World Journal of Advanced Research and Reviews
eISSN: 2581-9615 CODEN (USA): WJARAI
Cross Ref DOI: 10.30574/wjarr
Journal homepage: https://wjarr.com/

(RESEARCH ARTICLE)

Check for updates

Exploring machine learning algorithms to predict health risks and outcomes

Paulami Bandyopadhyay *

Senior Data Engineer

World Journal of Advanced Research and Reviews, 2020, 07(03), 313–327

Publication history: Received on 07 September 2020; revised on 20 September 2020; accepted on 24 September 2020

Article DOI: https://doi.org/10.30574/wjarr.2020.7.3.0341

Abstract

This study investigates the pertinence of machine learning techniques on various datasets and how we can leverage it in prediction of health risks. I investigated how well two algorithms—Logistic Regression and Multi-Layered Perceptron (MLP)—predict health outcomes and risk. To be more precise, I evaluated the model's capacity to recognize stroke risk using a dataset of stroke predictions. By means of comparison analysis, this study seeks to clarify the advantages and disadvantages of each algorithm when used with these disparate data kinds, providing information about how well-suited they are for different prediction tasks. Additionally, I provided a framework for data analysis that outlines crucial procedures for data preparation, cleaning, and exploration. This framework may be used to improve the efficacy of machine learning models on a variety of datasets.

Keywords: Machine Learning Heterogeneous Data; Comparative Analysis; Prediction Modeling; Data Analysis Techniques; Stroke Prediction; Logistic Regression; Multi-Layered Perceptron; Data Preprocessing

1. Introduction

1.1. Motivation: The Power and Nuance of Machine Learning Data

In many fields, machine learning (ML) is becoming a key component of advancement. Its capacity to glean insightful information from large, intricate datasets has propelled advances in the social sciences, healthcare, and finance. Nevertheless, there is no one-size-fits-all approach to ML model efficacy. Selecting the best machine learning algorithms requires an awareness of the subtle differences among the various types of data. Data can be numerical or categorical, structured (tables) or unstructured (text, pictures), and feature connections can be either linear or non-linear. These considerations play a major role in selecting the best algorithm. This study explores the performance of two different algorithms on disparate datasets in order to delve into this important area of machine learning application.

1.2. Research Focus: Delving into Stroke Prediction

The utilization of machine learning techniques for stroke prediction is the main emphasis of this work. Stroke is a major global cause of mortality and disability that has a substantial impact on public health. In order to enable early intervention and preventive measures, stroke prediction models seek to identify those who are at a high risk of having a stroke. Usually, these models look at things like smoking history, blood pressure, cholesterol, and age. This study attempts to provide a more comprehensive knowledge of how machine learning algorithms function on various data types with differing underlying structures and complexities by examining these discrete datasets.

1.3. Methodology: Unveiling the Algorithms - Logistic Regression and Multi-Layered Perceptron

This section explores the two main approaches used in this study: Multi-Layered Perceptron (MLP) and Logistic Regression. Both approaches are categorized under the general heading of supervised learning, in which a model learns

* Corresponding author: Paulami Bandyopadhyay

from labeled data to predict outcomes for examples that are not visible. Here, I explained each technique's fundamental ideas and features.

A fundamental algorithm for classification tasks is logistic regression. It creates a mathematical model that associates a probability of a particular outcome (like the occurrence of a stroke) with input features (like age and blood pressure). In essence, the model separates observations with high and low probabilities of the desired outcome by learning a decision boundary. Because of this methodology, logistic regression is ideally suited for evaluating datasets such as the stroke prediction dataset, which aims to classify people according to their risk level.

However, the Multi-Layered Perceptron (MLP), a kind of artificial neural network, is a more intricate design. It mimics the structure of the human brain by being made up of interconnected layers of artificial neurons. Activation functions are used by each layer to transform the received data, which eventually results in an output prediction. The power of MLP resides in its capacity to discover intricate, non-linear relationships in data. This makes it an effective tool for solving complex prediction problems, possibly surpassing logistic regression in situations where a linear model finds it difficult to capture the underlying relationships.

1.4. Research Objectives: Evaluating Algorithms, Unveiling Strengths and Weaknesses

This study compares the effectiveness of MLP and logistic regression on stroke prediction tasks in order to accomplish the following main goals:

Assess the Algorithms' Applicability to Various Data Types: This entails evaluating how well each algorithm captures the fundamental connections found in the datasets used to predict strokes. In order to provide insights into the algorithms' suitability for various data types, I will ascertain which algorithm performs better on each dataset.

Learn About Algorithmic Strengths and Weaknesses: By comparing the performance, I hope to show the situations in which each algorithm performs well and pinpoint areas in which one might perform better than the other. Researchers and practitioners will be able to choose the best algorithm for their particular prediction tasks with the help of this helpful guidance.

Showcase the Best Data Analysis Techniques for Machine Learning Applications: To create strong machine learning models, data analysis must be done well. In order to improve model performance on a variety of datasets, this study will highlight crucial procedures for data preparation, cleaning, and exploration. To enhance the model's capacity to learn from the data, these procedures could involve handling missing values, locating outliers, and feature engineering—the process of developing new features from preexisting data.

