Classifying patients at risk for heart failure using clinical data

By Mara Sánchez, Eric Yang & Omar Ramos 2025/11/21

In [1]:
```
import requests
import warnings
import pandas as pd
import altair_ally as aly
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.pipeline import make_pipeline
from sklearn.compose import make_column_transformer
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.model_selection import cross_validate, train_test_split
from sklearn.dummy import DummyClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay, classification_report
```

Summary

In this analysis, we explored various classification models with the intent of predicting whether a patient is at risk of heart failure based on clinical data and lifestyle factors of individuals. After evaluating multiple models through cross-validation, we selected Logistic Regression as our final model due to its overall superior performance across classification metrics. The model demonstrated promising results on the unseen test set, with an accuracy of 86% and F1-scores of 0.88 for the positive class (at risk) and 0.84 for the negative class (not at risk). From the 276 observations in the test set, the model correctly identified 144 cases at risk and 97 not at risk, reporting 23 false positives and 12 false negatives (cases predicted as not at risk when there is risk). Although the scores are encouraging for a first iteration, there is room for improvement to optimize the hyperparameters and the model's threshold settings to minimize false negative cases, which are critical in medical applications. Overall, this model shows potential to support clinical professionals in the assessment of patients during screening.

Introduction

Cardiovascular diseases (CVDs) represent the leading cause of death worldwide, responsible for an estimate of 19.8 million deaths in 2022 and accounting for one-third of all deaths globally in people under the age of 70 (World Health Organization, 2025)). Most CVDs have proven to have a big correlation with an individual's behavior, habits and environment (American Heart Association, 2025). Heart Failure (HF) is a multi-faceted and life-threatening syndrome where the heart's ability to pump and/or fill with blood is reduced, it is estimated to affect more than 64 million people globally (Savarese *et al.*, 2022). Risk factors such as high blood pressure, high blood glucose levels or pre-existing diseases require early intervention to avoid the risk of developing heart failure and other complications. Since many of these risk factors are clinically measurable and changeable, early intervention is highly possible and a critical opportunity in contributing to a patient's well-being.

This issue leads to the question of whether a machine learning (ML) model could reliably classify patients as 'at-risk' or 'not-at-risk' for heart failure based on clinical and lifestyle features. The study of this question is important because traditional risk assessment methods tend to overlook the variability and the complexity of the risk factors, meaning that subjectivity from healthcare professionals could also impact the outcome of the assessment (Barnett *et al.*, 2020). Missing the early detection of risk of HF could lead to chronic and progressive conditions associated with increase in mortality and decrease in quality of life. Additionally, ML algorithms could offer significant advantages in predicting risk of HF, when integrating it to clinical practice it could allow for a more personalized treatment for the patient (Kokori *et al.*, 2025).

Methods

The dataset used in this project is pulled from a repository of the University of Minho, Portugal. The dataset was created by Federico Soriano Palacios (2021), it integrates five different heart-related datasets combined over 11 common features that can be used to predict a possible heart disease. The five data sets are part of the "Heart Disease" dataset (Janosi *et al.*, 1989) that can be found in the UCI Machine Learning Repository that is originally sourced from the Hungarian Institute of Cardiology, the

University Hospital of Zurich, the University Hospital of Basel, the V.A. Medical Center of Long Beach and Cleveland Clinic Foundation. Each row of the dataset contains 11 attributes that describe the patient's age, sex, chest pain type, resting blood pressure, serum cholesterol, fasting blood sugar, resting ECG result, maximum heart rate achieved, exercise induced angina, ST depression induced by exercise relative to rest, slope of the peak exercise ST segment, and the presence or absence of heart disease.

