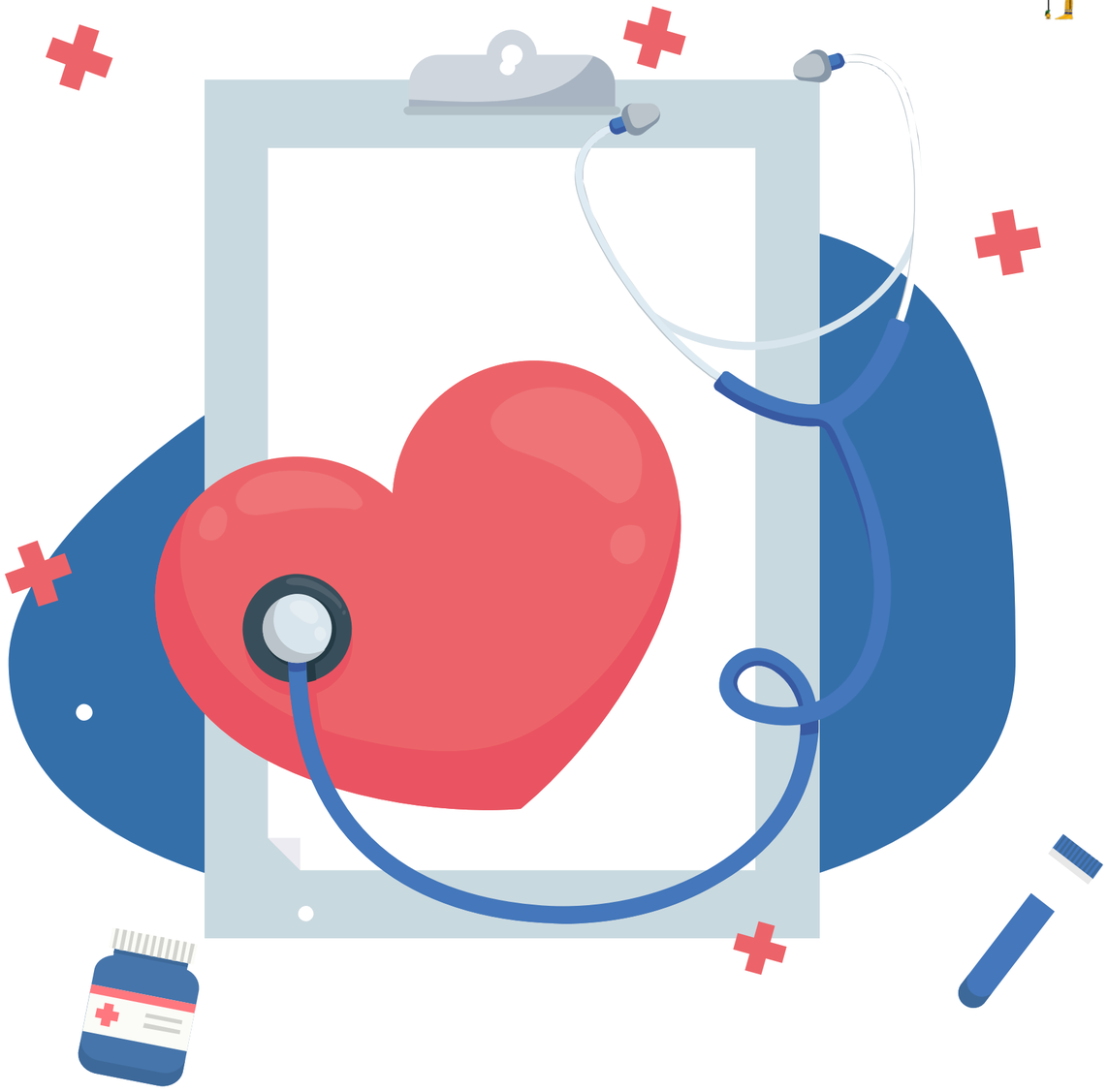
رواد مصر الرقمية

وزارة الاتصـــــالات
وتكنولوجيا المعلومات

Heart Disease Prediction



Supervised by Eng. Mahmoud Talaat

DS Healthcare Cairo
CAI3_AIS4_S2

eyouth®

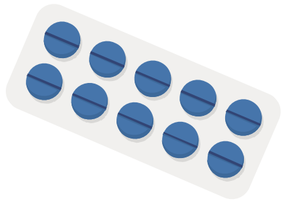
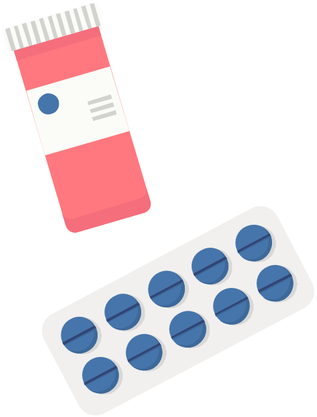


TABLE OF CONTENTS

01

INTRODUCTION

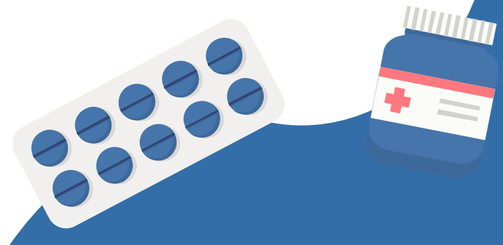


Heart disease is one of the leading causes of death worldwide. According to **the World Health Organization (WHO)**, cardiovascular diseases cause around **17.9** million deaths each year, representing 32% of all global deaths, with 85% of these deaths due to heart attacks and strokes.

The global burden is increasing, and by 2030, deaths from heart disease are expected to exceed 23 million if preventive measures are not strengthened.

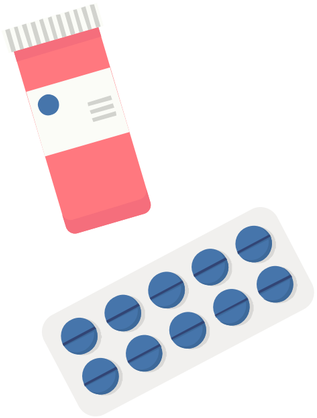
A major issue is that many cases remain undiagnosed in early stages, as symptoms often appear only after serious complications.

This project aims to build a machine learning model that **predicts** the risk of heart disease using personal health and lifestyle indicators. **The goal is** to support early detection, improve awareness, and contribute to better preventive healthcare decisions.



02

PROBLEM STATEMENT



Challenges in Diagnosing Heart Disease

- Many people, especially in rural or underserved areas, lack access to routine heart screenings, so early signs often go unnoticed.
- Heart disease often develops silently, with vague symptoms like fatigue or mild chest discomfort, making early detection difficult.
- Junior or non-specialist doctors may struggle to diagnose early cases, risking delays or misdiagnosis.
- In emergency situations, quick assessment is needed, but full diagnostic tools may not be available.
- Rising obesity, diabetes, smoking, and inactivity have increased the number of high-risk individuals, stressing the need for early screening.

These challenges emphasize the need for an accessible, intelligent prediction system to support early detection and informed decision-making.

OUR SOLUTION

We developed an AI-powered system designed to support early detection and clinical decision-making for heart disease.

The solution includes two core components:

1. Heart Disease Risk Classification Model

A machine-learning model that analyzes key patient features (age, BMI, cholesterol, lifestyle, medical history, etc.) to classify individuals into Low, Medium, or High heart-disease risk.

This helps clinicians quickly assess patient priority and identify those needing immediate attention.

2. Medical Recommendation Support

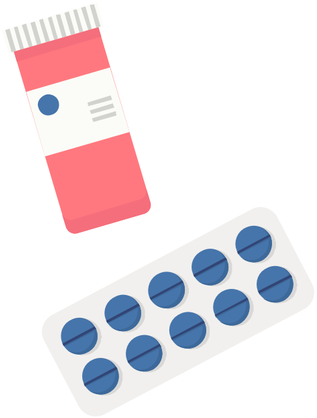
Based on the predicted risk level, the system provides structured medical recommendations to support doctors in making faster and more informed decisions—especially in busy or emergency environments.

Impact

This solution improves early identification of heart-disease cases, enhances diagnostic consistency, reduces missed high-risk patients, and supports healthcare teams with reliable, data-driven insights.

03

METHODOLOGY



Data Collection

Data Cleaning
& Preprosessing

EDA

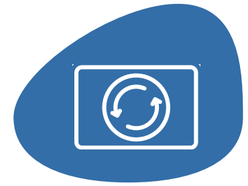
Feature Engineering
& Selection

Model Selection

Model
Training

Model Evaluation

Model
Deployment



Model Monitoring
& Maintenance

DATASET COLLECTION

The dataset contains 319,794 records and 18 features, representing personal and lifestyle health indicators.

Feature Types

- Numerical features (4):
  - BMI
  - PhysicalHealth
  - MentalHealth
  - SleepTime

- Categorical features (14):
  Smoking, AlcoholDrinking, Stroke, DiffWalking, Sex, AgeCategory, Race, Diabetic, PhysicalActivity, GenHealth, Asthma, KidneyDisease, SkinCancer, and the target HeartDisease.

