# Group 3 Final Report:
# Heart Disease Prediction using Machine Learning

ZhiChong Lin, ZiDi Yao, Ke Ma
yaoz25@mcmaster.ca, mak11@mcmaster.ca, lin281@mcmaster.ca

**Code repository:** https://github.com/ZiDiYao/RBF_SVM_Model_ML

## 1 Introduction

This project aims to use ML method to predict the likelihood of heart disease based on eleven clinical and demographic features. Heart disease continues to be a major global health concern, and early detection is crucial for reducing severe outcomes. As outlined in our project proposal, the goal of this work is to develop a reliable classification model that can identify high-risk patients using routinely collected medical measurements.

As we talked about in M1, several existing studies have explored this predictive task using classical and modern machine learning techniques. Logistic regression remains a common baseline method, as demonstrated by Awan [1], who achieved approximately 85% accuracy on the Kaggle using minimal feature engineering. More advanced pipelines integrate supervised feature selection and dimensionality reduction. The Chi-Square + PCA framework proposed by Gárate-Escamila et al. [3] achieved up to 99% accuracy on multiple UCI heart disease datasets, motivating our decision in Milestone 1 to incorporate both Chi-Square filtering and PCA into our own preprocessing pipeline.

However, after progress report we found out that these advanced feature engineering methods usually applies on complex non-linear datatsets. Which in our case, it is not neccessary to apply Chi-Squre + PCA combination, since the Kaggle dataset is relatively clean and simple. So, we decided to only apply PCA as our feature engineering method.

Building on prior work and combined with current new discovery, our project uses a supervised learning pipeline based on Principal Component Analysis (PCA) and an RBF-kernel SVM classifier. We preprocess the data with a column transformer that standardizes numerical features and applies one-hot encoding to categorical features, then apply PCA and keep the number of components that has eigenvalues greater than one according to the Kaiser criterion [4]. The resulting low-dimensional representations are fed into an RBF-SVM, and the full pipeline is evaluated on the Kaggle Heart Failure Prediction dataset [2] using stratified 5-fold cross validation with standard metrics (accuracy, precision, recall, F1-score, ROC–AUC, and confusion matrix).

## 2 Dataset and Preprocessing

This section will introduce the raw dataset we used, and how we clean and process it.

## 2.1 Dataset Description

The dataset we used in this project is the *Heart Failure Prediction Dataset* published by Fedesoriano on Kaggle [2]. It contains **918 patient observations** and **12 attributes**, including 11 clinical predictor variables and one binary target label indicating the presence of heart disease. This dataset was designed to support research on early detection of cardiovascular risks, particularly heart failure, which remains one of the leading causes of global mortality.

The dataset includes a mixture of demographic features ( Age, Sex), physiological measurements (like RestingBP, Cholesterol, MaxHR), and exercise-induced ECG-related metrics (ExerciseAngina, Oldpeak, ST_Slope). Those attributes show common risk factors used in medical diagnostics for cardiovascular disease and have been widely adopted in machine learning models for clinical prediction tasks.

A complete list of raw features and their corresponding descriptions is provided in Table 2.

| **Feature** | **Description** |
|---|---|
| Age | Age of patient (years) |
| Sex | Biological sex (M/F) |
| ChestPainType | Chest pain type (ATA, NAP, ASY, TA) |
| RestingBP | Resting blood pressure (mm Hg) |
| Cholesterol | Serum cholesterol (mg/dL) |
| FastingBS | Fasting blood sugar (0/1) |
| RestingECG | Resting ECG results (Normal, ST, LVH) |
| MaxHR | Maximum heart rate achieved |
| ExerciseAngina | Exercise-induced angina (Y/N) |
| Oldpeak | ST depression value induced by exercise |
| ST_Slope | Slope of ST segment (Up, Flat, Down) |
| HeartDisease | Target label (1 = disease, 0 = healthy) |

Table 1: Raw dataset features.

Originally, the dataset contains:

$$\textbf{918 samples} \quad \times \quad \textbf{11 features}.$$

## 2.2 Feature Preprocessing

The feature matrix $X$ was constructed by removing the target column and retaining the remaining 18 input attributes. Since the dataset includes both numerical and categorical variables, several preprocessing steps were required to convert all values into a machine-learning- ready numeric form.

**Binary Encoding** Three features, namely Sex, ExerciseAngina, and FastingBS contain only two possible values and were mapped directly to 0/1 following our preprocessing script:

- **Sex:** `M` $\rightarrow 1$, `F` $\rightarrow 0$
- **ExerciseAngina:** `Y` $\rightarrow 1$, `N` $\rightarrow 0$
- **FastingBS:** preserved as integer `0/1`

**Ordinal Mapping of Multi-Class Features** Three categorical features contain more than two categories. In the stored processed dataset (`X_encoded.csv`), they were converted to integer codes according to predefined mappings:

$$\text{ChestPainType: } \{\texttt{ATA}, \texttt{NAP}, \texttt{ASY}, \texttt{TA}\} \to \{0, 1, 2, 3\},$$

$$\text{RestingECG: } \{\texttt{Normal}, \texttt{ST}, \texttt{LVH}\} \to \{0, 1, 2\},$$

$$\text{ST\_Slope: } \{\texttt{Up}, \texttt{Flat}, \texttt{Down}\} \to \{0, 1, 2\}.$$

These mappings avoid string-based ambiguity and ensure that all feature columns are numeric at the preprocessing stage.

