ASSIGNMENT 2 PART2
(CLASSIFICATION)

محمد احسين عبدالواحد الازرق 2190203749
زياد خالد سعد الشريف 2190203644

1. Problem Overview

In this assignment, we build a binary classification problem using a provided dataset. The primary objective was to train and compare the performance of various classification algorithms and visualize their evaluation metrics.

The target variable (target) has two classes:

0: No heart disease
1: Have heart disease

We will compare 4 different models { }

2. Data Preprocessing

After loading the dataset, we renamed the target column from "condition" to "target" for clarity. To understand the data, we previewed the first few rows using df.head() and examined its structure.

The dataset contains 297 entries and 14 columns (features), all of which are numeric (either int64 or float64). No missing values were found, so no imputation or removal was necessary. Typically, if missing values exist, we might drop the rows, fill them with the mean/median/mode, or use interpolation, but this was not required here.

We also checked descriptive statistics using df.describe(). We noticed that some features had higher standard deviations than others, indicating varying value ranges across features.

Since several machine learning models, such as {Logistic Regression, SVM, and k-NN} perform better when features are scaled to a similar range, we applied feature scaling using StandardScaler after splitting the data into 80% training and 20% test sets.

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	69	1	0	160	234	1	2	131	0	0.1	1	1	0	0
1	69	0	0	140	239	0	0	151	0	1.8	0	2	0	0
2	66	0	0	150	226	0	0	114	0	2.6	2	0	0	0
3	65	1	0	138	282	1	2	174	0	1.4	1	1	0	1
4	64	1	0	110	211	0	2	144	1	1.8	1	0	0	0

Display first 5 rows

	age	sex	cp	trestbps	chol
count	297.000000	297.000000	297.000000	297.000000	297.000000
mean	54.542088	0.676768	2.158249	131.693603	247.350168
std	9.049736	0.468500	0.964859	17.762806	51.997583
min	29.000000	0.000000	0.000000	94.000000	126.000000
25%	48.000000	0.000000	2.000000	120.000000	211.000000
50%	56.000000	1.000000	2.000000	130.000000	243.000000
75%	61.000000	1.000000	3.000000	140.000000	276.000000
max	77.000000	1.000000	3.000000	200.000000	564.000000

summary statistics

```
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
target      0
dtype: int64
```

missing values

```
RangeIndex: 297 entries, 0 to 296
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       297 non-null    int64
 1   sex       297 non-null    int64
 2   cp        297 non-null    int64
 3   trestbps  297 non-null    int64
 4   chol      297 non-null    int64
 5   fbs       297 non-null    int64
 6   restecg   297 non-null    int64
 7   thalach   297 non-null    int64
 8   exang     297 non-null    int64
 9   oldpeak   297 non-null    float64
 10  slope     297 non-null    int64
 11  ca        297 non-null    int64
 12  thal      297 non-null    int64
 13  target    297 non-null    int64
dtypes: float64(1), int64(13)
memory usage: 32.6 KB
```

data info

3. Exploratory Data Analysis (EDA)

First, we summarized the data by target (`df.groupby('target').mean()`) to see average feature values for patients with and without heart disease. For example, patients with disease tend to have higher cholesterol and lower max heart rate. This helps understand group differences for prediction.

