UDACITY MACHINE LEARNING ENGINEER
NANODEGREE 2020

CAPSTONE PROJECT

# Predicting diabetes by using Pima Indian Diabetes dataset

Neelam Sinha
January 27th 2020

# 1 Definition

## 1.1 Project Overview

Diabetes is a common physiological health problem among humans across gender, race and age. The term diabetic is applied when an individual is unable to break down glucose, for lack of insulin. The human organ called pancreas is responsible for generating the hormone called insulin, which is a very important enzyme that regulates the sugar level in human blood stream. As a result of lack of insulin, the body cells didn't get the energy they need, and thus elevates the sugar level in the blood, and many problems/ diseases can emerge like heart attack, blindness etc.
Diabetes is not a curable disease; although, fortunately, it is treatable. Diabetes and related complications are responsible for the death of almost 200,000 Americans every year [1, 2]. In past there are many works done using standard statistical techniques such as discriminate analysis, regression analysis, and factor analysis, neural network models to provide prediction but these standard statistical methods may provide disappointing results when:

- The sample size is small.
- The form of the underlying functional relationship is not known.
- The underlying functional relationships involve complex interactions and intercorrelations among several variables.[3]
- Using neural network models[4] will lead black box situation, the understand and interpretation of the model, and its working become very difficult.

That's why I choose to use simple supervised machine learning models in my project to predict diabetic or non-diabetic patients based on their diagnostic measurements.

The goal of this capstone project is to apply supervised machine Learning techniques to the classification of patients to be diabetic or not.

## 1.2 Problem Statement

In modern healthcare, predicting and properly treating diseases have become of foremost importance in medical prognostics fields. But even in this era, the normal identifying, whether the person is diabetic or non-diabetic process need patients to visit a diagnostic center, consult their doctor, and wait for a day or more to get their report.

The objective of this project is to use Machine Learning methods to predict whether a person has diabetes or not, in a supervised fashion.

## 1.3 Metrics

As the dataset has classification problem, there are many pre-defined evaluation metrics which we can use. For example, F1_score, Recall, Precision, Accuracy, ROC_AUC. Etc. In used two metrics to evaluation.

The evaluation metric for this problem will be the 'Classification Accuracy' which is defined as the percentage of correct predictions.

**Accuracy = correct classifications / number of classifications**

Where
correct classification = True positive + true negative

Number of classifications= True positive + true negative + false positive +false negative

We can obtain the accuracy score from scikit-learn, which takes as inputs the actual labels and the predicted labels.

**ROC (Receiver Operating Characteristic)**- ROC Curve tells us about how good the model can distinguish between two things (e.g If a patient has a disease or no). Better models can accurately distinguish between the two. Whereas, a poor model will have difficulties in distinguishing between the two.

# 2 Analysis

## 2.1 Data Exploration and Visualisation

### 2.1.1 Dataset Used

The data was collected and made available by "National Institute of Diabetes and Digestive and Kidney Diseases" as part of the Pima Indians Diabetes Database. Several constraints were placed on the selection of these instances from a larger database. In particular, all patients here belong to the Pima Indian heritage (subgroup of Native Americans), and are females of ages 21 and above.

Data set can be downloaded from here: https://www.kaggle.com/uciml/pima-indians-diabetes-database

The dataset is saved in diabetes.csv format file and, from the domain knowledge, I have analyzed and found out the ranges of values and its effects on diabetes.

There are 8 independent variables:

- Pregnancies: No. of times pregnant
- Glucose: Plasma Glucose Concentration a 2 hour in an oral glucose tolerance test (mg/dl)
- Blood Pressure: Diastolic Blood Pressure(mmHg)
  - If Diastolic B.P > 90 means High B.P (High Probability of Diabetes)
  - Diastolic B.P < 60 means low B.P (Less Probability of Diabetes)
- Skin Thickness: Triceps Skin Fold Thickness (mm) –
  A value used to estimate body fat. Normal Triceps SkinFold Thickness in women is 23mm. Higher thickness leads to obesity and chances of diabetes increases.
- Insulin: 2-Hour Serum Insulin (mu U/ml)
- BMI: Body Mass Index (weight in kg/ height in m2) –
  BMI of **18.5 to 25** is within the normal range, BMI between **25 and 30** then it falls within the overweight range. A BMI of **30 or over** falls within the obese range.
- Diabetes Pedigree Function: It provides information about diabetes history in relatives and genetic relationship of those relatives with patients. Higher Pedigree Function means patient is more likely to have diabetes.
- Age (years)
- Outcome: Class Variable (0 if non-diabetic, 1 if diabetic)

