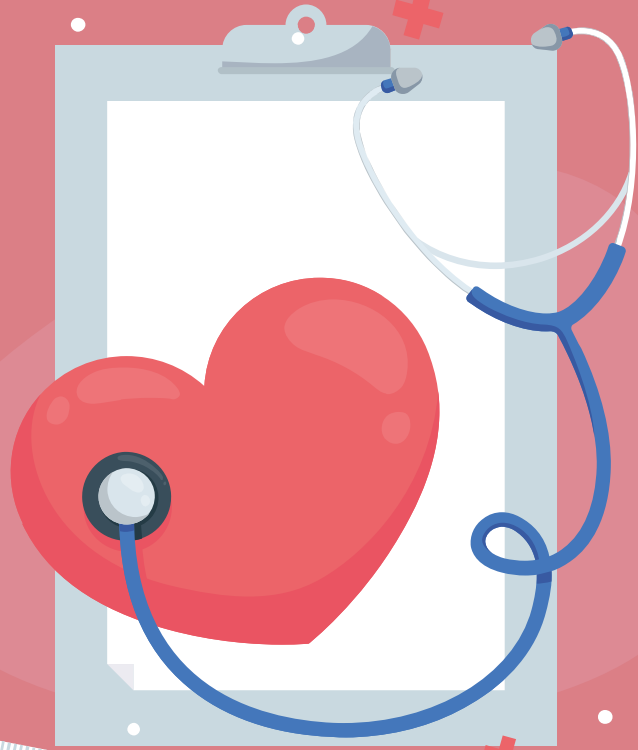
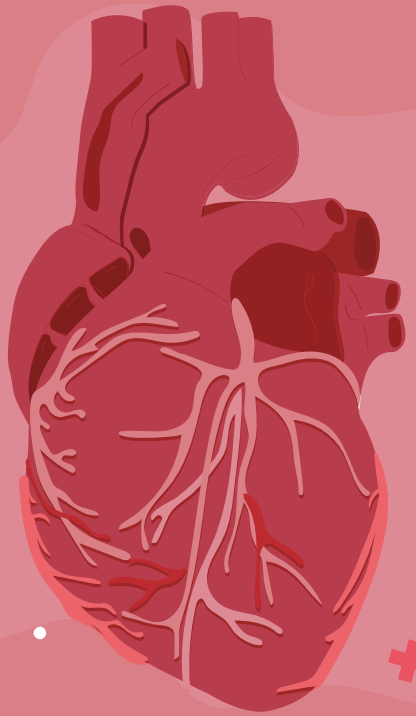
CARDIOVASCULAR DISEASE

By: Wing, Shannon, Joey and Carissa

CONTINUE



CONTEXT



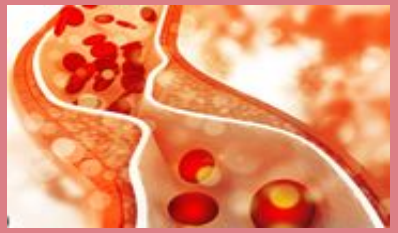
In Singapore, Cardiovascular disease accounted for 32% of all deaths in 2021 for adults

In Singapore, 21 people die from cardiovascular disease every day.

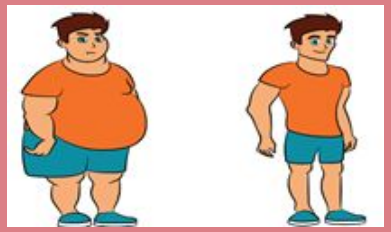
PREV NEXT

PROBLEM STATEMENT

Age

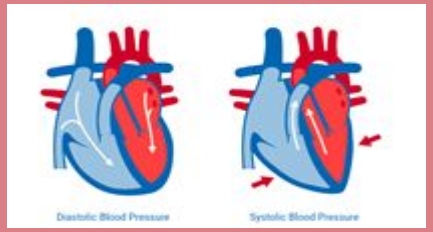


Cholesterol



Weight

Smoke



Blood pressure

Gender

Glucose

Height

In this project, we aim to predict what is the leading cause of cardiovascular disease in Singapore, using machine learning and data science, using the given data set.