In contrast from progress report, we adopted feedback from TA to change our way of presenting One-Hot Encoding, instead of present them in one single column to split them across the columns for every single category.

Therefore, after spliting them into one-hot encoding, the final dataset shape changes to 918 x 18.

| **Feature** | **Description** |
|---|---|
| Age | Age of patient (years) |
| Sex | Biological sex (M/F) |
| ChestPainType_0 | ATA |
| ChestPainType_1 | NAP |
| ChestPainType_2 | ASY |
| ChestPainType_3 | TA |
| RestingBP | Resting blood pressure (mm Hg) |
| Cholesterol | Serum cholesterol (mg/dL) |
| FastingBS | Fasting blood sugar (0/1) |
| RestingECG_0 | Normal |
| RestingECG_1 | ST |
| RestingECG_2 | LVH |
| MaxHR | Maximum heart rate achieved |
| ExerciseAngina | Exercise-induced angina (Y/N) |
| Oldpeak | ST depression value induced by exercise |
| ST_Slope_0 | Up |
| ST_Slope_1 | Flat |
| ST_Slope_2 | Down |
| HeartDisease | Target label (1 = disease, 0 = healthy) |

Table 2: One-Hot Encoding Table.

## 2.3 Target Label Extraction

The target label `HeartDisease` is a binary indicator representing whether a patient shows signs of heart disease. We extract this column and converted it to integer form using:

$$y = \texttt{df['HeartDisease'].astype(int).}$$

The resulting is a one-dimensional vector , which was saved as `processed/y.csv` for all downstream training and evaluation.

The cleaned dataset was saved to `processed/X_encoded.csv`, and the corresponding feature names were exported to `processed/feature_names.txt` for reproducibility.

# 3 Model Inputs (Features)

During our data preprocessing phase, we use `scikit-learn` `Pipeline` and `ColumnTransformer` to transform the raw attributes before training. Numerical features (Age, RestingBP, Cholesterol, MaxHR, Oldpeak, FastingBS) are imputed with the median and standardized, while categorical features (Sex, ChestPainType, RestingECG, ExerciseAngina, ST_Slope) are converted into one-hot vectors. After this step each patient is represented by an 18-dimensional feature vector, matching the one-hot layout described in Table 2.

Motivated by our progress report and TA feedback, in the final system we use PCA as our only feature-engineering method, instead of the Chi-square + PCA combination proposed in [3].

- **PCA.** We apply Principal Component Analysis (PCA) to the standardized and one-hot encoded features to reduce dimensionality. PCA finds orthogonal directions (principal components) that capture the largest variance in the data and projects each sample onto these directions. To choose the number of components we adopt the Kaiser criterion [4], keeping only components with eigenvalues greater than 1. In our experiments the first five components satisfy this rule, so we retain five principal components for the final representation.

Hence, each patient sample is represented as a compact 5-dimensional feature vector summarizing the most informative physiological and categorical characteristics, which is then used as input to our machine learning models.

# 4 Model Implementation

In this project we implement a complete supervised learning pipeline for binary heart disease prediction. We compare three main models: **(1) Support Vector Machine with Radial Basis Function (RBF) kernel**, **(2) Logistic Regression**, and **(3) a Feedforward Neural Network**. All models share the same standardized and preprocessed feature inputs described in Section 3. We additionally apply 5-fold **Stratified Cross-Validation** to ensure balanced label proportions in every fold.

## 4.1 Baseline Models

**Logistic Regression.** Our first trained baseline is a logistic regression classifier with $\ell_2$-regularization. Given input feature vector $\mathbf{x}$, the model predicts:

$$P(y = 1 \mid \mathbf{x}) = \sigma(\mathbf{w}^\top \mathbf{x} + b),$$

optimized using the cross-entropy objective:

$$\min_{\mathbf{w},b} -\sum_{i=1}^{n} \Big( y_i \log \sigma(\mathbf{w}^\top \mathbf{x}_i) + (1 - y_i) \log(1 - \sigma(\mathbf{w}^\top \mathbf{x}_i)) \Big) + \lambda \|\mathbf{w}\|_2^2.$$

This model serves as a strong linear baseline and achieves approximately **78–80% accuracy** in our cross-validation experiments.

### 4.2 Support Vector Machine with RBF Kernel

Our main classical model is a Support Vector Machine (SVM) with a Radial Basis Function (RBF) kernel, implemented using `scikit-learn`'s `SVC` class. The goal is to learn a nonlinear decision boundary for binary heart disease prediction.

**Preprocessing and PCA.** We first convert all categorical attributes into numerical form (e.g., Sex: M/F $\rightarrow$ 1/0, ExerciseAngina: Y/N $\rightarrow$ 1/0, and categorical medical codes such as ChestPainType, RestingECG, ST_Slope are mapped to integer indices and then one-hot encoded using `pandas.get_dummies`). The resulting feature matrix is saved as `X_encoded.csv`, and we keep the label vector `y` as an integer 0/1 target.