Target Variable

- HeartDisease (Yes / No)

```
print(df.info())
```

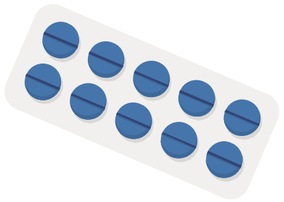
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 319794 entries, 0 to 319793
Data columns (total 18 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   HeartDisease      319794 non-null  object
 1   BMI               287815 non-null  float64
 2   Smoking           319794 non-null  object
 3   AlcoholDrinking   319794 non-null  object
 4   Stroke            319794 non-null  object
 5   PhysicalHealth    319794 non-null  int64
 6   MentalHealth      319794 non-null  int64
 7   DiffWalking       319794 non-null  object
 8   Sex               319794 non-null  object
 9   AgeCategory       319794 non-null  object
 10  Race              319794 non-null  object
 11  Diabetic          319794 non-null  object
 12  PhysicalActivity  319794 non-null  object
 13  GenHealth         319794 non-null  object
 14  SleepTime         310201 non-null  float64
 15  Asthma            319794 non-null  object
 16  KidneyDisease     319794 non-null  object
 17  SkinCancer        297409 non-null  object
dtypes: float64(2), int64(2), object(14)
memory usage: 43.9+ MB
None
```

DESCRIPTIVE STATISTICS SUMMARY

- Shows the basic statistical summary for BMI, PhysicalHealth, MentalHealth, SleepTime.
- BMI average ≈ 28 → indicates overweight trend in the population.
- Physical & Mental Health: median = 0 days, but max = 30 → presence of people with chronic issues.
- SleepTime average ≈ 7 hours → generally healthy sleep patterns.
- Extreme min/max values suggest possible outliers that may need cleaning.

```
df.describe()
```

	BMI	PhysicalHealth	MentalHealth	SleepTime
count	287815.000000	319794.000000	319794.000000	310201.000000
mean	28.320311	3.371721	3.898378	7.096618
std	6.348316	7.950860	7.955245	1.436282
min	12.020000	0.000000	0.000000	1.000000
25%	24.030000	0.000000	0.000000	6.000000
50%	27.320000	0.000000	0.000000	7.000000
75%	31.410000	2.000000	3.000000	8.000000
max	94.850000	30.000000	30.000000	24.000000



DUPLICATED DATA

Removing Duplicate Records

- Identified **21,039** duplicate rows in the dataset.
- Removed all duplicates to ensure data quality and prevent bias.
- Dataset size reduced from **319,794 → 298,755** rows.
- Final dataset contains 0 duplicates.

```
print("Number of duplicate rows:", df.duplicated().sum())
print("Before dropping duplicates:", df.shape)
```

```
Number of duplicate rows: 21039
Before dropping duplicates: (319794, 18)
```

```
df = df.drop_duplicates()
print("After dropping duplicates:", df.shape)
print("Number of duplicate rows:", df.duplicated().sum())
```

```
After dropping duplicates: (298755, 18)
Number of duplicate rows: 0
```

MISSING VALUES

Imputed missing values based on HeartDisease class:

- **BMI & SleepTime:** filled with median of each class
- **SkinCancer:** filled with mode of each class

All missing values are now handled. Dataset contains 0 missing values, ensuring data quality and consistency.

```
df.isna().sum()
```

HeartDisease	0
BMI	23850
Smoking	0
AlcoholDrinking	0
Stroke	0
PhysicalHealth	0
MentalHealth	0
DiffWalking	0
Sex	0
AgeCategory	0
Race	0
Diabetic	0
PhysicalActivity	0
GenHealth	0
SleepTime	9441
Asthma	0
KidneyDisease	0
SkinCancer	22046

dtype: int64

