IT4060 -Machine Learning

Assignment -2

Group Details

Name	Student ID
Athapaththu P.N.P.	IT19028774
Nandu Gamitha Manawadu	IT19140476
Sanduni Jayamali Gamage K.G.	IT19123578
M.G.D.D.B. Ekanayaka	IT19138732

Submitted to

Sri Lanka Institute of Information Technology

29 May 2022

Contents

INTRODUCTION

It is estimated that 12 million human lives are lost due to heart diseases in each year by the estimations done by the World Health Organization. Cardiovascular illnesses account for half of all fatalities in the industrialized countries. Early diagnosis of heart related deformities could help the patients at high risk to manage their lifestyles which will lessen the consequences. Using logistic regression, this study will identify the most relevant/risk variables for heart disease and forecast the total risk.

Logistic regression is one of most popular Machine learning Algorithm under the supervised learning approach . Logistic regression model predicts a dependent variable of type categorical using a collection of individual variables. Output of the predicted variable is predicting using machine learning algorithm of logistic regression. So as the outcome of the prediction will be a discreate or categorical result. Can be Yes/No or 0/1, True/False. Although returning numbers like 0 and 1 this will return probabilistic values in between 0 and 1. Except for how they are used, Logistic and Linear Regression algorithms are likely to be same. The difference is Linear Regression algorithm is used for regression related incidents, and Logistic Regression algorithm is utilized for classification related scenarios.

Data Set

The dataset is publicly found on the Kaggle website http://www.kaggle.com/ and comes from an ongoing cardiovascular study of Framingham, Massachusetts people. The categorization purpose is to determine whether the patient will develop heart related disease by the coming ten years (CHD). The dataset contains information on the patients. It consists of nearly 4,000 records and 15 qualities.Framingham.csv includes both demographic, behavioral and medical risk factors.

sex - male or female

age - age of the patient

currentSmoker-whether or not the patient is a current smoker (Nominal)

cigsPerDay-the number of cigarettes that the person smoked on average in one day

BPMeds-whether or not the patient was on blood pressure medication (Nominal)

prevalentStroke-whether or not the patient had previously had a stroke (Nominal)

prevalentHyp-whether or not the patient was hypertensive (Nominal)

diabetes-whether or not the patient had diabetes (Nominal)

totChol- total cholesterol level (Continuous)

sysBP- systolic blood pressure (Continuous)

diaBP -diastolic blood pressure (Continuous)

BMI - Body Mass Index (Continuous)

heartRate: heart rate (Continuous - In medical research, variables such as heart rate though in fact discrete, yet are considered continuous because of large number of possible values.)

glucose: glucose level (Continuous)

TenYearCHD -10 year risk of coronary heart disease CHD (binary: "1", means "Yes", "0" means "No")

	male	age	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	diabetes	totChol	sysBP	diaBP	BMI	heartRate	glucose	TenYearCHD
0	1	39	0	0.0	0.0	0	0	0	195.0	106.0	70.0	26.97	80.0	77.0	0
1	0	46	0	0.0	0.0	0	0	0	250.0	121.0	81.0	28.73	95.0	76.0	0
2	1	48	1	20.0	0.0	0	0	0	245.0	127.5	80.0	25.34	75.0	70.0	0
3	0	61	1	30.0	0.0	0	1	0	225.0	150.0	95.0	28.58	65.0	103.0	1
4	0	46	1	23.0	0.0	0	0	0	285.0	130.0	84.0	23.10	85.0	85.0	0

Figure 1- image of the head of dataset in tabular format

Methodology

Data Preprocessing

Different libraries and frameworks such as sklearn, numpy, pandas are used for the implementation. Dataset was loaded in .csv format and as the dataset is unknown, information of the data set was extracted.

```
import pandas as pd
import numpy as np
import statsmodels.api as sm
import scipy.stats as st
import matplotlib.pyplot as plt
import seaborn as sn
from sklearn.metrics import confusion_matrix
import matplotlib.mlab as mlab
import warnings
%matplotlib inline
warnings.filterwarnings("ignore")
sn.set_style("darkgrid")
```

Figure 2 - Libraries imported

```
#nimesha
# Loading Heart Data from framingham.csv
chd_df=pd.read_csv("framingham.csv")
#drop a column
chd_df.drop(['education'],axis=1,inplace=True)
chd_df.head()
```

Figure 3- Image loading the dataset and drop the unwanted colmun

```
#Find missing values
chd_df.isnull().sum()
```

```
sex_male             0
age                  0
currentSmoker        0
cigsPerDay          29
BPMeds              53
prevalentStroke      0
prevalentHyp         0
diabetes             0
totChol             50
sysBP                0
diaBP                0
BMI                 19
heartRate            1
glucose            388
TenYearCHD           0
dtype: int64
```

Figure 4 -Checking for null values

```
#Counting total no of rows with missing values
count=0
for i in chd_df.isnull().sum(axis=1):
    if i>0:
        count=count+1
print('Total number of rows with missing values =', count)
print('Percentage of rows with missing values in the dataset =
print('Therefore, the missing values are eliminated.')
```

```
Total number of rows with missing values = 489
Percentage of rows with missing values in the dataset = 12 %
Therefore, the missing values are eliminated.
```

Figure 5 -Checking for null values with rows

For preprocessing, dataset was checked for null values using isnull () and sum () methods of pandas library.Cigsperday ,BPMeds , totChol colmuns includes null values.
After the data preprocessing data visualization done. Used pandas unique () methods to extract information about the dataset and used matplotlib and seaborn to visualize whether there are any relationship s among the attributes.

Data Vizulization

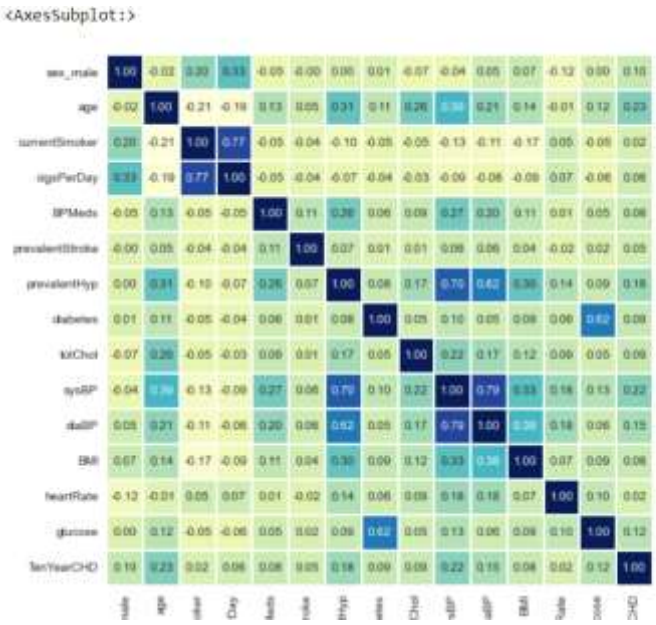


Figure-6 Heatmap of the data set

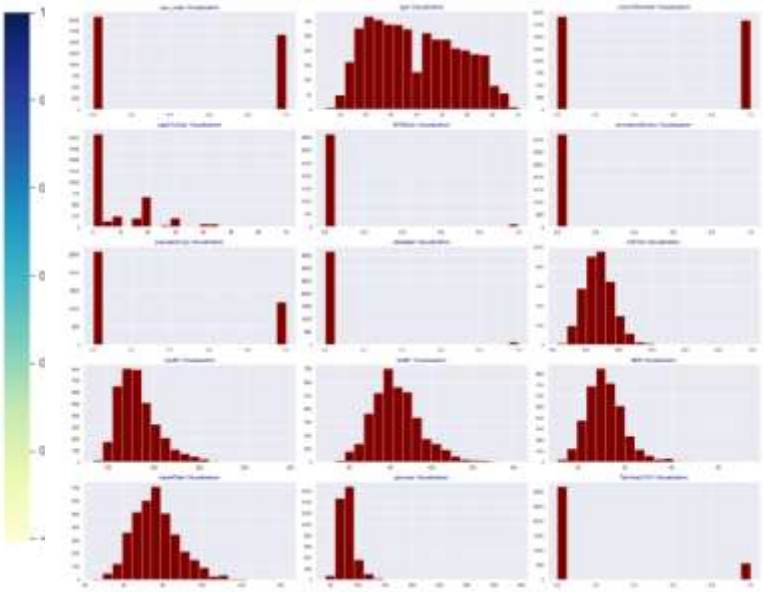


Figure -7 analysis of the attributes

```
1]: #Plot a graph for the TenYearCHD feature value data
    sn.countplot(x='TenYearCHD',data=chd_df)

1]: <AxesSubplot:xlabel='TenYearCHD', ylabel='count'>
```

