Accredited SINTA 2 Ranking
Decree of the Director General of Higher Education, Research, and Technology, No. 158/E/KPT/2021
Validity period from Volume 5 Number 2 of 2021 to Volume 10 Number 1 of 2026

Published online at: **http://jurnal.iaii.or.id**

JURNAL RESTI
(Rekayasa Sistem dan Teknologi Informasi)
Vol. 9 No. 2 (2025) 266 - 275 e-ISSN: 2580-0760

Optimizing Multilayer Perceptron for Car Purchase Prediction with GridSearch and Optuna

Ginanti Riski[1*], Dedy Hartama[2], Solikhun[3]
[1,2,3]Department of Informatics Engineering, STIKOM Tunas Bangsa, Pematangsiantar, Indonesia
[1]ginantiriski@gmail.com, [2]dedyhartama@amiktunasbangsa.ac.id, [3]solikhun@amiktunasbangsa.ac.id

Abstract

Multilayer Perceptron (MLP) is a powerful machine learning algorithm capable of modeling complex, non-linear relationships, making it suitable for predicting car purchasing power. However, its performance depends on hyperparameter tuning and data quality. This study optimizes MLP performance using GridSearch and Optuna for hyperparameter tuning while addressing data imbalance with the Synthetic Minority Over-sampling Technique (SMOTE). The dataset comprises demographic and financial attributes influencing car purchasing power. Initially, the dataset exhibited class imbalance, which could lead to biased predictions; SMOTE was applied to generate synthetic samples, ensuring a balanced class distribution. Two hyperparameter tuning approaches were implemented: GridSearch, which systematically explores a predefined parameter grid, and Optuna, an adaptive optimization framework utilizing a Bayesian approach. The results show that Optuna achieved the highest accuracy of 95.00% using the Adam optimizer, whereas GridSearch obtained the best accuracy of 94.17% with the RMSProp optimizer, demonstrating Optuna's superior ability to identify optimal hyperparameters. Additionally, SMOTE significantly improved model stability and predictive performance by ensuring adequate class representation. These findings offer insights into best practices for optimizing MLP in predictive modeling. The combination of SMOTE and advanced hyperparameter tuning techniques is applicable to various domains requiring accurate predictive analytics, such as finance, healthcare, and marketing. Future research can explore alternative optimization algorithms and data augmentation techniques to further enhance model robustness and accuracy.

Keywords: *multilayer perceptron; hyperparameter optimization; gridsearch; optuna; SMOTE*

How to Cite: Ginanti Riski, Dedy Hartama, and Solikhun, "Optimizing Multilayer Perceptron for Car Purchase Prediction with GridSearch and Optuna ", *J. RESTI (Rekayasa Sist. Teknol. Inf.)* , vol. 9, no. 2, pp. 266 - 275, Mar. 2025.
DOI: https://doi.org/10.29207/resti.v9i2.6328

1. Introduction

In recent years, the use of *Machine Learning* (ML) algorithms, particularly *Multilayer Perceptron* (MLP), has become increasingly popular in various predictive applications. One of its applications is in predicting consumer purchasing power, especially in the automotive market. Vehicle purchasing power prediction plays an important role in the automotive industry for designing sales strategies, marketing, and product development. As a type of *Artificial Neural Network* (ANN), MLP has proven effective in handling prediction problems involving non-linear relationships between inputs and outputs, making it highly relevant in modeling purchasing power, which is complex and dynamic [1]-[3].

Technological advancements have driven various studies on the development and optimization of MLP for prediction, including in the automotive sector. One study showed that hyperparameter tuning significantly improved MLP performance, where the default Adam configuration achieved an accuracy of 89.50% and RMSProp of 87.50%. However, after tuning with a learning rate of 0.001, their accuracy increased to 91.5% and 92.00%, respectively [1].

Another study on predicting Toyota car sales in Indonesia found that the MLP 10-15-1 architecture provided the best results, with an MAE of 1879.29 and a MAPE of 6.78% [2]. Meanwhile, research on chili price prediction in Tangerang compared MLP and RNN, showing that MLP was more accurate, with a loss of 0.0038, MSE of 10,271,959.0, and MAPE of 3.79% [3]. These findings suggest that similar techniques can be applied to vehicle purchasing power prediction,

Received: 25-01-2025 | Accepted: 14-03-2025 | Published Online: 25-03-2025

given the data patterns involving economic factors and market trends [4].

Although MLP has shown promising results in various studies, several key challenges remain in the modeling process, particularly in hyperparameter optimization, data imbalance, and feature selection. One of the challenges in using MLP is hyperparameter optimization. Traditional techniques such as *GridSearch* are commonly used but require very long computation times and do not always produce optimal accuracy [1], [4]. Some studies have highlighted the limitations of *GridSearch* and proposed alternatives such as *Optuna*, which has proven to be more efficient in hyperparameter optimization through a faster and more accurate algorithmic search approach, as applied in heart disease prediction [5]-[8].

Additionally, data imbalance is a major challenge in many ML applications, including vehicle purchasing power prediction. This imbalance occurs when the number of samples in one category is significantly smaller than in another, which can lead to bias in the prediction model. One widely used solution is SMOTE (*Synthetic Minority Over-sampling Technique*), One way to enhance model performance is by generating synthetic data for the underrepresented class., thereby enhancing prediction accuracy [8], [9]. Further studies have also developed variations of SMOTE, such as *GeometricSMOTE*, which allows for faster and more efficient data balancing [10], [11]. Therefore, this study will explore the use of SMOTE to address data imbalance issues in vehicle purchasing power prediction.

