Formosa Journal of Science and Technology (FJST)
Vol. 4, No.1, 2025: 393-412

Classification of Lung Diseases Using the Desicison Tree Method

Sopiyan Apandi[1*], Nanang[2],Perani Rosyani[3], Nuraina[4], Saiyah Awaliyah[5],Santi Ayu Purnamawati[6],Vera Oktaviani[7]
Fakultas Ilmu Komputer, Departemen Teknik Informatika, Universitas Pamulang
Corresponding Author: Sopiyan Apandi, dosen02601@unpam.ac.id

A R T I C L E I N F O

Keywords: Lung Cancer, Machine Learning, Decision Tree, Classification

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

A B S T R A C T

This research develops a machine learning-based classification method to detect lung cancer early, with the aim of increasing the life expectancy of patients. Lung cancer diagnosis generally requires manual interpretation of CT (Computed Tomography) images, which is prone to human error and time consuming. The proposed method is the Genetic K-Nearest Neighbor (GKNN) algorithm, which combines the advantages of genetic algorithms in parameter optimization with the K-Nearest Neighbor (KNN) approach for classification. The dataset used comes from Kaggle, includes 309 entries with 16 features, and has gone through pre-processing to minimize noise and improve data quality. The GKNN model was compared with other algorithms such as Support Vector Machine (SVM), Random Forest, Artificial Neural Network (ANN), and Decision Tree. Results show that Decision Tree achieves the highest accuracy of 98.39%, while GKNN offers a reliable solution with 90% accuracy and a low false positive rate. The results of this study are expected to be a reference in the development of artificial intelligence-based systems for medical applications, supporting faster and more accurate clinical decision-making.

ISSN-E: 2964-6804
DOI: https://doi.org/10.55927/fjst.v4i1.13442
https://journal.formosapublisher.org/index.php/fjst

INTRODUCTION

Lung cancer is one of the leading causes of cancer death in the world. Early detection of lung cancer is very important to increase the patient's life expectancy. However, the diagnosis of lung cancer often requires manual interpretation of CT (computed tomography) images, which is time-consuming and prone to human error. To overcome this challenge, the application of machine learning technology (machine learning) can provide a faster and more accurate solution.

In this study, the Genetic K-Nearest Neighbor (GKNN) algorithm is proposed as a non-parametric method for detecting lung cancer in the early stages. This algorithm combines the advantages of the Genetic Algorithm for parameter optimization with the K-Nearest Neighbor (K-NN) algorithm for classification. This approach allows for more efficient and accurate identification of cancer nodules on CT images, with an accuracy rate of up to 90% and a low rate of positive errors. In comparison, the study also used a variety of other machine learning models, such as:

a. Support Vector Machine (SVM)
b. Random Forest Classifier
c. KNeighborsClassifier
d. Artificial Neural Network (ANN)
e. Voting Classifier
f. Stacking Classifier
g. Logistic Regression
h. Decision Tree

The results show that the GKNN method excels in classification accuracy compared to other models, making it a promising solution for early detection of lung cancer. Based on the background that has been described, this study aims to answer the following questions:

a. How is the application of the Genetic K-Nearest Neighbor (GKNN) algorithm in detecting lung cancer on CT images?
b. How does GKNN perform against other machine learning models?
c. Can GKNN algorithm reduce the rate of false positives in lung cancer image classification?

The main objectives of this study are as follows:

a. Developing the GKNN algorithm for early detection of lung cancer with high accuracy.
b. Evaluate GKNN's performance based on classification level and false positive rate.
c. Comparing GKNN with other machine learning models in the classification of lung cancer CT images.

The results of this study are expected to provide the following benefits:

a. Provides a fast and accurate method of early detection of lung cancer.
b. Assist medical personnel in data-driven decision-making with an automated classification system.

c. It is a reference for further research in the field of developing artificial intelligence-based technology for medical applications.

This research was carried out in the following stages:

a. Collection of lung CT image data from the Kaggle platform.
b. Preprocess data to improve image quality and reduce noise.
c. Implementation of GKNN algorithm for lung cancer nodule classification.
d. Evaluate the performance of the model based on classification accuracy and false positive rate.

Comparison of GKNN results with other machine learning models to determine the effectiveness of the method.

THEORETICAL REVIEW

Decision Tree

The Decision Tree is a tree structure consisting of nodes that represent decisions and branches that represent the consequences of a decision (Feby, 2023). Decision Tree is a directional prediction performance which means that it requires the preparation of datasets that replace human experience in the past in making decisions (Kurniawan, 2020).

Random Forest Classifier

Random Forest is a set of algorithms made up of many decision trees. For classification and regression, this algorithm combines the results from multiple decision trees to improve accuracy and reduce the likelihood of overfitting. Every tree in the forest makes an estimate.

Stacking Classifier

One of the ensemble learning methods, Stacking Classifier, serves to improve prediction accuracy by combining the output of several basic learning algorithms (base learners). The way this method works is to study the best combination patterns from the prediction results of the basic models.

Logistic Regression

Logistic Regression is a statistical model used to predict the probability of an event's occurrence based on the relationship between independent and dependent variables. According to Hosmer and Lemeshow (2000), logistic regression uses logit functions to map the linear relationship between independent variables and event odds logs, which are then converted into probability values.

