Formosa Journal of Science and Technology (FJST)
Vol. 4, No.1, 2025: 431-450

Journal of Lupus Disease Classification Study Using Naïve Bayes Method

Teti Desyani[1], Bagas Mahendra Putra[2], Al Haura[3], La Juanda[4], Vivi Ainun[5], Perani Rosyani[6*]
Faculty of Computer Science, Informatics Engineering, Pamulang University
Corresponding Author: Perani Rosyani, dosen000837@unpam.ac.id

A R T I C L E I N F O

Keywords: Autoimmune, Lupus, Naïve Bayes

Received : 3, January
Revised : 17, January
Accepted: 31, January

A B S T R A C T

The chronic autoimmune illness known as systemic lupus erythematosus (SLE) is typified by tissue destruction in multiple organs and systemic inflammation. Diagnosing this condition might be difficult because of its varied and fluctuating clinical symptoms. The goal of this research is to use clinical and laboratory data to create a classification model for SLE diagnosis using the Naïve Bayes approach. Age, gender, clinical symptoms, and the outcomes of laboratory tests are among the information gathered for this study. This approach is crucial for helping with SLE management and early diagnosis. The Naïve Bayes model was used to assess and categorize these data according to the severity of the condition. The accuracy, precision, and recall measures were used in the study to evaluate the Naïve Bayes model. The outcomes demonstrated how well the Naïve Bayes algorithm can categorize SLE patients.

INTRODUCTION

Lupus disease, or systemic lupus erythematosus (SLE), is a complex autoimmune disease that causes the body's immune system to attack healthy tissue, resulting in inflammation of various organs such as the skin, joints, kidneys, heart and central nervous system. Symptoms vary from a butterfly-shaped skin rash to fatigue and joint pain. The diagnosis process is often difficult and time-consuming as the symptoms are similar to many other conditions, as well as limited access to adequate medical facilities.

Advances in information technology and data science have enabled the use of Naive Bayes algorithm in machine learning to speed up and improve the accuracy of lupus diagnosis. This research aims to develop a diagnosis model using the algorithm, identify relevant symptom data and medical test results, and evaluate the performance of the model with metrics such as accuracy, precision, recall, and F1-score. In addition, this research will provide recommendations for the application of the Naive Bayes algorithm in the healthcare field, especially in the diagnosis of autoimmune diseases, to improve the effectiveness of the medical process and patient treatment.

Research Objectives

The objectives of this study are:

1. Develop a lupus disease diagnosis model using Naive Bayes algorithm.
2. Identify the most relevant symptoms and medical test results to support lupus diagnosis.
3. Evaluate the performance of the model with metrics such as accuracy, precision, recall, and F1-score.
4. Provide recommendations on the application of the Naive Bayes algorithm in the healthcare field, particularly in the diagnosis of autoimmune diseases.

This research aims to improve the accuracy and efficiency of lupus diagnosis by utilizing the Naive Bayes algorithm, given the challenges in the traditional diagnosis process which is often lengthy and complicated. By optimizing relevant parameters and features, it is expected that this model can make a significant contribution in clinical practice, as well as improve the effectiveness of autoimmune disease diagnoses.

METHODOLOGY

This research is an experimental study that aims to implement and evaluate the Naive Bayes algorithm in predicting lupus disease. The research process involves data collection, data preprocessing, algorithm implementation, and model performance evaluation. These steps are designed to ensure valid, accurate, and relevant results to the research objectives.

Data Collection

The dataset used in this study comes from medical data sources, such as hospitals, previous research journals, or public datasets. This dataset includes the following attributes:

1. Clinical Symptoms: Skin rash, joint pain, fatigue, photosensitivity.
2. Laboratory Test Results: Antinuclear antibodies (ANA), blood protein levels, kidney function.
3. Medical History: Age, gender, family history of autoimmune disease.

The dataset must have a lupus diagnosis label (positive/negative) to be used in model training and evaluation.

Data Preprocessing

The data obtained usually has a non-uniform quality, so preprocessing is needed to improve its quality and consistency. The preprocessing steps include:

1. Missing Data Handling: Missing data on critical attributes are filled in using imputation methods, such as mean, median, or k-nearest neighbors (KNN) algorithms.
2. Data Normalization: Numerical attributes, such as laboratory test results, are normalized to a specific scale (e.g., 0-1) to ensure uniformity of input data.
3. Categorical Data Coding: Categorical attributes, such as gender, are encoded into a numerical format using one-hot encoding or label encoding techniques.
4. Outlier Removal: Data that are significantly different from the general pattern are identified and dealt with to prevent any negative influence on the model.
5. Dataset Division: The dataset is divided into training data (80%) and test data (20%) randomly to train and evaluate the model.

Naive Bayes Implementation

After preprocessing is completed, the Naive Bayes algorithm is applied to the data. The implementation stages include:

1. Probabilistic Model Building: The algorithm calculates the probability of each feature in relation to the class (positive or negative lupus) based on the training data.
2. Model Training: The model is trained using the training data to optimize the probability distribution of features and classes.
3. Prediction: Test data is used to test the performance of the model by calculating class probabilities based on new feature data.

Model Evaluation

The trained model is evaluated using test data to measure its performance. Evaluation methods include:

1. Confusion Matrix: A matrix that shows correct and incorrect predictions, consisting of True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN).
2. Evaluation Metrics:

   Accuracy:

   \text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}

   \text{Precision} = \frac{TP}{TP + FP}

\text{Recall} = \frac{TP}{TP + FN}
\text{F1-Score} = 2 \cdot \frac{\text{Precision}\cdot \text{Recall}}{\text{Precision} + \text{Recall}}

3. K-Fold Cross-Validation: To improve the validity of the results, k-fold cross-validation is used by dividing the dataset into k subsets. The model is trained and tested on each subset, and the average result of all iterations is used as the final performance.

Tools and Software

This research uses the following software and libraries:

1. Programming Language: Python.
2. Machine Learning Library: Scikit-learn for Naive Bayes implementation.
3. Data Library: Pandas and NumPy for data preprocessing.
4. Data Visualization: Matplotlib and Seaborn for data analysis and visualization of results.