1.5. Expected Contribution: Advancing the Application of ML on Heterogeneous Data

The goal of this investigation is to add significant knowledge to the field of machine learning, specifically the use of ML on diverse datasets. The results can help practitioners and researchers choose the best algorithms for their particular data types and prediction tasks. Additionally, by illuminating optimal methodologies for data analysis, this study can aid in the advancement of more resilient and dependable machine learning models in a variety of application areas. Advances in fields such as healthcare (better stroke prediction for preventive measures) may result from this. The ultimate goal of the research is to support the ethical and efficient application of ML to address challenging issues in a variety of domains.

2. Exploratory Data Analysis

2.1. Datasets attributes description

Understanding the data that will be used for training is the first and most important step in creating any machine learning algorithm. This understanding is attained by a thorough examination of the features of the datasets. Accordingly, the subsequent subsections will explore the particular characteristics of the dataset used in this research: stroke prediction.

A detailed description of each stroke prediction attribute is provided in Table 1.

Table 1 Stroke Prediction Attributes

List of all attributes in the Stroke Prediction dataset			
Attribute name	Type	Details	Possible Values
mean_blood_sugar_level	numeric	The average value of blood glucose throughout the duration of observation of the subject	
cardiovascular issues	categorical	Whether or not the subject has a medical history of cardiovascular	0,1
job_category	categorical	The field in which the person walks	child, entrepreneuurial, N_work_history, private_sector, public_sector
body_mass_indicator	numeric	Cbody mass index, which indicates if the person is underweigth, within normal limits, overweight or obese	
Sex	categorical	the gender of the person	F, M
tobacoo_usage	categorical	current or past smoker indicator	ex-smoker, smoker, non-smoker
high_blood_pressure	categorical	Binary attribute indicating whether the person suffer from high blood pressure or not	0,1
married	categorical	Binary attribute indicating whether the person is married or not	Y,N
living_area	categorical	The type of area where he lived most of his lives	city, countryside
years_old	numeric	The age of the person	
chaotic_sleep	categorical	Binary attribute for sleep program irregularity	0,1
analysis_results	numeric	The results of medical analysis of the person which may include various measurements and indicators relevant to her health	
biological_age_index	numeric	An index that estimates the biological age of a person based on different factors such as lifestyle, health status, measured in an unknown unit	
cerebrovascular_accident	categorical	Binary attribute indicating whether the person had an earlier stroke or not	0,1

2.2. Exploration of Attribute Types and Value Ranges

Finding the kinds of attributes (features) that are present and the ranges of values that correspond to them is an essential step before applying a machine learning model to a dataset. For the purpose of choosing suitable algorithms and guaranteeing ideal model performance, this analysis is crucial. Three main attribute types will be discussed in the paragraphs that follow.

- Continuous Numeric Attributes: These characteristics have numerical values that, in theory, can be any value within a given range. Age, weight, temperature, and so forth are a few examples.
- Discrete Nominal Attributes: Categorical data with discrete, non-ordered values is represented by these attributes. The days of the week (Monday, Tuesday, etc.) and disease types (cancer, diabetes, etc.) are two examples.
- Ordinal Attributes: These characteristics show categorical data with values that are inherently ordered. On the other hand, it might not be possible to interpret the difference between successive values in terms of a consistent unit. Movie ratings (G, PG, PG-13, etc.) and customer satisfaction ratings (one, two, etc.) are two examples. The relative order that a numerical value represents in ordinal attributes may be more significant than the value itself.

I can determine the datasets' Continuous Numerical Attributes and Discrete Nominal Attributes by using the analysis attributes.py script. Statistics that can be displayed in Tables 2 for numerical attributes and Table 3 for discrete attributes will be produced by the script.

Furthermore, there are 5110 items in the Stroke Prediction dataset overall.

Table 2 Continuous Numeric Attributes in Stroke Prediction Dataset

List of all Continuous Numeric Attributes in the Stroke Prediction dataset					
	mean_blood_sugar_level	body_mass_indicator	years_old	analysis_results	biological_age index
count	5110.000000	4909.000000	5110.000000	4599.000000	5110.000000
mean	106.147677	28.893237	46.568665	323.523446	134.784256
std	45.283560	7.854067	26.593912	101.577442	50.399352
min	55.120000	10.300000	0.080000	104.829714	-15.109456
25%	77.245000	23.500000	26.000000	254.646209	96.710581
50%	91.885000	28.100000	47.000000	301.031628	136.374631
75%	114.090000	33.100000	63.750000	362.822769	172.507322
max	271.740000	97.600000	134.000000	756.807975	266.986321

An initial inspection of the data reveals that there are missing attributes in Stroke Prediction datasets. In the Stroke Prediction dataset two attributes are missing: 'body mass indicator' and 'analysis results'.

To better understand the distribution of the continuous numeric attributes within the datasets, boxplots have been generated for each attribute. These visualizations are located in the 'plots' folder at the root of the project directory. The name of each boxplot starts with 'box plot '.

Boxplots are a standardized method for visually representing the distribution of data. They provide insights into several key characteristics of the data, including the median, quartiles, and outliers.