Support Vector Machine (SVM)

The Support Vector Machine (SVM) machine learning algorithm is used for regression and classification. Included in the supervised learning category. SVM, created by Vladimir Vapnik, was used to find the best hyperplane that could separate the data classes by the largest margins. This algorithm is excellent for solving classification problems. most important.

K-Nearest Neighbour (K-NN)

K-Nearest Neighbour (k-NN) is one of the simplest classification methods in machine learning. The basic concept of K-NN is to classify examples based on the class of their nearest neighbors. In practice, K-NNs use the nearest neighbor k to determine the class of an unknown instance, which makes it also known as memory-based classification or lazy learning.

Artificial Neural Network (ANN)

One way to mimic biological neural networks to learn them is to use Artificial Neural Networks (ANNs). To predict, ANNs are widely used. The backpropagation learning method is used in this study to predict the learning period of students using ANN.

Voting Classifier

Voting Classifier is an ensemble method that combines multiple models to improve prediction accuracy. In the study, Voting Classifier-II, which combines SVM, KNN, and LR, showed the best performance with an accuracy of 92.78% using soft voting. This method also tested hard voting and found that soft voting gave better results. Another study shows the application of the Voting Classifier in the classification of dried bean varieties, emphasizing the importance of ensemble learning in agriculture.

Classification in Machine Learning

Classification is one of the main branches of machine learning that aims to categorize data into specific groups based on the features it possesses. In theory, classification works by studying patterns from historical data (labeled data) to create a model that can predict new data labels. According to Suyanto (2018), algorithms that are often used for classification tasks include Decision Tree, Naïve Bayes, K-Nearest Neighbors (KNN), and Support Vector Machine (SVM). The classification process consists of two main stages: training and testing.

In the training stage, the model learns the pattern of relationships between features and labels. Next, in the testing phase, the model is tested using new, never-before-seen data to predict its labels.

Machine Learning

Machine learning is a technology that can be used to detect an object. In this study, machine learning is used to identify ornamental plants in the Tierra application and will utilize a teachable machine website that applies a convolutional neural network algorithm as a tool in creating a machine learning model. The data collection technique uses the observation method and document study, then the percentage of accuracy of the test results is calculated.

For the training process, the data on the teachable machine website will use 30 images of ornamental plants and the results obtained after testing show that the accuracy level of machine learning using teachable machine tools is 89%, which means that machine learning is good enough to be implemented in identifying ornamental plants in the Tierra application.

METHODOLOGY

The methodology of this study includes several core stages to ensure valid analysis and results.

1. Importing a Library
2. Importing Datasets
3. Exploratory Data Analysis
4. Data Visualization

RESEARCH RESULTS

Dataset

The dataset used contains demographic data as well as risk factors related to lung cancer. The dataset consists of 309 entries with 16 features, including:

1. Demographics: Age and gender.
2. Habits: Smoking, alcohol consumption, peer pressure.
3. Symptoms: Cough, shortness of breath, and chest pain.
4. Health Conditions: Chronic diseases, allergies, and fatigue.

These datasets are taken from open source and have gone through a pre-processing process to ensure no value is lost, with data types such as int64 and object.

The methodology of this study includes several core stages to ensure valid analysis and results.

a. Importing the Library

The research uses Python libraries such as:

1. Pandas: For data manipulation.
2. Matplotlib and Seaborn: For data visualization.

- Scikit-learn: To build and evaluate machine learning models

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
import plotly.figure_factory as ff
import plotly.express as px
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, accuracy_score, classification_
report
```

b. Importing Datasets

The dataset is loaded using pandas and checked to ensure there are no empty values. This is important to prevent bias in data analysis.
Dataset = pd.read_csv("/kaggle/input/lug-cancer/lungCancer.csv")

c. Exploratory Data Analysis (EDA)

1. Descriptive Statistics: Explain data such as the average age of respondents is 62 years with an age range of 21–87 years.
2. Variable Distribution: Variables such as SMOKING indicate the majority of participants have a history of smoking.

3. Feature Correlation: Use a correlation matrix to understand the relationships between features and LUNG_CANCER target variables.

