Leveraging Machine Learning Algorithms for cardiovascular disease predictions

Research Question

Can machine learning classification algorithms accurately predict the presence of cardiovascular diseases using readily available clinical measurements, and screen for cardio vascular diseases effectively.

Relevance

Cardiovascular disease (CVD) remains the leading cause of death globally, accounting for approximately 17.9 million deaths annually (WHO, 2023). Early detection through accurate risk assessment enables timely intervention—lifestyle modifications, medication, or surgical procedures—that can significantly improve patient outcomes and reduce mortality.

Machine learning offers the potential to analyze complex patterns across multiple clinical measurements simultaneously, potentially exceeding the accuracy of traditional single-factor risk assessments or simple clinical scoring systems.

Theory and Background

Clinical Context

Heart Disease (Coronary Artery Disease)

Coronary artery disease (CAD), the most common form of heart disease, occurs when coronary arteries—vessels supplying blood to the heart muscle—become narrowed or blocked due to atherosclerotic plaque buildup. This process, often developing over decades, reduces blood flow to the heart (ischemia), potentially causing chest pain (angina), heart attacks (myocardial infarction), or sudden cardiac death.

Stroke (Cerebrovascular Disease)

Stroke, or cerebrovascular accident (CVA), occurs when blood flow to the brain is interrupted, causing brain cells to die within minutes due to oxygen deprivation. Stroke is the second leading cause of death globally and a leading cause of long-term disability.

In this project, we use two types of classification model namely Logistic Regression and Random Forest. Logistic Regression assumes Linear Relationship and outputs the probabilities of the instance. This is a simple classification model and serves as a good starting point for classification tasks.

For Capturing Non Linear Relationships, Random Forest Classification is used. This model combines many decision trees and outputs the best node.

Both of these models fall under Supervised Learning and it learns from the input features and predicts the Output Features.

Problem Statement

Dataset-1
Given a patient's clinical measurements and diagnostic test results, predict whether the patient has stroke or not.

- Input (Features)
  1. Patient ID
  2. Patient Name
  3. Age
  4. Gender
  5. Hypertension
  6. Heart Disease
  7. Marital Status
  8. Work Type
  9. Residence Type
  10. Average Glucose Level
  11. Body Mass Index (BMI)
  12. Smoking Status
  13. Alcohol Intake
  14. Physical Activity
  15. Stroke History
  16. Family History of Stroke
  17. Dietary Habits
  18. Stress Levels
  19. Blood Pressure Levels
  20. Cholesterol Levels
  21. Symptoms

Output (Target)

Diagnosis

Dataset-2

Given a set of Clincal Measurements, predict whether a patient will have heart disease or not
Input Features

1. age
2. sex
3. cp
4. trestbps
5. chol
6. fbs
7. restecg
8. thalach
9. exang
10. oldpeak
11. slope
12. ca
13. thal

Output Features

target

Problem Analysis

Dataset-1 Stroke

Data Characteristics

Dataset: 15,000 patients, 22 features (4 continuous, 6 binary, 12 categorical)
Target: Binary - Stroke diagnosis
Quality: Some missing values in BMI, otherwise complete
Distributions: All continuous features (Age, Glucose, BMI, Stress) approximately normal (skewness: -0.01 to 0.02)

Assumptions:

- Self-reported data (diet, activity, stress) is reasonably accurate (questionable—introduces noise)
- Current health status predicts stroke risk (vs longitudinal trends)

Approach

Strategy:

1. **EDA:** Distribution analysis , data visualization, missing data assessment
2. **Preprocessing:**
   - Binary features → Label Encoding
   - Ordinal features → Order-preserving encoding (Low=0, Moderate=1, High=2)
   - Nominal features → One-Hot Encoding (Marital Status, Work Type)
   - Impute missing BMI with median
3. **Feature Scaling:** StandardScaler for all continuous features
4. **Modeling:** Train Logistic Regression and Random Forest with class_weight='balanced'
5. **Evaluation:** ROC-AUC, recall , precision, F1-score

Dataset-2 Heart Disease

Data Characteristics

Dataset: 1,025 patients, 13 features (5 continuous, 3 binary, 5 categorical)
Target: Binary - 30% no disease, 70% disease (moderate imbalance)
Quality: Complete data, no missing values
Distributions: Continuous features approximately normal; some right-skewed (trestbps, chol, oldpeak)

