MITA Capstone Project

(22:544:688)

Framingham Heart study

By APURVA SUNIL SARODE

RUID: 195000707

Under the guidance of
PROF. MICHAIL XYNTARAKIS

CONTENT

1. INTRODUCTION

CVDs are the number 1 cause of death globally: more people die annually from CVDs than from any other cause. An estimated 17.9 million people died from CVDs in 2016, representing 31% of all global deaths. Of these deaths, 85% are due to heart attack and stroke. Over three-quarters of CVD deaths take place in low- and middle-income countries. Of the 17 million premature deaths (under 70 years of age) caused by non-communicable diseases in 2015, 82% were in low- and middle-income countries, and 37% were caused by CVDs. People with cardiovascular disease or who are at high cardiovascular risk (due to the presence of one or more risk factors such as hypertension, diabetes, hyperlipidemia, or already established disease) need early detection and management using counseling and medicines, as appropriate.

Our objective is to find the most relevant factor or risk, which results in heart disease. We will also predict whether the patient has a 10 – year risk of future CVDs or not by using different models with their accuracy. We will analyze the data and find the trends of various variables and threats involved in multiple graphs and visualizations.

This future subject demonstrates exploratory analysis to classify the matrix and explore the data using various methods and techniques to help find the best model of accuracy and to predict cardiovascular disease using an accuracy matrix.

"The dataset is available on National Heart, Lung, and Blood Institute (NHLBI), also publicly available on the Kaggle website is about an ongoing cardiovascular study on residents of the town of Framingham, Massachusetts."[2] The dataset provides the patients' information, which includes over 4,240 records and 16 attributes.

2. DATASET

The dataset contains 16 attributes. Each attribute is considered as a potential risk factor. It is distributed in demographic, behavioral, and medical risk factors.

Demographic:

- Gender: male or female (1 = male, 0 = female; Nominal)
- Age: age of the patient (Although the recorded ages have been truncated to whole numbers, the concept of age is continuous)
- Education: 1 = High School, 2 = High School or GED, 3 = College or Vocational School, 4 = College or University.

Behavioral:

- "CurrentSmoker: whether the patient is a current smoker (0 = nonsmoker, 1 = smoker; Nominal)
- CigsPerDay: The number of cigarettes that the person smoked on average in one day. (It is considered continuous as one can have any number of cigarettes, even half a cigarette.)"[2]

Medical history:

- BPMeds: whether the patient was on blood pressure medication (0 = Not on Blood Pressure medications, 1 = Is on Blood Pressure medications; Nominal)
- PrevalentStroke: whether the patient had previously had a stroke (0 = No, 1 = Yes; Nominal)
- PrevalentHyp: whether the patient was hypertensive (0 = No, 1 = Yes; Nominal)

- Diabetes: whether the patient had diabetes (0 = No, 1 = Yes; Nominal)

Medical current record:

- TotChol: total cholesterol level in mg/dL (Continuous)
- SysBP: systolic blood pressure in mmHg (Continuous)
- DiaBP: diastolic blood pressure in mmHg(Continuous)
- BMI: Body Mass Index calculated as Weight (kilogram) / Height (meter-squared)(Continuous)
- HeartRate: heart rate in Beats/Min (Ventricular)
- Glucose: glucose level in mg/dL (Continuous)
- Ten Year CHD: Coronary heart disease (CHD) diagnosed in last 10 year (0 = No; 1 = Yes)

3. PROBLEM STATEMENT

The vital part of the human body and disease-related to it is more visible to the heart. The term heart disease refers to the condition or cardiovascular disease of the heart and blood vessels. The goal of this project is to decide which cause or risk contributes to heart disease, and ten years into the future, the person will have heart disease or not. Using multiple machine learning models, based on the different outcomes, we can predict the outcome and can be used for further analysis.

4. METHODOLOGY (DATA PREPROCESSING)

Data preprocessing is a data mining technique that involves transforming raw data into an understandable format. Real-world data is often incomplete, inconsistent, lacking in certain behaviors or trends, and is likely to contain many errors.

Data preprocessing is a proven method of resolving such issues. Data preprocessing prepare raw data for further processing. Below is the process that is done to process the data.

4.1. LOADING DATA

```
# Read the data
heart_dataset = pd.read_csv("C:\\Users\\Apurva Sarode\\Desktop\\Capstone_Heart\\framingham.csv")

# Display the first few lines
heart_dataset.head()
```

	male	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	diabetes	totChol	sysBP	diaBP	BMI	heartRate
0	1	39	4.0	0	0.0	0.0	0	0	0	195.0	106.0	70.0	26.97	80.0
1	0	46	2.0	0	0.0	0.0	0	0	0	250.0	121.0	81.0	28.73	95.0
2	1	48	1.0	1	20.0	0.0	0	0	0	245.0	127.5	80.0	25.34	75.0
3	0	61	3.0	1	30.0	0.0	0	1	0	225.0	150.0	95.0	28.58	65.0
4	0	46	3.0	1	23.0	0.0	0	0	0	285.0	130.0	84.0	23.10	85.0

There is a total of 4240 rows and 16 columns.

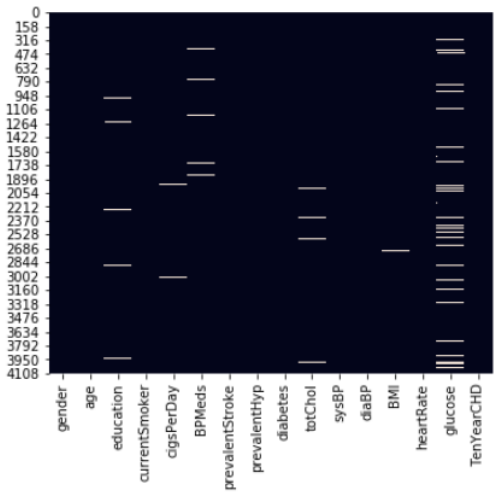
4.2. MISSING / NULL VALUE

```
# Formating the Columns
heart_dataset.rename(columns={'male':'gender'},inplace=True)

# Identify missing values
heart_dataset.isna().sum()
```

```
gender               0
age                  0
education          105
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
```

```
# Identify missing values
plt.figure(figsize=(6,5,))
sns.heatmap(heart_dataset.isnull(), cbar = False)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1df84d48f88>
```



```
# Identify count of different types of objects.
heart_dataset.get_dtype_counts()
```

```
C:\Users\Apurva Sarode\Anaconda3\lib\site-packages\ipykernel_launcher.py:2: FutureWarning: `get_dtype_counts` has been depre
cated and will be removed in a future version. For DataFrames use `.dtypes.value_counts()
```

```
float64    9
int64      7
dtype: int64
```

```
# Showing data types of variables
heart_dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4240 entries, 0 to 4239
Data columns (total 16 columns):
gender             4240 non-null int64
age                4240 non-null int64
education          4135 non-null float64
currentSmoker      4240 non-null int64
cigsPerDay         4211 non-null float64
BPMeds             4187 non-null float64
prevalentStroke    4240 non-null int64
prevalentHyp       4240 non-null int64
diabetes           4240 non-null int64
totChol            4190 non-null float64
sysBP              4240 non-null float64
diaBP              4240 non-null float64
BMI                4221 non-null float64
heartRate          4239 non-null float64
glucose            3852 non-null float64
TenYearCHD         4240 non-null int64
dtypes: float64(9), int64(7)
memory usage: 530.1 KB
```

