MACHINE LEARNING: CLASSIFICATION

Heart Disease Prediction

By
Siddarth CB

CONTENTS

- Data Description
- Main Objectives
- Various Classifier Models
- Key Findings
- Flaws and Improvement

Data

Description

INTRODUCTION

Predicting and diagnosing heart disease is a challenging task in the medical industry since it depends on several factors and includes analysis of several physical examinations and patient symptoms.

Heart diseases are some of the most deadliest diseases since heart is one of the most vital organs responsible for most body functions.

Heart disease can be predicted based on different symptoms like age, gender, heart rate etc. which in turn reduces the death rate for heart patients. In this report we are going to use machine learning algorithms and Python language to do that!

	Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	Slope	Ca	Thal	AHD
1	63	1	typical	145	233	1	2	150	0	2.3	3	0	fixed	No
2	67	1	asymptomatic	160	286	0	2	108	1	1.5	2	3	normal	Yes
3	67	1	asymptomatic	120	229	0	2	129	1	2.6	2	2	reversable	Yes
4	37	1	nonanginal	130	250	0	0	187	0	3.5	3	0	normal	No
5	41	0	nontypical	130	204	0	2	172	0	1.4	1	0	normal	No

Age: age of person in years

Sex: gender of person
- 1: Male
- 0: Female

ChestPain: chest pain type of person (asymptomatic=silent ischemia

RestBP: resting blood pressure of person (in mm of Hg)

Chol: cholesterol of person in mg/dl

Fbs: fasting blood sugar of the person (>120 md/dl = 1)

MaxHR: maximum heart rate achieved

RestECG: resting electrocardiograph results
0: Left Ventricular Hypertrophy
1: Normal
2: ST-T Wave Abnormality

ExAng: exercise induced angina

Oldpeak: exercise induced ST depression

Slope: slope of peak exercise ST
1: Upsloping
2: Flat
3: Downsloping

Ca: number of major vessels

Thal: thalassemia

AHD: target variable (yes/no)

	SNo	Age	Sex	RestBP	Chol	Fbs
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000
mean	152.000000	54.438944	0.679868	131.689769	246.693069	0.148515
std	87.612784	9.038662	0.467299	17.599748	51.776918	0.356198
min	1.000000	29.000000	0.000000	94.000000	126.000000	0.000000
25%	76.500000	48.000000	0.000000	120.000000	211.000000	0.000000
50%	152.000000	56.000000	1.000000	130.000000	241.000000	0.000000
75%	227.500000	61.000000	1.000000	140.000000	275.000000	0.000000
max	303.000000	77.000000	1.000000	200.000000	564.000000	1.000000

	RestECG	MaxHR	ExAng	Oldpeak	Slope	Ca
count	303.000000	303.000000	303.000000	303.000000	303.000000	299.000000
mean	0.990099	149.607261	0.326733	1.039604	1.600660	0.672241
std	0.994971	22.875003	0.469794	1.161075	0.616226	0.937438
min	0.000000	71.000000	0.000000	0.000000	1.000000	0.000000
25%	0.000000	133.500000	0.000000	0.000000	1.000000	0.000000
50%	1.000000	153.000000	0.000000	0.800000	2.000000	0.000000
75%	2.000000	166.000000	1.000000	1.600000	2.000000	1.000000
max	2.000000	202.000000	1.000000	6.200000	3.000000	3.000000

Remove Nulls

```
[15]  heart.dropna(axis=0, inplace=True)
      ✓ 0.0s
```

```
[16]  print(heart.isna().sum())
      ✓ 0.0s

...   SNo         0
      Age         0
      Sex         0
      ChestPain   0
      RestBP      0
      Chol        0
      Fbs         0
      RestECG     0
      MaxHR       0
      ExAng       0
      Oldpeak     0
      Slope       0
      Ca          0
      Thal        0
      AHD         0
      dtype: int64
```

