Part II:

Machine Learning on Heart Disease Classification

Agenda

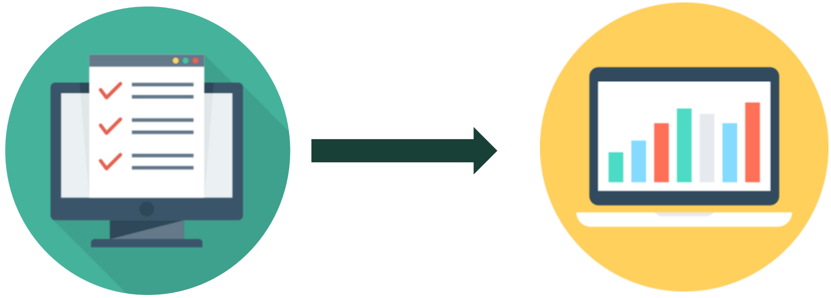
01 Project Objective & Flow

02 Data Preprocessing

03 Model Implementation

04 Comparisons of Optimized Models

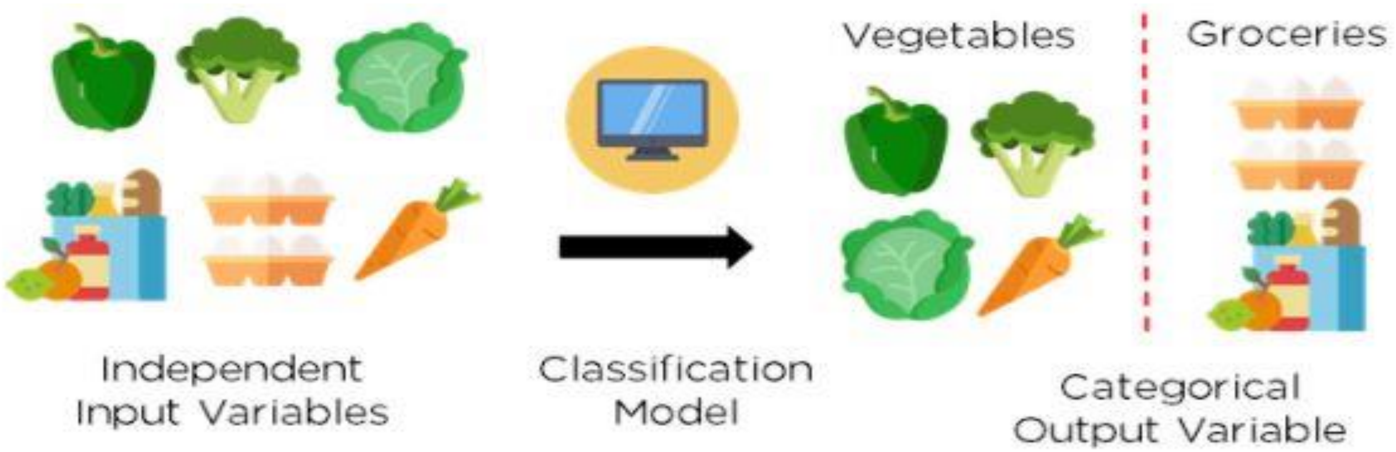
05 Conclusions and Future Work



Project Objective

<u>Supervised Learning</u>

Binary Classification of Heart Disease
(**Class 1: Have heart disease**, Class 0: No heart disease)



Project Flow

Data Preprocessing	01 Data Exploration	02 Feature Selection	03 Feature Scaling	04 Dataset Splitting
Model Training	05 Approach Comparison	06 Hyperparameter Tuning	07 Model Training	08 Performance Evaluation
Model Retraining	09 Computing Permutation Importance	10 Step 06 - Step 08	11 Model Comparison	

Dataset

Categorical

- Binary
  - Target
  - Sex
  - Exercise-induced angina
  - Fasting blood sugar (> 120 mg/dL)
- Nominal
  - Chest pain type
  - Resting electrocardiogram results
  - Slope of the peak exercise ST segment

Numeric

- Age
- Resting blood pressure
- Serum cholesterol
- Maximum heart rate
- Oldpeak

Data Exploration

Overview Alerts 3 Reproduction

Dataset statistics

Number of variables	12
Number of observations	1048
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	98.4 KiB
Average record size in memory	96.1 B

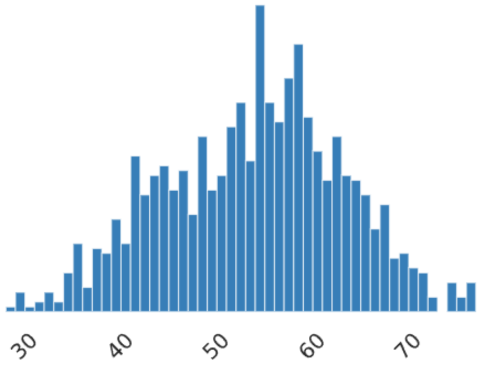
Variable types

Numeric	5
Categorical	7

Data Exploration

1. Mean and median (50%) of all columns, except oldpeak, are almost the same, meaning that the data is symmetrically distributed.

2. The distribution of age follows a normal distribution pattern.



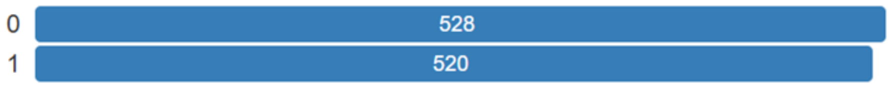
	age	resting bps	cholesterol	max heart rate	oldpeak
count	1048.000000	1048.000000	1048.000000	1048.000000	1048.000000
mean	53.325382	132.613550	245.172710	142.918893	0.942366
std	9.397822	17.367605	57.101359	24.427115	1.100429
min	28.000000	92.000000	85.000000	69.000000	-0.100000
25%	46.000000	120.000000	208.000000	125.000000	0.000000
50%	54.000000	130.000000	239.000000	144.000000	0.600000
75%	60.000000	140.000000	275.000000	162.000000	1.600000
max	77.000000	200.000000	603.000000	202.000000	6.200000

target

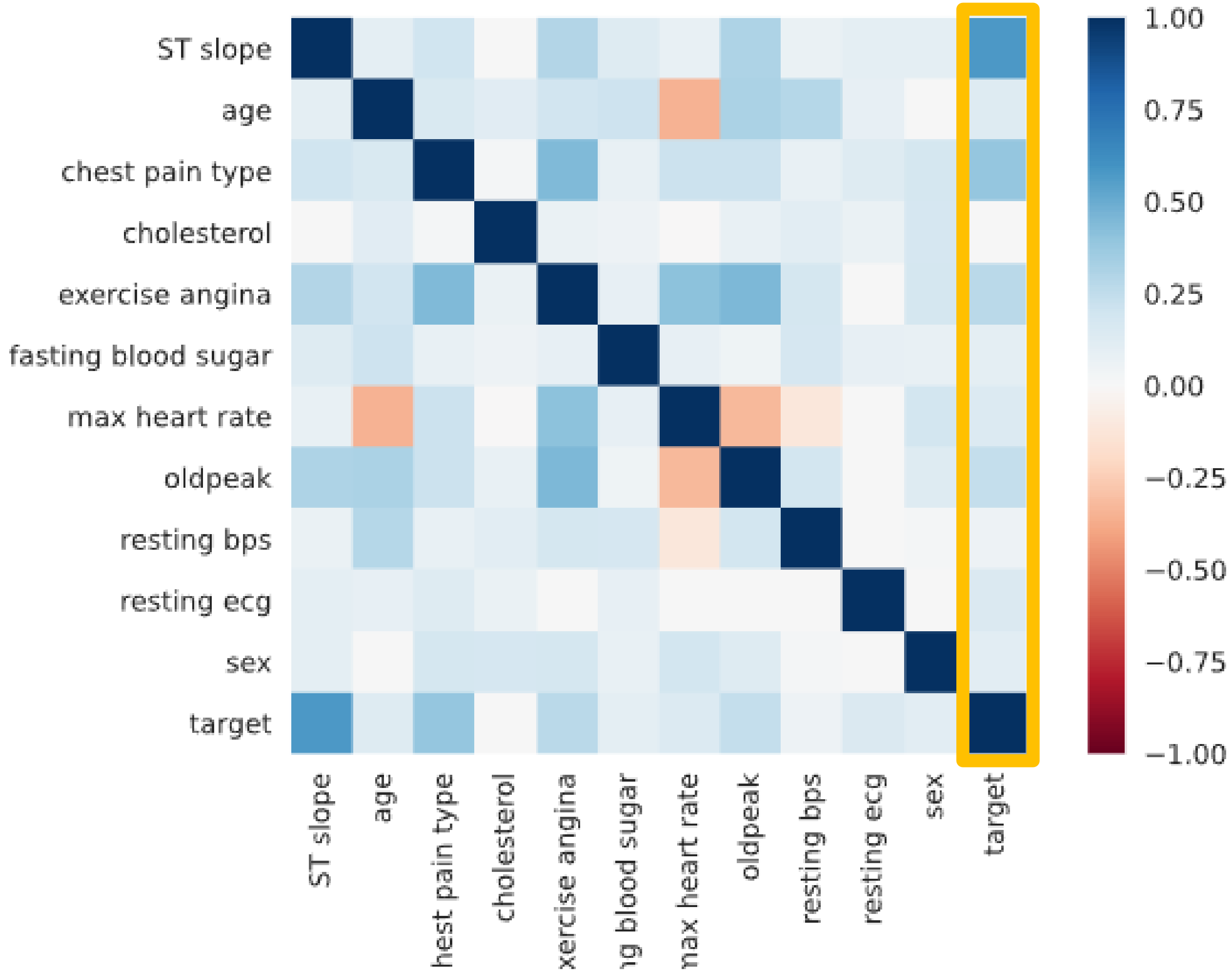
Categorical

HIGH CORRELATION

Distinct	2
Distinct (%)	0.2%
Missing	0
Missing (%)	0.0%
Memory size	8.3 KiB

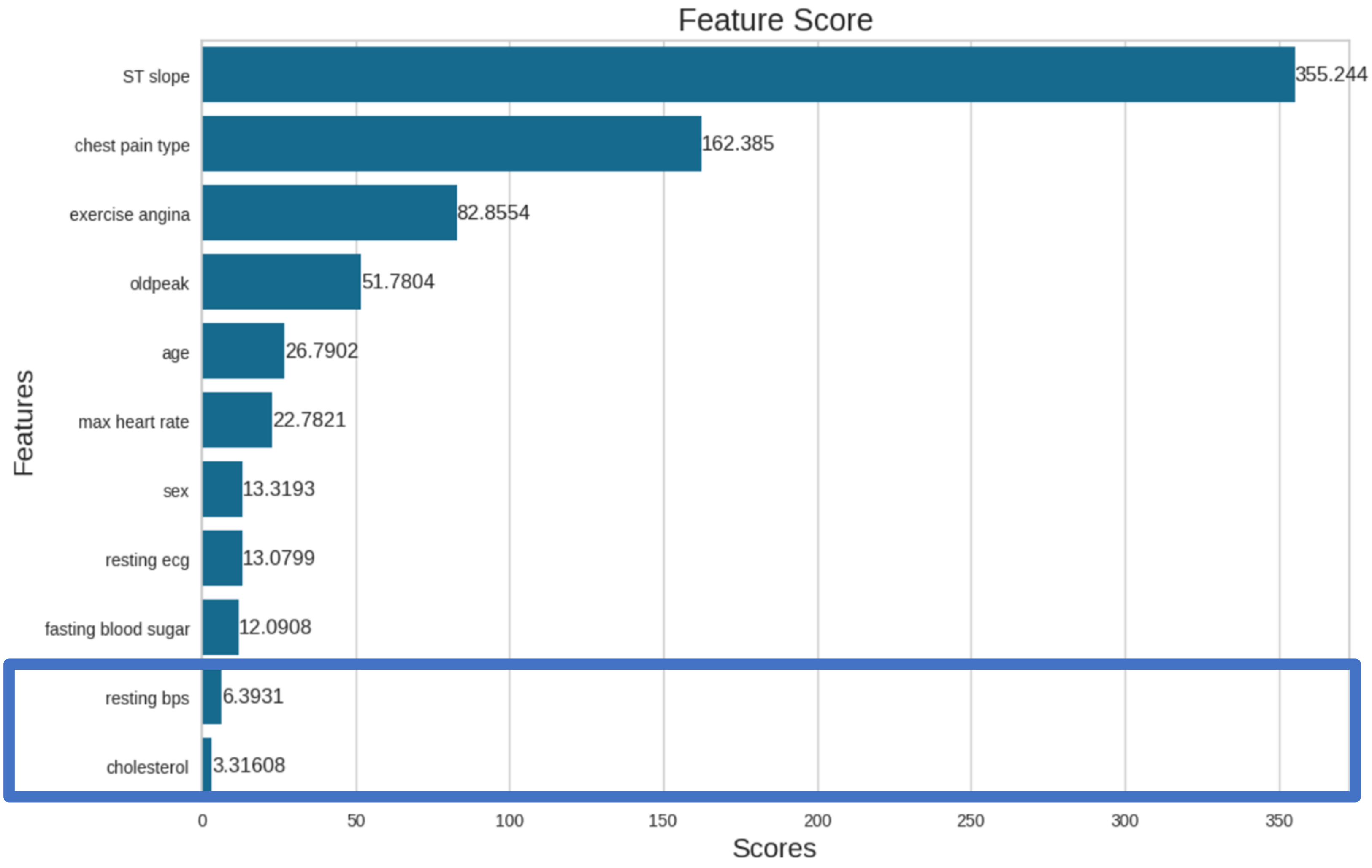


Data Exploration



	target
ST slope	0.574
age	0.132
chest pain type	0.390
cholesterol	0.000
exercise angina	0.267
fasting blood sugar	0.100
max heart rate	0.138
oldpeak	0.241
resting bps	0.052
resting ecg	0.147
sex	0.106
target	1.000

Feature Selection, Scaling and Dataset Splitting



1. **Drop the low scored features**
   score < 10
   (resting bps and cholesterol)

2. **Standardize features using**
   Standard Scaler
   mean to 0
   standard deviation to 1

3. **Split train and test dataset**
   train set : 80%
   test set : 20%

Model Building

01
Model with
Default Settings

02
Model with
Manual Tuning

03
Model with “Best” hyperparameters
(Tuned by GridSearchCV)

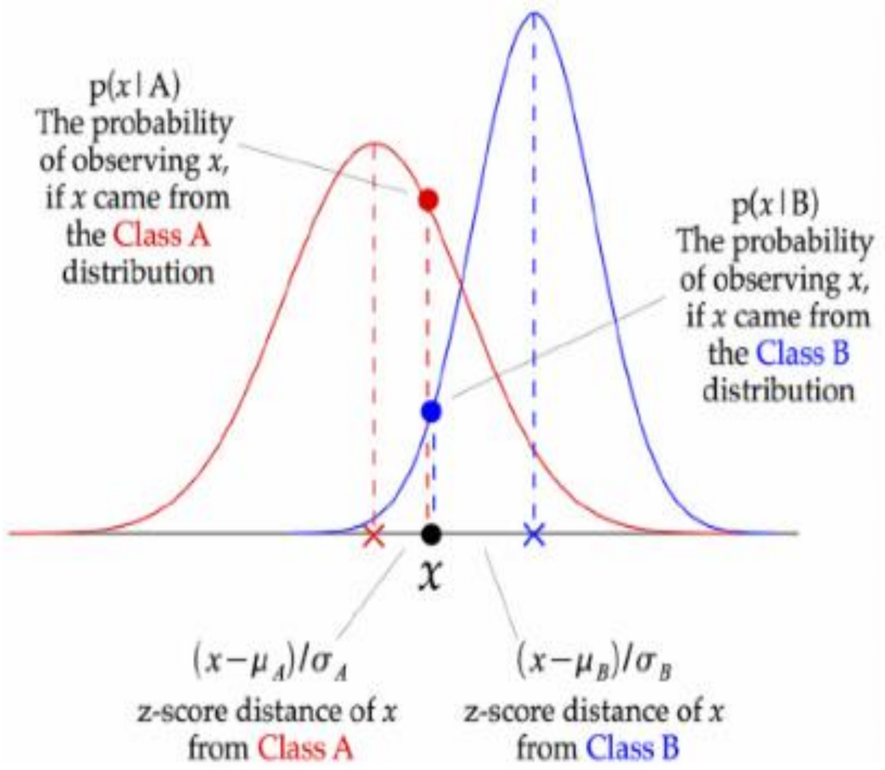
04
Computing
Permutation Importance

05
Model Retraining with
“Best” hyperparameters
(Tuned by GridSearchCV)

Model 1 : Gaussian Naive Bayes (GNB)

<u>hyperparameter tuning</u>

- variance smoothing



Model 1 : Gaussian Naive Bayes (GNB)

Approach 1: Default Settings

```
Train Accuracy using default settings: 73.63%
Test Accuracy using default settings: 75.71%
```

Approach 2: Hyperparameter tuning using GridSearchCV (Automated Tuning)

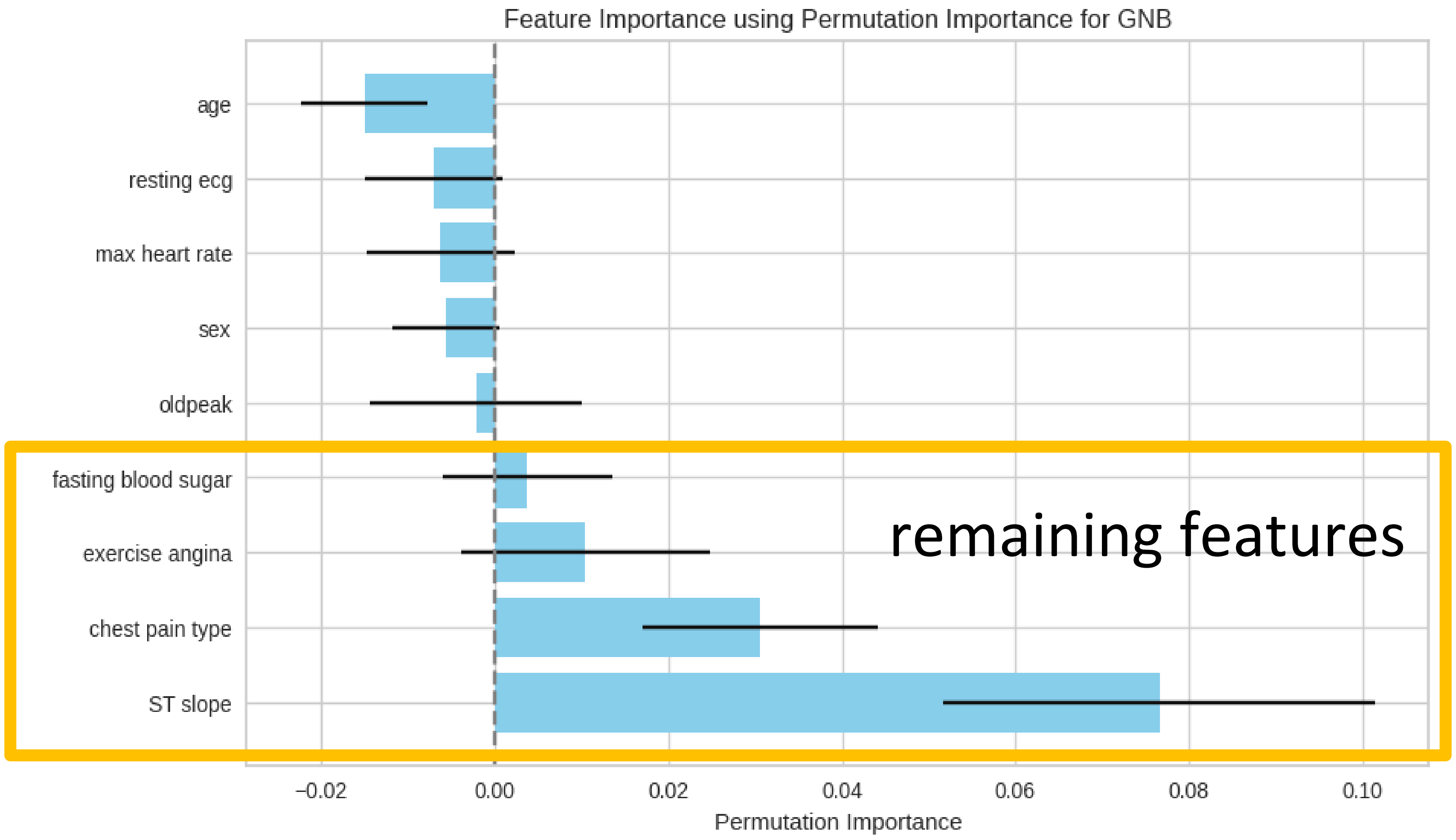
```
Best Hyperparameter: {'var_smoothing': 0.3511191734215131}
Best Cross-Validation Accuracy: 0.7397
Train Accuracy with the best hyperparameter: 73.87%
Test Accuracy with the best hyperparameter: 74.76%
```

* After eliminating negative features determined by permutation importance:

```
Best Hyperparameters after dropping negative features: {'var_smoothing': 0.8111308307896871}
Best Cross-Validation Accuracy after dropping negative features: 0.7660
Train Accuracy with best hyperparameters after dropping negative features: 76.73%
Test Accuracy with best hyperparameters after dropping negative features: 77.62%
```

Permutation Importance for GNB

	Feature	Importance	Std
8	ST slope	0.076508	0.024863
2	chest pain type	0.030635	0.013563
6	exercise angina	0.010476	0.014307
3	fasting blood sugar	0.003810	0.009712
7	oldpeak	-0.002063	0.012214
1	sex	-0.005556	0.006158
5	max heart rate	-0.006190	0.008532
4	resting ecg	-0.006984	0.007943
0	age	-0.014921	0.007350



Performance Evaluations
before and after Feature Elimination(GNB)

before

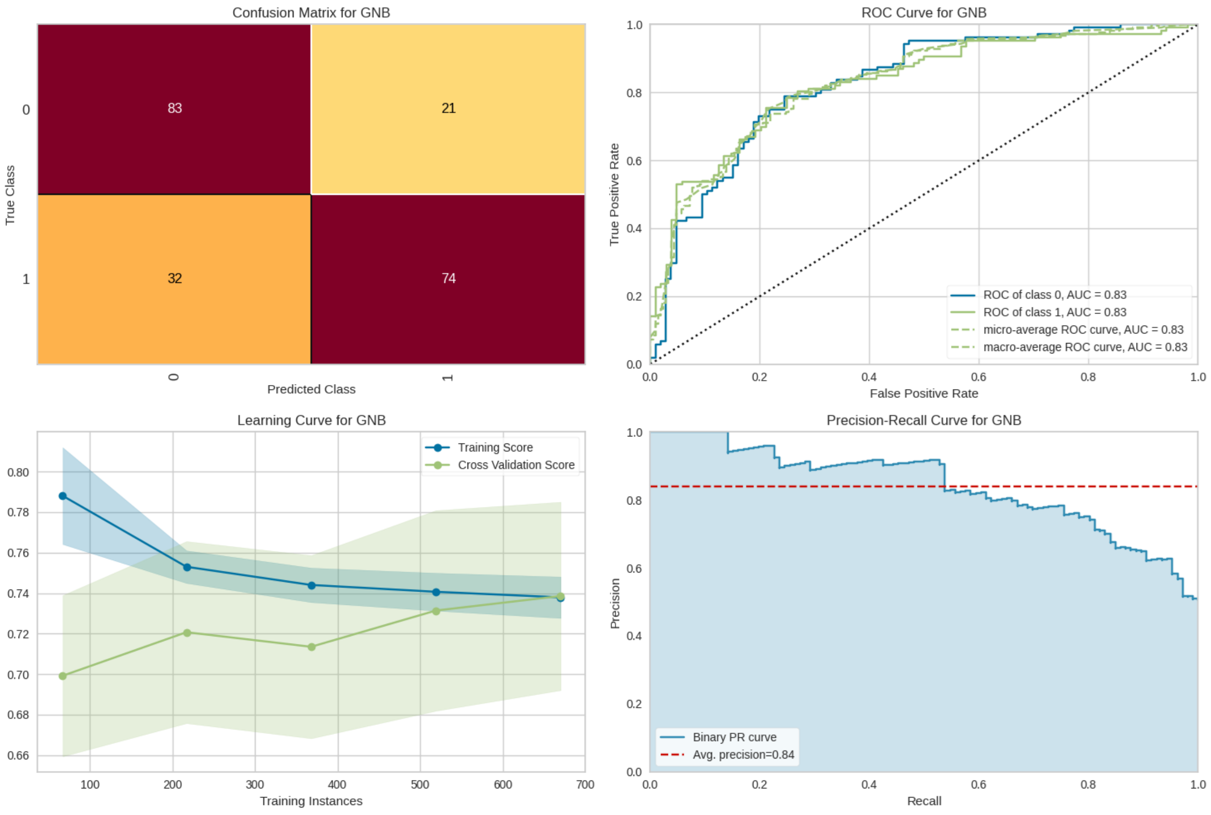
	precision	recall	f1-score	support
0	0.72	0.80	0.76	104
1	0.78	0.70	0.74	106
accuracy			0.75	210
macro avg	0.75	0.75	0.75	210
weighted avg	0.75	0.75	0.75	210

after

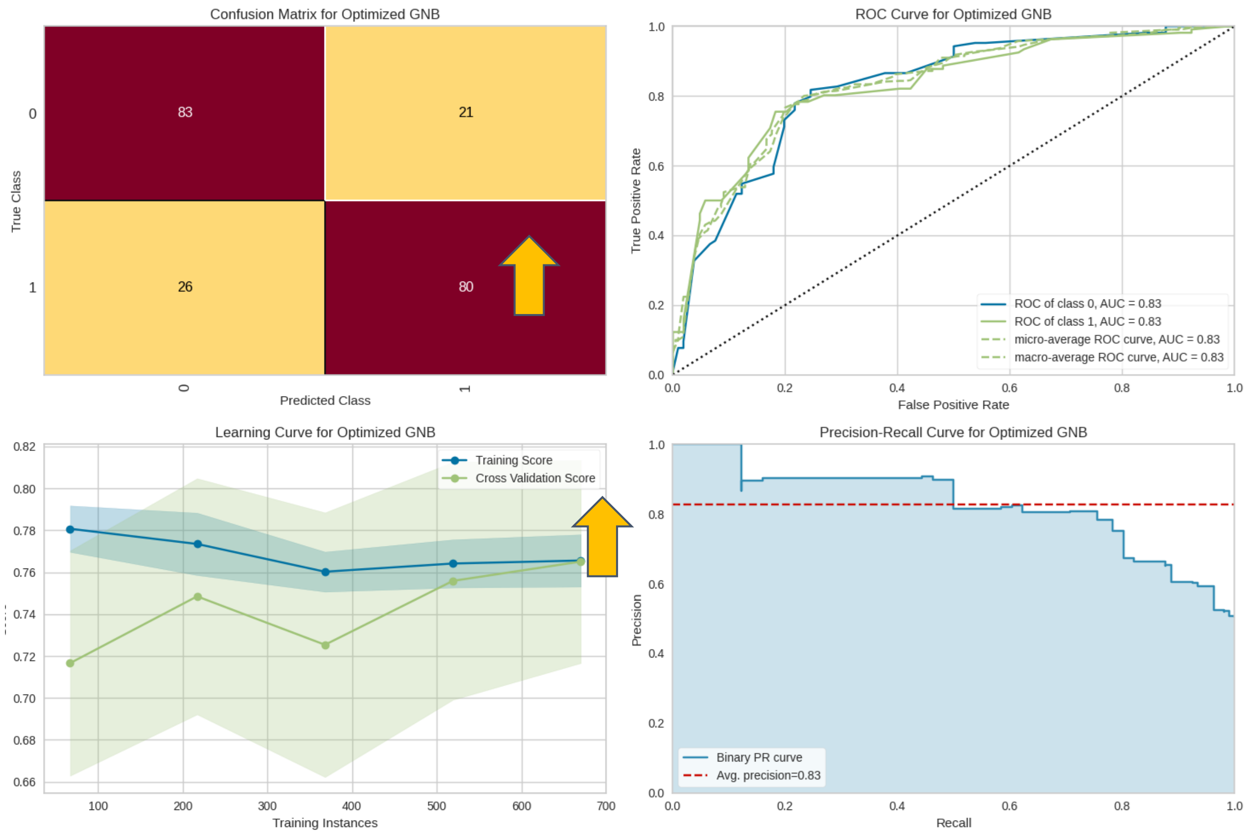
	precision	recall	f1-score	support
0	0.76	0.80	0.78	104
1	0.79	0.75	0.77	106
accuracy			0.78	210
macro avg	0.78	0.78	0.78	210
weighted avg	0.78	0.78	0.78	210

Performance Evaluations
before and after Feature Elimination(GNB)

before



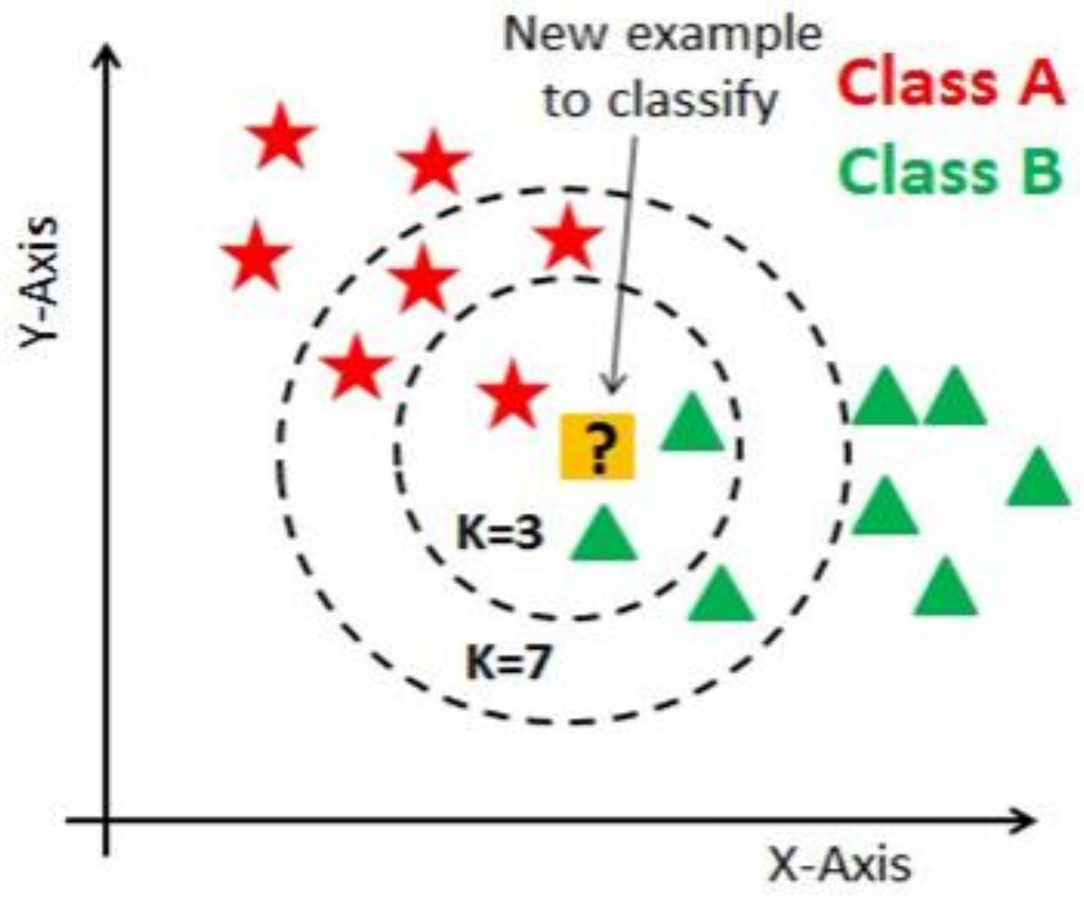
after



Model 2 : K-Nearest Neighbor (KNN)