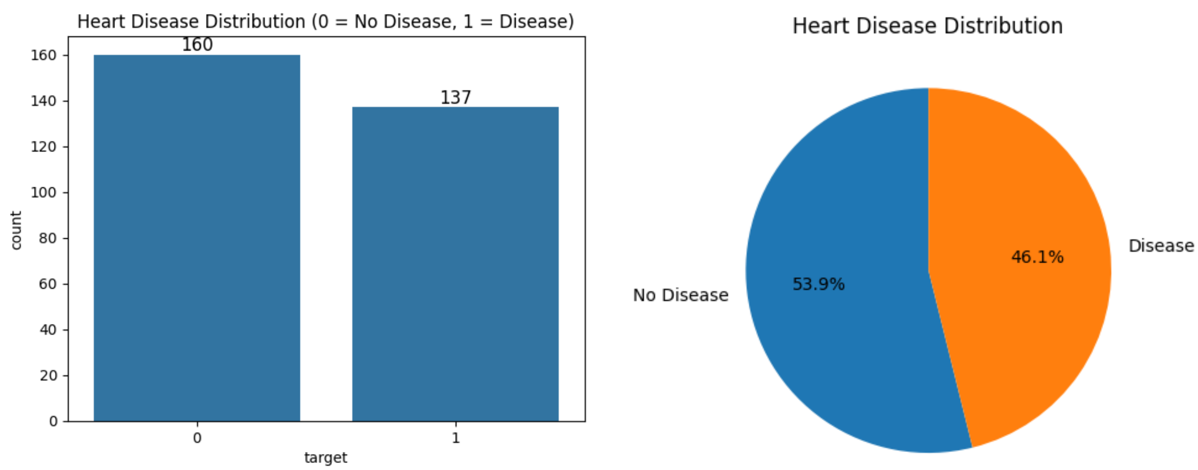
Then, we checked the class distribution. It's nearly balanced: 53.9% no disease, 46.1% disease so no need for special imbalance handling.

Next, we computed the correlation matrix. Most features have low correlation with each other (good). The most correlated features with the target are thal, ca, oldpeak, exang, and cp.

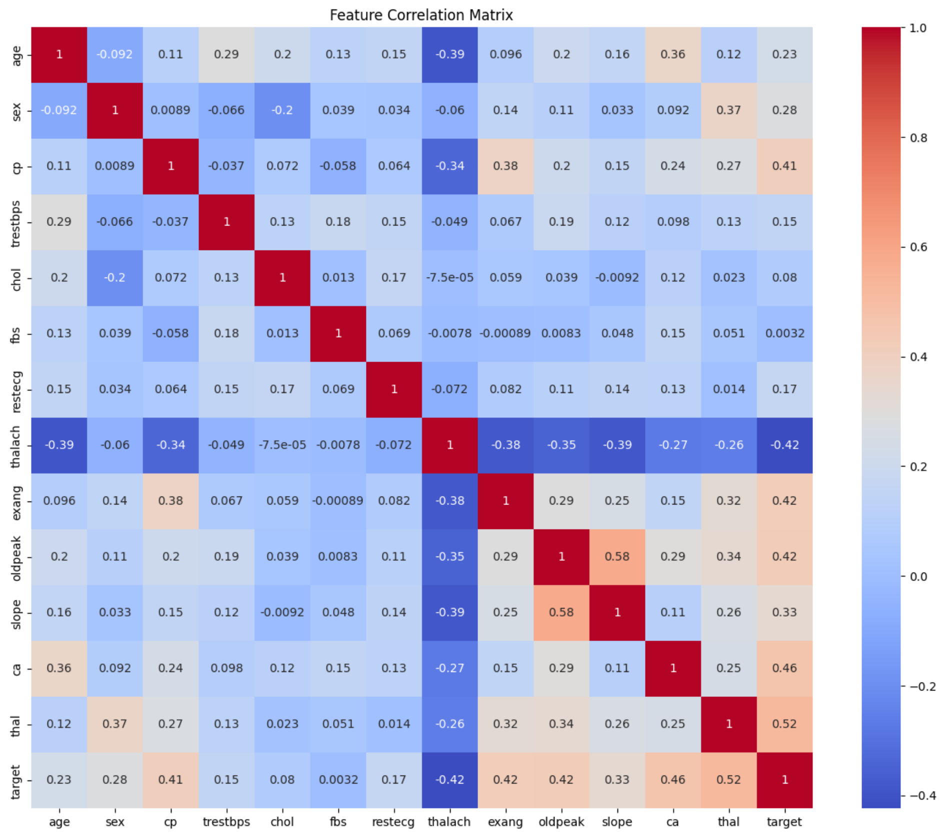
Finally, we used a pairplot to visualize relationships between features and the target. It helps spot patterns and differences between classes and guides us on choosing suitable models and decision boundaries.

target	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
0	52	0.5	1.7	129	243	0.1	0.8	158	0.1	0.5	0.4	0.2	0.3
1	56	0.8	2.5	134.6	25	0.1	1.1	139	0.5	1.5	0.8	1.1	1.3

