JUSIKOM PRIMA (Jurnal Sistem Informasi dan Ilmu Komputer Prima)
Vol. 7 No. 1, August 2023 E-ISSN : 2580-2879

COMPARATIVE ANALYSIS OF STROKE CLASSIFICATION USING THE K-NEAREST NEIGHBOR, DECISION TREE, AND MULTILAYER PERCEPTRON METHODS

Ertina Sabarita Barus*[*1], Jenny Evans Halim[2], Sally Yessica[3]
[1,2,3]Universitas Prima Indonesia
Jl.Sampul No.3, Sei Putih Bar., Kec. Medan Petisah, Medan City, North Sumatra 20118
Email : baruschild2@gmail.com*

ABSTRACT- Stroke has become a serious health problem; the leading cause of stroke is usually a blood clot in the arteries that supply blood to the brain. Strokes can also be caused by bleeding when blood vessels burst and blood leaks into the brain. In one year, about 12.2 million people will have their first stroke, and 6.5 million people will die from a stroke. More than 110 million people worldwide have had a stroke. Handling that is done quickly can minimize the level of brain damage and the potential adverse effects. Therefore, it is essential to predict whether a patient has the potential to experience a stroke. The K-Nearest Neighbor, Decision Tree, and Multilayer Perceptron algorithms are applied as a classification method to identify symptoms in patients and achieve an optimal accuracy level. The results of making the three algorithms are pretty good, where K-Nearest Neighbor (K-NN) has an accuracy value of 93.84%, Decision Tree is 93.97%, and Multilayer Perceptron (MLP) is 93.91%. The best accuracy value is the Decision Tree algorithm with an accuracy difference of no more than 0.10% with the two algorithms used.

KEYWORDS: K-Nearest Neighbor, Decision Tree, Multilayer Perceptron, Stroke

1. INTRODUCTION

A stroke is when blood flow to the brain stops, which results in a lack of oxygen, damages the brain and causes the loss of various vital functions [1], [2]. The leading cause of a stroke is usually a blood clot in an artery that carries blood to the brain. Strokes can also be caused by bleeding when blood vessels burst and blood leaks into the brain [3], [4]. The consequences of a stroke can have permanent effects, including partial paralysis and disturbances in speech, comprehension, and memory. The brain area affected and the length of time the blood flow is stopped influence the type and severity of damage [5], [6].

Worldwide, stroke has become a severe health problem. One in four adults over 25 will have a stroke in their lifetime. In one year, about 12.2 million people will have their first stroke, and 6.5 million people will die from a stroke. More than 110 million people worldwide have had a stroke [7]. Although the risk of stroke increases with age, more than 60% of stroke cases occur in people under 70 and 16% under 50 years [8].

Handling that is done quickly can minimize the level of brain damage and the potential adverse effects. Therefore, it is imperative to predict whether a patient has the potential to experience a stroke. One strategy for predicting stroke is through the classification method. Classification related to stroke conditions is an essential element in accurately predicting the potential for stroke. [9]

The K-Nearest Neighbor, Decision Tree, and Multilayer Perceptron algorithms are applied as a classification method to identify symptoms in patients, aiming to achieve an optimal level of

accuracy [10]. In this study, a comparison will be made to determine which method has a higher level of accuracy in the classification process. This study uses three classification algorithms: the K-Nearest Neighbor, Decision Tree, and Multilayer Perceptron.

Stroke classification analysis has been carried out by [11], [12], [13] using several classification methods such as K-Nearest Neighbor, Gaussian Naive Bayes, Decision Tree Algorithm C.45, Multilayer Perceptron where several researchers carry out the results of the classification before still getting an accuracy value below 90%, therefore it is necessary to develop it to get an even better accuracy value.

Based on the explanation above, the researcher will research stroke titled "Comparative Analysis of Stroke Classification Using the K-Nearest Neighbor, Decision Tree, and Multilayer Perceptron Methods."

2. RESEARCH METHODS

2.1 Research Methodology

Stroke is a devastating medical disorder that ranks among the leading killers and causes of morbidity in the world. To increase the patient's chances of recovery and reduce the danger of complications, it is imperative to identify and diagnose stroke accurately and early [14]. The classification method is a strategy that can be applied in disease detection and diagnosis.

Famous classification approaches widely applied in many domains, including health, are the K-Nearest Neighbor (K-NN), Decision Tree, and Multilayer Perceptron (MLP) methods. Each approach to stroke diagnosis has advantages and disadvantages and may be more or less accurate.

K-Nearest Neighbor is the only method that can be used to classify data by looking for information that relates to specific evidence according to the number of related neighbors analyzed using K [15].

One technique for classifying data is a decision tree. A tree with root, internal, and terminal nodes forms a decision tree model. Leaf nodes are class labels, while roots and inner nodes are variables or characteristics. The data query will follow the root node and internal node until it reaches the leaf node when it performs classification. Based on the inner node labels, query the data layer labels. Data with specific feature values are used in conventional decision trees [16].

Multilayer Perceptron (MLP) is an artificial neural network that operates on a feed-forward basis and consists of one or more hidden layers. The MLP architecture consists of an initial layer of neurons responsible for receiving input data, followed by one or more hidden layers of computational neurons, and finally, an output layer containing neurons that store the computational results. 17].

The three algorithms used in this study for data analysis and prediction are K-Nearest Neighbor (K-NN), Decision Tree, and Multilayer Perceptron (MLP). The K-Nearest Neighbors (K-NN) algorithm is a direct approach that relies on the proximity of neighboring data points to make predictions. In contrast, the Decision Tree algorithm uses a hierarchical structure to handle various data types and offers high interpretability. In contrast, the Multilayer Perceptron (MLP) is a specific variant of artificial neural networks characterized by a layered structure of interconnected neurons, which makes it highly suitable for dealing with complex and non-linear associations in data sets.

2.2 Research Stages

In carrying out a classification analysis, stroke detection has several stages. The stages carried out by researchers in conducting research can be seen in Diagram 1 below.

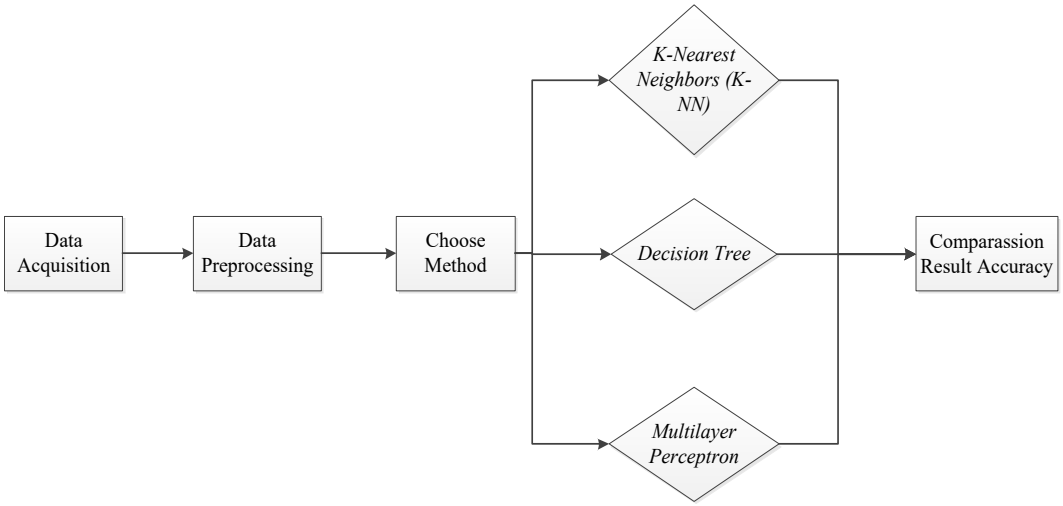

Figure 1 Research Stages Diagram

1. Data Acquisition
   The Data Acquisition Phase involves collecting data from reliable sources, which are then processed and used to develop Machine Learning models. The process ends by calling Data Frame in Google Colaboratory. The dataset used in this study is sourced from [18], specifically from the repository "Brain Stroke Dataset" by Jillani Soft Tech [19].
2. Data Preprocessing
   Preprocessing involves optimizing data quality, mitigating the impact of noise or outliers, dealing with missing or incomplete data, and adapting data formats to align with algorithm or model requirements.
3. Choose Methode
   At this stage, classification algorithms are selected in machine learning, which are 3 K-Nearest Neighbor (K-NN), Decision Tree, and Multilayer Perceptron (MLP).
4. Comparison Result Accuracy
   After selecting the model, testing the training data, and testing the data for each algorithm model, the last step is to compare the values of each algorithm to see which algorithm is the best in classifying stroke predictions.

3. RESULTS AND DISCUSSION

3.1 Analysis of Research Problems

Stroke is a significant health problem resulting in long-term damage and death. Therefore, it is essential to have the ability to quickly recognize and diagnose this disease to provide appropriate and effective treatment and care. This study aims to analyze several problems, which are the main focus of the investigation. This study aims to evaluate the performance of three classification methods, namely K-Nearest Neighbor (K-NN), Decision Tree, and Multilayer Perceptron (MLP), in classifying stroke patients accurately. This classification is based on available medical data.

3.2 Stages of Data Processing

Figure 2 illustrates the flow of data processing, starting with data acquisition, where collecting data from the Kaggle dataset provider website; the second stage is the preparation / preprocessing process to prepare the training model. After that, the Exploratory Data Analysis (EDA) stage was carried out to visualize the dataset. After exploratory data analysis (EDA) has been carried out, the next step is Data splitting/Partitioning to separate test data from training data. Model Building is carried out in the next stage, where classification algorithm models such as K-Nearest Neighbor (K-NN), Decision Tree, and Multilayer Perceptron (MLP) are used. And in the final

stage, a comparison of the accuracy values of each method used is carried out.

