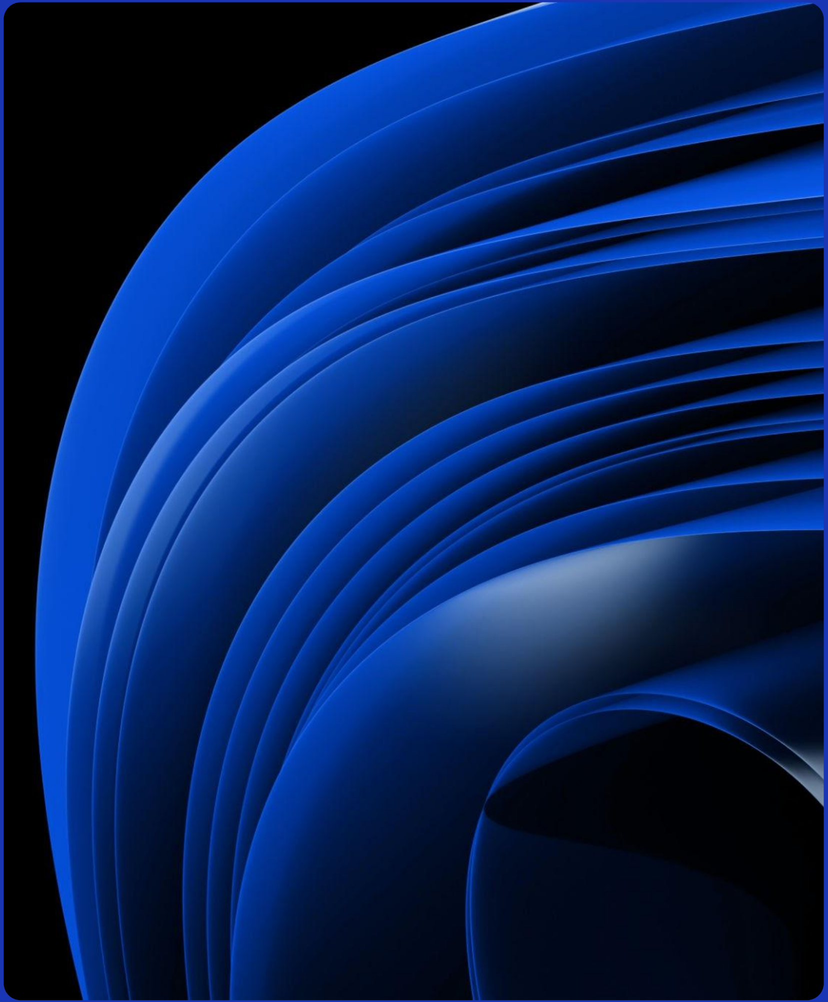
Breast Cancer Classification

By Ardiansyah Ihsan Prakoso



Introduction

Breast cancer is one of the most common types of cancer and a major concern in the medical world. Early detection of breast cancer is essential to increase the chances of cure and the effectiveness of treatment. One method that can be used in breast cancer analysis is machine learning, which allows classification of tumors based on medical characteristics.

In this project, we will use the Breast Cancer Wisconsin (Diagnostic) Dataset from Scikit-Learn to build a breast cancer classification model. The model aims to distinguish between benign and malignant tumors based on various cellular characteristics obtained from digital images of fine needle aspiration (FNA) biopsies.

The project involved data exploration, dataset processing, application of machine learning algorithms, and model evaluation to determine the most effective classification method.

Goals Project

Develop a machine learning model to classify tumors as benign or malignant based on the features available in the dataset.