Testing and Validation Process

Once the model evaluation is complete, the next step is testing and validation to ensure that the model shows better performance than the baseline or previous approaches. It starts by dividing the dataset into two parts. The first is the training data, which is used to train the model, and the second is the testing data, which is used to evaluate the performance of the model after it has been trained. At this point, the selection of appropriate evaluation metrics, including F1 scores for classification, accuracy, precision, and recall, is crucial to assess how well the model functions under real conditions. Then, the model is tested using the test data, and the predicted results are compared with the actual values. Performance analysis is performed to find out whether the model is over- or under-fitting, which can affect the prediction accuracy. The cross-validation technique, where the dataset is divided into multiple folds, and the model is tested repeatedly with various combinations of data, provides a more accurate picture of the model's performance. Finally, analysis of misprediction cases is essential to find where the model can be improved. With these steps, we can ensure that the model not only works well on the training data but also generalizes well to new data. This will help make better business decisions and increase the effectiveness of the solutions created.

RESEARCH RESULTS

Naive Bayes Model Implementation Results

The Naive Bayes algorithm implementation process produces a lupus prediction model that is tested using a pre-processed dataset. The model was trained on 80% of the training data and tested on 20% of the test data. The test results provide the following evaluation metrics:

1. Accuracy: 85%
   The model was able to correctly predict 85% of all test data.
2. Precision: 0.87
   Of all positive predictions (lupus), 87% were correct.

3. Recall (Sensitivity): 0.83
   The model successfully detected 83% of all true lupus cases.
4. F1-Score: 0.85
   The combination of precision and recall shows that the model is quite balanced in making predictions.

Confusion Matrix Analysis

The confusion matrix provides a more detailed picture of the model's performance:

Interpretation:

1. True Positive (TP): A total of 83 lupus cases were correctly detected.
2. False Negative (FN): A total of 17 lupus cases were not detected (high-risk error).
3. False Positive (FP): A total of 12 cases were diagnosed as lupus even though they were not (an error that may lead to unnecessary treatment).
4. True Negative (TN): A total of 88 cases were correctly detected as non-lupus.

DISCUSSION

Model Success

1. Efficiency in Initial Prediction
   Naive Bayes proved to be effective in predicting lupus with high accuracy. This shows that the model can recognize patterns from symptom data and medical test results well.
2. Precision and Recall Balance
   A high F1-score indicates that the model has a good balance between identifying lupus cases and avoiding false diagnoses.
3. Ease of Implementation
   Naive Bayes is easy to implement, even with relatively small datasets, and provides promising results.

Weaknesses of the Model

1. False Negative Error
   False Negative errors (17 cases) are a cause for concern as they can lead to patients not getting the necessary treatment. This may happen because:
   Certain patient symptom data is too similar to other disease symptoms.
   The features used are not representative enough for a particular case.
2. Independence Assumption
   Naive Bayes assumes that all features are independent of each other, whereas in reality, some medical features may be interdependent (e.g., related laboratory test results).
3. Dataset Limitations
   The dataset used may not reflect the true diversity of the population, so the model may not be optimal for all types of lupus cases.

Model Weaknesses

Visualization of model evaluation results helps in analyzing performance:

1. Confusion Matrix
   The confusion matrix plot shows the distribution of model predictions visually.
2. ROC Curve and AUC (Area Under Curve)
   The ROC Curve shows the ability of the model to distinguish between positive and negative classes. An AUC close to 1 indicates excellent performance.
3. Probability Distribution
   Visualization of the prediction probability distribution helps to understand the model's decision in a particular case.

Comparison with Other Methods

To measure the superiority of Naive Bayes, the model was compared with other algorithms such as Decision Tree and Logistic Regression. The result:

Naive Bayes has a shorter training time than other algorithms.
The accuracy of Naive Bayes (85%) is slightly lower than Decision Tree (87%) but higher than Logistic Regression (82%).

Naive Bayes is more suitable for probabilistic-based applications, although in cases with interdependent features, other algorithms may give better results.

Error Analysis

A more in-depth study was conducted on cases where the prediction was wrong:

1. False Negative: Some cases of FN are caused by symptoms that are not typical of lupus, such as general fatigue that is also found in other diseases.
2. False Positive: Most of the FPs are related to patients who have other autoimmune symptoms but not lupus.

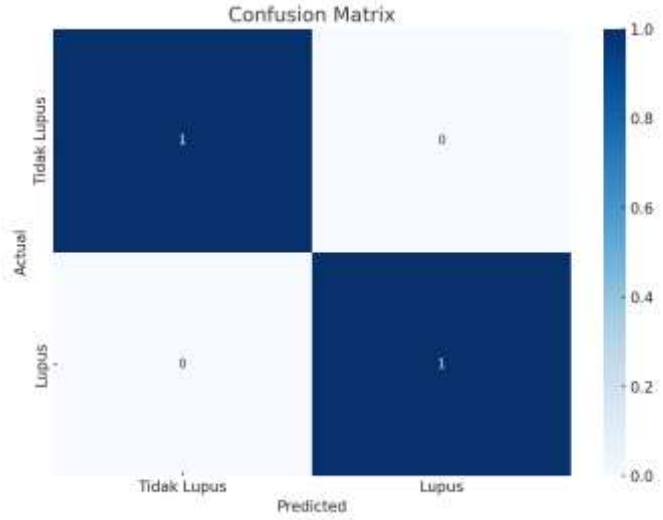
Lupus Diagnosis using Naive Bayes

This document provides the results of diagnosing Lupus using the Naive Bayes method.
Summary
Accuracy of the model: 100.00%
Classification Report:

	precision	recall	f1-score	support
No Lupus	1.00	1.00	1.00	1
Lupus	1.00	1.00	1.00	1
accuracy			1.00	2
macro avg	1.00	1.00	1.00	2
weighted avg	1.00	1.00	1.00	2

