

# A UI Based Co-relational Analysis for Diabetic Prognosis using SVM Techniques

## Keywords:

Diabetes , Correlation , SVM , Superior label , Hyperplane

## Abstract

Diabetes is one of the common and growing concern disease in several countries and all of them are working to prevent this disease at early stage .The typical recognizing process is that patients need to visit the diagnostic center, consult specialist and sit tight for a day or more to get their reports .

The objective of this model is to predict whether someone is diabetic or not, based upon input attributes like : **age**, **body mass index(BMI)** and **blood pressure(BP)** .This model uses **support vector machine (SVM)** which is a supervised classification machine learning algorithm .

Roll No	Name
15BD1A054M	B. Nagaraju
15BD1A055T	Suraj Sura
15BD1A0556	G. Manasa

# 1. INTRODUCTION

Diabetes is one of the common and rapidly increasing diseases in the world. It is a major health problem in most of the countries. Diabetes is a condition in which your body is unable to produce the required amount of insulin needed to regulate the amount of sugar in the body. This leads to various diseases including heart disease, kidney disease, blindness, nerve damage and blood vessels damage. There are two general reasons for diabetes:

- (1) The pancreas does not make enough insulin or the body does not produce enough insulin. Only 5-10 % of people with diabetes have this form of the disease (Type-1).
- (2) Cells do not respond to the insulin that is produced (Type-2).

Diabetes disease diagnosis and interpretation of the diabetes data is an important classification problem. A classifier is required and to be designed that is cost efficient, convenient and accurate . **Support Vector Machines (SVM)** have shown remarkable success in the area of employee computer aided diagnostic systems to improve diagnostic decisions.

Artificial intelligence provide a great deal of human ideologies and are involved in human related fields of application. These systems find a place in the medical diagnosis.

## **2. BUSINESS PROBLEM AND SOLUTIONS**

### **2.1 Existing System**

Now a days, the report for diabetes test is a very lengthy process. The typical recognizing process is that patients need to visit the diagnostic center, consult specialist and sit tight for a day or more to get their reports.

### **2.2 Client Expectations**

To make the diagnosis a much easy process , the clients are needed a system which outputs most accurate results and does not consume individual's time and deviate him/her from his day to day routine

### **2.3 Proposed Solution**

A medical diagnosis is a classification process. A physician has to analyze lot of factors before diagnosing the diabetes which makes physician's job difficult. So in order to make the procedure much easy and cost efficient one, we are using machine learning technique for predicting the status of diabetes in any person.

### **2.4 Functional Requirements**

For documenting the functional requirements, the set of functionalities supported by the system are to be specified. A function can be specified by identifying the state at which data is to be input to the system, its input data domain, the output domain, and the type of processing to be carried on the input data to obtain the output.

Functional requirements define specific behaviour or function of the application. Following are the functional requirements:

- i. Should enter the details like Name, Age, BMI and Blood Pressure (BP).
- ii. Should press the PREDICT button to know the result.

### **2.5 Non Functional Requirements**

A non-functional requirement is the one that specifies criteria that can be used to judge the operation of a system, rather than specific behaviour. Especially these are the constraints that the system must work within. Following are the non functional requirements:

- i. Should be available for installation on computer.
- ii. Crashing of application should not occur.
- iii. Response time for each query sent by the user should be minimum.
- iv. Application should be updated properly.

## 2.6 Software and Hardware requirements

Operating System	: Windows 7/8/10
Platform	: Anaconda 1.6.14
Programming Language	: Python 3.6.5
External Files	: Dataset
Processor	: Intel i3
Hard Disk	: 500GB or more
RAM	: 4GB or more

## **3. LITERATURE SURVEY**

### **3.1 Machine Learning**

Machine learning is a branch of artificial intelligence that aims at solving real life engineering problems. It provides the opportunity to learn without being explicitly programmed and it is based on the concept of learning from data. It is so much ubiquitously used dozen a times a day that we may not even know it. The advantage of machine learning (ML) methods is that it uses mathematical models, heuristic learning, knowledge acquisitions and decision trees for decision making. Thus, it provides controllability, observability and stability. It updates easily by adding a new patient's record

The application of machine learning models on human disease diagnosis aids medical experts based on the symptoms at an early stage, even though some diseases exhibit similar symptoms. One of the important problems in multivariate techniques is to select relevant features from the available set of attributes. The common feature selection techniques include wrapper subset evaluation, filtering and embedded models. Embedded models use classifiers to construct ensembles, the wrapper subset evaluation method provides ranks to features based on their importance and filter methods rank the features based on statistical measurements.

A computer aided medical diagnosis system generally consists of a knowledge base and a method for solving an intended problem. On the basis of the query posted to the system, it provides assistance to the physicians in diagnosing the patients accurately. The knowledge base of such medical systems relies on the inputs that spring up from the clinical experience of field experts. Knowledge acquisition is the process of transforming human expert knowledge and skills acquired through clinical practice to software. It is quite time consuming and labour intensive task. Common methods like Case Based Reasoning (CBR) solves the knowledge acquisition problem to some extent because the past records are maintained in a database, including possible remedies, past clinical decisions, preventive measures and expected diagnostic outcome measures. During patient diagnosis, the clinical database is matched for analogous past patient's record for taking suitable decisions.

Some of the major problems faced during the development of an expert diagnosis system are: medical experts are less interested to share their knowledge with others, experience knowledge (called common sense) is practically impossible to be separated and designing a unique expert system for diagnosing all diseases is difficult.

### **3.2 Software reliability**

Software reliability is defined as the probability that a system will not have a failure over a specified period of time under specific conditions. The knowledge of software reliability is very vital in critical systems because it indicates the design perfection. In this

work, the primary aim is to enhance the software reliability of the computer aided diagnosis systems using machine learning algorithms.

To provide quality treatment and prevent misdiagnosis are the prime motivations for developing a medical diagnosis system. Diagnosing a disease of a patient accurately is a great challenge in medical field. A huge amount is spent on advanced primary health care devices based on software reliability research as they are considered as critical systems.