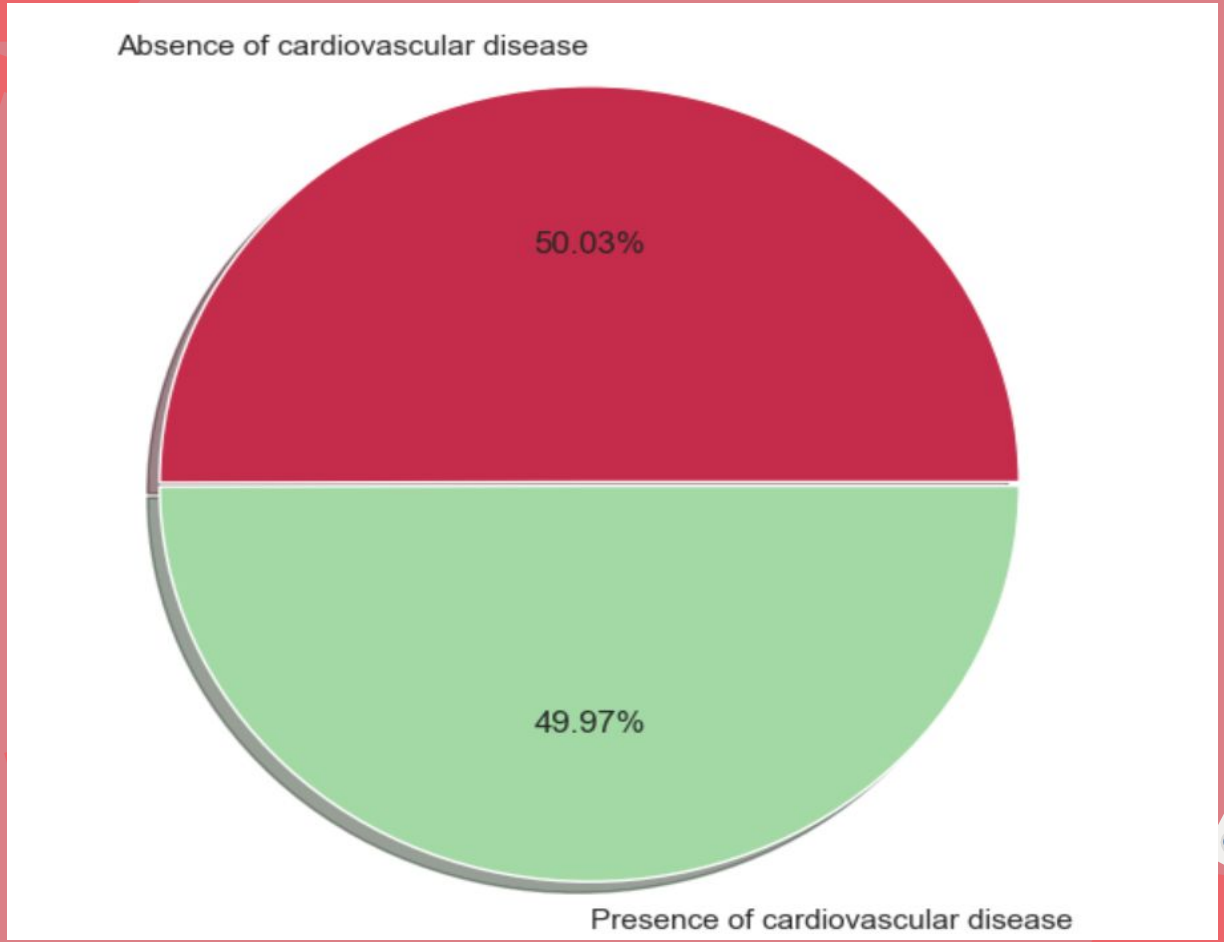
Exploratory Data Analysis

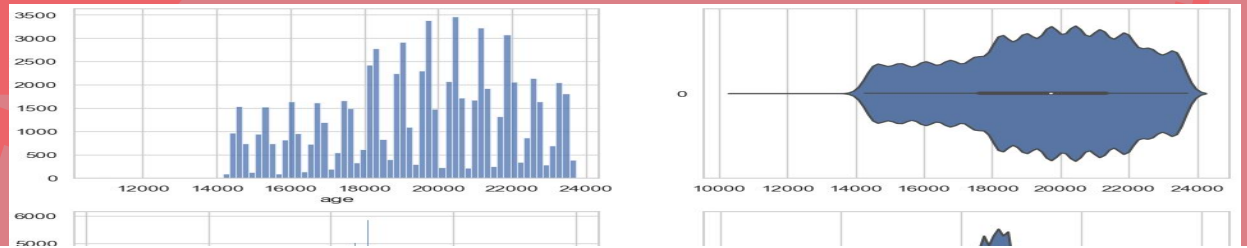
```
#Reading the file
cardio_data=pd.read_cs
cardio_data.head()
```

	id	age	gender	heig	alco	active	cardio
0	0	18393	2	16	0	1	0
1	1	20228	1	15	0	1	1
2	2	18857	1	16	0	0	1
3	3	17623	2	16	0	1	1
4	4	17474	1	15	0	0	0

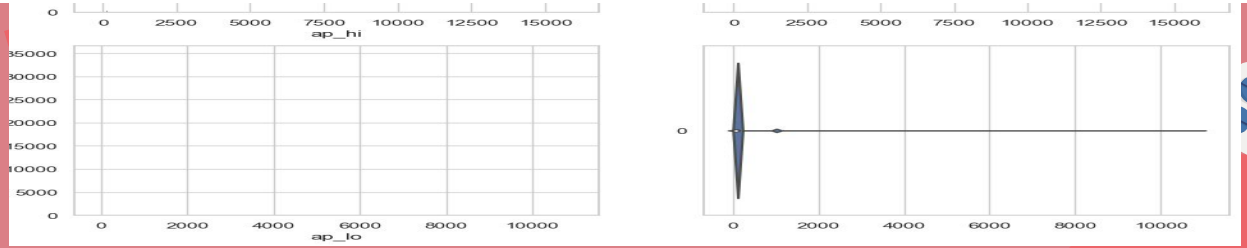
```
#Information of dataset
cardio_data.info()
```

```
<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
```

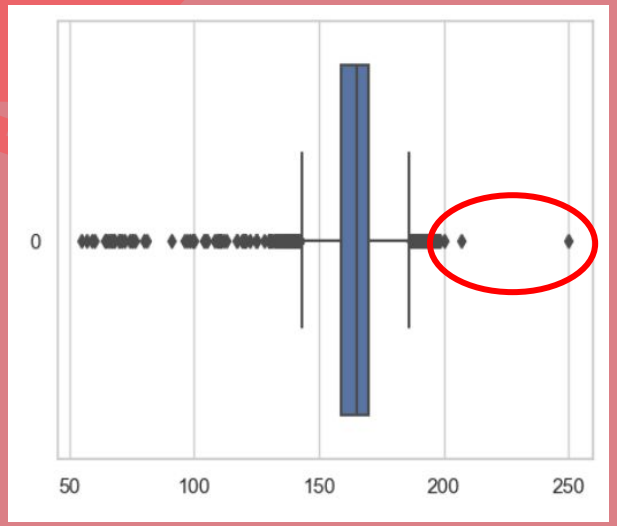

Absence of cardiovascular disease
50.03%
49.97%
Presence of cardiovascular disease



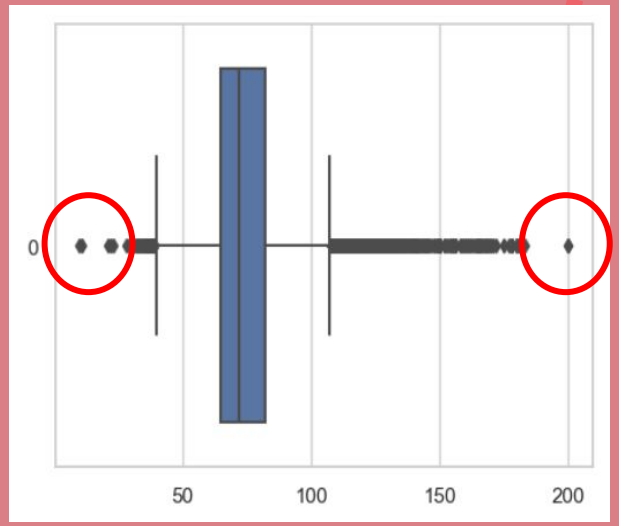
	age	height	weight	ap_hi	ap_lo	cardio
count	70000.000000	70000.000000	70000.000000	70000.000000	70000.000000	70000.000000
mean	19468.865814	164.359229	74.205690	128.817286	96.630414	0.499700
std	2467.251667	8.210126	14.395757	154.011419	188.472530	0.500003
min	10798.000000	55.000000	10.000000	-150.000000	-70.000000	0.000000
25%	17664.000000	159.000000	65.000000	120.000000	80.000000	0.000000
50%	19703.000000	165.000000	72.000000	120.000000	80.000000	0.000000
75%	21327.000000	170.000000	82.000000	140.000000	90.000000	1.000000
max	23713.000000	250.000000	200.000000	16020.000000	11000.000000	1.000000



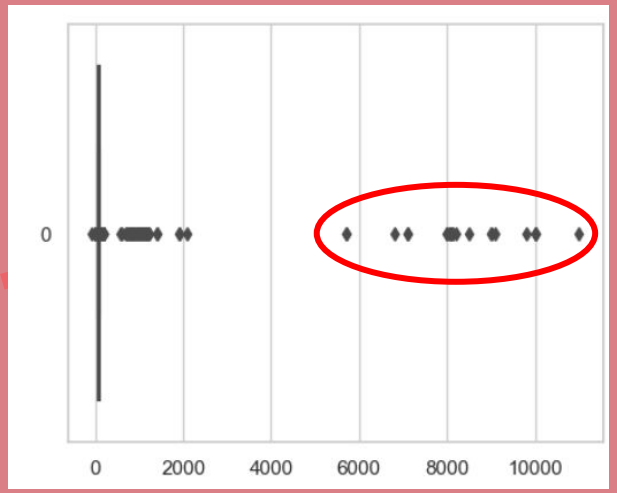
height



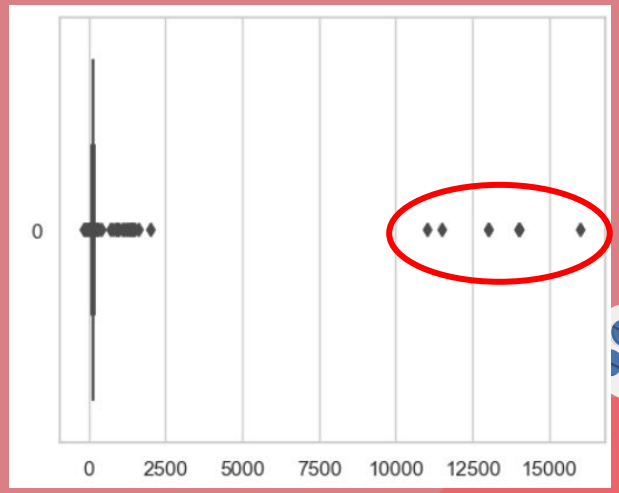
weight



ap_lo



ap_hi



Problems

Blood Pressure

The maximum and minimum values of blood pressure is impossible.

