A
Mini Skill Based Project Report
On
Machine learning and Optimization (280404)

In fulfilment of the requirement for the award of the degree

SUBMITTED BY

Name of student: Rudraksh Saraf
Enrollment No: 0901AM211046
IV semester
Artificial Intelligence and Machine Learning

SUBMITTED TO

Dr. Vibha Tiwari

Department Of Information Technology

Madhav Institute of Technology and Science, Gwalior
(A Govt. Aided UGC Autonomous & NAAC Accredited Institute Affiliated to RGPV, Bhopal)

Session: Jan-June 2023

DECLARATION

I hereby declare that the mini skill-based project for the course Machine learning and optimization (280404) is being submitted in the partial fulfilment of the requirement for the award of **Bachelor of Technology** in Artificial Intelligence and Machine Learning**.**

All the information in this document has been obtained and presented in accordance with academic rule and ethical conduct.

Date:

Place: Gwalior

Name: Rudraksh Saraf
Enrolment No. 0901AM211046

ACKNOWLEDGEMENT

I am deeply indebted to all those who have significantly contributed towards the successful completion of this project. At the outset, I would like to express my sincere gratitude to our project supervisor Dr. Vibha Tiwari for her invaluable guidance, encouragement, and support throughout the project. The insightful feedback and constructive criticism has been instrumental in shaping the project's outcome. I would also like to extend my sincere thanks to our institute and department for providing us with the necessary resources and infrastructure required to complete this project. I would like to acknowledge the contributions of our team members and peers who have collaborated with us throughout the project. Their dedication, teamwork, and expertise have been vital in achieving our project goals. Finally, I would like to appreciate the efforts of the open-source community and researchers who have developed the necessary tools and technologies used in this project, particularly in the field of machine learning. Thank you all again for your outstanding support and encouragement throughout the project execution.

Project Title: Random Forest

To build an application to classify the patient to be healthy or suffering from cardiovascular disease based on the given attributes: such as

- age,
- gender,
- systolic blood pressure,
- diastolic blood pressure etc.

Macro Skills: EDA, Feature Selection, Performance Metrics

Micro Skills: Plot a suitable graph for analysing given dataset

References used in this project:

1. SCIKIT Learn Library Documentation
2. 20 models for Cardiovascular Disease prediction | Kaggle
3. Heart Attack: EDA + CORR + ML + ML metrics | Kaggle
4. Ensemble Learning Scikitlearn.org
5. Visual Data Representation using Seaborn
6. How to Find Outliers | 4 Ways Scribbr

Introduction

Problem Formulation:

The problem formulation of the ML project to classify the patient to be healthy or suffering from cardiovascular disease based on the given attributes.

The goal is to minimize false negatives (predicting a person is not suffering from cardiovascular based diseases when they actually are) while also minimizing false positives (predicting a person is suffering from cardiovascular disease when they are actually not).

The model should be trained and evaluated on a dataset of labelled examples, and feature engineering techniques may be used to extract relevant information from the input data. Ultimately, the goal is to develop a model that can be used to predict cardiovascular based diseases risk in new individuals based on their personal characteristics and medical history.

Conceptual Background of the Domain Problem:

The domain problem of Cardiovascular Diseases(CVD) is a complex and multifaceted issue that involves various medical, environmental, and lifestyle factors. It is a leading cause of death and disability worldwide, with an estimated 17.9 million deaths attributed to CVDs each year.

The development of CVDs involves the interplay of various risk factors, including high blood pressure, high cholesterol, smoking, obesity, diabetes, and physical inactivity. These risk factors can lead to the accumulation of plaque in the arteries, which can result in atherosclerosis, coronary heart disease, stroke, and other cardiovascular complications.

The management of CVDs typically involves a combination of lifestyle modifications, medications, and procedures. However, early detection and accurate risk prediction are critical for effective prevention and treatment.

Machine learning techniques, such as Random Forest, have been applied in the domain of CVDs to analyse large datasets and identify patterns and predictive models that can aid in risk assessment and personalized treatment. Random Forest is a decision tree-based algorithm that is particularly useful for handling large and complex datasets with high-dimensional features.

By leveraging the power of Random Forest, researchers and healthcare professionals can better understand the complex interactions between various risk factors and their impact on cardiovascular health. This can lead to more accurate risk prediction, earlier detection, and more personalized and effective treatment for CVDs**.**

The conceptual background of a machine learning project for predicting whether a person is suffering from cardiovascular disease or not involves several key steps. These include:

1. Data collection: The first step in any machine learning project is to gather relevant data for the problem at hand. For the problem of predicting cardiovascular disease, this might involve collecting medical records, demographic information, lifestyle factors, and other relevant data points for a large number of individuals.
2. Data pre-processing: Once the data has been collected, it will need to be cleaned and pre-processed to prepare it for analysis. This might involve tasks such as removing missing values, removing outliers and encoding categorical variables.
3. Feature selection: Next, the most important features or variables that are relevant to the prediction of cardiovascular disease will need to be identified. This step is critical for reducing the dimensionality of the dataset and improving the accuracy of the model.
4. Model selection: After selecting the relevant features, the appropriate machine learning algorithm will need to be chosen. For the problem of predicting cardiovascular disease, a popular algorithm is Random Forest, which is a decision tree-based algorithm that can handle large and complex datasets.
5. Model training: Once the algorithm has been selected, the data can be used to train the model. During this step, the algorithm will learn the patterns and relationships between the features and the outcome variable (whether a person has cardiovascular disease or not).
6. Model evaluation: After the model has been trained, it will need to be evaluated to determine its accuracy and performance. This might involve using techniques such as cross-validation and calculating metrics such as precision, recall, and F1 score.
7. Model deployment: Once the model has been evaluated and deemed accurate and reliable, it can be deployed and used for prediction on new

data. This might involve creating a web application or integrating the model into an existing healthcare system.