In Figure 1, I can see a boxplot for the body mass indicator attribute in the Stroke Prediction dataset. As described above, the boxplot provides a visual representation of the data's distribution, highlighting key statistical measures such as the median, quartiles, and potential outliers. This information is also presented in Table 3. One of the main insights that can be derived from the boxplot is the presence of outliers, which are data points that lie significantly outside the range of the rest of the data. Outliers can have a significant impact on the performance of machine learning models, and identifying and handling them appropriately is an essential step in the data preprocessing process

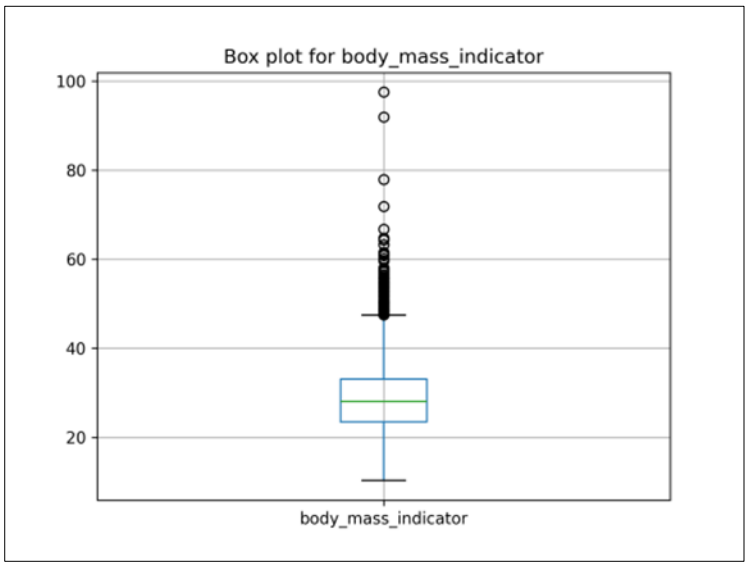

Figure 1 Boxplot for the body mass indicator attribute in the Stroke Prediction dataset

From the Discreet Nominal Attributes Table 3, I can see that each dataset contains only one attribute with missing values. In the Stroke Prediction dataset, the married attribute is missing. Also, the number of unique values for each attribute describes the diversity of the data.

Table 3 Discrete Nominal Attributes in Stroke Prediction Dataset

List of all Discrete Nominal Attributes in the Stroke Prediction dataset		
	Non-missing count	Unique values count
cardiovascular_issues	5110	2
job_category	5110	5
sex	5110	2
tobacco_usage	5110	4
high_blood_pressure	5110	2
married	4599	2
living_area	5110	2
chaotic_sleep	5110	2
cerebrovascular_accident	5110	2

In the histograms for the discrete nominal attributes, I can see the distribution of the unique values for each attribute. These visualizations can provide insights into the frequency of each category within the dataset, which can be useful for understanding the data's composition and identifying potential imbalances or biases. The histograms for the discrete nominal attributes are located in the 'plots' folder at the root of the project directory. The name of each histogram starts with 'histogram '.

In Figure 2, we can see a histogram of the tobacco usage attribute in the Stroke Prediction dataset,. The histogram shows that the majority of individuals are non-smokers, with a significant portion having undefined tobacco usage status. This imbalance and the presence of missing data need to be addressed appropriately.

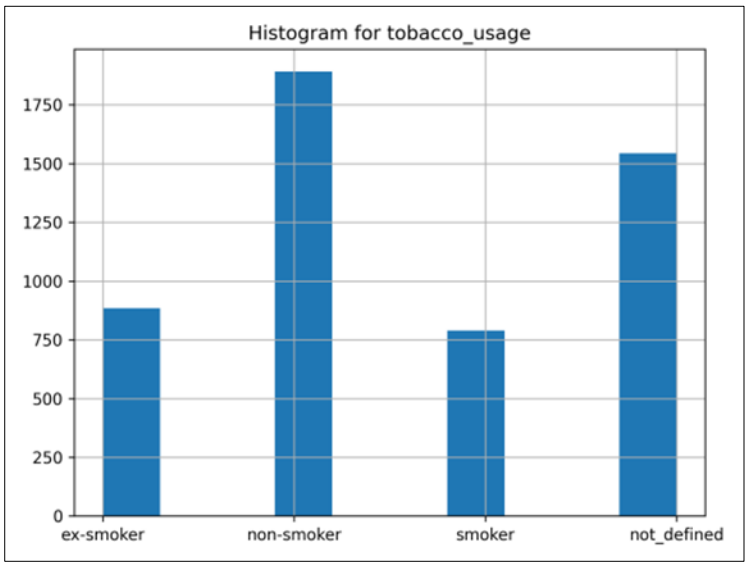

Figure 2 Histogram for the tobacco usage attribute in the Stroke Prediction dataset

2.3. Investigation of Class Distribution

In machine learning, it is common practice to split a dataset into two distinct subsets: a training set and a test set. This division is crucial for ensuring robust-ness and generalizability of the models developed using the data.

- Training Set: The primary purpose of the training set is to train the machine learning model. The model learns from patterns and relationships within the data to develop a predictive capability.
- Test Set: The test set, unseen by the model during training, serves to evaluate the model's generalizability. By applying the trained model to the test set, we can assess its performance on new, unseen data. This helps prevent overfitting, where the model performs well on the training data but fails to generalize to real-world scenarios.

Looking at how data is distributed is key. Imbalanced data, where some classes have far more examples than others, throws off classification tasks: high accuracy can hide poor performance on rare classes; models struggle to learn patterns from underrepresented classes; inaccurate predictions, especially for the minority class.

By checking the distribution, we can address imbalance:

- Balance the data: Oversample rare examples or under sample common ones.
- Cost-sensitive learning: Penalize the model more for mistakes on rare classes.
- Better metrics: Use precision, recall, and F1-score to get a clearer picture.

