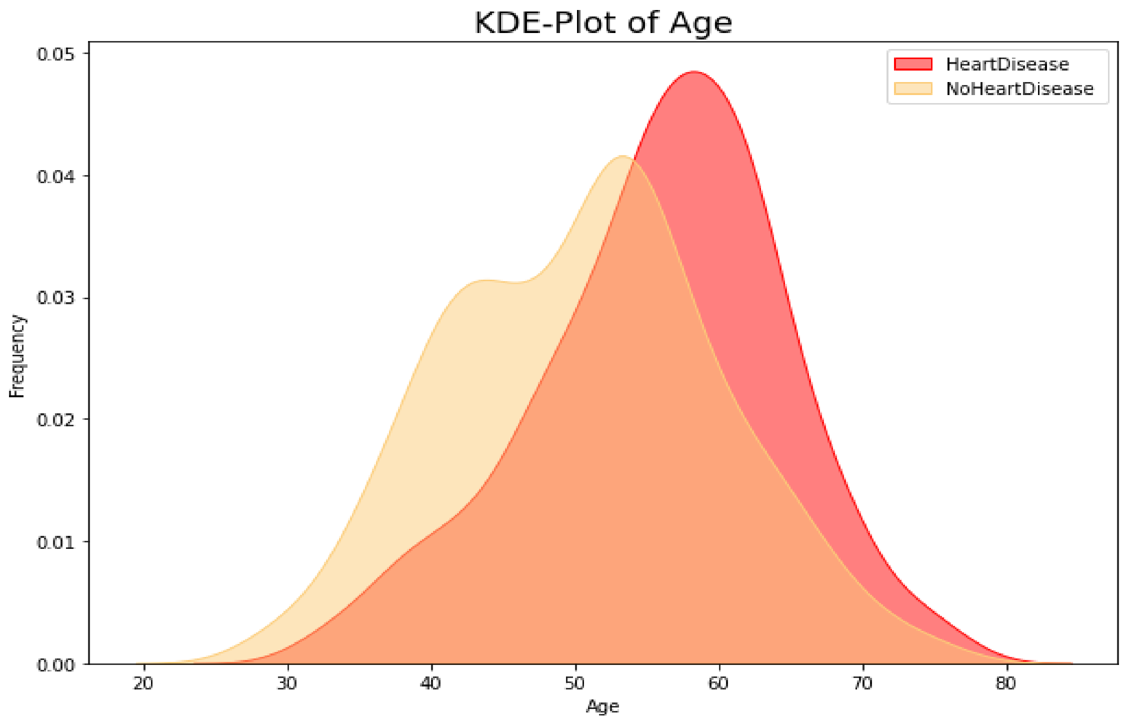
Heart Failure Prediction

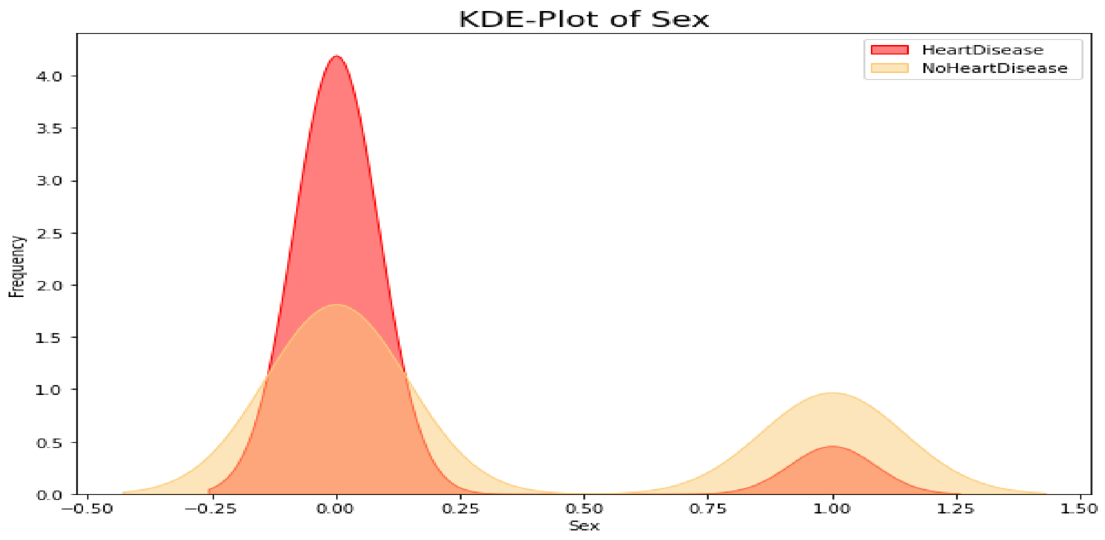
Sudip Kumar (B20CS072) and Rahul Saha (B20CS081)