<u>hyperparameter tuning</u>

- number of neighbors (k)
- distance metric



Model 2 : K-Nearest Neighbor (KNN)

Approach 1: Default settings

```
Train Accuracy with default settings: 84.01%
Test Accuracy with default settings: 75.71%
```

Approach 2: Direct Training and Testing (Manual tuning)

K = 14 / 18

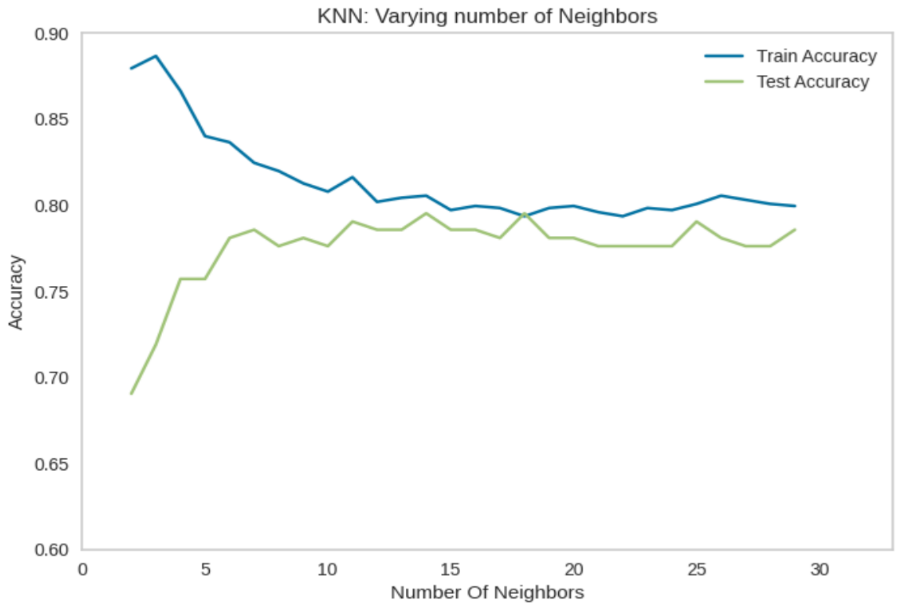
```
Train Accuracy with n_neighbors=14: 80.55%
Test Accuracy with n_neighbors=14: 79.52%
```

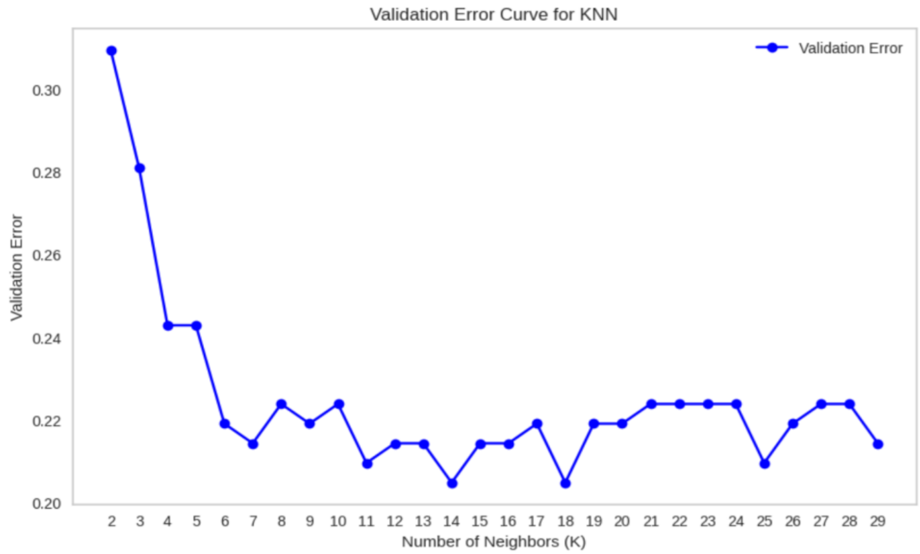
```
Train Accuracy with n_neighbors=18: 79.36%
Test Accuracy with n_neighbors=18: 79.52%
```

Approach 3 : Cross-Validation (Manual tuning)

K = 7

```
Train Accuracy with n_neighbors=7: 82.46%
Test Accuracy with n_neighbors=7: 78.57%
```





Model 2 : K-Nearest Neighbor (KNN)

Approach 4: Hyperparameter Tuning using GridSearchCV (Automated Tuning)

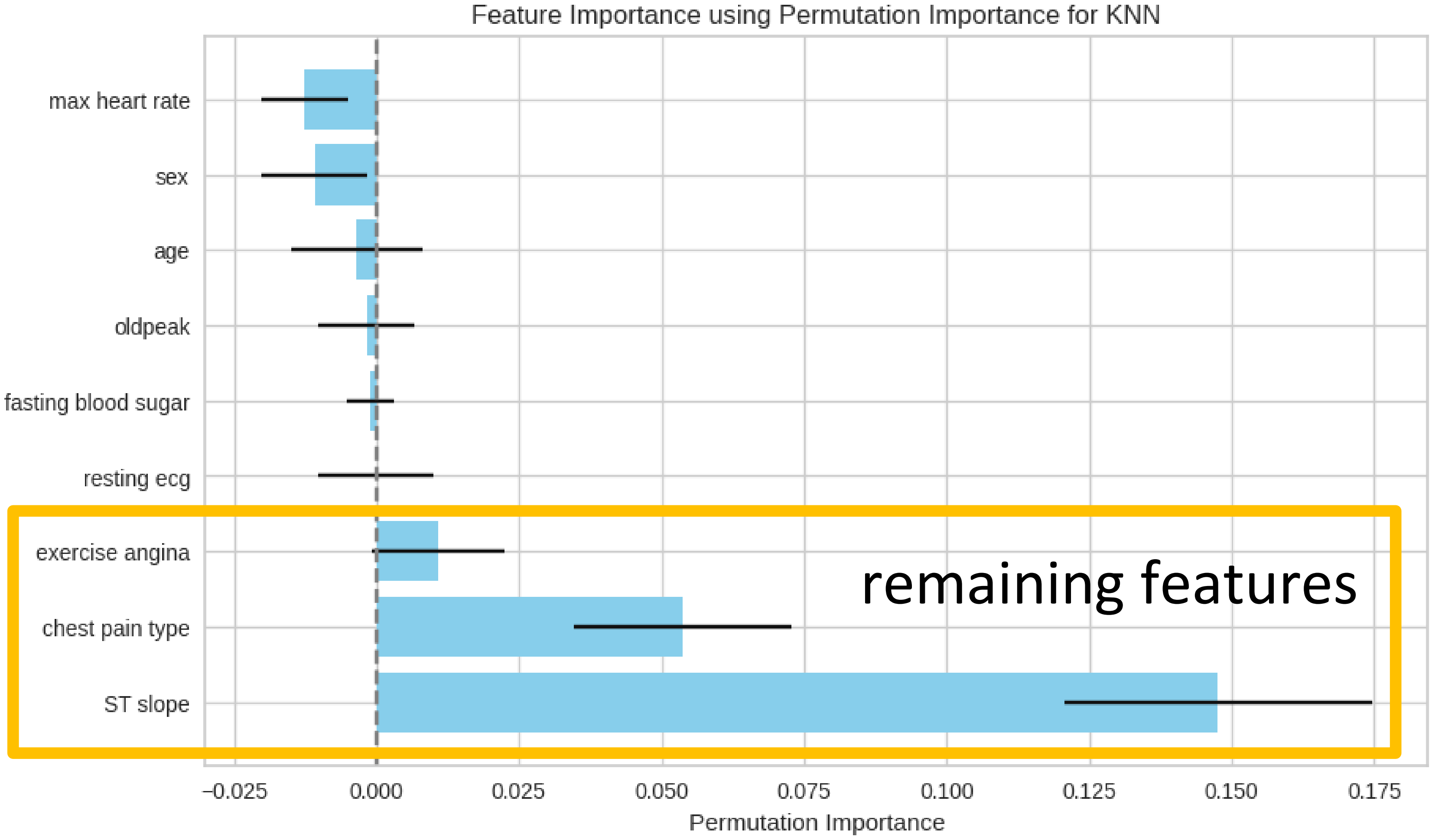
before permutation importance	params_knn_1	params_knn_2	params_knn_3	**params_knn_4**
(1) N-neigbor (2-29)	22	17	20	**25**
(2) metric	NA	manhattan	NA	**manhattan**
(3) weight	NA	distance	distance	**NA**
train accuracy	79.36%	**100.00%**	**100.00%**	**80.31%**
test accuracy	77.62%	77.62%	77.14%	**78.57%**

* After eliminating negative features determined by permutation importance:

after permutation importance	**knn_reduced**
(1) N-neigbor (2-29)	**28**
(2) metric	**manhattan**
(3) weight	**NA**
train accuracy	**80.31%**
test accuracy	**80.00%**

Permutation Importance for KNN

	Feature	Importance	Std
8	ST slope	0.147619	0.027077
2	chest pain type	0.053651	0.019085
6	exercise angina	0.010794	0.011725
4	resting ecg	-0.000159	0.010175
3	fasting blood sugar	-0.001111	0.004203
7	oldpeak	-0.001746	0.008383
0	age	-0.003492	0.011530
1	sex	-0.010952	0.009213
5	max heart rate	-0.012698	0.007612