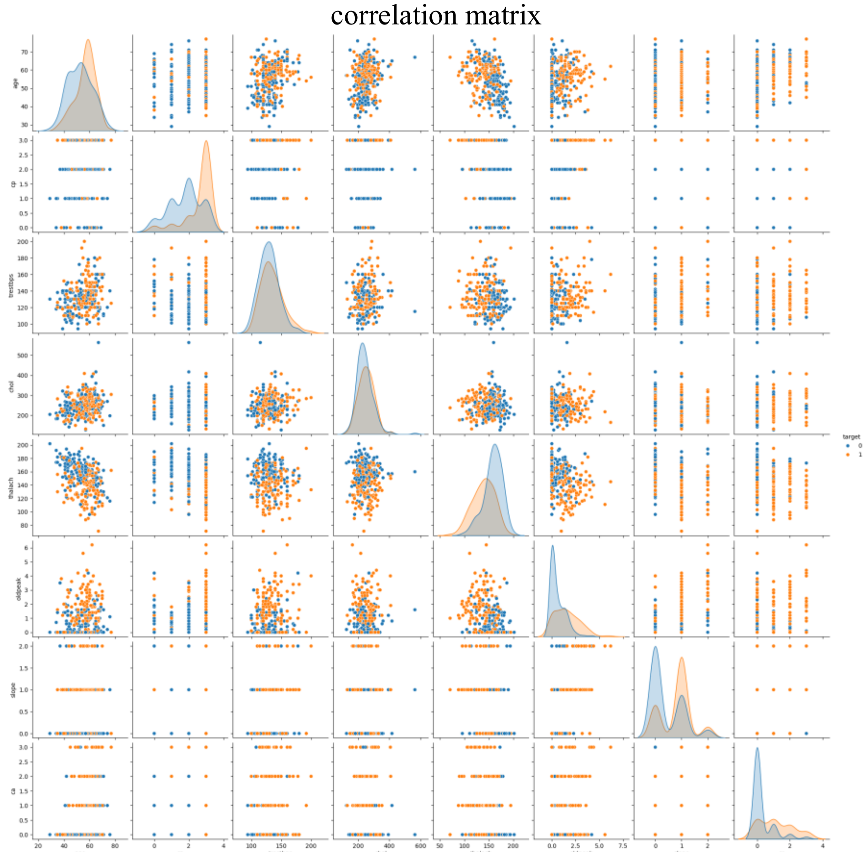
Summary statistics grouped by target



class distribution



correlation matrix



pairwise relationships

Feature	Correlation to target
target	1.000000
thal	0.520516
ca	0.463189
oldpeak	0.424052
exang	0.421355
cp	0.408945
slope	0.333049
sex	0.278467
age	0.227075
restecg	0.166343
trestbps	0.153490
chol	0.080285
fbs	0.003167
thalach	-0.423817

Correlation with target

4. Feature Selection / Engineering

We first drop features with low correlation to target (means they don't affect the prediction much).
From above, "fbs" has the lowest correlation (0.003) and gets dropped.
You might think "thalach" (-0.42) is lower because it's negative, but only correlations close to 0 (like |0.003| < 0.05) are weak.
Strong negative or positive values (like -0.42) still matter for prediction.

Next, we check for highly correlated features (above 0.8), since they bring redundant info, but in our case, no pairs passed that, so nothing removed.

So the new selected features will be "13" since we remove "fbs"

```
['age', 'sex', 'cp', 'trestbps', 'chol', 'restecg', 'thalach', 'exang',
'oldpeak', 'slope', 'ca', 'thal', 'target']
```

5. Model Selection and Training

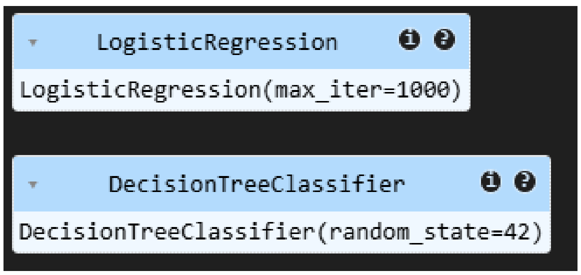
After preprocessing and feature engineering, we move to building models. In this assignment, we use 4 classification models: Logistic Regression, Decision Tree, k-NN, and SVM.

