Name : Divya Khiani

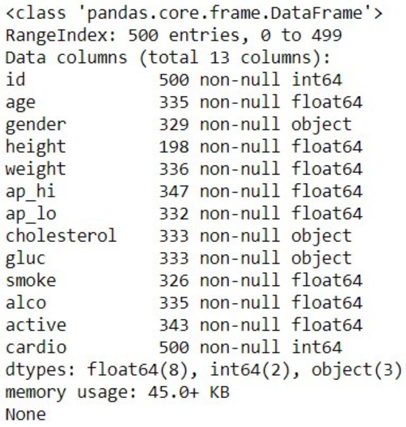
SJSU ID : 015273245

Report on Cardio Vascular Disease Prediction

TASK 1

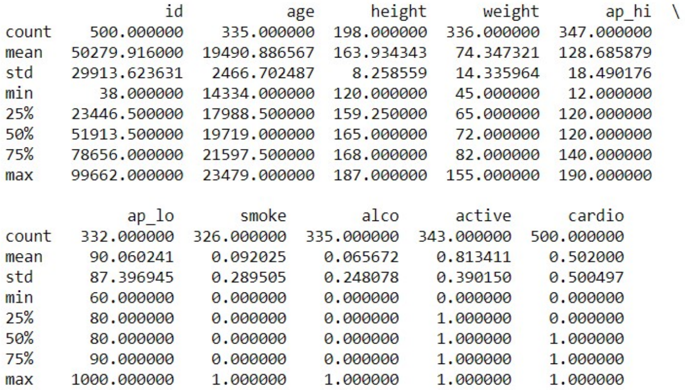
1. **Identify columns into nominal,categorical,continuous etc**
   Nominal -> id, gender, smoke, alco, cardio
   Ordinal -> cholesterol, gluc
   Interval -> height, weight, ap_hi, ap_lo
   Ratio -> age

2. **Insights of the data**
   Data.info()



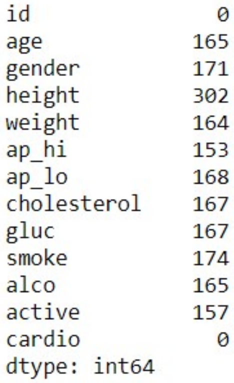
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 13 columns):
id             500 non-null int64
age            335 non-null float64
gender         329 non-null object
height         198 non-null float64
weight         336 non-null float64
ap_hi          347 non-null float64
ap_lo          332 non-null float64
cholesterol    333 non-null object
gluc           333 non-null object
smoke          326 non-null float64
alco           335 non-null float64
active         343 non-null float64
cardio         500 non-null int64
dtypes: float64(8), int64(2), object(3)
memory usage: 45.0+ KB
None
```

Data.describe()



```
                 id           age      height      weight       ap_hi  \
count    500.000000    335.000000  198.000000  336.000000  347.000000
mean   50279.916000  19490.886567  163.934343   74.347321  128.685879
std    29913.623631   2466.702487    8.258559   14.335964   18.490176
min       38.000000  14334.000000  120.000000   45.000000   12.000000
25%    23446.500000  17988.500000  159.250000   65.000000  120.000000
50%    51913.500000  19719.000000  165.000000   72.000000  120.000000
75%    78656.000000  21597.500000  168.000000   82.000000  140.000000
max    99662.000000  23479.000000  187.000000  155.000000  190.000000

             ap_lo       smoke        alco      active      cardio
count   332.000000  326.000000  335.000000  343.000000  500.000000
mean     90.060241    0.092025    0.065672    0.813411    0.502000
std      87.396945    0.289505    0.248078    0.390150    0.500497
min      60.000000    0.000000    0.000000    0.000000    0.000000
25%      80.000000    0.000000    0.000000    1.000000    0.000000
50%      80.000000    0.000000    0.000000    1.000000    1.000000
75%      90.000000    0.000000    0.000000    1.000000    1.000000
max    1000.000000    1.000000    1.000000    1.000000    1.000000
```

3. Number of null values in each column