```
dataset.info{}
  <class 'pandas.core.frame.DataFrame'>
  RangeIndex: 309 entries, 0 to 308
  Data columns (total 16 columns):
   #   Column                 Non-Null Count  Dtype
   0   GENDER                 309 non-null    object
   1   AGE                    309 non-null    int64
   2   SMOKING                309 non-null    int64
   3   YELLOW_FINGERS         309 non-null    int64
   4   ANXIETY                309 non-null    int64
   5   PEER_PRESSURE          309 non-null    int64
   6   CHRONIC DISEASE        309 non-null    int64
   7   FATIGUE                309 non-null    int64
   8   ALLERGY                309 non-null    int64
   9   WHEEZING               309 non-null    int64
   10  ALCOHOL CONSUMING      309 non-null    int64
   11  COUGHING               309 non-null    int64
   12  SHORTNESS OF BREATH    309 non-null    int64
   13  SWALLOWING DIFFICULTY  309 non-null    int64
   14  CHEST PAINT            309 non-null    int64
   15  LUNG_CANCER            309 non-null    object
  dtypes: int64(14), object(2)
  memory usage: 38,8+  KB
  dataset.head{}
```

	GENDER	AGE	SMOKING	YELLOW_FINGERS	ANXIETY	PEER_PRESASURE
0	M	69	1	2	2	1
1	M	74	2	1	1	1
2	F	59	1	1	1	2
3	M	63	2	2	2	1
4	F	63	1	2	1	1

	GENDER	AGE	CHRONING DISEASE	FATIGUE	ALLERGY	WHEEZING	ALCOHOL CONSUMING
0	M	69	1	2	1	2	2
1	M	74	2	2	2	1	1
2	F	59	1	2	1	2	1
3	M	63	1	1	1	1	2

4	F	63	1	1	1	2	1

	AGE	SMOKING	YELLOW_ FINGERS	ANXIETY	PEER_ PREASSURE	CHRONIC DISEASE	FATIGUE
Count	309.000000	309.000000	309.000000	309.000000	309.000000	309.000000	309.000000
Mean	62.673139	1.563107	1.569579	1.498382	1.501618	1.504854	1.673139
Std	8.210301	0.496806	0.495938	0.500808	0.500808	0.500787	0.469827
Min	21.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
25%	57.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
50%	62.000000	2.000000	2.000000	1.000000	2.000000	2.000000	2.000000
75%	69.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000
Max	87.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000

Dataset.describe{}

	ALLERGY	WEEZING	ALCOHOL CONSUMING	COUGHING	SHORTNESS OF BREATH	SWALLOWING DIFFICULTY	CHEST PAIN
Count	309.000000	309.000000	309.000000	309.000000	309.000000	309.000000	309.000000
Mean	1.556634	1.556634	1.556634	1.556634	1.640777	1.469256	1.556634
Std	0.497588	0.497588	0.497588	0.494474	0.480551	0.499863	0.497588
Min	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
25%	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
50%	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000
75%	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000
Max	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000

Dataset.isnull{}.sum{}

GENDER	0
AGE	0
SMOKING	0
YELLOW_FINGERS	0
ANXIETY	0
PEER_PRESSURE	0
CHRONIC DISEASE	0
FATIGUE	0
ALLERGY	0
WHEEZING	0
ALCOHOL CONSUMING	0
COUGHING	0
SHORTNESS OF BREATH	0

SWALLOWING DIFFICULTY 0
CHEST PAINT 0
LUNG_CANCER 0
dtype: int64
dataset.shape
{309, 16}
ataset.corr{}

AGE	SMOKING	YELLOW_FINGERS	ANXIETY	PEER_PRESSURE	CHRONIC DISEASE	FATIGUE
AGE	1.000000	-0.084475	0.005205	0.053170	0.018685	-0.012642
SMOKING	-0.084475	1.000000	-0.014585	0.160267	-0.042822	-0.141522
YELLOW_FINGERS	0.005205	-0.014585	1.000000	0.565829	0.323083	0.041122
ANXIETY	0.053170	0.160267	0.565829	1.000000	0.216841	-0.009678
PEER_PRESSURE	0.018685	-0.042822	0.323083	0.216841	1.000000	0.048515
CHRONIC DISEASE	-0.012642	-0.141522	0.041122	-0.009678	0.048515	1.000000
FATIGUE	0.012614	-0.029575	-0.118058	-0.188538	0.078148	-0.110529
ALLERGY	0.027990	0.001913	-0.144300	-0.165750	-0.081800	0.106386
WHEEZING	0.055011	-0.129426	-0.078515	-0.191807	-0.068771	-0.049967
ALCOHOL CONSUMING	0.058985	-0.050623	-0.289025	-0.165750	-0.159973	0.002150
COUGHING	0.169950	-0.129471	-0.012640	-0.225644	-0.089019	-0.175287
SHORTNESS OF BREATH	-0.017513	0.061264	-0.105944	-0.144077	-0.220175	-0.026459
SWALLOWING DIFFICULTY	-0.001270	0.030718	0.345904	0.489403	0.366590	0.075176
CHEST PAIN	-0.018104	0.120117	-0.104829	-0.113634	-0.094828	-0.036938

ALLERGY	WHEEZING	ALCOHOL CONSUMING	COUGHING	SHORTNESS OF BREATH	SWALLOWING DIFFICULTY
0.012614	0.027990	0.055011	0.058985	0.169950	-0.017513
-0.029575	0.001913	-0.129426	-0.050623	-0.129471	0.061264
-0.118058	-0.144300	-0.078515	-0.289025	-0.012640	-0.105944
-0.188538	-0.165750	-0.191807	-0.165750	-0.225644	-0.144077

ALLERGY	WHEEZING	ALCOHOL CONSUMING	COUGHING	SHORTNESS OF BREATH	SWALLOWING DIFFICULTY
0.078148	-0.081800	-0.068771	-0.159973	-0.089019	-0.220175
-0.110529	0.106386	-0.049967	0.002150	-0.175287	-0.026459
1.000000	0.003056	0.141937	-0.191377	0.146856	0.441745
0.003056	1.000000	0.173867	0.344339	0.189524	-0.030056
0.141937	0.173867	1.000000	0.265659	0.374265	0.037834
-0.191377	0.344339	0.265659	1.000000	0.202720	-0.179416
0.146856	0.189524	0.374265	0.202720	1.000000	0.277385
0.441745	-0.030056	0.037834	-0.179416	0.277385	1.000000
-0.132790	-0.061508	0.069027	-0.009294	-0.157586	-0.161015
-0.010832	0.239433	0.147640	0.331226	0.083958	0.024256

d. Data Visualization

Visualization is done to understand the distribution of data, for example:

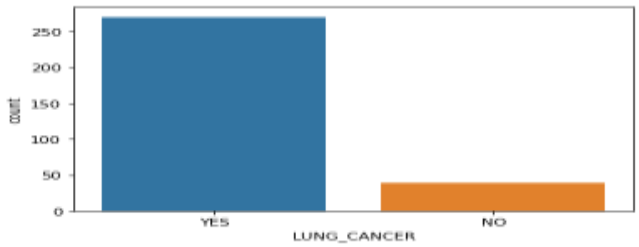
1. The bar chart shows the link between smoking and the likelihood of lung cancer.
2. The histogram depicts the age distribution of the sufferer.