### 2.1.2 Sample of data and data overview

The Dataset is available in .csv file. Here is

Loading Data in python:

- Library Used:

```
from scipy import optimize
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

- **Loading Data:**

```
diabetes = pd.read_csv("C:/Users/sapph/Desktop/capston_project/diabetes.csv",
header = 0, sep = ',' ,index_col = None)
diabetes.head(10)
```

- **Sample Data:**

| Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | DiabetesPedigreeFunction | Age | Outcome |
|---|---|---|---|---|---|---|---|---|
| 6 | 148 | 72 | 35 | 0 | 33.6 | 0.627 | 50 | 1 |
| 1 | 85 | 66 | 29 | 0 | 26.6 | 0.351 | 31 | 0 |
| 8 | 183 | 64 | 0 | 0 | 23.3 | 0.672 | 32 | 1 |
| 1 | 89 | 66 | 23 | 94 | 28.1 | 0.167 | 21 | 0 |
| 0 | 137 | 40 | 35 | 168 | 43.1 | 2.288 | 33 | 1 |
| 5 | 116 | 74 | 0 | 0 | 25.6 | 0.201 | 30 | 0 |
| 3 | 78 | 50 | 32 | 88 | 31 | 0.248 | 26 | 1 |
| 10 | 115 | 0 | 0 | 0 | 35.3 | 0.134 | 29 | 0 |
| 2 | 197 | 70 | 45 | 543 | 30.5 | 0.158 | 53 | 1 |
| 8 | 125 | 96 | 0 | 0 | 0 | 0.232 | 54 | 1 |
| 4 | 110 | 92 | 0 | 0 | 37.6 | 0.191 | 30 | 0 |

### 2.1.2 Class distributions