```
id               0
age            165
gender         171
height         302
weight         164
ap_hi          153
ap_lo          168
cholesterol    167
gluc           167
smoke          174
alco           165
active         157
cardio           0
dtype: int64
```

4. Analysis of Patients

a. Oldest Person

Age of oldest person is 64.0

b. Youngest Person

Age of youngest person is 39.0

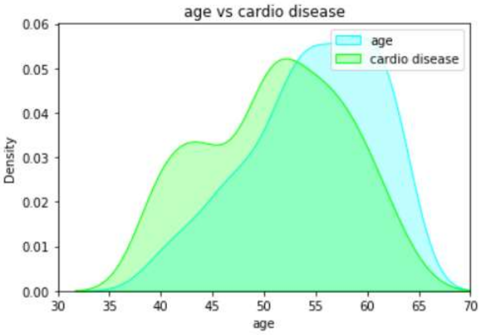
c. Average age of Person

Average age group is 52.91

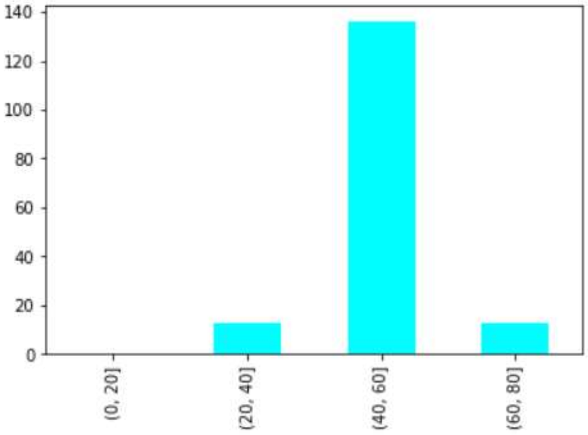
d. Median Age