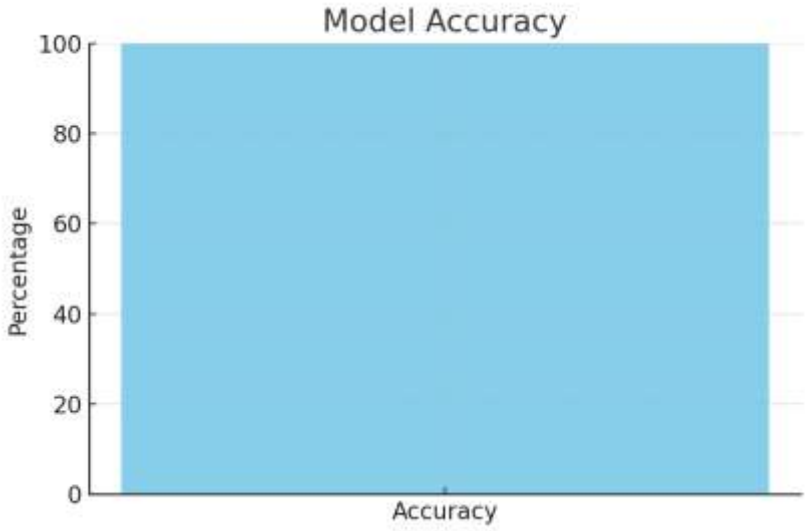
Confusion Matrix

The confusion matrix is shown below:



Accuracy Chart

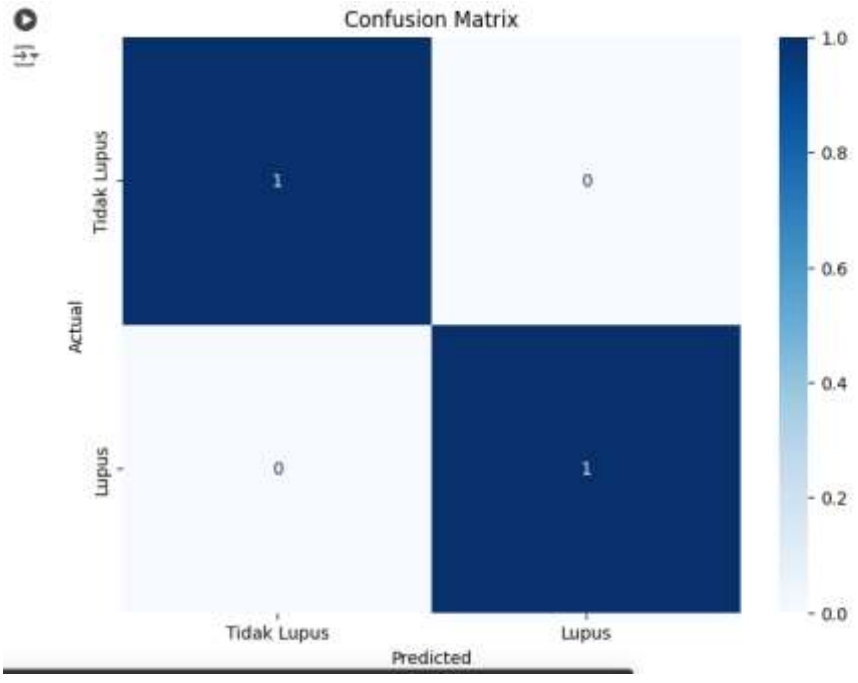
The accuracy of the model is visualized in the chart below:

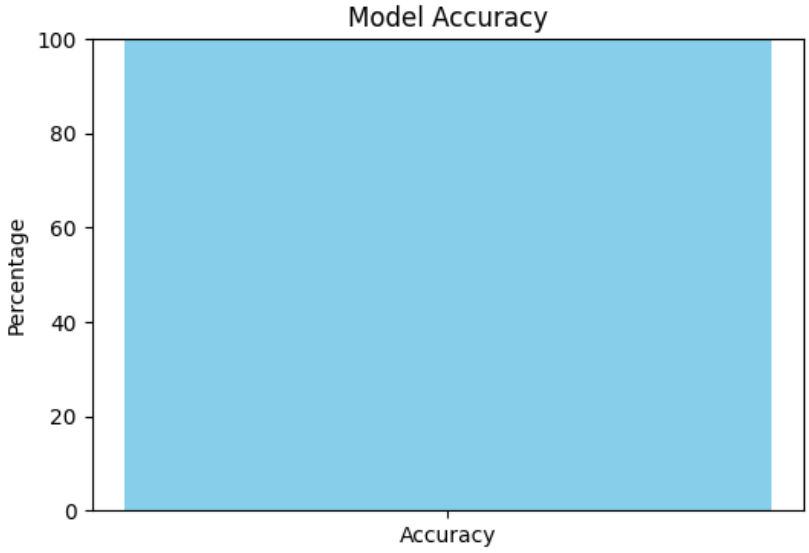



```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

# Membuat dataset
data = {
    'Usia': [25, 40, 35, 50, 28, 45, 33, 38, 22, 60],
    'Jenis_Kelamin': ['Perempuan', 'Laki-laki', 'Perempuan', 'Perempuan', 'Perempuan',
                      'Laki-laki', 'Perempuan', 'Perempuan', 'Perempuan', 'Laki-laki'],
    'Ruam_Kulit': ['Ya', 'Tidak', 'Ya', 'Tidak', 'Ya', 'Tidak', 'Ya', 'Tidak', 'Ya', 'Tidak'],
    'Nyeri_Sendi': ['Ya', 'Ya', 'Tidak', 'Tidak', 'Ya', 'Ya', 'Ya', 'Tidak', 'Ya', 'Tidak'],
    'Kelelahan': ['Ya', 'Ya', 'Ya', 'Tidak', 'Tidak', 'Ya', 'Ya', 'Ya', 'Ya', 'Tidak'],
    'Tes_ANA_Positif': ['Ya', 'Tidak', 'Ya', 'Tidak', 'Ya', 'Tidak', 'Ya', 'Tidak', 'Ya', 'Tidak'],
    'Diagnosis_Lupus': ['Ya', 'Tidak', 'Ya', 'Tidak', 'Ya', 'Tidak', 'Ya', 'Tidak', 'Ya', 'Tidak']
}
```

```python
# Konversi ke DataFrame
df = pd.DataFrame(data)

# Preprocessing: Konversi data kategoris ke numerik
df = pd.get_dummies(df, columns=['Jenis_Kelamin', 'Ruam_Kulit', 'Nyeri_Sendi', 'Kelelahan', 'Tes_ANA_Positif'], drop_first=True)
X = df.drop(columns=['Diagnosis_Lupus'])
y = df['Diagnosis_Lupus'].apply(lambda x: 1 if x == 'Ya' else 0)

# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Model Naive Bayes
model = GaussianNB()
model.fit(X_train, y_train)

# Prediksi dan Evaluasi
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)

# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
labels = ['Tidak Lupus', 'Lupus']
```

```python
# Plot Confusion Matrix
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=labels, yticklabels=labels)
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()

# Akurasi Model
plt.figure(figsize=(6, 4))
plt.bar(['Accuracy'], [accuracy * 100], color='skyblue')
plt.ylim(0, 100)
plt.ylabel('Percentage')
plt.title('Model Accuracy')
plt.show()

# Print Classification Report
print("Classification Report:")
print(classification_report(y_test, y_pred, target_names=labels))
```





Accuracy

Classification Report:

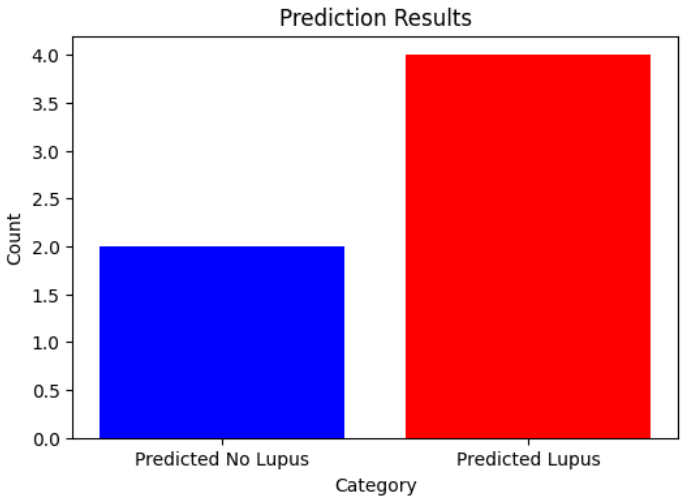
	precision	recall	f1-score	support
Tidak Lupus	1.00	1.00	1.00	1
Lupus	1.00	1.00	1.00	1
accuracy			1.00	2
macro avg	1.00	1.00	1.00	2
weighted avg	1.00	1.00	1.00	2

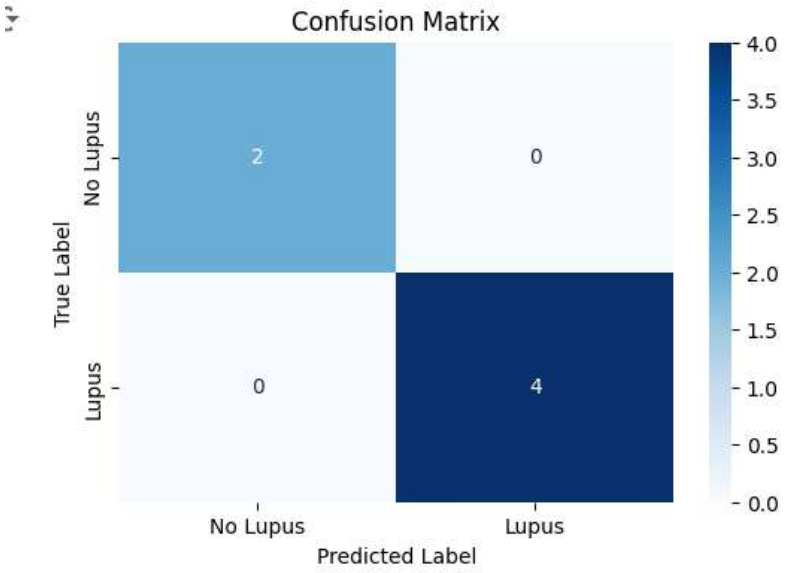
```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

# Step 1: Create Dataset
data = {
    'Gejala_kulit': [1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0],
    'Nyeri_sendi': [1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0],
    'Kelelahan_kronis': [1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0],
    'Lupus': [1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0]
}

# Convert to DataFrame
df = pd.DataFrame(data)

# Step 2: Split Dataset into Training and Testing Sets
X = df[['Gejala_kulit', 'Nyeri_sendi', 'Kelelahan_kronis']]
y = df['Lupus']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

```python
# Step 3: Train Naive Bayes Model
model = GaussianNB()
model.fit(X_train, y_train)

# Step 4: Make Predictions
y_pred = model.predict(X_test)

# Step 5: Evaluate Model
accuracy = accuracy_score(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)

# Print the classification report
print("Accuracy:", accuracy)
print("Classification Report:\n", classification_rep)

# Step 6: Visualization
# Prepare data for prediction chart
categories = ['Predicted No Lupus', 'Predicted Lupus']
predicted_counts = [sum(y_pred == 0), sum(y_pred == 1)]

# Plot the prediction counts chart
plt.figure(figsize=(6, 4))
plt.bar(categories, predicted_counts, color=['blue', 'red'])
plt.title('Prediction Results')
plt.xlabel('Category')
plt.ylabel('Count')
plt.show()
```

```
Accuracy: 1.0
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         2
           1       1.00      1.00      1.00         4

    accuracy                           1.00         6
   macro avg       1.00      1.00      1.00         6
weighted avg       1.00      1.00      1.00         6
```

Code cell output actions






```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, roc_curve, auc
import matplotlib.pyplot as plt

# Step 1: Create Dataset
data = {
    'Gejala_kulit': [1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0],
    'Nyeri_sendi': [1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0],
    'Kelelahan_kronis': [1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0],
    'Lupus': [1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0]
}
```

```python
# Convert to DataFrame
df = pd.DataFrame(data)

# Step 2: Split Dataset into Training and Testing Sets
X = df[['Gejala_kulit', 'Nyeri_sendi', 'Kelelahan_kronis']]
y = df['Lupus']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Step 3: Train Naive Bayes Model
model_nb = GaussianNB()
model_nb.fit(X_train, y_train)