In Figure 3, we can see the distribution of each class in the datasets. The class distributions provide insights into the balance of the data and can help guide the selection of appropriate strategies for handling imbalanced classes. For example, in the Stroke Prediction dataset, the cerebrovascular accident class is highly imbalanced, with a significantly higher number of negative instances compared to positive instances. This imbalance can impact the model's ability to learn patterns from the minority class and may require resampling techniques or cost-sensitive learning to address.

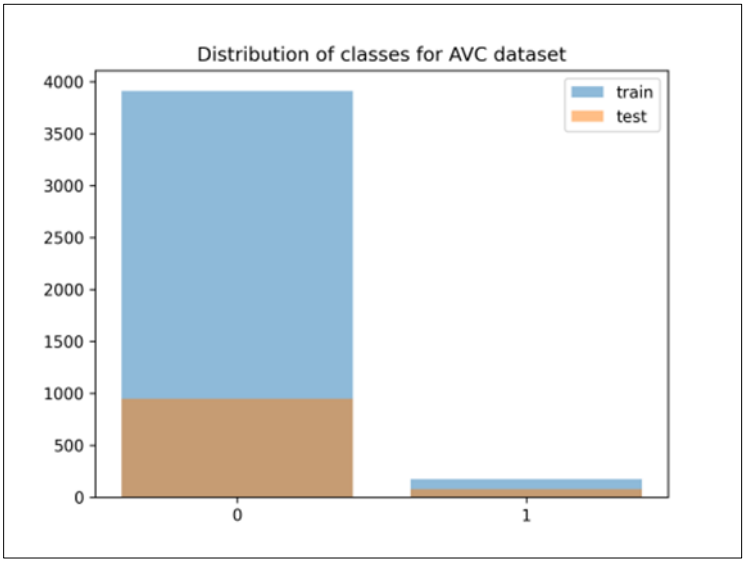

Figure 3 Distribution of the cerebrovascular accident class in the Stroke Prediction dataset

2.4. Analysis of Feature Correlations

Feature correlation analysis is a critical step in understanding the relationships between different attributes in a dataset. By examining how attributes are related to each other, we can identify patterns, dependencies, and redundancies that can inform feature selection, model building, and interpretation.

Correlation analysis typically involves calculating correlation coefficients be-tween pairs of attributes. The correlation coefficient quantifies the strength and direction of the linear relationship between two variables. A correlation coefficient close to 1 indicates a strong positive relationship, while a value close to -1 indicates a strong negative relationship. A correlation coefficient near 0 suggests no linear relationship between the variables.

In the 'correlation analysis.py' script, we calculate the correlation coefficients between all pairs of continuous numeric attributes in the datasets, generating a correlation matrix for each dataset. Moreover, we calculate the Cram´er's V coefficient for all pairs of discrete nominal attributes in the datasets, generating a Cram´er's V matrix for each dataset to measure the association between categorical variables. In Figures 4, we can see the correlation matrix for the Stroke Prediction datasets, respectively, for the continuous numeric attributes. In Figure 5, we can see the Cram´er's V matrix for the discrete nominal attributes.

The correlation matrix and Cram´er's V matrix provide valuable insights into the relationships between attributes in the datasets. By examining these matrices, In Figure 4, we can see that the mean blood sugar level attribute is highly correlated with the analysis results attribute, while the body mass indicator at-tribute is negatively correlated with the analysis results attribute.

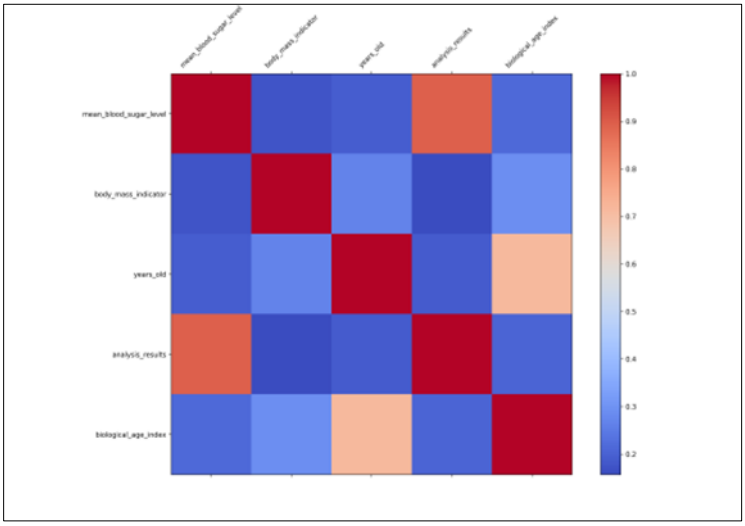

Figure 4 Correlation matrix for the Stroke Prediction dataset

The Cram´er's V matrix in Figure 5 provides insights into the association between discrete nominal attributes. In the Stroke Prediction dataset, the cardiovascular issues attribute is strongly associated with the chaotic sleep attribute.