Constraints and Assumptions

Constraints:

- Moderate class imbalance (70-30) requires stratified splitting and class weighting
- Different feature scales (age: 29-77, cholesterol: 126-564) require StandardScaler for Logistic Regression
- Sample size (n=1,025) adequate for chosen algorithms but limits deep learning

Assumptions:

- Clinical measurements are accurate and standardized
- Binary classification (presence/absence) sufficient for screening
- 1988 Cleveland data generalizes to current populations (questionable)
- Features contain sufficient information for prediction

Approach

Strategy:

1. **EDA:** Distribution analysis, correlation study, class balance assessment
2. **Preprocessing:** Stratified 80-20 split, StandardScaler for continuous features

3. **Modeling:** Train Logistic Regression (linear baseline) and Random Forest (non-linear)
4. **Evaluation:** ROC-AUC, precision, recall, F1-score, confusion matrix
5. **Validation:** Compare feature importance to medical literature

The Approach for both the datasets look similar and most of data science projects follow the same step for all Supervised Learning Problems.

Solution Explanation

This project implements a comprehensive binary classification pipeline for predicting cardiovascular events across two distinct datasets: stroke prediction (Dataset 1) and heart disease diagnosis (Dataset 2). We systematically compare Logistic Regression (linear baseline) and Random Forest (ensemble approach) to determine which algorithm provides optimal performance for each cardiovascular condition. The solution encompasses data preprocessing tailored to each dataset's characteristics, model training with imbalance handling, rigorous multi-metric evaluation, and feature importance analysis validated against medical literature.

Step-by-step solution

- Load Data
  -> Read the Respective data
- Clean
  Check for null values and if the features are in same scale or not and then fill null if any
- Data Visualization
  Check for Normality and perform univariate analysis
- Transform Features
  Perform Binary, Ordinal and One hot encoding as needed.
- Split the data
  Split the data into train and test set.
- Train Models
  Train Logistic Regression and Random Forest Models
- Evaluate
  Evaluate models using F1-Score, Precision, Recall and Accuracy

ALGORITHM:

INPUT: Dataset D (stroke or heart disease) with features X and target y

OUTPUT: Trained models (LR, RF), performance metrics, feature importance

1. LOAD DATA
D ← READ_CSV(filepath)

2. IDENTIFY FEATURE TYPES
continuous ← [Age, Glucose, BMI, Stress, ...]
binary ← [Gender, Hypertension, ...]
ordinal ← [Smoking, Activity, Symptoms, ...]
nominal ← [Marital Status, Work Type, ...]

3. EXPLORATORY DATA ANALYSIS
FOR each feature f in continuous:
TEST_NORMALITY(f) //
PLOT_HISTOGRAM(f)
IDENTIFY missing_data_patterns()

4. HANDLE MISSING DATA
IF missing_percentage < 20%:
IMPUTE with median (continuous) or mode (categorical)
ELSE:
CONSIDER dropping feature or advanced imputation

5. ENCODE CATEGORICAL FEATURES
FOR each binary feature:
APPLY LabelEncoder() // 0/1
FOR each ordinal feature with categories:
CREATE mapping: {category[i]: i for i in range(len(categories))}
APPLY mapping
FILL unmapped with -1
FOR each nominal feature:
APPLY OneHotEncoder(drop_first=True)

6. PREPARE FEATURES AND TARGET
X ← DROP ['Patient ID', 'Patient Name', 'Diagnosis']
y ← ENCODE_TARGET(Diagnosis) // "stroke"→1, "no stroke"→0VERIFY y has both classes {0, 1}

7. STRATIFIED SPLIT
X_train, X_test, y_train, y_test ← SPLIT(
X, y,
test_size=0.2,
random_state=42,
stratify=y // Maintain class distribution
)VERIFY both sets have similar class proportions

8. STANDARDIZE FEATURES
scaler ← StandardScaler()
X_train_scaled ← scaler.FIT_TRANSFORM(X_train) // Learn μ, σ from train

X_test_scaled ← scaler.TRANSFORM(X_test) // Apply train μ, σ to testVERIFY mean ≈ 0, std ≈ 1 for X_train_scaled

9. TRAIN LOGISTIC REGRESSION
lr ← LogisticRegression(
max_iter=1000,
random_state=42,
class_weight='balanced' // Handle imbalance
)lr.FIT(X_train_scaled, y_train) // Use scaled data!

