By: Tim Remmert and Anam Ejaz

About the Project

- Cardiovascular diseases (CVDs) are the number 1 cause of death globally, taking an estimated 17.9 million lives each year, which accounts for 31% of all deaths worldwide.
- Four out of 5 CVD deaths are due to heart attacks and strokes, and one-third of these deaths occur prematurely in people under 70 years of age.
- Heart failure is a common event caused by CVDs and the dataset used for this project contains 11 features that can be used to predict possible heart disease.
- People with cardiovascular disease or who are at high cardiovascular risk (due to the presence of one or more risk factors such as hypertension, diabetes, hyperlipidaemia or already established disease) need early detection and management that a accurate machine learning model can provide.

About the Features in the Dataset

1. **Age:** age of the patient [years]
2. **Sex:** sex of the patient [M: Male, F: Female]
3. **ChestPainType:** chest pain type [TA: Typical Angina, ATA: Atypical Angina, NAP: Non-Anginal Pain, ASY: Asymptomatic]
4. **RestingBP:** resting blood pressure [mm Hg]
5. **Cholesterol:** serum cholesterol [mm/dl]
6. **FastingBS:** fasting blood sugar [1: if FastingBS > 120 mg/dl, 0: otherwise]
7. **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]
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: Normal]

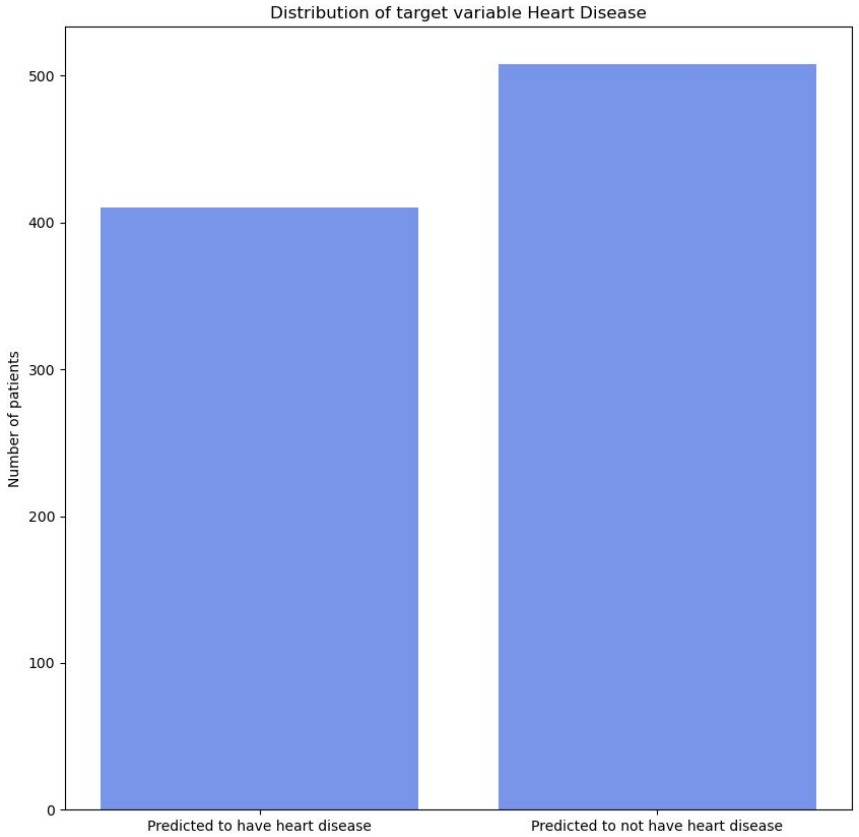
*Angina is chest pain or discomfort caused when your heart muscle doesn't get enough oxygen-rich blood