As we can see, there are in total of 9 float variables and 7 int variables.

```
# Identify numeric and categorical columns
numeric_columns = ['age', 'cigsPerDay', 'totChol', 'sysBP', 'diaBP', 'BMI', 'heartRate', 'glucose']
categorical_columns = [c for c in heart_dataset.columns if c not in numeric_columns]
print(categorical_columns)
```

```
['gender', 'education', 'currentSmoker', 'BPMeds', 'prevalentStroke', 'prevalentHyp', 'diabetes', 'TenYearCHD']
```

So now, as we have distinguished the numeric and categorical variables. We can now replace the null or missing values of the numeric column with the mean and categorical column with mode. We won't delete the missing or Null values as the data is less and can also be replaced by simple methods.

```
# Fill missing values with Mean
heart_dataset["glucose"].fillna(round(heart_dataset["glucose"].mean()) , inplace = True)
heart_dataset["BMI"].fillna(round(heart_dataset["BMI"].mean()) , inplace = True)
heart_dataset["totChol"].fillna(round(heart_dataset["totChol"].mean()) , inplace = True)
heart_dataset["heartRate"].fillna(round(heart_dataset["heartRate"].mean()) , inplace = True)
heart_dataset["cigsPerDay"].fillna(round(heart_dataset["cigsPerDay"].mean()) , inplace = True)
```

```
# fill missing values with mode
heart_dataset['education'].fillna(heart_dataset['education'].mode()[0], inplace=True)
heart_dataset['BPMeds'].fillna(heart_dataset['BPMeds'].mode()[0], inplace=True)
```

```
df= heart_dataset.dropna().copy()
```

After replacing with the technique, we discussed above, here is the count of each variable showing where all NA and Null values have been replaced.

```
# Show total Data columns count
df.count()
```

```
gender             4240
age                4240
education          4240
currentSmoker      4240
cigsPerDay         4240
BPMeds             4240
prevalentStroke    4240
prevalentHyp       4240
diabetes           4240
totChol            4240
sysBP              4240
diaBP              4240
BMI                4240
heartRate          4240
glucose            4240
TenYearCHD         4240
dtype: int64
```

4.3. FINAL DATA

```
# Storing final and cleaned file
df.to_csv("Final_CVD_Data.csv")
df= pd.read_csv('Final_CVD_Data.csv', index_col=0) #removed unnamed column
```

5. EXPLORATORY ANALYSIS

Exploratory Data Analysis refers to the critical process of performing initial investigations on data so as to discover patterns, to spot anomalies, to test hypothesis and to check assumptions with the help of summary statistics and graphical representations.

5.1. DATA OVERVIEW

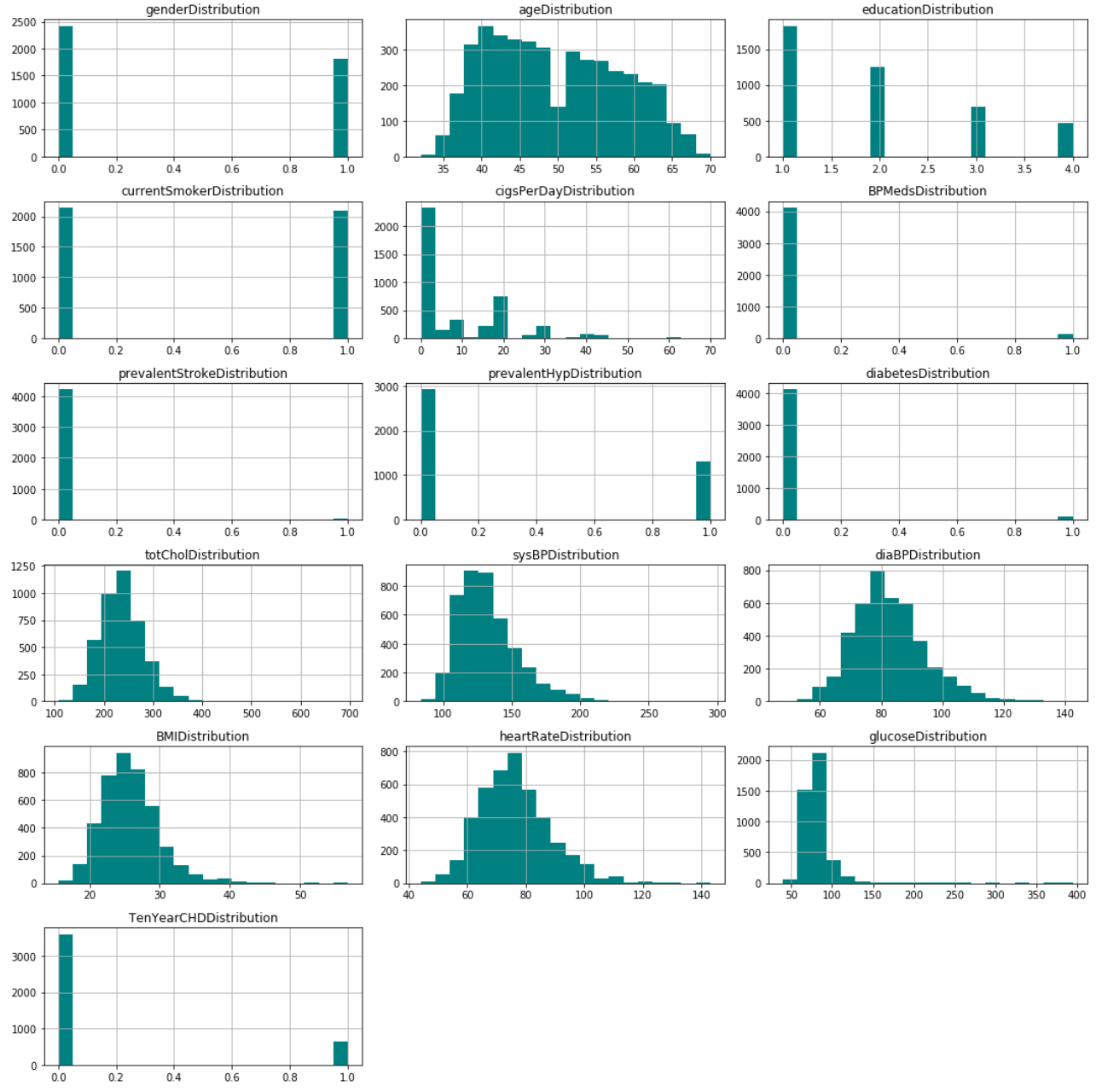
```
# View of descriptive statistics
df.describe()
```

	gender	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	diabetes	totChol
count	4240.000000	4240.000000	4240.000000	4240.000000	4240.000000	4240.000000	4240.000000	4240.000000	4240.000000	4240.000000
mean	0.429245	49.580189	1.955189	0.494104	9.005896	0.029245	0.005896	0.310613	0.025708	236.703066
std	0.495027	8.572942	1.018522	0.500024	11.881610	0.168513	0.076569	0.462799	0.158280	44.327533
min	0.000000	32.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	107.000000
25%	0.000000	42.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	206.000000
50%	0.000000	49.000000	2.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	234.000000
75%	1.000000	56.000000	3.000000	1.000000	20.000000	0.000000	0.000000	1.000000	0.000000	262.000000
max	1.000000	70.000000	4.000000	1.000000	70.000000	1.000000	1.000000	1.000000	1.000000	696.000000

We can see some necessary statistical details like count, mean, min/max, etc., for each variable in the data frame.