10. TRAIN RANDOM FOREST
rf ← RandomForestClassifier(
n_estimators=100,
max_depth=10,
random_state=42,
class_weight='balanced'
)rf.FIT(X_train, y_train) // Use unscaled data (trees don't need scaling)

11. MAKE PREDICTIONS
// Logistic Regression
y_pred_lr ← lr.PREDICT(X_test_scaled)
y_proba_lr ← lr.PREDICT_PROBA(X_test_scaled)[:, 1]// Random Forest
y_pred_rf ← rf.PREDICT(X_test)
y_proba_rf ← rf.PREDICT_PROBA(X_test)[:, 1]

12. CALCULATE METRICS
FOR each model m in [lr, rf]:
accuracy ← ACCURACY_SCORE(y_test, y_pred_m)
precision ← PRECISION_SCORE(y_test, y_pred_m)
recall ← RECALL_SCORE(y_test, y_pred_m)
f1 ← F1_SCORE(y_test, y_pred_m)
roc_auc ← ROC_AUC_SCORE(y_test, y_proba_m)

cm ← CONFUSION_MATRIX(y_test, y_pred_m)

STORE metrics_m ← {accuracy, precision, recall, f1, roc_auc, cm}

13. GENERATE VISUALIZATIONS
PLOT confusion_matrices (both models side-by-side)
PLOT ROC_curves (both models on same plot)
PLOT metrics_comparison (bar chart)

14. EXTRACT FEATURE IMPORTANCE
Logistic Regression
importance_lr ← ABSOLUTE(lr.coef_[0])
feature_ranking_lr ← SORT_DESCENDING(importance_lr)**Random Forest**
importance_rf ← rf.feature_importances_
feature_ranking_rf ← SORT_DESCENDING(importance_rf)

15. VISUALIZE IMPORTANCE
PLOT bar_chart(feature_ranking_lr)
PLOT bar_chart(feature_ranking_rf)
PLOT comparison_heatmap(importance_lr, importance_rf)

Why this approach is sound
We Preprocess data end to end and perform univariate analysis and then split the data into train and test sets, we then train model on train set and evaluate it using test set

- No Informaion Leakage
  The test set remains statistically independent of all training procedures.
- Stratification
  Both train and test sets maintain original class distribution
- Empirical Validation
  Proves that data quality is the foremost thing

Results

1.6 Results and Discussion

Performance Summary

Dataset	LR Accuracy	RF Accuracy	Best ROC-AUC	Outcome
Stroke	50%	51%	0.51	Failed (random)
Heart Disease	78%	99%	~0.99	Excellent

Dataset 1: Stroke Prediction

Results: Both models ~50% accuracy (random guessing level)