Median of age is 54.00

e. Relationship between cardio and age

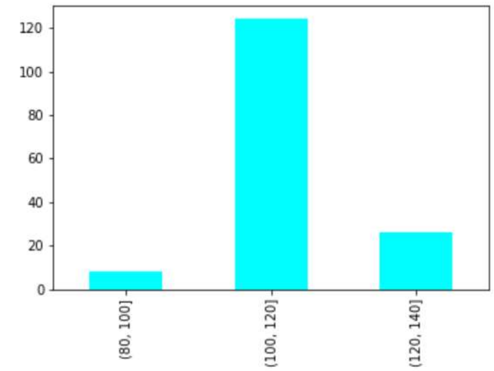


f. Age group whose survival rate is largest



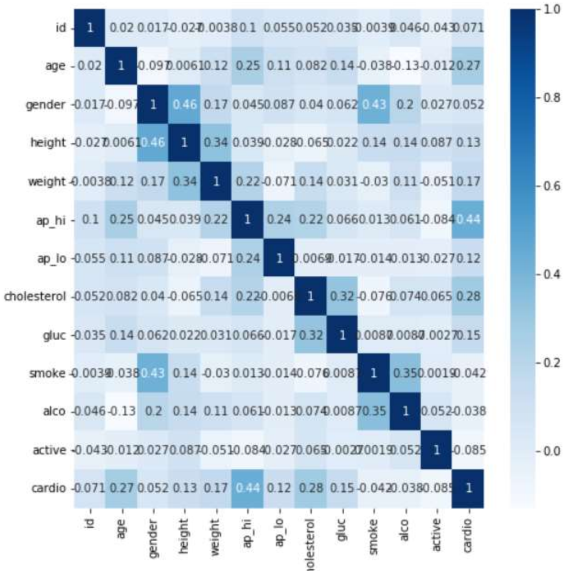
Highest survival rate is of age group 20-40 that is 81.2%

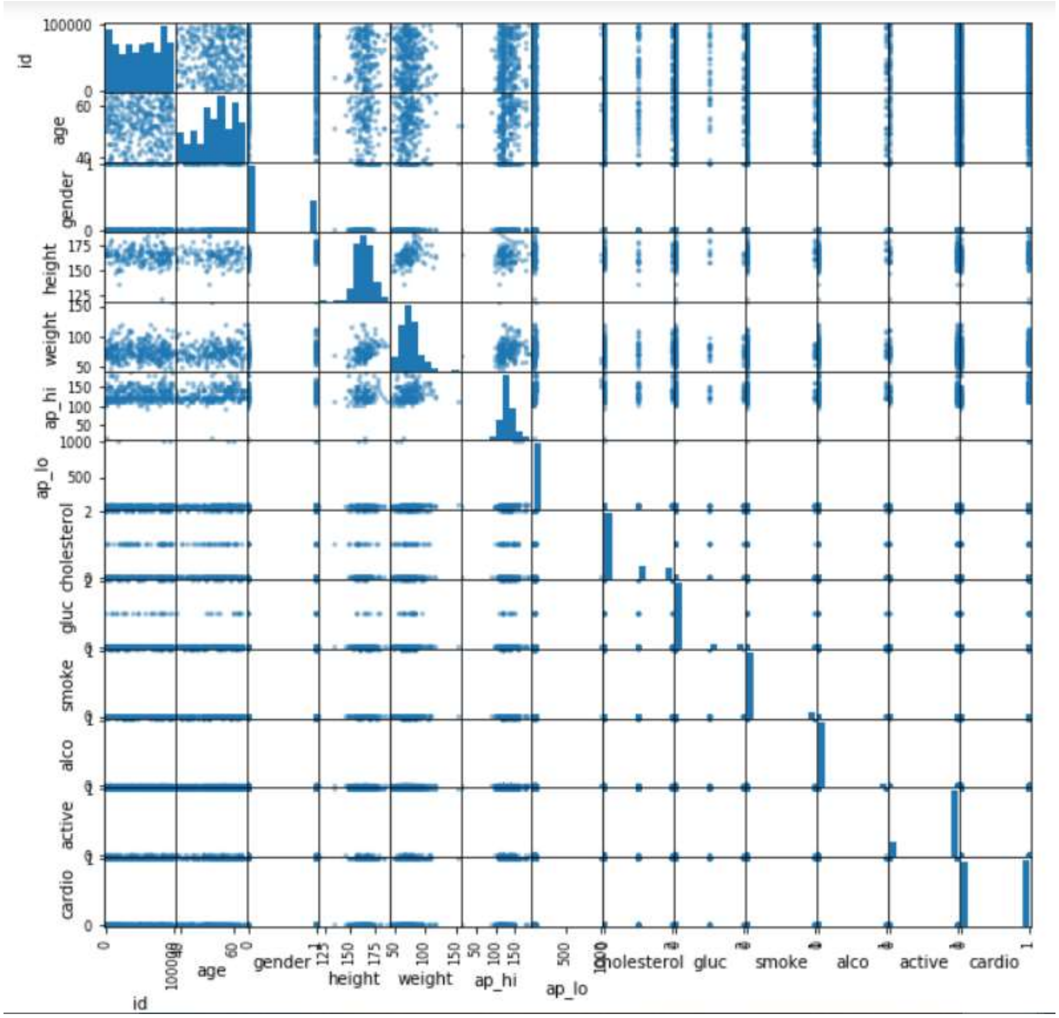
g. Similar relationship for ap_hi

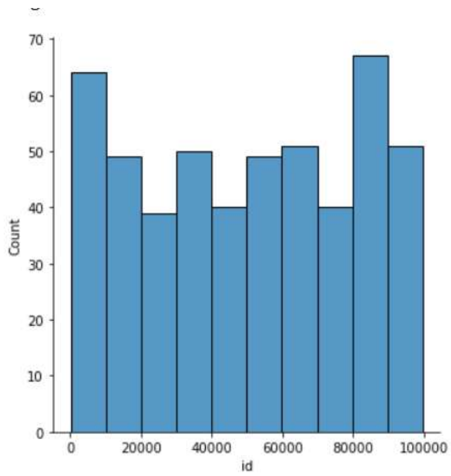


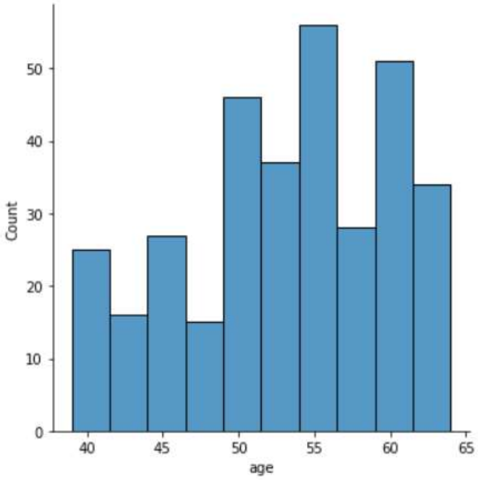
```
Survival rate of 20-40 age group is 50.00
Survival rate of 40-60 age group is 45.76
Survival rate of 60-80 age group is 54.17 ----> highest