52. VISUALIZATION IN PYTHON

```
def draw_histograms(dataframe, features, rows, cols):
    fig=plt.figure(figsize=(15,15))
    for i, feature in enumerate(features):
        ax=fig.add_subplot(rows,cols,i+1)
        dataframe[feature].hist(bins=20,ax=ax,facecolor='teal')
        ax.set_title(feature+"Distribution",color='black')

    fig.tight_layout()
    plt.show()
draw_histograms(df,df.columns,6,3)
```



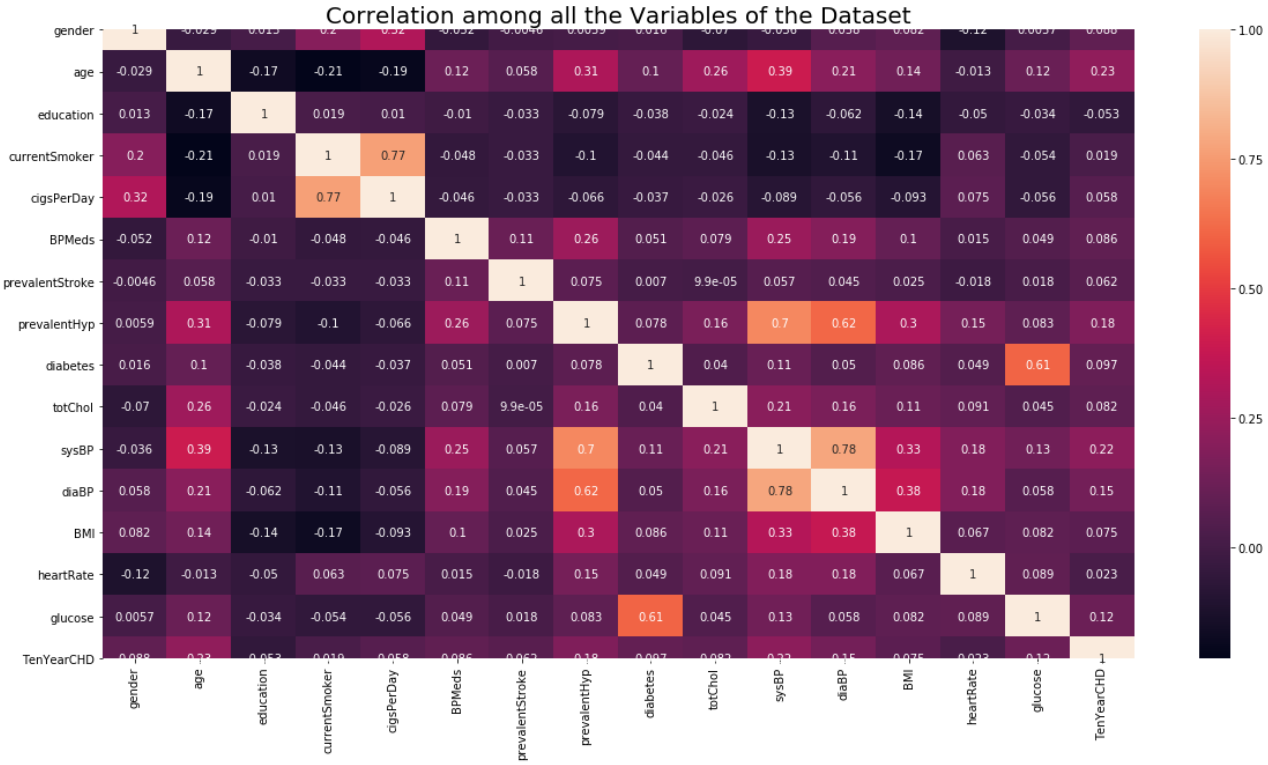
We can understand the following points:

- There are more females than males.

- The dataset has a greater number of peoples age 40-50.
- There is a large population who are educated till High School, and the number of populations is decreasing to a college education.
- The distribution for totChol, sysBP, diaBP, BMI, heartRate, and glucose are Right Screwed as the distribution is towards the right. This indicates that the mean is higher than the median.

```
#Checking relationship between variables
cor=df.corr()
plt.figure(figsize=(20,10), facecolor='w')
sns.heatmap(cor,xticklabels=cor.columns,yticklabels=cor.columns,annot=True)
plt.title("Correlation among all the Variables of the Dataset", size=20)
cor
```

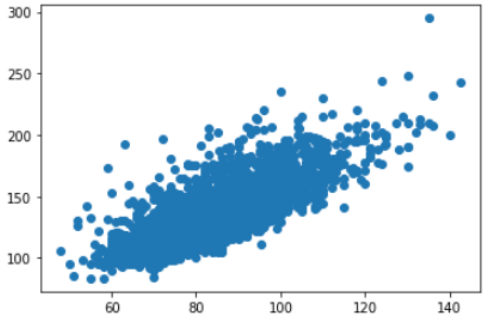
	gender	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	diabetes	totChol	sysBP
gender	1.000000	-0.029014	0.013361	0.197026	0.316023	-0.051544	-0.004550	0.005853	0.015693	-0.070111	-0.035879
age	-0.029014	1.000000	-0.165283	-0.213662	-0.192534	0.121011	0.057679	0.306799	0.101314	0.260709	0.394053
education	0.013361	-0.165283	1.000000	0.019399	0.010217	-0.010231	-0.032910	-0.078565	-0.038215	-0.024038	-0.126062
currentSmoker	0.197026	-0.213662	0.019399	1.000000	0.767051	-0.048348	-0.032980	-0.103710	-0.044285	-0.046191	-0.130281
cigsPerDay	0.316023	-0.192534	0.010217	0.767051	1.000000	-0.045683	-0.032710	-0.066444	-0.037085	-0.026165	-0.088523
BPMeds	-0.051544	0.121011	-0.010231	-0.048348	-0.045683	1.000000	0.114614	0.258580	0.051407	0.078789	0.251479
prevalentStroke	-0.004550	0.057679	-0.032910	-0.032980	-0.032710	0.114614	1.000000	0.074791	0.006955	0.000099	0.057000
prevalentHyp	0.005853	0.306799	-0.078565	-0.103710	-0.066444	0.258580	0.074791	1.000000	0.077752	0.162681	0.696656
diabetes	0.015693	0.101314	-0.038215	-0.044285	-0.037085	0.051407	0.006955	0.077752	1.000000	0.040158	0.111265
totChol	-0.070111	0.260709	-0.024038	-0.046191	-0.026165	0.078789	0.000099	0.162681	0.040158	1.000000	0.207445
sysBP	-0.035879	0.394053	-0.126062	-0.130281	-0.088523	0.251479	0.057000	0.696656	0.111265	0.207445	1.000000
diaBP	0.058199	0.205586	-0.062334	-0.107933	-0.056474	0.192254	0.045153	0.615840	0.050260	0.163424	0.783952
BMI	0.081631	0.135630	-0.139743	-0.167537	-0.092948	0.099710	0.025141	0.300625	0.086391	0.114965	0.325210
heartRate	-0.116911	-0.012835	-0.049582	0.062681	0.075258	0.015136	-0.017674	0.146780	0.048986	0.090693	0.182088
glucose	0.005679	0.116941	-0.034413	-0.054052	-0.056018	0.048873	0.018059	0.082750	0.605694	0.044706	0.134555



Correlation heatmap help to find out the correlation between each variable. Some of the examples are:

- dia BP and sysBP are highly correlated.
- sysBP and prevalentHyp are highly correlated.

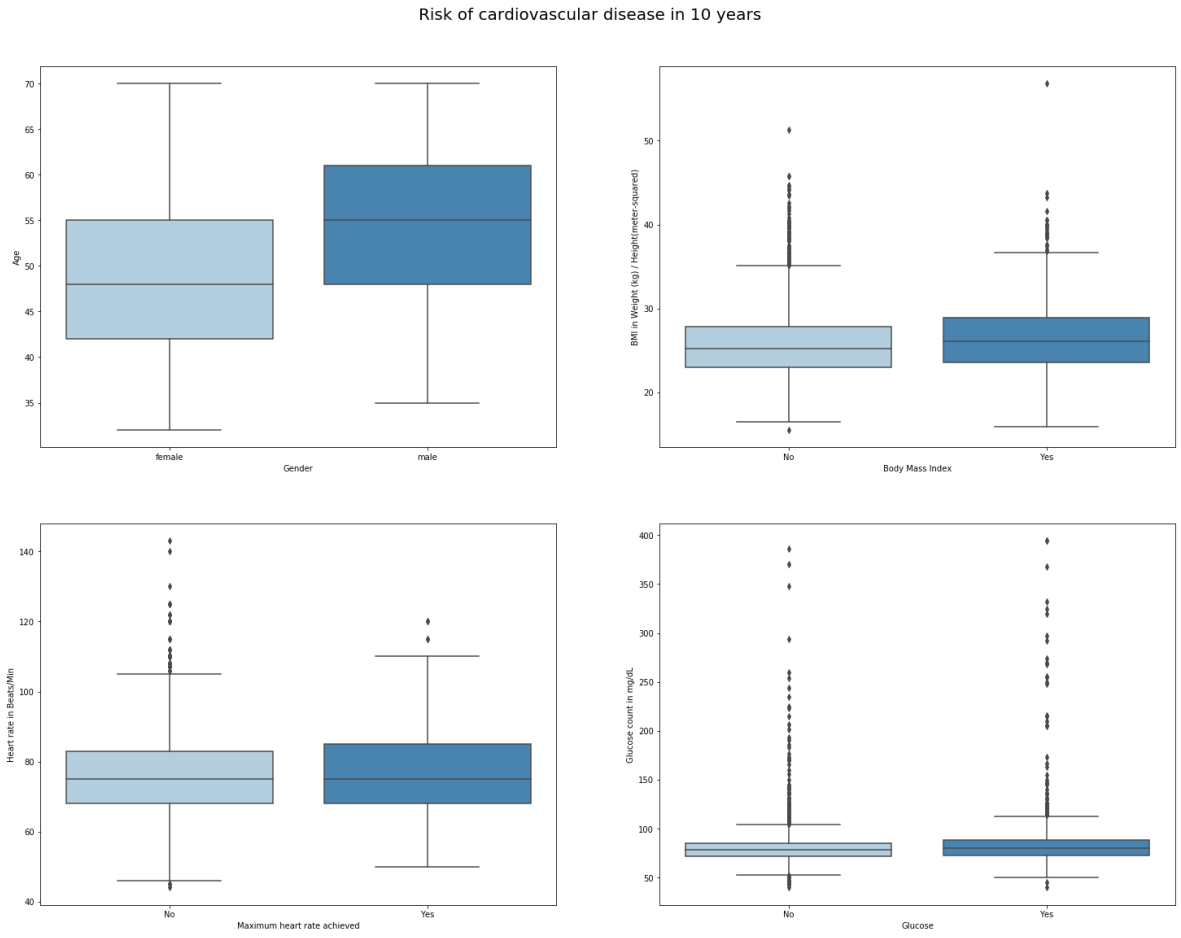
```
x = df['diaBP']
y = df['sysBP']
plt.scatter(x, y)
plt.show()
```



- Scatterplot between highly correlated field diaBP and sysBP with very less Outliers.

```
plt.figure(figsize = (25, 50))
plt.suptitle('Risk of cardiovascular disease in 10 years', y = 0.90, fontsize = 20)
gs = gridspec.GridSpec(5, 2)

plt.subplot(gs[0, 0])
sns.boxplot(df['TenYearCHD'].replace({0:'female', 1:'male'}), df['age'], palette = 'Blues')
plt.xlabel('Gender')
plt.ylabel('Age')

plt.subplot(gs[0, 1])
sns.boxplot(df['TenYearCHD'].replace({0:'No', 1:'Yes'}), df['BMI'], palette = 'Blues')
plt.xlabel('Body Mass Index')
plt.ylabel('BMI in Weight (kg) / Height(meter-squared)')

plt.subplot(gs[1, 0])
sns.boxplot(df['TenYearCHD'].replace({0:'No', 1:'Yes'}), df['heartRate'], palette = 'Blues')
plt.xlabel('Maximum heart rate achieved')
plt.ylabel('Heart rate in Beats/Min')


plt.subplot(gs[1, 1])
sns.boxplot(df['TenYearCHD'].replace({0:'No', 1:'Yes'}), df['glucose'], palette = 'Blues')
plt.xlabel('Glucose')
plt.ylabel('Glucose count in mg/dL')
```

```
Text(0, 0.5, 'Glucose count in mg/dL')
```