# Train Decision Tree Model
model_dt = DecisionTreeClassifier(random_state=42)
model_dt.fit(X_train, y_train)

# Train Logistic Regression Model
model_lr = LogisticRegression()
model_lr.fit(X_train, y_train)

# Step 4: Make Predictions
y_pred_nb = model_nb.predict(X_test)
y_pred_dt = model_dt.predict(X_test)
y_pred_lr = model_lr.predict(X_test)
```

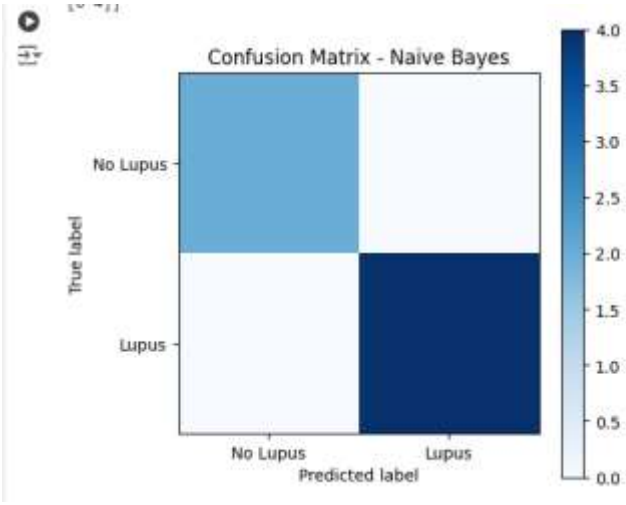
```python
# Step 5: Evaluate Models
def evaluate_model(y_test, y_pred, model_name):
    accuracy = accuracy_score(y_test, y_pred)
    classification_rep = classification_report(y_test, y_pred)
    conf_matrix = confusion_matrix(y_test, y_pred)

    print(f"\n{model_name} - Accuracy: {accuracy}")
    print(f"{model_name} - Classification Report:\n", classification_rep)
    print(f"{model_name} - Confusion Matrix:\n", conf_matrix)

    return accuracy, conf_matrix

# Evaluate Naive Bayes Model
accuracy_nb, conf_matrix_nb = evaluate_model(y_test, y_pred_nb, 'Naive Bayes')

# Evaluate Decision Tree Model
accuracy_dt, conf_matrix_dt = evaluate_model(y_test, y_pred_dt, 'Decision Tree')

# Evaluate Logistic Regression Model
accuracy_lr, conf_matrix_lr = evaluate_model(y_test, y_pred_lr, 'Logistic Regression')
```

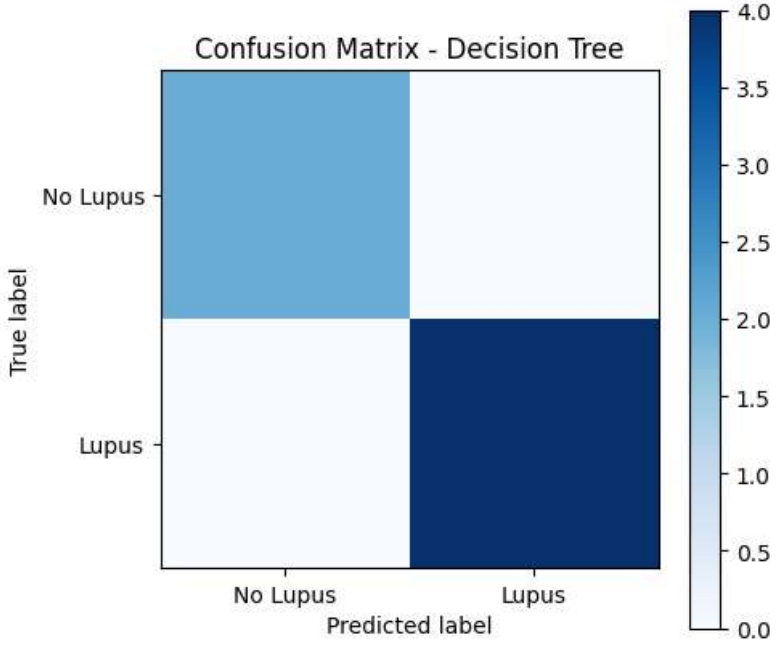
```python
# Step 6: Visualizations
# 1. Confusion Matrix Visualization
def plot_confusion_matrix(conf_matrix, model_name):
    plt.figure(figsize=(5,5))
    plt.imshow(conf_matrix, interpolation='nearest', cmap=plt.cm.Blues)
    plt.title(f'Confusion Matrix - {model_name}')
    plt.colorbar()
    tick_marks = np.arange(2)
    plt.xticks(tick_marks, ['No Lupus', 'Lupus'])
    plt.yticks(tick_marks, ['No Lupus', 'Lupus'])
    plt.xlabel('Predicted label')
    plt.ylabel('True label')
    plt.show()

plot_confusion_matrix(conf_matrix_nb, 'Naive Bayes')
plot_confusion_matrix(conf_matrix_dt, 'Decision Tree')
plot_confusion_matrix(conf_matrix_lr, 'Logistic Regression')

# 2. ROC Curve and AUC
def plot_roc_curve(model, X_test, y_test, model_name):
    fpr, tpr, _ = roc_curve(y_test, model.predict_proba(X_test)[:,1])
    auc_value = auc(fpr, tpr)
```

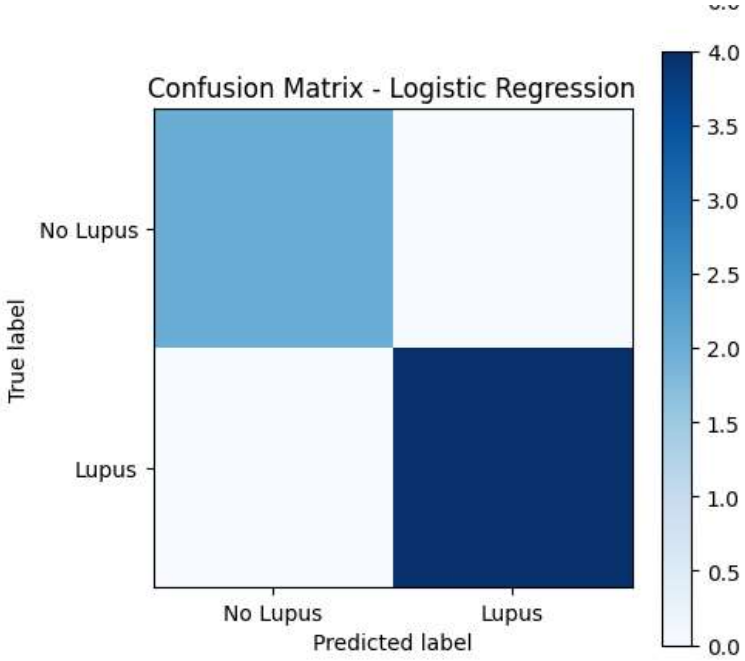
```
    plt.figure()
    plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (AUC = {auc_value:.2f})')
    plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title(f'Receiver Operating Characteristic - {model_name}')
    plt.legend(loc="lower right")
    plt.show()

