Exploring the Utility of Machine Learning Across Varied Data Formats

Cărămidă Iustina-Andreea - 332CA

Faculty of Automatic Control and Computer Science
University Politehnica of Bucharest
iustina.caramida@stud.acs.upb.ro

Abstract. This study investigates the applicability of machine learning techniques on diverse datasets. We explore the effectiveness of two algorithms, Logistic Regression and Multi-Layered Perceptron (MLP), on predicting both health outcomes and financial well-being. Specifically, we utilize a stroke prediction dataset to assess the model's ability to identify individuals at risk of stroke. Additionally, we employ a salary prediction dataset to evaluate the model's capacity to classify individuals earning above a specific income threshold (e.g., $50,000 per year). Through comparative analysis, this research aims to elucidate the strengths and limitations of each algorithm when applied to these contrasting data types, offering insights into their suitability for various prediction tasks. Furthermore, we present a framework for data analysis, outlining essential steps for data cleaning, exploration, and preparation, which can be applied to enhance the effectiveness of machine learning models across diverse datasets.

Keywords: *Machine Learning · Heterogeneous Data · Comparative Analysis · Prediction Modeling · Data Analysis Techniques · Stroke Prediction · Salary Prediction · Logistic Regression · Multi-Layered Perceptron · DataPreprocessing*

1 Introduction

1.1 Motivation: The Power and Nuance of Machine Learning Data

Machine learning (ML) has become a cornerstone of progress in numerous disciplines. Its ability to extract valuable insights from vast and complex datasets has fueled breakthroughs in healthcare, finance, and social sciences. However, the effectiveness of ML models is not a one-size-fits-all proposition. Different data types possess unique characteristics, and understanding these nuances is essential for selecting the most appropriate ML algorithms. Data can be structured (organized in tables) or unstructured (text, images), numerical or categorical, and may exhibit linear or non-linear relationships between features. Choosing the right algorithm depends heavily on these factors. This research delves into this crucial aspect of ML application by exploring the performance of two distinct algorithms on contrasting datasets.

1.2 Research Focus: Delving into Stroke Prediction and Salary Prediction

This study focuses on the application of ML techniques to two contrasting datasets: stroke prediction and salary prediction. Stroke, a leading cause of disability and death globally, poses a significant public health burden. Stroke prediction models aim to identify individuals at high risk of experiencing a stroke, allowing for preventive measures and early intervention. These models typically analyze factors such as age, blood pressure, cholesterol levels, and smoking history.

Conversely, salary prediction models attempt to classify individuals based on income thresholds. This information can offer valuable insights into economic trends, such as income inequality, and inform policy decisions. Salary prediction models might analyze factors like education level, work experience, and industry sector. By investigating these two distinct datasets, this research aims to gain a broader understanding of how ML algorithms perform on different data types with varying underlying structures and complexities.

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

This section delves into the core methodologies employed in this study: Logistic Regression and Multi-Layered Perceptron (MLP). Both algorithms fall under the umbrella of supervised learning, where a model learns from labeled data to make predictions on unseen examples. Here, we unveil the underlying principles and functionalities of each technique.

Logistic regression serves as a foundational algorithm for classification tasks. It establishes a mathematical model that maps input features (e.g., age, blood pressure) to a probability of a specific outcome (e.g., stroke occurrence). The model essentially learns a decision boundary, separating observations with high and low probabilities of the target outcome. This approach makes logistic regression well-suited for analyzing datasets like the stroke prediction one, where the goal is to categorize individuals based on their risk level.

On the other hand, Multi-Layered Perceptron (MLP) represents a more complex architecture - a type of artificial neural network. It consists of interconnected layers of artificial neurons, mimicking the structure of the human brain. Each layer transforms the received data using activation functions, ultimately leading to an output prediction. MLP's strength lies in its ability to learn complex, non-linear relationships within data. This makes it a powerful tool for tackling intricate prediction problems, potentially outperforming logistic regression when the underlying relationships are not easily captured by a linear model. The salary prediction dataset, where various factors might influence income levels, could be a prime candidate for exploration with MLP.