To build the classification model for the prediction of heart failure risk, an exploratory data analysis was conducted. Considering the distribution of the data, the features and their correlation to the target variable, it was decided to include all of them in the model with some standardization to improve results. Data was split into 70% for the training set and 30% for the test set. Different approaches were taken to evaluate which model would have the best accuracy in classification. We trained five different models, including Decision Tree, k-nearest neighbors (k-nn), Support Vector Machine (SVC) with an RBF kernel, Logistic Regression, and a Dummy Classifier (baseline). Each of the models were evaluated using a 5-fold cross-validation strategy. Model performance was evaluated using metrics such as accuracy, precision, recall, and F1-score. The model with the highest performance scores across all metrics is the Logistic Regression which is the model that will be used for the rest of the project. The Python programming language (Van Rossum and Drake 2009) and the following Python packages were used to perform the analysis: NumPy (Harris et al. 2020), pandas (McKinney 2010), Matplotlib (Hunter 2007), seaborn (Waskom 2021), Altair (VanderPlas et al. 2018), scikit-learn (Pedregosa et al. 2011), and requests (Reitz 2011).

Results & Discussion

We conducted some initial exploratory data analysis by observing the number of observations, data types of the features, and checked for missing values. We also inspected the categorical counts for some features to better understand their distribution. The distribution of the target variable was observed to be balanced with approximately equal representation of both classes.

To evaluate the usefulness of each feature to predict the heart disease, we investigated the distribution of each feature with respect to the target variable. We also visualized the correlation between features using a heatmap to identify any strong relationships.

In [2]:
```
url = "https://epl.di.uminho.pt/~jcr/AULAS/ATP2021/datasets/heart.csv"

response = requests.get(url)

with open("../data/raw/heart.csv", "wb") as f:
    f.write(response.content)
```

In [3]:
```
df = pd.read_csv('../data/raw/heart.csv')
df
```

Out[3]:

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope
0	40	M	ATA	140	289	0	Normal	172	N	0.0	Up
1	49	F	NAP	160	180	0	Normal	156	N	1.0	Flat
2	37	M	ATA	130	283	0	ST	98	N	0.0	Up
3	48	F	ASY	138	214	0	Normal	108	Y	1.5	Flat
4	54	M	NAP	150	195	0	Normal	122	N	0.0	Up
...	...	...	...	...	...	...	...	...	...	...	...
913	45	M	TA	110	264	0	Normal	132	N	1.2	Flat
914	68	M	ASY	144	193	1	Normal	141	N	3.4	Flat
915	57	M	ASY	130	131	0	Normal	115	Y	1.2	Flat
916	57	F	ATA	130	236	0	LVH	174	N	0.0	Flat
917	38	M	NAP	138	175	0	Normal	173	N	0.0	Up

918 rows × 12 columns

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 918 entries, 0 to 917
Data columns (total 12 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   Age             918 non-null    int64  
 1   Sex             918 non-null    object 
 2   ChestPainType   918 non-null    object 
 3   RestingBP       918 non-null    int64  
 4   Cholesterol     918 non-null    int64  
 5   FastingBS       918 non-null    int64  
 6   RestingECG      918 non-null    object 
 7   MaxHR           918 non-null    int64  
 8   ExerciseAngina  918 non-null    object 
 9   Oldpeak         918 non-null    float64
 10  ST_Slope        918 non-null    object 
 11  HeartDisease    918 non-null    int64  
dtypes: float64(1), int64(6), object(5)
memory usage: 86.2+ KB
```

In [5]:
```
df.describe()
```

Out[5]:

	Age	RestingBP	Cholesterol	FastingBS	MaxHR	Oldpeak	HeartDisease
count	918.000000	918.000000	918.000000	918.000000	918.000000	918.000000	918.000000
mean	53.510893	132.396514	198.799564	0.233115	136.809368	0.887364	0.553377
std	9.432617	18.514154	109.384145	0.423046	25.460334	1.066570	0.497414
min	28.000000	0.000000	0.000000	0.000000	60.000000	-2.600000	0.000000
25%	47.000000	120.000000	173.250000	0.000000	120.000000	0.000000	0.000000
50%	54.000000	130.000000	223.000000	0.000000	138.000000	0.600000	1.000000
75%	60.000000	140.000000	267.000000	0.000000	156.000000	1.500000	1.000000
max	77.000000	200.000000	603.000000	1.000000	202.000000	6.200000	1.000000

In [6]:
```
df.isna().sum()
```

Out[6]:
```
Age               0
Sex               0
ChestPainType     0
RestingBP         0
Cholesterol       0
FastingBS         0
RestingECG        0
MaxHR             0
ExerciseAngina    0
Oldpeak           0
ST_Slope          0
HeartDisease      0
dtype: int64
```

In [7]:
```
df['ChestPainType'].value_counts()
```

Out[7]:
```
ChestPainType
ASY    496
NAP    203
ATA    173
TA      46
Name: count, dtype: int64
```

In [8]:
```
df['ST_Slope'].value_counts()
```

Out[8]:
```
ST_Slope
Flat    460
Up      395
Down     63
Name: count, dtype: int64
```

In [9]:
```
df['HeartDisease'] = df['HeartDisease'].astype('bool')
```

In [10]:
```
warnings.filterwarnings("ignore", module="altair")