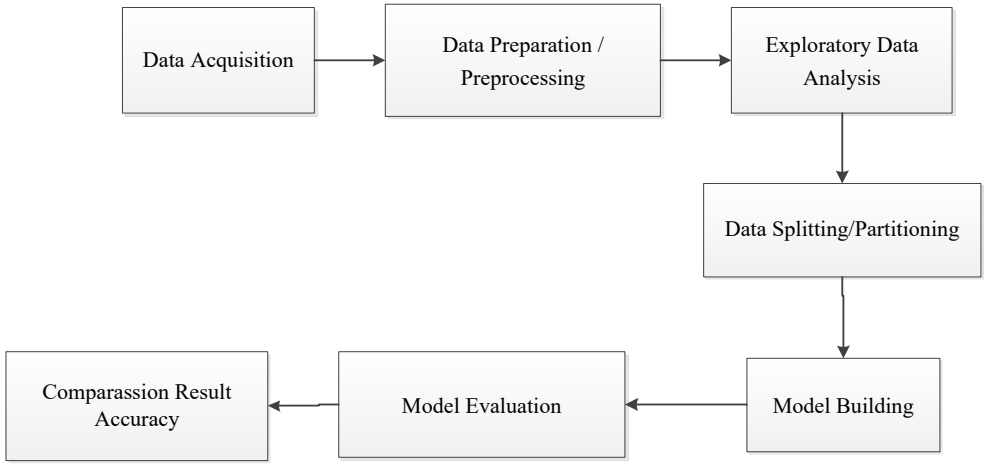

Figure 2 Research Stages Diagram

3.2.1. Data Acquisition

Stroke is a significant global health problem that affects many individuals worldwide. For this data analysis, researchers used a dataset from [18], which includes 4980 rows of data and ten columns related to stroke examination. The following are the steps that have been taken for data analysis:

```
# MENGAKSES DATA DARI DRIVE KE COLAB
df = pd.read_csv("/content/drive/MyDrive/Dataset/BRAINSTROKE/brain_stroke.csv")
df.head(5000)
```

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	Male	67.0	0	1	Yes	Private	Urban	228.69	36.6	formerly smoked	1
1	Male	80.0	0	1	Yes	Private	Rural	105.92	32.5	never smoked	1
2	Female	49.0	0	0	Yes	Private	Urban	171.23	34.4	smokes	1
3	Female	79.0	1	0	Yes	Self-employed	Rural	174.12	24.0	never smoked	1
4	Male	81.0	0	0	Yes	Private	Urban	186.21	29.0	formerly smoked	1
...	...	...	...	...	...	...	...	...	...	...	...
4976	Male	41.0	0	0	No	Private	Rural	70.15	29.8	formerly smoked	0
4977	Male	40.0	0	0	Yes	Private	Urban	191.15	31.1	smokes	0
4978	Female	45.0	1	0	Yes	Govt_job	Rural	95.02	31.8	smokes	0
4979	Male	40.0	0	0	Yes	Private	Rural	83.94	30.0	smokes	0
4980	Female	80.0	1	0	Yes	Private	Urban	83.75	29.1	never smoked	0

4981 rows × 11 columns

Figure 3. Stroke Patient Dataset

3.2.2 Data Preparation/Preprocessing

The data that has been acquired will be prepared/preprocessed. Preparing related data for machine learning model construction and training is called preprocessing. Data must also be converted into a format that will be processed in the model. Preprocessing also improves data quality, and the steps are carried out as follows:

1. **Data Manipulation**

   At the data manipulation stage, changes are made to the name of each column which we can see in Figure 4 below:

```
df.rename(columns={"gender   " : "JenisKelamin",
                   "age" : "Usia",
                   "hypertension" : "TekananDarahTinggi",
                   "heart_disease" : "PenyakitJantung",
                   "ever_married" : "StatusPernikahan",
                   "work_type" : "StatusPerkerjaan",
                   "Residence_type" : "JenisTempatTinggal",
                   "avg_glucose_level" : "TingkatGlukosa",
                   "bmi" : "BodyMassIndex",
                   "smoking_status" : "TingkatPerokok",
                   "stroke" : "Stroke",}, inplace=True)
df.head(50000)
```

	gender	Usia	TekananDarahTinggi	PenyakitJantung	StatusPernikahan	StatusPerkerjaan	JenisTempatTinggal	TingkatGlukosa	BodyMassIndex	TingkatPerokok	Stroke
0	Male	67.0	0	1	Yes	Private	Urban	228.69	36.6	formerly smoked	1
1	Male	80.0	0	1	Yes	Private	Rural	105.92	32.5	never smoked	1
2	Female	49.0	0	0	Yes	Private	Urban	171.23	34.4	smokes	1
3	Female	79.0	1	0	Yes	Self-employed	Rural	174.12	24.0	never smoked	1
4	Male	81.0	0	0	Yes	Private	Urban	186.21	29.0	formerly smoked	1
...	...	...	...	...	...	...	...	...	...	...	...
4976	Male	41.0	0	0	No	Private	Rural	70.15	29.8	formerly smoked	0
4977	Male	40.0	0	0	Yes	Private	Urban	191.15	31.1	smokes	0
4978	Female	45.0	1	0	Yes	Govt_job	Rural	95.02	31.8	smokes	0
4979	Male	40.0	0	0	Yes	Private	Rural	83.94	30.0	smokes	0
4980	Female	80.0	1	0	Yes	Private	Urban	83.75	29.1	never smoked	0

Figure 4 Data Manipulation

2. **Duplicate Removal**

This stage deletes data that has duplicate or multiple values, and we can see the elimination of same data in Figure 5 below:

```
[9]  df=df.drop_duplicates()
     df.duplicated()

     0       False
     1       False
     2       False
     3       False
     4       False
             ...
     4976    False
     4977    False
     4978    False
     4979    False
     4980    False
     Length: 4981, dtype: bool
```

