Using Classification to Predict Heart Failure

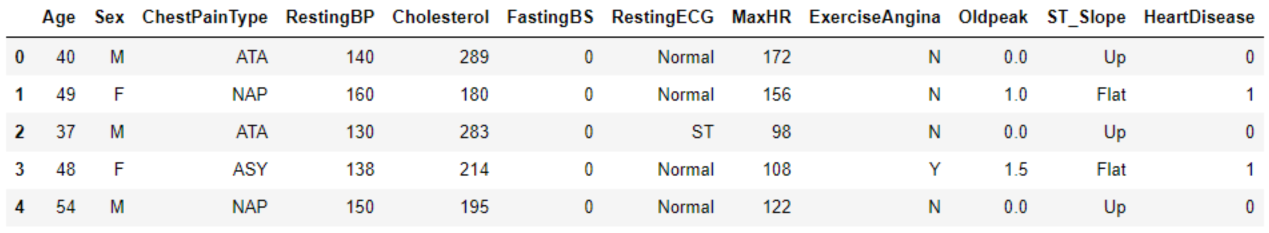
Nan He

1 Main objective of the analysis

The data I will be using is a heart failure prediction dataset from Kaggle, the link for the data set is here. The goal of this report is to find a model that predict possible heart failure. I will clean the data, and use three different classification method to build the model. The hyperparameters of each model will be benchmarked using cross-validation. The model will be tested on the hold out set, using accuracy, F1 score, and ROC curve as metrics.

2 Description of the data set

The data set includes 11 clinical features for predicting heart disease events and 918 labels in total. The categorical features are "Sex", "ChestPainType", "FastingBS", "RestingECG", "ExerciseAngina", and "ST_Slope", of which "Sex", "FastingBS", and "ExerciseAngina" are binary. The numerical features are "Age", "RestingBP", "Cholesterol", "MaxHR", and "Oldpeak". The target is "HeartDisease", which is a binary feature, indicating the occurance of the heart disease event. Example data looks like:



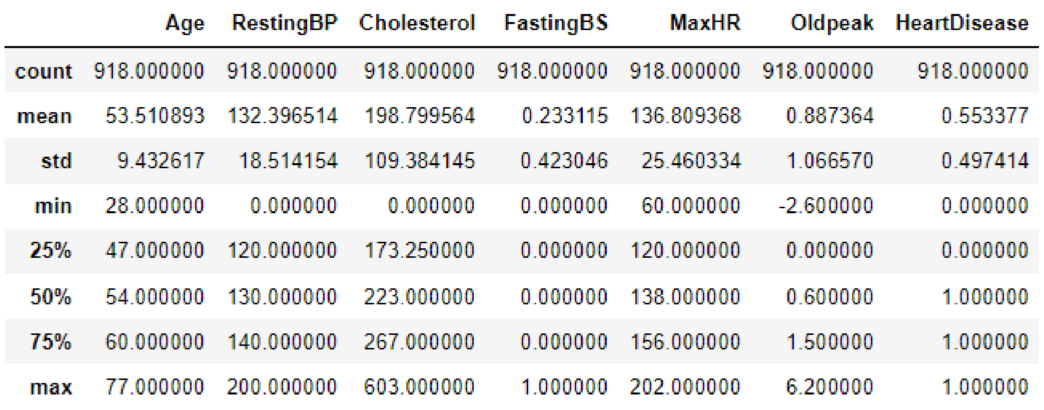
	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope	HeartDisease
0	40	M	ATA	140	289	0	Normal	172	N	0.0	Up	0
1	49	F	NAP	160	180	0	Normal	156	N	1.0	Flat	1
2	37	M	ATA	130	283	0	ST	98	N	0.0	Up	0
3	48	F	ASY	138	214	0	Normal	108	Y	1.5	Flat	1
4	54	M	NAP	150	195	0	Normal	122	N	0.0	Up	0

Figure 1: Example data before cleaning.

3 Data exploration, cleaning, and feature engineering

The description of the data is:

I firstly check whether there is any empty data, and find none. The pairplot of all numerical data is shown, which show good predictability.



	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

Figure 2: Data descriptions.

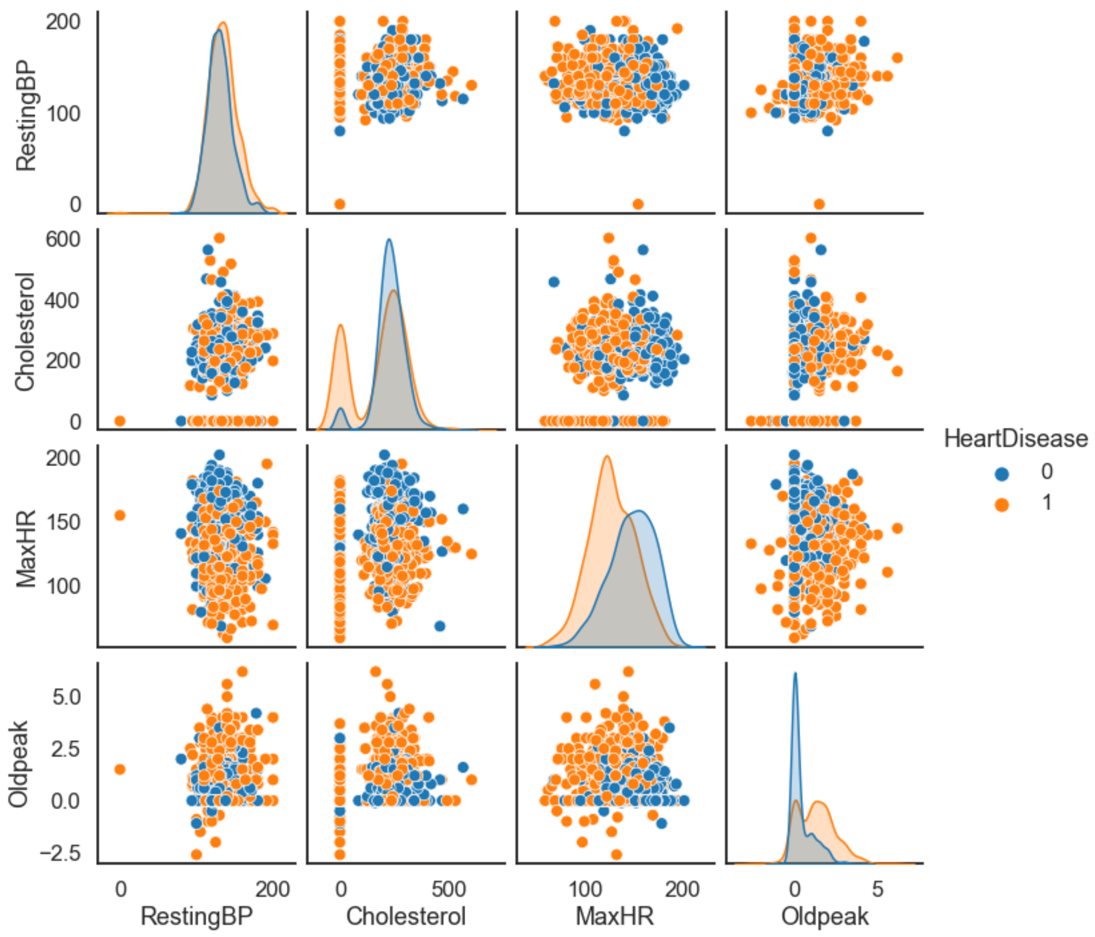


Figure 3: Pairplot of all numerical data.

The next step is to encode the categorical data, since we are going to use some models that requires numerical inputs. For binary data ”Sex”, ”FastingBS”, and ”ExerciseAngina”, I encode them as 0 and 1 in place; while for "ChestPainType", "RestingECG", and "ST_Slope", I do one-hot encoding. The number of features after encoding is 17.

The next step is to check whether the data is balances, to decide whether we need to upsample or downsample after the train-test split. The target "HeartDisease" has 508 "1" and 410 "0", which is fairly balanced; thus no need for using resample methods.

Before running the model, we split the data into a train set and a test set, holding 25% of data out of the model training procedure. The labels in the training set is 688. The numerical features are then scaled using standard scalar. This step eliminate the bias in algorithms based on distances. The data is now ready to train.