aly.alt.data_transformers.enable('vegafusion')
```

```
aly.dist(df, color='HeartDisease')
```

Out[10]:

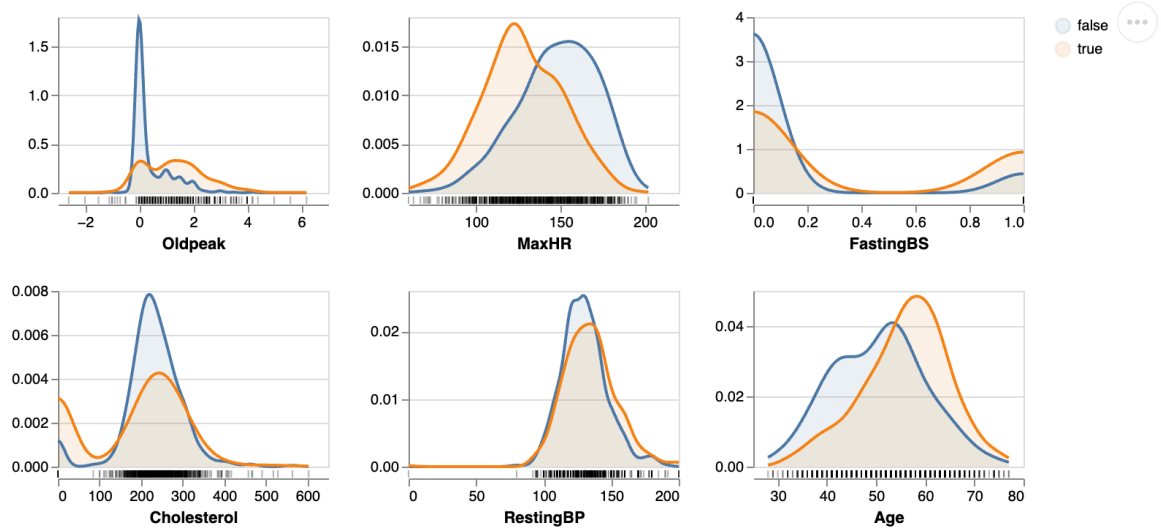


Figure 1. Comparison of numerical predictors, coloured by heart disease status.

In [11]:
```
aly.dist(df.assign(HeartDisease=lambda df: df['HeartDisease'].astype(object)),
         dtype='object', color='HeartDisease')
```

Out[11]:

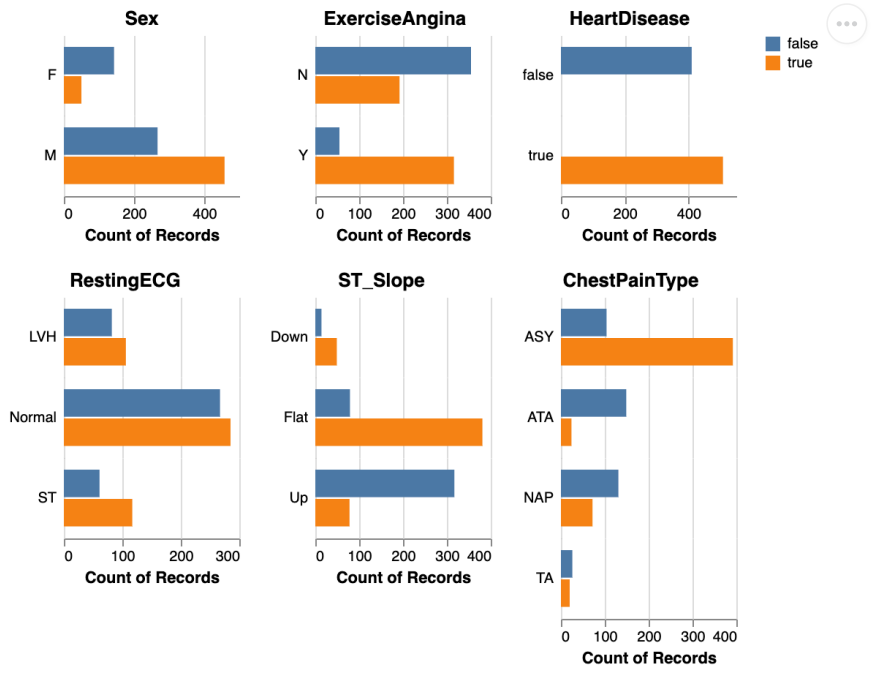


Figure 2. Distribution of categorical predictors, coloured by heart disease status.

In [12]:
```
corr = df.corr(numeric_only=True)