Main
Objectives

IN THIS SECTION

- Corelation between features to find most influential features on the target

- Build different models based on advanced classification techniques

- Discuss flaws of different models

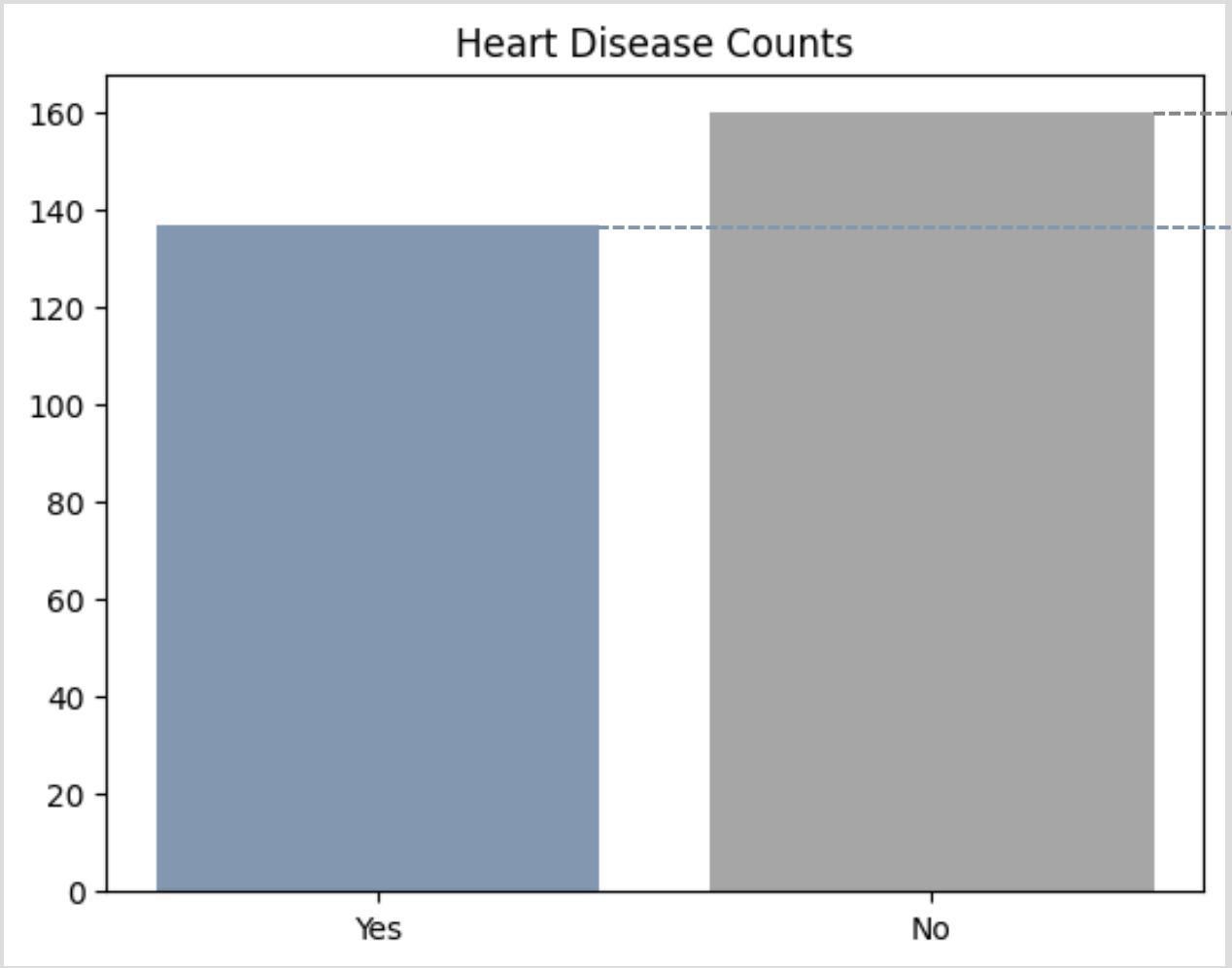
Possible Snags:

The analysis may encounter several snags in the preliminary data, including missing or imbalanced values, noisy or inconsistent records, and potential biases due to limited representation of certain age, gender, or ethnic groups. Addressing these challenges through preprocessing and appropriate validation strategies will be essential to ensure the reliability of model comparisons.

FEATURES

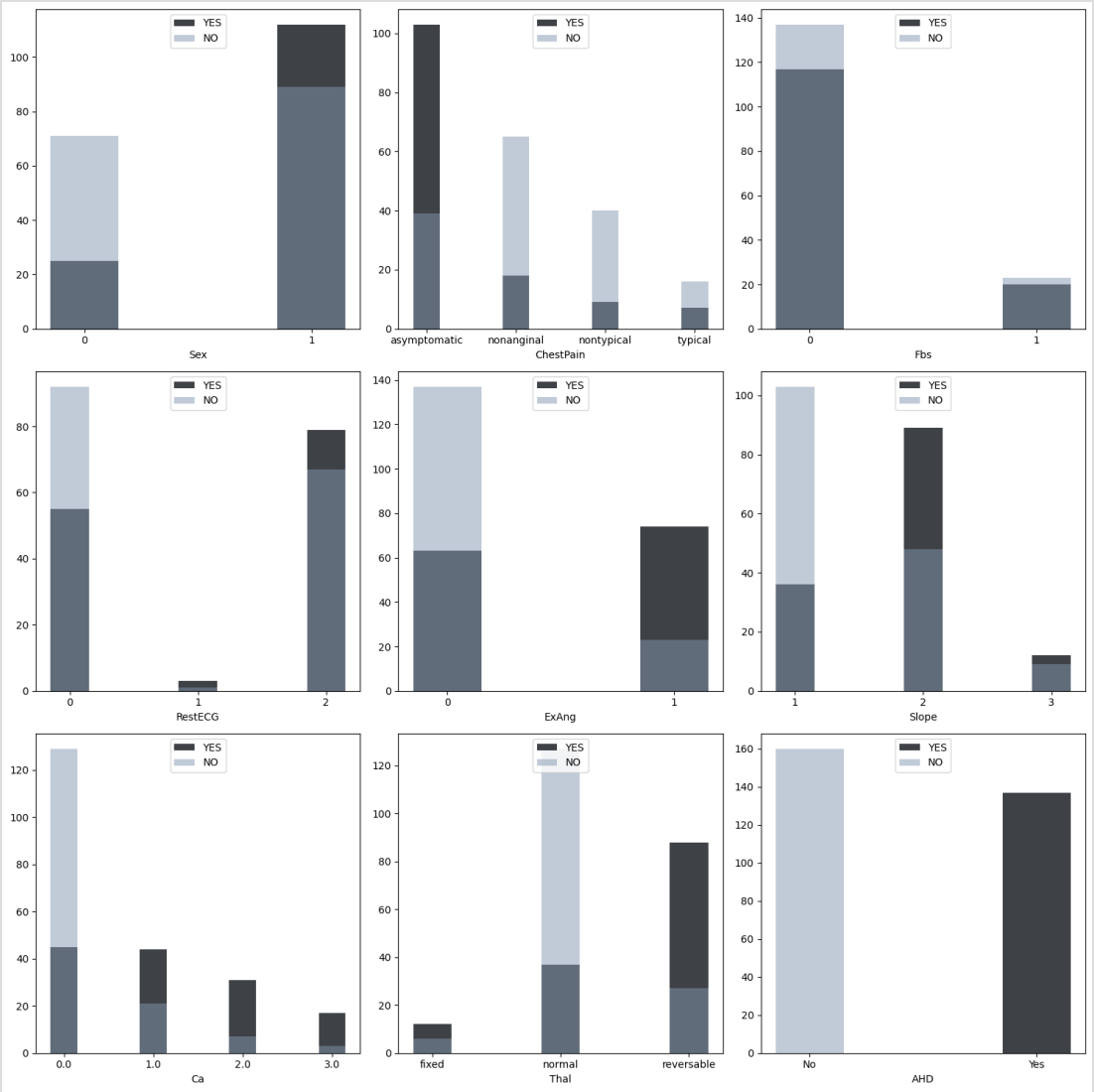
Categorical Features: [‘Sex’, ’ChestPain’, ‘Fbs’, ‘RestECG’, ‘ExAng’, ‘Slope’, ‘Ca’, ‘Thal’, ‘AHD’]

Continuous Features: [‘Age’, ’RestBP’, ‘Chol’, ‘MaxHR’, ‘OldPeak’]



160 No Heart Disease

137 With Heart Disease



INFERENCES

- Heart diseases more common in male subjects than female subjects

- Heart diseases found more in subjects with asymptomatic chest pain (silent ischemia)