1.4 Research Objectives: Evaluating Algorithms, Unveiling Strengths and Weaknesses

By comparatively analyzing the performance of Logistic Regression and MLP on stroke and salary prediction tasks, this research seeks to achieve several key objectives:

Evaluate the Suitability of Algorithms for Diverse Data Types: This involves assessing the effectiveness of each algorithm in capturing the underlying relationships within the stroke prediction and salary prediction datasets. We will determine which algorithm performs better on each dataset, offering insights into their suitability for different data types.

Gain Insights into Algorithmic Strengths and Weaknesses: By analyzing the comparative performance, we aim to highlight the scenarios where each algorithm excels and identify areas where one might outperform the other. This will provide valuable guidance for researchers and practitioners in selecting the most appropriate algorithm for their specific prediction tasks.

Demonstrate Best Practices for Data Analysis in ML Applications: Effective data analysis is crucial for building robust ML models. This research will showcase essential steps for data cleaning, exploration, and preparation, emphasizing their importance in enhancing model performance across diverse datasets. These steps may include handling missing values, identifying outliers, and feature engineering (creating new features from existing data) to improve the model's ability to learn from the data.

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

Through this exploration, the research aims to contribute valuable knowledge to the field of machine learning, particularly the application of ML on heterogeneous datasets. The findings can guide researchers and practitioners in selecting appropriate algorithms for their specific prediction tasks and data types. Furthermore, by demonstrating best practices for data analysis, this research can contribute to the development of more robust and reliable ML models across diverse application domains. This can lead to advancements in areas like healthcare (improved stroke prediction for preventive measures) and economics (better understanding of factors influencing income inequality). Ultimately, the research aims to contribute to the responsible and effective use of ML for tackling complex problems across various fields.

2 Exploratory Data Analysis

2.1 Datasets attributes description

The initial and crucial step in developing any machine learning algorithm involves a thorough understanding of the data it will be trained on. This understanding is achieved through a comprehensive analysis of the datasets' characteristics. In this vein, the following sub sections will delve into the specific attributes of the two datasets employed in this study: stroke prediction and salary prediction.

A detailed description of each salary prediction attribute is provided in Table 1 and of each stroke prediction attribute in Table 2.

List of all attributes in the Salary Prediction dataset		
Attribute name	Type	Details
fnl	numeric	Socio-economic characteristic of the population from which the individual comes
hpw	numeric	Number of work hours per week
relation	categorical	The type of relationship in which the individual is involved
gain	numeric	Capital gain
country	categorical	Country of origin
job	categorical	The individual's job
edu_int	numeric	Number of years of study
years	numeric	Age of the individual
loss	numeric	Loss of capital
work_type	categorical	The job's type
partner	categorical	The type of partner the individual has
edu	categorical	The individual's type of education
gender	categorical	Individual's gender
race	categorical	Individual's race
prod	numeric	Capital production
gtype	categorical	Type of employment contract
money	categorical	Whether the individual earns more than $50,000 per year

Table 1: Salary 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 observation of the subject	
cardiovascular_issues	categorical	Whether or not the subject has a medical history cardiovascular	0, 1
job_category	categorical	The field in which the person works	child, entrepreneurial, N_work_history, private_sector, public_sector
body_mass_indicator	numeric	Body mass index, which indicates if the person is underweight, within limits normal, overweight or obese	
sex	categorical	The gender of the person	F, M
tobacco_usage	categorical	Current or past smoker indicator	ex-smoker, smoker, non-smoker
high_blood_pressure	categorical	Binary attribute indicating whether a person suffer from high blood pressure or not	0, 1
married	categorical	Binary attribute indicating whether the person a ever been married	Y, N
living_area	categorical	The type of area where he lived most of his life	City, Countryside
years_old	numeric	The person's age in years	
chaotic_sleep	categorical	Binary attribute for a sleep program irregular	0, 1

analysis_results	numeric	The results of medical analyzes 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 indicator indicating whether the person a had a stroke or not	0, 1