Data Preprocessing and Exploration

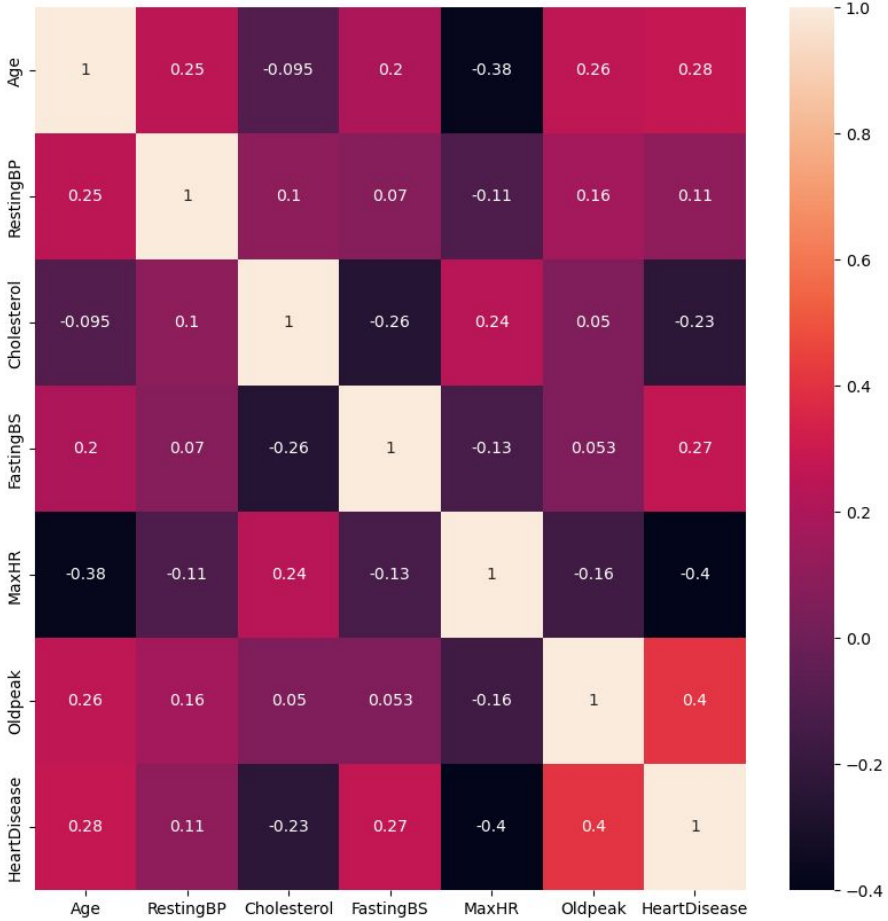
- The dataset set has 918 rows and 12 columns in total
- There is no missing values present and no duplicate rows

	Age	RestingBP	Cholesterol	FastingBS	MaxHR	Oldpeak	HeartDisease
Count	918	918	918	918	918	918	918
Mean	53.51	132.40	198.80	0.23	136.80	0.89	0.55
Std	9.43	18.514	109.38	0.42	25.46	1.07	0.50
Min	28	0	0	0	60	-2.6	0
25%	47	120	173.25	0	120	0	0
50%	54	130	223	0	138	0.6	1
75%	60	140	267	0	156	1.5	1
Max	77	200	603	1	202	6.2	1

Distribution of Target Variable Heart Disease



Quantitative Value Correlation Heatmap



Conversion of Categorical Variables

- We can convert our categorical variables Sex, ChestPainType, RestingECG, ExerciseAngina, ST_Slope into numerical ones by using Python pandas get_dummies function to create dummy variables for each
- The function will create a new column for each categorical variable possible value and fill each row with 0 or 1 with 1 meaning it has the variable value and 0 meaning it does not
- For example for Sex variable it will create two new variables M and F for male and female and fill all male rows with 1 and 0 otherwise
- We can use the drop_first = True option to omit the unnecessary extra column for binary categorical variables such as sex since the value of one can completely determine the other and prevent introducing colinearity to our dataset which may impact the model

Splitting the Data for training and testing

- We split the data by using 30% to use for testing and 70% to use for training
- Our training data set is now 642 rows and the test data set is 276 rows

About the ROC and AUC

The Receiver Operator Curve (ROC) is a plot that shows the effect on the true positive rate and false positive rate as the threshold for classification decreases.

- **True Positive Rate (TPR) = TP/(TP +FN)**
- **False Positive Rate (FPR) = FP/(FP + TN)**

The Area under the curve (AUC) is the area under the constructed ROC curve (a value between 0 and 1) and can be used to compare two different ROC curves. If the AUC for one classifier is greater than the other we can then consider that classifier more accurate under all possible thresholds**.**