```

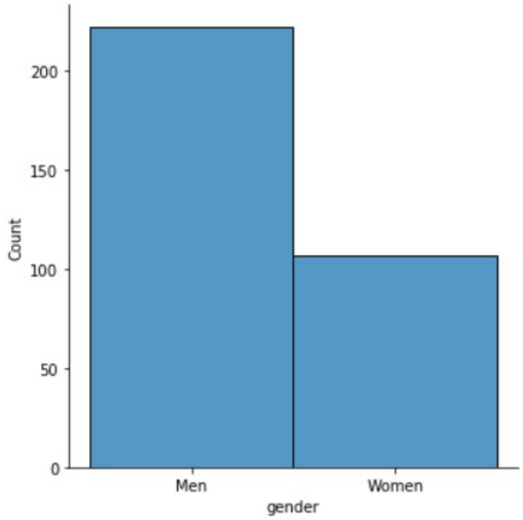
h. Visuals on data distribution




id
100000
0
age
60
40
gender
1
0
height
175
150
125
weight
150
100
50
ap_hi
150
100
50
ap_lo
1000
500
cholesterol
2
0
gluc
2
0
smoke
1
0
alco
1
0
active
1
0
cardio
1
0
0
100000
40
60
0
1
125
150
175
50
100
150
50
100
150
500
1000
id
age
gender
height
weight
ap_hi
ap_lo
cholesterol
gluc
smoke
alco
active
cardio

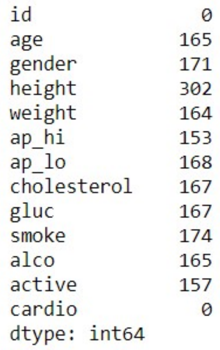






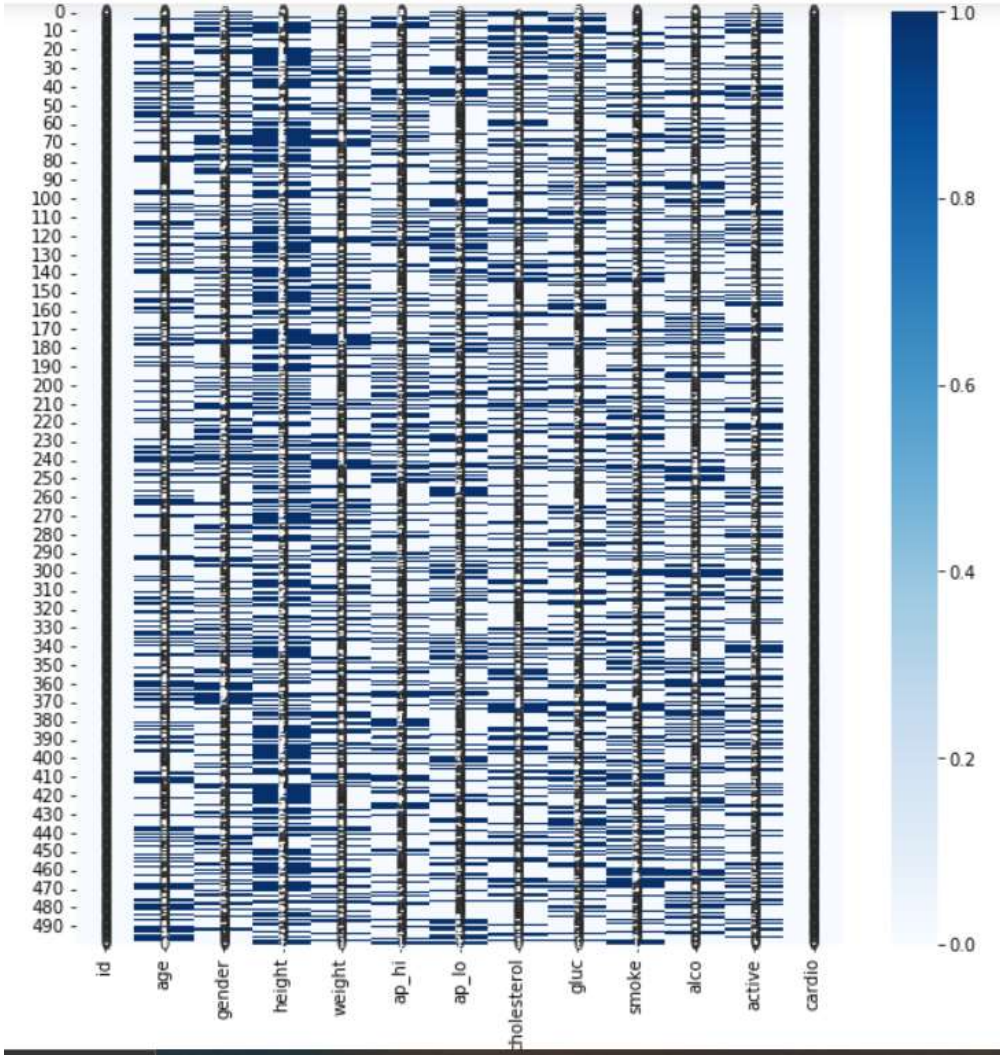
i. Missing values

- **Count missing values**