Figure 5 Duplicate Removal

3. **Feature Engineering**

Feature engineering refers to systematic procedures for selecting, manipulating, and transforming unprocessed data. To perform Feature Engineering, the LabelEncoder library from Scikit Learn is used. Feature engineering is applied to the gender column and smoker level, which we can see in Figure 3.5 below:

```
# IMPORT LIBRARY LABEL ENCODING (OBJECT -> INT)
from sklearn.preprocessing import LabelEncoder
```

```
# LABEL ENCODING MASING MASING DATA KATEGORIKAL
le1 = LabelEncoder()
df['gender'] =le1.fit_transform(df['gender'])
le2 = LabelEncoder()
df['Stroke'] =le2.fit_transform(df['Stroke'])
df.head(5000)
```

	gender	Usia	TekananDarahTinggi	PenyakitJantung	StatusPernikahan	StatusPerkerjaan	JenisTempatTinggal	TingkatGlukosa	BodyMassIndex	TingkatPerokok	Stroke
0	1	67.0	0	1	Yes	Private	Urban	228.69	36.6	formerly smoked	1
1	1	80.0	0	1	Yes	Private	Rural	105.92	32.5	never smoked	1
2	0	49.0	0	0	Yes	Private	Urban	171.23	34.4	smokes	1
3	0	79.0	1	0	Yes	Self-employed	Rural	174.12	24.0	never smoked	1
4	1	81.0	0	0	Yes	Private	Urban	186.21	29.0	formerly smoked	1
...	...	...	...	...	...	...	...	...	...	...	...
4976	1	41.0	0	0	No	Private	Rural	70.15	29.8	formerly smoked	0
4977	1	40.0	0	0	Yes	Private	Urban	191.15	31.1	smokes	0
4978	0	45.0	1	0	Yes	Govt_job	Rural	95.02	31.8	smokes	0
4979	1	40.0	0	0	Yes	Private	Rural	83.94	30.0	smokes	0
4980	0	80.0	1	0	Yes	Private	Urban	83.75	29.1	never smoked	0

4981 rows × 11 columns

```
# LABEL ENCODING MASING MASING DATA KATEGORIKAL
le1 = LabelEncoder()
df['TingkatPerokok'] =le1.fit_transform(df['TingkatPerokok'])
le2 = LabelEncoder()
df['Stroke'] =le2.fit_transform(df['Stroke'])

df.head(5000)
```

	gender	Usia	TekananDarahTinggi	PenyakitJantung	StatusPernikahan	StatusPerkerjaan	JenisTempatTinggal	TingkatGlukosa	BodyMassIndex	TingkatPerokok	Stroke
0	1	67.0	0	1	Yes	Private	Urban	228.69	36.6	1	1
1	1	80.0	0	1	Yes	Private	Rural	105.92	32.5	2	1
2	0	49.0	0	0	Yes	Private	Urban	171.23	34.4	3	1
3	0	79.0	1	0	Yes	Self-employed	Rural	174.12	24.0	2	1
4	1	81.0	0	0	Yes	Private	Urban	186.21	29.0	1	1
...	...	...	...	...	...	...	...	...	...	...	...
4976	1	41.0	0	0	No	Private	Rural	70.15	29.8	1	0
4977	1	40.0	0	0	Yes	Private	Urban	191.15	31.1	3	0
4978	0	45.0	1	0	Yes	Govt_job	Rural	95.02	31.8	3	0
4979	1	40.0	0	0	Yes	Private	Rural	83.94	30.0	3	0
4980	0	80.0	1	0	Yes	Private	Urban	83.75	29.1	2	0

4981 rows × 11 columns

Figure 6 Feature Enginereeng

4. Unused column removal

The final stage in data preprocessing is deleting columns not used to classify the model to be applied. The columns to be deleted are Marital Status, Employment Status, and Type of Residence. The following stages of unused column removal can be seen in Figure 7 below:

```
# TERDAPAT KOLOM YANG TIDAK MEMPENGARUHI KLASIFIKASI SEPERTI KOLOM NO DAN NAMA
df.drop(['StatusPernikahan','StatusPerkerjaan','JenisTempatTinggal'], axis=1, inplace=True)
df.head(5000)
```

	gender	Usia	TekananDarahTinggi	PenyakitJantung	TingkatGlukosa	BodyMassIndex	TingkatPerokok	Stroke
0	1	67.0	0	1	228.69	36.6	1	1
1	1	80.0	0	1	105.92	32.5	2	1
2	0	49.0	0	0	171.23	34.4	3	1
3	0	79.0	1	0	174.12	24.0	2	1
4	1	81.0	0	0	186.21	29.0	1	1
...	...	...	...	...	...	...	...	...
4976	1	41.0	0	0	70.15	29.8	1	0
4977	1	40.0	0	0	191.15	31.1	3	0
4978	0	45.0	1	0	95.02	31.8	3	0
4979	1	40.0	0	0	83.94	30.0	3	0
4980	0	80.0	1	0	83.75	29.1	2	0

4981 rows × 8 columns

Figure 7 Unsuded Column Removal

3.2.3 Exploratory Data Analysis

This study included multiple stages of exploratory data analysis (EDA), which involved examining the distribution of stroke cases by sex, age, and presence of heart disease. Figure 8 illustrates some examples of this stage.

```
sns.heatmap(df.corr(),annot=True, cmap='terrain', linewidths=0.1)
fig=plt.gcf()
#fig.set_size_inches(13,10)
plt.show()
```