Performance Evaluations before and after Feature Elimination (KNN)

before

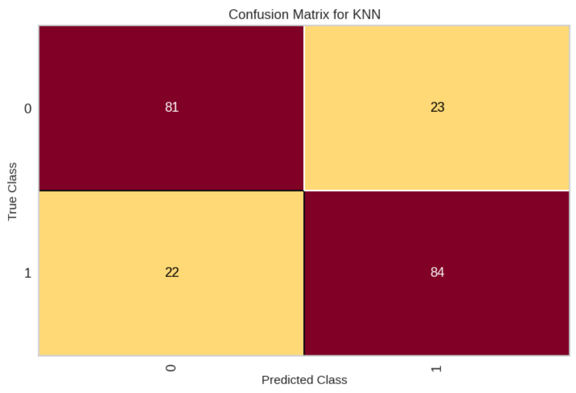
	precision	recall	f1-score	support
0	0.79	0.78	0.78	104
1	0.79	0.79	0.79	106
accuracy			0.79	210
macro avg	0.79	0.79	0.79	210
weighted avg	0.79	0.79	0.79	210

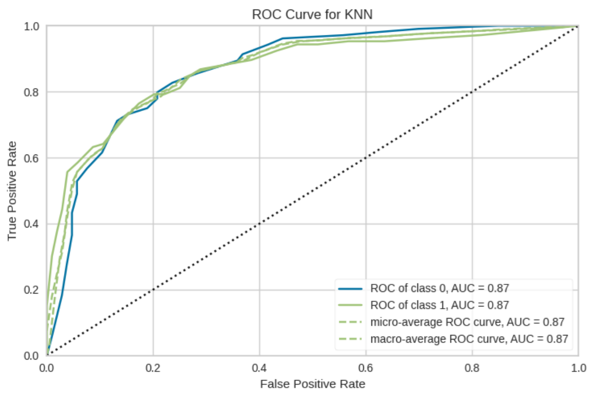
after

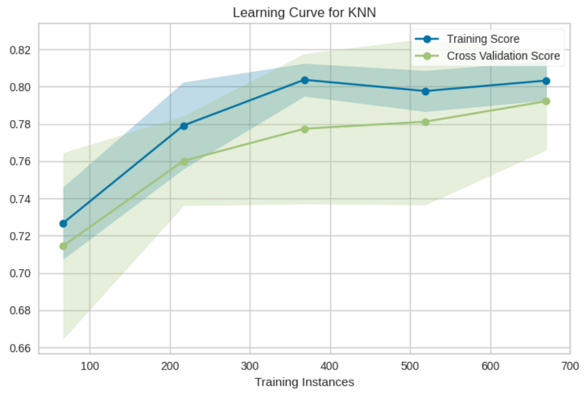
	precision	recall	f1-score	support
0	0.82	0.77	0.79	104
1	0.79	0.83	0.81	106
accuracy			0.80	210
macro avg	0.80	0.80	0.80	210
weighted avg	0.80	0.80	0.80	210

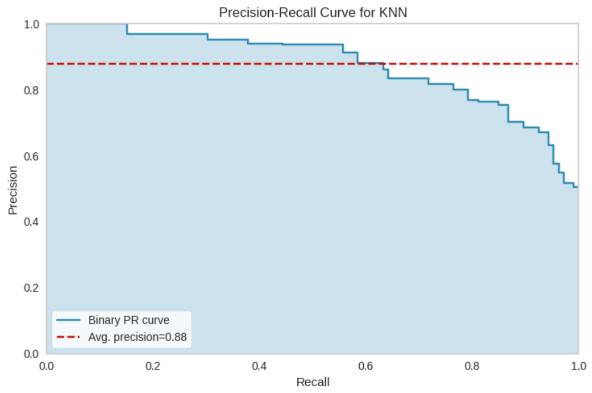
Performance Evaluations before and after Feature Elimination (KNN)

before

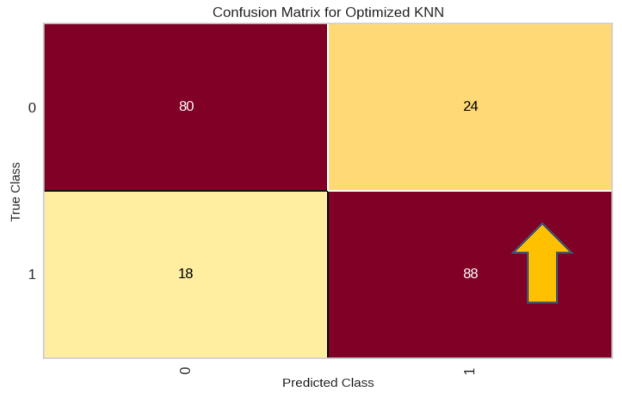


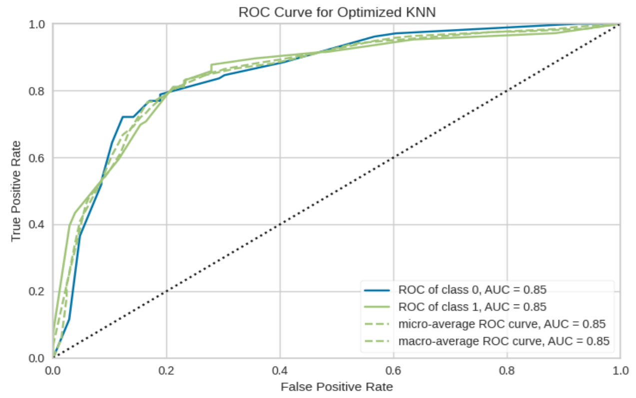


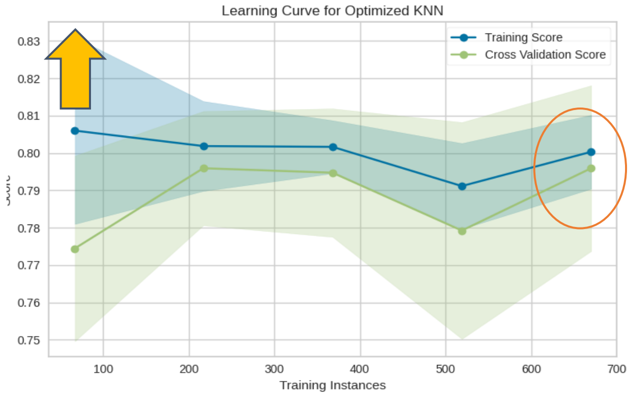


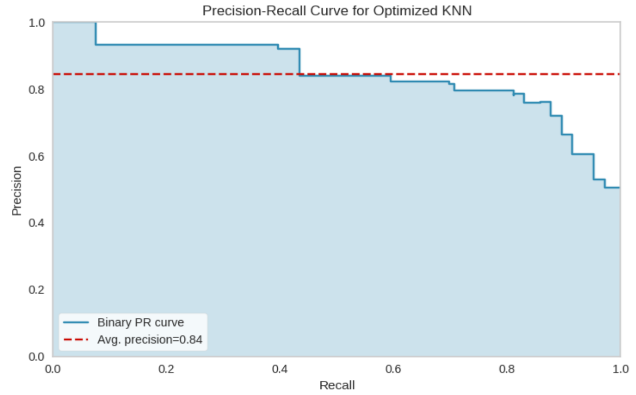


after









Model 3 : Support Vector Machine (SVM)

hyperparameter tuning

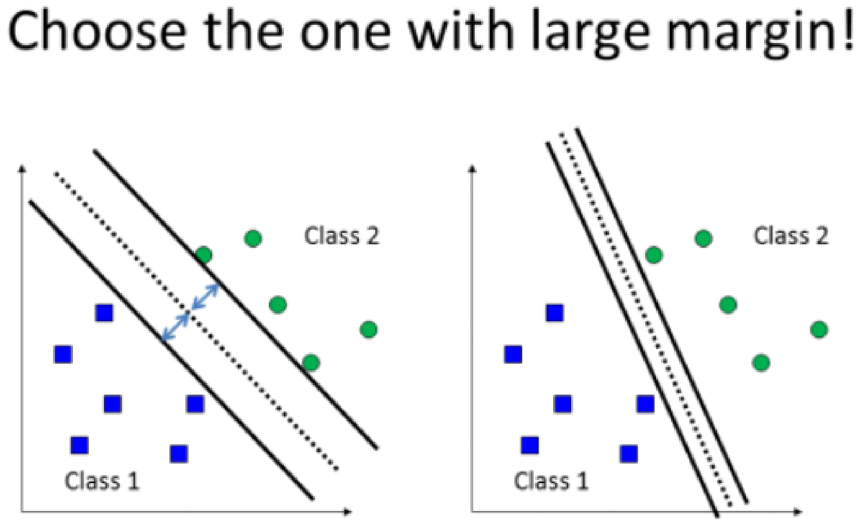
■ **C** (fault tolerance)

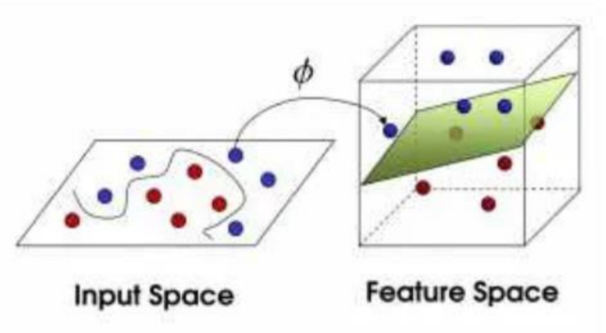
the trade-off between maximizing the margin and minimizing the classification error on the training data

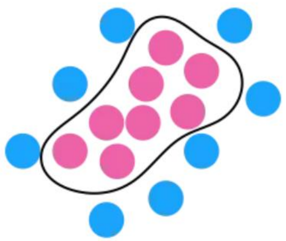
Smaller C: allows more misclassifications -> more support vectors

■ **gamma** (Coefficient of radial basis function (RBF) kernel)

Smaller gamma: the influence of a single training example extends further







Model 3 : Support Vector Machine (SVM)

Approach 1: Default settings

We get an accuracy of 78.1% without tuning the hyperparameters.