```
df.isna().sum()
```

HeartDisease	0
BMI	0
Smoking	0
AlcoholDrinking	0
Stroke	0
PhysicalHealth	0
MentalHealth	0
DiffWalking	0
Sex	0
AgeCategory	0
Race	0
Diabetic	0
PhysicalActivity	0
GenHealth	0
SleepTime	0
Asthma	0
KidneyDisease	0
SkinCancer	0

dtype: int64

DATA TRANSFORMATION

1. Encoding

- **Binary Encoding:** Converted Yes/No variables to numeric (0/1).
- **One-Hot Encoding:** Applied to multi-category features (Race, Diabetic).
- **Ordinal Encoding:** Used for ordered categories (AgeCategory, GenHealth).

2. Scaling

- **RobustScaler:** Applied to features affected by outliers (BMI, PhysicalHealth, MentalHealth, SleepTime)
- **StandardScaler:** Applied to normally distributed numeric features (AgeCategory)

Why These Steps Matter

- Prepare features for ML algorithms
- Reduce outlier impact
- Improve model performance and stability
- Ensure consistent feature scaling

OUTLIER ANALYSIS

1. BMI has many extreme outliers

- A large number of BMI values were far outside the normal range, with some exceeding 60, which is medically unrealistic.
- These were treated as data-entry errors, so extreme BMI values were capped at the median to avoid skewing the model.

2. PhysicalHealth and MentalHealth high values are NOT errors

- High values in these two columns represent the number of days with poor health in the last month (0–30 days).
- These “outliers” are meaningful real cases, not invalid data — so they were kept.

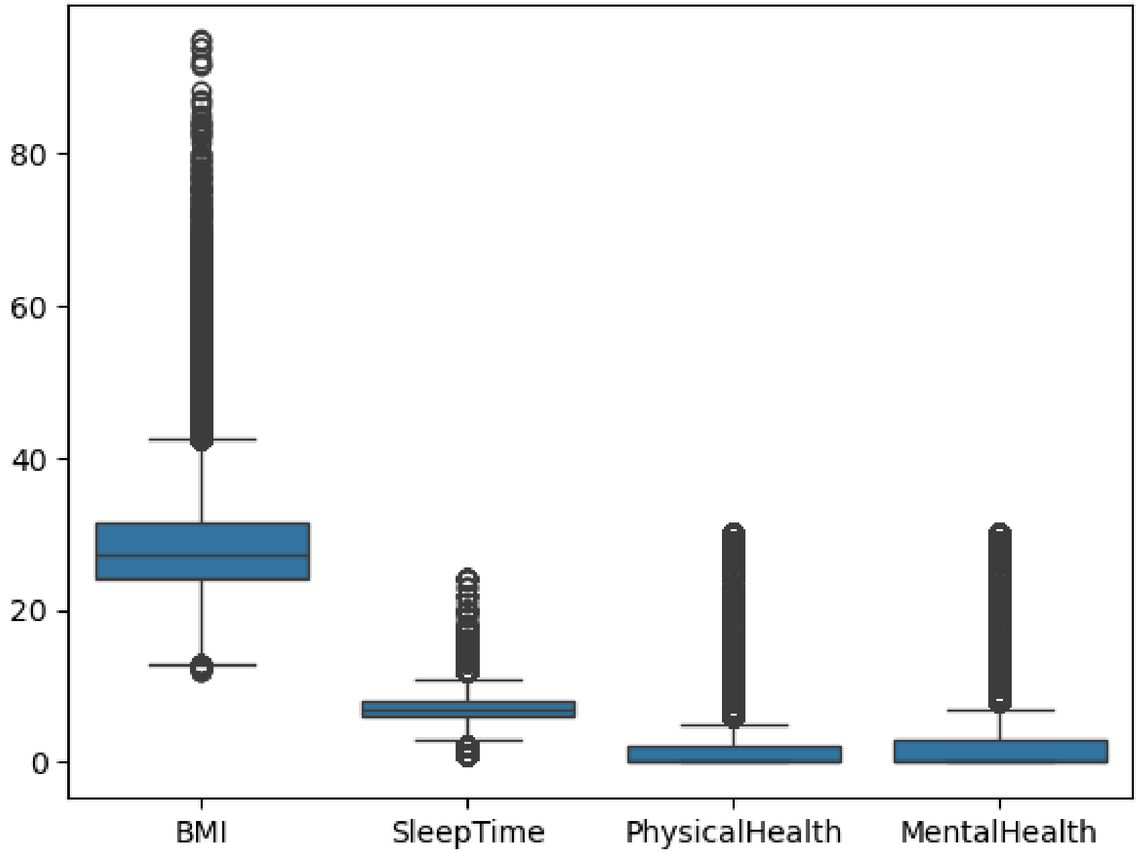
3. SleepTime contained unrealistic values

- Some participants reported 1 hour or 27 hours of sleep, which is clearly impossible.
- Values below 3 hours or above 18 hours were removed to improve dataset quality.

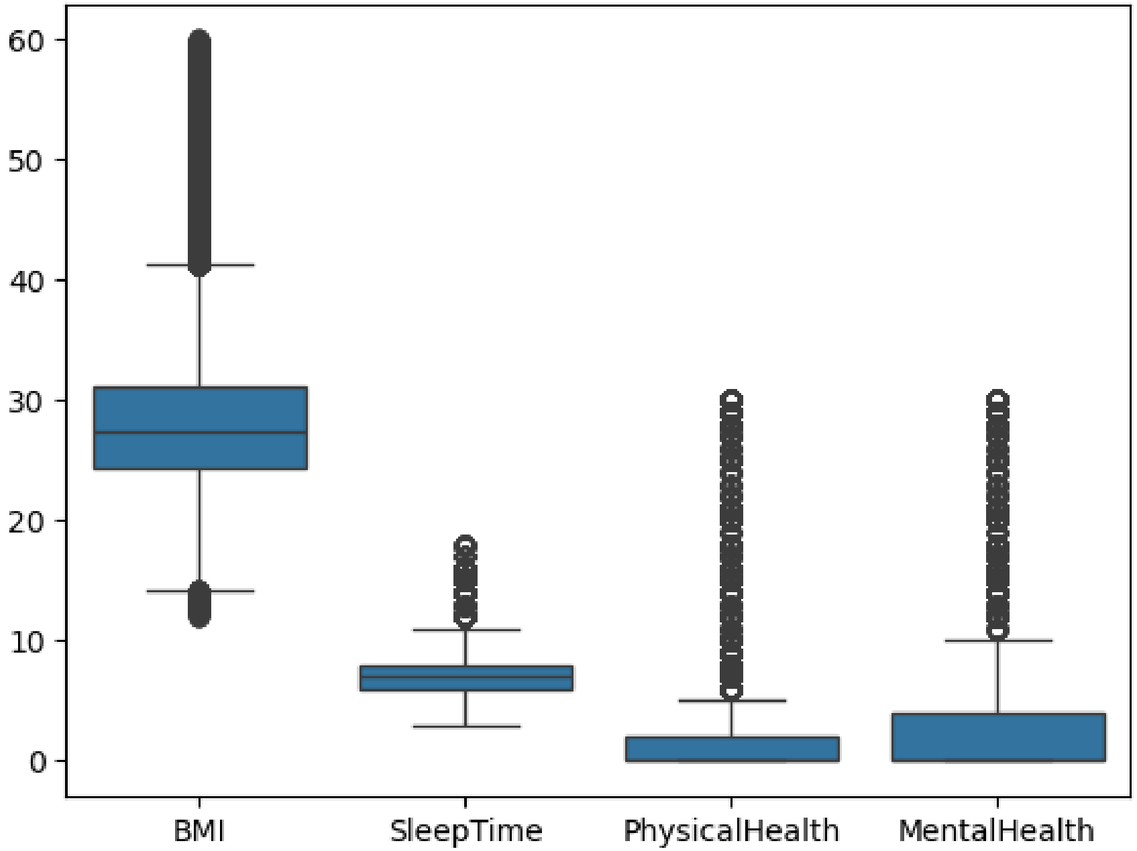
4. Final data cleaning improved reliability

- After handling outliers, the dataset became more realistic and consistent, enhancing the accuracy of later modeling steps.

FROM

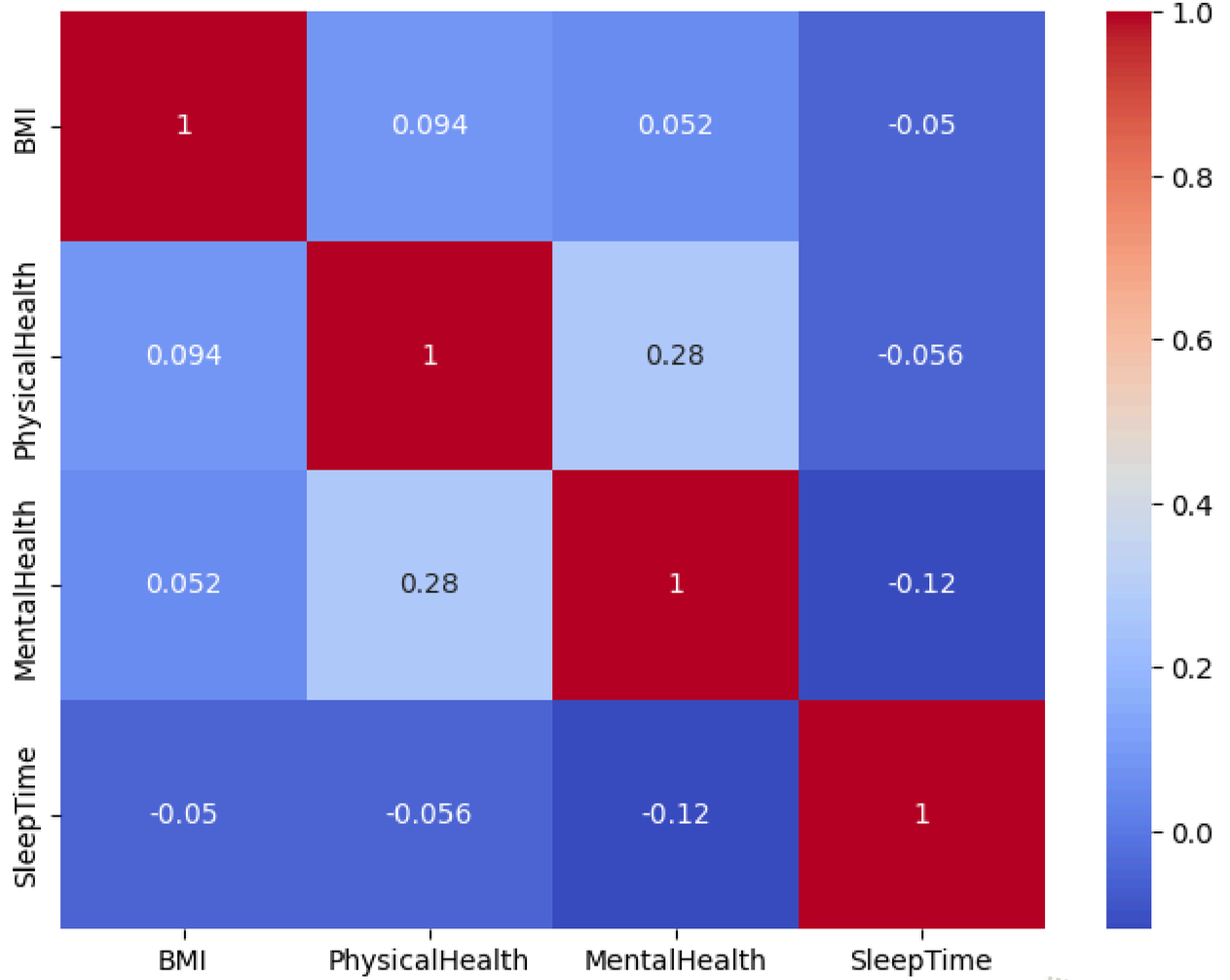

80
60
40
20
0
BMI
SleepTime
PhysicalHealth
MentalHealth

TO


60
50
40
30
20
10
0
BMI
SleepTime
PhysicalHealth
MentalHealth

INSIGHTS

- **Physical vs Mental Health (0.28):** Clear mind–body connection; worse physical health often means worse mental health.
- **Mental Health & Sleep (-0.12):** Poor mental health reduces sleep duration → increases heart-risk.
- **Weak BMI Correlations:** BMI has almost no daily effect on sleep or health symptoms → acts as a silent risk factor.
- **Low Multicollinearity:** No strong correlations → all features are valid inputs for ML modeling.



CLASS IMBALANCE PROBLEM

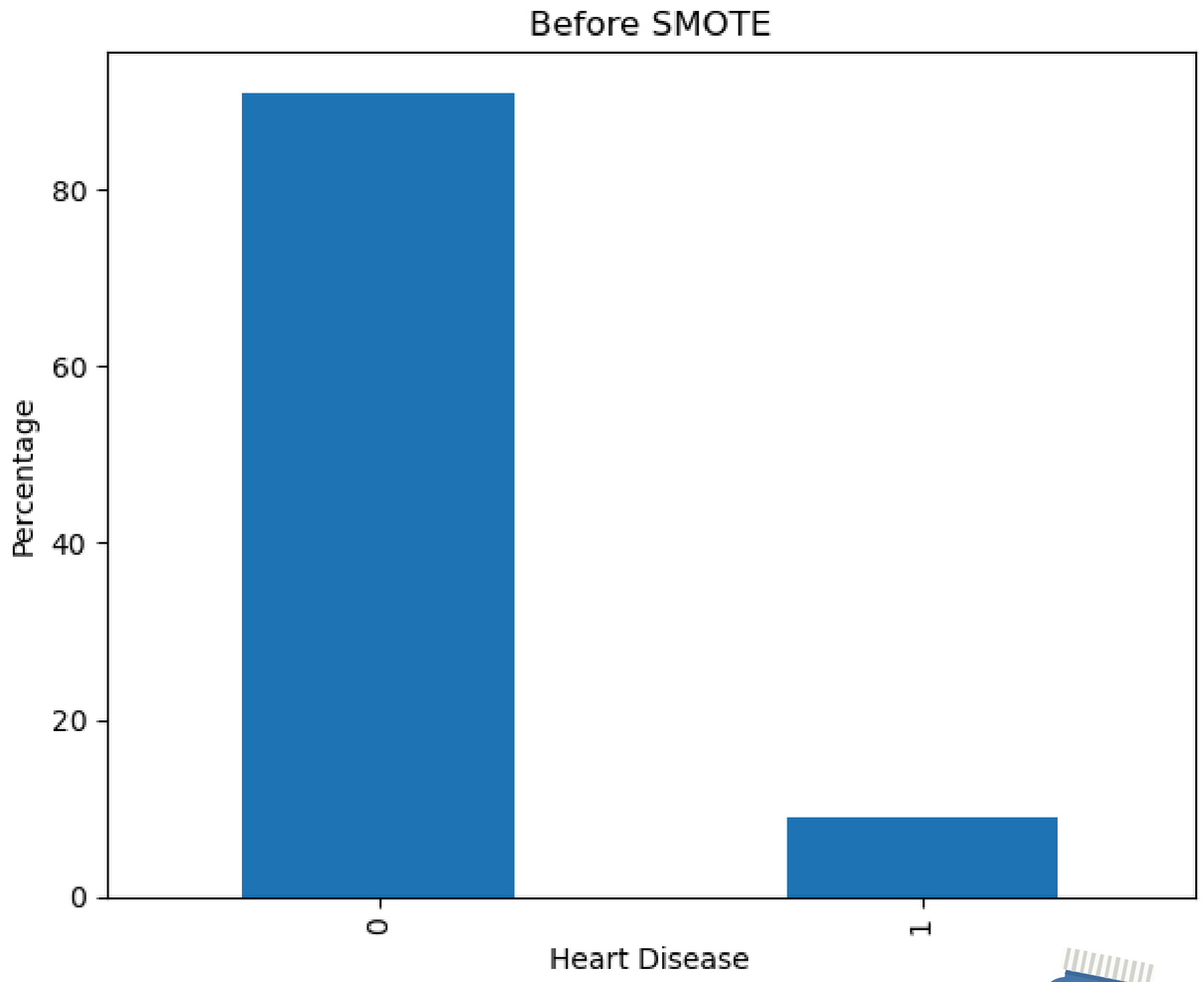
The target variable is highly **imbalanced:**

- No Heart Disease: 209,435 (≈ 91%)
- Heart Disease: 20,889 (≈ 9%)

Why This Is a Problem?

Class imbalance creates major challenges for machine learning models:

- **Model Bias:** The model tends to predict the majority class, leading to low recall for heart-disease cases.
- **Poor Minority Learning:** Not enough examples for the model to learn real patterns → missed detections.
