MINI PROJECT
DIGITAL SKILL FAIR DATA SCIENCE 35.0

EVALUATING THE PERFORMANCE OF SEVERAL CLASSIFICATION ALGORITHMS FOR BREAST CANCER PREDICTION

Ketrin Natasya Stefany

Get to Know

BREAST CANCER

Breast cancer is a type of cancer that starts in the breast and while most lumps are benign (non-cancerous) rather than malignant (cancerous), certain benign lumps can increase the risk of developing breast cancer.

10 Top Cancer Types in Female for Estimated New Cancer Cases and Deaths, 2024

©2024, American Cancer Society. Inc, Survellance and Health Wquity Science

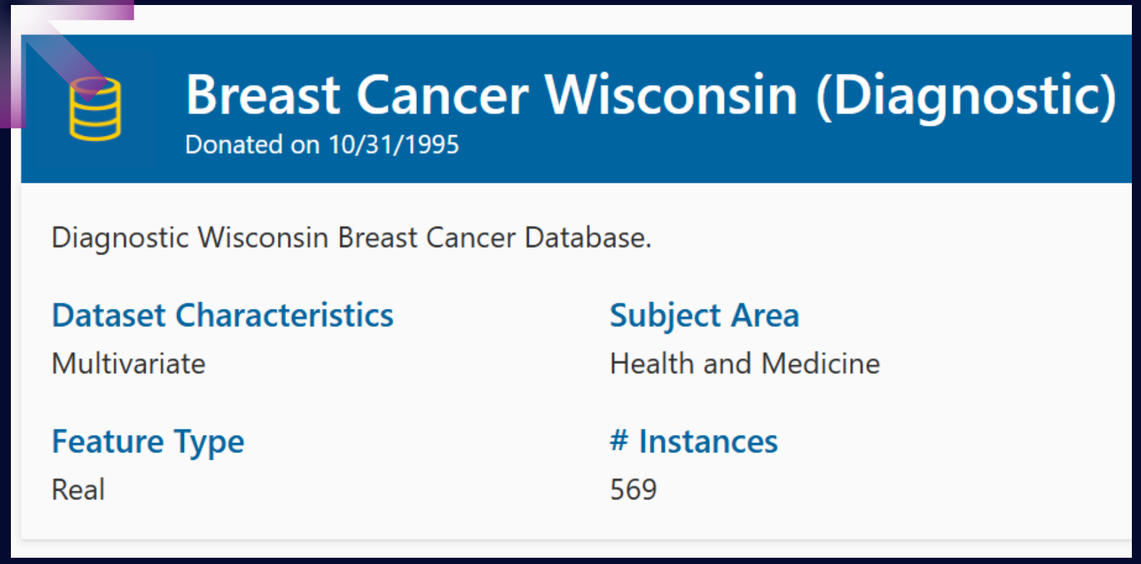
Estimated New Cases

Female		
Breast	310,720	32%
Lung & bronchus	118,270	12%
Colon & rectum	71,270	7%
Uterine corpus	67,880	7%
Melanoma of the skin	41,470	4%
Non-Hodgkin lymphoma	36,030	4%
Pancreas	31,910	3%
Thyroid	31,520	3%
Kidney & renal pelvis	29,230	3%
Leukemia	26,320	3%
All sites	**972,060**	

Estimated Deaths

Female		
Lung & bronchus	59,280	21%
Breast	42,250	15%
Pancreas	24,480	8%
Colon & rectum	24,310	8%
Uterine corpus	13,250	5%
Ovary	12,740	4%
Liver & intrahepatic bile duct	10,720	4%
Leukemia	10,030	3%
Non-Hodgkin lymphoma	8,360	3%
Brain & other nervous system	8,070	3%
All sites	**288,920**	

ASSETS



Datasets

The dataset used in this project is the **Breast Cancer (Diagnostic) Dataset** provided by scikit-learn, which contains 569 instances of breast tumor data collected by the University of Wisconsin, Madison.