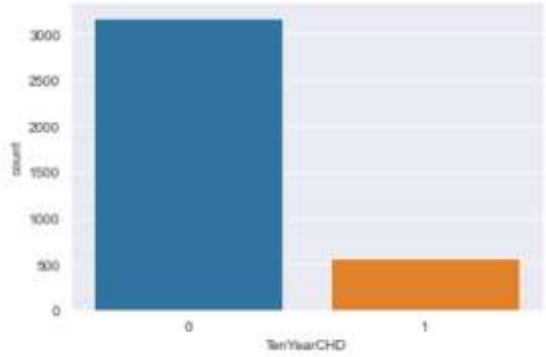


Figure -8 plot graph of the Feature Value data

Feature Selection

'age','sex_male','cigsPerDay','totChol','sysBP','glucose','TenYearCHD' selected as features

```
def backward_elemination (hd_frame,dependent_var,column_list):
    while len(column_list)>0 :
        model=sm.Logit(dependent_var,hd_frame[column_list])
        result=model.fit(disp=0)
        largest_pvalue=round(result.pvalues,3).nlargest(1)
        if largest_pvalue[0]<(0.05):
            return result
            break
        else:
            column_list=column_list.drop(largest_pvalue.index)

result=backward_elemination(heart_details_constant,chd_df.TenYearCHD,col)
```

Figure -9 Feature Selection

In here hd_frame dependent variable and a list of column names, runs the regression repeatedly eleminating feature with the highest P-value above alpha one at a time and returns the regression summary with all p-values below alpha.

```
result.summary()
```

Logit Regression Results

Dep. Variable:	TenYearCHD	**No. Observations:**	3749
Model:	Logit	**Df Residuals:**	3742
Method:	MLE	**Df Model:**	6
Date:	Sun, 29 May 2022	**Pseudo R-squ.:**	0.1148
Time:	08:06:53	**Log-Likelihood:**	-1417.6
converged:	True	**LL-Null:**	-1601.4
Covariance Type:	nonrobust	**LLR p-value:**	2.548e-76

	coef	std err	z	P>|z|	[0.025	0.975]
const	-9.1211	0.468	-19.491	0.000	-10.038	-8.204
sex_male	0.5813	0.105	5.521	0.000	0.375	0.788
age	0.0654	0.006	10.330	0.000	0.053	0.078
cigsPerDay	0.0197	0.004	4.803	0.000	0.012	0.028
totChol	0.0023	0.001	2.099	0.036	0.000	0.004
sysBP	0.0174	0.002	8.166	0.000	0.013	0.022
glucose	0.0076	0.002	4.573	0.000	0.004	0.011

Figure -10 Summary of the Feature Selection

After All Features Plughed in the logistic regression equation according below ,

$$logit(p) = log(p/(1-p)) = \beta_0 + \beta_1 * Sexmale + \beta_2 * age + \beta_3 * cigsPerDay + \beta_4 * totChol + \beta_5 * sysBP + \beta_6 * glucose$$

Splitting dataset for training and testing

```
#Nimesha
import sklearn
new_hd_features=chd_df[['age','sex_male','cigsPerDay','totChol','sysBP','glucose','TenYearCHD']]
x=new_hd_features.iloc[:,:-1]
y=new_hd_features.iloc[:,-1]
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.20,random_state=5)
print(x_train,x_test,y_train,y_test)
```

Figure -11 splitting the dataset.

'age','sex_male','cigsPerDay','totChol','sysBP','glucose','TenYearCHD' selected as features
The dataset was splitted for training and testing in the ratio of 80: 20 which gives the highest accuraccay.

Algorithem Used

We have selected the Logistic Regression since it's a type of statistical regression analysis that predicts the outcome of a categorical variable using a set of prediction or independent variables. In logistic regression methods , the dependent variable will be binary. Prediction and estimating the chance of success are two of the main uses of logistic regression models.
Equation of the Logistic regression

$$P = e^{\beta_0+\beta_1 X_1} / 1 + e^{\beta_0+\beta_1 X_1}$$

With Logistic regression we go the highest accuracy for the model which was 0.88 accuracy.

Results and Discussion

According to the below figure -12 these results showed ,

- Using this fitted model, the probability of being diagnosed with heart disease for men (sex male = 1) over females (sex male = 0) is exp(0.5815) = 1.788687, while all other factors remain constant.
- In terms of percentage change, males have 78.8 percent greater probabilities than females. Since exp(0.0655) = 1.067644, the coefficient for age states that for every one year rise in age, we will observe a 7% increase in the chances of being diagnosed with CDH.
- Similarly, every additional cigarette smoked increases one's chances of developing CDH by 2%.
- There is no substantial change in total cholesterol or glucose levels.
- Every increase in systolic blood pressure raises the risk by 1.7 percent.

```
hd_params = np.exp(result.params)
conf_intervals = np.exp(result.conf_int())
conf_intervals['OR'] = hd_params
pvalue=round(result.pvalues,3)
conf_intervals['pvalue']=pvalue
conf_intervals.columns = ['CI 95%(2.5%)', 'CI 95%(97.5%)', 'Odds Ratio','pvalue']
print ((conf_intervals))
```

	CI 95%(2.5%)	CI 95%(97.5%)	Odds Ratio	pvalue
const	0.000044	0.000274	0.000109	0.000
sex_male	1.454877	2.198166	1.788313	0.000
age	1.054409	1.080897	1.067571	0.000
cigsPerDay	1.011730	1.028128	1.019896	0.000
totChol	1.000150	1.004386	1.002266	0.036
sysBP	1.013299	1.021791	1.017536	0.000
glucose	1.004343	1.010895	1.007614	0.000

Figure -12 Interpreting the results: Odds Ratio, Confidence Intervals and Pvalues

```
sklearn.metrics.accuracy_score(y_test,y_prediction)
```

```
0.8706666666666667
```

Figure -13 Final Accuracy of the model

As conclusion finally we can say ,

- After the elimination process, all of the traits picked had Pvalues of less than 5%, indicating that they have a substantial influence in the prediction of heart disease.
- It appears that men are at a high risk to heart related disease than women.
- Increases in age, daily cigarette smoking, and systolic blood pressure all indicate an increased risk of heart disease.**
- Total cholesterol has no effect on the risk of coronary heart disease. This might be because the total cholesterol result includes "good cholesterol" (HDL). Glucose, however, has a minor effect on the chances (0.2 percent)
- The model has an accuracy of 0.88. The model is more sensitive than specific.
- The Area Under the ROC Curve (AUC) is 73.5, which is acceptable.
- More data might enhance the overall model.

Appendix

Video Link –

https://mysliit-my.sharepoint.com/:v:/g/personal/it19028774_my_sliit_lk/Edb7n5ZsTAVKlqfGTf4Wbe8BgMEBnSpbeNb81X0Npz4GMw?e=y43Fk2

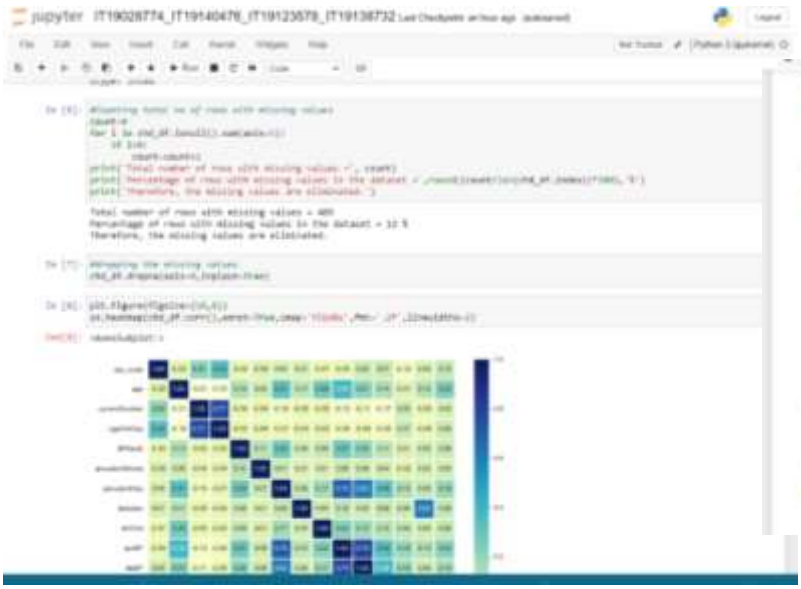
Git hub Link

https://github.com/NimeshaPrasadini/ML_Assignment2_IT19028774_IT19140476_IT19123578_IT19138732.git

Contribution of Members

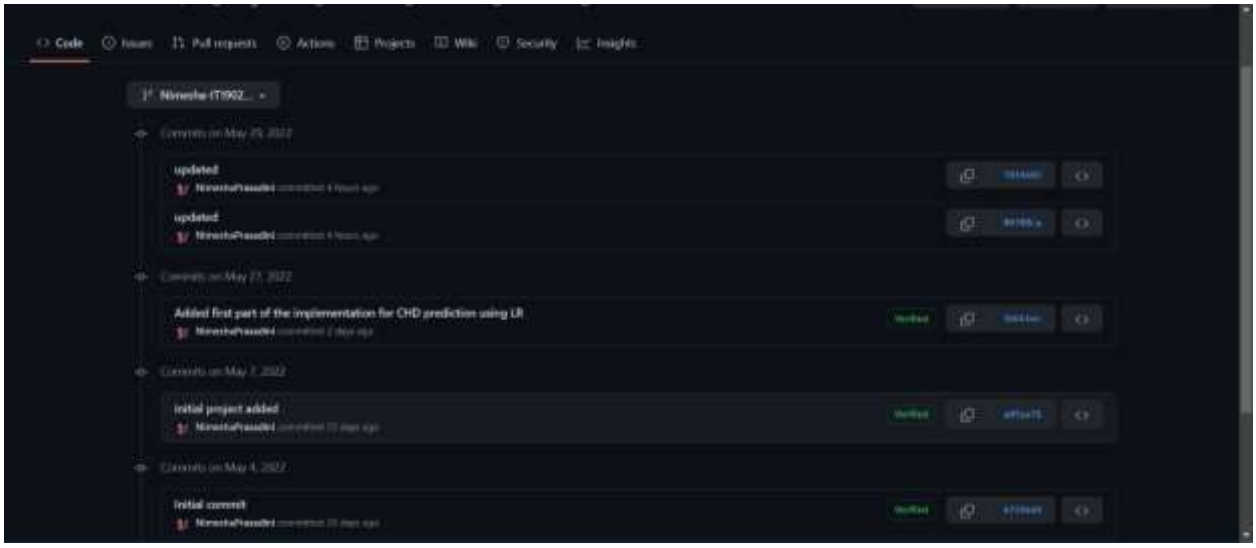
Athapaththu P.N.P.	IT19028774

Import required libraries and packages
Drop education column because it's not necessary for implementations. Check data shape and data types. Rename 'male' column name as 'sex_male'. Check whether there is any duplicate columns
Find missing values.Count total no of rows with missing values. Drop the missing values
Visualize heatmap for dataframe using sn.heatmap().Exploratory Analysis by drawing histograms for CHD features in the dataframe(using draw_chd_histograms function)
Count'TenYearCHD' feature values using value_counts()
Plot a graph for the 'TenYearCHD' feature value data using sn.countplot()
Plot graphs for all feature data in the dataframe using sn.pairplot()
Describe all feature data in the dataframe using .describe()
- count - no of non-empty values
- mean - average (mean) value
- std - standard deviation
- min - minimum value
- 25% - 25% percentile
- 50% - 50% percentile
- 75% - 75% percentile



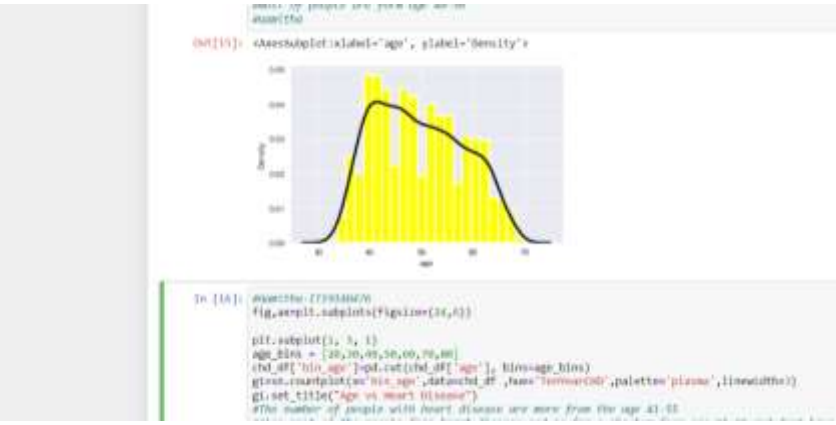

In [9]:
#Exploratory Analysis by drawing histograms for CHD features
def draw_chd_histograms(dataframe, features, rows, cols):
fig_chd=plt.figure(figsize=(20,20))
for i, feature in enumerate(features):
ax_chd=fig_chd.add_subplot(rows,cols,i+1)
dataframe[feature].hist(bins=20,ax=ax_chd,facecolor='maroon')
ax_chd.set_title(feature+" Visualization",color='navy')
fig_chd.tight_layout()
plt.show()
#Call the histogram function
draw_chd_histograms(chd_df,chd_df.columns,6,3)

Git Hub


Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
updated
updated
Added first part of the implementation for CHD prediction using LR
initial project added
Initial commit

Nandu Gamitha Manawadu IT19140476

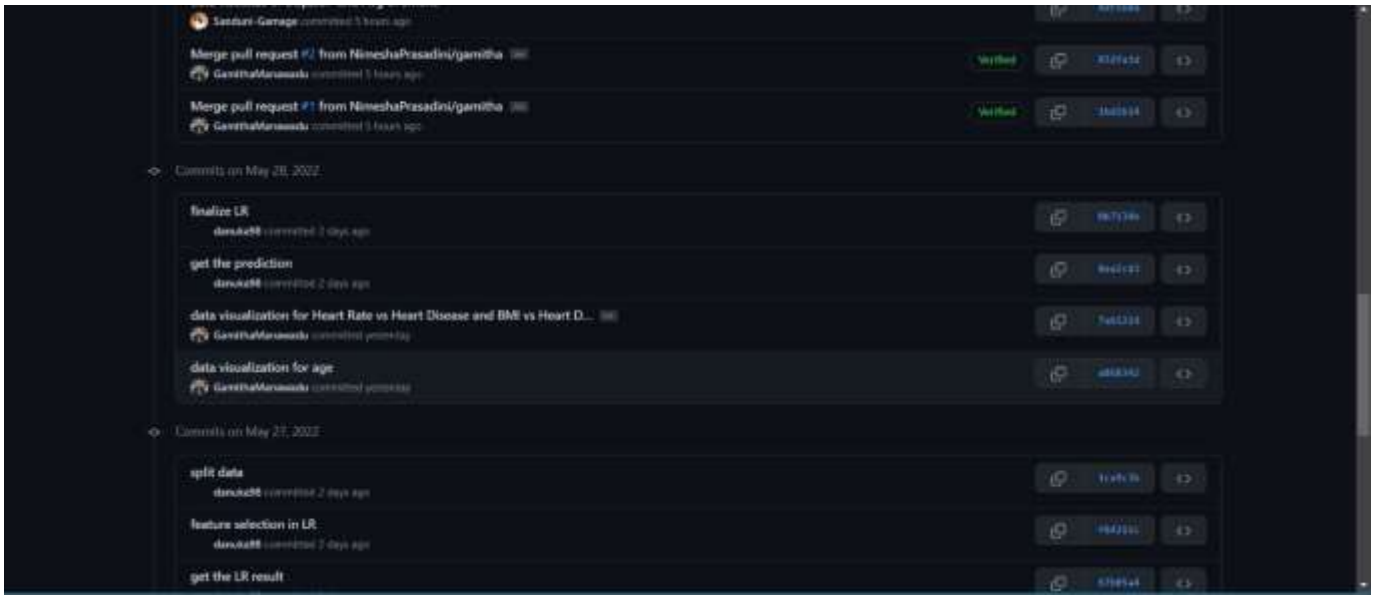
Logistic regression algorithm was used as the machine learning algorithm for the problem .
I contributed to visualize the age density of the dataset and also, I visualized the age vs heart diseases graph, cholesterol level vs heart disease graph, heart rate vs heart disease graph and also BMI vs heart disease graph.
I used Matplotlib libraries such as Seaborn to visualize random visualizations. Confusion matrix was used in the model evaluation to calculate sensitivity. As the sensitivity of the false negative was high
I used Sklearn preprocessing library of Binarize to implement a method to test the sensitivity using lowering the threshold of the model.