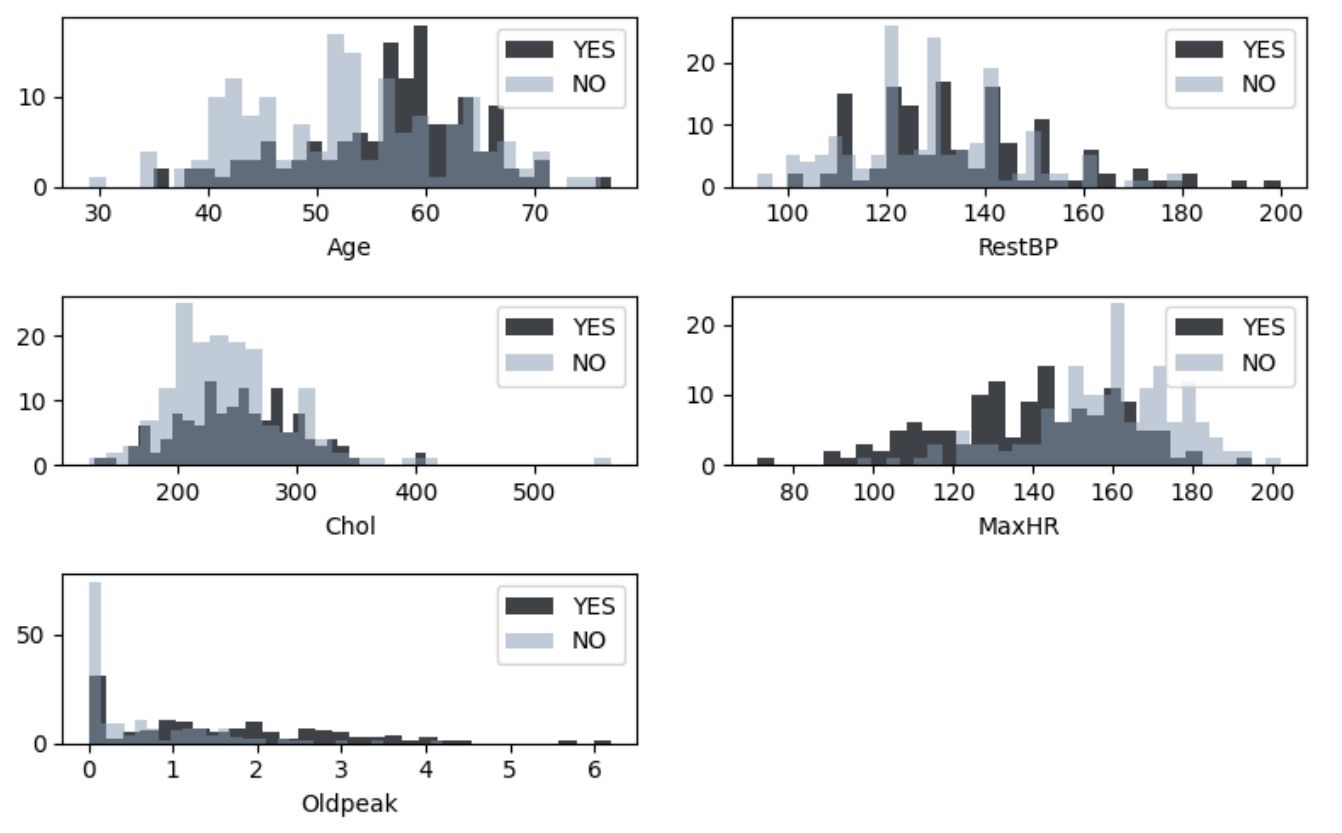
- Heart diseases found in subjects with ST wave abnormality

- Heart diseases indicated by exercise induced angina

- Heart diseases indicated by flat and down sloping peak exercise ST

- More blood vessels is more prone to heart disease

- Subjects with fixed and reversible thalassemia are more prone to heart disease while subjects with normal thalassemia are less prone to heart disease