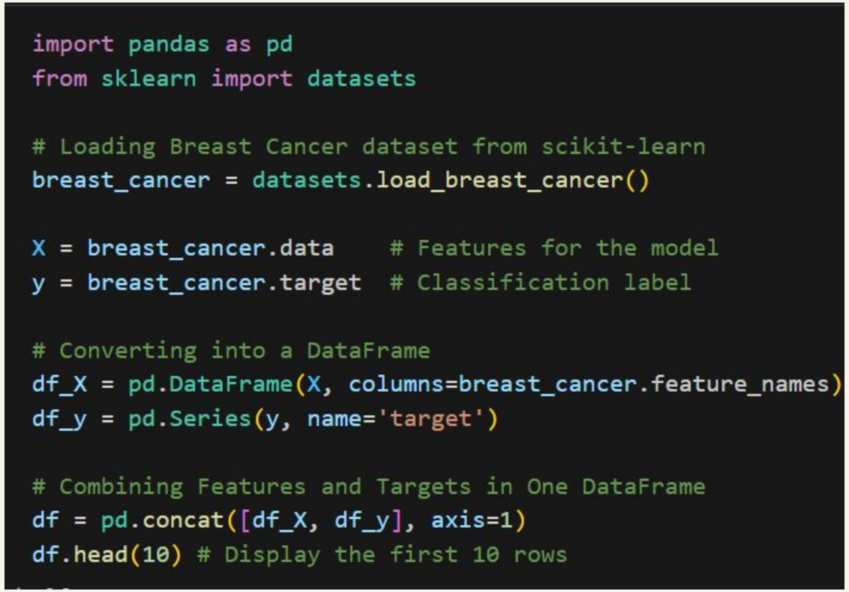
Analyzed tumor cell characteristics based on 30 numerical features to understand the key differences between benign and malignant tumors.

Testing the performance of two classification algorithms, namely Naïve Bayes and Support Vector Machine (SVM), in detecting breast cancer.

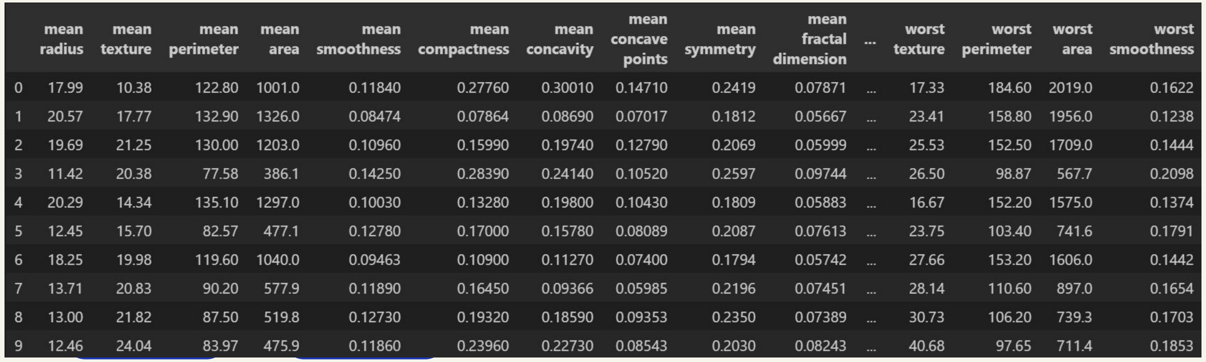
Evaluate the model performance using metrics such as accuracy, precision, recall, f1-score, and confusion matrix to ensure the developed model has high reliability in tumor classification.

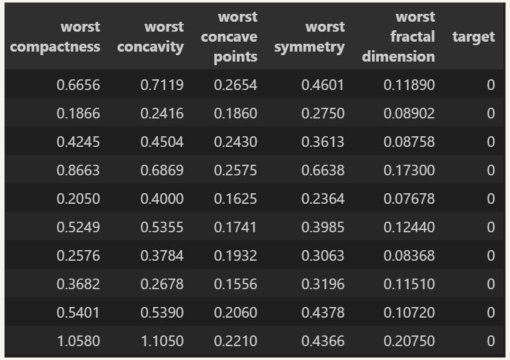
Dataset

The Breast Cancer dataset is a dataset available in the Scikit-Learn library and is often used in machine learning research for breast cancer classification. This dataset contains information on various tumor cell characteristics obtained from digital images of fine needle aspiration (FNA) of breast tissue.



```python
import pandas as pd
from sklearn import datasets

# Loading Breast Cancer dataset from scikit-learn
breast_cancer = datasets.load_breast_cancer()

X = breast_cancer.data    # Features for the model
y = breast_cancer.target  # Classification label

# Converting into a DataFrame
df_X = pd.DataFrame(X, columns=breast_cancer.feature_names)
df_y = pd.Series(y, name='target')

# Combining Features and Targets in One DataFrame
df = pd.concat([df_X, df_y], axis=1)
df.head(10) # Display the first 10 rows
```

