MINISTRY OF EDUCATION AND TRAINING

HUNGYEN UNIVERSITY OF TECHNOLOGY AND EDUCATION

FINAL PROJECT

DATA MINING

APPLYING MACHINE LEARNING ON DIAGNOSING AND ANALYZING ALZHEIMER'S DISEASE DATA

MAJOR: DATA SCIENCE

STUDENT: **VU QUANG PHUC**

CLASS: **124221**

MENTOR: **Ph.D NGUYEN VAN QUYET**

HUNG YEN – 2025

COMMENT

Comment from mentor:

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

MENTOR

Nguyen Van Quyet

COMMITMENT

I solemnly declare that the project for the Advanced Python Programming course, titled “Applying machine learning on diagnosing and analyzing Azheimer's Disease Data” is the result of my independent work.

All references and sources used in the project are appropriately cited in the References section. The findings and analyses presented in the project are entirely my own.

I accept full accountability for any discrepancies or violations of this declaration, as per the regulations of the faculty and the university.

Hung Yen, January 12, 2024

Student

Vu Quang Phuc

ACKNOWLEDGEMENT

Completing this project, “Applying machine learning on diagnosing and analyzing Azheimer's Disease Data” has been a challenging yet rewarding journey, and it would not have been possible without the invaluable guidance and support I have received along the way

First and foremost, I would like to express my deepest gratitude to the Department of Computer Science, Faculty of Information Technology – Hung Yen University of Technical Education. The resources, academic environment, and opportunities provided by the department have been instrumental in enabling me to carry out this project. I feel incredibly fortunate to have been part of a program that prioritizes academic excellence and hands-on learning.

I owe a special debt of gratitude to my mentor, Ph.D Ngue, whose dedication, patience, and expertise have been a cornerstone of my progress. His thoughtful feedback and encouragement throughout the process not only enhanced the quality of this project but also deepened my understanding of advanced Python programming and data analysis. His ability to inspire confidence while challenging me to push my boundaries has left a lasting impact on my learning journey.

I would also like to thank all the faculty members of Hung Yen University of Technical Education. Their dedication to teaching and their tireless efforts to impart knowledge have equipped me with a strong foundation in computer science. The skills and insights I gained during their lectures and practical sessions have proven invaluable in overcoming the obstacles encountered in this project.

I acknowledge that, despite my best efforts, there may still be areas for improvement in this work. I warmly welcome constructive criticism and feedback from my professors, as I believe that every critique is an opportunity for growth.

Thank you for your encouragement and support throughout this journey

TABLE OF CONTENTS

TABLE OF FIGURES

CHƯƠNG 1: INTRODUCTION TO THE PROBLEM

1.1 Problem

Alzheimer's disease is a progressive neurodegenerative disorder that severely impacts memory, thinking, and behavior, and it is the most common cause of dementia. Currently, there are approximately 55 million people worldwide living with various forms of dementia, with Alzheimer’s accounting for 60-70% of these cases. The number of individuals affected is projected to triple by 2050. In Vietnam, around 500,000 people are currently diagnosed with Alzheimer's, and this number is expected to rise rapidly due to the aging population.

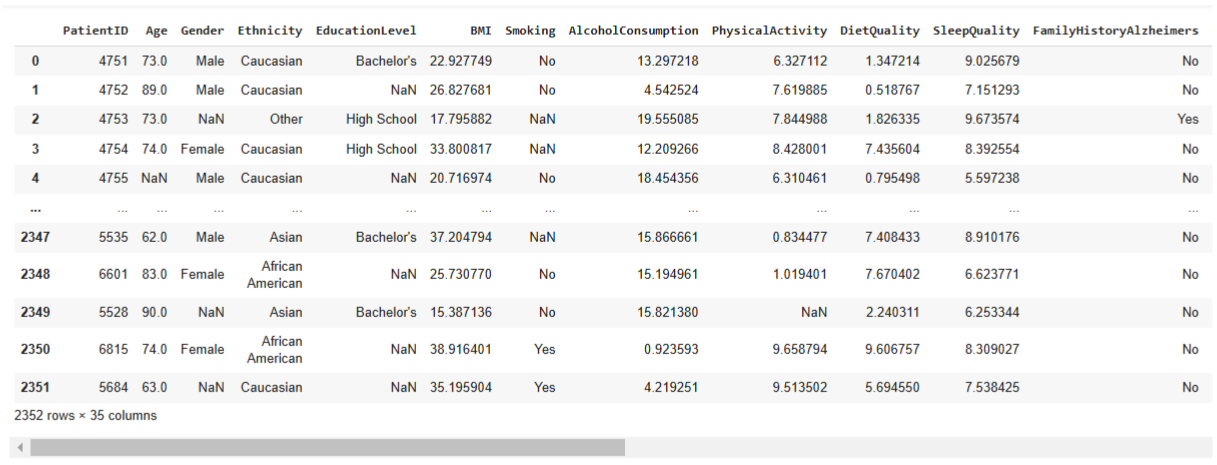
Machine learning (ML) is opening up numerous opportunities for diagnosing Alzheimer’s through medical imaging analysis (MRI, PET), early predictions using biological data, behavioral and linguistic analysis, as well as supporting treatment and patient management. However, the implementation of such technologies in Vietnam faces challenges, including the lack of high-quality data and limited healthcare infrastructure. To fully harness the potential of machine learning in this field, close collaboration among stakeholders is essential to enhance diagnostic and treatment efficacy while improving the quality of life for patients and their families.

Machine learning offers remarkable advancements over traditional Alzheimer’s diagnostic methods. It enables the early detection of subtle signs of brain damage, processes large volumes of complex data (e.g., imaging, biological, genetic), and automates diagnostic workflows, saving time and reducing errors. Additionally, ML facilitates personalized treatment plans tailored to the unique characteristics of each patient and utilizes unconventional data sources such as speech patterns and behavior.

Favorable conditions, such as the abundance of medical data, robust computational technologies, and growing interest from the scientific community, provide a strong foundation for the practical application of machine learning in Alzheimer’s diagnosis and treatment. These advancements hold the promise of significantly enhancing the efficiency and accuracy of diagnosing and managing Alzheimer’s disease, paving the way for improved patient care and outcomes.

1.2 Dataset overview

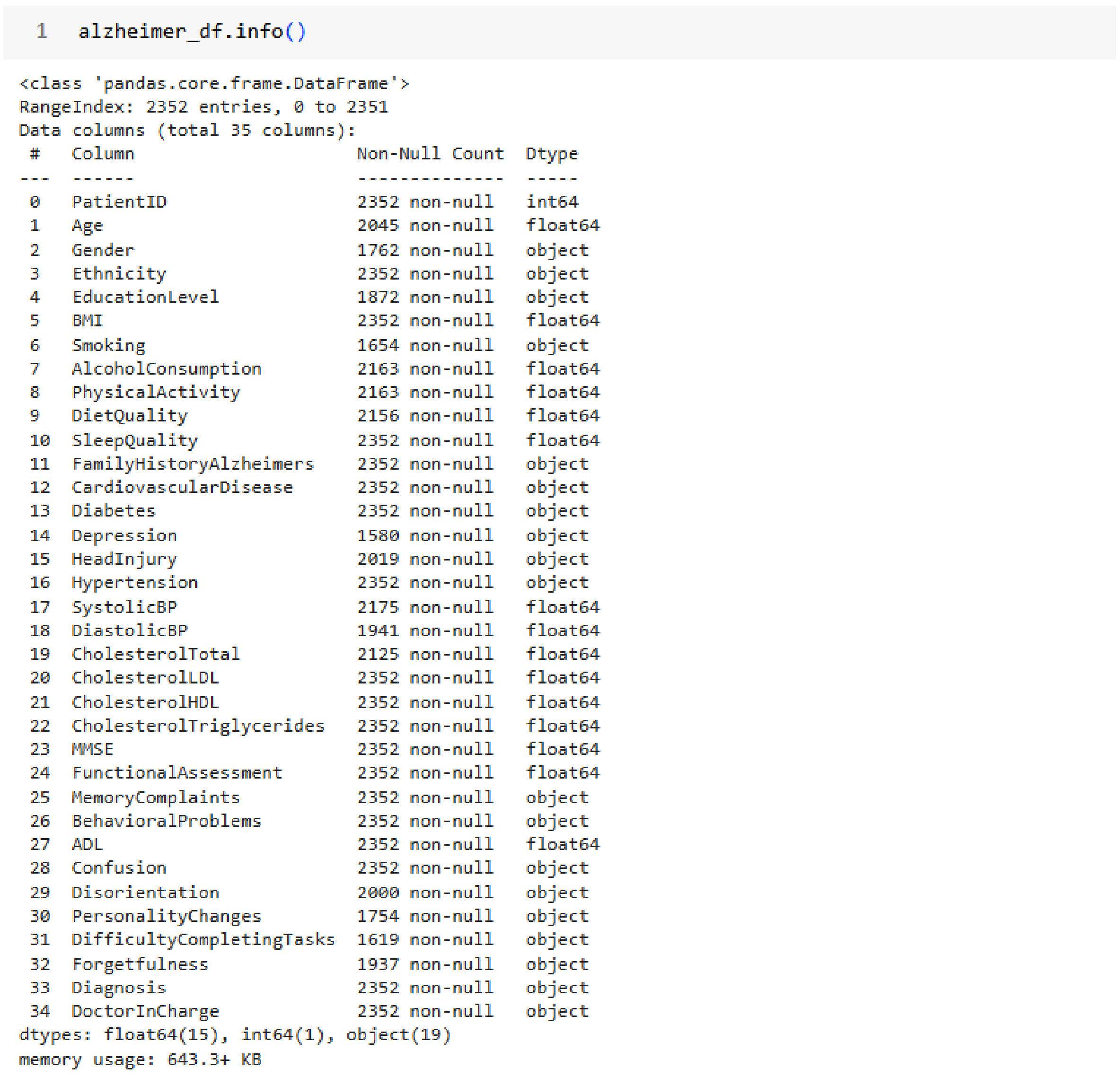
Dataset link in Kaggle: *Alzheimer's Desease Dataset | Kaggle*



	PatientID	Age	Gender	Ethnicity	EducationLevel	BMI	Smoking	AlcoholConsumption	PhysicalActivity	DietQuality	SleepQuality	FamilyHistoryAlzheimers
0	4751	73.0	Male	Caucasian	Bachelor's	22.927749	No	13.297218	6.327112	1.347214	9.025679	No
1	4752	89.0	Male	Caucasian	NaN	26.827681	No	4.542524	7.619885	0.518767	7.151293	No
2	4753	73.0	NaN	Other	High School	17.795882	NaN	19.555085	7.844988	1.826335	9.673574	Yes
3	4754	74.0	Female	Caucasian	High School	33.800817	NaN	12.209266	8.428001	7.435604	8.392554	No
4	4755	NaN	Male	Caucasian	NaN	20.716974	No	18.454356	6.310461	0.795498	5.597238	No
...	...	...	...	...	...	...	...	...	...	...	...	...
2347	5535	62.0	Male	Asian	Bachelor's	37.204794	NaN	15.866661	0.834477	7.408433	8.910176	No
2348	6601	83.0	Female	African American	NaN	25.730770	No	15.194961	1.019401	7.670402	6.623771	No
2349	5528	90.0	NaN	Asian	Bachelor's	15.387136	No	15.821380	NaN	2.240311	6.253344	No
2350	6815	74.0	Female	African American	NaN	38.916401	Yes	0.923593	9.658794	9.606757	8.309027	No
2351	5684	63.0	NaN	Caucasian	NaN	35.195904	Yes	4.219251	9.513502	5.694550	7.538425	No

2352 rows × 35 columns

Figure 1.1 Overview Alzheimer's Desease dataset



```
alzheimer_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2352 entries, 0 to 2351
Data columns (total 35 columns):
 #   Column                     Non-Null Count  Dtype  
---  ------                     --------------  -----  
 0   PatientID                  2352 non-null   int64  
 1   Age                        2045 non-null   float64
 2   Gender                     1762 non-null   object 
 3   Ethnicity                  2352 non-null   object 
 4   EducationLevel             1872 non-null   object 
 5   BMI                        2352 non-null   float64
 6   Smoking                    1654 non-null   object 
 7   AlcoholConsumption         2163 non-null   float64
 8   PhysicalActivity           2163 non-null   float64
 9   DietQuality                2156 non-null   float64
 10  SleepQuality               2352 non-null   float64
 11  FamilyHistoryAlzheimers    2352 non-null   object 
 12  CardiovascularDisease      2352 non-null   object 
 13  Diabetes                   2352 non-null   object 
 14  Depression                 1580 non-null   object 
 15  HeadInjury                 2019 non-null   object 
 16  Hypertension               2352 non-null   object 
 17  SystolicBP                 2175 non-null   float64
 18  DiastolicBP                1941 non-null   float64
 19  CholesterolTotal           2125 non-null   float64
 20  CholesterolLDL             2352 non-null   float64
 21  CholesterolHDL             2352 non-null   float64
 22  CholesterolTriglycerides   2352 non-null   float64
 23  MMSE                       2352 non-null   float64
 24  FunctionalAssessment       2352 non-null   float64
 25  MemoryComplaints           2352 non-null   object 
 26  BehavioralProblems         2352 non-null   object 
 27  ADL                        2352 non-null   float64
 28  Confusion                  2352 non-null   object 
 29  Disorientation             2000 non-null   object 
 30  PersonalityChanges         1754 non-null   object 
 31  DifficultyCompletingTasks  1619 non-null   object 
 32  Forgetfulness              1937 non-null   object 
 33  Diagnosis                  2352 non-null   object 
 34  DoctorInCharge             2352 non-null   object 
dtypes: float64(15), int64(1), object(19)
memory usage: 643.3+ KB
```

Figure 1.2 Feature information and datatype

1.3 Feature descriptions

1. **PatientID**: A unique identifier assigned to each patient (4751 to 6900).
2. **Age**: The age of the patients ranges from 60 to 90 years.
3. **Gender**: Gender of the patients, where 0 represents Male and 1 represents Female.
4. **Ethnicity**: The ethnicity of the patients, coded as follows:
    - 0: Caucasian
    - 1: African American
    - 2: Asian
    - 3: Other
5. **EducationLevel**: The education level of the patients, coded as follows:
    - 0: None
    - 1: High School
    - 2: Bachelor's
    - 3: Higher
6. **BMI**: Body Mass Index of the patients, ranging from 15 to 40.
7. **Smoking**: Smoking status, where 0 indicates No and 1 indicates Yes.
8. **AlcoholConsumption**: Weekly alcohol consumption in units, ranging from 0 to 20.
9. **PhysicalActivity**: Weekly physical activity in hours, ranging from 0 to 10.
10. **DietQuality**: Diet quality score, ranging from 0 to 10.
11. **SleepQuality**: Sleep quality score, ranging from 4 to 10.
12. **FamilyHistoryAlzheimers**: Family history of Alzheimer's Disease, where 0 indicates No and 1 indicates Yes.
13. **CardiovascularDisease**: Presence of cardiovascular disease, where 0 indicates No and 1 indicates Yes.
14. **Diabetes**: Presence of diabetes, where 0 indicates No and 1 indicates Yes.
15. **Depression**: Presence of depression, where 0 indicates No and 1 indicates Yes.
16. **HeadInjury**: History of head injury, where 0 indicates No and 1 indicates Yes.
17. **Hypertension**: Presence of hypertension, 0 indicates No, 1 indicates Yes.
18. **SystolicBP**: Systolic blood pressure, ranging from 90 to 180 mmHg.
19. **DiastolicBP**: Diastolic blood pressure, ranging from 60 to 120 mmHg.

20. **CholesterolTotal**: Total cholesterol levels, ranging from 150 to 300 mg/dL.
21. **CholesterolLDL**: Low-density lipoprotein cholesterol levels, ranging from 50 to 200 mg/dL.
22. **CholesterolHDL**: High-density lipoprotein cholesterol levels, ranging from 20 to 100 mg/dL.
23. **CholesterolTriglycerides**: Triglycerides levels, ranging from 50 to 400 mg/dL.
24. **MMSE**: Mini-Mental State Examination score, ranging from 0 to 30. Lower scores indicate cognitive impairment.
25. **FunctionalAssessment**: Functional assessment score, ranging from 0 to 10. Lower scores indicate greater impairment.
26. **MemoryComplaints**: Presence of memory complaints, where 0 indicates No and 1 indicates Yes.
27. **BehavioralProblems**: Presence of behavioral problems, where 0 indicates No and 1 indicates Yes.
28. **ADL**: Activities of Daily Living score, ranging from 0 to 10. Lower scores indicate greater impairment.
29. **Confusion**: Presence of confusion, where 0 indicates No and 1 indicates Yes.
30. **Disorientation**: Presence of disorientation, where 0 indicates No and 1 indicates Yes.
31. **PersonalityChanges**: Presence of personality changes, where 0 indicates No and 1 indicates Yes.
32. **DifficultyCompletingTasks**: Presence of difficulty completing tasks, where 0 indicates No and 1 indicates Yes.
33. **Forgetfulness**: Presence of forgetfulness, where 0 indicates No and 1 indicates Yes.
34. **Diagnosis**: Diagnosis status for Alzheimer's Disease, where 0 indicates No and 1 indicates Yes.
35. **DoctorInCharge**: This column contains confidential information about the doctor in charge, with "XXXConfid" as the value for all patients.

- This dataset was collected in csv format including 2352 rows and 35 columns Coresponding to 35 features and 2352 samples
- This dataset offers extensive insights into the factors associated with Alzheimer's Disease, including demographic, lifestyle, medical, cognitive, and functional variables. It is ideal for developing predictive models, conducting statistical analyses, and exploring the complex interplay of factors contributing to Alzheimer's Disease.

1.4 Statistics and visualization

a) Compare diagnosis rates across age groups and genders.

b) Explore how physical activity and diet affect MMSE scores.

c) Check if family history or conditions like cardiovascular disease increase diagnosis risk.

d) Study the link between blood pressure and MMSE scores.

e) Compare cholesterol levels in patients with and without Alzheimer’s.

f) Analyze how common symptoms are in diagnosed patients.

g) See how behavioral issues relate to MMSE scores.

h) Examine how smoking and alcohol affect Alzheimer’s symptoms.

i) Compare daily living scores between diagnosed and non-diagnosed patients.

j) Look at cognitive and functional differences across ethnic groups.

1.5 Data preparing

a) Handle duplicated samples

b) Remove irrelevant data

1.6 Data preprocessing

a) Classify features

b) Text-based features features encoder

c) Data splitting (train/test)

d) Impute missing values

e) Outlier, noise

f) Scaling features

g) Imbalance class

h) Multicolinerity

i) Feature mapping

j) Dimensionality reduction

1.7 Modeling

a) Naïve Bayes

b) Support Vector Machine

c) Logistic Regression

d) XGBoost

CHƯƠNG 2: ESSENTIAL KNOWLEDGE

2.1 Pandas

Pandas is a robust open-source library in Python widely used for data manipulation and analysis, especially in data science and machine learning. Its name originates from "Panel Data," highlighting its ability to handle multidimensional datasets. Pandas offers two primary data structures:

- DataFrame: A two-dimensional, tabular data structure with labeled rows and columns, akin to a spreadsheet or database table.
- Series: A one-dimensional array, similar to a single column in a DataFrame.

Key Features:

1. Handles data from various sources like CSV, Excel, SQL, and JSON.
2. Provides advanced methods for sorting, filtering, grouping, merging, and summarizing data.
3. Supports time-series data manipulation and handling of missing values.
4. Seamlessly integrates with libraries such as NumPy, Matplotlib, and Scikit-learn, making it an essential tool for data preprocessing and analysis.

Advantages of Using Pandas

1. Ease of Use: Simple syntax and extensive documentation make it beginner-friendly.
2. Optimized for Large Data: Built on NumPy, it efficiently handles large datasets.
3. Visualization Capabilities: Offers basic plotting tools and integrates with advanced libraries for visualization.
4. Data Management: Facilitates efficient and error-free data handling for projects demanding high accuracy.

Application to Alzheimer’s Dataset Analysis

- Pandas’ features are particularly well-suited for working with complex medical datasets, such as those related to Alzheimer’s disease. Key advantages include:

1. **Data Management**

   Handles diverse features like numerical data (age, blood pressure, cholesterol), categorical data (gender, ethnicity), and binary values (smoking status, family history).

   The DataFrame structure organizes data clearly, making it accessible for analysis.

2. **Missing Data Handling**

   Provides methods like isnull() and fillna() to detect and manage missing values, ensuring clean and consistent datasets.

3. **Data Transformation**

   Enables efficient transformation, such as normalizing numerical columns (age, BMI) or encoding categorical data (gender, ethnicity) for machine learning models using methods like apply(), map(), and get_dummies().

4. **Exploratory Data Analysis (EDA)**

   Quickly calculates basic statistics (mean, standard deviation, frequency) to understand features.

   Methods like describe() help identify outliers and unusual trends.

5. **Filtering and Grouping**

   Supports grouping and filtering by conditions, such as comparing patient characteristics by gender, age, or diagnosis, using groupby() and filter().

   6. Integration with Machine Learning

   Prepares data for machine learning models in libraries like Scikit-learn and TensorFlow by handling tasks like train-test splitting and exporting data as NumPy arrays.

7. **Efficiency with Large Datasets**

   Optimized for managing extensive data (e.g., thousands of patients and features) efficiently, ensuring smooth analysis workflows.

8. **Data Visualization**

   Facilitates visualization of relationships between features, such as age and Alzheimer’s risk, through plots like histograms, scatter plots, and boxplots.

2.2 Matplotlib

Matplotlib is a versatile plotting library in Python widely used for creating static, animated, and interactive visualizations. It is particularly beneficial for visualizing complex datasets like Alzheimer’s data:

1. **Comprehensive Visualization Tools:**
   Matplotlib allows the creation of various chart types such as line plots, bar charts, scatter plots, histograms, and heatmaps, which can be used to display trends, distributions, and relationships within Alzheimer’s datasets.
2. **Extensive Customization Options:**
   The library offers extensive control over plot elements, including titles, labels, colors, and line styles, allowing users to tailor visuals for specific needs, such as highlighting key variables or comparing groups within a dataset.
3. **Relationship Exploration:**
   Matplotlib supports visualizing relationships between different variables in the dataset. For example, scatter plots can be used to explore correlations between features like cholesterol and blood pressure, or between age and cognitive function scores.
4. **Comparative Analysis:**
   Matplotlib is excellent for comparing different groups or categories within the data. Bar charts, box plots, and violin plots are especially useful for comparing measures like disease prevalence, symptoms, or risk factors across different demographic groups.
5. **Time-Series Data Handling:**
   Matplotlib excels in visualizing time-based data. Line plots, for example, can show how cognitive function scores or symptom progression change over time, which is essential for tracking Alzheimer’s disease development.
6. **Seamless Integration with Pandas:**
   Matplotlib integrates seamlessly with Pandas DataFrames, making it easy to pass data directly from a DataFrame into a plot, simplifying the workflow for users who are already manipulating data with Pandas.

7. **Effective Outlier Detection:**

   The library supports detecting anomalies or outliers in the data through visualization techniques like box plots and histograms, which are crucial for identifying unexpected patterns or errors in data, such as extreme values in health parameters.

8. **Data Distribution Representation:**

   With tools like histograms and density plots, Matplotlib helps in visualizing the distribution of numerical features such as age, cholesterol levels, or MMSE scores, providing insight into the data spread and identifying skewed distributions.

9. **High-Quality Reporting and Presentation:**

   Matplotlib enables the creation of publication-quality visualizations that are suitable for academic reports, research papers, and presentations. The library’s flexibility makes it possible to generate visually appealing charts for data analysis discussions.

10. **Integration with Other Visualization Libraries**:

    While Matplotlib is highly capable on its own, it also integrates well with higher-level libraries like Seaborn and Plotly, allowing users to create even more sophisticated and aesthetically pleasing visualizations, including statistical plots and interactive charts.

2.3 Sklearn

Scikit-learn (sklearn) is a powerful machine learning library in Python that provides a comprehensive suite of tools for data preprocessing, model selection, evaluation, and optimization. It plays a crucial role in working with healthcare datasets, such as Alzheimer’s, by offering methods to process and analyze complex data efficiently.

1. **Data Preprocessing and Standardization:**

   Scikit-learn offers tools like StandardScaler and MinMaxScaler to standardize and scale input features, such as blood pressure, cholesterol, and BMI, ensuring that machine learning models process the data effectively, especially when features have different ranges or units.

2. **Train-Test Split:**

   The train_test_split function allows for easy partitioning of datasets into training and test sets, which is essential for model validation and preventing overfitting. This ensures that the model generalizes well on unseen data.

3. **Model Selection:**

   Scikit-learn supports a wide range of supervised and unsupervised machine learning algorithms, such as linear regression, decision trees, SVM, and k-means clustering. For instance, logistic regression can be used to predict Alzheimer’s risk based on features like age, blood pressure, and cholesterol, while SVM can be employed for classification tasks.

