Internship Report
On

(Machine Learning)

Submitted by
[Shubham Mishra]
[CSJMA21001390163]
[Chhatrapati Shahu Ji Maharaj University Kanpur]

Submitted to
Mallika Srivastava
Head, Training
EISystems Services

&

Mayur Dev Sewak
Head, Internships & Trainings
EISystems Services

Self`s Declaration

I, [ShubhamMishra],a student of [B.Tech] program, Roll No. [CSJMA21001390163] of the Department of [CSE(AI)] College do hereby declare that I have completed the mandatory internship in Eisystems Technologies under the faculty guideship of [Dr Alok Kumar], Department of [CSE]

, [Chhatrapati Shahu Ji Maharaj University Kanpur]

Table of Content

Executive Summary

Week 1: Introduction to Machine Learning

The first week was dedicated to understanding the basics of machine learning. This included an overview of what machine learning is, its applications, types of machine learning (supervised, unsupervised, and reinforcement learning), and key concepts such as features, labels, and datasets. We also learned about the typical workflow of a machine learning project, from data collection and preprocessing to model training and evaluation.

Week 2: Looping and Control Structures

In the second week, we focused on foundational programming skills essential for machine learning. We learned about looping constructs (for and while loops) and control structures (if-else statements) in Python. These constructs are crucial for data manipulation and iterative processes in machine learning algorithms. Practical exercises included writing scripts to process data and implement basic algorithms.

Week 3: NumPy, Pandas, and Matplotlib

The third week was dedicated to data manipulation and visualization, key skills for any machine learning practitioner. We learned:

- **NumPy**: For numerical operations and array manipulations.
- **Pandas**: For data analysis and manipulation, working with DataFrames.
- **Matplotlib**: For data visualization, creating plots, and visualizing the results of data analysis.

Hands-on sessions involved using these libraries to load, clean, and visualize datasets, providing a strong foundation for data preprocessing and exploratory data analysis.

Week 4: Building a Project with Scikit-learn

The final week culminated in applying the skills learned to build a machine learning project using Scikit-learn. This involved:

- **Data Preprocessing**: Using techniques learned in previous weeks to clean and prepare the data.
- **Model Training**: Implementing and training various machine learning models.
- **Model Evaluation**: Using metrics to evaluate model performance and selecting the best model.
- **Deployment**: Understanding the basics of deploying a machine learning model for practical use.

The project chosen for this week was the Heart Disease Detection project, where we developed a predictive model to identify the presence of heart disease in patients based on health parameters.

Overall, this internship provided a thorough grounding in machine learning, equipping participants with the knowledge and practical skills needed to undertake machine learning projects. The structured approach ensured a progressive learning curve, building from basic concepts to practical implementation.

Overview of Organization

About EISystems

India's leader in workshops & trainings at IITs, NITs & top engineering colleges

EISystems Services is a leading Indian technology identity with operations across India. EISystems (We call it EISys) offers trainings in Cybersecurity, Machine Learning, Automobiles, Internet of Things, Robotics and Socialmedia for enterprises and student community. Till date we have trained approximately 50000 students and impacted around 2 lakhs students through our various outreach initiatives since our founding.

Our Presence

Some of the colleges where we had already felt our presence are given below:-

Indian Institute of Science, Bangalore
Indian Institute of Technology, Bombay
Indian Institute of Technology, Delhi
Indian Institute of Technology, Madras
Indian Institute of Technology, Kanpur
Indian Institute of Technology, Roorkee
Indian Institute of Technology, Guwahati
Indian Institute of Technology (Banaras Hindu University), Varanasi
Indian Institute of Technology, Indore
Indian Institute of Technology, Jodhpur
Indian Institute of Technology, Hyderabad
National Institute of Technology, Tiruchirappalli
National Institute of Technology, Warangal
National Institute of Technology, Calicut
National Institute of Technology, Patna
National Institute of Technology, Jalandhar
National Institute of Technology, Jaipur
National Institute of Technology, Durgapur
National Institute of Technology, Surat
National Institute of Technology, Allahabad
Indian Institute of Information Technology, Allahabad
ABV Indian Institute of Information Technology, Gwalior
PDP Indian Institute of Information Technology, Jabalpur
Jawahar Lal Nehru Technological University, Hyderabad
College of Engineering, Guindy
Delhi Technological University, New Delhi
& around 100 engineering colleges.

Project Summary

The Heart Disease Detection project aims to develop a predictive model to identify the presence of heart disease in patients based on various health parameters. Using machine learning techniques, the project seeks to provide an efficient, accurate, and non-invasive method for early detection, which is crucial for timely treatment and better patient outcomes.

Details of Projects

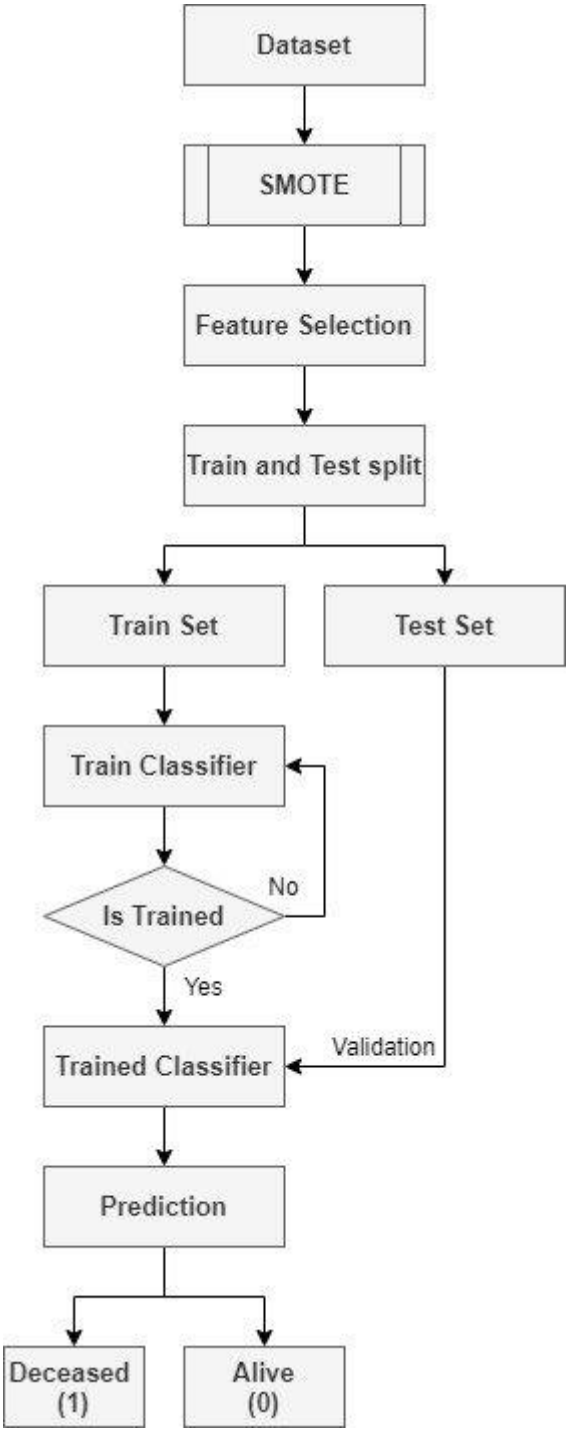
Problem Definition

data. Heart disease is one of the leading causes of death globally. Early detection can significantly improve the chances of successful treatment and management. The goal of this project is to create a model that can predict the likelihood of heart disease in a patient using historical health

Objectives

- To collect and preprocess relevant health data.
- To build and train a machine learning model for heart disease prediction.
- To evaluate the model's performance and fine-tune it for optimal accuracy.
- To create a user-friendly interface for practical application.

Data Flow Diagram / Algorithms


Dataset
SMOTE
Feature Selection
Train and Test split
Train Set
Test Set
Train Classifier
Is Trained
No
Yes
Trained Classifier
Validation
Prediction
Deceased
(1)
Alive
(0)

Algorithms

1. **Data Collection and Preprocessing**:
   - Collect data from reliable medical sources.
   - Clean the data by handling missing values and outliers.
   - Normalize and scale the data for consistency.
2. **Feature Selection**:
   - Use statistical methods and domain knowledge to select the most relevant features for prediction.
3. **Model Training**:
   - Split the data into training and testing sets.
   - Train different machine learning models (e.g., Logistic Regression, Decision Trees, Random Forest, etc.).
   - Use cross-validation techniques to evaluate model performance.
4. **Model Evaluation**:
   - Evaluate the models using metrics such as accuracy, precision, recall, F1-score, and ROC-AUC.
   - Select the best-performing model based on evaluation metrics.
5. **Prediction**:
   - Deploy the selected model to predict heart disease on new patient data.

Input / Output Datasets / Screenshots

.

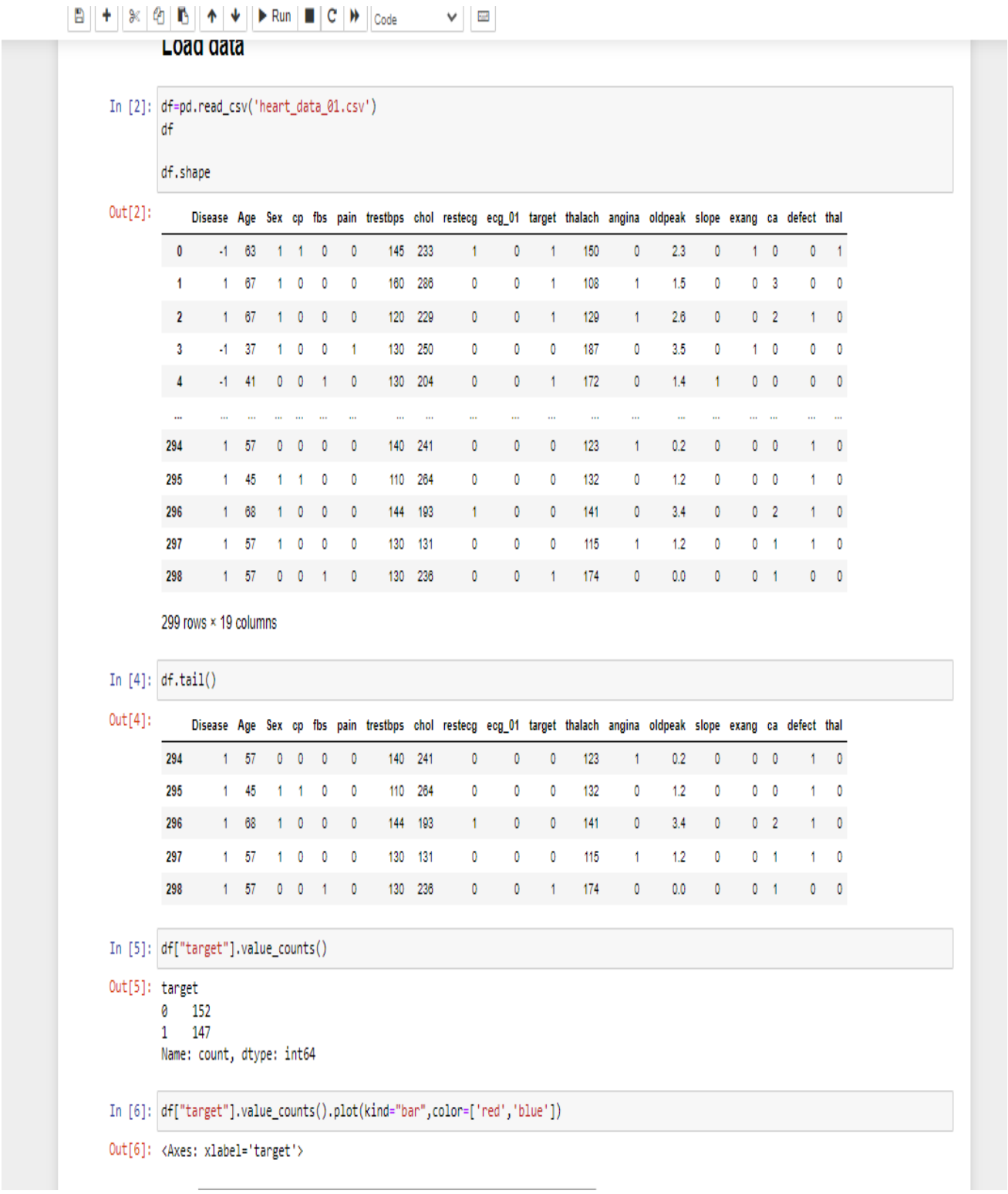
Input Datasets

The input dataset consists of patient records with various health parameters such as age, sex, chest pain type, resting blood pressure, serum cholesterol, fasting blood sugar, resting electrocardiographic results, maximum heart rate achieved, exercise-induced angina, oldpeak (ST depression induced by exercise relative to rest), the slope of the peak exercise ST segment, number of major vessels colored by fluoroscopy, and thalassemia.

Output Datasets

The output is a prediction of the presence or absence of heart disease for each patient record in the dataset.

Screenshots



Load data

In [2]:
```
df=pd.read_csv('heart_data_01.csv')
df

df.shape
```

Out[2]:

	Disease	Age	Sex	cp	fbs	pain	trestbps	chol	restecg	ecg_01	target	thalach	angina	oldpeak	slope	exang	ca	defect	thal
0	-1	63	1	1	0	0	145	233	1	0	1	150	0	2.3	0	1	0	0	1
1	1	67	1	0	0	0	160	286	0	0	1	108	1	1.5	0	0	3	0	0
2	1	67	1	0	0	0	120	229	0	0	1	129	1	2.6	0	0	2	1	0
3	-1	37	1	0	0	1	130	250	0	0	0	187	0	3.5	0	1	0	0	0
4	-1	41	0	0	1	0	130	204	0	0	1	172	0	1.4	1	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
294	1	57	0	0	0	0	140	241	0	0	0	123	1	0.2	0	0	0	1	0
295	1	45	1	1	0	0	110	264	0	0	0	132	0	1.2	0	0	0	1	0
296	1	68	1	0	0	0	144	193	1	0	0	141	0	3.4	0	0	2	1	0
297	1	57	1	0	0	0	130	131	0	0	0	115	1	1.2	0	0	1	1	0
298	1	57	0	0	1	0	130	236	0	0	1	174	0	0.0	0	0	1	0	0

299 rows × 19 columns

In [4]:
```
df.tail()
```

Out[4]:

	Disease	Age	Sex	cp	fbs	pain	trestbps	chol	restecg	ecg_01	target	thalach	angina	oldpeak	slope	exang	ca	defect	thal
294	1	57	0	0	0	0	140	241	0	0	0	123	1	0.2	0	0	0	1	0
295	1	45	1	1	0	0	110	264	0	0	0	132	0	1.2	0	0	0	1	0
296	1	68	1	0	0	0	144	193	1	0	0	141	0	3.4	0	0	2	1	0
297	1	57	1	0	0	0	130	131	0	0	0	115	1	1.2	0	0	1	1	0
298	1	57	0	0	1	0	130	236	0	0	1	174	0	0.0	0	0	1	0	0

In [5]:
```
df["target"].value_counts()
```

Out[5]:
```
target
0    152
1    147
Name: count, dtype: int64
```

In [6]:
```
df["target"].value_counts().plot(kind="bar",color=['red','blue'])
```

Out[6]:
```
<Axes: xlabel='target'>
```

Modeling

In [21]:
```
df.head()
```

Out[21]:

	Disease	Age	Sex	cp	fbs	pain	trestbps	chol	restecg	ecg_01	target	thalach	angina	oldpeak	slope	exang	ca	defect	thal
0	-1	63	1	1	0	0	145	233	1	0	1	150	0	2.3	0	1	0	0	1
1	1	67	1	0	0	0	160	286	0	0	1	108	1	1.5	0	0	3	0	0
2	1	67	1	0	0	0	120	229	0	0	1	129	1	2.6	0	0	2	1	0
3	-1	37	1	0	0	1	130	250	0	0	0	187	0	3.5	0	1	0	0	0
4	-1	41	0	0	1	0	130	204	0	0	1	172	0	1.4	1	0	0	0	0

