```
import numpy as np
import pandas as pd
pd.plotting.register_matplotlib_converters()
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
print("Setup Complete")
```

```
Setup Complete
```

Load the Dataset

```
# Read the file into a variable heart
heart= pd.read_csv('/content/dataset_heart.csv')
```

Exploratory Data Analysis

Explore the Data: Get a basic understanding of the dataset.

```
heart.head()
```

	age	sex	chest pain type	resting blood pressure	serum cholestoral	fasting blood sugar	resting electrocardiographic results	max heart rate	exercise induced angina	oldpeak	ST segment	major vessels	thal	heart disease
0	70	1	4	130	322	0	2	109	0	2.4	2	3	3	2
1	67	0	3	115	564	0	2	160	0	1.6	2	0	7	1
2	57	1	2	124	261	0	0	141	0	0.3	1	0	7	2
3	64	1	4	128	263	0	0	105	1	0.2	2	1	7	1
4	74	0	2	120	269	0	2	121	1	0.2	1	1	3	1

```
heart.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 270 entries, 0 to 269
Data columns (total 14 columns):
 #   Column                                Non-Null Count  Dtype  
---  ------                                --------------  -----  
 0   age                                   270 non-null    int64  
 1   sex                                   270 non-null    int64  
 2   chest pain type                       270 non-null    int64  
 3   resting blood pressure                270 non-null    int64  
 4   serum cholestoral                     270 non-null    int64  
 5   fasting blood sugar                   270 non-null    int64  
 6   resting electrocardiographic results  270 non-null    int64  
 7   max heart rate                        270 non-null    int64  
 8   exercise induced angina               270 non-null    int64  
 9   oldpeak                               270 non-null    float64
 10  ST segment                            270 non-null    int64  
 11  major vessels                         270 non-null    int64  
 12  thal                                  270 non-null    int64  
 13  heart disease                         270 non-null    int64  
dtypes: float64(1), int64(13)
memory usage: 29.7 KB
```

```
# print the number of rows and columns
print("Number of Rows: ", heart.shape[0])
print("Number of Columns: ", heart.shape[1])
```

```
Number of Rows:  270
Number of Columns:  14
```

```
heart.isnull().sum()
```

```
age                                     0
sex                                     0
chest pain type                         0
resting blood pressure                  0
serum cholestoral                       0
fasting blood sugar                     0
resting electrocardiographic results    0
max heart rate                          0
```

```
exercise induced angina              0
oldpeak                              0
ST segment                           0
major vessels                        0
thal                                 0
heart disease                        0
dtype: int64
```

```
heart.describe().T.style.background_gradient(cmap = "Reds")
```