Apart from data balancing, feature selection also plays a crucial role in improving MLP model quality. Effective feature selection techniques can help enhance model accuracy by reducing data dimensions and eliminating irrelevant features [4]. Other studies have also emphasized the importance of optimizing feature selection in neural networks to improve predictive performance [6]. Therefore, this study will apply correlation-based feature selection to improve the quality of input data used in MLP model training [12].

Previous research has mostly addressed individual aspects like hyperparameter optimization or data imbalance management, without adopting a holistic approach. Few studies have integrated advanced techniques such as Optuna for hyperparameter tuning, SMOTE for data balancing, and effective feature selection in vehicle purchasing power prediction. This highlights a research gap that needs to be filled. Hence, this study seeks to enhance prediction accuracy and efficiency by utilizing GridSearch and Optuna for hyperparameter optimization, SMOTE for handling data imbalance, and correlation-based feature selection to refine input data quality.

This study enhances vehicle purchasing power prediction accuracy, aiding the automotive industry in refining marketing and sales strategies. It is the first to integrate hyperparameter optimization, data balancing, and feature selection in this context.

2. Research Methods

2.1 Research Dataset

The research dataset used in this study is taken from the *Cars Purchase Decision Dataset* on *Kaggle*, which contains 1000 entries and five key features [13]:

The dataset includes *"User ID"***,** which is an integer that uniquely identifies each user; *"Gender"*, a categorical variable with values 'Male' or 'Female'; *"Age"*, an integer representing the user's age; *"Annual Salary"*, an integer indicating the user's yearly income; and *"Purchased"*, a binary variable where 0 indicates no purchase and 1 indicates a purchase decision.

This dataset predicts car purchase decisions based on Age, Gender, and Annual Salary. Preprocessing techniques, including oversampling with SMOTE and feature selection, are applied to handle class imbalance and improve model performance.

2.2 Research Stages

The research follows a structured workflow to ensure systematic handling of each step and improve model performance. Figure 1 illustrates the step-by-step process followed in this research.

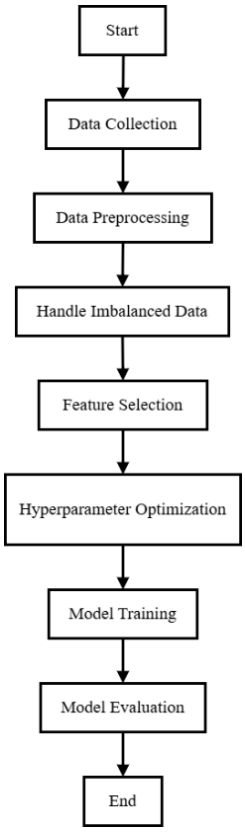

Figure 1. Research Stages

The dataset used in this study is sourced from *Kaggle* and consists of 1,000 data entries for analysis. To ensure data quality, an initial exploration is conducted to

identify and handle missing values, outliers, and duplicate records. Categorical features, such as Gender, are then converted into numerical values using label encoding. For optimal model performance, a feature selection process is applied to remove low-correlation features. Additionally, to address the class imbalance, the *Synthetic Minority Over-sampling Technique* (SMOTE) is implemented to generate synthetic data for the minority class. The dataset is subsequently split into training and test sets to facilitate model evaluation. Before training, feature standardization is performed to ensure that all features are on the same scale. A *Multilayer Perceptron* (MLP) model is then built and assessed using *accuracy*, *precision*, *recall*, and *F1-score* metrics. To further enhance model performance, hyperparameter tuning is conducted using *GridSearch* and *Optuna*. Finally, the model's performance is evaluated to measure the impact of hyperparameter tuning and ensure optimal predictive accuracy.

2.3 Multilayer Perceptron (MLP)

MLP is applied in this study as an *artificial neural network* to handle non-linear relationships between input and output variables, enabling the model to capture complex patterns in the data [14]. MLP consists of an input layer, hidden layers, and an output layer [15]. The training process uses optimization algorithms such as *Adam* or *RMSProp* with *backpropagation* to minimize model errors [16]. Following previous studies, the ReLU activation function is applied to the hidden layers, while the sigmoid function is used for the output layer [10], [15].

One of the key advantages of MLP is its ability to learn complex decision boundaries through the hierarchical representation of features. By adjusting the number of hidden layers and neurons, the model can capture intricate patterns that traditional machine-learning algorithms might struggle with. However, selecting the optimal network architecture is crucial, as an excessive number of hidden layers may lead to overfitting, while too few may result in underfitting. To address this, techniques such as dropout and batch normalization are commonly implemented to enhance generalization and improve model stability.

2.4 Oversampling with SMOTE

To handle class imbalance, SMOTE (*Synthetic Minority Over-sampling Technique*) is utilized. This technique generates synthetic data for the minority class by interpolating existing data points, enhancing model accuracy when working with imbalanced datasets. [7], [16], [17].

In addition to increasing the number of samples in the minority class, SMOTE helps the model recognize more diverse patterns compared to traditional oversampling methods, such as simple data duplication. However, SMOTE must be applied carefully, as excessive use can lead to overfitting. To mitigate this, SMOTE is often combined with undersampling techniques for the majority class or other regularization methods, such as dropout in neural networks [18], [19]. Furthermore, previous studies have shown that the effectiveness of SMOTE depends on the original data distribution and the appropriate selection of parameters, such as the number of neighbors used for synthetic data interpolation [20].