There are several software reliability models available in the literature; however, none of the models are perfect. An important research issue is choosing a suitable estimation model based on a specific application. One advantage of software reliability over hardware reliability is that a mechanical part surely undergoes ageing; suffer from wear and tear problem over time and usage; however software do not rust or wear out. Software reliability is a vital parameter for software quality, functionality and performance. Some common software reliability models are prediction and estimation models like bathtub curve, exponential, Putnam etc.

### **3.3 Supervised learning**

Supervised learning is the most common form of machine learning scheme used in solving the engineering problems. It can be thought as the most appropriate way of mapping a set of input variables with a set of output variables. The system learns to infer a function from a collection of labelled training data. The training dataset contains a set of input features and several instance values for respective features. The predictive performance accuracy of a machine learning algorithm depends on the supervised learning scheme. The aim of the inferred function may be to solve a regression or classification problem. There are several metrics used in the measurement of the learning task like accuracy, sensitivity, specificity, kappa value, area under the curve etc. In this work, the aim is to classify the patients as healthy or ill based on the past medical records. Before solving any engineering problem, it is vital that it is necessary to choose a suitable algorithm for the training purpose based on the type of the data. The selection of a method depends primarily on the type of the data as the field of machine learning is data driven. The next important aspect is the optimization of the chosen machine learning algorithms

### **3.4 Classification task**

Classification task is a classical problem in the field of data mining which deals with assigning a pre-specified class to an unknown data. A learning model is built based on the relationship between the predictor attribute values and the value of the target. The challenge is to correctly predict the class based on learning of past data. In machine learning, this kind of classification problems are referred to as supervised learning. Hence, we need to provide a data set containing instances with known classes and a test data set for which the class has to be determined. The success of the classification ability largely depends on the quality of data

provided for learning and also the type of machine learning algorithm used. For example, the classification techniques can be used to predict the fraud customers in a bank who apply for a loan or classify mangoes whether they are good or bad and lots of other real time applications. The most common type of classification problem is binary classification, where the target has two possible values like good or bad, yes or no etc. There are several methods for measuring the classification performance like confusion matrix, lift curve, receiver operator characteristics etc.

### **3.5 Optimization**

Every machine learning algorithm has a specific technique of learning and is based on the values of their parameters. When an algorithm is applied to solve a classification problem with a different set of parameters, the classification accuracy also differs abruptly in each case. The challenge in machine learning to find the most suitable parameter values of the algorithms that solves an engineering problem to the best possible way in terms of performance metrics. Therefore, one has to fine tune the algorithm parameters that best suits the problem. There are several optimization techniques like genetic algorithm, particle swarm optimization, Tabu search methods etc. The focus of the study is to calibrate the algorithm parameters using design of experiment method.

### **3.6 Main challenges of Machine Learning**

In short, since the main task is to select a learning algorithm and train it on some data, the two things that can go wrong are "bad algorithm" and "bad data."

1. Insufficient Quantity of Training Data
2. Non-representative Training Data
3. Poor-Quality Data
4. Irrelevant Features
5. Over fitting the Training Data
6. Under fitting the Training Data

## 4. DESIGN

### 4.1 Introduction to UML

The UML (Unified Modeling Language) is a way of visualizing a software program using a collection of diagrams. Today, UML is accepted by the Object Management Group (OMG) as the standard for modelling software development. UML stands for Unified Modelling Language. UML helped extend the original UML specification to cover a wider portion of software development efforts including agile practices.

- Improved integration between structural models like class diagrams and behaviour models like activity diagrams.
- Added the ability to define a hierarchy and decompose a software system into components and sub-components

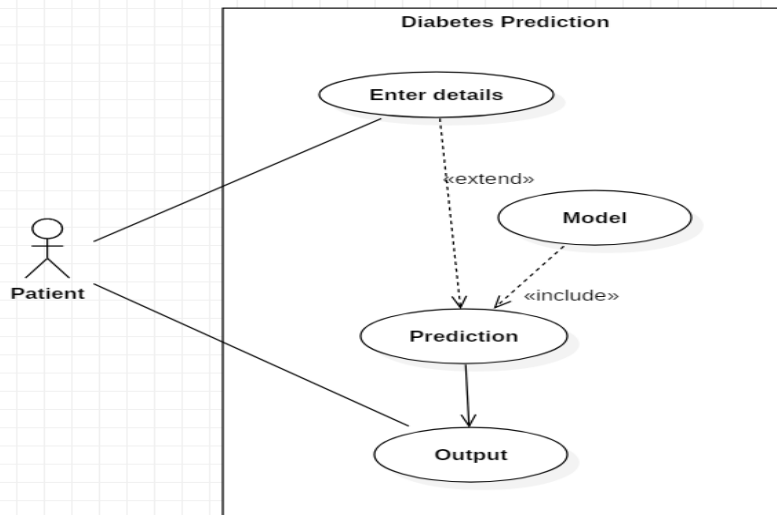
### 4.2 UML Diagrams

#### 4.2.1 Use case Diagram

A cornerstone part of the system is the functional requirements that the system fulfils. Use Case diagrams are used to analyze the system's high-level requirements. These requirements are expressed through different use cases. We notice three main components of this UML diagram:

- **Functional requirements** – represented as use cases; a verb describing an action
- **Actors** – they interact with the system; an actor can be a human being, an organization or an internal or external application
- **Relationships** between actors and use cases – represented using straight arrows