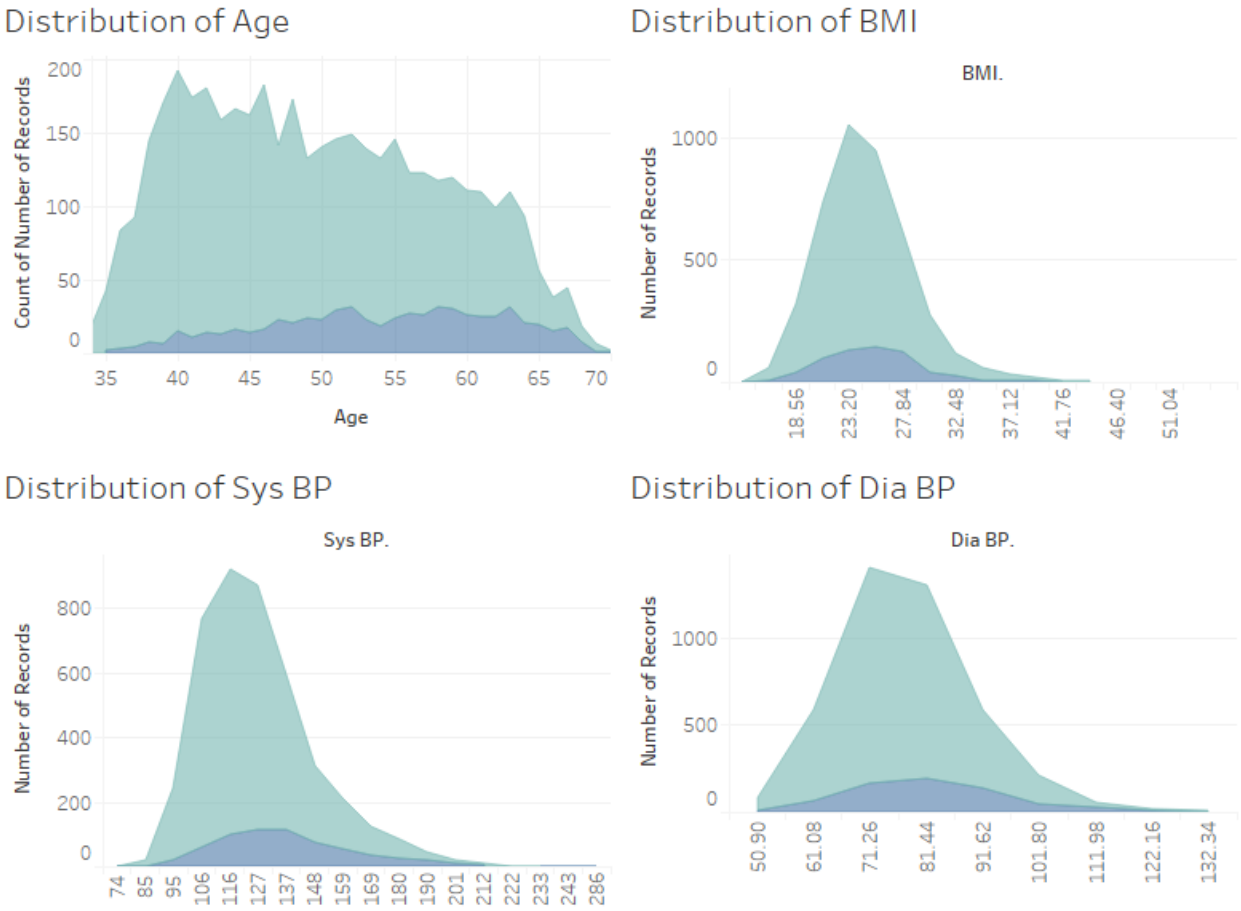


The above boxplot shows as follows:

- The male population is much older than the female population, and the third quartile of the female is equal to the median of males.
- The Body mass index and heart rate have a similar kind of data with multiple outliers.

- For glucose, the values are closed to mean; hence, there is minimal variance.

5.3. VISUALIZATION USING TABLEAU

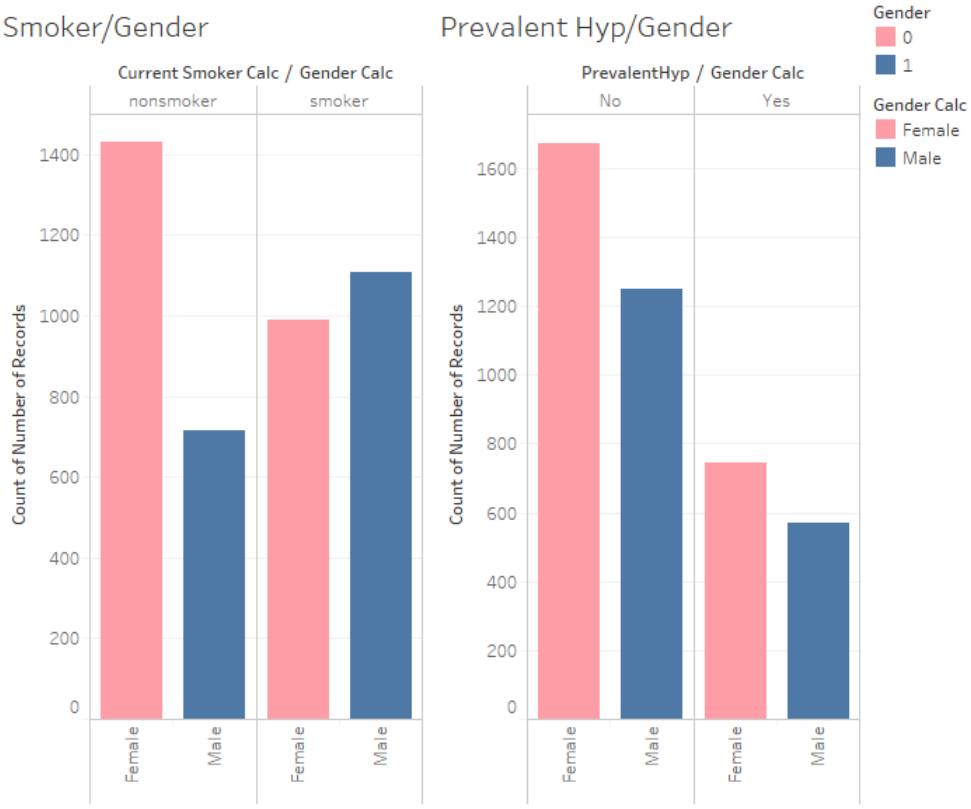


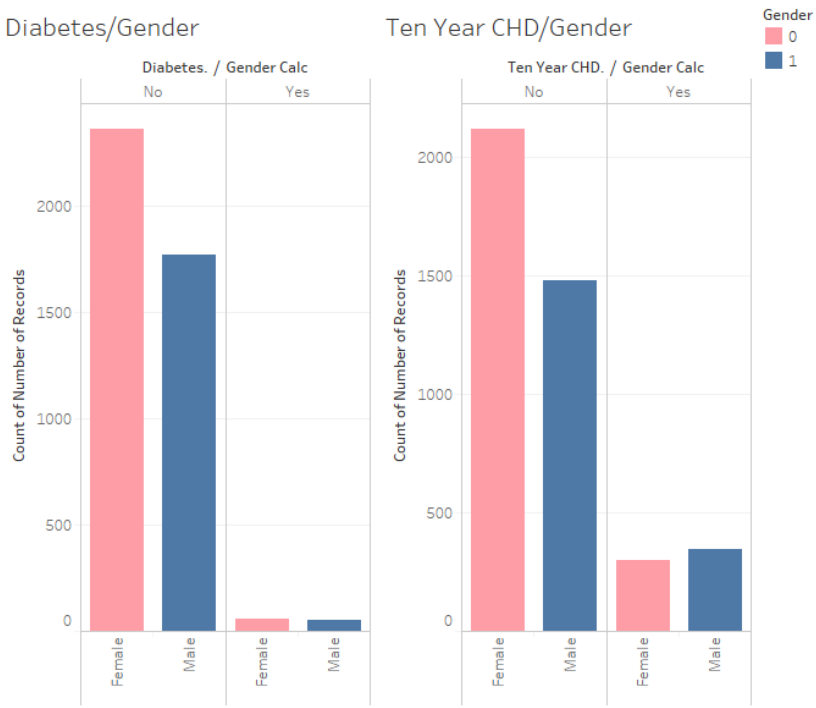
Here we can see the simple distribution count of variables Age, BMI, SysBP, and DiaBP.