4. **Model Evaluation and Optimization:**

   Scikit-learn provides tools like cross-validation and grid search to evaluate model performance and optimize hyperparameters. cross_val_score allows for model evaluation across multiple splits of data, while GridSearchCV helps find the best combination of parameters to enhance prediction accuracy.

5. **Handling Missing Values:**

   Missing data is a common issue in healthcare datasets, and Scikit-learn’s SimpleImputer can replace missing values with mean, median, or mode values, ensuring the integrity of the dataset without disrupting model training.

6. **Feature Transformation:**

   Scikit-learn includes tools like OneHotEncoder and PolynomialFeatures to transform categorical features into binary format and enhance the learning capabilities of models, which is essential for handling non-numeric data such as gender and ethnicity in Alzheimer’s datasets.

7. **Model Interpretability**:

   Scikit-learn offers interpretable models like decision trees and linear regression, which allow practitioners to understand how input features influence outcomes. This is particularly valuable in healthcare contexts, where the explanation of model decisions can help clinicians understand the factors driving the risk of Alzheimer’s.

8. **Integration with Other Libraries:**

   Scikit-learn integrates seamlessly with libraries like Pandas and Matplotlib, enabling users to manipulate data, visualize results, and analyze key metrics like accuracy, sensitivity, and specificity, thus facilitating end-to-end analysis workflows.

9. **Handling Large Datasets:**

   Scikit-learn is optimized for efficiency, making it capable of handling large datasets, such as those used in Alzheimer’s research, without compromising performance. This is especially important for training complex models on large-scale medical data.

10. **Support for Deep Learning:**

    Although primarily focused on traditional machine learning algorithms, Scikit-learn also supports simple deep learning models through tools like MLPClassifier and MLPRegressor. These models are useful for more complex tasks, such as predicting Alzheimer’s when there are intricate interactions between features.

2.4 Feature Engineering

Feature engineering refers to the process of transforming raw data into meaningful features that enhance the performance of machine learning models. It involves selecting, modifying, or creating new features from the existing dataset to better represent the underlying patterns in the data. The goal is to improve model accuracy, reduce overfitting, and make the model learn more efficiently.

Feature engineering can include:

1. **Handling Missing Values:**

   Alzheimer’s datasets often contain missing values, especially in medical data where certain tests may not be available for every patient. Feature engineering involves choosing strategies for imputing or removing missing data. For instance, using imputation techniques like replacing missing values with the mean, median, or mode, or using more advanced methods such as KNN imputation.

2. **Categorical Data Encoding:**

   Medical datasets often include categorical variables, such as gender, ethnicity, or medical history. These non-numeric features must be encoded into a numeric format before feeding them into machine learning models. Techniques like One-Hot Encoding or Label Encoding can be used to convert these categorical variables into a format that algorithms can process effectively.

3. **Feature Scaling:**

   Variables such as blood pressure, cholesterol levels, and age can have different units and scales, making them incompatible with certain machine learning algorithms (e.g., distance-based algorithms like k-NN or SVM). Feature scaling techniques like Standardization (z-score normalization) or Min-Max Scaling can help standardize the features, ensuring that they contribute equally to the model.

4. **Creating New Features:**

   In the Alzheimer’s project, you may need to create new features from existing ones. For example:

   - Age ranges: Convert continuous age data into age ranges (e.g., 50-60, 60-70), which can be more informative for predicting Alzheimer’s risk.
   - BMI categories: Create a categorical variable based on BMI (underweight, normal, overweight, obese), which might be more useful for classification models than raw BMI values.
   - Risk score: Combine several health factors (e.g., cholesterol, blood pressure) into a single composite risk score that reflects overall cardiovascular health, which could influence Alzheimer’s risk.

5. **Handling Outliers:**

   Medical datasets often contain outliers, such as extremely high or low blood pressure or cholesterol values, which can negatively impact model performance. Techniques like winsorizing (capping extreme values), or using robust scaling methods, can mitigate the impact of outliers.

6. **Feature Interaction:**

   Certain features might work together to improve the prediction. For instance:

   - The combination of age and family history of Alzheimer’s could provide better predictive power than age alone. Creating a new feature representing this interaction could help the model learn more complex patterns.
   - Gender and blood pressure could also interact, as the relationship between these features and Alzheimer’s risk might differ by gender. Interaction features can be created by multiplying or combining features based on domain knowledge.

7. **Temporal Features:**

   If the dataset includes temporal information (e.g., the age of diagnosis, progression of symptoms, or longitudinal medical history), feature engineering might involve creating time-based features such as:

   - Time since diagnosis: The difference between the current age and the age of diagnosis could be important in understanding the disease progression.
   - Symptom progression: Creating features that represent the change in cognitive scores over time could be useful for predicting the stage of Alzheimer’s.

8. **Dimensionality Reduction:**

   If the dataset has a large number of features, Principal Component Analysis (PCA) or feature selection methods can be used to reduce the dimensionality of the data while retaining most of the variance, improving model efficiency, and potentially boosting performance.

9. **Domain-Specific Features:**

   Incorporating domain-specific knowledge (e.g., expert medical insights) into feature engineering can help create meaningful features.

2.5 Principal Component Analysis

What is PCA? PCA is a technique for dimensionality reduction that transforms high-dimensional data into a smaller set of orthogonal variables (principal components), retaining most of the data's variance. It is used to simplify large datasets, reduce noise, and improve model performance.

Pros of PCA:

1. Dimensionality Reduction: Reduces features while retaining most information.
2. Noise Reduction: Removes less important components, improving signal-to-noise ratio.
3. Uncorrelated Features: Eliminates multicollinearity by creating uncorrelated components.
4. Machine Learning Performance: Improves model efficiency and prevents overfitting.
5. Data Compression: Reduces data size for storage and processing.

Cons of PCA:

1. Loss of Interpretability: New components may not have clear meanings.
2. Linear Assumption: PCA only captures linear relationships.
3. Scaling Sensitivity: Data must be standardized before applying PCA.
4. Loss of Variance: Some information may be lost during dimensionality reduction.
5. Computational Complexity: Can be expensive for large datasets.
6. Not Effective for All Data Types: Works best with continuous data, not suitable for categorical or complex structures.

2.6 Synthetic Minority Over-sampling Technique

SMOTE is an oversampling technique used to address class imbalance in datasets by generating synthetic examples of the minority class. Rather than duplicating existing minority instances, it creates new, synthetic samples by interpolating between existing ones, improving the model's ability to learn the minority class characteristics.

Pros of SMOTE:

1. Improved Class Balance: Helps balance the number of samples in each class, preventing models from being biased toward the majority class (e.g., healthy individuals vs. Alzheimer's patients).
2. Better Model Performance: Leads to improved accuracy and predictive performance by allowing models to learn better from the minority class.
3. Prevents Overfitting: Reduces overfitting compared to simple oversampling techniques (e.g., duplicating data).
4. Supports Rare Event Prediction: Enhances the model's ability to predict less frequent but important events (e.g., Alzheimer's disease diagnosis).
5. Works Well with Imbalanced Datasets: Particularly useful in datasets with skewed class distributions like healthcare data.

Cons of SMOTE:

1. Risk of Overfitting: If synthetic data is not carefully generated, it could introduce noise, leading to overfitting.
2. Generates Noisy Data: Synthetic data might not always represent real-world variability, potentially distorting the model.
3. Complexity: Adds complexity to data preprocessing and may require tuning of parameters like the number of neighbors used for interpolation.
4. Not Suitable for All Models: Some models may not benefit from synthetic data generation.
5. Computational Overhead: SMOTE can be computationally intensive, especially with large datasets.

2.7 HyperParameter Tuning

Hyperparameter Tuning involves finding the optimal set of hyperparameters for a given model to improve its performance. This can be done using techniques such as Grid Search, Random Search, or Bayesian Optimization.

Pros of Hyperparameter Tuning:

1. Improved Model Performance: Helps achieve the best possible performance for a model by optimizing key parameters, making the predictions more accurate (e.g., in predicting Alzheimer’s risk).
2. Better Generalization: Proper tuning reduces overfitting and underfitting, ensuring the model generalizes better to unseen data.
3. Customization for Complex Datasets: Allows the model to be tailored specifically to the complexities of the Alzheimer’s dataset (e.g., dealing with imbalanced classes or diverse features).
4. Automates Model Optimization: Tools like Grid Search and Random Search automate the process, saving time while finding the best hyperparameter configuration.
5. Improved Precision: Fine-tuning hyperparameters can make the model more sensitive to important trends in the data, leading to better detection of patterns such as early signs of Alzheimer's disease.

Cons of Hyperparameter Tuning:

1. Computationally Expensive: Hyperparameter tuning can be time-consuming, especially when using exhaustive search methods like Grid Search or working with large datasets.
2. Risk of Overfitting: If not cross-validated properly, hyperparameter tuning may lead to overfitting to the training data, where the model performs well on training data but fails to generalize to new data.
3. Difficult to Tune: Some models (e.g., deep learning) have many hyperparameters, making the tuning process complex and difficult to manage effectively.
4. Requires Expert Knowledge: While tools like Grid Search and Random Search simplify the process, understanding which hyperparameters are critical to tune and how they affect model behavior still requires some domain expertise.

2.8 Cross Validation

Cross-validation is a technique used to assess the generalization ability of a machine learning model. It involves splitting the dataset into multiple subsets (folds), training the model on some folds, and testing it on the remaining fold(s). This process is repeated several times, and the model's performance is averaged to provide a more reliable estimate of its true performance.

The most common form of cross-validation is k-fold cross-validation, where the data is divided into k subsets (or folds), and the model is trained and tested k times, each time using a different fold as the test set and the remaining folds as the training set.

Types of Cross-Validation:

- k-Fold Cross-Validation: The data is split into k equal parts. The model is trained on k-1 folds and tested on the remaining fold. This is repeated for each fold.
- Stratified k-Fold Cross-Validation: A variation of k-fold that ensures each fold has a proportional representation of each class in classification problems, useful when the dataset is imbalanced.
- Leave-One-Out Cross-Validation (LOOCV): Each data point is used as a test set once, and the remaining data points are used for training.
- Leave-P-Out Cross-Validation: Similar to LOOCV, but more than one data point is left out during each iteration.
- Repeated k-Fold Cross-Validation: k-fold cross-validation is repeated several times with different random splits of the data to increase the robustness of the performance estimate.

Pros of Cross-Validation in this Project:

1. Better Model Evaluation: Cross-validation provides a more accurate estimate of the model’s performance, as it uses all data points for both training and testing. This is important in the Alzheimer’s project to ensure the model generalizes well.
2. Reduced Bias: By using multiple training and testing splits, cross-validation reduces the bias associated with random train-test splits, providing a more robust evaluation.

3. Prevents Overfitting: Cross-validation helps to detect overfitting by evaluating the model on multiple different sets of data, ensuring the model doesn’t just memorize the training data.
4. Works Well for Small Datasets: If the Alzheimer’s dataset is small, cross-validation helps in utilizing the available data efficiently by ensuring each data point is used for both training and testing.
5. Helps with Hyperparameter Tuning: Cross-validation is often used alongside hyperparameter tuning (e.g., grid search) to evaluate the performance of different hyperparameter combinations.

Cons of Cross-Validation in this Project:

1. Computationally Expensive: Cross-validation requires training the model multiple times, which can be computationally expensive, especially with large datasets or complex models.
2. Increased Training Time: With k-fold cross-validation, the model needs to be trained k times, which increases the training time proportionally to k.
3. Not Suitable for Real-Time Applications: Cross-validation can be slow, which may not be ideal in real-time systems that require immediate predictions.
4. Possible Data Leakage: If the data is not properly split, there is a risk of data leakage, where information from the test set might influence the training process.
5. Can Be Inappropriate for Time Series: Cross-validation might not be suitable for time-series data where the temporal order of the data should be preserved. In this case, time series cross-validation methods would be more appropriate.

2.9 Support Vector Machine (SVM)

Support Vector Machine (SVM) is a supervised machine learning algorithm primarily used for classification and regression tasks. It aims to find the optimal hyperplane that best separates different classes in a feature space.

Key Principles of SVM:

- Maximizing the Margin: SVM seeks to find the hyperplane that maximizes the margin between the nearest data points of each class, known as support vectors.
- Handling Non-Linear Data: For problems that cannot be linearly separated, SVM uses a kernel function to map the data to a higher-dimensional space where a linear separation is possible.

Kernel Functions: When data is not linearly separable, SVM applies kernel functions to map the data to a higher-dimensional space. Commonly used kernels include:

- *Linear Kernel*: Used for linearly separable data.
- *Polynomial Kernel*: Handles polynomially separable data.
- *RBF Kernel*: A popular kernel for non-linear classification.