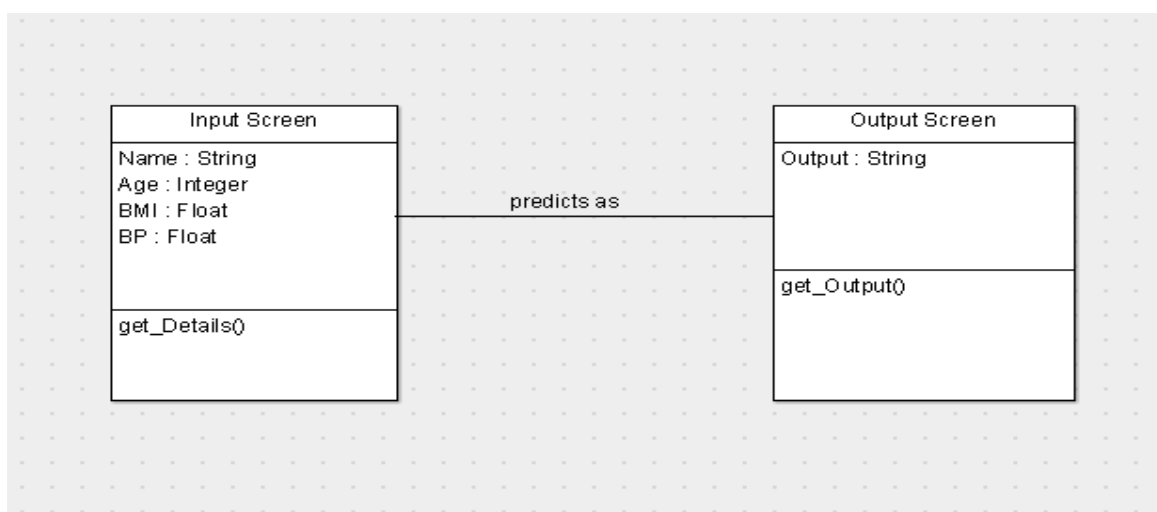




## 4.2.2 Class Diagram

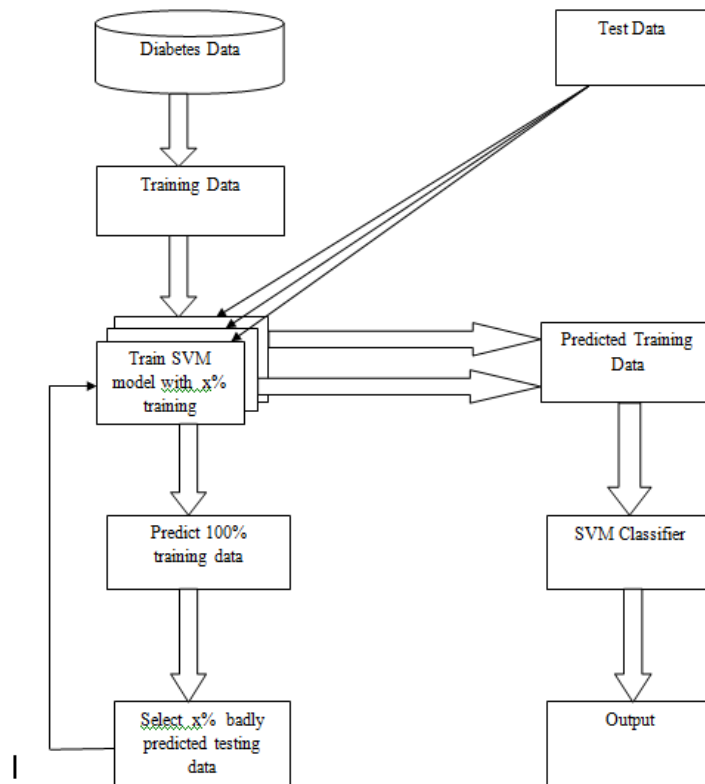
Class UML diagram is the most common diagram type for software documentation. Since most software being created nowadays is still based on the Object-Oriented Programming paradigm, using class diagrams to document the software turns out to be a common-sense solution. This happens because OOP is based on classes and the relations between them.

Class diagrams contain classes, alongside with their attributes (also referred to as data fields) and their behaviors (also referred to as member functions). More specifically, each class has 3 fields: the class name at the top, the class attributes right below the name, the class operations/behaviors at the bottom. The relation between different classes (represented by a connecting line), makes up a class diagram.



### 4.2.3 Architecture Diagram

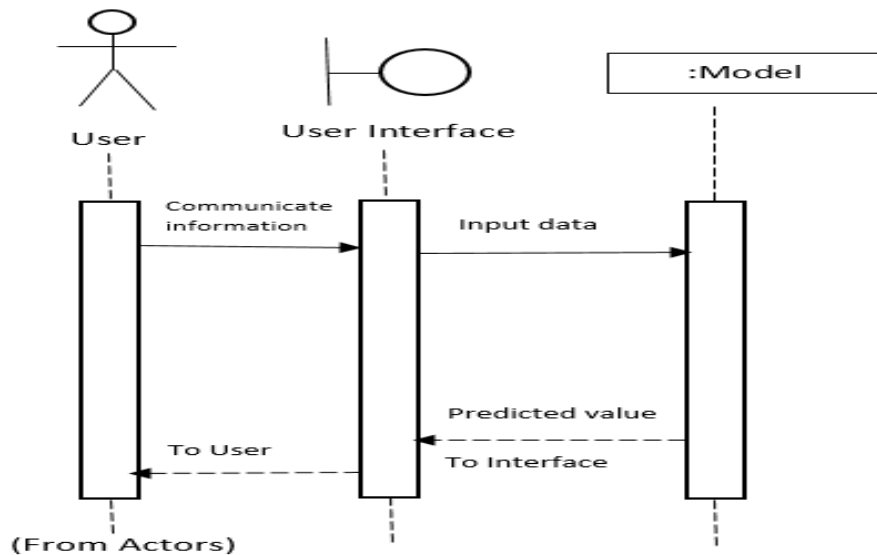
An architecture diagram is a graphical representation of a set of concepts, that are part of an architecture, including their principles, elements and components.



### 4.2.4 Sequence Diagram

Sequence diagrams are probably the most important UML diagrams among not only the computer science community but also as design-level models for business application development. Lately, they have become popular in depicting business processes, because of their visually self-explanatory nature.

As the name suggests, sequence diagrams describe the sequence of messages and interactions that happen between actors and objects. Actors or objects can be active only when needed or when another object wants to communicate with them. All communication is represented in a chronological manner. To get a better idea, check the example of a UML sequence diagram below.