In [22]:
```
x=df.drop("target",axis=1)

y=df["target"]
```

In [23]:
```
y
```

Out[23]:
```
0      1
1      1
2      1
3      0
4      1
      ..
294    0
295    0
296    0
297    0
298    1
Name: target, Length: 299, dtype: int64
```

In [35]:
```
log_grid={"C":np.logspace(-4,4,20),
          "solver":["liblinear"]}
```

In [36]:
```
np.random.seed(42)
log_reg=RandomizedSearchCV(LogisticRegression(),
                           param_distributions=log_grid,
                           cv=5,
                           n_iter=20,
                           verbose=True)
# fit the model
log_reg.fit(x_train,y_train)
```

```
Fitting 5 folds for each of 20 candidates, totalling 100 fits
```

Out[36]:
```
RandomizedSearchCV(cv=5, estimator=LogisticRegression(), n_iter=20,
                   param_distributions={'C': array([1.00000000e-04, 2.63665090e-04, 6.95192796e-04, 1.83298071e-03,
       4.83293024e-03, 1.27427499e-02, 3.35981829e-02, 8.85866790e-02,
       2.33572147e-01, 6.15848211e-01, 1.62377674e+00, 4.28133240e+00,
       1.12883789e+01, 2.97635144e+01, 7.84759970e+01, 2.06913808e+02,
       5.45559478e+02, 1.43844989e+03, 3.79269019e+03, 1.00000000e+04]),
                                        'solver': ['liblinear']},
                   verbose=True)
```

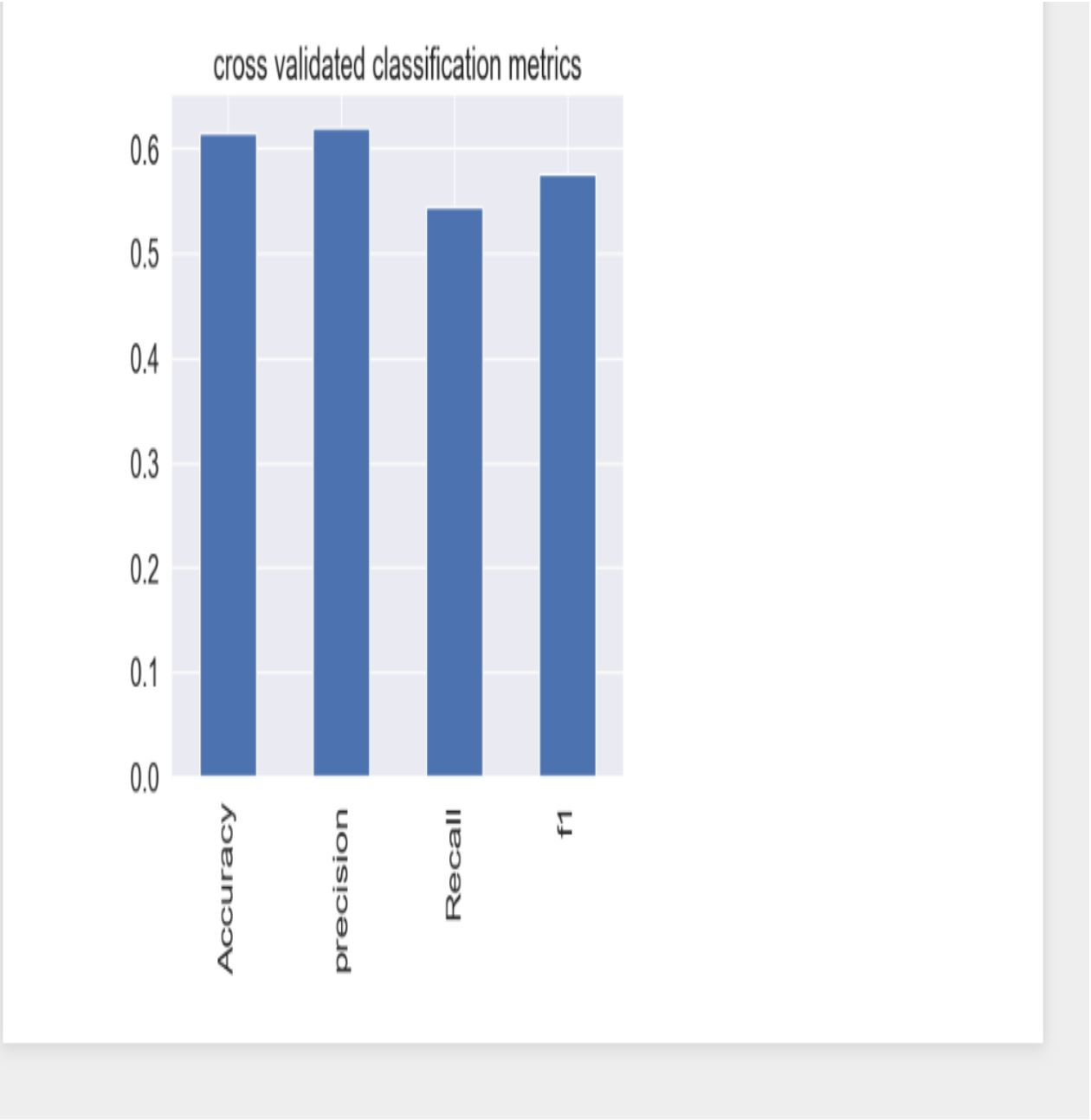
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [37]:
```
log_reg.best_params_
```

Out[37]:
```
{'solver': 'liblinear', 'C': 0.0001}
```

In [38]:
```
log_reg.score(x_test,y_test)
```

Out[38]:
```
0.5833333333333334
```


cross validated classification metrics
0.6
0.5
0.4
0.3
0.2
0.1
0.0
Accuracy
precision
Recall
f1

Text / Code / Program Used

HEART DISEASE PREDICTION MODEL

```
import numpy as np
import pandas as pd
# % matplotlib inline
import matplotlib.pyplot as plt
# import seaborn as sns

# model implimplement

from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
#model evaluation

from sklearn.model_selection  import train_test_split,cross_val_score
from sklearn.model_selection  import RandomizedSearchCV,GridSearchCV
from sklearn.metrics import confusion_matrix,classification_report
from sklearn.metrics import precision_score,recall_score,f1_score
from sklearn.metrics import RocCurveDisplay
```

Load data

```
df=pd.read_csv('heart_data_01.csv')
df

df.shape
```

```
     Disease  Age  Sex  cp  fbs  pain  trestbps  chol  restecg  ecg_01
\
0         -1   63    1   1    0     0       145   233        1       0
1          1   67    1   0    0     0       160   286        0       0
2          1   67    1   0    0     0       120   229        0       0
3         -1   37    1   0    0     1       130   250        0       0
4         -1   41    0   0    1     0       130   204        0       0
..       ...  ...  ...  ..  ...   ...       ...   ...      ...     ...
294        1   57    0   0    0     0       140   241        0       0
295        1   45    1   1    0     0       110   264        0       0
296        1   68    1   0    0     0       144   193        1       0
```

```
297        1   57    1   0    0     0       130   131        0       0
298        1   57    0   0    1     0       130   236        0       0

     target  thalach  angina  oldpeak  slope  exang  ca  defect  thal
0         1      150       0      2.3      0      1   0       0     1
1         1      108       1      1.5      0      0   3       0     0
2         1      129       1      2.6      0      0   2       1     0
3         0      187       0      3.5      0      1   0       0     0
4         1      172       0      1.4      1      0   0       0     0
..      ...      ...     ...      ...    ...    ...  ..     ...   ...
294       0      123       1      0.2      0      0   0       1     0
```

