| **Method** | **Use Case** | **When to Avoid** |
| --- | --- | --- |
| Mean | Quick fix for numerical data | If data is skewed |
| Median | Safer when outliers exist | Doesn’t adapt to context |
| Mode | Categorical variables | Not useful for numerical |
| **KNNImputer** | Best for both numeric and mixed data | Need enough complete rows |

**3. Class Imbalance — Explained**

In your heart disease dataset:

python

CopyEdit

print(df['TenYearCHD'].value\_counts())

You will most likely see:

yaml

CopyEdit

0 3200+

1 600+

This means:

* **Most people don’t have heart disease (0)**
* **Very few people have it (1)**

**❌ Problem:**

The model **learns to always predict 0 (No risk)** just to increase accuracy.

🔴 Even if someone has heart disease, the model may predict **"No"** — because it saw more 0s in training.

**✅ Fix: Use SMOTE (Synthetic Minority Oversampling Technique)**

SMOTE **generates fake but realistic "1" rows** (patients with disease) by learning from real ones.

This helps balance the dataset.

**✅ Step-by-Step Code (with explanation)**

python

CopyEdit

# Step 1: Import SMOTE

from imblearn.over\_sampling import SMOTE

# Step 2: Apply SMOTE on your scaled training data

sm = SMOTE(random\_state=42)

X\_train\_resampled, y\_train\_resampled = sm.fit\_resample(X\_train\_scaled, y\_train)

# Step 3: Train your model on this balanced dataset

from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier()

model.fit(X\_train\_resampled, y\_train\_resampled)

# Step 4: Test on real test set

y\_pred = model.predict(X\_test\_scaled)

# Step 5: Evaluate

from sklearn.metrics import classification\_report

print(classification\_report(y\_test, y\_pred))

**🧠 What This Code Does:**

| **Code** | **Explanation** |
| --- | --- |
| sm = SMOTE() | Initializes SMOTE to create synthetic examples |
| .fit\_resample() | Creates new "1" rows to match the number of "0"s |
| model.fit() | Trains your model on the balanced data |
| classification\_report() | Shows precision, recall, F1 score → these are better for imbalanced data |

**✅ How SMOTE Improves Your Model**

| **Without SMOTE** | **With SMOTE** |
| --- | --- |
| High accuracy, but low recall for 1s | Slightly lower accuracy but **much better at detecting 1s** |
| Biased toward 0 (No Risk) | Balanced — detects both No and Yes fairly |
| Bad model in real life | **Useful medical model** ✅ |

**🔎 Bonus: Visual Check Before & After SMOTE**

python

CopyEdit

import seaborn as sns

import matplotlib.pyplot as plt

# Before

sns.countplot(x=y\_train)

plt.title("Before SMOTE")

plt.show()

# After

sns.countplot(x=y\_train\_resampled)

plt.title("After SMOTE")

plt.show()

sample\_input = [[1, 62, 1, 1, 1, 280, 160, 100, 32.0, 160]]

* **male = 1** → Male
* **age = 62**
* **BPMeds = 1** → On BP meds
* **prevalentHyp = 1** → Has hypertension
* **diabetes = 1** → Diabetic
* **totChol = 280** → High cholesterol
* **sysBP = 160**, **diaBP = 100** → High blood pressure
* **BMI = 32.0** → Overweight
* **glucose = 160** → High glucose

This person has multiple risks → likely **high risk (1)**.