Advantages of SVM:

1. High Generalization: By optimizing the margin between classes, SVM tends to perform well on unseen data.
2. Effective with Non-Linear Data: The kernel trick allows SVM to handle complex, non-linear problems.
3. Good Performance with High-Dimensional Data: SVM is effective for datasets with many features, such as text data, where the dimensionality is typically very large.

Disadvantages of SVM:

1. Computationally Expensive: SVM struggles with large datasets as its computation time increases significantly with data size.
2. Difficulty in Choosing Parameters and Kernel: Selecting the right kernel and tuning hyperparameters (such as C and γ) can be challenging and often requires extensive experimentation.

2.10 Naive Bayes

Naive Bayes is a supervised machine learning algorithm based on Bayes' Theorem, with the simplifying assumption that the input features are independent of each other (the "naive" assumption). Despite this assumption often being unrealistic, Naive Bayes performs well, particularly in classification tasks.

Key Features of Naive Bayes:

- Independence Assumption: Naive Bayes assumes that all features in the dataset are independent, which simplifies the computation of probabilities for each feature within a class.
- Classification: Naive Bayes calculates the probability of each class given the features and selects the class with the highest probability.

Types of Naive Bayes:

- Multinomial Naive Bayes: Used for discrete data, such as text classification where words are features.
- Bernoulli Naive Bayes: Used for binary data.
- Gaussian Naive Bayes: Used for continuous data.

Advantages of Naive Bayes:

1. Fast: Naive Bayes is lightweight and performs well with large datasets.
2. Good for High-Dimensional Data: Particularly effective for text data, where each word acts as a feature.
3. Simple to Use: Requires minimal hyperparameter tuning, making it easy to implement.
4. Scalable: Can be extended with different probability distributions like Multinomial or Gaussian.

Disadvantages of Naive Bayes:

1. Unrealistic Independence Assumption: Features are often dependent, especially in text data, which can reduce accuracy.
2. Limited Flexibility: Struggles to capture complex relationships between features.
3. Imbalanced Data: Naive Bayes can perform poorly when classes are imbalanced, as it may not predict the minority class well.

2.11 Logistic Regression

Logistic Regression is a supervised machine learning algorithm commonly used for binary classification tasks, where the goal is to predict a probability that an instance belongs to one of two classes (e.g., 0 or 1). Despite its name, it is used for classification, not regression. It’s especially useful when you need to understand the relationship between features and the probability of an outcome.

Extension to Multi-Class Problems: While primarily used for binary classification, Logistic Regression can handle multi-class classification through two approaches:

- *One-vs-Rest (OvR):* This method involves training a separate binary classifier for each class, where each model distinguishes one class from the others. The class with the highest predicted probability is chosen as the final result.
- *Softmax Regression (Multinomial Logistic Regression):* This approach generalizes Logistic Regression to multiple classes by assigning probabilities to each class and choosing the class with the highest probability.

Advantages:

1. Simplicity and Interpretability: Logistic Regression is easy to implement and interpret, making it ideal for understanding feature influences and making transparent predictions, especially in fields like healthcare.
2. Computational Efficiency: It’s a computationally simple model that works well for smaller datasets and provides fast results.
3. Scalability: Logistic Regression can be adapted to multi-class problems with ease using OvR or Softmax, making it versatile across different types of classification tasks.
4. Probabilistic Output: It provides probabilities, offering more than just a class prediction. This is useful in applications like risk assessment.

2.12 XGBoost

XGBoost (Extreme Gradient Boosting) is a powerful machine learning algorithm primarily used for classification tasks. It is an enhanced version of Gradient Boosting, utilizing decision trees in a boosting framework to improve model performance.

Boosting: Boosting is an ensemble learning technique where multiple weak models (learners) are trained sequentially. Each subsequent model attempts to correct the errors of the previous one. The goal is to combine these weak models to form a strong model with better performance.

Gradient Boosting: Gradient Boosting optimizes the model by minimizing a loss function, building decision trees sequentially and using the gradient of the loss function to guide the learning process.

XGBoost Features:

1. Performance Optimization: XGBoost integrates parallel processing and memory management techniques, enhancing speed and efficiency.
2. Customizable Loss Functions: It supports log-loss for binary classification and other customized loss functions.
3. Regularization: XGBoost reduces overfitting through L1 (Lasso) and L2 (Ridge) regularization.
4. Handling Missing Data: It can handle missing values in the dataset automatically, making it more robust for real-world applications.

Key Parameters:

- max_depth: Maximum depth of each decision tree, affecting the model's complexity.
- learning_rate: Controls how much the model adjusts at each iteration.
- n_estimators: Number of decision trees in the ensemble.
- subsample: Fraction of data used for each tree, helping to reduce overfitting.
- objective: Defines the optimization goal (e.g., "binary: logistic" for binary classification).
- reg_lambda and reg_alpha: L2 and L1 regularization to reduce overfitting.

Advantages:

1. High Performance: XGBoost often delivers superior results on various real-world tasks.
2. Good Generalization: Its regularization techniques help prevent overfitting, ensuring that the model generalizes well on unseen data.
3. High Customization: XGBoost supports numerous parameters and customizable loss functions, providing flexibility.
4. Handles Missing Data: It can work with datasets containing missing values without requiring preprocessing.

Disadvantages:

1. High Computational Cost: Training XGBoost models can be time-consuming, particularly with large datasets.
2. Sensitive to Hyperparameters: Optimal performance requires careful tuning of hyperparameters.
3. Data Requirements: XGBoost performs best with a significant amount of data; it is less effective with small datasets.

2.13 Confusion Matrix

A Confusion Matrix is an evaluation tool for classification models, helping to compare predicted values with actual values in the test set. It provides a comprehensive overview of the model's performance and helps identify errors made by the model.

For a binary classification problem, the confusion matrix has four components:

- **True Positive (TP):** The number of samples correctly predicted as positive class.
- **False Positive (FP)**: The number of samples incorrectly predicted as positive when they are actually negative.
- **True Negative (TN):** The number of samples correctly predicted as negative class.
- **False Negative (FN)**: The number of samples incorrectly predicted as negative when they are actually positive.

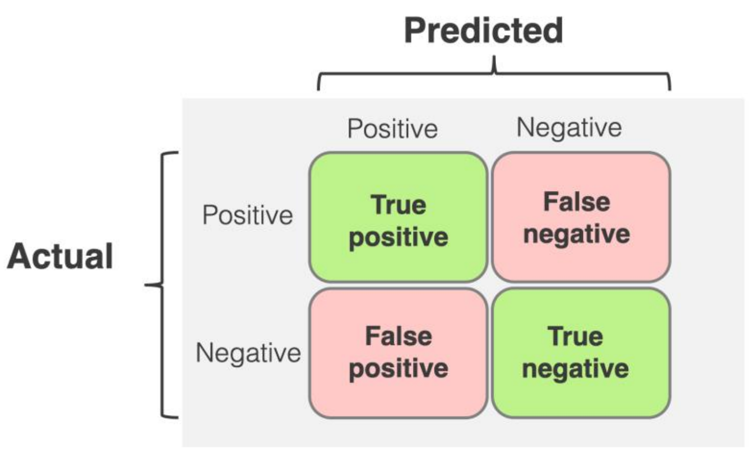


Figure 2.1 Confusion matrix

Advantages:

1. Allows detailed analysis of the types of errors the model makes (e.g., predicting positive when it should be negative or vice versa).
2. Provides a clear view of the model's performance for each class.

Disadvantages:

1. It does not provide information on the model’s overall accuracy; it focuses primarily on the correct and incorrect classifications between the classes.

2.14 Evaluation Metrics

Evaluation metrics help measure a model's performance from different perspectives. Here are some important metrics for classification models:

1. Accuracy:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

- **Definition**: Accuracy is the percentage of correct predictions out of all the predictions made.
- **Pros**: It's simple to understand and easy to calculate.
- **Cons**: Accuracy is not reliable when dealing with imbalanced data. For example, if one class is much more frequent than the other, the model could achieve high accuracy simply by predicting the majority class, ignoring the minority class.

2. **Precision:**

$$\text{Precision} = \frac{TP}{TP + FP}$$

- **Definition**: Precision measures the accuracy of the positive predictions. It is the proportion of actual positive cases among all the cases that were predicted as positive by the model.
- **Pros**: Precision helps to understand how accurate the model is when predicting the positive class.
- **Cons:** Precision can be misleading when the positive class is rare, as it may not reflect the model's ability to distinguish between classes.

3. **Recall:**

$$\text{Recall} = \frac{TP}{TP + FN}$$

- **Definition**: Recall, or sensitivity, measures how well the model identifies all the actual positive instances. It is the proportion of actual positive cases that the model correctly predicted as positive.
- **Pros**: Recall is useful when you want to know how good the model is at identifying all relevant instances, especially in tasks like detecting rare diseases.
- **Cons**: A high recall can result in many false positives (incorrectly predicting positives), which can be undesirable depending on the context.

4. **F1-Score:**

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

- **Definition**: F1-Score is the harmonic mean of precision and recall. It is especially useful when you need to balance both precision and recall, particularly in situations with imbalanced data.
- **Pros**: F1-Score provides a balanced view of both precision and recall, making it a good metric for imbalanced classification problems.
- **Cons**: F1-Score might not always offer a complete picture if not considered along with other factors like accuracy.

2.15 ROC-AUC

ROC-AUC (Receiver Operating Characteristic - Area Under the Curve) is a performance evaluation metric primarily used in binary classification problems to assess how well a model distinguishes between two classes.

ROC Curve:

The ROC curve is a graphical representation of a model's performance across different thresholds. It plots:

- True Positive Rate (TPR) or Recall: The proportion of actual positives correctly identified by the model.
- False Positive Rate (FPR): The proportion of actual negatives incorrectly classified as positive by the model.

As the threshold changes, the curve shows how well the model distinguishes between the two classes.

AUC (Area Under the Curve):

- AUC is the area under the ROC curve, quantifying the model's ability to distinguish between positive and negative classes.
- AUC = 0.5: The model is no better than random guessing.
- AUC = 1: The model perfectly classifies all points.
- AUC < 0.5: The model performs worse than random guessing.

Pros of ROC-AUC:

- Threshold-Independent: ROC-AUC evaluates the model's performance over all possible thresholds, making it a robust metric regardless of where the decision boundary is set.
- Effective for Imbalanced Data: It provides a more balanced evaluation of the model, unlike accuracy, which can be misleading when the dataset is imbalanced.

Cons of ROC-AUC:

- Not Ideal for Multi-Class Problems: While ROC-AUC works well for binary classification, it becomes more complex and less interpretable in multi-class scenarios.
- Doesn't Reflect Precision-Recall Trade-off