2.5 Hyperparameter Tuning with GridSearch and Optuna

GridSearch is employed to examine all possible hyperparameter combinations to identify the optimal one, though it can be time-intensive [21], [22]. As an alternative, *Optuna* is applied to efficiently search for optimal hyperparameters, reducing the search time compared to *GridSearch* [20]-[25].

In practice, *GridSearch* is best suited for smaller search spaces due to its exhaustive nature, whereas *Optuna* leverages efficient sampling techniques, such as *Tree-structured Parzen Estimator* (TPE) and pruning strategies, to focus on the most promising hyperparameter configurations. This makes *Optuna* particularly effective when working with deep learning models or complex machine learning pipelines that require extensive computational resources. Moreover, *Optuna* supports adaptive learning rates and parallel processing, further enhancing optimization efficiency. By combining both approaches, researchers can initially use *GridSearch* for a broad search and refine the results with *Optuna* for a more precise and efficient hyperparameter tuning process.

2.6 Model Evaluation

The model's performance is assessed using metrics like *Accuracy*, *Precision*, *Recall*, and *F1-Score* to evaluate its effectiveness in predicting car purchase decisions. *Accuracy* is determined using Formula 1:

$$Accuracy = \frac{(a+d)}{Total\ Sampel} \times 100\% \quad (1)$$

"a" represents the number of correct positive predictions, *"d"* is the number of incorrectly predicted positives, and *"Total Samples"* refers to the total number of tested data points.

Additionally, *Precision* is computed by dividing the number of true positive predictions (TP) by the total number of positive predictions (TP + FP), following Formula 2:

$$Precision = \frac{TP}{(TP+FP)} \quad (2)$$

Recall is determined by comparing the number of true positive predictions (TP) with the total actual positive cases (TP + FN), using Formula 3:

$$Recall = \frac{TP}{(TP+FN)} \quad (3)$$

Finally, *F1-Score* is calculated as the harmonic mean of Precision and Recall, formulated as Formula 4.

$$1 - Score = 2 \times \frac{(Precision \times Recall)}{(Precision + Recall)} \quad (4)$$

These metrics provide a comprehensive evaluation of the model's ability to balance correct predictions and misclassifications, offering deeper insight into its overall performance in making purchase decisions.

3. Results and Discussions

3.1 Data Collection

Once the data was collected, it was imported into the *Kaggle Notebook* for further processing and analysis. Figure 2 presents the dataset successfully loaded into *Kaggle Notebook*, displaying the first few rows along with their column names and values.

	User ID	Gender	Age	AnnualSalary	Purchased
0	385	Male	35	20000	0
1	681	Male	40	43500	0
2	353	Male	49	74000	0
3	895	Male	40	107500	1
4	661	Male	25	79000	0

Figure 2. Importing Data into Kaggle Notebook

From this view, we can confirm that the dataset has been correctly imported and structured, with clearly defined columns and data types. This preliminary verification confirms that the data is prepared for the next preprocessing steps, including handling categorical variables, scaling numerical values, and preparing it for model training. These steps will be discussed in the next section.

3.2 Data Exploration

To ensure data quality, several checks were performed in the data exploration phase, including missing value detection, outlier identification, and duplicate data removal. These steps are essential to maintaining dataset integrity and ensuring accurate model predictions. Figure 3 presents the results of these checks.:

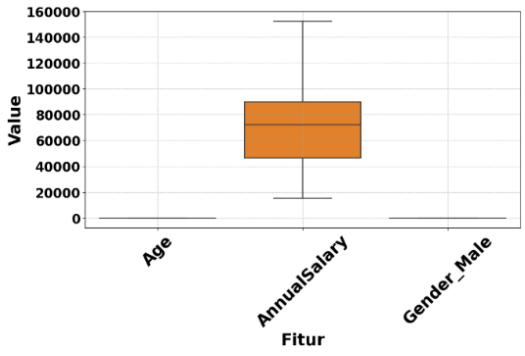

Figure 3. Data without Outliers

The analysis revealed that there were no missing values in the dataset, meaning that all records were complete and no imputation was required. Additionally, no duplicate entries were found, ensuring that the dataset does not contain redundant records that could bias the model's learning process.

Outlier detection was conducted using boxplots and the *Interquartile Range* (IQR) method. The IQR method identifies outliers by calculating the range between the *first quartile* (Q1) and the *third quartile* (Q3). Data points that fall below Q1 - 1.5IQR or above Q3 + 1.5IQR are regarded as outliers. The results confirmed that all feature values fall within an acceptable range, indicating that the dataset is free from extreme deviations.

Since the dataset does not contain missing values, duplicates, or significant outliers, no additional data cleaning or transformation was necessary at this stage. This ensures that the dataset is in optimal condition for further preprocessing, feature selection, and model training.

3.3 Label Encoding

At this stage, *Label Encoding* was applied to convert categorical features into a numerical format suitable for machine learning models. This transformation is necessary because most machine learning algorithms only accept numerical data.

In this dataset, the *Gender* column contained two categories: '*Male*' and '*Female*'. Using *Label Encoding*, the '*Male*' category was given a value of 1, and the '*Female*' category was assigned 0. This encoding enables the model to handle gender data efficiently without adding to the dimensionality.

Label Encoding was chosen over *One-Hot Encoding* because the feature has only two categories (*binary classification*). *One-Hot Encoding* is more suitable when dealing with categorical features with multiple unique values, as it prevents the model from interpreting numerical labels as ordinal relationships. However, since *Gender* is a non-ordinal variable with only two categories, *Label Encoding* is a more efficient and appropriate choice.

By applying this encoding, the dataset remains compact, avoids unnecessary feature expansion, and ensures that *machine learning* models can correctly interpret and utilize the *Gender* feature during training.