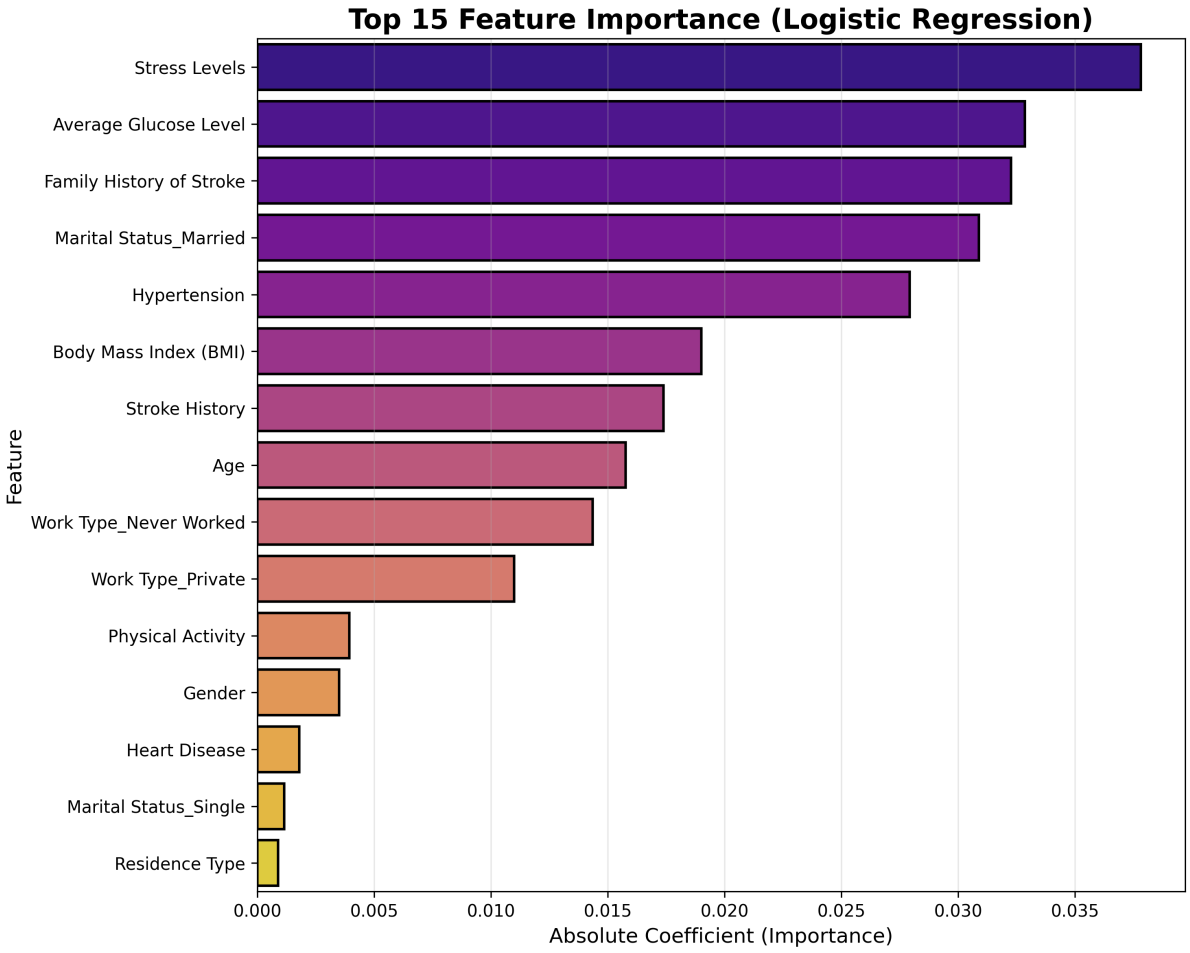
Analysis:

- Recall: 50% (missing half of stroke cases)
- Both classes equally misclassified
- No better than coin flip

Root Cause: Weak features (self-reported lifestyle data) lack predictive power

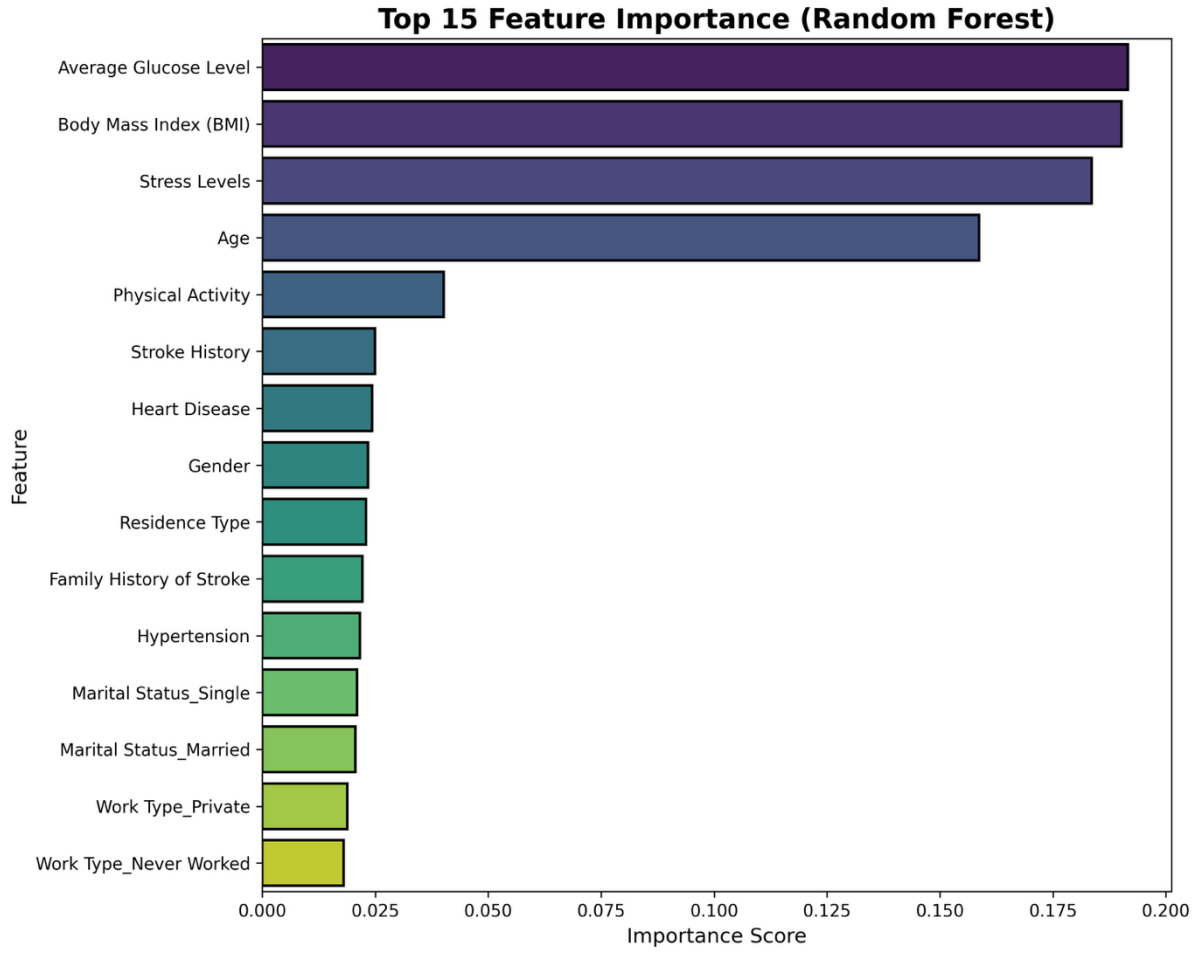
Feature Importance: Unreliable; models disagreed on top predictors

Clinical Utility: Not deployable



Logistic Regression Feature Importance

For Logistic Regression, Top three Features are Stress Levels, Average Glucose Level and Family History. However, it gives importance to Marital Status which may not be a good indicator.



Random Forest Feature Importance

Random Forest Assigns top 3 importance to Average Glucose Level, BMI and Stress Levels while some features which maybe less relevant in clinical context are assigned lower importance.

Dataset 2: Heart Disease

Results:

- **Logistic Regression:** 78% accuracy, 87% sensitivity
- **Random Forest:** 99% accuracy, 97% sensitivity, 100% specificity
- **Improvement:** +21% with Random Forest

Confusion Matrix:

- **LR:** 13 false negatives, 32 false positives

- **RF:** 3 false negatives, 0 false positives (near-perfect!)

Feature Importance (Both Models Agree):

1. cp (Chest Pain)
2. ca (Vessels Blocked)
3. oldpeak (ST Depression)
4. thalach (Max Heart Rate)
5. thal (Thallium Test)