Overall, the success of a machine learning project for predicting cardiovascular disease will depend on the quality of the data, the choice of algorithm, and the accuracy of the model evaluation. By following these key steps, researchers and healthcare professionals can develop effective tools for early detection and personalized treatment of cardiovascular disease.

Review of Literature

The following is a brief review of some of the relevant studies and literature in this area:

1. "Machine learning techniques in cardiovascular disease prediction" by Shameer et al. (2018): This review article discusses the various machine learning techniques that have been used for cardiovascular disease prediction, including logistic regression, decision trees, random forests, and neural networks. The authors also highlight the challenges and limitations of using machine learning in this domain, such as the need for large and high-quality datasets.
2. "Deep learning for cardiac imaging: A review" by Bai et al. (2018): This review article focuses on the use of deep learning techniques, such as convolutional neural networks (CNNs), for analysing medical images in the context of cardiovascular disease. The authors provide an overview of the various CNN architectures that have been used for cardiac imaging, as well as their applications in diagnosis and prognosis.
3. "Machine learning in cardiovascular medicine: Are we there yet?" by Musunuru et al. (2020): This review article examines the current state of machine learning in cardiovascular medicine and highlights the potential for these techniques to improve patient outcomes. The authors discuss the challenges and limitations of using machine learning in this domain, as well as the need for more rigorous evaluation and validation of these models.

Overall, these studies and others in the literature demonstrate the potential for machine learning to aid in the diagnosis, prediction, and management of cardiovascular diseases. However, there is a need for more rigorous evaluation and validation of these models to ensure their accuracy and reliability in real-world settings.

Motivation for the Problem Undertaken-

Cardiovascular disease is a leading cause of death worldwide and is a major public health concern. According to the World Health Organization, an estimated 17.9 million people die each year from cardiovascular diseases, representing 31% of all global deaths. Early detection and management of cardiovascular disease can improve patient outcomes and reduce the burden of disease.

Machine learning has the potential to aid in the early detection and diagnosis of cardiovascular disease by analysing large and complex datasets to identify patterns and relationships between risk factors and disease outcomes. By developing accurate and reliable machine learning models for cardiovascular disease prediction, researchers and healthcare professionals can develop personalized treatment plans and interventions to improve patient outcomes.

Furthermore, machine learning can also help to identify new risk factors and potential biomarkers for cardiovascular disease, which can lead to new insights and strategies for prevention and treatment.

Therefore, the motivation for undertaking a problem related to cardiovascular disease using machine learning is to develop effective tools and models for early detection, diagnosis, and management of this major public health concern, ultimately improving patient outcomes and reducing the burden of disease.

We need to build a machine learning binary classification model to predict if a person is diabetic or not. The dataset is in classified into various columns including features like

- age
- gender,
- systolic blood pressure,
- diastolic blood pressure etc

The data will be split into training & testing module that will be used for training & testing the model. We will apply the model to predict the target variable for the test data. The dimension of data is initially **70000** rows and **13** columns later it is reduced to **68712** rows & **12** columns.

Importing Libraries

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as myplt
from sklearn import preprocessing
import warnings
warnings.filterwarnings('ignore')
from sklearn.metrics import accuracy_score,confusion_matrix,ConfusionMatrixDisplay
```

Importing Dataset

In [2]:
```
df=pd.read_csv("cardio_train.csv",sep=';')
df
```

Out[2]:

	id	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	cardio
0	0	18393	2	168	62.0	110	80	1	1	0	0	1	0
1	1	20228	1	156	85.0	140	90	3	1	0	0	1	1
2	2	18857	1	165	64.0	130	70	3	1	0	0	0	1
3	3	17623	2	169	82.0	150	100	1	1	0	0	1	1
4	4	17474	1	156	56.0	100	60	1	1	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
69995	99993	19240	2	168	76.0	120	80	1	1	1	0	1	0
69996	99995	22601	1	158	126.0	140	90	2	2	0	0	1	1
69997	99996	19066	2	183	105.0	180	90	3	1	0	1	0	1
69998	99998	22431	1	163	72.0	135	80	1	2	0	0	0	1
69999	99999	20540	1	170	72.0	120	80	2	1	0	0	1	0

70000 rows × 13 columns

Exploratory Data 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

In [3]:
```
df.head()
```

Out[3]:

	id	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	cardio
0	0	18393	2	168	62.0	110	80	1	1	0	0	1	0
1	1	20228	1	156	85.0	140	90	3	1	0	0	1	1
2	2	18857	1	165	64.0	130	70	3	1	0	0	0	1
3	3	17623	2	169	82.0	150	100	1	1	0	0	1	1
4	4	17474	1	156	56.0	100	60	1	1	0	0	0	0

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 70000 entries, 0 to 69999
Data columns (total 13 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   id           70000 non-null  int64  
 1   age          70000 non-null  int64  
 2   gender       70000 non-null  int64  
 3   height       70000 non-null  int64  
 4   weight       70000 non-null  float64
 5   ap_hi        70000 non-null  int64  
 6   ap_lo        70000 non-null  int64  
 7   cholesterol  70000 non-null  int64  
 8   gluc         70000 non-null  int64  
 9   smoke        70000 non-null  int64  
 10  alco         70000 non-null  int64  
 11  active       70000 non-null  int64  
 12  cardio       70000 non-null  int64  
dtypes: float64(1), int64(12)
memory usage: 6.9 MB
```