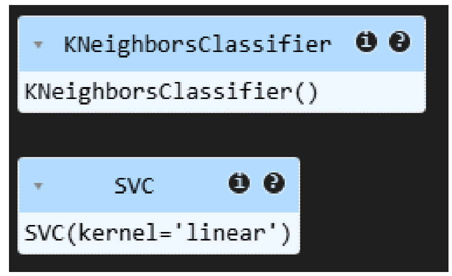
These models are suitable because we need to classify the target as 0 or 1.

- **Logistic Regression:** predicts probability of heart disease (good for linear relations).
- **Decision Tree:** captures non-linear relations between features and target.
- **k-NN:** makes predictions based on similar patient records.
- **SVM:** separates patients into two groups with the best boundary.

We initialize and train these models using the training data we split earlier, for k-NN , we choose k to be 5.

Even though this section is called Model Selection, we didn't actually select one model yet — we're trying multiple models first to compare their performance later.





6. Model Evaluation

This is the main part of the assignment. Here we evaluate the models we trained, using known evaluation metrics: **Accuracy, Precision, Recall, F1-Score, and ROC AUC**.

- We first start by making predictions on test data.
For **Logistic Regression** and **Decision Tree** we use the normal test data `X_test` (without scaling).

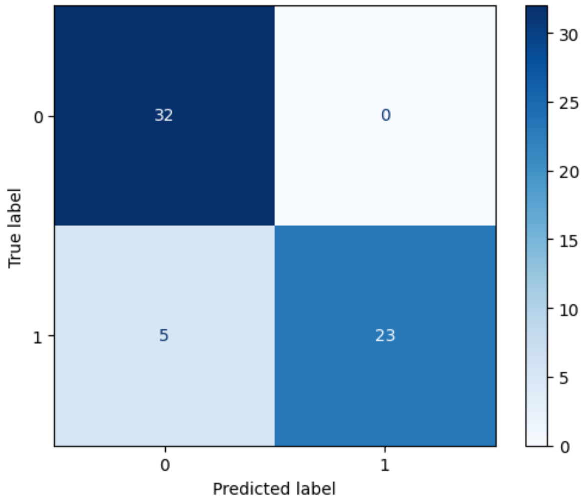
But for **k-NN** and **SVM**, we use the scaled data `X_test_scaled`.
As we said before, this is because **k-NN is very sensitive to unscaled data**, and its performance drops without it. **SVM is sensitive too, but less than k-NN**. ^ ^

- Then we compute the evaluation metrics.
By the way, they measure how good the model's predictions are compared to the true values.

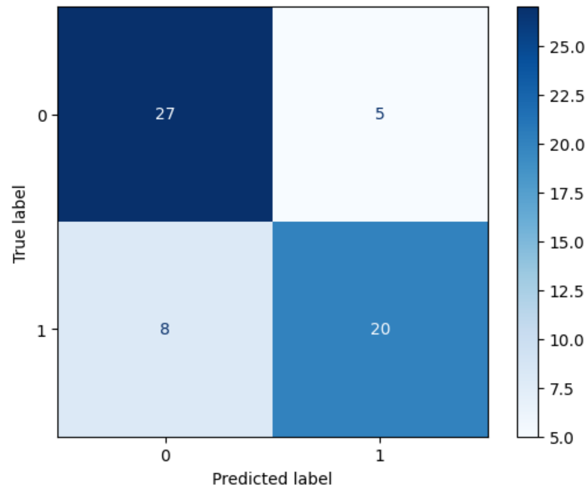
- After that, we get the evaluation results, shown in the table below.

Model	Accuracy	Precision	Recall	F1-score
Logistic Regression	0.91667	0.92793	0.91667	0.91560
Decision Tree	0.78333	0.78476	0.78333	0.78205
k-NN	0.90000	0.91579	0.90000	0.89829
SVM	0.90000	0.91579	0.90000	0.89829

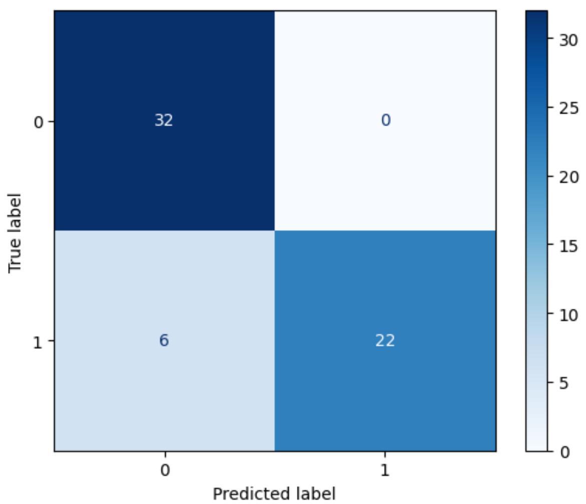
- We use confusion matrices to visualize how many predictions were correct and wrong for each class. As we can see below:



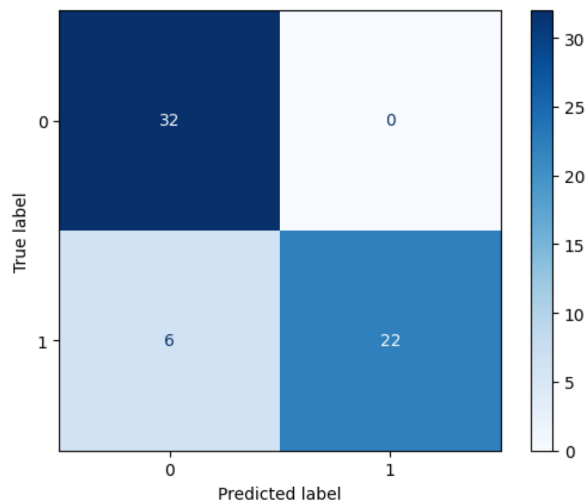
Logistic Regression



Decision Tree

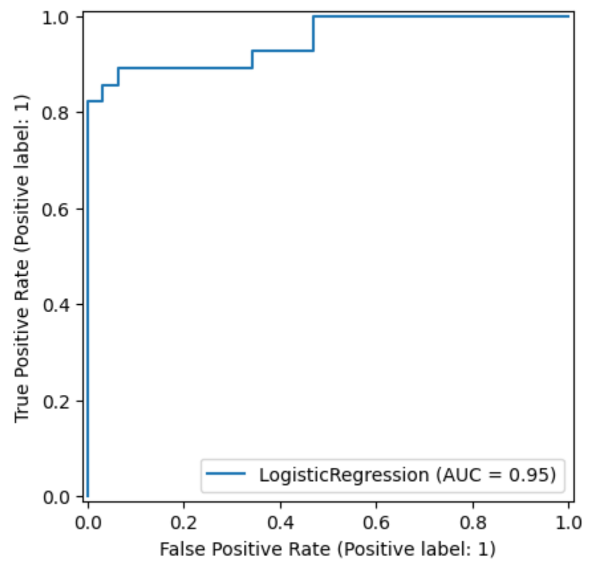


k-NN

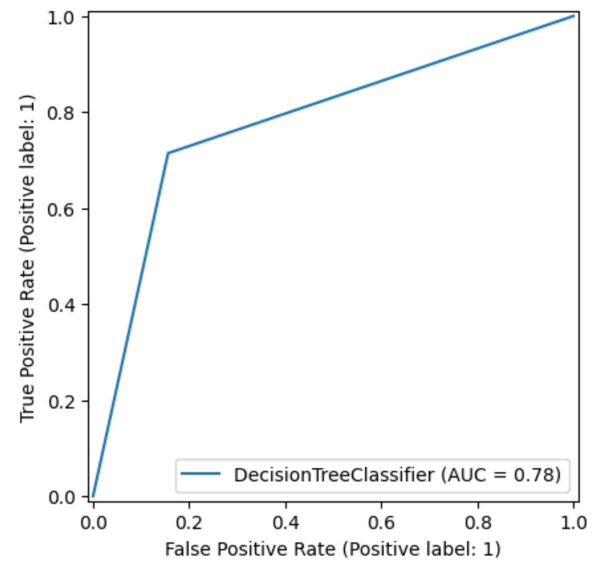


SVM

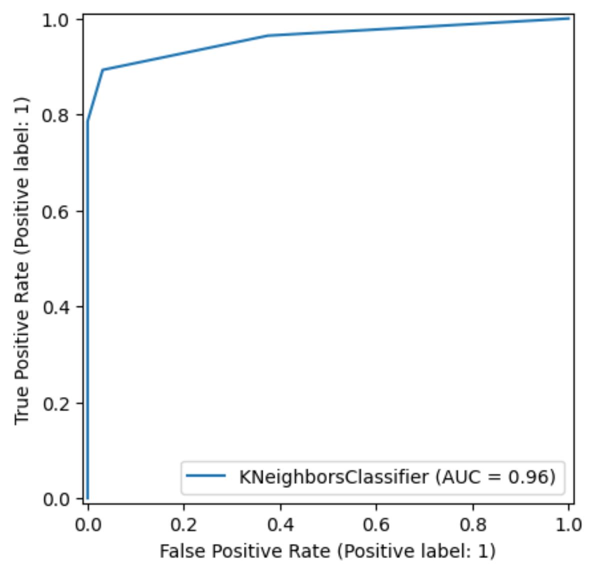
- Also, we plot the ROC curves for all models.
The ROC curve shows the trade-off between **True Positive** Rate and **False Positive** Rate.
We use it here to visualize how well the model separates the two classes.



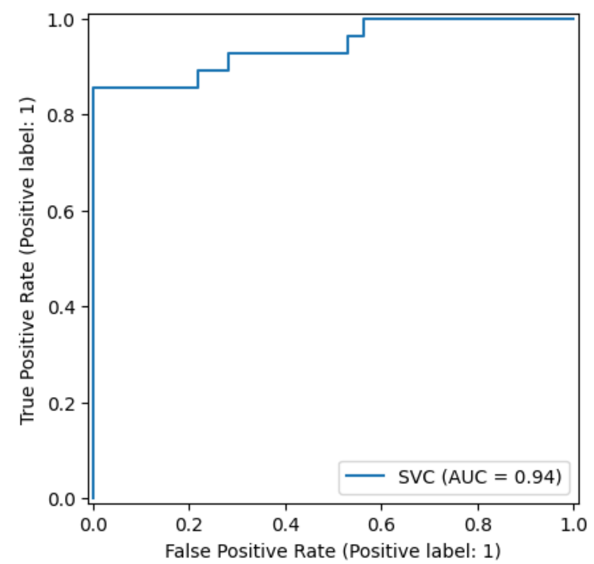
Logistic Regression



Decision Tree



k-NN

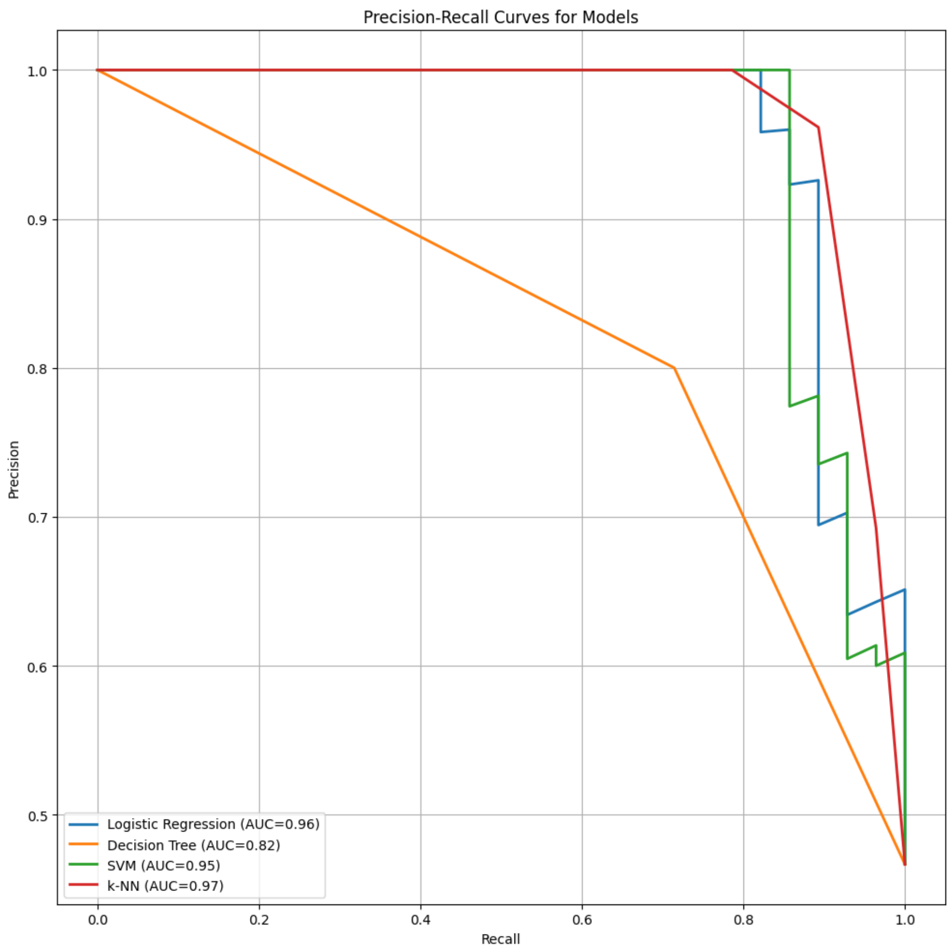


SVC

k-NN (0.96) and Logistic Regression (0.95) performed best.
SVM (0.94) is close behind.
Decision Tree (0.78) performed worst.

Overall: k-NN, Logistic, and SVM are good , Decision Tree is weaker.

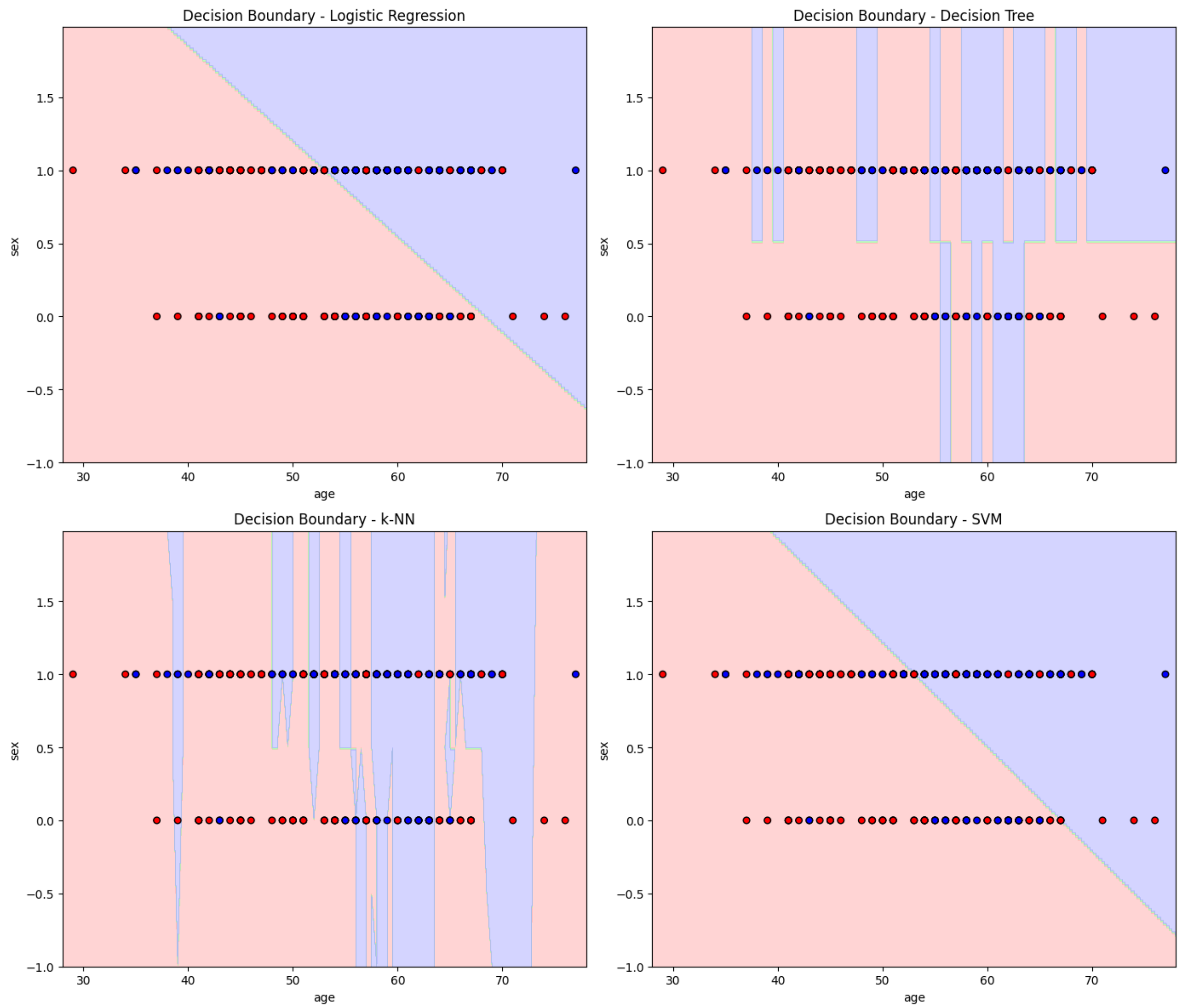
- We also compute and plot **the Precision-Recall** curves for all models.
Precision-Recall focuses on the positive class performance.
It lets us understand how well the model finds positive cases without too many false alarms, as we can see in the figure.