## 4.2.5 Activity Diagram

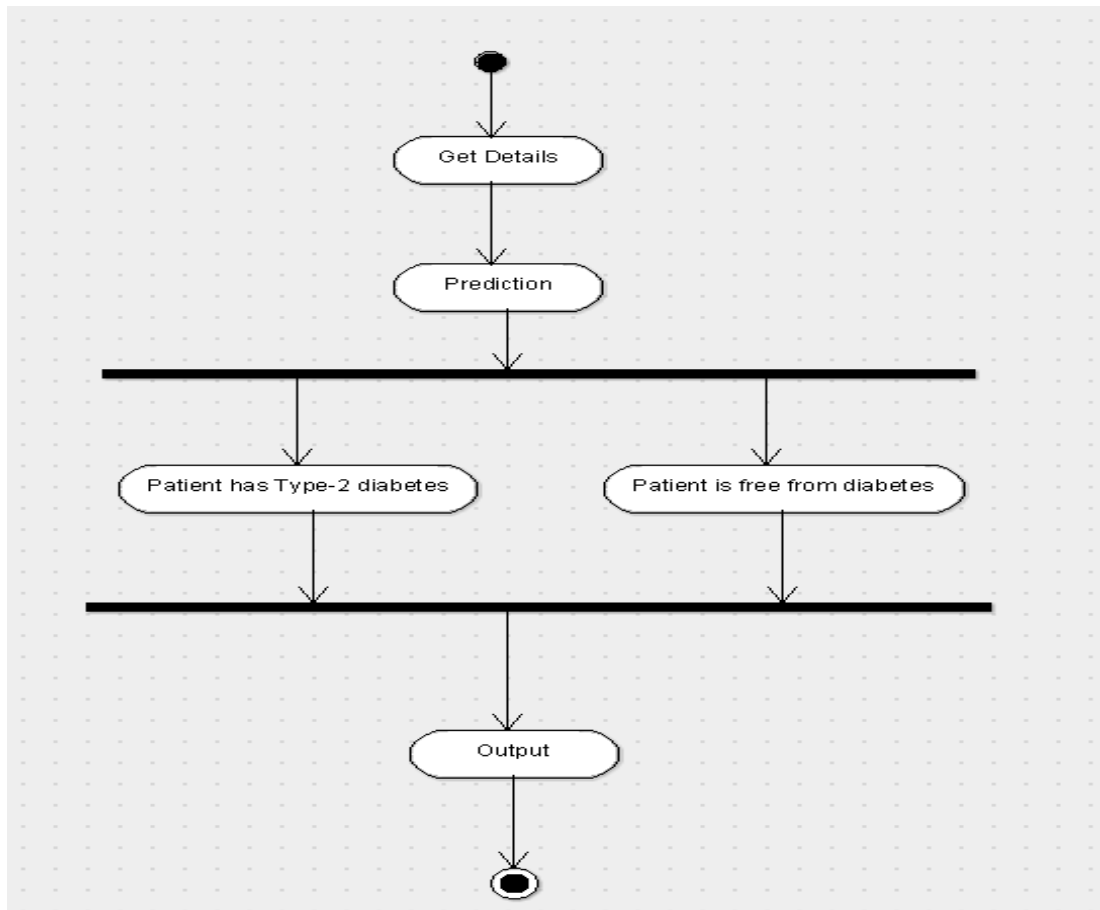
Activity diagram is another important diagram in UML to describe the dynamic aspects of the system. Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all type of flow control by using different elements such as fork, join, etc

### 4.2.5.1 Purpose of Activity Diagrams

The basic purposes of activity diagrams is similar to other diagrams. It captures the dynamic behaviour of the system. Other diagrams are used to show the message flow from one object to another but activity diagram is used to show message flow from one activity to another. Activity is a particular operation of the system. Activity diagrams are not only used for visualizing the dynamic nature of a system, but they are also used to construct the executable system by using forward and reverse engineering techniques. The only missing thing in the activity diagram is the message part. It shows different flows such as parallel, branched, concurrent, and single.

The purpose of an activity diagram can be described as –

- Draw the activity flow of a system.
- Describe the sequence from one activity to another.
- Describe the parallel, branched and concurrent flow of the system.



## 4.2.6 Modules

A module is a separate unit of software or hardware. Typical characteristics of modular components include portability, which allows them to be used in a variety of systems, and interoperability, which allows them to function with the components of other systems. The term was first used in architecture.

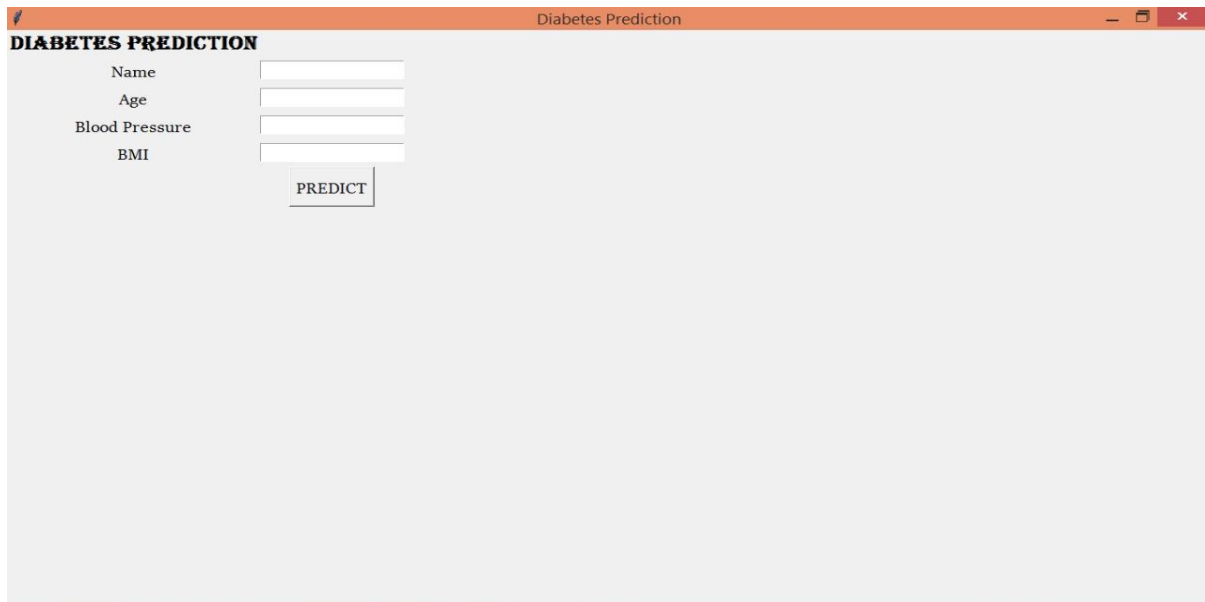
In computer programming, especially in older languages such as PL/1, the output of the language compiler was known as an object module to distinguish it from the set of source language statements, sometimes known as the source module. In mainframe systems such as IBM's OS/360, the object module was then linked together with other object modules to form a load module. The load module was the executable code that you ran in the computer.