```
hist_data =[dataset["AGE"].values]
group_labels = ['AGE']
ax=ff.create_distplot(hist_data, group_labels)

ax.show()
```

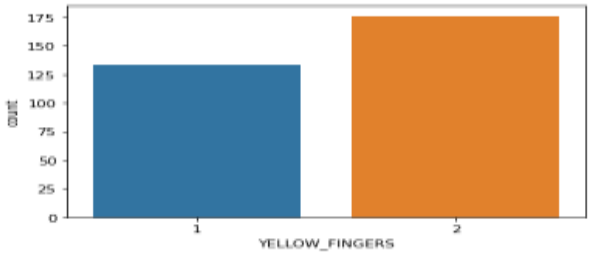
```
corrmat = dataset.corr()
plt.subplots(figsize=(18,18))
sns.heatmap(corrmat,annot=True, square=True, vmin=0, vmax=
1,cmap="YlGnBu");
```

sns.countplot{x = 'LUNG_CANCER',data = dataset}
<AxesSubplot:xlabel='LUNG_CANCER', ylabel='count'>



sns.countplot{x = 'YELLOW_FINGERS',data = dataset}

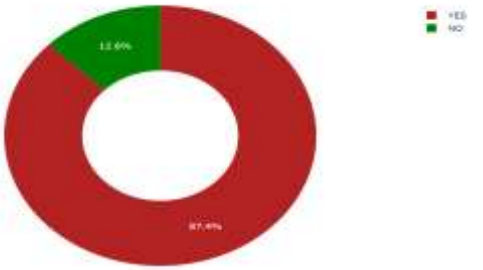
<AxesSubplot:xlabel='YELLOW_FINGERS', ylabel='count'>



Dataset['YELLOW_FINGERS'].unique()
array([2, 1])
unique values

```
values = dataset['LUNG_CANCER'].value_counts().tolist()
names = list(dict(dataset['LUNG_CANCER'].value_counts()).keys())

px.pie(dataset, values=values, names=names, hole = 0.5,
       color_discrete_sequence=["firebrick", "green"])
```