```
id               0
age            165
gender         171
height         302
weight         164
ap_hi          153
ap_lo          168
cholesterol    167
gluc           167
smoke          174
alco           165
active         157
cardio           0
dtype: int64
```

- **Heatmap of missing values**



j. **Handle missing data**

- **Use Dropna**

  If we drop all the rows containing any NA value only 7 rows are left, so the dimension of the returned dataframe will be (7,13)

- **Replace NA with mean**

To replace the null entries with mean of the column but for that we first need to replace null values with 0 and then find the mean for every column.

- **Additional techniques (pad,bfill)**
  The columns in dataset that are not numeric wont be filled using mean values hence we need to use bfill or pad method. In bfill method the null value is replaced by next valid observation, while in pad method the null value is replaced with the previous valid observation.

k. Regression Models

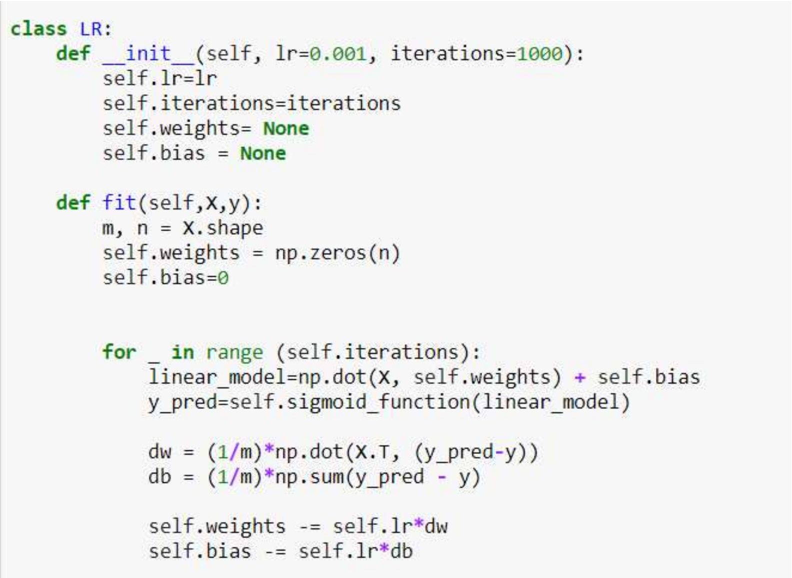
- Simple Logistic Regression
  Using only cardio-train.csv → Accuracy = 56.7 %
  Using cardio-train.csv+cardio-validation.csv → Accuracy = 65.6 %

- Feature Selection + Logistic Regression
  Using only cardio-train.csv → Accuracy = 60.8 %
  Using cardio-train.csv+cardio-validation.csv → Accuracy = 61.6 %

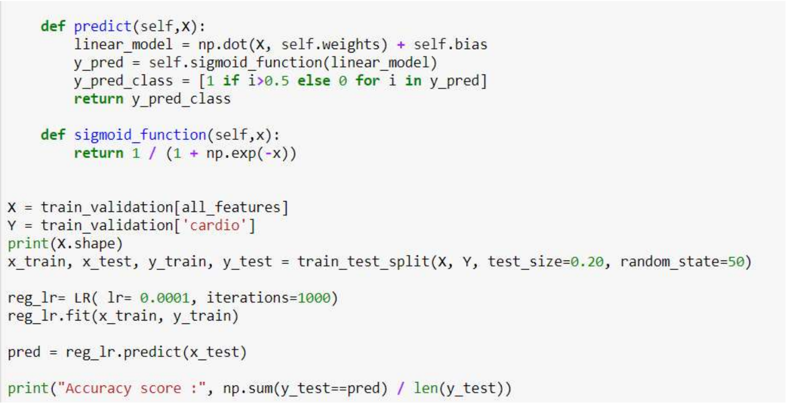
- Random Forest
  Using only cardio-train.csv → Accuracy = 68.8 %
  Using cardio-train.csv+cardio-validation.csv → Accuracy = 62.4 %

- Random Forest with Grid Search CV
  Using only cardio-train.csv → Accuracy = 65.6 %
  Using cardio-train.csv+cardio-validation.csv → Accuracy = 69.6 %
  Using cardio-train.csv+cardio-validation.csv → Accuracy = 70.4 %

l. Own implementation of Logistic Regression



```
class LR:
    def __init__(self, lr=0.001, iterations=1000):
        self.lr=lr
        self.iterations=iterations
        self.weights= None
        self.bias = None

    def fit(self,X,y):
        m, n = X.shape
        self.weights = np.zeros(n)
        self.bias=0


        for _ in range (self.iterations):
            linear_model=np.dot(X, self.weights) + self.bias
            y_pred=self.sigmoid_function(linear_model)

            dw = (1/m)*np.dot(X.T, (y_pred-y))
            db = (1/m)*np.sum(y_pred - y)

            self.weights -= self.lr*dw
            self.bias -= self.lr*db
```



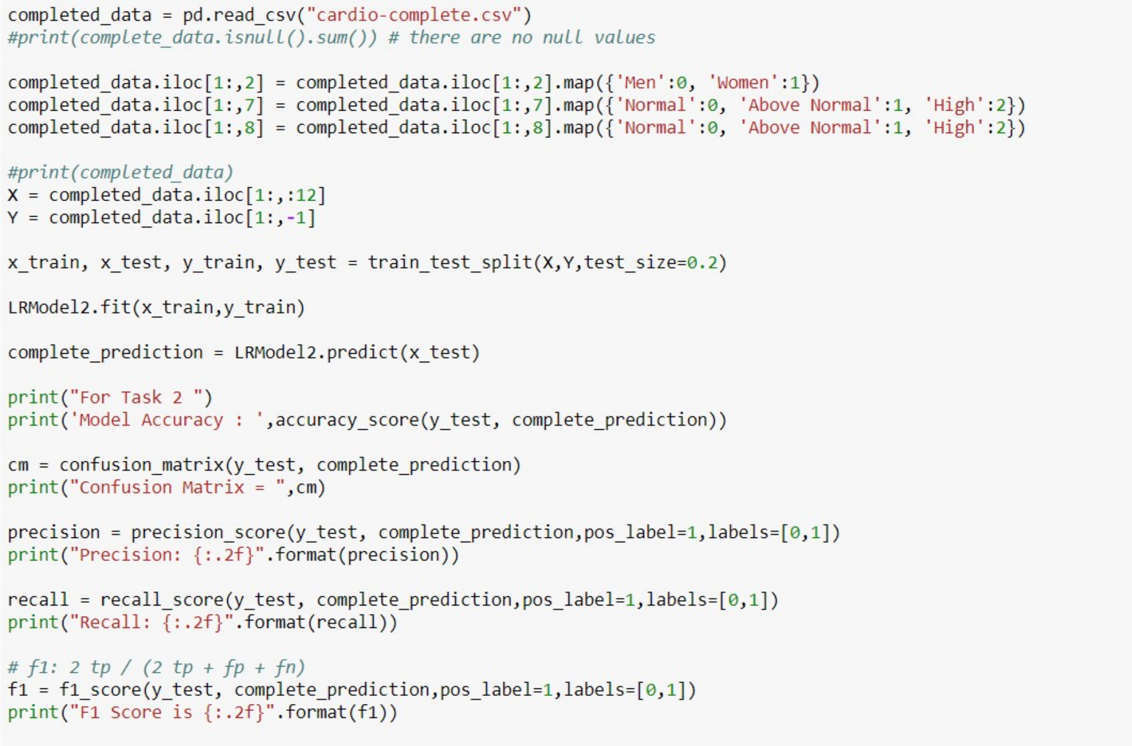
```
    def predict(self,X):
        linear_model = np.dot(X, self.weights) + self.bias
        y_pred = self.sigmoid_function(linear_model)
        y_pred_class = [1 if i>0.5 else 0 for i in y_pred]
        return y_pred_class