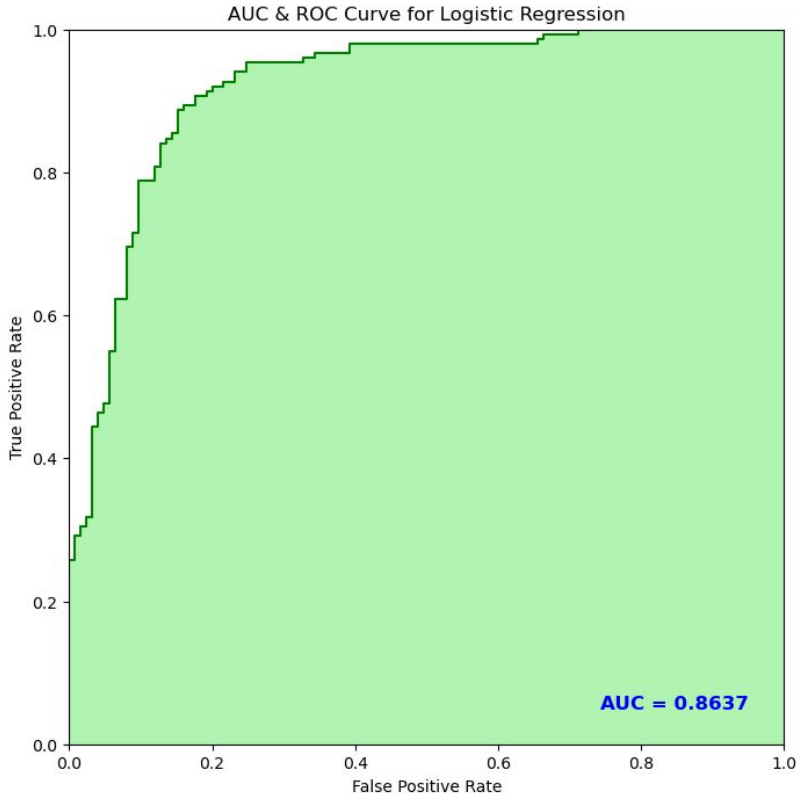
Classification with Logistic Regression

- Accuracy: 87%

Confusion Matrix:

	Will Develop Heart Disease	Will Not Develop Heart Disease
Will Develop Heart Disease	103	21
Will Not develop Heart Disease	12	140

How will changing the threshold value effect the TPR and FPR?