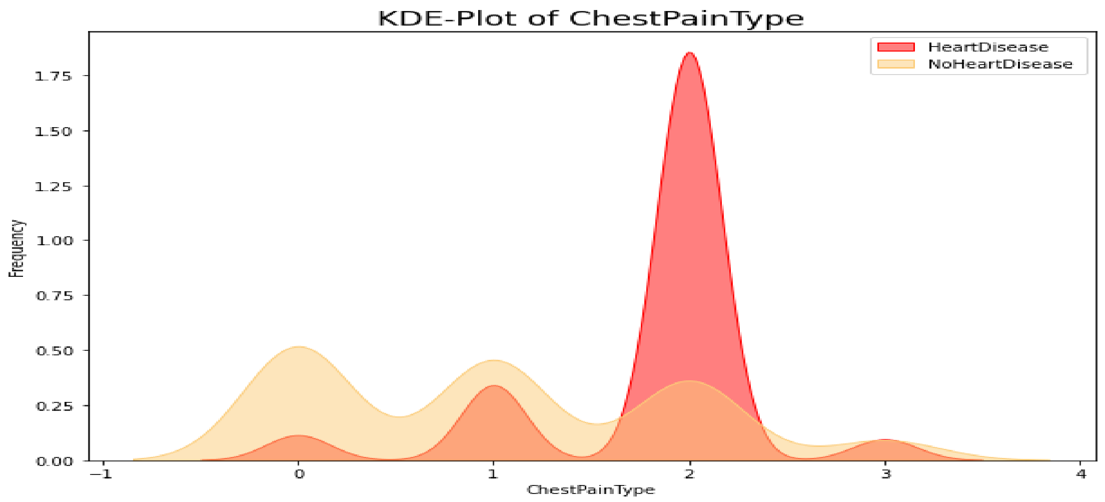
Data set:
https://drive.google.com/file/d/1osorOQ4uUz-iqpAJ1rwfiJY4xzcyU9e9/view?usp=sharing

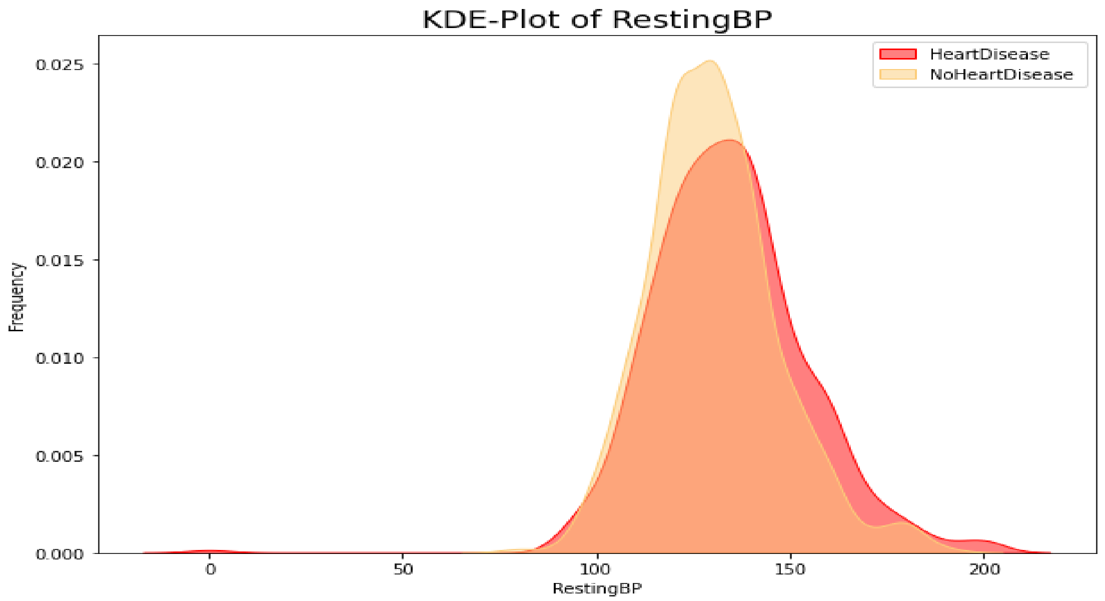
We have done our project by going through these several steps:

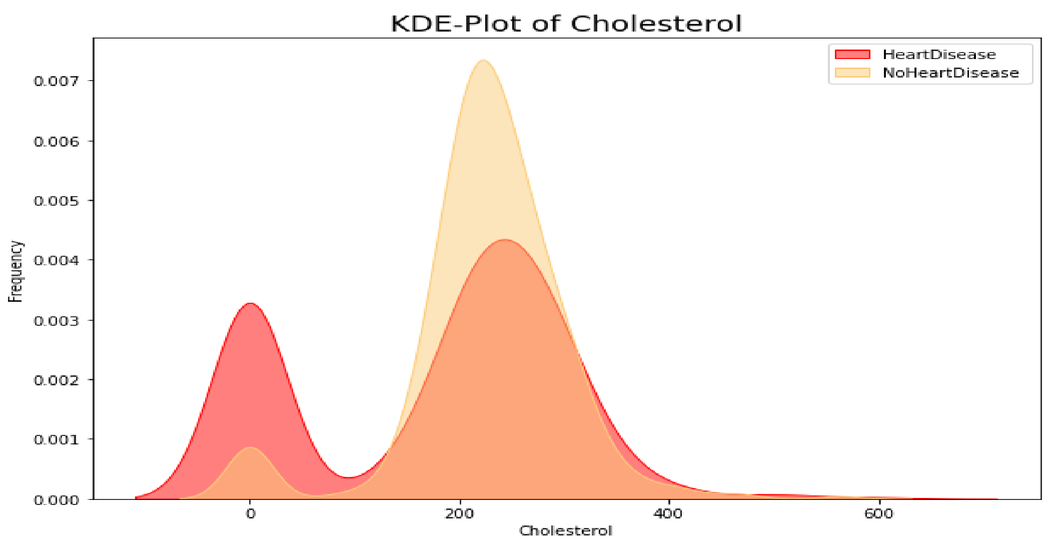
- Step 1: we imported the provided data set in our colaboratory file by importing pandas.
- Step 2: we go through the preprocessing procedure.
- Step 3: we had done the data visualization
- Step 4: then plotted the graph for keeping X- axis as age and Y-axis as the disease by importing matplotlib.pyplot.