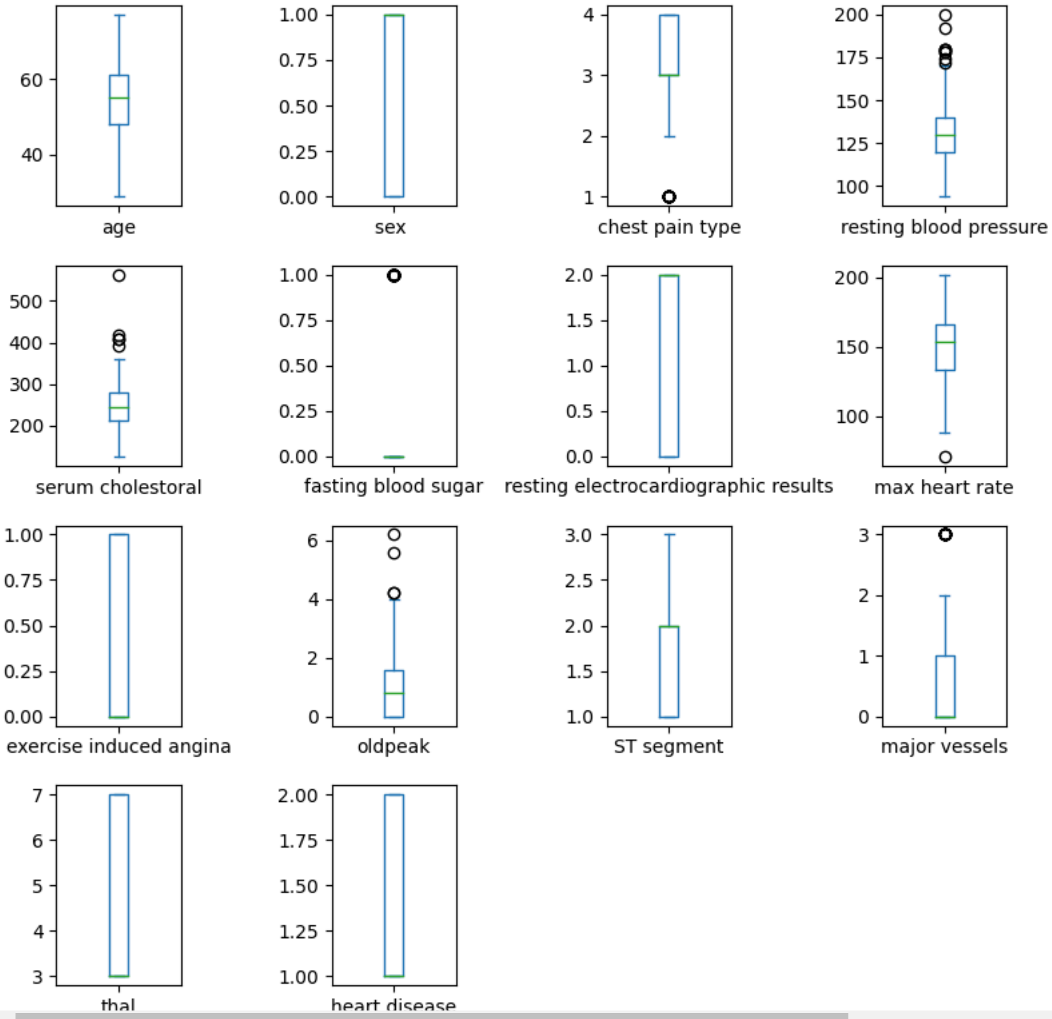
	count	mean	std	min	25%	50%	75%	max
age	270.000000	54.433333	9.109067	29.000000	48.000000	55.000000	61.000000	77.000000
sex	270.000000	0.677778	0.468195	0.000000	0.000000	1.000000	1.000000	1.000000
chest pain type	270.000000	3.174074	0.950090	1.000000	3.000000	3.000000	4.000000	4.000000
resting blood pressure	270.000000	131.344444	17.861608	94.000000	120.000000	130.000000	140.000000	200.000000
serum cholestoral	270.000000	249.659259	51.686237	126.000000	213.000000	245.000000	280.000000	564.000000
fasting blood sugar	270.000000	0.148148	0.355906	0.000000	0.000000	0.000000	0.000000	1.000000
resting electrocardiographic results	270.000000	1.022222	0.997891	0.000000	0.000000	2.000000	2.000000	2.000000
max heart rate	270.000000	149.677778	23.165717	71.000000	133.000000	153.500000	166.000000	202.000000
exercise induced angina	270.000000	0.329630	0.470952	0.000000	0.000000	0.000000	1.000000	1.000000
oldpeak	270.000000	1.050000	1.145210	0.000000	0.000000	0.800000	1.600000	6.200000
ST segment	270.000000	1.585185	0.614390	1.000000	1.000000	2.000000	2.000000	3.000000
major vessels	270.000000	0.670370	0.943896	0.000000	0.000000	0.000000	1.000000	3.000000
thal	270.000000	4.696296	1.940659	3.000000	3.000000	3.000000	7.000000	7.000000
heart disease	270.000000	1.444444	0.497827	1.000000	1.000000	1.000000	2.000000	2.000000

Data Visualization: Create visualizations to understand the data.

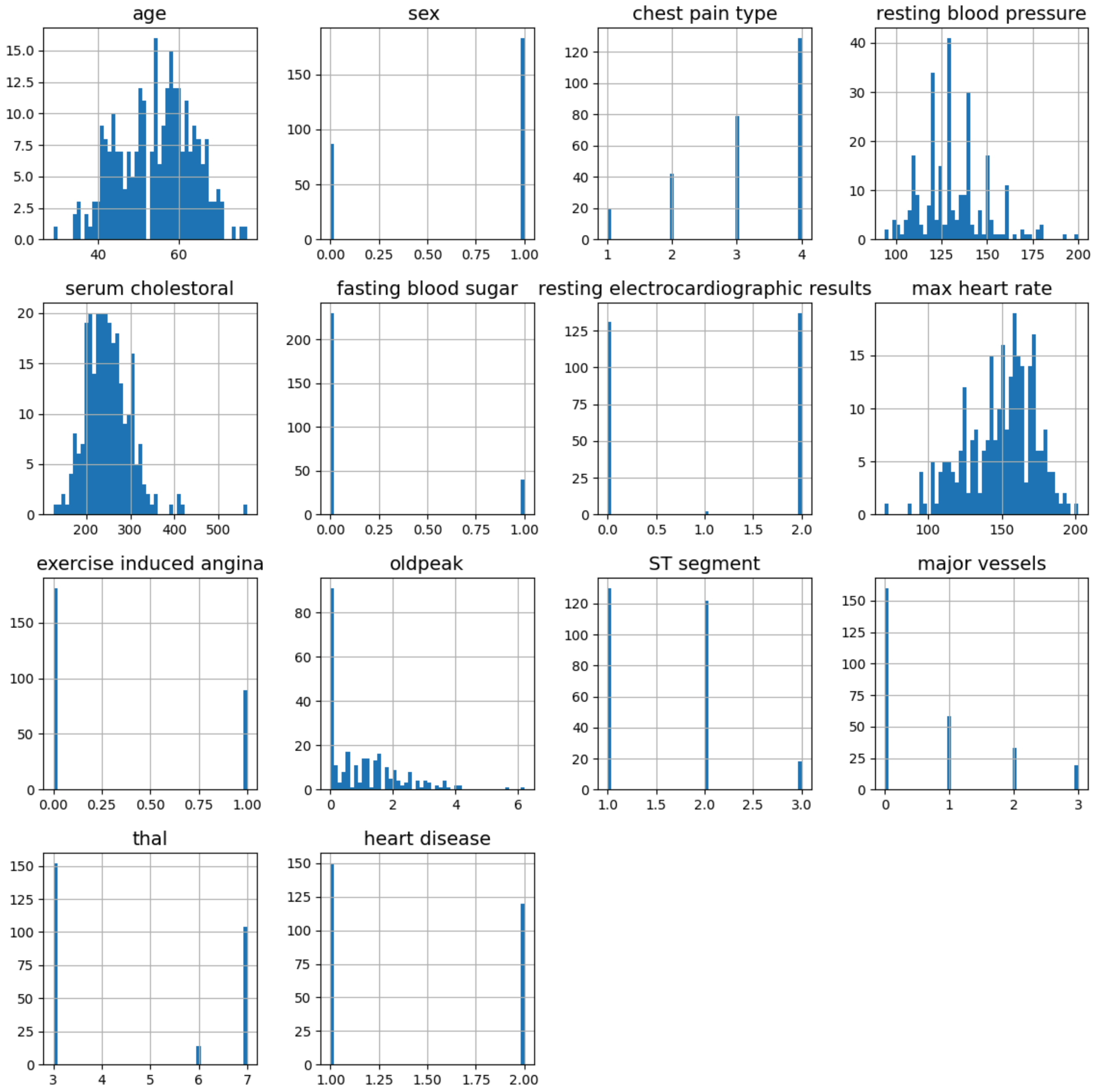
```
heart.columns
```

```
Index(['age', 'sex ', 'chest pain type', 'resting blood pressure',
       'serum cholestoral', 'fasting blood sugar',
       'resting electrocardiographic results', 'max heart rate',
       'exercise induced angina', 'oldpeak', 'ST segment', 'major vessels',
       'thal', 'heart disease'],
      dtype='object')
```

```
#Take the column values
names = ['age', 'sex ', 'chest pain type', 'resting blood pressure',
       'serum cholestoral', 'fasting blood sugar',
       'resting electrocardiographic results', 'max heart rate',
       'exercise induced angina', 'oldpeak', 'ST segment', 'major vessels',
       'thal', 'heart disease']

# Set the custom font sizes
plt.rc('font', size=14)
plt.rc('axes', labelsize=14, titlesize=14)
plt.rc('legend', fontsize=14)
plt.rc('xtick', labelsize=10)
plt.rc('ytick', labelsize=10)

# Create the box plots
heart.plot(kind='box', subplots=True, layout=(4, 4), sharex=False, sharey=False, figsize=(8, 8), y=names)

# Adjust the layout and spacing
plt.tight_layout()
plt.show()