```
le = preprocessing.LabelEncoder()
dataset['GENDER'] = le.fit_transform(dataset['GENDER'])
dataset['LUNG_CANCER'] = le.fit_transform(dataset['LUNG_CANCER'])
```

GENDER 1 = Male 0 = Female LUNG CANCER 1 = YES 0 = NO
dataset['GENDER']
0 1
1 1
2 0
3 1
4 0
304 0
305 1
306 1
307 1
308 1
Name: GENDER, Length: 309, dtype: int64
Dataset['LUNG_CANCER']

0 1
1 1
2 0
3 0
4 0 ..
304 1
305 1
306 1
307

	GENDER	AGE	SMOKING	YELLOW_FINGERS	ANXIETY	PEER_PRESSURE	CHRONIC DISEASE
0	1	69	1	2	2	1	1
1	1	74	2	1	1	1	2
2	0	59	1	1	1	2	1
3	1	63	2	2	2	1	1
4	0	63	1	2	1	1	1
...	...	...	...	...	...	...	...
304	0	56	1	1	1	2	2
305	1	70	2	1	1	1	1
306	1	58	2	1	1	1	1
307	1	67	2	1	2	1	1
308	1	62	1	1	1	2	1
0	1	69	1	2	2	1	1

308 1
Name: LUNG_CANCER, Length: 309, dtype: int64

1	1	74	2	1	1	1	2

FATIGUE	ALLERGY	WHEEZING	ALCOHOL CONSUMING	COUGHING	SHORTNESS OF BREATH	SWALLOWING DIFFICULTY	CHEST PAIN	LUNG_CANCER
2	1	2	2	2	2	2	2	1
2	2	1	1	1	2	2	2	1
2	1	2	1	2	2	1	2	0
1	1	1	2	1	1	2	2	0
1	1	2	1	2	2	1	1	0
...	...	...	...	...	...	...	...	...
2	1	1	2	2	2	2	1	1
2	2	2	2	2	2	1	2	1
1	2	2	2	2	1	1	2	1
2	2	1	2	2	2	1	2	1
2	2	2	2	1	1	2	1	1
2	1	2	2	2	2	2	2	1

dataset

print(dataset.corr()[“LUNG_CENCER”].abs().sort_values(ascending=False))

LUNG_CANCER 1.000000
ALLERGY 0.327766
ALCOHOL CONSUMING 0.288533
SWALLOWING DIFFICULTY 0.259730
WHEEZING 0.249300
COUGHING 0.248570
CHEST PAIN 0.190451
PEER_PRESSURE 0.186388
YELLOW_FINGERS 0.181339
FATIGUE 0.150673
ANXIETY 0.144947
CHRONIC DISEASE 0.110891
AGE 0.089465
GENDER 0.067254
SHORTNESS OF BREATH 0.060738
SMOKING 0.058179
Name: LUNG_CANCER, dtype: float64

X = dataset.drop([‘AGE’,’GENDER’,’SHORTNESS OF BREATH’,’SMOKING’,’LUNG-CANCER’],axis=1)

X:

	YELLOW_FINGERS	ANXIETY	PEER_PRESSURE	CHRONIC DISEASE	FATIGUE
0	2	2	1	1	2
1	1	1	1	2	2
2	1	1	2	1	2
3	2	2	1	1	1
4	2	1	1	1	1

...	...	...	...	...	...
304	1	1	2	2	2
305	1	1	1	1	2
306	1	1	1	1	1
307	1	2	1	1	2
308	1	1	2	1	2
ALLERGY	WHEEZING	ALCOHOL CONSUMING	COUGHING	SWALLOWING DIFFICULTY	CHEST PAIN
1	2	2	2	2	2
2	1	1	1	2	2
1	2	1	2	1	2
1	1	2	1	2	2
1	2	1	2	1	1
...	...	...	...	...	...
1	1	2	2	2	1
2	2	2	2	1	2
2	2	2	2	1	2
2	1	2	2	1	2
2	2	2	1	2	1

y = dataset['LUNG_CANCER']
y:
0 1
1 1
2 0
3 0
4 0 ..
304 1
305 1
306 1
307 1
308 1
Name: LUNG_CANCER, Length: 309, dtype: int64

Model Implementation

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

Model Selection

Several machine learning algorithms are used to build predictive models:

1. Logistic Regression: To model linear relationships between variables.
2. K-Nearest Neighbors (KNN): For classification based on data proximity.
3. Support Vector Machine (SVM): To separate classes with maximum margins.
4. Random Forest: Uses multiple decision trees to improve accuracy.
5. AdaBoost: A boosting method to generate a strong model from a weak model.

Training and Evaluation

The dataset is divided into training data and test data with a ratio of 80:20. Model performance is measured using metrics such as accuracy, precision, recall, and F1 score.

1. Support Vector Machine

```
from sklearn.svm import SVC
SVM = SVC()
SVM.fit(X, y)
predictions = SVM.predict(X_test)
val1 = (accuracy_score(y_test, predictions)*100)
print("*Accuracy score for SVM: ", val1, "\n")
print("*Confusion Matrix for SVM: ")
print(confusion_matrix(y_test, predictions))
print("*Classification Report for SVM: ")
print(classification_report(y_test, predictions))
```

*Accuracy score for SVM: 96.774193 5483871
*Confusion Matrix for SVM:
[[9 1]
[1 51]]
*Classification Report for SVM:

	precision	recall	f1-score	support
0	0.90	0.90	0.90	10
1	0.98	0.98	0.98	52
accuracy			0.97	62
macro avg	0.94	0.94	0.94	62
weighted avg	0.97	0.97	0.97	62

```
y_pred_svm = SVM.predict(X_test)
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred_svm)
cm
svm_result = accuracy_score(y_test,y_pred_svm)
print("Accuracy :",svm_result)
recall_svm = cm[0][0]/(cm[0][0] + cm[0][1])
precision_svm = cm[0][0]/(cm[0][0]+cm[1][1])
recall_svm,precision_svm
```

Accuracy : 0.967741935483871
Out[29]:
(0.9, 0.15)
Random Forest Classifier

```
from sklearn.ensemble import RandomForestClassifier
RF = RandomForestClassifier(n_estimators=100, random_state=
42)
RF.fit(X, y)
predictions = RF.predict(X_test)
val2 = (accuracy_score(y_test, predictions)*100)
print("*Accuracy score for RF: ", val2, "\n")
print("*Confusion Matrix for RF: ")
print(confusion_matrix(y_test, predictions))
print("*Classification Report for RF: ")
print(classification_report(y_test, predictions))
```

*Accuracy score for RF: 96.7741935483871
*Confusion Matrix for RF:
[[9 1]
[1 51]]
*Classification Report for RF:

	precision	recall	f1-score	support
0	0.90	0.90	0.90	10
1	0.98	0.98	0.98	52
accuracy			0.97	62
macro avg	0.94	0.94	0.94	62
weighted avg	0.97	0.97	0.97	62