3.4 Feature Selection

The first step in *Feature Selection* was to analyze the correlation between features and the target variable (*Purchased*) to identify which features have significant relationships with the target. Pearson's correlation coefficient was used for this calculation, as it is a widely accepted method for measuring linear relationships between numerical variables.

To better understand the relationships between features and their impact on the target variable, an initial correlation analysis was conducted. Figure 4 presents

the Correlation Matrix, highlighting the strength of relationships among the features in the dataset.

From these results, features with a correlation greater than 0.1 with the Purchased column were selected for inclusion in the model. The threshold of 0.1 was chosen to ensure that only features with at least a weak to moderate correlation with the target variable were considered, helping to reduce noise in the dataset and improve model efficiency.

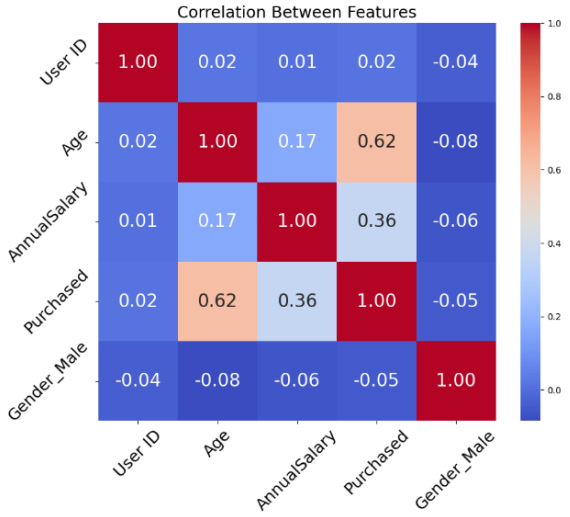

Figure 4. Initial Correlation Between Features

To prevent multicollinearity, a second correlation analysis was performed on the chosen features. *Multicollinearity* arises when independent variables are strongly correlated, causing redundancy and instability in the model. A high correlation (typically above 0.8) between independent features can cause inflated variance in the regression coefficients, reducing the model's interpretability and robustness.

The results of the heatmap visualization, which illustrate the correlation between the selected features, are presented in Figure 5.

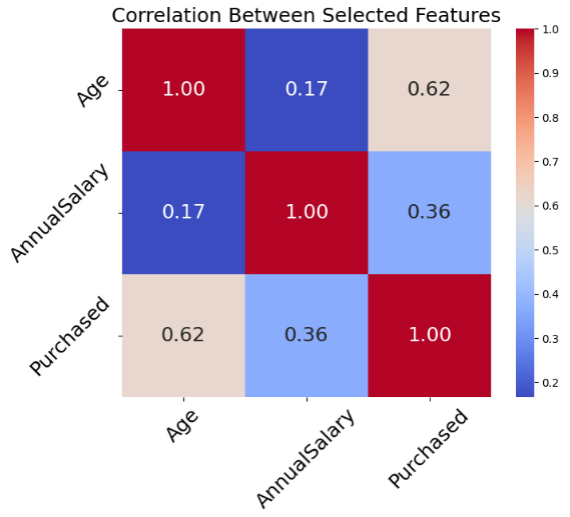

Figure 5. Correlation Between Features After Feature Selection

3.5 Oversampling with SMOTE

To tackle the class imbalance problem in the dataset, the *Synthetic Minority Over-sampling Technique* (SMOTE) was applied at this stage. SMOTE is a commonly used oversampling technique that creates synthetic samples for the minority class rather than just replicating existing data. This method helps avoid overfitting and improves the model's ability to generalize more effectively.

Before applying SMOTE, the class distribution in the *Purchased* column was imbalanced, with significantly more samples in Class 0 (*not purchasing a car*) than in Class 1 (*purchasing a car*). This imbalance can cause the model to favor the majority class, reducing its accuracy in correctly classifying the minority class. Figure 6 visualizes the class distribution before oversampling.

Figure 6. Class Distribution Before Oversampling

To resolve this imbalance, SMOTE was applied to generate synthetic samples for Class 1 until its count equaled that of Class 0. This process ensures that the model does not develop a bias toward the majority class, thereby improving its ability to classify both categories accurately. Figure 7 illustrates the class distribution after oversampling.:

Figure 7. Class Distribution After Oversampling

The class distribution before and after applying SMOTE is presented in Table 1, highlighting the increase in samples for the minority class:

Table 1. Data Distribution Before and After SMOTE Application

Class	Before SMOTE	After SMOTE
Class 0 (Not Purchased)	598	402
Class 1 (Purchased)	598	598

3.6 Split Data

At this point, the dataset, which had been resampled using SMOTE, was split into training and testing sets. This separation is crucial to ensure that the model learns from one portion of the data and is tested on a different portion that it has not encountered during training. This approach allows us to evaluate the model's ability to generalize to new, unseen data instead of merely memorizing the training data.

The data was split using the *train_test_split* method, with 80% allocated for training and 20% for testing. The 80-20 split is a commonly used ratio in *machine learning* because it provides the model with sufficient data for training while keeping a reasonable portion for evaluation. This ensures that the model has enough samples to learn from while preventing overfitting to the training data.

To maintain reproducibility, a random state was used during the split. This ensures that every time the split is performed, the data is divided in the same way, which is crucial for achieving consistent and comparable results across multiple runs.

3.7 Data Standardization

After the dataset was split, data standardization was performed to ensure all features were on a comparable scale. *Standardization* is crucial because many *Machine Learning* (ML) models, particularly those based on distance calculations (such as *logistic regression*, *neural networks*, and *support vector machines*), perform better when the input features have similar scales. Without *standardization*, features with larger numerical ranges could dominate and negatively affect the learning process.

The *StandardScaler* method was used for this purpose. *StandardScaler* adjusts the data so that each feature has a mean of 0 and a standard deviation of 1, ensuring that no single feature dominates others due to differences in magnitude. The transformation follows Formula 5.

$$X' = \frac{X - \mu}{\sigma} \quad (5)$$

X' represents the standardized value, X is the original feature value, μ denotes the mean of the feature, and σ is the standard deviation. By transforming the data in this manner, the model benefits from improved numerical stability and better convergence during training.

Standardization was applied to both the training and testing data, but the testing data was transformed using the scaling parameters calculated from the training data. This is necessary to prevent data leakage, ensuring that the test data remains unseen and does not influence the model's learning process.

3.8 Multilayer Perceptron (MLP) Model

At this stage, a *Multilayer Perceptron* (MLP) model was built and tested using three different optimizers: Adam, LBFGS, and RMSProp. These optimizers were chosen because they each have distinct characteristics and strengths, which can significantly impact model performance. The comparison between these optimizers helps determine which one provides the best accuracy, *precision*, *recall*, and *F1 score* for car purchase predictions.

The choice of optimizer is critical in influencing model performance. Adam (*Adaptive Moment Estimation*) integrates the benefits of momentum-based optimization techniques and adaptive learning rates, making it effective for non-stationary problems and deep networks with faster convergence. LBFGS (*Limited-memory Broyden-Fletcher-Goldfarb-Shanno*) It is a second-order optimization technique that approximates the *Hessian matrix*, offering stability for convex problems and faster convergence for small to medium datasets, though it is computationally expensive for large-scale deep learning tasks. RMSProp (*Root Mean Square Propagation*) adjusts learning rates dynamically by dividing the gradient by an exponentially decaying average of squared gradients, making it ideal for non-stationary objectives and *recurrent networks* like RNN and LSTM while providing more stability than Adam when handling noisy gradient updates.

The MLP model was trained using three different optimizers, each with specific hyperparameter settings, as shown in Table 2.

Table 2. Optimizer Parameters for Adam, LBFGS, and RMSProp

Optimization	Hidden Layer	Iterations	Learning Rate
Adam	100	500	0,1
Lbfgs	100	500	0,1
Rmsprop	100	50	0,001

After the model was trained, predictions were made, and its performance was evaluated using four key metrics. Accuracy measures the percentage of correct predictions across the entire test data, providing an overall assessment of model performance. *Precision* indicates how many of the predicted positive cases are actually correct, which is crucial in imbalanced datasets to avoid excessive false positives. *Recall* (Sensitivity) evaluates how well the model identifies actual positive cases, ensuring that the minority class is not overlooked. Finally, *F1-Score* is the harmonic mean of precision and recall, making it the best metric for assessing the balance between these two aspects in an imbalanced dataset.

To compare the performance of each optimizer, the results of the initial model evaluation are presented in Table 3.

Table 3. Initial Model Evaluation with Adam, LBFGS, and RMSProp Optimizers

	Model Evaluation			
Optimization	Accuracy	Precision	Recall	F1-Score
Adam	93,33%	90,60%	95,50%	92,98%
Lbfgs	90,42%	90,74%	88,29%	89,50%
Rmsprop	93,75%	92,11%	94,59%	93,33%

RMSProp achieved the highest accuracy (93.75%) and F1-score (93.33%), indicating the best balance between *precision* and *recall*, making it the most effective optimizer in this comparison. Adam followed closely with 93.33% accuracy but had the highest *recall* (95.50%), demonstrating its strength in identifying positive cases due to its adaptive learning rate, which efficiently adjusts to small gradient updates. LBFGS had the lowest accuracy (90.42%), likely because it is more suited for convex optimization problems and struggles to generalize well in complex, non-linear deep learning models. Thus, RMSProp appears to be the most effective optimizer for this model, though further tuning may enhance performance.

3.9 Hyperparameter Tuning

In this phase, Hyperparameter Tuning was conducted to identify the best parameter combinations that could enhance the performance of the MLP model. Tuning hyperparameters is an essential step in the development of a model, ensuring that the model can handle data variations effectively and improve overall predictive capability.

This study utilized two optimization methods: *GridSearch* and *Optuna*. Both methods were applied to three different optimizers: Adam, LBFGS, and RMSProp, to explore optimal parameter combinations that improve model accuracy. Each optimizer has unique characteristics and performance, which can significantly impact prediction results. Therefore, conducting experiments with various hyperparameter combinations is essential to achieve a more efficient and accurate model.

3.9.1 GridSearch

In this phase, the best parameters were determined using *GridSearchCV*. *GridSearch* performs an exhaustive search by evaluating all possible parameter combinations within a predefined range. This approach ensures that the best combination is found, but it has significant computational costs due to its brute-force nature. In this study, *GridSearch* was applied to the three optimizers (Adam, LBFGS, and RMSProp) with the following hyperparameters being optimized: hidden layer size, activation function, alpha regularization, and learning rate.

The *GridSearch* results for each optimizer are summarized in Table 4.

Table 4. Best Parameters with GridSearch

Optimizer	Best Parameters	Best Score
Adam	{'activation': 'logistic', 'alpha': 0.0001, 'hidden_layer_sizes': (100,), 'learning_rate_init': 0.1}	90,59%
LBFGS	{'activation': 'logistic', 'alpha': 0.1, 'hidden_layer_sizes': (100,), 'learning_rate_init': 0.1}	89,85%
RMSProp	{'batch_size': 32, 'epochs': 50, 'learning_rate': 0.01, 'hidden_units': 50}	94,58%

After hyperparameter tuning, model evaluation was conducted using *Accuracy*, *Precision*, *Recall*, and *F1-Score* to assess the performance of each optimizer. Table 5 presents the model evaluation results for each optimizer tested using *GridSearch*.