- **Overfitting Risk:** The model may memorize patterns of the majority class while ignoring minority examples.



SOLUTION — BorderlineSMOTE + ENN

What We Did

- Applied **BorderlineSMOTE** → generated realistic minority samples.
- Used **ENN** → removed noisy majority samples.

What Happened

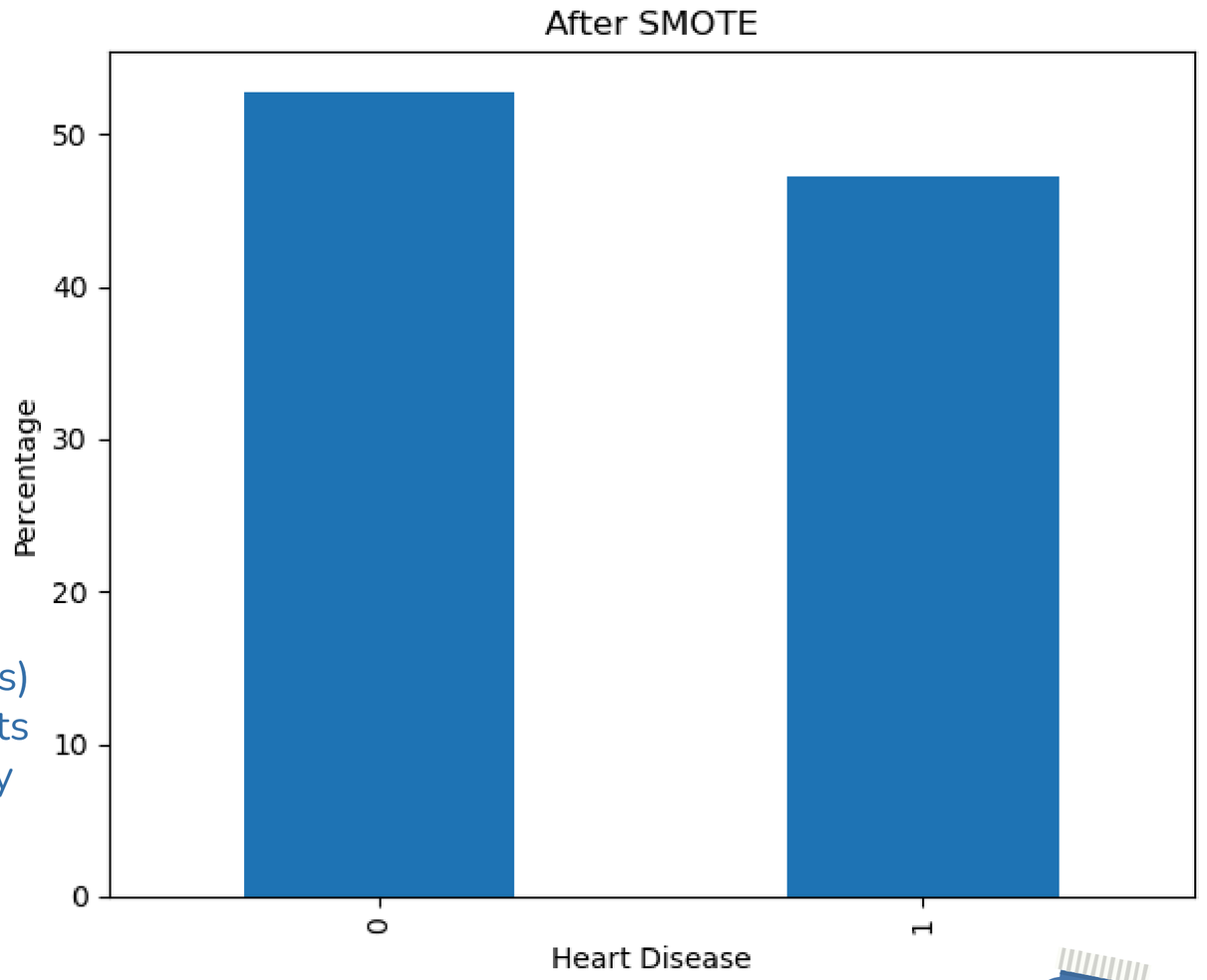
- Minority class increased from: 20,889 → 187,605
- Majority class remained: 209,435
- Duplicates after resampling: 4,307 (expected due to oversampling)
- AgeCategory distribution: Slight shift but overall preserved (no extreme distortion).

Why This Works

- Balances the dataset (from 9% minority → nearly equal classes)
- Improves class separation after **ENN** removes ambiguous points
- Enhances recall for heart-disease cases (model now sees many more positive examples)

Outcome

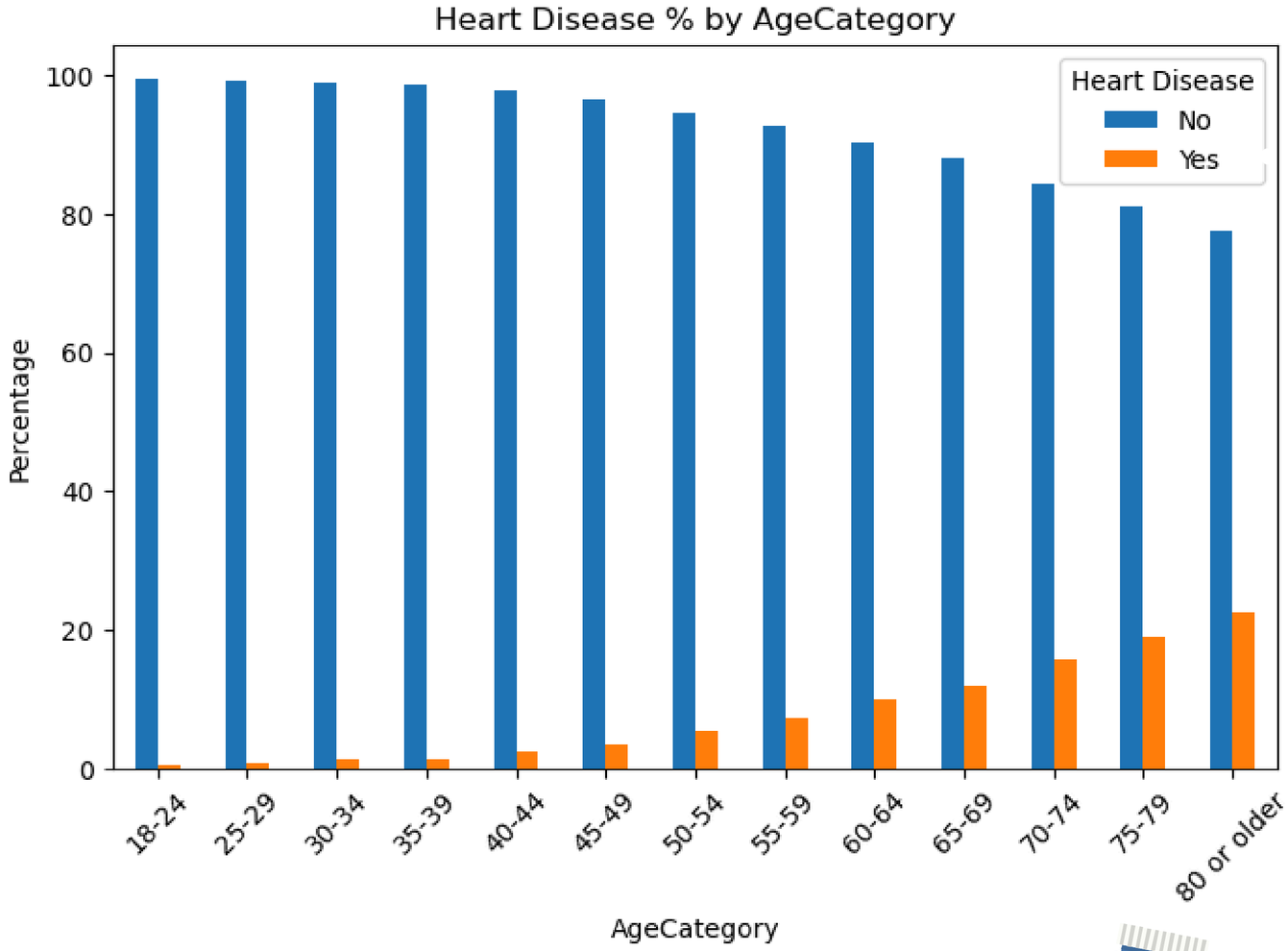
A balanced, cleaner, and more informative dataset — much better for model training.



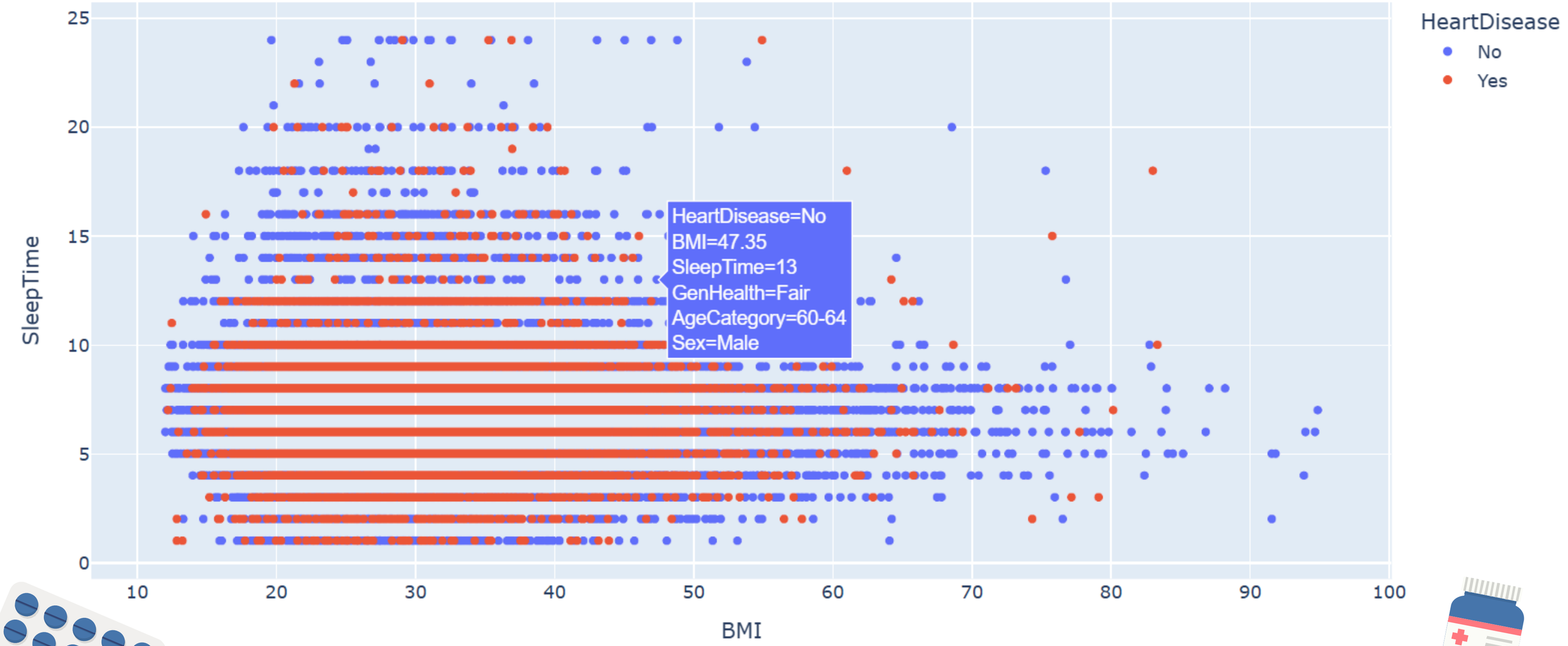
DATA VISUALIZATION

Insights:

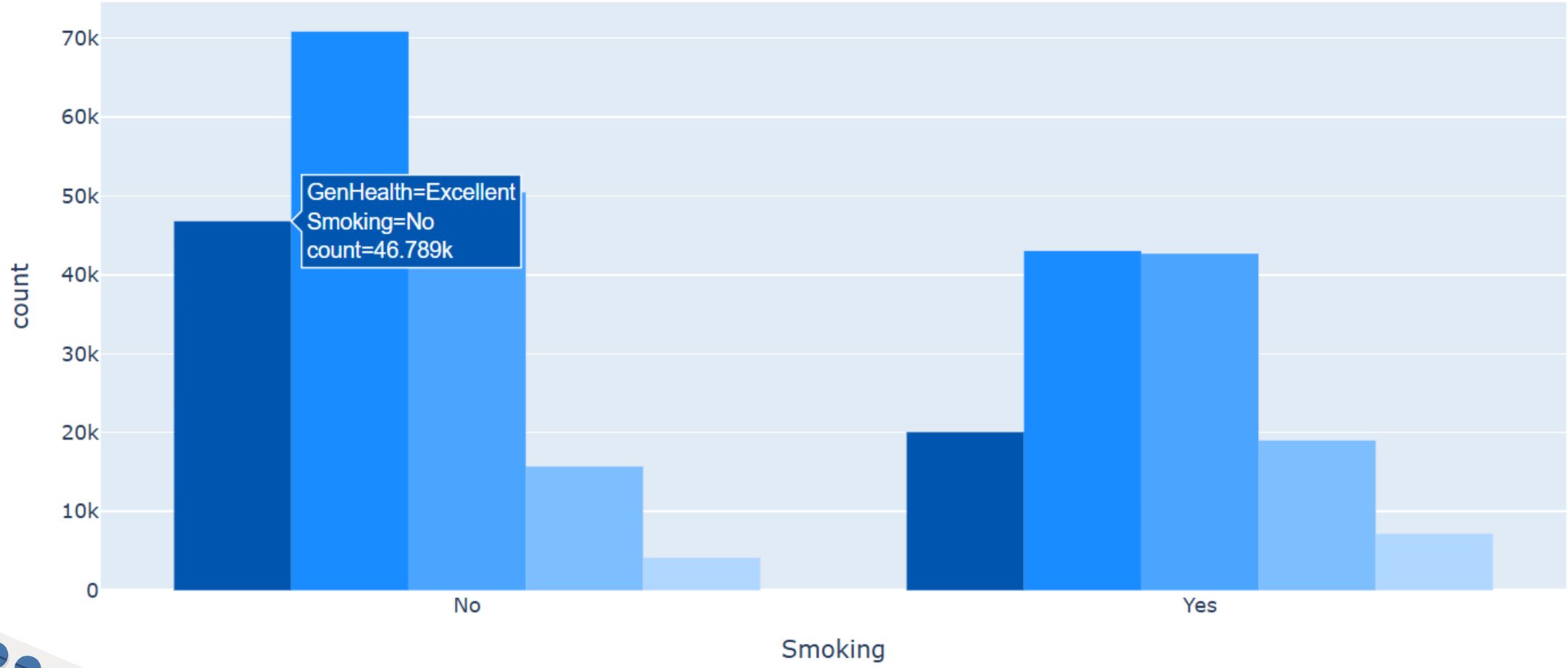
- Heart disease increases clearly with age.
- Very low in younger groups (18–44), then rises starting from age 50+.
- Highest prevalence appears in the 70–79 and 80+ groups.
- Age is a strong and consistent risk factor.



BMI VS SLEEPTIME (INTERACTIVE)



SMOKING VS SELF-REPORTED HEALTH

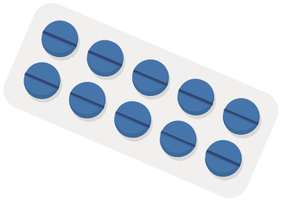


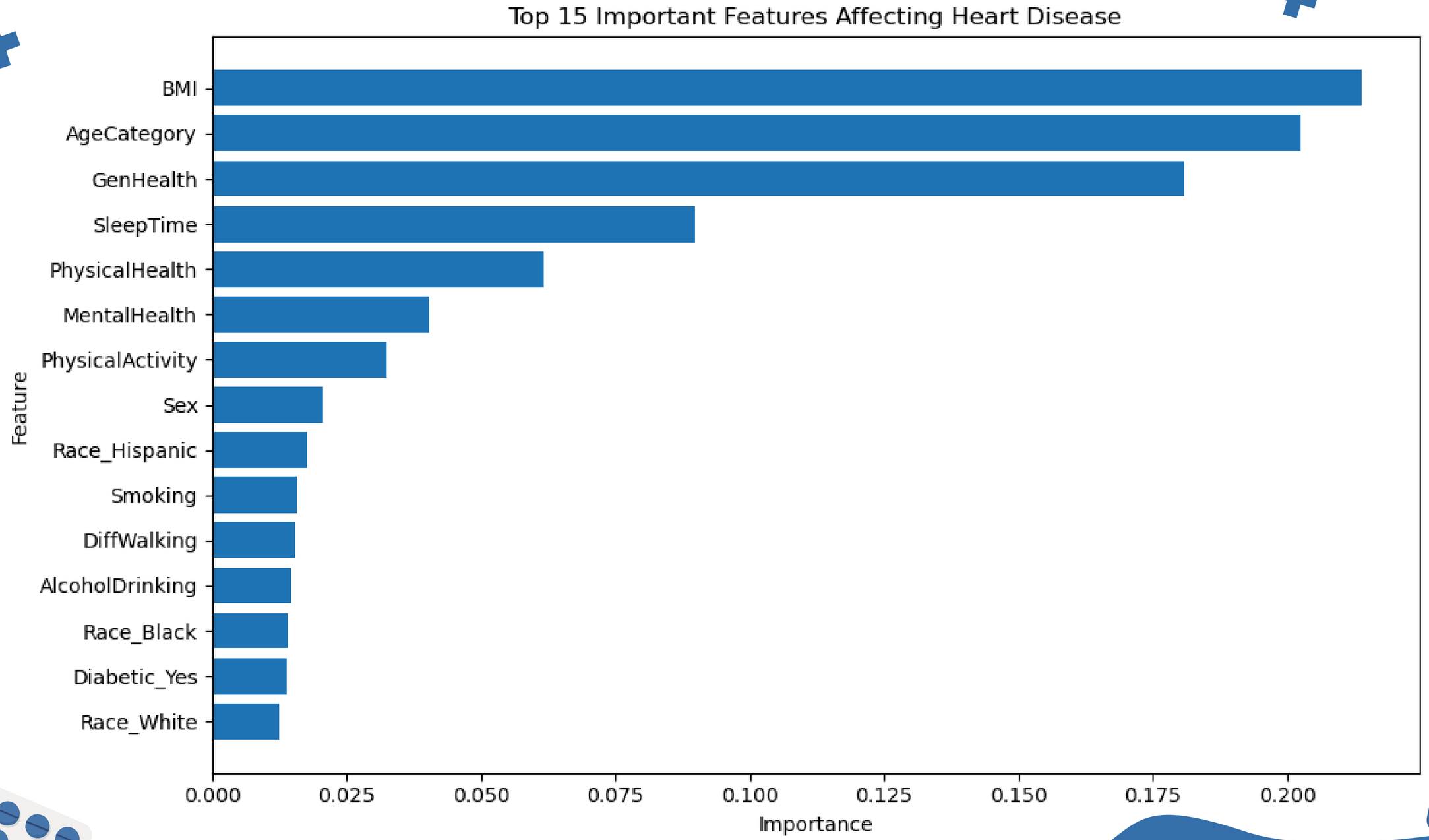
FEATURE IMPORTANCE

