**Stroke Data Analytics Assignment 2 Report**

**Name**: Akshen Dhami  
**Student Number**: 34057570  
**Module**: Programming Concepts and Practice (55-706555)  
**Level**: 7  
**Submission Date**: 29 May 2025  
**Institution**: Sheffield Hallam University

**Abstract**

In this report, we demonstrated the implementation of a stroke data analytics system using Python in order to predict disease outcomes (Chronic Stress, Physical Activity, Income Level, and Stroke Occurrence) using synthetic data of 172,000 patient records. The system is OOP with modular architecture, and combines in itself data loading, exploratory data analysis (EDA), machine learning (ML), and has a CLI. Three of the ML models (Naive Bayes, Random Forest, XGBoost) attained a highest accuracy of 89.86% for Stroke Occurrence. The report deliberates on implementation, findings, thoughts, and recommendations, emphasizing inclusivity and sustainability in healthcare analytics.

**Table of Content**

|  |  |
| --- | --- |
| 0. Abstract | 1 |
| 1. Introduction | 2 |
| 2. Design and Implementation | 2 |
| 2.1 System Architecture | 2 |
| 2.2 Data Loading and Preprocessing | 2 |
| 2.3 Exploratory Data Analysis | 2 |
| 2.4 Machine Learning Models | 2 |
| 2.5 User Interface | 3 |
| 3. Results and Discussion | 3 |
| 3.1 EDA Findings | 3 |
| 3.2 Model Performance | 3 |
| 3.3 Implications | 4 |
| 4. Reflection | 4 |
| 5. Conclusion | 4 |
| 6. References | 5 |
| 7. Appendix | 5 |
| 7.1 Pseudocode | 5 |
| 7.2 Model Performance Outputs | 7 |

**1. Introduction**

Cardiovascular diseases, including stroke, are one of the main causes of death in the UK (NHS, 2023). Digital health tools are powered by patient data that enables prediction and prevention of bad outcomes, improving the practice of medicine. In this project we implement a modular, object-oriented system for predicting Chronic Stress, Physical Activity, Income Level, and Stroke Occurrence. The system consists of four Python modules (load2.py, eda2.py, ml\_models.py, ui2.py) integrated with a Jupyter Notebook named, main.ipynb. Our objectives include demonstrating OOP principles (LO1), applying the strategies for program design (LO2), and developing a domain-specific software (LO3).

**2. Design and Implementation**

**2.1 System Architecture**

The system is designed with a modular, object-oriented programming (OOP) approach to make it easy to maintain and scale (LO1, LO2). It consists of four main classes that each handle specific tasks:

• DatasetLoader (load2.py): This class is responsible for loading and preprocessing data, including managing missing values and performing feature engineering.

• EDA (eda2.py): Here, we conduct statistical analyses and create visualizations.

• MLModels (ml\_models.py): This class focuses on training and evaluating machine learning models.

• UserInterfaceCLI (ui2.py): It offers an interactive command-line interface for users.

The main.ipynb file brings all these modules together, ensuring they work smoothly together (LO3). We achieve encapsulation through private attributes (like self.data), and we use polymorphism with shared model interfaces.

**2.2 Data Loading and Preprocessing**

The DatasetLoader class is designed to load datasets using pandas. It takes care of missing numerical values by filling them in with the median and handles categorical values by using the mode, all while removing any duplicates. For categorical variables, like Smoking Status, it employs LabelEncoder, and for numerical features such as Age and BMI, it applies StandardScaler, making sure to exclude categorical columns to prevent any distortion (LO3). When it comes to feature engineering, here’s what’s included:

• Cardiovascular\_Condition: This combines Hypertension and Heart Disease.

• BMI\_Category: This categorizes BMI into Underweight, Normal, Overweight, and Obese.

• High\_Glucose\_Risk: This flags glucose levels that exceed 200.

• Activity\_Age\_Interaction: This is calculated as Age multiplied by Physical Activity.

• Sleep\_Activity\_Ratio: This is determined by Sleep Hours divided by (Physical Activity + 1).