Minimum Value: impossible to be negative

Maximum Value: Highest recorded human blood pressure was 370/360 mmhg and BP lower than 90/60 mmhg are considered hypotension.

In addition, diastolic pressure cannot be higher than systolic pressure

Height & Weight

The average height for a 2 y/o child is ~86.8 cm & 12.46 kg.

Hence, impossible for the values of height and weight of an average adult to be lower than that of a child.

Data Set Preparation

Removing Impossible Values

Problem 2: The max and min of blood pressure is off. Highest recorded huma
considered hypotension.

Problem 3: Diastillic pressure cannot be hgiher than systolic pressure.

```
    print("Diastilic pressure is higher than systolic in {0} cases".format
```

75]

```
Diastilic pressure is higher than systolic in 1234 cases
```

Removing Outliers

```
cardio_data_numerics=cardio_data[['age', 'height', 'weight', 'ap_hi', 'ap_lo','cardio']]
cardio_data_numerics.describe()
```

	age	height	weight	ap_hi	ap_lo	cardio
count	70000.000000	70000.000000	70000.000000	70000.000000	70000.000000	70000.000000
mean	19468.865814	164.359229	74.205690	128.817286	96.630414	0.499700
std	2467.251667	8.210126	14.395757	154.011419	188.472530	0.500003
min	10798.000000	55.000000	10.000000	-150.000000	-70.000000	0.000000
25%	17664.000000	159.000000	65.000000	120.000000	80.000000	0.000000
50%	19703.000000	165.000000	72.000000	120.000000	80.000000	0.000000
75%	21327.000000	170.000000	82.000000	140.000000	90.000000	1.000000
max	23713.000000	250.000000	200.000000	16020.000000	11000.000000	1.000000

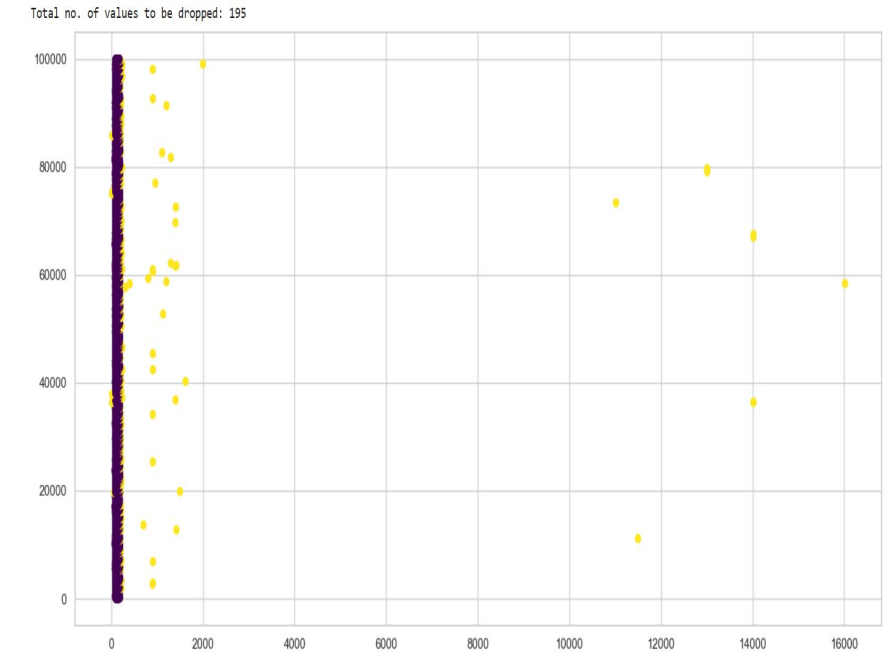
TURKEY FENCE METHOD

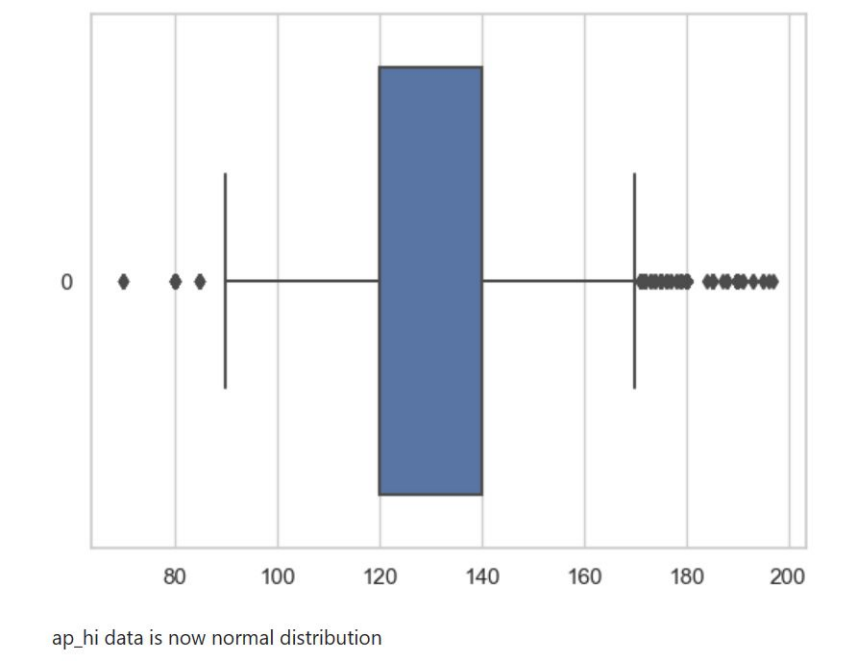
```python
#Making a function to remove outliers
def flag_outliers(df, var, degree):
    #Getting the interquartile range
    lq = df[var].quantile(0.25)
    uq = df[var].quantile(0.75)
    iqr = uq - lq
    #Generating Upper/Lower Bounds
    lowerbound = lq - degree*iqr
    upperbound = uq + degree*iqr
    #Adding a column with a boolean flag to flag out anomalies
    df['flag'] = 0
    df.loc[(df[var] <= lowerbound) | (df[var] >= upperbound), 'flag'] = 1
    #Show a scatterplot of the values to be dropped
    f, axes = plt.subplots(1, 1, figsize=(16,8))
    plt.scatter(x = var, y = "id", c = "flag", cmap = 'viridis', data = df)
    total=df['flag'].sum()
    print('Total no. of values to be dropped:',total)
    del df['flag']
    return lowerbound, upperbound
```

```python
def drop_outliers(df, var, lowerbound, upperbound):
    df.drop(df[(df[var] <= lowerbound) | (df[var] >= upperbound)].index,inplace=True)

    #Plot boxplot of variable after dropping
    sb.boxplot(data = df[var], orient = "h")
```

