**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 21, 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 Nguyen Van Quyet, 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**

[COMMITMENT 1](#_Toc187438613)

[ACKNOWLEDGEMENT 2](#_Toc187438614)

[TABLE OF FIGURES 5](#_Toc187438615)

[CHAPTER 1: INTRODUCTION TO THE PROBLEM 6](#_Toc187438616)

[1.1 Problem 6](#_Toc187438617)

[1.2 Dataset overview 7](#_Toc187438618)

[1.3 Feature descriptions 8](#_Toc187438619)

[1.4 Statistics and visualization 11](#_Toc187438620)

[1.5 Data preparing 11](#_Toc187438621)

[1.6 Data preprocessing 11](#_Toc187438622)

[1.7 Modeling 11](#_Toc187438623)

[CHAPTER 2: ESSENTIAL KNOWLEDGES 12](#_Toc187438624)

[2.1 Pandas 12](#_Toc187438626)

[2.2 Matplotlib 14](#_Toc187438627)

[2.3 Sklearn 15](#_Toc187438628)

[2.4 Feature Engineering 17](#_Toc187438629)

[2.5 Principal Component Analysis 20](#_Toc187438630)

[2.6 Synthetic Minority Over-sampling Technique 20](#_Toc187438631)

[2.7 HyperParameter Tuning 21](#_Toc187438632)

[2.8 Cross Validation 23](#_Toc187438633)

[2.9 Support Vector Machine (SVM) 24](#_Toc187438634)

[2.10 Naive Bayes 25](#_Toc187438635)

[2.11 Logistic Regression 26](#_Toc187438636)

[2.12 XGBoost 27](#_Toc187438637)

[2.13 Confusion Matrix 29](#_Toc187438638)

[2.14 Evaluation Metrics 30](#_Toc187438639)

[2.15 ROC-AUC 32](#_Toc187438640)

[CHAPTER 3: SOLUTIONS 33](#_Toc187438641)

[3.1. Statistics and visualization 33](#_Toc187438642)

[3.2. Data preparing 36](#_Toc187438643)

[3.3. Data preprocessing 38](#_Toc187438644)

[3.1. Modeling 45](#_Toc187438645)

[CONCLUSION 51](#_Toc187438646)

[REFERENCES 52](#_Toc187438647)

# TABLE OF FIGURES

[Figure 1.1 Overview Alzheimer’s Desease dataset 7](#_Toc187417607)

[Figure 1.2 Feature information and datatype 7](#_Toc187417608)

[Figure 2.1 Confusion matrix 30](#_Toc187417609)

[Figure 3.1 Dataset overview 36](#_Toc187417610)

[Figure 3.2 Statistic the duplicated samples 37](#_Toc187417611)

[Figure 3.3 Source code drop duplicated samples 37](#_Toc187417612)

[Figure 3.4 Source code drop irrelevant features 38](#_Toc187417613)

[Figure 3.5 Preprocessing steps outline 38](#_Toc187417614)

[Figure 3.6 Classify features 39](#_Toc187417615)

[Figure 3.7 Source code text-based features encoding 40](#_Toc187417616)

[Figure 3.8 Source code splitting data (train/test set) 41](#_Toc187417617)

[Figure 3.9 Source code reporting missing values 41](#_Toc187417618)

[Figure 3.10 Statistic report missing value 42](#_Toc187417619)

[Figure 3.11 Souce code imputing missing values 42](#_Toc187417620)

[Figure 3.12 Handle outlier 43](#_Toc187417621)

[Figure 3.13 Source code scaling features 44](#_Toc187417622)

[Figure 3.14 Source code handling class imbalance using SMOTE 45](#_Toc187417623)

[Figure 3.15 List of base-models would be experimented 46](#_Toc187417624)

[Figure 3.16 Source code evaluating models 47](#_Toc187417625)

[Figure 3.17 Source code ploting the confusion matrix for the training set 48](#_Toc187417626)

[Figure 3.18 Confusion matrix of the base models on train set 48](#_Toc187417627)

[Figure 3.19 Source code ploting the confusion matrix for the test set 49](#_Toc187417628)

[Figure 3.20 Confusion matrix of the base models on test set 49](#_Toc187417629)

[Figure 3.21 Source code turning classification reports into Dataframe 49](#_Toc187417630)

[Figure 3.22 Classification report for all model in Dataframe form 50](#_Toc187417631)

[Figure 3.23 Cross validation for single model 51](#_Toc187417632)

[Figure 3.24 Source code Logistic regression 52](#_Toc187417633)

[Figure 3.25 Cross validation result of Logistic Regression 52](#_Toc187417634)

# CHAPTER 1: INTRODUCTION TO THE PROBLEM

## 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.

## Dataset overview

* Dataset link in Kaggle: *[Alzheimer's Desease Dataset | Kaggle](https://www.kaggle.com/datasets/rabieelkharoua/alzheimers-disease-dataset/data)*

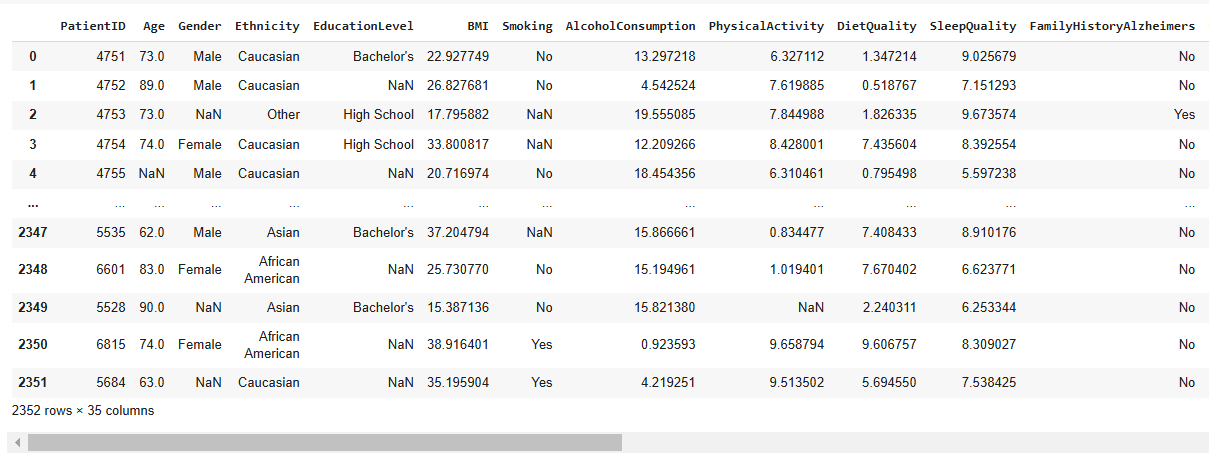


Figure 1.1 Overview Alzheimer’s Desease dataset

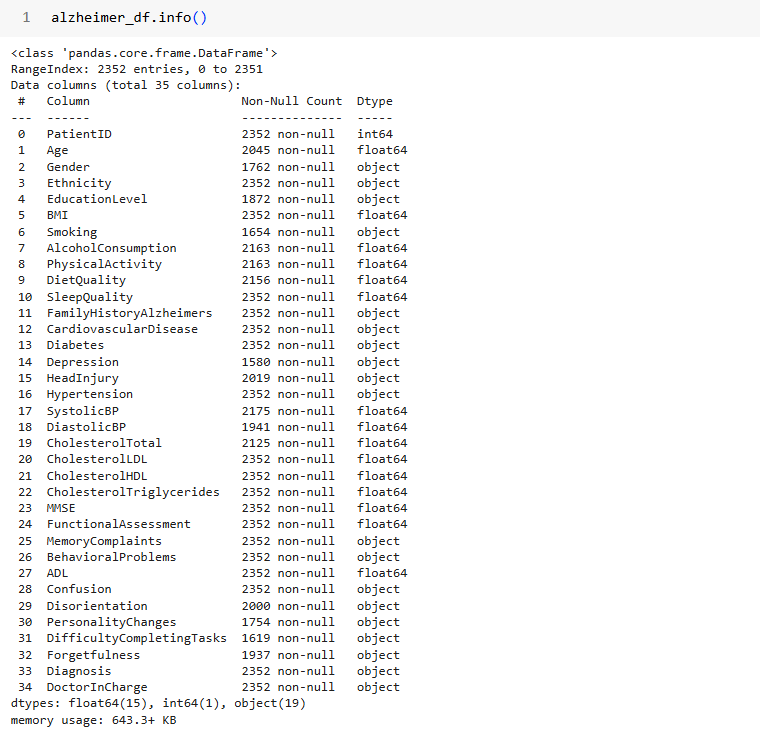


Figure 1.2 Feature information and datatype

## 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

1. **BMI**: Body Mass Index of the patients, ranging from 15 to 40.
2. **Smoking**: Smoking status, where 0 indicates No and 1 indicates Yes.
3. **AlcoholConsumption**: Weekly alcohol consumption in units, ranging from 0 to 20.
4. **PhysicalActivity**: Weekly physical activity in hours, ranging from 0 to 10.
5. **DietQuality**: Diet quality score, ranging from 0 to 10.
6. **SleepQuality**: Sleep quality score, ranging from 4 to 10.
7. **FamilyHistoryAlzheimers**: Family history of Alzheimer's Disease, where 0 indicates No and 1 indicates Yes.
8. **CardiovascularDisease**: Presence of cardiovascular disease, where 0 indicates No and 1 indicates Yes.
9. **Diabetes**: Presence of diabetes, where 0 indicates No and 1 indicates Yes.
10. **Depression**: Presence of depression, where 0 indicates No and 1 indicates Yes.
11. **HeadInjury**: History of head injury, where 0 indicates No and 1 indicates Yes.
12. **Hypertension**: Presence of hypertension, 0 indicates No, 1 indicates Yes.
13. **SystolicBP**: Systolic blood pressure, ranging from 90 to 180 mmHg.
14. **DiastolicBP**: Diastolic blood pressure, ranging from 60 to 120 mmHg.
15. **CholesterolTotal**: Total cholesterol levels, ranging from 150 to 300 mg/dL.
16. **CholesterolLDL**: Low-density lipoprotein cholesterol levels, ranging from 50 to 200 mg/dL.
17. **CholesterolHDL**: High-density lipoprotein cholesterol levels, ranging from 20 to 100 mg/dL.
18. **CholesterolTriglycerides**: Triglycerides levels, ranging from 50 to 400 mg/dL.
19. **MMSE**: Mini-Mental State Examination score, ranging from 0 to 30. Lower scores indicate cognitive impairment.
20. **FunctionalAssessment**: Functional assessment score, ranging from 0 to 10. Lower scores indicate greater impairment.
21. **MemoryComplaints**: Presence of memory complaints, where 0 indicates No and 1 indicates Yes.
22. **BehavioralProblems**: Presence of behavioral problems, where 0 indicates No and 1 indicates Yes.
23. **ADL**: Activities of Daily Living score, ranging from 0 to 10. Lower scores indicate greater impairment.
24. **Confusion**: Presence of confusion, where 0 indicates No and 1 indicates Yes.
25. **Disorientation**: Presence of disorientation, where 0 indicates No and 1 indicates Yes.
26. **PersonalityChanges**: Presence of personality changes, where 0 indicates No and 1 indicates Yes.
27. **DifficultyCompletingTasks**: Presence of difficulty completing tasks, where 0 indicates No and 1 indicates Yes.
28. **Forgetfulness**: Presence of forgetfulness, where 0 indicates No and 1 indicates Yes.
29. **Diagnosis**: Diagnosis status for Alzheimer's Disease, where 0 indicates No and 1 indicates Yes.
30. **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.

## Statistics and visualization

1. Check if family history or conditions like cardiovascular disease increase diagnosis risk.
2. Analyze how common symptoms are in diagnosed patients.
3. Look at cognitive and functional differences across ethnic groups.

## Data preparing

1. Handle duplicated samples
2. Remove irrelevant data

## Data preprocessing

1. Classify features
2. Text-based features encoder
3. Data splitting (train/test)
4. Impute missing values
5. Outlier, noise
6. Scaling features
7. Imbalance class

## Modeling

1. Base models, ensemble models
2. Validating models
3. Confusion matrices (train and test)
4. Classification reports (train and test)
5. Hyperparameter tuning.

# CHAPTER 2: ESSENTIAL KNOWLEDGES



## 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.

## 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.

1. **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.

1. **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.

1. **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.

1. **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.

1. **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.

1. **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.

1. **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.

1. **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.

1. **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.

## 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.

1. **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.

1. **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.

1. **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.

1. **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.

1. **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.

1. **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.

1. **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.

1. **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.

1. **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.

## 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.

1. **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.

1. **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.

1. **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.

1. **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.

1. **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.

1. **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.

1. **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.

1. **Domain-Specific Features:**

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

## Principal Component Analysis

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.

## 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.

## 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.

## 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.

## 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.

## 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.

## 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.

## 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.

## 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.

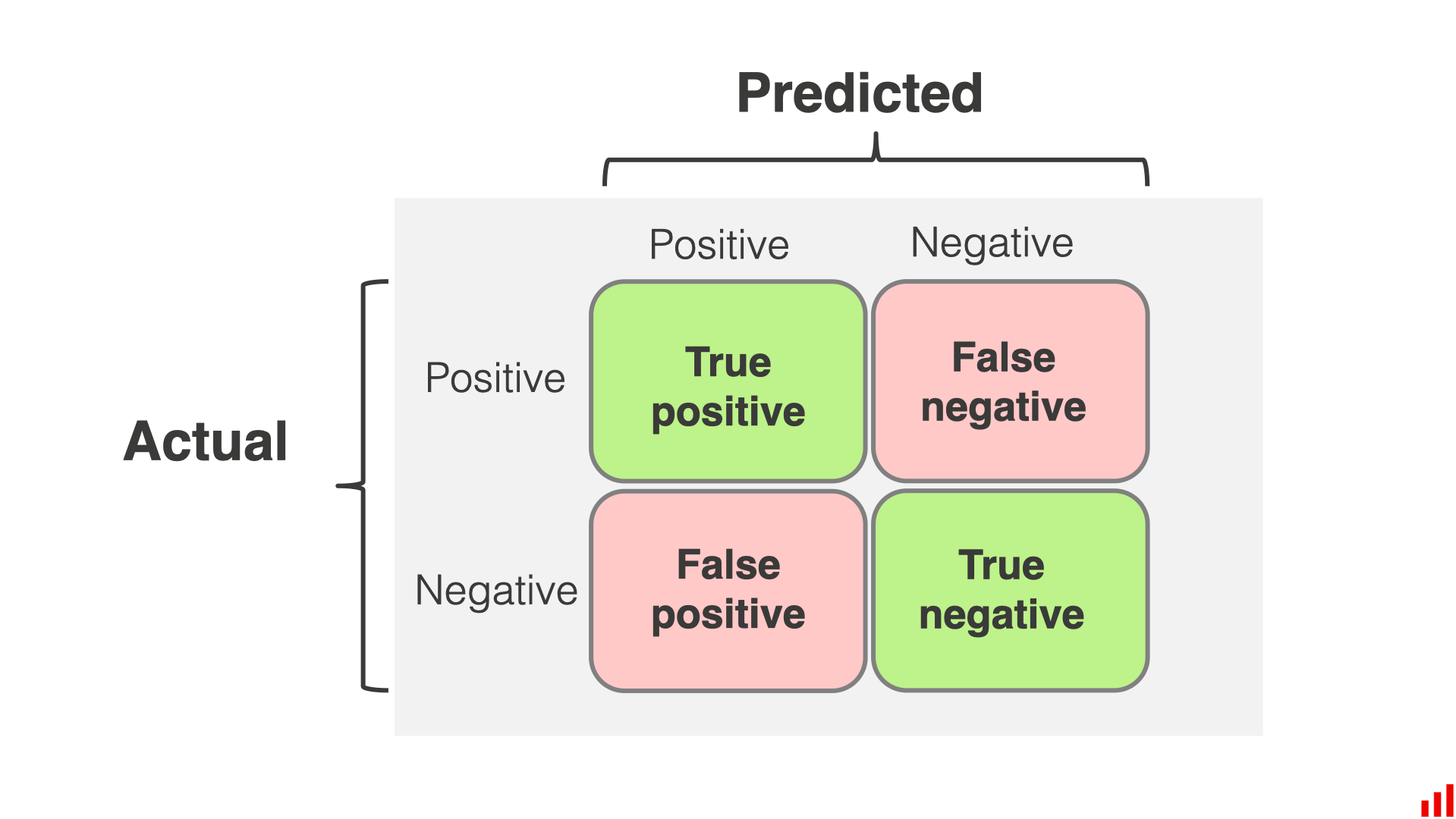


Figure 2.1 Confusion matrix

**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.

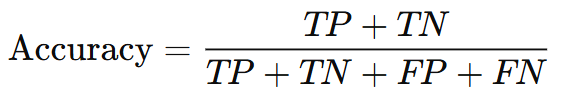
**Disadvantages:**

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

## Evaluation Metrics

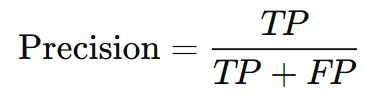
Evaluation metrics help measure a model's performance from different perspectives. Here are some important metrics for classification models:

1. **Accuracy:**



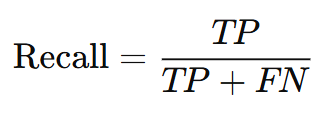
* **Definition**: Accuracy is the percentage of correct predictions out of all the predictions made.
* **Pros**: It's simple to understand and easy to calculate.
* **Cons**: Accuracy is not reliable when dealing with imbalanced data. For example, if one class is much more frequent than the other, the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

1. **Precision:**



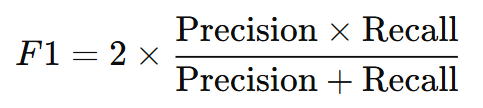
* **Definition**: Precision measures the accuracy of the positive predictions. It is the proportion of actual positive cases among all the cases that were predicted as positive by the model.
* **Pros**: Precision helps to understand how accurate the model is when predicting the positive class.
* **Cons:**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

1. **Recall:**



* **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.

1. **F1-Score:**



* **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.

## 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:**

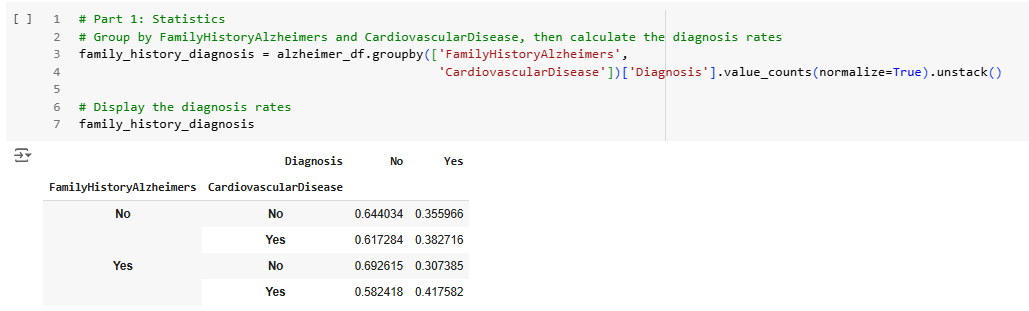
* Not Ideal for Multi-Class Problems: While ROC-AUC works well for binary classification, it becomes more complex and less interpretable in multi-class scenarios.
* Doesn't Reflect Precision-Recall Trade-off

# CHAPTER 3: SOLUTIONS

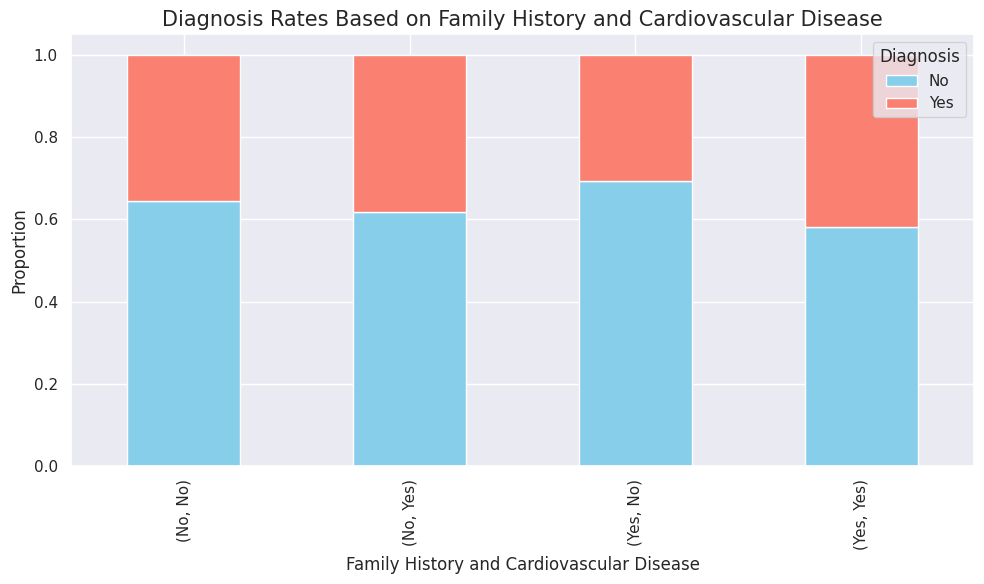
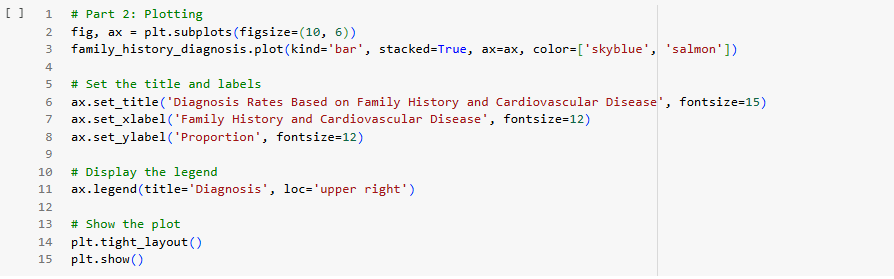
## Statistics and visualization

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

As you can see in the code below calculating the Diagnosis rate based on the Family History and Cardiovascular Disease.

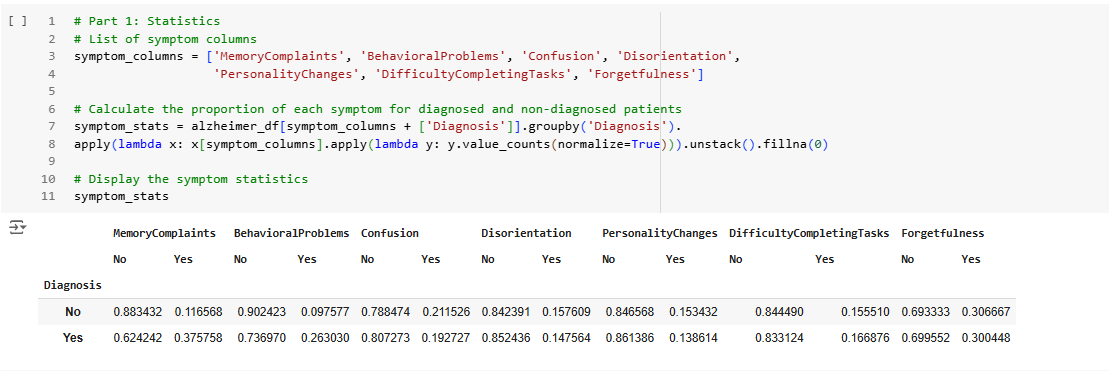


You can read the code below for more understandable



1. Analyze how common symptoms are in diagnosed patients.

The code below was created to statistic and plot the symptoms and diagnosis patients relationships.

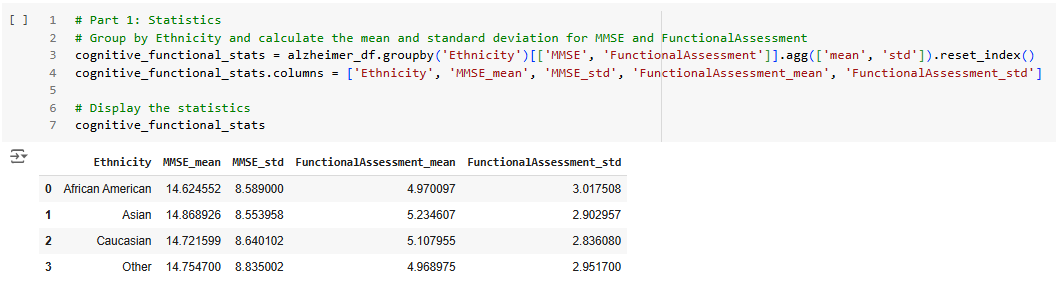


The code generates a series of bar plots comparing the proportions of symptoms between diagnosed and non-diagnosed patients for each symptom listed in *symptom\_columns*. Using a loop, it dynamically creates subplots, where each plot corresponds to a specific symptom, and visualizes the data using skyblue bars for non-diagnosed patients and salmon bars for diagnosed patients. The plots are customized with titles, axis labels, and a legend positioned in the upper-right corner. The layout is adjusted for proper spacing, and all the plots are displayed together.



1. Look at cognitive and functional differences across ethnic groups.

The code calculates the mean and standard deviation of `MMSE` (Mini-Mental State Examination) and `FunctionalAssessment` scores for different ethnic groups in an Alzheimer dataset. The statistics are grouped by `Ethnicity` and displayed in a table with columns for mean and standard deviation values. The table shows results for African American, Asian, Caucasian, and Other groups, highlighting variations in cognitive and functional assessment scores across these groups.