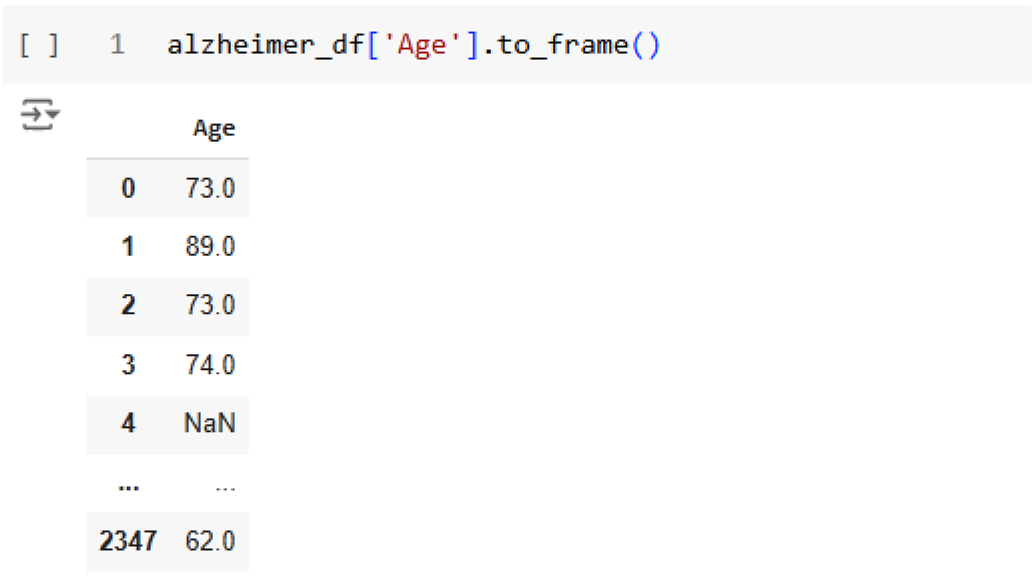
CHƯƠNG 3: SOLUTIONS

3.1. Statistics and visualization

```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Khai báo các thư viện cần thiết để trực quan hóa dữ liệu

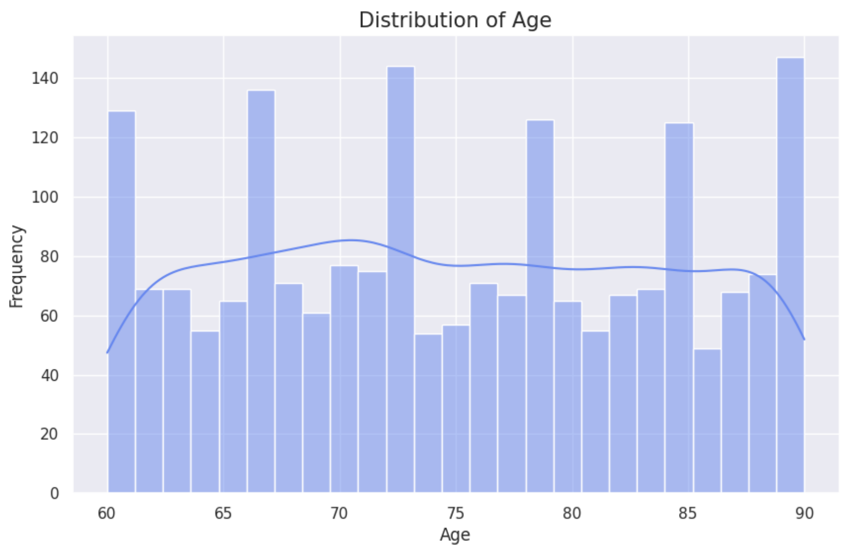
a) Biểu đồ phân phối tuổi bệnh nhân



```python
alzheimer_df['Age'].to_frame()
```

```
       Age
0     73.0
1     89.0
2     73.0
3     74.0
4      NaN
...    ...
2347  62.0
```

```
[ ]  # plot a distribution of age
     plt.figure(figsize=(10, 6))
     sns.histplot(alzheimer_df['Age'], kde=True, bins=25, palette='Blues')
     plt.title('Distribution of Age', fontsize=15)
     plt.xlabel('Age', fontsize=12)
     plt.ylabel('Frequency', fontsize=12)
     plt.show()

```



Hình 3.1 Biểu đồ trực quan phân phối dữ liệu tuổi của bệnh nhân

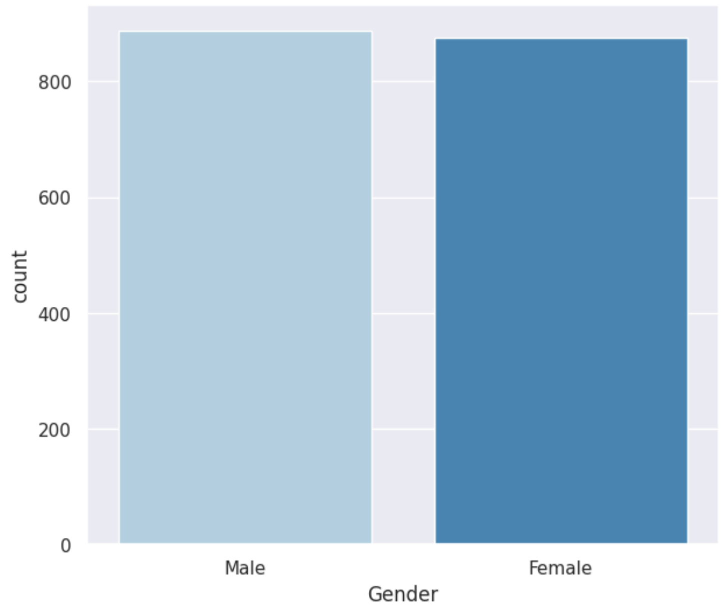
Em sử dụng biểu đồ histogram để có thể trực quan hóa được phân phối của dữ liệu tuổi. Thư viện seaborn hỗ trợ 1 hàm histplot() để có thể làm việc này 1 cách đơn giản hơn. Hàm histplot nhận đầu vào là 1 cột dữ liệu số liên tục từ DataFrame. Sau đó vẽ lên biểu đồ phân phối của dữ liệu đó.

b) Biểu đồ cột tỷ lệ giới tính

```
[ ]  alzheimer_df["Gender"].to_frame()
```

	Gender
0	Male
1	Male
2	NaN
3	Female
4	Male
...	...
2347	Male
2348	Female

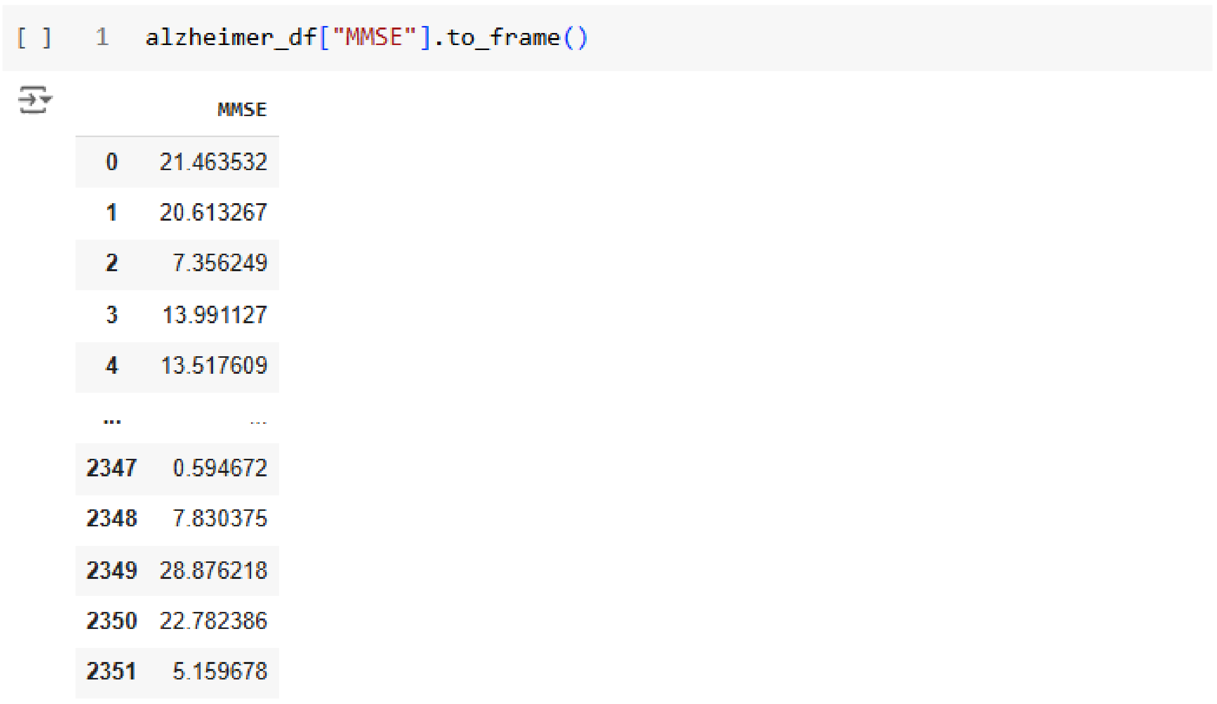
```
[ ]  # visualize the proportion of gender
     plt.figure(figsize=(7, 6))
     sns.countplot(data=alzheimer_df, x='Gender', palette='Blues')
```



Hình 3.2 Biểu đồ trực quan tỉ lệ dữ liệu giới tính của bệnh nhân

Em sử dụng hàm countplot() được hỗ trợ trong thư viện seaborn để trực quan được dữ liệu tỉ lệ giới tính trong bộ dữ liệu này, Tỉ lệ nam và nữ được thể hiện tương đối cân bằng thông qua 2 cột trên. Điều đó cho thấy việc mắc các căn bệnh về trí nhớ không liên quan nhiều đến việc giới tính của bệnh nhân là nam hay nữ.

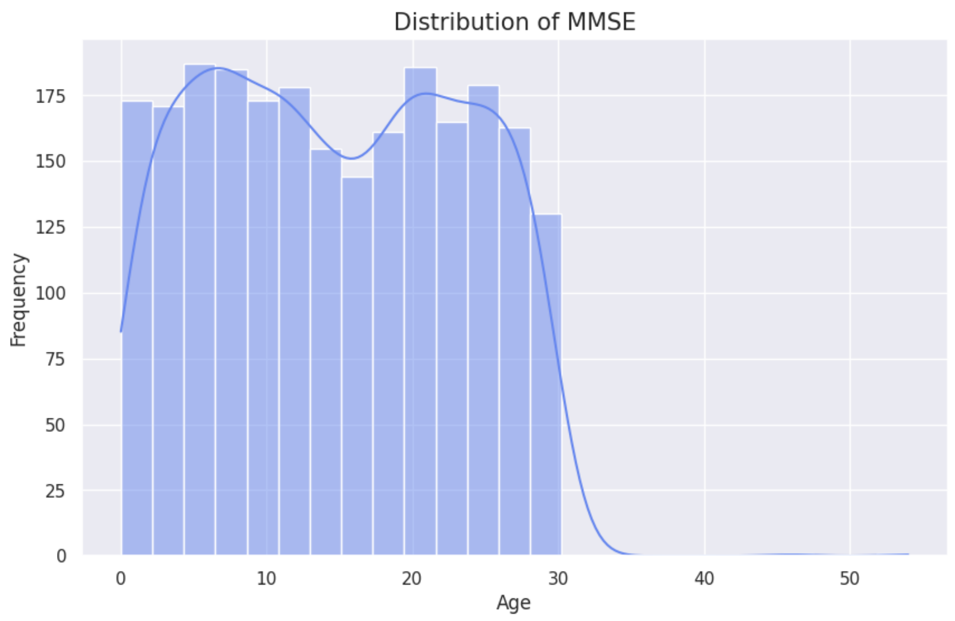
c) Biểu đồ phân phối điểm MMSE



```
[ ]  alzheimer_df["MMSE"].to_frame()
```

```
           MMSE
   0  21.463532
   1  20.613267
   2   7.356249
   3  13.991127
   4  13.517609
 ...        ...
2347   0.594672
2348   7.830375
2349  28.876218
2350  22.782386
2351   5.159678
```

```
plt.figure(figsize=(10, 6))
sns.histplot(alzheimer_df['MMSE'], kde=True, bins=25, palette='Blues')
plt.title('Distribution of MMSE', fontsize=15)
plt.xlabel('Age', fontsize=12)
plt.ylabel('Frequency', fontsize=12)
plt.show()
```



Hình 3.3 Biểu đồ trực quan phân phối dữ liệu MMSE của bệnh nhân

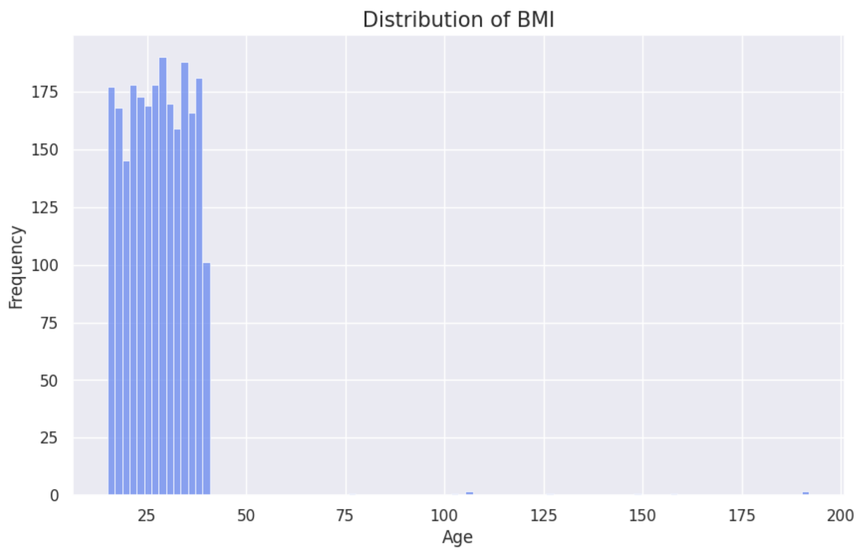
Biểu đồ trên cho thấy phân phối dữ liệu MMSE (Mini mental state examination) của bệnh nhân khá đồng đều ngoại trừ nửa từ 40-50 nhìn có vẻ như nằm ngoài phân phối của dữ liệu, khả năng cao sẽ là nhiễu có thể cần phải loại bỏ về sau.

d) Biểu đồ phân bố BMI

```
alzheimer_df["BMI"].to_frame()
```

	BMI
0	22.927749
1	26.827681
2	17.795882
3	33.800817
4	20.716974
...	...
2347	37.204794
2348	25.730770
2349	15.387136
2350	38.916401
2351	35.195904

```
[ ]  # visulize the BMI
     plt.figure(figsize=(10, 6))
     sns.histplot(data=alzheimer_df, x='BMI',palette='Blues')
     plt.title('Distribution of BMI', fontsize=15)
     plt.xlabel('Age', fontsize=12)
     plt.ylabel('Frequency', fontsize=12)
     plt.show()
```



Hình 3.4 Biểu đồ trực quan phân phối dữ liệu BMI của bệnh nhân

Biểu đồ phân bố BMI (Body Mass Index) cũng cho thấy nhiều điểm dữ liệu nhiễu nằm ngoài phân phối tập trung của đa số các điểm dữ liệu còn lại.