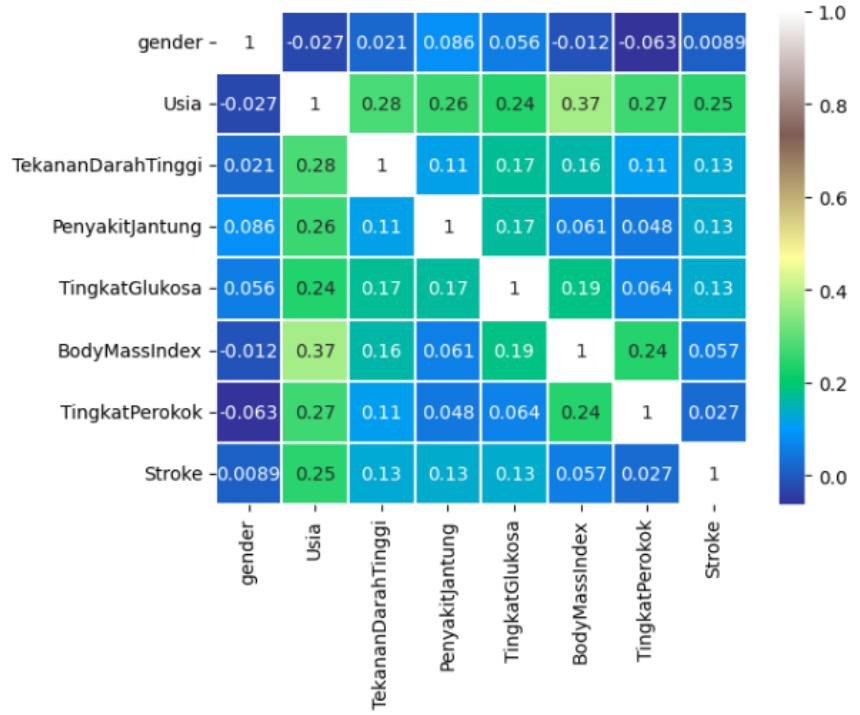

Figure 8 Stroke Data Distribution

3.2.4 Data Splitting

Data splitting refers to partitioning the dataset into two distinct subsets: training and test data. The purpose of data splitting is to facilitate the movement of machine learning models using a defined training data set while also evaluating the performance of a model trained using a separate test data set.

1. **Creating Defining Variables**

   Six columns are used to make the determining variable (X), as shown in Figure 9 below.

```
# MEMBUAT VARIABEL PENENTU (X)
X = df.drop('Stroke', axis=1)
X.head(5000)
```

	gender	Usia	TekananDarahTinggi	PenyakitJantung	TingkatGlukosa	BodyMassIndex	TingkatPerokok
0	1	67.0	0	1	228.69	36.6	1
1	1	80.0	0	1	105.92	32.5	2
2	0	49.0	0	0	171.23	34.4	3
3	0	79.0	1	0	174.12	24.0	2
4	1	81.0	0	0	186.21	29.0	1
...	...	...	...	...	...	...	...
4976	1	41.0	0	0	70.15	29.8	1
4977	1	40.0	0	0	191.15	31.1	3
4978	0	45.0	1	0	95.02	31.8	3
4979	1	40.0	0	0	83.94	30.0	3
4980	0	80.0	1	0	83.75	29.1	2

4981 rows × 7 columns

Figure 9 Creating Determinant Variable (X)

2. **Create a Target Variable (Y)**

   In making the target variable (Y), 1 column is used, as shown in Figure 10 below.

```
# MEMBUAT VARIABEL PENENTU (X)
X = df.drop('Stroke', axis=1)
X.head(5000)
```

	gender	Usia	TekananDarahTinggi	PenyakitJantung	TingkatGlukosa	BodyMassIndex	TingkatPerokok
0	1	67.0	0	1	228.69	36.6	1
1	1	80.0	0	1	105.92	32.5	2
2	0	49.0	0	0	171.23	34.4	3
3	0	79.0	1	0	174.12	24.0	2
4	1	81.0	0	0	186.21	29.0	1
...	...	...	...	...	...	...	...
4976	1	41.0	0	0	70.15	29.8	1
4977	1	40.0	0	0	191.15	31.1	3
4978	0	45.0	1	0	95.02	31.8	3
4979	1	40.0	0	0	83.94	30.0	3
4980	0	80.0	1	0	83.75	29.1	2

4981 rows × 7 columns

Figure 10 Create a Target Variable (Y)

3. **Data Splitting**

The data is divided into two parts, namely 70% training data and 30% test data which can be seen in Figure 11 below.

```
# IMPORT LIBRARY UNTUK DATA SPLITTING
from sklearn.model_selection import train_test_split

# IMPORT LIBRARY UNTUK MENGUKUR AKURASI MODEL
from sklearn.metrics import accuracy_score
```

```
# MEMBUAT FUNGSI UNTUK SPLITTING DATA

X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size=0.3, random_state=101, shuffle=True)
```

Figure 11 Data Splitting

3.3.5. Bangunan Model

Algoritma K-Nearest Neighbor, Decision Tree, dan Multilayer Perceptron diaplikasikan sebagai model klasifikasi dalam upaya pengidentifikasian gejala pada pasien, dengan tujuan mencapai tingkat akurasi yang optimal

1. **K-Nearest Neighbor (K-NN)**

This research creates a K-Nearest Neighbor (K-NN) with a n_neighbors=4 configuration using the KNeighborsClassifier library.

```
#import library KNN
from sklearn.neighbors import KNeighborsClassifier
```

```
knn= KNeighborsClassifier(n_neighbors=4)
knn.fit(X_train,Y_train)
```

```
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=4)
```