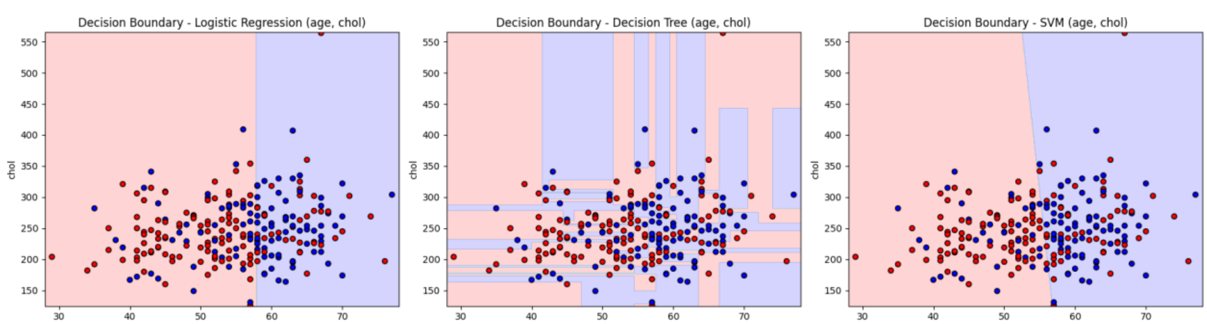
k-NN (0.97) and Logistic Regression (0.96) performed best.
SVM (0.95) also strong.
Decision Tree (0.82) weakest.

Overall: nearly same for k-NN, Logistic, and SVM good, Decision Tree weaker.

- As a final step, we visualized some features vs target and features vs features in a 2D plane and drew the decision boundaries between the classes.
As we can see below:



We removed **k-NN** from this part because it took too long to compute (around **20 minutes**)!!!. and that's because k-NN needs to compare each test point to all training points, which is slow for large or complex data.



6. Conclusion

From the results, **Logistic Regression** performed the best overall, with the highest **accuracy (91.67%)**, **F1-score (91.56%)**, and **AUC (0.95)**. It also showed a clean confusion matrix with only 5 false negatives.

k-NN and **SVM** gave very similar good results (**accuracy 90%**, **AUC ~0.94-0.96**), but **k-NN was slow to compute** for the decision boundary plots, so it was excluded there.

Decision Tree had the weakest performance (**accuracy 78.33%**, **AUC 0.78**), misclassifying more cases in the confusion matrix.

In summary, Logistic Regression is the best model for this dataset, being simple, fast, and highly accurate.