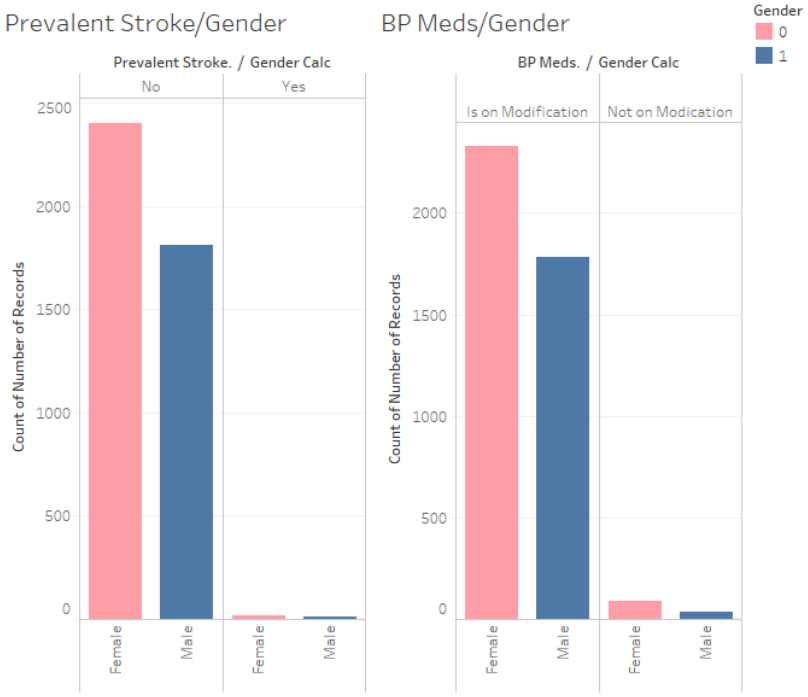
- The age is generally distributed at the range of 38 – 62 wrt Ten Year CHD.
- BMI is distributed at the range of 16 – 35 wrt Ten Year CHD
- SysBP and DiaBP are distributed at the range of 88 – 187 and 50 – 112, respectively.

This gives a broad idea of the distribution in the dataset of some of the essential variables. Also, this will help to get an overview of the dataset variables.

Below is some visualization wrt Gender showing the count of the variables.


Smoker/Gender
Current Smoker Calc / Gender Calc
nonsmoker
smoker
Count of Number of Records
0
200
400
600
800
1000
1200
1400
Female
Male
Female
Male
Prevalent Hyp/Gender
PrevalentHyp / Gender Calc
No
Yes
Count of Number of Records
0
200
400
600
800
1000
1200
1400
1600
Female
Male
Female
Male
Gender
0
1
Gender Calc
Female
Male


Diabetes/Gender
Diabetes. / Gender Calc
No
Yes
Count of Number of Records
0
500
1000
1500
2000
Female
Male
Female
Male
Ten Year CHD/Gender
Ten Year CHD. / Gender Calc
No
Yes
Count of Number of Records
0
500
1000
1500
2000
Female
Male
Female
Male
Gender
0
1



- We can observe that males were diagnosed with CVDs than females.
- Females are more prone to hypertension and blood pressure.
- Males smoked more than females, as shown in the graph.

6. EXPERIMENT (BUILDING MODELS FOR PREDICTION)

For predictive analytical modeling, numerous statistical and machine learning algorithms are available. After you have established the purpose of your model, you may pick your model. Herewith the patient data we have, we will be forecasting the incidence of cardiovascular diseases in the next ten years. Based on machine learning techniques, we will build multiple models and algorithms and focus on further analysis with the highest accurate model.

6.1. SPLITTING THE DATASET

Firstly, we will set up sci-kit-learn, which helps split the data into two, i.e., Test and Train. Splitting the data is very important as most of the part of the dataset is used to train the model, whereas the small portion is used to test the model which we have built. The general ratio is 80-20%. While splitting the data also, the overfitting and underfitting should be kept in mind.

```
# extract the target variable
X, y = df.iloc[:, :-1], df.iloc[:, -1]
print(X.shape)
print(y.shape)
```

```
(4240, 15)
(4240,)
```

```
#split the data in 80% as Training and 20% as Test dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=879)
print ("train_set_x shape: " + str(X_train.shape))
print ("train_set_y shape: " + str(y_train.shape))
print ("test_set_x shape: " + str(X_test.shape))
print ("test_set_y shape: " + str(y_test.shape))
```

```
train_set_x shape: (3392, 15)
train_set_y shape: (3392,)
test_set_x shape: (848, 15)
test_set_y shape: (848,)
```

```
# scale feature matrices
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```
def train_model(X_train, y_train, X_test, y_test, classifier, **kwargs):
    """     Fit the chosen model and print out the score.   """

    # instantiate model
    model = classifier(**kwargs)

    # train model
    model.fit(X_train,y_train)

    # check accuracy and print out the results
    fit_accuracy = model.score(X_train, y_train)
    test_accuracy = model.score(X_test, y_test)
    print(f"Train accuracy: {fit_accuracy:0.2%}")
    print(f"Test accuracy: {test_accuracy:0.2%}")
    return model
```

6.2. DECISION TREE

A Decision Tree algorithm for supervised machine learning has a predefined target variable that is often used in problems with classification. A decision tree is like a flow-chart, where every internal node is tested on an attribute, every branch represents the outcome of a trial, and every leaf (or terminal) node holds a class label. In a tree, the topmost node is the root node. Some of the trees in the Decision tree are precise than others; it is computationally infeasible to find the optimal tree.

The following algorithm for Decision tree induction: Tree Growth (E, F)

1. if stopping_cond(E,') = true then
2. leaf = createNode().
3. leaf.label: Classify(E).
4. return leaf.
5. else

6. root = createNode().
7. root.test_cond: find_best_split(E, F).
8. let V = {v|v is a possible outcome of root.test_cond }.
9. for each v ∈ V do
10. Ev = {e I root.test_cond(e) = v and e ∈ E}.
11. child = TreeGrowth(Ev, F).
12. add child as descendent of root and Iabel the edge (root ◊ child) as v.
13. end for 1
14. end if
15. return root.

Following the code used for using the Decision Tree algorithm:

```
# Decision Tree
model = train_model(X_train, y_train, X_test, y_test, DecisionTreeClassifier, random_state=1060)
```

```
Train accuracy: 100.00%
Test accuracy: 76.18%
```

We can see that the accuracy of models is 76.18%, and that's good to compare.

6.3. SUPPORT VECTOR MACHINES

SVM is a supervised machine learning algorithm that can be used for classification or regression problems. It uses a technique called the kernel trick to transform your data, and then based on these transformations, it finds an optimal boundary between the possible outputs.

Following the code used for using the Support Vector Machine:

```
#Support Vector Machines
model = train_model(X_train, y_train, X_test, y_test, SVC)
```

```
Train accuracy: 86.20%
Test accuracy: 85.02%
```

Accuracy of SVM is better than Decision tree, i.e., 85.02%

6.4. LOGISTIC REGRESSION

In statistics, logistic regression is a form of regression analysis used to predict an outcome from a collection of predictor or independent variables for a categorical dependent variable. The dependent variable is always binary in the logistic regression algorithm. It is mainly used for predicting and calculating the probability of success.

Following the code used for using the logistic regression algorithm:

```
# Logistic Regression
model = train_model(X_train, y_train, X_test, y_test, LogisticRegression)
```

```
Train accuracy: 85.64%
Test accuracy: 85.26%
```

```
C:\Users\Apurva Sarode\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:432: FutureWarning: Default solver will
be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
```

Accuracy for logistic regression, i.e., 85.26% is better than SVM

6.5. K-NEAREST NEIGHBORS (KNN)

The KNN algorithm is a method for classifying objects based on closest training examples in the feature space. It is also a type of instance-based learning where the function is only approximated locally, and all computation is delayed until classification. The KNN is the fundamental and most straightforward classification technique when there is little or no prior knowledge about distributing the data. The K in KNN refers to the number of nearest neighbors that the classifier will use to predict. It uses a majority voting scheme. It

Assigns data points based on whichever is in the majority. In case there is a tie between the classes, we may randomly choose one of them to classify the data point.