Modular programming is the concept that similar functions should be contained within the same unit of programming code and that separate functions should be developed as separate units of code so that the code can easily be maintained and reused by different programs. Object-oriented programming is a newer idea that inherently encompasses modular programming.

## 4.2.7 User Interface

In information technology, the user interface (UI) is everything designed into an information device with which a person may interact. This can include display screens, keyboards, a mouse and the appearance of a desktop. It is also the way through which a user interacts with an application or a website. The growing dependence of many companies on web applications and mobile applications has led many companies to place increased priority on UI in an effort to improve the user's overall experience.

The UI interface is often talked about in conjunction with user experience (UX), which may include the aesthetic appearance of the device, response time and the content that is presented to the user within the context of the user interface. An increasing focus on creating an optimized user experience has led some to carve out careers as UI and UX experts. Certain languages, such as HTML and CSS, have been geared toward making it easier to create a strong user interface and experience.

A screenshot of a web application window titled "Diabetes Prediction". The window has a light gray background and a dark orange header bar. In the top left corner of the header bar, there is a small icon of a person. The title "Diabetes Prediction" is centered in the header bar. On the left side of the main content area, the text "DIABETES PREDICTION" is displayed in bold. Below this text, there are four input fields labeled "Name", "Age", "Blood Pressure", and "BMI". To the right of these fields is a button labeled "PREDICT".

(UI main)

## 5. IMPLEMENTATION

### 5.1 Data Collection

**Data collection** is the process of gathering and measuring information on targeted variables in an established systematic fashion, which then enables one to answer relevant questions and evaluate outcomes

The goal for all data collection is to capture quality evidence that allows analysis to lead to the formulation of convincing and credible answers to the questions that have been posed.

### 5.2 Data Pre Processing

Data pre-processing is a **data mining** technique that involves transforming raw data into an understandable format. Real-world data is often incomplete, inconsistent, and/or lacking in certain behaviours or trends, and is likely to contain many errors. Data pre-processing is a proven method of resolving such issues. Data pre-processing prepares raw data for further processing.

Data pre-processing is used database-driven applications such as customer relationship management and rule-based applications (like neural networks).

Data goes through a series of steps during pre-processing:

- **Data Cleaning:** Data is cleansed through processes such as filling in missing values, smoothing the noisy data, or resolving the inconsistencies in the data.
- **Data Integration:** Data with different representations are put together and conflicts within the data are resolved.
- **Data Transformation:** Data is normalized, aggregated and generalized.
- **Data Reduction:** This step aims to present a reduced representation of the data in a data warehouse.
- **Data Discretization:** Involves the reduction of a number of values of a continuous attribute by dividing the range of attribute intervals

### 5.3 Code Snippets

#### Imports

```
import numpy as np
```

```
import pandas as pd
```

```
import matplotlib.pyplot as plt
```

```

import seaborn as sns

from sklearn.preprocessing import MinMaxScaler

from sklearn.model_selection import train_test_split, GridSearchCV

from sklearn.svm import SVC

%matplotlib inline

sns.set()

#load the pima indian diabetes dataset

diabetes=pd.read_csv("E:\Machine_Learning\DataSets\diabetes.csv")

Inspect the dataset

print("diabetes shape is :",diabetes.shape)

print("Dataset Description:\n")

diabetes.describe()

diabetes shape is : (768, 9)

```

## Dataset Description:

```
diabetes shape is : (768, 9)
Dataset Description:
```

	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

```
# Visualise a table with the first rows of the dataset, to better understand the data format
```

```
print("Dataset head :\n")
```

```
diabetes.head()
```

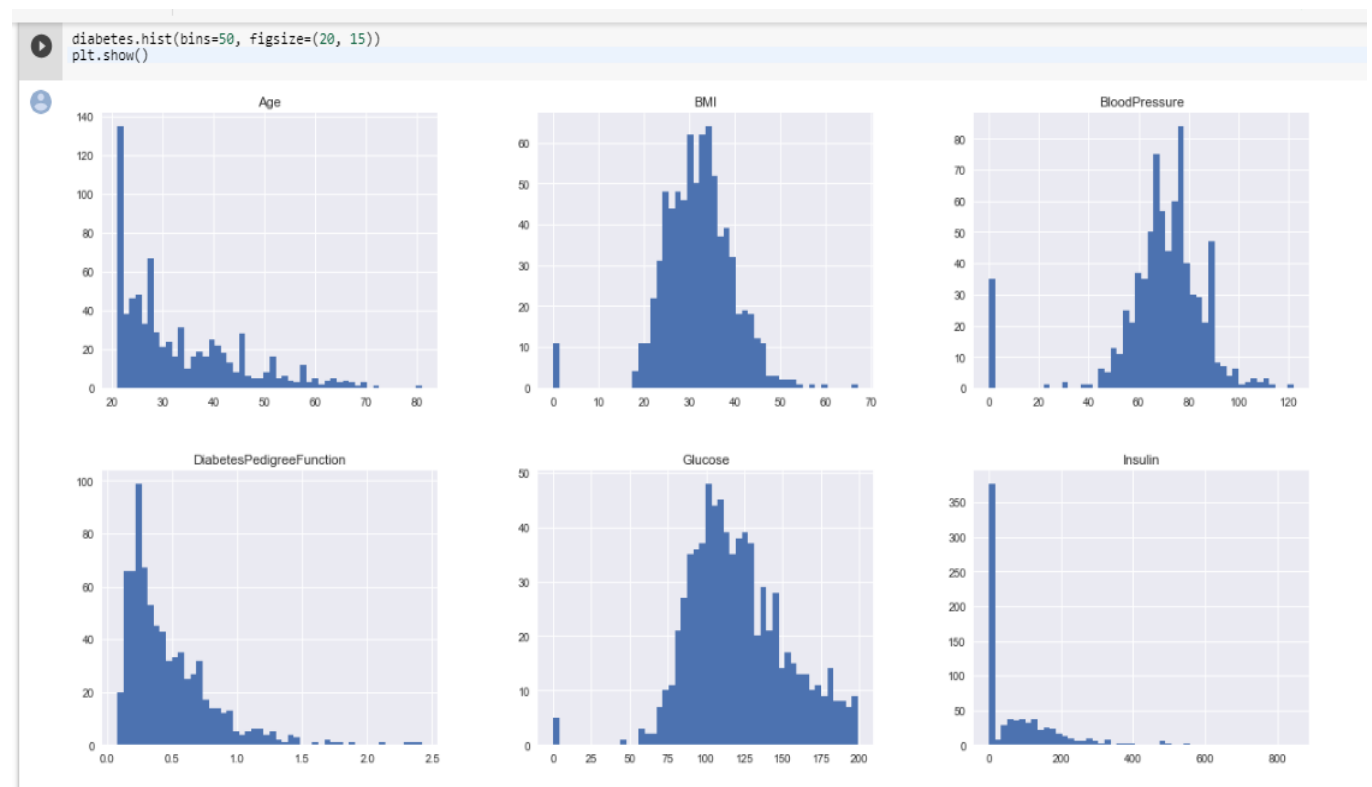
**Dataset head :**

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

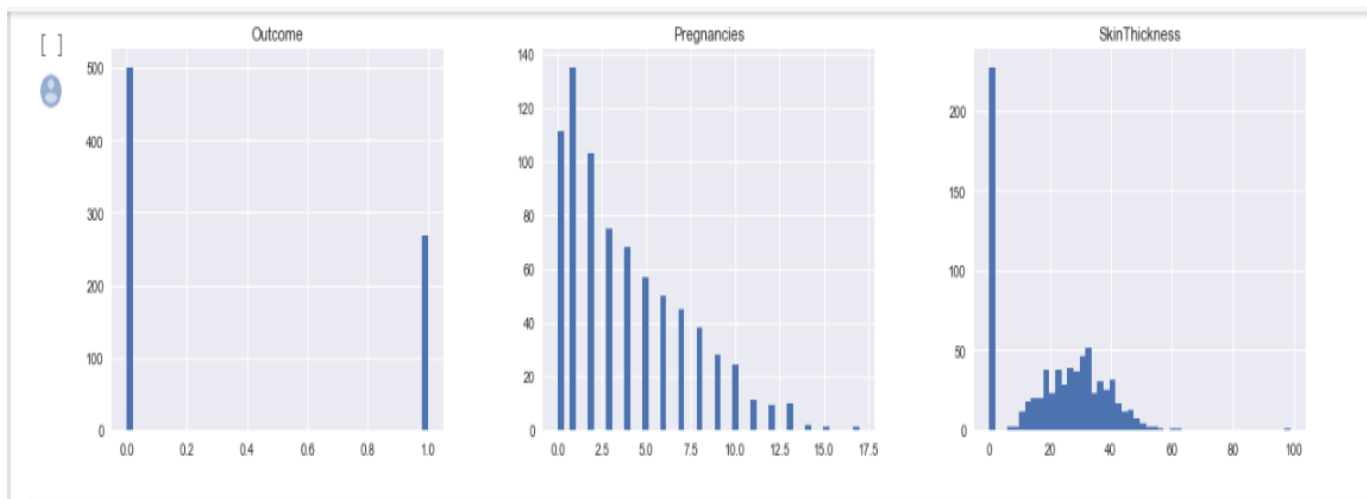
**Visualize the dataset:**

```
diabetes.hist(bins=50, figsize=(20, 15))
```

```
plt.show()
```







#print the Outcome counts 1/0

```
OutCount=diabetes.groupby("Outcome").size()
```

```
print(OutCount)
```

```
OutCount.plot(kind="bar",title="Outcome Count")
```

