I : Learning the DataSet

Preface:

We have a diabetes dataset and we want to know the basic information about it. The volumn of the dataset, the meaning of each column, etc.

Steps:

1. Import the dataset using pd.read_csv( )

```
import pandas as pd
df = pd.read_csv("diabetes.csv")
✓ 0.2s
```

figure 1.1: Import the dataset

2. Basic infomation

a) According to df.shape, we know we have 768 instances of the diabetes dataset, each instance contains 9 attributes.

```
df.shape
✓ 0.7s
(768, 9)
```

figure 1.2.a: Shape of the dataset

b) Using df.info( ) to get a close look of each columns. We know the name/ type/ count/ non-null of each columns. About non-null we will mention the details in the preprocessing part.

```
df.info()
✓ 0.7s
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               768 non-null    int64
 1   Glucose                   768 non-null    int64
 2   BloodPressure             768 non-null    int64
 3   SkinThickness             768 non-null    int64
 4   Insulin                   768 non-null    int64
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64
 8   Outcome                   768 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

figure 1.2.b: df.info( )

3. The meaning of the each columns

- Pregnancies: Number of times pregnant
- Glucose: Plasma glucose concentration a 2 hours in an oral glucose tolerance test
- BloodPressure: Diastolic blood pressure (mm Hg)
- SkinThickness: Triceps skin fold thickness (mm)
- Insulin: 2-Hour serum insulin (mu U/ml)
- BMI: Body mass index (weight in kg/(height in m)^2)
- DiabetesPedigreeFunction: Diabetes pedigree function
- Age: Age (years)
- Outcome: Class variable (0 or 1, class value 1 is interpreted as "tested positive for diabetes")

4. Extra information from the internet

- This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases. The objective is to predict based on diagnostic measurements whether a patient has diabetes.
- Several constraints were placed on the selection of these instances from a larger database. In particular, all patients here are females at least 21 years old of Pima Indian heritage.

II : More Details and Preprocessing the Dataset

Preface:

Now, we know the basic information about the dataset, however, we haven't look at the details about the dataset. We gonna dig in to the real values in the dataset and see something like abnormal value, outliers, max, min, range, etc. Basically anything we find interesting.

Does it have abnormal data? Does it have meaningless / missing data? If it has, we want to replace / process them.

Steps:

1. Look at the real value in the dataset and recognize abnormal / rare values:

we predefine three functions:
First is print the number of 0 value in the column.
Second is print the value range of the column without taking 0 value into account.
Third is plot the distribution of the column without taking account the missing value.

```
# define a method that return the range of a columns without take 0 into account
def column_range_without0(column):
    a = column[column != 0]
    print(column.name + "'s range = " + "(" + str(a.min()) + ", " + str(a.max()) + ")")
```

✓ 0.5s

```
# define a method that return the number of 0 value in the column
def number_of_0_value(column):
    print("Number of missing value = " + str((column == 0).sum()))
```

✓ 0.7s