Tools and Libraries

Python and its powerful libraries like scikit-learn, pandas, NumPy, matplotlib, and seaborn provide essential tools for data processing, visualization, and machine learning.

```
##Informasi tentang Dataframe
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 31 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   mean radius             569 non-null    float64
 1   mean texture            569 non-null    float64
 2   mean perimeter          569 non-null    float64
 3   mean area               569 non-null    float64
 4   mean smoothness         569 non-null    float64
 5   mean compactness        569 non-null    float64
 6   mean concavity          569 non-null    float64
 7   mean concave points     569 non-null    float64
 8   mean symmetry           569 non-null    float64
 9   mean fractal dimension  569 non-null    float64
 10  radius error            569 non-null    float64
 11  texture error           569 non-null    float64
 12  perimeter error         569 non-null    float64
 13  area error              569 non-null    float64
 14  smoothness error        569 non-null    float64
 15  compactness error       569 non-null    float64
 16  concavity error         569 non-null    float64
 17  concave points error    569 non-null    float64
 18  symmetry error          569 non-null    float64
 19  fractal dimension error 569 non-null    float64
 20  worst radius            569 non-null    float64
 21  worst texture           569 non-null    float64
 22  worst perimeter         569 non-null    float64
 23  worst area              569 non-null    float64
 24  worst smoothness        569 non-null    float64
 25  worst compactness       569 non-null    float64
 26  worst concavity         569 non-null    float64
 27  worst concave points    569 non-null    float64
 28  worst symmetry          569 non-null    float64
 29  worst fractal dimension 569 non-null    float64
 30  target                  569 non-null    int64  
dtypes: float64(30), int64(1)
memory usage: 137.9 KB
```

DATA UNDERSTANDING

The dataset consists of 31 main variables, including **30 key features** that provide information for training machine learning models and **1 target label** that contains the final values the model will predict, with the class labels indicating whether the tumor is benign (0) or malignant (1).

DATA PREPROCESSING

1 MISSING VALUE

During this step, the data is inspected to identify any missing values. In this dataset, no missing data was found, allowing the preprocessing to proceed without further action.

```
#Memeriksa data hilang
df.isna().sum()
```

	0
mean radius	0
mean texture	0
mean perimeter	0
mean area	0
mean smoothness	0
mean compactness	0
mean concavity	0
mean concave points	0
mean symmetry	0
mean fractal dimension	0
radius error	0
texture error	0
perimeter error	0
area error	0
smoothness error	0
compactness error	0
concavity error	0
concave points error	0
symmetry error	0
fractal dimension error	0
worst radius	0
worst texture	0
worst perimeter	0
worst area	0
worst smoothness	0
worst compactness	0
worst concavity	0
worst concave points	0
worst symmetry	0
worst fractal dimension	0
target	0

dtype: int64

DATA PREPROCESSING

2 DATA DUPLICATE

This step involves checking for duplicate records in the dataset. After analysis, no duplicate entries were identified, so no additional handling was needed.

```
#Memeriksa data duplikat
df.duplicated().sum()
```