```python
from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier(n_estimators=200, random_state=42)
rf.fit(X_train_resampled, y_train_resampled)

importances = rf.feature_importances_

indices = np.argsort(importances)[::-1]

feature_names = X_train_resampled.columns
top_features = pd.DataFrame({
    'Feature': feature_names[indices],
    'Importance': importances[indices]
})
```

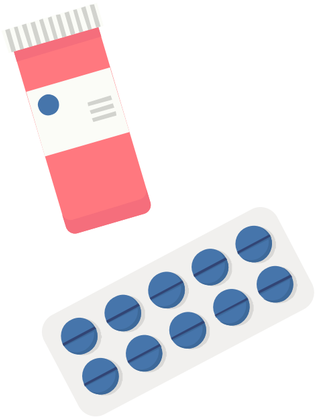
	Feature	Importance
0	BMI	0.213998
1	AgeCategory	0.202626
2	GenHealth	0.181045
3	SleepTime	0.089910
4	PhysicalHealth	0.061678
5	MentalHealth	0.040303
6	PhysicalActivity	0.032561
7	Sex	0.020670
8	Race_Hispanic	0.017770
9	Smoking	0.015896
10	DiffWalking	0.015526
11	AlcoholDrinking	0.014580
12	Race_Black	0.014212
13	Diabetic_Yes	0.013835
14	Race_White	0.012403

We trained a Random Forest on the balanced dataset to identify the most important features influencing heart disease.




Top 15 Important Features Affecting Heart Disease
BMI
AgeCategory
GenHealth
SleepTime
PhysicalHealth
MentalHealth
PhysicalActivity
Sex
Race_Hispanic
Smoking
DiffWalking
AlcoholDrinking
Race_Black
Diabetic_Yes
Race_White
Feature
0.000
0.025
0.050
0.075
0.100
0.125
0.150
0.175
0.200
Importance

MODEL SELECTION



BASELINE MODEL EVALUATION

Objective:

- Evaluate multiple machine learning models to establish baseline performance before hyperparameter tuning or optimization.

```
# Take a stratified sample (e.g., 10%) for fast experimentation
X_sample, _, y_sample, _ = train_test_split(
    X_train, y_train, test_size=0.9, stratify=y_train, random_state=42
)

print("Sample shape:", X_sample.shape)
```

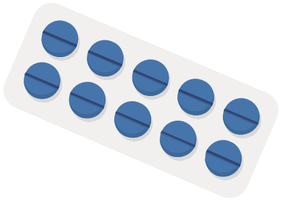
```
Sample shape: (39704, 23)
```

Models Tested:

We trained and evaluated the following models to compare their baseline performance:

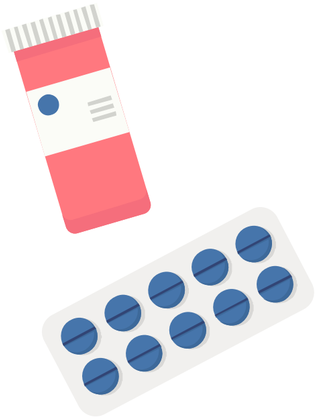
- Logistic Regression
- Decision Tree Classifier
- Random Forest Classifier
- Gradient Boosting Classifier
- MLP Classifier (Neural Network)

```
models = {
    "Logistic Regression": LogisticRegression(max_iter=1000, random_state=42),
    "Decision Tree": DecisionTreeClassifier(random_state=42),
    "Random Forest": RandomForestClassifier(n_estimators=100, random_state=42),
    "Gradient Boosting": GradientBoostingClassifier(random_state=42),
    "Neural Network (MLP)": MLPClassifier(hidden_layer_sizes=(64, 32),
                                          max_iter=300, random_state=42)
}