Table 5. Model Evaluation Results with GridSearch

Optimizer	Accuracy	Precision	Recall	F1-Score
Adam	93,75%	92,86%	93,69%	93,27%
LBFGS	92,92%	90,52%	94,59%	92,51%
RMSProp	94,17%	93,69%	93,69%	93,69%

3.9.2 Optuna

In this phase, hyperparameter optimization was performed using *Optuna*, which employs *Bayesian Optimization*, a more efficient approach compared to traditional *GridSearch*. *Optuna* uses probabilistic models to intelligently explore the hyperparameter space instead of exhaustively testing all combinations.

Unlike *GridSearch*, which performs *brute-force* searching, *Optuna* dynamically adjusts its search based on prior trials, allowing it to converge to optimal parameters faster and with less computational cost.

In this study, *Optuna* was used to fine-tune several key hyperparameters, including *hidden_layer_size*, *learning_rate_init*, *max_iter*, *epochs*, and *batch_size*. These hyperparameters play a critical role in model performance, and their optimal configuration can lead to significantly improved predictive accuracy.

Table 6 summarizes the hyperparameter tuning results obtained using *Optuna*.

Table 6. Best Parameters with Optuna

Optimizer	Best Parameters	Best Score
Adam	{'hidden_layer_size': 168, 'learning_rate_init': 0.00083, 'max_iter': 562}	95,00%
LBFGS	{'hidden_layer_size': 50, 'learning_rate_init': 0.000191, 'max_iter': 720}	93,33%
RMSProp	{'hidden_layer_size': 186, 'learning_rate': 0.0039, 'epochs': 39, 'batch_size': 64}	94,58%

After optimization, the model was re-evaluated, and the results are summarized in Table 7.

Table 7. Model Evaluation Results with Optuna

Optimizer	Accuracy	Precision	Recall	F1-Score
Adam	95,00%	94,59%	94,59%	94,59%
LBFGS	93,33%	92,79%	92,79%	92,79%
RMSProp	94,58%	93,75%	94,59%	94,17%

3.10 Model Evaluation

The results of the *Optuna-based* model evaluation indicate that the Adam optimizer achieved the best performance among all tested optimizers. The confusion matrix obtained after tuning the Adam optimizer with *Optuna* is presented in Figure 7.

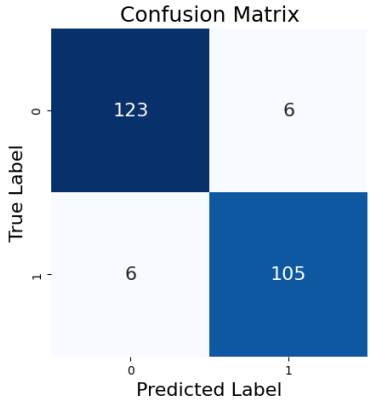

Figure 7. Confusion Matrix for Adam Optimizer with Optuna

In the *GridSearch* model evaluation, RMSProp outperformed the other optimizers. The confusion matrix obtained after tuning RMSProp with *GridSearch* is presented in Figure 8.

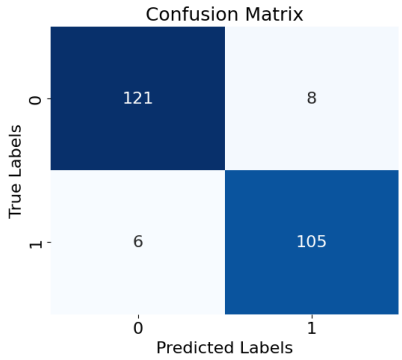

Figure 8. Confusion Matrix for Adam Optimizer with GridSearch

By implementing *GridSearch* and *Optuna*, this study successfully explored more optimal hyperparameter combinations for the *Multilayer Perceptron* (MLP) model.

Optuna outperforms *GridSearch* in computational efficiency by utilizing *Bayesian Optimization* to direct hyperparameter searches toward more promising areas. In contrast, *GridSearch* requires significantly more time as it exhaustively tests all possible combinations using a *brute-force* approach, which increases computational costs. On the other hand, *Optuna* accelerates the process by leveraging previous trial results to focus on the most promising hyperparameters. Additionally, the study results indicate that Adam consistently delivers the best performance compared to LBFGS and RMSProp. This is due to Adam's ability to adjust the learning rate adaptively, making it more effective in handling gradient variations and improving model training stability.

3.11 Research Contribution

This study provides a valuable contribution to the optimization of *Multilayer Perceptron* (MLP) for kidney disease classification by adopting a more comprehensive approach compared to previous research. One of the key contributions is the comparative analysis between *GridSearch* and *Optuna* for hyperparameter tuning, demonstrating that *Optuna* is more efficient and achieves higher accuracy than *GridSearch*. Unlike prior studies that relied on a single tuning method, this research provides a direct comparison of both approaches in medical data classification, offering deeper insights into the effectiveness of each method.

Additionally, this study integrates multiple model optimization techniques, such as SMOTE for handling imbalanced data and correlation-based feature selection to improve input quality. This combination has rarely been explored in kidney disease classification, making this research one of the first to integrate these three approaches in a single study. The results also indicate that MLP optimized with *Optuna*, combined with SMOTE and feature selection, achieves superior *accuracy* and *F1-score* compared to previous methods. This confirms that effective hyperparameter tuning can significantly enhance disease classification performance, which is crucial for early diagnosis and treatment planning.