```
# define a method that plot the column distribution
import seaborn as sns
def plot_column_distribution(column):
    a = column
    if (a.name != 'Pregnancies'):
        a = a[a != 0]
    fig = sns.histplot(a)
    fig.set(xlabel = (a.name + "'s Range"), title = (a.name + " value distribution histogram"))
```

✓ 0.6s

figure 2.1: predefine functions

a) Pregnancies: Some femal conceive more than 10 times. It's possibile, however, it is unusual, we don't know if this will affect the outcome? The maximum is 17 times! And in the dataset, there are 111 woman never pregnant. It's similar to a exponential distribution.

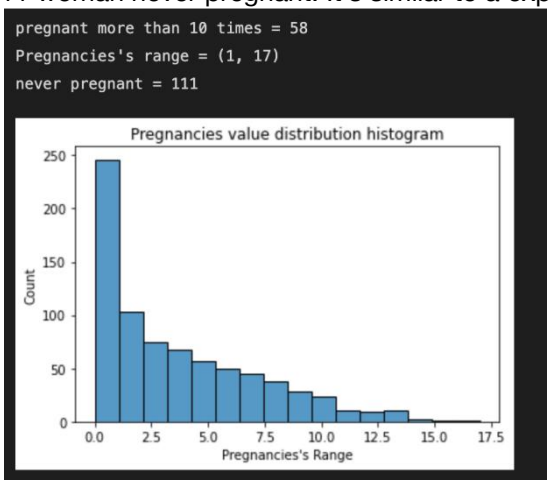


figure 2.1.a: Pregnant details information

b) Glucose: There are just 5 missing value in the Glucose column, and the range of Glucose is (44,199).

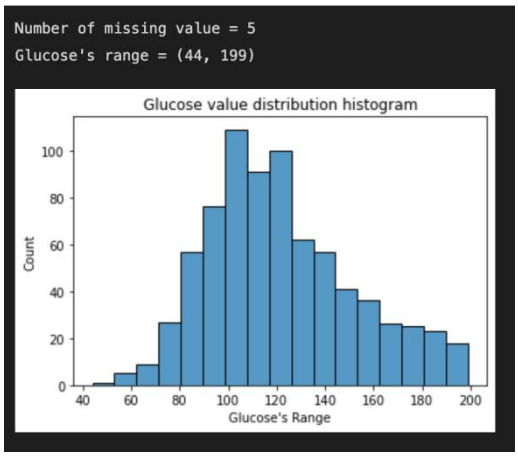


figure 2.1.b: Glucose detials information

c) BloodPressure: 35 missing values, range from (24, 122)

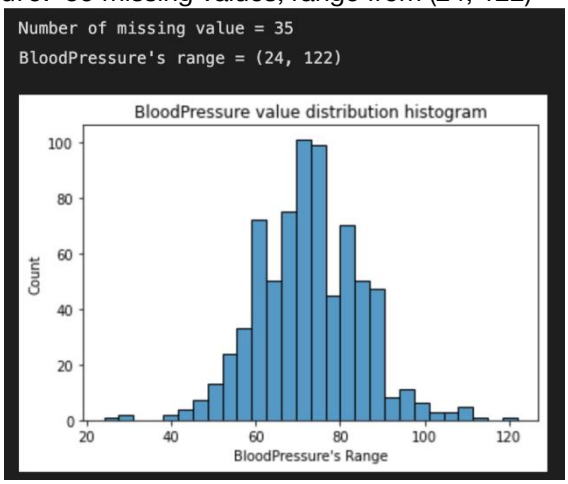


figure 2.1.c: Blood Pressure details information

d) SkinThickness: 227 missing data, this column miss a lot of data, maybe because is more complicated to measure the skin thickness? But for sure, no one can alive with 0 skin thickness.

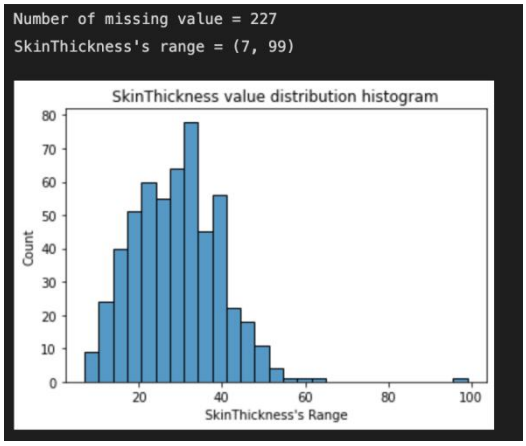


figure 2.1.d: Skin Thickness details information

e) Insulin: This one is the very tricky. We need to understand the type of diabetes and a normal person can never have zero insulin(sometimes lower than 3 if hungry but never go to 0). There are three types of diabetes.(mu U/ml)

Type 1 diabetes is thought to be caused by an autoimmune reaction (the body attacks itself by mistake). This reaction stops your body from making insulin. So in this case, the patient can have 0 in insulin with the outcome equals to 1.

In type 2 diabetes, the pancreas makes insulin, but the cells don't respond to it as they should. This is called insulin resistance. When glucose can't get into cells, the blood sugar level rises. Then the pancreas works harder to make even more insulin.

Gestational diabetes develops in pregnant women who have never had diabetes. If you have gestational diabetes, your baby could be at higher risk for health problems. Gestational diabetes usually goes away after your baby is born. However, it increases your risk for type 2 diabetes later in life. Your baby is more likely to have obesity as a child or teen and develop type 2 diabetes later in life.

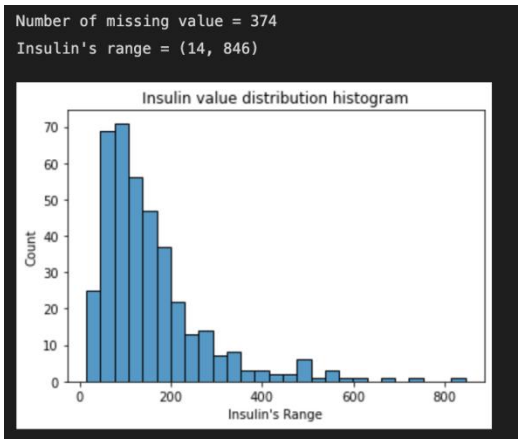


figure 2.1.e1: Wrong Insulin details information

The missing value is when 'Insulin' == 0 and 'Outcome' == 0. When 'Insulin' == 0 and 'Outcome' == 1, 'Insulin' == 0 is not missing data, is the data from type1 diabetes patients. So we drop the missing data and plot the real Insulin data. It can be very easy to ignore this part and handle the wrong data.

