World Journal of Advanced Engineering Technology and Sciences

eISSN: 2582-8266

Cross Ref DOI: 10.30574/wjaets

Journal homepage: https://wjaets.com/

(Research Article)

A study of machine learning algorithms for predicting financial well-being: Logistic regression vs. MLP

Paulami Bandyopadhyay *

Senior Data Engineer, Independent Researcher, India.

World Journal of Advanced Engineering Technology and Sciences, 2021, 03(01), 084–096

Publication history: Received on 08 July 2021; revised on 22 August 2021; accepted on 25 August 2021

Article DOI: https://doi.org/10.30574/wjaets.2021.3.1.0058

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 financial well-being. Specifically, 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; Salary Prediction; Logistic Regression; Multi-Layered Perceptron; Data Preprocessing

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 Salary Prediction

This study focuses on the application of ML techniques to salary prediction study. 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.

* Corresponding author: Paulami Bandyopadhyay.

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 to a probability of a specific outcome. The model essentially learns a decision boundary, separating observations with high and low probabilities of the target outcome.

On the other hand, Multi-Layered Perceptron (MLP) represents a more com-plex 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.

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

By comparatively analyzing the performance of Logistic Regression and MLP on 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 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 to 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. 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 in-volves a thorough understanding of the data it will be trained on. This under-standing is achieved through a comprehensive analysis of the dataset's characteristics. In this vein, the following sub sections will delve into the specific attributes of the datasets employed in this study: salary prediction.

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

Table 1 Salary Prediction Attribute

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 int	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

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 analysis is essential for selecting appropriate algorithms and ensuring optimal model performance. In the following paragraphs 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 for numeric attributes and Table 5 for discrete attributes.

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

Table 2 Continuous Numeric Attributes in Salary 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	2014.927593
std	1.060709e+05	12.517356	7003.795382	24.770835	13.745101	3394.035484	14007.604496
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	99.000000	99999.0000	206.000000	90.000000	3770.000000	200125.000

An initial inspection of the data reveals that there are missing attributes in the Salary Prediction datasets. In the Salary Prediction dataset, the 'hpw' attribute is missing.

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.

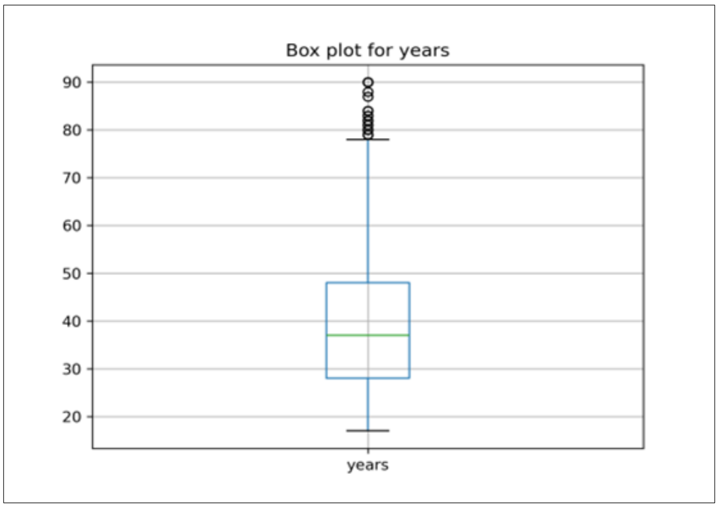

Figure 1 Boxplot for the years attribute in the Salary Prediction dataset

From the Discreet Nominal Attributes table (Tables 3) we can see that the dataset contains only one attribute with missing values. In the Salary Prediction dataset, the gender attribute is missing. Also, the number of unique values for each attribute describes 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.