## Data preparing

First, let’s take a look at this information below. The information below is created by the *.info()* function supported directly as the method of class pd.DataFrame. The informations give us a general overview of the dataset including some basic information such as: column name, data-type, size, non-null count,..

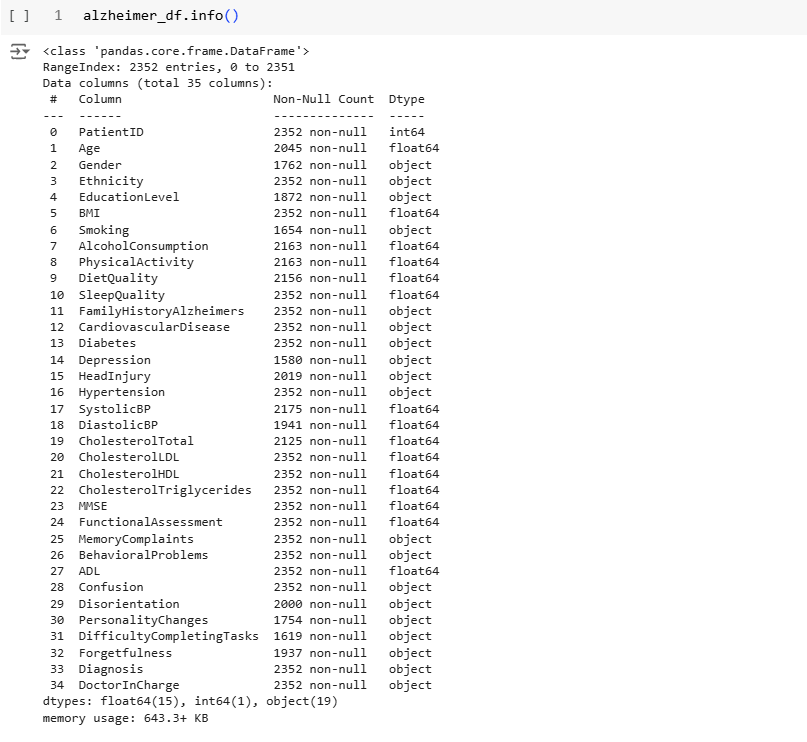


Figure 3.1 Dataset overview

1. Handle duplicated samples

I utilized the *filter\_duplicated\_records()* function to calculate the duplication rate of records within this dataset. As shown in the figure, the percentage of duplicated or completely identical records is approximately **0.08631%**, with a total of 203 duplicated entries.

These duplicate records will be removed to avoid redundancy in the dataset, which could otherwise negatively impact the data preprocessing and model training stages. By eliminating these duplicates, we aim to ensure the dataset remains clean, efficient, and representative, improving the overall quality and performance of the machine learning pipeline. This step is a crucial part of data preparation, ensuring the integrity and accuracy of subsequent analysis.

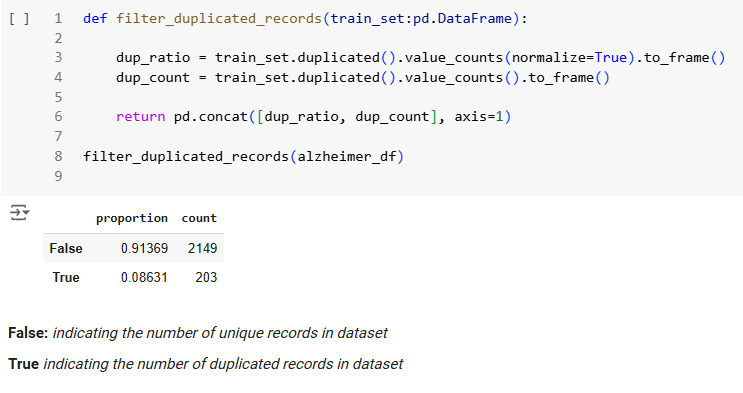


Figure 3.2 Statistic the duplicated samples

The function below removes all duplicate records from the DataFrame using the *drop\_duplicates()* method provided by the Pandas library.

After calling the function and rechecking the dataset, it is evident that all duplicated data has been successfully eliminated. This ensures a cleaner and more efficient dataset for further analysis and processing.

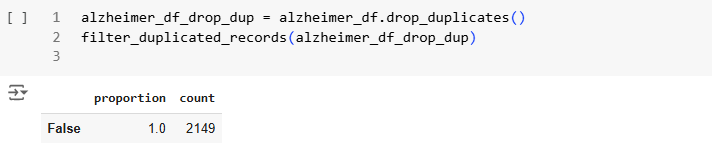


Figure 3.3 Source code drop duplicated samples

1. Remove irrelevant data

The columns listed below, including “DoctorInCharge” and “PatientID,” are deemed non-essential for data analysis and machine learning model training. Therefore, they will be removed from the dataset to streamline the process and focus on more relevant features.

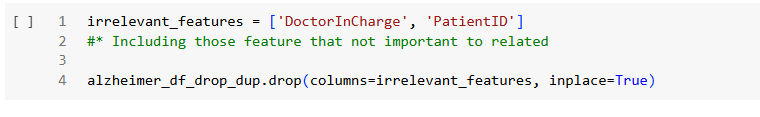


Figure 3.4 Source code drop irrelevant features

## Data preprocessing

The outline below describe all of the steps that would be applied on this dataset, The whole process of each step was construct as the pipelines for more readable and reproducible, All of these preprocessing techniques may not be enough or suitable for some kind of models, so you should be carefull utilizing it. Only experimenting those techniques and evaluate the model performances can find out what’s the main, core preprocessing or maybe redundant at worst.



Figure 3.5 Preprocessing steps outline

1. Classify features

Organizing features into distinct categories—such as binary categorical, multi-categorical, ordinal categorical, continuous numerical, and the target feature—is an essential part of preparing data for machine learning. This classification helps us choose and apply the right preprocessing techniques for each type of feature, ensuring the data is properly formatted and meaningful for the model.

By categorizing the features in this way, we can utilize tools like ColumnTransformer to systematically preprocess the data. Each group of features can have its corresponding technique applied in a structured and efficient manner, reducing errors and ensuring the dataset is model-ready. This approach not only simplifies the preprocessing pipeline but also improves the consistency and effectiveness of the machine learning model.



Figure 3.6 Classify features

1. Text-based features encoder

This code is part of the preprocessing pipeline for preparing data to train a machine learning model. It utilizes `ColumnTransformer` from scikit-learn to apply specific transformations to different types of features based on their characteristics, as categorized earlier.

The `ColumnTransformer` allows you to handle each feature group separately while keeping the process efficient and organized. Here's how it works:

1. **Binary Categorical Features**: These are features with only two possible values (e.g., Yes/No). Instead of using one-hot encoding, which could introduce multicollinearity, you’ve opted for OrdinalEncoder(). This approach reduces redundancy and keeps the feature representation.
2. **Multi-Categorical Features**: For features with multiple unique values (e.g., Ethnicity), you use *OneHotEncoder().* This method creates a binary column for each category, ensuring the model can understand the categorical nature of these features without assuming any implicit order.
3. **Remainder Handling**: The *remainder="passthrough"* argument ensures that all features not explicitly mentioned in the transformer list (likely continuous numerical features) remain unchanged and pass through the

By categorizing the features into binary categorical, multi-categorical, ordinal categorical, and so on, made it clear how each group should be processed. This classification was crucial for designing the pipeline because each feature type requires a different transformation technique. Without this organization, the preprocessing would be less systematic, leading to potential errors or inefficiencies.

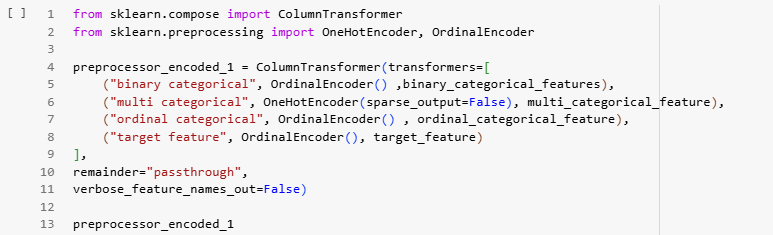


Figure 3.7 Source code text-based features encoding

1. Data splitting

This code splits the dataset into training and testing sets to prepare for model building and evaluation. It separates the target variable, **Diagnosis**, from the features, creating **X** for input variables and **y** for the output. Using train\_test\_split, 80% of the data is allocated for training (**X\_train**, **y\_train**), and 20% for testing (**X\_test**, **y\_test**). The *stratify=y* parameter ensures the class distribution in the target variable remains balanced across both sets, while *random\_state=16* makes the split reproducible. This step ensures a fair and reliable setup for training and testing the machine learning model.