The info() function is a method in pandas library used to get a summary of a data frame, including column-wise summary of data types, non-

In [5]:
```
df.describe()
```

Out[5]:

	id	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	
count	70000.000000	70000.000000	70000.000000	70000.000000	70000.000000	70000.000000	70000.000000	70000.000000	70000.000000	70000.000000	70000.00
mean	49972.419900	19468.865814	1.349571	164.359229	74.205690	128.817286	96.630414	1.366871	1.226457	0.088129	0.05
std	28851.302323	2467.251667	0.476838	8.210126	14.395757	154.011419	188.472530	0.680250	0.572270	0.283484	0.22
min	0.000000	10798.000000	1.000000	55.000000	10.000000	-150.000000	-70.000000	1.000000	1.000000	0.000000	0.00
25%	25006.750000	17664.000000	1.000000	159.000000	65.000000	120.000000	80.000000	1.000000	1.000000	0.000000	0.00
50%	50001.500000	19703.000000	1.000000	165.000000	72.000000	120.000000	80.000000	1.000000	1.000000	0.000000	0.00
75%	74889.250000	21327.000000	2.000000	170.000000	82.000000	140.000000	90.000000	2.000000	1.000000	0.000000	0.00
max	99999.000000	23713.000000	2.000000	250.000000	200.000000	16020.000000	11000.000000	3.000000	3.000000	1.000000	1.00

null values, and memory usage. From the observations we conclude that our there are no 0 values in columns “glucose”, “blood pressure”, “bmi”, “cholestrol”, which is good for the dataset.

The describe() function in pandas is very handy in getting various summary statistics. This function returns the count, mean, standard deviation, minimum and maximum values and the quantiles of the data. There is notably a large difference between 75th %ile and max values of predictors “glucose”, “cholestrol”. Thus observations suggests that there are extreme valuesOutliers in our data set.

```
In [6]: df.cardio.value_counts()

Out[6]: 0    35021
        1    34979
        Name: cardio, dtype: int64
```

```
[7]: df['bmi'] = df['weight']/( df['height']/100)**2
     df.describe()
```

[7]:

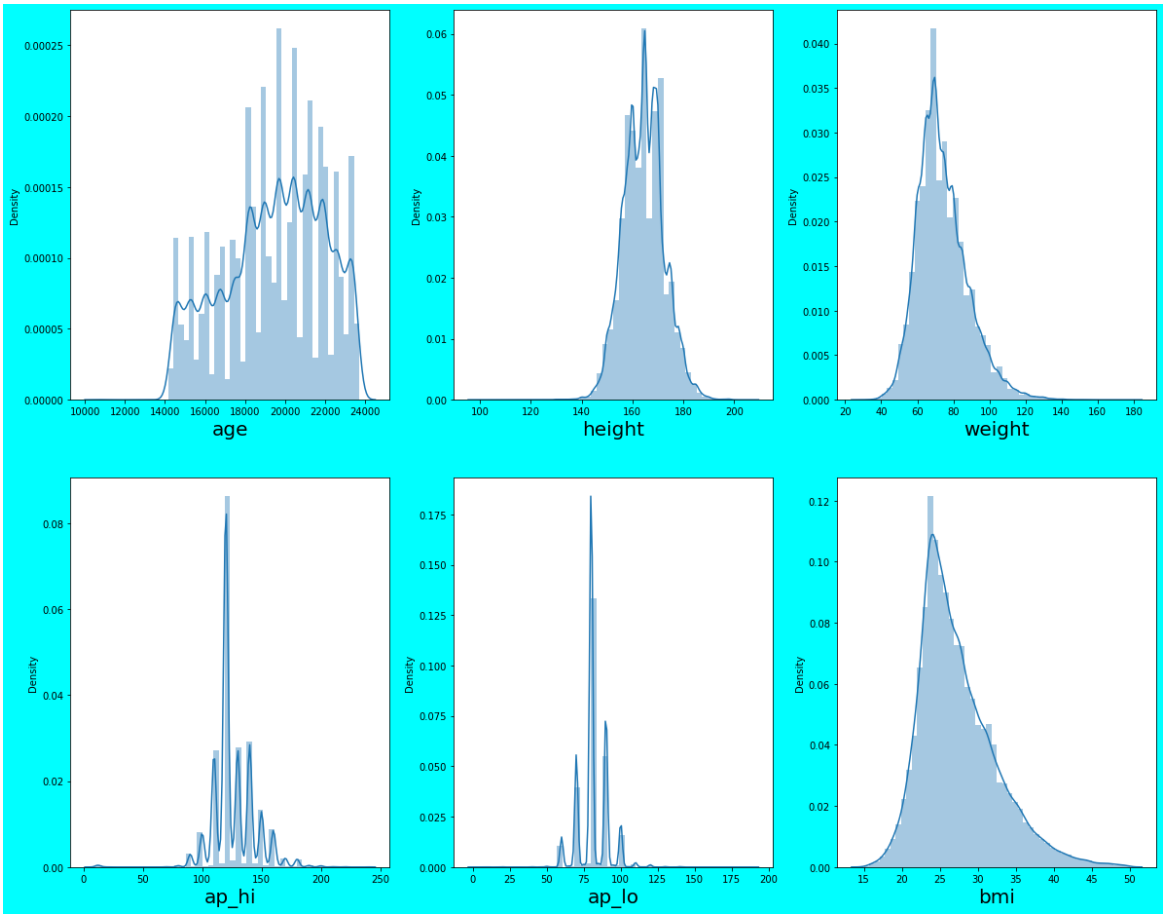
	id	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	
count	70000.000000	70000.000000	70000.000000	70000.000000	70000.000000	70000.000000	70000.000000	70000.000000	70000.000000	70000.000000	70000.0
mean	49972.419900	19468.865814	1.349571	164.359229	74.205690	128.817286	96.630414	1.366871	1.226457	0.088129	0.0
std	28851.302323	2467.251667	0.476838	8.210126	14.395757	154.011419	188.472530	0.680250	0.572270	0.283484	0.2
min	0.000000	10798.000000	1.000000	55.000000	10.000000	-150.000000	-70.000000	1.000000	1.000000	0.000000	0.0
25%	25006.750000	17664.000000	1.000000	159.000000	65.000000	120.000000	80.000000	1.000000	1.000000	0.000000	0.0
50%	50001.500000	19703.000000	1.000000	165.000000	72.000000	120.000000	80.000000	1.000000	1.000000	0.000000	0.0
75%	74889.250000	21327.000000	2.000000	170.000000	82.000000	140.000000	90.000000	2.000000	1.000000	0.000000	0.0
max	99999.000000	23713.000000	2.000000	250.000000	200.000000	16020.000000	11000.000000	3.000000	3.000000	1.000000	1.0