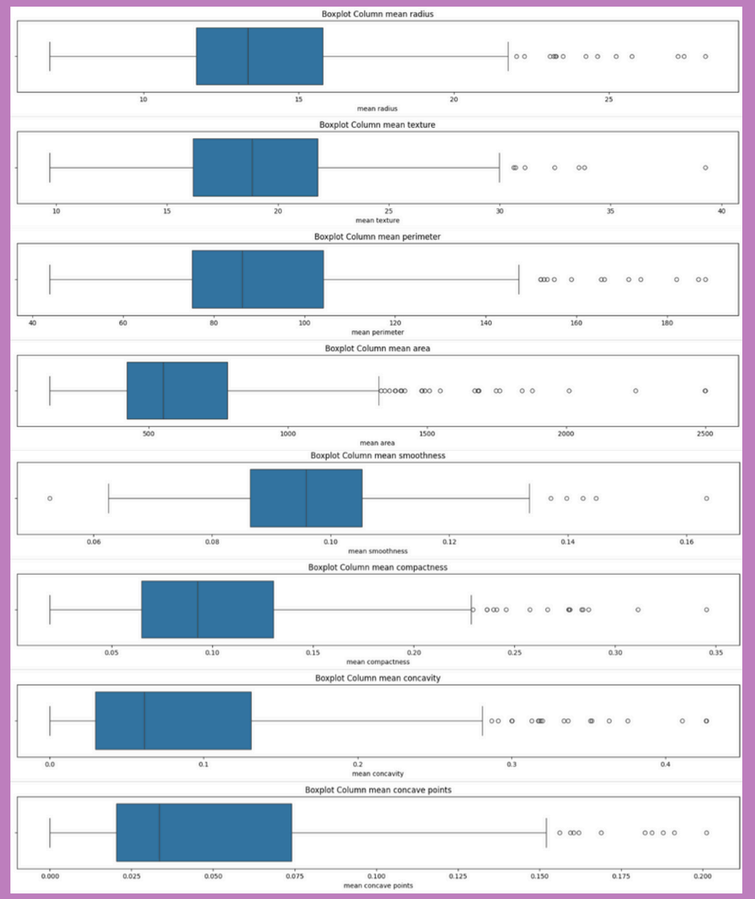
```
0
```

DATA PREPROCESSING

3 OUTLIER

Outliers in the dataset are identified using a boxplot visualization. It was observed that there are outliers present in the data, which can negatively impact model accuracy and need to be addressed to ensure robust analysis.