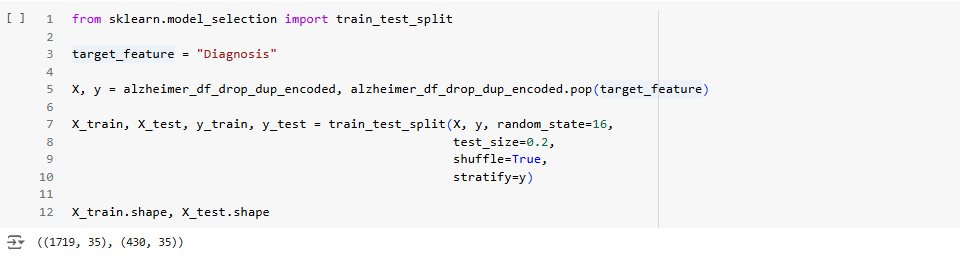


Figure 3.8 Source code splitting data (train/test set)

1. Impute missing values

This function, *report\_missing\_values()*, generates a summary report of missing values in the provided datasets. It calculates both the count and percentage of missing values for each column in the training dataset. If a test dataset is also provided, it includes the missing value count and percentage for the test dataset in the report. The results are presented in a DataFrame, sorted by the number of missing values in the training set.

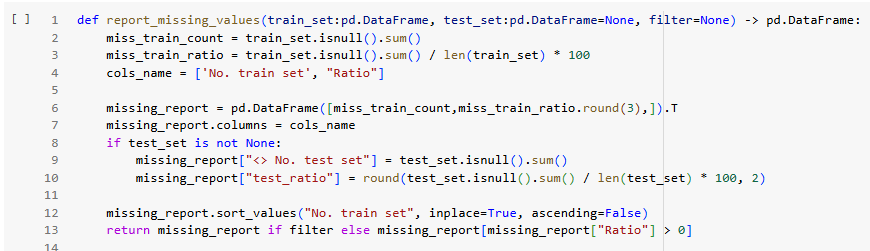


Figure 3.9 Source code reporting missing values

The number and percentage of missing values in the training set ("No. train set" and "Ratio") and the test set ("No. test set" and "test\_ratio"). The features are sorted by the number of missing values in the training set, with ***EducationLevel*** having the highest proportion of missing data (20.88%) in the training set. This report helps identify which features require data imputation or handling to improve the quality and consistency of the datasets.

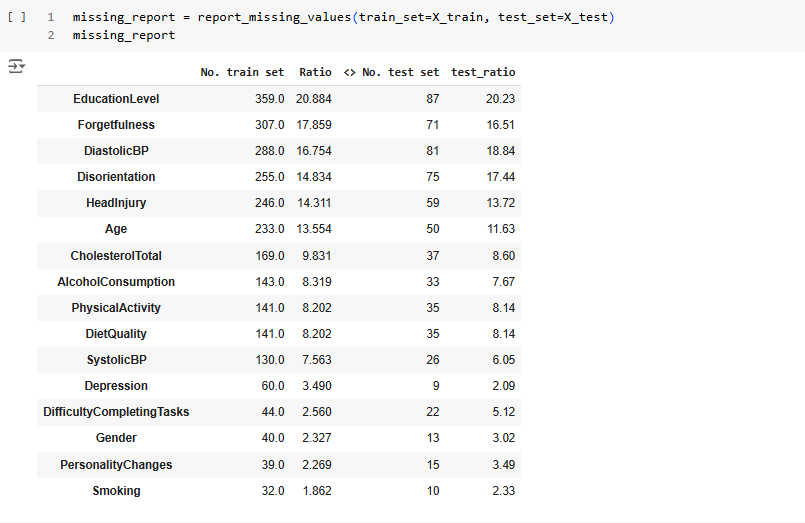


Figure 3.10 Statistic report missing value

The source code below take all responsibilities handling missing values in this data set. It’s using SimpleImputer to impute the missing values combining with columns transformer for organized and clean readable visual. The strategies I’m using for those missing values is *most\_frequent* all of those missing values’ll be imputed by the most\_frequent value in its own distribution.

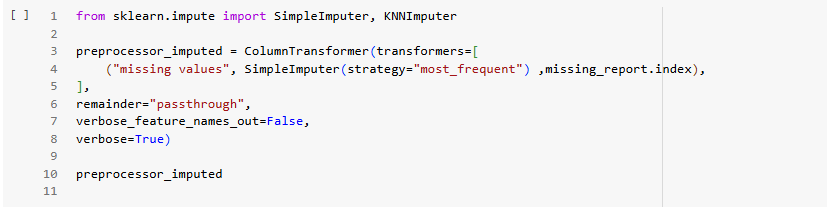
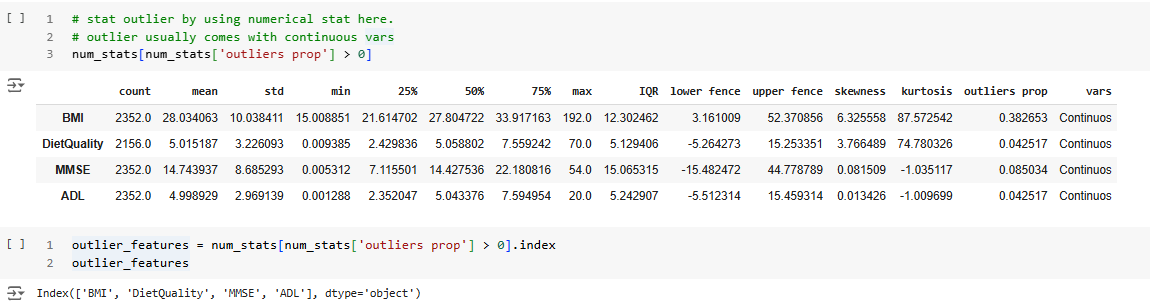


Figure 3.11 Souce code imputing missing values

1. Outlier, noise

The table summarizes outlier detection in four continuous variables: "BMI," "DietQuality," "MMSE," and "ADL," based on statistical metrics like mean, IQR, skewness, and kurtosis. The `outliers prop` column identifies the proportion of outliers for each variable, with all four features exhibiting outliers. These features are flagged for further analysis to address their potential impact on the dataset.



The dataset preprocessing applies the `RobustScaler` to mitigate outliers in selected continuous variables: "BMI," "DietQuality," "MMSE," and "ADL." A `ColumnTransformer` is used to scale these features while leaving other columns unchanged. The transformation is applied to both training and test datasets, producing a scaled training dataset with 1,719 rows and 35 columns. The scaled values address the influence of outliers while preserving the integrity of categorical and other variables for further analysis.

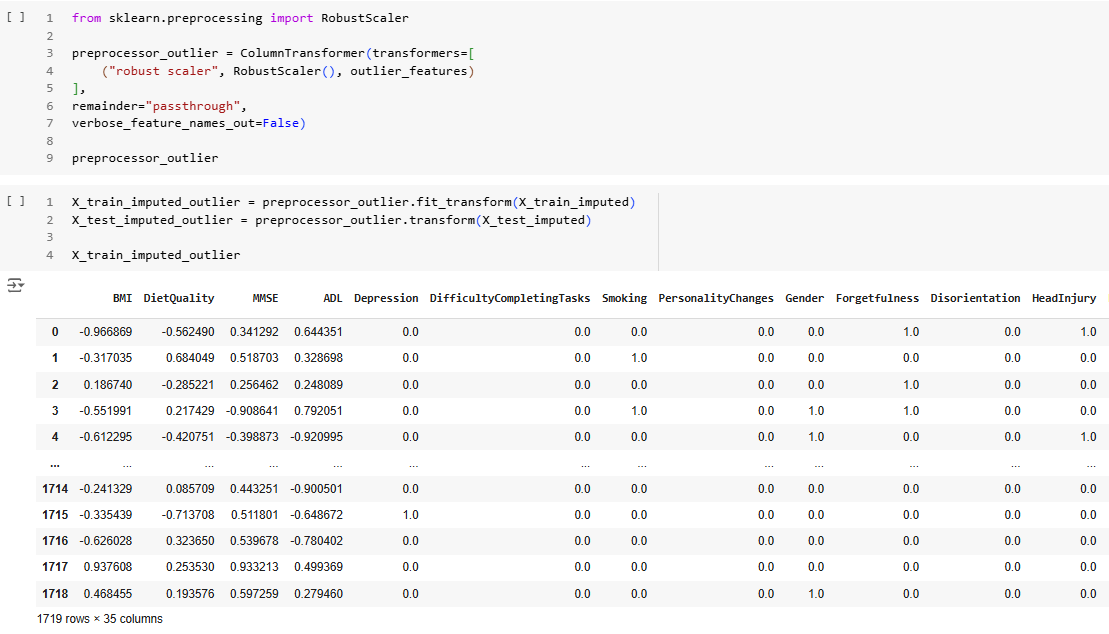


Figure 3.12 Handle outlier

1. Scaling features

This code defines a preprocessing pipeline that applies different transformations to numerical and categorical features in a dataset. The **numerical\_*transformer*** pipeline first standardizes continuous numerical features using *StandardScaler*, which centers the data around zero, and then scales them to a range between 0 and 1 using *MinMaxScaler*. The ***preprocessor\_rescaled*** uses a *ColumnTransformer* to apply the *MinMaxScaler* to the specified categorical features ***mul\_cat\_features*** and the ***numerical\_transformer*** to continuous numerical features ***con\_num\_features***. Any other columns not specified are left unchanged with *remainder="passthrough*. The *verbose=True* option ensures that the transformation steps are logged for transparency. This approach helps prepare the data by rescaling different types of features appropriately before model training.