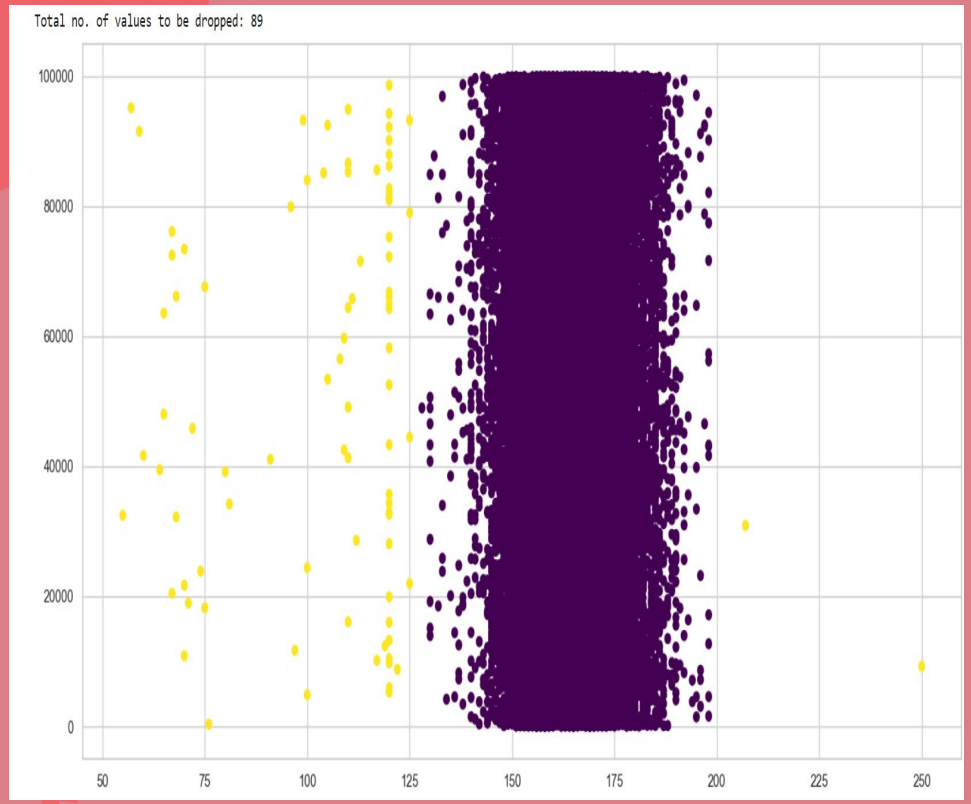
Systolic Blood pressure

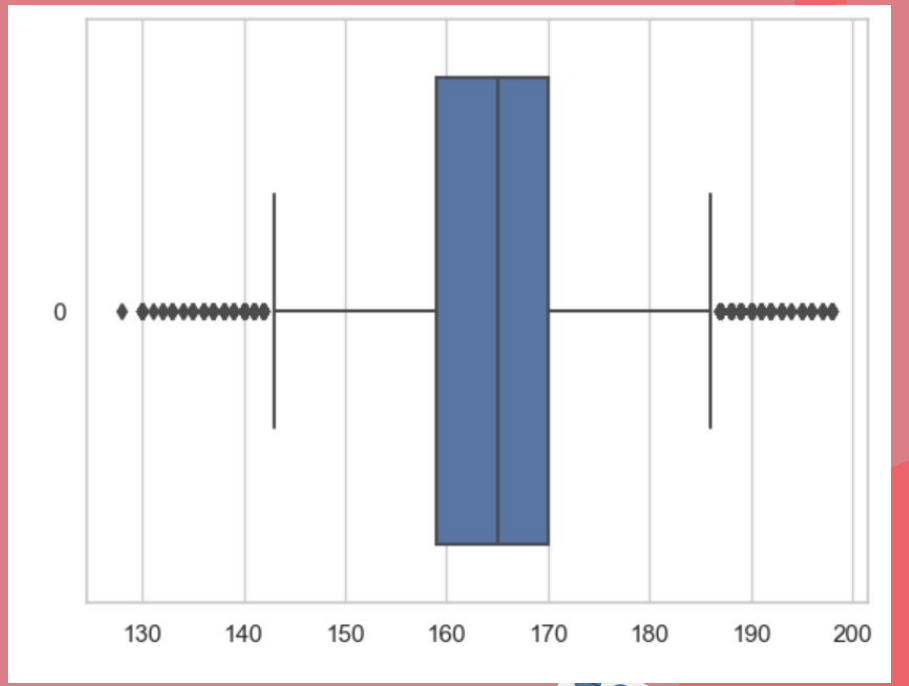




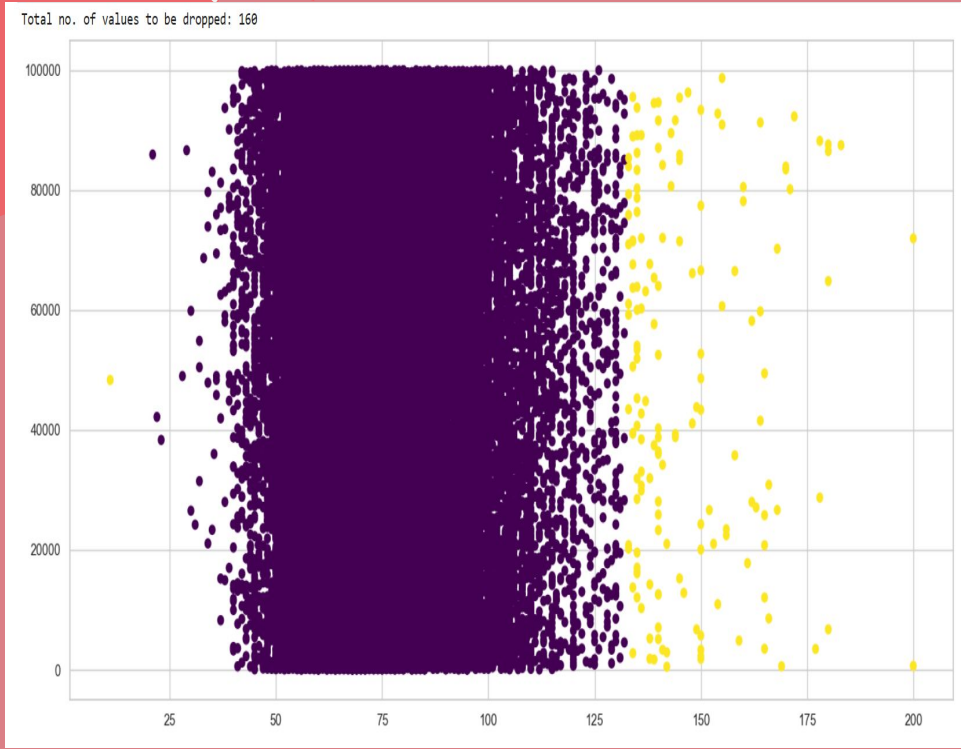
ap_hi data is now normal distribution

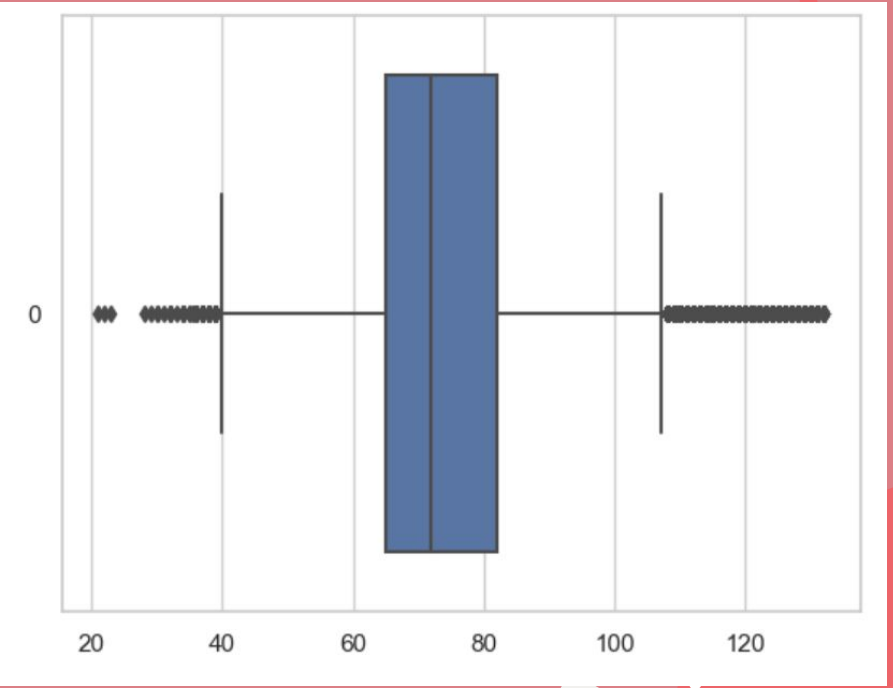
Height


Total no. of values to be dropped: 89
100000