The algorithm is as follows

1: Let k be the number of nearest neighbors and D be the set of training examples.

2: for each test example z = (x', y') do

3: Compute d(x',x), the distance between z and every example, (x,y) ∈ D.

4: Select Dz ⊆ D, the set of k closest training examples to z.

5: y': argmax ∑v (xi ,yi) ∈ Dz I(v = yi)

6: end for

```
# KNN
model = train_model(X_train, y_train, X_test, y_test, KNeighborsClassifier)
```

```
Train accuracy: 86.38%
Test accuracy: 84.32%
```

The K nearest neighbor algorithm gave me an accuracy of 84.32%

6.6. RANDOM FORESTS

Random forest is a classifier that evolves from decision trees. It consists of many decision trees. To classify a new instance, each decision tree provides a classification for input data; a random forest collects the classifications and chooses the most voted prediction as a result. The input of each tree is sampled data from the original dataset. Besides, a subset of features is randomly selected from the optional features to grow the tree at each node. Each tree is grown without pruning. Essentially, a random forest enables many weak or weakly-correlated classifiers to form a robust classifier.

```
# Random Forests
model = train_model(X_train, y_train, X_test, y_test, RandomForestClassifier, random_state=1060)
```

```
Train accuracy: 97.73%
Test accuracy: 84.91%
```

```
C:\Users\Apurva Sarode\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:245: FutureWarning: The default value of n_est
imators will change from 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
```

The Random Forest algorithm gave me an accuracy of 84.91%

6.7. GAUSSIAN NAIVE BAYES

Naive Bayes classifiers are built on Bayesian classification methods. These rely on Bayes's theorem, an equation describing the relationship of conditional probabilities of statistical quantities.

```
#Gaussian Naive Bayes
model = train_model(X_train, y_train, X_test, y_test, GaussianNB)
```

```
Train accuracy: 82.75%
Test accuracy: 82.19%
```

The Gaussian Naïve Bayes algorithm gave me an accuracy of 82.19%

6.8. COMPARE DIFFERENT MACHINE LEARNING ALGORITHMS

```
# initialize an empty list
accuracy = []

# list of algorithms names
classifiers = ['KNN', 'Decision Trees', 'Logistic Regression', 'Naive Bayes', 'SVM', 'Random Forests']

# list of algorithms with parameters
models = [KNeighborsClassifier(n_neighbors=5), DecisionTreeClassifier(max_depth=6, random_state=1060), LogisticRegression(),
        GaussianNB(), SVC(C=0.05, kernel='linear'), RandomForestClassifier(n_estimators=110, random_state=1060)]

# loop through algorithms and append the score into the list
for i in models:
    model = i
    model.fit(X_train, y_train)
    score = model.score(X_test, y_test)
    accuracy.append(score)
```

```
C:\Users\Apurva Sarode\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:432: FutureWarning: Default solver will
be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
```

```
# create a dataframe from accuracy results
summary = pd.DataFrame({'accuracy':accuracy}, index=classifiers)
summary
```

	accuracy
KNN	0.843160
Decision Trees	0.836085
Logistic Regression	0.852594
Naive Bayes	0.821934
SVM	0.847877
Random Forests	0.854953

We can have a bright look that Logistic Regression has the highest accuracy, i.e., 85.25%, which is more than the rest of the models we created. Hence, we can use Logistic Regression for further analysis of our project.

7. RELATED WORK

7.1 PREDICTION OF CVD's USING LOGISTIC REGRESSION

As we got to know that Logistic regression only has two possible outcomes. It is mostly used for prediction and calculating the probability of the results. To explain the relation between one dependent variable and one or more independent variables, logistic regression is used.

Here, we will try to take TenYearCHD as the dependent variable and comparing it with the rest of the column.

```
st.chisqprob = lambda chisq, df: st.chi2.sf(chisq, df)
cols=df.columns[:-1]
model=sm.Logit(df.TenYearCHD,df[cols])
result=model.fit()
result.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.397430
         Iterations 6
```

Logit Regression Results

Dep. Variable:	TenYearCHD	**No. Observations:**	4240
Model:	Logit	**Df Residuals:**	4225
Method:	MLE	**Df Model:**	14
Date:	Thu, 21 Nov 2019	**Pseudo R-squ.:**	0.06699
Time:	01:20:31	**Log-Likelihood:**	-1685.1
converged:	True	**LL-Null:**	-1806.1
		LLR p-value:	1.294e-43

	coef	std err	z	P>|z|	[0.025	0.975]
gender	0.3737	0.097	3.853	0.000	0.184	0.564
age	0.0285	0.005	5.221	0.000	0.018	0.039
education	-0.1618	0.044	-3.637	0.000	-0.249	-0.075
currentSmoker	-0.2528	0.141	-1.796	0.073	-0.529	0.023
cigsPerDay	0.0229	0.006	4.036	0.000	0.012	0.034
BPMeds	0.4103	0.217	1.895	0.058	-0.014	0.835
prevalentStroke	0.8768	0.436	2.009	0.045	0.021	1.732
prevalentHyp	0.9059	0.116	7.837	0.000	0.679	1.132
diabetes	0.8511	0.277	3.074	0.002	0.309	1.394
totChol	-0.0013	0.001	-1.317	0.188	-0.003	0.001
sysBP	0.0115	0.004	3.249	0.001	0.005	0.018
diaBP	-0.0244	0.006	-4.288	0.000	-0.036	-0.013
BMI	-0.0504	0.011	-4.416	0.000	-0.073	-0.028
heartRate	-0.0199	0.004	-5.422	0.000	-0.027	-0.013
glucose	0.0014	0.002	0.709	0.478	-0.003	0.005

The above result indicates the P-value of all the dependent variable variables i.e. TenyearCHD, indicating a low statistically significant relationship with the probability of heart disease. Thus we can eliminate the variables with a P-value greater than 0.05 one by one by processing the removal until all the variables with a P-value greater than 0.05 have been removed.

```
def back_feature_elem (data_frame,dep_var,col_list):

    while len(col_list)>0 :
        model=sm.Logit(dep_var,data_frame[col_list])
        result=model.fit(disp=0)
        largest_pvalue=round(result.pvalues,3).nlargest(1)
        if largest_pvalue[0]<(0.05):
            return result
            break
        else:
            col_list=col_list.drop(largest_pvalue.index)

result=back_feature_elem(df,df.TenYearCHD,cols)

result.summary()
```

Logit Regression Results

Dep. Variable:	TenYearCHD	**No. Observations:**	4240
Model:	Logit	**Df Residuals:**	4229
Method:	MLE	**Df Model:**	10
Date:	Thu, 21 Nov 2019	**Pseudo R-squ.:**	0.06452
Time:	01:20:46	**Log-Likelihood:**	-1689.6
converged:	True	**LL-Null:**	-1806.1
		LLR p-value:	1.969e-44

	coef	std err	z	P>|z|	[0.025	0.975]
gender	0.3777	0.096	3.934	0.000	0.190	0.566
age	0.0267	0.005	5.130	0.000	0.016	0.037
education	-0.1676	0.044	-3.801	0.000	-0.254	-0.081
cigsPerDay	0.0149	0.004	3.888	0.000	0.007	0.022
prevalentStroke	0.9785	0.435	2.249	0.025	0.126	1.831
prevalentHyp	0.9564	0.112	8.517	0.000	0.736	1.176
diabetes	0.9918	0.218	4.554	0.000	0.565	1.419
sysBP	0.0119	0.003	3.409	0.001	0.005	0.019
diaBP	-0.0257	0.006	-4.526	0.000	-0.037	-0.015
BMI	-0.0504	0.011	-4.493	0.000	-0.072	-0.028
heartRate	-0.0215	0.004	-6.070	0.000	-0.028	-0.015

As we can see above, all the variable now has a P value less than 0.05.