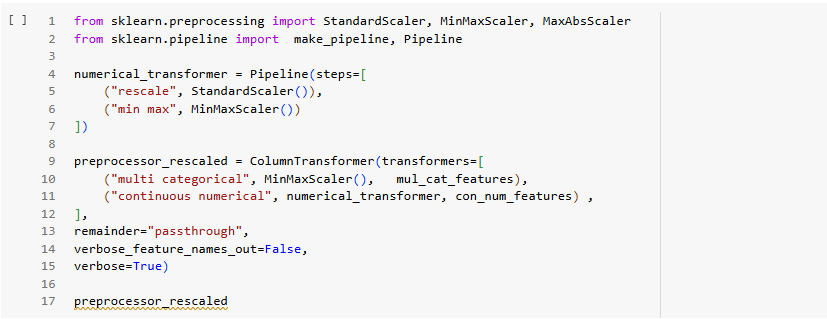


Figure 3.13 Source code scaling features

1. Imbalance class

This code snippet creates a pipeline using SMOTE (Synthetic Minority Over-sampling Technique) from the *imblearn* library to address class imbalance in the training data. The pipeline, *smote\_pipe*, applies the *SMOTE* technique to generate synthetic samples for the minority class in the dataset. The *fit\_resample* method is then used on the training data

(X\_train\_imputed\_rescaled and y\_train), creating a balanced dataset with more samples for the minority class. This technique is commonly used to improve the performance of machine learning models when the classes are imbalanced, helping the model generalize better.

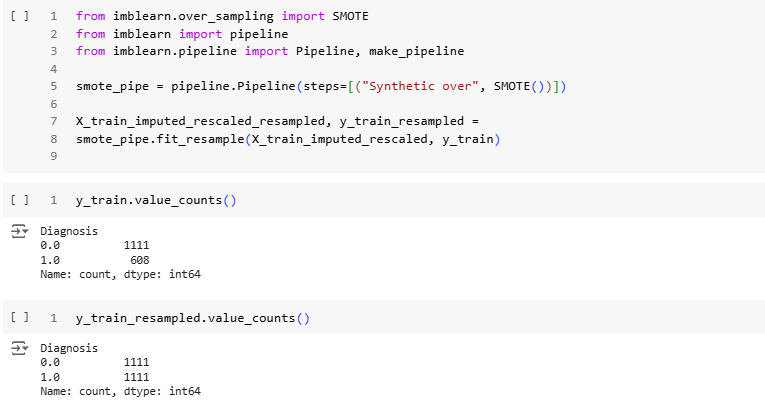


Figure 3.14 Source code handling class imbalance using SMOTE

## Modeling

1. Base models, ensemble models

In this picture below, as you can see I have a list of base-models and advanced models, in this case is ensemble models. I’ll try to experiment all of these base models and evaluated each model performance to find out which models best learn the pattern of dataset by the performance metrics such as accuracy, precision, recall, f1-score, confusion matrix, classification reports.

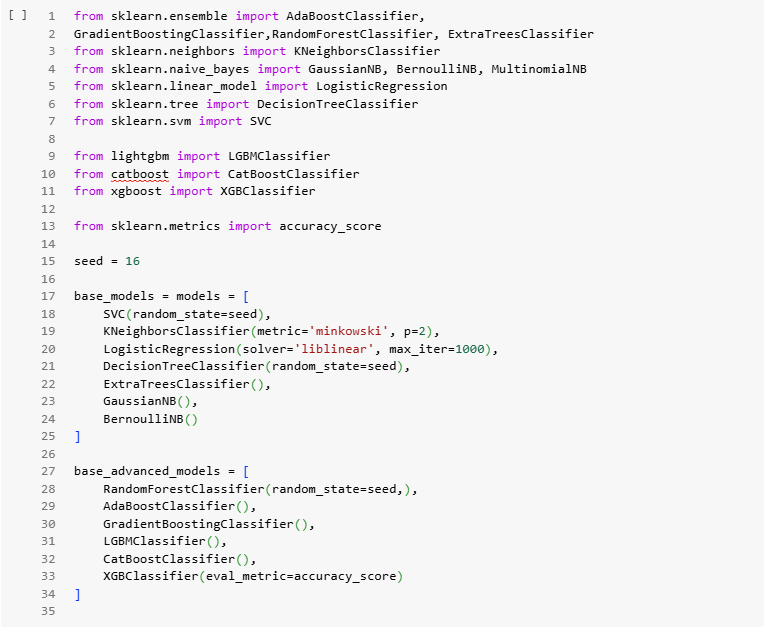


Figure 3.15 List of base-models would be experimented

These base models above was initialized with the default hyperparameters, I’m not trying to hyperparameter tuning at this step. The later step’ll take care of it.

1. Validating models

The function *validate\_models* will take the list of base models and the corresponding train, test set as parameters. The inner evaluating process followed by taking the train set and using it to train models, after training try some prediction using the test set and calculating the confusion matrix, classification report, iterate the process through each models and return the output as the dictionary of confusion matrix, clf report of each model

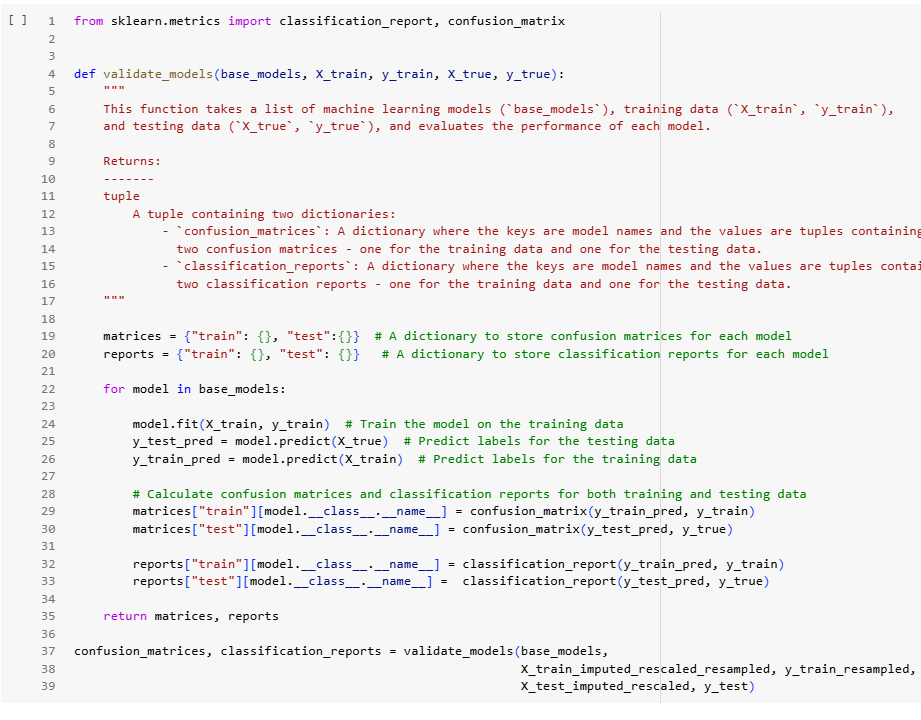


Figure 3.16 Source code evaluating models

1. Confusion matrices (train and test)

The confusion matrix is one of the essential metric to evaluate the classification’s model performances , especially when dealing with imbalanced datasets or multi-class classification problems. It provides a more detailed understanding of how well the model is performing, beyond just the accuracy metric. The confusion matrix helps you see the distribution of errors (false positives and false negatives) and gives insight into areas where the model is performing poorly, allowing you to refine your model