```

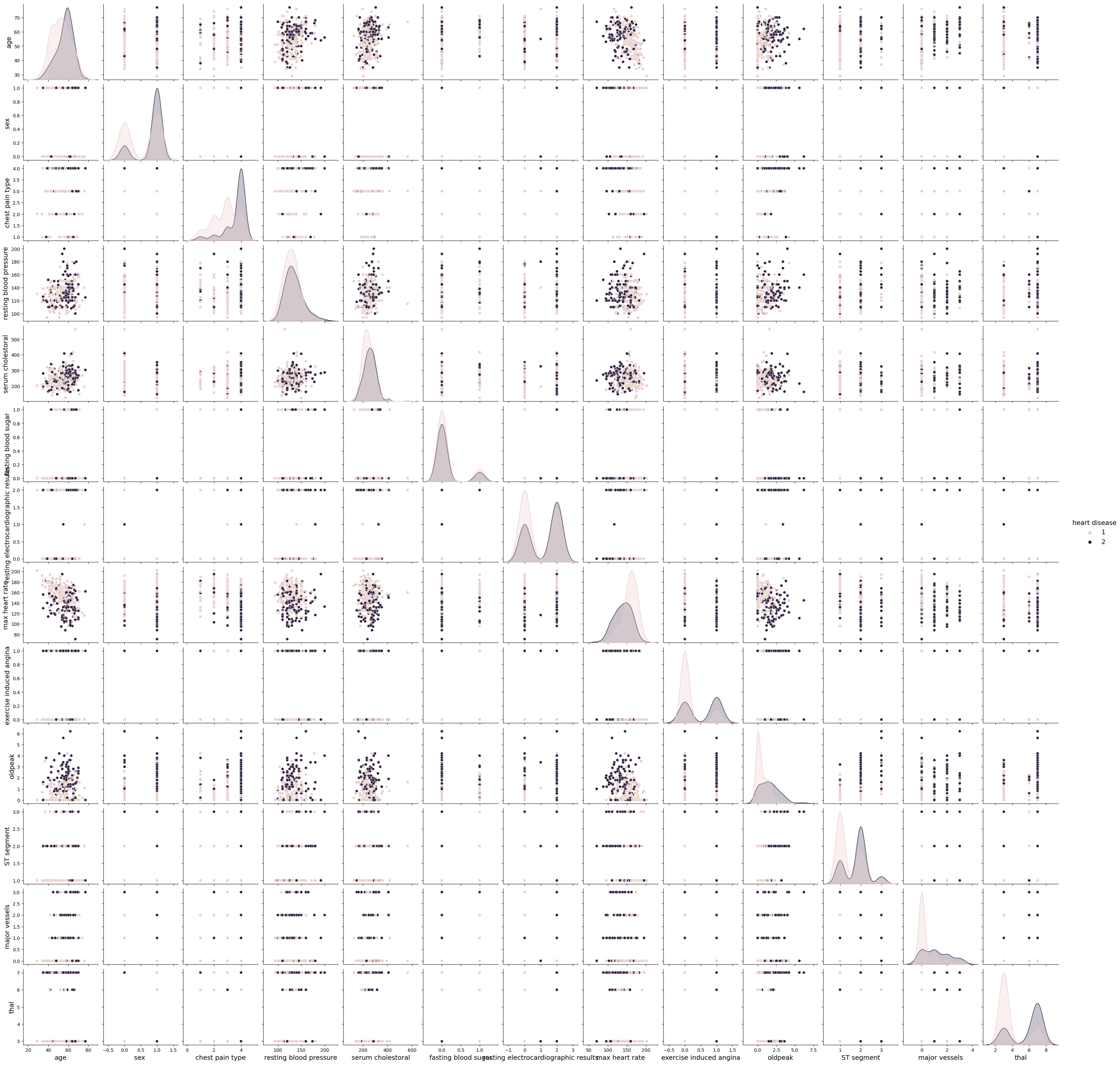


```
# extra code – the next 5 lines define the default font sizes
plt.rc('font', size=14)
plt.rc('axes', labelsize=14, titlesize=14)
plt.rc('legend', fontsize=14)
plt.rc('xtick', labelsize=10)
plt.rc('ytick', labelsize=10)

heart.hist(bins=50, figsize=(14, 14))

plt.show()
```



```
sns.pairplot(heart,hue='heart disease')
```

```
<seaborn.axisgrid.PairGrid at 0x7e33aa84dc00>
```



Spliting Independent And Dependent features

```
X=heart.drop('heart disease',axis=1)
y=heart['heart disease']
```

```
X
```

	age	sex	chest pain type	resting blood pressure	serum cholestoral	fasting blood sugar	resting electrocardiographic results	max heart rate	exercise induced angina	oldpeak	ST segment	major vessels	thal
0	70	1	4	130	322	0	2	109	0	2.4	2	3	3
1	67	0	3	115	564	0	2	160	0	1.6	2	0	7
2	57	1	2	124	261	0	0	141	0	0.3	1	0	7
3	64	1	4	128	263	0	0	105	1	0.2	2	1	7
4	74	0	2	120	269	0	2	121	1	0.2	1	1	3
...	...	...	...	...	...	...	...	...	...	...	...	...	...
265	52	1	3	172	199	1	0	162	0	0.5	1	0	7
266	44	1	2	120	263	0	0	173	0	0.0	1	0	7
267	56	0	2	140	294	0	2	153	0	1.3	2	0	3
268	57	1	4	140	192	0	0	148	0	0.4	2	0	6
269	67	1	4	160	286	0	2	108	1	1.5	2	3	3

270 rows × 13 columns

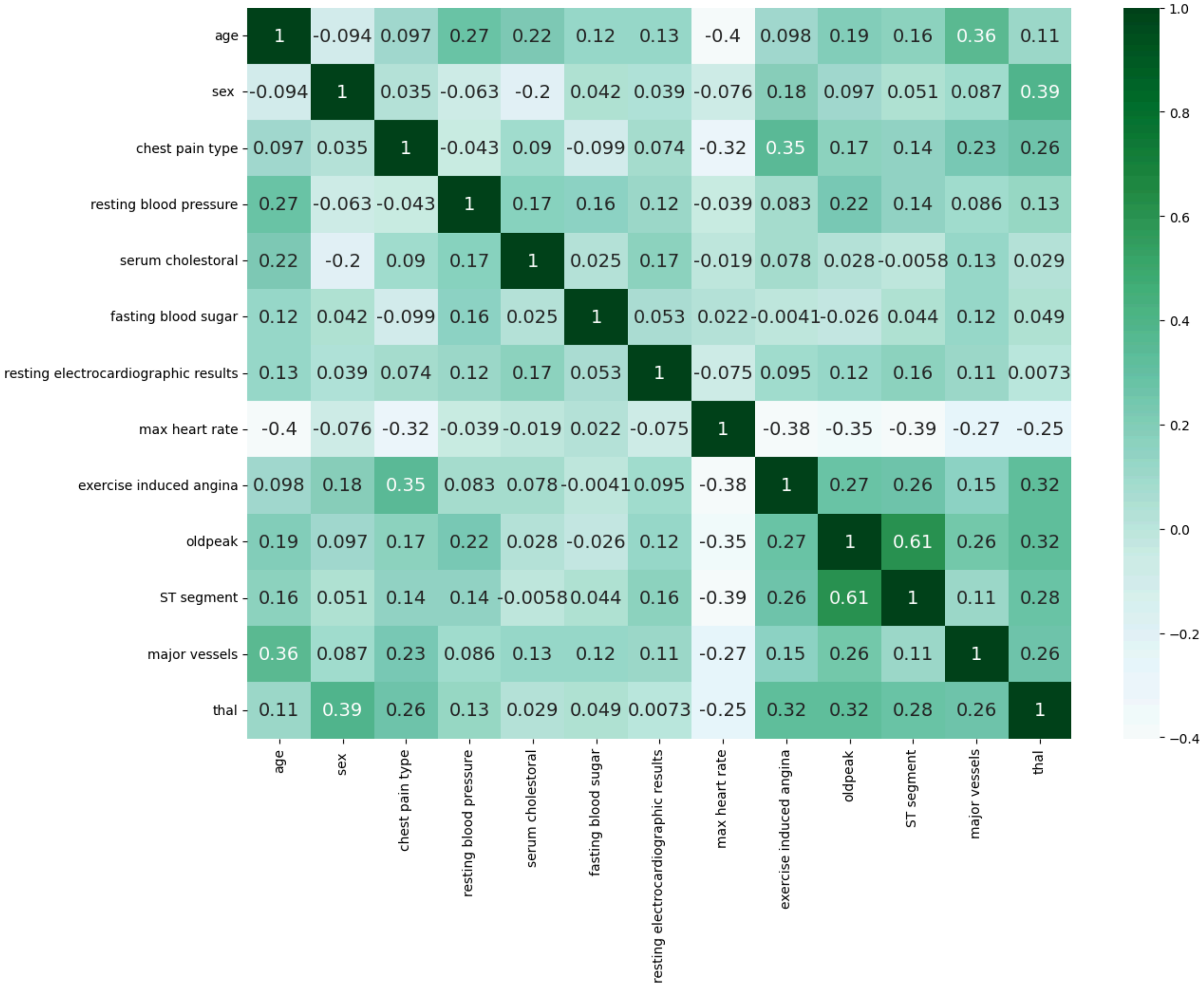
```
y
```

```
0      2
1      1
2      2
3      1
4      1
      ..
265    1
266    1
267    1
268    1
269    2
Name: heart disease, Length: 270, dtype: int64
```

Looking for Correlation

```
corr_matrix = X.corr()
```

```
import seaborn as sns
plt.figure(figsize = (14,10))
sns.heatmap(corr_matrix, annot = True, cmap = 'BuGn')
plt.show()
```



Stratified Train Test Split Data

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state = 42, stratify = y)
```

```
X_train
```

	age	sex	chest pain type	resting blood pressure	serum cholestoral	fasting blood sugar	resting electrocardiographic results	max heart rate	exercise induced angina	oldpeak	ST segment	major vessels	thal
78	42	0	3	120	209	0	0	173	0	0.0	2	0	3
121	54	1	4	122	286	0	2	116	1	3.2	2	2	3
27	51	0	3	120	295	0	2	157	0	0.6	1	0	3
198	69	0	1	140	239	0	0	151	0	1.8	1	2	3
218	54	1	3	120	258	0	2	147	0	0.4	2	0	7
...	...	...	...	...	...	...	...	...	...	...	...	...	...
86	62	1	2	128	208	1	2	140	0	0.0	1	0	3
109	45	0	2	112	160	0	0	138	0	0.0	2	0	3
225	41	1	2	135	203	0	0	132	0	0.0	2	0	6
128	52	1	2	134	201	0	0	158	0	0.8	1	1	3
130	63	0	4	108	269	0	0	169	1	1.8	2	2	3

216 rows × 13 columns

```
print(X.shape, X_train.shape, X_test.shape)
```

```
(270, 13) (216, 13) (54, 13)
```

Data Preprocessing

Standard Scaler

```
from sklearn.preprocessing import StandardScaler
df2 = heart.copy()
ss = StandardScaler()
df2[['age', 'resting blood pressure','serum cholestoral', 'max heart rate','oldpeak']] = ss.fit_transform(df2[['age','resting blood
```

Handling Outliers

```
for col in heart.columns:
    if heart[col].dtypes != 'object':
        lower_limit, upper_limit = heart[col].quantile([0.25,0.75])
        IQR = upper_limit - lower_limit
        lower_whisker = lower_limit - 1.5 * IQR
        upper_whisker = upper_limit + 1.5 * IQR
        heart[col] = np.where(heart[col]>upper_whisker,upper_whisker,np.where(heart[col]<lower_whisker,lower_whisker,heart[col]))
```

Model Training & Evaluation

```
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
```

```
knn =KNeighborsClassifier(n_neighbors=5)
logreg = LogisticRegression()
dt = DecisionTreeClassifier(random_state=0)
rf = RandomForestClassifier()
```

```
knn.fit(X_train, y_train)
logreg.fit(X_train, y_train)
dt.fit(X_train, y_train)
rf.fit(X_train, y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=
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(
▾ RandomForestClassifier
RandomForestClassifier()
```

```python
from sklearn.metrics import accuracy_score

y_pred = knn.predict(X_test)
print('K-Nearest Neighbors  Test Accuracy ', accuracy_score(y_test, y_pred ))

y_pred = logreg.predict(X_test)
print('Logistic Regression Test Accuracy ', accuracy_score(y_test, y_pred ))

y_pred = dt.predict(X_test)
print('Decision Tree Test Accuracy ', accuracy_score(y_test, y_pred ))

y_pred = rf.predict(X_test)
print('Random Forest Test Accuracy ', accuracy_score(y_test, y_pred ))

```

```
K-Nearest Neighbors  Test Accuracy  0.7222222222222222
Logistic Regression Test Accuracy  0.8518518518518519
Decision Tree Test Accuracy  0.7592592592592593
Random Forest Test Accuracy  0.8333333333333334
```

```python
from sklearn.metrics import classification_report

def plot_classification_report(y_train, y_pred1, y_test, y_pred2, c_name):
    print("-"*25,c_name,"(TRAIN SET)","-"*25)
    print(classification_report(y_train, y_pred1))
    print("-"*25,c_name,"(Test SET)","-"*25)
    print(classification_report(y_test, y_pred2))
```

```python
c_name= "K-Nearest Neighbors"
plot_classification_report(y_train, knn.predict(X_train), y_test, knn.predict(X_test), c_name)

c_name= "Logistic Regression"
plot_classification_report(y_train, logreg.predict(X_train), y_test, logreg.predict(X_test), c_name)

c_name= "Decision Tree"
plot_classification_report(y_train, dt.predict(X_train), y_test, dt.predict(X_test), c_name)

c_name= "Random Forest"
plot_classification_report(y_train, rf.predict(X_train), y_test, rf.predict(X_test), c_name)
```

```
------------------------- K-Nearest Neighbors (TRAIN SET) -------------------------
              precision    recall  f1-score   support

           1       0.75      0.78      0.76       120
           2       0.71      0.67      0.69        96

    accuracy                           0.73       216
   macro avg       0.73      0.72      0.73       216
weighted avg       0.73      0.73      0.73       216

------------------------- K-Nearest Neighbors (Test SET) -------------------------
              precision    recall  f1-score   support

           1       0.76      0.73      0.75        30
           2       0.68      0.71      0.69        24

    accuracy                           0.72        54
   macro avg       0.72      0.72      0.72        54
weighted avg       0.72      0.72      0.72        54

------------------------- Logistic Regression (TRAIN SET) -------------------------
              precision    recall  f1-score   support

           1       0.86      0.90      0.88       120
           2       0.87      0.82      0.84        96

    accuracy                           0.87       216
   macro avg       0.87      0.86      0.86       216
weighted avg       0.87      0.87      0.87       216

------------------------- Logistic Regression (Test SET) -------------------------
```