Heart Disease Predictions

GA Final Project - Justin Moss

Objective

Based on the observed parameters, predict if a given set of characteristics puts a patient at risk for heart disease

Data Dictionary

- Age: age of the patient [years]
- Sex: sex of the patient [M: Male, F: Female] / [0: Female, 1: Male]
- ChestPainType: chest pain type [TA: Typical Angina, ATA: Atypical Angina, NAP: Non-Anginal Pain, ASY: Asymptomatic]
- RestingBP: resting blood pressure [mm Hg]
- Cholesterol: serum cholesterol [mm/dl]
- FastingBS: fasting blood sugar [1: if FastingBS > 120 mg/dl, 0: otherwise]
- RestingECG: resting electrocardiogram results [Normal: Normal, ST: having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV), LVH: showing probable or definite left ventricular hypertrophy by Estes' criteria]
- MaxHR: maximum heart rate achieved [Numeric value between 60 and 202]
- ExerciseAngina: exercise-induced angina [Y: Yes, N: No]
- Oldpeak: oldpeak = ST [Numeric value measured in depression]
- ST_Slope: the slope of the peak exercise ST segment [Up: upsloping, Flat: flat, Down: downsloping]
- HeartDisease: output class [1: heart disease, 0: Normal]

First Steps:

Loading in the data, renaming columns for ease of use, and some EDA.

In [5]:
```
# import data
heart = pd.read_csv('C:/Users/Magnus/Desktop/School-Stuff/General-Assembly/GitBash/DS/unit-4_project/heart.csv')
```

In [6]:
```
# rename some columns for easier reference
heart = heart.rename(columns={
    'Age': 'age',
    'RestingBP': 'resting_bp',
    'Cholesterol': 'cholesterol',
    'FastingBS': 'fastingbs',
    'MaxHR': 'max_hr',
    'Oldpeak': 'oldpeak',
    'HeartDisease': 'heart_disease'
})
```

In [7]:
```
heart.head()
```

Out[7]:

	age	Sex	ChestPainType	resting_bp	cholesterol	fastingbs	RestingECG	max_hr	ExerciseAngina	oldpeak	ST_Slope	heart_disease
0	40	M	ATA	140	289	0	Normal	172	N	0.0	Up	0
1	49	F	NAP	160	180	0	Normal	156	N	1.0	Flat	1
2	37	M	ATA	130	283	0	ST	98	N	0.0	Up	0
3	48	F	ASY	138	214	0	Normal	108	Y	1.5	Flat	1
4	54	M	NAP	150	195	0	Normal	122	N	0.0	Up	0

In [8]:
```
# create dummies for the codified non-ineger parameters
heart_sex_dummies = pd.get_dummies(heart['Sex'], prefix='sex')
heart_pain_dummies = pd.get_dummies(heart['ChestPainType'], prefix='pain_type')
heart_pain_ecg = pd.get_dummies(heart['RestingECG'], prefix='rest_ecg')
heart_pain_ex_angina = pd.get_dummies(heart['ExerciseAngina'], prefix='exercise_angina')
heart_pain_st_slope = pd.get_dummies(heart['ST_Slope'], prefix='st_slope')
```

In [9]:
```
# combine dummies into one dataframe
dummies = pd.concat([
    heart_sex_dummies,
    heart_pain_dummies,
    heart_pain_ecg,
    heart_pain_ex_angina,
    heart_pain_st_slope
], axis=1)
```

In [151]:
```
dummies.head(5)
```

Out[151]:

pain_type_NAP	pain_type_TA	rest_ecg_LVH	rest_ecg_Normal	rest_ecg_ST	exercise_angina_N	exercise_angina_Y	st_slope_Down	st_slope_Flat	st_slope_Up
0	0	0	1	0	1	0	0	0	1
1	0	0	1	0	1	0	0	1	0
0	0	0	0	1	1	0	0	0	1
0	0	0	1	0	0	1	0	1	0
1	0	0	1	0	1	0	0	0	1

In [11]:
```
# combine original data with dummies
heart_dummies = pd.concat([heart, dummies], axis=1)
heart_dummies.head()
```

Out[11]:

	age	Sex	ChestPainType	resting_bp	cholesterol	fastingbs	RestingECG	max_hr	ExerciseAngina	oldpeak	ST_Slope	heart_disease	sex_F	sex_M	pain_t
0	40	M	ATA	140	289	0	Normal	172	N	0.0	Up	0	0	1	
1	49	F	NAP	160	180	0	Normal	156	N	1.0	Flat	1	1	0	
2	37	M	ATA	130	283	0	ST	98	N	0.0	Up	0	0	1	
3	48	F	ASY	138	214	0	Normal	108	Y	1.5	Flat	1	1	0	
4	54	M	NAP	150	195	0	Normal	122	N	0.0	Up	0	0	1	

In [14]:
```
# some EDA looking at the numbers of the dummies
heart_dummies.groupby(by='heart_disease')[['sex_F', 'sex_M', 'pain_type_ASY', 'pain_type_ATA',
      'pain_type_NAP', 'pain_type_TA', 'rest_ecg_LVH', 'rest_ecg_Normal',
      'rest_ecg_ST', 'exercise_angina_N', 'exercise_angina_Y',
      'st_slope_Down', 'st_slope_Flat', 'st_slope_Up']].agg(['mean'])
```

Out[14]:

	sex_F	sex_M	pain_type_ASY	pain_type_ATA	pain_type_NAP	pain_type_TA	rest_ecg_LVH	rest_ecg_Normal	rest_ecg_ST	exercise_angina_N	exercise
	mean	mean	mean	mean	mean	mean	mean	mean	mean	mean	
ase											
0	0.348780	0.651220	0.253659	0.363415	0.319512	0.063415	0.200000	0.651220	0.148780	0.865854	
1	0.098425	0.901575	0.771654	0.047244	0.141732	0.039370	0.208661	0.561024	0.230315	0.377953	

In [15]:
```
# some EDA looking at the numbers of the non-dummies
heart_dummies.groupby(by='heart_disease')[['age', 'resting_bp',
                                           'cholesterol', 'fastingbs', 'max_hr',
                                          'oldpeak']].agg(['mean', 'min', 'max'])
```

Out[15]:

	age			resting_bp			cholesterol			fastingbs			max_hr			oldpeak		
	mean	min	max	mean	min	max	mean	min	max	mean	min	max	mean	min	max	mean	min	max
heart_disease																		
0	50.551220	28	76	130.180488	80	190	227.121951	0	564	0.107317	0	1	148.151220	69	202	0.408049	-1.1	4.2
1	55.899606	31	77	134.185039	0	200	175.940945	0	603	0.334646	0	1	127.655512	60	195	1.274213	-2.6	6.2

In [16]:
```
# due to the over-representation of males with heart disease, I was curious about how much of the sample they represent
heart_dummies['Sex'].value_counts(normalize=True)
```

Out[16]:
```
M    0.78976
F    0.21024
Name: Sex, dtype: float64
```

EDA Observations:

- 55% of samples have heart_disease
- Age Range is from 28 to 77, with no significant separation in average age
- 90% of heart disease cases are men despite making up only 79% of the samples
- No significant separation in average resting Resting BP
- Heart disease cases typically have lower cholesterol
- Heart disease cases typically have lower max heart rates
- 33% of heart disease cases have fasting blood sugar over 120 mg/dl
- 77% of heart disease cases have ASY pain type (asymptomatic ie: no pain)
- 56% of heart disease cases have normal resting ECG
- 62% of heart disease cases have Exercise Angina
- Higher oldpeak positively associated with higher risk of heart disease
- Flat ST Slope for 75% of heart disease cases

Moving forward:

Since we are predicting a binary value, we will use a Logistic Regression.
We'll use all of the parameters as our features.

Fitting with all of the data gave an accuracy of ~87%

```
# for whatever reason, I had to increase max iterations to make the code work
logreg = LogisticRegression(max_iter=1000)

feature_cols = ['age', 'resting_bp','cholesterol', 'fastingbs', 'max_hr', 'oldpeak',
                'sex_F', 'sex_M', 'pain_type_ASY', 'pain_type_ATA', 'pain_type_NAP',
                'pain_type_TA', 'rest_ecg_LVH', 'rest_ecg_Normal', 'rest_ecg_ST',
                'exercise_angina_N', 'exercise_angina_Y', 'st_slope_Down', 'st_slope_Flat',
                'st_slope_Up'
               ]

X = heart_dummies[feature_cols]
y = heart_dummies['heart_disease']

# First I'll fit all of the data and see what results we get
logreg.fit(X,y)
pred = logreg.predict(X)
logreg.score(X,y)*100
```

```
87.25490196078431
```

More False Negatives or Positives?

As this is something as serious as a heart attack, I would rather a potential patient received a false positive and sought care instead of a false negative. There will always be a tradeoff there. (more on that later)

```
# Lets stack the predcitions and their probabilities against the actual data
heart_dummies['predict_prob'] = logreg.predict_proba(X)[:, 1]
heart_dummies['predict'] = logreg.predict(X)
heart_dummies[['heart_disease', 'predict', 'predict_prob']].head(10)
```

	heart_disease	predict	predict_prob
0	0	0	0.034930
1	1	0	0.254823
2	0	0	0.045156
3	1	1	0.839502
4	0	0	0.093832
5	0	0	0.030736
6	0	0	0.011728
7	0	0	0.061855
8	1	1	0.941827
9	0	0	0.012803

```
false_neg = heart_dummies[heart_dummies['heart_disease'] > heart_dummies['predict']]
false_neg.shape
```

```
(48, 28)
```

```
false_pos = heart_dummies[heart_dummies['heart_disease'] < heart_dummies['predict']]
false_pos.shape
```

```
(69, 28)
```

```
f'Percent false readings: {((false_neg.shape[0]+false_pos.shape[0])/heart_dummies.shape[0])*100} %'
```

```
'Percent false readings: 12.745098039215685 %'
```

Train-Test-Split?

Our score falls slightly with a train-test-split, which of course means we're more generalized now, but we can do better.

```
# Going with a test-train-split:
logreg1 = LogisticRegression(max_iter=1000)
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42)

logreg1.fit(X_train, y_train);
```

```
# scoring the train-test-split model
y_pred = logreg1.predict(X_test)
logreg1.score(X_test, y_test)
```

```
0.8652173913043478
```

Confusion Matrixing, Altering Threshold

Using a confusion matrix, we will change the threshold for a positive indication of heart disease to '.3' for the probability. We now have significantly fewer false scores.

```
# Build a confusion matrix to send more potential false negative to coming back as positive
logit_simple = linear_model.LogisticRegression(max_iter=1000, C=1e9).fit(X_train, y_train)
logit_pred_proba = logit_simple.predict_proba(X_test)[:,1]
tn, fp, fn, tp = metrics.confusion_matrix(y_true=y_test, y_pred=logit_pred_proba > .3).ravel()
(tn, fp, fn, tp)
```

```
(83, 15, 13, 119)
```

```
# percent of false readings
(fp+fn)/len(y_test)*100
```

```
12.173913043478262
```

```
# percent of 'true' readings
(tn+tp)/len(y_test)*100
```

```
87.82608695652175
```

Am I at risk?

I made some assumptions as I've never heard of oldpeak or ST Slope before this project, but I should be fine for now.

```
me = [[31, 80, 160, 0, 190, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1]]
print(logit_simple.predict(me))
print(logit_simple.predict_proba(me))
```

```
[0]
[[0.75145737 0.24854263]]
```

Conclusions:

Based on our findings with the confusion matrix possibly creating more false positives, I am personally not presently at risk for heart disease.

In the future, we could expand the number of observations and take in more health data such as:
- Is the patient a smoker?
- Do they exercise regularly?
- Body-mass Index
- etc.

Additionally, I could clean up the dummies to bring it closer to a binary. In particular:
- Sex could be 0=male 1=female
- Pain types could have three columns for the actual pains, with asymptomatic being all 0s
- same with resting ecg, st slope, exercise angina

Questions?

Data sourced from:
https://www.kaggle.com/datasets/fedesoriano/heart-failure-prediction