results_sample = []
```



04

RESULTS



🔸 Model: Logistic Regression

	precision	recall	f1-score	support
0	0.8428	0.8094	0.8258	20943
1	0.7962	0.8315	0.8135	18761
accuracy			0.8198	39704
macro avg	0.8195	0.8204	0.8196	39704
weighted avg	0.8208	0.8198	0.8199	39704

🔸 Model: Decision Tree

	precision	recall	f1-score	support
0	0.9999	1.0000	1.0000	20943
1	1.0000	0.9999	0.9999	18761
accuracy			0.9999	39704
macro avg	1.0000	0.9999	0.9999	39704
weighted avg	0.9999	0.9999	0.9999	39704

🔸 Model: Random Forest

	precision	recall	f1-score	support
0	1.0000	0.9999	1.0000	20943
1	0.9999	1.0000	0.9999	18761
accuracy			0.9999	39704
macro avg	0.9999	1.0000	0.9999	39704
weighted avg	0.9999	0.9999	0.9999	39704

🔸 Model: Gradient Boosting

	precision	recall	f1-score	support
0	0.9034	0.8827	0.8930	20943
1	0.8724	0.8947	0.8834	18761
accuracy			0.8884	39704
macro avg	0.8879	0.8887	0.8882	39704
weighted avg	0.8887	0.8884	0.8884	39704

🔸 Model: Neural Network (MLP)

	precision	recall	f1-score	support
0	0.8748	0.8314	0.8526	20943
1	0.8217	0.8672	0.8438	18761
accuracy			0.8483	39704
macro avg	0.8483	0.8493	0.8482	39704
weighted avg	0.8497	0.8483	0.8484	39704

🔷 Baseline Results on Sample:

	Model	Accuracy	Recall	Precision	F1
2	Random Forest	0.999950	1.000000	0.999893	0.999947
1	Decision Tree	0.999950	0.999893	1.000000	0.999947
3	Gradient Boosting	0.888374	0.894675	0.872356	0.883375
4	Neural Network (MLP)	0.848328	0.867225	0.821676	0.843836
0	Logistic Regression	0.819817	0.831459	0.796233	0.813465

KEY INSIGHTS

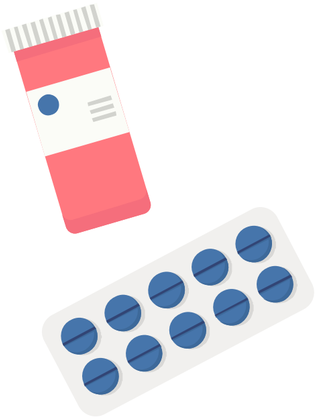
- Tree-based models (Decision Tree & Random Forest) showed perfect accuracy, indicating extreme overfitting → not suitable without regularization.
- Logistic Regression offers stable—but limited—performance.
- MLP performs well but needs tuning for better precision.
- Gradient Boosting showed the best balance across all metrics (Accuracy, Precision, Recall, F1).

WHY WE SELECTED GRADIENT BOOSTING

Gradient Boosting was chosen as the main model for further optimization because:

- Strong overall performance on the dataset
- Higher Recall compared to Logistic Regression and MLP
- Not overfitted, unlike Decision Tree and Random Forest
- Well-suited for tabular data and handles feature interactions effectively

MODEL TRAINING



GRADIENT BOOSTING TRAINING WITH CROSS-VALIDATION

We used 5-Fold Stratified Cross-Validation to evaluate the model reliably and avoid bias due to class imbalance.

Key insights:

- Very stable results (low standard deviation)
- Strong balance between Recall and Precision
- High ROC-AUC → excellent class separability

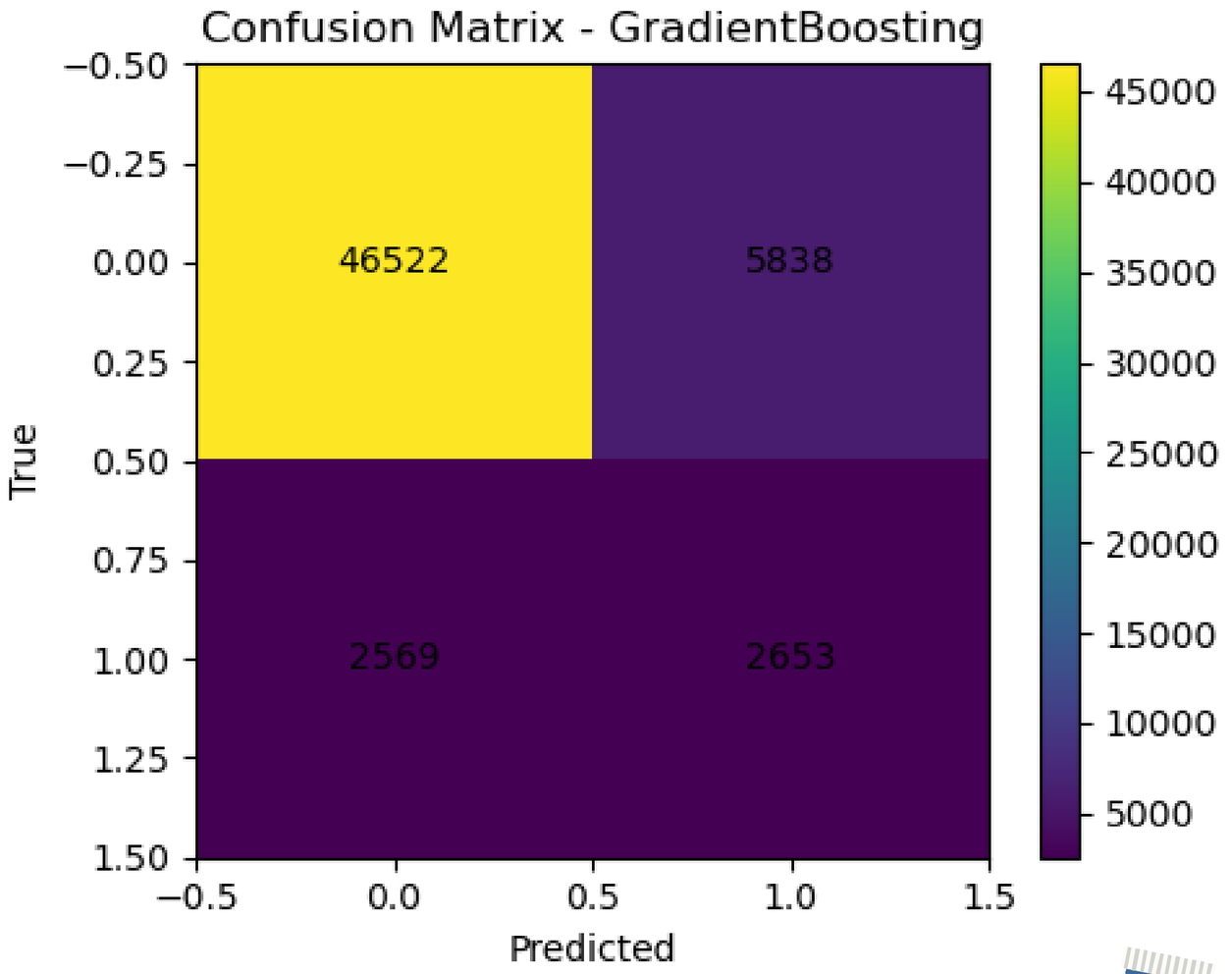
```python
# --- CV on training set ---
from joblib import parallel_backend

gb = GradientBoostingClassifier(random_state=42)
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
scoring = ['accuracy', 'precision', 'recall', 'f1', 'roc_auc']

# Use threading backend to avoid pickling/un-serialization errors
# when parallelizing in some environments
with parallel_backend('threading'):
    cv_res = cross_validate(gb, X_train, y_train, cv=cv, scoring=scoring, n_jobs=-1)

for metric in scoring:
    key = 'test_' + metric
    if key in cv_res:
        print(f"CV {metric}: mean={np.mean(cv_res[key]):.4f}, std={np.std(cv_res[key]):.4f}")
```

