Documentation
Assignment 1
GEN-511
15th September, 2019

1. Problem Statement

We are given a dataset having certain attributes and various classes. Using the dataset, we need to train a model which classifies when a patient has diabetes or not. In the process, we need to

- Explore Data
- Fill the missing values
- Add new features (if required)
- Select Features
- Train model

2. Dataset Description

This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases. The objective of the dataset is to diagnostically predict whether or not a patient has diabetes, based on certain diagnostic measurements included in the dataset. 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. The datasets consists of several medical predictor variables and one target variable, "Outcome". Predictor variables includes the number of pregnancies the patient has had, their BMI, insulin level, age, glucose, blood pressure, skin thickness, body mass index, and the diabetes pedigree function. This is a binary classified data set.

- Exploratory Data Analysis

The data provided has 768 observations. Each observation has eight of the above described columns. Some of the cells contain NaN entries and some values were 0. Apart from the class labels, the value 0 in each of the cell is treated as a missing value. The table below shows the details of the above inferred information. The column Pregnancies has the maximum number of null values. In the section Data Preprocessing we describe the steps taken to handle this data and the experiments conducted on the data is described in the experimental scenario.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
Pregnancies                 742 non-null float64
Glucose                     752 non-null float64
BloodPressure               768 non-null float64
SkinThickness               746 non-null float64
Insulin                     768 non-null float64
BMI                         757 non-null float64
DiabetesPedigreeFunction    768 non-null float64
Age                         749 non-null float64
Outcome                     768 non-null int64
dtypes: float64(8), int64(1)
memory usage: 54.1 KB
```

Another observation that can be taken from the table is that blood pressure, insulin and DPF have no missing data. But they might have some zero values. They can be used assisting in finding correlation in the later stages of the assignment.

Illustration 1: Data Information

Moving further, we later present the data description in the illustration table 2. The data description shows up that apart from NaN value and 0, some attributes even have the minimum value as negative floating point integers. Also, in the given data, without any preprocessing, the values have approximately close mean and median values. Thus in the later part, we need to deal with these negative values of the attributes as well.

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	742.000000	752.000000	768.000000	746.000000	768.000000	757.000000	768.000000	749.000000	768.000000
mean	3.866601	119.966097	68.886078	20.309879	79.799479	31.711151	0.471876	33.761336	0.348958
std	3.479971	32.367659	19.427448	15.974523	115.244002	8.544789	0.331329	12.297409	0.476951
min	-5.412815	0.000000	-3.496455	-11.945520	0.000000	-16.288921	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.100000	0.243750	24.000000	0.000000
50%	3.000000	116.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.000000	80.000000	32.000000	127.250000	36.500000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

Illustration 2: Data Description

We shall now be checking the correlation between each of the two attributes. For the instance, we replace all the missing and the negative values to 0. This was done because on handling the missing values by dropping them reduced the dataset to 292 values. This could have resulted in miscalculation from the current result. Calculating correlation between the two attributes helps us to conclude the type of relation between the two attributes. This also helps us as some of the stronlgy related attributed could be feature engineered in the further part of the assigment to form a new feature. Correlation may also help us in filling up the missing data as the data set can be sorted with respect to the strongly correlated attributes and later the values of the missing cells could have been taken from the neighbour cells.

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
Pregnancies	1	0.0754641	0.112882	-0.107483	-0.0914524	-0.0166993	-0.0324557	0.400865	-0.187193
Glucose	0.0754641	1	0.102135	0.0475497	0.255817	0.169892	0.117387	0.136966	-0.424455
BloodPressure	0.112882	0.102135	1	0.197882	0.0891479	0.244393	0.0399594	0.204735	-0.0628176
SkinThickness	-0.107483	0.0475497	0.197882	1	0.413899	0.329112	0.167316	-0.140969	-0.0811122
Insulin	-0.0914524	0.255817	0.0891479	0.413899	1	0.18201	0.185071	-0.0590377	-0.130548
BMI	-0.0166993	0.169892	0.244393	0.329112	0.18201	1	0.134505	-0.0135726	-0.261728
DiabetesPedigreeFunction	-0.0324557	0.117387	0.0399594	0.167316	0.185071	0.134505	1	0.0243998	-0.173844
Age	0.400865	0.136966	0.204735	-0.140969	-0.0590377	-0.0135726	0.0243998	1	-0.164639
Outcome	-0.187193	-0.424455	-0.0628176	-0.0811122	-0.130548	-0.261728	-0.173844	-0.164639	1

Illustration 3: Correlation Matrix

From the above table, some of the clear observations are that insulin level in the body is strongly correlated with the skin thickness of the patient. Also the pregnancy of the patient goes along with the age. Calculating this correlation matrix gives us a deep insight between how connected the corresponding attributes are.

- Data Preprocessing

```
In [7]: # Removing all the rows having any of the null values or numeric value 0

        data["Outcome"] = data["Outcome"].replace(to_replace = 0, value = 2)
        data = data.replace(to_replace = 0, value = np.nan)
        print(data.isnull().sum())
```