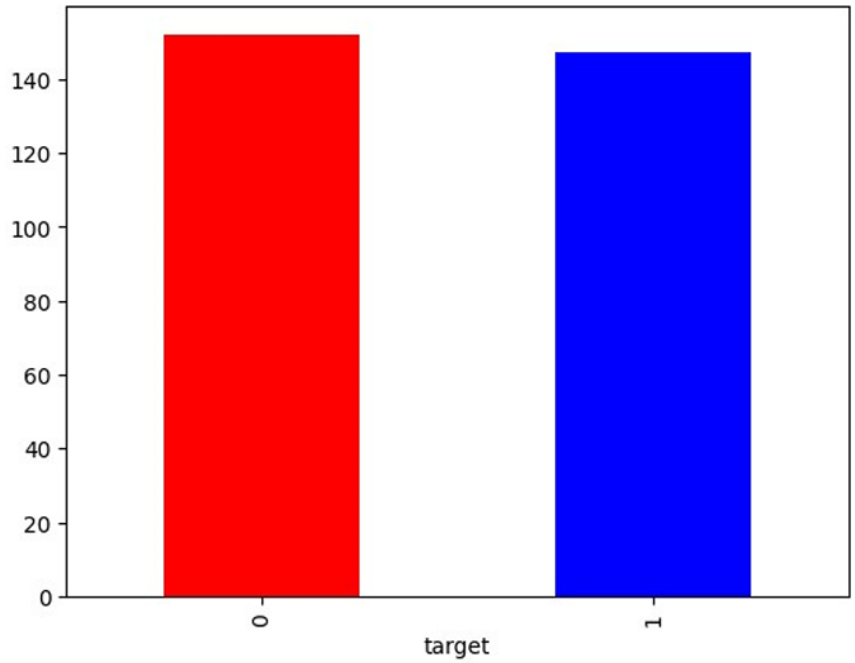
```
296       0      141       0      3.4      0      0   2       1     0

297       0      115       1      1.2      0      0   1       1     0

298       1      174       0      0.0      0      0   1       0     0
```

```
df["target"].value_counts()
```

```
target
0    152
1    147
Name: count, dtype: int64
```

```
df["target"].value_counts().plot(kind="bar",color=['red','blue'])
```

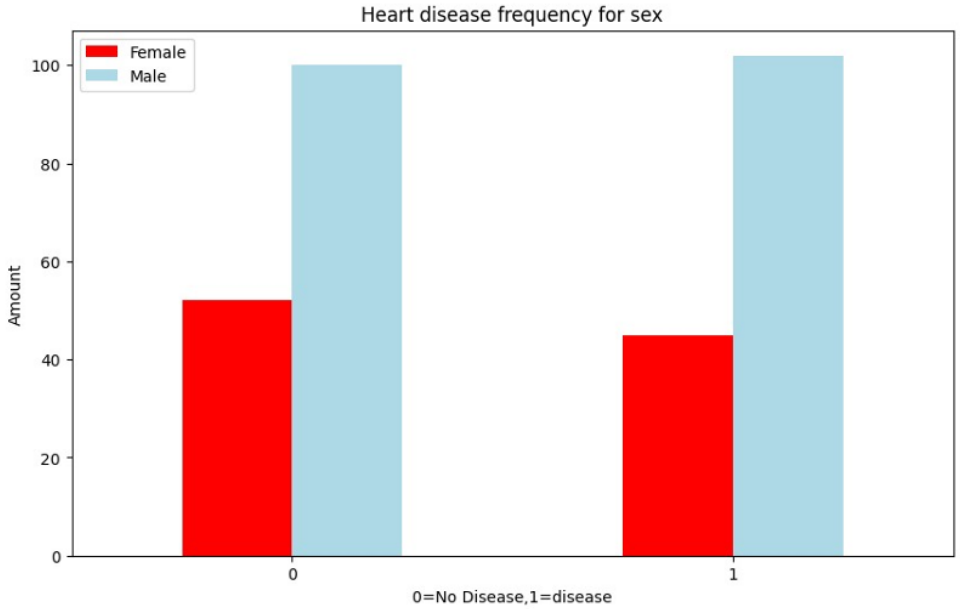
```
<Axes: xlabel='target'>
```




```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 299 entries, 0 to 298
Data columns (total 19 columns):
 #   Column    Non-Null Count  Dtype
```

```
---  ------    --------------  -----
 0   Disease   299 non-null    int64
 1   Age       299 non-null    int64
 2   Sex       299 non-null    int64
 3   cp        299 non-null    int64
 4   fbs       299 non-null    int64
 5   pain      299 non-null    int64
 6   trestbps  299 non-null    int64
 7   chol      299 non-null    int64
 8   restecg   299 non-null    int64
```




```
df["thalach"].value_counts()
```

```
thalach
162    11
160     9
163     9
152     8
150     7
       ..
177     1
127     1
97      1
190     1
90      1
Name: count, Length: 91, dtype: int64
```

Age vs Max Heart rate

```
plt.figure(figsize=(10,6))
plt.scatter(df.Age[df.target==1],
            df.thalach[df.target==1],
            color='blue')
#negative result
plt.scatter(df.Age[df.target==0],
            df.thalach[df.target==0],
```

336

337

338

339

340

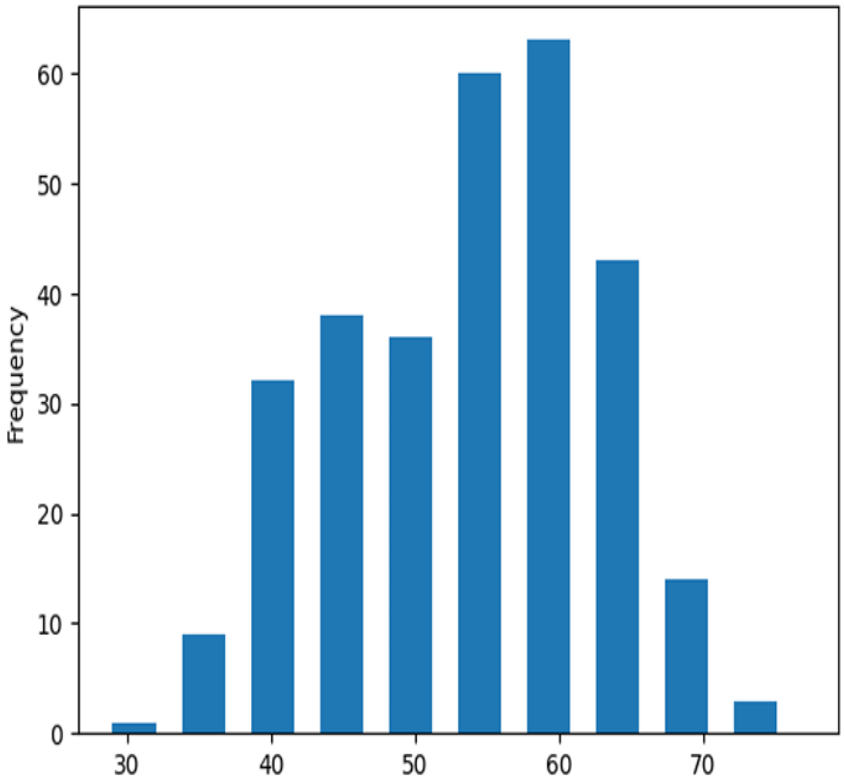
```
            color='lightblue')
plt.title("Heart Disease in function of Age and Max Heart Rate")
plt.xlabel("Age")
plt.ylabel("Max Heart Rate")
plt.legend(["Disease","No disease"]);
```


```
# check the distributon of age
df.Age.plot.hist(width=3);
```

341

342

343




```
# compair chest pain vs target
pd.crosstab(df.cp,df.target)
```

```
target    0    1
cp
0       143  133
1         9   14
```

```
pd.crosstab(df.cp,df.target).plot(kind='bar',
                                  figsize=(10,6),
                                  color=["green","red"])
plt.title("Heart Disease frequency per chest pain type")
plt.xlabel("chest pain type")
plt.ylabel("Amount")
plt.legend([" No Disease","disease"]);
plt.xticks(rotation=0)
```

