Predictive Analytics: Data Preprocessing

This notebook handles the preprocessing of the Breast Cancer dataset as part of the AI in Software Engineering group assignment. The goa

```
# Import necessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
```

📁 Step 1: Load Dataset

We upload the dataset (data.csv) into Google Colab and load it using pandas.

```
# Upload CSV file
from google.colab import files
uploaded = files.upload()
```

Choose files breast_cancer_data.csv

- **breast_cancer_data.csv**(text/csv) - 125204 bytes, last modified: 19/09/2019 - 100% done

```
# Load the dataset
df = pd.read_csv('breast_cancer_data.csv')
df.head()
```

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	poi
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	

5 rows × 33 columns

🧹 Step 2: Clean the Data

We check for missing values, drop duplicates, and remove irrelevant columns such as `id` and any unnamed columns.

```
# Check for missing values
print("Missing values per column:\n", df.isnull().sum())

# Check for duplicates
print("\nNumber of duplicate rows:", df.duplicated().sum())
df.drop_duplicates(inplace=True)

# Drop irrelevant or empty columns
if 'Unnamed: 32' in df.columns:
    df.drop('Unnamed: 32', axis=1, inplace=True)
if 'id' in df.columns:
    df.drop('id', axis=1, inplace=True)

# Dataset info after cleaning
df.info()
```

What can I help you build?

```
compactness_worst        0
compactness_worst          0
concavity_worst            0
concave points_worst       0
symmetry_worst             0
fractal_dimension_worst    0
Unnamed: 32              569
dtype: int64

Number of duplicate rows: 0
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 31 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   diagnosis                569 non-null    object 
 1   radius_mean              569 non-null    float64
 2   texture_mean             569 non-null    float64
 3   perimeter_mean           569 non-null    float64
 4   area_mean                569 non-null    float64
 5   smoothness_mean          569 non-null    float64
 6   compactness_mean         569 non-null    float64
 7   concavity_mean           569 non-null    float64
 8   concave points_mean      569 non-null    float64
 9   symmetry_mean            569 non-null    float64
 10  fractal_dimension_mean   569 non-null    float64
 11  radius_se                569 non-null    float64
 12  texture_se               569 non-null    float64
 13  perimeter_se             569 non-null    float64
 14  area_se                  569 non-null    float64
 15  smoothness_se            569 non-null    float64
 16  compactness_se           569 non-null    float64
 17  concavity_se             569 non-null    float64
 18  concave points_se        569 non-null    float64
 19  symmetry_se              569 non-null    float64
 20  fractal_dimension_se     569 non-null    float64
 21  radius_worst             569 non-null    float64
 22  texture_worst            569 non-null    float64
 23  perimeter_worst          569 non-null    float64
 24  area_worst               569 non-null    float64
 25  smoothness_worst         569 non-null    float64
 26  compactness_worst        569 non-null    float64
 27  concavity_worst          569 non-null    float64
 28  concave points_worst     569 non-null    float64
 29  symmetry_worst           569 non-null    float64
 30  fractal_dimension_worst  569 non-null    float64
dtypes: float64(30), object(1)
memory usage: 137.9+ KB
```

🔤 Step 3: Encode Categorical Values

The `diagnosis` column contains categorical labels: "M" for malignant and "B" for benign. We'll use `LabelEncoder` to convert them to num

```
# Encode the 'diagnosis' column
le = LabelEncoder()
df['diagnosis'] = le.fit_transform(df['diagnosis'])  # M = 1, B = 0

# Check encoded values
df['diagnosis'].value_counts()
```

	count
diagnosis	
0	357
1	212

dtype: int64

🧪 Step 4: Split Dataset into Train/Test Sets

We separate the features and the target variable, then split the dataset into 80% training and 20% testing sets.

```
# Define features and target
X = df.drop('diagnosis', axis=1)
y = df['diagnosis']

# Split the dataset
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42)

# Output shapes
```

```
print("X_train shape:", X_train.shape)
print("X_test shape:", X_test.shape)
print("y_train shape:", y_train.shape)
print("y_test shape:", y_test.shape)
```

```
X_train shape: (455, 30)
X_test shape: (114, 30)
y_train shape: (455,)
y_test shape: (114,)
```

```
## ✅ Summary

We successfully:
- Loaded and previewed the dataset.
- Cleaned missing values, duplicates, and irrelevant columns.
- Encoded the categorical target (`diagnosis`) to numeric.
- Split the dataset into training and testing sets for model development.

This preprocessed dataset is now ready for modeling in the next phase of the project.
```