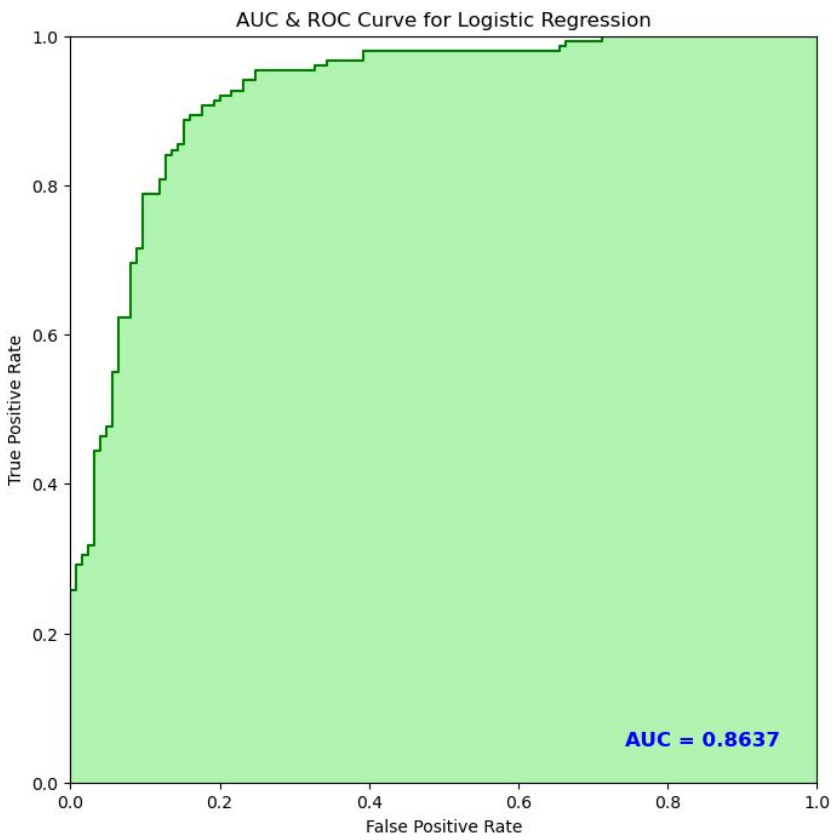
Classification Using LDA

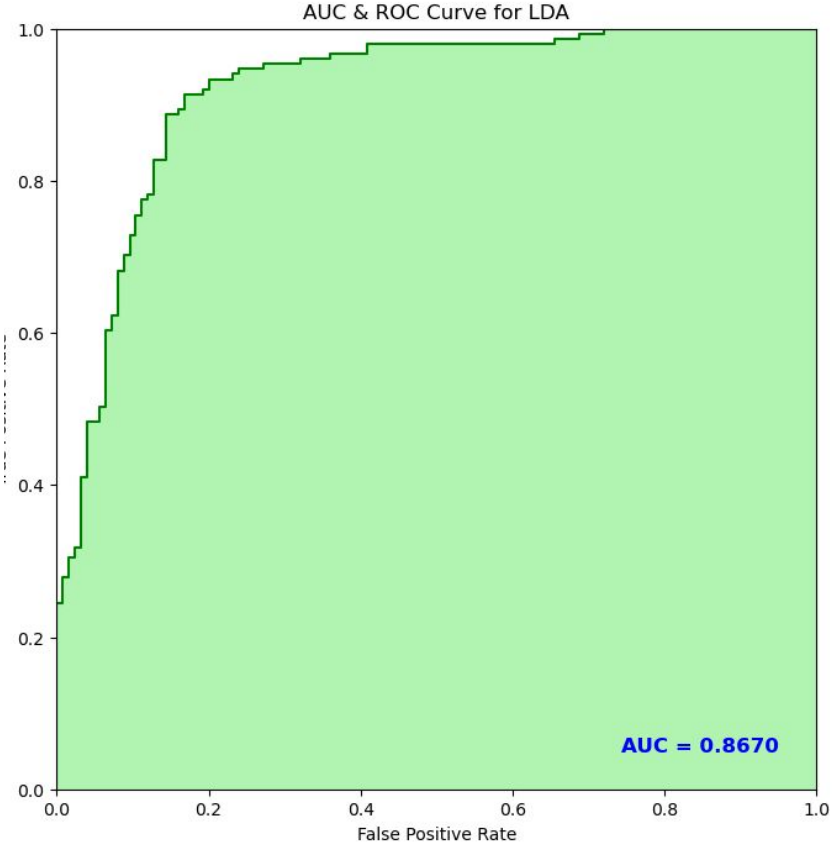
- Accuracy: 87%

Confusion Matrix:

	Will Develop Heart Disease	Will Not Develop Heart Disease
Will Develop Heart Disease	105	20
Will Not Develop Heart Disease	16	135

There is no significant difference between LDA and Logistic Regression Performance