Clinical Validation: 100% alignment with established diagnostic criteria

---

Key Findings

1. Data Quality > Algorithm Choice

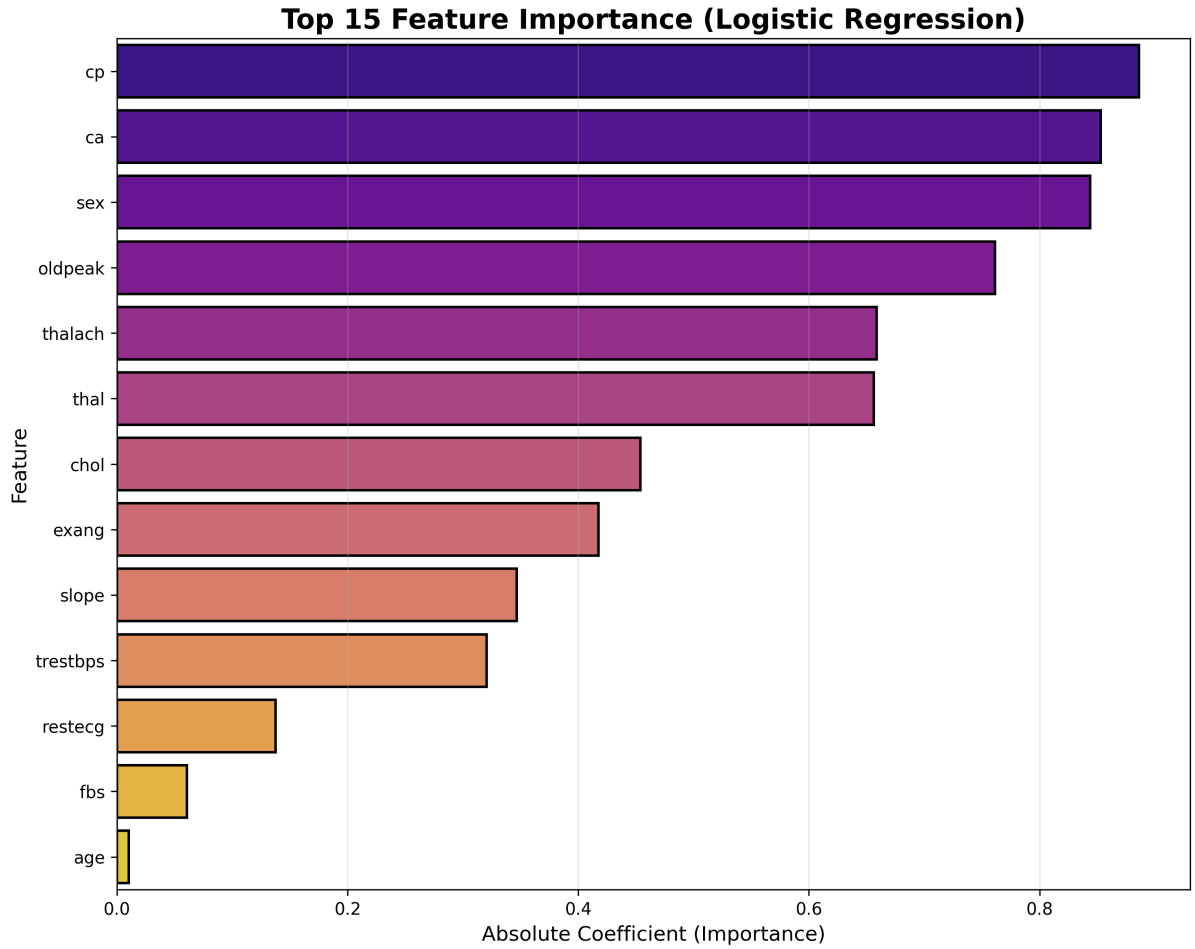
- Heart (1,025 samples, quality features) → 99%
- Stroke (15,000 samples, weak features) → 50%
- More data can't fix poor features

2. Random Forest Excels with Complex Patterns

- 21% improvement (78%→99%) proves non-linear relationships exist
- Captures threshold effects and feature interactions