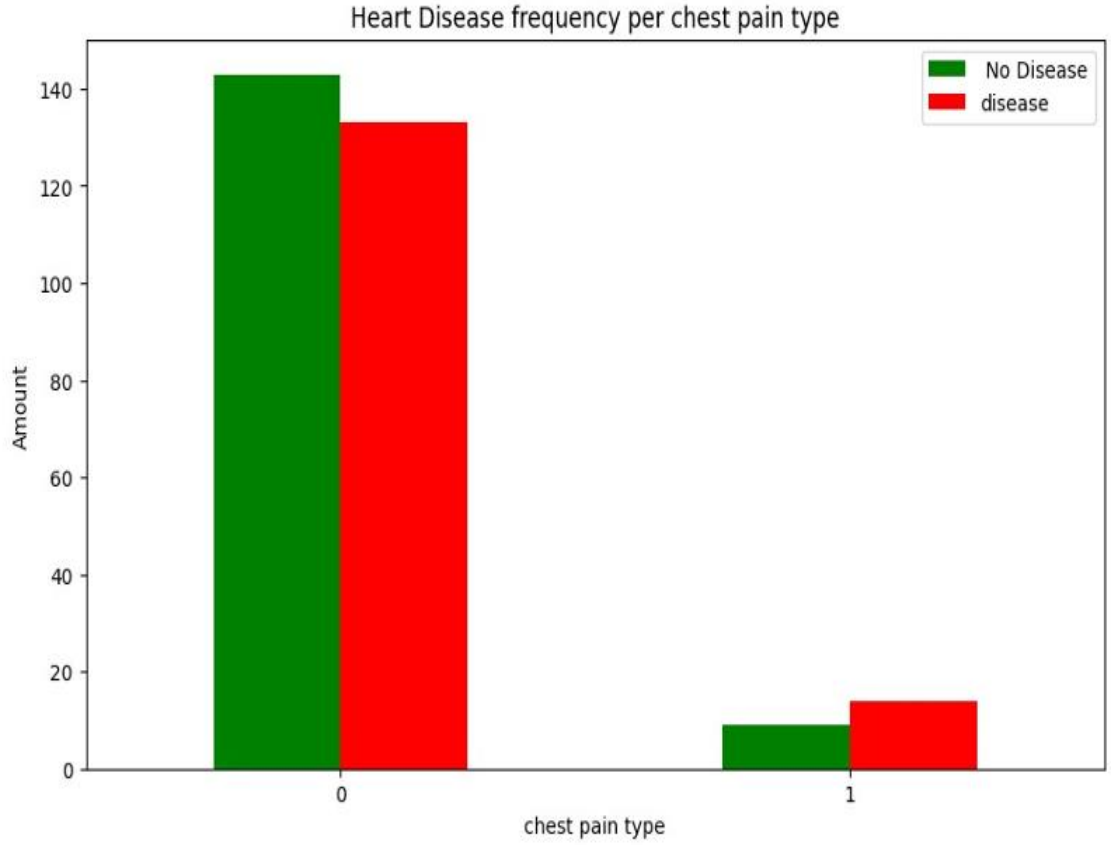
```
(array([0, 1]), [Text(0, 0, '0'), Text(1, 0, '1')])
```

344

345

346

347




```
df.head()
```

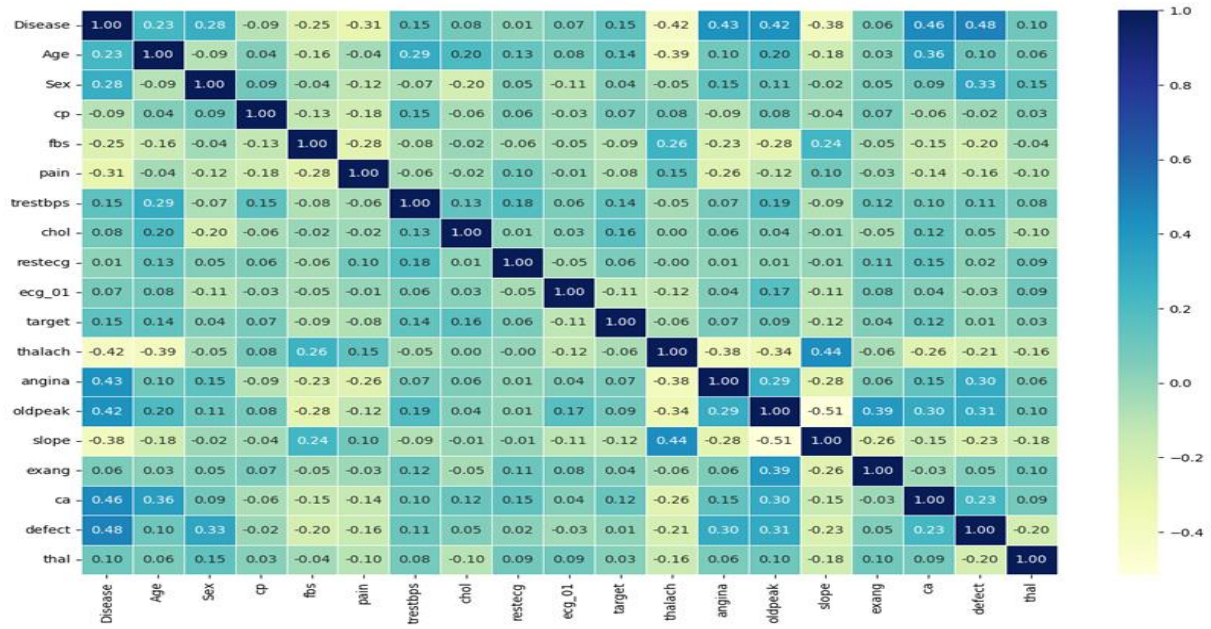
	Disease	Age	Sex	cp	fbs	pain	trestbps	chol	restecg	ecg_01 target
0	-1	63	1	1	0	0	145	233	1	0 1
1	1	67	1	0	0	0	160	286	0	0 1
2	1	67	1	0	0	0	120	229	0	0 1
3	-1	37	1	0	0	1	130	250	0	0 0
4	-1	41	0	0	1	0	130	204	0	0 1

	thalach	angina	oldpeak	slope	exang	ca	defect	thal
0	150	0	2.3	0	1	0	0	1
1	108	1	1.5	0	0	3	0	0
2	129	1	2.6	0	0	2	1	0
3	187	0	3.5	0	1	0	0	0
4	172	0	1.4	1	0	0	0	0

```
# correlation matrics
df.corr()
```

```
             Disease       Age       Sex        cp       fbs
pain  \
Disease    1.000000  0.225775  0.283300 -0.091024 -0.246763 -0.310013
Age        0.225775  1.000000 -0.091813  0.042989 -0.162418 -0.043286
Sex        0.283300 -0.091813  1.000000  0.092803 -0.040600 -0.123170
cp        -0.091024  0.042989  0.092803  1.000000 -0.127802 -0.180439
fbs       -0.246763 -0.162418 -0.040600 -0.127802  1.000000 -0.276725
pain      -0.310013 -0.043286 -0.123170 -0.180439 -0.276725  1.000000
trestbps   0.152840  0.290696 -0.065521  0.150260 -0.080136 -0.055227
chol       0.078722  0.203377 -0.195907 -0.055531 -0.015501 -0.024900
restecg    0.013111  0.128676  0.045862  0.057231 -0.056382  0.097436
ecg_01     0.067379  0.083677 -0.105856 -0.033615 -0.051552 -0.008015
target     0.149680  0.139149  0.038425  0.067592 -0.091995 -0.078853
thalach   -0.415031 -0.392342 -0.052064  0.081268  0.256764  0.150852
angina     0.425476  0.095108  0.149038 -0.094614 -0.232138 -0.262072
oldpeak    0.424672  0.197376  0.110237  0.084343 -0.281040 -0.121395
slope     -0.384730 -0.183068 -0.022648 -0.044501  0.236417  0.099450
exang      0.060585  0.026018  0.050676  0.068006 -0.050966 -0.026198
ca         0.460442  0.362605  0.093185 -0.059834 -0.153886 -0.138891
defect     0.481585  0.104754  0.327572 -0.021830 -0.201429 -0.157658
thal       0.104142  0.060092  0.145351  0.032472 -0.036080 -0.095631

             trestbps      chol   restecg    ecg_01    target
thalach  \
Disease    0.152840  0.078722  0.013111  0.067379  0.149680 -0.415031
Age        0.290696  0.203377  0.128676  0.083677  0.139149 -0.392342
Sex       -0.065521 -0.195907  0.045862 -0.105856  0.038425 -0.052064
cp         0.150260 -0.055531  0.057231 -0.033615  0.067592  0.081268
fbs       -0.080136 -0.015501 -0.056382 -0.051552 -0.091995  0.256764
```

```
pain      -0.055227 -0.024900  0.097436 -0.008015 -0.078853  0.150852
trestbps   1.000000  0.132284  0.177623  0.058177  0.141046 -0.048053
chol       0.132284  1.000000  0.006664  0.032914  0.160090  0.002179
restecg    0.177623  0.006664  1.000000 -0.048370  0.063599 -0.003387
ecg_01     0.058177  0.032914 -0.048370  1.000000 -0.114513 -0.120705
target     0.141046  0.160090  0.063599 -0.114513  1.000000 -0.063417
thalach   -0.048053  0.002179 -0.003387 -0.120705 -0.063417  1.000000
angina     0.065885  0.056388  0.011637  0.042728  0.068687 -0.376359
```