plt.figure(figsize=(10, 6))
sns.heatmap(
    corr,
    annot=True,
    fmt=".2f",
    cmap="coolwarm",
    vmin=-1, vmax=1,
    square=True
)

plt.title("Correlation Heatmap of All Numeric Features")
plt.tight_layout()
plt.show()
```

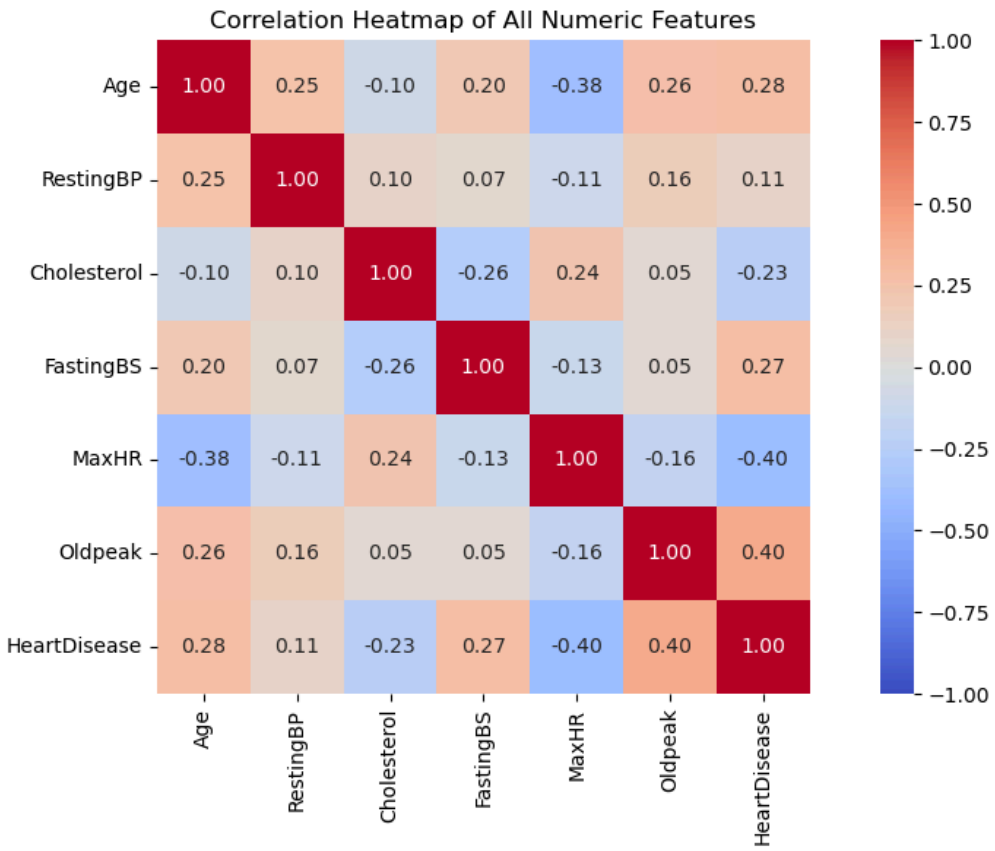


Figure 3. Correlation heatmap of all numerical features in the dataset.

We are not dropping any features as all features seem to be relevant to predicting heart disease based on the EDA.

In [13]:
```
numerical_features = ['Age', 'RestingBP', 'Cholesterol', 'FastingBS', 'MaxHR', 'Oldpeak']
categorical_features = ['Sex', 'ChestPainType', 'RestingECG', 'ExerciseAngina', 'ST_Slope']
```

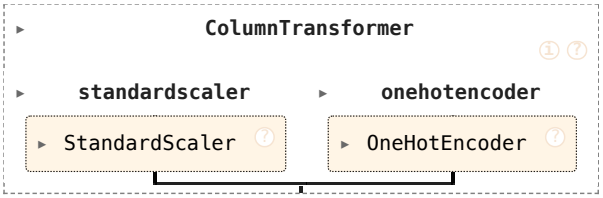
In [14]:
```
# --- Start of code block copied from another author ---
# Title: Function to consolidate cross validation scores into a pandas series.
# Author: Varada Kolhatkar & Michael Gelbart
# Source: https://pages.github.ubc.ca/mds-2025-26/DSCI_571_sup-learn-1_students/README.html
# Taken from: DSCI-571: Laboratory 2
def mean_std_cross_val_scores(model, X_train, y_train, **kwargs):
    """
    Returns mean and std of cross validation