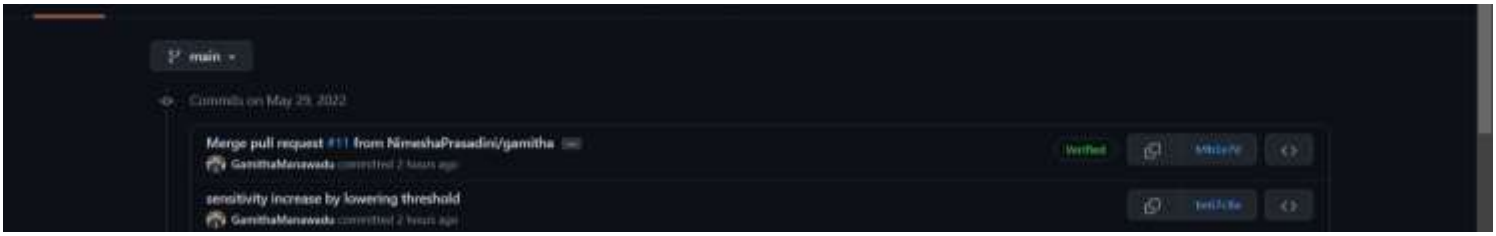


```
In [16]: #use(the-777016047b
         fig,ax=plt.subplots(figsize=(24,5))

         plt.subplot(1, 3, 1)
         age_bins = [30,35,40,50,60,70,80]
         chd_df['bin_age']=pd.cut(chd_df['age'], bins=age_bins)
         g1=sn.countplot(x='bin_age',data=chd_df ,hue='TenYearCHD',palette='plasma',linewidth=1)
         g1.set_title("Age vs Heart Disease")
         #The number of people with heart disease are more from the age 41-55
         #Also most of the people four heart disease and go for a checkup from age 55-65 and dont have heart disease (Precautions)

         plt.subplot(1, 3, 2)
         chol_bins = [100,150,200,250,300,350,600,650]
         chd_df['bin_chol']=pd.cut(chd_df['totChol'], bins=chol_bins)
         g2=sn.countplot(x='bin_chol',data=chd_df,hue='TenYearCHD',palette='plasma',linewidth=1)
         g2.set_title("Cholestoral vs Heart Disease")
         #Most people get the heart disease with 200-250 cholestrol
         #The others with cholestrol of above 250 tend to think they have heart disease but the rate of heart disease fails

         plt.subplot(1, 3, 3)
         hr_bins = [40,60,80,100,120,140]
         chd_df['bin_hr']=pd.cut(chd_df['heartRate'], bins=hr_bins)
         g3=sn.countplot(x='bin_hr',data=chd_df,hue='TenYearCHD',palette='plasma',linewidth=1)
         g3.set_title("Heart Rate vs Heart Disease")
         #People who have thalach between 100-100 have a very high chance of getting the heart disease

Out[16]: Text(0.5, 1.0, 'Heart Rate vs Heart Disease')
```

```
In [73]: #use(the-777016047b
         TN=cm[0,0]
         TP=cm[1,1]
         FN=cm[1,0]
         FP=cm[0,1]
         sensitivity=TP/float(TP+FN)
         specificity=TN/float(TN+FP)

In [74]: print('The accuracy of the model = TP+TN/(TP+TN+FP+FN) = ',(TP+TN)/float(TP+TN+FP+FN),'\n',

         'The Misclassification = 1-Accuracy = ',1-((TP+TN)/float(TP+TN+FP+FN)),'\n',

         'Sensitivity or True Positive Rate = TP/(TP+FN) = ',TP/float(TP+FN),'\n',

         'Specificity or True Negative Rate = TN/(TN+FP) = ',TN/float(TN+FP),'\n',

         'Positive Predictive value = TP/(TP+FP) = ',TP/float(TP+FP),'\n',

         'Negative predictive Value = TN/(TN+FN) = ',TN/float(TN+FN),'\n',

         'Positive Likelihood Ratio = Sensitivity/(1-Specificity) = ',sensitivity/(1-specificity),'\n',

         'Negative Likelihood Ratio = (1-Sensitivity)/Specificity = ',(1-sensitivity)/specificity)

         The accuracy of the model = TP+TN/(TP+TN+FP+FN) =  0.8706666666666667
          The Misclassification = 1-Accuracy =  0.1293333333333333
          Sensitivity or True Positive Rate = TP/(TP+FN) =  0.07766990291262135
          Specificity or True Negative Rate = TN/(TN+FP) =  0.9969088098918083
          Positive Predictive value = TP/(TP+FP) =  0.8
```