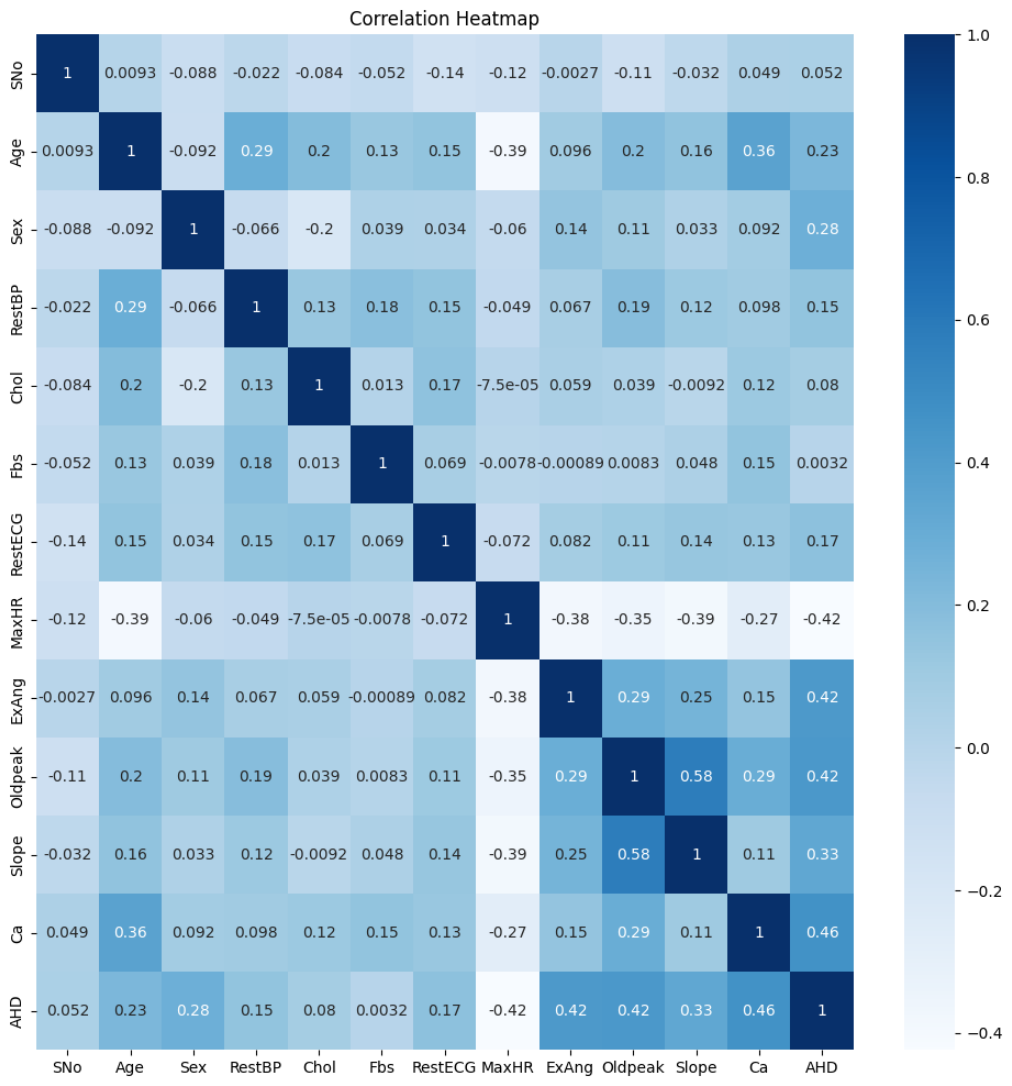
INFERENCES

- Heart diseases more common in subjects above 55 years of age

- Rest BP above 130 indicator of heart disease

- High cholesterol indicator of heart disease

- Lower max heart rate indicates lack of fitness and subsequently heart disease



INFERENCES

- Cholesterol and Fbs are least correlated with target variable

- Ca and Oldpeak are most correlated with target variable

Feature Encoding

	SNo	Age	RestBP	Chol	MaxHR	Oldpeak	Sex_0	Sex_1	ChestPain_asymptomatic	ChestPain_nonanginal	ChestPain_nontypical	ChestPain_typical	Fbs_0	Fbs_1	RestECG_0
0	1	63	145	233	150	2.3	False	True	False	False	False	True	False	True	False
1	2	67	160	286	108	1.5	False	True	True	False	False	False	True	False	False
2	3	67	120	229	129	2.6	False	True	True	False	False	False	True	False	False
3	4	37	130	250	187	3.5	False	True	False	True	False	False	True	False	True
4	5	41	130	204	172	1.4	True	False	False	False	True	False	True	False	False

RestECG_1	RestECG_2	ExAng_0	ExAng_1	Slope_1	Slope_2	Slope_3	Ca_0.0	Ca_1.0	Ca_2.0	Ca_3.0	Thal_fixed	Thal_normal	Thal_reversable
False	True	True	False	False	False	True	True	False	False	False	True	False	False
False	True	False	True	False	True	False	False	False	False	True	False	True	False
False	True	False	True	False	True	False	False	False	True	False	False	False	True
False	False	True	False	False	False	True	True	False	False	False	False	True	False
False	True	True	False	True	False	False	True	False	False	False	False	True	False

Standard Scaling

SNo	Age	RestBP	Chol	MaxHR	Oldpeak
1	0.9361806492869552	0.750380041408663	-0.27644339140393	0.01749443399309793	1.0689652869493862
2	1.3789285041248942	1.5962664494237417	0.744555065679425	-1.816333882224527	0.381773316767638
3	1.3789285041248942	-0.6594306386164683	-0.35349987873097566	-0.8994197241157145	1.3266622757675421
4	-1.9416804071596487	-0.09550636660641582	0.05104667973601406	1.6330098554229102	2.0997532422220093
5	-1.4989325523217096	-0.09550636660641582	-0.835102924525011	0.9780711710594728	0.29587432049491935

ML Analysis & Findings

IN THIS SECTION

- Comparison between 4 different classification models

- Namely Logistic Regression, KNN, SVM and XGBoost

- Techniques Used:
  Cross-Validation
  Scaling
  Grid Search
  Metric Measurements

