment (MCI) progression to Alzheimer’s Disease. | XGB | 88% |
| KR Kruthika et al. [22] | Limited development of accurate and efficient diagnostic tools for Alzheimer's disease, particularly integrating advanced image processing and machine learning techniques. | BN + SVM + KNN + PSO | 96.31 ± 1.22 % |
| C Kavitha et al. [23] | Limited exploration and validation of machine learning models for predicting Alzheimer's disease progression using diverse and real-world clinical datasets beyond OASIS, potentially impacting broader applicability and generalizability in clinical settings. | Voting classifier | 85.12% |
| G Gupta et al [24] | Limited exploration of diverse datasets beyond OASIS in evaluating AI-based machine learning techniques for early Alzheimer’s disease detection and classification, despite notable classifier performance with Random Forest. | Random forest | 86.8% |
| A D Arya et al. [25] | need for advanced techniques in feature selection and engineering to enhance the predictive performance of machine learning models for Alzheimer's disease. | XGBoost Tree | 96.75% |
| P Rani et al. [26] | limited demographic diversity and potential biases in existing neuroimaging datasets, which may hinder the generalization of the machine learning model to real-world clinical scenarios. | RF | 87.84% (Imbalanced dataset)  94.03%  (Balanced dataset) |

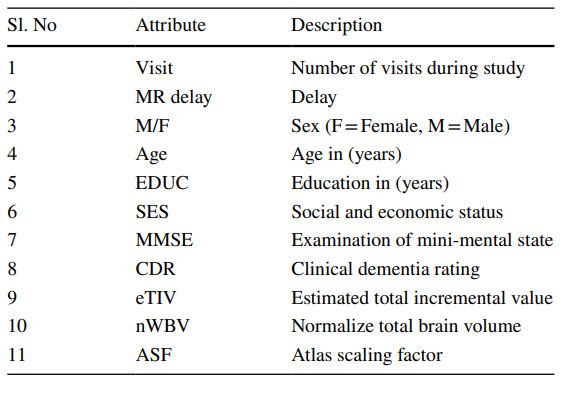
**Material and Methods:**

**Dataset Description:**

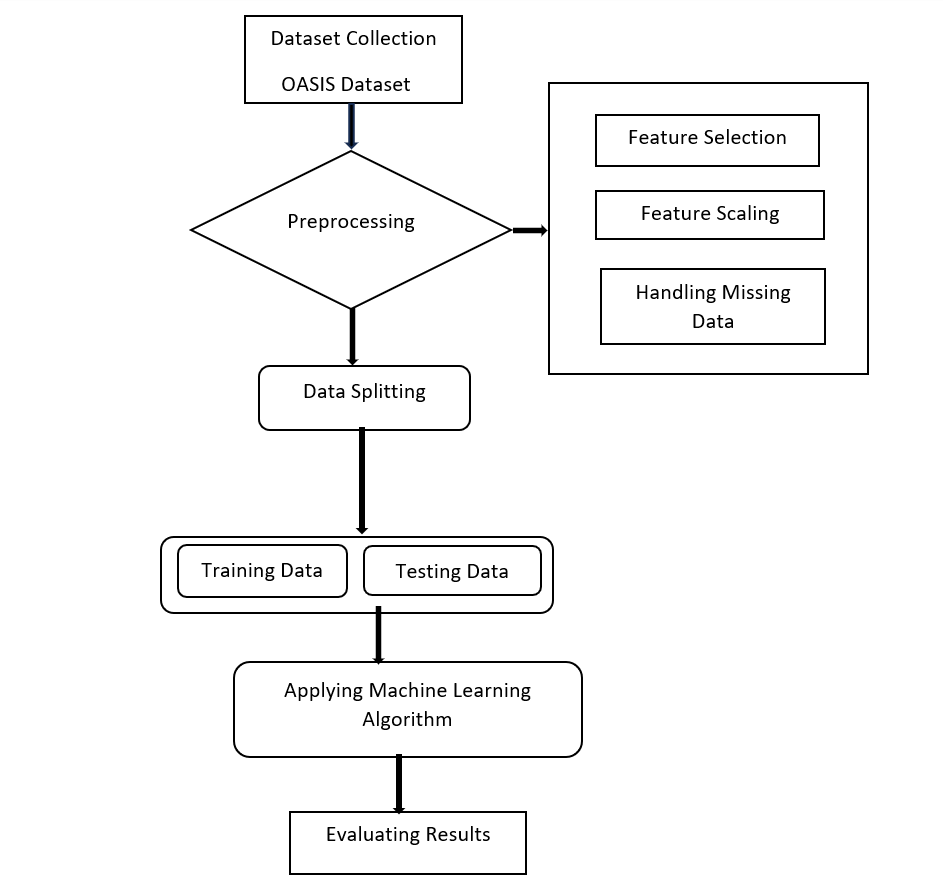
The dataset is available on Kaggle. (Link - [MRI and Alzheimers (kaggle.com)](https://www.kaggle.com/datasets/jboysen/mri-and-alzheimers?select=oasis_longitudinal.csv))

The dataset used in this study is Longitudinal MRI Data in NonDemented and Demented Older Adults.

This set consists of a longitudinal collection of 150 subjects aged 60 to 96. Each subject was scanned on two or more visits, separated by at least one year for a total of 373 imaging sessions. For each subject, 3 or 4 individual T1-weighted MRI scans obtained in single scan sessions are included. The subjects are all right-handed and include both men and women. 72 of the subjects were characterized as nondemented throughout the study. 64 of the included subjects were characterized as demented at the time of their initial visits and remained so for subsequent scans, including 51 individuals with mild to moderate Alzheimer’s disease. Another 14 subjects were characterized as nondemented at the time of their initial visit and were subsequently characterized as demented at a later visit.



**Workflow:**



**Details of the Hardware and Software used:**

**Hardware Used:**

1. Computing device (Computer/Server)
2. Storage (Hard Drive or SSD)

**Software Used:**

1. IDE: Jupyter Notebook
2. Environment Used: python
3. Machine Learning Libraries:

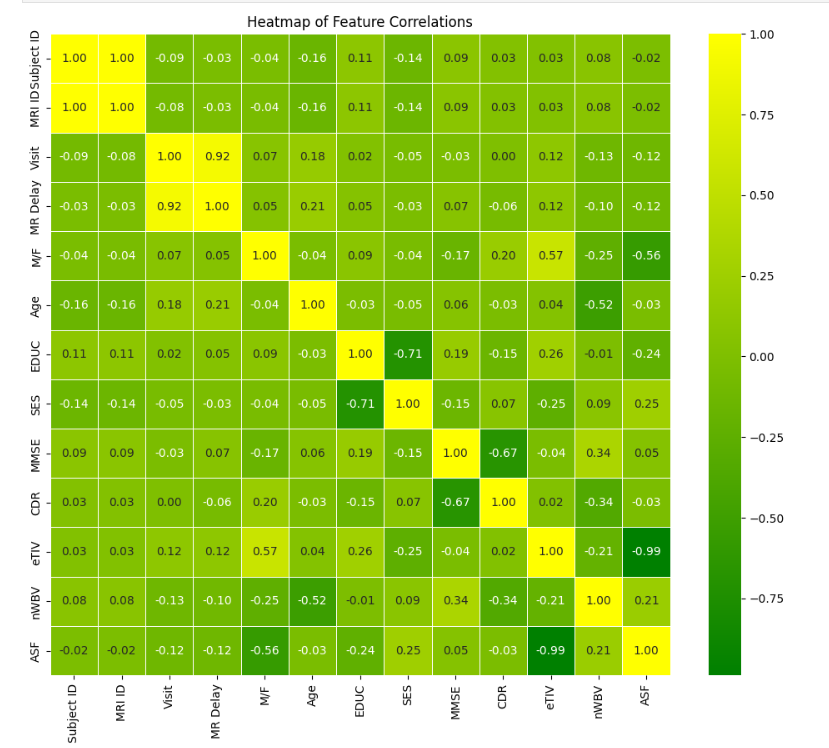
* NumPy and Pandas
* Matplotlib and seaborn
* Scikit-learn
* Imbalanced-learn

**Dataset Split-up:**

* The data is categorised into two Data Frames: X(Features) and Y(Target).
* The ratio of dividing data is 80:20 randomly which means 80% of the data are used to train the model and 20% for testing..
* In this Dataset, the Target feature is labeled with numerical values.