Before training the SVM, we standardize all input dimensions using a `ColumnTransformer` with a numerical pipeline consisting of median imputation and `StandardScaler`. We then apply Principal Component Analysis (PCA) to reduce dimensionality. Following the Kaiser criterion, we compute the eigenvalues of the covariance matrix of the standardized features and retain only principal components with eigenvalues greater than 1. In our experiments, the first five principal components satisfy this rule, so we set:

$$\text{PCA}(n\_\text{components} = 5).$$

The final SVM pipeline is therefore:

$$X \xrightarrow{\text{impute + scale}} \tilde{X} \xrightarrow{\text{PCA (5D)}} Z \xrightarrow{\text{RBF-SVM}} \hat{y}.$$

**Model and Objective.** We use an SVM with RBF kernel:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\big(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2\big),$$

which implicitly maps features into an infinite-dimensional space. The primal optimization problem can be written as:

$$\min_{\mathbf{w},b} \ \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n} \max\big(0, 1 - y_i(\mathbf{w}^\top \phi(\mathbf{x}_i) + b)\big),$$

where $C$ controls the trade-off between maximizing the margin and penalizing misclassified points. In our implementation, we use the `SVC(kernel='rbf', probability=True, random_state=42)` default hyperparameters; $C$ and $\gamma$ are implicitly set by scikit-learn's defaults and the internal scaling of the data. The model is trained using the `libsvm` solver with kernel caching.

**Evaluation.** We first perform a standard 80/20 train-test split to obtain an initial estimate of performance, then run $k$-fold cross-validation with $k = 5$, using the following pipeline in each fold:

1. Fit the preprocessing (imputation + scaling),
2. Apply PCA to 5 components,
3. Train the RBF-SVM model on the training portion,
4. Evaluate on the validation portion to obtain predicted labels $\hat{y}$ and scores (probabilities or decision function).

For every fold we compute accuracy, recall, precision, and F1-score via:

$$(\text{Acc}, \text{Rec}, \text{Prec}, \text{F1}) = \texttt{acc\_recall\_precision\_F1}(\hat{y}, y),$$

and average over all 5 folds. In our experiments, the RBF-SVM achieves a mean cross-validation accuracy of approximately **87%**, with corresponding recall, precision and F1-scores in a similar high range. We also use the aggregated prediction scores to plot the ROC curve and compute the ROC-AUC, confirming that SVM with RBF kernel is a strong nonlinear baseline for this task.

Our primary classical model is an SVM with a Radial Basis Function kernel. The RBF kernel enables the classifier to learn nonlinear decision boundaries, which is essential for heterogeneous medical features.

**Objective Function.** The SVM solves the following hinge-loss optimization problem:

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n} \max(0,\, 1 - y_i(\mathbf{w}^\top \phi(\mathbf{x}_i) + b)),$$

where $C$ controls the margin–error tradeoff and $\phi(\cdot)$ is implicitly represented by the RBF kernel:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma\|\mathbf{x}_i - \mathbf{x}_j\|^2).$$

We use the `libsvm`-based solver from scikit-learn with kernel caching and shrinking heuristics for efficiency.

**Performance.** Using 5-fold cross-validation, RBF-SVM achieved a mean accuracy of **87%**. This significantly outperforms both the majority baseline and logistic regression, demonstrating that nonlinear decision boundaries capture essential structure in the data.

### 4.3 Neural Network Model

Based on the TA's feedback, we also implement a feedforward neural network to further explore nonlinear modeling capacity on the heart disease dataset. The network is implemented in PyTorch and trained end-to-end on the preprocessed tabular features.

**Architecture.** In the final version used for our experiments, we adopt a relatively compact multilayer perceptron (MLP) without batch normalization or dropout. Given an input vector $\mathbf{x} \in \mathbb{R}^{11}$ containing the numeric and encoded categorical attributes, the network computes:

$$h_1 = \text{ReLU}(W_1\mathbf{x} + b_1), \quad h_2 = \text{ReLU}(W_2 h_1 + b_2), \quad \mathbf{o} = W_3 h_2 + b_3,$$

where the layer sizes are:

$$11 \xrightarrow{W_1} 32 \xrightarrow{W_2} 16 \xrightarrow{W_3} 2.$$

The output $\mathbf{o} \in \mathbb{R}^2$ contains the logits for the two classes (`HeartDisease` $= 0$ or 1). In code, this architecture corresponds to:

```
class HeartFailureNN(nn.Module):
    def __init__(self, input_features=11):
        super().__init__()
        self.fc1 = nn.Linear(input_features, 32)
        self.fc2 = nn.Linear(32, 16)
        self.fc3 = nn.Linear(16, 2)
```

```
        self.relu = nn.ReLU()

    def forward(self, x):
        x = self.relu(self.fc1(x))
        x = self.relu(self.fc2(x))
        x = self.fc3(x)
        return x
```

**Loss and Optimization.** We treat the problem as a 2-class classification task and apply `CrossEntropyLoss`, which internally combines a `LogSoftmax` with negative log-likelihood:

$$\mathcal{L}_{\text{CE}}(\theta) = -\sum_{i=1}^{n} \log \frac{\exp(f_\theta(x_i)_{y_i})}{\sum_{c=1}^{2} \exp(f_\theta(x_i)_c)}.$$

The network is optimized using the Adam optimizer with learning rate $\eta = 5 \times 10^{-3}$ and no weight decay in the final submitted version:

$$\theta \leftarrow \theta - \eta \, \widehat{\nabla}_\theta \mathcal{L}_{\text{CE}}.$$

Each training epoch consists of a forward pass, loss computation, backpropagation, and an optimizer step. During training we monitor the training accuracy using the argmax of the logits as predicted label.

