In [66]:

```python
# Import necessary libraries
import pandas as pd

# Load the dataset
data = pd.read_csv("Breast Cancer Wisconsin.csv")

# Display the first few rows
data.head()
```

Out[66]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothn
0	842302	M	17.99	10.38	122.80	1001.0	
1	842517	M	20.57	17.77	132.90	1326.0	
2	84300903	M	19.69	21.25	130.00	1203.0	
3	84348301	M	11.42	20.38	77.58	386.1	
4	84358402	M	20.29	14.34	135.10	1297.0	

5 rows × 32 columns

In [68]:

```python
# View dataset shape and column information
print("Shape of the dataset:", data.shape)
print("\nColumns and Data Types:")
print(data.dtypes)

# Check for missing values
print("\nMissing Values:")
print(data.isnull().sum())

# Check class distribution
print("\nTarget Distribution:")
print(data["diagnosis"].value_counts())
```

```
Shape of the dataset: (569, 32)

Columns and Data Types:
id                           int64
diagnosis                   object
radius_mean                float64
texture_mean               float64
perimeter_mean             float64
area_mean                  float64
smoothness_mean            float64
compactness_mean           float64
concavity_mean             float64
concave points_mean        float64
symmetry_mean              float64
fractal_dimension_mean     float64
radius_se                  float64
texture_se                 float64
perimeter_se               float64
area_se                    float64
smoothness_se              float64
compactness_se             float64
concavity_se               float64
concave points_se          float64
symmetry_se                float64
fractal_dimension_se       float64
radius_worst               float64
texture_worst              float64
perimeter_worst            float64
area_worst                 float64
smoothness_worst           float64
compactness_worst          float64
concavity_worst            float64
concave points_worst       float64
symmetry_worst             float64
fractal_dimension_worst    float64
dtype: object

Missing Values:
id                         0
diagnosis                  0
radius_mean                0
texture_mean               0
perimeter_mean             0
area_mean                  0
smoothness_mean            0
compactness_mean           0
concavity_mean             0
concave points_mean        0
symmetry_mean              0
fractal_dimension_mean     0
radius_se                  0
texture_se                 0
perimeter_se               0
area_se                    0
smoothness_se              0
compactness_se             0
```

```
concavity_se               0
concave points_se          0
symmetry_se                0
fractal_dimension_se       0
radius_worst               0
texture_worst              0
perimeter_worst            0
area_worst                 0
smoothness_worst           0
compactness_worst          0
concavity_worst            0
concave points_worst       0
symmetry_worst             0
fractal_dimension_worst    0
dtype: int64

Target Distribution:
diagnosis
B    357
M    212
Name: count, dtype: int64
```

In [70]:

```python
# Drop unnecessary columns
data.drop("id", axis=1, inplace=True)

# Encode the target variable: M → 1, B → 0
data["diagnosis"] = data["diagnosis"].map({"M": 1, "B": 0})
```

In [72]:

```python
from sklearn.model_selection import train_test_split
from imblearn.over_sampling import SMOTE
from collections import Counter

# Split features and target
X = data.drop("diagnosis", axis=1)
y = data["diagnosis"]

# Perform train-test split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, stratify=y, random_state=42
)

# Apply SMOTE to balance the training data
smote = SMOTE(random_state=42)
X_train_smote, y_train_smote = smote.fit_resample(X_train, y_train)

# Show class distribution before and after SMOTE
print("Before SMOTE:", Counter(y_train))
print("After SMOTE:", Counter(y_train_smote))
```

```
Before SMOTE: Counter({0: 250, 1: 148})
After SMOTE: Counter({1: 250, 0: 250})
```

```
C:\Users\PC\anaconda3\Lib\site-packages\sklearn\base.py:474: FutureWarning: `BaseEst
imator._validate_data` is deprecated in 1.6 and will be removed in 1.7. Use `sklear
n.utils.validation.validate_data` instead. This function becomes public and is part
of the scikit-learn developer API.
  warnings.warn(
```

In [74]:

```
from xgboost import XGBClassifier

# Initialize and train XGBoost model
xgb = XGBClassifier(use_label_encoder=False, eval_metric='logloss', random_state=42
xgb.fit(X_train_smote, y_train_smote)
```

```
C:\Users\PC\anaconda3\Lib\site-packages\xgboost\training.py:183: UserWarning: [00:4
5:28] WARNING: C:\actions-runner\_work\xgboost\xgboost\src\learner.cc:738:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
```

Out[74]:

```
▾ XGBClassifier

XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds
=None,
              enable_categorical=False, eval_metric='logloss',
              feature_types=None, feature_weights=None, gamma=None,
              grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin
=None,
```

In [76]:

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

# Predict on the test set
y_pred = xgb.predict(X_test)

# Evaluate predictions
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
print("Accuracy Score:", accuracy_score(y_test, y_pred))
```

```
Confusion Matrix:
 [[107   0]
 [  4  60]]

Classification Report:
               precision    recall  f1-score   support

           0       0.96      1.00      0.98       107
           1       1.00      0.94      0.97        64

    accuracy                           0.98       171
   macro avg       0.98      0.97      0.97       171
weighted avg       0.98      0.98      0.98       171

Accuracy Score: 0.9766081871345029
```

In [ ]:

In [ ]: