Predicting Heart Failure

Brian Guo, Winston Yang, Tiffany Yan

Can we predict heart failure given other medical information about the patient?

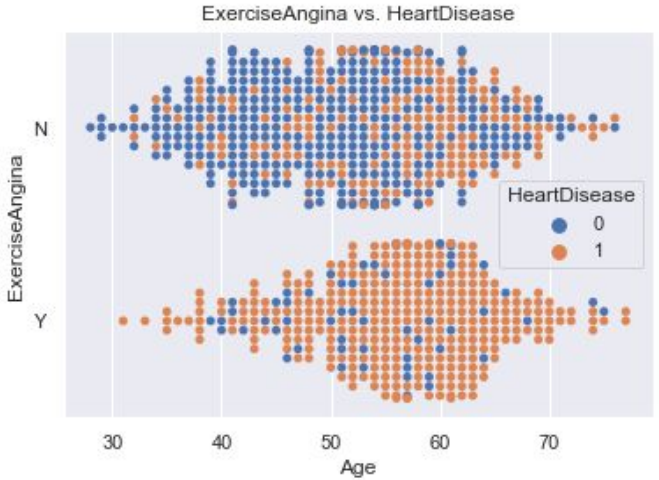
Introduction

- Heart disease is the leading cause of death in America with one person dying every 36 seconds due to heart disease
- Over 600,000 people die from heart disease every year in America
- Doctors can take precautions to prevent these deaths if such future can be predicted
- This data set contains 918 observations and 12 variables containing 11 features and 1 target variable

Features

1) Age [years]
2) Sex [M: Male, F: Female]
3) ChestPainType [TA: Typical Angina, ATA: Atypical Angina, NAP: Non-Anginal Pain, ASY: Asymptomatic]
4) RestingBP [mm Hg]
5) Cholesterol [mm/dl]
6) FastingBS [1: if FastingBS > 120 mg/dl, 0: otherwise]
7) RestingECG [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]
8) MaxHR: maximum heart rate achieved [Numeric value between 60 and 202]
9) ExerciseAngina: exercise-induced angina [Y: Yes, N: No]
10) Oldpeak: oldpeak = ST [Numeric value measured in depression]
11) ST_Slope: the slope of the peak exercise ST segment [Up: upsloping, Flat: flat, Down: downsloping]
12) HeartDisease: output class [1: heart disease, 0: No heart disease]

Initial Analysis



Dataset Example

Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	xerciseAngin	Oldpeak	ST_Slope	HeartDisease
40	M	ATA	140	289	0	Normal	172	N	0	Up	0
49	F	NAP	160	180	0	Normal	156	N	1	Flat	1
37	M	ATA	130	283	0	ST	98	N	0	Up	0
48	F	ASY	138	214	0	Normal	108	Y	1.5	Flat	1
54	M	NAP	150	195	0	Normal	122	N	0	Up	0
39	M	NAP	120	339	0	Normal	170	N	0	Up	0
45	F	ATA	130	237	0	Normal	170	N	0	Up	0
54	M	ATA	110	208	0	Normal	142	N	0	Up	0
37	M	ASY	140	207	0	Normal	130	Y	1.5	Flat	1
48	F	ATA	120	284	0	Normal	120	N	0	Up	0
37	F	NAP	130	211	0	Normal	142	N	0	Up	0
58	M	ATA	136	164	0	ST	99	Y	2	Flat	1
39	M	ATA	120	204	0	Normal	145	N	0	Up	0
49	M	ASY	140	234	0	Normal	140	Y	1	Flat	1
42	F	NAP	115	211	0	ST	137	N	0	Up	0
54	F	ATA	120	273	0	Normal	150	N	1.5	Flat	0
38	M	ASY	110	196	0	Normal	166	N	0	Flat	1
43	F	ATA	120	201	0	Normal	165	N	0	Up	0