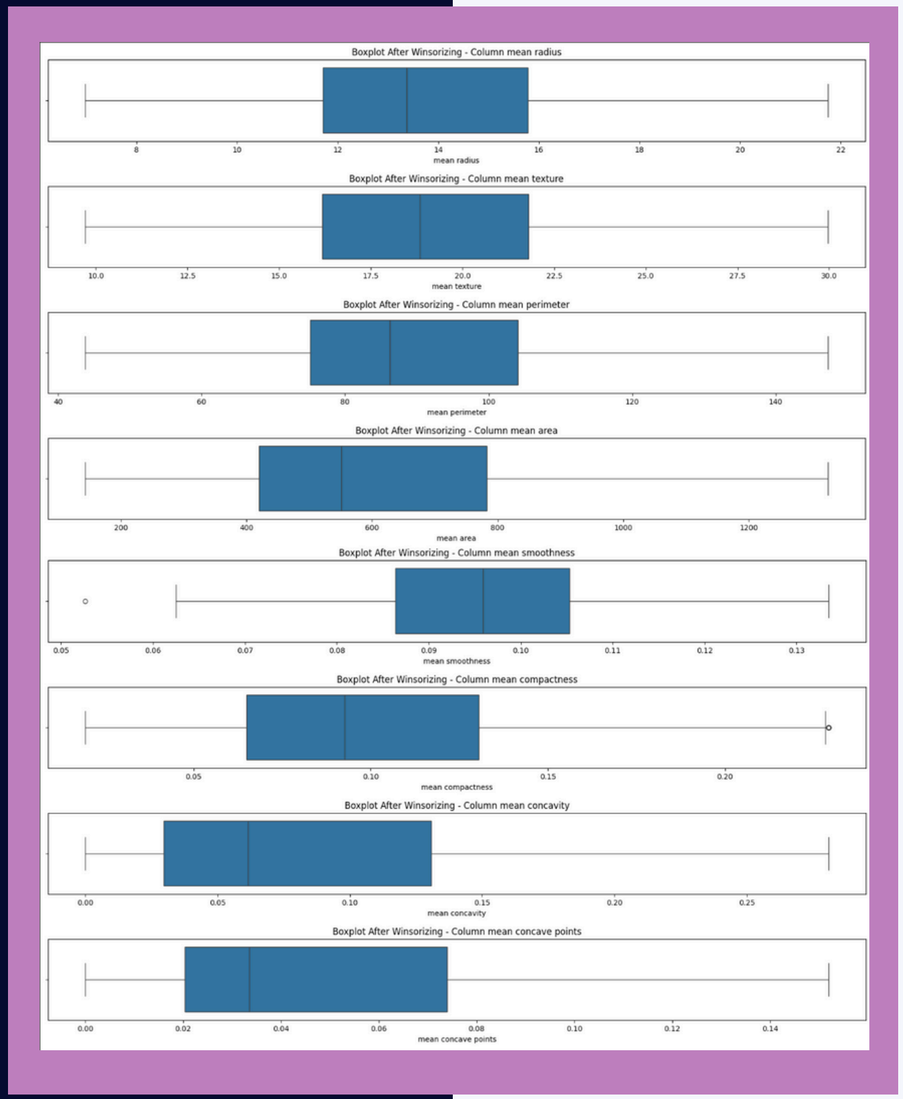
```
#Memeriksa outlier dengan boxplot
import matplotlib.pyplot as plt
import seaborn as sns
for kol in df_x:
  plt.figure(figsize = (20,2))
  sns.boxplot(df, x=kol)
  plt.title(f"Boxplot Column {kol}")
  plt.show()
```



DATA PREPROCESSING

4 WINSORIZED (OUTLIER HANDLING)

Winsorization is applied to handle outliers by replacing extreme values with the nearest value within a specified range, ensuring data consistency without removing significant points.



DATA PREPROCESSING

5 STANDARIZATION

Standardization is performed to scale the features to a uniform range, typically with a mean of 0 and a standard deviation of 1. This process ensures that all features are comparable and improves the performance of the machine learning model.

```
#Standarisasi Data
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
df = pd.DataFrame(scaler.fit_transform(df), columns=df.columns)
```

EXPLORATORY DATA ANALYSIS

1 DESCRIPTIVE STATISTICS

```
#Menampilkan statistik deskriptif data
df.describe()
```

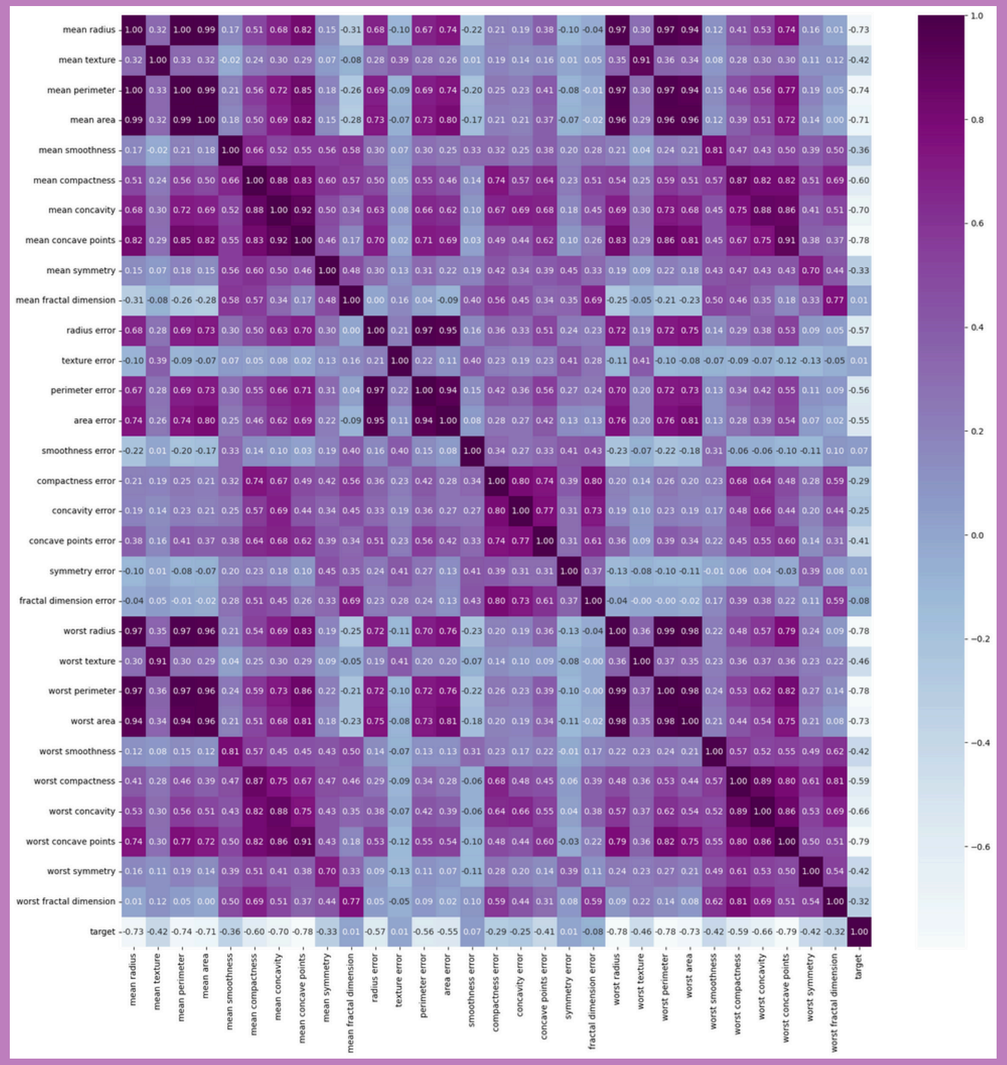
	mean radius	mean texture	mean perimeter	m
count	5.690000e+02	5.690000e+02	5.690000e+02	5.690
mean	-2.622390e-16	3.277987e-16	2.497514e-17	-2.49
std	1.000880e+00	1.000880e+00	1.000880e+00	1.000
min	-2.126557e+00	-2.285331e+00	-2.074601e+00	-1.626
25%	-7.087972e-01	-7.380411e-01	-7.112108e-01	-7.19
50%	-2.070683e-01	-9.852672e-02	-2.302441e-01	-2.90
75%	5.169836e-01	6.104481e-01	5.457329e-01	4.68
max	2.310589e+00	2.567314e+00	2.422676e+00	2.249

8 rows × 31 columns

Provides a comprehensive overview of patterns, distributions, and key characteristics of the dataset through summary measures like mean, median, variance, standard deviation, skewness, and more.