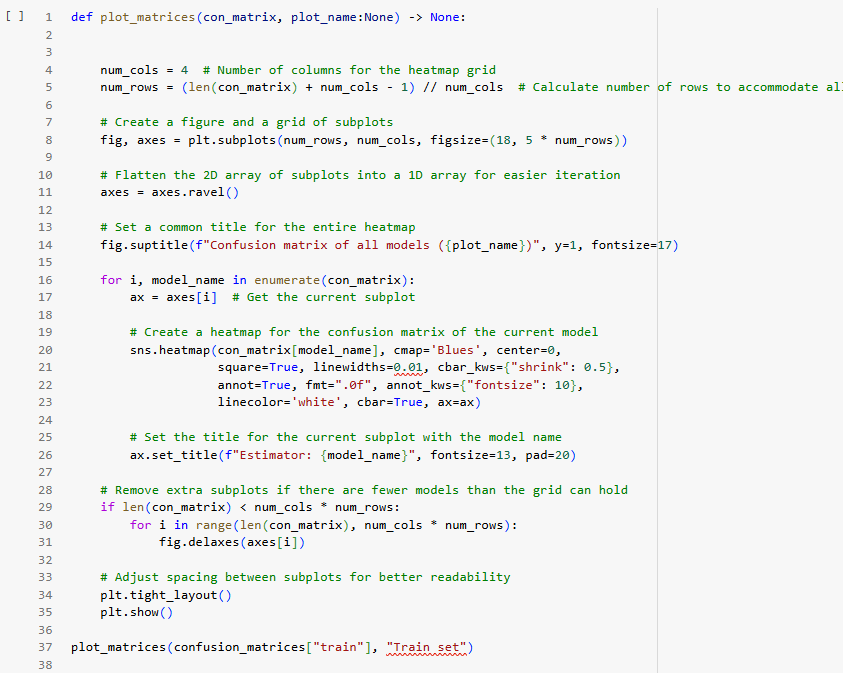


Figure 3.17 Source code ploting the confusion matrix for the training set

The output of the function above would be liked this.

You can adding new models if you want without hesitating the function will crash. These plot below was evaluating on traning set. With the visualizations below you can have a pretty clear overview of each model performances as well.

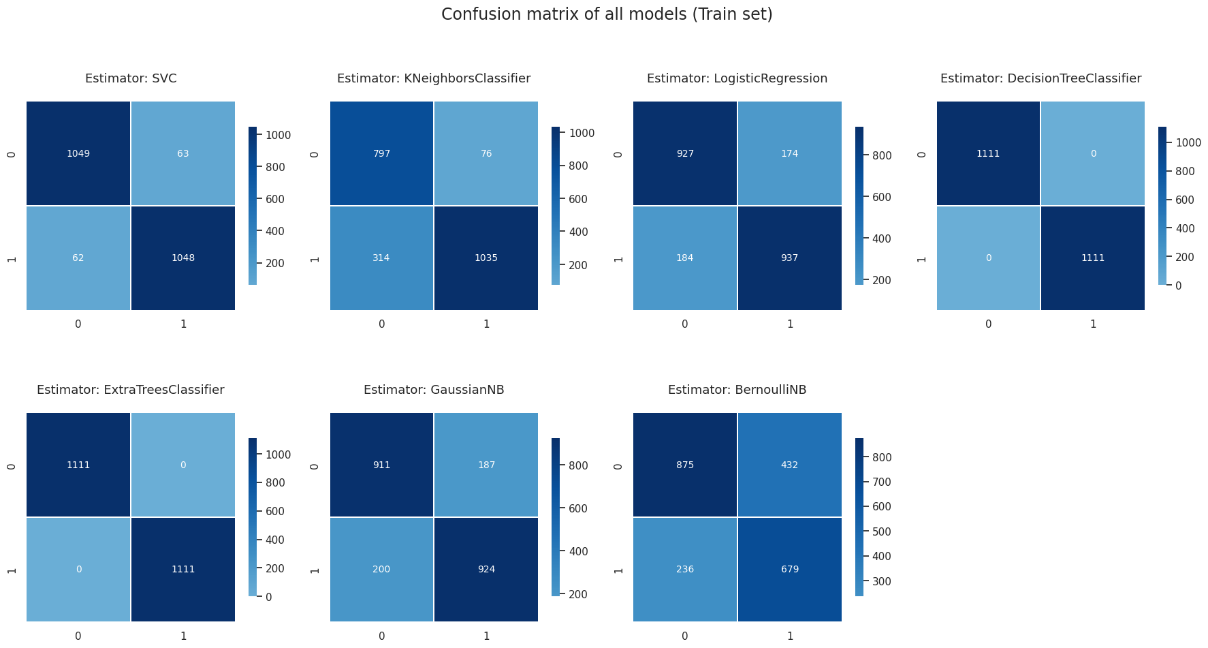


Figure 3.18 Confusion matrix of the base models on train set

The function below’ll let us see the base model performances on the test set.

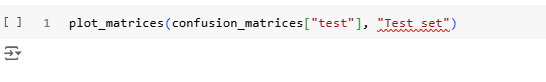


Figure 3.19 Source code ploting the confusion matrix for the test set

As you can see the model performaces on the test set are not good as training set. This can indicate that the model might be overfiting the training data so it’s not have good pattern to generalize the test set. That’s why we need classification report and hyperparameter tuning for better model performance.

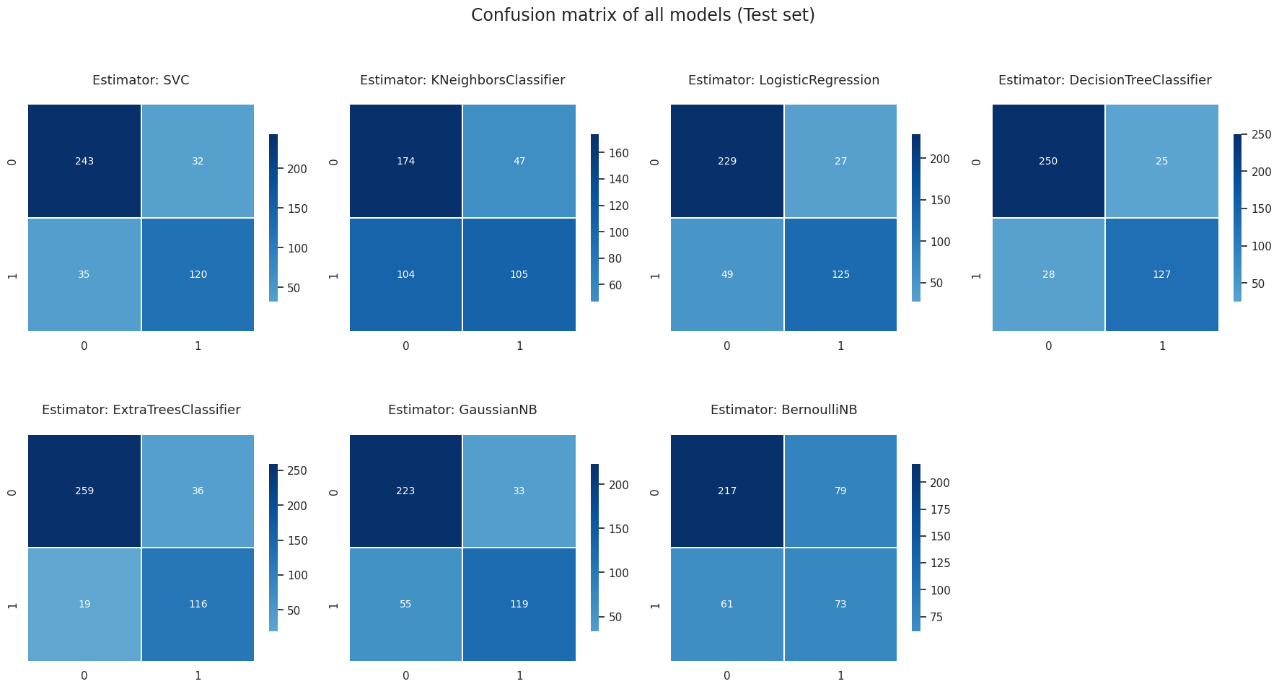


Figure 3.20 Confusion matrix of the base models on test set

1. Classification reports (train and test)

The function below merge all the classification reports of model above into a list of dictionary, and I’ll turn it into a Dataframe for more visual and readability.

