# Logistic Regression: Classifying Heart Disease in the Binary Classification Case

October 21, 2023

## 1 Introduction

In this project, I will use logistic regression and classification on the the Heart Disease Data Set from the Cleveland Clinic Foundation (https://archive.ics.uci.edu/dataset/45/heart+disease). With a focus on real-world application, the objective is to decipher the information within the dataset, ranging from patient characteristics like age and chest pain, to construct a robust classification model.

By delving into the machine learning pipeline, I aim to use the power of logistic regression to address the question: Can we effectively predict the presence of heart disease in individuals based on these diverse attributes?

Here are the features in the dataset:

1. 'Unnamed: 0'
2. 'age'
3. 'sex'
4. 'cp' (Chest Pain)
5. 'trestbps' (Resting Blood Pressure)
6. 'chol' (Serum Cholesterol)
7. 'fbs' (Fasting Blood Sugar)
8. 'restecg' (Resting Electrocardiographic Results)
9. 'thalach' (Maximum Heart Rate Achieved)
10. 'exang' (Exercise-Induced Angina)
11. 'oldpeak' (ST Depression Induced by Exercise Relative to Rest)
12. 'slope'
13. 'ca' (Number of Major Vessels Colored by Fluoroscopy)
14. 'thal' (Thalassemia)
15. 'present' (Target Variable indicating the Presence of Heart Disease)

[58]:
```
import pandas as pd
heart = pd.read_csv('heart_disease.csv')
```

## 2 Exploring the Data

[59]:
```
heart.head()
```

[59]:
```
   Unnamed: 0  age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  \
0           1   63    1   1       145   233    1        2      150      0
1           2   67    1   4       160   286    0        2      108      1
2           3   67    1   4       120   229    0        2      129      1
3           4   37    1   3       130   250    0        0      187      0
4           5   41    0   2       130   204    0        2      172      0

   oldpeak  slope   ca thal  present
0      2.3      3  0.0  6.0        0
1      1.5      2  3.0  3.0        1
2      2.6      2  2.0  7.0        1
3      3.5      3  0.0  3.0        0
4      1.4      1  0.0  3.0        0
```

[60]:
```
heart.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 15 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Unnamed: 0  303 non-null    int64
 1   age         303 non-null    int64
 2   sex         303 non-null    int64
 3   cp          303 non-null    int64
 4   trestbps    303 non-null    int64
 5   chol        303 non-null    int64
 6   fbs         303 non-null    int64
 7   restecg     303 non-null    int64
 8   thalach     303 non-null    int64
 9   exang       303 non-null    int64
 10  oldpeak     303 non-null    float64
 11  slope       303 non-null    int64
 12  ca          303 non-null    object
 13  thal        303 non-null    object
 14  present     303 non-null    int64
dtypes: float64(1), int64(12), object(2)
memory usage: 35.6+ KB
```

Converting `ca` and `thal` to numeric data types:

[61]:
```
heart['ca'] = pd.to_numeric(heart['ca'], errors='coerce')
heart['thal'] = pd.to_numeric(heart['thal'], errors='coerce')
```

[62]:
```
heart.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 15 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   Unnamed: 0  303 non-null    int64  
 1   age         303 non-null    int64  
 2   sex         303 non-null    int64  
 3   cp          303 non-null    int64  
 4   trestbps    303 non-null    int64  
 5   chol        303 non-null    int64  
 6   fbs         303 non-null    int64  
 7   restecg     303 non-null    int64  
 8   thalach     303 non-null    int64  
 9   exang       303 non-null    int64  
 10  oldpeak     303 non-null    float64
 11  slope       303 non-null    int64  
 12  ca          299 non-null    float64
 13  thal        301 non-null    float64
 14  present     303 non-null    int64  
dtypes: float64(3), int64(12)
memory usage: 35.6 KB
```