80000
60000
40000
20000
0
50
75
100
125
150
175
200
225
250

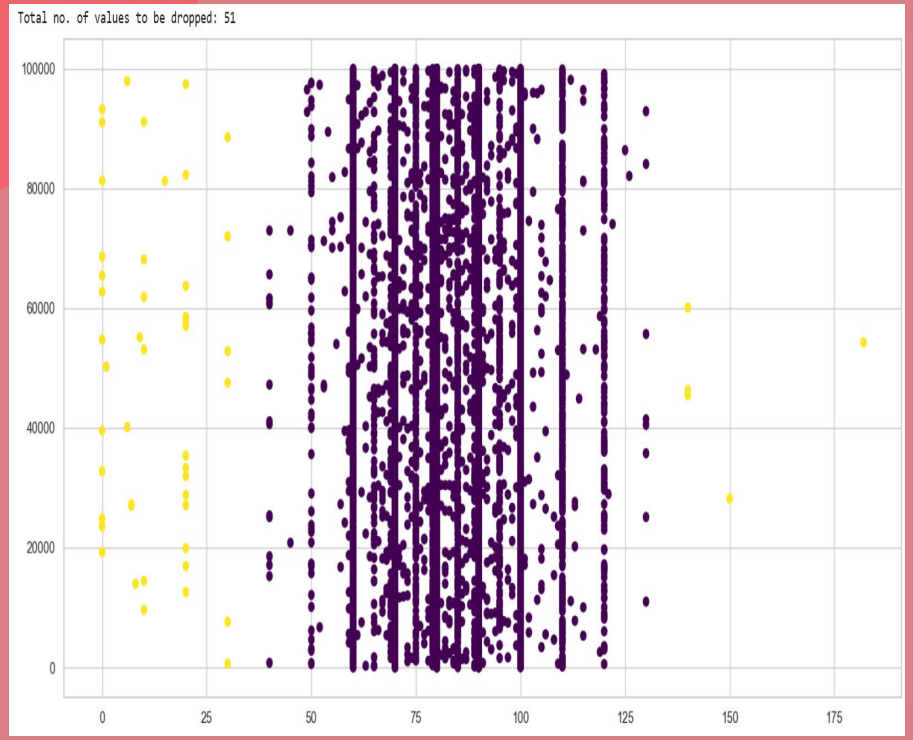

0
130
140
150
160
170
180
190
200

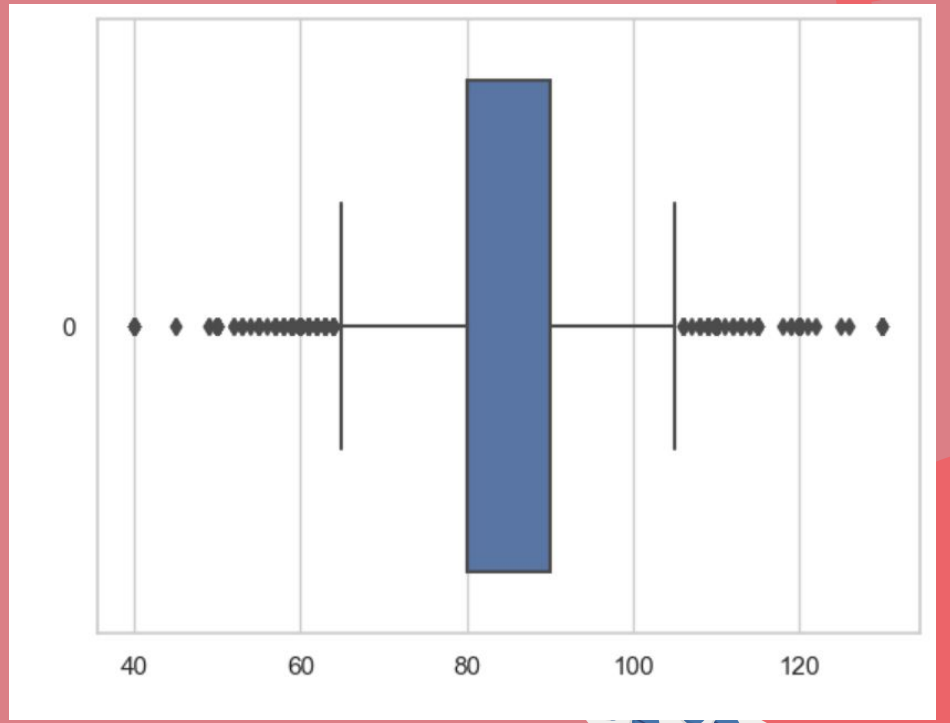
Weight


Total no. of values to be dropped: 160
100000
80000
60000
40000
20000
0
25
50
75
100
125
150
175
200