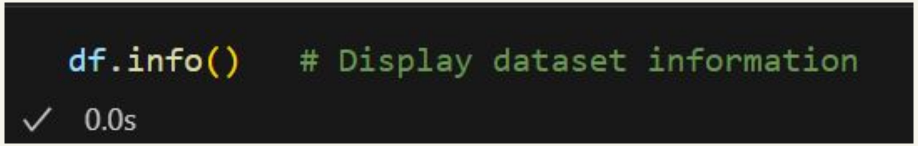




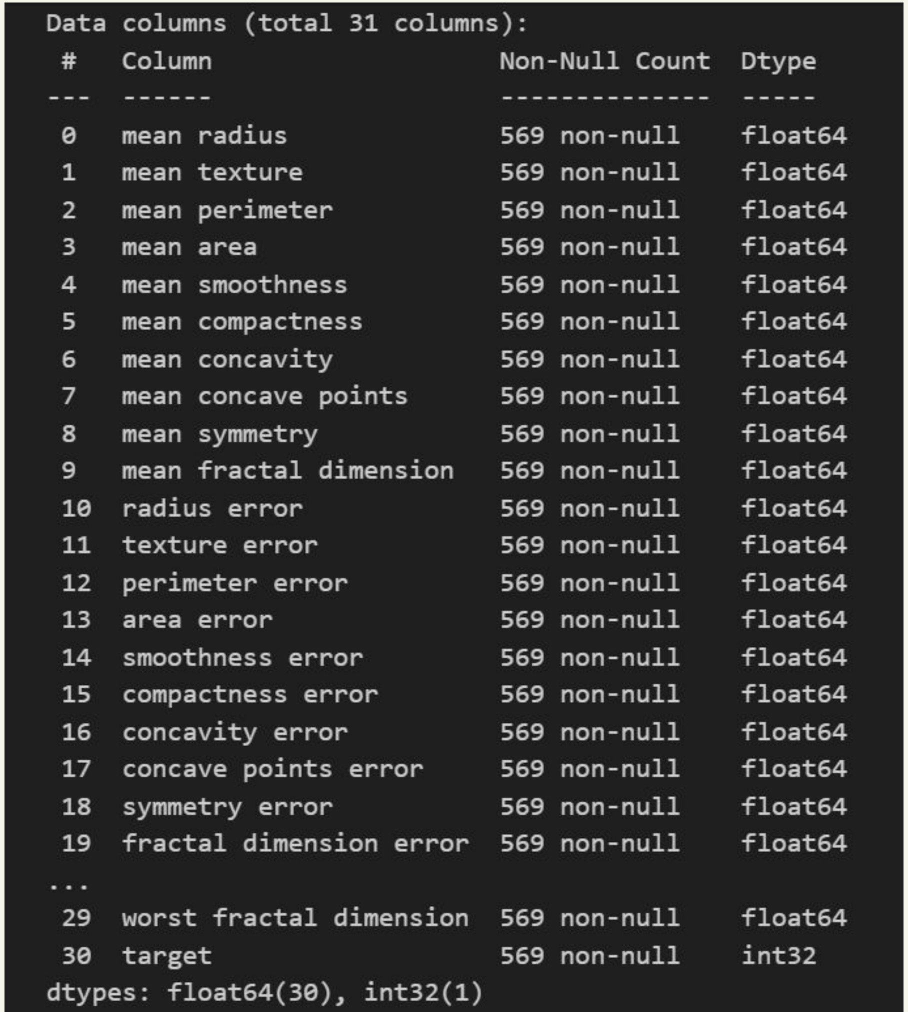
	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst texture	worst perimeter	worst area	worst smoothness	worst compactness	worst concavity	worst concave points	worst symmetry	worst fractal dimension	target
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710	0.2419	0.07871	...	17.33	184.60	2019.0	0.1622	0.6656	0.7119	0.2654	0.4601	0.11890	0
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017	0.1812	0.05667	...	23.41	158.80	1956.0	0.1238	0.1866	0.2416	0.1860	0.2750	0.08902	0
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790	0.2069	0.05999	...	25.53	152.50	1709.0	0.1444	0.4245	0.4504	0.2430	0.3613	0.08758	0
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520	0.2597	0.09744	...	26.50	98.87	567.7	0.2098	0.8663	0.6869	0.2575	0.6638	0.17300	0
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430	0.1809	0.05883	...	16.67	152.20	1575.0	0.1374	0.2050	0.4000	0.1625	0.2364	0.07678	0
5	12.45	15.70	82.57	477.1	0.12780	0.17000	0.15780	0.08089	0.2087	0.07613	...	23.75	103.40	741.6	0.1791	0.5249	0.5355	0.1741	0.3985	0.12440	0
6	18.25	19.98	119.60	1040.0	0.09463	0.10900	0.11270	0.07400	0.1794	0.05742	...	27.66	153.20	1606.0	0.1442	0.2576	0.3784	0.1932	0.3063	0.08368	0
7	13.71	20.83	90.20	577.9	0.11890	0.16450	0.09366	0.05985	0.2196	0.07451	...	28.14	110.60	897.0	0.1654	0.3682	0.2678	0.1556	0.3196	0.11510	0
8	13.00	21.82	87.50	519.8	0.12730	0.19320	0.18590	0.09353	0.2350	0.07389	...	30.73	106.20	739.3	0.1703	0.5401	0.5390	0.2060	0.4378	0.10720	0
9	12.46	24.04	83.97	475.9	0.11860	0.23960	0.22730	0.08543	0.2030	0.08243	...	40.68	97.65	711.4	0.1853	1.0580	1.1050	0.2210	0.4366	0.20750	0