```
df_RealDataInsulin = df['Insulin'].drop(df.query('Insulin == 0 & Outcome != 1').index)
fig = sns.histplot(df_RealDataInsulin)
fig.set(xlabel = 'Real Insulin Value', title = ' Real Insulin distribution histogram')
```

✓ 0.4s

```
[Text(0.5, 0, 'Real Insulin Value'),
 Text(0.5, 1.0, ' Real Insulin distribution histogram')]
```

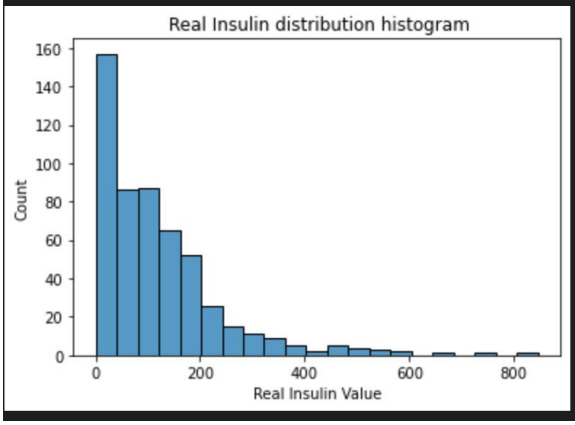


figure 2.1.e2: Right Insulin details information

f) BMI: 11 missing value. range from (18.2, 67.1)

```
Number of missing value = 11
BMI's range = (18.2, 67.1)
```

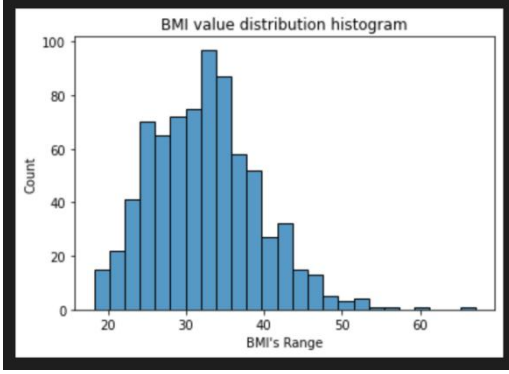


figure 2.1.f: BMI details information

g) DiabetesPedigreeFunction: 0 missing value

```
Number of missing value = 0
DiabetesPedigreeFunction's range = (0.078, 2.42)
```

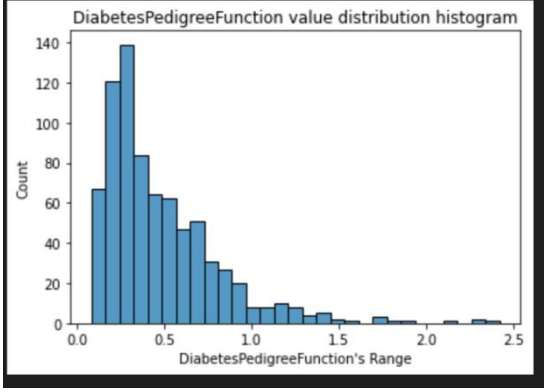


figure 2.1.g:Diabetes Pedifree Function details information

h) Age: 0 missing value

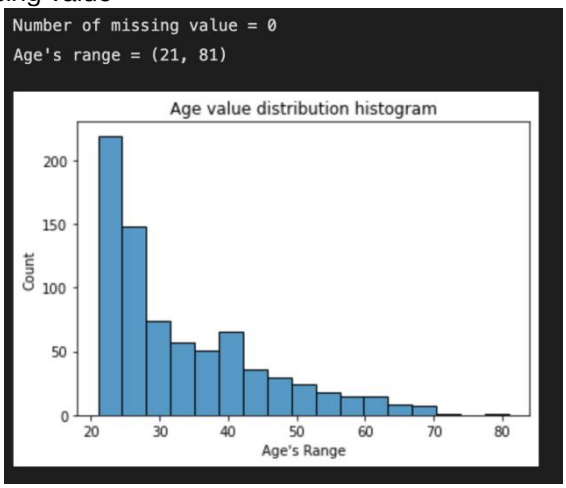


figure 2.1.h: Age details information

i) Outcome: 500 have no diabetes while 268 hve diabetes.

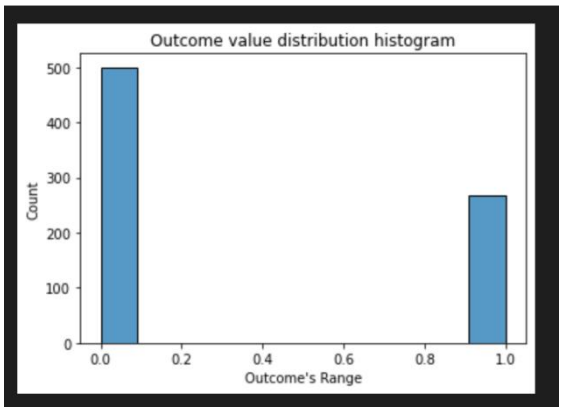


figure 2.1.i: Outcome details information

2. Replace the value using different imputation method :

The columns with the missing data are Glucose(5), Blood Presure(35), Skin Thickness(227), Insulin(236), BMI(11).

a) replace the 0 value to np.NaN:

```
import numpy as np
# Define a method to replace the missing value in the pandas.serials column from 0 to np.NaN
# return the column
def replace0toNaN(column):
    column = column.replace(0,np.NaN)
    return column
```

```
# look at the replace dataframe first
df.info()
```
✓ 0.9s

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               768 non-null    int64
 1   Glucose                   763 non-null    float64
 2   BloodPressure             733 non-null    float64
 3   SkinThickness             541 non-null    float64
 4   Insulin                   532 non-null    float64
 5   BMI                       757 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64
 8   Outcome                   768 non-null    int64
dtypes: float64(6), int64(3)
memory usage: 54.1 KB
```

figure 2.2.a: replace missing data from 0 to np.NaN

b) use KNN Imputer to replace the missing values:

```
import numpy as np
from sklearn.impute import KNNImputer

# using default settings, n_neighbors = 5, weights = uniform
knn_imputer = KNNImputer()
imputation_npnd = knn_imputer.fit_transform(df)
imputation_npnd
```
✓ 0.2s

```
array([[  6.   , 148.   ,  72.   , ...,   0.627,  50.   ,   1.   ],
       [  1.   ,  85.   ,  66.   , ...,   0.351,  31.   ,   0.   ],
       [  8.   , 183.   ,  64.   , ...,   0.672,  32.   ,   1.   ],
       ...,
       [  5.   , 121.   ,  72.   , ...,   0.245,  30.   ,   0.   ],
       [  1.   , 126.   ,  60.   , ...,   0.349,  47.   ,   1.   ],
       [  1.   ,  93.   ,  70.   , ...,   0.315,  23.   ,   0.   ]])
```

figure 2.2.b: Using KNNImputer

c) convert numpy.ndarray to pandas dataframe

```
# convert numpy.ndarray to pandas dataframe
imputation_df = pd.DataFrame(imputation_npnd, columns=df.columns)
imputation_df
```

✓ 0.1s Python

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6.0	148.0	72.0	35.0	0.0	33.6	0.627	50.0	1.0
1	1.0	85.0	66.0	29.0	58.6	26.6	0.351	31.0	0.0
2	8.0	183.0	64.0	33.0	0.0	23.3	0.672	32.0	1.0
3	1.0	89.0	66.0	23.0	94.0	28.1	0.167	21.0	0.0
4	0.0	137.0	40.0	35.0	168.0	43.1	2.288	33.0	1.0
...	...	...	...	...	...	...	...	...	...
763	10.0	101.0	76.0	48.0	180.0	32.9	0.171	63.0	0.0
764	2.0	122.0	70.0	27.0	127.0	36.8	0.340	27.0	0.0
765	5.0	121.0	72.0	23.0	112.0	26.2	0.245	30.0	0.0
766	1.0	126.0	60.0	33.2	0.0	30.1	0.349	47.0	1.0
767	1.0	93.0	70.0	31.0	66.6	30.4	0.315	23.0	0.0