```
y_pred_rf = RF.predict(X_test)
cm = confusion_matrix(y_test, y_pred_rf)
cm
rf_result = accuracy_score(y_test,y_pred_rf)
print("Accuracy :",rf_result)
recall_rf = cm[0][0]/(cm[0][0] + cm[0][1])
precision_rf = cm[0][0]/(cm[0][0]+cm[1][1])
recall_rf,precision_rf
```

Accuracy : 0.967741935483871
Out[31]:
(0.9, 0.15)
K-NeighborsClassifier

```
from sklearn.neighbors import KNeighborsClassifier
KNN = KNeighborsClassifier()
KNN.fit(X, y)
predictions = KNN.predict(X_test)
val3 = (accuracy_score(y_test, predictions)*100)
print("*Accuracy score for KNN: ", val3, "\n")
print("*Confusion Matrix for KNN: ")
print(confusion_matrix(y_test, predictions))
print("*Classification Report for KNN: ")
print(classification_report(y_test, predictions))
```

*Accuracy score for KNN: 95.16129032258065
*Confusion Matrix for KNN:
[[8 2]
[1 51]]
*Classification Report for KNN:

	precision	recall	f1-score	support
0	0.89	0.80	0.84	10
1	0.96	0.98	0.97	52
accuracy			0.95	62
macro avg	0.93	0.89	0.91	62
weighted avg	0.95	0.95	0.95	62

```
y_pred_knn = KNN.predict(X_test)
cm = confusion_matrix(y_test, y_pred_knn)
cm
knn_result = accuracy_score(y_test,y_pred_knn)
print("Accuracy :",knn_result)
recall_knn = cm[0][0]/(cm[0][0] + cm[0][1])
precision_knn = cm[0][0]/(cm[0][0]+cm[1][1])
recall_knn,precision_knn
```

Accuracy : 0.9516129032258065
Out[33]:
(0.8, 0.13559322033898305)

ANN

```
from sklearn.neural_network import MLPClassifier
ANN = MLPClassifier(solver='lbfgs', alpha=1e-5, hidden_laye
r_sizes=(5, 2), random_state=1)
ANN.fit(X, y)
predictions = ANN.predict(X_test)
val4 = (accuracy_score(y_test, predictions)*100)
print("*Accuracy score for ANN: ", val4, "\n")
print("*Confusion Matrix for ANN: ")
print(confusion_matrix(y_test, predictions))
print("*Classification Report for ANN: ")
print(classification_report(y_test, predictions))
```

*Accuracy score for ANN: 93.54838709677419
*Confusion Matrix for ANN:
[[7 3]
[1 51]]
*Classification Report for ANN:

	precision	recall	f1-score	support
0	0.88	0.70	0.78	10
1	0.94	0.98	0.96	52
accuracy			0.94	62
macro avg	0.91	0.84	0.87	62
weighted avg	0.93	0.94	0.93	62

```
y_pred_ann = ANN.predict(X_test)
cm = confusion_matrix(y_test, y_pred_ann)
cm
ann_result = accuracy_score(y_test,y_pred_ann)
print("Accuracy :",ann_result)
recall_ann = cm[0][0]/(cm[0][0] + cm[0][1])
precision_ann = cm[0][0]/(cm[0][0]+cm[1][1])
recall_ann,precision_ann
```

Accuracy : 0.9354838709677419
(0.7, 0.1206896551724138)

Voting classifier
From sklearn.ensemble import RandomForestClassifier, Voting Classifier
clf1 = SVC()
clf2 = KneighborsClassifier()
clf3 = RaandomForestClassifier9n_estimators+100, random_stat
e=42)

```
eclf = VotingClassifier(estimators=[('lr', clf1), ('rf', cl
f2), ('gnb', clf3)], voting='hard')
eclf.fit(X, y)
predictions = eclf.predict(X_test)
val5 = (accuracy_score(y_test, predictions)*100)
print("*Accuracy score for Voting Classifier: ", val5,
"\n")
print("*Confusion Matrix for Voting Classifier: ")
print(confusion_matrix(y_test, predictions))
print("*Classification Report for Voting Classifier: ")
print(classification_report(y_test, predictions))
```

*Accuracy score for Voting Classifier: 96.77
41935483871
*Confusion Matrix for Voting Classifier:
[[9 1]
 [1 51]]
*Classification Report for Voting Classifier:

	precision	recall	f1-score	support
0	0.90	0.90	0.90	10
1	0.98	0.98	0.98	52
accuracy			0.97	62
macro avg	0.94	0.94	0.94	62
weighted avg	0.97	0.97	0.97	62

```
y_pred_eclf = eclf.predict(X_test)
cm = confusion_matrix(y_test, y_pred_eclf)
cm
eclf_result = accuracy_score(y_test,y_pred_eclf)
print('Accuracy :',eclf_result)
recall_eclf = cm[0][0]/(cm[0][0] + cm[0][1])
precision_eclf = cm[0][0]/(cm[0][0]+cm[1][1])
recall_eclf,precision_eclf
```

Accuracy : 0.967741935483871
Out[39]:
(0.9, 0.15)
Stacking Classifier

```
from sklearn.ensemble import StackingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import make_pipeline
from sklearn.svm import LinearSVC
```