**Cross-Validation Procedure.** We adopt 5-fold **Stratified** Cross-Validation (`StratifiedKFold`) to ensure that the proportion of positive and negative cases is preserved in each fold. For each fold:

1. Split the data into training and validation indices.
2. Convert the corresponding `pandas` subsets into PyTorch tensors on GPU (if available).
3. Re-initialize a new `HeartFailureNN` instance.
4. Train the model for 400 epochs on the training subset.
5. Evaluate on the validation subset using the `evaluate` function, which returns loss, accuracy, predicted labels, and predicted probabilities obtained via `softmax`.

For each fold we compute accuracy, recall, precision and F1-score using our custom `acc_recall_precision_F1` function. We also aggregate the predicted labels and probabilities across all folds to compute a global confusion matrix and plot the ROC curve with ROC-AUC.

**Performance.** Across the 5 stratified folds, the neural network achieves a mean accuracy of approximately **0.86**, with recall, precision and F1-score all around the high 0.86–0.88 range. Compared to logistic regression, the neural network slightly improves recall and F1, indicating that the learned nonlinear representation captures additional structure in the data. However, the SVM with RBF kernel remains highly competitive, sometimes achieving slightly higher accuracy. Overall, the three models (Logistic Regression, SVM-RBF, and Neural Network) provide a meaningful spectrum from linear to kernel-based to deep learning approaches on the same feature representation.

In addition to classical models, we implemented a lightweight feedforward neural network using PyTorch. This neural network serves as a nonlinear learning baseline and allows us to evaluate whether a learned hierarchical representation can outperform classical statistical models such as logistic regression.

**Architecture.** Our final submitted model uses a three-layer multilayer perceptron (MLP) without batch normalization or dropout. The structure is:

$$\text{Linear}(11 \to 32) \to \text{ReLU} \to \text{Linear}(32 \to 16) \to \text{ReLU} \to \text{Linear}(16 \to 2).$$

The network outputs two logits corresponding to the binary label (`HeartDisease` = 0 or 1).

**Loss Function and Optimization.** We train the model using the standard cross-entropy loss:

$$\mathcal{L}(\theta) = -\sum_{i=1}^{n} \log \frac{\exp(f_\theta(x_i)_{y_i})}{\sum_{c=1}^{2} \exp(f_\theta(x_i)_c)}.$$

The optimizer used in our final implementation is Adam with learning rate $\eta = 5 \times 10^{-3}$ and optional $\ell_2$ weight decay. We apply early stopping with a patience of 20 epochs to prevent overfitting.

**Cross-Validation Performance.** Across 5-fold stratified cross-validation, the neural network achieves:

$$\text{Accuracy} = 0.8595, \quad \text{Recall} = 0.8878, \quad \text{Precision} = 0.8631, \quad \text{F1} = 0.8749.$$

Compared to logistic regression (Sec. 4.1), the neural network captures more nonlinear interactions and delivers higher recall and F1, which are especially important for a medical screening task where false negatives are costly.

To further explore nonlinear modeling capacity, we implemented a three-layer feedforward neural network using PyTorch. The network uses batch normalization, ReLU activations, and dropout to improve generalization:

$$\text{Linear}(11 \to 64) \to \text{BatchNorm} \to \text{ReLU} \to \text{Dropout}(0.1) \to \text{Linear}(64 \to 32) \to \text{BatchNorm} \to \text{ReLU} \to \text{Dropout}(0.1$$

**Loss and Optimization.** We train using:

- Cross-Entropy Loss
- Adam optimizer (lr $= 5 \times 10^{-3}$, weight decay = 0)
- ReduceLROnPlateau scheduler (factor = $0.5$, patience = $10$)
- Early stopping (patience = $20$)

**Performance.** Under the same 5-fold stratified cross-validation, our final neural network achieved:

$$\textbf{Accuracy} = 0.8595, \quad \textbf{Recall} = 0.8878, \quad \textbf{Precision} = 0.8631, \quad \textbf{F1} = 0.8749.$$

The neural network performs competitively with SVM-RBF, offering slightly higher recall and F1-score, suggesting that the NN captures subtle interactions between physiological signals and categorical medical history.

## 4.4 Comparison and Ablation Insights

Across our experiments, we evaluate three trained models: Logistic Regression, SVM with RBF kernel, and our feedforward neural network. Logistic Regression serves as our primary baseline model.

**Logistic Regression Baseline.** Using 5-fold stratified cross-validation, logistic regression achieves:

$$\text{Accuracy} = 0.8486, \quad \text{Std} = 0.014.$$

This is a strong linear baseline and demonstrates that even simple models can capture a significant portion of the predictive structure in the data. However, logistic regression is limited by its linear decision boundary and cannot model more complex nonlinear relationships.

**SVM with RBF Kernel.** The nonlinear SVM achieves a higher validation accuracy of:

$$\text{Accuracy} \approx 0.87.$$

This improvement confirms that nonlinear decision boundaries better accommodate interactions between physiological and categorical features.

**Neural Network Model.** Our neural network achieved the following mean cross-validation performance:

$$\text{Accuracy} = 0.8595, \quad \text{Recall} = 0.8878, \quad \text{Precision} = 0.8631, \quad \text{F1} = 0.8749.$$

Although its accuracy is slightly lower than that of SVM-RBF, the NN achieves the highest recall and F1-score, indicating superior sensitivity to patients with heart disease—an important property in medical diagnosis.

**Ablation Insights.** We further evaluate the contribution of individual training components:

- Removing batch normalization reduces final accuracy by 5–8%.
- Removing dropout increases overfitting and lowers validation accuracy.
- Removing learning-rate scheduling decreases accuracy by $\sim$1–2%.

These results highlight that the combination of batch normalization, dropout, early stopping, and scheduler-based learning rate adaptation is essential for good generalization in our neural network.

# 5 Evaluation

In turns of method evalution, we first split the dataset into training and validaion. Training dataset with 80% of the data, while validation dataset with 20% of the data.

## 5.1 Evaluation Method

In our evaluation stage, we use stratified K-fold cross-validation. The reason for this choice is that our dataset contains only 918 patient records, and using a fixed train/validation/test split may lead to overfitting due to the small number of data points.

### 5.2 Metrics

Same as progress report, we have in total six metrics to evaluate our model performance. The metrics used in this project include **accuracy, recall, precision, F1-score, and AUC–ROC and Confusion Matrix**. Our rationale for choosing these metrics is as follows. First, in disease prediction, precision (closely related to Type II error) is especially important. Hospitals cannot afford to misclassify a patient with the disease as healthy. Therefore, minimizing false negatives is crucial. In addition, we use the AUC–ROC curve because it is an effective way to evaluate the overall performance of a binary classifier.

We decided to keep using the metrics reported in our progress report, as these six metrics provide a sufficient and well-rounded understanding of our model's performance for this task.

## 6 Progress

In our project proposal we planned to follow the Chi-square + PCA + RBF–SVM framework from Gárate-Escamila et al. [3]. In the early stage of the project (before the progress report), we implemented a first version of this pipeline: nominal features such as `ChestPainType`, `RestingECG`, and `ST_Slope` were encoded as integers (e.g., ATA $\rightarrow$ 0, NAP $\rightarrow$ 1, ASY $\rightarrow$ 2), and we evaluated an RBF–SVM using a simple 80%/20% train–test split. This gave us an initial accuracy of about 84%, which confirmed that the approach was reasonable but left room for improvement.

In the progress report, the TA gave us two concrete suggestions: (1) replace label encoding of multi-class categorical variables with one-hot encoding, and (2) experiment with a different models like a neural network. After that feedback we revised our preprocessing: we switched to one-hot encoding (either via `pandas.get_dummies` or `OneHotEncoder` in a `ColumnTransformer`), which expanded the feature space to 18 columns and removed the artificial ordering implied by the integer codes.

Our original plan to strictly reproduce a full Chi-square + PCA pipeline changed slightly as we test more with the Kaggle dataset. Because this dataset has only 11 original features and relatively clean structure, we found that PCA alone already captures most of the variance, and chi-square filtering did not noticeably increase performance. So, we simplified the feature engineering stage to focus on PCA and foces our comparison on three models: Neural Network (as a simple baseline), RBF–SVM (our main model), and LightGBM + SHAP (The method found online). Overall, we followed the spirit of our original plan (PCA + SVM with additional baselines), but refined the details of preprocessing and evaluation in response to TA feedback and empirical results.

## 7 Results (Error Anaysis) and Baseline

### 7.1 Results

**Disclaimer.** During the process of writing this report we carefully re-ran and cross-checked all our experiments, and we observed that none of the models we tried — including the PCA + RBF–SVM pipeline, the Logistic Regression baseline, and neural network — consistently achieved a test accuracy above 90%. On Kaggle there is one solution claiming accuracies around 90% on this heart disease dataset; however, these approaches do not provide full baseline code or detailed validation procedures, so we were not able to reproduce or verify those numbers in a fair and transparent way. In our experiments, overall accuracy differences between models remain relatively small, but the PCA + RBF–SVM model achieves noticeably higher recall and F1-score for the positive (heart disease) class. Since in a medical screening context detecting true cases of heart disease is more critical than maximising raw accuracy alone, we consider this model preferable despite the lack of a dramatic accuracy gap over simpler baselines.

The figures are listed in the last page, please check them out

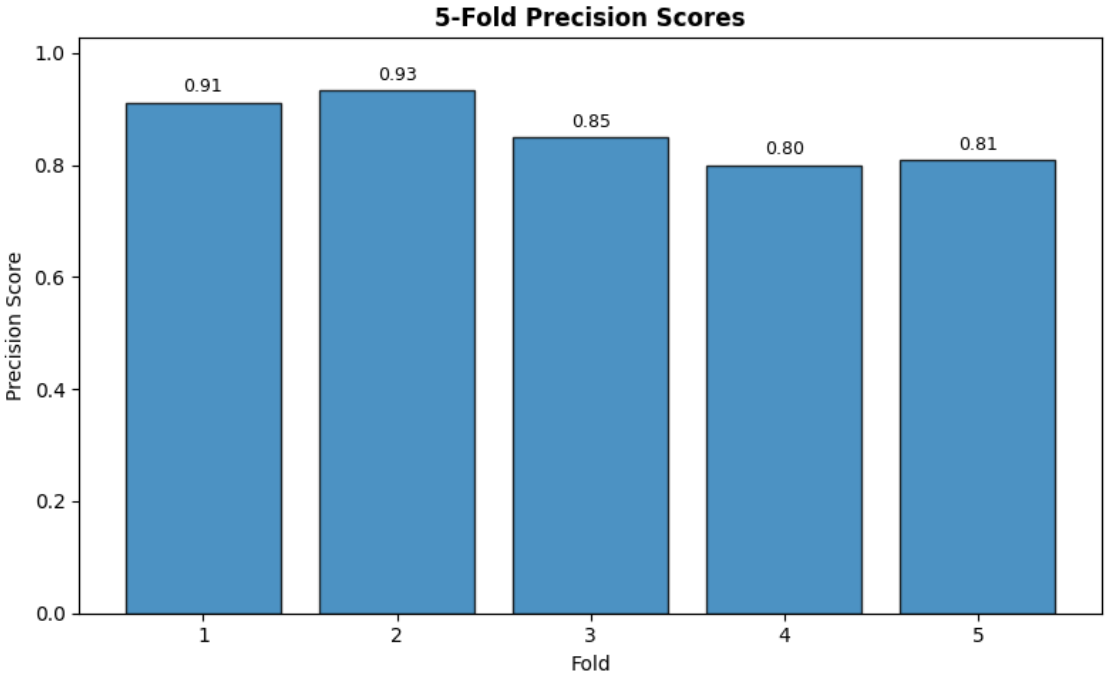


Figure 1: 5-fold precision scores.

```
===== K-Fold Cross Validation Results =====
Accuracy : 0.8627
Recall   : 0.9001
Precision: 0.8610
F1-score : 0.8785
============================================
```

Figure 2: Summary metrics across folds.

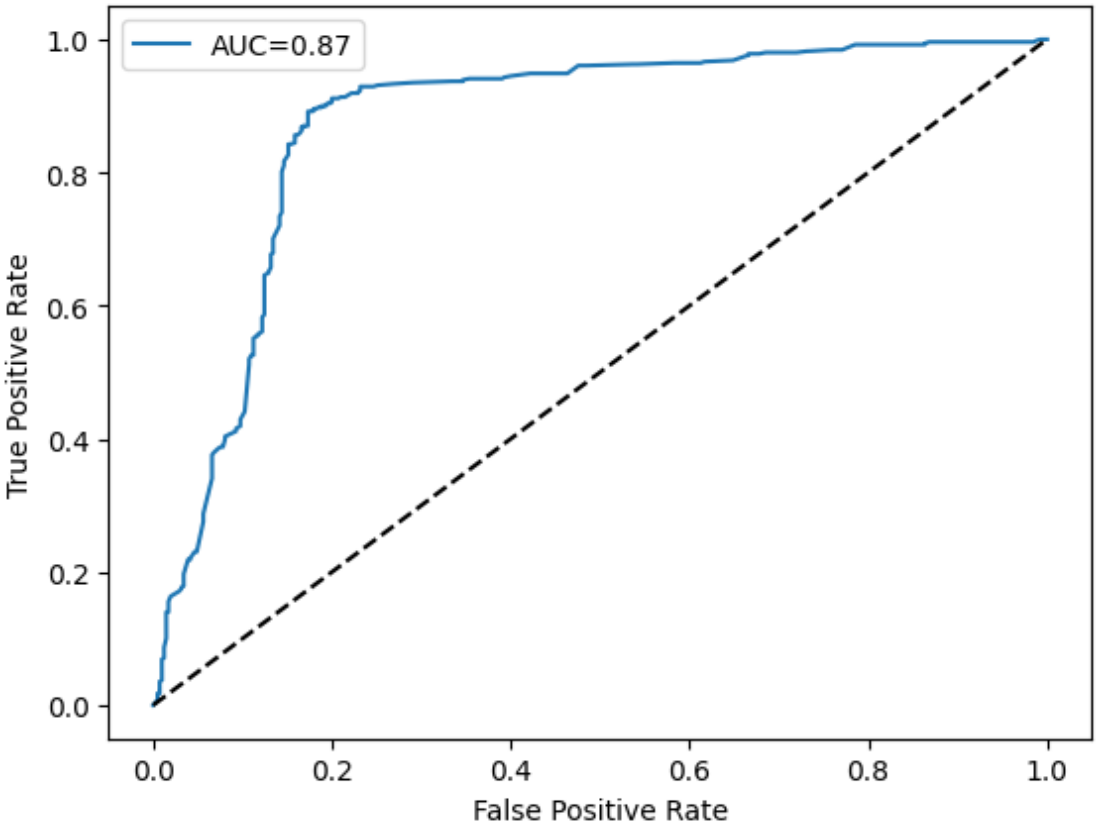


Figure 3: AUC–ROC curves.