Table 2: Stroke Prediction Attributes

2.2 Exploration of Attribute Types and Value Ranges

Prior to applying a machine learning model to a dataset, a crucial step involves in identifying the types of attributes (features) present and their corresponding values ranges. This anaysis is essential for selecting appropriate algorithms and ensuring optimal model performance. In the following pharagraphs we will describe three primary attribute types.

- *Continuous Numeric Attributes:* These attributes possess numerical values that can theoretically take on any value within a specific range. Examples might include: age, weight, temperature etc.
- *Discrete Nominal Attributes:* These attributes represent categorical data with distinct, non-ordered values. Examples include days of the week (Monday, Tuesday, etc.) or types of diseases (cancer, diabetes, etc.).
- *Ordinal Attributes:* These attributes represent categorical data with values that exhibit an inherent order. However, the difference between consecutive values may not be interpretable in terms of a consistent unit. Examples include customer satisfaction ratings (1-star, 2-star, etc.) or movie ratings (G, PG, PG-13, etc.). In ordinal attributes, the numerical value itself might not be as important as the relative order it represents.

Using the *analysis_attributes.py* script, we can identify the Continuous Numeric Attributes and Discrete Nominal Attributes in the datasets. The script will output statistics that can be showed in Tables 3 and 4 for numeric attributes and Table 5 and 6 for discrete attributes.

Moreover, the total number of items in the full dataset is 9999 for the Salary Prediction dataset and 5110 for the Stroke Prediction dataset.

List of all Continuous Numeric Attributes in the Salary Prediction dataset							
	fnl	hpw	gain	edu_int	years	loss	prod
count	9.999000e +03	9199.00000	9999.00000	9999.00000	9999.00000	9999.00000	9999.00000
mean	1.903529e +05	40.416241	979.853385	14.262026	38.646865	84.111411	2014.9275 93
std	1.060709e +05	12.517356	7003.7953 82	24.770835	13.745101	3394.0354 84	14007.6044 96
min	1.921400e +04	1.000000	0.000000	1.000000	17.000000	0.000000	-28.000000
25%	1.182825e +05	40.000000	0.000000	9.000000	28.000000	0.000000	42.000000
50%	1.784720e +05	40.000000	0.000000	10.000000	37.000000	0.000000	57.000000
75%	2.373110e +05	45.000000	0.000000	13.000000	48.000000	0.000000	77.000000
max	1.455435e +06	99.00000	99999.0000	206.000000	90.000000	3770.00000	200125.000

Table 3: Continuous Numeric Attributes in Salary 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

Table 4: Continuous Numeric Attributes in Stroke Prediction Dataset

An initial inspection of the data reveals that there are missing attributes in both the Salary Prediction and Stroke Prediction datasets. In the Salary Prediction datase the *'hpw'* attribute is missing, while int 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 the Figure 1 we can see a boxplot for the *years* attribute in the Salary Prediction dataset. The box in the middle of the plot contains the middle 50% of the data, and the line in the middle represents the median. The whiskers extend to the minimum and maximum values within 1.5 times the interquartile range (the difference between the first and third quartiles). Points outside this range are considered outliers.

Also, in Figure 2 we 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. These information is also presented in Tables 3 and 4. 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.