**Data Preprocessing, Scaling and Data Visualisation:**

* The dataset retrieved from the source had been pre-processed to numerical values, with no missing data. We have converted object datatype features to numeric datatype features. To avoid biases, all features are scaled to a range using mapping technique.
* For observing relation between the features various plots were analysed. The Correlation Heatmap below displays the relation between features.

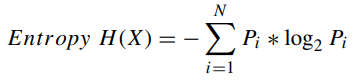


* This Heatmap shows that features are not correlated. Hence, these features should be used for constructing the model.

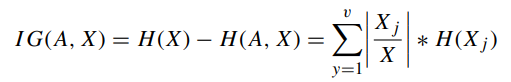
**Details of Models Proposed:**

**Decision Tree:**

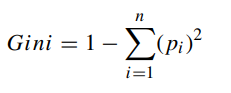
Another supervised probabilistic decision-making algorithm (Decision tree) [27] is used for regression and classification of targets based on the training features. The probability tree is penned down using the top-down divide and conquer method. The root of the tree is selected on the variable that provides the maximum information gain, mathematically represented in Eqs. (1), (2) and the next nodes are recursively selected in this manner. The goal is to make the tree as small as possible. Different attribute selection method is applied for selecting the node like GINI Index, shown in Eq. (3)

 (1)

where X, N and Pi represent the set of all instances in the dataset, number of distinct class values and even probability respectively.

 (2)

where H(X) Entropy of dataset X, |Xj| number of instances with y value of an attribute A, |X| total number of instances in dataset X, v set of distinct value of an attribute A, H(Xj) entropy of subset of instance for attribute A, H(A,X) entropy of an attribute A.

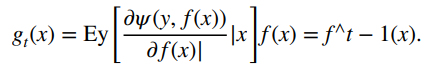
 (3)

**Random Forest:**

Many times, a single tree is not effective to capture mapping to target variables with observations. Therefore, supervised ensemble method such as random forest [27] is also applied over decision trees for decision making in which some number of randomly created decision trees are employed in a small sample of training data and majority decision is used for prediction/classification of target variables.

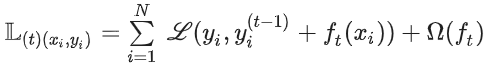
**Gradient Boosting:**

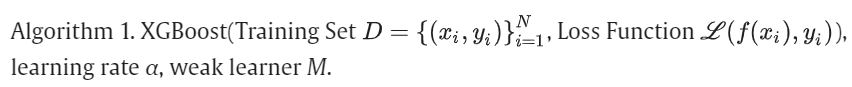
Both the base-learner models and the loss function are freely securable. When given a particular loss function (y, f) and/ or a particular base-learner (x, θt), the answer to the parameter estimates could be complicated to calculate in actuality [29]. It was suggested to address this by choosing a diferent functionh(x, θt ) that is most parallel to the observed data's negative gradient, gt(xi)Ni=1.

**** (4)

**XGBoost**:

XGBoost [28] approximates a function by optimizing a specific loss function where different regularization methods are applied. The algorithm for XGBoost is given in Algorithm 1. The weak learner learned using an objective (loss function and regularization) function at iteration *t* as the following:

 (5)



**Voting classifier:**

One of the simplest methods of merging the forecasts from several learning algorithms is by voting. Voting classifers are not really classifers, but rather wrappers for multiple ones that are trained and evaluated concurrently to beneft from their unique qualities. To predict the fnal result, datasets are trained using various algorithms and ensembles. A qualifed majority on a forecast can be obtained in two ways: Hard Voting Hard voting is the simplest type of majority voting. The class with the most votes (Nc) will be selected in this instance. The majority vote of each classifer is used to make prediction [30]. In Eq. (6), the class label j is predicted using majority (plurality) voting of each classifer M

 (6)

**CatBoost:**

CatBoost proposed by Yandex [31] is a high-performance machine learning algorithm for gradient boosting on decision trees, which attempts to handle categorical and ordered features using a permutation driven alternative compared with the classical algorithms. Thus, the modified target-based statistics offers a more effective implementation with lower computational complexity, and Bayesian optimization was performed to overcome model overfitting. The CatBoost uses greedy search to create a strong competitive model by combining many weak models sequentially. The decision trees are fitted sequentially by ordered boosting; each tree is learned using information from former trees to reduce the errors. Unlike gradient boosting models, CatBoost employs the oblivious tree procedure which generates a simple fitting scheme and high computational efficiency, and uses loss function change to rank the feature importance of the developed model.

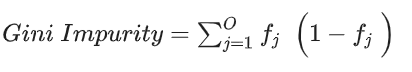
**Extra Trees:**

Learning in this method is done through the aggregation of outputs received from varied non-correlated decision trees assembled in a forest to yield the results of segregation. The way it operates is synonymous with the forest's random. However, it differentiates specifically from this in the way how the DTs are constructed in the forest. The DT is built in Extra Trees Forest, where it is generated from training at its initial stages. It is required to choose some prominent features to categorize the stats depending upon specific mathematical norms, which are done after every node of the test, followed by a random selection of k-features allotted to each tree from the pre-decided set of features. This allotment of features assists the further formulation of new de-correlated DTs [32].

The same forest structure is used to ascertain feature selection at the time of forest construction as for every selected feature; there is an overall decrease in the criteria of mathematics being used in split feature decisions. This is what we term “The Featured Gini Importance.” The user picks the high-ranked k-feature depending upon his/her choices for the feature selection; all of the features are organized in descending sequence according to the GINI value of every feature.

The classification of extra tree classifier uses Gini impurity by default as in equation (7) and uses Entropy as an alternative for the classification as in equation (8), whereas for regression, [Mean Square Error](https://www.sciencedirect.com/topics/engineering/mean-square-error) and [Mean Absolute Error](https://www.sciencedirect.com/topics/engineering/mean-absolute-error) are used:

 (7)

(8)

**Program:**

**Importing necessary libraries**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.impute import SimpleImputer

from sklearn.feature\_selection import SelectKBest, f\_classif

from sklearn.preprocessing import LabelEncoder, StandardScaler

from sklearn.model\_selection import train\_test\_split, cross\_val\_score, Kfold

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score, confusion\_matrix

from imblearn.combine import SMOTEENN

from sklearn.naive\_bayes import GaussianNB

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier, ExtraTreesClassifier, BaggingClassifier

import xgboost as xgb

from catboost import CatBoostClassifier

import warnings

from sklearn.exceptions import UndefinedMetricWarning

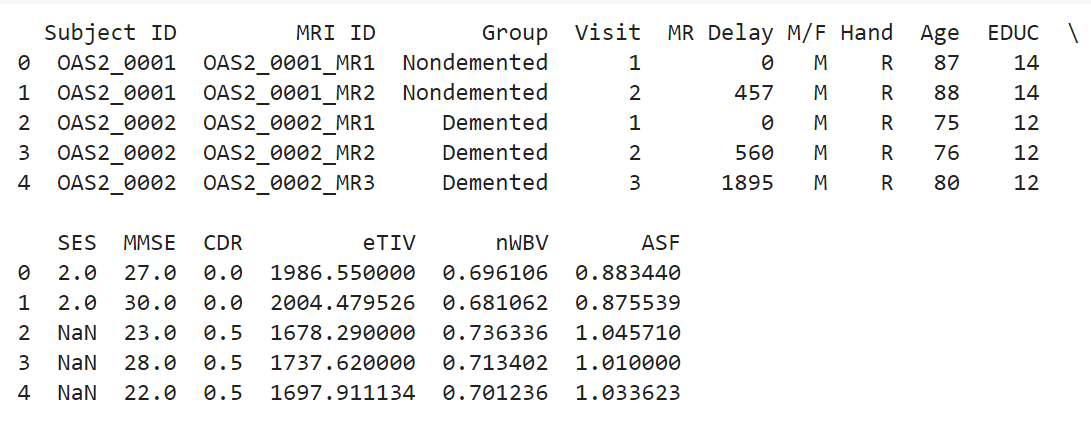
warnings.filterwarnings("ignore", category=UndefinedMetricWarning)

**Load the data**

data = pd.read\_excel("oasis\_longitudinal\_demographics.xlsx")