Furthermore, the methodology developed in this study is not limited to kidney disease classification but also has the potential to be applied to other medical classification tasks, such as heart disease and diabetes prediction. Thus, this research not only improves MLP accuracy for kidney disease classification but also provides a foundation for developing similar models in broader healthcare analytics.

4. Conclusions

This study successfully demonstrated that *Multilayer Perceptron* (MLP) optimized with hyperparameter tuning techniques can significantly improve accuracy in predicting car purchase decisions. By utilizing *GridSearch* and *Optuna*, the best hyperparameter combinations were identified, where RMSProp achieved the highest performance in *GridSearch* with an accuracy of 94.17%, while Adam achieved the highest accuracy of 95.00% in *Optuna*. Compared to other optimizers, LBFGS reached an accuracy of 92.92% with *GridSearch* and 93.33% with *Optuna*, while RMSProp achieved 94.58% with *Optuna*. These results indicate that Adam performs better with *Optuna*, whereas RMSProp demonstrated the best performance within *GridSearch*. Additionally, this study shows that *Optuna* is more efficient than *GridSearch* in hyperparameter tuning. With *Optuna*, an accuracy of 95.00% was achieved using 562 iterations and a learning rate of 0.00083, whereas *GridSearch* required significantly more computational time due to its

exhaustive search of multiple parameter combinations. This confirms that *Optuna* is not only faster in finding optimal parameters but also yields a more accurate model. The application of SMOTE successfully addressed data imbalance, raising the number of minority class samples from 402 to 598, resulting in a more stable and accurate model for predicting car purchase decisions. Correlation-based feature selection also contributed to improving input data quality, which ultimately enhanced the model's predictive performance. Although the results are promising, this study has several limitations. The hyperparameter tuning process, especially with *GridSearch*, remains computationally expensive, as it requires testing a vast number of parameter combinations. Additionally, the dataset used had a limited number of features, which means the model's ability to fully capture the complexity of real-world purchasing behavior may not be fully optimized. Furthermore, this study did not evaluate the model's performance on real-time datasets or with additional, more dynamic variables, which could enhance predictive accuracy in more complex environments. For future research, it is recommended to use a larger and more diverse dataset, incorporating external factors such as consumer preferences, economic indicators, and historical purchase data to improve model robustness. Additionally, exploring *ensemble learning* techniques such as *Random Forest* or *XGBoost* could provide valuable comparisons to the MLP model. Testing the model in real-time scenarios or with continuously updated datasets would also enhance validation and improve its applicability in dynamic environments. Furthermore, integrating multiple hyperparameter optimization methods, such as combining *GridSearch* with *Bayesian Optimization*, could be explored to achieve even more optimal results. With these future directions, upcoming research is expected to enhance the efficiency and accuracy of car purchase prediction models, providing valuable insights for further studies and industry applications.

References

[1] M. Iqbal, M. N. Hendri, M. R. Ramadhan Saelan, M. Sony Maulana, Yudhistira, and A. Mustopa, "Optimasi Hyperparameter Multilayer Perceptron Untuk Prediksi Daya Beli Mobil," *J. Manaj. Inform. dan Sist. Inf.*, vol. 6, no. 1, pp. 73–81, 2023, doi: 10.36595/misi.v6i1.739.

[2] M. N. Afkar, D. T. Randa, and R. A. Saputra, "Prediksi Penjualan Mobil Toyota Di Indonesia Menggunakan Multi-Layer Perceptron," *J. Inform. Polinema*, pp. 91–98, 2024.

[3] K. H. Suradiradja, "Algoritme Machine Learning Multi-Layer Perceptron dan Recurrent Neural Network untuk Prediksi Harga Cabai Merah Besar di Kota Tangerang," *Fakt. Exacta*, vol. 14, no. 4, pp. 194–205, 2021, doi: 10.30998/faktorexacta.v14i4.10376.

[4] M. Wahyuni, "Klasifikasi Penyakit Daun Tomat dengan Perbandingan Fungsi Aktivasi Multi Layer Perceptron," *Minfo Polgan*, vol. 13, no. 2, pp. 1988–1998, 2024, doi: https://doi.org/10.33395/jmp.vl3i2.14351.

[5] Z. N. Nugroho, A. Harjoko, and M. Auzan, "Klasifikasi Eritrosit Pada Thalasemia Minor Menggunakan Fitur Konvolusi dan Multi-Layer Perceptron," *Indones. J. Electron. Instrum. Syst.*, vol. 13, no. 1, pp. 91–100, 2023, doi: 10.22146/ijeis.83473.

[6] L. A. Zahir and S. Mafiroh, "Metode Multi Layer Perceptron (Optimization Of Concrete Compressive Strength With Artificial Neural Networks Using Multi Layer Perceptron)," *Tek. Sipil Univ. Tulungagung*, vol. 4, no. 1, pp. 45–55, 2024.

[7] P. I. Ashuri, I. A. Cahyani, and C. S. K. Aditya, "MIND (Multimedia Artificial Intelligent Networking Database) Klasifikasi Penyakit Stunting Menggunakan Algoritma Multi-Layer Perceptron," *Multimed. Artif. Intell. Netw. Database*, vol. 9, no. 1, pp. 52–63, 2024, doi: 10.26760/mindjournal.v9i1.52-63.

[8] D. Dablain, B. Krawczyk, and N. V. Chawla, "DeepSMOTE: Fusing Deep Learning and SMOTE for Imbalanced Data," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 34, no. 9, pp. 6390–6404, 2023, doi: 10.1109/TNNLS.2021.3136503.

[9] A. Ranggana, R. C. Putra, and W. Wahyudin, "Multilayer Perceptrons Dalam Memprediksi Kemenangan Pertandingan Sepak Bola UEFA EURO 2016," *Digit. Transform. Technol.*, vol. 3, no. 2, pp. 629–642, 2023, doi: 10.47709/digitech.v3i2.3123.

[10] L. Camacho, G. Douzas, and F. Bacao, "Geometric SMOTE for regression," *Expert Syst. Appl.*, vol. 193, p. 116387, 2022, doi: 10.1016/j.eswa.2021.116387.

[11] M. R. Aohana, F. Bimantoro, R. N. L. Hidhayah, and D. Swanjaya, "Komparasi Algoritma MLP+LBP dan CNN Sebagai solusi Inovatif Untuk Deteksi Dini Korosi," in *Seminar Nasional Inovasi Teknologi*, Kediri, 2024, pp. 528–536. [Online]. Available: https://proceeding.unpkediri.ac.id/index.php/inotek/

[12] D. Wintana, Gunawan, H. Sulaeman, and S. Bahri, "Penerapan Multi Layer Perceptron dan Diskrit pada Prediksi Cacat Software," *J. Inf. Technol.*, pp. 321–329, 2022.

[13] S. Gabriel, "Cars - Purchase Decision Dataset," 2022, *Kaggle*. [Online]. Available: https://www.kaggle.com/datasets/gabrielsantello/cars-purchase-decision-dataset

[14] D. Mardinah and M. Thoriq, "Algoritma Multi Layer Perceptron sebagai Prediksi Kelulusan Mahasiswa Universitas Adzkia Tepat Waktu berdasarkan jenis kelamin , Indeks Prestasi Semester , dan Jumlah SKS," *J. Technol. Comput.*, vol. 1, no. 2, pp. 26–35, 2024, [Online]. Available: https://http//ojs.adzkia.ac.id/index.php/jtech

[15] F. Penalun, A. Hermawan, D. Avianto, and A. Pramudwiatmoko, "A Multi-Layer Perceptron Regression and Variant Windowing for Estimating Rainfall Based on Weather Radar Data," *J. Educ. Sci.*, vol. 33, no. 2, pp. 58–71, 2024, doi: 10.33899/edusj.2024.146355.1421.

[16] A. Gupta, R. Ramanath, J. Shi, and S. S. Keerthi, "Adam vs. SGD: Closing the generalization gap on image classification," in *OPT2021: 13th Annual Workshop on Optimization for Machine Learning*, 2021, pp. 1–7.

[17] Y. Elor and H. Averbuch-Elor, "To SMOTE, or not to SMOTE?," *Proc. ACM Int. Conf. Inf. Knowl. Manag.*, vol. 1, no. 1, 2022, [Online]. Available: http://arxiv.org/abs/2201.08528

[18] M. M. Ahsan, M. S. Ali, and Z. Siddique, "Imbalanced Class Data Performance Evaluation and Improvement using Novel Generative Adversarial Network-based Approach: SSG and GBO," no. Ml, pp. 1–13, 2022, [Online]. Available: http://arxiv.org/abs/2210.12870

[19] J. H. Joloudari, A. Marefat, M. A. Nematollahi, S. S. Oyelere, and S. Hussain, "Effective Class-Imbalance Learning Based on SMOTE and Convolutional Neural Networks," *Appl. Sci.*, vol. 13, no. 6, pp. 1–43, 2023, doi: 10.3390/app13064006.

[20] S. Wang, Y. Dai, J. Shen, and J. Xuan, "Research on expansion and classification of imbalanced data based on SMOTE algorithm," *Sci. Rep.*, vol. 11, no. 1, pp. 1–11, 2021, doi: 10.1038/s41598-021-03430-5.

[21] J. Taylor, W. Wang, B. Bala, and T. Bednarz, "Optimizing the optimizer for data driven deep neural networks and physics informed neural networks," 2022, [Online]. Available: http://arxiv.org/abs/2205.07430

[22] M. Juez-Gil, Á. Arnaiz-González, J. J. Rodríguez, C. López-Nozal, and C. García-Osorio, "Approx-SMOTE: Fast SMOTE for Big Data on Apache Spark," *Neurocomputing*, vol. 464, pp. 432–437, 2021, doi: 10.1016/j.neucom.2021.08.086.

[23] N. K. C. Pratiwi, N. Ibrahim, and S. Saidah, "Prediksi Kanker Paru menggunakan Grid search untuk Optimasi Hyperparameter pada Algoritma MLP dan Logistic Regression," *J. Tek. Energi Elektr. Tek. Telekomun. Tek. Elektron.*, vol. 12, no. 3, p. 556, 2024, doi: 10.26760/elkomika.v12i3.556.

[24] B. Fu *et al.*, "Quantifying scattering characteristics of mangrove species from Optuna-based optimal machine learning classification using multi-scale feature selection and SAR image time series," *Int. J. Appl. Earth Obs. Geoinf.*, vol. 122, p. 103446, 2023, doi: 10.1016/j.jag.2023.103446.

[25] W. A. G. Kodri and S. Hadianti, "Optimization of The Machine Learning Approach using Optuna in Heart Disease Prediction," *J. Med. Informatics Technol.*, vol. 1, no. 3, pp. 59–64, 2023, doi: 10.37034/medinftech.v1i3.15.