Modeling

```
df.head()
```

```
   Disease  Age  Sex  cp  fbs  pain  trestbps  chol  restecg  ecg_01
target  \
0       -1   63    1   1    0     0       145   233        1       0
1
1        1   67    1   0    0     0       160   286        0       0
1
2        1   67    1   0    0     0       120   229        0       0
1
3       -1   37    1   0    0     1       130   250        0       0
0
4       -1   41    0   0    1     0       130   204        0       0
1

   thalach  angina  oldpeak  slope  exang  ca  defect  thal
0      150       0      2.3      0      1   0       0     1
1      108       1      1.5      0      0   3       0     0
2      129       1      2.6      0      0   2       1     0
```

```
3      187       0      3.5      0      1   0       0     0
4      172       0      1.4      1      0   0       0     0
```

```
x=df.drop("target",axis=1)
```

```
y=df["target"]
```

```
y
```

```
0      1
1      1
2      1
3      0
4      1
      ..
294    0
295    0
```

```
2         129        1      2.6       0       0   2        1     0
3         187        0      3.5       0       1   0        0     0
4         172        0      1.4       1       0   0        0     0
..        ...      ...      ...     ...     ...  ..      ...   ...
294       123        1      0.2       0       0   0        1     0
295       132        0      1.2       0       0   0        1     0
296       141        0      3.4       0       0   2        1     0
297       115        1      1.2       0       0   1        1     0
298       174        0      0.0       0       0   1        0     0

[299 rows x 18 columns]
```

```
#split data
np.random.seed(42)
x_train,x_test,y_train,y_test=train_test_split(x,
                                               y,test_size=0.2)

x_train
```

```
     Disease  Age  Sex  cp  fbs  pain  trestbps  chol  restecg  ecg_01
\
6          1   62    0   0    0     0       140   268        0       0

183        1   60    0   0    0     0       158   305        0       0

185       -1   42    1   0    0     1       120   240        1       0

146        1   57    1   0    0     0       165   289        1       0

30        -1   69    0   1    0     0       140   239        0       0

..       ...  ...  ...  ..  ...   ...       ...   ...      ...     ...

188        1   69    1   0    0     1       140   254        0       0

71         1   67    1   0    0     0       125   254        1       0

106        1   59    1   0    0     0       140   177        0       0

270        1   46    1   0    0     0       140   311        0       0

102       -1   57    0   0    0     0       128   303        0       0


     thalach  angina  oldpeak  slope  exang  ca  defect  thal
6        160       0      3.6      0      1   2       0     0
183      161       0      0.0      1      0   0       0     0
185      194       0      0.8      0      1   0       1     0
146      124       0      1.0      0      0   3       1     0
30       151       0      1.8      1      0   2       0     0
..       ...     ...      ...    ...    ...  ..     ...   ...
```

```
188       146        0      2.0       0       0   3        1     0
71        163        0      0.2       0       0   2        1     0
106       162        1      0.0       1       0   1        1     0
270       120        1      1.8       0       0   2        1     0
102       159        0      0.0       1       0   1        0     0

[239 rows x 18 columns]
```

```
y_train
```

```
6      1
183    1
185    0
146    1
30     0
      ..
188    1
71     0
```

352

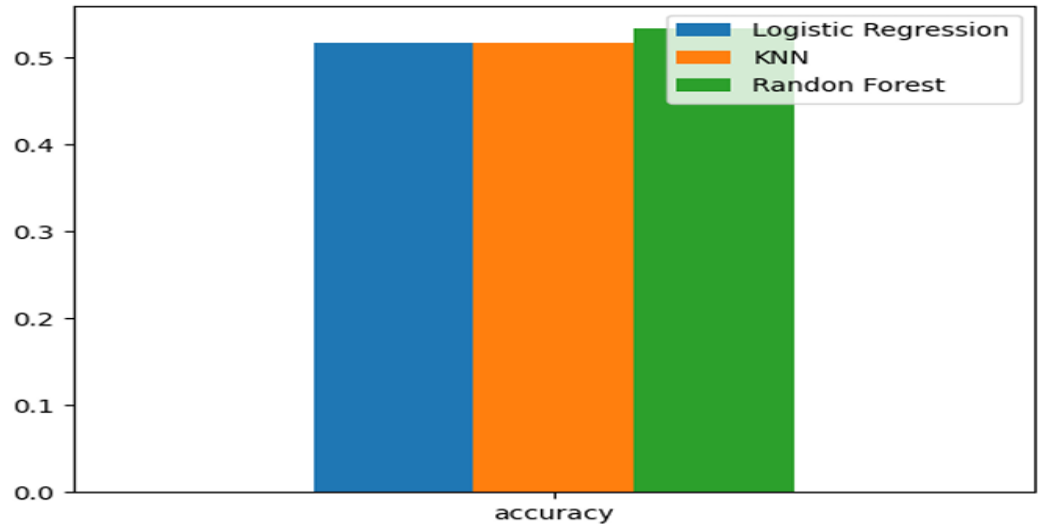
```
                                y_test=y_test)
model_score
```

```
C:\Users\dell\miniconda3\lib\site-packages\sklearn\linear_model\
_logistic.py:458: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as
shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

```
{'Logistic Regression': 0.5166666666666667,
 'KNN': 0.5166666666666667,
 'Randon Forest': 0.5333333333333333}
```

```
model_compare=pd.DataFrame(model_score,index=["accuracy"])
model_compare.plot.bar()
plt.xticks(rotation=0)
```