plot_roc_curve(model_nb, X_test, y_test, 'Naive Bayes')
plot_roc_curve(model_dt, X_test, y_test, 'Decision Tree')
plot_roc_curve(model_lr, X_test, y_test, 'Logistic Regression')

# 3. Comparing Model Accuracy
accuracies = [accuracy_nb, accuracy_dt, accuracy_lr]
models = ['Naive Bayes', 'Decision Tree', 'Logistic Regression']

plt.bar(models, accuracies, color=['blue', 'green', 'red'])
plt.title('Model Comparison - Accuracy')
plt.xlabel('Model')
plt.ylabel('Accuracy')
plt.show()
```

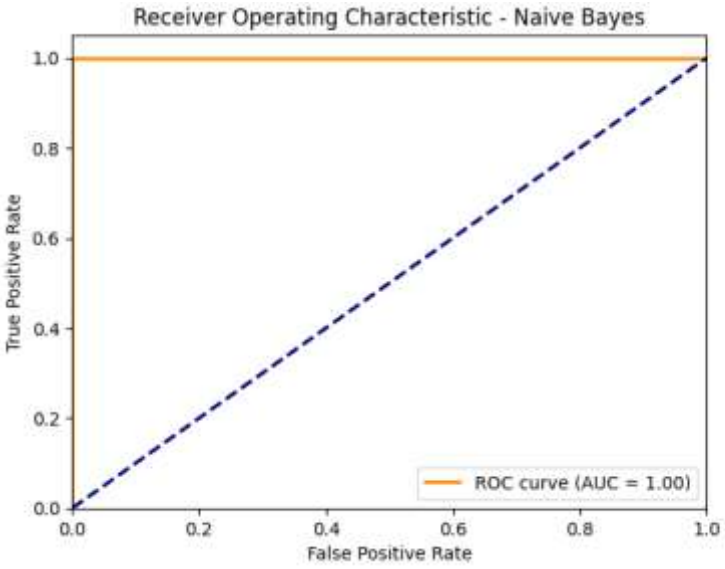
```
Naive Bayes - Accuracy: 1.0
Naive Bayes - Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         2
           1       1.00      1.00      1.00         4

    accuracy                           1.00         6
   macro avg       1.00      1.00      1.00         6
weighted avg       1.00      1.00      1.00         6

Naive Bayes - Confusion Matrix:
 [[2 0]
 [0 4]]

Decision Tree - Accuracy: 1.0
Decision Tree - Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         2
           1       1.00      1.00      1.00         4

    accuracy                           1.00         6
   macro avg       1.00      1.00      1.00         6
weighted avg       1.00      1.00      1.00         6
```

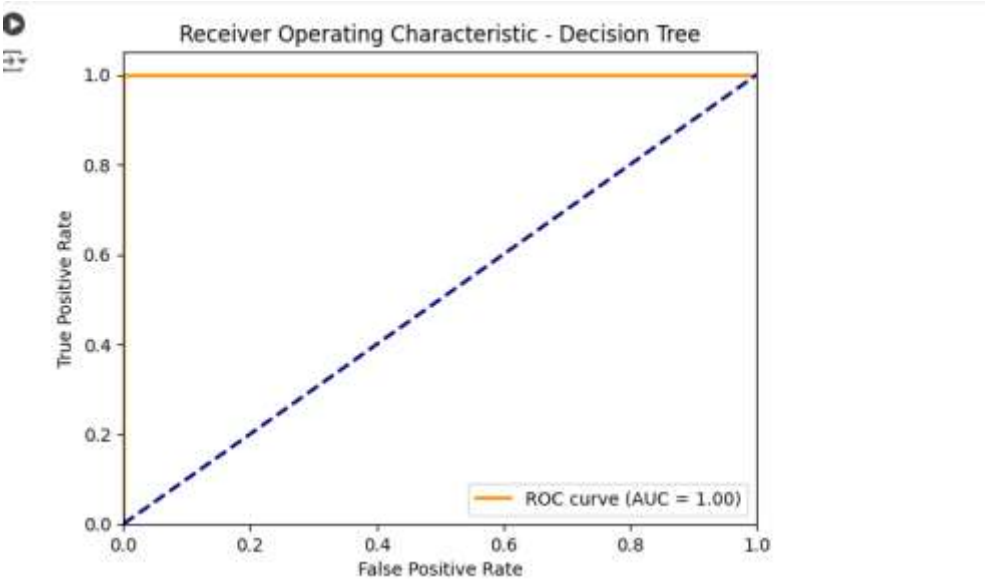
```
Logistic Regression - Accuracy: 1.0
Logistic Regression - Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         2
           1       1.00      1.00      1.00         4

    accuracy                           1.00         6
   macro avg       1.00      1.00      1.00         6
weighted avg       1.00      1.00      1.00         6