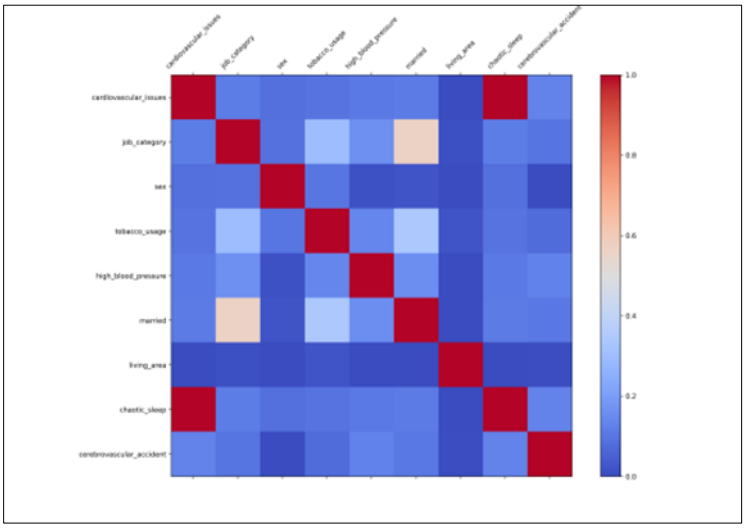

Figure 5 Cram´er's V matrix for the Stroke Prediction dataset

3. Data Preprocessing

As highlighted in the previous section, high-quality data is the cornerstone of effective machine learning models. However, real-world datasets often exhibit various imperfections that can impede model performance. Our exploration of the datasets revealed the presence of several such issues, including:

- Missing values for specific attributes.
- Extreme values (outliers) within certain attributes. – Redundant attributes with high correlation.
- Inconsistent value ranges for numeric attributes.

These imperfections necessitate data preprocessing, a crucial step aimed at transforming the raw data into a clean and consistent format. This section delves into the specific data preprocessing techniques employed in this study. By addressing these issues, we aim to optimize the data for subsequent machine learning algorithms, ultimately enhancing their effectiveness in extracting valuable insights.

As a note, all the scripts for data preprocessing are located in the 'preprocessing' folder at the root of the project directory.

3.1. Handling Missing Values

Missing data, a common issue in real-world datasets, necessitates the application of imputation procedures to address these missing values. Imputation techniques can be categorized as either univariate or multivariate:

– Univariate Imputation: This approach focuses solely on the attribute with missing values. Common univariate techniques include replacing missing val-ues with the mean, median, or most frequent value within the attribute. These methods are simple to implement but may not effectively capture the underlying relationships between attributes.

– Multivariate Imputation: This more sophisticated approach leverages the values of other attributes within a sample to estimate the missing value. Techniques like regression analysis are often employed to establish relation-ships between the missing attribute and the remaining attributes. Based on these relationships, a predicted value can be imputed for the missing data point. Multivariate imputation offers a more nuanced approach but requires careful consideration of the relationships between attributes and potential biases in the imputation process.

In the 'impute values.py' script, in the 'missing values' function, we used the IterativeImputer class from the sklearn.impute module to apply multivariate imputation to address missing values in the datasets for continuous numeric attributes. The script uses the most frequent value strategy for categorical at-tributes. The imputed datasets are saved in the same folder as the original datasets, with the prefix 'preprocessed missing'.

3.2. Outlier Detection and Treatment

Outliers, data points that deviate significantly from the rest of the dataset, can adversely affect the performance of machine learning models. Outliers can skew statistical measures, distort relationships between attributes, and lead to poor generalization of the model. Detecting and treating outliers is essential for ensuring the robustness and reliability of the model.

We purpose to impute the outliers using the IsolationForest algorithm from the sklearn.ensemble module. The script 'outlier detection.py' detects outliers in the continuous numeric attributes of the datasets and replaces them with the imputed values. The preprocessed datasets with imputed outliers are saved in the same folder as the original datasets, with the prefix 'preprocessed outliers'.

3.3. Analysis of Attribute Correlations

As previously discussed, attribute correlations can provide valuable insights into the relationships between different attributes in the dataset. By identifying highly correlated attributes, we can eliminate redundant information and re-duce the dimensionality of the data, leading to more efficient model training and improved interpretability.

We choose to remove highly correlated attributes found in the section of Exploratory Data Analysis. These attributes are:

– analysis results: it is correlated with body mass indicator in the Stroke Pre-diction dataset.

– chaotic sleep: it is correlated with cardiovascular issues in the Stroke Pre-diction dataset.

The script 'remove correlated attributes.py' removes those attributes from the train dataset and saves the preprocessed dataset in the same folder as the original datasets, with the prefix 'preprocessed correlated'.

3.4. Normalization and Standardization

The numerical attributes in the dataset can vary significantly in their value scales. For example, some attributes may have values in the thousands, while others have values in the single digits. This disparity in scales can significantly affect algorithms like Logistic Regression.

In algorithms like Logistic Regression, which rely on a linear combination of attribute values, attributes with larger numerical values can disproportionately influence the model. This dominance can lead to biased results and reduce the model's effectiveness.

To mitigate this issue, it is essential to standardize the values of the numeric attributes. Standardization adjusts the scales of the attributes, ensuring that each one contributes equally to the model's predictions. This process improves the performance and accuracy of the model by creating a more balanced and fair representation of the data.

4. Algorithms Designs

Algorithm design is a critical aspect of computer science and machine learning, focusing on creating efficient and effective methods to solve complex problems. The process involves the careful selection of algorithms based on the specific characteristics of the data and the desired outcomes. This document explores the application of two prominent machine learning algorithms, Logistic Regression and Multi-Layered Perceptron (MLP), on diverse datasets. The goal is to compare their performance and suitability for different types of prediction tasks, particularly in the contexts of stroke prediction.