768 rows × 9 columns

figure 2.2.c: new replaced DataFrame

3. Plot the distribution of each columns of new replaced DataFrame ：

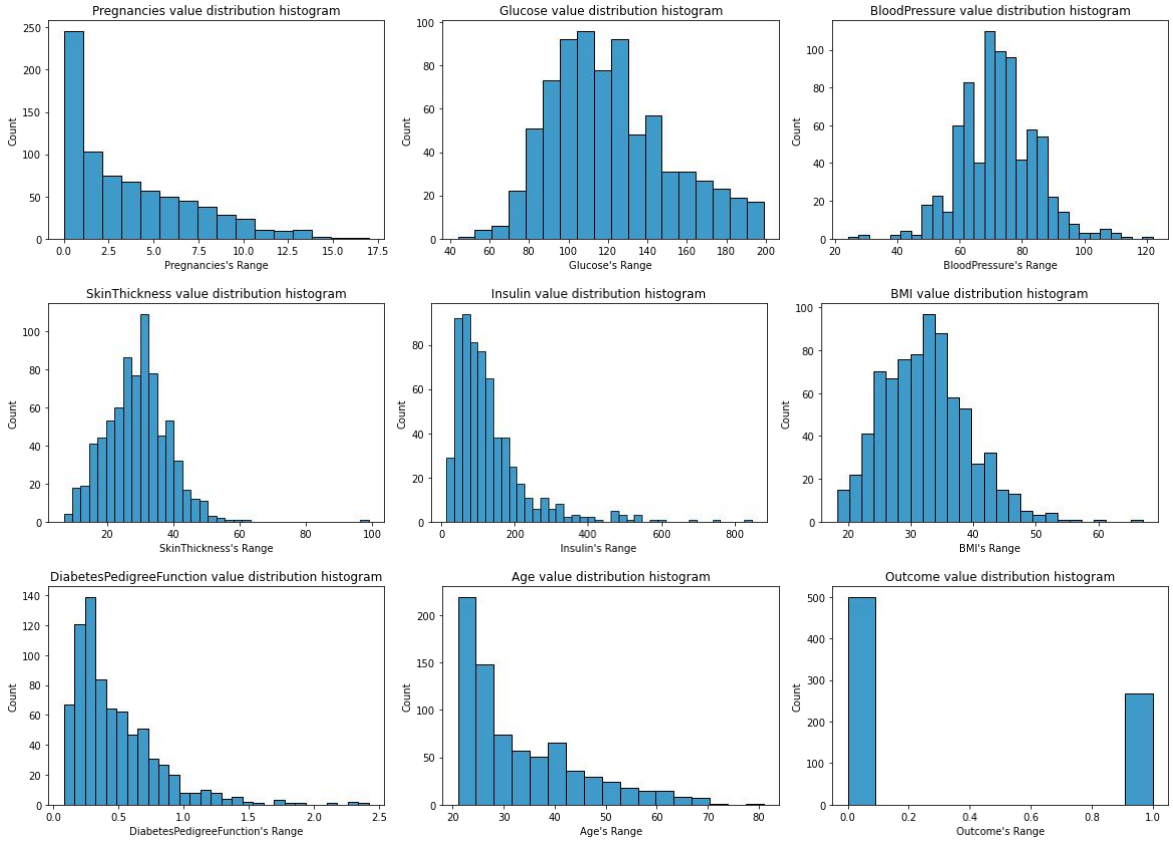


figure2.3: new replaced DataFrame each column’s histogram

4. Find the relationship of each columns :

a) HeatMap using plotly. As you can see in the plot, (Age, Pregnancies), (Glucose,Outcome), (BMI, SkinThickness) have good positive linear relationship.

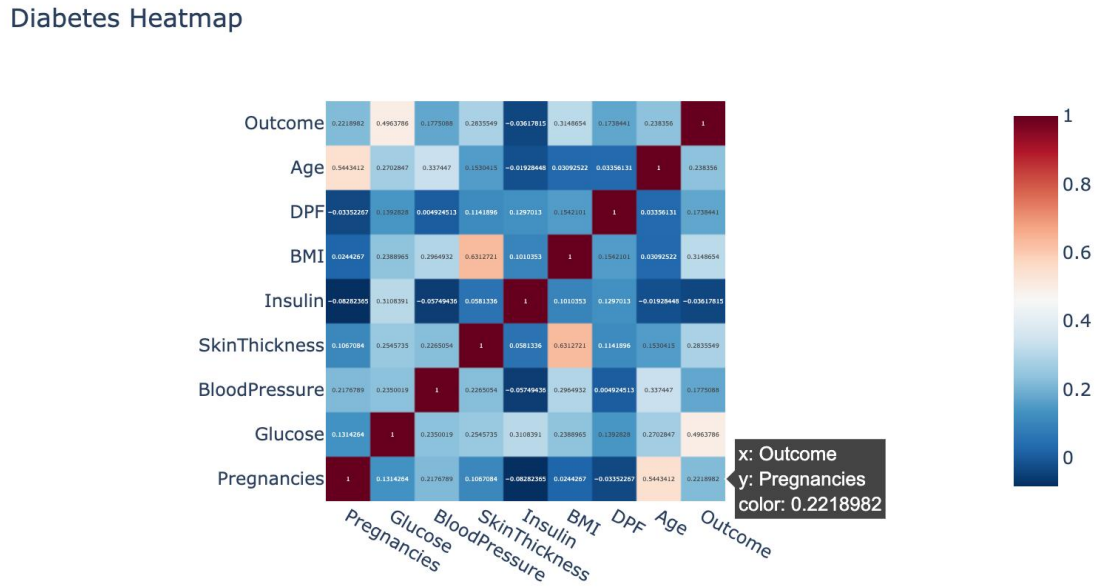


figure 2.4.a: Heatmap

b) PariMap using plotly

In this piar plot, only Glucose and Insulin seems to have a exponential relationship

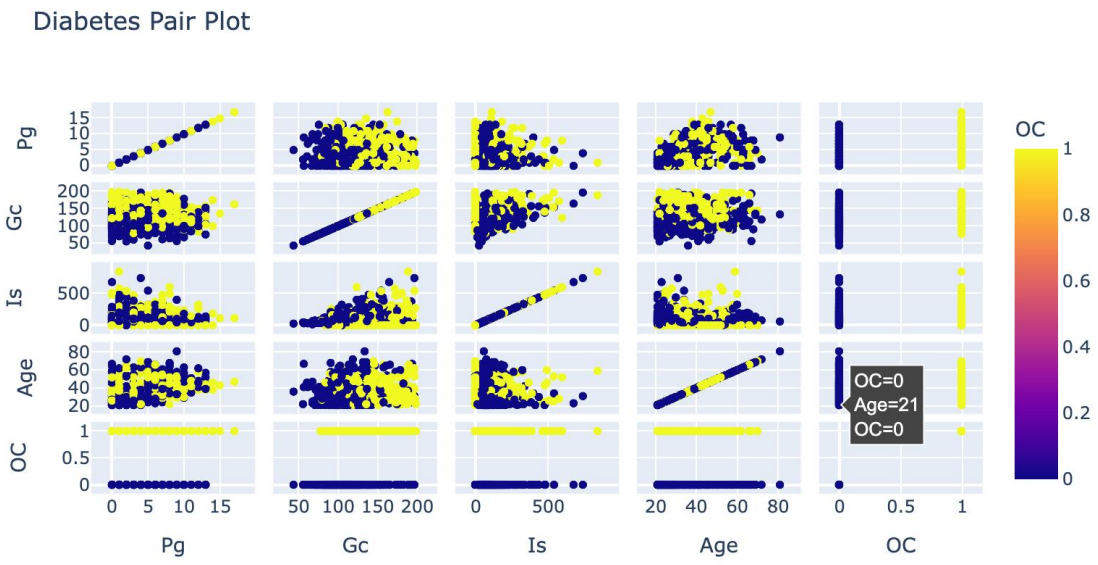


figure 2.4.b1 : Pair Plot 1

In this pair plot, we can not find some obvious relationship between columns.

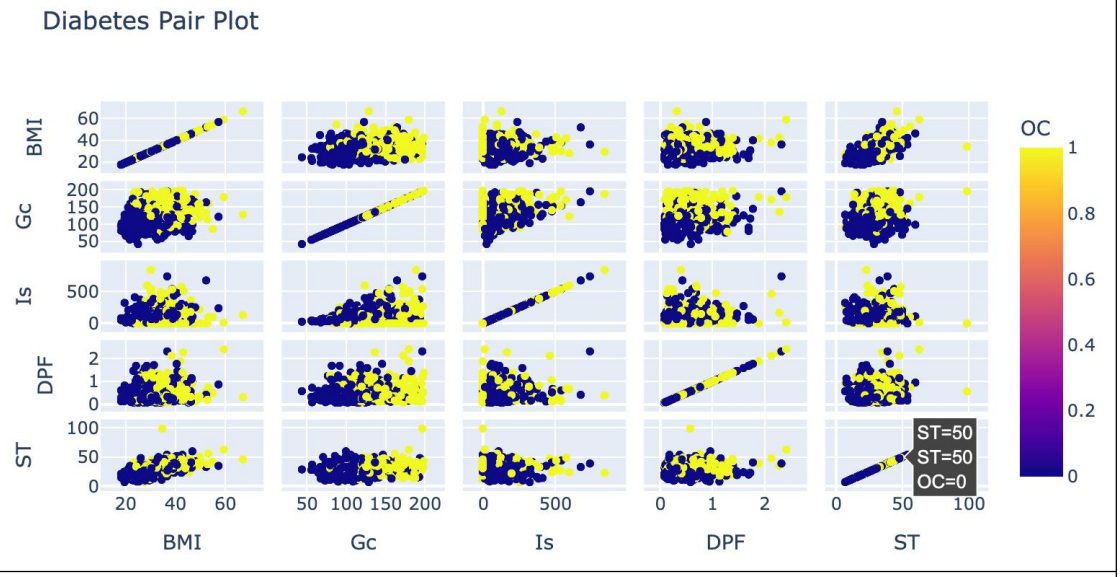


figure 2.4.b2: Pair Plot 2

III: Come up with Meaningful Questions

Preface:

After dig into the details of this dataset, we are now have an idea about this dataset, combine this information and with our knowledge. We are trying to come up with some meaningful questions and hopefully these questions can lead us to a deeper understanding of the diabetes.

Questions:

Woman are very care about their beauty. So I want to know the relationship between their beauty and diabetes.

Here, the beauty can be represent as skin thickness, BMI and pregnancies, ages.

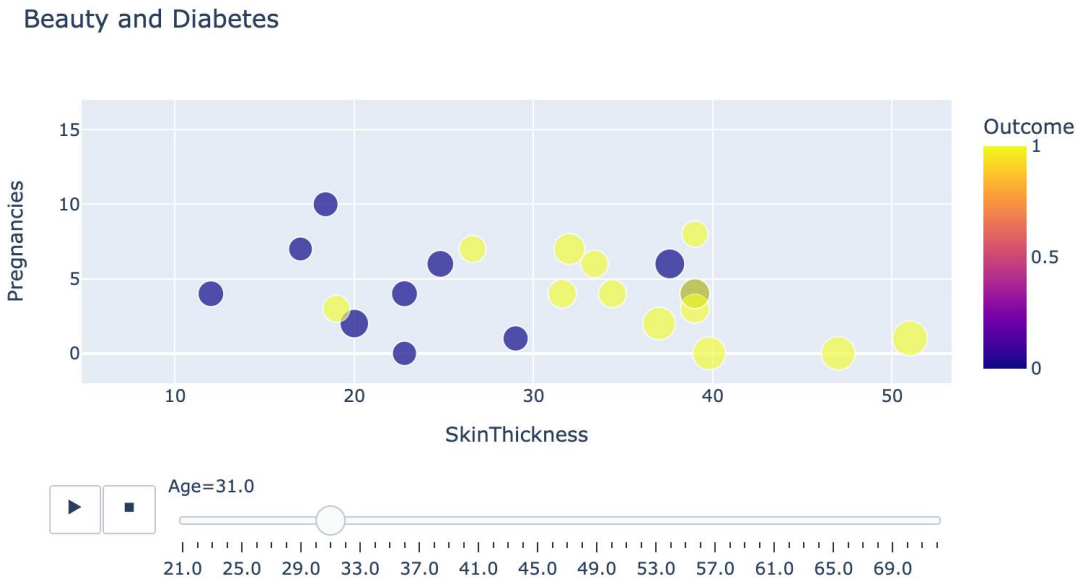
Because in my experience, the beautiful woman may have relatively low skin thickness in triceps for better shape; Suitable BMI for better shape ; Relatively low pregnancies and ages.