```
print(diabetes.info(verbose=True))
```

```
Data columns (total 9 columns):
Pregnancies                 768 non-null int64
Glucose                     768 non-null int64
BloodPressure               768 non-null int64
SkinThickness               768 non-null int64
Insulin                     768 non-null int64
BMI                         768 non-null float64
DiabetesPedigreeFunction    768 non-null float64
Age                         768 non-null int64
Outcome                     768 non-null int64
```

- **Data Summary:**

| | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | DiabetesPedigreeFunction | Age | Outcome |
|---|---|---|---|---|---|---|---|---|---|
| count | 768.000000 | 768.000000 | 768.000000 | 768.000000 | 768.000000 | 768.000000 | 768.000000 | 768.000000 | 768.000000 |
| mean | 3.845052 | 120.894531 | 69.105469 | 20.536458 | 79.799479 | 31.992578 | 0.471876 | 33.240885 | 0.348958 |
| std | 3.369578 | 31.972618 | 19.355807 | 15.952218 | 115.244002 | 7.884160 | 0.331329 | 11.760232 | 0.476951 |
| min | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.078000 | 21.000000 | 0.000000 |
| 25% | 1.000000 | 99.000000 | 62.000000 | 0.000000 | 0.000000 | 27.300000 | 0.243750 | 24.000000 | 0.000000 |
| 50% | 3.000000 | 117.000000 | 72.000000 | 23.000000 | 30.500000 | 32.000000 | 0.372500 | 29.000000 | 0.000000 |
| 75% | 6.000000 | 140.250000 | 80.000000 | 32.000000 | 127.250000 | 36.600000 | 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 |

From above table, we analyse that the min value is zero for more than one variable. If a variable other than(pregnancies, outcome), have value of zero, it does not make sense and thus indicates missing value, as those variables represents important in medical diagnostics. These are the variables that have an invalid zero value:

- Glucose
- BloodPressure
- SkinThickness
- Insulin
- BMI

Therefore, the dataset needs handling of missing values.

### 2.1.4 Visualization of the Pima dataset:

- **Find distribution of each variable before handling missing data:**

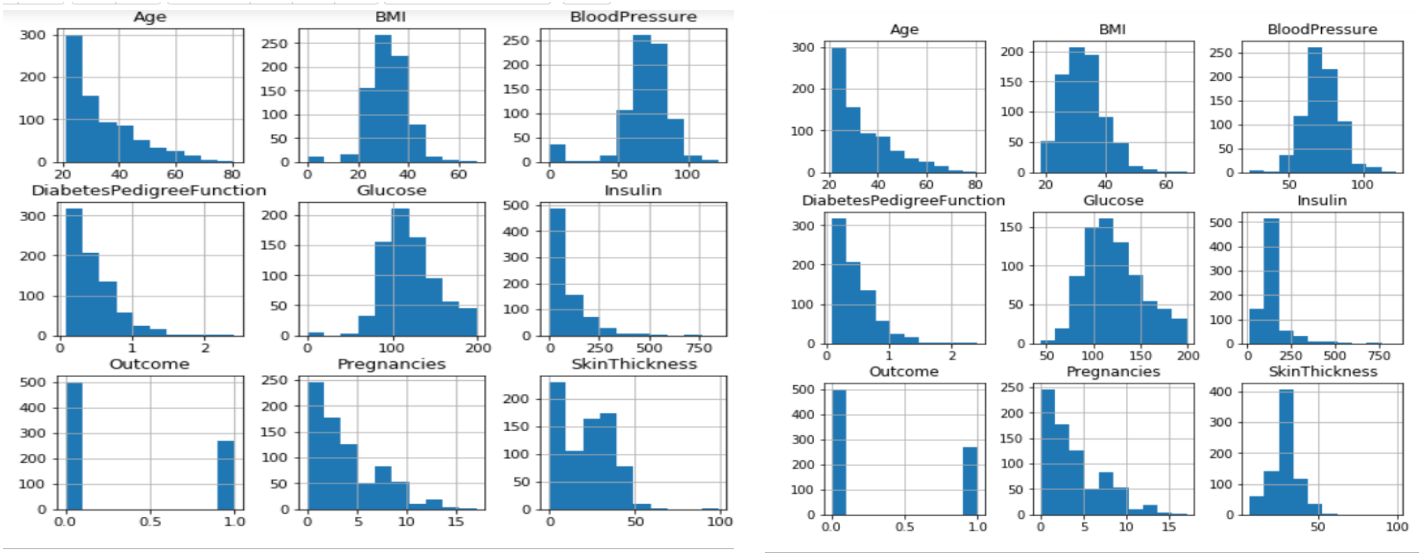


Figure1: (a) Shows histogram plot of the distribution of each variable before imputation. (b) Shows histogram plot of the distribution of each variable after imputation.

Observation: From figure 1(a), We can see from the distribution of each variable that it has data have missing values, So we aim to Impute the Dataset.

### 2.1.5 Handling the missing Value:

To Handle the missing value, I choose to Impute the columns in accordance with their distribution. First I replace the zeroes value with NAN values of the variables(**Glucose, Blood Pressure, Skin Thickness, Insulin, BMI**). So that we should be aware of counts which were missing.

```
# number of notnull values before imputation
diabetes.count():

Pregnancies                 768
Glucose                     763
BloodPressure               733
SkinThickness               541
Insulin                     394
BMI                         757
DiabetesPedigreeFunction    768
Age                         768
Outcome                     768
```

After Finding counts of missing value I Imputed each dataset variable with their column mean. Here is the Imputation code and distribution of data after imputation:

```
diabetes['Glucose'].fillna(diabetes['Glucose'].mean(), inplace = True)
diabetes['BloodPressure'].fillna(diabetes['BloodPressure'].mean(), inplace = True)
diabetes['SkinThickness'].fillna(diabetes['SkinThickness'].mean(), inplace = True)
diabetes['Insulin'].fillna(diabetes['Insulin'].mean(), inplace = True)
diabetes['BMI'].fillna(diabetes['BMI'].mean(), inplace = True)
diabetes.count()
```

```
# number of notnull values before imputation
Pregnancies                 768
Glucose                     768
BloodPressure               768
SkinThickness               768
Insulin                     768
BMI                         768
DiabetesPedigreeFunction    768
Age                         768
Outcome                     768
```

- **The correlation plot:**

To understand data better I plot the correlation:

```
corr= diabetes.corr()
corr.style.background_gradient(cmap='RdBu_r', axis=None)
```

| | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | DiabetesPedigreeFunction | Age | Outcome |
|---|---|---|---|---|---|---|---|---|---|
| Pregnancies | 1 | 0.128135 | 0.214178 | 0.100239 | 0.082171 | 0.0217189 | -0.0335227 | 0.544341 | 0.221898 |
| Glucose | 0.128135 | 1 | 0.223192 | 0.228043 | 0.581186 | 0.232771 | 0.137246 | 0.267136 | 0.49465 |
| BloodPressure | 0.214178 | 0.223192 | 1 | 0.226839 | 0.0982723 | 0.28923 | -0.00280453 | 0.330107 | 0.170589 |
| SkinThickness | 0.100239 | 0.228043 | 0.226839 | 1 | 0.184888 | 0.648214 | 0.115016 | 0.166816 | 0.259491 |
| Insulin | 0.082171 | 0.581186 | 0.0982723 | 0.184888 | 1 | 0.22805 | 0.130395 | 0.220261 | 0.303454 |
| BMI | 0.0217189 | 0.232771 | 0.28923 | 0.648214 | 0.22805 | 1 | 0.155382 | 0.0258415 | 0.31368 |
| DiabetesPedigreeFunction | -0.0335227 | 0.137246 | -0.00280453 | 0.115016 | 0.130395 | 0.155382 | 1 | 0.0335613 | 0.173844 |
| Age | 0.544341 | 0.267136 | 0.330107 | 0.166816 | 0.220261 | 0.0258415 | 0.0335613 | 1 | 0.238356 |
| Outcome | 0.221898 | 0.49465 | 0.170589 | 0.259491 | 0.303454 | 0.31368 | 0.173844 | 0.238356 | 1 |

Figure2: Correlation plot

### 2.2 Algorithms and Techniques

**Solution:** As the problem statement have to classify the data into patients having diabetes or not, the best method which can be used is Classification Tree Algorithm or Classification Ensemble Algorithm. Further we can easily classify and predict the outcome using nodes and internodes.
**Software Package Used:** Python, Libraries Used: Pandas, Scikit Learn, Numpy, Scipy, Matplotlib,Seaborn.

### 2.3 Benchmark Model

For the benchmark model, we will use the algorithms outlined in the paper "Using the Adap Learning Algorithm to Forecast the Onset of Diabetes Mellitus" [4]. The paper describes They describe ADAP as "*an adaptive learning routine that generates and executes digital analogs of perceptron-like devices*".
This algorithm will make predictions based on a function of input variables and will make internal adjustments if predictions are incorrect. The network is split into 3 main layers:

- **Input, partitioned into "sensors"**: Represents a discrete value. These are organized into partitions and are "excited" by input.
- **Association Units**: Uses a threshold function to activate a specific responder value. Connected to adjustable weights that change based on said function.
- **Responder**: Responder values are summed and constitute a specific prediction.

| Algorithm | AUC |
|---|---|
| ADAP | 76% |

## 3 Methodology

### 3.1 Data Preprocessing and Data Splitting

#### 3.1.1 Data cleaning and Imputation:

In the dataset Data cleaning will take place as data has got lot of Zeroes in some columns like Glucose, Blood Pressure, Skin Thickness, Insulin, BMI, which does not make sense as these columns are important and required in patient diagnosis data.

So, I consider them as missing value and Impute the columns in accordance with their mean. In above figure 1(a), 1(b) we can see the histogram plot of both when data was not imputed and have missing values, and when data is imputed with the mean of the corresponding variables.

#### 3.1.2 Libraries used for machine learning.

Basically, I used Scikit learn to import Classifier algorithm, metric (to evaluate) which I will use in my model pipeline.

```
from sklearn import svm
from sklearn.svm import SVC
from sklearn.neighbors  import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn import metrics
```

#### 3.1.3 Split the dataset

To have unknown datapoints to test the data and model rather than testing with the same points with which the model was trained, data split is required. This helps capture the model performance much better.

Here I use `sklearn.model_selection.train_test_split` to split the dataset into training and testing sets. The testing set size will be 25% and set a random state to 100.