4.1. Logistic Regression

Logistic regression is a fundamental statistical method employed for classification tasks in machine learning. It establishes a mathematical model that maps a set of input features (independent variables) to a probability of a specific out-come (dependent variable). The core functionality lies in estimating the odds of a particular class membership (e.g., presence of stroke) based on the input features. The resulting model essentially learns a decision boundary, separating observations with high and low probabilities of belonging to the target class. This characteristic makes logistic regression particularly well-suited for analyzing datasets where the outcome variable is binary (e.g., stroke occurrence vs. no stroke occurrence).

In the logistic regression folder at the root of the project directory, we implemented the Logistic Regression algorithm in two different ways:

- Logistic Regression with Scikit-Learn: We used the Scikit-Learn library to implement Logistic Regression on the preprocessed datasets.
- Logistic Regression from Scratch: We implemented Logistic Regression from scratch using the Negative Log-Likelihood method and the Gradient Descent optimization algorithm.

Before starting the implementation of the Logistic Regression algorithm, we need to encode the categorical attributes in the datasets. Categorical attributes are non-numeric attributes that represent discrete categories or groups. These attributes need to be encoded into a numerical format before they can be used in machine learning algorithms.

For encoding the categorical attributes except the target attribute, I used the OneHotEncoder class from the sklearn.preprocessing module. This class encodes categorical attributes as one-hot vectors, creating a binary representation of each category. This encoding is essential for feeding categorical attributes into machine learning models, as most algorithms require numerical input data. For the target attribute, I used the LabelEncoder class from the sklearn.preprocessing module to encode the target attribute as integer values.

In the Logistic Regression with Scikit-Learn implementation, we used the LogisticRegression class from the sklearn.linear model module to train the model on the preprocessed datasets, without setting any hyperparameters, using the default values. For the Logistic Regression from Scratch implementation, I implemented the Negative Log-Likelihood loss function and the Gradient Descent optimization algorithm. We trained the model on the preprocessed datasets, setting the learning rate to 0.01 and the number of epochs to 10000. For the regularization, we used the Ridge Regression technique.

The results of both implementations for each dataset are saved in the Logis-ticRegression folder at the specific dataset's root.

4.2. Multi-Layered Perceptron (MLP)

A Multilayer Perceptron (MLP) is a class of feedforward artificial neural network that consists of at least three layers of nodes: an input layer, one or more hidden layers, and an output layer. Each node, except for the input nodes, is a neuron that uses a nonlinear activation function. MLPs are capable of modeling com-plex relationships in data, making them suitable for tasks such as classification, regression, and pattern recognition. The network learns by adjusting the weights through a process called backpropagation, which minimizes the error between the predicted outputs and the actual

targets. This adaptability and learning capability make MLPs powerful tools in machine learning and artificial intelligence applications.

In the mlp folder at the root of the project directory, we implemented the Multi-Layered Perceptron algorithm in two different ways:

– MLP with Scikit-Learn: We used the Scikit-Learn library to implement the MLP algorithm on the preprocessed datasets.

– MLP from Scratch: We implemented the MLP algorithm from scratch us-ing the Negative Log-Likelihood method, the Gradient Descent optimization algorithm, and the Sigmoid activation function.

Before starting the implementation of the MLP algorithm, we need to standardize the numeric attributes in the datasets as in the Logistic Regression algorithm.

For the MLP with Scikit-Learn implementation, we used the MLPClassifier class from the sklearn.neural network module to train the model on the prepro-cessed datasets, without setting any hyperparameters, using the default values. For the MLP from Scratch implementation, we implemented the Negative Log-Likelihood loss function, the Gradient Descent optimization algorithm, and the Sigmoid activation function. We trained the model on the preprocessed datasets, setting the learning rate to 0.01, the number of epochs to 10000, and the number of hidden units to 100. For the regularization, we used the Ridge Regression technique.

The results of both implementations for each dataset are saved in the MLP folder at the specific dataset's root.

5. Evaluation

The evaluation of machine learning models is a critical step in assessing their performance and effectiveness. By comparing the model's predictions to the actual ground truth, we can determine the model's accuracy, precision, recall, and other metrics that quantify its performance. This section delves into the evaluation of the Logistic Regression and Multi-Layered Perceptron (MLP) models on the Stroke Prediction datasets.

5.1. Hyperparameter Tuning

Hyperparameters are parameters that are set before the learning process begins. They control the learning process and the behavior of the model. Hyperparame-ter tuning is the process of selecting the optimal hyperparameters for a machine learning model to achieve the best performance. This process involves search-ing through different hyperparameter configurations and evaluating the model's performance on a validation set to find the optimal settings.

In the context of the Logistic Regression (manual implementation), the hy-perparameters that were used are:

– learning rate: The rate at which the model updates the weights - 0.01 – num iterations The number of iterations the model trains for - 10000 – regularization The regularization parameter to prevent overfitting - 0.1

In the context of the Logistic Regression (Scikit-Learn implementation), the majority of hyperparameters that were used are the default values provided by the Scikit-Learn library. The only hyperparameters that were set are:

- solver: The optimization algorithm used in the model - 'sag' for AVC
- max iter: The maximum number of iterations for the optimization algorithm - 500 for AVC
- C: The regularization parameter to prevent overfitting - 2.56050926 for AVC

In the context of the Multi-Layered Perceptron (manual implementation), the hyperparameters that were used are:

- hidden sizes: The sizes of the hidden layers are defined as a list - [256, 128, 64]
- num epochs: The number of epochs the model trains for - 100
- learning rate: The rate at which the model updates the weights - 0.01
- Loss Function : The loss function used to optimize the model - Negative Log-Likelihood

In the context of the Multi-Layered Perceptron (Scikit-Learn implementation), the hyperparameters that were used are the default values provided by the Scikit-Learn library.