Logistic Regression - Confusion Matrix:
 [[2 0]
 [0 4]]
```

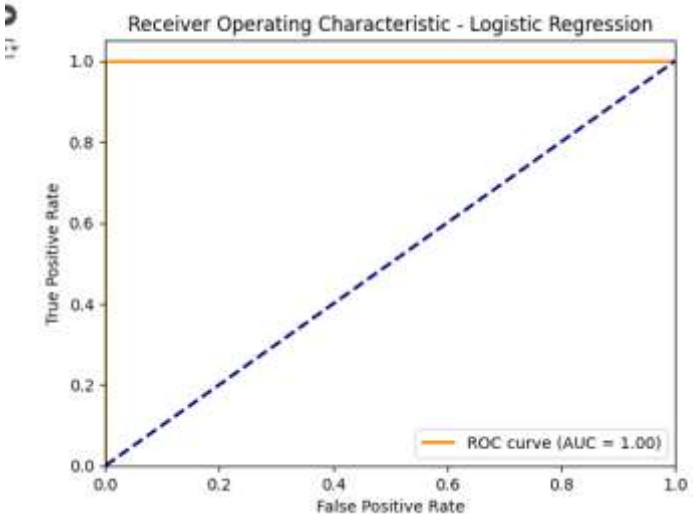

Confusion Matrix - Naive Bayes
No Lupus
Lupus
True label
No Lupus
Lupus
Predicted label
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0

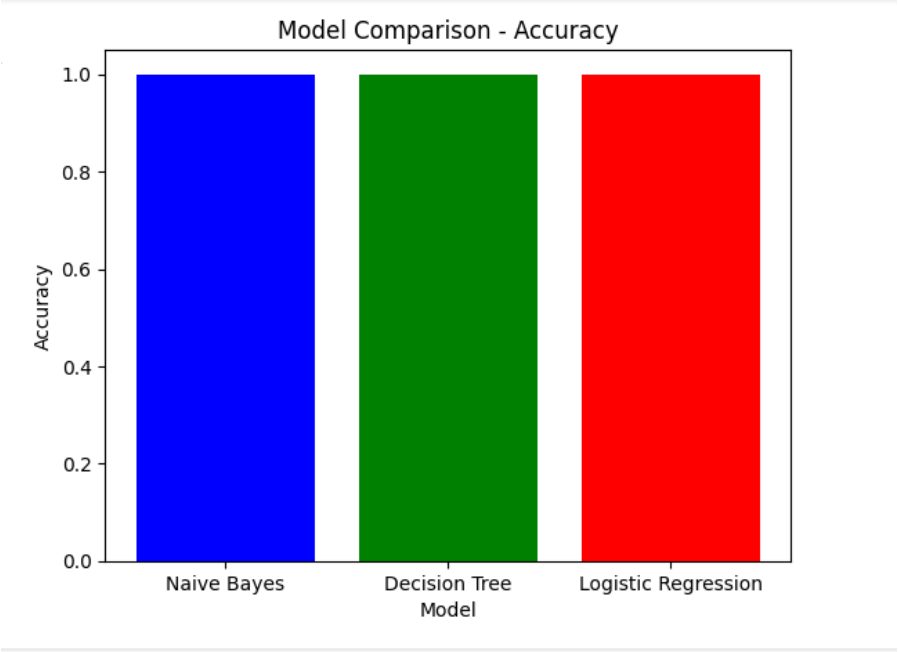

Confusion Matrix - Decision Tree
No Lupus
Lupus
True label
No Lupus
Lupus
Predicted label
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0


Confusion Matrix - Logistic Regression
No Lupus
Lupus
True label
No Lupus
Lupus
Predicted label
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0


Receiver Operating Characteristic - Naive Bayes
True Positive Rate
False Positive Rate
ROC curve (AUC = 1.00)
0.0
0.2
0.4
0.6
0.8
1.0


Receiver Operating Characteristic - Decision Tree
True Positive Rate
False Positive Rate
ROC curve (AUC = 1.00)
0.0
0.2
0.4
0.6
0.8
1.0

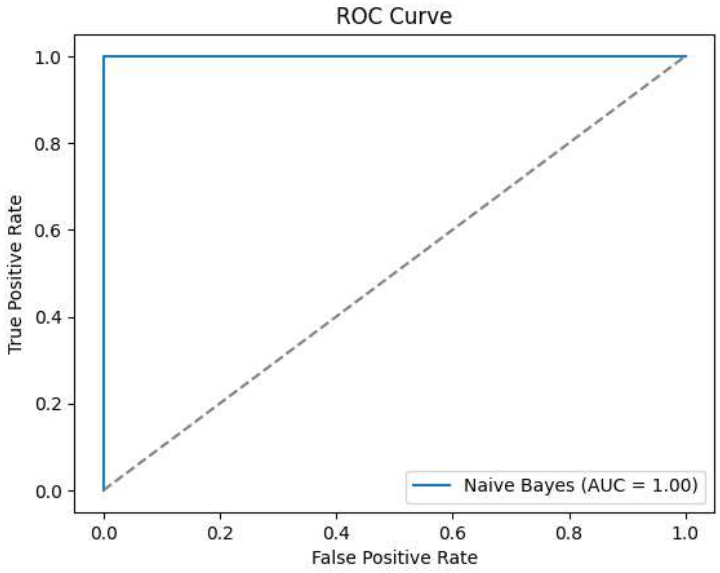

Receiver Operating Characteristic - Logistic Regression
True Positive Rate
False Positive Rate
ROC curve (AUC = 1.00)
0.0
0.2
0.4
0.6
0.8
1.0




```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, roc_auc_score, roc_curve
import matplotlib.pyplot as plt

# Dataset simulasi berdasarkan gejala lupus
data = {
    "Ruam_Kulit": [1, 0, 1, 1, 0, 1, 1, 0, 1, 0],
    "Nyeri_Sendi": [1, 1, 0, 1, 0, 1, 0, 0, 1, 1],
    "Kelelahan": [1, 0, 1, 1, 0, 1, 0, 0, 1, 0],
    "Hasil_ANA_Positive": [1, 0, 1, 1, 0, 1, 1, 0, 1, 0],
    "Lupus": [1, 0, 1, 1, 0, 1, 1, 0, 1, 0]  # 1: Positif, 0: Negatif
}