```
# split the dataset
outcome=diabetes['Outcome']
data=diabetes[diabetes.columns[:8]]
train,test=train_test_split(diabetes,test_size=0.25,random_state=100,stratify=diabetes[
'Outcome'])
train_X=train[train.columns[:8]]
test_X=test[test.columns[:8]]
train_Y=train['Outcome']
test_Y=test['Outcome']
```

### 3.2 Implementation

#### 3.2.1 Model Selection

Model selection is one of the difficult jobs, but as the problem statement is about classification. So, first to best guess optimal model I use Classifier tree algorithm(DecisionTreeClassifier()),

Classifier ensemble algorithm(RandomForestClassifier(), GradientBoostingClassifier(), AdaBoostClassifier()) and use KNN and SVM model to check which model performs well on he given dataset.

### 3.2.2 Training and Testing data

After model selection, I train all the models using `model.fit(train_X,train_Y)` on training data. Then predict the target outcome using the test feature data, `prediction=model.predict(test_X)`. After prediction I validate the target outcome which I predict with the target outcome of testing data. For validation/ metric evaluation I use Accuracy score `metrics.accuracy_score(prediction,test_Y)` and ROC_AUC score `metrics.roc_auc_score(prediction,test_Y)`.

Here is the following code:

```
abc=[]
roc=[]
classifiers=['Linear Svm','KNN','Decision Tree', 'Random forest', 'gbc', 'adaboost']
models=[svm.SVC(kernel='linear'),KNeighborsClassifier(),
DecisionTreeClassifier(random_state=0), RandomForestClassifier(random_state=0),
GradientBoostingClassifier(random_state= 0), AdaBoostClassifier(random_state=0)]
for i in models:
    model = i
    model.fit(train_X,train_Y)
    prediction=model.predict(test_X)
    abc.append(metrics.accuracy_score(prediction,test_Y))
    roc.append(metrics.roc_auc_score(prediction,test_Y))
models_df=pd.DataFrame(abc,index=classifiers)
models_df.columns=['Accuracy' ]
models_df['Roc_auc']= roc
```

| | Accuracy | Roc_auc |
|---|---|---|
| **Linear Svm** | 0.765625 | 0.748626 |
| **KNN** | 0.723958 | 0.694673 |
| **Decision Tree** | 0.661458 | 0.628321 |
| **Random forest** | 0.750000 | 0.726875 |
| **gbc** | 0.755208 | 0.730645 |
| **adaboost** | 0.760417 | 0.737479 |

Figure3: Metric Evaluation of different models

**Observation:** From this brief sanity check the model seems to predict not well.

After running all model, the accuracy and AUC score I got was not very high. During this phase of training and testing I am getting maximum AUC of ~74% which show my models are performing bad than the benchmark model. which can be further improved by using only relevant features (figure4) and then Standardize the dataset based on these four important features (first four with highest weight in figure 4).