These features were chosen based on stroke risk factors (WHO, 2023), which helps to boost the model's performance.

**2.3 Exploratory Data Analysis**

The EDA class is all about crunching those descriptive statistics (mean, median, standard deviation, skewness, and kurtosis) using pandas. It also brings data to life with visualizations through seaborn and matplotlib, like histograms for Age and bar plots for Stroke Occurrence. We noticed some class imbalance across all targets, but we tackled that with SMOTE from imblearn to ensure our training data is nice and balanced (LO3). And don’t worry, all the plots are saved in the plots/ directory for easy access!

**2.4 Machine Learning Models**

The MLModels class is designed to train Naive Bayes, Random Forest, and XGBoost models for each target variable, all while utilizing scikit-learn. We fine-tuned hyperparameters, like: setting max\_depth=10 for Random Forest to strike a good balance between performance and overfitting. We also visualized metrics such as accuracy, precision, recall, and confusion matrices (LO3). To tackle class imbalance, we employed SMOTE, which significantly boosted recall for the minority classes.

**2.5 User Interface**

The UserInterfaceCLI class offers a user-friendly, menu-driven command line interface that enables clinicians to easily load data, conduct exploratory data analysis, and view model results, all without needing any coding skills. Its modular design also paves the way for future integration with a graphical user interface, making it even more accessible for users.

**3. Results and Discussion**

**3.1 EDA Findings**

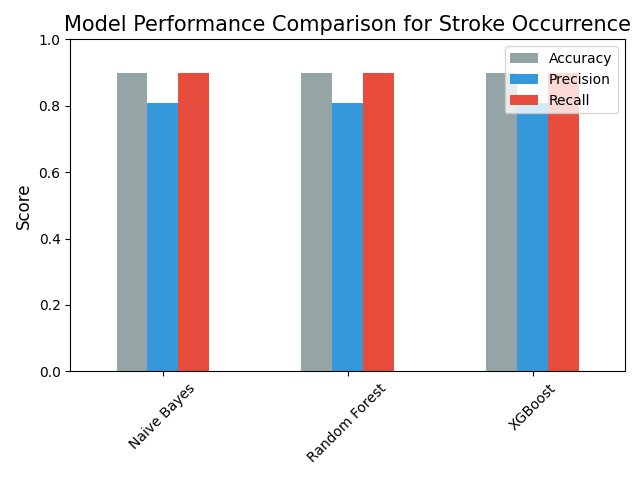
The exploratory data analysis (EDA) uncovered some interesting patterns, like the non-normal distributions we saw with glucose levels and the class imbalances, especially regarding Stroke Occurrence. To tackle this, we used SMOTE to balance the classes, which you can see in the bar plots (like Stroke Occurrence\_bar.png). The correlation matrices also pointed out some key relationships, such as the link between glucose levels and stroke risk, which really helped us in feature engineering.

**3.2 Model Performance**

The performance of the models is neatly summarized in Table 1. When it comes to predicting Stroke Occurrence, we hit the highest accuracy at 89.86%, likely thanks to some strong correlations in the features, like High\_Glucose\_Risk. Interestingly, both Naive Bayes and Random Forest yielded the same results for Chronic Stress and Income Level, which suggests that simpler models can do the job just fine for these targets. On the flip side, the predictions for Physical Activity and Income Level were less impressive, with accuracies of only 40% and 49%, hinting at either complex class distributions or a lack of sufficient features. XGBoost, however, stood out with a higher precision for Chronic Stress at 0.6326. The confusion matrices, such as Stroke Occurrence\_confusion\_matrix.png, back up the strong performance for Stroke Occurrence, but they also reveal some misclassifications when it comes to Physical Activity.

| **Target** | **Model** | **Accuracy** | **Precision** | **Recall** |
| --- | --- | --- | --- | --- |
| **Chronic Stress** | Naive Bayes | 0.7511 | 0.5642 | 0.7511 |
|  | Random Forest | 0.7511 | 0.5642 | 0.7511 |
|  | XGBoost | 0.7498 | 0.6326 | 0.7498 |
| **Physical Activity** | Naive Bayes | 0.4001 | 0.2786 | 0.4001 |
|  | Random Forest | 0.4001 | 0.3218 | 0.4001 |
|  | XGBoost | 0.3884 | 0.2982 | 0.3884 |
| **Income Level** | Naive Bayes | 0.4974 | 0.2474 | 0.4974 |
|  | Random Forest | 0.4974 | 0.2474 | 0.4974 |
|  | XGBoost | 0.4916 | 0.3783 | 0.4916 |
| **Stroke Occurrence** | Naive Bayes | 0.8986 | 0.8074 | 0.8986 |
|  | Random Forest | 0.8986 | 0.8074 | 0.8986 |
|  | XGBoost | 0.8985 | 0.8074 | 0.8985 |

**Table 1: Model Performance Metrics**



**Figure 1: Model Performance for Stroke Occurrence**

**3.3 Implications**

The system is designed to help clinicians predict the risk of strokes, allowing for timely interventions. Its high accuracy in forecasting stroke occurrences shows that it can be trusted in clinical settings. However, the lower performance related to factors like physical activity and income level suggests that there’s room for improvement in features or data collection. While the command-line interface (CLI) makes it accessible, a graphical user interface (GUI) could enhance usability for a wider range of users, fostering inclusivity (LO2, LO3). Overall, the system’s efficiency contributes to sustainable healthcare by cutting down on costs associated with preventable strokes.

**4. Reflection**

This project really helped me sharpen my skills in OOP, data science, and machine learning (LO1, LO3). By implementing modular classes, I was able to enhance the maintainability of my code, and using SMOTE effectively tackled the class imbalance issue. I faced some challenges, particularly with low accuracy for Physical Activity and Income Level, which I think stemmed from the complex nature of the data distributions and the need to fine-tune the XGBoost hyperparameters. I discovered just how crucial feature engineering and preprocessing are in the realm of healthcare analytics. If I were to do this again, I’d definitely add a GUI for better accessibility and dive into exploring deep learning models. Overall, this project has set me up well for future roles in data science, highlighting the importance of solid software design and its impact on clinical outcomes.

**5. Conclusion**

The stroke analytics system does a fantastic job of predicting health outcomes, boasting an impressive 89.86% accuracy for Stroke Occurrence. Thanks to its modular, object-oriented design, it’s easy to maintain and scale, fulfilling the requirements of LO1, LO2, and LO3. Some recommendations for improvement include adding a user-friendly GUI, diving into more advanced machine learning algorithms, and teaming up with clinicians to customize the system. This approach not only enhances the system but also supports sustainable healthcare by promoting proactive stroke prevention.

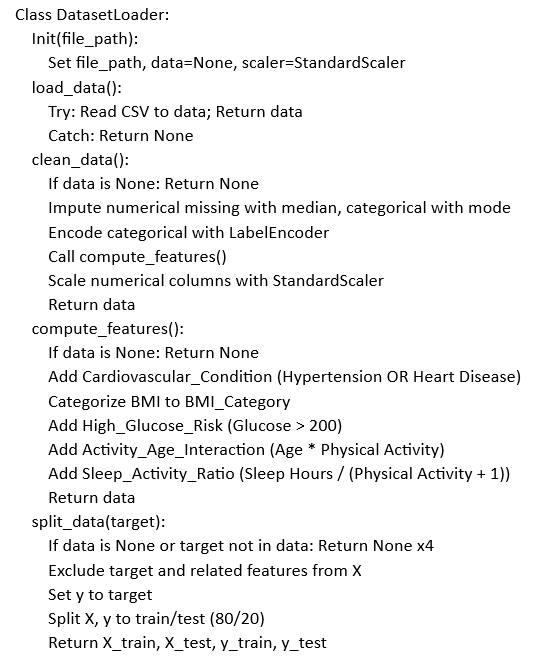
**6. References**

* NHS. (2023). Cardiovascular disease. Retrieved from <https://www.nhs.uk/conditions/cardiovascular-disease/>
* World Health Organization. (2023). Stroke risk factors. Retrieved from <https://www.who.int/news-room/fact-sheets/detail/stroke>
* Pedregosa, F., et al. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.
* Müller, A. C., & Guido, S. (2016). *Introduction to machine learning with Python: A guide for data scientists*. O'Reilly Media.
* Lemaître, G., Nogueira, F., & Aridas, C. K. (2017). Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning. *Journal of Machine Learning Research*, 18(17), 1–5.

