Predicting Heart Disease Presence Through Clinical Results

Joshua Ewer

DSC680 - Applied Data Science

2026/01/18

Business Problem

Heart disease is a leading cause of mortality worldwide and places ongoing strain on healthcare systems (World Health Organization [WHO], 2023). Physicians often collect a wide range of clinical measurements, including blood pressure, cholesterol levels, stress test results, and electrocardiogram (ECG) reading, but using these indicators as a clear assessment of disease risk can be challenging. The business problem addressed in this project is whether machine learning models can use commonly available clinical data to predict the presence of heart disease with reasonable accuracy. From a healthcare analytics perspective, the value of a model is not in replacing medical judgment, but in supporting clinicians by identifying patients who may need closer attention or additional testing.

Background / History

This project was motivated by both personal experience and analytical relevance. A close family member was recently diagnosed with a heart arrhythmia, and required a pacemaker. At the same time, I was personally diagnosed with high blood pressure. These unrelated (though maybe related through genetics) events highlight how cardiovascular risk can be discovered through routine measurements before more serious outcomes occur.

From a data science standpoint, cardiovascular health is a well-established research area with a large variety of publicly available datasets. Repositories such as the UCI Machine Learning Repository provide structured clinical data suitable for predictive modeling, without the risk of exposing personally identifiable information (Dua & Graff, 2019). This combination of personal relevance, public health importance, and data availability made heart disease prediction an excellent subject for the exploration of machine learning models.

Data Explanation (Data Preparation and Structure)

The dataset used in this analysis is derived from the UCI Heart Disease dataset and accessed via Kaggle. It contains 1,025 patient observations and 14 variables representing clinical and diagnostic measurements commonly used in cardiovascular assessment. The target variable is binary, indicating the presence or absence of heart disease. Independent variables include age, chest pain type, resting blood pressure, cholesterol levels, fasting blood sugar, ECG results, exercise-induced angina, ST depression (oldpeak), thalassemia-related imaging outcomes, number of vessels observed during fluoroscopy, and maximum heart rate achieved.

Initial data preparation confirmed that the dataset contained no missing values or single-value columns. Numeric variables were examined for scale and distribution, while categorical variables were encoded to support modeling. Correlation analysis was conducted to assess relationships between predictors and the target variable, which helped to inform feature selection.

Methods

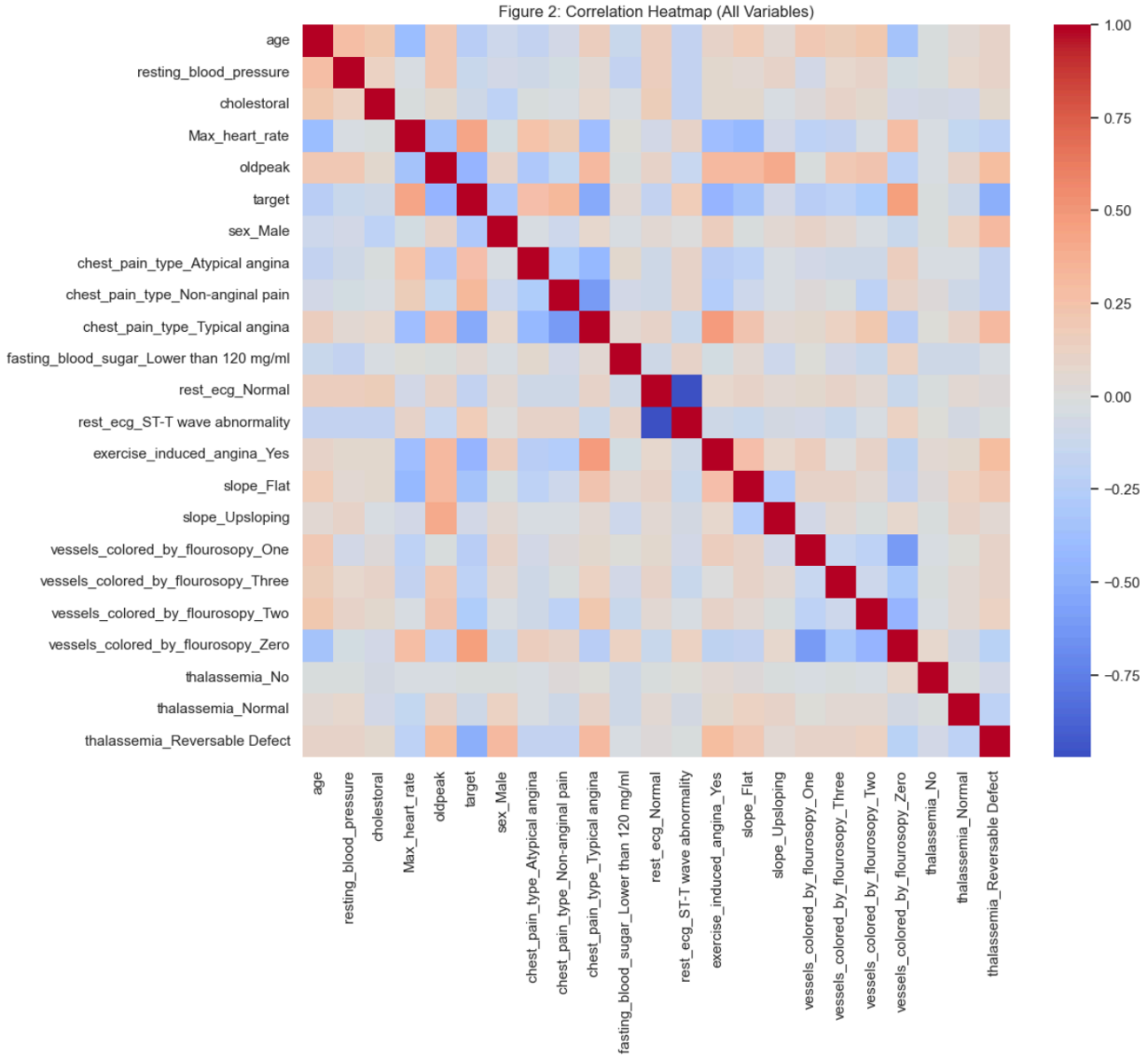
The analytical approach followed a usual data science workflow, focusing on interpretability, model comparison, and careful validation. Exploratory data analysis was first conducted to assess class balance, feature distributions, and potential anomalies. This step ensured that model performance metrics would not be skewed by class imbalance or data leakage. A Logistic Regression was selected as the baseline model because of its interpretability and long-standing use in medical research for binary classification (Hosmer et al., 2013). To better capture complex, non-linear relationships that are often present in clinical data, a Gradient Boosting model was used. Ensemble methods, including Gradient Boosting, are excellent

choices for structured datasets and have been shown to outperform single-model approaches in many predictive tasks (Zhou, 2012).

Model evaluation relied on k-fold cross-validation to reduce overfitting risk, given the relatively small dataset size. Performance was assessed using accuracy, recall, and the area under the receiver operating characteristic curve (ROC-AUC). Recall was emphasized because of false negatives in a healthcare context. While false negatives are not a concern in an experiment like this, in a real world scenario, false negatives represent an almost unpayable cost.

Analysis

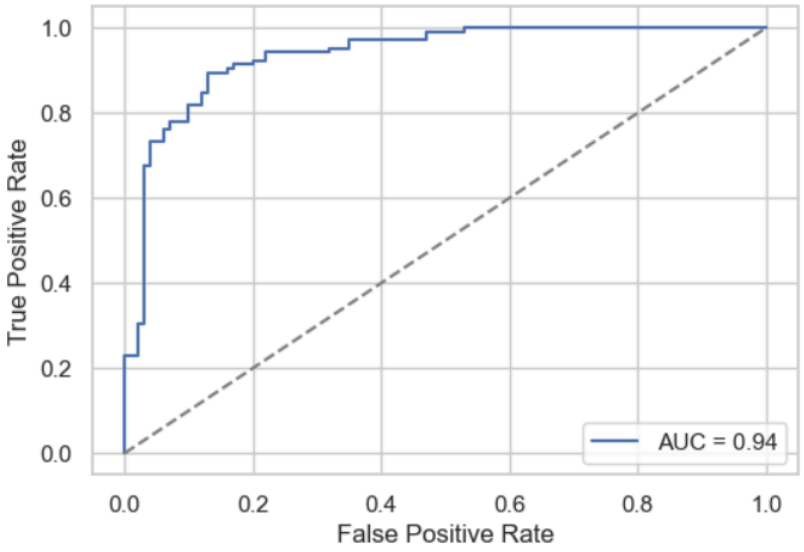
Exploratory analysis confirmed that the target variable was reasonably balanced, reducing the likelihood of misleading accuracy scores. Correlation analysis identified several features with strong relationships to heart disease presence, including typical angina, thalassemia-related imaging defects, exercise-induced angina, ST depression (oldpeak), number of vessels observed



via fluoroscopy, and maximum heart rate achieved. These findings align with established clinical knowledge, since many of these variables are associated with impaired blood flow or abnormal cardiac stress responses.

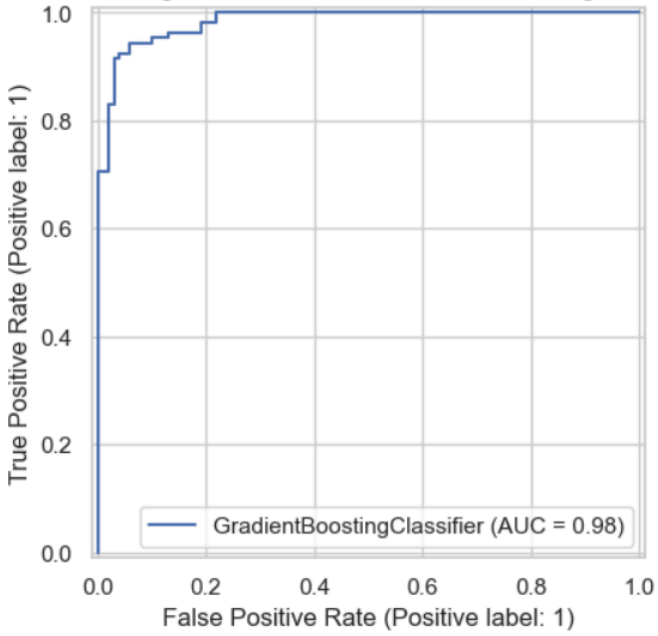
Logistic Regression produced a strong baseline performance, achieving a ROC-AUC of approximately 0.82. The model's coefficients show the importance of the previously identified predictors. It also made the direction and relative magnitude of the features simpler to interpret.

Figure 4: ROC Curve - Logistic Regression



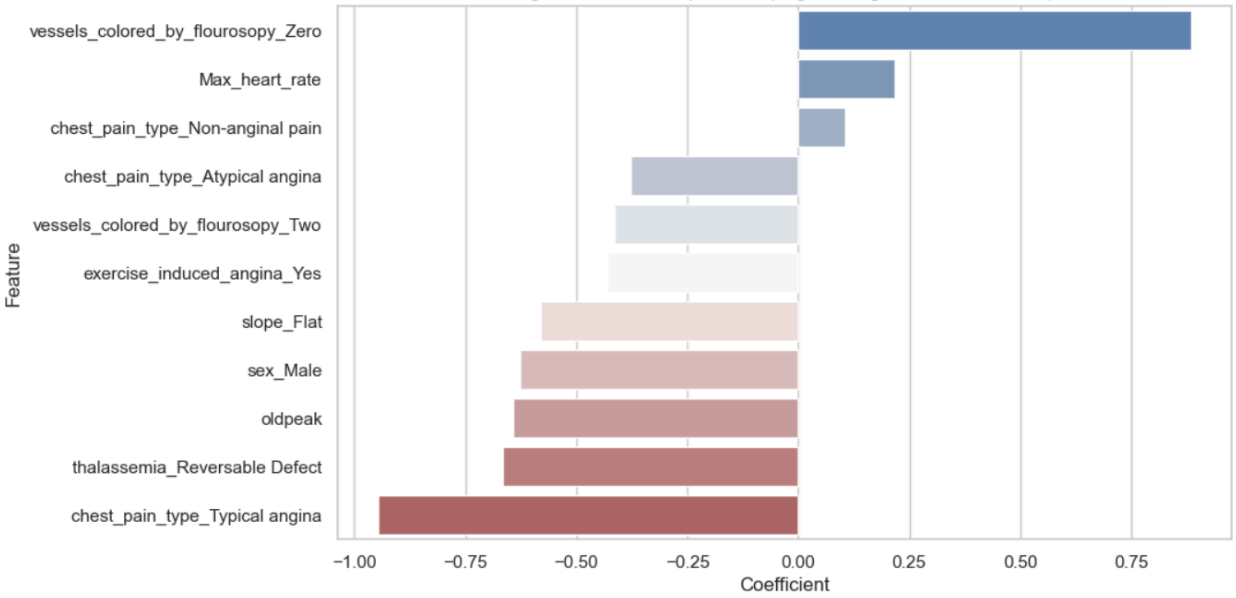
Gradient Boosting substantially improved predictive performance, achieving a ROC-AUC of approximately 0.98 and higher recall.

Figure 6: ROC Curve - Gradient Boosting



This improvement in performance could suggest that non-linear interactions between clinical variables play an important role in heart disease prediction. Feature importance rankings from the ensemble model consistently highlighted chest pain type, oldpeak, and thalassemia-related variables, reinforcing the validity of earlier exploratory findings.

Figure 5: Feature Importance (Logistic Regression Coefficients)



Conclusion

The objective of this project was to show whether commonly collected clinical measurements could be used to predict heart disease presence, and to evaluate how different modeling approaches perform on this task. A relatively small set of clinically meaningful variables provided strong predictive power, and both models demonstrated reasonable performance when appropriate feature selection and validation techniques were applied.

Logistic Regression provided a very interpretable baseline, while Gradient Boosting delivered superior accuracy by capturing complex relationships in the data. These results are consistent with prior research showing that ensemble methods often outperform linear models on structured clinical datasets (Zhou, 2012). While the findings are exploratory, they demonstrate how predictive analytics can meaningfully support healthcare-related decision-making when applied responsibly.

Assumptions

This analysis assumes that the clinical measurements in the dataset were collected accurately and consistently, and that the dataset reasonably represents patients undergoing cardiovascular evaluation. It also assumes that observed correlations reflect genuine physiological relationships rather than artifacts of encoding or sampling.

Limitations

Several limitations affect the generalizability of the results. The dataset lacks detailed demographic and lifestyle variables, limiting insight into how predictions may vary across different populations. The relatively small sample size increases the risk of overfitting, even with cross-validation. For this very important reason, the model outputs should not be interpreted as clinically actionable predictions; they are simply exploratory results.

Challenges

Key challenges included managing overfitting risk, selecting features without introducing bias, and interpreting unexpected correlations such as the relationship between maximum heart

rate and disease presence. Balancing model interpretability with predictive performance was also a central challenge in this analysis.

Future Uses and Additional Applications

With a larger or more diverse dataset, this modeling approach could be extended to preventive screening tools, or continual monitoring of cardiovascular health. Incorporating lifestyle or socioeconomic variables could also help to make the model more accurate when applied to a general population.

Recommendations

For healthcare-related predictive tasks, interpretable models such as Logistic Regression should be used as baselines, with ensemble methods applied when higher accuracy is required. Emphasis should be placed on recall and false-negative reduction, and models should be validated on external datasets before broader application.

Implementation Plan

A practical implementation would involve validating the model on additional datasets, integrating explainability tools, and presenting outputs through dashboards designed for exploratory use. Clear disclaimers should accompany any deployment to ensure predictions are interpreted strictly as decision-support information.

Ethical Assessment

Predictive models trained on limited datasets may reflect underlying demographic or clinical biases, potentially leading to unequal performance across patient groups. False negatives could delay necessary care, while false positives may cause unnecessary anxiety or testing. Consistent with ethical guidance in applied machine learning, this model should be used purely as a support tool and never as a substitute for professional medical judgment (Pedregosa et al., 2011; WHO, 2023).

References

Dua, D., & Graff, C. (2019). *UCI Machine Learning Repository: Heart Disease Data Set*. University of California, Irvine.

Hosmer, D. W., Lemeshow, S., & Sturdivant, R. X. (2013). *Applied logistic regression* (3rd ed.). Wiley.

Pedregosa, F., et al. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research, 12*, 2825–2830.

World Health Organization. (2023). *Cardiovascular diseases (CVDs)*.

Zhou, Z.-H. (2012). *Ensemble methods: Foundations and algorithms*. Chapman & Hall/CRC.

Appendix

Data Preparation, Modeling, Evaluation, and Interpretation

[1]:
```
import warnings
warnings.filterwarnings('ignore')

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import (
    accuracy_score,
    classification_report,
    confusion_matrix,
    roc_auc_score,
    roc_curve
)

# Set up some basic configurations for consistency
pd.set_option("display.max_columns", None)
sns.set(style="whitegrid", palette="deep")
```

[2]:
```
df = pd.read_csv("HeartDiseaseTrain-Test.csv")

print("Dataset shape:", df.shape)
df.head()
```

```
Dataset shape: (1025, 14)
```

[2]:

	age	sex	chest_pain_type	resting_blood_pressure	cholestoral	fasting_blood_sugar	rest_ecg	Max_heart_rate	exercise_induced_angina	oldpeak	slope	vessels_colored_by_flourosopy
0	52	Male	Typical angina	125	212	Lower than 120 mg/ml	ST-T wave abnormality	168	No	1.0	Downsloping	Two
1	53	Male	Typical angina	140	203	Greater than 120 mg/ml	Normal	155				
2	70	Male	Typical angina	145	174	Lower than 120 mg/ml	ST-T wave abnormality	125				
3	61	Male	Typical angina	148	203	Lower than 120 mg/ml	ST-T wave abnormality	161				
4	62	Female	Typical angina	138	294	Greater than 120 mg/ml	ST-T wave abnormality	106				

```
1                    Yes      3.1    Upsloping                         Zero
2                    Yes      2.6    Upsloping                         Zero
3                     No      0.0  Downsloping                          One
4                     No      1.9         Flat                        Three

         thalassemia  target
0  Reversable Defect       0
1  Reversable Defect       0
2  Reversable Defect       0
3  Reversable Defect       0
4       Fixed Defect       0
```

[3]:
```
categorical_cols = df.select_dtypes(include=["object"]).columns.tolist()
numeric_cols = df.select_dtypes(include=[np.number]).columns.tolist()

print("\nCategorical Columns:", categorical_cols)
print("Numeric Columns:", numeric_cols)
```

```
Categorical Columns: ['sex', 'chest_pain_type', 'fasting_blood_sugar',
'rest_ecg', 'exercise_induced_angina', 'slope', 'vessels_colored_by_flourosopy',
'thalassemia']
Numeric Columns: ['age', 'resting_blood_pressure', 'cholestoral',
'Max_heart_rate', 'oldpeak', 'target']
```

[4]:
```
print("\nMissing values per column:\n", df.isnull().sum())
print("\nData types:\n", df.dtypes)
```

```
Missing values per column:
 age                             0
sex                              0
chest_pain_type                  0
resting_blood_pressure           0
cholestoral                      0
fasting_blood_sugar              0
rest_ecg                         0
Max_heart_rate                   0
exercise_induced_angina          0
oldpeak                          0
slope                            0
vessels_colored_by_flourosopy    0
thalassemia                      0
target                           0
dtype: int64

Data types:
 age                               int64
```

```
sex                                 object
chest_pain_type                     object
resting_blood_pressure               int64
cholestoral                          int64
fasting_blood_sugar                 object
rest_ecg                            object
Max_heart_rate                       int64
exercise_induced_angina             object
oldpeak                            float64
slope                               object
vessels_colored_by_flourosopy       object
thalassemia                         object
target                               int64
dtype: object
```

Initial analysis of the data shows the following characteristics:

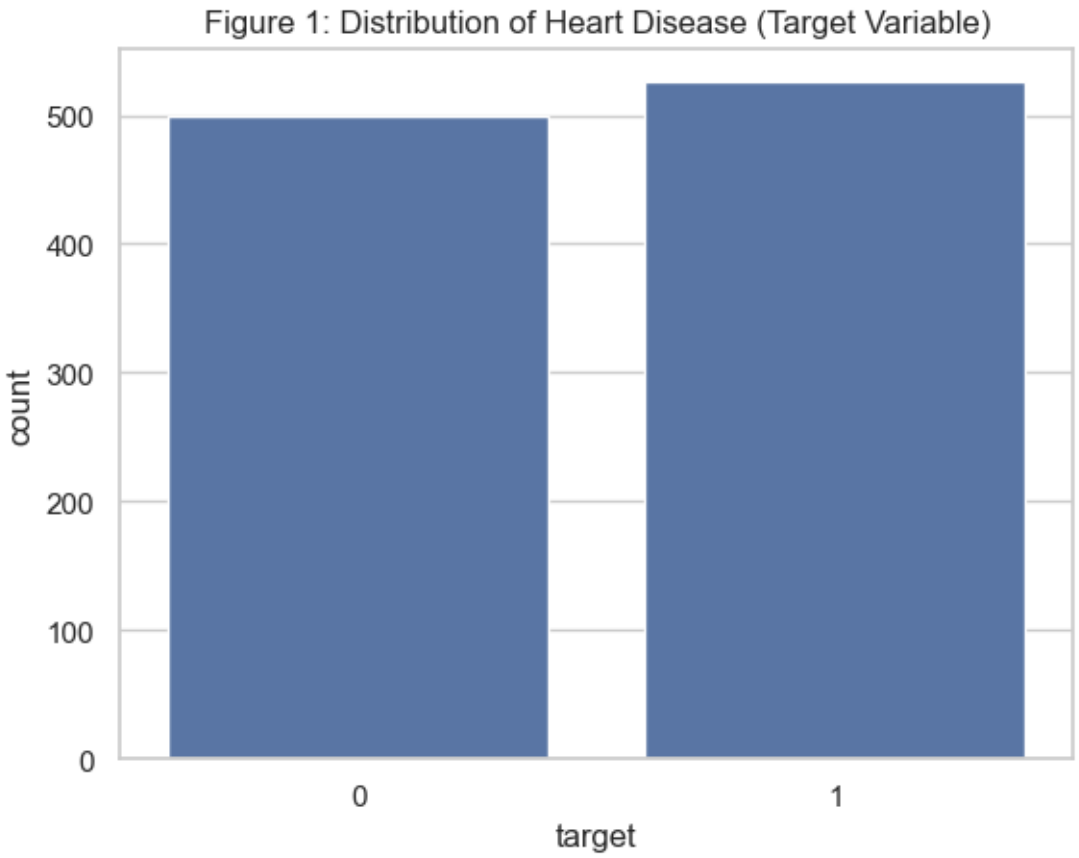
- There are no columns with missing data, so this data set looks to be of high quality. I will not need to take any further action to deal with bad data
- There is a mix of categorical and numeric columns, so I will need to encode some of the columns before I can use them in my model

[5]:
```
# Get dummy variables for categorical values (one-hot encoding)
df_encoded = pd.get_dummies(df, drop_first=True)
```

Exploratory Data Analysis

My exploratory data analysis will dig deeper into the data to ensure I do not need to modify or impute any values to handle skewed data or prevent class imbalances

[6]:
```
sns.countplot(x="target", data=df_encoded)
plt.title("Figure 1: Distribution of Heart Disease (Target Variable)")
plt.show()
```



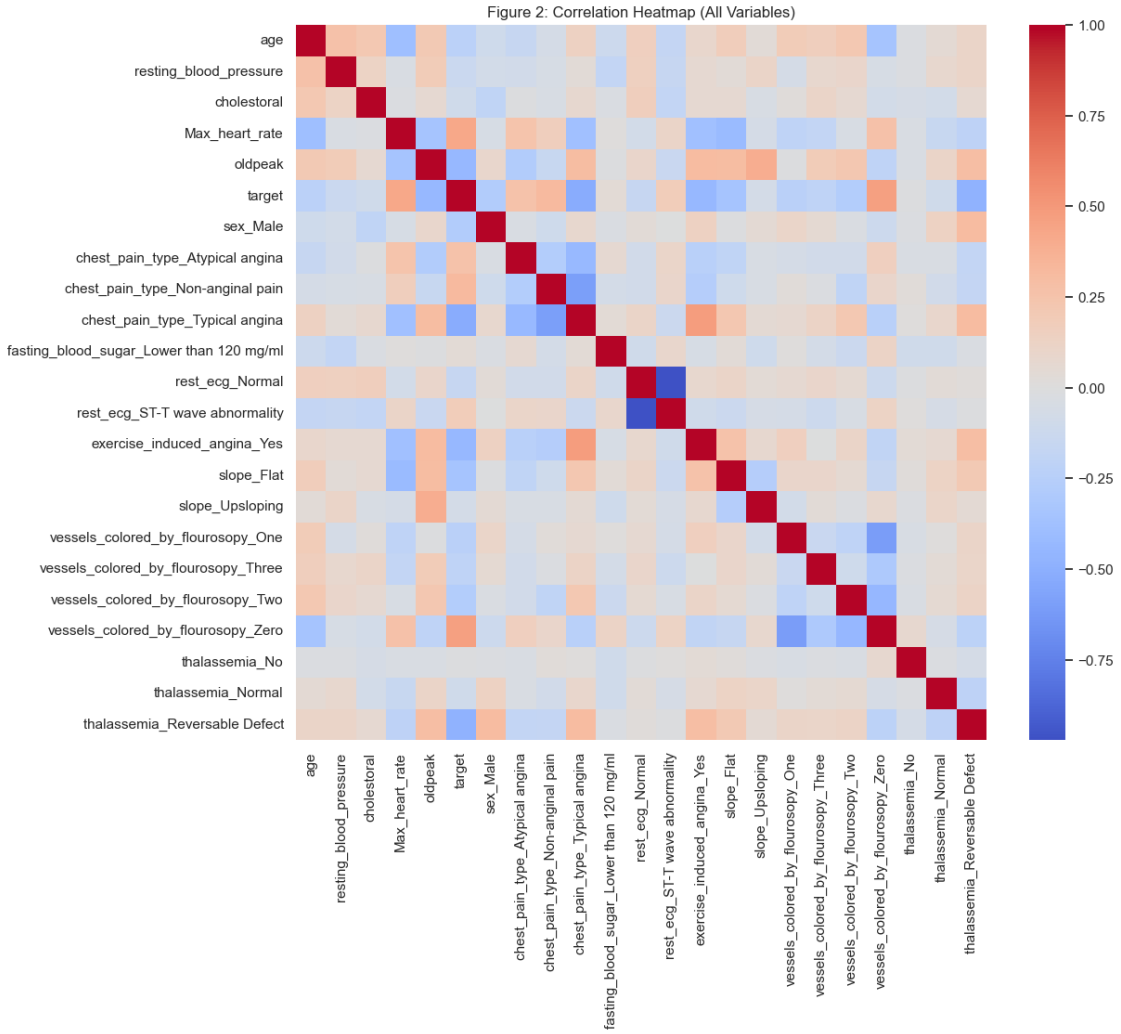
The target value looks to have a fair distribution so I don't need to worry about class imbalances or misleading accuracy metrics.

Encode the categorical values so that I can see which values have high correlation with our target variable

[7]:
```
plt.figure(figsize=(12,10))
corr = df_encoded.corr()

top_features = corr["target"].abs().sort_values(ascending=False).head(11).
  ↪drop("target")
print("Top 10 most correlated features with the target variable:\n")
print(top_features)

sns.heatmap(corr, cmap="coolwarm", center=0)
plt.title("Figure 2: Correlation Heatmap (All Variables)")
plt.show()
```

```
Top 10 most correlated features with the target variable:

chest_pain_type_Typical angina        0.519621
```

```
thalassemia_Reversable Defect          0.479709
vessels_colored_by_flourosopy_Zero     0.465981
oldpeak                                0.438441
exercise_induced_angina_Yes            0.438029
Max_heart_rate                         0.422895
slope_Flat                             0.349417
chest_pain_type_Non-anginal pain       0.319504
sex_Male                               0.279501
vessels_colored_by_flourosopy_Two      0.276566
Name: target, dtype: float64
```



Results from exploration (including LOTs of research into heart disease testing and what all the symptoms and terminology means):

The correlation analysis found several features in the dataset that have strong correlation with heart disease.

- The variable `chest_pain_type_Typical` angina showed the highest positive correlation (0.52) with the target variable, which suggests that individuals who experience typical angina are more likely to have underlying heart disease. Angina is a common symptom of restricted blood flow to the heart.
- `thalassemia_Reversable Defect` (0.48), suggests that patients with reversible thalassemia defects (abnormal results from cardiac imaging tests) are more likely to have heart disease. The next closest correlated variable, `vessels_colored_by_flourosopy_Zero` (0.47), implies that individuals showing no visible blood vessels during fluoroscopy (which an indicator of possible blockage) also tend to have higher disease likelihood.
- The next two variables are related to exercise and stress testing. The continuous variable `oldpeak` (0.44) measures ECG abnormalities and suggests higher disease risk. The categorical variable `exercise_induced_angina_Yes` (0.44) supports this pattern—those who develop angina during exercise are more likely to have heart disease.
- The last strongly correlated variable is `Max_heart_rate` (0.42) showed a positive correlation, which might suggest that a higher maximum heart rate is associated with heart disease. In my research, I noticed that this relationship can often be negative, so (pending time) I will dig into this further to see if this is an issue with encoding or the dataset, or is a true representation of correlation.

There are other features that have moderate correlation will be kept in the model, though their correlation may be too weak to be meaningful: - `slope_Flat` (0.35), another exercise test showing a flat ECG slope - `chest_pain_type_Non-anginal pain` (0.32), is a self-reported value, which tend to be less reliable - `sex_Male` (0.28) confirms that males in this dataset have higher disease prevalence, which is consistent with well-known demographic characteristics - `vessels_colored_by_flourosopy_Two` (0.28) has a weaker or non-linear correlation than the similar highly corrrelated variable noted above.

Overall, these patterns align with well-known cardiovascular risk indicators (especially those related to chest pain characteristics, ECG abnormalities, and imaging results) so the initial exploration confirms that this data is realistic and useful for modeling. I will keep only the strong features in the features used in the model.

Feature Selection and Model training

[8]:
```
threshold = 0.25  # cutoff to only get

strong_features = corr["target"][abs(corr["target"]) >= threshold].index.
 ↪tolist()
strong_features.remove("target")
print(f"Number of strong features: {len(strong_features)}")
print("Selected features:\n", strong_features)

df_strong = df_encoded[strong_features + ["target"]]
print("\nNew dataset shape:", df_strong.shape)
```

```
Number of strong features: 11
Selected features:
```

```
 ['Max_heart_rate', 'oldpeak', 'sex_Male', 'chest_pain_type_Atypical angina',
'chest_pain_type_Non-anginal pain', 'chest_pain_type_Typical angina',
'exercise_induced_angina_Yes', 'slope_Flat',
'vessels_colored_by_flourosopy_Two', 'vessels_colored_by_flourosopy_Zero',
'thalassemia_Reversable Defect']

New dataset shape: (1025, 12)
```

[9]:
```
# Split features and target
X = df_strong.drop("target", axis=1)
y = df_strong["target"]

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, stratify=y, random_state=42
)

# Scale numeric features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

A logistic regression is a simple place to start, though I plan on using a boosted algorithm because they tend to work well for tabular data (like medical test results)

[10]:
```
log_reg = LogisticRegression(max_iter=1000, random_state=42)
log_reg.fit(X_train_scaled, y_train)
y_pred = log_reg.predict(X_test_scaled)
y_prob = log_reg.predict_proba(X_test_scaled)[:, 1]
```

Model Evaluation and Conclusions

[11]:
```
# Accuracy and report
accuracy = accuracy_score(y_test, y_pred)
auc = roc_auc_score(y_test, y_prob)
print(f"\nAccuracy: {accuracy:.3f}")
print(f"AUC: {auc:.3f}")
print("\nClassification Report:\n", classification_report(y_test, y_pred))

# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6,4))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")
plt.title("Figure 3: Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```

```python
# ROC Curve
fpr, tpr, thresholds = roc_curve(y_test, y_prob)
plt.figure(figsize=(6,4))
plt.plot(fpr, tpr, label=f"AUC = {auc:.2f}")
plt.plot([0,1], [0,1], linestyle="--", color="gray")
plt.title("Figure 4: ROC Curve - Logistic Regression")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.legend()
plt.show()

cv_auc = cross_val_score(log_reg, X_train_scaled, y_train, cv=5,␣
 ↪scoring="roc_auc")
print(f"Average CV AUC: {cv_auc.mean():.3f}")
```

```
Accuracy: 0.873
AUC: 0.936

Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.84      0.87       100
           1       0.86      0.90      0.88       105

    accuracy                           0.87       205
   macro avg       0.87      0.87      0.87       205
weighted avg       0.87      0.87      0.87       205
```

Figure 3: Confusion Matrix

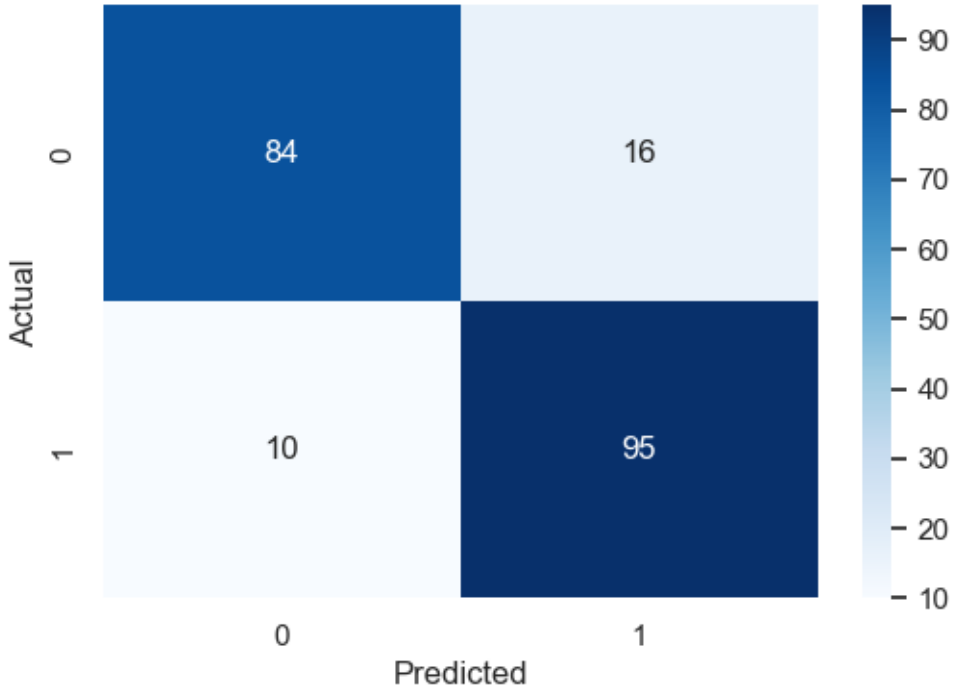
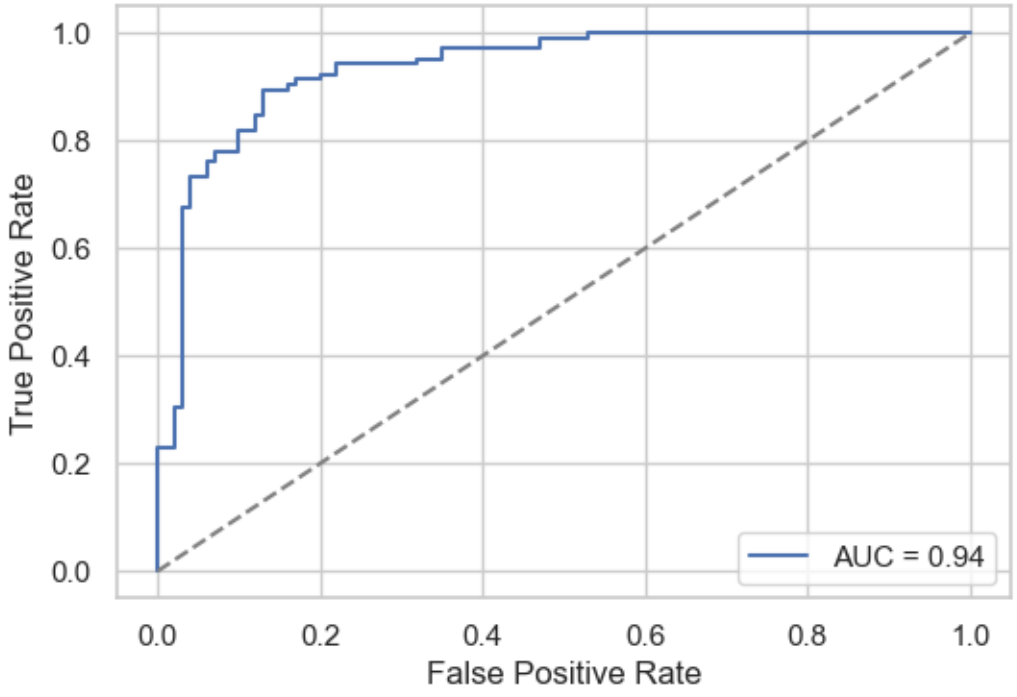


Figure 4: ROC Curve - Logistic Regression

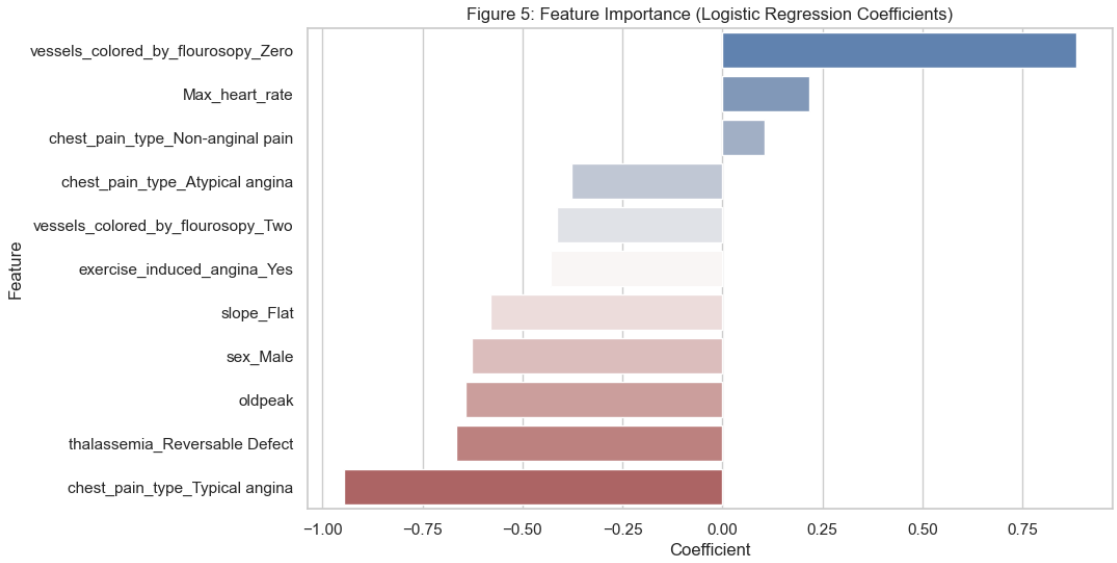


```
Average CV AUC: 0.932
```

The logistic regression shows an AUC 0.94, which means this model can correctly rank a randomly chosen heart disease patient higher than a randomly chosen healthy patient 94% of the time, which is *very* good performance for a medical classification problem. I mentioned the concern about overfitting above, so cross-validation (or k-fold) may help to make sure we're not overfitting.

I also want to make sure that the features selected are actually important.

[12]:
```
importance = pd.DataFrame({
    "Feature": X.columns,
    "Coefficient": log_reg.coef_[0]
}).sort_values(by='Coefficient', ascending=False)

plt.figure(figsize=(10,6))
sns.barplot(x="Coefficient", y="Feature", data=importance, palette="vlag",␣
 ↪hue="Feature")
plt.title("Figure 5: Feature Importance (Logistic Regression Coefficients)")
plt.show()

print("\nTop 5 positive predictors:\n", importance.head())
print("\nTop 5 negative predictors:\n", importance.tail())
```




```
Top 5 positive predictors:
                              Feature  Coefficient
9  vessels_colored_by_flourosopy_Zero     0.883677
0                       Max_heart_rate     0.217071
4    chest_pain_type_Non-anginal pain     0.105380
3     chest_pain_type_Atypical angina    -0.376664
```

```
8   vessels_colored_by_flourosopy_Two    -0.413367

Top 5 negative predictors:
                            Feature  Coefficient
7                       slope_Flat    -0.580790
2                         sex_Male    -0.628162
1                          oldpeak    -0.641473
10   thalassemia_Reversable Defect    -0.665211
5   chest_pain_type_Typical angina    -0.946265
```

The logistic regression model achieved very good accuracy (concerns about overfitting, so that will require more confirmation) and AUC, suggesting this model has reliable predictive power for identifying heart disease. Key predictors included chest pain type, maximum heart rate, and exercise-induced angina.

Future improvements could include: - Testing a Gradient Boosting model - Hyperparameter tuning - Running a similar analysis and modeling process on a data that that incorporates demographic or lifestyle data (to get better generalization

[13]:
```
#### Comparing a Gradient Boosting algorithm
```

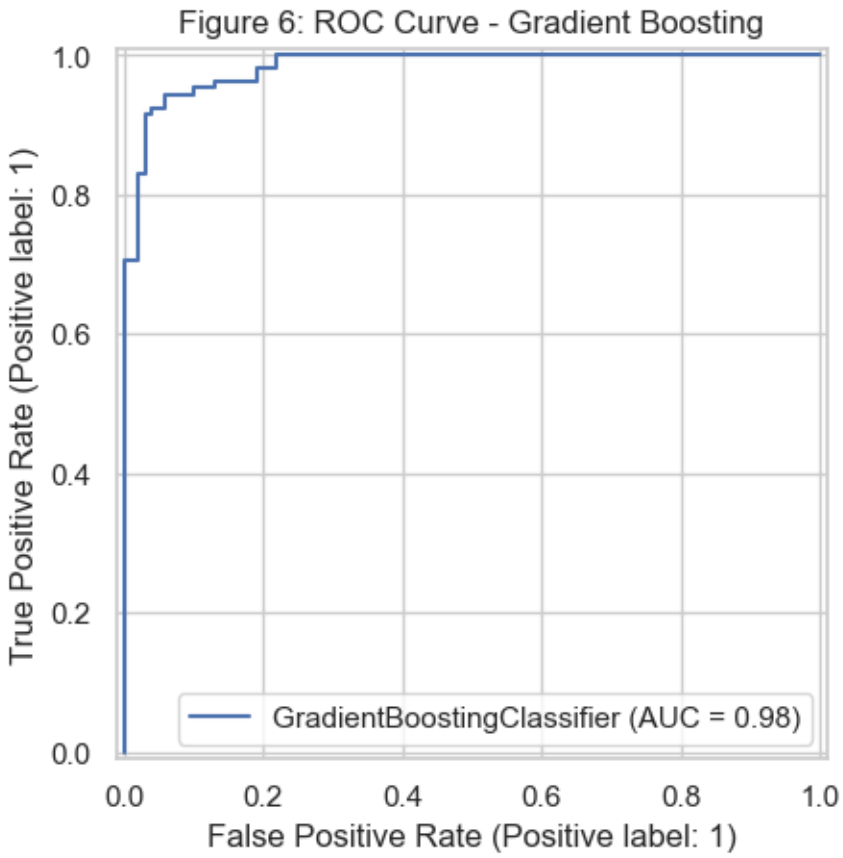
I'll begin with simple hyperparameters that can balance bias and variance well.

- A max depth of 3 will keep the tree complexity shallow enough to reduce overfitting but can still capture interactions between features.
- Lowering the learning rate to 0.05 will help reduce the risk of overfitting. A smaller learning rate can help the model learn gradually instead of overfitting based on subtle patterns.
- I needed to increase the number of boosting rounds to 200 because too small of a number with that slower learning rate can often underfit a model. This should still be small enough not to blow up the model's complexity, but deal with the slower learning rate.

[14]:
```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import classification_report, confusion_matrix,␣
 ↪roc_auc_score
from sklearn.metrics import RocCurveDisplay

gb = GradientBoostingClassifier(
    n_estimators=200,
    learning_rate=0.05,
    max_depth=3,
    random_state=42
)

gb.fit(X_train, y_train)
gb_pred = gb.predict(X_test)
gb_pred_prob = gb.predict_proba(X_test)[:, 1]

print("=== Gradient Boosting Classifier Report ===")
print(classification_report(y_test, gb_pred))
```

```python
print("AUC:", roc_auc_score(y_test, gb_pred_prob))

# Confusion Matrix
cm = confusion_matrix(y_test, gb_pred)
print("Confusion Matrix:")
print(cm)

# ROC Curve
RocCurveDisplay.from_estimator(gb, X_test, y_test)
plt.title("Figure 6: ROC Curve - Gradient Boosting")
plt.show()
```

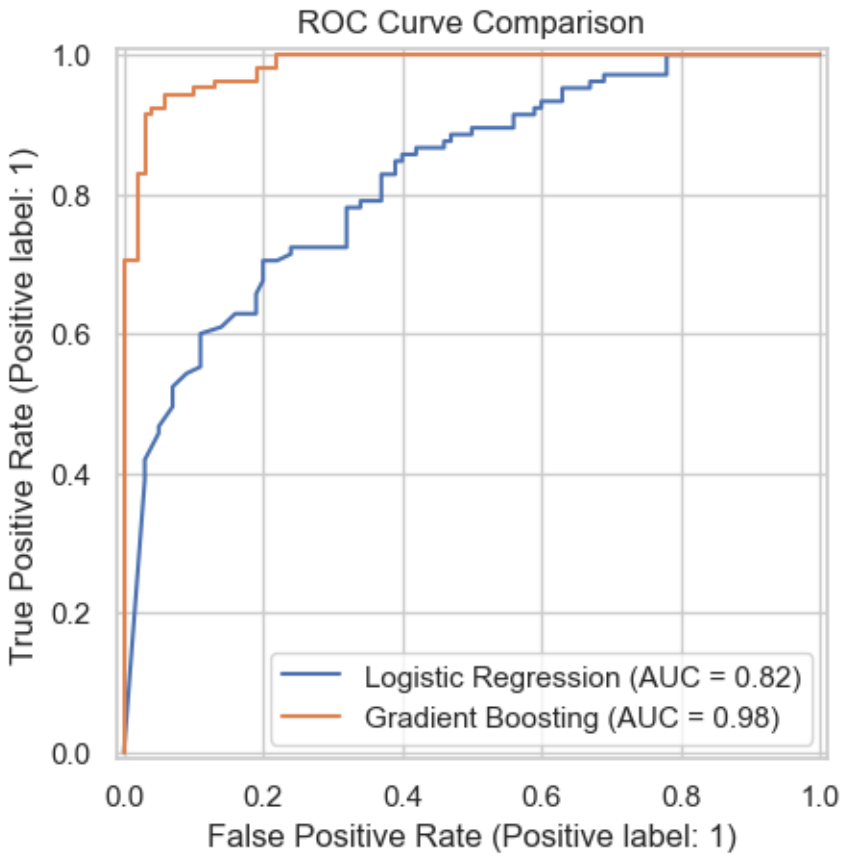
```
=== Gradient Boosting Classifier Report ===
              precision    recall  f1-score   support

           0       0.94      0.94      0.94       100
           1       0.94      0.94      0.94       105

    accuracy                           0.94       205
   macro avg       0.94      0.94      0.94       205
weighted avg       0.94      0.94      0.94       205

AUC: 0.9834285714285714
Confusion Matrix:
[[94  6]
 [ 6 99]]
```



[15]:
```
log_reg_pred = log_reg.predict(X_test)
log_reg_pred_prob = log_reg.predict_proba(X_test)[:, 1]
print("Logistic Regression AUC:", roc_auc_score(y_test, log_reg_pred_prob))
print("Gradient Boosting AUC:", roc_auc_score(y_test, gb_pred_prob))
```

```
Logistic Regression AUC: 0.8227619047619047
Gradient Boosting AUC: 0.9834285714285714
```

[16]:
```
fig, ax = plt.subplots()

RocCurveDisplay.from_predictions(
    y_test, log_reg_pred_prob, name="Logistic Regression", ax=ax
)

RocCurveDisplay.from_predictions(
    y_test, gb_pred_prob, name="Gradient Boosting", ax=ax
)

plt.title("ROC Curve Comparison")
plt.show()
```



I chose to explore Gradient Boosting because it often outperforms a logistic regression. This is because it is able to handle complex relationships between variables, and captures non-linear interactions automatically. The means it can often achieve a higher AUC and better recall in imbalanced problems like predicting heart disease.

The regression model is a good baseline, but gradient boosting demonstrates what a modern model can do on the same kind of data.

[ ]: