Dataset Merging for ML Model for Detecting if Patient is Sick or Healthy

Breif Description about Problem Statment and its Solution

For creating an ML model to detect is patient is sick or not we have given `4 dataset` files (`BloodPressure.csv` , `Glucose.csv` , `Oximetry.csv` , `Weight_Height.csv`). Below are the given steps which have been taken to solve the problem and build a ML model:

- **Step1:** `Merging Data Files`
  - For a single patient we have multiple set of data in all these 4 files. So for a complete set of data for a single patient, we have to merge these files according to `Patient id` which is `Patient column` and `Date Column` is also important in mergining the data files. **We will merge each patient's data which have been recorded on same date**.

- **Step2 - Step7 are completed in ML Model File**
- **Step2:** `Data Cleaning`
  - Now after step 1, we have a complete dataset, but now we have some missing entries and outliers in the dataset. So for this purpose we will clean the dataset. We will first compute and remove the missing values and then we will check the outliers. For checking outliers we will use `Z function` and for outliers visualization we will be using `box plot` provided by `seaborn library` .

- **Step3:** `Feature Analysis`
  - In this step we will use correlation analysis to analyze all the features which are given in the dataset. We have **removed 2 features** after correlation analysis which are `Patient` and `Date` , they both have same values on different rows which effected their correlation analysis result very badly. So after deleting them we will be using 9 features for Training and Testing the Model.

- **Step4:** `K-means Clustring`
  - As you know we don't have label/target/class of data. So in this case it is best to use unsupervised learning models so we can detect the clusters in the dataset. Each cluster will have different class/label. In our case we have 2 different classes to detect `Sick` and `Healthy` , so we have 2 clusters of data.

- **Step5:** `Neural Network`
  - After applying K-means clustring we got 2 clusters of data and then we merged the class/label of each cluster with the dataset. Now we have a dataset with class/label, so we will train `Neural Networks` model to detect which patient is healthy/sick. Before applying `Neural Networks` we will scale and convert the data and we will split the dataset into training and teseting subsets.
  - We will define a `Neural Network` architecture which we will use to detect healthy/sick patients.
  - We will use `cross validation` technique to prevent overfitting.
  - After all these steps we will train and test the model.

- **Step6:** `Saving information`
  - For our last presentation step we will need to save some of the information in files. We will store the `Neural Network model` to predict the classes, `Training and Testing scores` , `Each Feature Maximum value` which will be used to scale the data when we will predict the classes.

- **Step7:** `Model Presentation`
  - For model presentation we will be using `Dash by Plotly` . Dash is python library same as Shiny which is for R language. Model presentation files are provided in `Model Presentation Folder` .

Importing Libraries

For data merging we will only use `pandas` library.

In [1]:

```
import pandas as pd
```

Reading data

We will read all the data files using `.read_csv()` function

In [2]:

```
bp_data = pd.read_csv('bloodPressure.csv')
gl_data = pd.read_csv('Glucose.csv')
ox_data = pd.read_csv('Oximetry.csv')
wh_data = pd.read_csv('Weight_Height.csv')
```

Data Preprocessing before merging

As we know in all datasets we have a column `Date` . In `Date` column we have a mixture of **date and time**. If we use both **date and time** format for `Date` column then it will be a lot difficult to merge all these files together. So for that purpose we will only use **date** and we will delete the **time** for each entry and then we will sae it again.

In the given cell below we will iterate over each row in the dataset by **for loop**, then we will only select `Date` column and split the information inside it by a `space` . So now in the index `0` we have data and on index `1` we have time. So we will just save the `0` index value which is only date. We will do that step for each dataset.

In [3]:

```
for index, data in bp_data.iterrows():
    date = data["Date"]
    splited_date = date.split(' ')
    bp_data.at[index, "Date"] = splited_date[0]

bp_data
```

Out[3]:

	Patient	Systolic	Diastolic	AvBloodPressure	HeartRate	Date
0	1369	113	93	93	109	1/1/2013
1	1410	91	87	110	99	1/1/2013
2	1156	91	58	92	93	1/1/2013
3	663	114	73	121	62	1/1/2013
4	1198	100	60	99	61	1/1/2013
...	...	...	...	...	...	...
10162	81	111	78	85	78	9/18/2015
10163	81	104	66	88	91	9/18/2015
10164	81	106	69	85	101	9/18/2015
10165	8	114	81	96	72	9/21/2015
10166	8	110	78	90	79	9/23/2015

10167 rows × 6 columns

In [4]:

```
for index, data in gl_data.iterrows():
    date = data["Date"]
    splited_date = date.split(' ')
    gl_data.at[index, "Date"] = splited_date[0]

gl_data
```

Out[4]:

	Patient	Glucose	Date
0	1369	61	1/1/2013
1	1410	82	1/1/2013
2	1156	89	1/1/2013
3	663	68	1/1/2013
4	1198	135	1/1/2013
...	...	...	...
10069	81	89	9/11/2015
10070	8	101	9/21/2015
10071	8	75	9/21/2015
10072	81	110	9/21/2015
10073	8	201	9/23/2015

10074 rows × 3 columns

In [5]:

```
for index, data in ox_data.iterrows():
    date = data["Date"]
    splited_date = date.split(' ')
    ox_data.at[index, "Date"] = splited_date[0]

ox_data
```

Out[5]:

	Patient	SpO2	HeartRate	Date
0	1369	96	73	1/1/2013
1	1410	73	61	1/1/2013
2	1156	85	57	1/1/2013
3	663	66	87	1/1/2013
4	1198	88	75	1/1/2013
...	...	...	...	...
9867	81	90	85	9/18/2015
9868	81	93	71	9/18/2015
9869	81	97	99	9/18/2015
9870	81	89	66	9/21/2015
9871	8	94	76	9/23/2015

9872 rows × 4 columns

In [6]:

```
for index, data in wh_data.iterrows():
    date = data["Date"]
    splited_date = date.split(' ')
    wh_data.at[index, "Date"] = splited_date[0]

wh_data
```

Out[6]:

	Patient	Weight	Height	IMC	Date
0	1369	94.89	1.59	37.53	1/1/2013
1	1410	64.56	1.67	23.15	1/1/2013
2	1156	119.04	1.49	53.62	1/1/2013
3	663	61.56	1.86	17.79	1/1/2013
4	1198	50.42	1.57	20.46	1/1/2013
...	...	...	...	...	...
10165	8	72.60	1.82	21.92	9/21/2015
10166	8	72.60	1.82	21.92	9/21/2015
10167	81	72.60	1.82	21.92	9/21/2015
10168	8	73.20	1.67	26.25	9/23/2015
10169	81	82.00	1.72	27.72	9/23/2015

10170 rows × 5 columns

Now after manipulating `Date` column, we will group the dataset by `Patient id` , we will do that with the help of `groupby()` function. Grouping the data with `Patient id` will help us alot when we will merge the datasets in a new dataframe.

In [7]:

```
grouped_bp_data = bp_data.groupby("Patient")
grouped_gl_data = gl_data.groupby("Patient")
grouped_ox_data = ox_data.groupby("Patient")
grouped_wh_data = wh_data.groupby("Patient")
```

In the cell below we will create a new dataframe for saving the merged datasets

In [8]:

```
# Taking only bp heart_rate, dropping oximetry heart_rate
merged_df = pd.DataFrame(columns=['Date', 'Patient', 'Systolic', 'Diastolic', 'AvBl

merged_df
```

Out[8]:

Date	Patient	Systolic	Diastolic	AvBloodPressure	HeartRate	Glucose	SpO2	Weight	Heig

These are helper functions which will help us in adding a new entry in new dataframe

In [9]:

```
def add_bp_data(bp_data, row_index):
    merged_df.at[row_index, "Date"] = bp_data[1]["Date"]
    merged_df.at[row_index, "Patient"] = bp_data[1]["Patient"]
    merged_df.at[row_index, "Systolic"] = bp_data[1]["Systolic"]
    merged_df.at[row_index, "Diastolic"] = bp_data[1]["Diastolic"]
    merged_df.at[row_index, "AvBloodPressure"] = bp_data[1]["AvBloodPressure"]
    merged_df.at[row_index, "HeartRate"] = bp_data[1]["HeartRate"]

def add_gl_data(gl_data, row_index):
    merged_df.at[row_index, "Date"] = gl_data[1]["Date"]
    merged_df.at[row_index, "Glucose"] = gl_data[1]["Glucose"]

def add_ox_data(ox_data, row_index):
    merged_df.at[row_index, "Date"] = ox_data[1]["Date"]
    merged_df.at[row_index, "SpO2"] = ox_data[1]["SpO2"]

def add_wh_data(wh_data, row_index):
#     print(wh_data)/
    merged_df.at[row_index, "Date"] = wh_data[1]["Date"]
    merged_df.at[row_index, "Weight"] = wh_data[1]["Weight"]
    merged_df.at[row_index, "Height"] = wh_data[1]["Height"]
    merged_df.at[row_index, "IMC"] = wh_data[1]["IMC "]
```

The cell below contains the logic for merging the datasets. I have merged the data for each patient by Date. I will explain that in steps so you will understand it in a better way:

- **Step1**: We have group all the datasets with `Patient id` , now first we will iterate over the groups of Patient data by the help of for loop. I'm using zip function so then I can take one group from each dataset at a time.
- **Step2**: After applying step 1 we have 4 groups of Patient data, and these 4 groups will be of same Patient, Like they all have same `pateint id` , so now we will iterate over each single entry in the group, again I'm using zip function to iterate over multiple entries at the same time. You will notice that in this step I'm using itertools.zip function, this function will help us to iterate over the data even if all the groups are not of same length.
- **Step3**: When Iterating over the data notice that we don't have same length for each group for his purpose we have to check if the `data == None` or `data != None` , you will notice that we have many if conditions to fulfill that None condition. We are dealing with a combination of 4 data variables at the same time that's why we have too much if conditions
- **Step4**: After checking the None condition now it's time to check if all the available data variables have same `date` , if that is the case then we will simply add them together without increasing the row_index, if that was not the case we will store them one by one and also we will increase the row_index after saving each one.

In [11]:

```
import itertools
row_index = 0

for bp_group, gl_group, ox_group, wh_group in zip(grouped_bp_data, grouped_gl_data,
    for bp_data, gl_data, ox_data, wh_data in itertools.zip_longest(bp_group[1].ite
#       for bp
#       add only bp data in row
        if bp_data != None and gl_data == None and ox_data == None and wh_data == N
            add_bp_data(bp_data, row_index)

#       add bp and gl data
        elif bp_data != None and gl_data != None and ox_data == None and wh_data ==
            if bp_data[1]["Date"] == gl_data[1]["Date"]:
                add_bp_data(bp_data, row_index)
                add_gl_data(gl_data, row_index)
            else:
                add_bp_data(bp_data, row_index)
                row_index = row_index + 1
                add_gl_data(gl_data, row_index)
#       add bp and ox data
        elif bp_data != None and gl_data == None and ox_data != None and wh_data ==
            if bp_data[1]["Date"] == ox_data[1]["Date"]:
                add_bp_data(bp_data, row_index)
                add_ox_data(ox_data, row_index)
            else:
                add_bp_data(bp_data, row_index)
                row_index = row_index + 1
                add_ox_data(ox_data, row_index)

#       add bp and wh data
        elif bp_data != None and gl_data == None and ox_data == None and wh_data !=
            if bp_data[1]["Date"] == wh_data[1]["Date"]:
                add_bp_data(bp_data, row_index)
                add_wh_data(wh_data, row_index)
            else:
                add_bp_data(bp_data, row_index)
                row_index = row_index + 1
                add_wh_data(wh_data, row_index)

#       add bp, gl ad ox data
        elif bp_data != None and gl_data != None and ox_data != None and wh_data ==
            if bp_data[1]["Date"] == gl_data[1]["Date"] and gl_data[1]["Date"] == o
                add_bp_data(bp_data, row_index)
                add_gl_data(gl_data, row_index)
                add_ox_data(ox_data, row_index)
            else:
                add_bp_data(bp_data, row_index)
                if bp_data[1]["Date"] == gl_data[1]["Date"]:
                    add_gl_data(gl_data, row_index)
                if bp_data[1]["Date"] == ox_data[1]["Date"]:
                    add_ox_data(ox_data, row_index)
                if gl_data[1]["Date"] == ox_data[1]["Date"]:
                    row_index = row_index+1
                    add_gl_data(gl_data, row_index)
                    add_ox_data(ox_data, row_index)

#       add bp, gl, and wh data
        elif bp_data != None and gl_data != None and ox_data == None and wh_data !=
            if bp_data[1]["Date"] == gl_data[1]["Date"] and gl_data[1]["Date"] == w
```

```
                add_bp_data(bp_data, row_index)
                add_gl_data(gl_data, row_index)
                add_wh_data(wh_data, row_index)
            else:
                add_bp_data(bp_data, row_index)
                if bp_data[1]["Date"] == gl_data[1]["Date"]:
                    add_gl_data(gl_data, row_index)
                if bp_data[1]["Date"] == wh_data[1]["Date"]:
                    add_wh_data(wh_data, row_index)