```
CV accuracy: mean=0.8866, std=0.0015
CV precision: mean=0.8729, std=0.0023
CV recall: mean=0.8895, std=0.0019
CV f1: mean=0.8811, std=0.0015
CV roc_auc: mean=0.9642, std=0.0007
```

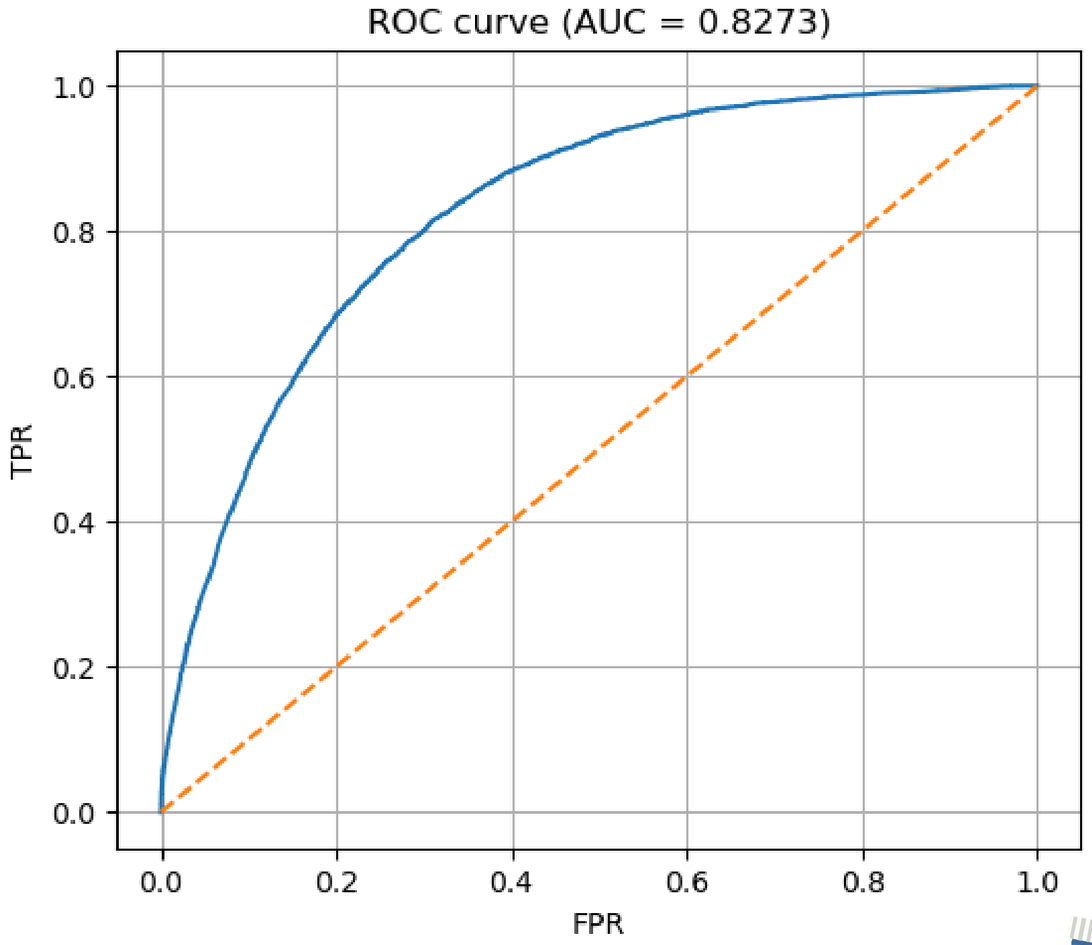
MODEL EVALUATION

- 46,522 true negatives → the model is very good at identifying class 0.
- 2,653 true positives → it correctly detects some positive cases.
- 5,838 false positives → model often predicts 1 when it is actually 0.
- 2,569 false negatives → many positive cases are missed.

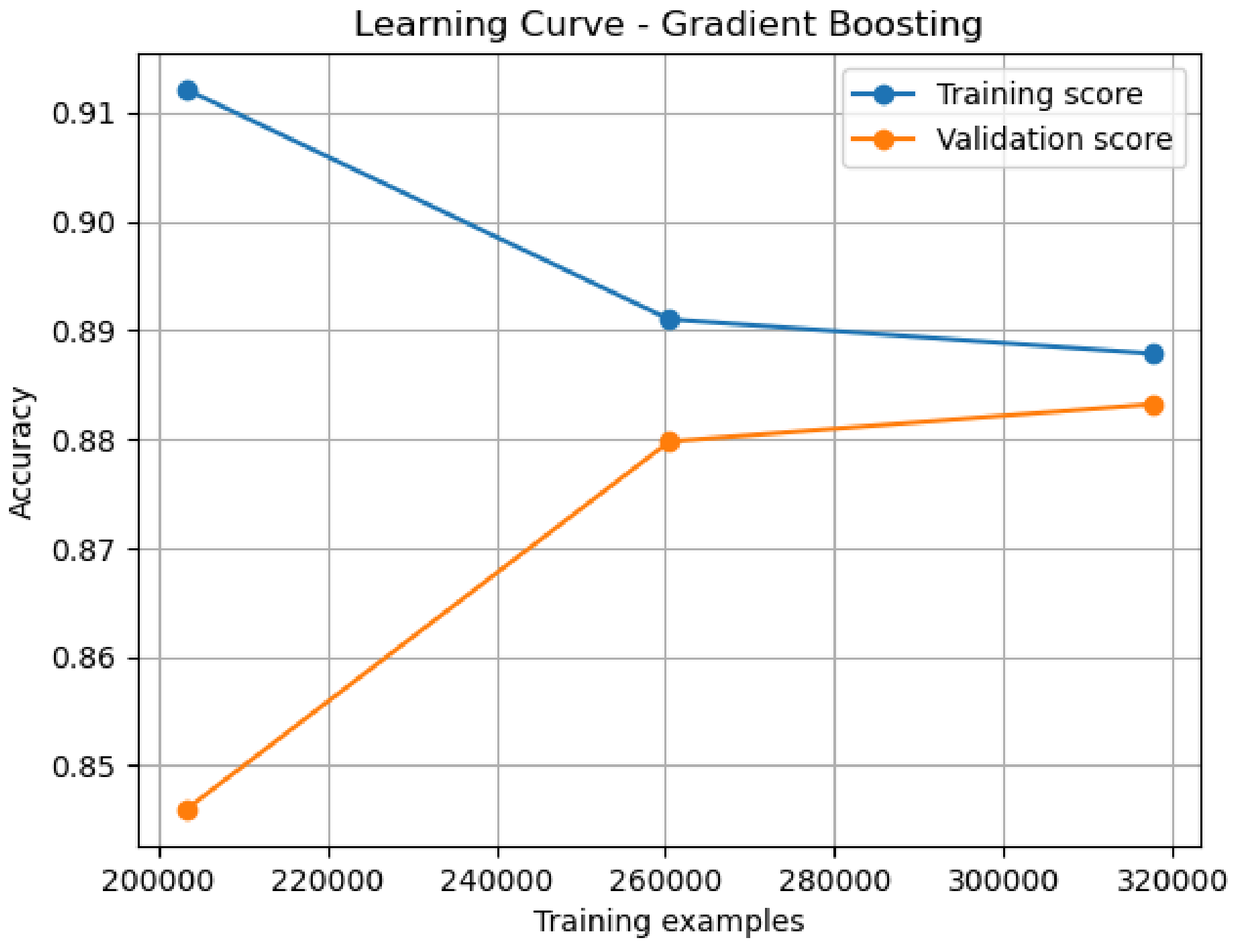
Overall: the model performs well on the majority class, but struggles with the minority class, leading to low precision and moderate recall for class 1.



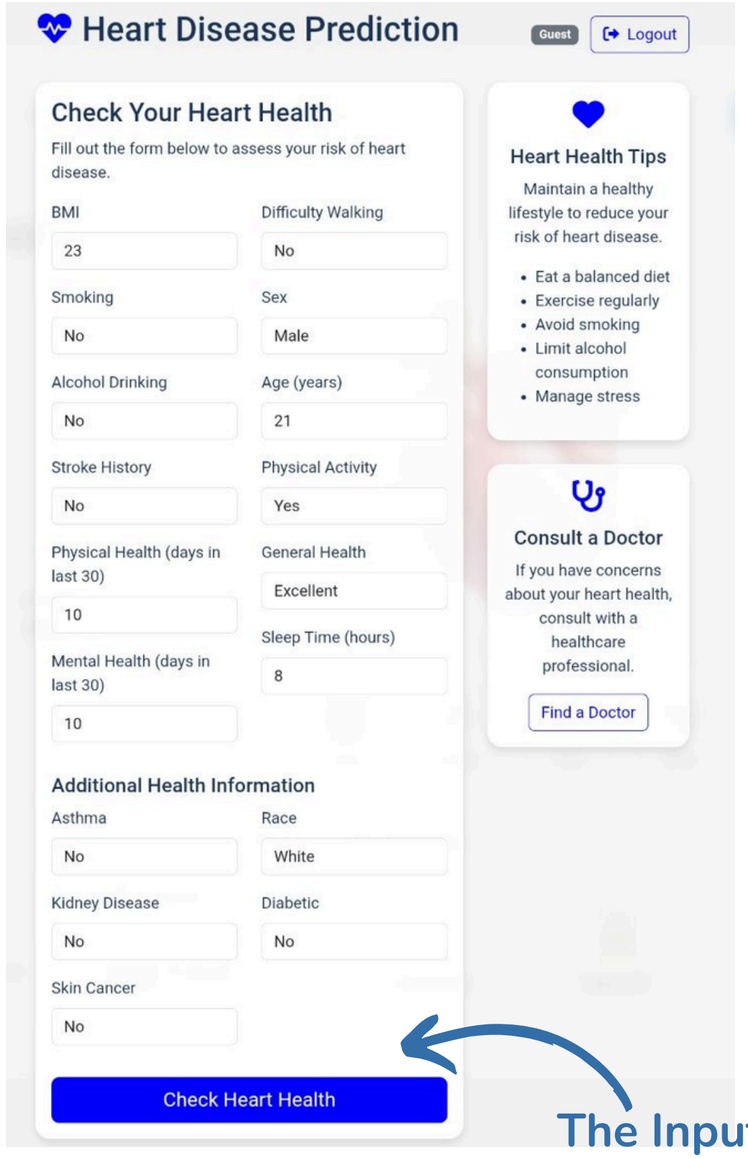
- The ROC curve shows good separation between the classes.
- AUC = 0.8273, indicating strong discriminative ability.
- The model maintains a high True Positive Rate across most False Positive Rates.
- Performance is clearly better than random guessing (dashed line).
- Overall, the model can rank positive vs. negative cases effectively despite class imbalance.



- Training accuracy decreases slightly as data increases → the model is not memorizing.
- Validation accuracy increases steadily → the model benefits from more data.
- The gap between training and validation scores becomes small → no major overfitting.
- Overall, Gradient Boosting shows good generalization and is still improving with larger training sizes.



DEPLOYMENT

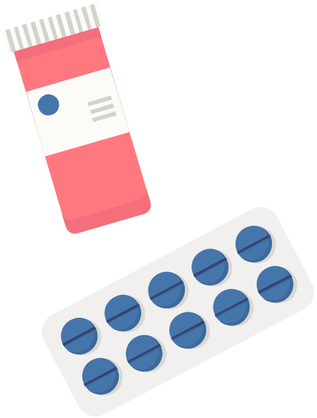


The Input form for the user

The output:

06

FUTURE WORK



- Advanced Data Balancing: Evaluate more robust imbalance-handling techniques such as SMOTE-ENN, Borderline-SMOTE, and ADASYN on the full dataset.

- Hyperparameter Optimization: Run a full GridSearchCV or Bayesian Optimization to fine-tune Gradient Boosting for maximum recall.

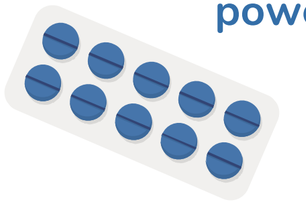
- Model Exploration: Experiment with stronger tree-based models like XGBoost, LightGBM, and CatBoost.

- Expanded Data Collection: Incorporate more detailed clinical attributes (labs, ECG readings, lifestyle factors).

Computer Vision Integration:

- Use medical imaging (e.g., chest X-rays, CT scans, echocardiography) with deep learning models to detect early structural signs of heart disease and enrich predictions beyond tabular data.

- This would allow combining clinical data + image data for a more powerful and realistic diagnostic system.



Team Members

Salma Salah

Fatema Taher

Heba Adel Ali

Raghad Hamdy

Abdelrahman Ebrahim

THANKS!