[63]:
```
heart.describe()
```

[63]:

| | Unnamed: 0 | age | sex | cp | trestbps | chol | \ |
|---|---|---|---|---|---|---|---|
| count | 303.000000 | 303.000000 | 303.000000 | 303.000000 | 303.000000 | 303.000000 | |
| mean | 152.000000 | 54.438944 | 0.679868 | 3.158416 | 131.689769 | 246.693069 | |
| std | 87.612784 | 9.038662 | 0.467299 | 0.960126 | 17.599748 | 51.776918 | |
| min | 1.000000 | 29.000000 | 0.000000 | 1.000000 | 94.000000 | 126.000000 | |
| 25% | 76.500000 | 48.000000 | 0.000000 | 3.000000 | 120.000000 | 211.000000 | |
| 50% | 152.000000 | 56.000000 | 1.000000 | 3.000000 | 130.000000 | 241.000000 | |
| 75% | 227.500000 | 61.000000 | 1.000000 | 4.000000 | 140.000000 | 275.000000 | |
| max | 303.000000 | 77.000000 | 1.000000 | 4.000000 | 200.000000 | 564.000000 | |

| | fbs | restecg | thalach | exang | oldpeak | slope | \ |
|---|---|---|---|---|---|---|---|
| count | 303.000000 | 303.000000 | 303.000000 | 303.000000 | 303.000000 | 303.000000 | |
| mean | 0.148515 | 0.990099 | 149.607261 | 0.326733 | 1.039604 | 1.600660 | |
| std | 0.356198 | 0.994971 | 22.875003 | 0.469794 | 1.161075 | 0.616226 | |
| min | 0.000000 | 0.000000 | 71.000000 | 0.000000 | 0.000000 | 1.000000 | |
| 25% | 0.000000 | 0.000000 | 133.500000 | 0.000000 | 0.000000 | 1.000000 | |
| 50% | 0.000000 | 1.000000 | 153.000000 | 0.000000 | 0.800000 | 2.000000 | |
| 75% | 0.000000 | 2.000000 | 166.000000 | 1.000000 | 1.600000 | 2.000000 | |
| max | 1.000000 | 2.000000 | 202.000000 | 1.000000 | 6.200000 | 3.000000 | |

```
                ca        thal     present
count   299.000000  301.000000  303.000000
mean      0.672241    4.734219    0.458746
std       0.937438    1.939706    0.499120
min       0.000000    3.000000    0.000000
25%       0.000000    3.000000    0.000000
50%       0.000000    3.000000    0.000000
75%       1.000000    7.000000    1.000000
max       3.000000    7.000000    1.000000
```

There seems to be missing values in the `ca` and `thal` columns. I will investigate these and if needed will replace them with the mean.

This table also shows the presence of binary features. Below is a list of the binary features:

**Binary Variables:** 1. `sex` (Gender, assuming 0 or 1 values) 2. `fbs` (Fasting Blood Sugar, assuming 0 or 1 values) 3. `exang` (Exercise-Induced Angina, assuming 0 or 1 values) 4. `present` (Target Variable indicating the Presence of Heart Disease, assuming 0 or 1 values)

[64]:
```
heart['ca'].fillna(heart['ca'].mean(), inplace=True)
heart['thal'].fillna(heart['thal'].mean(), inplace=True)
```

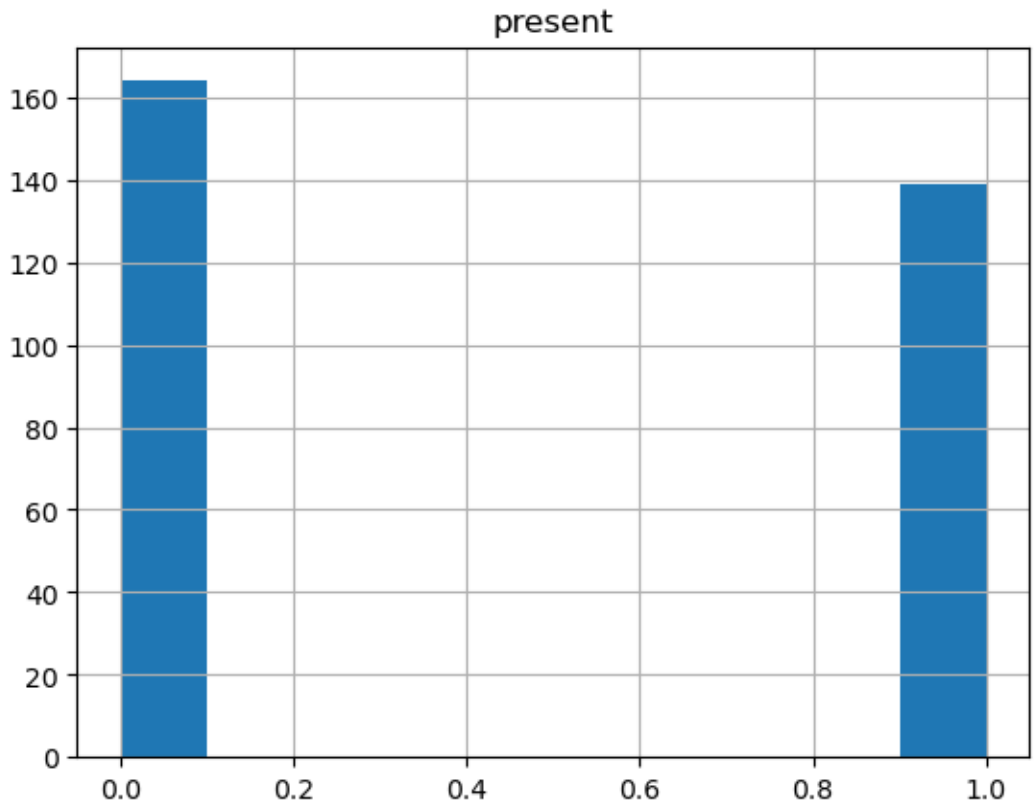
[65]:
```
print(heart.isnull().sum())

print("Missing values in 'ca':", heart['ca'].isnull().sum())
print("Missing values in 'thal':", heart['thal'].isnull().sum())
```

```
Unnamed: 0    0
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
present       0
dtype: int64
Missing values in 'ca': 0
Missing values in 'thal': 0
```

[66]:
```
heart.hist("present")
```

[66]: array([[<AxesSubplot:title={'center':'present'}>]], dtype=object)




Looking at the histogram of the target variable `present`, I can see that the number of cases are not too different between cases where heart disease is present or not.

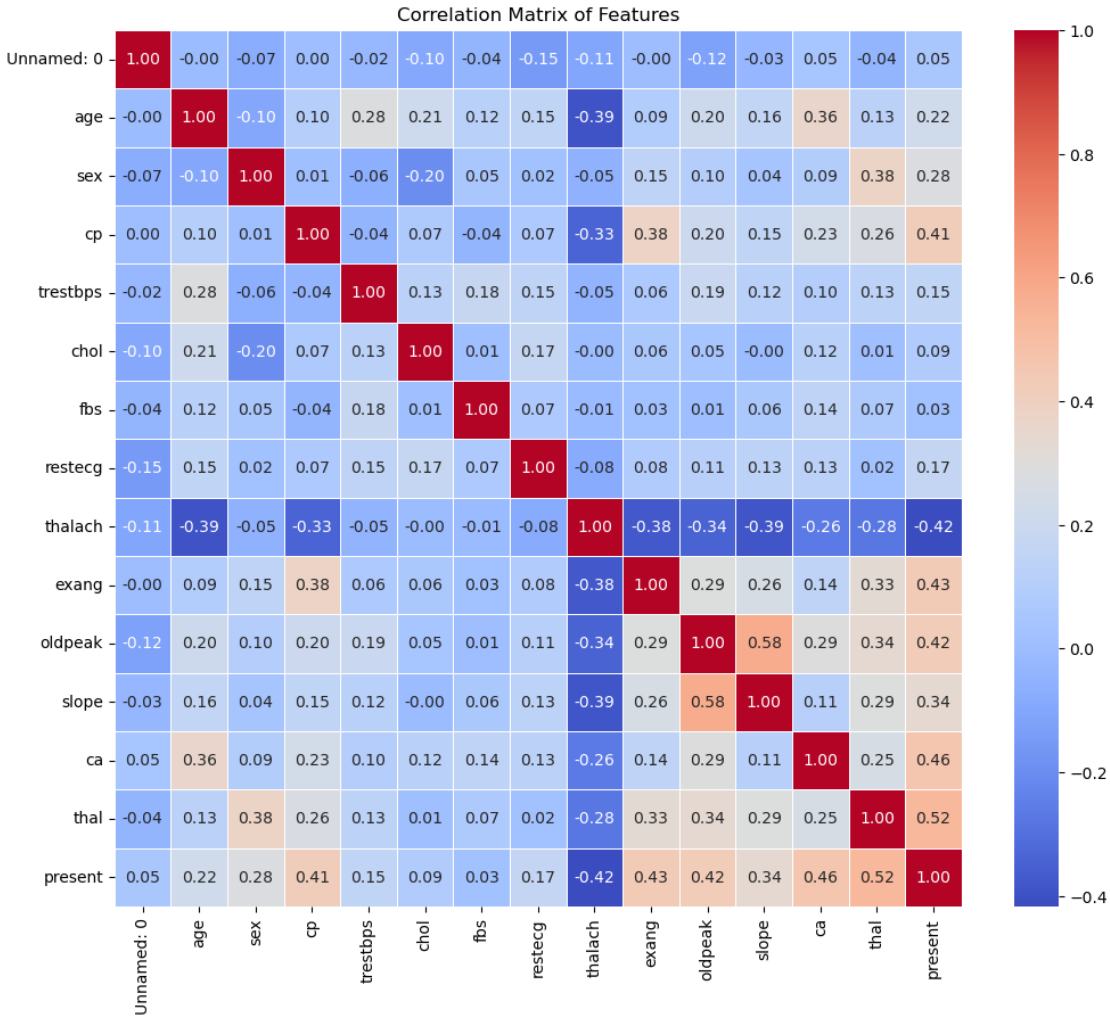
## 3 Feature Selection

Now, I will select the features that I will use in this logistic regression model by seeing if there are any features are highly correlated to the target variable `present`.

[67]:
```
import seaborn as sns
import matplotlib.pyplot as plt

correlation_matrix = heart.corr()

plt.figure(figsize=(12, 10))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f",␣
 ↪linewidths=.5)
plt.title('Correlation Matrix of Features')
plt.show()
```




[68]:
```
correlation_matrix = heart.corr()
correlation_with_target = correlation_matrix['present'].abs().
  ↪sort_values(ascending=False)
print(correlation_with_target)
```

```
present     1.000000
thal        0.523928
ca          0.457598
exang       0.431894
oldpeak     0.424510
thalach     0.417167
cp          0.414446
slope       0.339213
sex         0.276816
age         0.223120
restecg     0.169202
```

```
trestbps      0.150825
chol          0.085164
Unnamed: 0    0.048765
fbs           0.025264
Name: present, dtype: float64
```

Based off of the correlations, the features that are the most correlated with the target variable `present` are `thal` (Thalassemia) and `ca` (Number of Major Vessels Colored by Fluoroscopy).

Now, I will use the Split-Apply-Combine method to check

[69]:
```
import pandas as pd
from sklearn.feature_selection import SelectKBest, f_classif
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

# Assuming 'heart' is your DataFrame
# Assuming 'present' is your target variable indicating the presence of heart␣
 ↪disease
X = heart.drop(['present', 'Unnamed: 0'], axis=1)  # Drop the target variable␣
 ↪and 'Unnamed: 0'
y = heart['present']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)

# Initialize SelectKBest with ANOVA F-statistic
selector = SelectKBest(f_classif, k=2)  # Choose the desired number of features

# Fit and transform the training data
X_train_selected = selector.fit_transform(X_train, y_train)

# Get the selected feature indices
selected_indices = selector.get_support(indices=True)

# Get the names of the selected features
selected_features = X.columns[selected_indices]

# Display the selected features
print("Selected Features:")
print(selected_features)
```

```
Selected Features:
Index(['ca', 'thal'], dtype='object')
```

From looking at the correlations to using the SelectKBest test, I can see that the features that I should use for the model are `ca` and `thal`.

## 4 Dividing the Data

Now, I will divide the data into test and training sets.

[70]:
```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression

X = heart.drop(["present"], axis = 1)
y = heart["present"]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20,␣
↪random_state = 734)
```

## 5 Building the Model

I will now build and test three models with the following features:

- `ca`
- `thal`
- `ca` and `thal`

All models will still have the same target variable, which is `present`.

[71]:
```
X1 = X_train[["ca"]]
X2 = X_train[["thal"]]
X3 = X_train[["ca", "thal"]]

X1_test = X_test[["ca"]]
X2_test = X_test[["thal"]]
X3_test = X_test[["ca", "thal"]]

model1 = LogisticRegression()
model2 = LogisticRegression()
model3 = LogisticRegression()

model1.fit(X1, y_train)
model2.fit(X2, y_train)
model3.fit(X3, y_train)

y_train_pred1 = model1.predict(X1)
y_train_pred2 = model2.predict(X2)
y_train_pred3 = model3.predict(X3)
```

## 6 Evaluating the Models

[72]:
```
from sklearn.metrics import accuracy_score
train_accuracy1 = accuracy_score(y_train, y_train_pred1)
train_accuracy2 = accuracy_score(y_train, y_train_pred2)
train_accuracy3 = accuracy_score(y_train, y_train_pred3)
train_accuracies = [train_accuracy1, train_accuracy2, train_accuracy3]
print(train_accuracies)
```

```
[0.7231404958677686, 0.7644628099173554, 0.7892561983471075]
```

From the three models, I can see that the model with the most complexity which uses both features has the highest accuaracy of around 79%.

[73]:
```
predictions1 = y_train_pred1

tp = sum((y_train == 1) & (predictions1 == 1))
fn = sum((y_train == 1) & (predictions1 == 0))
sensitivity1 = tp / (tp + fn)

predictions2 = y_train_pred2

tp = sum((y_train == 1) & (predictions2 == 1))
fn = sum((y_train == 1) & (predictions2 == 0))
sensitivity2 = tp / (tp + fn)

predictions3 = y_train_pred3

tp = sum((y_train == 1) & (predictions3 == 1))
fn = sum((y_train == 1) & (predictions3 == 0))
sensitivity3 = tp / (tp + fn)

train_sensitivities = [sensitivity1, sensitivity2, sensitivity3]
print(train_sensitivities)
```

```
[0.6363636363636364, 0.7363636363636363, 0.8090909090909091]
```

The third model demonstrated the highest sensitivity among the three, correctly identifying approximately 80.91% of the actual positive instances. This indicates that Model 3 has the highest ability to capture instances of heart disease among the positive cases.

Now, I will calculate the specifity of each model.

[74]:
```
tn = sum((y_train == 0) & (predictions1 == 0))
fp = sum((y_train == 0) & (predictions1 == 1))
specificity1 = tn / (tn + fp)

tn = sum((y_train == 0) & (predictions2 == 0))
fp = sum((y_train == 0) & (predictions2 == 1))
```

```
specificity2 = tn / (tn + fp)

tn = sum((y_train == 0) & (predictions3 == 0))
fp = sum((y_train == 0) & (predictions3 == 1))
specificity3 = tn / (tn + fp)

train_specificities = [specificity1, specificity2, specificity3]
print(train_specificities)
```

```
[0.7954545454545454, 0.7878787878787878, 0.7727272727272727]
```

The first model correctly identified the most of the actual negative instances. However, the difference is specitivities is small between the three models.

## 7 Interpreting the Model Coefficients for Model 3

[75]:
```
print("Intercept:", model3.intercept_[0])
print("Coefficients:")
for feature, coefficient in zip(X3.columns, model3.coef_[0]):
    print(f"{feature}: {coefficient}")


odds_ratios = [round(np.exp(coef), 4) for coef in model3.coef_[0]]
print("\nOdds Ratios:")
for feature, odds_ratio in zip(X3.columns, odds_ratios):
    print(f"{feature}: {odds_ratio}")
```

```
Intercept: -3.7350523077255913
Coefficients:
ca: 1.0322442647345986
thal: 0.5998385489388353

Odds Ratios:
ca: 2.8074
thal: 1.8218
```

Interpretation for the odds ratios of Model 3:

1. **Odds Ratio for `ca` (Number of Major Vessels Colored by Fluoroscopy): 2.8074**
   - Interpretation: A one-unit increase in the number of major vessels colored by fluoroscopy (`ca`) is associated with an approximately 2.81 times higher odds of the presence of heart disease, holding other variables constant. This implies that individuals with more major vessels colored are more likely to have heart disease compared to those with fewer colored vessels.
2. **Odds Ratio for `thal` (Thalassemia): 1.8218**
   - Interpretation: A one-unit increase in the thalassemia level (`thal`) is associated with an approximately 1.82 times higher odds of the presence of heart disease, holding other variables constant. This suggests that individuals with a higher thalassemia level are

somewhat more likely to have heart disease compared to those with a lower thalassemia level.

## 8 Final Model Evaluation

Now, I will use Model 3 to calculate the test predictions and then evaluate its accuracy, sensitivity, and specificity.

[76]:
```
y_test_pred3 = model3.predict(X3_test)
test_accuracy3 = accuracy_score(y_test, y_test_pred3)
print(test_accuracy3)
```

```
0.7868852459016393
```

Model 3 correctly predicted the target variable (presence or absence of heart disease) for about 78.79% of the instances in the test set.

[77]:
```
predictions3 = y_test_pred3

tp = sum((y_test == 1) & (predictions3 == 1))
fn = sum((y_test == 1) & (predictions3 == 0))
sensitivity3 = tp / (tp + fn)
print(sensitivity3)
```

```
0.7586206896551724
```

Model 3 correctly identified about 75.86% of the actual positive instances (cases with heart disease) in the test set.

[78]:
```
tn = sum((y_test == 0) & (predictions3 == 0))
fp = sum((y_test == 0) & (predictions3 == 1))
specificity3 = tn / (tn + fp)
print(specificity3)
```

```
0.8125
```

Model 3 correctly identified about 81.25% of the actual negative instances (cases without heart disease) in the test set.

## 9 Conclusion

The logistic regression model developed for predicting heart disease based on the features `ca` (number of major vessels colored by fluoroscopy) and `thal` (thalassemia) demonstrates meaningful interpretability. The odds ratios indicate that an increase in the number of major vessels colored by fluoroscopy (ca) and a higher thalassemia level (thal) are associated with higher odds of the presence of heart disease. This aligns with our expectations, as it is plausible that individuals with more major vessels colored or higher thalassemia levels may be more likely to have heart disease.

Comparing the model's performance in predicting cases and non-cases, the sensitivity of 75.86% indicates a relatively good ability to identify individuals with heart disease among the actual positive

cases. The specificity of 81.25% suggests a strong capability to correctly classify individuals without heart disease among the actual negative cases. The balance between sensitivity and specificity is crucial, and the model appears to perform reasonably well in both aspects.

The accuracy of approximately 78.79% signifies the proportion of correct predictions in the test set. While accuracy is an essential metric, its acceptability in an actual clinical setting depends on the specific context and the consequences of false positives and false negatives. Given that the model demonstrates a balanced performance with respect to sensitivity and specificity, it could potentially be useful as a screening tool. However, before deployment in a clinical setting, further validation, and potentially refinement of the model would be necessary. Additionally, collaboration with domain experts and consideration of the potential impact on patient outcomes are crucial steps in evaluating the model's suitability for clinical use.

In summary, the logistic regression model exhibits coherent interpretation, aligning with expectations regarding the influence of features on heart disease prediction. Its balanced performance in sensitivity and specificity, coupled with a reasonably high accuracy, suggests promising potential for application, pending thorough validation and consideration of clinical implications.