```
[ ]  from imblearn.over_sampling import SMOTE
     from imblearn import pipeline
     from imblearn.pipeline import Pipeline, make_pipeline

     smote_pipe = pipeline.Pipeline(steps=[("Synthetic over", SMOTE())])

     X_train_imputed_rescaled_resampled, y_train_resampled =
     smote_pipe.fit_resample(X_train_imputed_rescaled, y_train)


```

```
[ ]  y_train.value_counts()
```

```
Diagnosis
0.0        1111
1.0         608
Name: count, dtype: int64
```

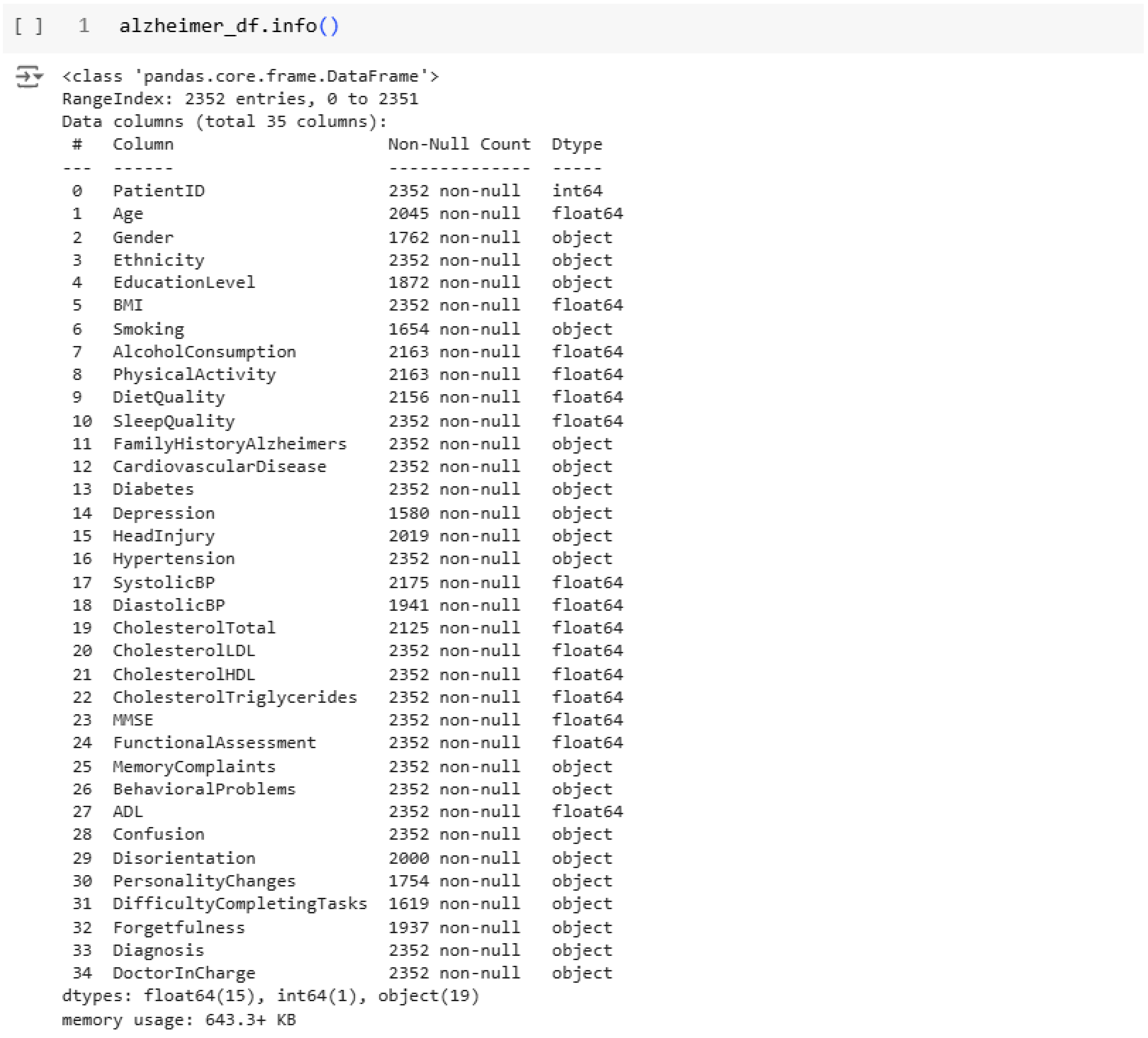
```
[ ]  y_train_resampled.value_counts()
```

```
Diagnosis
0.0        1111
1.0        1111
Name: count, dtype: int64
```

Ta sử dụng SMOTE, một kĩ thuật sử dụng để lấy mẫu lại cho class có số lượng các bản ghi thấp hơn. Ta sử dụng SMOTE và kết quả cho thấy các nhãn đã được cân bằng về mức 50:50.

3.2. Data preparing

Đầu tiên em sử dụng hàm Alzheimer_df.info() để kiểm tra các đặc trưng nào bị thiếu và có kiểu dữ liệu thế nào. Qua đó có thể biết thêm 1 số thông tin về kích thước của bộ dữ liệu, tên của các đặc trưng đó sẽ như nào, kiểu dữ liệu đã chuẩn hóa đúng chưa.

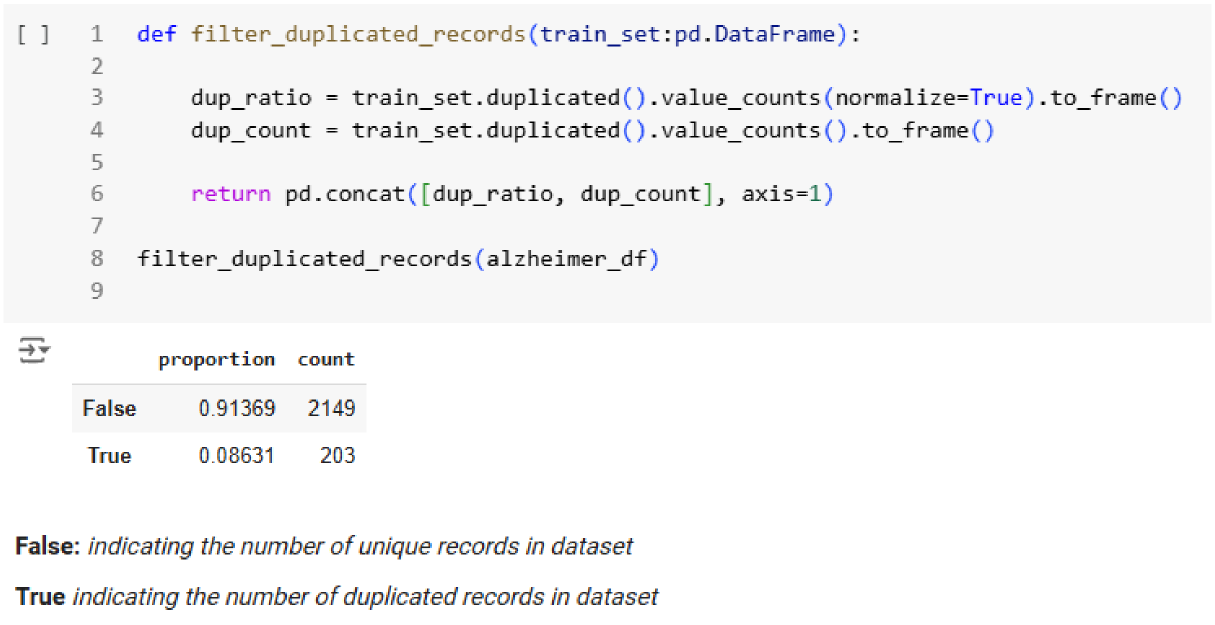


```python
alzheimer_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2352 entries, 0 to 2351
Data columns (total 35 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   PatientID                  2352 non-null   int64
 1   Age                        2045 non-null   float64
 2   Gender                     1762 non-null   object
 3   Ethnicity                  2352 non-null   object
 4   EducationLevel             1872 non-null   object
 5   BMI                        2352 non-null   float64
 6   Smoking                    1654 non-null   object
 7   AlcoholConsumption         2163 non-null   float64
 8   PhysicalActivity           2163 non-null   float64
 9   DietQuality                2156 non-null   float64
 10  SleepQuality               2352 non-null   float64
 11  FamilyHistoryAlzheimers    2352 non-null   object
 12  CardiovascularDisease      2352 non-null   object
 13  Diabetes                   2352 non-null   object
 14  Depression                 1580 non-null   object
 15  HeadInjury                 2019 non-null   object
 16  Hypertension               2352 non-null   object
 17  SystolicBP                 2175 non-null   float64
 18  DiastolicBP                1941 non-null   float64
 19  CholesterolTotal           2125 non-null   float64
 20  CholesterolLDL             2352 non-null   float64
 21  CholesterolHDL             2352 non-null   float64
 22  CholesterolTriglycerides   2352 non-null   float64
 23  MMSE                       2352 non-null   float64
 24  FunctionalAssessment       2352 non-null   float64
 25  MemoryComplaints           2352 non-null   object
 26  BehavioralProblems         2352 non-null   object
 27  ADL                        2352 non-null   float64
 28  Confusion                  2352 non-null   object
 29  Disorientation             2000 non-null   object
 30  PersonalityChanges         1754 non-null   object
 31  DifficultyCompletingTasks  1619 non-null   object
 32  Forgetfulness              1937 non-null   object
 33  Diagnosis                  2352 non-null   object
 34  DoctorInCharge             2352 non-null   object
dtypes: float64(15), int64(1), object(19)
memory usage: 643.3+ KB
```

a) Xóa các bản ghi trùng lặp

Em sử dụng hàm filter_duplicated_records() để xuất ra tỉ lệ lặp lại của các bản ghi trong bộ dữ liệu này, như dưới hình có thể thấy, tỉ lệ các bản ghi trùng lặp hay giống y đúc nhau vào khoảng 0.08631 phần trăm, số lượng cụ thể là 203 bản ghi. Các bản ghi trùng lặp sẽ bị xóa bỏ để tránh dư thừa dữ liệu cho quá trình huấn luyện mô hình và tiền xử lý dữ liệu.



```python
def filter_duplicated_records(train_set:pd.DataFrame):

    dup_ratio = train_set.duplicated().value_counts(normalize=True).to_frame()
    dup_count = train_set.duplicated().value_counts().to_frame()

    return pd.concat([dup_ratio, dup_count], axis=1)

filter_duplicated_records(alzheimer_df)
```

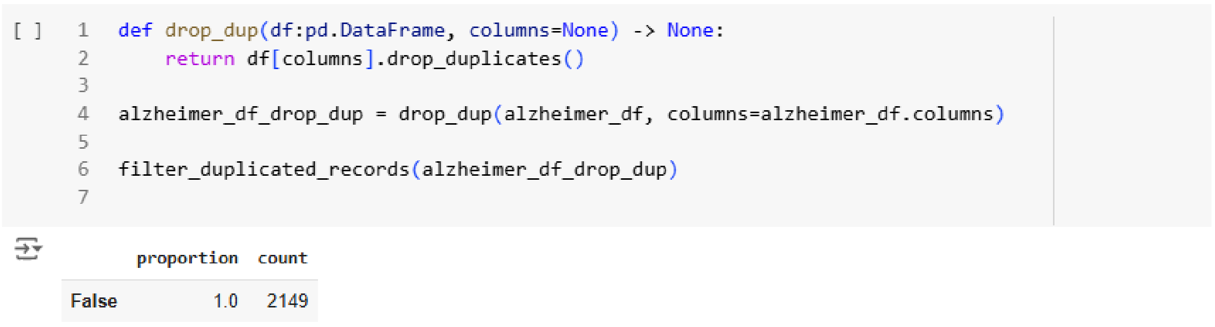
```
       proportion  count
False     0.91369   2149
True      0.08631    203
```

False: *indicating the number of unique records in dataset*

True *indicating the number of duplicated records in dataset*

Hàm bên dưới đây sẽ xóa tất cả các bản ghi trùng lặp trong DataFrame thông qua phương thức drop_duplicated() được hỗ trợ trong thư viện pandas.

Sau khi gọi hàm và kiểm tra lại có thể thấy dữ liệu trùng lặp đã được loại bỏ.

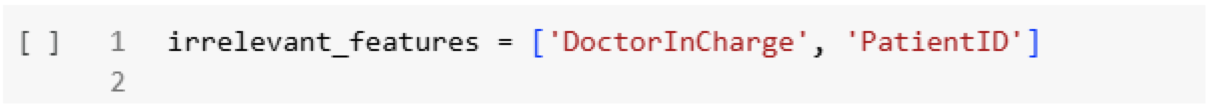


```python
def drop_dup(df:pd.DataFrame, columns=None) -> None:
    return df[columns].drop_duplicates()

alzheimer_df_drop_dup = drop_dup(alzheimer_df, columns=alzheimer_df.columns)

filter_duplicated_records(alzheimer_df_drop_dup)
```

```
       proportion  count
False         1.0   2149
```

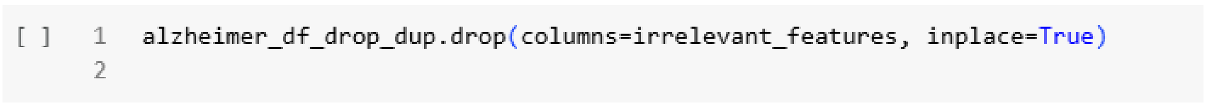
b) Xóa các cột không quan trọng trong bộ dữ liệu

Các cột bên dưới đây bao gồm “DoctorInCharge” and “PatientID” đều không quan trọng trong quá trình phân tích và huấn luyện mô hình học máy cho dữ liệu nên sẽ bị loại bỏ.



```python
irrelevant_features = ['DoctorInCharge', 'PatientID']
```

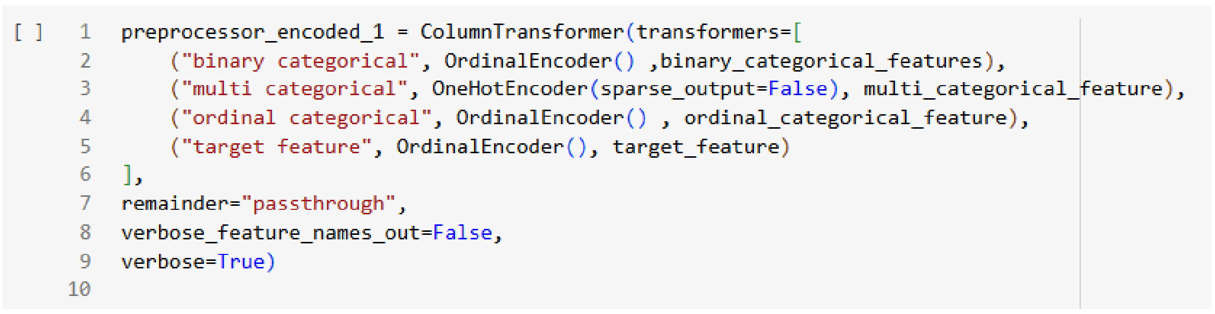
Ta sẽ loại bỏ các cột hay các đặc trưng này bằng cách sử dụng phương thức drop() được hỗ trợ trong thư viện pandas.



```python
alzheimer_df_drop_dup.drop(columns=irrelevant_features, inplace=True)
```

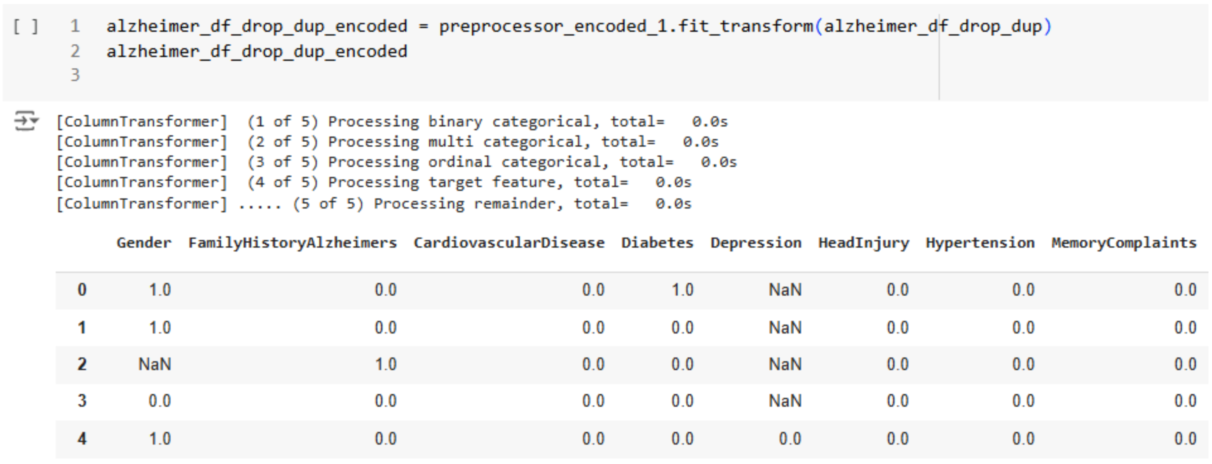
c) Mã hóa các cột chứa dữ liệu văn bản

Ta sẽ mã hóa các giá trị văn bản bằng cách sử dụng ColumnTransformer(), 1 công cụ cho phép thay đổi, mã hóa dữ liệu dạng văn bản xang dạng số 1 cách linh hoạt và tối ưu nhất. Như hình bên dưới em có định nghĩa ColumnsTransformer() cộng thêm danh sách các cột các kiểu mã hóa khác nhau cho từng loại đặc trưng, với kiểu dữ liệu và tính chất khác nhau.



```
preprocessor_encoded_1 = ColumnTransformer(transformers=[
    ("binary categorical", OrdinalEncoder() ,binary_categorical_features),
    ("multi categorical", OneHotEncoder(sparse_output=False), multi_categorical_feature),
    ("ordinal categorical", OrdinalEncoder() , ordinal_categorical_feature),
    ("target feature", OrdinalEncoder(), target_feature)
],
remainder="passthrough",
verbose_feature_names_out=False,
verbose=True)

```

Quá trình bên dưới mô tả cách áp dụng hàm và kết quả đầu ra của dữ liệu ban đầu Đã được mã hóa, không còn các dữ liệu văn bản nữa.



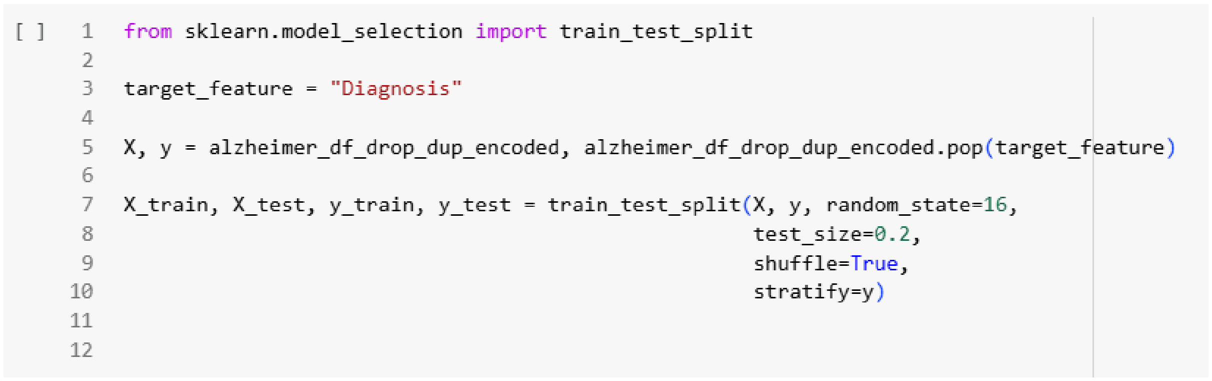
```
alzheimer_df_drop_dup_encoded = preprocessor_encoded_1.fit_transform(alzheimer_df_drop_dup)
alzheimer_df_drop_dup_encoded

```

```
[ColumnTransformer]  (1 of 5) Processing binary categorical, total=   0.0s
[ColumnTransformer]  (2 of 5) Processing multi categorical, total=   0.0s
[ColumnTransformer]  (3 of 5) Processing ordinal categorical, total=   0.0s
[ColumnTransformer]  (4 of 5) Processing target feature, total=   0.0s
[ColumnTransformer] ..... (5 of 5) Processing remainder, total=   0.0s
```

	Gender	FamilyHistoryAlzheimers	CardiovascularDisease	Diabetes	Depression	HeadInjury	Hypertension	MemoryComplaints
0	1.0	0.0	0.0	1.0	NaN	0.0	0.0	0.0
1	1.0	0.0	0.0	0.0	NaN	0.0	0.0	0.0
2	NaN	1.0	0.0	0.0	NaN	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	NaN	0.0	0.0	0.0
4	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

d) Phân chia dữ liệu

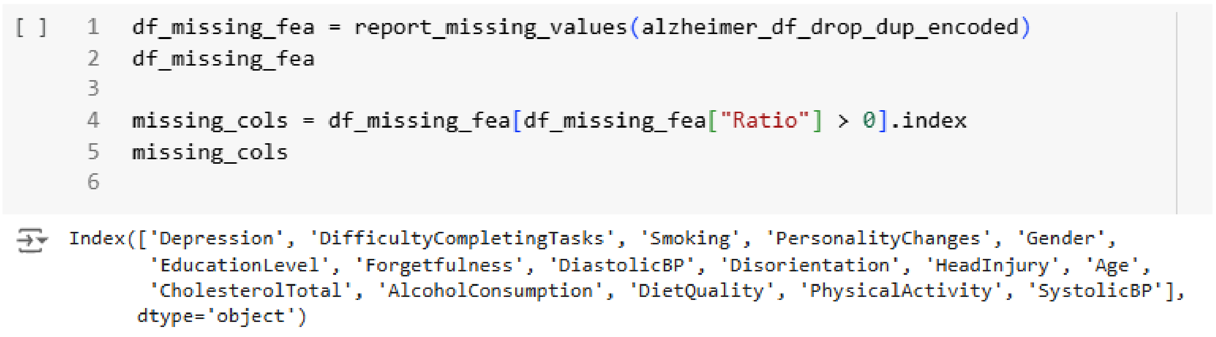
Em sẽ sử dụng hàm train_test_split được hỗ trợ trong sklearn để có thể chưa dữ liệu ra làm 2 bộ, bộ train và bộ test với tỉ lệ train:test là 80:20. Ta sẽ sử dụng Stratified sampling, hay chọn mẫu phân tổ để có thể chia đều label cho từng tập



```
from sklearn.model_selection import train_test_split

target_feature = "Diagnosis"

X, y = alzheimer_df_drop_dup_encoded, alzheimer_df_drop_dup_encoded.pop(target_feature)

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=16,
                                                    test_size=0.2,
                                                    shuffle=True,
                                                    stratify=y)


```

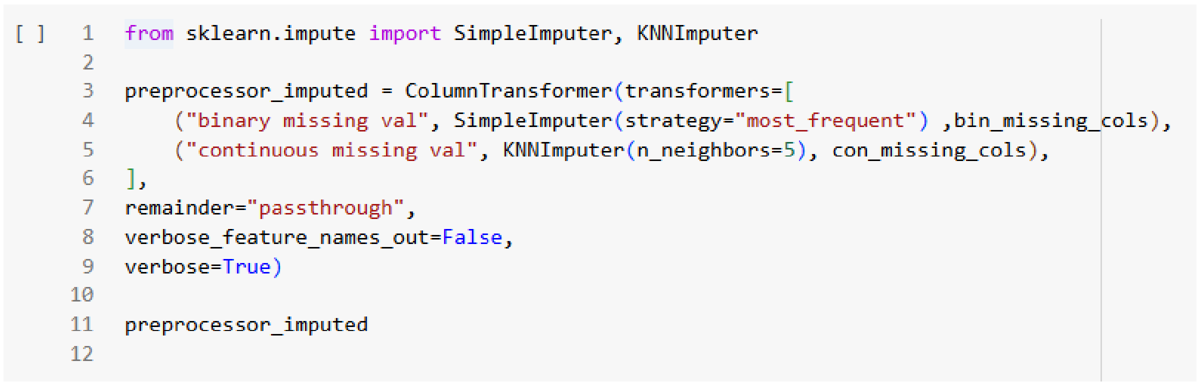
e) Xử lý dữ liệu mất mát

Các cột bên dưới được in ra dưới màn hình là những cột đều mang các giá trị thiếu Các cột này sẽ cần phải được thêm dữ liệu vào những chỗ trống để có thể phù hợp cho mô hình huấn luyện và phân tích.