```
(array([0]), [Text(0, 0, 'accuracy')])
```



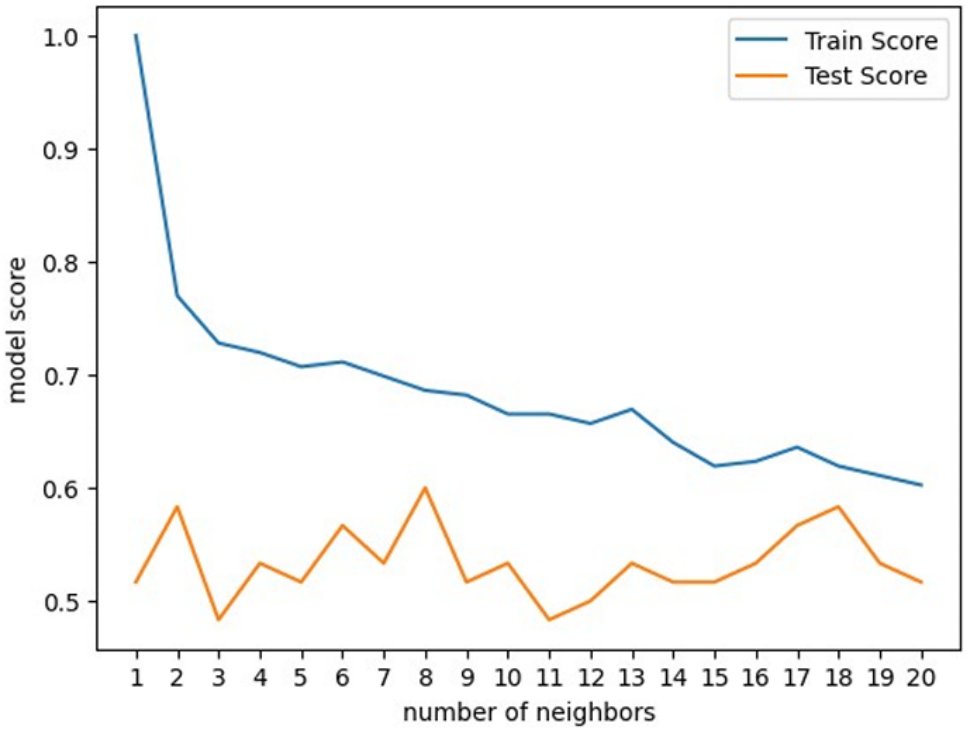

```
# hyperparameter tuning
train_score=[]
test_score=[]
neighbors=range(1,21)
knn=KNeighborsClassifier()
for i in neighbors:
    knn.set_params(n_neighbors=i)
    # fit algrithm
    knn.fit(x_train,y_train)
    train_score.append(knn.score(x_train,y_train))
    test_score.append(knn.score(x_test,y_test))


train_score
```

```
[1.0,
 0.7698744769874477,
```

```
 0.5333333333333333,
 0.5166666666666667,
 0.5166666666666667,
 0.5333333333333333,
 0.5666666666666667,
 0.5833333333333334,
 0.5333333333333333,
 0.5166666666666667]
```

```
plt.plot(neighbors,train_score,label="Train Score")
plt.plot(neighbors,test_score,label="Test Score")
plt.xlabel("number of neighbors")
plt.xticks(np.arange(1,21,1))
plt.ylabel("model score")
plt.legend()
print(f"Maximun knn score:{max(test_score)*100:0.2f}")
```

```
Maximun knn score:60.00
```




```
log_grid={"C":np.logspace(-4,4,20),
          "solver":["liblinear"]}
```

```
np.random.seed(42)
log_reg=RandomizedSearchCV(LogisticRegression(),
                           param_distributions=log_grid,
                           cv=5,
                           n_iter=20,
                           verbose=True)
# fit the model
log_reg.fit(x_train,y_train)
```

```
Fitting 5 folds for each of 20 candidates, totalling 100 fits
```

```
RandomizedSearchCV(cv=5, estimator=LogisticRegression(), n_iter=20,
                   param_distributions={'C': array([1.00000000e-04,
2.63665090e-04, 6.95192796e-04, 1.83298071e-03,
       4.83293024e-03, 1.27427499e-02, 3.35981829e-02, 8.85866790e-02,
       2.33572147e-01, 6.15848211e-01, 1.62377674e+00, 4.28133240e+00,
       1.12883789e+01, 2.97635144e+01, 7.84759970e+01, 2.06913808e+02,
       5.45559478e+02, 1.43844989e+03, 3.79269019e+03,
1.00000000e+04]),
                                        'solver': ['liblinear']},
                   verbose=True)
```

```
log_reg.best_params_
```

```
{'solver': 'liblinear', 'C': 0.0001}
```

```
log_reg.score(x_test,y_test)
```

```
0.5833333333333334
```

```
# rf_grid={"n_estimators": np.arange(10.1000,50),
#          'bootstrap': [True, False],
# #          "max_depth": [None]
#         "min_samples_split": np.arange(2,20,2),
#         "min_samples_leaf":np.arange(1,20,2)}

# np.random.seed(42)
# rsr=RandomizedSearchCV(RandomForestClassifier(),
#                        param_distributions= rf_grid,
#                        cv=5,
#                        n_iter=20,
#                        verbose=True)
# rsr.fit(x_train,y_train)

# hyperparameter tuning with grid search

log_grid={"C":np.logspace(-4,4,20),
          "solver":["liblinear"]}
log_reg=GridSearchCV(LogisticRegression(),
                           param_grid=log_grid,
```

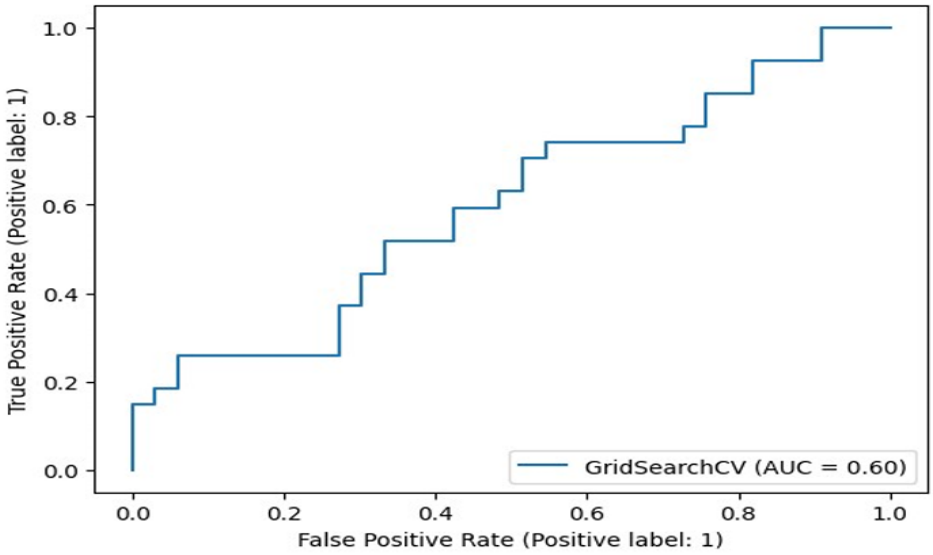
```
                           cv=5,

                           verbose=True)
# fit the model
log_reg.fit(x_train,y_train);
```

```
Fitting 5 folds for each of 20 candidates, totalling 100 fits
```

```
log_reg.best_params_
```

```
{'C': 0.0001, 'solver': 'liblinear'}
```

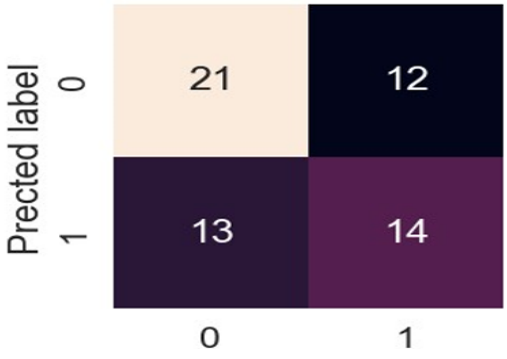
```
log_reg.score(x_test,y_test)
```




```
print(confusion_matrix(y_test,y_pre))
```

```
[[21 12]
 [13 14]]
```

```
sns.set(font_scale=1.5)
def plots(y_test,y_pre):
    fig,ax=plt.subplots(figsize=(3,3))
    ax=sns.heatmap(confusion_matrix(y_test,y_pre),
                   annot=True,
                   cbar=False)
    plt.xlabel("True label")
    plt.ylabel("Prected label")
plots(y_test,y_pre)
```



Classification Report

```
print(classification_report(y_test,y_pre))
```

```
              precision    recall  f1-score   support

           0       0.62      0.64      0.63        33
           1       0.54      0.52      0.53        27

    accuracy                           0.58        60
   macro avg       0.58      0.58      0.58        60
weighted avg       0.58      0.58      0.58        60
```

```
log_reg.best_params_
```

```
{'C': 0.0001, 'solver': 'liblinear'}
```

```
clf=LogisticRegression(C=0.0001,
                       solver="liblinear")

cv_acc=cross_val_score(clf,
                       x,
                       y,
                       cv=5,
                       scoring="accuracy")
cv_acc
```

```
array([0.61666667, 0.55      , 0.63333333, 0.7       , 0.57627119])
```

```
cv_acc=np.mean(cv_acc)
cv_acc
```

```
0.6152542372881357
```

```
cv_precision=cross_val_score(clf,
                       x,
                       y,
                       cv=5,
                       scoring="precision")
cv_precision=np.mean(cv_precision)
cv_precision
```

```
0.6206652802630942
```

```
cv_recall =cross_val_score(clf,
                       x,
                       y,
                       cv=5,
                       scoring="recall")
cv_recall= np.mean(cv_recall)
cv_recall
```

```
0.5450574712643678
```

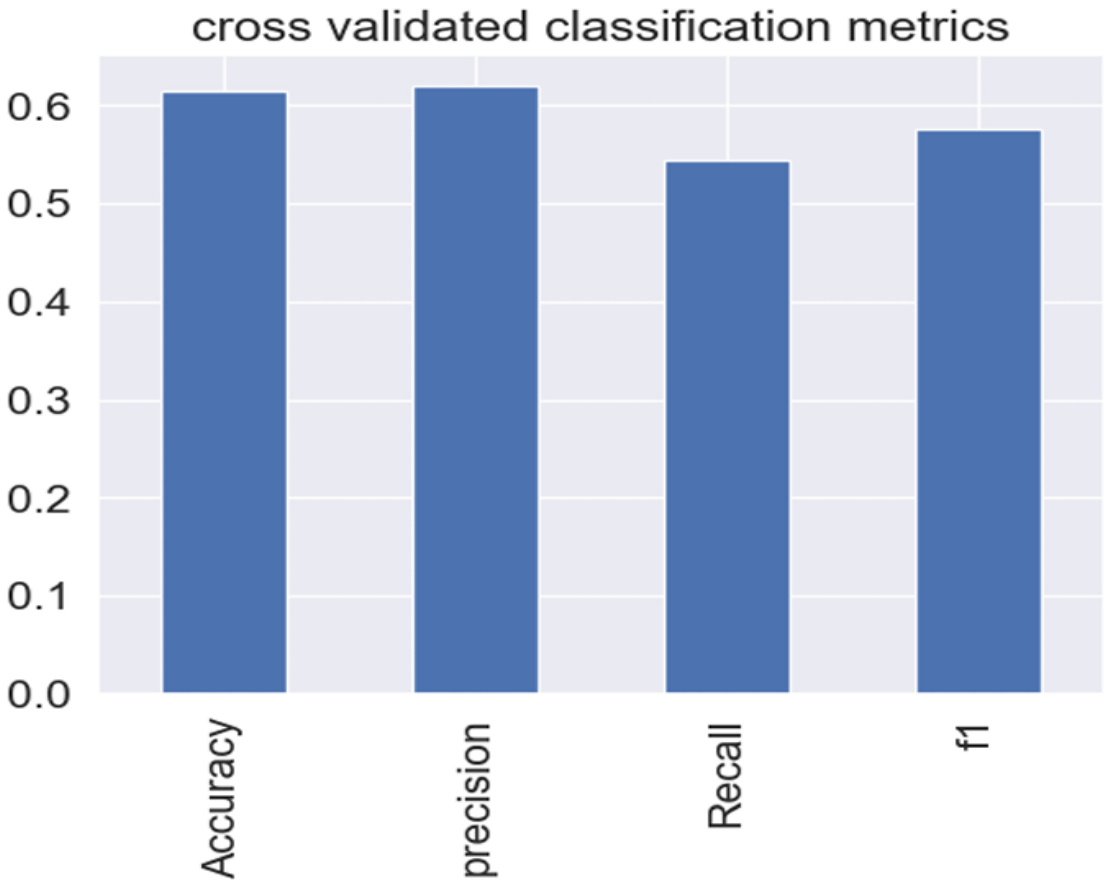
```
cv_f1score=cross_val_score(clf,
                       x,
                       y,
                       cv=5,
                       scoring="f1")
cv_f1score=np.mean(cv_f1score)
cv_f1score
```

```
0.5765739586187583
```

```
cvplot=pd.DataFrame({"Accuracy":cv_acc,
                     "precision": cv_precision,
                     "Recall": cv_recall,
```

```
cvplot=pd.DataFrame({"Accuracy":cv_acc,
                     "precision": cv_precision,
                     "Recall": cv_recall,
                     "f1":cv_f1score},index=[0])
cvplot.T.plot.bar(title="cross validated classification
metrics",legend=False)
```

```
<Axes: title={'center': 'cross validated classification metrics'}>
```



Program Used

- **Python**: For data preprocessing, model training, and evaluation.
- **Libraries**: Pandas, Scikit-learn, NumPy, Matplotlib (for visualization if needed).

References

- **UCI Machine Learning Repository**: Heart Disease Dataset.
- **Scikit-learn Documentation**: https://scikit-learn.org/
- **Pandas Documentation**: https://pandas.pydata.org/
- **Matplotlib Documentation**: https://matplotlib.org/

Student *Self Evaluation of the Short-Term Internship*

Please rate your performance in the following areas:

1) Oral communication	**1**	**2**	**3**	**4**	**5**
2) Written communication	**1**	**2**	**3**	**4**	**5**
3) Initiative	**1**	**2**	**3**	**4**	**5**
4) Interaction with staff	**1**	**2**	**3**	**4**	**5**
5) Attitude	**1**	**2**	**3**	**4**	**5**
6) Dependability	**1**	**2**	**3**	**4**	**5**
7) Ability to learn	**1**	**2**	**3**	**4**	**5**
8) Planning and organization	**1**	**2**	**3**	**4**	**5**
9) Professionalism	**1**	**2**	**3**	**4**	**5**
10) Creativity	**1**	**2**	**3**	**4**	**5**
11) Quality of work	**1**	**2**	**3**	**4**	**5**
12) Productivity	**1**	**2**	**3**	**4**	**5**
13) Progress of learning	**1**	**2**	**3**	**4**	**5**
14) Adaptability to organization's culture/policies	**1**	**2**	**3**	**4**	**5**
15) OVERALL PERFORMANCE	**1**	**2**	**3**	**4**	**5**

Rating Scale: 5

Annexure 1
Weekly Progress Report

Week(s)	Summary of Weekly Activity
Week 1	*Introduction to Machine Learning* The first week was dedicated to understanding the basics of machine learning. This included an overview of what machine learning is, its applications, types of machine learning (supervised, unsupervised, and reinforcement learning), and key concepts such as features, labels, and datasets. We also learned about the typical workflow of a machine learning project, from data collection and preprocessing to model training and evaluation.
Week 2	*Looping and Control Structures* In the second week, we focused on foundational programming skills essential for machine learning. We learned about looping constructs (for and while loops) and control structures (if-else statements) in Python. These constructs are crucial for data manipulation and iterative processes in machine learning algorithms. Practical exercises included writing scripts to process data and implement basic algorithms.
Week 3	*NumPy, Pandas, and Matplotlib* The third week was dedicated to data manipulation and visualization, key skills for any machine learning practitioner. We learned: • **NumPy**: For numerical operations and array manipulations.

	• **Pandas**: For data analysis and manipulation, working with DataFrames. • **Matplotlib**: For data visualization, creating plots, and visualizing the results of data analysis.
Week 4	*Building a Project with Scikit-learn* The final week culminated in applying the skills learned to build a machine learning project using Scikit-learn. This involved: • **Data Preprocessing**: Using techniques learned in previous weeks to clean and prepare the data. • **Model Training**: Implementing and training various machine learning models. • **Model Evaluation**: Using metrics to evaluate model performance and selecting the best model. • **Deployment**: Understanding the basics of deploying a machine learning model for practical use.