Data Cleaning

	Cholesterol	RestingBP
count	918.000000	918.000000
mean	198.799564	132.396514
std	109.384145	18.514154
min	0.000000	0.000000
25%	173.250000	120.000000
50%	223.000000	130.000000
75%	267.000000	140.000000
max	603.000000	200.000000

	Cholesterol	RestingBP
count	918.000000	918.000000
mean	244.635389	132.540894
std	53.318029	17.989932
min	85.000000	80.000000
25%	214.000000	120.000000
50%	244.635389	130.000000
75%	267.000000	140.000000
max	603.000000	200.000000

Data Cleaning (cont.)

```
# replacing missing cholesterol values
# with the mean cholesterol
arr = np.array([X['Cholesterol']])
mean = arr[np.nonzero(arr)].mean()
X['Cholesterol'].replace({0: mean}, inplace=True)

# replacing missing restingbp values
# with the mean restingbp
arr = np.array([X['RestingBP']])
mean = arr[np.nonzero(arr)].mean()
X['RestingBP'].replace({0: mean}, inplace=True)
```

Data Transformation

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope	HeartDisease
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

	Age	RestingBP	FastingBS	MaxHR	Oldpeak	HeartDisease	Cholesterol	Male	Female	CPT_ASY	...	CPT_NAP	CPT_TA	ECG_LVH	ECG_Normal	ECG_ST	EA_N	EA_Y	ST_Down	ST_Flat	ST_Up
0	40	140	0	172	0.0	0	289.0	1	0	0	...	0	0	0	1	0	1	0	0	0	1
1	49	160	0	156	1.0	1	180.0	0	1	0	...	1	0	0	1	0	1	0	0	1	0
2	37	130	0	98	0.0	0	283.0	1	0	0	...	0	0	0	0	1	1	0	0	0	1
3	48	138	0	108	1.5	1	214.0	0	1	1	...	0	0	0	1	0	0	1	0	1	0
4	54	150	0	122	0.0	0	195.0	1	0	0	...	1	0	0	1	0	1	0	0	0	1

Data Transformation (cont.)

```
# doing one hot encoding for the categorical features
ohe_sex = pd.get_dummies(df.Sex, prefix='Sex')
ohe_chestpain = pd.get_dummies(df.ChestPainType, prefix='ChestPainType')
ohe_restingecg = pd.get_dummies(df.RestingECG, prefix='RestingECG')
ohe_exerciseangina = pd.get_dummies(df.ExerciseAngina, prefix='ExerciseAngina')
ohe_stslope = pd.get_dummies(df.ST_Slope, prefix='ST_Slope')
```

```
# adding the encoded features
X['Male'] = ohe_sex['Sex_M']
X['Female'] = ohe_sex['Sex_F']

X['CPT_ASY'] = ohe_chestpain['ChestPainType_ASY']
X['CPT_ATA'] = ohe_chestpain['ChestPainType_ATA']
X['CPT_NAP'] = ohe_chestpain['ChestPainType_NAP']
X['CPT_TA'] = ohe_chestpain['ChestPainType_TA']
```

```
X['ECG_LVH'] = ohe_restingecg['RestingECG_LVH']
X['ECG_Normal'] = ohe_restingecg['RestingECG_Normal']
X['ECG_ST'] = ohe_restingecg['RestingECG_ST']

X['EA_N'] = ohe_exerciseangina['ExerciseAngina_N']
X['EA_Y'] = ohe_exerciseangina['ExerciseAngina_Y']

X['ST_Down'] = ohe_stslope['ST_Slope_Down']
X['ST_Flat'] = ohe_stslope['ST_Slope_Flat']
X['ST_Up'] = ohe_stslope['ST_Slope_Up']
```

Procedures

1) Logistic Regression
2) Support Vector Machine (SVM)
3) Cross-Validation for Both Models
4) Results

Logistic Regression

- Out of box sklearn Logistic Regression
- C value of 1
- Used ⅔ of the data for training, ⅓ for testing
- Scaled our data with StandardScaler()

```
# scaling the data with StandardScaler()
scaler = preprocessing.StandardScaler().fit(X)
X = scaler.transform(X)
# split into training and test data
X_train, X_test, Y_train, Y_test = train_test_split(X, y, test_size=0.33, random_state=42)
```

```
# doing the logreg
logreg = linear_model.LogisticRegression(C = 1)
logreg.fit(X_train, Y_train)
logreg_pred = logreg.predict(X_test)
```

Logistic Regression Results

LogReg Confusion Matrix	Predicted Positive	Predicted Negative
Actual Positive	161	19
Actual Negative	19	123

$$\textit{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Logistic Regression Accuracy: 0.858

SVM

- Out of box sklearn support vector machine
- C value of 1
- Linear kernel
- Used ⅔ of the data for training, ⅓ for testing
- Scaled our data with StandardScaler()

```
# scaling the data with StandardScaler()
scaler = preprocessing.StandardScaler().fit(X)
X = scaler.transform(X)
# split into training and test data
X_train, X_test, Y_train, Y_test = train_test_split(X, y, test_size=0.33, random_state=42)
```

```
# doing the svm
clf = svm.SVC(kernel='linear')
clf.fit(X_train, Y_train)
svm_pred = clf.predict(X_test)
```

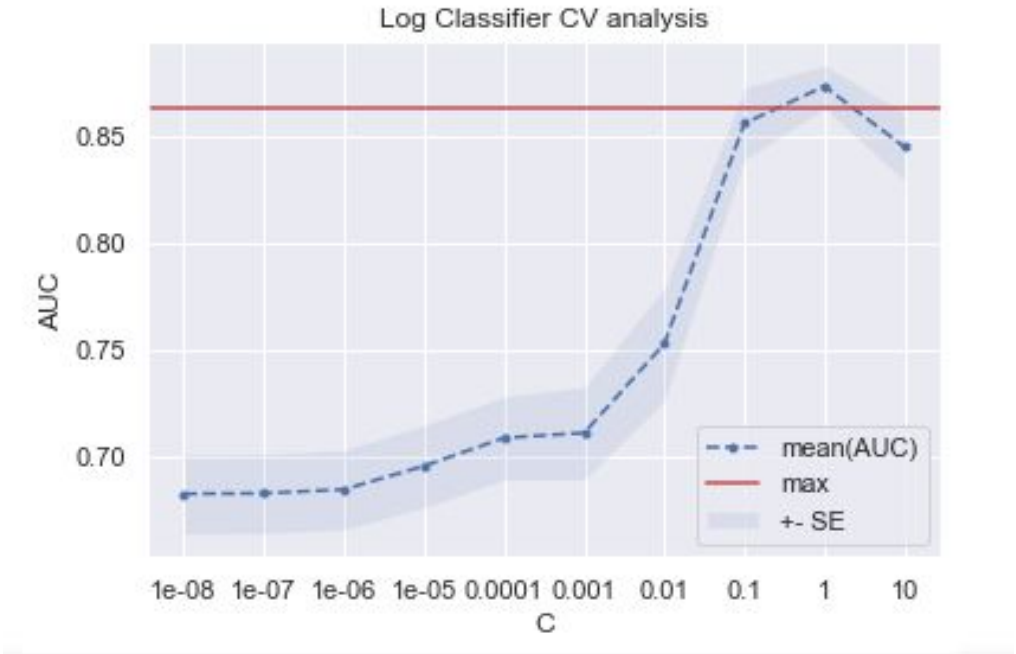
SVM Results

SVM Confusion Matrix	Predicted Positive	Predicted Negative
Actual Positive	167	13
Actual Negative	13	136

$$\textit{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

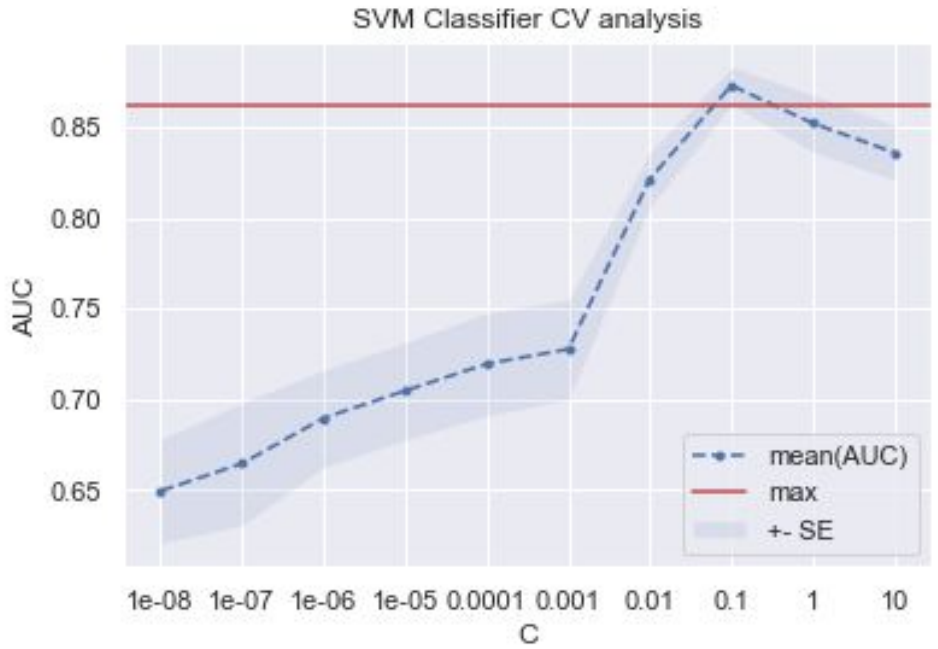
SVM Accuracy: 0.871

Cross Validation(LogReg)



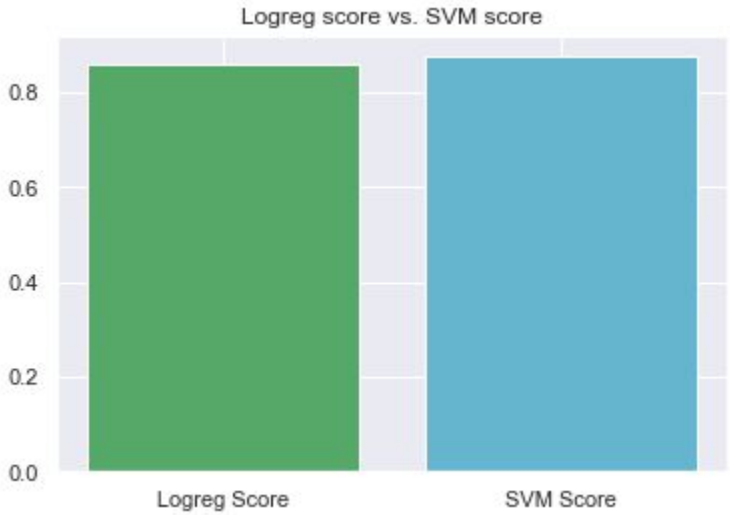
Logistic Regression Optimal C = 0.1

Cross Validation(SVM)



SVM Optimal C = 0.1

Results



LogReg2 Confusion Matrix	Predicted Positive	Predicted Negative
Actual Positive	163	17
Actual Negative	17	123

SVM2 Confusion Matrix	Predicted Positive	Predicted Negative
Actual Positive	168	12
Actual Negative	12	123

	C = 0.1	C = 1
LogReg Accuracy	0.865	0.858
SVM Accuracy	0.875	0.871

Conclusion

- We can predict heart failure given other medical information about the patient with ~87% accuracy.
- Both models are able to perform very well in predicting the results.
- Importance of Data Cleaning
- Future Plans

Works Cited

- https://www.kaggle.com/datasets/fedesoriano/heart-failure-prediction?resource=download

Predicting Heart Failure

Brian Guo, Winston Yang, Tiffany Yan