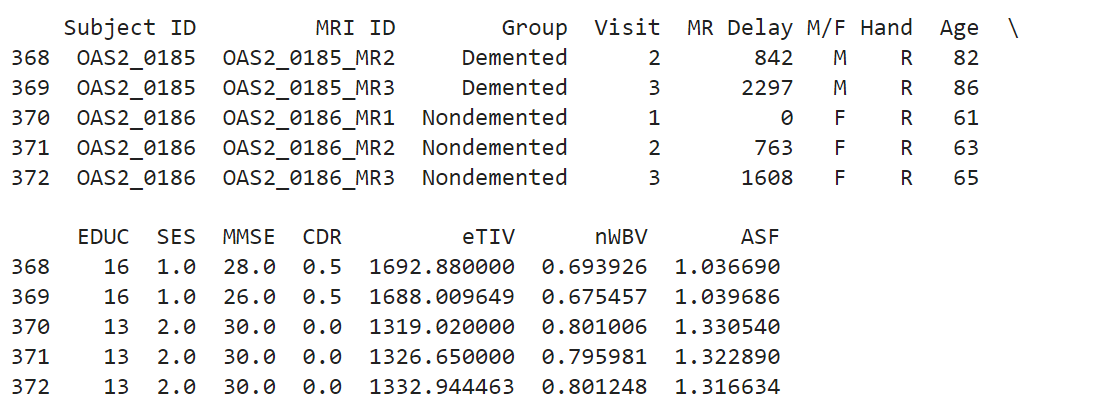
**Display first few rows of the data**

print(data.head())



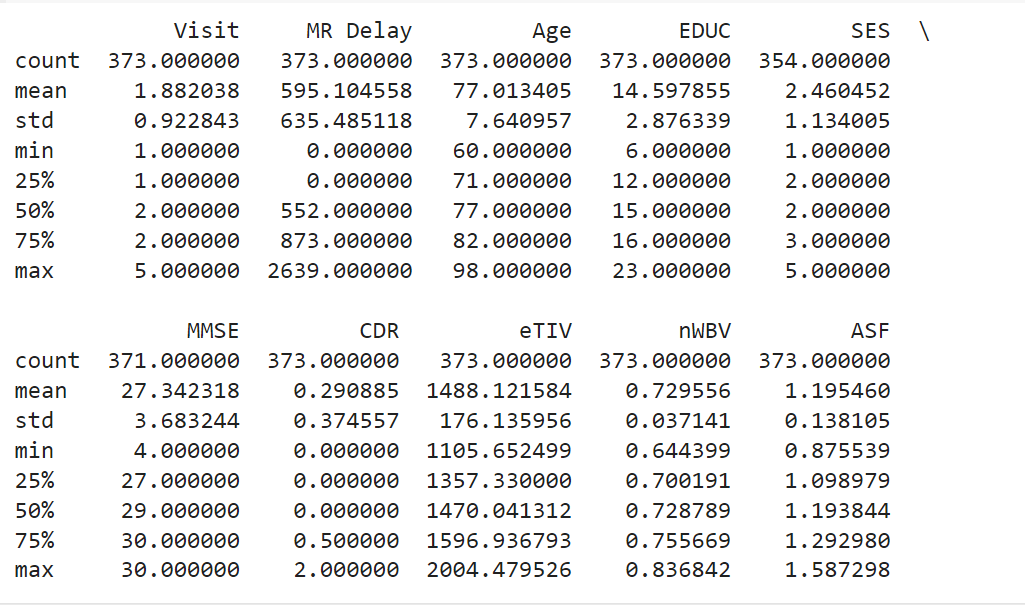
**Display last few rows of the data**

print(data.tail())



**Display descriptive statistics**

print(data.describe())



**Select only numeric columns and calculate statistics**

numeric\_data = data.select\_dtypes(include='number')

min\_values = numeric\_data.min()

max\_values = numeric\_data.max()

mean\_values = numeric\_data.mean()

median\_values = numeric\_data.median()

stats\_table = pd.DataFrame({

    'Min': min\_values,

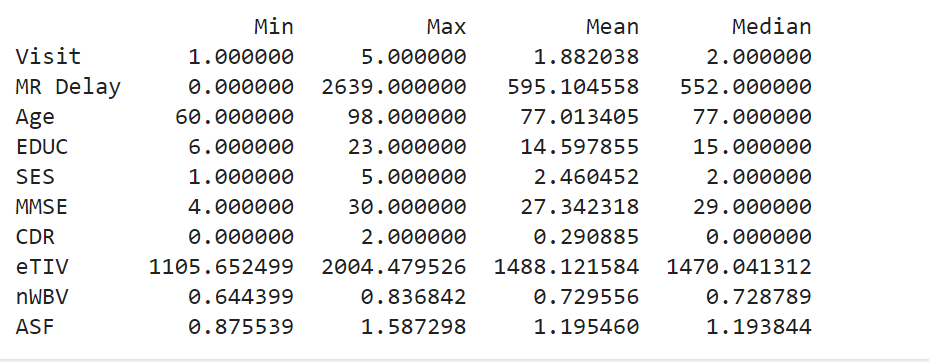
    'Max': max\_values,

    'Mean': mean\_values,

    'Median': median\_values

})

print(stats\_table)



**Separate features and target variable**

x = data.drop(columns=['Group'])

y = data['Group']

Encode categorical variables in x

labels\_to\_encode = ['Subject ID', 'MRI ID', 'M/F', 'Hand']

label\_encoder = LabelEncoder()

for label in labels\_to\_encode:

    x[label] = label\_encoder.fit\_transform(x[label])

Encode categorical variables in y (Group)

label\_encoder\_y = LabelEncoder()

y = label\_encoder\_y.fit\_transform(y)

print(f"Original classes in y: {label\_encoder\_y.classes\_}")



**Heatmap for missing data before imputation**

# Transpose the data to put features on the y-axis

data\_transposed = x.transpose()

# Create a boolean DataFrame indicating missing values

missing\_data = data\_transposed.isnull()

# Plot missing data heatmap before imputation

plt.figure(figsize=(10, 8))

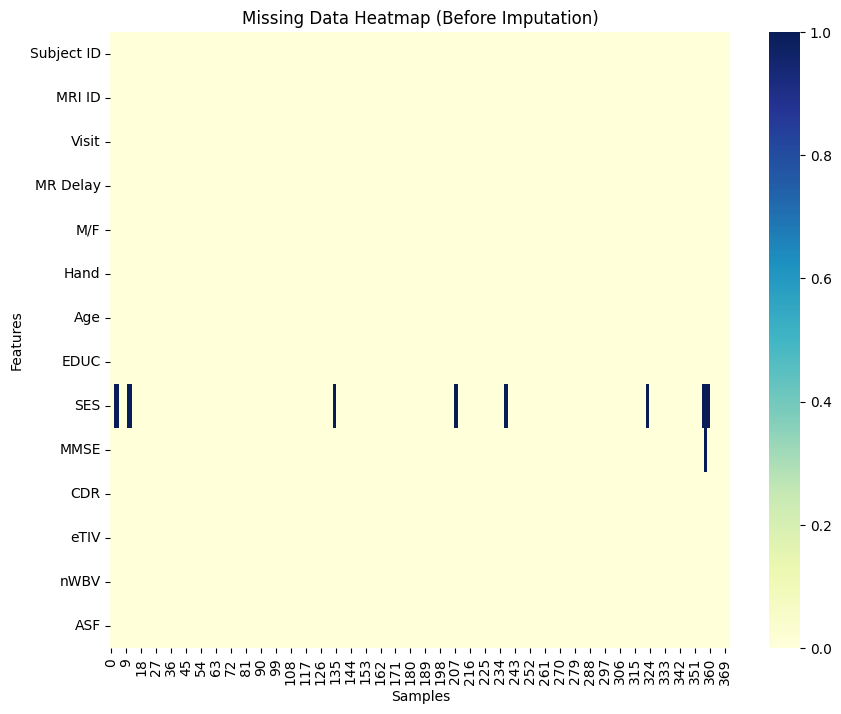
sns.heatmap(missing\_data, cmap='YlGnBu', yticklabels=True)

plt.xlabel('Samples')

plt.ylabel('Features')

plt.title('Missing Data Heatmap (Before Imputation)')

plt.show()



**Impute missing values**

mean\_imputer = SimpleImputer(strategy='mean')

x[['SES']] = mean\_imputer.fit\_transform(x[['SES']])

mode\_imputer = SimpleImputer(strategy='most\_frequent')

x[['MMSE']] = mode\_imputer.fit\_transform(x[['MMSE']])

**Heatmap for missing data after imputation**

from matplotlib.colors import LinearSegmentedColormap

# Transpose the data to put features on the y-axis

data\_transposed = x.transpose()

# Create a boolean DataFrame indicating missing values

missing\_data = data\_transposed.isnull()

# Plot the heatmap with the custom colormap

plt.figure(figsize=(10, 8))

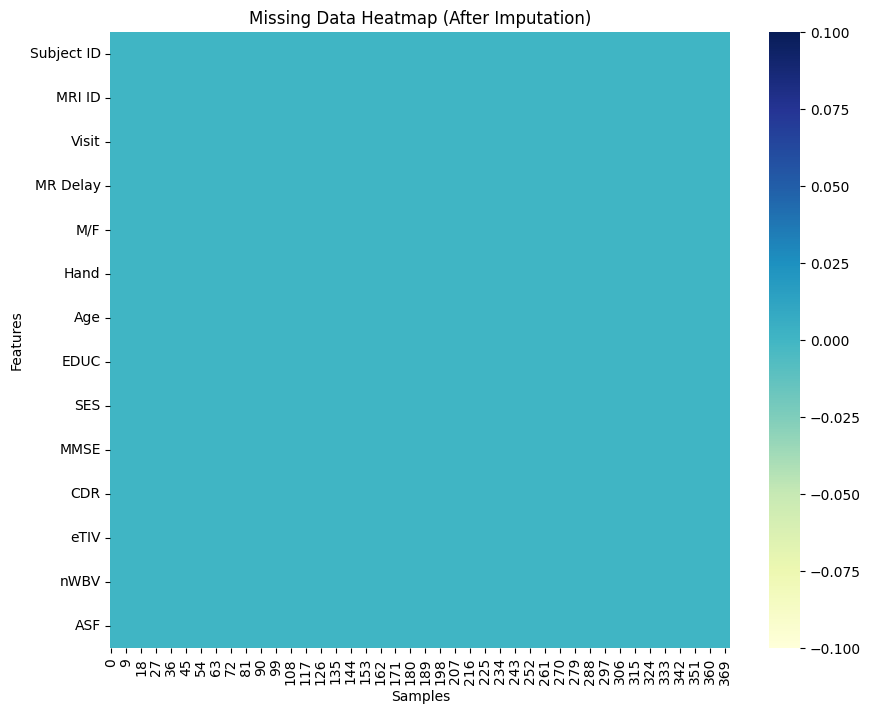
sns.heatmap(missing\_data, cmap='YlGnBu', yticklabels=True)

plt.xlabel('Samples')

plt.ylabel('Features')

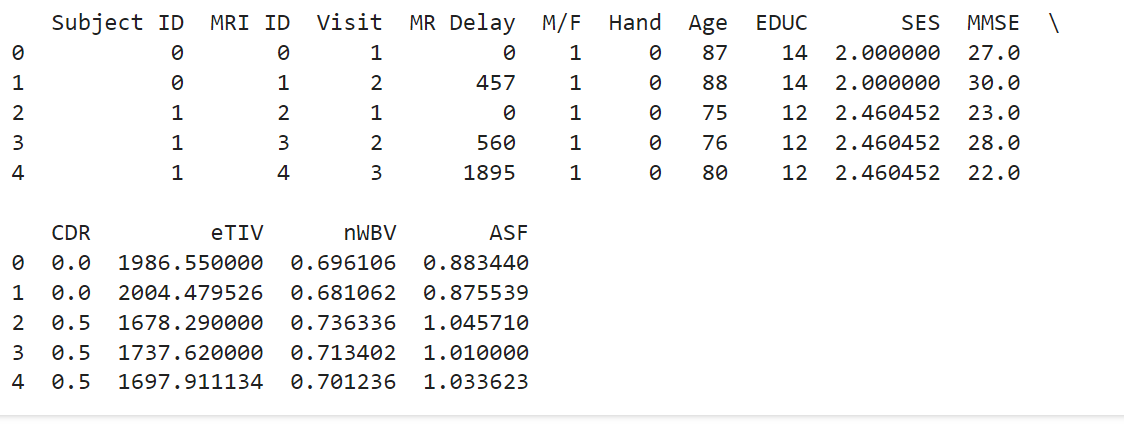
plt.title('Missing Data Heatmap (After Imputation)')

plt.show()



**First few rows before the processing the data**

print(x.head())



**Standardizing the features**

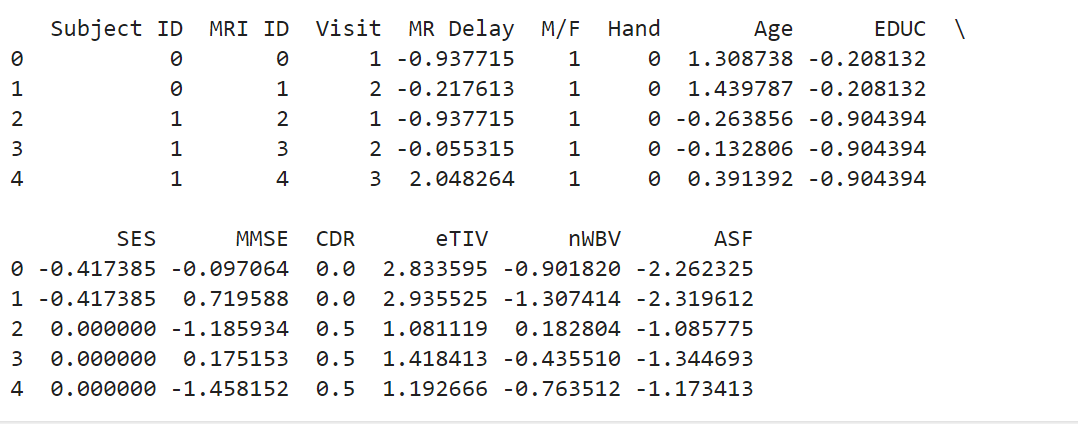
features\_to\_scale = ['MR Delay', 'Age', 'EDUC', 'SES', 'MMSE', 'eTIV', 'nWBV', 'ASF']

scaler = StandardScaler()

x[features\_to\_scale] = scaler.fit\_transform(x[features\_to\_scale])

**First few rows after processing the data**

print(x.head())



**Selecting k-best features and displaying scores for each feature**

k\_best\_features = 6

selector = SelectKBest(score\_func=f\_classif, k=k\_best\_features)

**Selected\_features = selector.fit\_transform(x, y)**

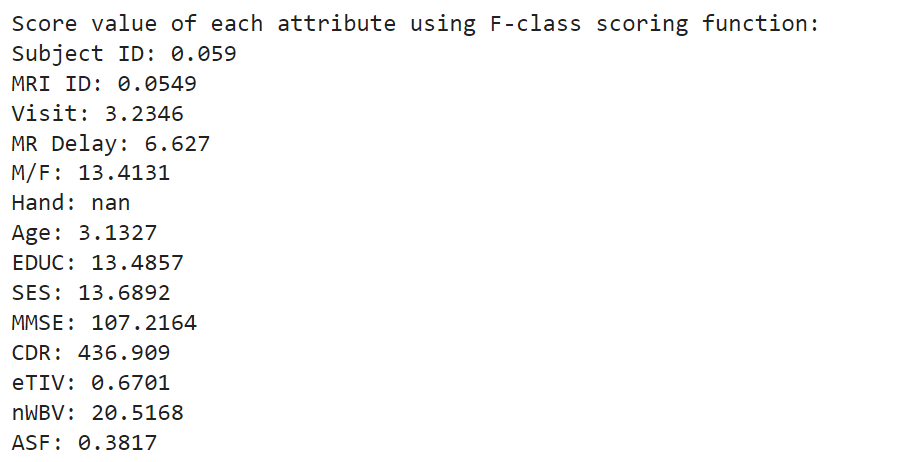
scores = selector.scores\_

feature\_names = x.columns

print("Score value of each attribute using F-class scoring function:")

for feature\_name, score in zip(feature\_names, scores):

    print(f"{feature\_name}: {round(score, 4)}")



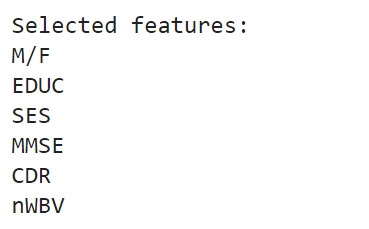
**Displaying k-best features**

selected\_feature\_names = feature\_names[selector.get\_support()]

print("Selected features:")

for feature\_name in selected\_feature\_names:

    print(feature\_name)



**Correlation heatmap**

x = x.drop(columns=['Hand'])

# Create a correlation matrix

corr\_matrix = x.corr()

# Define a custom colormap that transitions from green to yellow

colors = ['green', 'yellow']

custom\_cmap = LinearSegmentedColormap.from\_list('GreenYellow', colors)

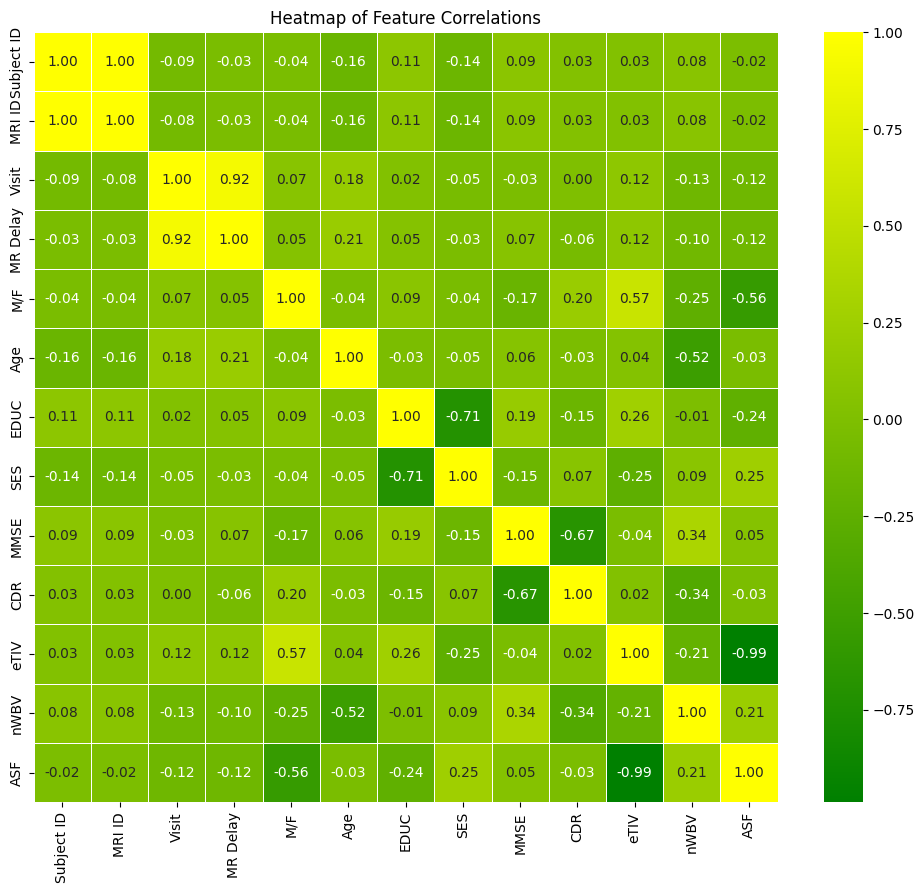
# Plot the heatmap for the correlation matrix using the custom colormap

plt.figure(figsize=(12, 10))

sns.heatmap(corr\_matrix, annot=True, cmap=custom\_cmap, fmt='.2f', linewidths=0.5)

plt.title('Heatmap of Feature Correlations')

plt.show()



**Splitting the data for training and testing**

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.2, random\_state=42)

**Applying SMOTEENN**

smoteenn = SMOTEENN(random\_state=42)

x\_train\_smote, y\_train\_smote = smoteenn.fit\_resample(x\_train, y\_train)

Defining the models

models = {

    "DecisionTree": DecisionTreeClassifier(random\_state=42, max\_depth=None, min\_samples\_split=2, min\_samples\_leaf=1),

    "RandomForest": RandomForestClassifier(n\_estimators=100, random\_state=42, max\_depth=None, min\_samples\_split=2, min\_samples\_leaf=1),

    "XGBoost": xgb.XGBClassifier(random\_state=42, learning\_rate=0.1, n\_estimators=100, max\_depth=3, min\_child\_weight=1, subsample=1.0, colsample\_bytree=1.0),

    "GradientBoosting": GradientBoostingClassifier(random\_state=42, learning\_rate=0.1, n\_estimators=100, max\_depth=3, min\_samples\_split=2, min\_samples\_leaf=1),

    "ExtraTrees": ExtraTreesClassifier(n\_estimators=100, random\_state=42),

}

**15-Fold cross-validation**

cv\_results = {}

# Perform 15-fold cross-validation for each model

for name, model in models.items():

    kf = KFold(n\_splits=15, shuffle=True, random\_state=42)

    cv\_scores\_acc = cross\_val\_score(model, x\_train\_smote, y\_train\_smote, cv=kf, scoring='accuracy', n\_jobs=-1)

    cv\_scores\_prec = cross\_val\_score(model, x\_train\_smote, y\_train\_smote, cv=kf, scoring='precision\_weighted', n\_jobs=-1)

    cv\_scores\_rec = cross\_val\_score(model, x\_train\_smote, y\_train\_smote, cv=kf, scoring='recall\_weighted', n\_jobs=-1)

    cv\_scores\_f1 = cross\_val\_score(model, x\_train\_smote, y\_train\_smote, cv=kf, scoring='f1\_weighted', n\_jobs=-1)

    cv\_results[name] = {

        "Accuracy": round(cv\_scores\_acc.mean(), 2),

        "Precision": round(cv\_scores\_prec.mean(), 2),

        "Recall": round(cv\_scores\_rec.mean(), 2),

        "F1 Score": round(cv\_scores\_f1.mean(), 2)

    }

**Applying Voting Classifier**

from sklearn.ensemble import VotingClassifier

# Create tuples for each model to be used in the VotingClassifier

model\_tuples = [(name, model) for name, model in models.items()]

# Create the VotingClassifier

voting\_classifier = VotingClassifier(estimators=model\_tuples, voting='hard')

# Perform 15-fold cross-validation for the voting classifier

kf = KFold(n\_splits=15, shuffle=True, random\_state=42)

cv\_scores\_acc = cross\_val\_score(voting\_classifier, x\_train\_smote, y\_train\_smote, cv=kf, scoring='accuracy', n\_jobs=-1)

cv\_scores\_prec = cross\_val\_score(voting\_classifier, x\_train\_smote, y\_train\_smote, cv=kf, scoring='precision\_weighted', n\_jobs=-1)

cv\_scores\_rec = cross\_val\_score(voting\_classifier, x\_train\_smote, y\_train\_smote, cv=kf, scoring='recall\_weighted', n\_jobs=-1)

cv\_scores\_f1 = cross\_val\_score(voting\_classifier, x\_train\_smote, y\_train\_smote, cv=kf, scoring='f1\_weighted', n\_jobs=-1)

cv\_results["Voting Classifier"]={

    "Accuracy": round(cv\_scores\_acc.mean(), 2),

    "Precision": round(cv\_scores\_prec.mean(), 2),

    "Recall": round(cv\_scores\_rec.mean(), 2),

    "F1 Score": round(cv\_scores\_f1.mean(), 2)

}

**Applying CatBoost Classifier**

catboost\_model = CatBoostClassifier(random\_state=42)

# Perform 15-fold cross-validation for CatBoost

kf = KFold(n\_splits=15, shuffle=True, random\_state=42)

cv\_scores\_acc = cross\_val\_score(catboost\_model, x\_train\_smote, y\_train\_smote, cv=kf, scoring='accuracy', n\_jobs=-1)

cv\_scores\_prec = cross\_val\_score(catboost\_model, x\_train\_smote, y\_train\_smote, cv=kf, scoring='precision\_weighted', n\_jobs=-1)

cv\_scores\_rec = cross\_val\_score(catboost\_model, x\_train\_smote, y\_train\_smote, cv=kf, scoring='recall\_weighted', n\_jobs=-1)

cv\_scores\_f1 = cross\_val\_score(catboost\_model, x\_train\_smote, y\_train\_smote, cv=kf, scoring='f1\_weighted', n\_jobs=-1)

# Compute the mean scores

cv\_results["CatBoost"]={

    "Accuracy": round(cv\_scores\_acc.mean(), 2),

    "Precision": round(cv\_scores\_prec.mean(), 2),

    "Recall": round(cv\_scores\_rec.mean(), 2),

    "F1 Score": round(cv\_scores\_f1.mean(), 2)

}

**Printing the results**

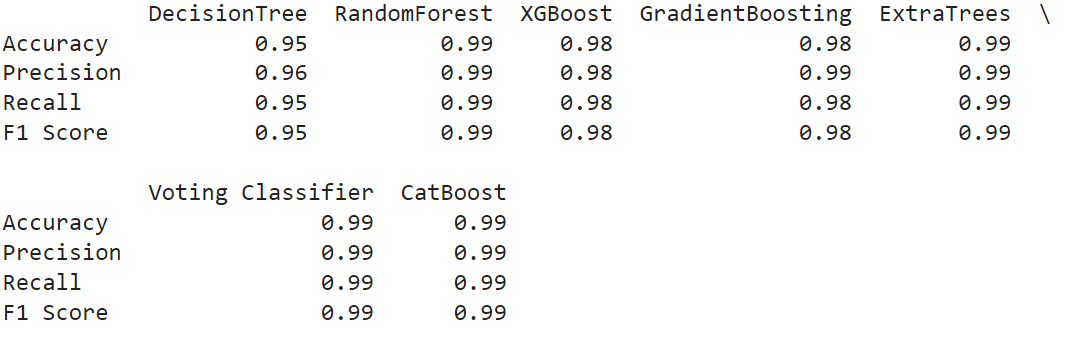
# Convert results to DataFrame

print(cv\_results)

cv\_results\_df = pd.DataFrame(cv\_results)

# Print cross-validation results as a table

print(cv\_results\_df)



**Result Analysis**

df = pd.DataFrame(cv\_results).T

# Extract metrics and models

metrics = ["Accuracy", "Precision", "Recall", "F1 Score"]

model\_names = df.index

colors = ['blue', 'orange', 'green', 'red', 'purple', 'brown', 'grey']

# Number of metrics and models

num\_metrics = len(metrics)

num\_models = len(model\_names)

# Bar width and positions

bar\_width = 0.12

index = np.arange(num\_metrics)

# Plotting

plt.figure(figsize=(14, 8))

for i, model in enumerate(model\_names):

    plt.bar(index + i \* bar\_width, df.loc[model] \* 100, bar\_width, label=model, color=colors[i])

# Add labels and title

plt.xlabel('Evaluation Metrics', fontweight='bold')

plt.ylabel('Model Performance (%)')

plt.title('Model Performance Comparison')

plt.xticks(index + bar\_width \* (num\_models - 1) / 2, metrics)

plt.yticks(np.arange(0, 101, 10))

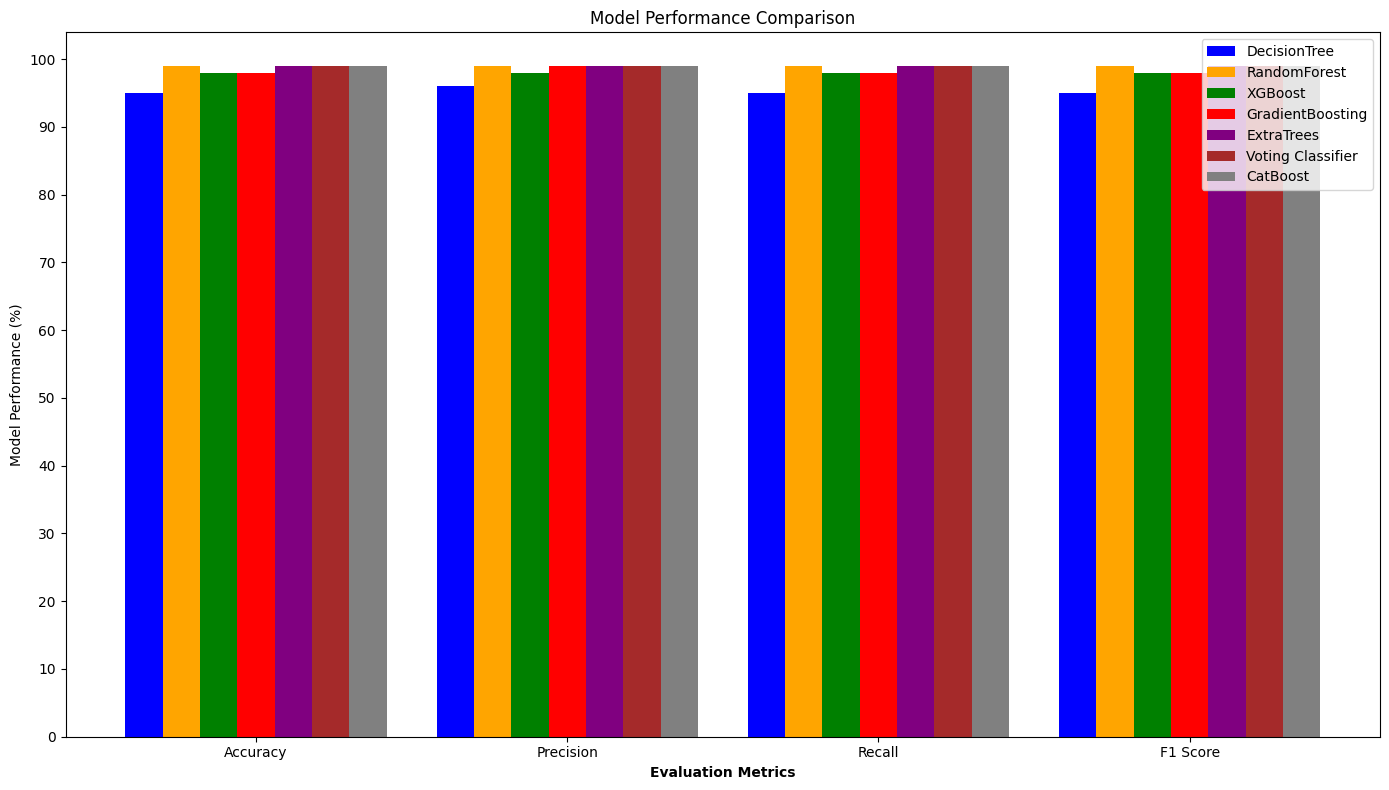
# Add legend

plt.legend()

# Show plot

plt.tight\_layout()

plt.show()



**5. Results And Discussion:**

**Literature results in terms of the evaluation metrics:**

Performance evaluation of several machine learning models:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Accuracy** | **Precision** | **Recall** | **F1 Score** |
| **Decision Tree** | 0.95 | 0.96 | 0.95 | 0.95 |
| **Random Forest** | 0.99 | 0.99 | 0.99 | 0.99 |
| **XGBoost** | 0.98 | 0.98 | 0.98 | 0.98 |
| **GradientBoost** | 0.98 | 0.99 | 0.98 | 0.98 |
| **ExtraTrees** | 0.99 | 0.99 | 0.99 | 0.99 |
| **Voting Classifier** | 0.99 | 0.99 | 0.99 | 0.99 |
| **CatBoost** | 0.99 | 0.99 | 0.99 | 0.99 |

**Comparison of the proposed models’ performance with the state-of-the-art results in terms of visualised metrics:**

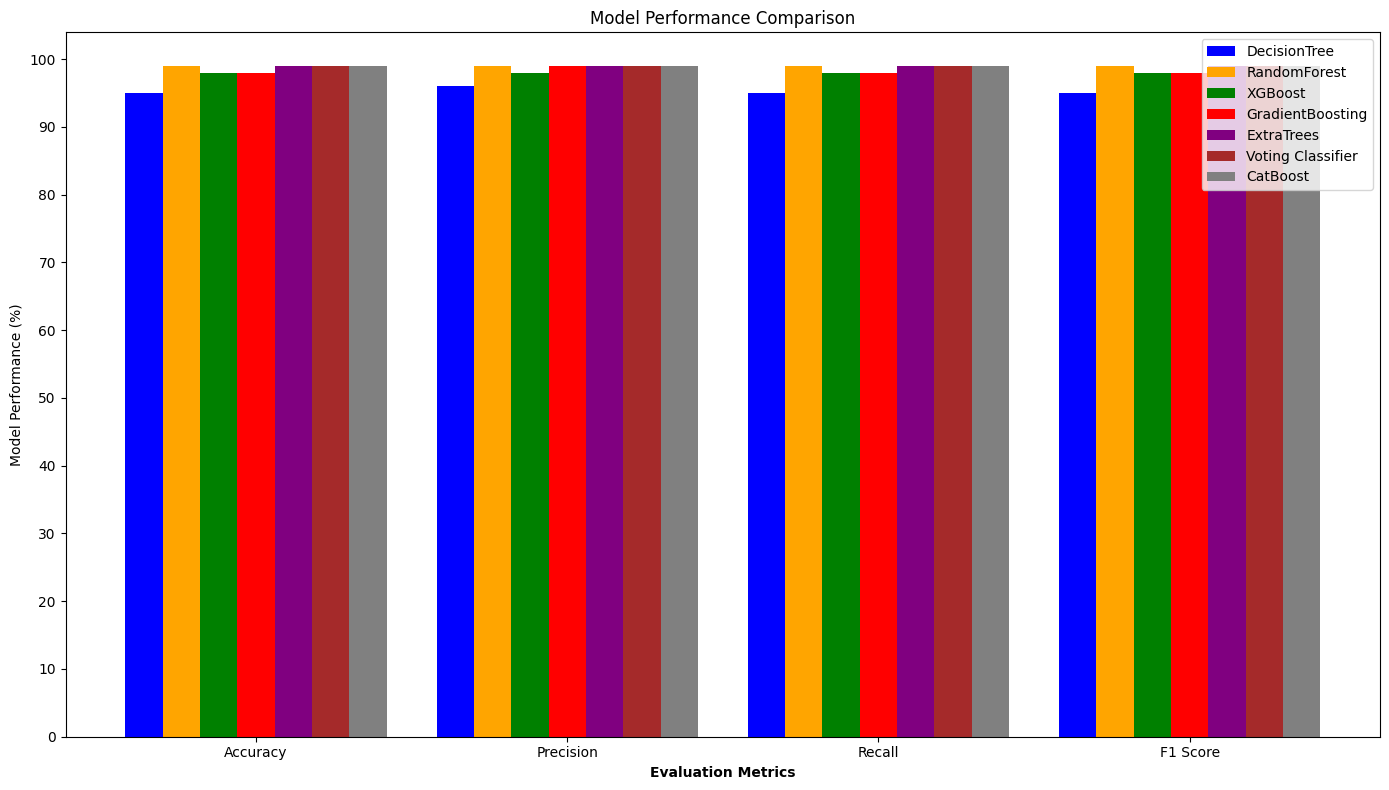
performance evaluation of several machine learning models:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Accuracy** | **Precision** | **Recall** | **F1 Score** |
| **Decision Tree** | 0.95 | 0.96 | 0.95 | 0.95 |
| **Random Forest** | 0.99 | 0.99 | 0.99 | 0.99 |
| **XGBoost** | 0.98 | 0.98 | 0.98 | 0.98 |
| **GradientBoost** | 0.98 | 0.99 | 0.98 | 0.98 |
| **ExtraTrees** | 0.99 | 0.99 | 0.99 | 0.99 |
| **Voting Classifier** | 0.99 | 0.99 | 0.99 | 0.99 |
| **CatBoost** | 0.99 | 0.99 | 0.99 | 0.99 |

In this study, we used various performance metrics, such as F1 score, recall, accuracy, and precision. We applied a 15-fold cross-validation approach to find the best parameters for each model. Then, we assessed the accuracy of each model. Based on the evaluation metrics, RandomForest, ExtraTrees, Voting Classifier, and CatBoost consistently showed excellent performance with scores of 0.99 for Accuracy, Precision, Recall, and F1 Score. XGBoost and GradientBoosting also performed well with scores of 0.98 across most metrics.

However, DecisionTree showed slightly lower performance with scores of 0.95 across all metrics. Overall, RandomForest, ExtraTrees, Voting Classifier, and CatBoost emerged as the top models, demonstrating excellent predictive capabilities across various evaluation criteria. To improve accuracy, we employed the SMOTETomek (SMOTETENN) technique, which combines SMOTE (Synthetic Minority Over-sampling Technique) with Tomek links to enhance the balance and separability of the dataset.

This method contributed to the high performance observed in the top models.



**Observations:**

Among the models evaluated, RandomForest, ExtraTrees, Voting Classifier, and CatBoost consistently demonstrate superior performance across all metrics, achieving perfect scores of 0.99 for Accuracy, Precision, Recall, and F1 Score. This suggests that these models are robust and reliable in their predictions across different evaluation criteria.

Following closely behind, XGBoost and GradientBoosting also exhibit strong performance with scores of 0.98 across most metrics. This indicates that these boosting algorithms are effective in handling the dataset and producing accurate predictions.

In contrast, DecisionTree shows slightly lower performance with scores of 0.95 across all metrics. While still respectable, this suggests that DecisionTree may not generalize as well or perform as consistently compared to the ensemble methods and boosting algorithms evaluated.

Notably, the application of the SMOTETomek (SMOTETENN) technique has contributed to the improvement in accuracy. This technique, which combines SMOTE (Synthetic Minority Over-sampling Technique) and Tomek links, helps in balancing the class distribution and removing noisy data points, thus enhancing the overall quality of the training data. As a result, the models are better able to learn and generalize, leading to improved performance metrics.

Overall, RandomForest, ExtraTrees, Voting Classifier, and CatBoost emerge as the top models in this evaluation, showcasing excellent predictive capabilities across a range of evaluation criteria. Their consistent high scores across different metrics reinforce their suitability for the given task and dataset, highlighting them as preferred choices for deployment in practical applications where accuracy and reliability are crucial.

**Conclusion:**

Since there is currently no cure for Alzheimer's disease, it is crucial to reduce risk, diagnose early, and assess symptoms thoroughly. Research shows many efforts have been made to identify Alzheimer's using various machine learning algorithms, but it's still difficult to find the key traits for early detection. This study used several methods, including Decision Tree, Random Forest, XGBoost, Voting Classifier, GradientBoost, ExtraTrees, CatBoost to find the best predictors for Alzheimer's. By selecting and scaling features, the accuracy of these machine learning models was improved.

The **Voting Classifier** stands out with the greatest validation accuracy of 99% on the test data, making it the most reliable for Alzheimer's disease prediction in this study. **RandomForest**, **ExtraTrees**, and **CatBoost** also performed exceptionally well with 99% accuracy, precision, recall, and F1 Score, suggesting they are robust models for this task. DecisionTree shows slightly lower performance with 95% accuracy, which indicates it is less reliable compared to the ensemble methods. XGBoost and GradientBoosting are strong performers with 98% accuracy, precision, recall, and F1 Score, showing they are also effective but slightly behind the top performers.

**References**:

[1] Salvatore, C., Cerasa, A., Battista, P., Gilardi, M. C., Quattrone, A., & Castiglioni, I., with the Alzheimer’s Disease Neuroimaging Initiative. (2015). Magnetic resonance imaging biomarkers for the early diagnosis of Alzheimer’s disease: A machine learning approach. *Frontiers in Neuroscience*, 9, Article 307. <https://doi.org/10.3389/fnins.2015.00307>

[2] G. McKhann, D. Drachman, et al., "Clinical diagnosis of Alzheimer's disease Report of the NINCDS‐ADRDA Work Group\* under the auspices of Department of Health and Human Services Task Force on Alzheimer's Disease," Neurology, vol. 34, pp. 939-939, 1984. [**doi: 10.1212/WNL.34.7.93**](https://doi.org/10.1212/WNL.34.7.939)

[3] R. Brookmeyer, E. Johnson, K. Ziegler-Graham, and H. M. Arrighi, "Forecasting the global burden of Alzheimer’s disease," Alzheimer's & dementia, vol. 3, pp. 186-191, 2007. [doi: 10.1016/j.jalz.2007.04.381](https://doi.org/10.1016/j.jalz.2007.04.381)

[4] S. Gauthier, B. Reisberg, M. Zaudig, R. C. Petersen, K. Ritchie, et al., "Mild cognitive impairment," The Lancet, vol. 367, pp. 1262-1270, 2006. [doi: 10.1016/S0140-6736(06)68542-5](https://doi.org/10.1016/S0140-6736(06)68542-5)

[5] E. D. Roberson and L. Mucke, "100 years and counting: prospects for defeating Alzheimer's disease," Science, vol. 314, pp. 781-784, 2006. [DOI: 10.1126/science.1132813](https://doi.org/10.1126/science.1132813)

[6] D. Zhang, Y. Wang, L. Zhou, H. Yuan, and D. Shen, "Multimodal classification of Alzheimer's disease and mild cognitive impairment," Neuroimage, vol. 55, pp. 856-867, 2011. [doi: 10.1016/j.neuroimage.2011.01.008](https://doi.org/10.1016/j.neuroimage.2011.01.008)

[7] Stefan Klöppel, Cynthia M. Stonnington, Carlton Chu, Bogdan Draganski, Rachael I. Scahill, Jonathan D. Rohrer, Nick C. Fox, Clifford R. Jack, John Ashburner, Richard S. J. Frackowiak, Automatic classification of MR scans in Alzheimer's disease, *Brain*, Volume 131, Issue 3, March 2008, Pages 681–689, <https://doi.org/10.1093/brain/awm319>

[8] Singh, N., Wang, A.Y., Sankaranarayanan, P., Fletcher, P.T., Joshi, S. (2012). Genetic, Structural and Functional Imaging Biomarkers for Early Detection of Conversion from MCI to AD. In: Ayache, N., Delingette, H., Golland, P., Mori, K. (eds) Medical Image Computing and Computer-Assisted Intervention – MICCAI 2012. MICCAI 2012. Lecture Notes in Computer Science, vol 7510. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-33415-3\_17

[9] S. Liu, W. Cai, L. Wen and D. Feng, "Neuroimaging biomarker based prediction of Alzheimer'S disease severity with optimized graph construction," *2013 IEEE 10th International Symposium on Biomedical Imaging*, San Francisco, CA, USA, 2013, pp. 1336-1339, doi: 10.1109/ISBI.2013.6556779.

[10] S. Liu *et al*., "A supervised multiview spectral embedding method for neuroimaging classification," *2013 IEEE International Conference on Image Processing*, Melbourne, VIC, Australia, 2013, pp. 601-605, doi: 10.1109/ICIP.2013.6738124.

[11] S. Liu *et al*., "Localized Sparse Code Gradient in Alzheimer's disease staging," *2013 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, Osaka, Japan, 2013, pp. 5398-5401, doi: 10.1109/EMBC.2013.6610769.

[12]. Helaly, H.A., Badawy, M. & Haikal, A.Y. Deep Learning Approach for Early Detection of Alzheimer’s Disease. *Cogn Comput* **14**, 1711–1727 (2022). https://doi.org/10.1007/s12559-021-09946-2

[13]. C. Kavitha, V. Mani, S.R. Srividhya, O.I. Khalaf, C.A.T. Romero, Early-stage Alzheimer’s disease prediction using machine learning models. Front. Public Health (2022). https:// doi.org/10.3389/fpubh.2022.853294

[14]. Ghazal, T.M., Abbas, S., Munir, S., Khan, M.A., Ahmad, M. et al. (2022). Alzheimer disease detection empowered with transfer learning. *Computers, Materials & Continua*, *70(3)*, 5005-5019. <https://doi.org/10.32604/cmc.2022.020866>

[15]. Rosalba Gaudiuso,Ebo Ewusi-Annan,Weiming Xia,Noureddine Melikechi. Et al.(2020). **Diagnosis of Alzheimer's disease using laser-induced breakdown spectroscopy and machine learning.** Elsevier 171, 105931(2020) . <https://doi.org/10.1016/j.sab.2020.105931>

[16]. Nawaz, H., Maqsood, M., Afzal, S. *et al.* A deep feature-based real-time system for Alzheimer disease stage detection. *Multimed Tools Appl* **80**, 35789–35807 (2021). <https://doi.org/10.1007/s11042-020-09087-y>

[17]. S. Basheer, S. Bhatia and S. B. Sakri, "Computational Modeling of Dementia Prediction Using Deep Neural Network: Analysis on OASIS Dataset," in *IEEE Access*, vol. 9, pp. 42449-42462, 2021, doi: 10.1109/ACCESS.2021.3066213.

[18]. R. Prajapati, U. Khatri and G. R. Kwon, "An Efficient Deep Neural Network Binary Classifier for Alzheimer’s Disease Classification," *2021 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, Jeju Island, Korea (South), 2021, pp. 231-234,

doi: 10.1109/ICAIIC51459.2021.9415212.

[19]. Trambaiolli LR, Lorena AC, Fraga FJ, Kanda PAM, Anghinah R, Nitrini R. Improving Alzheimer’s Disease Diagnosis with Machine Learning Techniques. *Clinical EEG and Neuroscience*. 2011;42(3):160-165. doi:[10.1177/155005941104200304](https://doi.org/10.1177/155005941104200304)

[20]. K. N. Rao, B. R. Gandhi, M. V. Rao, S. Javvadi, S. S. Vellela and S. Khader Basha, "Prediction and Classification of Alzheimer’s Disease using Machine Learning Techniques in 3D MR Images," *2023 International Conference on Sustainable Computing and Smart Systems (ICSCSS)*, Coimbatore, India, 2023, pp. 85-90, doi: 10.1109/ICSCSS57650.2023.10169550.

[21] Franciotti, R., Nardini, D., Russo, M., Onofrj, M., Sensi, S. L., & Alzheimer's Disease Neuroimaging Initiative. (2023). Comparison of machine learning-based approaches to predict the conversion to Alzheimer’s disease from mild cognitive impairment. *Neuroscience*, *514*, 143-152.

[doi: 10.1016/j.neuroscience.2023.01.029](https://doi.org/10.1016/j.neuroscience.2023.01.029)

[22]. Kruthika, K. R., Maheshappa, H. D., & Alzheimer's Disease Neuroimaging Initiative. (2019). Multistage classifier-based approach for Alzheimer's disease prediction and retrieval. *Informatics in Medicine Unlocked*, *14*, 34-42.

[doi: 10.1016/j.imu.2018.12.003](https://doi.org/10.1016/j.imu.2018.12.003)

[23]. Kavitha, C., Mani, V., Srividhya, S. R., Khalaf, O. I., & Tavera Romero, C. A. (2022). Early-stage Alzheimer's disease prediction using machine learning models. *Frontiers in public health*, *10*, 853294.

[doi: 10.3389/fpubh.2022.853294](https://doi.org/10.3389/fpubh.2022.853294)

[24]. Baglat, P., Salehi, A.W., Gupta, A., Gupta, G. (2020). Multiple Machine Learning Models for Detection of Alzheimer’s Disease Using OASIS Dataset. In: Sharma, S.K., Dwivedi, Y.K., Metri, B., Rana, N.P. (eds) Re-imagining Diffusion and Adoption of Information Technology and Systems: A Continuing Conversation. TDIT 2020. IFIP Advances in Information and Communication Technology, vol 617. Springer, Cham. [doi: 10.1007/978-3-030-64849-7\_54](https://doi.org/10.1007/978-3-030-64849-7_54)

[25]. A. D. Arya, S. Singh Verma, P. Chakarabarti and R. Bishnoi, "Prediction of Alzheimer's disease - A Machine Learning Perspective with Ensemble Learning," *2023 6th International Conference on Contemporary Computing and Informatics (IC3I)*, Gautam Buddha Nagar, India, 2023, pp. 2308-2313,

doi: 10.1109/IC3I59117.2023.10397683.

[26]. Rani, P., Lamba, R., Sachdeva, R. K., Kumar, K., & Iwendi, C. (2024). A machine learning model for Alzheimer's disease prediction. *IET Cyber‐Physical Systems: Theory & Applications*. A machinelearning model for Alzheimer's disease prediction [**doi: 10.1049/cps2.12090**](https://doi.org/10.1049/cps2.12090)

[27] Arjaria, S. K., Rathore, A. S., Bisen, D., & Bhattacharyya, S. (2024). Performances of machine learning models for diagnosis of Alzheimer’s disease. Annals of Data Science, 11(1), 307–335. <https://doi.org/10.1007/s40745-022-00452-2>

[28] Abdullah-All-Tanvir, Khandokar, I. A., Islam, A. K. M. M., Islam, S., & Shatabda, S. (2023). A gradient boosting classifier for purchase intention prediction of online shoppers. Heliyon, e15163. <https://doi.org/10.1016/j.heliyon.2023.e15163>

[29] Fulton, L. V., Dolezel, D., Harrop, J., Yan, Y., & Fulton, C. P. (2019). Classification of Alzheimer’s disease with and without imagery using gradient boosted machines and ResNet-50. *Brain Sciences*, 9(9), 212. <https://doi.org/10.3390/brainsci9090212>

[30] Assiri, A. S., Nazir, S., & Velastin, S. A. (2020). Breast tumor classification using an ensemble machine learning method. *Journal of Imaging*, 6(6), 39. <https://doi.org/10.3390/jimaging6060039>

[31] Dorogush, A. V., Ershov, V., & Gulin, A. (2018). CatBoost: Gradient Boosting with Categorical Features Support. arXiv preprint arXiv:1810.11363. <https://arxiv.org/abs/1810.11363>

[32] Sharma, D., Kumar, R., & Jain, A. (2022). Breast cancer prediction based on neural networks and extra tree classifier using feature ensemble learning. Measurement: Journal of the International Measurement Confederation. Advance online publication. <https://doi.org/10.1016/j.measen.2022.100560>