5.2. Confusion Matrix

A confusion matrix is a table that summarizes the performance of a classification model on a set of test data for which the true values are known. It provides insights into the model's predictions, including true positive, true negative, false positive, and false negative instances. These metrics are essential for evaluating the model's performance and identifying potential areas for improvement.

In the context of the Logistic Regression and Multi-Layered Perceptron mod-els, we generated confusion matrices to analyze the model's predictions on the test data. The confusion matrices provide a detailed breakdown of the model's performance, highlighting the number of correct and incorrect predictions for each class.

In Figures 6 and 7 we can see the confusion matrices for the Logistic Re-gression model on the Stroke Prediction dataset, implemented manually and with Scikit-Learn, respectively. In Figures 8 and 9 we can see the confusion matrices for the Multi-Layered Perceptron model on the Stroke Prediction dataset, implemented manually and with Scikit-Learn, respectively.

Because of the high-class imbalance in the Stroke Prediction dataset, the models can't predict the positive class effectively (the positive class is less than 5% of the dataset). On the other hand, the models can predict the negative class effectively.

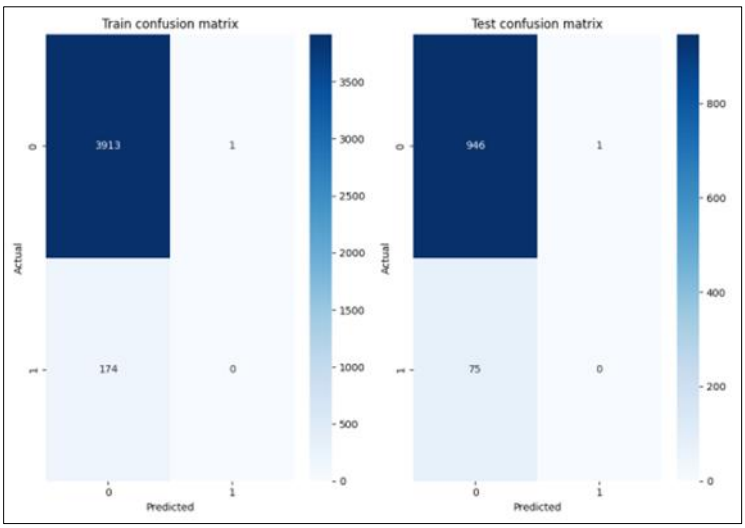

Figure 6 Confusion Matrix for the Logistic Regression model on the Stroke Prediction dataset (Manual Implementation)

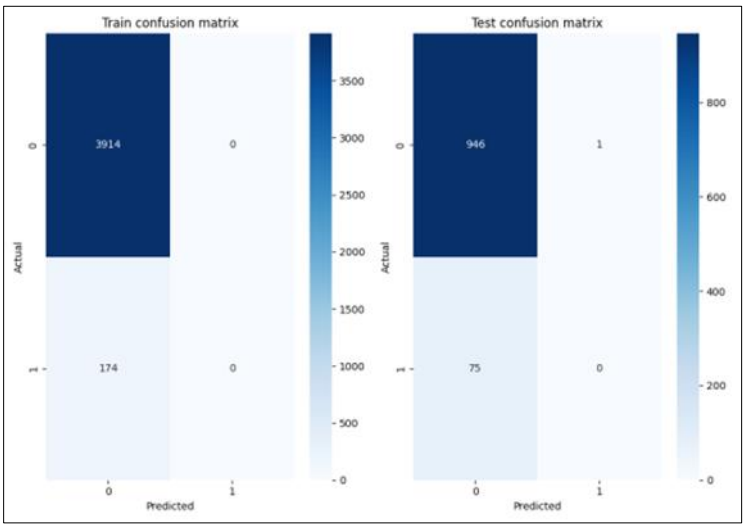

Figure 7 Confusion Matrix for the Logistic Regression model on the Stroke Prediction dataset (Scikit-Learn Implementation)

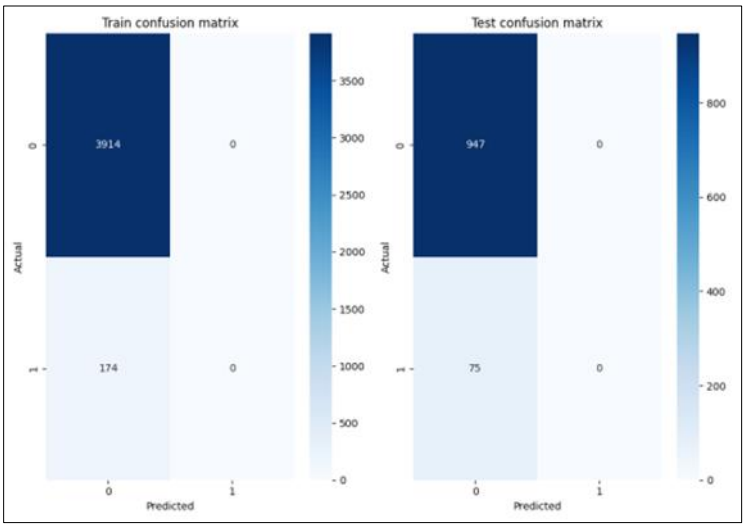

Figure 8 Confusion Matrix for the Multi-Layered Perceptron model on the Stroke Pre-diction dataset (Manual Implementation)