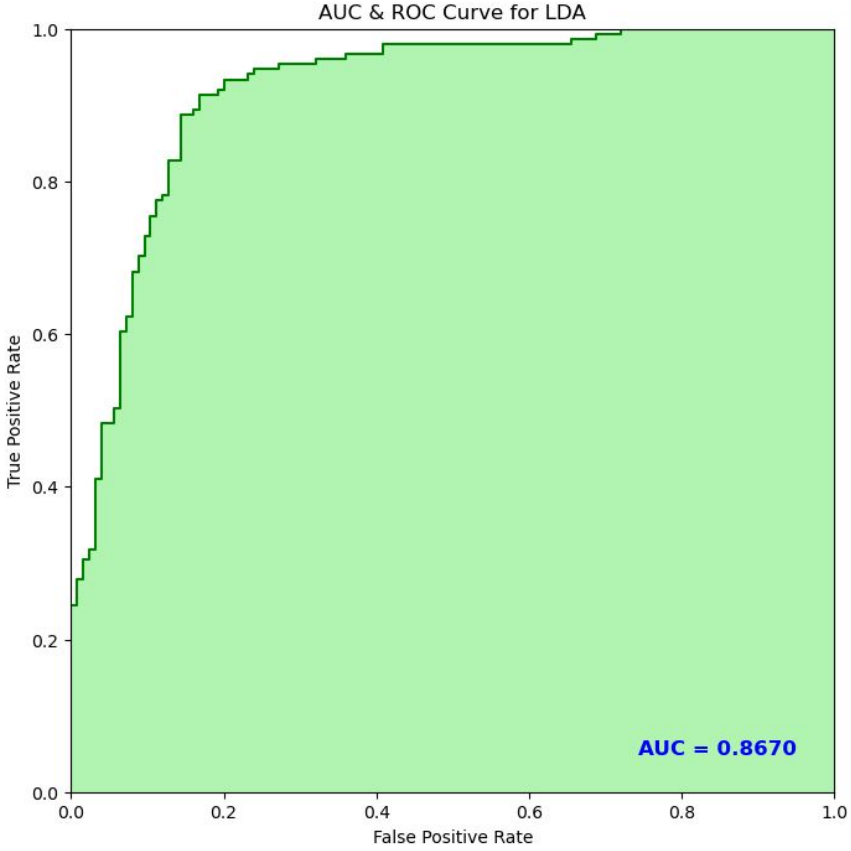
QDA Classification Results

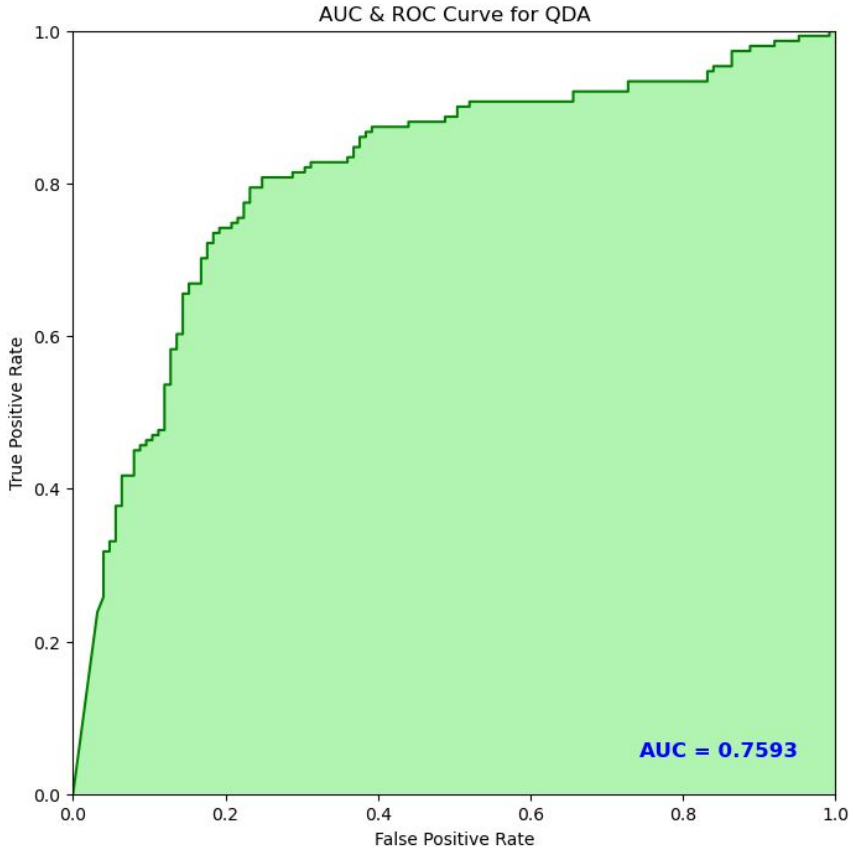
- Accuracy: 76%

Confusion Matrix:

	Will Develop Heart Disease	Will Not Develop Heart Disease
Will Develop Heart Disease	88	37
Will Not Develop Heart Disease	28	123

QDA is the Worse Classification performance so far suggesting the model is too flexible for the data





Using Random Forest As Classifier

- How can we set the hyperparameter n_estimator (the number of decision trees to create) before training and testing the model?
- We can use grid search in sklearn package to use cross validation on the training set for each possible combination of parameters we need to set and determine which combination has the highest accuracy
- For possible values of 50, 100, 500 total decision trees our grid search determined 500 decision trees had the best accuracy of 0.87 on the training data