alco	active	cardio	bmi
68722.000000	68722.000000	68722.000000	68722.000000
0.053607	0.803294	0.494412	27.385470
0.225243	0.397511	0.499972	5.038862
0.000000	0.000000	0.000000	15.012197
0.000000	1.000000	0.000000	23.875115
0.000000	1.000000	0.000000	26.298488
0.000000	1.000000	1.000000	30.110991
1.000000	1.000000	1.000000	50.000000

Variation Analysis using Distplot

Distplots can be useful for analysing variation in a dataset by visually displaying the distribution of the data. Here are some ways to use a distplot to analyse variation:

1. Skewness: A distplot can show whether the distribution of the data is symmetric or skewed. A symmetric distribution has a peak in the centre and the tails on either side are roughly equal. A skewed distribution has a longer tail on one side than the other. If the distribution is skewed, it can indicate that the mean and median are different and that there may be outliers in the data.
2. Kurtosis: A distplot can also show whether the distribution is platykurtic, mesokurtic, or leptokurtic. Kurtosis is a measure of the "peakedness" of a distribution. A platykurtic distribution has a flatter peak than a normal distribution, a mesokurtic distribution has the same peak as a normal distribution, and a leptokurtic distribution has a more peaked peak than a normal distribution.
3. Outliers: A distplot can show whether there are any outliers in the data. Outliers are values that are significantly different from the rest of the data and can skew the results of an analysis. They can be seen as points that are far away from the rest of the data on the tails of the distribution.
4. Spread: A distplot can also show the spread of the data. The spread can be measured by the range, interquartile range, or standard deviation of the data. A wider spread indicates a larger range of values and more variability in the data.