    Parameters
    ----------
    model :
        scikit-learn model
    X_train : numpy array or pandas DataFrame
        X in the training data
    y_train :
        y in the training data

    Returns
    ----------
        pandas Series with mean scores from cross_validation
    """

    scores = cross_validate(model, X_train, y_train, **kwargs)

    mean_scores = pd.DataFrame(scores).mean()
    std_scores = pd.DataFrame(scores).std()
    out_col = []

    for i in range(len(mean_scores)):
        out_col.append((f"%0.3f (+/- %0.3f)" % (mean_scores.iloc[i], std_scores.iloc[i])))
```

```
    return pd.Series(data=out_col, index=mean_scores.index)
# --- End of code block copied from another author ---
```

In [15]:
```
numeric_transformer = StandardScaler()
```

In [16]:
```
preprocessor = make_column_transformer(
    (StandardScaler(), numerical_features),
    (OneHotEncoder(drop='if_binary',
                   handle_unknown="ignore",
                   sparse_output=False
                  ), categorical_features),
)
preprocessor
```

Out[16]:



In [17]:
```
train_df, test_df = train_test_split(df, test_size=0.3, random_state=123)
```

In [18]:
```
X_train = train_df.drop(columns=["HeartDisease"])
X_test = test_df.drop(columns=["HeartDisease"])
y_train = train_df["HeartDisease"]
y_test = test_df["HeartDisease"]
```

In [19]:
```
# Define Classification Metrics for scoring
classification_metrics = ["accuracy", "precision", "recall", "f1"]
```

Models Evaluation

We decided to initiate our analysis considering a set of classification models including Decision Tree, kNN, SVM (with RBF kernel), Logistic Regression, and a dummy classifier as a baseline. All models were trained and evaluated using a 5-fold cross-validation strategy. The performance classification metrics used to evaluate the models were accuracy, precision, recall and F1-score.

Initially, the models were trained using default hyperparameters to define a model that would be further optimized in subsequent steps for this project.

The configuration and results of the cross-validations are detailed below:

In [20]:
```
# Define models to evaluate
models = {
    "dummy": DummyClassifier(random_state=123),
    "decision_tree": DecisionTreeClassifier(random_state=123),
    "kNN": KNeighborsClassifier(),
    "SVM": SVC(random_state=123),
    #"naive_bayes": MultinomialNB(),
    "logistic_regression": LogisticRegression(random_state=123, max_iter=1000)
}
```

In [21]:
```
# Build pipeline function
def build_pipe(model):
    return make_pipeline(preprocessor, model)
```

In [22]:
```
# Execute cross-validation for each model and store results
results_dict = {}