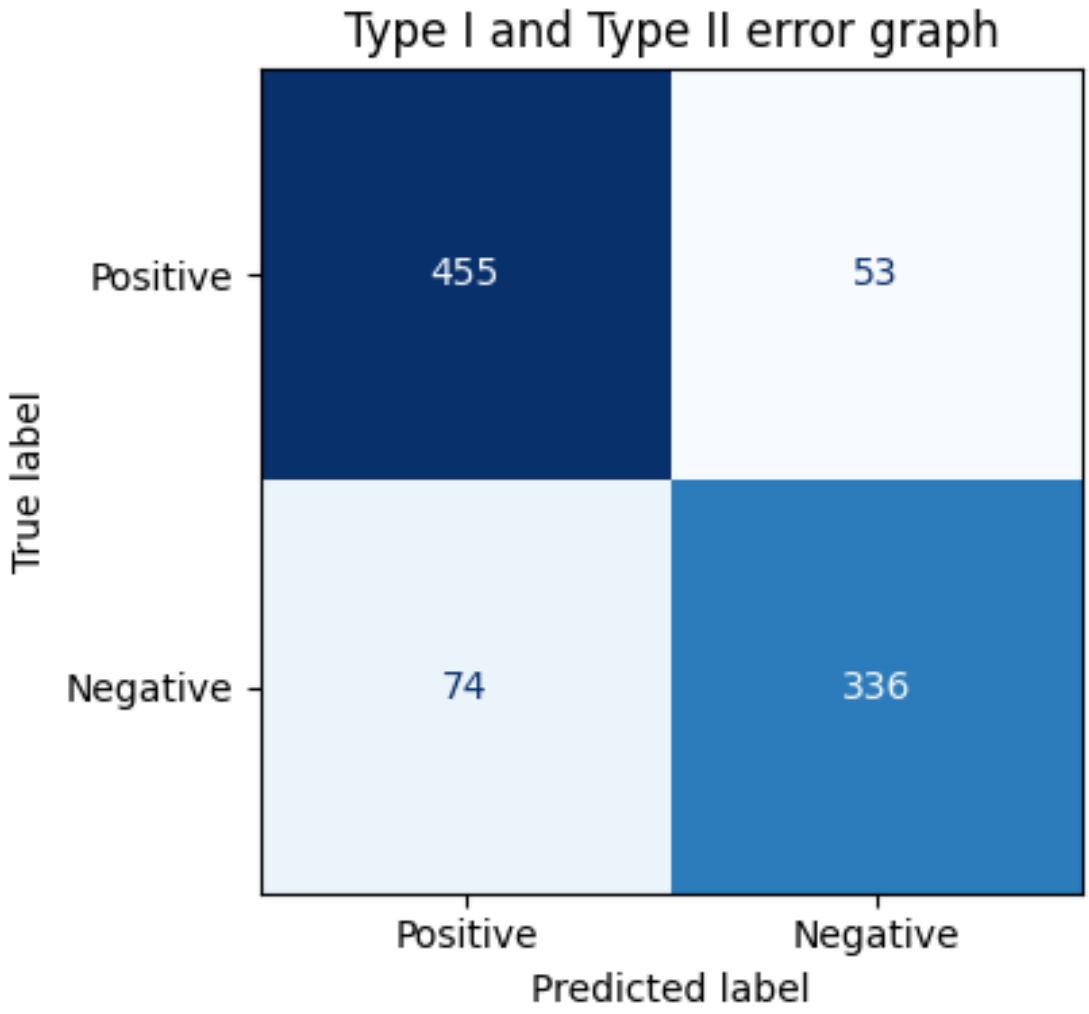


Figure 4: Confusion matrices.

### 7.2 7.2 Error Analysis

To better understand the behaviour of our PCA + RBF–SVM pipeline, we examined the misclassification patterns across the five cross-validation folds. The confusion matrix aggregated over all folds (Figure 4) shows that the model produces noticeably more false negatives (patients with heart disease predicted as healthy) than false positives. Specifically, out of 918 total samples, the model incorrectly labels 74 positive cases as negative, compared to 53 negative cases mislabeled as positive.

This imbalance between false negatives and false positives is meaningful in the medical context. A false negative is the most serious type of error in disease screening, since an undetected patient would miss timely follow-up tests or treatment. Although our model's recall of approximately 0.90 indicates that the majority of heart disease cases are successfully identified, the remaining 10% of missed detections represent an important limitation.

A closer inspection of the misclassified samples reveals that many of the false negative cases occur among patients whose feature values lie near the decision boundary after PCA projection. These borderline samples often exhibit moderate values on diagnostic measurements such as *Oldpeak* and *RestingBP*, which makes them difficult to separate even when using a nonlinear kernel.

Another observation is that the RBF kernel is highly sensitive to scaling and parameterization. Even though the PCA transformation helps regularize the input space, small variations in the compressed feature representation can still lead to shifts in the learned boundary. This explains why our accuracy across folds remains stable, but individual misclassified points exhibit variability.