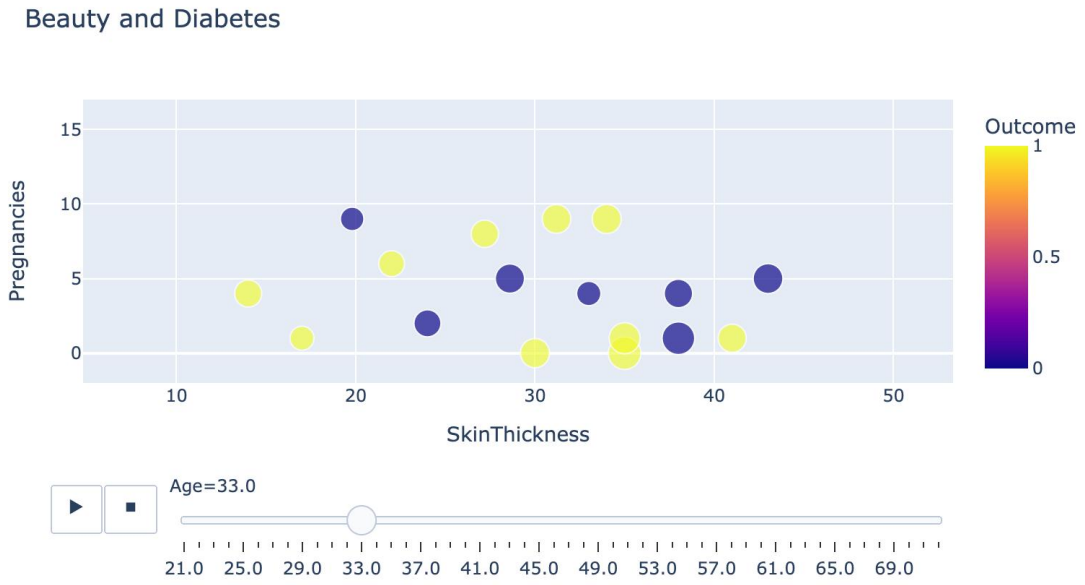
I want to know if there are clear relationship between them. I want to know the precentage if I saw a Pima Indian heritage woman in the street and she is beautiful, how many precentage she may suffer diabetes.

IV: Visualization

Plotyly Animation:

As you can see in the plots below, the yellow represent the diabetes and blue represent the normal people. The size is defined by BMI, larger the BMI, larger the cirle. Animation frame is Age. In the x-axis, we have skin thickness. In the y-axis, we have pregnancies.
Go through the animation, we can have a view of the relationship and diabetes. The trend is if a woman tend to have a thick skin, large BMI, high preganancies, she is more possible to have diabetes.
However, we can't know tell the relationship between age and outcome due to we have very small part of old people in the dataset. Most of testers are smaller than 40 and almost half is between 20-30. Even though there is good positive relationship between age and pregnancies.