```
Pregnancies                 135
Glucose                      21
BloodPressure                35
SkinThickness               242
Insulin                     374
BMI                          25
DiabetesPedigreeFunction      0
Age                          19
Outcome                       0
dtype: int64
```

Illustration 4: NaN values per attributes

As discussed in the above section, the PIMA Indian diabetes dataset contains missing and random values. Some cells even have the negative values. We replaced these negative values, 0 and null values with NaN. One evaluating the number of cells having NaN values, we find that at most 374 observations of insulin attribute have NaN value. Overall, there are only 292 observations with complete data. Dropping this data will lead to a huge loss of information. In the

later sections, we showed the experimental results after dropping the null values and by replacing these null values with optimum fillers.
We now fill the missing data values. For imputation, we chose three basic methods.

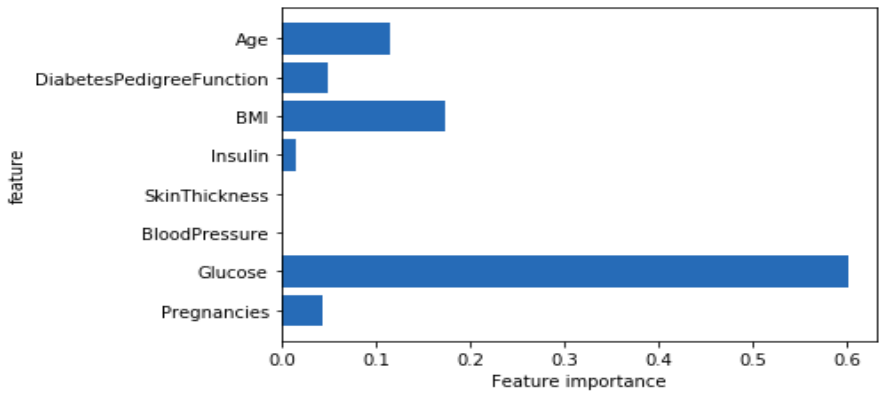
i. **Mean:** We computed the mean of the remaining observations left in the attribute and filled the missing values with the mean. This is the most basic approach and it requires the least overhead time. One of the drawback of using mean is that it reduces the variance. But this can also be taken as a silver lining as an easy hyposis would even work in this case. Another reason of replacing the data with mean value was as in the given dataset, the mean and median values are approximately equal. Thus choosing these values gave us the best of both the worlds.

ii. **Sklearn.preprocessing Impute.SimpleImputer:** The scikit-learn library provides the Imputer() pre-processing class that can be used to replace the missing values. It is a flexible class that allows you to specify the value to replace (it can be something other than NaN) and the technique used to replace it (such as mean, median, or mode). The Imputer class operates directly on the NumPy array instead of the DataFrame. This uses a predictive model to find the best value to be fitted in the position.

iii. **Tweaked Rand():** Another method to impute the missing value was to fill the required location with missing values. We tweaked the method by calculating the minimum value and the maximum value of each attribute and later filled the missing value with a random value in the given range.

- Feature Extraction

After filling the missing values and conducting experiments described in section 3 of the document, we computed the importance and contribution of each feature towards the required problem statement. Using the decision tree model, we computed the importance of each attribute. The illustration below shows the importance of each attribute.