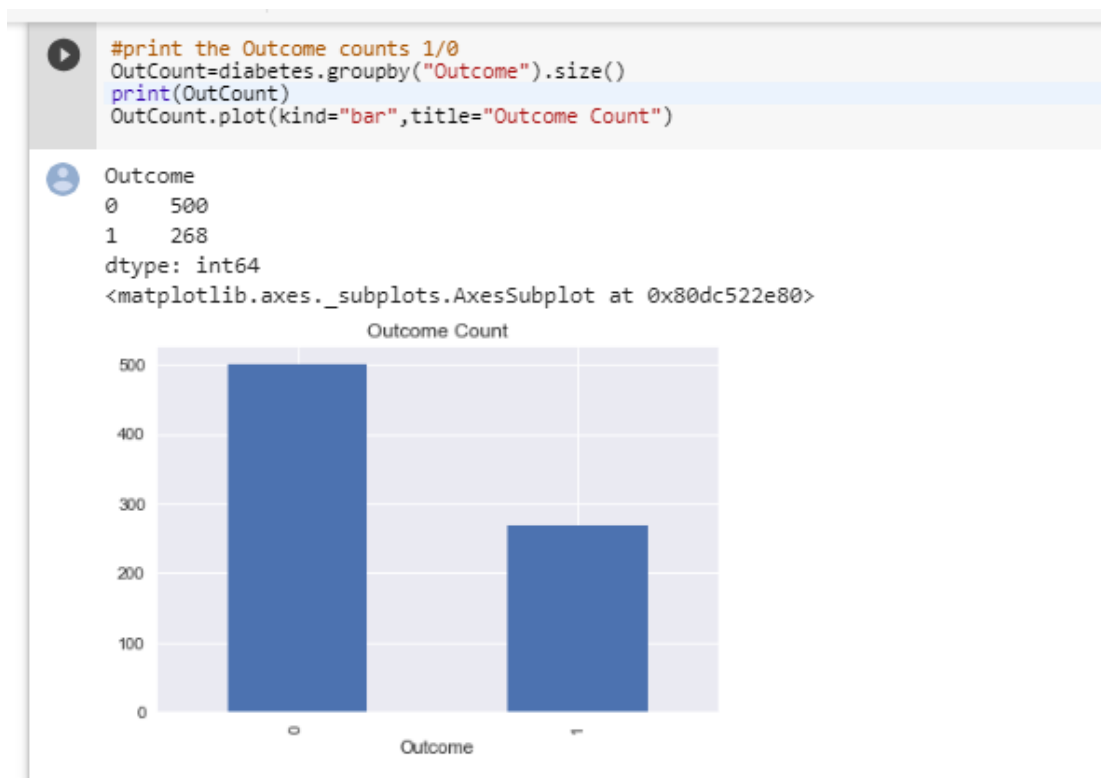
Outcome

0 500

1 268

dtype: int64

<matplotlib.axes.\_subplots.AxesSubplot at 0x80dc522e80>



## Data Correlataion Matrix

#Finding Correlation of attributes with outcome

```
corr_mat=diabetes.corr()
```

#correaltion matrix

corr\_mat

▼ Data correlation Matrix:

```
[ ] #Finding Correlation of attributes with outcome
    corr_mat=diabetes.corr()
```

```
#correaltion matrix
corr_mat
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
Pregnancies	1.000000	0.129459	0.141282	-0.081672	-0.073535	0.017683	-0.033523	0.544341	0.221898
Glucose	0.129459	1.000000	0.152590	0.057328	0.331357	0.221071	0.137337	0.263514	0.466581
BloodPressure	0.141282	0.152590	1.000000	0.207371	0.088933	0.281805	0.041265	0.239528	0.065068
SkinThickness	-0.081672	0.057328	0.207371	1.000000	0.436783	0.392573	0.183928	-0.113970	0.074752
Insulin	-0.073535	0.331357	0.088933	0.436783	1.000000	0.197859	0.185071	-0.042163	0.130548
BMI	0.017683	0.221071	0.281805	0.392573	0.197859	1.000000	0.140647	0.036242	0.292695
DiabetesPedigreeFunction	-0.033523	0.137337	0.041265	0.183928	0.185071	0.140647	1.000000	0.033561	0.173844
Age	0.544341	0.263514	0.239528	-0.113970	-0.042163	0.036242	0.033561	1.000000	0.238356
Outcome	0.221898	0.466581	0.065068	0.074752	0.130548	0.292695	0.173844	0.238356	1.000000

Visualize predictor's correlation with outcome

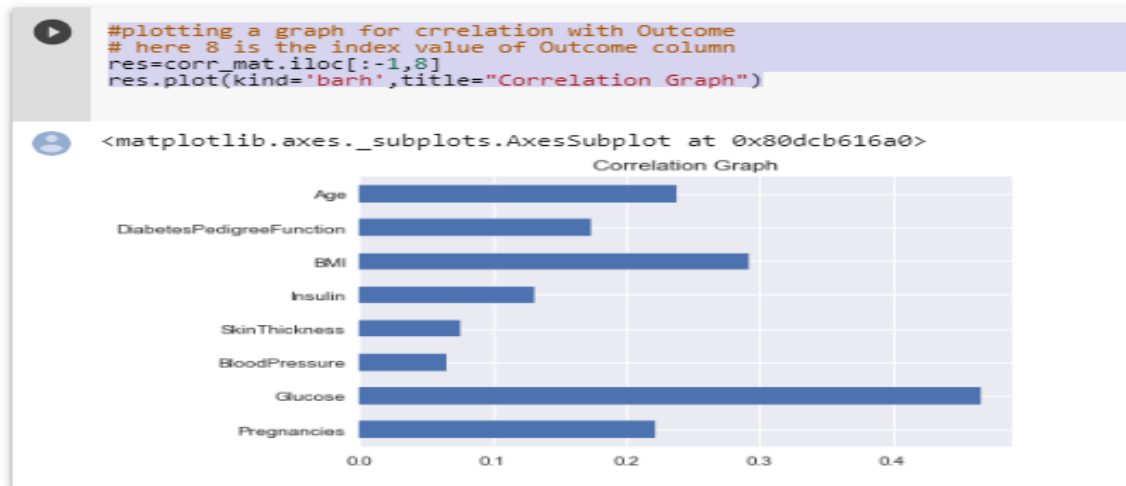
#plotting a graph for crrelation with Outcome

# here 8 is the index value of Outcome column

```
res=corr_mat.iloc[:-1,8]
```

```
res.plot(kind='barh',title="Correlation Graph")
```

#### Visualize Predictors Correlation with Outcome:



From the above Correlation graph, It can be inferred that factors like "Age, BMI and Blood Pressure" which can be measured without taking the blood sample, influence the Outcome(0/1)

From the above Correlation graph, It can be inferred that factors like "Age, BMI and Blood Pressure" which can be measured without taking the blood sample, influence the Outcome(0/1)

There are one zero value records in dataset.

#### Data Cleaning and Transformation:

```
zeros_Age=(diabetes["Age"]==0).sum()
zeros_BMI=(diabetes["BMI"]==0).sum()
zeros_BP=(diabetes["BloodPressure"]==0).sum()
print("Count of Zero values in Age : ",zeros_Age)
print("Count of Zero values in BMI : ",zeros_BMI)
print("Count of Zero values in BP : ",zeros_BP)
```

Count of Zero values in Age : 0  
Count of Zero values in BMI : 11  
Count of Zero values in BP : 35

Remove these records (zero value) from the dataset and create the required dataset for the model prediction.

## Creating dataset for model