12 RATIO, CONFIDENCE INTERVALS AND P-VALUES

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

```
                 CI 95%(2.5%)  CI 95%(97.5%)  Odds Ratio  pvalue
gender               1.208651       1.761038    1.458931   0.000
age                  1.016630       1.037574    1.027048   0.000
education            0.775721       0.922059    0.845731   0.000
cigsPerDay           1.007426       1.022696    1.015032   0.000
prevalentStroke      1.134040       6.242023    2.660583   0.025
prevalentHyp         2.088175       3.242908    2.602261   0.000
diabetes             1.759391       4.131389    2.696058   0.000
sysBP                1.005082       1.018964    1.011999   0.001
diaBP                0.963901       0.985558    0.974669   0.000
BMI                  0.930186       0.971992    0.950859   0.000
heartRate            0.971982       0.985561    0.978748   0.000
```

Following are the points that can be inferred from the above table:

- Odds of getting diagnosed in Ten years for male over the female is 1.4589, i.e., the odds for the male is 45.89% higher than the female.
- For age, we can say that there will be a 2.70% increase in the odds of getting diagnosed.
- The additional cigarette will give rise to 1.5% odd of CHD
- For sysBP, there is 1.19% odd for CHD

13 MODEL EVALUATION - CONFUSION MATRIX

One of the significant items often used in machine learning to explain the classification model's efficiency is a confusion matrix. It is also known as an error matrix. Let’s see the confusion matrix for our model.

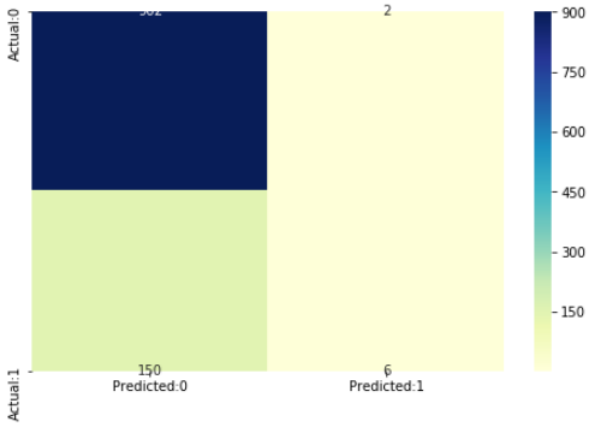
```
import sklearn
new_features=df[['age','gender','cigsPerDay','sysBP','glucose','TenYearCHD']]
x=new_features.iloc[:,:-1]
y=new_features.iloc[:,-1]
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.25,random_state=1060)
```

```
logreg=LogisticRegression()
logreg.fit(x_train,y_train)
y_pred=logreg.predict(x_test)
```

```
C:\Users\Apurva Sarode\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:432: FutureWarning: Default solver will
be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
```

```
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,y_pred)
conf_matrix=pd.DataFrame(data=cm,columns=['Predicted:0','Predicted:1'],index=['Actual:0','Actual:1'])
plt.figure(figsize = (8,5))
sns.heatmap(conf_matrix, annot=True,fmt='d',cmap="YlGnBu")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x22f9a26e848>
```



Our confusion matrix shows:

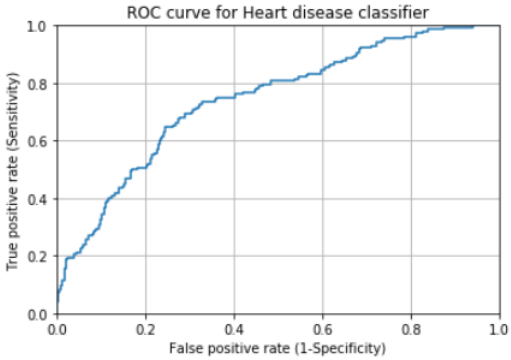
Correct Prediction : 902 + 6 = 908

Incorrect Prediction: 150 + 2 = 152

7.4 RECEIVER OPERATING CHARACTERISTIC (ROC) CURVE

"ROC curves are typically used in binary classification to study the output of a classifier."[8] It typically features a true positive rate on the Y-axis and a false positive rate on the X-axis. "It means that the top left corner of the plot is the "ideal" point - a false positive rate of zero, and a true positive rate of one."[8] It is not very realistic, but it does mean that a larger area under the curve (AUC) is usually better.

```
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)
```



The optimum position for the roc curve is towards the top left corner, where the specificity and sensitivity are at optimum levels.

8. CONCLUSION

We concluded the following things:

- The logistic model was the best model with a predicting value of 86%. The model is more specific than sensitive.
- Men are more likely to have heart disease compared to women.
- An increase in Age, Cigarettes per day, and Systolic Blood Pressure would also increase the probability of having CHD.
- The attributes which were eliminated having the P value less than 0.05 and thus suggested a significant role in the prediction of CHD.
- We could not see any significant change in the odds of CHD with respect to Total cholesterol. This could be due to the presence of 'good cholesterol(HDL) in the total cholesterol reading. Also, Glucose too causes a very negligible change in odds.
- The ROC curve is about 74%, which is much satisfactory. Also, the overall model can be improved by getting a greater number of data.

9. REFERENCE

1. Cardiovascular Disease. (n.d.). Retrieved from https://www.who.int/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds)
2. Singh, N. (2019, September 23). LOGISTIC REGRESSION. Retrieved from https://medium.com/@lucky16785/logistic-regression-5ad83b0913ed.
3. By. (2019, November 6). 2022 CRM: Cardiac Rhythm Management Market Share, Size, Trends and Forecasts Report by TBRC. Retrieved from https://www.marketwatch.com/press-release/2022-crm-cardiac-rhythm-management-market-share-size-trends-and-forecasts-report-by-tbrc-2019-11-06.
4. Data Mining Techniques in Healthcare: A Case Study from http://www.ijcstjournal.org/volume-5/issue-6/IJCST-V5I6P11.pdf
5. Data Preprocessing restored from https://www.techopedia.com/definition/14650/data-preprocessing
6. Exploratory Data Analysis retrieved from https://towardsdatascience.com/exploratory-data-analysis-8fc1cb20fd15
7. FreeCodeCamp.org. (2017, July 31). The Hitchhiker's Guide to Machine Learning in Python. Retrieved from https://medium.freecodecamp.org/the-hitchhikers-guide-to-machine-learning-algorithms-in-python-bfad66adb378.
8. Support Vector Machine Definition from https://medium.com/datadriveninvestor/support-vector-machine-svm-algorithm-in-a-fun-easy-way-fc23a008c22
9. Receiver Operating Characteristic (ROC)¶. (n.d.). Retrieved from https://scikit-learn.org/stable/auto_examples/model_selection/plot_roc.html