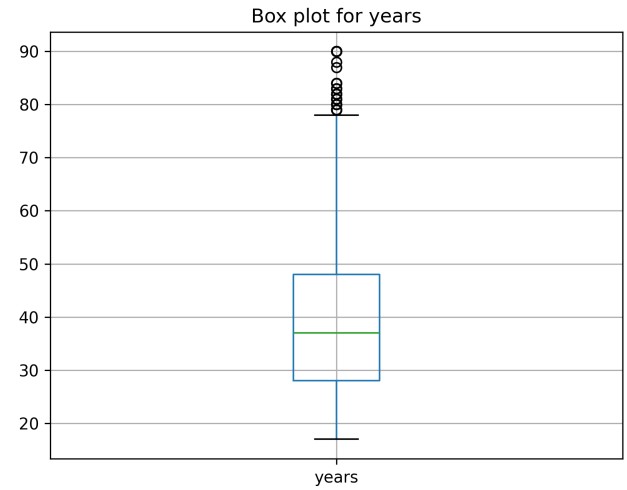

Fig. 1. Boxplot for the *years* attribute in the Salary Prediction dataset

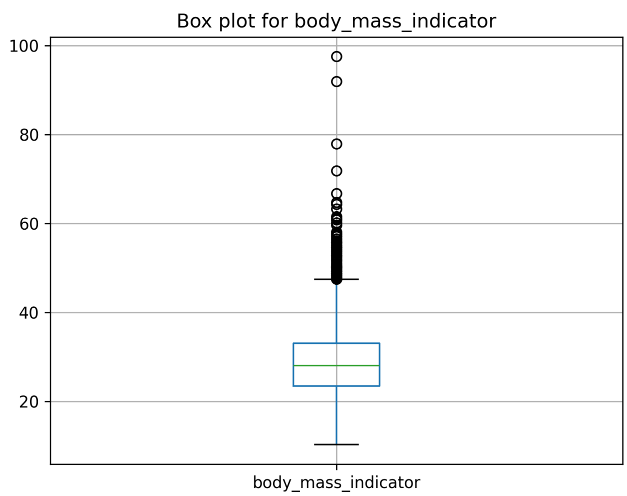

Fig. 2. Boxplot for the *body_mass_indicator* attribute in the Stroke Prediction dataset

List of all Discrete Nominal Attributes in the Salary Prediction dataset		
	Non-missing count	Unique values count
relation	9999	6
country	9999	41
job	9999	14
work_type	9999	9
partner	9999	7
edu	9999	16
gender	9199	2
race	9999	5
gtype	9999	2
money	9999	2

Table 5: Discrete Nominal Attributes in Salary 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

Table 6: Discrete Nominal Attributes in Stroke Prediction Dataset

From the Discret Nominal Attributes tables (Tables 5 and 6) we can see that each dataset contains only one attribute with missing values. In the Salary Prediction dataset the *gender* attribute is missing, while the Stroke Prediction dataset the *married* attribute is missing. Also, the number of unique values for each attribute descibes the diversity of the data. For example, the *country* attribute in the Salary Prediction dataset has 41 unique values, indicating that the data contains information from 41 different countries.

In the historigrams for the discrete nominal attributes, we 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_'*.

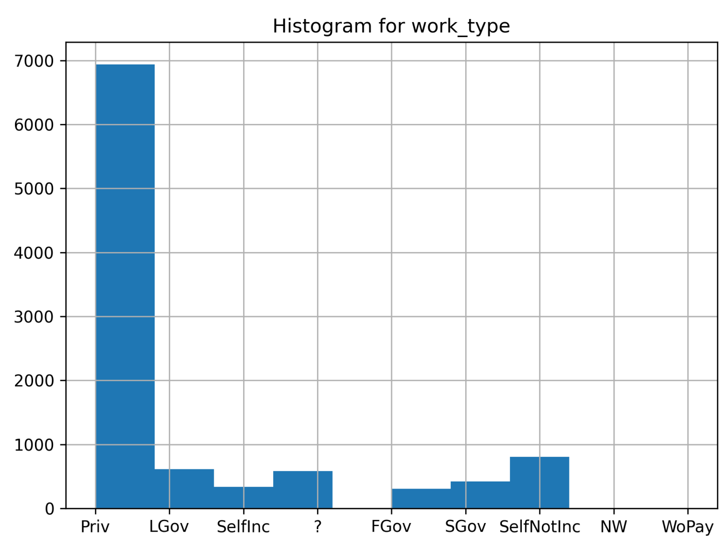

Fig. 3. Histogram for the *work_type* attribute in the Salary Prediction dataset

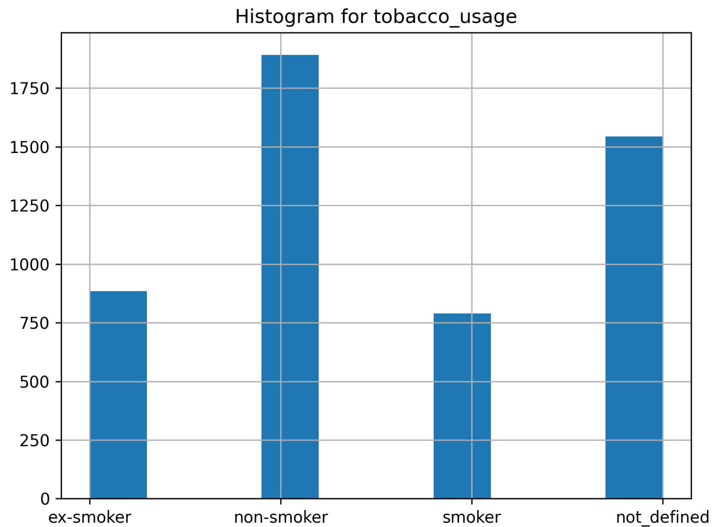

Fig. 4. Histogram for the *tobacco_usage* attribute in the Stroke Prediction dataset

In Figure 3 we can see a histogram of the *work_type* attribute in the Salary Prediction dataset. The dominance of the 'Priv' category indicates a severe class imbalance. For classification tasks, the model might predict Priv most of the time since it's the majority class, leading to a high overall accuracy but poor precision, recall, and F1 scores for minority classes.

Also, in Figure 4 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.

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 robustness 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 undersample 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 Figures 5 and 6 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. On the other hand, the Salary Prediction dataset exhibits a more balanced distribution of the *money* class, which may require less intervention to handle class imbalance.