4 Testing different models

The first model I tested is a simple **logistic regression** with L2 regularization. The second model is a **SVM classifier** with different kernels and regularizations. The third model is a **Random Forest classifier** with various criterion and max features for tree split.

To tune the model hyperparameter, the grid search cross-validation is performed on regularization parameter "C" for all model. For SVM classifier, the kernel type (linear, polynomial, radial basis functions, sigmoid) and the gamma value of the gaussian functions are tuned. For Ramdom Forest classifier, the tree-split criterion (Gini, entropy), the max features for each split, and the number of estimators are tuned.

After cross-validation, all model are tested on the test set, the results are shown in various metrics:

Table 1: Results for all three models on the test set, LR stands for Logistic regression, SVC for SVM classifier, RFC for Random Forest classifier.

Model Name	True Positive	True Negative	False Negative	False Positive	Accuracy	F1 score	ROC AUC
LR	112	86	20	12	0.861	0.875	0.937
SVC	112	87	20	11	0.865	0.878	0.932
RFC	117	88	15	10	0.891	0.903	0.945

The ROC curves for all three models are:

5 Final model choices and analysis

From the results, the Random forest model with hyper-parameter tuned performs best on the test set. The hyperparameter for the model is:
Criterion: Gini, Max features: 2, Estimators: 200. However, the Random forest model tooks the longest time to train and tune, while logistic regression and Support Vector Machine are much faster. If the data set is larger, the hyper-parameter tuning of Random forest could be difficult.

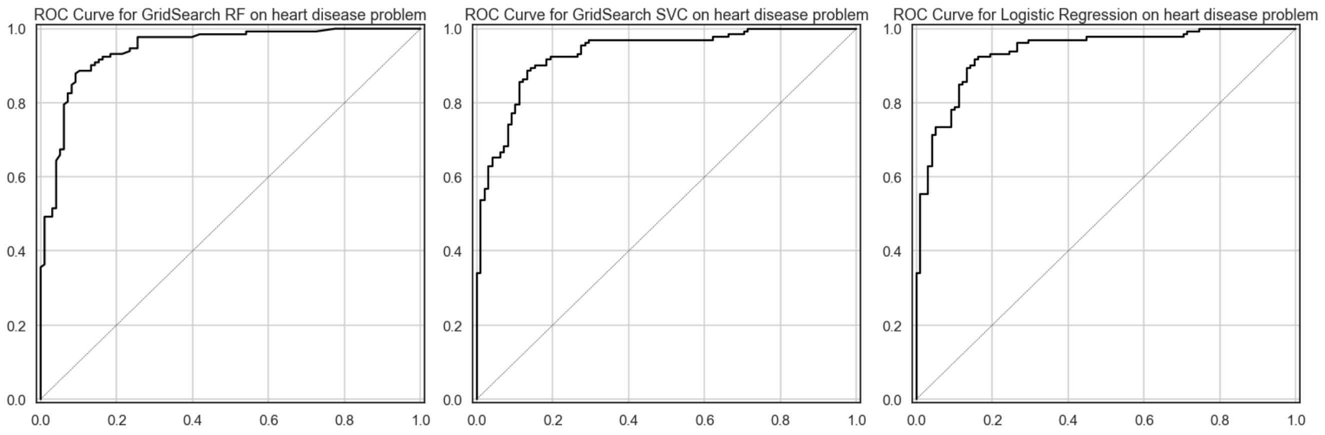


Figure 4: ROC curve for LR, SVC, and RFC models.

6 Key Findings and Insights

One of the major finding is that the grid-search cross-validation is vital for Support Vector Machine classifiers and ensemble methods like Random forest classifiers. For example, if I simply use SVC with radial basis function, gamma= 1.0 and $C = 10.0$, the accuracy on the test set will be only 0.748, and produce 50 false positive, much worse than simple logistic regression. Therefore, to use SVC and RFC, the cross-validation procedure is very important and cannot be skipped.

7 Summary and suggestions for next steps

In summary, this report provide a tuned model to predict heart-failure events, with 0.891 accuracy. The possible next step is to further tun the hyper-parameters, and try more models. Another possible direction is to consider ignoring unimportant features.