Table 3 Discrete Nominal Attributes in Salary 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	9999	2
race	9999	5
gtype	9999	2
money	9999	2

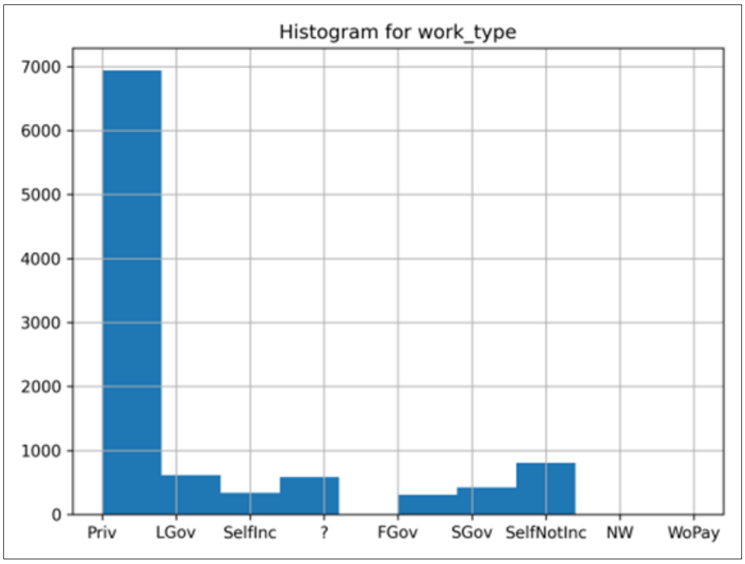

Figure 2 Histogram for the work type attribute in the Salary Prediction dataset

In the histograms 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_'*.

In Figure 2 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.

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.

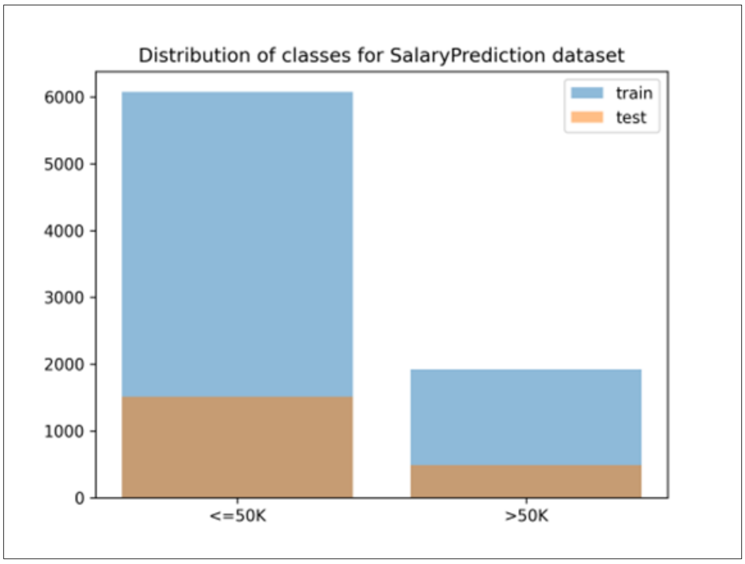

Figure 3 Distribution of the money class in the Salary Prediction dataset

In Figures 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. The Salary Prediction dataset exhibits a more balanced distribution of the money class, which may require less intervention to handle class imbalance.

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 Figure 4, we can see the correlation matrix for the Salary 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, we can see that Figure 4 the prod attribute is highly correlated with the gain attribute, while the years attribute is negatively correlated with the fnl attribute.

The Cram´er's V matrix in Figures 5 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.

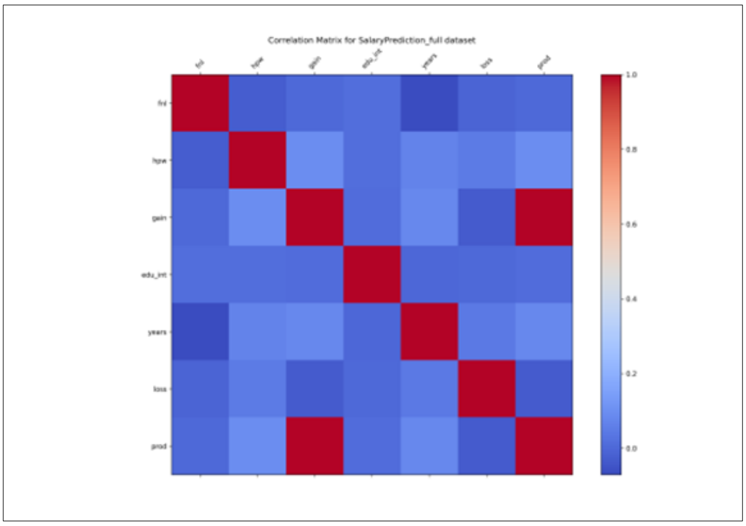

Figure 4 Correlation matrix for the Salary Prediction dataset

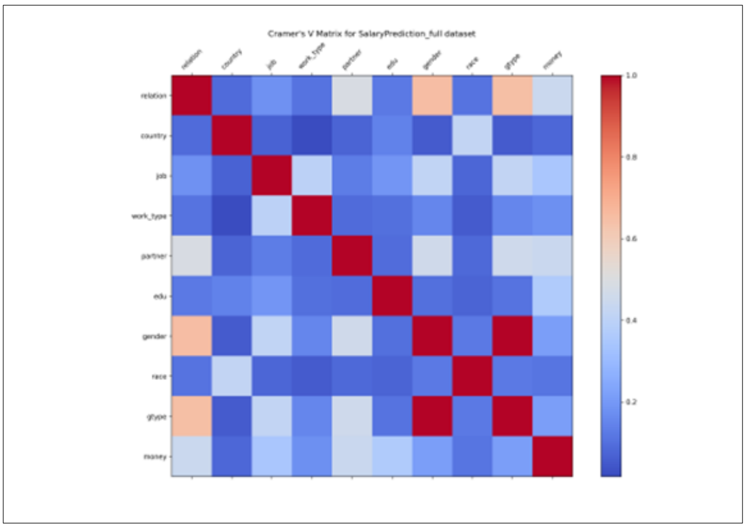

Figure 5 Cram´er's V matrix for the Salary Prediction dataset

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 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 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.
- 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 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 out-come (dependent variable). The core functionality lies in estimating the odds of a particular class membership 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.

In the logistic regression folder at the root of the project directory, we im-lamented 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 Lo-gistic Regression 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 Logistic Regression 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 ca-pability 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 stan-dardize 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 ac-tual ground truth, we can determine the model's accuracy, precision, recall, and other metrics that quantify its performance. This section delves into the evalua-tion of the Logistic Regression and Multi-Layered Perceptron (MLP) models on the Salary Prediction techniques.

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. Hyperparameter tuning is the process of selecting the optimal hyperparameters for a machine learning model to achieve the best performance. This process involves searching 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 hyperparameters 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 - 'saga' for Salary Pre-diction.
– max iter: The maximum number of iterations for the optimization algorithm - 200 for Salary Prediction.
– C: The regularization parameter to prevent overfitting - 0.4342470001 for Salary Prediction.

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 models, 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 11 and 12 we can see the confusion matrices for the Logistic Regression model on the Salary Prediction dataset, implemented manually and with Scikit-Learn, respectively. In Figures 13 and 14 we can see the confusion matrices for the Multi-Layered Perceptron model on the Salary Prediction dataset, implemented manually and with Scikit-Learn, respectively.

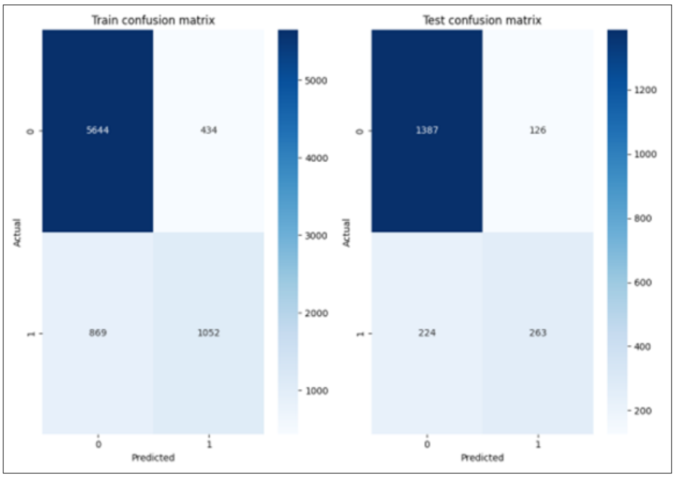

Figure 6 Confusion Matrix for the Logistic Regression model on the Salary Prediction dataset (Manual Implementation)

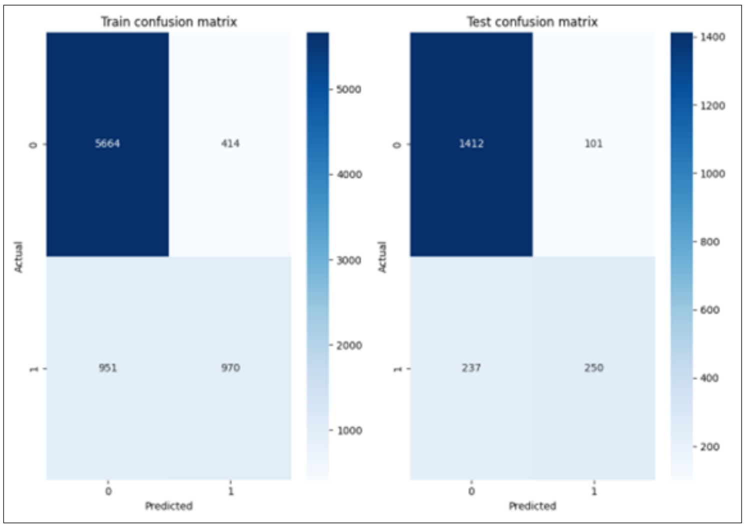

Figure 7 Confusion Matrix for the Logistic Regression model on the Salary Prediction dataset (Scikit-Learn Implementation)

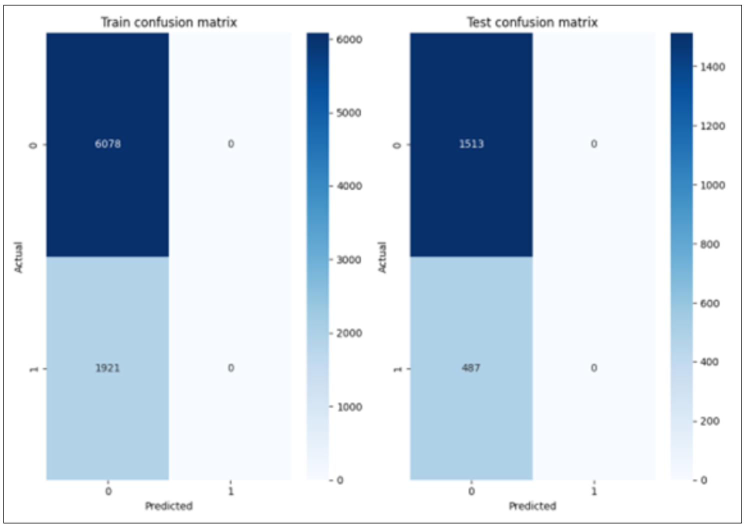

Figure 8 Confusion Matrix for the Multi-Layered Perceptron model on the Salary Pre-diction dataset (Manual Implementation)

We can see that the Logistic Regression model implemented with Scikit-Learn achieved almost the same results as the manual implementation and the MLP model implemented with Scikit-Learn. All 4 models achieved a high number of true positive predictions for the negative class, indicating that the models are effective at identifying negative instances in the dataset. The true positive predictions for the positive class are lower, but the models still achieved a rea-sonable prediction but the MLP model with manual implementation achieved the worst results. It may be due to the hyperparameters that were set for the model. The false positive and false negative predictions are also lower than the true positive and true negative predictions, which is a good sign for the model's performance.