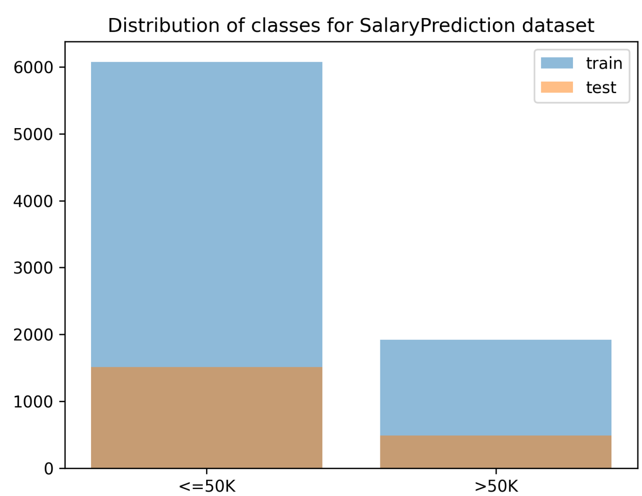

Fig. 5. Distribution of the *money* class in the Salary Prediction dataset

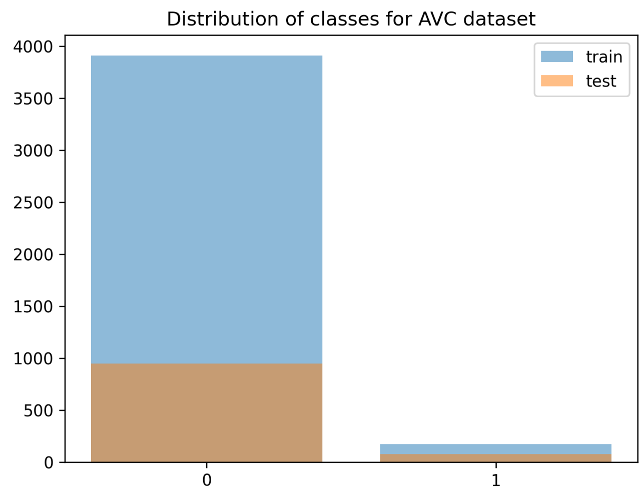

Fig. 6. Distribution of the *cerebrovascular_accident* class in the Stroke Prediction dataset

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 between 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 Cramér's V coefficient for all pairs of discrete nominal attributes in the datasets, generating a Cramér's V matrix for each dataset to measure the association between categorical variables. In Figures 7 and 8 we can see the correlation matrix for the Salary Prediction and Stroke Prediction datasets, respectively, for the continuous numeric attributes. In Figures 9 and 10 we can see the Cramér's V matrix for the discrete nominal attributes.

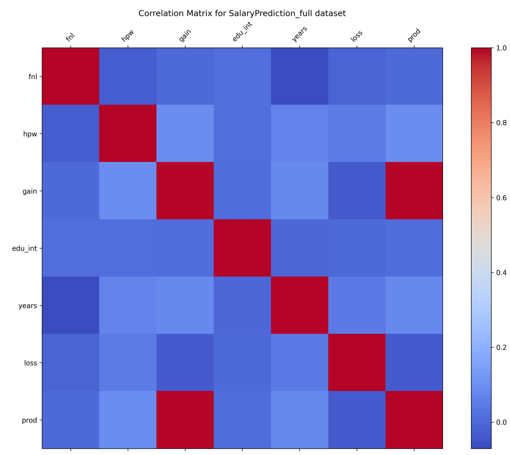

Fig. 7. Correlation matrix for the Salary Prediction dataset

The correlation matrix and Cramér's V matrix provide valuable insights into the relationships between attributes in the datasets. By examining these matrices, we can see that Figure 7 the *prod* attribute is highly correlated with the *gain* attribute, while the *years* attribute is negatively correlated with the *fnl* attribute.

In Figure 8 we can see that the *mean_blood_sugar_level* attribute is highly correlated with the *analysis_results* attribute, while the *body_mass_indicator* attribute is negatively correlated with the *analysis_results* attribute.

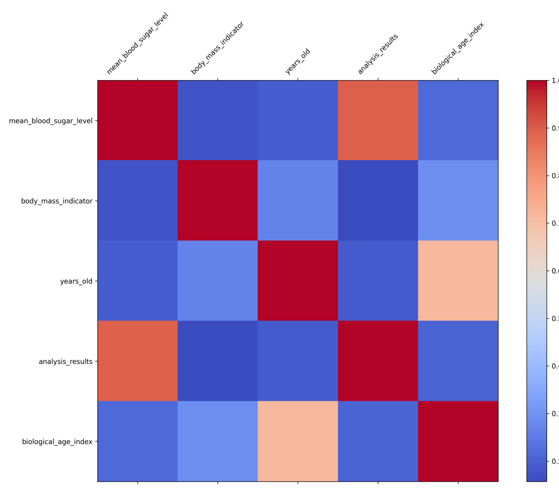

Fig. 8. Correlation matrix for the Stroke Prediction dataset

The Cramér's V matrix in Figures 9 and 10 provides insights into the association between discrete nominal attributes. For example, in the Salary Prediction dataset, the *gtype* attribute is strongly associated with the *gender* attribute, while in the Stroke Prediction dataset, the *cardiovascular_issues* attribute is strongly associated with the *chaotic_sleep* attribute.