Approach 2 : Direct Training and Testing (Manual tuning)

```
Train Accuracy using manual tuning: 94.75%
Test Accuracy using manual tuning: 73.81%
```

Approach 3: Hyperparameter tuning using GridSearchCV (Automated Tuning)

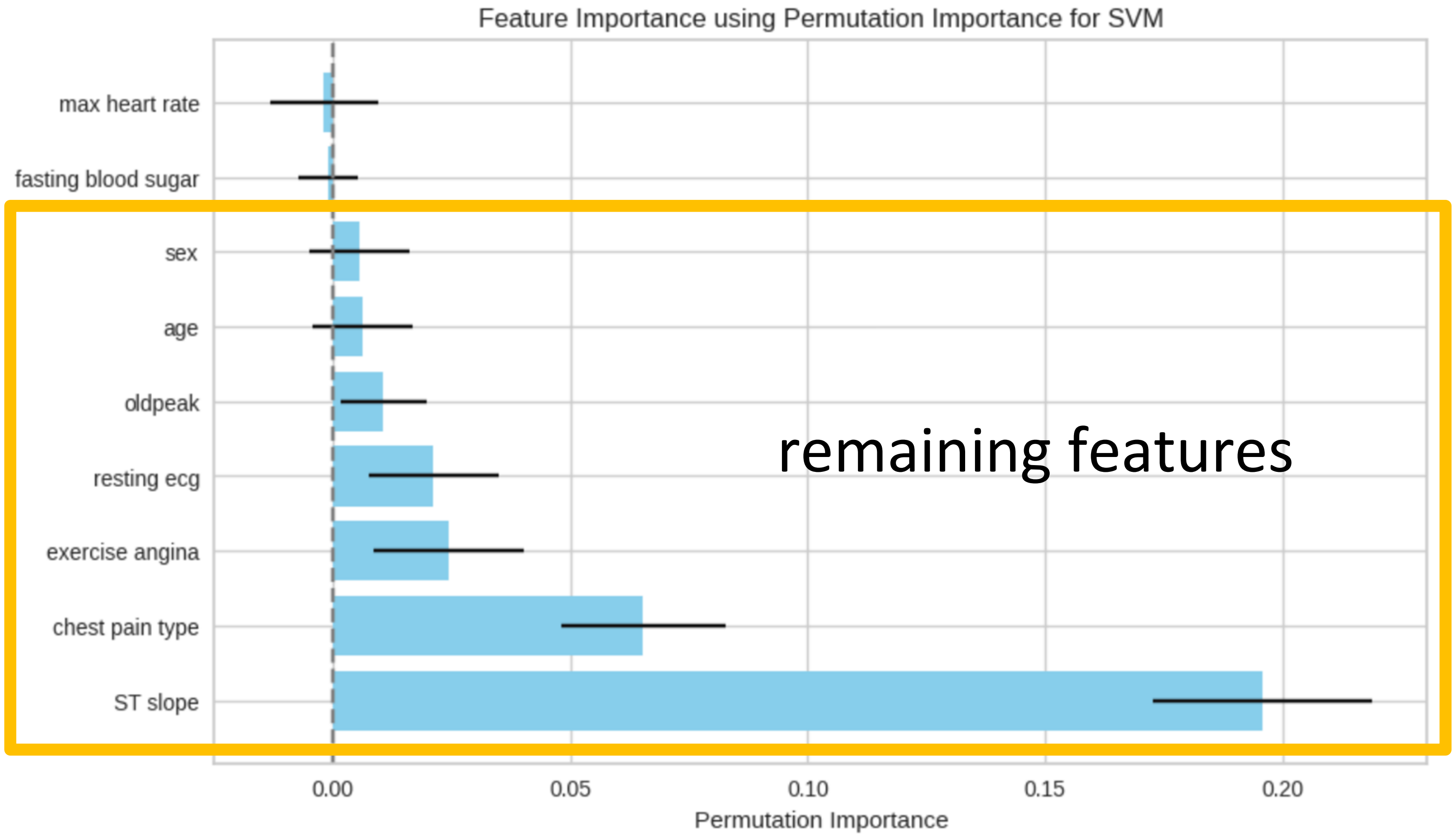
```
Best Hyperparameter: {'C': 1000, 'gamma': 0.01}
Best Cross-Validation Accuracy: 0.7923
Train Accuracy with the best hyperparameter: 85.20%
Test Accuracy with the best hyperparameter: 79.05%
```

*** After eliminating negative features determined by permutation importance:**

```
Best Hyperparameter: {'C': 1000, 'gamma': 0.01}
Best Cross-Validation Accuracy: 0.7923
Train Accuracy with the best hyperparameter: 82.58%
Test Accuracy with the best hyperparameter: 80.00%
```

Permutation Importance for SVM

	Feature	Importance	Std
8	ST slope	0.195873	0.023092
2	chest pain type	0.065238	0.017307
6	exercise angina	0.024286	0.015810
4	resting ecg	0.021111	0.013729
7	oldpeak	0.010635	0.009175
0	age	0.006190	0.010516
1	sex	0.005556	0.010511
3	fasting blood sugar	-0.000952	0.006197
5	max heart rate	-0.001905	0.011442



Performance Evaluations before and after Feature Elimination (SVM)

before

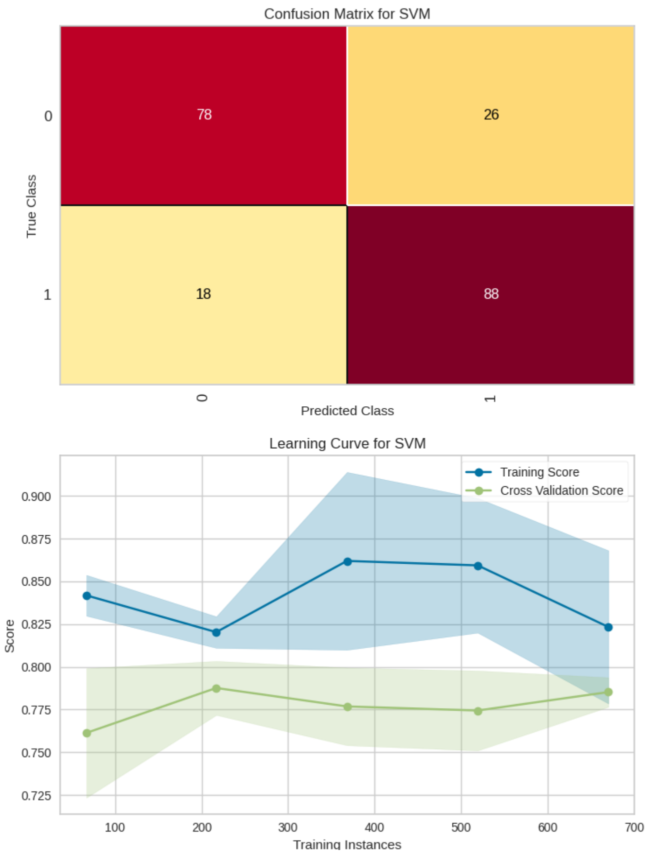
	precision	recall	f1-score	support
0	0.81	0.75	0.78	104
1	0.77	0.83	0.80	106
accuracy			0.79	210
macro avg	0.79	0.79	0.79	210
weighted avg	0.79	0.79	0.79	210

after

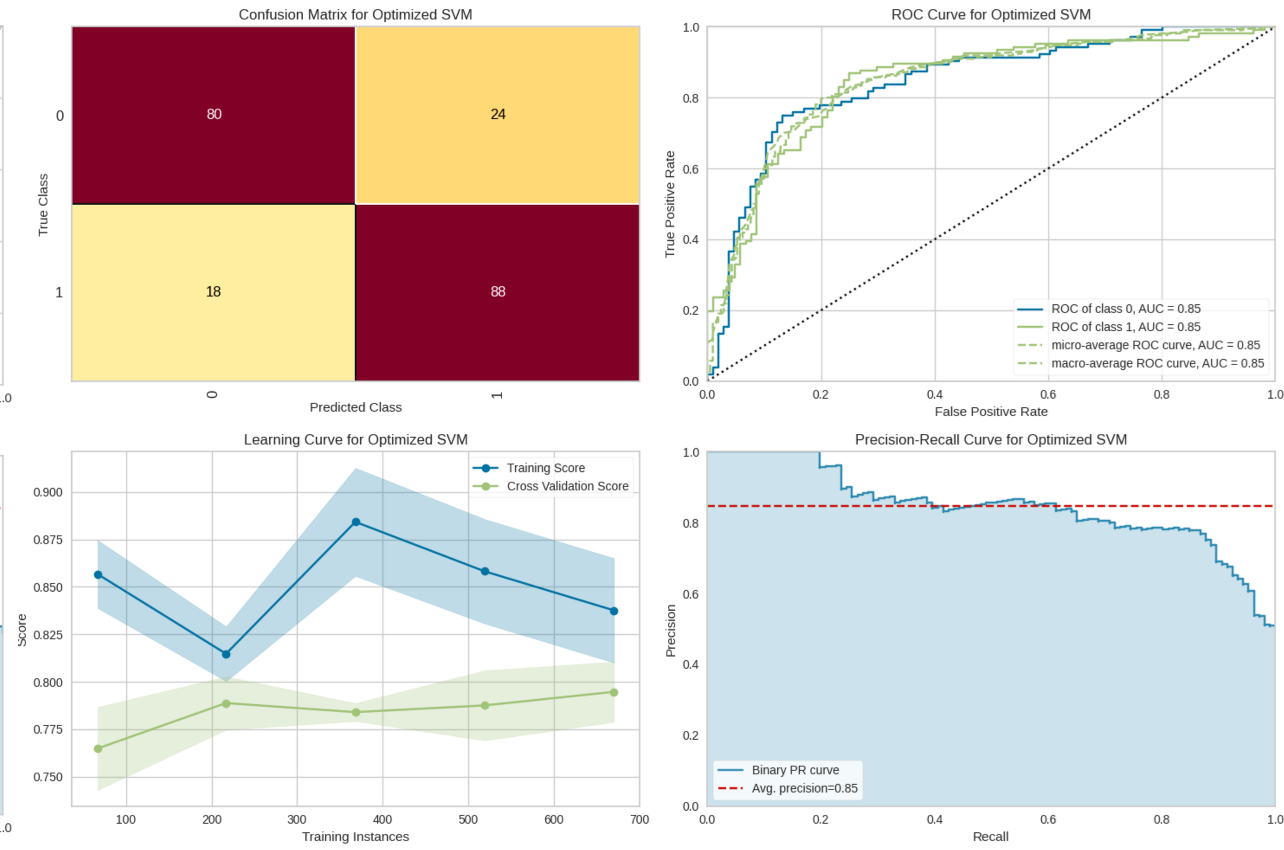
	precision	recall	f1-score	support
0	0.82	0.77	0.79	104
1	0.79	0.83	0.81	106
accuracy			0.80	210
macro avg	0.80	0.80	0.80	210
weighted avg	0.80	0.80	0.80	210

Performance Evaluations before and after Feature Elimination (SVM)

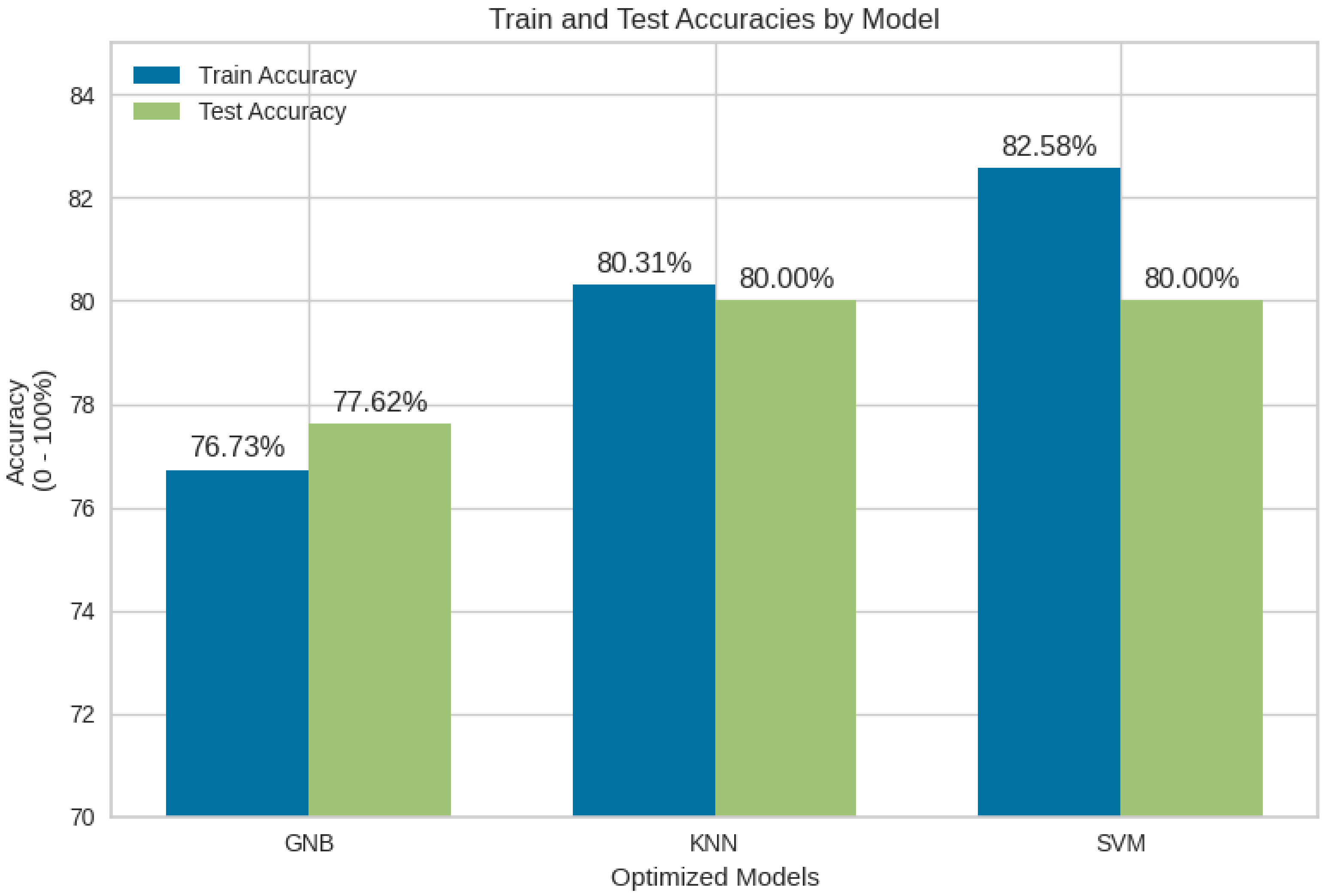
before



after

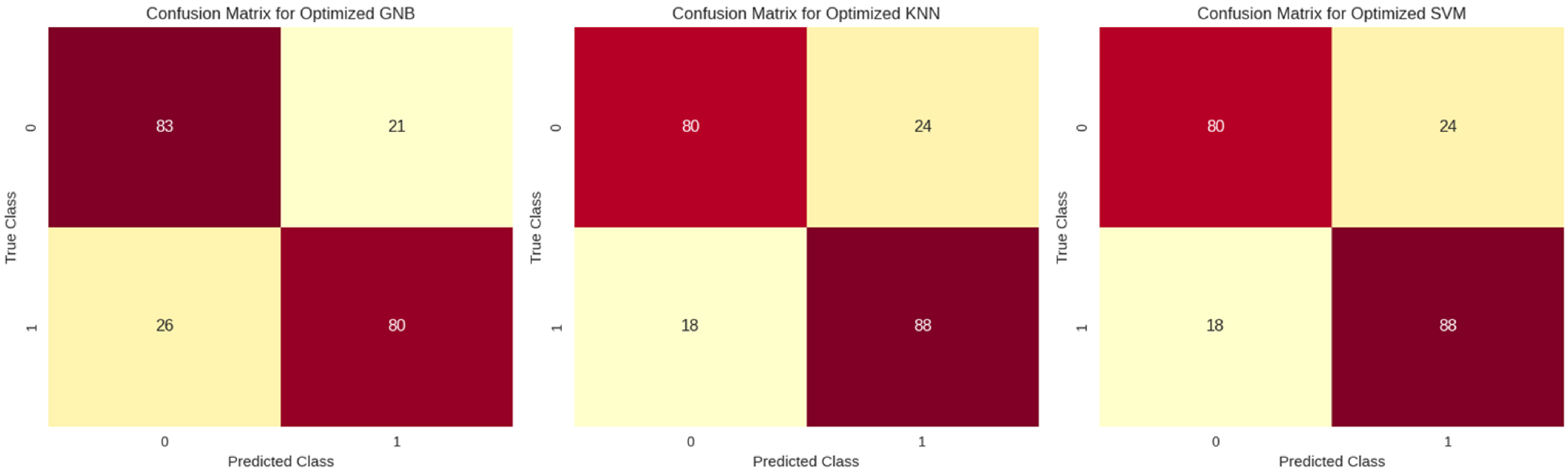


Comparisons of Optimized Models



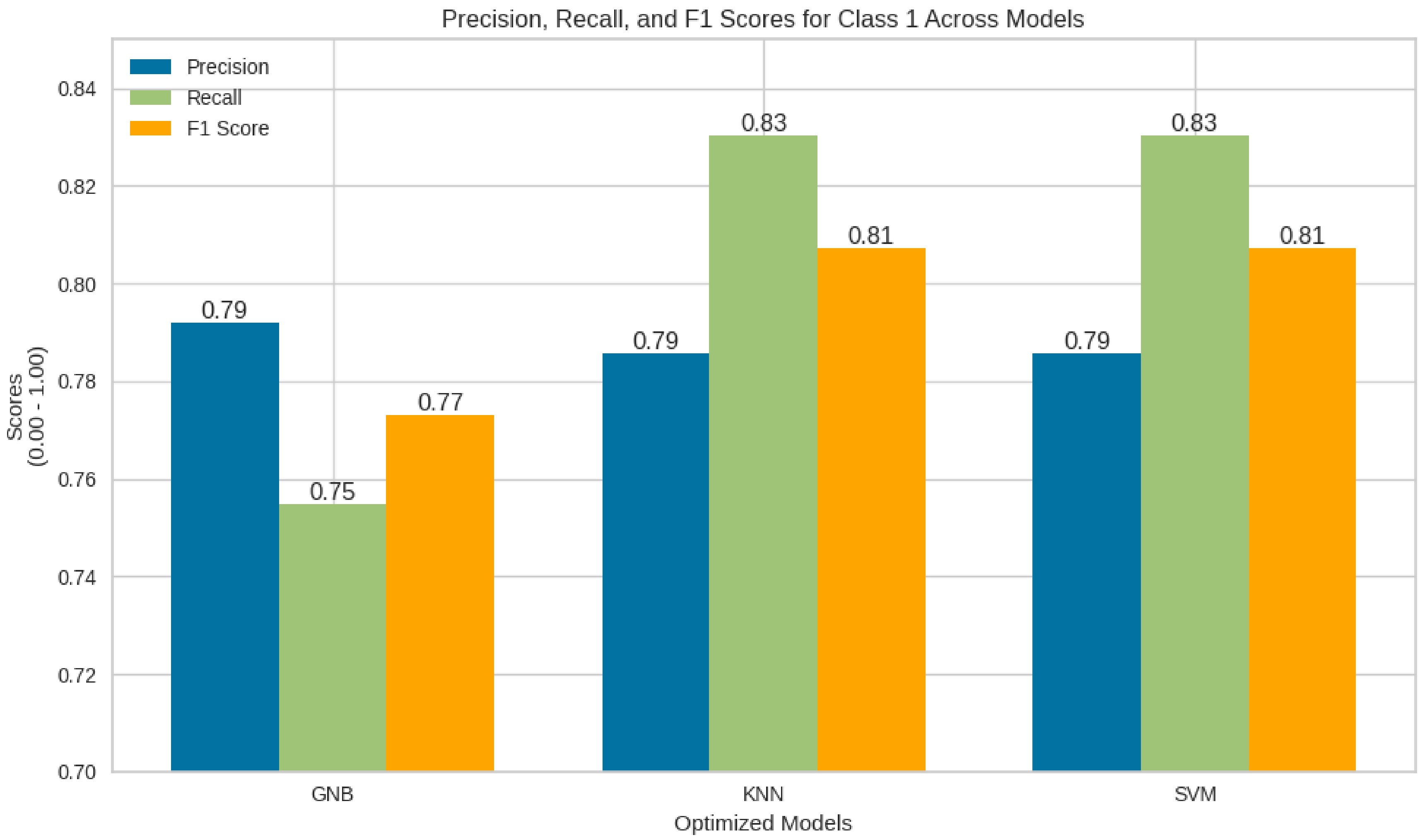
- Similar accuracies among models
- GNB:
  Less effective
- KNN and SVM:
  Equal test accuracies
- KNN:
  Best Generalization