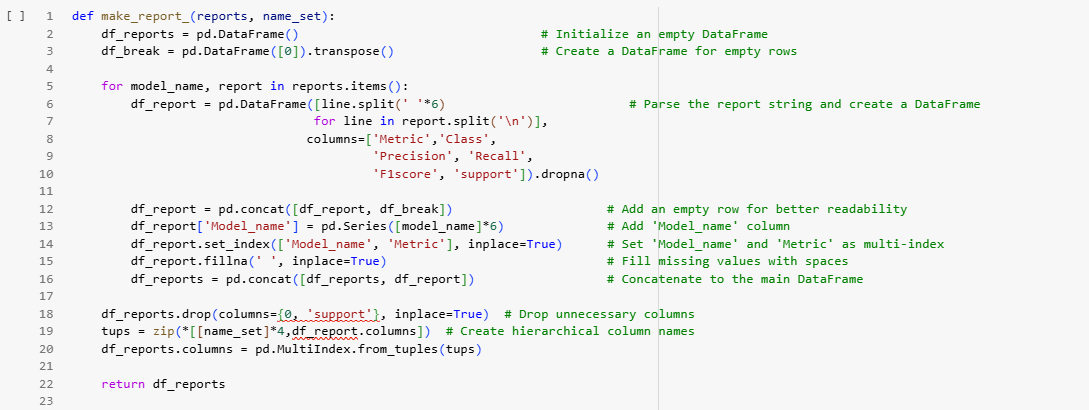


Figure 3.21 Source code turning classification reports into Dataframe

The output of the function above would be liked this. Base on this visualization you can absolutely do some comparations along with these model performances. You can easily interprete which model has the best accuracy, precision, recall, f1-score or which model has the highest recall to data. It’s essential to make a report like this.

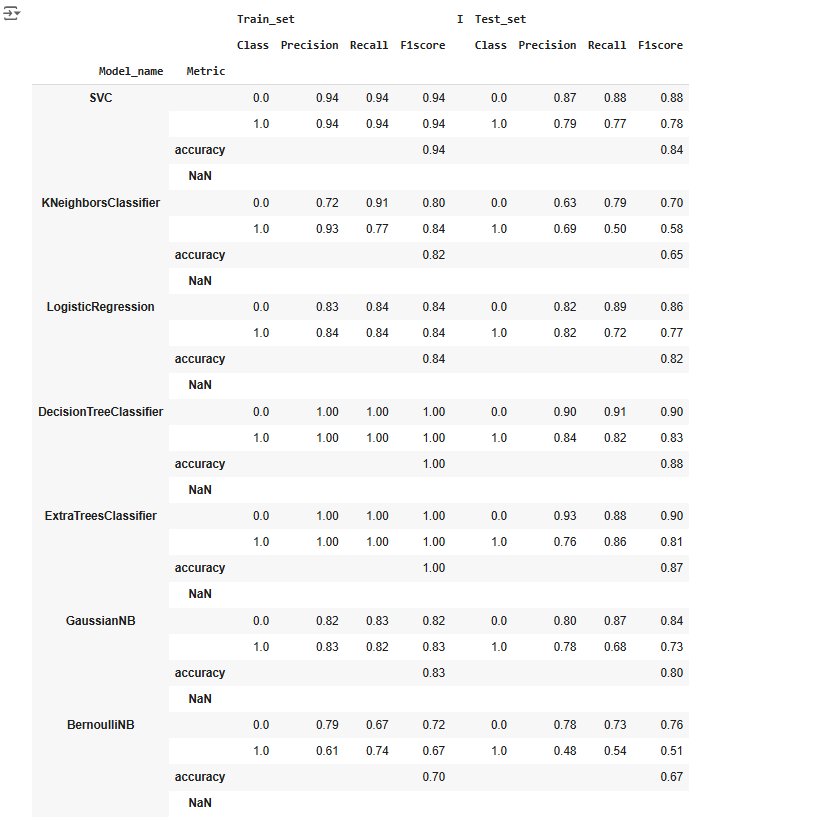


Figure 3.22 Classification report for all model in Dataframe form

1. Hyperparameter tuning.

The code defines a function, cross\_val\_df`, to perform cross-validation on a single machine learning model using sklearn.model\_selection.cross\_validate. It evaluates the model's performance on a dataset (X and y) with a 5-fold cross-validation strategy (cv=5). The function computes metrics including precision, recall, F1-score, and accuracy for both training and testing data, while also tracking fit and score times.

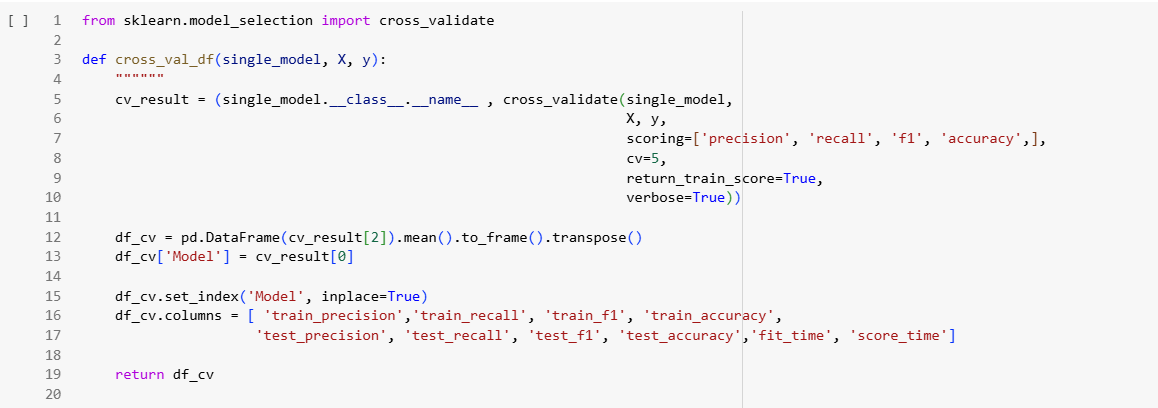


Figure 3.23 Cross validation for single model

**Logistic Regression**

The code performs hyperparameter tuning for a logistic regression model using GridSearchCV. A logistic regression model is initialized with a maximum of 1000 iterations (max\_iter=1000). A parameter grid (param\_grid) is defined to explore different values for the penalty (l1, l2, elasticnet) and solver (lbfgs, liblinear) options.

GridSearchCV is set up with the logistic regression model and the parameter grid, using 5-fold cross-validation (cv=5) and parallel processing (n\_jobs=-1). The process searches for the best combination of hyperparameters based on cross-validation performance. The verbose output is enabled (verbose=True) for detailed progress logs. The tuned model can be evaluated further using cross-validation or other techniques, as hinted by the commented-out line for cross-validation.`



Figure 3.24 Source code Logistic regression

The code performs cross-validation using the `GridSearchCV` object (`grid\_search`) on a resampled and preprocessed training dataset. Metrics evaluated include precision, recall, F1-score, and accuracy for both training and testing data, with results averaged across 5 folds (`cv=5`). The output is stored in a DataFrame, showing metrics like `fit\_time`, `score\_time`, `test\_precision`, `train\_precision`, `test\_recall`, `train\_recall`, `test\_f1`, `train\_f1`, `test\_accuracy`, and `train\_accuracy` for each hyperparameter combination.

The log indicates that 6 hyperparameter combinations were tested, resulting in 30 fits. The results highlight the performance of each combination across various metrics, providing insights into the model's behavior and enabling the selection of the best configuration for further use.

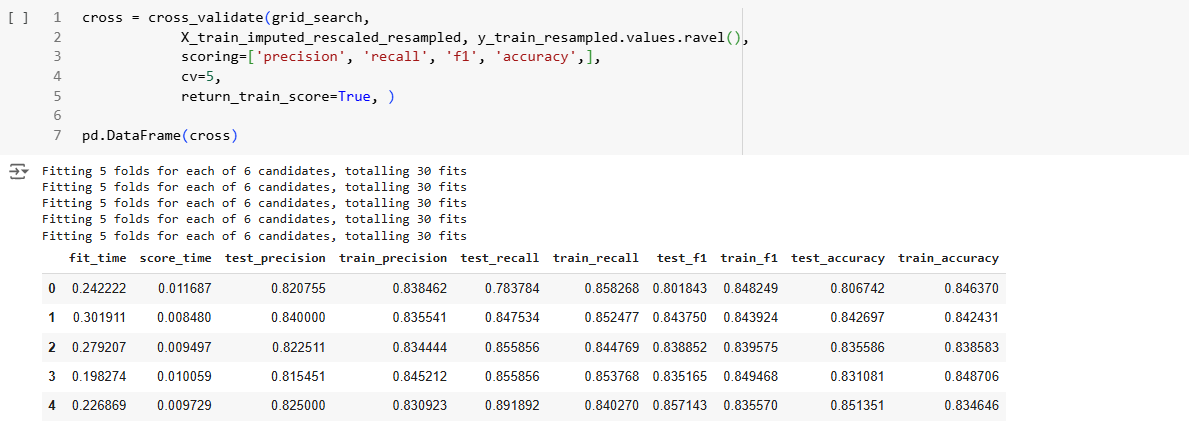


Figure 3.25 Cross validation result of Logistic Regression

# 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,. 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.

1. **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.

1. **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 |