                if gl_data[1]["Date"] == wh_data[1]["Date"]:
                    row_index = row_index+1
                    add_gl_data(gl_data, row_index)
                    add_wh_data(wh_data, row_index)

#       add bp, ox, and wh data
        elif bp_data != None and gl_data == None and ox_data != None and wh_data !=
            if bp_data[1]["Date"] == ox_data[1]["Date"] and ox_data[1]["Date"] == w
                add_bp_data(bp_data, row_index)
                add_ox_data(ox_data, row_index)
                add_wh_data(wh_data, row_index)
            else:
                add_bp_data(bp_data, row_index)
                if bp_data[1]["Date"] == ox_data[1]["Date"]:
                    add_ox_data(ox_data, row_index)
                if bp_data[1]["Date"] == wh_data[1]["Date"]:
                    add_wh_data(wh_data, row_index)
                if ox_data[1]["Date"] == wh_data[1]["Date"]:
                    row_index = row_index+1
                    add_ox_data(ox_data, row_index)
                    add_wh_data(wh_data, row_index)

#       for ox
#       add only ox data
        elif bp_data == None and gl_data == None and ox_data != None and wh_data ==
            add_ox_data(ox_data, row_index)

#       add ox and gl data
        elif bp_data == None and gl_data != None and ox_data != None and wh_data ==
            if ox_data[1]["Date"] == gl_data[1]["Date"]:
                add_ox_data(ox_data, row_index)
                add_gl_data(gl_data, row_index)
            else:
                add_ox_data(ox_data, row_index)
                row_index = row_index + 1
                add_gl_data(gl_data, row_index)

#       add ox and wh data
        elif bp_data == None and gl_data == None and ox_data != None and wh_data !=
            if ox_data[1]["Date"] == wh_data[1]["Date"]:
                add_ox_data(ox_data, row_index)
                add_wh_data(wh_data, row_index)
            else:
                add_ox_data(ox_data, row_index)
                row_index = row_index + 1
                add_wh_data(wh_data, row_index)

#       add gl, ox, and wh data
        elif bp_data == None and gl_data != None and ox_data != None and wh_data !=
            if gl_data[1]["Date"] == ox_data[1]["Date"] and ox_data[1]["Date"] == w
                add_gl_data(gl_data, row_index)
                add_ox_data(ox_data, row_index)
```

```
                add_wh_data(wh_data, row_index)
            else:
                add_ox_data(ox_data, row_index)
                if gl_data[1]["Date"] == ox_data[1]["Date"]:
                    add_gl_data(gl_data, row_index)
                if ox_data[1]["Date"] == wh_data[1]["Date"]:
                    add_wh_data(wh_data, row_index)
                if gl_data[1]["Date"] == wh_data[1]["Date"]:
                    row_index = row_index+1
                    add_gl_data(gl_data, row_index)
                    add_wh_data(wh_data, row_index)

#       for gl
#       add only gl data
        elif bp_data == None and gl_data != None and ox_data == None and wh_data ==
            add_gl_data(gl_data, row_index)
#       add gl, wh data
        elif bp_data == None and gl_data != None and ox_data == None and wh_data !=
            if gl_data[1]["Date"] == wh_data[1]["Date"]:
                add_gl_data(gl_data, row_index)
                add_wh_data(wh_data, row_index)
            else:
                add_gl_data(gl_data, row_index)
                row_index = row_index + 1
                add_wh_data(wh_data, row_index)
#       for wh
#       add only wh data
        elif bp_data == None and gl_data == None and ox_data == None and wh_data !=
            add_wh_data(wh_data, row_index)
#       add all data variables
        else:
            add_bp_data(bp_data, row_index)
            add_gl_data(gl_data, row_index)
            add_ox_data(ox_data, row_index)
            add_wh_data(wh_data, row_index)

        row_index = row_index + 1
```

After merging the data we have a new dataframe `merged_df`

In [12]:

```
merged_df
```

Out[12]:

	Date	Patient	Systolic	Diastolic	AvBloodPressure	HeartRate	Glucose	SpO2	W
0	2/24/2014	1	93	69	75	95	78	97	
1	8/29/2014	1	108	76	87	81	92	96	
2	11/13/2014	1	96	73	81	100	150	95	
3	4/30/2015	1	98	74	83	87	86	96	
4	4/30/2015	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
...	...	...	...	...	...	...	...	...	
12139	8/13/2013	1444	120	60	142	60	150	66	
12140	1/1/2015	1444	117	50	102	100	137	78	
12141	1/5/2015	1444	139	74	136	75	83	70	
12142	1/9/2015	1444	92	82	99	74	119	78	
12143	1/13/2015	1444	106	64	124	85	148	67	

12144 rows × 11 columns

Now in the cell below we will simply store the dataframe into the .csv file

In [13]:

```
merged_df.to_csv('merged_df.csv')
```

In [ ]:

```

```