Data Preprocessing

The dataset has been checked and confirmed to have no missing data or duplicate values, so it can be directly used in machine learning modeling without further cleaning.



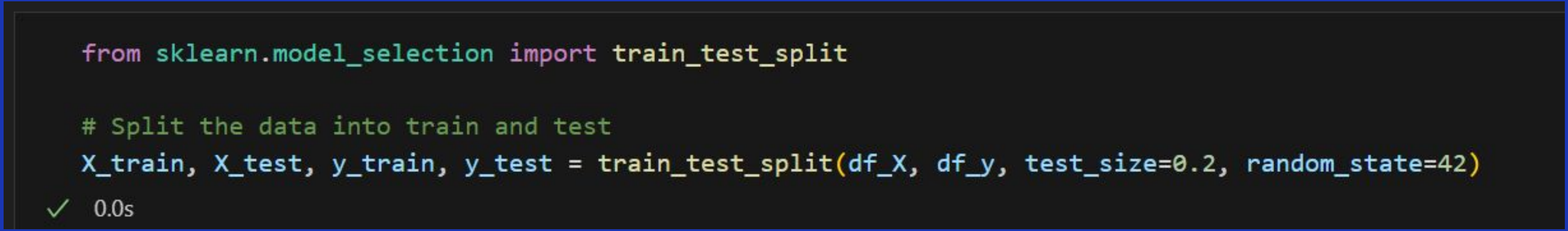
```
df.info()   # Display dataset information
✓ 0.0s
```



```
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
...
 29  worst fractal dimension  569 non-null    float64
 30  target                   569 non-null    int32
dtypes: float64(30), int32(1)
```

Modelling

The dataset is divided into two main parts, the training set and the testing set. This division is done to ensure that the model can learn from a portion of the data and then be tested on data that has never been seen before. In this project, the dataset was divided into 80% training set and 20% testing set using Scikit-Learn's train-test split method. The training data is used to build and adjust the model parameters, while the test data is used to evaluate the performance of the model in performing breast cancer classification. This split was done randomly by setting the random_state parameter = 42 so that the experimental results can be replicated consistently.