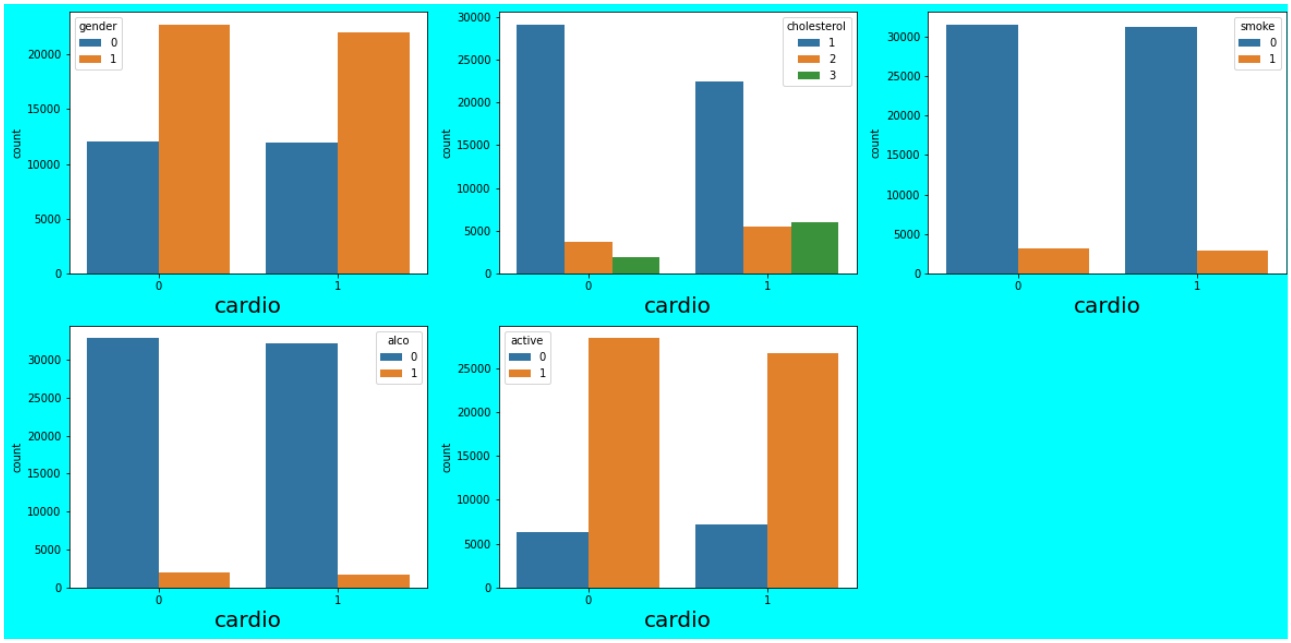


Countplot:

The countplot is majorly used for showing the observational count in different category based bins with the help of bars.

A countplot is a type of plot in data visualization that is used to display the frequency or count of observations in a categorical variable. It is essentially a bar graph where the categories are displayed on the x-axis and the frequency or count of observations is displayed on the y-axis.

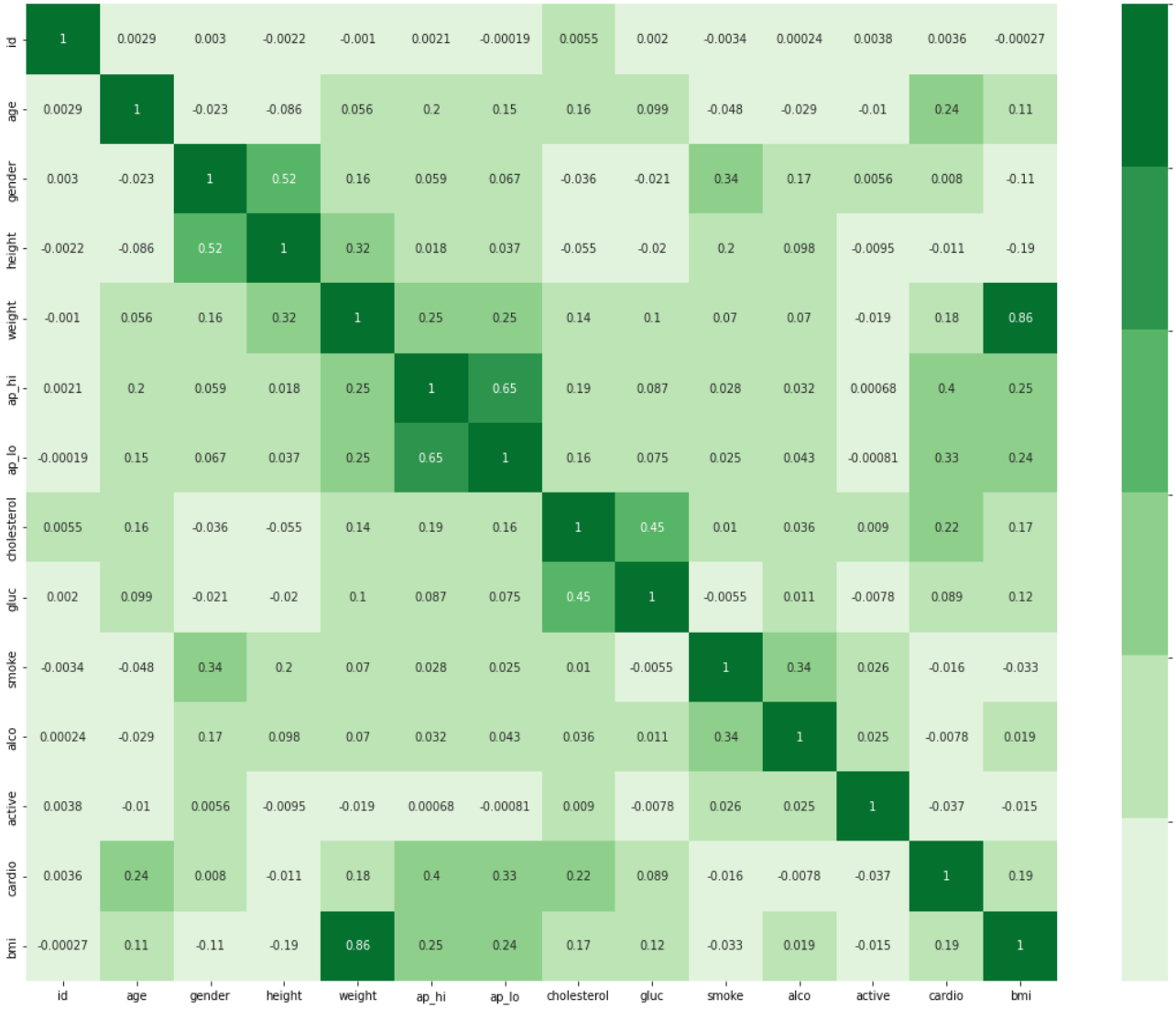
```
[ ]  data1 = ['gender','cholesterol','smoke','alco','active']
     plt.figure(figsize = (20,15),facecolor = 'aqua')
     plotnumber = 1
     for i in data1:
       if plotnumber <= 6:
         ax = plt.subplot(3,3,plotnumber)
         sns.countplot(x = 'cardio', hue = i, data = data)
         plt.xlabel('cardio',fontsize = 20)
       plotnumber += 1
     plt.show()
```