Figure 12 *Import Librabry KNN*

Based on the modeling above, the results obtained an accuracy of 93.84% for the K-Nearest Neighbor (K-NN) algorithm. In Figure 13, we can see the accuracy value of KNN:

```
y_pred_knn = knn.predict(X_test)
accuracy_knn=accuracy_score(Y_test,y_pred_knn)*100
print(f"Accuracy KNN : {accuracy_knn}")
print(f"{classification_report(Y_test,y_pred_knn)}")
```

```
Accuracy KNN : 93.84615384615384
              precision    recall  f1-score   support

           0       0.94      1.00      0.97      1405
           1       0.25      0.01      0.02        90

    accuracy                           0.94      1495
   macro avg       0.60      0.50      0.49      1495
weighted avg       0.90      0.94      0.91      1495
```

Figure 13 ***Result Accuracy KNN***

2. Decision Tree

This research creates a Decision Tree with a max_depth = 3 configuration using the Decision TreeClassifier library.

```
# INISIALISASI MODEL
dtc = DecisionTreeClassifier(max_depth=3)

# MELAKUKAN FITTING MODEL
dtc.fit(X_train, Y_train)
dtc
```

```
DecisionTreeClassifier
DecisionTreeClassifier(max_depth=3)
```

Figure 14 ***Import Library Decision Tree***

Based on the modeling above, the results obtained an accuracy of 93.97% for the DecisionTreeClassifier algorithm. In Figure 15, we can see the DCT accuracy value.

```
Y_pred_mlp = mlp.predict(X_test)
accuracy_mlp=accuracy_score(Y_test,Y_pred_mlp)*100
print(f"Accuracy DTC : {accuracy_mlp}")
print(f"{classification_report(Y_test,Y_pred_mlp)}")
```

```
Accuracy DTC : 93.9799331103679
              precision    recall  f1-score   support

           0       0.94      1.00      0.97      1405
           1       0.00      0.00      0.00        90

    accuracy                           0.94      1495
   macro avg       0.47      0.50      0.48      1495
weighted avg       0.88      0.94      0.91      1495
```

Figure 15 ***Result Accuracy DCT***

3. Multilayer Perceptron(MLP)

This research creates a Multilayer Perceptron(MLP) with a max_iter=2 configuration using the Multilayer Perceptron (MLP) library.

```
#import library MLP
from sklearn.neural_network import MLPClassifier

mlp= MLPClassifier(max_iter=2)
mlp.fit(X_train, Y_train)

/usr/local/lib/python3.10/dist-packages/sklearn/n
  warnings.warn(
▾      MLPClassifier
MLPClassifier(max_iter=2)
```

Figure 16 Import Librabry Multilayer Perceptron(MLP)

Based on the modeling above, the results obtained an accuracy of 93.9% for the Multilayer Perceptron (MLP) algorithm. In Figure 17, we can see the MLP accuracy value.

```
Y_pred_mlp = mlp.predict(X_test)
accuracy_mlp=accuracy_score(Y_test,Y_pred_mlp)*100
print(f"Accuracy MLP : {accuracy_mlp}")
print(f"{classification_report(Y_test,Y_pred_mlp)}")
```

```
Accuracy MLP : 93.91304347826087
              precision    recall  f1-score   support

           0       0.94      1.00      0.97      1405
           1       0.00      0.00      0.00        90

    accuracy                           0.94      1495
   macro avg       0.47      0.50      0.48      1495
weighted avg       0.88      0.94      0.91      1495
```

Figure 17 *Result Accuracy MLP*

3.2.6 Model Evaluation

To see whether the algorithm used has functioned well in classifying stroke, it is necessary to look at the model evaluation of the three algorithm methods. The library used in conducting the model evaluation is Skicit Lear which we can see in Figure 18 below.

```
# IMPORT LIBRARY UNTUK MODEL EVALUATION
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from sklearn.metrics import roc_curve, roc_auc_score
from sklearn.metrics import classification_report
```

Figure 18 Library Model Evaluation

1. **K-Nearest Neighbor (K-NN)**
   The results of the Confusion Matrix algorithm K-Nearest Neighbor (K-NN) process can be seen in Figure 19. The results were true positive 1491 times, true negative 3 times, false positive 80 times, and false negative 1.

```
# MEMBUAT CONFUSION MATRIX
grid_knn_matrix = confusion_matrix(Y_test, y_pred_knn)
sns.heatmap(grid_knn_matrix, annot=True)
```

```
<Axes: >
```