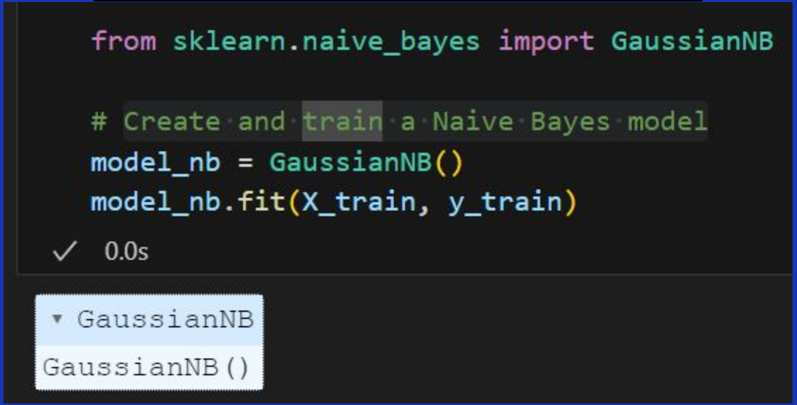
```
from sklearn.model_selection import train_test_split

# Split the data into train and test
X_train, X_test, y_train, y_test = train_test_split(df_X, df_y, test_size=0.2, random_state=42)
```

✓ 0.0s

Modelling

In this project, the machine learning used to detect breast cancer are Naïve Bayes and Support Vector Machine (SVM). The Naïve Bayes model works by calculating the likelihood of a tumor being benign or malignant based on patterns in the data. Meanwhile, SVM tries to find the best dividing line between benign and malignant tumors in order to make more accurate predictions. Both models are trained using cleaned data and then tested to see which one gives the best results in detecting breast cancer.

Naive Bayes

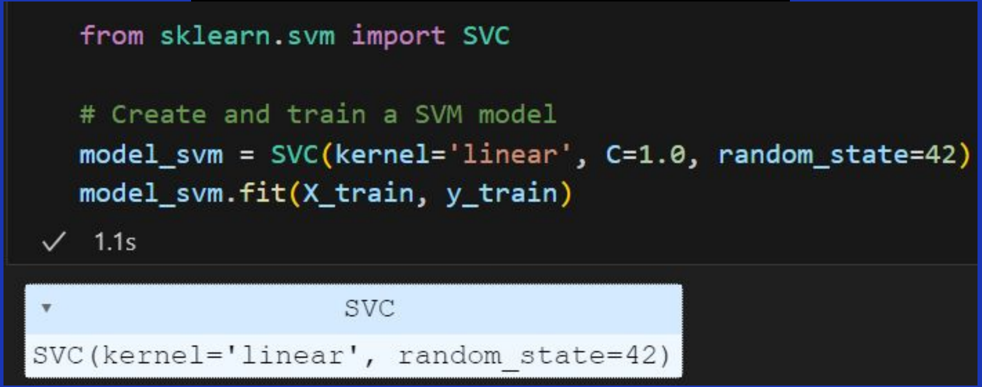


```
from sklearn.naive_bayes import GaussianNB

# Create and train a Naive Bayes model
model_nb = GaussianNB()
model_nb.fit(X_train, y_train)
```

✓ 0.0s

```
▾ GaussianNB
GaussianNB()
```

SVM



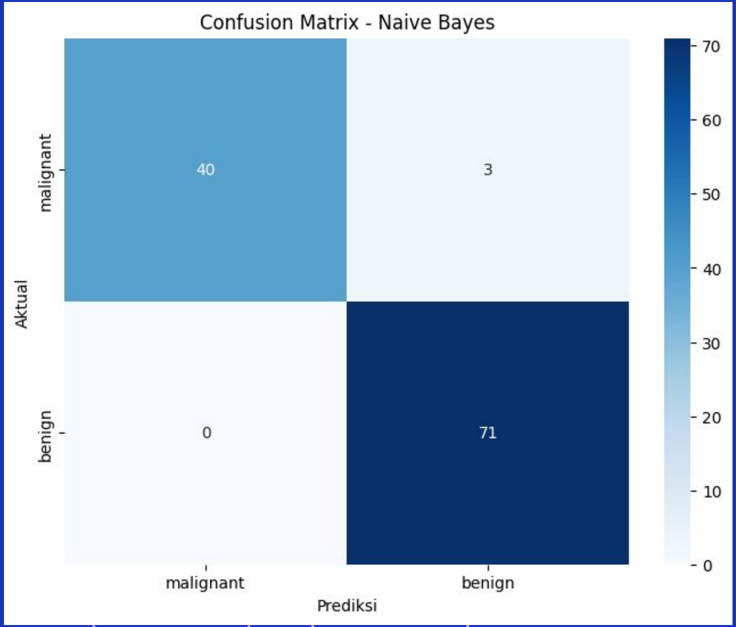
```
from sklearn.svm import SVC

# Create and train a SVM model
model_svm = SVC(kernel='linear', C=1.0, random_state=42)
model_svm.fit(X_train, y_train)
```

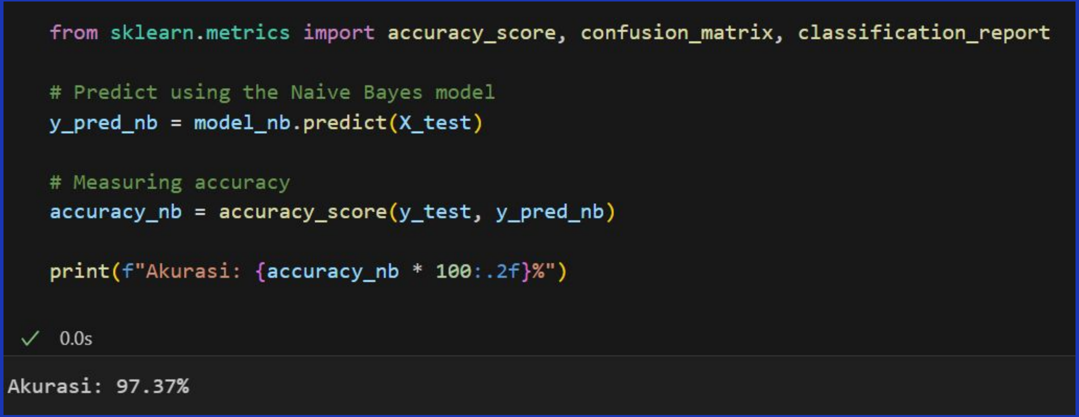
✓ 1.1s

```
▾ SVC
SVC(kernel='linear', random_state=42)
```