This figure helps us to conclude that attriubtes like skin thickness and blood pressure have no contribution on the result. Moreover, attribute like insulin has a very low impact

on the result but these attributes are strongly correlated to each other. Thus, we computed a unified value of the three attributes using the polynomial expression

new_attribute = skinthickness^3+ bloodpressure^2+insulin

and added this new attribute and dropped the previous three. We also combined the attribute glucose with insulin and tested the required data on the model. The illustration below gives the pair wise representation of data over all the attributes.

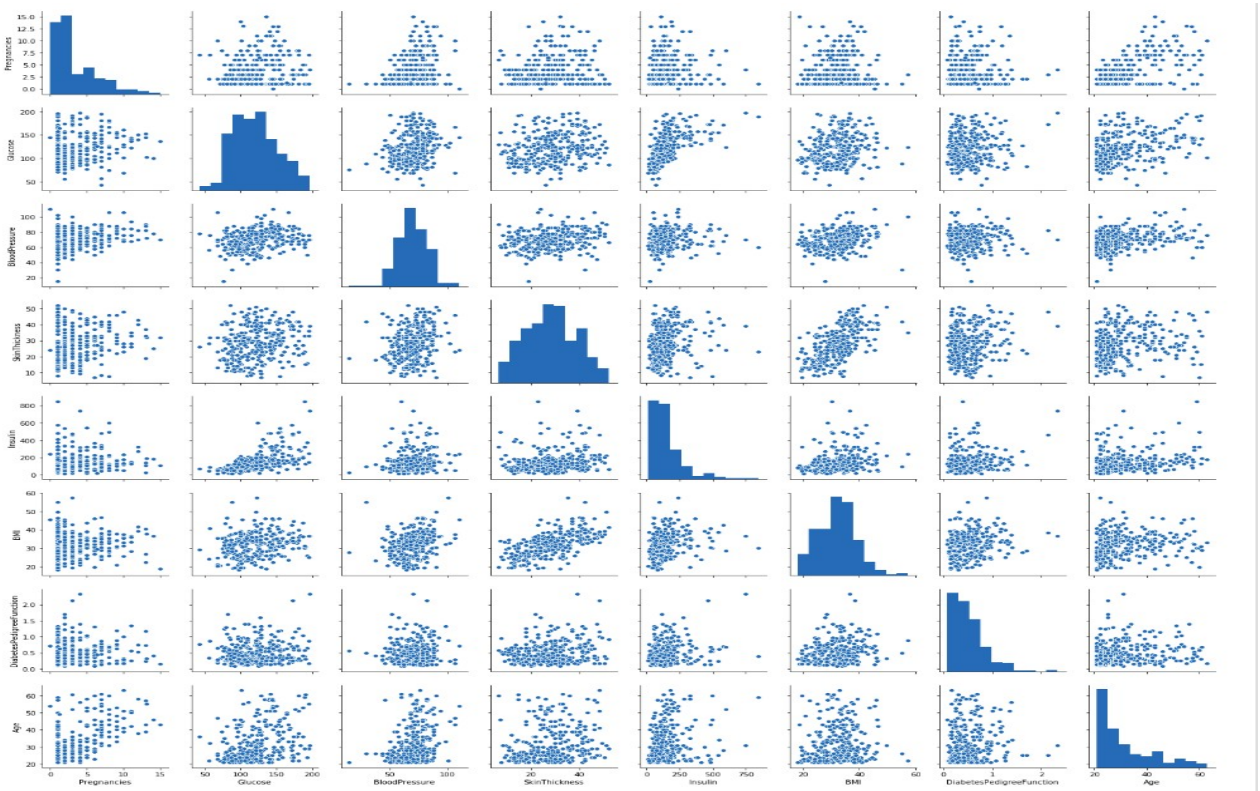


Illustration 5: Pairwise attribute representation

3. Experimental Scenario

- In this experiment, we dropped all the missing and misinterpreted values and exposed out dataset to various models. For instance, we only used two models on the dataset, namely, Support Vector Machine and Logistic Regression. This enabled us to get a general idea on the performance of our data on the model. The data being binary classified, logistic regression gives a better response on such data.

	precision	recall	f1-score	support
1.0	0.65	0.38	0.48	29
2.0	0.75	0.90	0.82	59
accuracy			0.73	88
macro avg	0.70	0.64	0.65	88
weighted avg	0.71	0.73	0.70	88

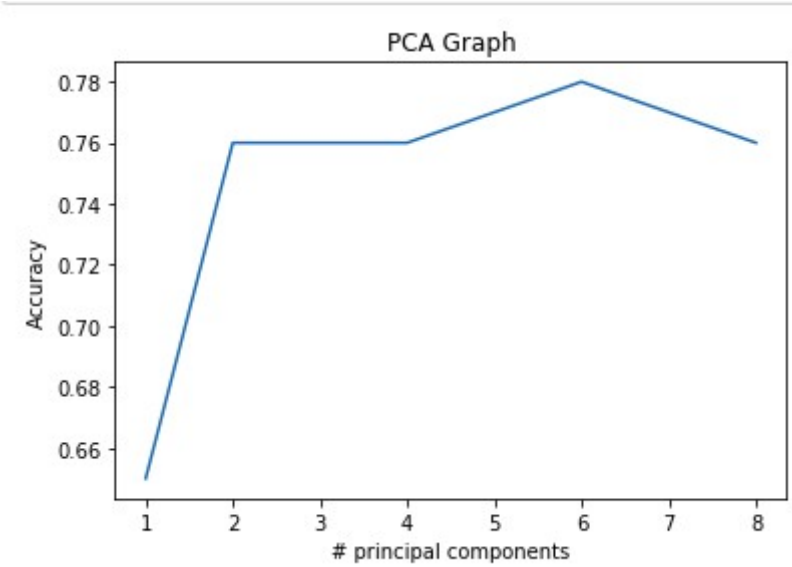
The table shows the result of the above conducted experiment. The

	precision	recall	f1-score	support
1.0	0.71	0.41	0.52	29
2.0	0.76	0.92	0.83	59
accuracy			0.75	88
macro avg	0.73	0.66	0.68	88
weighted avg	0.74	0.75	0.73	88

Illustration 6: Results just after dropping null values

SVM model reports an accuracy of 73% whereas the LR reports 75% on the other hand.

- In the second experimental scenario, we filled the missing data values with the mean of the remaining values of the attribute and trained the model on this data. Now, we extented this experiment by fluctuating the dimensions of the given data. We reduced the dimensions using the principal component analysis. The graph illustrates the performance of model over various dimensions.



With tweaking the dimension, both the models recorded an accuracy of 75% over the dataset.

- At last, we added two new features, we combined blood pressure, skin thickness and insulin with the given expression above. We even combined glucose and insulin, and on reducing the dimensions to 4, we obtained a paradigm which was 79% accurate.

4. Results

The last experiment conducted gave us a maximum accuracy of 79%. Thus, after dropping skin thickness and blood pressure attribute and replacing them with new attributes, the paradigm on dimension reduction, stated the above accuracy.