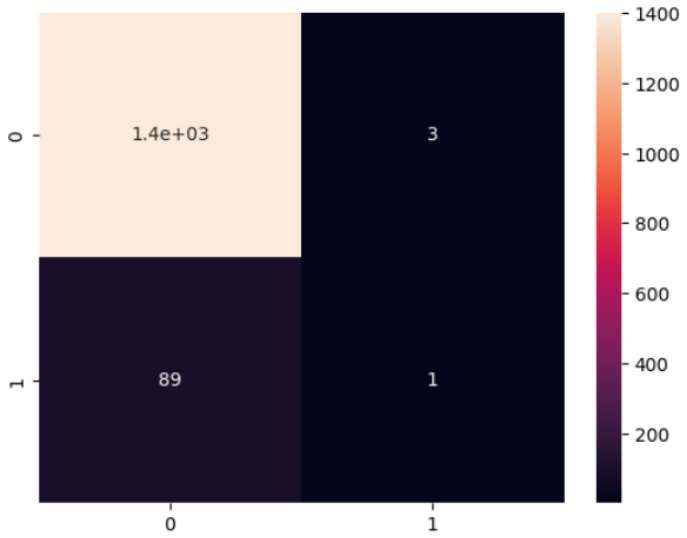

Figure 19 Confusion Matrix KNN

2. **Decision Tree**

The results of the Confusion Matrix Decision Tree algorithm process can be seen in Figure 20 that the results were valid positive as many as 1494, accurate negative 0 times, false positive 90, and false negative 0 times.

```
# MEMBUAT CONFUSION MATRIX
grid_dtc_matrix = confusion_matrix(Y_test, y_pred_dtc)
# VISUALISASI CONFUSION MATRIX
sns.heatmap(grid_dtc_matrix, annot=True)
```

```
<Axes: >
```

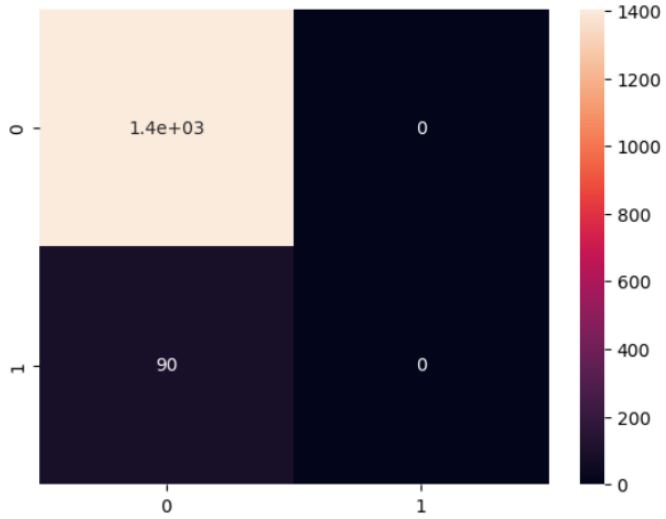

Figure 20 DCT Confusion Matrix

3. **Multi-Layer Precepton(MLP)**

The results of the Confusion Matrix Multi-Layer Perceptron algorithm process can be seen in Figure 21 that the results were valid positive as many as 1493, accurate negative 1 time, false positive 90, and incorrect negative 0 times.

```
# MEMBUAT CONFUSION MATRIX
grid_mlp_matrix = confusion_matrix(Y_test, Y_pred_mlp)
# VISUALISASI CONFUSION MATRIX
sns.heatmap(grid_mlp_matrix, annot=True)
```

```
<Axes: >
```

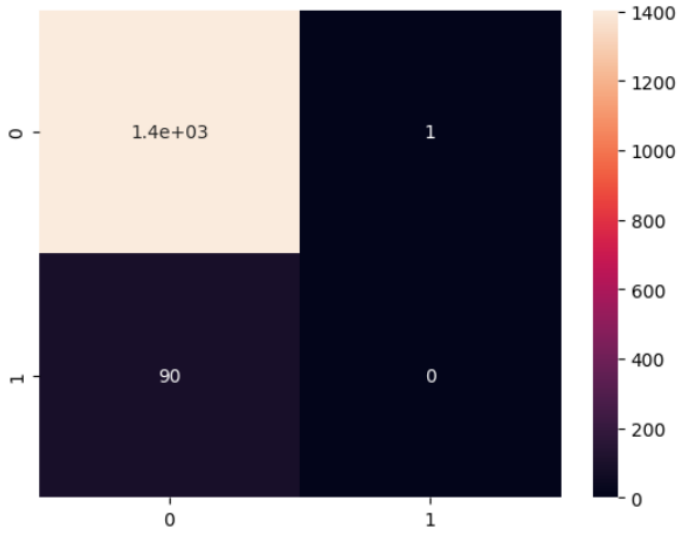

Figure 21 MLP Confusion Matrix

4. Comparison Result Accuracy

In each model a pretty good value is obtained where K-Nearest Neighbor (K-NN) with an accuracy value of 93.84%, Decision Tree of 93.97%, and Multilayer Perceptron (MLP) of 93.91% in classifying stroke patients accurately.

```
print(f"Accuracy KNN : {accuracy_knn}")
print(f"Accuracy DTC : {accuracy_dtc}")
print(f"Accuracy MLP : {accuracy_mlp}")
```

```
Accuracy KNN : 93.84615384615384
Accuracy DTC : 93.9799331103679
Accuracy MLP : 93.91304347826087
```

Figure 22 Comparison of Classification Results

CONCLUSION

With the help of Machine Learning techniques based on classification algorithms, stroke can now be predicted using technology in the medical field. By using predetermined medical parameters, the Machine Learning model that has been developed can predict whether a person will have a stroke. The results of making the three algorithms are quite good, where K-Nearest Neighbor (K-NN) has an accuracy value of 93.84%, Decision Tree is 93.97%, and Multilayer Perceptron (MLP) is 93.91%. Therefore, the three algorithms can make good predictions from the True Positive confusion matrix results of more than 1490. The best accuracy value is the Decision Tree algorithm with an accuracy difference of no more than 0.10% with the other two algorithms

BIBLIOGRAPHY

[1] D. Rahmadani, A. Alamuddin Muzafar, A. Hamid, and R. Annisa, “Analisis Perbandingan Algoritma C4.5 dan CART untuk Klasifikasi Penyakit Stroke.” [Online]. Available: https://journal.irpi.or.id/index.php/sentimas

[2] F. Adha, H. Airi, T. Suprapti, and A. Bahtiar, "Komparasi Metode Klasifikasi Data Mining Untuk Prediksi Penyakit Stroke," vol. 18, pp. 73–79, 2023.
[3] A. Alkafi and F. Hasnah, "Meta-Analisis Risiko Hipertensi dengan Penyakit Stroke di Asia," JIK J. ILMU Kesehat., vol. 6, no. 2, p. 302, Oct. 2022, doi: 10.33757/jik.v6i2.539.
[4] A. Perbandingan Metode Dempster Shafer, I. Lina Kendayto Panjaitan, E. Panggabean, and T. Informatika, "Analisis Perbandingan Metode Dempster Shafer Dengan Metode Certainty Factor Untuk Mendiagnosa Penyakit Stroke," STMIK Pelita Nusant. Medan Jl. Iskandar Muda, vol. 3, no. 1, 2018.
[5] K. Akmal, A. Faqih, and F. Dikananda, "Perbandingan Metode Algoritma Naïve Bayes Dan K-Nearest Neighbors Untuk Klasifikasi Penyakit Stroke," 2023. [Online]. Available: www.researchgate.net
[6] Indarto, E. Utami, and S. Raharjo, "Mortality Prediction Using Data Mining Classification Techniques in Patients with Hemorrhagic Stroke," in 2020 8th International Conference on Cyber and IT Service Management, CITSM 2020, Oct. 2020. doi: 10.1109/CITSM50537.2020.9268802.
[7] "Learn about stroke | World Stroke Organization." https://www.world-stroke.org/world-stroke-day-campaign/why-stroke-matters/learn-about-stroke# (accessed Jul. 28, 2023).
[8] F. Nurlan, "Analisis Survival Sstroke Berulang Menurut Umur Dan Jenis Kelamin Pasien Stroke Di Kota Makassar Survival," 2018. [Online]. Available: https://jurnal.unismuhpalu.ac.id/index.php/MPPKI/article/view/1086
[9] Y. Ayuningtyas and I. Made Suartana, "Klasifikasi Penyakit Stroke Menggunakan Support Vector Machine (SVM) dan Particle Swarm Optimization (PSO)".
[10] Y. A. Rindri and A. Fitriyani, "Analisis Perbandingan Kinerja Algoritma Multilayer Perceptron dan K-Nearest Neighbor pada Klasifikasi Tipe Migrain," J. Teknol. dan Inf., vol. 13, no. 1, pp. 44–55, 2023, doi: 10.34010/jati.v13i1.9111.
[11] D. Ulfatul, M. Rachmad, H. Oktavianto, and M. Rahman, "Perbandingan Metode K-Nearest Neighbor Dan Gaussian Naive Bayes Untuk Klasifikasi Penyakit Stroke," 2022. [Online]. Available: http://jurnal. unmuhjember.ac.id/index.php/JST
[12] J. Teknika and R. Estian Pambudi, "Klasifikasi Penyakit Stroke Menggunakan Algoritma Decision Tree C.45," IJCCS, vol. x, No.x, pp. 1–5.
[13] M. H. Ariansyah, S. Winarno, E. Nur Fitri, and H. M. Arga Retha, "Multilayer Perceptron For Diagnosing Stroke With The SMOTE Method In Overcoming Data Imbalances," Innov. Res. Informatics, vol. 5, no. 1, pp. 1–8, 2023, doi: 10.37058/innovatics.v5i1.6565.
[14] W. Nugraha and R. Sabaruddin, "Teknik Resampling untuk Mengatasi Ketidakseimbangan Kelas pada Klasifikasi Penyakit Diabetes Menggunakan C4.5, Random Forest, dan SVM Resampling Technique for Handling Class Imbalance in the Classification of Diabetes using C4.5, Random Forest, and SVM," Agustus, vol. 20, no. 3, pp. 352–361, 2021, [Online]. Available: https://www.kaggle.com/uciml/pima-indians-diabetes-database.
[15] "A Comparison of Heart Abnormalities Detection on ECG using KNN and Decision Tree".
[16] A. R. Oktavyani et al., "Perbandingan Metode Naive Bayes, K-NN dan Decision Tree Terhadap Dataset Healthcare Stroke", doi: 10.31284/p.snestik.2023.4067.
[17] A. Wahab, S. Samarinda, I. Lishania, R. Goejantoro, and Y. N. Nasution, "Perbandingan Klasifikasi Metode Naive Bayes dan Metode Decision Tree Algoritma (J48) pada Pasien Penderita Penyakit Stroke di RSUD Abdul Wahab Sjahranie Samarinda Hospital," J. EKSPONENSIAL, vol. 10, no. 2, 2019.
[18] "Brain stroke classification | Kaggle." https://www.kaggle.com /code/maihesham103/brain-stroke-classification (accessed Jul. 29, 2023).
[19] "Brain Stroke Dataset | Kaggle." https://www.kaggle.com/datasets/ jillanisofttech/brain-stroke-dataset (accessed Jul. 29, 2023).