for estimator in models:
    results_dict[estimator] = mean_std_cross_val_scores(
        build_pipe(models[estimator]),
        X_train,
        y_train,
        cv=5,
        return_train_score=True,
        scoring=classification_metrics
    )
    results_df = pd.DataFrame(results_dict)
```

```
results_df
```

Out[22]:

	dummy	decision_tree	kNN	SVM	logistic_regression
fit_time	0.003 (+/- 0.001)	0.003 (+/- 0.000)	0.005 (+/- 0.003)	0.006 (+/- 0.002)	0.004 (+/- 0.001)
score_time	0.004 (+/- 0.000)	0.003 (+/- 0.000)	0.015 (+/- 0.013)	0.006 (+/- 0.002)	0.005 (+/- 0.002)
test_accuracy	0.548 (+/- 0.002)	0.773 (+/- 0.020)	0.844 (+/- 0.045)	0.860 (+/- 0.026)	0.864 (+/- 0.031)
train_accuracy	0.548 (+/- 0.000)	1.000 (+/- 0.000)	0.893 (+/- 0.006)	0.906 (+/- 0.008)	0.870 (+/- 0.008)
test_precision	0.548 (+/- 0.002)	0.803 (+/- 0.014)	0.853 (+/- 0.028)	0.860 (+/- 0.018)	0.859 (+/- 0.018)
train_precision	0.548 (+/- 0.000)	1.000 (+/- 0.000)	0.895 (+/- 0.007)	0.907 (+/- 0.010)	0.872 (+/- 0.008)
test_recall	1.000 (+/- 0.000)	0.776 (+/- 0.052)	0.864 (+/- 0.067)	0.889 (+/- 0.057)	0.900 (+/- 0.057)
train_recall	1.000 (+/- 0.000)	1.000 (+/- 0.000)	0.912 (+/- 0.006)	0.923 (+/- 0.011)	0.893 (+/- 0.009)
test_f1	0.708 (+/- 0.002)	0.788 (+/- 0.025)	0.858 (+/- 0.045)	0.874 (+/- 0.028)	0.879 (+/- 0.032)
train_f1	0.708 (+/- 0.000)	1.000 (+/- 0.000)	0.904 (+/- 0.006)	0.915 (+/- 0.007)	0.882 (+/- 0.007)

Table 1. Cross-validation 5-folds results for all models.

After performing the cross-validation, we observed that both the Logistic Regression and SVM models had the best performance across all metrics, with Logistic Regression slightly outperforming SVM all-around.

Based on this, we decided to choose the Logistic Regression model as our final model for predicting heart disease risk.

In [23]:
```
# Confusion Matrix for Logistic Regression
lr_pipe = make_pipeline(preprocessor,
                        LogisticRegression(random_state=123, max_iter=1000))

conf_mat_logreg = ConfusionMatrixDisplay.from_estimator(
    #build_pipe["logistic_regression"].fit(X_train, y_train),
    lr_pipe.fit(X_train, y_train),
    X_train,
    y_train,
    values_format="d",
    cmap='Blues'
)
conf_mat_logreg
```

Out[23]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x108741400>
```

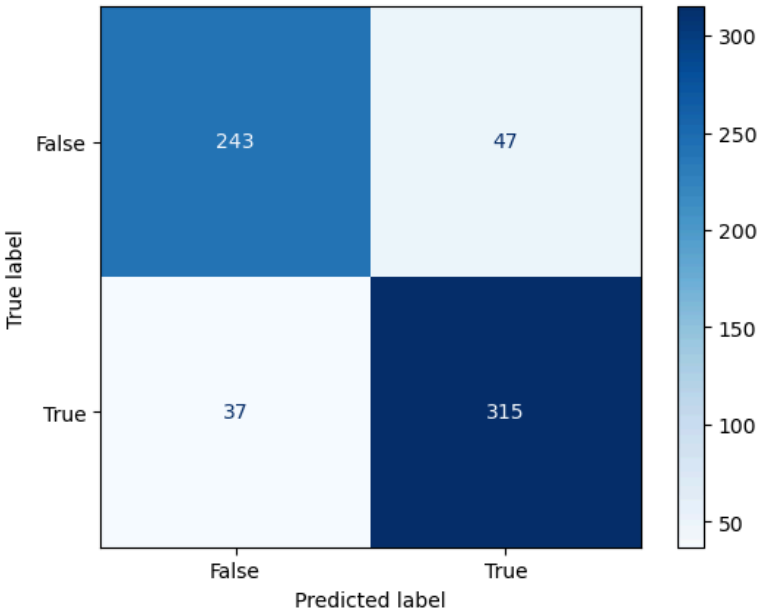


Figure 4. Confusion matrix for Logistic Regression model - Train set.

Evaluation on Test Set

In [24]:
```
# Evaluate on Test Set
# Classification report

y_pred = lr_pipe.predict(X_test)

classification_rep = classification_report(y_test, y_pred)
print(classification_rep)
```

```
              precision    recall  f1-score   support

       False       0.87      0.81      0.84       120
        True       0.86      0.90      0.88       156

    accuracy                           0.86       276
   macro avg       0.86      0.86      0.86       276
weighted avg       0.86      0.86      0.86       276
```

Table 2. Classification Report for Test Set.

In [25]:
```
# Confusion Matrix for Logistic Regression model on Test Set

conf_mat_logreg = ConfusionMatrixDisplay.from_estimator(
    lr_pipe.fit(X_test, y_test),
    X_test,
    y_test,
    values_format="d",
    cmap='Blues'
)
conf_mat_logreg
```

Out[25]: `<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x321f81820>`

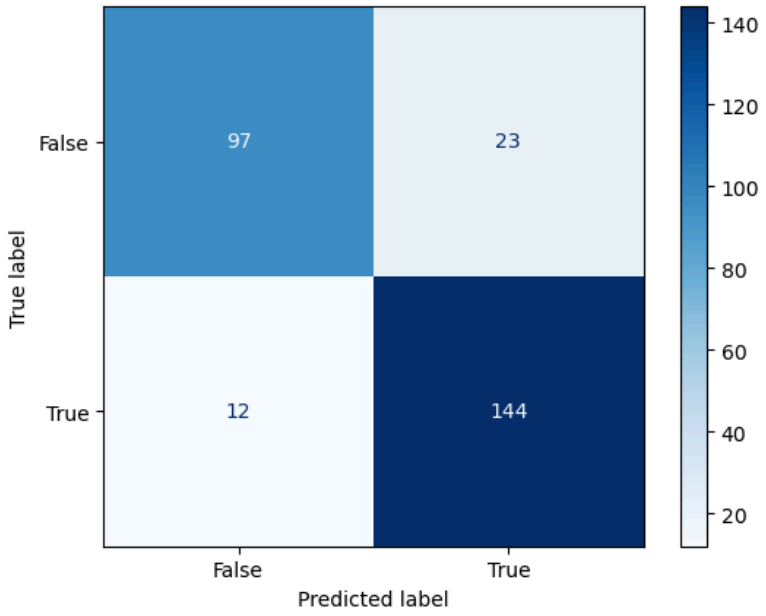


Figure 5. Confusion matrix for Logistic Regression model - Test set.

The initial results of the Logistic Regression prediction model on the test set showed good performance across all metrics, as detailed in Table 2. This matched our expectations based on the cross-validation results, indicating that the model would generalize well on unseen data. This is already a positive outcome for a first iteration of the model and provides a solid foundation for further improvements. It also has the potential to be useful as a support tool for clinical screening of patients.

Aside from the hyperparameter tuning that will be explored on future stages, this use case is well suited for Precision-Recall and Receiver Operating Characteristic (ROC) curve analysis to further evaluate the model's threshold settings and trade-offs between precision and recall. This is a medical-related application, and it is crucial to minimize false negatives i.e., predicting a patient is not at risk when they actually are. Also, making the probability scores available for the predictions could increase the model's utility, supporting the decision-making of healthcare professionals.

Finally, we could review the features coefficients from the Logistic Regression model to understand their influence on the predictions, enabling to better interpret the model's decisions and confirm if we could drop any features in future iterations.

References

1. American Heart Association. (2025). 2025 Heart & Stroke Statistics Update Fact Sheet: Global burden of disease [PDF]. https://professional.heart.org/-/media/phd-files-2/science-news/2/2025-heart-and-stroke-stat-update/factsheets/2025-stats-update-fact-sheet-global-burden-of-disease.pdf

2. Barnett, M. P., Koppes, L. L. J., & … [et al.]. (2020). Cardiovascular risk factors: It's time to focus on variability! Frontiers in Cardiovascular Medicine, 7, Article 80. https://doi.org/10.3389/fcvm.2020.00080 (PMC published version) https://pmc.ncbi.nlm.nih.gov/articles/PMC7379092/

3. Federico Soriano Palacios. (September 2021). Heart Failure Prediction Dataset. Retrieved [Date Retrieved] from https://www.kaggle.com/fedesoriano/heart-failure-prediction.

4. Harris, C.R. et al., 2020. Array programming with NumPy. Nature, 585, pp.357–362.

5. J. D. Hunter, "Matplotlib: A 2D Graphics Environment", Computing in Science & Engineering, vol. 9, no. 3, pp. 90-95, 2007.

6. Janosi, A., Steinbrunn, W., Pfisterer, M., & Detrano, R. (1989). Heart Disease [Dataset]. UCI Machine Learning Repository. https://doi.org/10.24432/C52P4X.

7. Kokori, E., Patel, R., Olatunji, G., Ukoaka, B. M., Abraham, I. C., Ajekiigbe, V. O., Kwape, J. M., Babalola, A. E., Udam, N. G., & Aderinto, N. (2025). Machine learning in predicting heart failure survival: A review of current models and future prospects. Heart Failure Reviews, 30(2), 431–442. https://doi.org/10.1007/s10741-024-10474-y https://pubmed.ncbi.nlm.nih.gov/39656330/

8. McKinney, Wes. 2010. "Data Structures for Statistical Computing in Python." In Proceedings of the 9th Python in Science Conference, edited by Stéfan van der Walt and Jarrod Millman, 51–56.

9. Pedregosa, F. et al., 2011. Scikit-learn: Machine learning in Python. Journal of machine learning research, 12(Oct), pp.2825–2830.

10. Reitz, Kenneth. 2011. Requests: HTTP for Humans. https://requests.readthedocs.io/en/master/.

11. Savarese, G., Lund, L. H., & Becher, P. M. (2023). Global burden of heart failure: A comprehensive and updated review of epidemiology. Cardiovascular Research, 118(17), 3272–3287. https://doi.org/10.1093/cvr/cvac013 https://pubmed.ncbi.nlm.nih.gov/35150240/

12. Van Rossum, Guido, and Fred L. Drake. 2009. Python 3 Reference Manual. Scotts Valley, CA: CreateSpace.

13. VanderPlas, J. et al., 2018. Altair: Interactive statistical visualizations for python. Journal of open source software, 3(32), p.1057.

14. World Health Organization. (2024, August 8). Cardiovascular diseases (CVDs) – fact sheet. https://www.who.int/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds)

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