EXPLORATORY DATA ANALYSIS

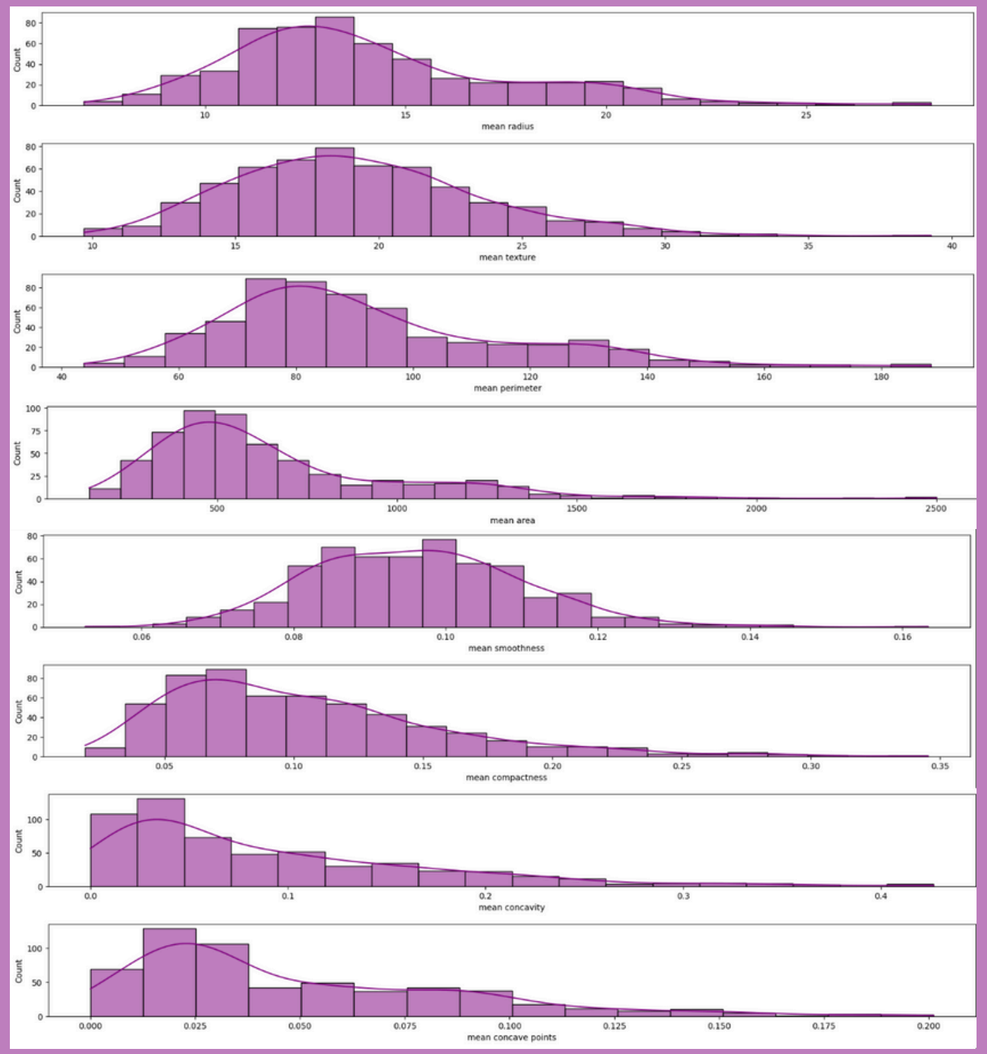
2 CORRELATION MATRIX



Examines relationships between features in the data to identify significant patterns and reduce dimensionality for better interpretation. In the accompanying correlation matrix, darker shades of purple indicate stronger positive correlations between variables, while lighter shades approaching white signify stronger negative correlations.

EXPLORATORY DATA ANALYSIS

3 DISTRIBUTION DATA



Reveals how data is distributed and its range, making it easier to select the most suitable model for analysis.

MODELLING - SPLIT DATA

```
#Split data
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(df_x, df_y, test_size=0.2, random_state=42)
x_train.shape, x_test.shape, y_train.shape, y_test.shape
```

```
((455, 30), (114, 30), (455,), (114,))
```

The processed data will be divided into two sets (training and testing data). The data will be split with 80% allocated for training and 20% for testing.

MODELLING - DECISION TREE

```
from sklearn.tree import DecisionTreeClassifier
DT = DecisionTreeClassifier(random_state=42)
DT.fit(x_train, y_train)
```

```
DecisionTreeClassifier
DecisionTreeClassifier(random_state=42)
```

```
[16] ypred_DT = DT.predict(x_test)
     ypred_DT
```

```
array([1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1,
       0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1,
       1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1,
       0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0,
       1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1,
       0, 1, 1, 0])
```

The Decision Tree model is a straightforward and widely-used classification model. It does not require data scaling and is known for its simplicity. For consistency, the model is set to use a random state value of 42, ensuring the same subset of data is used in each run.

MODELLING - RANDOM FOREST

```
from sklearn.ensemble import RandomForestClassifier
RF = RandomForestClassifier(random_state=42)
RF.fit(x_train, y_train)
```

```
RandomForestClassifier
RandomForestClassifier(random_state=42)
```

```
[18] ypred_RF = RF.predict(x_test)
     ypred_RF
```

```
array([1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1,
       0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1,
       1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1,
       0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0,
       1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1,
       0, 1, 1, 0])
```

Random Forest is an ensemble learning algorithm designed for both classification and regression tasks. It enhances the accuracy of modeling by combining multiple decision trees, making it a powerful tool for predictions.

MODELLING - LOGISTIC REGRESSION

```
from sklearn.linear_model import LogisticRegression
LR = LogisticRegression(random_state=42, max_iter=500)
LR.fit(x_train, y_train)
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/linear_model/_logistic.py:465: Converge
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

LogisticRegression

LogisticRegression(max_iter=500, random_state=42)

```
[20] ypred_LR = LR.predict(x_test)
     ypred_LR
```

```
array([1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1,
       0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1,
       1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1,
       0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0,
       1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1,
       0, 1, 0, 0])
```

Logistic Regression is a classification model used to predict the likelihood of an event's occurrence. It requires standardized data for effective model training.

MODELLING - NAIVE BAYES

```
from sklearn.naive_bayes import GaussianNB
NB = GaussianNB()
NB.fit(x_train, y_train)
```

```
GaussianNB
GaussianNB()
```

```
[22] ypred_NB = NB.predict(x_test)
     ypred_NB
```

```
array([1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1,
       0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1,
       1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1,
       0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0,
       1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1,
       0, 1, 1, 0])
```

Naive Bayes is a classification algorithm based on Bayes' Theorem, often applied in cases with Gaussian distribution. It calculates the probability of a class based on the distribution of the features in the data.

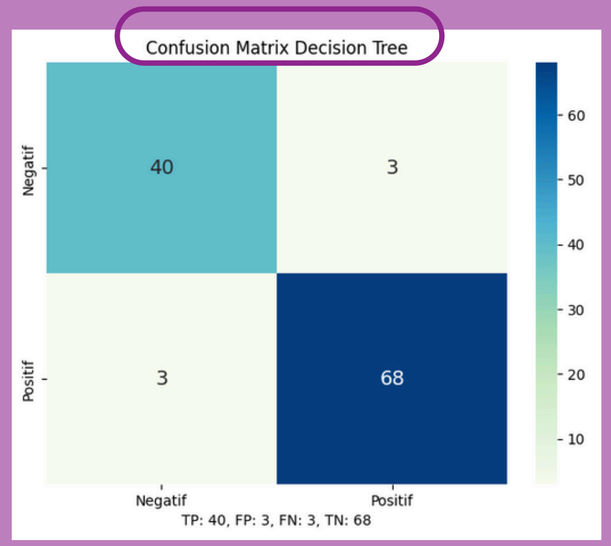
```python
#Akurasi model
from sklearn.metrics import accuracy_score
accuracyDT = accuracy_score(y_test, ypred_DT)
accuracyRF = accuracy_score(y_test, ypred_RF)
accuracyLR = accuracy_score(y_test, ypred_LR)
accuracyNB = accuracy_score(y_test, ypred_NB)