## 3.3
## Refinement

### Feature Extraction refinement

Prevoiously, I was using all features of dataset to train and test my model. I can improve my model by training my model only on relevant features. And Then and then Standardize the dataset based on these four important features .
For feature extraction I use `model.feature_importances_`, feature_importance function from the model I trained before .

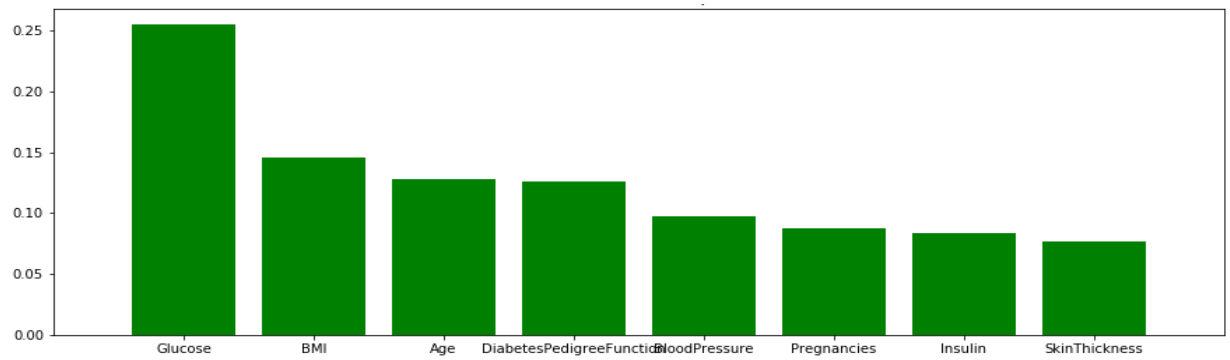


*Figure1: Important feature. We can see Glucose, BMI, Age and DiabetesPedigreeFunction*

### Standardisation:

Standardizing tends to make the training process well behaved because the numerical condition of the optimization problems is improved.
Formula: Here your data Z is rescaled such that **μ** = 0 and **σ** = 1 and is done through this formula.

$$Z = xi - \mu / \sigma$$

After Standardization the Accuracy and AUC were improved **by 6% and 7% respectively**.

Thus, we select our best classifier model i.e RandomForestClassifier() which was showing highest accuracy ans AUC score.

Then Finally I tried to optimize the Random forest and optimize my model with an accuracy of 81% and AUC score of 80%. Figure 2 shows the roc_auc curve plot.

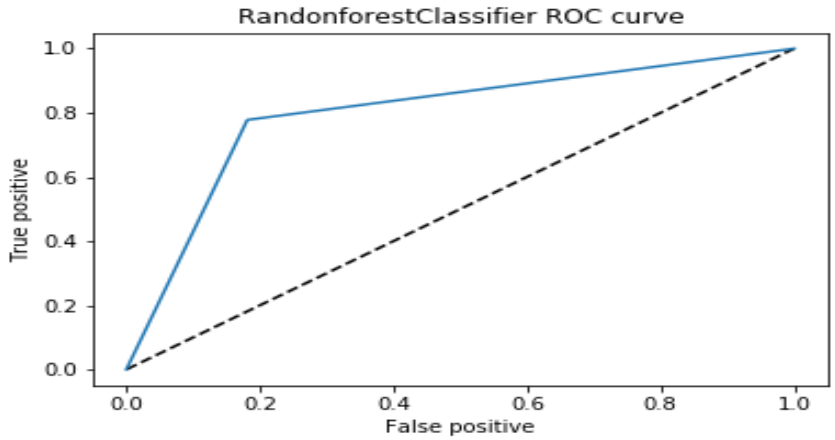


*Figure2: ROC_AUC curve of the Randomforest Classifier model with an AUC of 0.7983 (~0.80).*

# 4 Results

The final model achieved a classification accuracy of 80.72% and AUC of 79.83% on the testing data which exceeded my expectations given the benchmark was 76%.

| **Model** | ROC_AUC |
|---|---|
| RandomforestClassifier | 79.83% |
| Benchmark ADAP algorithm | 76% |

The final solution performs well.

# 5 Conclusion

Advantage of this project:

- The pipeline and algorithm predict and identify patients suffering from diabetes with good accuracy.
- Further predicting the disease at stage early leads to treating the patient before it becomes critical.

## 5.1 Improvement

This Model can be further improved by :

- Using the specific range of the variables.
- Optimizing the Classifier.
- Hyperparmeter Tuning.

# References

1. Alberti, K. G. M. M., and Zimmet, P. F.: Definition, Diagnosis and Classification of Diabetes Mellitus and Its Complications. Part 1: Diagnosis and Classification of Diabetes Mellitus. In: Provisional report of a WHO consultation. Diabetic medicine, 15(7), 539-553 (1998).

2. National Diabetes Data Group: Classification and diagnosis of diabetes mellitus and other categories of glucose intolerance. In: Diabetes, 28(12), 1039-1057 (1979).

3. Simmons, GJ., A Mathematical Model for an Associative Memaory. Sandia Coporation Monograph, SCR-641,1963).

4. Smith, J. W., Everhart, J., Dickson, W., Knowler W., and Johannes, R.: Using the Adap Learning Algorithm to Forecast the Onset of Diabetes Mellitus. In: Proceedings of the Annual Symposium on n Computer Application in Medical Care. American Medical Informatics Association, p. 261 (1988).