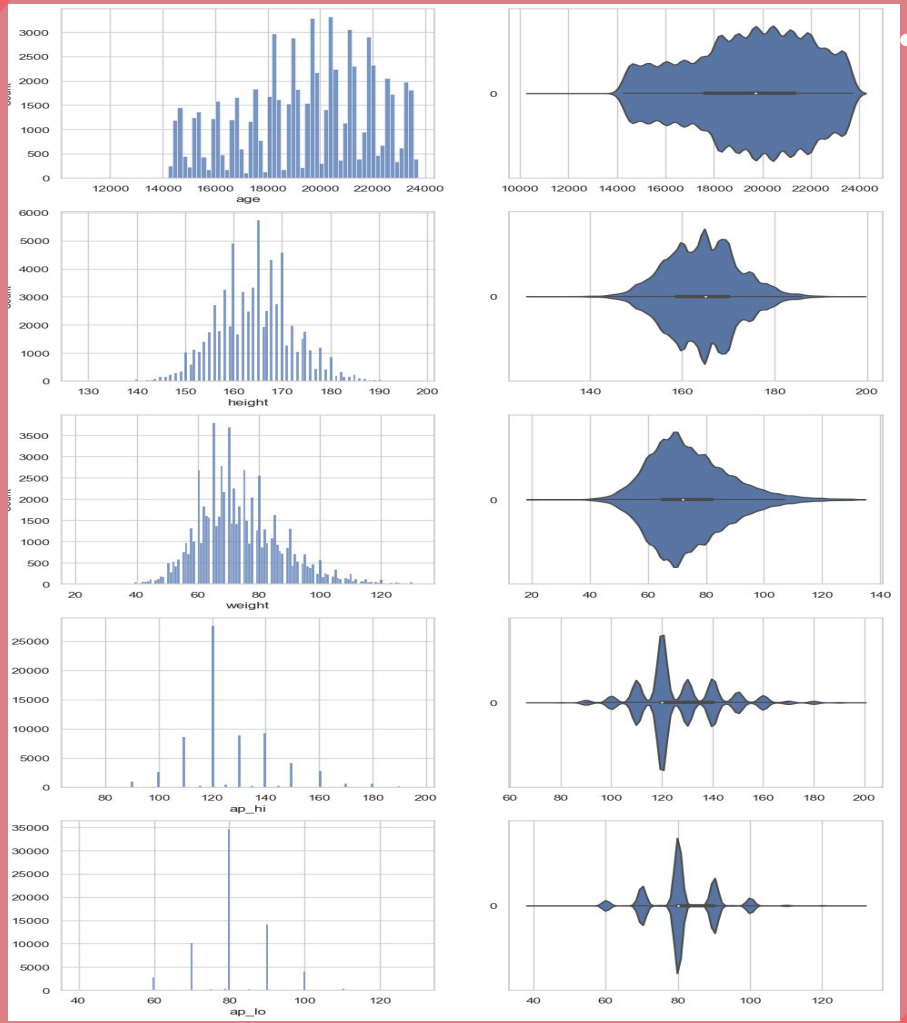

0
20
40
60
80
100
120

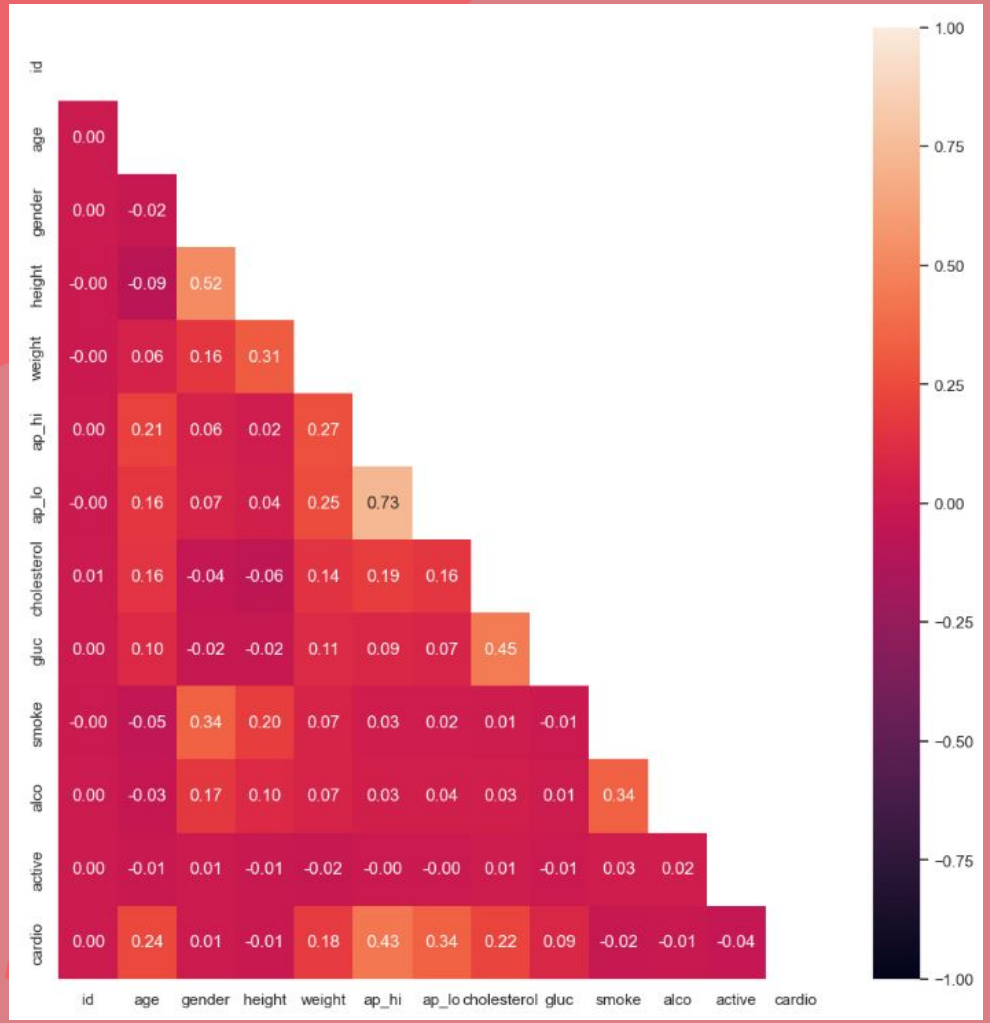
Diastolic Blood Pressure





	id	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	cardio
count	68322.000000	68322.00000	68322.000000	68322.000000	68322.000000	68322.000000	68322.000000	68322.000000	68322.000000	68322.000000	68322.000000	68322.000000	68322.000000
mean	49972.840783	19464.38440	1.348116	164.423861	73.912684	126.474401	81.178493	1.363558	1.224569	0.087644	0.053072	0.803519	0.493384
std	28842.164472	2468.54202	0.476377	7.836948	13.831609	16.270779	9.474229	0.678157	0.570487	0.282778	0.224179	0.397340	0.499960
min	0.000000	10798.00000	1.000000	128.000000	21.000000	70.000000	0.000000	1.000000	1.000000	0.000000	0.000000	0.000000	0.000000
25%	25006.250000	17657.00000	1.000000	159.000000	65.000000	120.000000	80.000000	1.000000	1.000000	0.000000	0.000000	1.000000	0.000000
50%	50011.000000	19701.00000	1.000000	165.000000	72.000000	120.000000	80.000000	1.000000	1.000000	0.000000	0.000000	1.000000	0.000000
75%	74856.750000	21325.00000	2.000000	170.000000	82.000000	140.000000	90.000000	1.000000	1.000000	0.000000	0.000000	1.000000	1.000000
max	99999.000000	23713.00000	2.000000	198.000000	132.000000	197.000000	182.000000	3.000000	3.000000	1.000000	1.000000	1.000000	1.000000

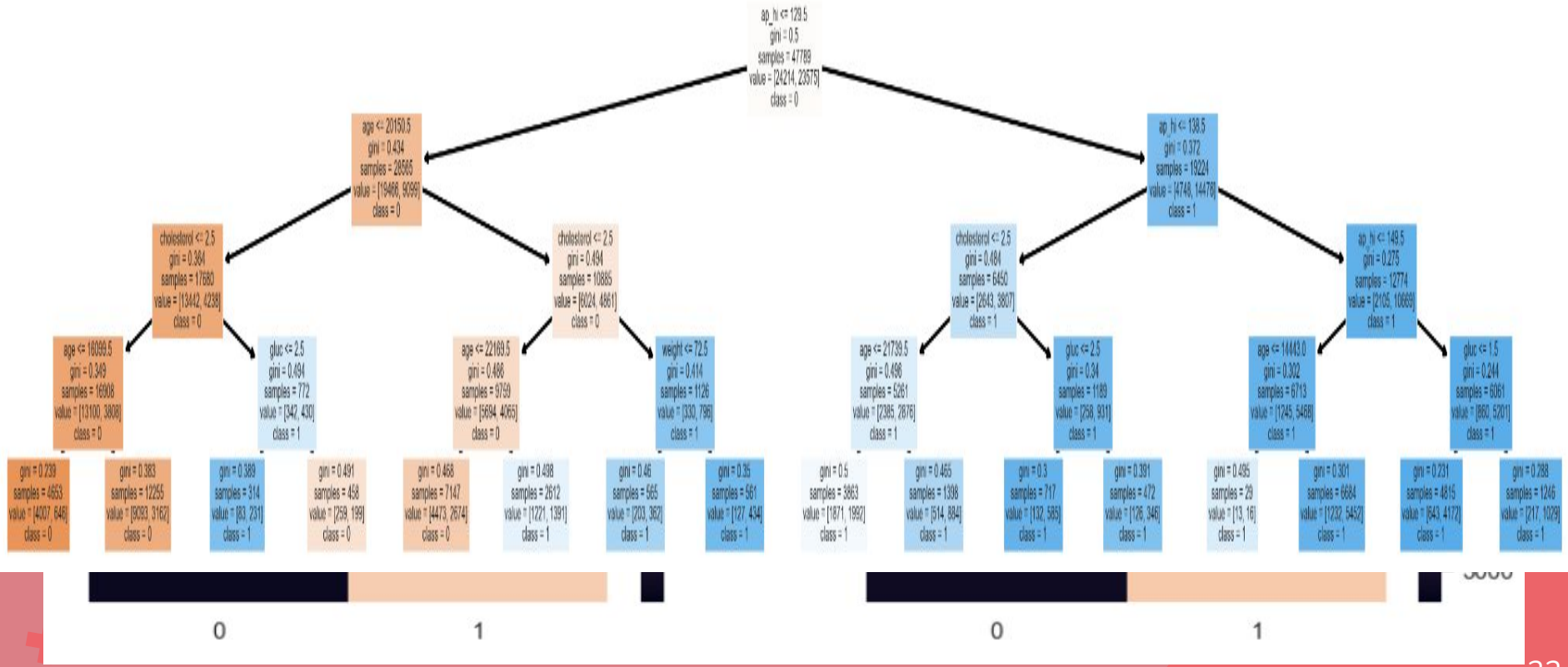

count
3000
2500
2000
1500
1000
500
0
12000 14000 16000 18000 20000 22000 24000
age
0
10000 12000 14000 16000 18000 20000 22000 24000
count
6000
5000
4000
3000
2000
1000
0
130 140 150 160 170 180 190 200
height
0
140 160 180 200
count
3500
3000
2500
2000
1500
1000
500
0
20 40 60 80 100 120
weight
0
20 40 60 80 100 120 140
25000
20000
15000
10000
5000
0
80 100 120 140 160 180 200
ap_hi
0
60 80 100 120 140 160 180 200
35000
30000
25000
20000
15000
10000
5000
0
40 60 80 100 120
ap_lo
0
40 60 80 100 120



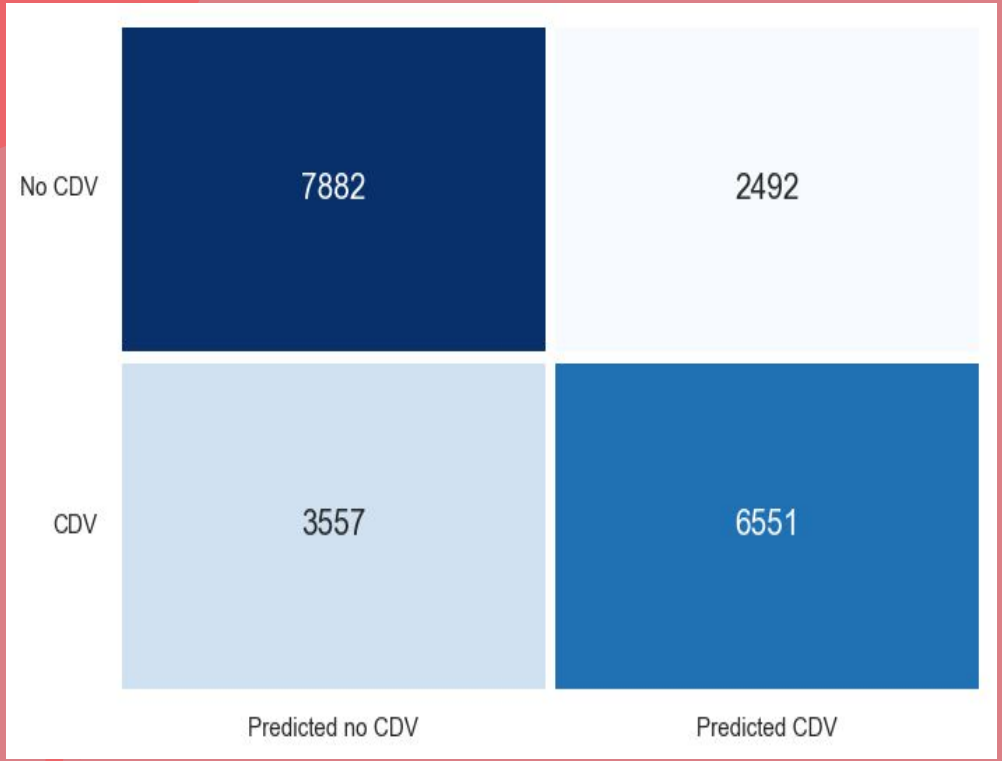
From this heatmap, ap_hi and ap_low seems to be a main factor of CDV, followed by cholesterol.

Data Science
Machine Learning

Decision Tree Classification

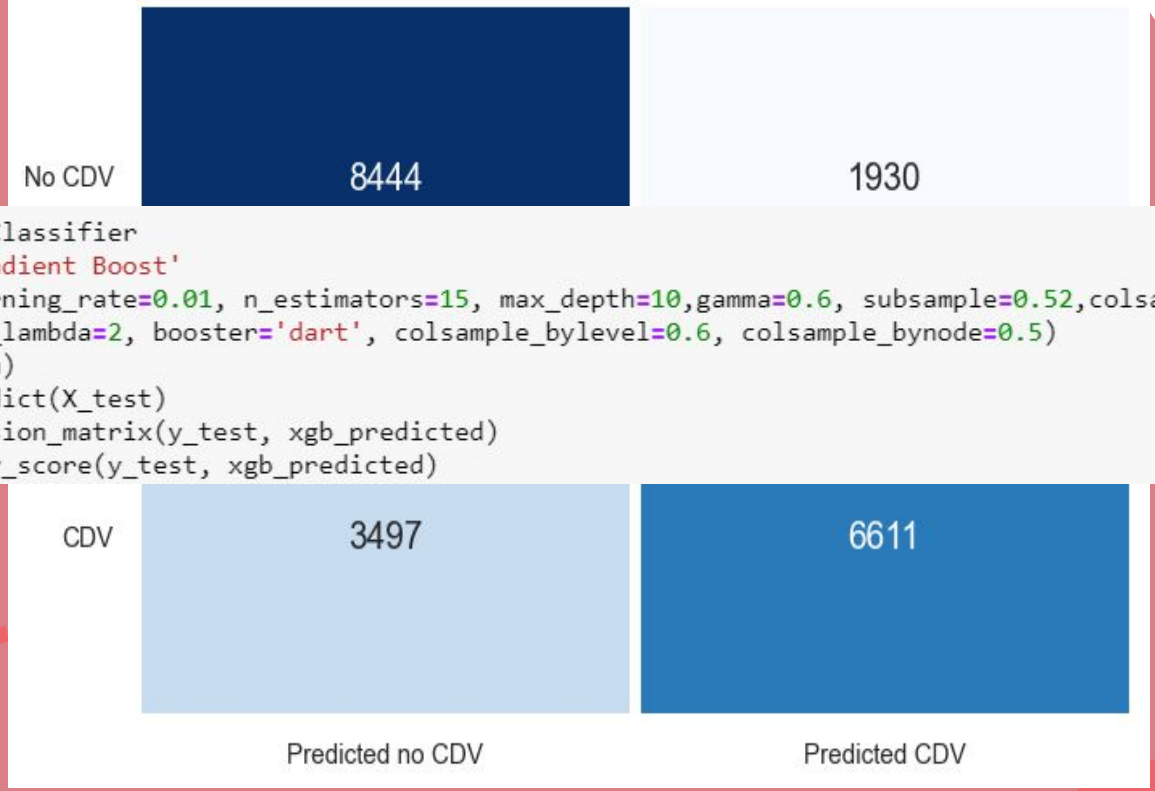


Logistic Regression



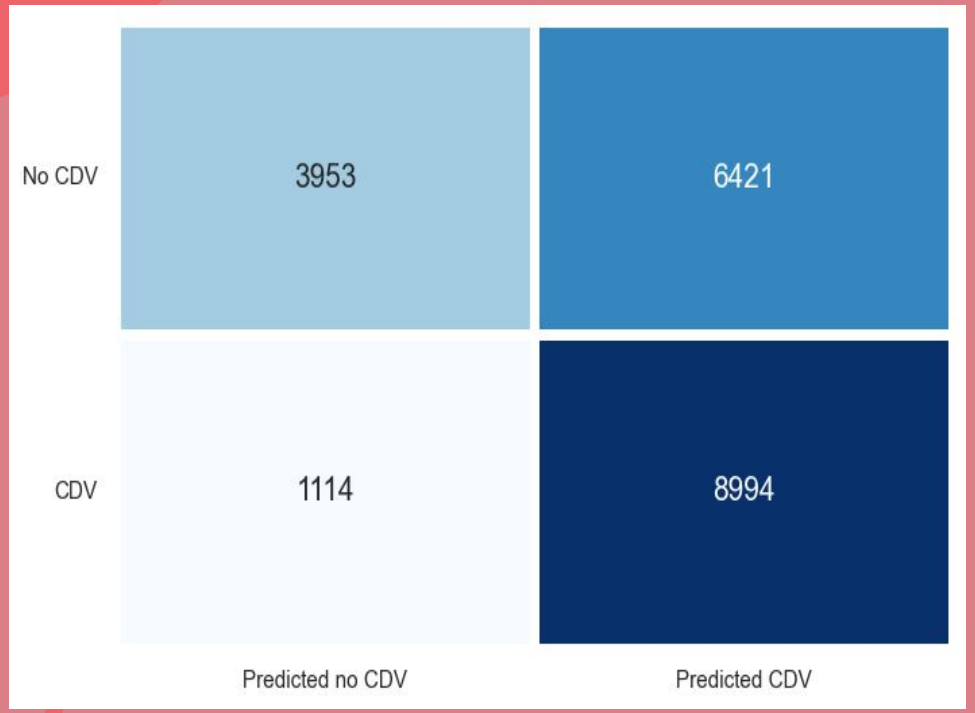

```
model_lgr = 'Logistic Regression'
lr = LogisticRegression()
model = lr.fit(X_train, y_train)
lr_predict = lr.predict(X_test)
lr_conf_matrix = confusion_matrix(y_test, lr_predict)
lr_acc_score = accuracy_score(y_test, lr_predict)
print("confussion matrix")
print(lr_conf_matrix)
print("------------------------------------------")
print("Accuracy of Logistic Regression:",lr_acc_score*100,'\n')
print("------------------------------------------")
print(classification_report(y_test,lr_predict))
```

Extreme Gradient Boost




```
from xgboost import XGBClassifier
model_egb = 'Extreme Gradient Boost'
xgb = XGBClassifier(learning_rate=0.01, n_estimators=15, max_depth=10,gamma=0.6, subsample=0.52,colsample_bytree=0.6,seed=27,
                    reg_lambda=2, booster='dart', colsample_bylevel=0.6, colsample_bynode=0.5)
xgb.fit(X_train, y_train)
xgb_predicted = xgb.predict(X_test)
xgb_conf_matrix = confusion_matrix(y_test, xgb_predicted)
xgb_acc_score = accuracy_score(y_test, xgb_predicted)
```

Stochastic Gradient Descent




```
model_sgd = 'Stochastic Gradient Descent'
sgdc = SGDClassifier(max_iter=5000, random_state=0)
sgdc.fit(X_train, y_train)
sgdc_predicted = sgdc.predict(X_test)
sgdc_conf_matrix = confusion_matrix(y_test, sgdc_predicted)
sgdc_acc_score = accuracy_score(y_test, sgdc_predicted)
print("confussion matrix")
print(sgdc_conf_matrix)
print("------------------------------------------")
print("Accuracy of : Stochastic Gradient Descent",sgdc_acc_score*100,'\n')
print("------------------------------------------")
print(classification_report(y_test,sgdc_predicted))
```

New Knowledge Learnt

Turkey Fence Method
Logistic Regression
Extreme Gradient Boost.
Stochastic Gradient Descent

Outcome

Outcome

	Model	Accuracy
0	Decision Tree	72.035388
1	Logistic Regression	71.888915
2	Extreme Gradient Boost	71.566674
3	Stochastic Gradient Descent	70.371455

Most of the models used have an accuracy of ≥ 70%, which suggests that they are accurate in predicting the presence of CDV based on it inputs variables.

Therefore, we can conclude that Decision Tree Classification yields an accuracy of 72%. This makes it a decent model for predicting what are the leading causes of cardiovascular diseases, in this case, blood pressure and cholesterol, which is the main objective of this project.

Individual Contribution	Wing	Shannon	Joey	Carissa
Modeling of data(Decision Tree Classification and Logistic regression)	X	X		
Data Preparation (cleaning of data)			X	X
Modeling of data(Extreme gradient boost and stochastic gradient descent)	X	X		
Exploratory Analysis			X	X

Q&A

Thank you for your kind attention!