print(f"Akurasi Model Decision Tree: {accuracyDT * 100:.2f}%")
print(f"Akurasi Model Random Forest: {accuracyRF * 100:.2f}%")
print(f"Akurasi Model Logistic Regression: {accuracyLR * 100:.2f}%")
print(f"Akurasi Model Naive Bayes: {accuracyNB * 100:.2f}%")
```

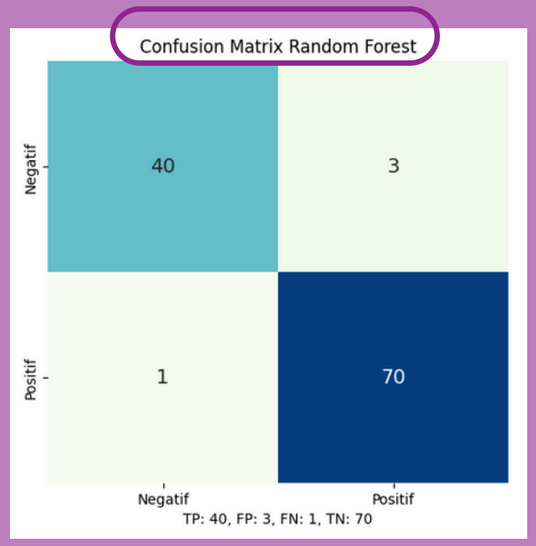
```
Akurasi Model Decision Tree: 94.74%
Akurasi Model Random Forest: 96.49%
Akurasi Model Logistic Regression: 95.61%
Akurasi Model Naive Bayes: 97.37%
```

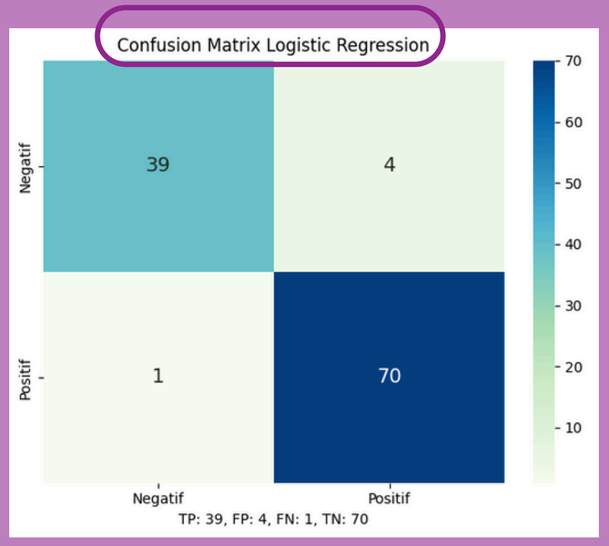
The classification algorithm's accuracy results:

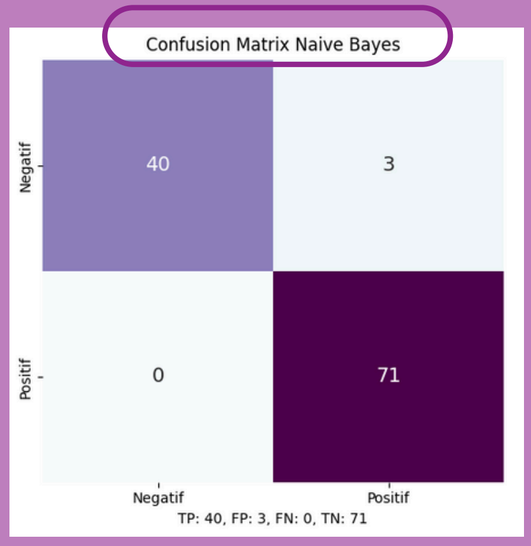
Decision Tree	: 94.74%
Random Forest	: 96.49%
Logistic Regression	: 95.61%
Naive Bayes	: 97.37%

EVALUATION - ACCURACY SCORE









All the classification algorithms show that their confusion matrices have higher TP and TN than FP and FN, which suggests that **all the classification algorithms used are considered effective**.

EVALUATION - CONFUSION MATRIX

CONCLUSION

This study evaluates four classification models: Decision Tree, Random Forest, Logistic Regression, and Naive Bayes. The accuracy rates for these models are as follows: Decision Tree at 94.74%, Random Forest at 96.49%, Logistic Regression at 95.61%, and Naive Bayes at 97.37%.

Based on these results, it can be concluded that **Naive Bayes is the most effective model for classifying breast cancer datasets**, as it achieved the highest accuracy rate among all the models tested.

GET IN TOUCH

Github
github.com/ketrinnatasya21

LinkedIn
linkedin.com/in/ketrinntsya

Email
ketrinnatasya21@gmail.com

I am always open to new opportunities for collaboration.
Let’s connect and explore how we can work together to
achieve meaningful outcomes!