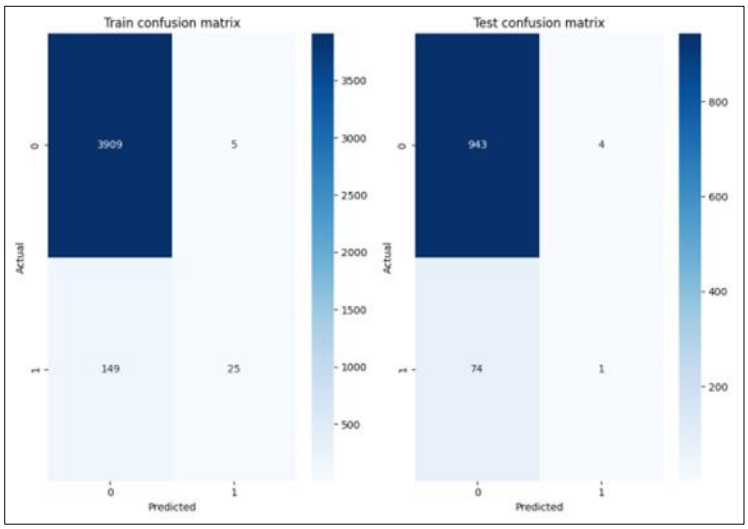

Figure 9 Confusion Matrix for the Multi-Layered Perceptron model on the Stroke Pre-diction dataset (Scikit-Learn Implementation)

5.3. Evaluation Metrics

To evaluate the performance of the machine learning models, we employ a range of evaluation metrics that provide insights into different aspects of the model's performance. These metrics include:

- Accuracy: The proportion of correctly classified instances out of the total instances. It provides a general measure of the model's correctness.
- Precision: The proportion of true positive predictions out of all positive predictions. It measures the model's ability to avoid false positives.
- Recall: The proportion of true positive predictions out of all actual positive instances. It measures the model's ability to capture all positive instances.
- F1-Score: The harmonic means of precision and recall. It provides a balanced measure of the model's performance.

These evaluation metrics help us understand the strengths and weaknesses of the machine learning models and guide us in improving their performance. By analyzing these metrics, we can identify areas for optimization and fine-tuning to enhance the model's predictive capabilities.

In the following table, Table 4, we present the evaluation metrics for the Logistic Regression and Multi-Layered Perceptron models on Stroke Prediction.

6. Conclusions

We investigated the use of machine learning algorithms for stroke prediction in this document. We performed a thorough examination of the datasets, encompassing preprocessing, feature correlation analysis, and data exploration. To forecast the results of the prediction, we used two well-known machine learning algorithms: Multi-Layered Perceptron (MLP) and Logistic Regression. We assessed the models' performance using a number of evaluation metrics, such as F1-Score, recall, accuracy, and precision.

The models performed differently on the prediction tasks, according to the evaluation metrics results. On the Stroke Prediction dataset, the Multi-Layered Perceptron and Logistic Regression models produced low F1-Score and recall values for the positive class.

Overall, it is evident that across both datasets, the Multi-Layered Perceptron (Scikit-Learn implementation) outperforms Logistic Regression. Specifically:

– Stroke Prediction: The MLP model outperforms Logistic Regression in terms of precision and recall, even though class imbalance presents a challenge. Logistic Regression is unable to predict the positive class at all, yielding an F1-score of 0.

Thus, based on these metrics, the Multi-Layered Perceptron (Scikit-Learn implementation) is the superior algorithm for the Stroke Prediction tasks based on these metrics.

References

[1] Rynkiewicz, Joseph. (2012). General bound of overfitting for MLP regression models. Neurocomputing. 90. 10.1016/j.neucom.2011.11.028.

[2] Jafar Alzubi et al 2018 J. Phys.: Conf. Ser. 1142 012012

[3] Chen JX. The evolution of computing: AlphaGo. Computing in Science & Engineering. 2016 Jul;18(4):4-7.4.

[4] Chantamit-O-Pas P, Goyal M. Prediction of stroke using deep learning model. In: Neural Information Processing: 24th International Conference, ICONIP 2017, Guangzhou, China, November 14e18, 2017, Proceedings, Part V 24. Berlin, Germany: Springer International Publishing; 2017:774e781.

[5] Katan M., Luft A. *Seminars in Neurology.* Volume 38. Thieme Medical Publishers; New York, NY, USA: 2018. Global burden of stroke; pp. 208–211.

[6] Boehme A.K., Esenwa C., Elkind M.S. Stroke risk factors, genetics, and prevention. *Circ. Res.* 2017;**120**:472–495. doi: 10.1161/CIRCRESAHA.116.308398.

[7] Heo J, Yoon JG, Park H, *et al*. Machine learning-based model for prediction of outcomes in acute stroke. Stroke 2019;**50**:1263–5. doi:10.1161/STROKEAHA.118.024293