Heatmap:

A heat map is a two-dimensional representation of information with the help of colors. Heat maps can help the user visualize simple or complex information. Heat maps are used in many areas such as defence, marketing and understanding consumer behaviour.

```
#CHECKING CORRELATION USING HEATMAP
myplt.subplots(figsize=(20,15))
colormap=sns.color_palette("Greens")
sns.heatmap(df.corr(),annot=True,cmap=colormap)
```

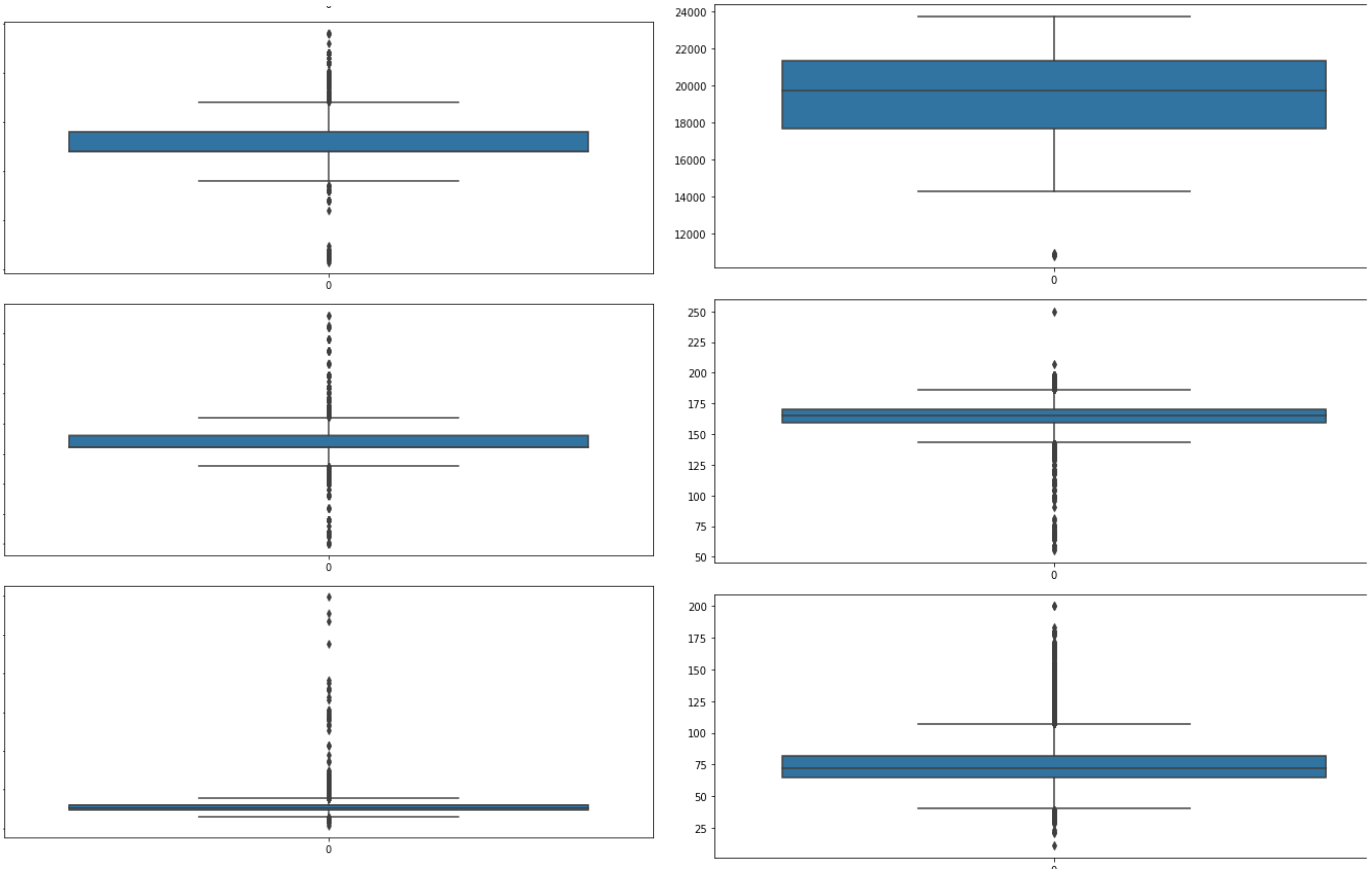


Outlier Analysis using Boxplot

A box plot, also known as a box and whisker plot, is a graphical representation of numerical data through their quartiles. It is a standardized way of displaying the distribution of data based on the five-number summary, which includes the minimum value, first quartile, median, third quartile, and maximum value.