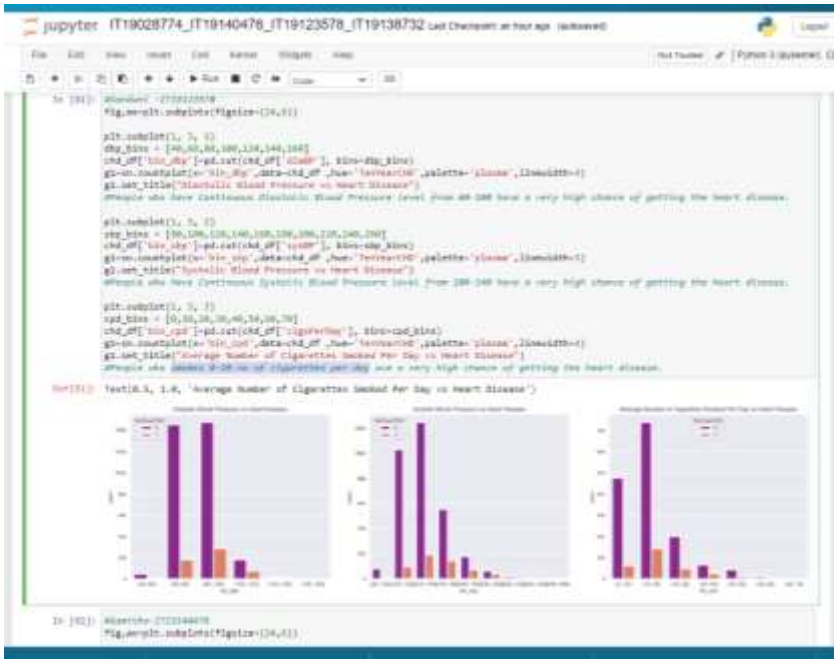
Git hub

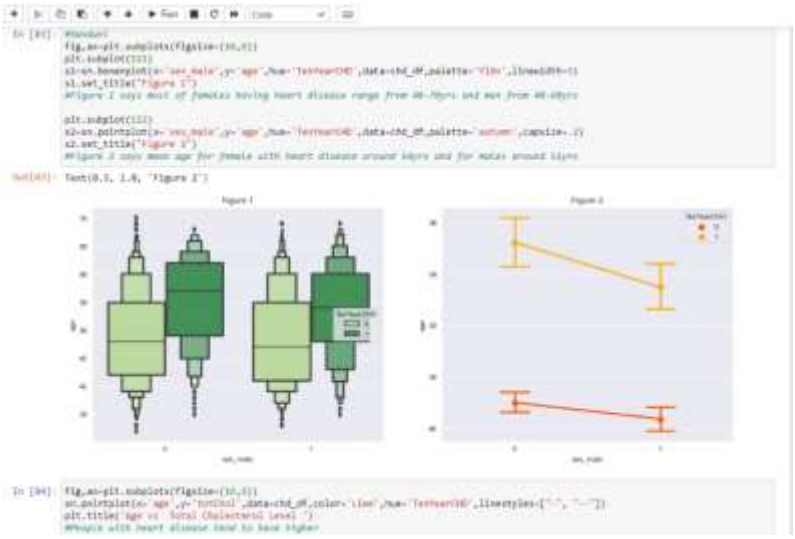

Merge pull request #2 from NimeshaPrasadini/gamitha
Merge pull request #1 from NimeshaPrasadini/gamitha
Commits on May 28, 2022
finalize LR
get the prediction
data visualization for Heart Rate vs Heart Disease and BMI vs Heart D...
data visualization for age
Commits on May 27, 2022
split data
feature selection in LR
get the LR result


main
Commits on May 29, 2022
Merge pull request #11 from NimeshaPrasadini/gamitha
sensitivity increase by lowering threshold

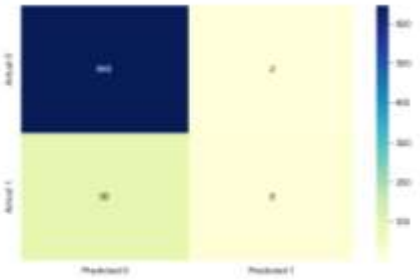
Sanduni Jayamali Gamage K.G. IT19123578

Logistic regression algorithm was used as the machine learning algorithm for the problem .
I contributed to data visualization
Get the plot of heart disease vs Continuous Diastolic Blood Pressure vs Continuous Systolic Blood vs smokes 0-20 no of cigarettes per day
Compare with the Accuracy with other Algorithms .
Also used Matplotlib libraries such as Seaborn to visualize random visualizations.
I get Confusion matrix for the data set
I get ROC Curve for best model.

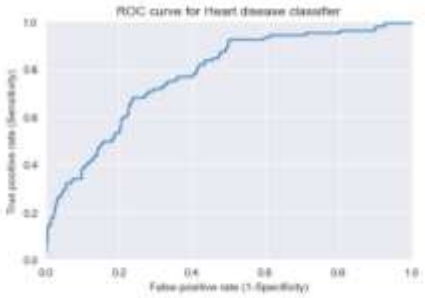




```
In [72]: #Sanduni -IT19123578
         from sklearn.metrics import confusion_matrix
         cm=confusion_matrix(y_test,y_prediction)
         conf_matrix=pd.DataFrame(data=cm,columns={'Predicted:0','Predicted:1'},index={'Actual:0','Actual:1'})
         plt.figure(figsize = (8,5))
         sn.heatmap(conf_matrix, annot=True,fmt='d',cmap="YlGnBu")

Out[72]: <AxesSubplot:>
```



```
In [77]: #Sanduni -IT19123578
         from sklearn.metrics import roc_curve
         fpr, tpr, thresholds = roc_curve(y_test, y_pred_prob_yes[:,1])
         plt.plot(fpr,tpr)
         plt.xlim([0.0, 1.0])
         plt.ylim([0.0, 1.0])
         plt.title('ROC curve for Heart disease classifier')
         plt.xlabel('False positive rate (1-Specificity)')
         plt.ylabel('True positive rate (Sensitivity)')
         plt.grid(True)
```



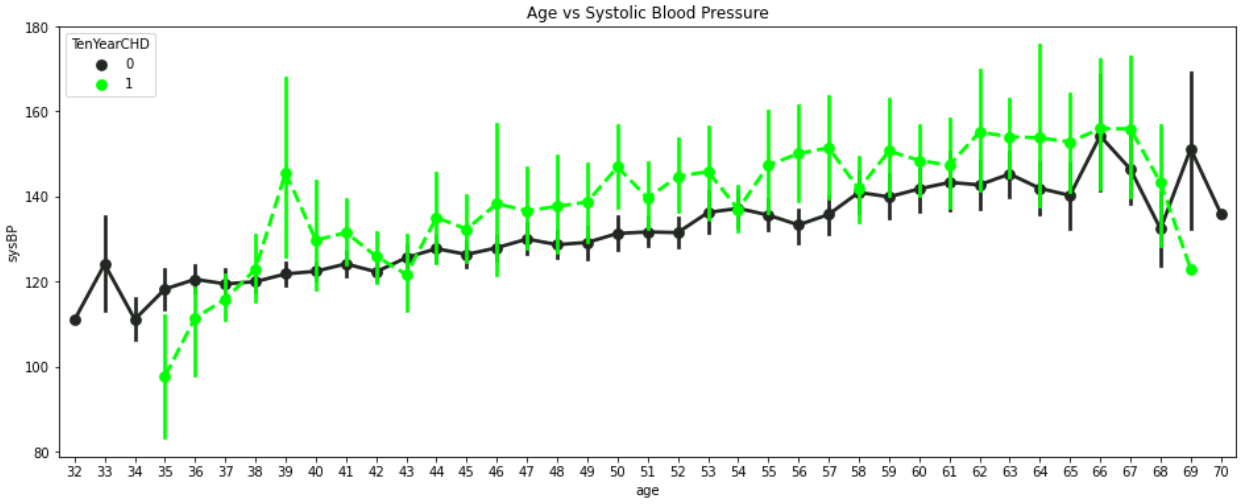

```
In [79]: #Sanduni -IT19123578
         sklearn.metrics.roc_auc_score(y_test,y_pred_prob_yes[:,1])

Out[79]: 0.7733827523596586
```

```
In [97]: #Sanduni
         from sklearn.linear_model import LogisticRegression
         from sklearn.tree import DecisionTreeClassifier
         from sklearn.ensemble import RandomForestClassifier
         from sklearn.ensemble import GradientBoostingClassifier
         from sklearn.ensemble import AdaBoostClassifier
         from sklearn.ensemble import ExtraTreesClassifier
         from sklearn.neighbors import KNeighborsClassifier
         from sklearn.svm import SVC
         from sklearn.naive_bayes import GaussianNB
         #from sklearn.model_selection import KFold
         from sklearn.metrics import accuracy_score

         classifiers=[['Logistic Regression :',LogisticRegression()],
               ['Decision Tree Classification :',DecisionTreeClassifier()],
               ['Gradient Boosting Classification :', GradientBoostingClassifier()],
               ['ada Boosting Classification :',AdaBoostClassifier()],
               ['K-Neighbors Classification :',KNeighborsClassifier()],
               ['Gaussian Naive Bayes :',GaussianNB()]]
         cla_pred=[]
         for name,model in classifiers:
             model=model
             model.fit(X_train,y_train)
             predictions = model.predict(X_test)
             cla_pred.append(accuracy_score(y_test,predictions))
             print(name,'{:.2f}%'.format(accuracy_score(y_test,predictions)*100))

         Logistic Regression : 84.63%
         Decision Tree Classification : 74.40%
         Gradient Boosting Classification : 84.63%
         ada Boosting Classification : 84.80%
         K-Neighbors Classification : 82.93%
         Gaussian Naive Bayes : 83.28%
```