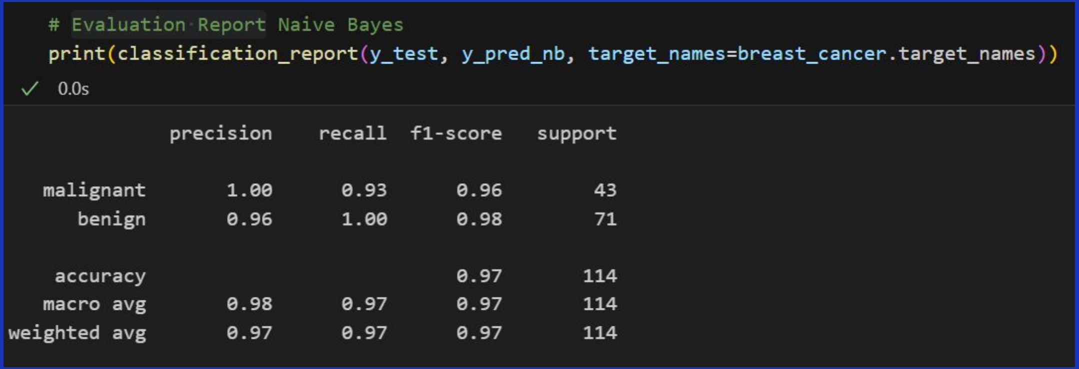
Naive Bayes Evaluation

The results of the Naive Bayes Model that has been trained and tested resulted in an accuracy of 97.37%, which means that this model is able to classify tumors with a very high success rate.





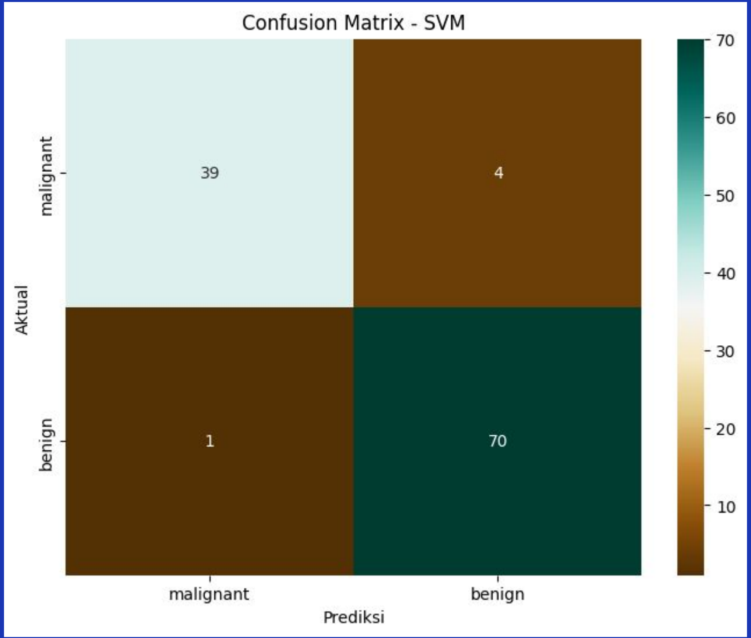
```
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

# Predict using the Naive Bayes model
y_pred_nb = model_nb.predict(X_test)

# Measuring accuracy
accuracy_nb = accuracy_score(y_test, y_pred_nb)

print(f"Akurasi: {accuracy_nb * 100:.2f}%")

✓ 0.0s

Akurasi: 97.37%
```

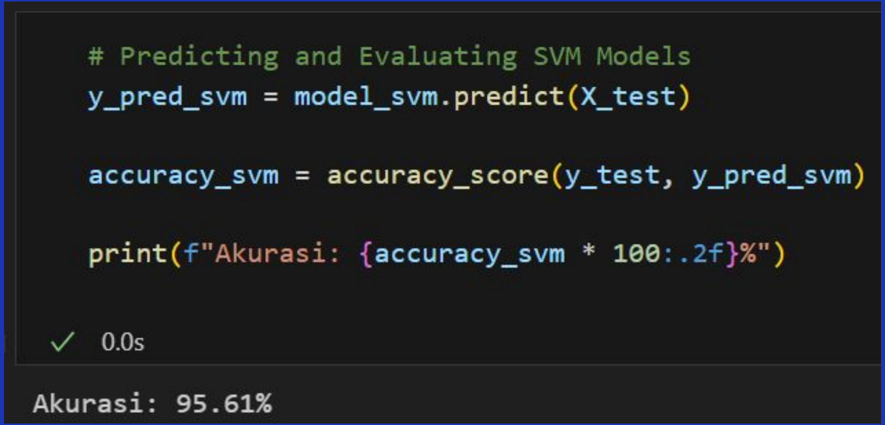


```
# Evaluation Report Naive Bayes
print(classification_report(y_test, y_pred_nb, target_names=breast_cancer.target_names))
✓ 0.0s

              precision    recall  f1-score   support

   malignant       1.00      0.93      0.96        43
      benign       0.96      1.00      0.98        71

    accuracy                           0.97       114
   macro avg       0.98      0.97      0.97       114
weighted avg       0.97      0.97      0.97       114
```