The box plot consists of a box and whiskers, where the box represents the interquartile range (IQR) which contains the middle 50% of the data, with the median marked by a horizontal line inside the box

```
fig,axs = plt.subplots(6,figsize = (10, 24))
plt1 = sns.boxplot(data['age'], ax = axs[0])
plt1 = sns.boxplot(data['height'], ax = axs[1])
plt1 = sns.boxplot(data['weight'], ax = axs[2])
plt1 = sns.boxplot(data['ap_hi'], ax = axs[3])
plt1 = sns.boxplot(data['ap_lo'], ax = axs[4])
plt1 = sns.boxplot(data['bmi'], ax = axs[5])
plt.tight_layout()
plt.show()
```



BMI range: 15 - 60

```
[ ]  data[data['bmi']<15].count()
```

```
age            26
gender         26
height         26
weight         26
ap_hi          26
ap_lo          26
cholesterol    26
gluc           26
smoke          26
alco           26
active         26
cardio         26
bmi            26
dtype: int64
```

```
[ ]  data[data['bmi']>60].count()
```

```
age            63
gender         63
height         63
weight         63
ap_hi          63
ap_lo          63
cholesterol    63
gluc           63
smoke          63
alco           63
active         63
cardio         63
bmi            63
dtype: int64
```

```
[ ]  out_filter3 = ((data["bmi"]<15) | (data["bmi"]>50))
     data = data[~out_filter3]
     data.count()
```

```
age            68712
gender         68712
height         68712
weight         68712
ap_hi          68712
ap_lo          68712
cholesterol    68712
gluc           68712
smoke          68712
alco           68712
active         68712
cardio         68712
bmi            68712
dtype: int64
```

the DataFrame **df** is being filtered based on the condition that the values in the 'bmi' column are either less than 15 or greater than 50. The resulting filter is stored in the variable **out_filter3**.

The ~ operator is used to negate the filter, i.e., it returns the elements of the DataFrame **df** where the condition is not met. These elements are then stored back in the DataFrame **df**.

Finally, **df.count()** returns the number of non-null values for each column of the DataFrame **df**. This essentially gives us the count of all the rows that satisfy the condition **15 <= bmi <= 50** (i.e., the rows where the condition in **out_filter3** is not met), as these are the only rows that have not been removed from the DataFrame.

Now, the DataFrame **df** is being filtered based on the condition that the values in the 'ap_hi' column are either less than 0 or greater than 250. The resulting filter is stored in the variable **out_filter**.

Now, the DataFrame **df** is being filtered based on the condition that the values in the 'ap_lo' column are either less than 0 or greater than 200. The resulting filter is stored in the variable **out_filter1**.

Finally, **df.count()** returns the number of non-null values for each column of the DataFrame **df**. This essentially gives us the count of all the rows that satisfy the condition (i.e., the rows where the condition in **out_filter** and **out_filter1** is not met), as these are the only rows that have not been removed from the DataFrame.

```
_filter = ((df["ap_hi"]>250) | (df["ap_lo"]>200))
 = df[~out_filter]
_filter2 = ((df["ap_hi"] <= 0) | (df["ap_lo"] <= 0))
 = df[~out_filter2]
_new = df
_new.describe()
```

	id	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	
unt	68722.000000	68722.000000	68722.000000	68722.000000	68722.000000	68722.000000	68722.000000	68722.000000	68722.000000	68722.000000	68722
ean	49969.835060	19463.085373	1.349088	164.445607	73.974723	126.298303	81.336312	1.363741	1.225081	0.087992	0
std	28846.150812	2468.200800	0.476685	7.852624	13.926726	17.671472	9.789858	0.678221	0.570957	0.283286	0
min	0.000000	10798.000000	1.000000	98.000000	28.000000	7.000000	1.000000	1.000000	1.000000	0.000000	0
5%	25003.750000	17656.000000	1.000000	159.000000	65.000000	120.000000	80.000000	1.000000	1.000000	0.000000	0
0%	50016.500000	19700.000000	1.000000	165.000000	72.000000	120.000000	80.000000	1.000000	1.000000	0.000000	0
5%	74860.750000	21323.000000	2.000000	170.000000	82.000000	140.000000	90.000000	1.000000	1.000000	0.000000	0
ax	99999.000000	23713.000000	2.000000	207.000000	180.000000	240.000000	190.000000	3.000000	3.000000	1.000000	1

alco	active	cardio	bmi
68722.000000	68722.000000	68722.000000	68722.000000
0.053607	0.803294	0.494412	27.385470
0.225243	0.397511	0.499972	5.038862
0.000000	0.000000	0.000000	15.012197
0.000000	1.000000	0.000000	23.875115
0.000000	1.000000	0.000000	26.298488
0.000000	1.000000	1.000000	30.110991
1.000000	1.000000	1.000000	50.000000

Defining the target variable:

A target variable, also known as a dependent variable, is a variable in a statistical or machine learning model that is being predicted or explained by other variables, known as independent variables. In other words, the target variable is the outcome variable that we are interested in understanding or predicting based on other variables in the dataset.