Comparisons of Optimized Models



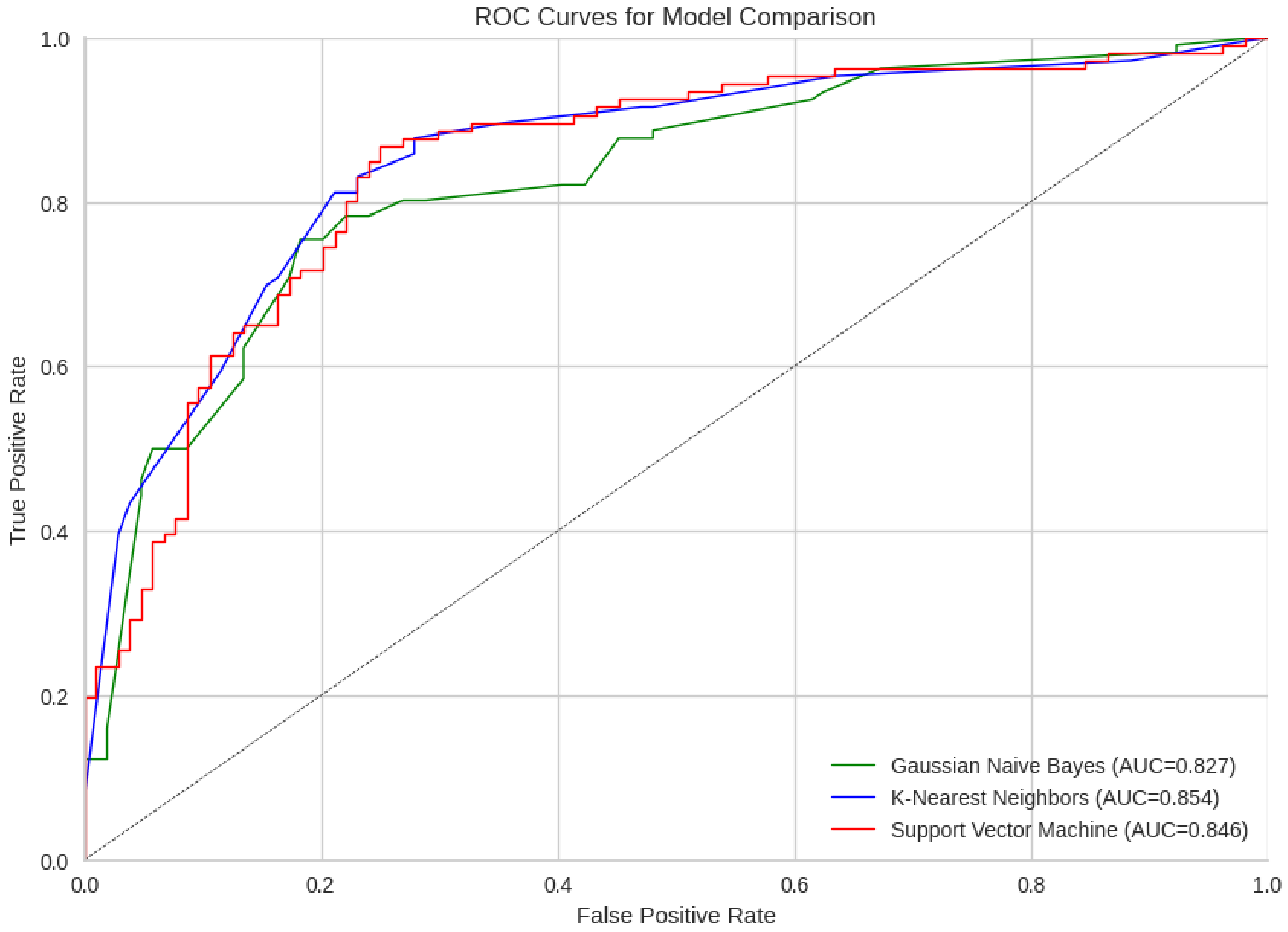
- GNB: Better at Negatives
- KNN and SVM:
  - Better at Positives
  - Identical TN, FP, FN, and TP values

Comparisons of Optimized Models



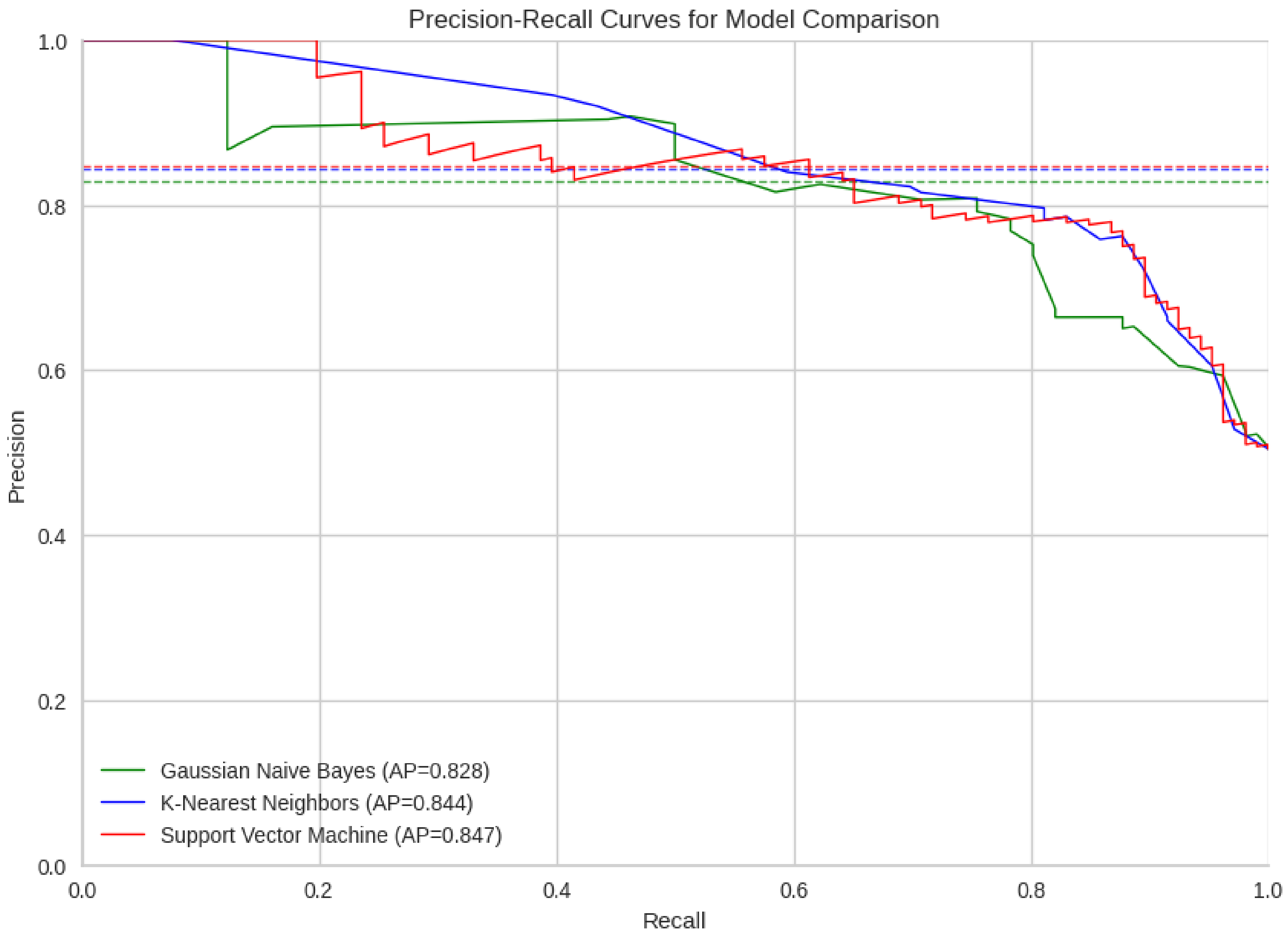
- Equal precision
- KNN and SVM:
  - Higher recall than GNB
  - Identical scores

Comparisons of Optimized Models



- GNB:
  Lower ROC curve
- KNN and SVM:
  Similar ROC curves
- KNN:
  Slightly higher AUC

Comparisons of Optimized Models



- GNB:
  Lowest in AP
- KNN and SVM:
  Similar AP
- KNN:
  More stable

Conclusions

1. **KNN Superiority (Test Accuracy:** 80.00%):
   leads with highest AUC and stable precision-recall,
   indicating consistent performance

2. **SVM Close Behind (Test Accuracy:** 80.00%)**:**
   closely follows KNN, with a slightly higher AP,
   but less stable precision-recall.

3. **GNB Lagging (Test Accuracy:** 77.62%)**:**
   trails in AUC, AP, and positive case classification,
   making it the least effective model.

Limitations & Challenges

1. Resource-Intensive Computation
   - especially in hyperparameter tuning and model training

2. Model optimization: time-consuming and complex (but it is worth it)

3. Limited time for further model building and evaluations

4. Difficult to make model comparisons due to similarities in findings

5. Lack of professional expertise to perform more precise feature selection

Future Work

1. Conduct an in-depth analysis on the significant feature ST-slope, by evaluating the impact of upsloping, flat, and downsloping characteristics
2. Further perform model evaluations for KNN and SVM by testing them under different scenarios (e.g. different train-test split ratios)
3. Experiment with more models and implement ensemble learning to synthesize predictions from various models for improved accuracy
4. Verify with medical domain knowledge to determine whether there is a better way to deal with feature elimination other than dropping the negative features
5. Establish a data pipeline to streamline the workflow of the code

Reference

Dataset:
Heart Disease Dataset (kaggle.com)

Heart Disease Dataset (Comprehensive) | IEEE DataPort (ieee-dataport.org)

Feature Selection Dropping the low scored features:
https://www.kaggle.com/code/totoro29/heart-disease-eda-prediction

Permutation Importance:
https://flageditors.medium.com/%E6%A9%9F%E5%99%A8%E5%AD%B8%E7%BF%92%E5%8B%95%E6%89%8B%E5%81%9Alesson-4-%E4%BD%BF%E7%94%A8permutation-importance%E4%BE%86%E9%81%B8%E5%8F%96%E9%87%8D%E8%A6%81%E7%89%B9%E5%BE%B5-%E4%B8%8A%E7%AF%87-7d9b8a8cab30

KNN tuning :
https://medium.com/@agrawalsam1997/hyperparameter-tuning-of-knn-classifier-a32f31af25c7

Appendix

Data Cleansing

- Remove the unnecessary column 'Unnamed: 0' (#1)
- Check null value (#2)

#1

	Unnamed: 0	age	sex	chest pain type	resting bps
0	0	40	1	2	140
1	1	49	0	3	160
2	2	37	1	2	130
3	3	48	0	4	138
4	4	54	1	3	150

#2

```
# Determine whether there are missing values
df.isnull().sum()
```

	0
age	0
sex	0
chest pain type	0
resting bps	0
cholesterol	0
fasting blood sugar	0
resting ecg	0
max heart rate	0
exercise angina	0
oldpeak	0
ST slope	0
target	0

dtype: int64

Data Cleansing

- Check duplicated value (#3)
- Check Dataset size (number of rows, number of columns) (#4)

#3

```
# Determine whether there are duplicated records
df.duplicated().sum()
```

```
0
```

#4

```
# Explore the data
print("Dataset size (number of rows, number of columns):", df.shape)
```

```
Dataset size (number of rows, number of columns): (1048, 12)
```