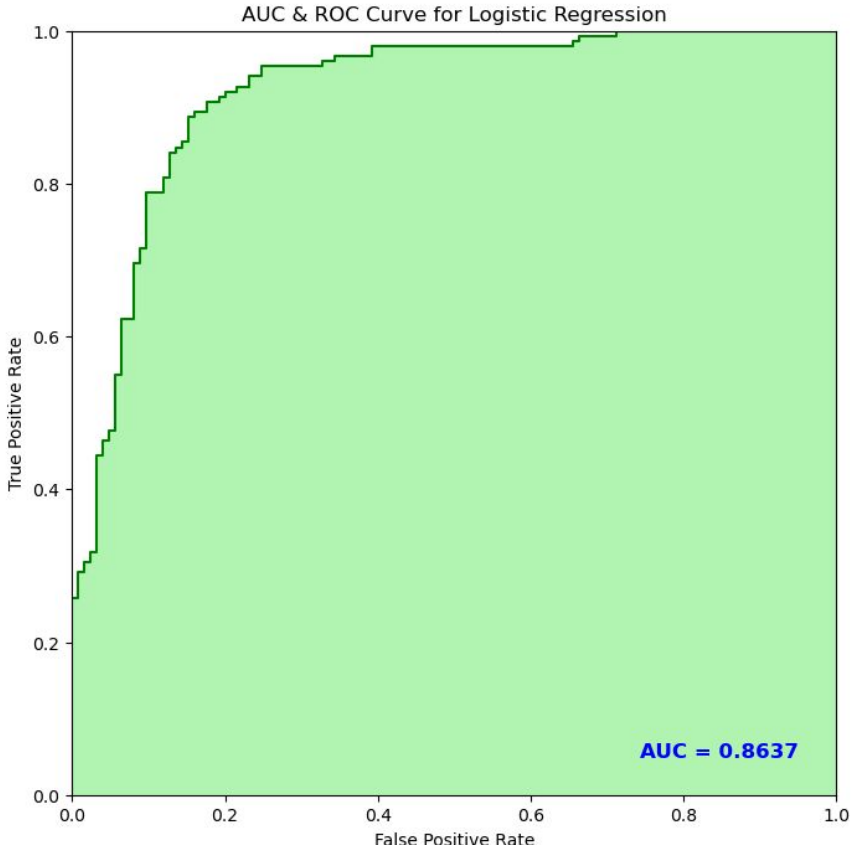
Random Forest Classification Results

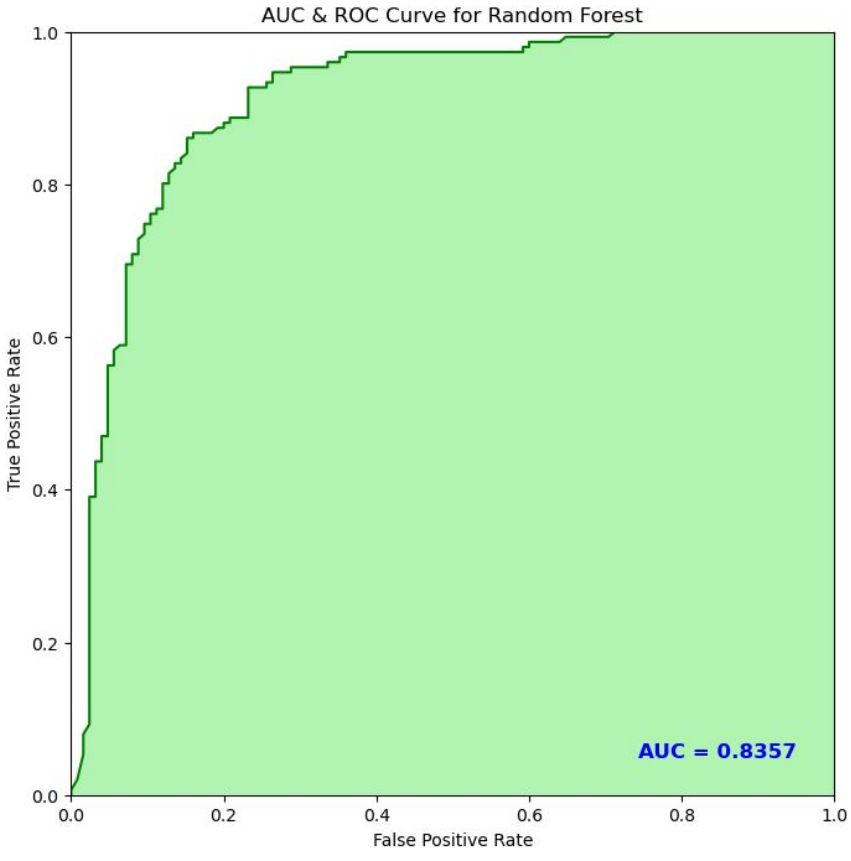
- Accuracy: 84%

Confusion Matrix:

	Will Develop Heart Disease	Will Not Develop Heart Disease
Will Develop Heart Disease	98	27
Will not Develop Heart Disease	17	134

Random Forest performs worse than LDA and Logistic Regression probably due to overfitting





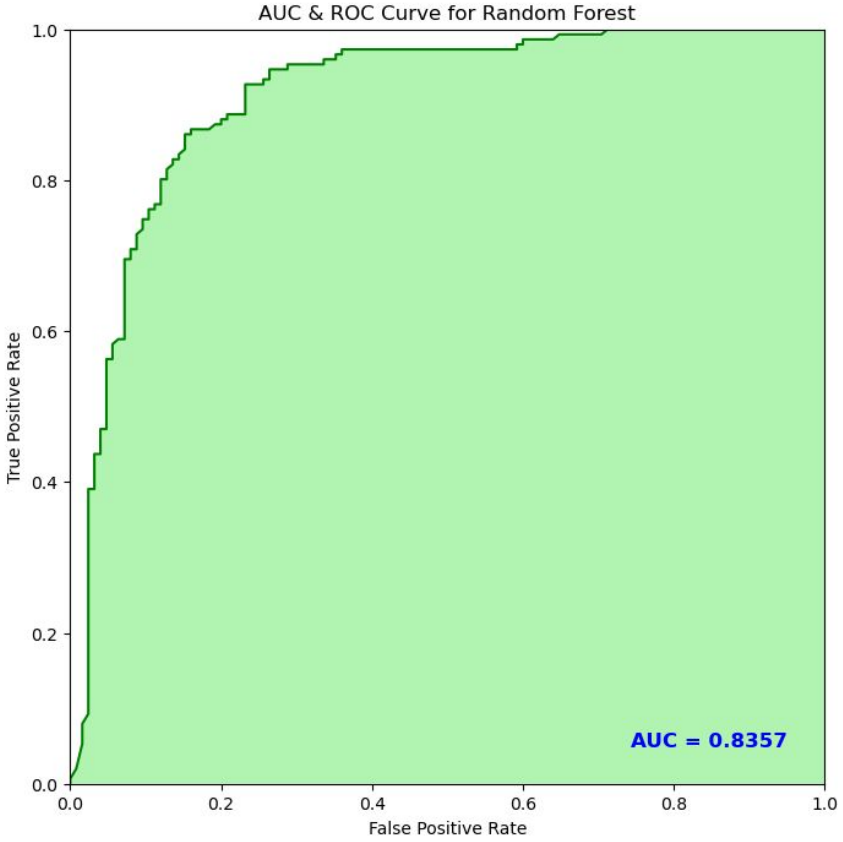
Will Bagging Improve the Test Results?