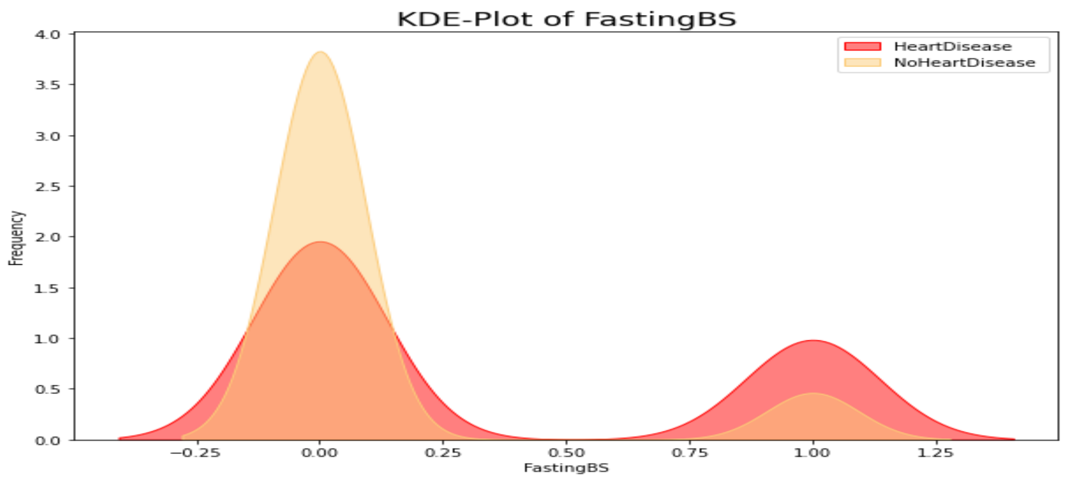


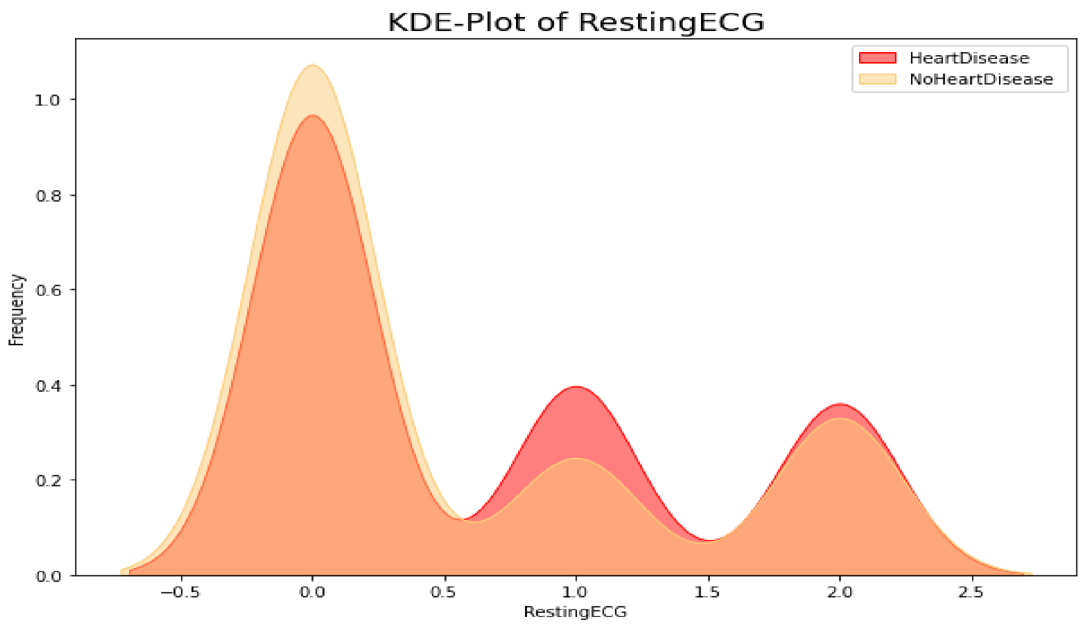

KDE-Plot of Sex
HeartDisease
NoHeartDisease
Frequency
Sex
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
1.25
1.50

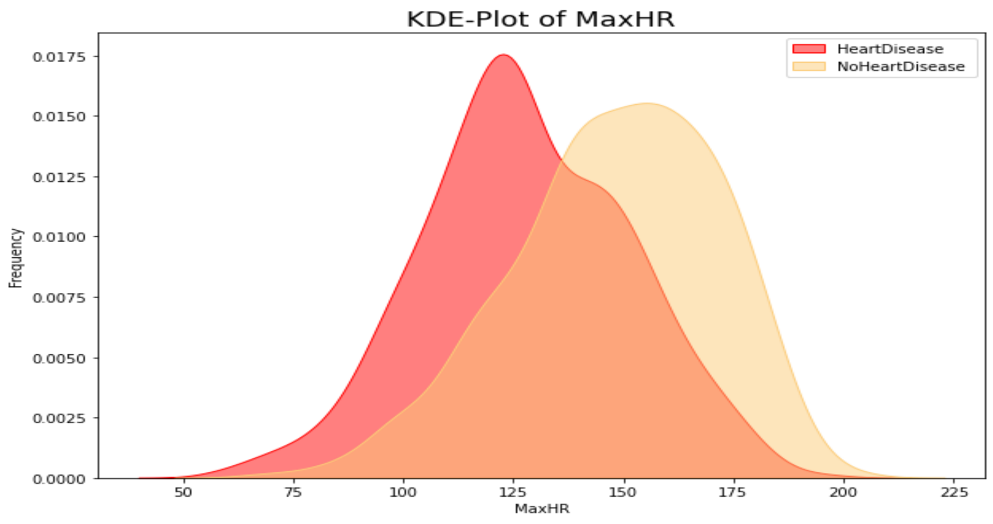

KDE-Plot of ChestPainType
HeartDisease
NoHeartDisease
Frequency
ChestPainType
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
−1
0
1
2
3
4

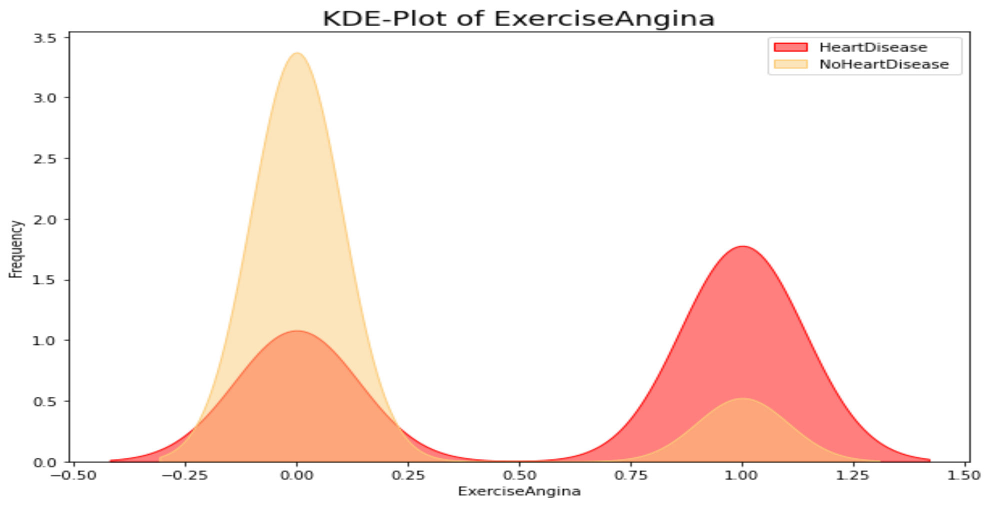

KDE-Plot of RestingBP
HeartDisease
NoHeartDisease
Frequency
RestingBP
0.000
0.005
0.010
0.015
0.020
0.025
0
50
100
150
200

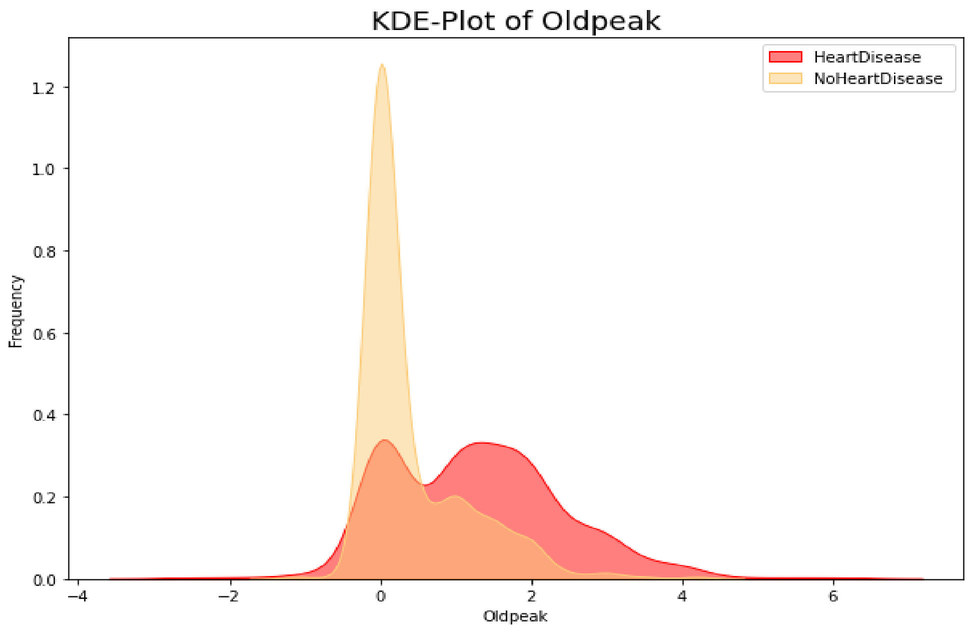

KDE-Plot of Cholesterol
HeartDisease
NoHeartDisease
Frequency
Cholesterol
0.000
0.001
0.002
0.003
0.004
0.005
0.006
0.007
0
200
400
600

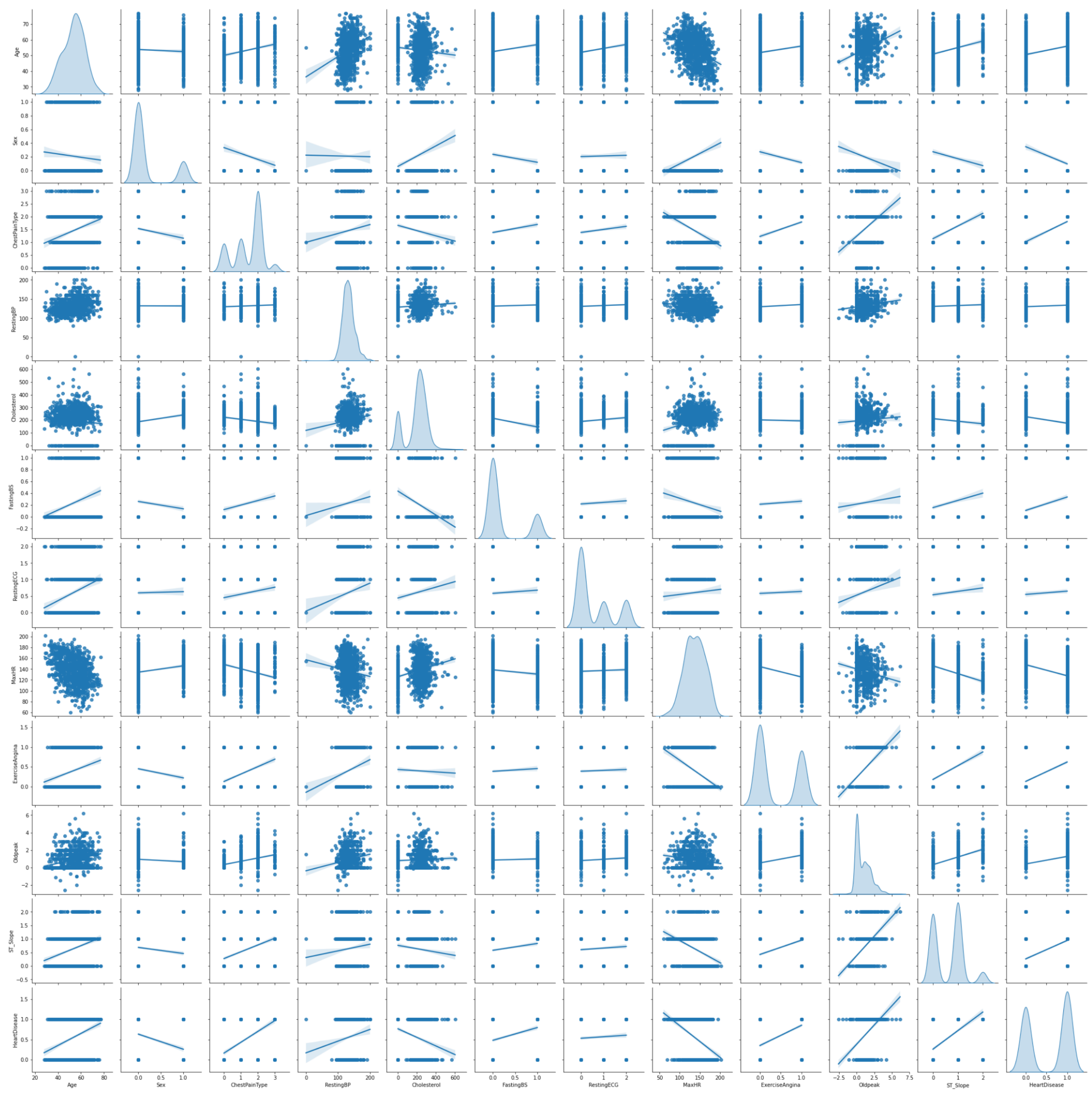

KDE-Plot of FastingBS
HeartDisease
NoHeartDisease
Frequency
FastingBS
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
−0.25
0.00
0.25
0.50
0.75
1.00
1.25


KDE-Plot of RestingECG
HeartDisease
NoHeartDisease
Frequency
RestingECG
0.0
0.2
0.4
0.6
0.8
1.0
−0.5
0.0
0.5
1.0
1.5
2.0
2.5


KDE-Plot of MaxHR
HeartDisease
NoHeartDisease
Frequency
0.0175
0.0150
0.0125
0.0100
0.0075
0.0050
0.0025
0.0000
50
75
100
125
150
175
200
225
MaxHR


KDE-Plot of ExerciseAngina
HeartDisease
NoHeartDisease
Frequency
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
1.25
1.50
ExerciseAngina


KDE-Plot of Oldpeak
HeartDisease
NoHeartDisease
Frequency
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−4
−2
0
2
4
6
Oldpeak



- Step 5: we have done the train and test splitting by importing train test split
- Step 6: Started with implementing different model applications. The models are as follows:
    1. from sklearn.linear_model import LogisticRegression
        - In the multiclass case, the training algorithm uses the one-vs-rest (OvR) scheme if the 'multi_class' option is set to

    ‘ovr’, and uses the cross-entropy loss if the ‘multi_class’ option is set to ‘multinomial’.
  - This class implements regularized logistic regression using the ‘liblinear’ library, ‘newton-cg’, ‘sag’, ‘saga’ and ‘lbfgs’ solvers. **Note that regularization is applied by default**.
  - Our accuracy using this model is 0.855072463768116
2. from sklearn.ensemble import BaggingClassifier
  - A Bagging classifier is an ensemble meta-estimator that fits base classifiers each on random subsets of the original dataset and then aggregate their individual predictions to form a final prediction.
  - Such a meta-estimator can typically be used as a way to reduce the variance of a black-box estimator, by introducing randomization into its construction procedure and then making an ensemble out of it.
  - In this model we have taken certain parameters like we have taken “The number of base estimators in the ensemble is 10” and “ random state is 0”.
  - Our accuracy using this model is 0.8369565217391305
3. from sklearn import svm
  - The sklearn.svm module includes Support Vector Machine algorithms.
  - We have used C-Support Vector Classification with random state = 42.
  - We have also imported cross validation model to cross validate the score.
  - We have checked our accuracy at different CV values and picked the best among them.

- Our best accuracy using this model is at CV = 10 i.e. 0.95196078

4. from sklearn.gaussian_process import GaussianProcessClassifier
    - Gaussian process classification (GPC) based on Laplace approximation.
    - The implementation is based on Algorithm 3.1, 3.2, and 5.1 of Gaussian Processes for Machine Learning (GPML) by Rasmussen and Williams.
    - We have used the parameter 'kernel'. The kernel "1.0 * RBF(1.0)" is used as default.
    - Our accuracy using this model is 0.8369565217391305

5. from sklearn.tree import DecisionTreeClassifier
    - The goal of this classifier is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.
    - A tree can be seen as a piecewise constant approximation.
    - We have also imported cross validation model to cross validate the score.
    - We have checked our accuracy at different CV values and picked the best among them.
    - Our best accuracy using this model is at CV = 16 i.e. 0.9122807