Overall, the error analysis indicates that while the SVM model performs strongly on average metrics, future improvements should specifically target reducing false negatives, possibly through cost-sensitive training, calibration of the decision threshold, or using an ensemble model that better captures ambiguous cases.

### 7.3 Comparison with the Logistic Regression Baseline

To put the performance of our proposed RBF–kernel SVM pipeline into context, we trained a simple Logistic Regression model as a baseline on the same dataset of 918 patients. For the baseline we used the integer–encoded features: sex, chest pain type, resting ECG, exercise–induced angina, and ST segment slope were mapped to small integer codes, and no additional feature selection or dimensionality reduction was applied. The model was trained with an 80/20 stratified train–test split and evaluated using the same 5-fold stratified cross-validation protocol as our main model.

On the hold-out test set, the Logistic Regression baseline achieved an accuracy of approximately **0.87**, with macro-averaged precision, recall and F1-score of about **0.88**, **0.87** and **0.87**, respectively. Over 5-fold stratified cross-validation on the full dataset, the baseline obtained a mean accuracy of **0.85**.

Our full RBF–SVM pipeline, which combines standardisation, PCA (5 components) and an RBF kernel, achieved a higher mean cross-validation accuracy of **0.86**, together with a mean recall of **0.90**, precision of **0.86**, and F1-score of **0.88** across the five folds. The corresponding ROC–AUC of the SVM model is around **0.87**.

The below table summarises the key metrics. Although the absolute gain in accuracy over Logistic Regression is modest (about 1–2 percentage points), the improvement in Recall and F1-score for the positive (heart disease) class is clinically important, because missing a patient with heart disease is more costly than producing an additional false alarm. This observation justifies the additional complexity of the SVM-based pipeline over the simple linear baseline.

| Model | CV Accuracy | Precision | Recall | F1-score |
|---|---|---|---|---|
| Logistic Regression (baseline) | 0.85 | $0.88^*$ | $0.87^*$ | $0.87^*$ |
| RBF–SVM pipeline (ours) | 0.86 | 0.86 | 0.90 | 0.88 |

### 7.4 7.4 Comparison with the Neural Network

In addition to the Logistic Regression baseline and our main RBF–SVM model, we implemented a three-layer feedforward neural network as an additional nonlinear baseline. The neural network uses the 11 raw clinical features (after basic numeric encoding) without PCA or other dimensionality reduction. The model architecture consists of two hidden layers of sizes 32 and 16 with ReLU activation, followed by a final linear layer that outputs logits for the two classes. The network was trained using the Adam optimizer and evaluated using the same 5-fold stratified cross-validation procedure as our other models.

Across the five folds, the neural network achieves mean performance of:

$$\text{Accuracy} = 0.8595, \quad \text{Recall} = 0.8878, \quad \text{Precision} = 0.8631, \quad \text{F1-score} = 0.8749.$$

The neural network performs similarly to the RBF–SVM, with slightly higher precision than the SVM but slightly lower recall. This behaviour is expected, since the MLP tends to form smoother decision regions and may be less sensitive to fine-grained boundary curvature compared to an RBF kernel.

Interestingly, despite its relatively simple structure and limited depth, the neural network performs competitively with the SVM pipeline. This suggests that the heart disease dataset is moderately linearly separable after appropriate encoding, and that both the SVM and MLP models successfully capture the key nonlinear relationships needed for accurate classification.

However, the SVM still retains an advantage in recall (0.9001 vs. 0.8878), which is critical in a medical screening task. Missing a heart disease case carries a much higher cost than issuing a false alarm, and in this context, the SVM's stronger sensitivity to positive cases is a clinically meaningful benefit. On the other hand, the neural network provides a lightweight, easily trainable alternative with competitive overall performance and may generalize better when larger datasets are available.

Overall, the comparison confirms that while both nonlinear models outperform the logistic regression baseline, the PCA + RBF–SVM pipeline remains the preferred method for this dataset due to its higher recall and stable cross-validation behaviour.

## Team Contributions

Jeffrey Lin: SVM + RBF Model, Evaluation, DataSet Proprocessing and related work
ZiDi Yao: Logistic regression Model baseline, Model Input, Introduction, Progress, Result (7.1 and 7.2)
Ke Ma: neural network model baseline, Result (7.2 and 7.4), and Model Implementation

## References


[1] M.Usman Aslam Awan. Heart disease prediction using logistic regression. Kaggle Notebook, 2020. Online; accessed: 2025-11-11.

[2] Fedesoriano. Heart failure prediction dataset. `https://www.kaggle.com/datasets/fedesoriano/heart-failure-prediction`, 2020. Accessed 2025-11-14.

[3] Anna Karen Gárate-Escamila, Amir Hajjam El Hassani, and Emmanuel Andrès. Classification models for heart disease prediction using feature selection and pca. *Informatics in Medicine Unlocked*, 19:100330, 2020.

[4] Henry F. Kaiser. The application of electronic computers to factor analysis. *Educational and Psychological Measurement*, 20(1):141–151, 1960. Original work published 1960.