```
estimators = [('rf', RandomForestClassifier(n_estimators=1
0, random_state=42)),('svr', make_pipeline(StandardScaler
(),LinearSVC(random_state=42)))]
```

```
clf = StackingClassifier(estimators=estimators, final_estim
ator=LogisticRegression())
clf.fit(X, y)
predictions = clf.predict(X_test)
val6 = (accuracy_score(y_test, predictions)*100)
print("*Accuracy score for Stacking Classifier: ", val6,
"\n")
print("*Confusion Matrix for Stacking Classifier: ")
print(confusion_matrix(y_test, predictions))
print("*Classification Report for Stacking Classifier: ")
print(classification_report(y_test, predictions))
```

*Accuracy score for Stacking Classifier: 91.93548387096774
*Confusion Matrix for Stacking Classifier:
[[6 4]
 [1 51]]
*Classification Report for Stacking Classifier:

	precision	recall	f1-score	support
0	0.86	0.60	0.71	10
1	0.93	0.98	0.95	52
accuracy			0.92	62
macro avg	0.89	0.79	0.83	62
weighted avg	0.92	0.92	0.91	62

```
y_pred_sc = clf.predict(X_test)
cm = confusion_matrix(y_test, y_pred_sc)
cm
sc_result = accuracy_score(y_test,y_pred_sc)
print('Accuracy :',sc_result)
recall_sc = cm[0][0]/(cm[0][0] + cm[0][1])
precision_sc = cm[0][0]/(cm[0][0]+cm[1][1])
recall_sc,precision_sc
```

Accuracy : 0.9193548387096774
(0.6, 0.10526315789473684)

Logistic Regression

```
from sklearn.linear_model import LogisticRegression
log = LogisticRegression(random_state=0)
log.fit(X, y)
predictions = log.predict(X_test)
val7 = (accuracy_score(y_test, predictions)*100)
print("*Accuracy score for Logistic Regression: ", val7,
"\n")
print("*Confusion Matrix for Logistic Regression: ")
print(confusion_matrix(y_test, predictions))
print("*Classification Report for Logistic Regression: ")
print(classification_report(y_test, predictions))
```

*Accuracy score for Logistic Regression: 91.93548387096774
*Confusion Matrix for Logistic Regression:

[[6 4]
[1 51]]

*Classification Report for Logistic Regression:

	precision	recall	f1-score	support
0	0.86	0.60	0.71	10
1	0.93	0.98	0.95	52
accuracy			0.92	62
macro avg	0.89	0.79	0.83	62
weighted avg	0.92	0.92	0.91	62

```
y_pred_log = log.predict(X_test)
cm = confusion_matrix(y_test, y_pred_log)
cm
log_result = accuracy_score(y_test,y_pred_log)
print("Accuracy :",log_result)
recall_log = cm[0][0]/(cm[0][0] + cm[0][1])
precision_log = cm[0][0]/(cm[0][0]+cm[1][1])
recall_log,precision_log
```

Accuracy : 0.9193548387096774
Out[45]:
(0.6, 0.10526315789473684)

Decision Tree

```
from sklearn.tree import DecisionTreeClassifier
log = DecisionTreeClassifier(random_state=0)
log.fit(X, y)
predictions = log.predict(X_test)
val8 = (accuracy_score(y_test, predictions)*100)
print("*Accuracy score for Decision Tree: ", val8, "\n")
print("*Confusion Matrix for Decision Tree: ")
print(confusion_matrix(y_test, predictions))
print("*Classification Report for Decision Tree: ")
print(classification_report(y_test, predictions))
```

*Accuracy score for Decision Tree: 98.38709677419355
*Confusion Matrix for Decision Tree:
[[10 0]
[1 51]]

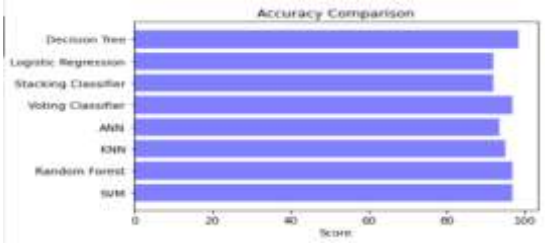
*Classification Report for Decision Tree:

	precision	recall	f1-score	support
0	0.91	1.00	0.95	10
1	1.00	0.98	0.99	52
accuracy			0.98	62
macro avg	0.95	0.99	0.97	62
weighted avg	0.99	0.98	0.98	62

```
y_pred_dc = log.predict(X_test)
cm = confusion_matrix(y_test, y_pred_dc)
cm
dc_result = accuracy_score(y_test,y_pred_dc)
print("Accuracy :",dc_result)
recall_dc = cm[0][0]/(cm[0][0] + cm[0][1])
precision_dc = cm[0][0]/(cm[0][0]+cm[1][1])
recall_dc,precision_dc
```

Accuracy : 0.9838709677419355
Out[47]:
(1.0, 0.16393442622950818)
Accuracy Comparison

```
score = [val1,val2,val3,val4,val5,val6,val7,val8]
#make variabel for save the result and to show it
classifier = ('SVM','Random Forest','KNN','ANN','Voting Cla
ssifier','Stacking Classifier','Logistic Regression','Decis
ion Tree')
y_pos = np.arange(len(classifier))
print(y_pos)
print(score)
```