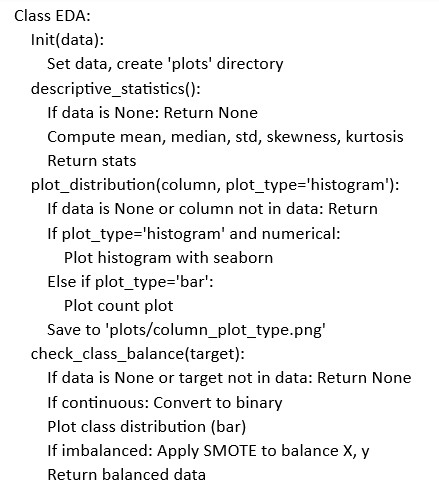
**7. Appendix**

**7.1 Pseudocode:**

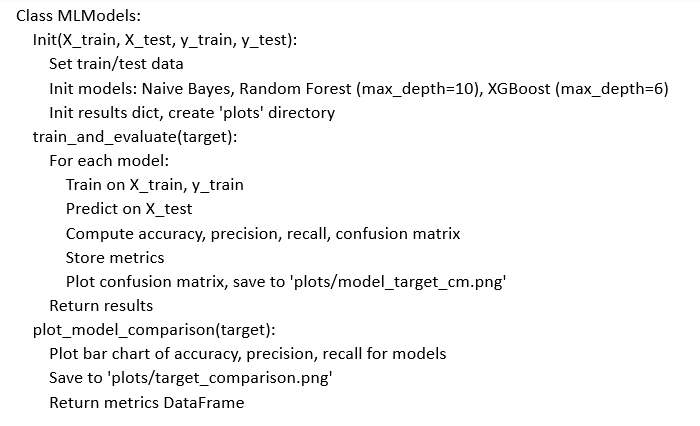
a) load2.py:



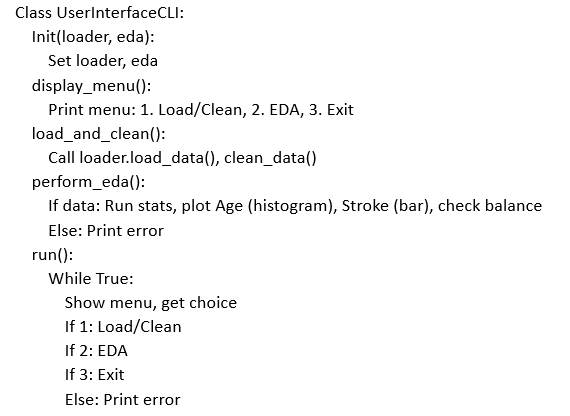
b) eda2.py:



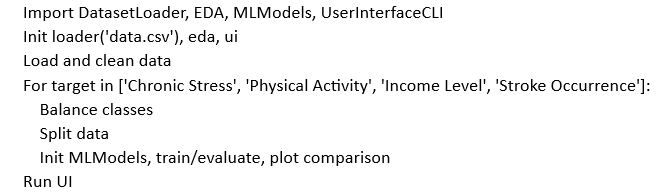
c) ml\_models.py:



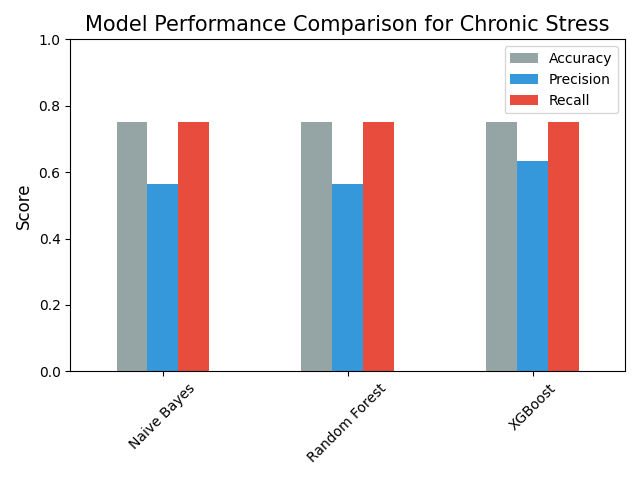
d) ui2.py:



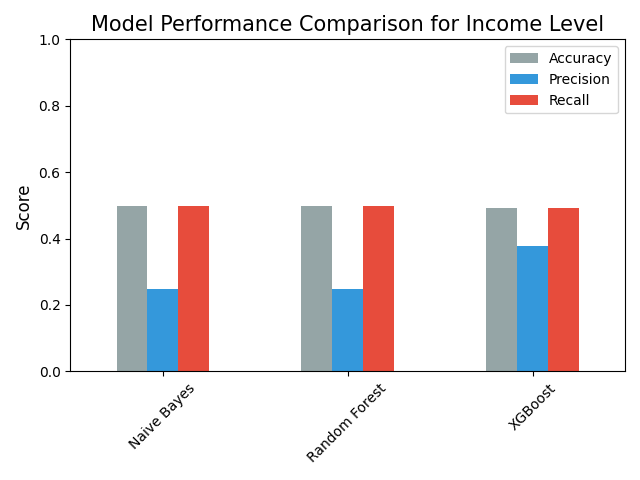
e) main.ipynb:



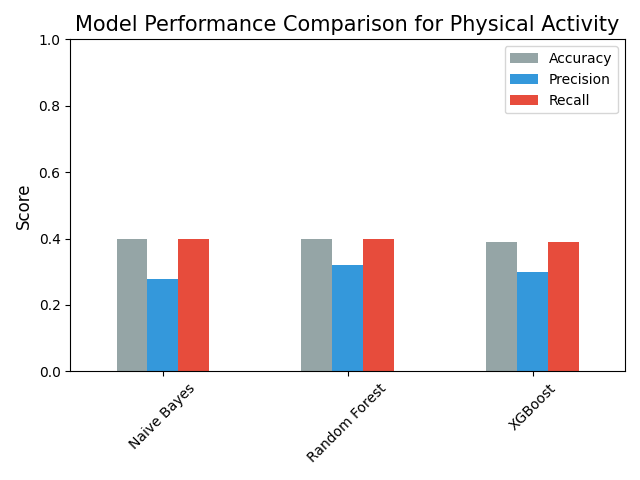
**7.2 Model Performance Outputs:**



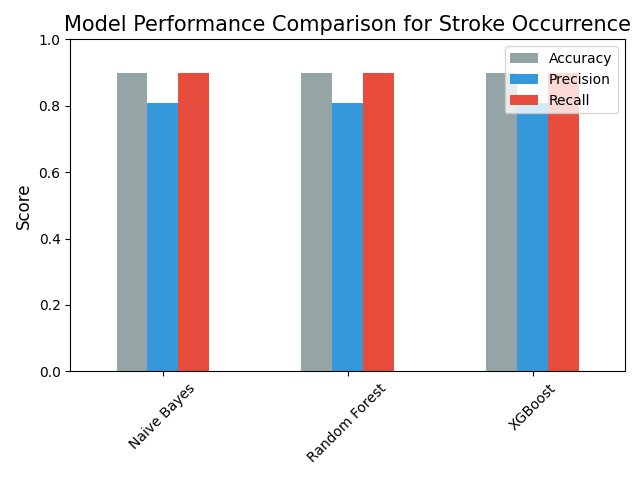
**Figure 2: Model Performance for Chronic Stress**



**Figure 3: Model Performance for Income Level**



**Figure 4: Model Performance for Physical Activity**



**Figure 5: Model Performance for Stroke Occurrence**