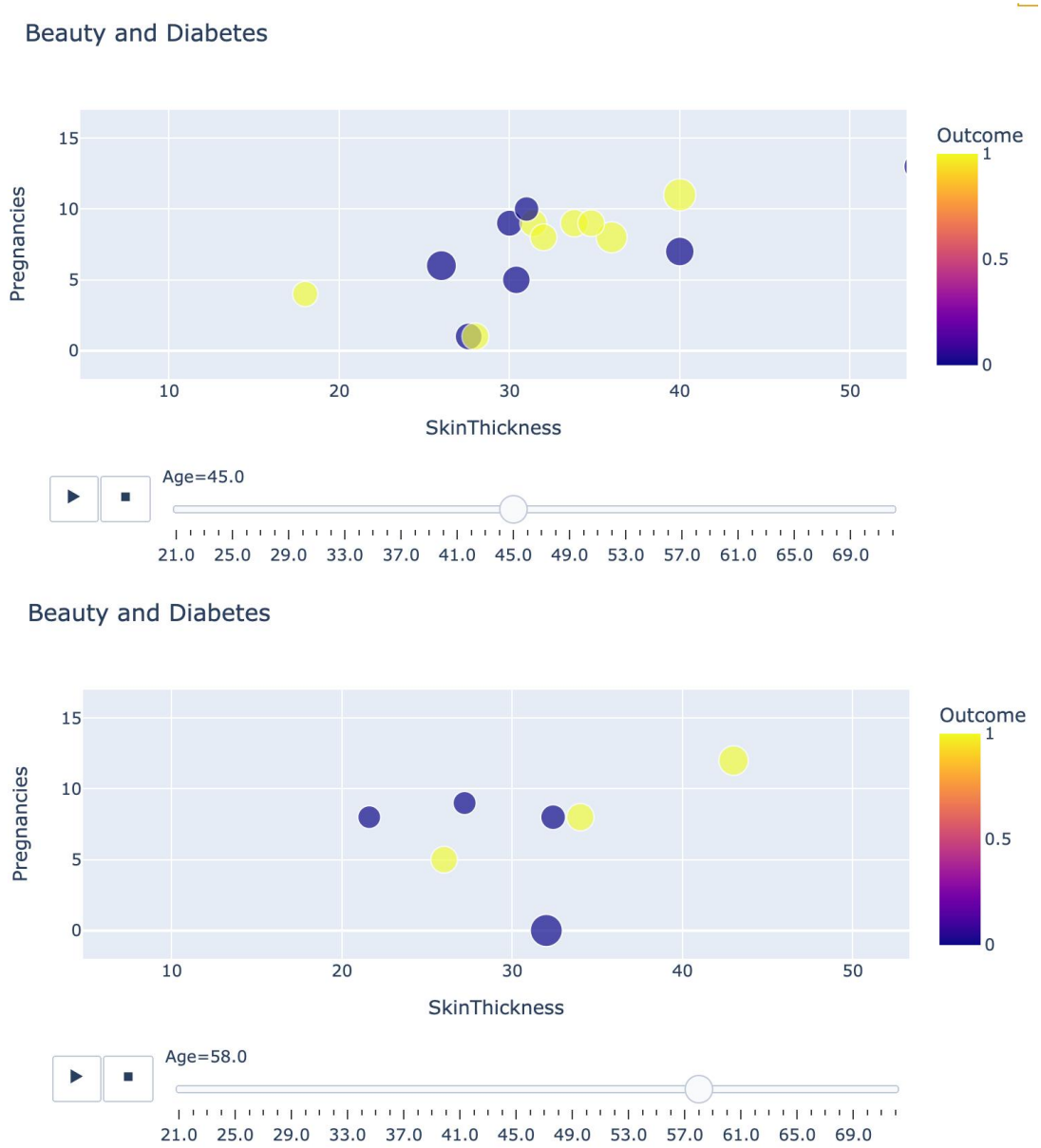


figure 4.1 : Beauty and Diabetes Animation

V: Summary and Chanllenge

Summary:

We look at the details of the diabetes dataset and have a better understanding of the relationship between columns. Our question is about beauty and diabetes.

We use four different visualization techics to find the relationship in the columns.
1. Searborn Histogram of each columns to see the distribution.
2. Plotly Heatmap to see the linear correlation between columns.
3. Plotly Pair Plot to see the linear and non-linear relationship between columns.
4. Plotly 5D animation plot to have a view of beauty factors and outcome.

We see the trend is if a woman tend to have a thick skin, large BMI, high preganancies, she is more possible to have diabetes. the possibility is not predominate, we only can say is around 60-70 percentage.

Chanllenge:

During the visualization of the diabetes dataset, we find lots of factors that constrain us to get a more precise prediction and deeper understanding.

1. Don't have expert knowledge in medical field, can not according to the relationship between columns and give further hypothesis. What we visualized is limited by our knowledge.

2. There are only 768 smaples in the dataset, may not big enough for more precise prediction.

3. I don't know the dataset is pregnancies columns have missing data or not. We can't tell when value is 0, it means never pregnant or missing. It might affect the result significantly. In this case, we can choose to drop the rows with 0 pregnancy, however, the draw back is we can't use this classifer to estimate the data with preganancy value is 0.

4. A chanllenge is always there: When you have missing data, and you have different types of imputation methods based on different aspect. Like mean, median, constant number, iterative imputation, KNN imputer, EM, replace according to the distributions using random method. But different type of imputation method can lead different estimator to differnt results. For example, if I use KNN imputer to replace the missing value. When I use KNN estimator to estimate the model, I probably will get a higer score than other estimators. What should we do in this situations are always chanllenge. And tries all possible combination seems impossible sometimes. Imputation in some sense is also a estimator. What should we stick to ? In the features of the dataset? If we know the dataset better , we can choose better, cause sometimes according to the reality and experince, we know the preference of each method. Do we have any other way to decide ?