Git Hub Commits

M.G.D.D.B. Ekanayaka IT19138732

I contributed to data visualization
Get the plot of Age vs Continuous Systolic Blood and Get the plot of Age vs Continues Heart Rate
Also used Matplotlib libraries such as Seaborn to visualize random visualizations. I created Logistic Regression model to predict the overall risk of Heart disease with the good predictions
I get feature selection for the data model
I use sklearn framework



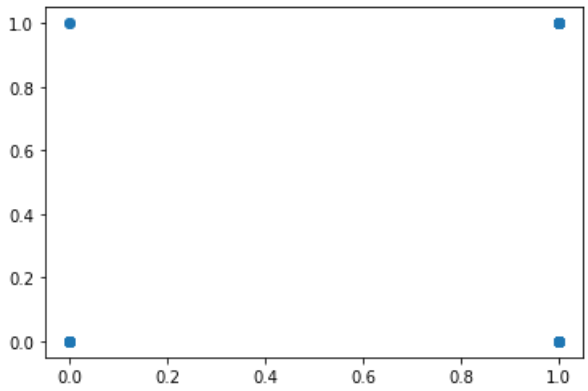
Logistic Model Evaluation

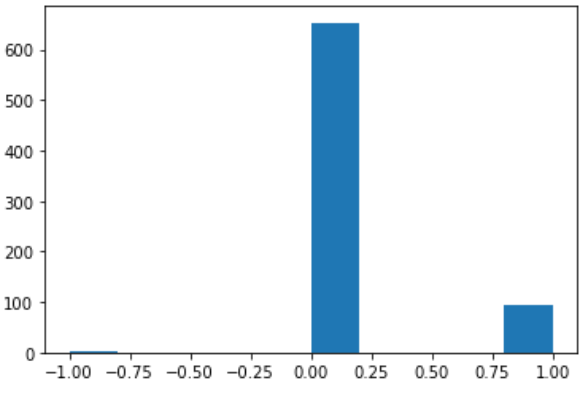
Logit Regression Results

Dep. Variable:	TenYearCHD	No. Observations:	3751
Model:	Logit	Df Residuals:	3744
Method:	MLE	Df Model:	6
Date:	Fri, 18 May 2018	Pseudo R-squ.:	0.1149
Time:	21:52:59	Log-Likelihood:	-1417.7
converged:	True	LL-Null:	-1601.7
		LLR p-value:	2.127e-76

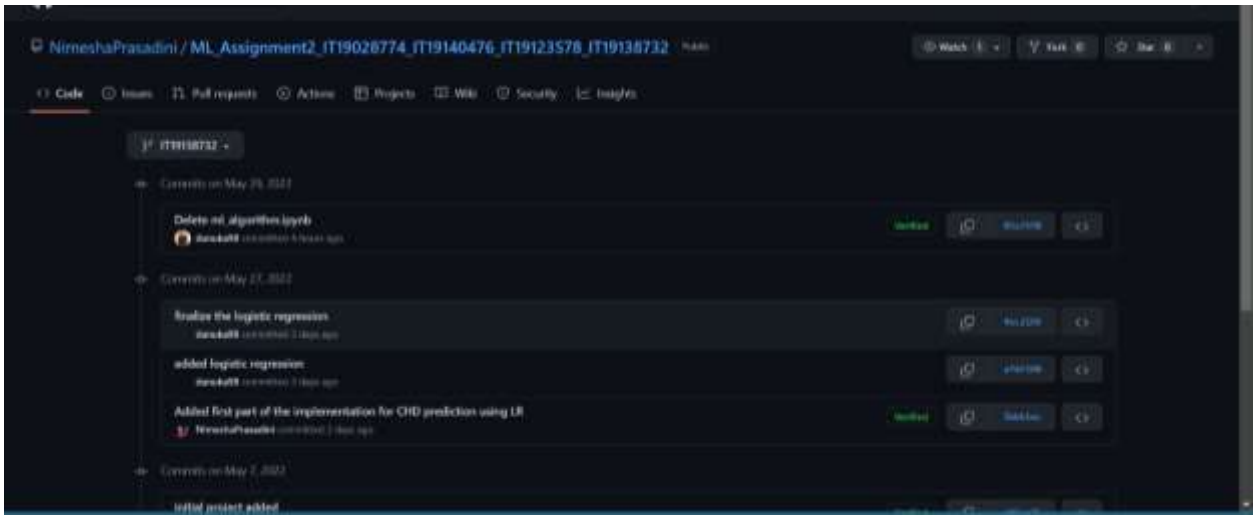
	coef	std err	z	P>|z|	[0.025	0.975]
const	-9.1264	0.469	-19.504	0.000	-10.043	-8.209
Sex_male	0.5815	0.105	5.524	0.000	0.375	0.788
age	0.0655	0.006	10.343	0.000	0.053	0.078
cigsPerDay	0.0197	0.004	4.905	0.000	0.012	0.028
totChol	0.0023	0.001	2.106	0.035	0.000	0.004
sysBP	0.0174	0.002	8.162	0.000	0.013	0.022
glucose	0.0076	0.002	4.574	0.000	0.004	0.011

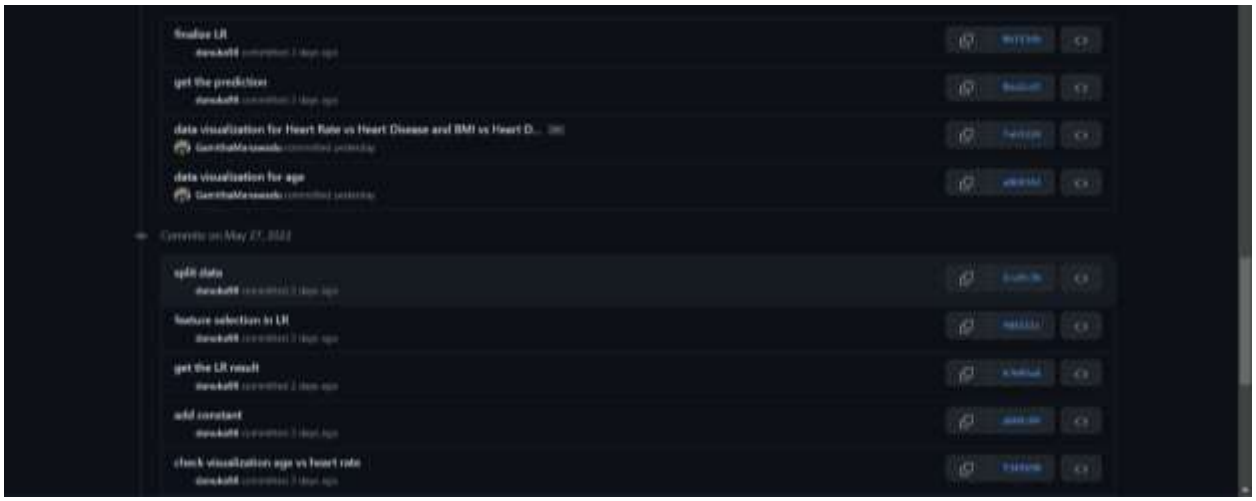
	CI 95%(2.5%)	CI 95%(97.5%)	Odds Ratio	pvalue
const	0.000044	0.000274	0.000109	0.000
sex_male	1.454877	2.198166	1.788313	0.000
age	1.054409	1.080897	1.067571	0.000
cigsPerDay	1.011730	1.028128	1.019896	0.000
totChol	1.000150	1.004386	1.002266	0.036
sysBP	1.013299	1.021791	1.017536	0.000
glucose	1.004343	1.010895	1.007614	0.000





Git hub


NimeshaPrasadini / ML_Assignment2_IT19028774_IT19140476_IT19123578_IT19138732
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Delete ml_algorithm.ipynb
finalize the logistic regression
added logistic regression
Added first part of the implementation for CHD prediction using LR
initial project added


finalize LR
get the prediction
data visualization for Heart Rate vs Heart Disease and BMI vs Heart D...
data visualization for age
split data
feature selection in LR
get the LR result
add constant
check visualization age vs heart rate

Code

```
#!/usr/bin/env python
# coding: utf-8

# In[ ]:

#Heart Disease Prediction using Logistic Regression
#The classification goal is to predict whether the patient has 10-year risk of future coronary heart disease (CHD)

# In[ ]:

import pandas as pd
import numpy as np
import statsmodels.api as sm
import scipy.stats as st
import matplotlib.pyplot as plt
import seaborn as sn
from sklearn.metrics import confusion_matrix
import matplotlib.mlab as mlab
import warnings
get_ipython().run_line_magic('matplotlib', 'inline')
warnings.filterwarnings("ignore")
sn.set_style("darkgrid")

# In[ ]:

#nimesha
# Loading Heart Data from framingham.csv
chd_df=pd.read_csv("framingham.csv")
#drop a column
chd_df.drop(['education'],axis=1,inplace=True)
chd_df.head()

# In[ ]:

#Rename 'male' column name
chd_df.rename(columns={'male':'sex_male'},inplace=True)

# In[ ]:


#Find missing values
chd_df.isnull().sum()
```

```
# In[ ]:

#Counting total no of rows with missing values
count=0
for i in chd_df.isnull().sum(axis=1):
    if i>0:
        count=count+1
print('Total number of rows with missing values =', count)
print('Percentage of rows with missing values in the dataset
=',round((count/len(chd_df.index))*100),'%')
print('Therefore, the missing values are eliminated.')

# In[ ]:

#dropping the missing values
chd_df.dropna(axis=0,inplace=True)


# In[ ]:

plt.figure(figsize=(10,8))
sn.heatmap(chd_df.corr(),annot=True,cmap='YlGnBu',fmt='.2f',linewidths=2)

# In[ ]:

#Exploratory Analysis by drawing histograms for CHD features
def draw_chd_histograms(dataframe, features, rows, cols):
    fig_chd=plt.figure(figsize=(20,20))
    for i, feature in enumerate(features):
        ax_chd=fig_chd.add_subplot(rows,cols,i+1)
        dataframe[feature].hist(bins=20,ax=ax_chd,facecolor='maroon')
        ax_chd.set_title(feature+" Visualization",color='navy')

    fig_chd.tight_layout()
    plt.show()
#Call the histogram function
draw_chd_histograms(chd_df,chd_df.columns,6,3)

# In[ ]:

#TenYearCHD feature values counting
chd_df.TenYearCHD.value_counts()

# In[ ]:

#Plot a graph for the TenYearCHD feature value data
```

```
sn.countplot(x='TenYearCHD',data=chd_df)

# In[ ]:

print('Therefore, there are',(chd_df.TenYearCHD == 1).sum(),'patients with risk of heart disease and',(chd_df.TenYearCHD == 0).sum(),'patents with no heart disease.')

# In[ ]:

# Plot graphs for all feature data in the dataframe
sn.pairplot(data=chd_df)

# In[ ]:

#Description of the all feature data in the dataframe
#count - no of non-empty values
#mean - average (mean) value
#std - standard deviation
#min - minimum value
#25% - 25% percentile
#50% - 50% percentile
#75% - 75% percentile
#max - maximum value
chd_df.describe()
#nimesha

# In[ ]:

sn.distplot(chd_df['age'],color='Yellow',hist_kws={'alpha':1,"linewidth": 2}, kde_kws={"color": "k", "lw": 3, "label": "KDE"})
#most of people are form age 40-50
#Gamitha

# In[ ]:

#Gamitha-IT19140476
fig,ax=plt.subplots(figsize=(24,6))

plt.subplot(1, 3, 1)
age_bins = [20,30,40,50,60,70,80]
chd_df['bin_age']=pd.cut(chd_df['age'], bins=age_bins)
g1=sn.countplot(x='bin_age',data=chd_df ,hue='TenYearCHD',palette='plasma',linewidth=3)
g1.set_title("Age vs Heart Disease")
#The number of people with heart disease are more from the age 41-55
#Also most of the people fear heart disease and go for a checkup from age 55-65 and dont have heart disease (Precautions)
```

```
plt.subplot(1, 3, 2)
chol_bins = [100,150,200,250,300,350,400,450]
chd_df['bin_chol']=pd.cut(chd_df['totChol'], bins=chol_bins)
g2=sn.countplot(x='bin_chol',data=chd_df,hue='TenYearCHD',palette='plasma',linewidth=3)
g2.set_title("Cholestoral vs Heart Disease")
#Most people get the heart disease with 200-250 cholestrol
#The others with cholestrol of above 250 tend to think they have heart disease but the rate of
heart disease falls

plt.subplot(1, 3, 3)
hr_bins = [40,60,80,100,120,140]
chd_df['bin_hr']=pd.cut(chd_df['heartRate'], bins=hr_bins)
g3=sn.countplot(x='bin_hr',data=chd_df,hue='TenYearCHD',palette='plasma',linewidth=3)
g3.set_title("Heart Rate vs Heart Disease")
#People who have thalach between 140-180 have a very high chance of getting the heart disease

# In[ ]:


#Sanduni -IT19123578
fig,ax=plt.subplots(figsize=(24,6))
plt.subplot(1, 3, 1)
dbp_bins = [40,60,80,100,120,140,160]
chd_df['bin_dbp']=pd.cut(chd_df['diaBP'], bins=dbp_bins)
g1=sn.countplot(x='bin_dbp',data=chd_df ,hue='TenYearCHD',palette='plasma',linewidth=3)
g1.set_title("Diastolic Blood Pressure vs Heart Disease")
#People who have Continuous Diastolic Blood Pressure level from 60-100 have a very high
chance of getting the heart disease.
plt.subplot(1, 3, 2)
sbp_bins = [80,100,120,140,160,180,200,220,240,260]
chd_df['bin_sbp']=pd.cut(chd_df['sysBP'], bins=sbp_bins)
g1=sn.countplot(x='bin_sbp',data=chd_df ,hue='TenYearCHD',palette='plasma',linewidth=3)
g1.set_title("Systolic Blood Pressure vs Heart Disease")
#People who have Continuous Systolic Blood Pressure level from 100-140 have a very high
chance of getting the heart disease.
plt.subplot(1, 3, 3)
cpd_bins = [0,10,20,30,40,50,60,70]
chd_df['bin_cpd']=pd.cut(chd_df['cigsPerDay'], bins=cpd_bins)
g1=sn.countplot(x='bin_cpd',data=chd_df ,hue='TenYearCHD',palette='plasma',linewidth=3)
g1.set_title("Average Number of Cigarettes Smoked Per Day vs Heart Disease")
#People who smokes 0-20 no of cigarettes per day ave a very high chance of getting the heart
disease.

# In[ ]:

#Gamitha-IT19140476
```

```
fig,ax=plt.subplots(figsize=(24,6))
plt.subplot(1, 2, 1)
bmi_bins = [0,10,20,30,40,50]
chd_df['bmi']=pd.cut(chd_df['BMI'], bins=bmi_bins)
x1=sn.countplot(x='bmi',data=chd_df,hue='TenYearCHD',palette='spring',linewidth=3)
x1.set_title('BMI vs Heart Disease')
#People with BMI value between 20-30 have highest chance of heart disease
```

In[ ]:

```
#Sanduni
fig,ax=plt.subplots(figsize=(16,6))
plt.subplot(121)
s1=sn.boxenplot(x='sex_male',y='age',hue='TenYearCHD',data=chd_df,palette='YlGn',linewidth
=3)
s1.set_title("Figure 1")
#Figure 1 says most of females having heart disease range from 40-70yrs and men from 40-60yrs
plt.subplot(122)
s2=sn.pointplot(x='sex_male',y='age',hue='TenYearCHD',data=chd_df,palette='autumn',capsize=
.2)
s2.set_title("Figure 2")
#Figure 2 says mean age for female with heart disease around 54yrs and for males around 51yrs
```

In[ ]:

```
fig,ax=plt.subplots(figsize=(16,6))
sn.pointplot(x='age',y='totChol',data=chd_df,color='Lime',hue='TenYearCHD',linestyles=["-", "-
-"])
plt.title('Age vs  Total Cholesterol Level ')
#People with heart disease tend to have higher
```