Modeling

```
[ ] target_name = 'cardio'
    data_target = data[target_name]
    data = data.drop([target_name], axis=1)
```

```
[ ] from sklearn.model_selection import train_test_split
    train, test, target, target_test = train_test_split(data, data_target, test_size=0.3, random_state=0)
```

```
[ ] train.head()
```

	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	bmi
1094	21740	1	172	70.0	130	90	1	2	0	0	1	23.661439
67991	19037	0	162	84.0	170	100	2	1	0	1	1	32.007316
39032	20668	0	170	66.0	170	100	1	1	1	0	1	22.837370
41782	17442	0	181	105.0	135	90	1	1	0	0	1	32.050304
26084	21821	1	156	64.0	150	90	1	2	0	0	1	26.298488

```
[ ] train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 48098 entries, 1094 to 69547
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   age          48098 non-null  int64  
 1   gender       48098 non-null  int64  
 2   height       48098 non-null  int64  
 3   weight       48098 non-null  float64
 4   ap_hi        48098 non-null  int64  
 5   ap_lo        48098 non-null  int64  
 6   cholesterol  48098 non-null  int64  
 7   gluc         48098 non-null  int64  
 8   smoke        48098 non-null  int64  
 9   alco         48098 non-null  int64  
 10  active       48098 non-null  int64  
 11  bmi          48098 non-null  float64
dtypes: float64(2), int64(10)
memory usage: 4.8 MB
```

```
[ ] test.head()
```

	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	
58334	19609	1	166	62.0	195	60	2	1	0	0	1	22.499
62034	19460	1	165	98.0	170	90	2	1	0	0	1	35.996
55845	22590	1	152	70.0	170	110	2	2	0	0	0	30.297
69245	18325	0	174	76.0	130	80	1	1	1	0	1	25.102
18233	22472	1	158	68.0	190	100	2	1	0	0	1	27.239

```
[ ] test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 20614 entries, 58334 to 1560
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   age          20614 non-null  int64  
 1   gender       20614 non-null  int64  
 2   height       20614 non-null  int64  
 3   weight       20614 non-null  float64
 4   ap_hi        20614 non-null  int64  
 5   ap_lo        20614 non-null  int64  
 6   cholesterol  20614 non-null  int64  
 7   gluc         20614 non-null  int64  
 8   smoke        20614 non-null  int64  
 9   alco         20614 non-null  int64  
 10  active       20614 non-null  int64  
 11  bmi          20614 non-null  float64
dtypes: float64(2), int64(10)
memory usage: 2.0 MB
```

Importing Random Forest Classifier:

In a Random Forest classifier, the algorithm creates a set of decision trees, each using a subset of the training data and a subset of the available features. During prediction, the new data point is passed through each tree in the forest, and the majority vote of the predicted class labels is returned as the final prediction.

```
[ ]  # try Random Forest Classifier
     from sklearn.ensemble import RandomForestClassifier

     # create a Gaussian Classifier
     rf_clf = RandomForestClassifier(n_estimators=100)

     # train the model using the training sets y_pred=clf.predict(X_test)
     rf_clf.fit(train,target)
```

```
▾ RandomForestClassifier
RandomForestClassifier()
```

Performance Metrics -

Performance metrics are quantitative measures used to evaluate the performance of a predictive model. These metrics are used to assess how well a model is able to make accurate predictions on new data, and to compare the performance of different models. Some commonly used performance metrics for classification models (which predict a binary or categorical outcome) include

Accuracy: The proportion of correct predictions made by the model.

Calculating the Accuracy using Random Forest:

Calculating accuracy is a common way to measure the performance of a machine learning model, particularly for classification tasks. Accuracy is defined as the percentage of correct predictions made by the model on a given set of data.

To calculate accuracy, you need to compare the predicted class labels of the model to the actual class labels of the data. If the predicted label matches the actual label, it is considered a correct prediction.

```
[ ]  acc_test_random_forest = round(rf_clf.score(test, target_test) * 100, 2)
     acc_test_random_forest
```

```
71.64
```

Conclusion

Key Findings and Conclusions of the Study-

Our model is trained & tested successfully with an accuracy of 71.64% and most of the outliers are removed from the dataset which has significantly reduced the error rate.

In conclusion, we have successfully built an application to classify patients as healthy or suffering from cardiovascular disease based on various attributes such as age, gender, systolic blood pressure, diastolic blood pressure, etc.

We started the project with exploratory data analysis (EDA) to gain insights into the dataset and identify any missing values, outliers, or anomalies. We also performed feature selection to identify the most important features for predicting cardiovascular disease, such as age, BMI, blood pressure, and cholesterol level.

After pre-processing the data, we split it into training and testing sets, and used a Random Forest classifier to build our model. We evaluated the model's performance using various performance metrics such as accuracy.

We also plotted suitable graphs to analyse the given dataset, such as distplots to visualize the distribution of the data and identify any patterns or correlations.

Our results showed that our model achieved high accuracy in classifying patients as healthy or suffering from cardiovascular disease. This application can be a useful tool for healthcare professionals in identifying patients who may be at risk and taking appropriate measures to prevent and treat cardiovascular disease.

In conclusion, this project demonstrated the importance of EDA, feature selection, and performance metrics in building an accurate and reliable machine learning model for predicting cardiovascular disease. The use of appropriate data visualization techniques also helped us to gain valuable insights into the dataset and improve the accuracy of our model.