ibimbing

DSF 36.0 - *Data Science*

BREAST CANCER CLASSIFICATION ANALYSIS USING K-NEAREST NEIGHBORS (KNN)

This project aims to analyze the classification of malignant and benign breast cancer tumors using a Machine Learning algorithm, namely K-Nearest Neighbors (KNN).

by Nisrina Asyifa Nur Azizah

TOOLS AND LIBRARIES

pandas

NumPy

matplotlib

seaborn

scikit learn

OUTLINE OF DATA ANALYSIS

01 Load Data

02 Preprocessing Data

03 Machine Learning Model

04 Model Performance Evaluation

05 Data Visualization

06 Conclusion

LOAD DATA

- Breast Cancer Wisconsin (Diagnostic) dataset from the scikit-learn library.
- It has 30 numerical features related to tumor characteristics.
- Target (Class Label):
    - 0: Malignant
    - 1: Benign
- A total of 569 samples, consisting of 357 benign and 212 malignant.

```
<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
...
 29  worst fractal dimension  569 non-null    float64
 30  target                   569 non-null    int64
dtypes: float64(30), int64(1)
memory usage: 137.9 KB
```

```
target
1    357
0    212
Name: count, dtype: int64
```

	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	0.3001	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	0.0869	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	0.1974	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	0.2414	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	0.1980	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 rows × 31 columns

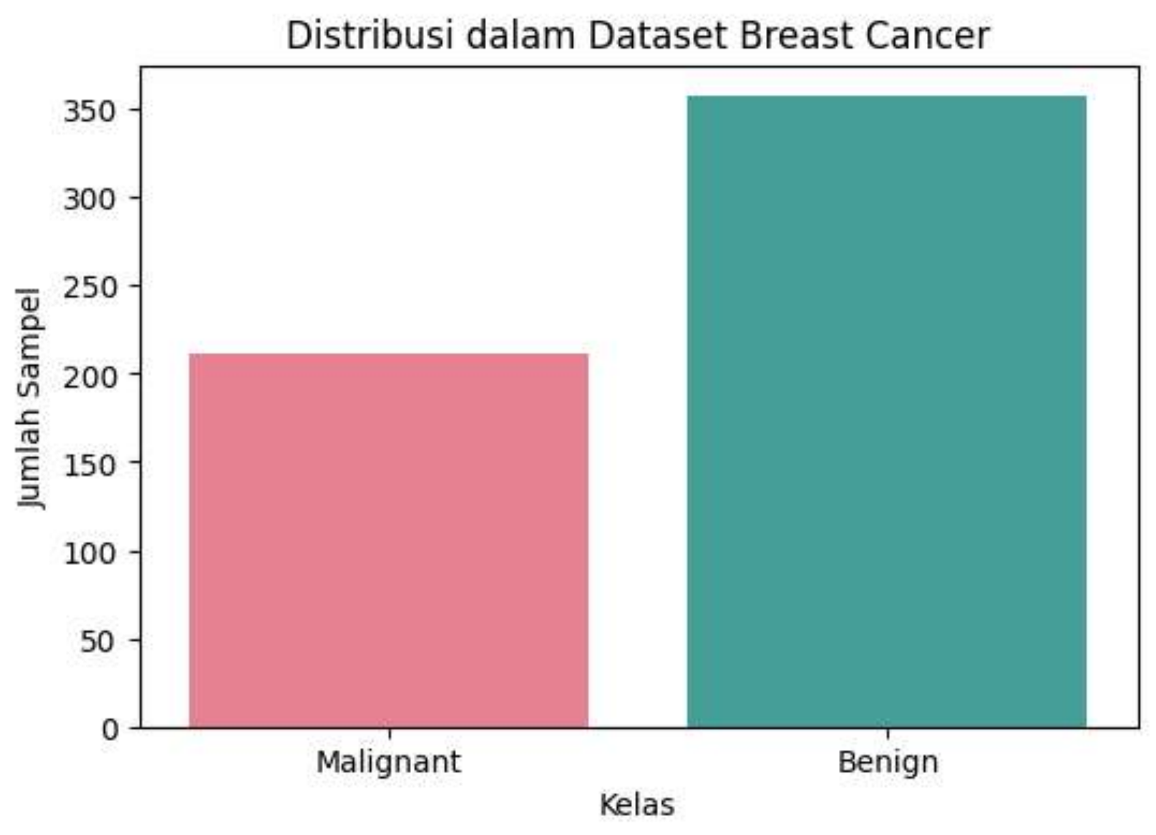
PREPROCESSING DATA

```
Missing Values:
 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 fractal dimension    0
target                     0
dtype: int64
Jumlah duplikat: 0
```

```python
# Normalisasi fitur untuk meningkatkan performa KNN
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

```python
# 3. Membagi dataset menjadi training dan testing (80:20)
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y,
                                                    test_size=0.2, random_state=42, stratify=y)
```

1. Checking for missing values and data duplication.
2. Feature normalization using StandardScaler
3. Divide the dataset into training data (80%) and testing data (20%)
4. Visualisasi distribusi data



MACHINE LEARNING MODEL

- The K-Nearest Neighbors (KNN) model was chosen for its simplicity and effectiveness in classification.
- Determining the optimal number of neighbors with GridSearchCV.
- Main parameters: k = 7

```python
# 5. Processing Data - Hyperparameter Tuning
param_grid_knn = {'n_neighbors': range(1, 20)}
grid_knn = GridSearchCV(KNeighborsClassifier(), param_grid_knn, cv=5)
grid_knn.fit(X_train, y_train)
print("Best parameters for KNN:", grid_knn.best_params_)
```

✓ 0.6s

```
Best parameters for KNN: {'n_neighbors': 7}
```

```python
# 6. Training model dengan KNN
knn_model = KNeighborsClassifier(n_neighbors=grid_knn.best_params_['n_neighbors'])
knn_model.fit(X_train, y_train)
```

✓ 0.0s

```
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=7)
```

```python
# Prediksi pada data uji
y_pred_knn = knn_model.predict(X_test)
```

✓ 0.0s

MODEL PERFORMANCE EVALUATION

```
Classification Report (KNN):
              precision    recall  f1-score   support

   Malignant       0.97      0.93      0.95        42
      Benign       0.96      0.99      0.97        72

    accuracy                           0.96       114
   macro avg       0.97      0.96      0.96       114
weighted avg       0.97      0.96      0.96       114