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 mean 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 tables, Table 4, we present the evaluation metrics for the Logistic Regression and Multi-Layered Perceptron models on the Salary Prediction task:

Table 4 Evaluation Metrics for Logistic Regression and Multi-Layered Perceptron Models on Salary Prediction Dataset

Evaluation Metrics for Salary Prediction Dataset - Train set				
Model	Accuracy	Precision	Recall	F1-Score
Logistic Regression (Manual)	0.829353669	0.700867052	0.50494534	0.586989409
Logistic Regression (Scikit-Learn)	0.8371046	0.70794078	0.54763144	0.617552098
Multi-Layered Perceptron (Manual)	0.75984498	1.0	0.0	0.0
Multi-Layered Perceptron (Scikit-Learn)	0.91098887	0.85537918	0.75741801	0.80342352

Evaluation Metrics for Salary Prediction Dataset - Test set				
Model	Accuracy	Precision	Recall	F1-Score
Logistic Regression(Manual)	0.831	0.712250712	0.51334702	0.596658711
Logistic Regression (Scikit-Learn)	0.825	0.676092544	0.54004106	0.60045662
Multi-Layered Perceptron (Manual)	0.7565	1.0	0.0	0.0
Multi-Layered Perceptron (Scikit-Learn)	0.811	0.635235732	0.52566735	0.575280898

6. Conclusions

In this document, we explored the application of machine learning algorithms to Salary Prediction technique. We conducted a comprehensive analysis of the datasets, including data exploration, feature correlation analysis, and data preprocessing. We implemented two prominent machine learning algorithms, Logistic Regression and Multi-Layered Perceptron (MLP), to predict the outcomes of the task. We evaluated the models using various evaluation metrics, including accuracy, precision, recall, and F1-Score, to assess their performance.

The results of the evaluation metrics indicate that the models achieved vary-ing levels of performance on the prediction task. The Logistic Regression model performed well on the Salary Prediction dataset, achieving high accuracy and F1-Score values. The Multi-Layered Perceptron model also demonstrated strong performance on the Salary Prediction dataset, achieving high accuracy and F1-Score values.

– Salary Prediction: The MLP model achieves higher accuracy, precision, re-call, and F1-score on the training set compared to Logistic Regression. While its test set performance is slightly lower than its training set performance, it still demonstrates a competitive F1-score.

Therefore, based on these metrics, the Multi-Layered Perceptron (Scikit-Learn implementation) is the better algorithm for the Salary Prediction task due to its superior performance across multiple evaluation criteria.

Compliance with ethical standards

Disclosure of conflict of interest

No conflict of interest to be disclosed.

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] Bhatia MPS, Kumar A., "Information Retrieval & Machine Learning: Supporting Technologies for Web Mining Research & Practice", Webology, Vol. 5, No. 2.6. Metrics for Multi-Class Classification: An Overview

[5] York, D.; Brown, T. Salary Surveys. In *The Compensation Handbook: A State of the Art Guide to Compensation Strategy and Design*; Berger, L.A., Berger, D.R., Eds.; McGraw Hill: New York, NY, USA, 2008.

[6] Johnson, C.B.; Riggs, M.L.; Downey, R.G. Fun with Numbers: Alternative Models for Predicting Salary Levels. *Res. High. Educ.* 1987, *27*, 349–362.

[7] Popescu, Marius-Constantin & Balas, Valentina & Perescu-Popescu, Liliana & Mastorakis, Nikos. (2009). Multilayer perceptron and neural networks. WSEAS Transactions on Circuits and Systems.