    def sigmoid_function(self,x):
        return 1 / (1 + np.exp(-x))


X = train_validation[all_features]
Y = train_validation['cardio']
print(X.shape)
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.20, random_state=50)

reg_lr= LR( lr= 0.0001, iterations=1000)
reg_lr.fit(x_train, y_train)

pred = reg_lr.predict(x_test)

print("Accuracy score :", np.sum(y_test==pred) / len(y_test))
```

m. Kaggle submission

First 8 submission were done by fitting the only with cardio-train.csv. Then I trained every model with cardio-train+cardio-validation and further submissions were using this.

TASK 2

Splitting cardio-complete.csv and applying regression model on it



```
completed_data = pd.read_csv("cardio-complete.csv")
#print(complete_data.isnull().sum()) # there are no null values

completed_data.iloc[1:,2] = completed_data.iloc[1:,2].map({'Men':0, 'Women':1})
completed_data.iloc[1:,7] = completed_data.iloc[1:,7].map({'Normal':0, 'Above Normal':1, 'High':2})
completed_data.iloc[1:,8] = completed_data.iloc[1:,8].map({'Normal':0, 'Above Normal':1, 'High':2})

#print(completed_data)
X = completed_data.iloc[1:,:12]
Y = completed_data.iloc[1:,-1]

x_train, x_test, y_train, y_test = train_test_split(X,Y,test_size=0.2)

LRModel2.fit(x_train,y_train)

complete_prediction = LRModel2.predict(x_test)

print("For Task 2 ")
print('Model Accuracy : ',accuracy_score(y_test, complete_prediction))

cm = confusion_matrix(y_test, complete_prediction)
print("Confusion Matrix = ",cm)

precision = precision_score(y_test, complete_prediction,pos_label=1,labels=[0,1])
print("Precision: {:.2f}".format(precision))

recall = recall_score(y_test, complete_prediction,pos_label=1,labels=[0,1])
print("Recall: {:.2f}".format(recall))

# f1: 2 tp / (2 tp + fp + fn)
f1 = f1_score(y_test, complete_prediction,pos_label=1,labels=[0,1])
print("F1 Score is {:.2f}".format(f1))
```

TASK 3

Applying feature transformation

- **Varying polynomial degree change the accuracy**
  if we choose all features ------->> for polynomial degree = 1 we are getting highest accuracy(71.5%), as we increase the degree >1 the accuracy goes on decreasing for degree=2 it was 70% for degree=3 it was 67.5%

  if we choose selected features ------>> degree=4 gives maximum accuracy of 71%

- **Is the model overfitting or underfitting**
  Min Cross validation Score is 0.575
  Mean Cross validation Score is 0.7133262976860537
  Max Cross validation Score is 0.926829268292683

  The mean cross validation score is 71%, hence we can say that the model is not overfitted