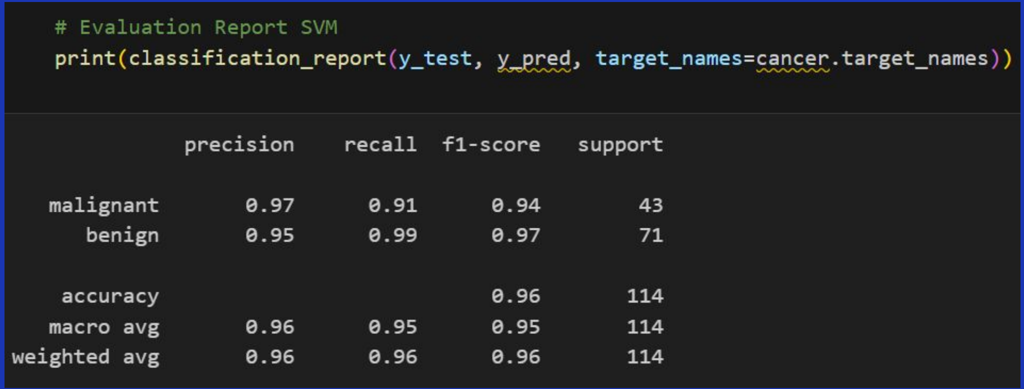
SVM Evaluation

The Support Vector Machine (SVM) model that has been trained and tested produces an accuracy of 95.61%, which means that this model is able to classify tumors quite well.





```
# Predicting and Evaluating SVM Models
y_pred_svm = model_svm.predict(X_test)

accuracy_svm = accuracy_score(y_test, y_pred_svm)

print(f"Akurasi: {accuracy_svm * 100:.2f}%")

✓ 0.0s

Akurasi: 95.61%
```



```
# Evaluation Report SVM
print(classification_report(y_test, y_pred, target_names=cancer.target_names))

              precision    recall  f1-score   support

   malignant       0.97      0.91      0.94        43
      benign       0.95      0.99      0.97        71

    accuracy                           0.96       114
   macro avg       0.96      0.95      0.95       114
weighted avg       0.96      0.96      0.96       114
```

Conclusion

Based on the evaluation results of Naïve Bayes and Support Vector Machine (SVM) models on breast cancer datasets, it can be concluded that Naïve Bayes has better performance than SVM with 97.37% accuracy, while SVM has 95.61% accuracy. The Naïve Bayes model excels in detecting malignant tumors with 100% precision, which means there are no errors in classifying malignant tumors as benign. However, the recall for malignant tumors is only 93%, which indicates that there are still some cases of malignant tumors that are misclassified as benign. Meanwhile, SVM has a precision of 97% and recall of 91% for malignant tumors, which means that this model is also quite good, but has slightly more errors in detecting malignant tumors than Naïve Bayes.

In terms of benign tumor detection, Naïve Bayes has 100% recall, which means all benign tumors were correctly classified, while SVM has 99% recall, which shows there is still a slight possibility of benign tumors being misclassified as malignant. From the balance of precision and recall measured by F1-score, Naïve Bayes obtained a higher value (0.96-0.98) than SVM (0.94-0.97), indicating that Naïve Bayes is more stable in detecting both tumor types.

Overall, Naïve Bayes is more recommended for this dataset as it has higher accuracy and is better at avoiding misclassification of malignant tumors as benign. However, if you want to use a model that is more flexible and less dependent on data distribution assumptions, SVM remains a good alternative. This model can be further improved through parameter optimization or data balancing techniques if needed.

Thank You!

Contact

Github	https://github.com/ArdiansyahIhsan
Email	ihsanprakoso25@gmail.com

Feel free to contact me if interested in this project or want to discuss further!