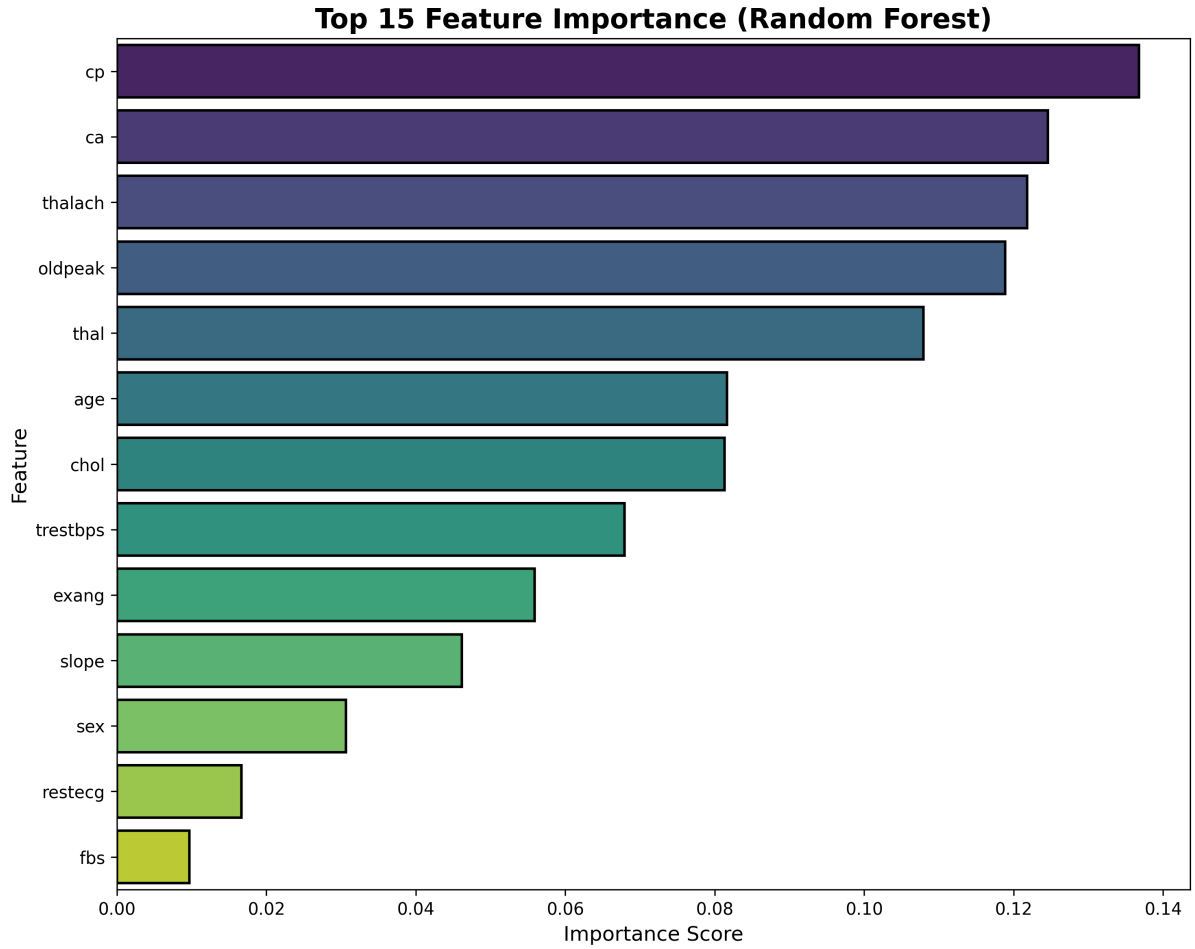
3. Feature Importance Validates Models

- Heart disease: Top features match medical literature perfectly
- Stroke: Unreliable rankings due to poor performance



Logistic Regression Feature Importance

The top 3 features are for cp, ca and sex for Logistic Regression



Random Forest Feature Importance

Top 3 features for Random Forest are also cp,ca and thalach. Since the data quality is good, Both Models Assigned Similar Feature Importance

Conclusion

Heart disease model demonstrates exceptional ML performance (99%) when features are high-quality clinical measurements. Stroke model failure (50%) illustrates that no algorithm can overcome weak features.

Key lesson: Feature quality fundamentally determines success; algorithm sophistication only matters when data permits.

Citations

1. Kaushik J. (2025). Predicting Airbnb Prices: Understanding What Drives Short-Term Rental Costs.
INFO 7390 — Art and Science of Data, Northeastern University.

2. Scikit-learn: Machine Learning in Python
Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, Édouard Duchesnay; 12(85):2825−2830, 2011.

3. Ahmad AA, Polat H. Prediction of Heart Disease Based on Machine Learning Using Jellyfish Optimization Algorithm. Diagnostics (Basel). 2023 Jul 17;13(14):2392. doi: 10.3390/diagnostics13142392. PMID: 37510136; PMCID: PMC10378171.

4. Melnykova, N., Patereha, Y., Skopivskyi, S. et al. Machine learning for stroke prediction using imbalanced data. Sci Rep 15, 33773 (2025). https://doi.org/10.1038/s41598-025-01855-w

5. Data structures for statistical computing in python, McKinney, Proceedings of the 9th Python in Science Conference, Volume 445, 2010.

6. Harris, C.R., Millman, K.J., van der Walt, S.J. et al. Array programming with NumPy. Nature 585, 357–362 (2020). DOI: 10.1038/s41586-020-2649-2. (Publisher link).

7. Waskom, M. L., (2021). seaborn: statistical data visualization. Journal of Open Source Software, 6(60), 3021, https://doi.org/10.21105/joss.03021.