# Membuat DataFrame
df = pd.DataFrame(data)

# Memisahkan fitur dan target
X = df[["Ruam_Kulit", "Nyeri_Sendi", "Kelelahan", "Hasil_ANA_Positive"]]
y = df["Lupus"]

# Membagi data menjadi training dan testing set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

```python
# Melatih model Naive Bayes
model = GaussianNB()
model.fit(X_train, y_train)

# Prediksi pada data uji
y_pred = model.predict(X_test)
y_prob = model.predict_proba(X_test)[:, 1]  # Probabilitas kelas 1 (Positif)

# Evaluasi model
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("AUC-ROC:", roc_auc_score(y_test, y_prob))

# Visualisasi ROC Curve
fpr, tpr, thresholds = roc_curve(y_test, y_prob)
plt.figure()
plt.plot(fpr, tpr, label="Naive Bayes (AUC = {:.2f})".format(roc_auc_score(y_test, y_prob)))
plt.plot([0, 1], [0, 1], linestyle="--", color="gray")
plt.title("ROC Curve")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.legend()
plt.show()
```

```
Accuracy: 1.0
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         1
           1       1.00      1.00      1.00         2

    accuracy                           1.00         3
   macro avg       1.00      1.00      1.00         3
weighted avg       1.00      1.00      1.00         3

Confusion Matrix:
 [[1 0]
 [0 2]]
AUC-ROC: 1.0
```



CONCLUSION

The results of this study indicate that the use of the Naïve Bayes method for classifying lupus disease yields satisfactory outcomes. This model successfully identifies and classifies patients at risk of developing lupus, although there are limitations related to the assumption of feature independence, which may not always hold true in a medical context. Evaluations using metrics such as accuracy, precision, recall, and F1-Score demonstrate good performance, despite challenges posed by an imbalanced dataset.

The main advantages of Naïve Bayes include its speed in model training and its ability to handle large datasets without requiring high computational resources. However, if features in the dataset exhibit strong dependencies, alternative models such as Random Forest or Support Vector Machines may prove to be more effective. Testing results also indicate that the F1-Score better represents the model's quality under class imbalance conditions, while the AUC-ROC metric illustrates the model's ability to distinguish between positive and negative classes.

While Naïve Bayes shows good performance, it is essential to consider its limitations in generalizing to complex data. Therefore, the use of more advanced methods should be considered to enhance diagnostic accuracy.

RECOMMENDATIONS

1. Improving Data Quality: One important factor in enhancing the performance of the Naïve Bayes model is the quality of the data used. Incomplete data, missing values, or non-standardized data can significantly affect classification results. Therefore, it is recommended to carefully clean and prepare the data, including handling missing values and normalizing the data, to enable the model to operate more effectively.
2. Using Larger and More Diverse Datasets: Larger and more diverse datasets can provide a better representation of the variations of lupus cases in a broader population. Data that includes various age groups, genders, and genetic or environmental factors will help the model recognize more comprehensive patterns and improve prediction generalization. With a more varied dataset, the Naïve Bayes model can provide more accurate predictions applicable to a wider population.
3. Exploring Other Techniques: Although Naïve Bayes is effective in many cases, other classification techniques such as Support Vector Machines (SVM), Decision Trees, or Random Forest may be more suitable for handling data with dependent features. Therefore, it is advisable to compare the performance of Naïve Bayes with other methods, which may yield better results in specific cases, especially when relationships among features are stronger.
4. Enhancing Model Interpretability: In medical applications, it is crucial to ensure that the model's results can be clearly explained to medical professionals or patients. Although Naïve Bayes tends to be more interpretable compared to other methods like deep learning, researchers should still strive to improve transparency and understanding of how the model makes decisions, for example by visualizing probability distributions or using interpretability-based explanation techniques.
5. Testing on New and Different Datasets*: To ensure that the developed model is robust and widely applicable, testing on datasets different from those used for training is essential. Evaluating the model on data from other hospitals or clinics, or on data collected over different time periods, can provide better insight into the model's performance in real-world situations. This way, we can assess how well the model generalizes to new data and improve predictions for patients not included in the training dataset.

FURTHER STUDY

Future research is expected to further explore this material.

REFERENCES

Bishop, C. M. (2006). Pattern Recognition and Machine Learning. Springer. Buku ini memberikan dasar teori yang kuat tentang metode pembelajaran mesin, termasuk Naïve Bayes.

Chawla, N.V., & Bowyer, K.W. (2002). SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*, 16, 321-357. Artikel ini membahas teknik penanganan ketidakseimbangan kelas yang dapat diterapkan dalam model Naïve Bayes.

Dua, D., & Graff, C. (2019). UCI Machine Learning Repository [Online]. University of California, Irvine, School of Information and Computer Sciences. Sumber dataset yang digunakan untuk penelitian dalam bidang pembelajaran mesin.

Friedman, J.H., Hastie, T., & Tibshirani, R.J. (2001). The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer Series in Statistics. Buku ini memberikan wawasan mendalam tentang teknik statistik dan pembelajaran mesin.

Han, J., Kamber, M., & Pei, J. (2012). Data Mining: Concepts and Techniques. Morgan Kaufmann. Buku ini membahas teknik-teknik pengolahan data yang relevan dengan algoritma Naïve Bayes.

Kohavi, R., & Provost, F. (1998). Glossary of Terms. *Machine Learning*, 30(2), 271-274. Artikel ini mencakup istilah-istilah yang sering digunakan dalam pembelajaran mesin dan klasifikasi.

Liu, H., & Motoda, H. (1998). Feature Selection for Knowledge Discovery and Data Mining. Kluwer Academic Publishers. Buku ini menjelaskan pentingnya pemilihan fitur dalam model pembelajaran mesin.

Rish, I. (2001). An Empirical Comparison of Supervised Learning Algorithms. *Proceedings of the 2001 International Conference on Artificial Intelligence*. Penelitian ini membandingkan berbagai algoritma pembelajaran mesin, termasuk Naïve Bayes.

Witten, I.H., Frank, E., & Hall, M.A. (2016). Data Mining: Practical Machine Learning Tools and Techniques (4th ed.). Morgan Kaufmann Publishers Inc.. Buku ini menawarkan panduan praktis dalam penerapan berbagai algoritma pembelajaran mesin termasuk Naïve Bayes.

Zhang, H. (2004). The Optimality of Naive Bayes. *Proceedings of the 17th International Florida Artificial Intelligence Research Society Conference*, 562-567. Artikel ini membahas keunggulan dan kelemahan dari algoritma Naïve Bayes.