Akurasi Model KNN: 0.96
```

- **Classification report** shows evaluation metrics (Accuracy, Precision, Recall, F1-Score)
- **Model accuracy,** i.e. the percentage of correct predictions compared to the total test data

1. **Malignant**
   - Precision = 0.97 → 97% of predicted Malignant cases are correct.
   - Recall = 0.93 → 93% of actual Malignant cases are correctly identified.
   - F1-score = 0.95 → 95% Balanced measure of precision & recall.
2. **Benign**
   - Precision = 0.96 → 96% of predicted Benign cases are correct.
   - Recall = 0.99 → 99% of actual Benign cases are correctly identified.
   - F1-score = 0.97 → 97% Excellent classification performance.

Overall Model Performance

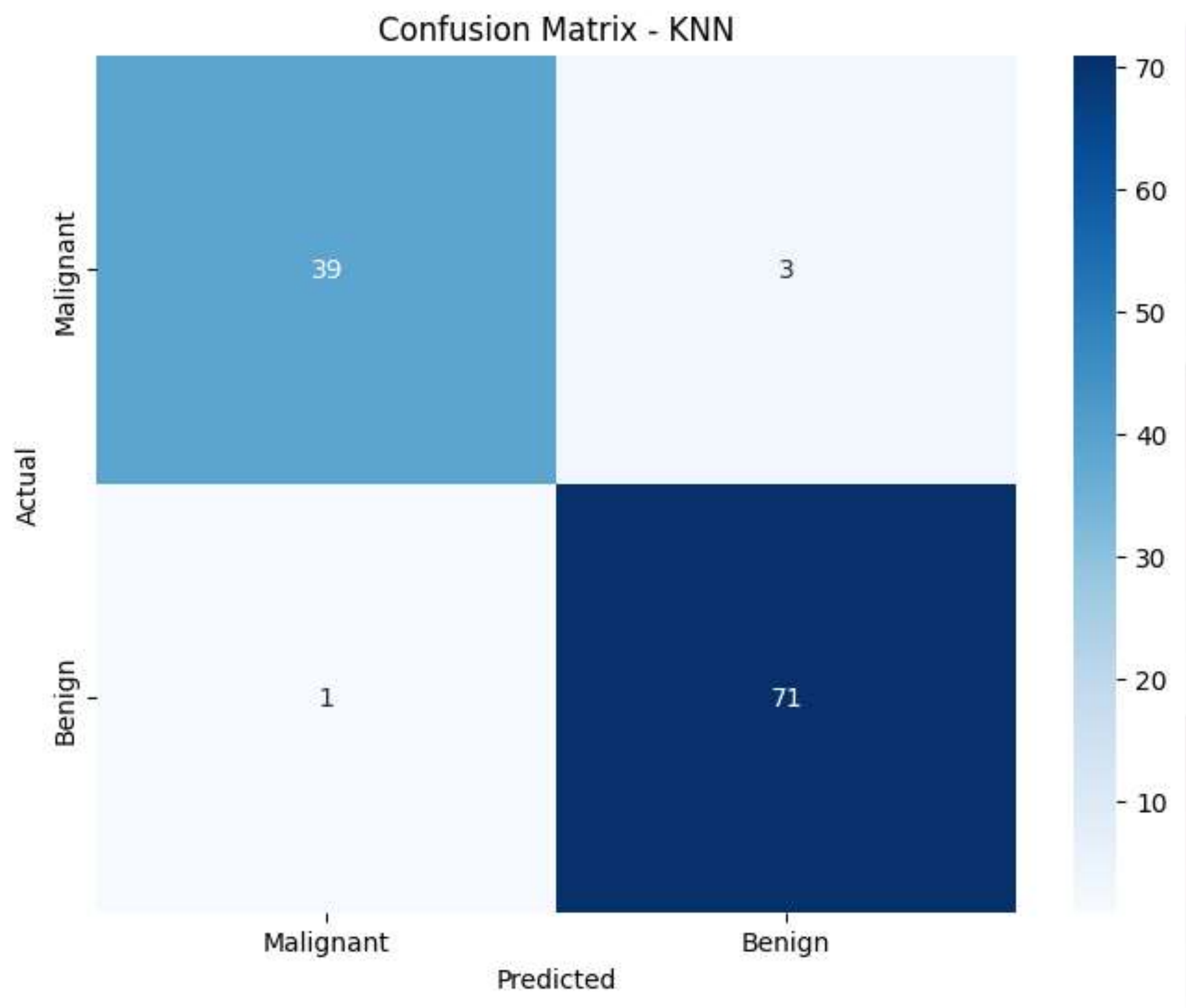
- Accuracy = 96%, meaning the model correctly classifies 96% of all cases.
- Macro & Weighted Averages confirm balanced performance across both classes.
- Slightly lower recall for Malignant cases (0.93) means some malignant cases are misclassified.

DATA VISUALIZATION

- **Confusion matrix,** which shows the number of correct and incorrect predictions for each class

Interpretation of the Confusion Matrix

- True Positives (Malignant correctly classified): 39 samples
- False Negatives (Malignant misclassified as Benign): 3 samples
- True Negatives (Benign correctly classified): 71 samples
- False Positives (Benign misclassified as Malignant): 1 sample



CONCLUSION

High Accuracy

- The KNN model achieved 96% accuracy, indicating excellent performance in classifying cancer data.

Performance on Malignant and Benign Classes

- High precision & recall for both classes.
- The recall for Malignant (0.93) is lower than Benign (0.99), meaning some cancer cases were misclassified.

Confusion Matrix Insights

- 3 Malignant cases were misclassified as Benign, which could be risky in medical diagnosis.
- 1 Benign case was misclassified as Malignant, potentially causing unnecessary anxiety for the patient.

THANK YOU

More information nisrinaasyifa56@gmail.com