In[ ]:

```
fig,ax=plt.subplots(figsize=(16,6))
sn.pointplot(x='age',y='cigsPerDay',data=chd_df,color='Lime',hue='TenYearCHD',linestyles=["-
", "--"])
plt.title('Age vs  Average Number of Cigarettes Smoked Per Day ')
#People with heart disease tend to have higher
```

In[ ]:

```
fig,ax=plt.subplots(figsize=(16,6))
sn.pointplot(x='age',y='diaBP',data=chd_df,color='Lime',hue='TenYearCHD',linestyles=["-", "--
"])
plt.title('Age vs Diastolic Blood Pressure')
```

#People with heart disease tend to have higher

In[ ]:

```
#Danuka
fig,ax=plt.subplots(figsize=(16,6))
sn.pointplot(x='age',y='sysBP',data=chd_df,color='Lime',hue='TenYearCHD',linestyles=["-", "--
"])
plt.title('Age vs Systolic Blood Pressure')
#People with heart disease tend to have higher
```

In[ ]:

```
#Danuka
fig,ax=plt.subplots(figsize=(16,6))
sn.lineplot(y='heartRate',x='age',data=chd_df,hue="TenYearCHD",style='TenYearCHD',palette=
'magma',markers=True, dashes=False,err_style="bars", ci=68)
plt.title('Age vs Continuous Heart Rate')
```

In[ ]:

```
y=chd_df['TenYearCHD']
```

In[ ]:

```
chd_df=pd.get_dummies(chd_df,drop_first=True)
chd_df.head()
```

In[ ]:

```
X=chd_df.drop('TenYearCHD',axis=1)
X.head()
```

In[ ]:

```
X.head()
```

In[ ]:

```
from sklearn.model_selection import train_test_split
X_train1, X_test1, y_train1, y_test1 = train_test_split(X,y, test_size=0.20, random_state=101)
```

In[ ]:

#Sanduni

```
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
#from sklearn.model_selection import KFold
from sklearn.metrics import accuracy_score

classifiers=[['Logistic Regression :',LogisticRegression()],
      ['Decision Tree Classification :',DecisionTreeClassifier()],
      ['Gradient Boosting Classification :', GradientBoostingClassifier()],
      ['Ada Boosting Classification :',AdaBoostClassifier()],
      ['K-Neighbors Classification :',KNeighborsClassifier()],
      ['Gaussian Naive Bayes :',GaussianNB()]]
cla_pred=[]
for name,model in classifiers:
   model=model
   model.fit(X_train1,y_train1)
   predictions = model.predict(X_test1)
   cla_pred.append(accuracy_score(y_test1,predictions))
   print(name,"{:.2f}%".format(accuracy_score(y_test1,predictions)*100))

# In[ ]:

#Danuka's Contribution
from statsmodels.tools import add_constant as add_constant
heart_details_constant = add_constant(chd_df)
heart_details_constant.head()

# In[ ]:

st.chisqprob = lambda chisq, df: st.chi2.sf(chisq, df)
col=heart_details_constant.columns[:-1]
hd_model=sm.Logit(chd_df.TenYearCHD,heart_details_constant[col])
lr_result=hd_model.fit()
lr_result.summary()

# In[ ]:

def backward_elemination (hd_frame,dependent_var,column_list):
   while len(column_list)>0 :
      model=sm.Logit(dependent_var,hd_frame[column_list])
```

```
        result=model.fit(disp=0)
        largest_pvalue=round(result.pvalues,3).nlargest(1)
        if largest_pvalue[0]<(0.05):
            return result
            break
        else:
            column_list=column_list.drop(largest_pvalue.index)

result=backward_elemination(heart_details_constant,chd_df.TenYearCHD,col)

# In[ ]:

result.summary()

# In[ ]:

hd_params = np.exp(result.params)
conf_intervals = np.exp(result.conf_int())
conf_intervals['OR'] = hd_params
pvalue=round(result.pvalues,3)
conf_intervals['pvalue']=pvalue
conf_intervals.columns = ['CI 95%(2.5%)', 'CI 95%(97.5%)', 'Odds Ratio','pvalue']
print ((conf_intervals))


# In[ ]:


#Nimesha
import sklearn
new_hd_features=chd_df[['age','sex_male','cigsPerDay','totChol','sysBP','glucose','TenYearCHD'
]]
x=new_hd_features.iloc[:,:-1]
y=new_hd_features.iloc[:,-1]
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.20,random_state=5)
print(x_train,x_test,y_train,y_test)


# In[ ]:


#Gamitha-IT19140476
from sklearn.linear_model import LogisticRegression
logistic_reg=LogisticRegression()
logistic_reg.fit(x_train,y_train)
```

```
y_prediction=logistic_reg.predict(x_test)
print(y_prediction)


# In[ ]:


sklearn.metrics.accuracy_score(y_test,y_prediction)


# In[ ]:


#Sanduni -IT19123578
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,y_prediction)
conf_matrix=pd.DataFrame(data=cm,columns=['Predicted:0','Predicted:1'],index=['Actual:0','Act
ual:1'])
plt.figure(figsize = (8,5))
sn.heatmap(conf_matrix, annot=True,fmt='d',cmap="YlGnBu")


# In[ ]:


#Gamitha-IT19140476
TN=cm[0,0]
TP=cm[1,1]
FN=cm[1,0]
FP=cm[0,1]
sensitivity=TP/float(TP+FN)
specificity=TN/float(TN+FP)


# In[ ]:


print('The acuuracy of the model = TP+TN/(TP+TN+FP+FN) =
',(TP+TN)/float(TP+TN+FP+FN),'\n',

'The Missclassification = 1-Accuracy = ',1-((TP+TN)/float(TP+TN+FP+FN)),'\n',

'Sensitivity or True Positive Rate = TP/(TP+FN) = ',TP/float(TP+FN),'\n',

'Specificity or True Negative Rate = TN/(TN+FP) = ',TN/float(TN+FP),'\n',
```

```
'Positive Predictive value = TP/(TP+FP) = ',TP/float(TP+FP),'\n',

'Negative predictive Value = TN/(TN+FN) = ',TN/float(TN+FN),'\n',

'Positive Likelihood Ratio = Sensitivity/(1-Specificity) = ',sensitivity/(1-specificity),'\n',

'Negative likelihood Ratio = (1-Sensitivity)/Specificity = ',(1-sensitivity)/specificity)


# In[ ]:


#Sanduni
y_pred_prob=logistic_reg.predict_proba(x_test)[:,:]
y_pred_prob_df=pd.DataFrame(data=y_pred_prob, columns=['Prob of no heart disease (0)','Prob
of Heart Disease (1)'])
y_pred_prob_df.head()


# In[ ]:


#Gamitha-IT19140476
from sklearn.preprocessing import binarize
for i in range(1,5):
    cm2=0
    y_pred_prob_yes=logistic_reg.predict_proba(x_test)
    y_prediction2=binarize(y_pred_prob_yes,i/10)[:,1]
    cm2=confusion_matrix(y_test,y_prediction2)
    print ('With',i/10,'threshold the Confusion Matrix is ','\n',cm2,'\n',
           'with',cm2[0,0]+cm2[1,1],'correct predictions and',cm2[1,0],'Type II errors( False
Negatives)','\n\n',
          'Sensitivity: ',cm2[1,1]/(float(cm2[1,1]+cm2[1,0])),'Specificity:
',cm2[0,0]/(float(cm2[0,0]+cm2[0,1])),'\n\n\n')



# In[ ]:


#Sanduni -IT19123578
from sklearn.metrics import roc_curve
fpr, tpr, thresholds = roc_curve(y_test, y_pred_prob_yes[:,1])
plt.plot(fpr,tpr)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.0])
```

```
plt.title('ROC curve for Heart disease classifier')
plt.xlabel('False positive rate (1-Specificity)')
plt.ylabel('True positive rate (Sensitivity)')
plt.grid(True)


# In[ ]:


#Sanduni -IT19123578
sklearn.metrics.roc_auc_score(y_test,y_pred_prob_yes[:,1])


# In[ ]:
```