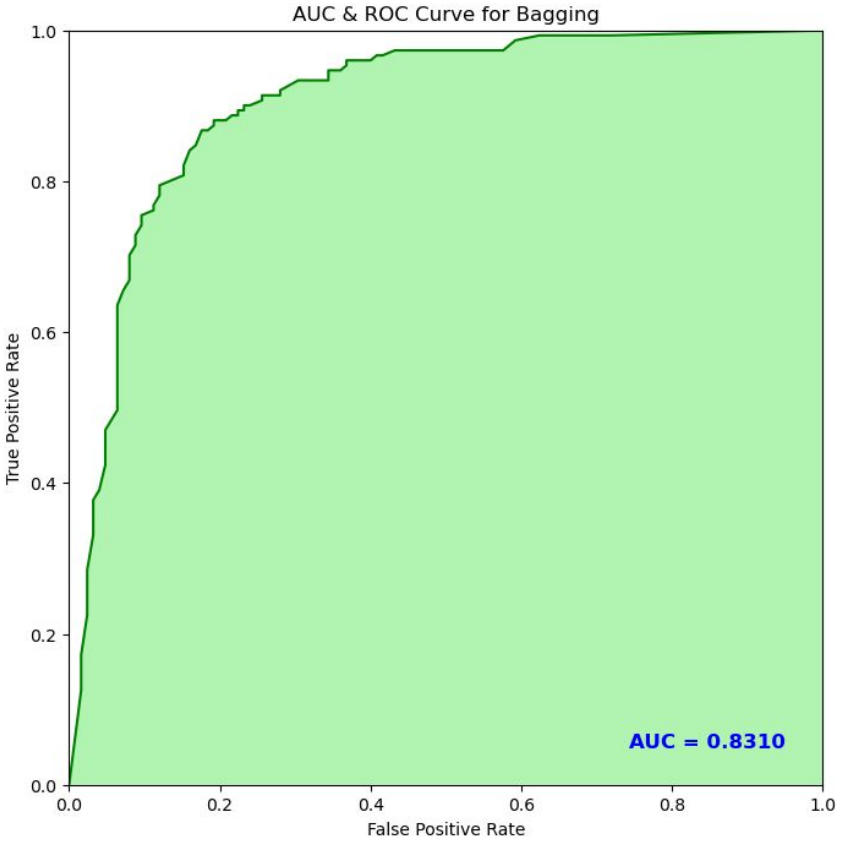
- Using Grid search again for the number of estimators returns 100
- Accuracy: 84%

Confusion Matrix:

	Will Develop Heart Disease	Will Not Develop Heart Disease
Will Develop Heart Disease	96	29
Will Not Develop Heart Disease	16	135

Bagging Did not Result in a Significant improvement over Random Forest





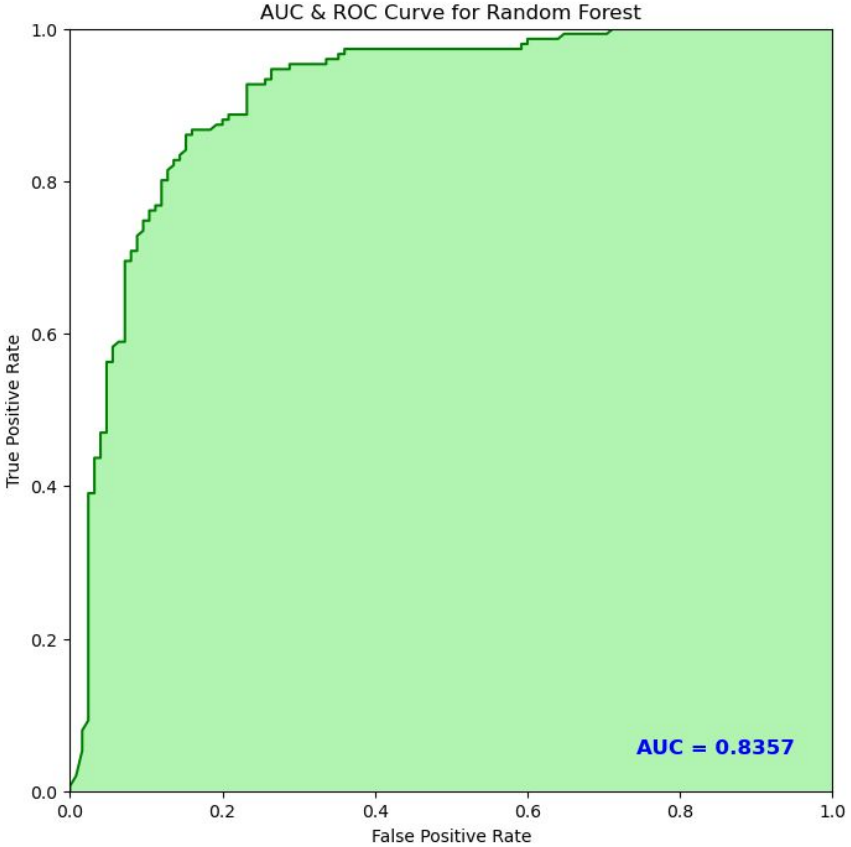
What about Boosting?

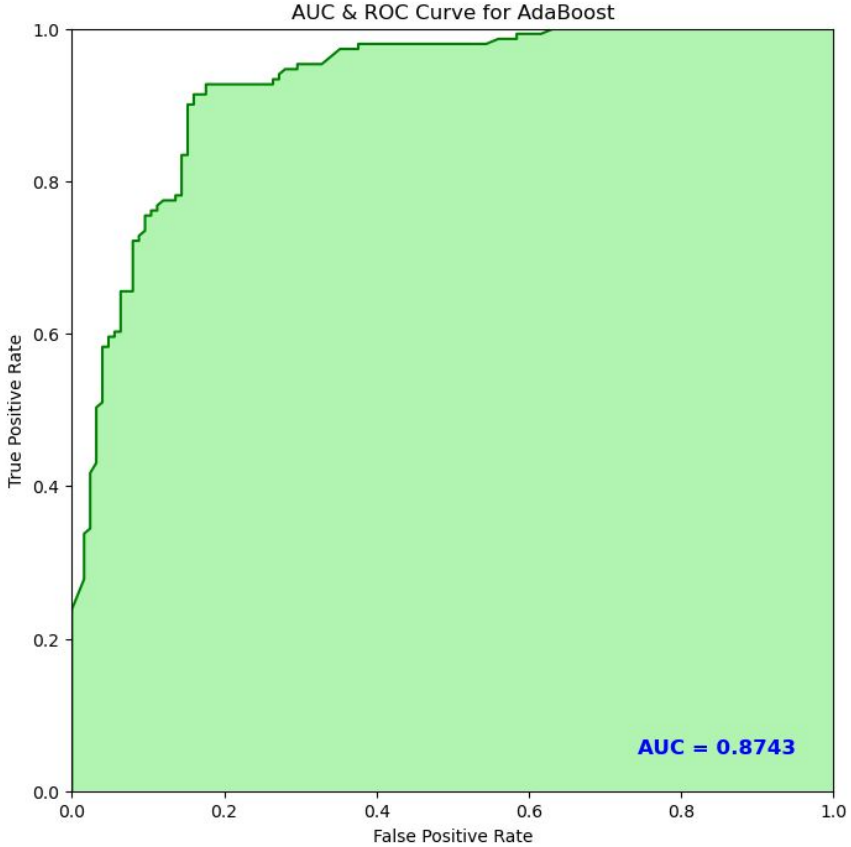
- Using Python AdaBoost implementation we use grid search to find the best combination of parameters number of estimators (50, 100, 500) and the learning rate (000.1, 00.1,0.1,1) to find the best combination as number of estimators = 50 and learning rate = 0.1
- Accuracy: 88%

Confusion Matrix:

	Will Develop Heart Disease	Will Not Develop Heart Disease
Will Develop Heart Disease	106	19
Will Not Develop Heart Disease	15	136

We Can See Boosting Significantly improved Random forest performance to the best classifier so far





Summary of Results

Rank	Classifier	Accuracy
1	AdaBoost	88%
2	LDA	87%
2	Logistic Regression	87%
3	Random Forest	84%
3	Bagging	84%
4	QDA	76%

Remarks

- Due to potential medical applications of this model we may want to decrease the threshold for classification to increase the amount of true positives we get so as few as possible at risk patients are classified as not at risk
- QDA worse performance than LDA may signify that flexible models will not be as effective for this data set
- We would Recommend AdaBoost (Adaptive boosting) on this classification task due to it superior accuracy to the other tested methods.