Using Stratified Shuffle Split

Train Test Split

```
from sklearn.model_selection import StratifiedShuffleSplit
from sklearn.metrics import classification_report, confusion_matrix, ConfusionMatrixDisplay

features=heart_encoded.columns.to_list()
features.remove('AHD')
# print(features)
split_obj=StratifiedShuffleSplit(n_splits=1, test_size=0.3, random_state=27)
train_idx, test_idx = next(split_obj.split(heart_scaled[features],heart_scaled['AHD']))
# print(train_idx)
# print(test_idx)
X_train = heart_scaled.iloc[train_idx, :][features]
X_test = heart_scaled.iloc[test_idx, :][features]
y_train = heart_scaled.iloc[train_idx]['AHD']
y_test = heart_scaled.iloc[test_idx]['AHD']
```

[36] ✓ 0.0s Python

Logistic Regression Without Penalty

Logistic Regression Model

```
from sklearn.linear_model import LogisticRegression

LR=LogisticRegression(solver='liblinear')
LR=LR.fit(X_train, y_train)
y_predict=LR.predict(X_test)

clfreport=pd.DataFrame(classification_report(y_true=y_test, y_pred=y_predict, output_dict=True))
print(clfreport)
```

[37] ✓ 0.0s Python

	No	Yes	accuracy	macro avg	weighted avg
precision	0.863636	0.782609	0.822222	0.823123	0.825823
recall	0.791667	0.857143	0.822222	0.824405	0.822222
f1-score	0.826087	0.818182	0.822222	0.822134	0.822398
support	48.000000	42.000000	0.822222	90.000000	90.000000

Logistic Regression With L1 Penalty

```python
from sklearn.linear_model import LogisticRegressionCV

LR_L1=LogisticRegressionCV(Cs=10, cv=4,solver='liblinear', penalty='l1')
LR_L1=LR_L1.fit(X_train, y_train)
y_predict=LR_L1.predict(X_test)

clfreport=pd.DataFrame(classification_report(y_true=y_test, y_pred=y_predict, output_dict=True))
print(clfreport)
```

[11] ✓ 0.2s Python

```
                    No        Yes  accuracy  macro avg  weighted avg
precision     0.857143   0.750000       0.8   0.803571      0.807143
recall        0.750000   0.857143       0.8   0.803571      0.800000
f1-score      0.800000   0.800000       0.8   0.800000      0.800000
support      48.000000  42.000000       0.8  90.000000     90.000000
```

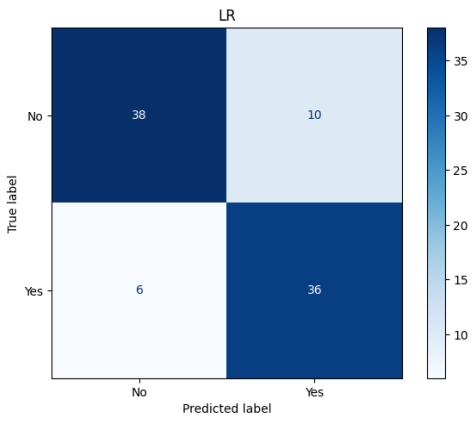
Logistic Regression With L2 Penalty

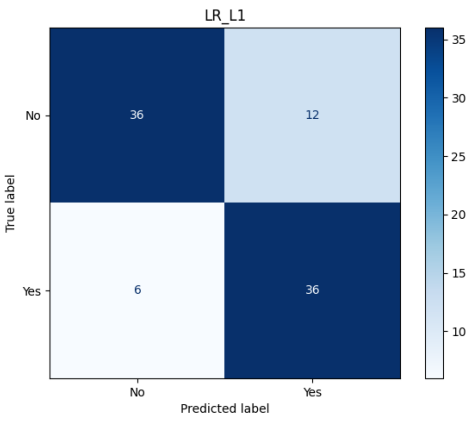
```
from sklearn.linear_model import LogisticRegressionCV

LR_L2=LogisticRegressionCV(Cs=10, cv=4,solver='liblinear', penalty='l2')
LR_L2=LR_L2.fit(X_train, y_train)
y_predict=LR_L2.predict(X_test)

clfreport=pd.DataFrame(classification_report(y_true=y_test, y_pred=y_predict, output_dict=True))
print(clfreport)
```

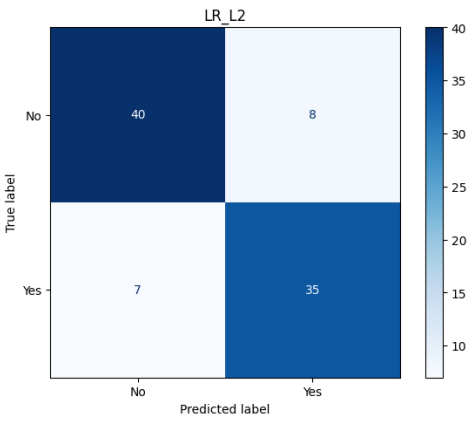
[12] ✓ 0.1s Python

```
                    No        Yes  accuracy  macro avg  weighted avg
precision     0.851064   0.813953  0.833333   0.832509      0.833746
recall        0.833333   0.833333  0.833333   0.833333      0.833333
f1-score      0.842105   0.823529  0.833333   0.832817      0.833437
support      48.000000  42.000000  0.833333  90.000000     90.000000
```

Confusion Matrices







Best performance is with penalty L2:

Accuracy = 83.33%

Precision = 83.37%

Recall = 83.33%

F1-Score = 83.34%

Support = 90.00%

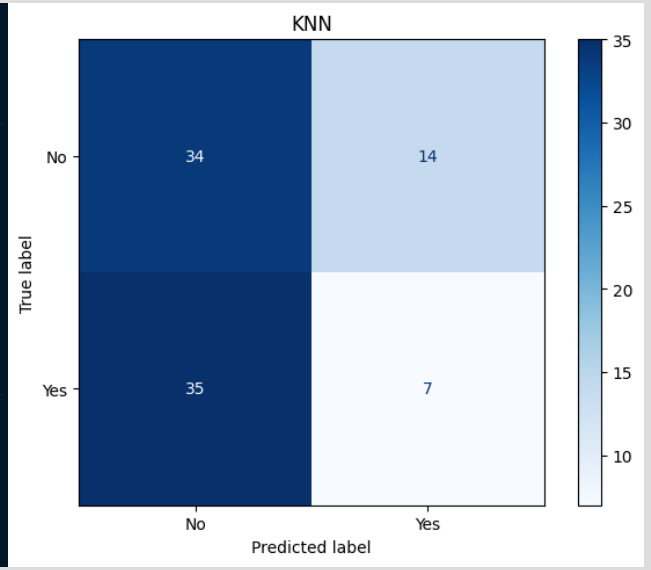
K Nearest Neighbours

KNN Model

```
from sklearn.neighbors import  KNeighborsClassifier

knn=KNeighborsClassifier(n_neighbors=25, weights='distance')
knn=knn.fit(X_train, y_train)
y_predict=knn.predict(X_test)

knn_report=pd.DataFrame(classification_report(y_true=y_test, y_pred=y_predict, output_dict=True))
print(knn_report)
```

[12] ✓ 0.0s Python

	No	Yes	accuracy	macro avg	weighted avg
precision	0.492754	0.333333	0.455556	0.413043	0.418357
recall	0.708333	0.166667	0.455556	0.437500	0.455556
f1-score	0.581197	0.222222	0.455556	0.401709	0.413675
support	48.000000	42.000000	0.455556	90.000000	90.000000



Accuracy = 45.56%
Precision = 41.83%
Recall = 45.56%
F1-Score = 41.37%
Support = 90.00%

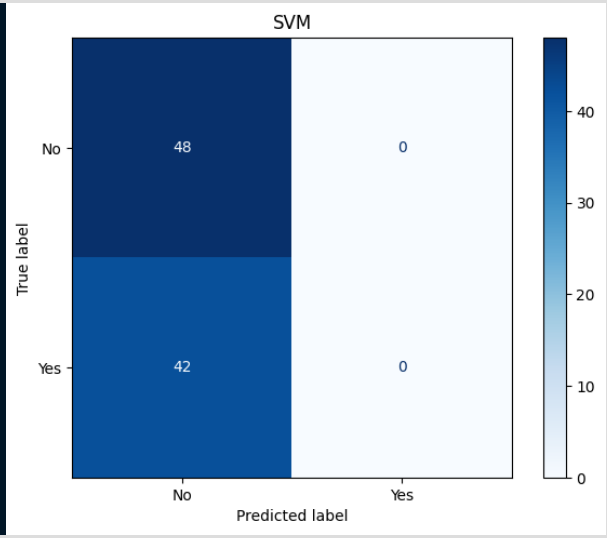
Support Vector Machines

SVM Model

```
from sklearn.svm import  SVC

kwargs={'kernel':'rbf'}
svc=SVC(**kwargs)
svc=svc.fit(X_train, y_train)
y_predict=svc.predict(X_test)

svc_report=pd.DataFrame(classification_report(y_true=y_test, y_pred=y_predict, output_dict=True))
print(svc_report)
```

[16] ✓ 0.0s Python

	No	Yes	accuracy	macro avg	weighted avg
precision	0.533333	0.0	0.533333	0.266667	0.284444
recall	1.000000	0.0	0.533333	0.500000	0.533333
f1-score	0.695652	0.0	0.533333	0.347826	0.371014
support	48.000000	42.0	0.533333	90.000000	90.000000



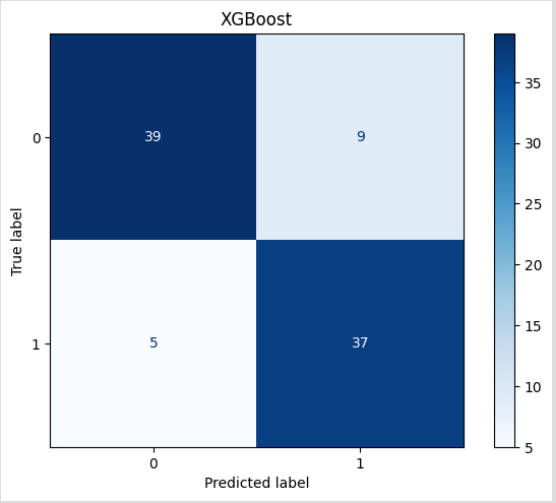
Accuracy = 53.33%
Precision = 28.44%
Recall = 53.33%
F1-Score = 37.10%
Support = 90.00%

XGBoost

```python
import xgboost as xgb
from sklearn.model_selection import GridSearchCV

heart_bool=heart_scaled.copy()
heart_bool['AHD']=heart_bool['AHD'].apply(lambda x:0 if x=='No' else 1)
X_train = heart_bool.iloc[train_idx, :][features]
X_test = heart_bool.iloc[test_idx, :][features]
y_train = heart_bool.iloc[train_idx]['AHD']
y_test = heart_bool.iloc[test_idx]['AHD']

param_grid={
    'max_depth':[5],
    'learning_rate':[0.05],
    'gamma':[0,0.25,1,10],
    'reg_lambda':[0],
    'scale_pos_weight':[1,3,5,7,10],
    'subsample':[0.1,0.2,0.3,0.4,0.5,0.8],
    'colsample_bytree':[0.5,0.7],
}

xgb_cl=xgb.XGBClassifier(object='binary:logistic')
grid_cv=GridSearchCV(xgb_cl, param_grid, n_jobs=-1, cv=3, scoring='roc_auc')
grid_cv=grid_cv.fit(X_train,y_train)
y_predict=grid_cv.predict(X_test)

report=pd.DataFrame(classification_report(y_true=y_test, y_pred=y_predict, output_dict=True))
print(report)
```

[21] ✓ 6.5s Python

```
                   0          1  accuracy  macro avg  weighted avg
precision   0.886364   0.804348  0.844444   0.845356      0.848090
recall      0.812500   0.880952  0.844444   0.846726      0.844444
f1-score    0.847826   0.840909  0.844444   0.844368      0.844598
support    48.000000  42.000000  0.844444  90.000000     90.000000
```



Accuracy = 84.44%
Precision = 84.81%
Recall = 84.44%
F1-Score = 84.46%
Support = 90.00%

Models Comparison

Though all models provide good support only Logistic Regression model and XGBoost Model provide precision, recall and f1-score.
Models can be ranked as follows:

1. XGBoost
2. Logistic Regression with L2 penalty
3. KNN
4. SVM

	0	1	accuracy	macro avg	weighted avg
precision	0.886364	0.804348	0.844444	0.845356	0.848090
recall	0.812500	0.880952	0.844444	0.846726	0.844444
f1-score	0.847826	0.840909	0.844444	0.844368	0.844598
support	48.000000	42.000000	0.844444	90.000000	90.000000

Model Flaws, Strengths & Improvements

Model Strengths, Flaws and Advanced steps:

In terms of simplicity, Logistic regression model provided viable and acceptable results while being the fastest and easiest in terms of parameters and training. KNN and SVM while being slower (in terms of prediction) still fail to produce results better than the Logistic regression model. KNN though easy to train takes time to predict since measuring distances and classifying datapoints is a tedious process.

On the other hand we have XGBoost model producing best results, however it takes longer to train since we use grid search technique to find the best fitting parameters. Since the dataset in the example taken in small we opt for XGBoost model, but with increase in the size of dataset the time taken by XGBoost increases and after a certain threshold its better to choose other models.

THANK YOU