```
[ ]  df_missing_fea = report_missing_values(alzheimer_df_drop_dup_encoded)
     df_missing_fea

     missing_cols = df_missing_fea[df_missing_fea["Ratio"] > 0].index
     missing_cols

Index(['Depression', 'DifficultyCompletingTasks', 'Smoking', 'PersonalityChanges', 'Gender',
       'EducationLevel', 'Forgetfulness', 'DiastolicBP', 'Disorientation', 'HeadInjury', 'Age',
       'CholesterolTotal', 'AlcoholConsumption', 'DietQuality', 'PhysicalActivity', 'SystolicBP'],
      dtype='object')
```

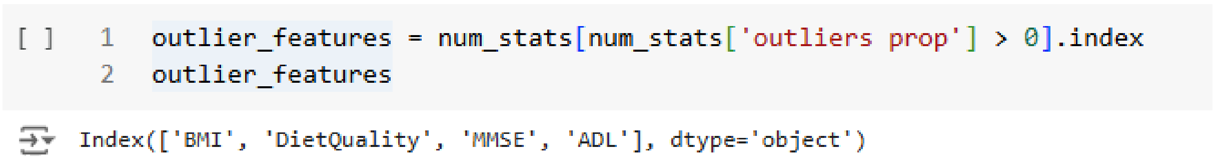
Để có thể thêm những dữ liệu mới vào chỗ trống ta sẽ sử dụng phương pháp Thế trong thư viện sklearn.impute. Hình bên dưới sử dụng SimpleImputer() and KNNImputer(), cả 2 phương pháp đều có những cách triển khai khác nhau. Chúng ta sẽ sử dụng 2 phương pháp này kết hợp với những kiểu dữ liệu khác nhau.



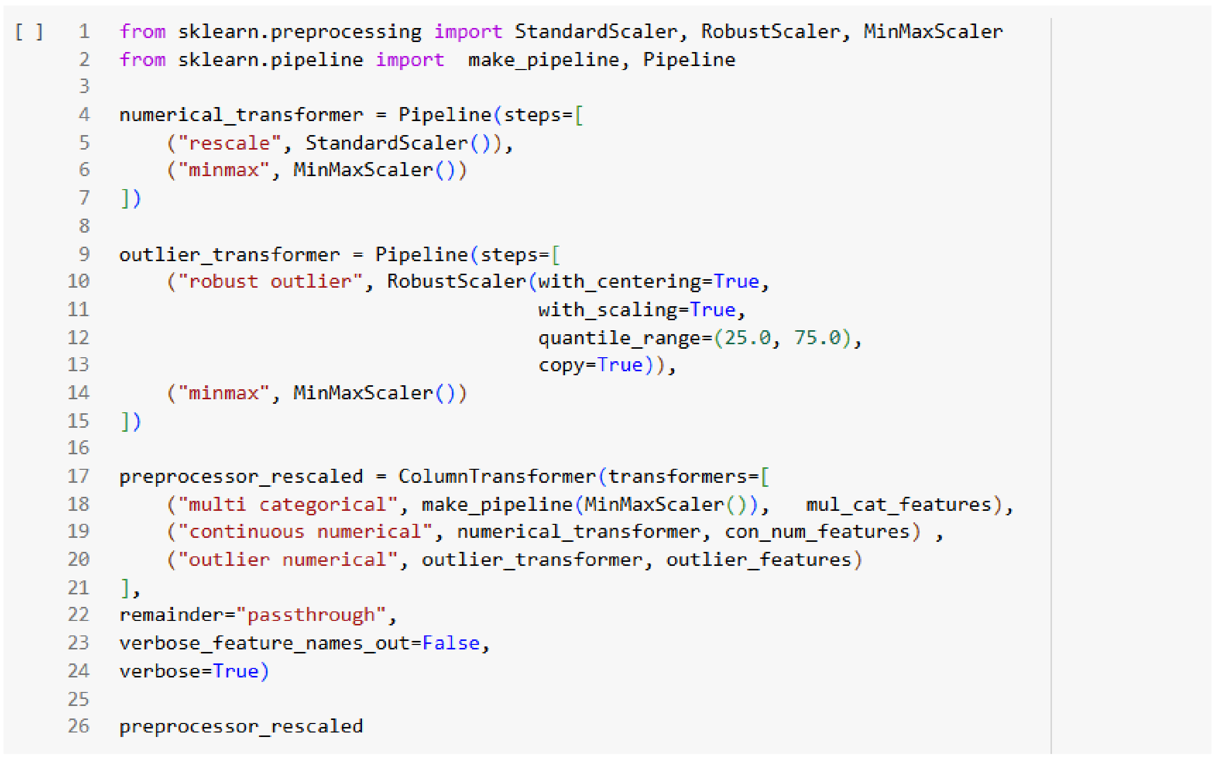
```
[ ]  from sklearn.impute import SimpleImputer, KNNImputer

     preprocessor_imputed = ColumnTransformer(transformers=[
         ("binary missing val", SimpleImputer(strategy="most_frequent") ,bin_missing_cols),
         ("continuous missing val", KNNImputer(n_neighbors=5), con_missing_cols),
     ],
     remainder="passthrough",
     verbose_feature_names_out=False,
     verbose=True)

     preprocessor_imputed
```

f) Xử lý dữ liệu ngoại lai

Bên dưới là danh sách các cột có khả năng cao sẽ chứa những dữ liệu ngoại lai hay dữ liệu nhiễu, ta cần phải tìm cách loại bỏ chúng để tối thiểu sai số cho mô hình và quá trình phân tích.



```
[ ]  outlier_features = num_stats[num_stats['outliers prop'] > 0].index
     outlier_features

Index(['BMI', 'DietQuality', 'MMSE', 'ADL'], dtype='object')
```

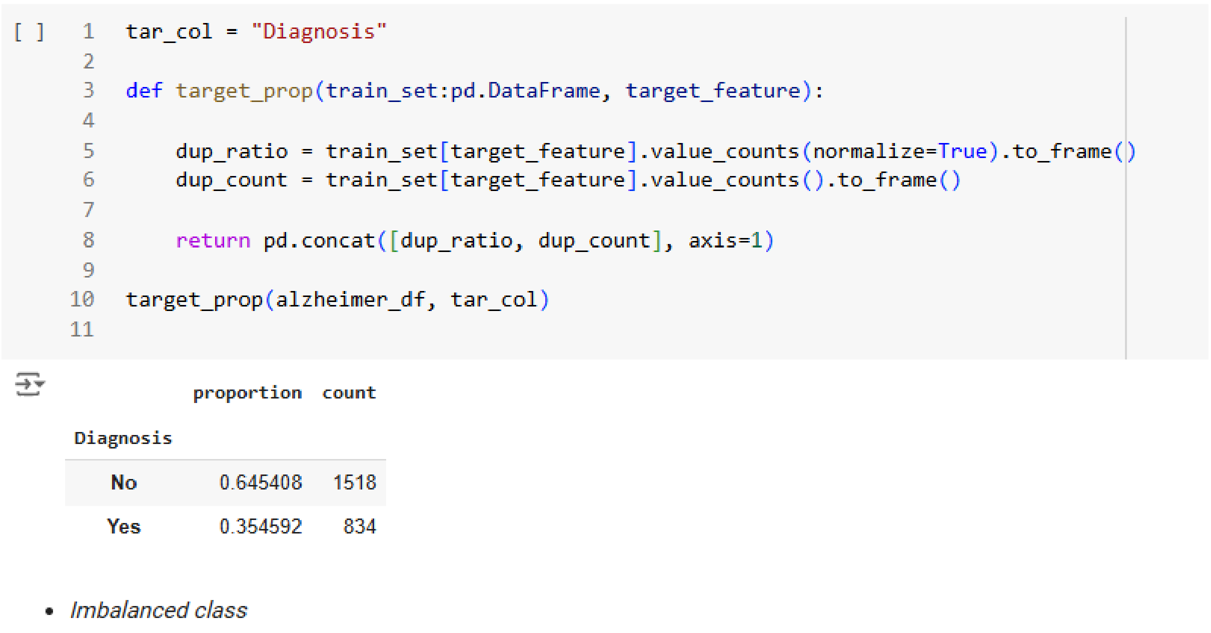


```
[ ]
from sklearn.preprocessing import StandardScaler, RobustScaler, MinMaxScaler
from sklearn.pipeline import  make_pipeline, Pipeline

numerical_transformer = Pipeline(steps=[
    ("rescale", StandardScaler()),
    ("minmax", MinMaxScaler())
])

outlier_transformer = Pipeline(steps=[
    ("robust outlier", RobustScaler(with_centering=True,
                                    with_scaling=True,
                                    quantile_range=(25.0, 75.0),
                                    copy=True)),
    ("minmax", MinMaxScaler())
])

preprocessor_rescaled = ColumnTransformer(transformers=[
    ("multi categorical", make_pipeline(MinMaxScaler()),   mul_cat_features),
    ("continuous numerical", numerical_transformer, con_num_features) ,
    ("outlier numerical", outlier_transformer, outlier_features)
],
remainder="passthrough",
verbose_feature_names_out=False,
verbose=True)

preprocessor_rescaled
```

Ta sẽ loại bỏ dữ liệu ngoại lai, hay nhiễu bằng cách sử dụng RobustScalar() cho những cột chứa dữ liệu ngoại lai này.

g) Xử lý mất cân bằng nhãn

Đầu tiên em sử dụng hàm target_prop() để hiển thị tỉ lệ mất cân bằng của nhãn trong dữ liệu. Cột nhãn mà ta đang xét là "Diagnosis" và tỉ lệ mất cân bằng của nhãn rơi vào khoảng 64:35, tỉ lệ khá lệch về nhãn No. Nhưng mà đây là vấn đề thường thấy ở những bộ dữ liệu liên quan đến y tế, là điều không thể tránh khỏi.



```
[ ]
tar_col = "Diagnosis"

def target_prop(train_set:pd.DataFrame, target_feature):

    dup_ratio = train_set[target_feature].value_counts(normalize=True).to_frame()
    dup_count = train_set[target_feature].value_counts().to_frame()

    return pd.concat([dup_ratio, dup_count], axis=1)

target_prop(alzheimer_df, tar_col)

```

```
           proportion  count
Diagnosis
   No       0.645408    1518
   Yes      0.354592     834
```

- *Imbalanced class*

3.3. Data preprocessing

3.1. Modeling

CONCLUSION

1. **Achievements:**

- The project has significantly improved the early detection and prediction of Alzheimer's disease by leveraging machine learning models such as Random Forest, XGBoost, and Artificial Neural Networks (ANNs). These models have shown high accuracy in identifying patterns in patient data, particularly in distinguishing early signs of the disease.
- The integration of AI into clinical workflows has helped doctors and healthcare providers make faster and more informed decisions, improving patient outcomes.
- The ability to process and analyze large datasets from diverse sources, including genetic, imaging, and cognitive test data, has enhanced the precision of predictions and personalized treatment approaches.

2. **Weaknesses:**

- Data Limitations: The quality and availability of data remain a challenge, particularly due to the lack of large-scale datasets that are comprehensive and well-annotated for Alzheimer's disease.
- Generalization Issues: The models trained on specific datasets may not perform equally well when applied to different patient populations or regions, leading to concerns about their generalizability.

3. **Future Enhancement:**

- Data Enhancement: Efforts should be made to acquire and integrate diverse datasets, including genetic data, biomarkers, and longitudinal studies, to improve the model’s robustness and generalizability.
- Explainable AI: Developing models with better interpretability will be crucial to gaining the trust of healthcare professionals and ensuring that these tools are used effectively in practice.
- Improved Accuracy: Ongoing research into advanced machine learning techniques, such as deep learning and transfer learning, could lead to higher accuracy rates and better detection of early-stage Alzheimer’s.

REFERENCES

[1] Dataset link: *https://www.kaggle.com/datasets/rabieelkharoua/alzheimers-disease-dataset/data*

[2] Documents and slides of mentor Nguyen Van Quyet