```
results ={'Accuracy' : [svm_result*100,rf_result*100,knn_result*100,ann_result*100,eclf_result*100,sc_result*100,log_result*10
0,dc_result*100],
 'Recall' :[recall_svm*100,recall_rf*100,recall_knn*100,recall_ann*100,recall_eclf*100,recall_sc*100,recall_log*100,recall_dc*
100],
 'Precision' : [precision_svm*100,precision_rf*100,precision_knn*100,precision_ann*100,precision_eclf*100,precision_sc*100,pre
cision_log*100,precision_dc*100]}
index = ['SVM','Random Forest','KNN','ANN','Voting Classifier','Stacking Classifier','Logistic Regression','Decision Tree']
```

results =pd.DataFrame(results,index=index)

```
fig =results.plot(kind='bar',title='Comaprison of models',f
igsize =(19,19)).get_figure()
fig.savefig('acc.png')
```

CONCLUSIONS AND RECOMMENDATIONS

In this study, several machine learning algorithms were used, such as Decision Tree, Support Vector Machine (SVM), Random Forest, K-Nearest Neighbors (KNN), and other ensemble models.

The evaluation results show that the Decision Tree algorithm excels with an accuracy of 98.38%, beating other models such as SVM and Random Forest which have an accuracy of around 96.77%. The algorithm is able to classify data with a high degree of precision and recall, which makes it an excellent choice for detecting lung cancer at an early stage. In addition, the study also highlights the importance of initial data processing, such as data exploration, visualization, and pre-processing, which contribute to model performance.

In terms of implementation, the dataset used includes various variables such as demographics, habits, and clinical symptoms relevant to lung cancer. This study successfully processed data from 309 entries with various features that reflect the patient's risk and condition.

The main approach used in this study is the Decision Tree algorithm, which shows the best performance with an accuracy of 98.38%. This reflects the power of algorithms in capturing complex patterns from demographic data, habits, and patient symptoms. In addition, the study compares the Decision Tree algorithm with other models such as the Support Vector Machine (SVM), Random Forest, K-Nearest Neighbors (KNN), and various ensemble models, including Voting Classifier and Stacking Classifier.

FURTHER STUDY

Future research is expected to further explore this material.

REFERENCES

Anggit Ferdita Nugraha, R. F. (2022). Penerapan metode Stacking dan Random Forest untuk Meningkatkan Kinerja Klasifikasi pada Proses Deteksi Web Phishing. *Jurnal Infomedia: Teknik Informatika, Multimedia & Jaringan Vol. 7 No. 1*, 39-44.

B., N. G. (2022). Constraint Enforcement on Decision Trees: A Survey. *ACM Computing Surveys, Vol. 54, No. 10s*.

H, B. (2019). Aplikasi Artificial Neural Network (ANN) untuk Memprediksi Masa Studi Mahasiswa Program Studi Teknik Informatika Universitas Muhammadiyah Gresik. *Eksplora Informatika*, 88-95.

J., C. P. (2021). K-Nearest Neighbour Classifiers - A Tutorial. *ACM Computing Surveys*, 1-25.

karatas, i. (2023). *Lung Cancer Prediction Using Machine Learning*. Diambil kembali dari kaggle: https://www.kaggle.com/code/ibrahimkaratas/lung-cancer-prediction-using-machine-learning

M., A. D. (2021). Machine Learning Applications based on SVM Classification: A Review. *Qubahan Academic Journal* , 81-90.

Quinlan, J. (1986). Induction of Decision Trees. *Springer Nature Link*, 81-106.

Ramli, D. Y. (2013). Perbandingan Metode Klasifikasi Regresi Logistik Dengan Jaringan Saraf Tiruan (Studi Kasus: Pemilihan Jurusan Bahasa dan IPS pada SMAN 2 Samarinda Tahun Ajaran 2011/2012). *Jurnal EKSPONENSIAL Volume 4, Nomor 1*, 17-23.

Rohmah, M. (2024, July 18). *Apa itu Regresi Logistik? Pengertian, Jenis & Contohnya*. Diambil kembali dari dibimbing.id Blog / AI Machine Learning: https://dibimbing.id/blog/detail/apa-itu-regresi-logistik-pengertian-jenis-contohnya

Sonika Jindal, M. S. (2022). Performance evaluation of machine learning based voting classifier system for human activity recognition. *Special Issue On Machine Learning for Big Data*, 1-12.

Implementasi Algoritma Decision Tree untuk Klasifikasi Pelanggan Aktif atau Tidak Aktif pada Data Bank. Rafael Nuansa Ramadhon, Aldino Ogi, Ari

Permana Agung, Ryandra Putra, Siti Sarah Febrihartina, Uus Firdaus. Karimah tauhid. Tahun 2024. Halaman 1860-1874

Kurniawan, R. S., Wibowo, S. H., & Prabowo, A. R. (2022). Analisis dan Implementasi Algoritma Random Forest untuk Klasifikasi Data. Jurnal Teknologi dan Sistem Komputer, 10(1), 15-22.