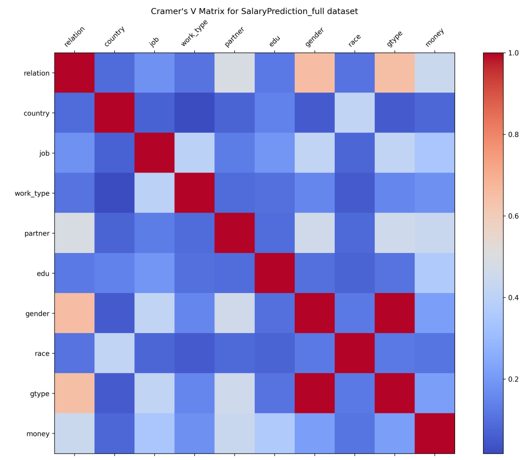

Fig. 9. Cramér's V matrix for the Salary Prediction dataset

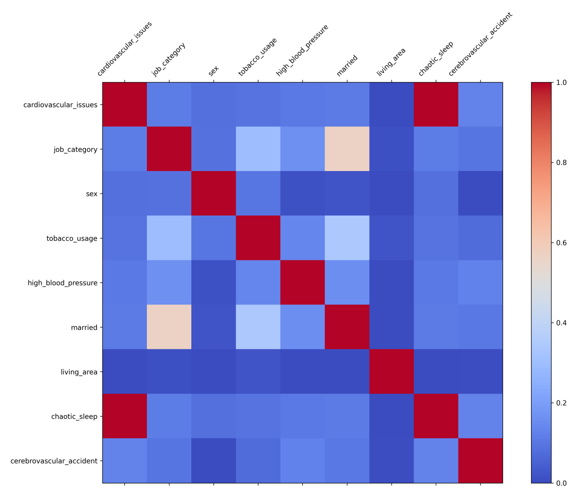

Fig. 10. Cramér's V matrix for the Stroke Prediction dataset

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 values 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 relationships 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 attributes. 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 reduce 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:

- *prod*: it is correlated with *gain* in the Salary Prediction dataset.
- *analysis_results*: it is correlated with *body_mass_indicator* in the Stroke Prediction dataset.
- *gtype*: it is correlated with *gender* in the Salary Prediction 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 and salary 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 outcome (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 enconding 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, we 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 *LogisticRegression* folder at the specific dataset's root.

4.2 Multi-Layered Perceptron (MLP)

Multi-Layer Perceptron (MLP) is a type of artificial neural network designed to capture complex, non-linear patterns in data. Unlike Linear Regression, which is limited to linear relationships, MLPs consist of multiple layers of interconnected nodes, or neurons, that can model intricate interactions between variables.

MLPs are particularly suitable for tasks involving complex datasets with non-linear relationships, such as salary prediction. In salary prediction, factors like education, work experience, industry sector, and other socio-economic variables interact in complex ways that Linear Regression might not fully capture. MLPs can learn these non-linear patterns, providing a more accurate and nuanced understanding of the factors influencing salary levels.

5 Evaluation

6 Conclusions

References

1. Moodle - Artifical Intelligence Course
2. Wikipedia - Linear Regression
3. Wikipedia - Linear Classifier
4. Wikipedia - Ridge Regression
5. Metrics for Multi-Class Classification: An Overview
6. Pandas - DataFrame
7. Wikipedia - Correlation
8. Thinking Neuron - How to Measure the Correlation Between Two Categorical Variables in Python
9. StrataScratch - Chi-Square Test in Python: A Technical Guide
10. Wikipedia - Cramér's V
11. Saturn Cloud - How to Detect and Exclude Outliers in a Pandas DataFrame
12. Kaggle - When to standardize test and train data?
13. Scikit-Learn - Standardization or Mean Removal and Variance Scaling
14. Scikit-Learn - OneHotEncoder
15. Scikit-Learn - LabelEncoder
16. Towards Data Science - Understanding Confusion Matrix