```
#temp_ds contains all non zero records of the diabetes dataset
temp_ds=pd.DataFrame(diabetes[(diabetes["Age"]>0) & (diabetes["BMI"]>0) &
(diabetes["BloodPressure"]>0)])
main_dataset=pd.DataFrame(data=temp_ds,columns=["Age","BMI","BloodPressure","Outcome"])
print("Original dataset dimesnions(diabetes): ",diabetes.shape)
print("Original dataset without zero value records dimensions(temp_ds): ",temp_ds.shape)
print("Dataset for Model without zero value records dimensions(main_dataset): ",main_dataset.shape)
```

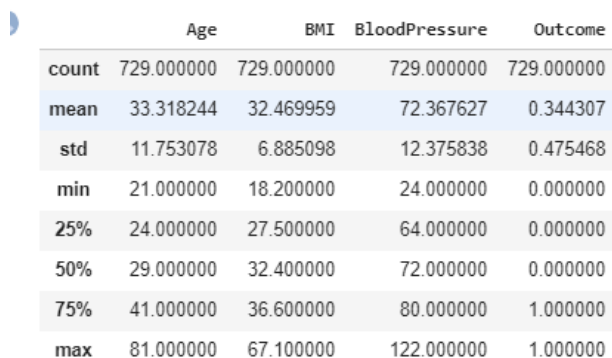
Original dataset dimesnions(diabetes): (768, 9)

Original dataset without zero value records dimensions(temp\_ds): (729, 9)

Dataset for Model without zero value records dimensions(main\_dataset): (729, 4)

main\_dataset.describe()

```
] main_dataset.describe()
```



	Age	BMI	BloodPressure	Outcome
count	729.000000	729.000000	729.000000	729.000000
mean	33.318244	32.469959	72.367627	0.344307
std	11.753078	6.885098	12.375838	0.475468
min	21.000000	18.200000	24.000000	0.000000
25%	24.000000	27.500000	64.000000	0.000000
50%	29.000000	32.400000	72.000000	0.000000
75%	41.000000	36.600000	80.000000	1.000000
max	81.000000	67.100000	122.000000	1.000000

main\_dataset contains 729 non zero records

```
out_count=main_dataset.groupby("Outcome").size()
```

```
print(out_count)
```

```
out_count.plot(kind="bar",title="Outcome label Count in main Dataset")
```

Outcome

0 478

1 251

dtype: int64

<matplotlib.axes.\_subplots.AxesSubplot at 0x80dc5d2278>



## Splitting the Dataset

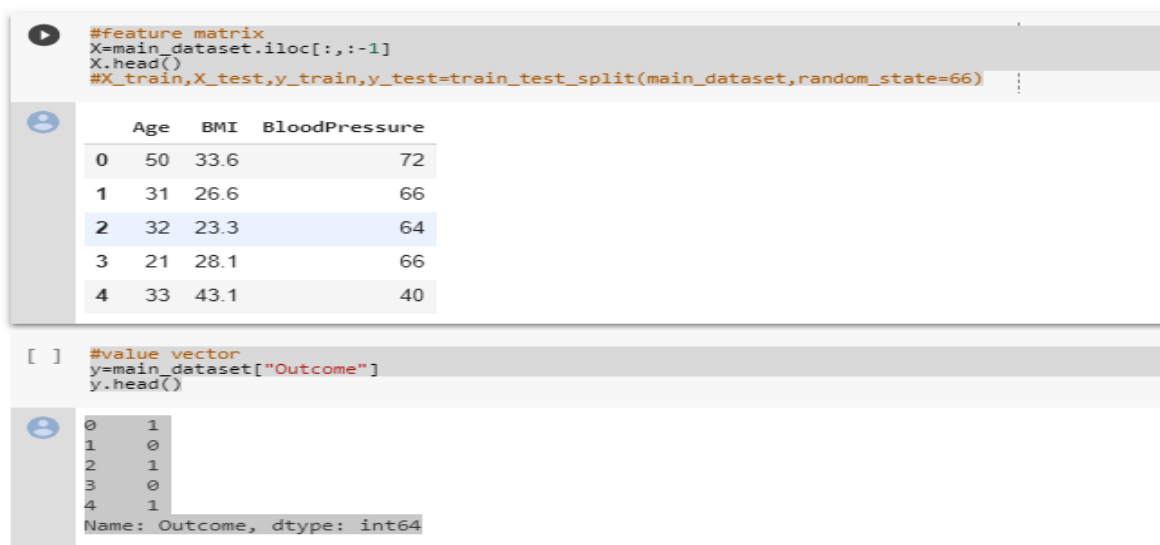
#feature matrix

X=main\_dataset.iloc[:, :-1]

X.head()

#X\_train,X\_test,y\_train,y\_test=train\_test\_split(main\_dataset,random\_state=66)

### Splitting the Dataset:



```
#value vector

y=main_dataset["Outcome"]

y.head()

0    1
1    0
2    1
3    0
4    1
Name: Outcome, dtype: int64
```

### **# Split the training dataset in 80% / 20%**

```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=60,stratify=y)

print("X_train shape : ",X_train.shape)

print("y_train shape : ",y_train.shape)

print("X_test shape : ",X_test.shape)

print("y_test shape : ",y_test.shape)
```

```
X_train shape : (583, 3)
y_train shape : (583,)
X_test shape : (146, 3)
y_test shape : (146,)
```

### **Feature Scaling**

```
scaler=MinMaxScaler()

X_train_scaled=scaler.fit_transform(X_train)

X_test_scaled=scaler.fit_transform(X_test)

print("type(X_train_scaled) : ",type(X_train_scaled))

print("type(X_test_scaled) : ",type(X_test_scaled))

type(X_train_scaled) : <class 'numpy.ndarray'>
type(X_test_scaled) : <class 'numpy.ndarray'>
```

## Scaled Values

#create a pandas dataframe to display the scaled values

```
sv=pd.DataFrame(data=X_train_scaled)
```

```
sv.head()
```

```
[ ] scaler=MinMaxScaler()  
X_train_scaled=scaler.fit_transform(X_train)  
X_test_scaled=scaler.fit_transform(X_test)  
print("type(X_train_scaled) : ",type(X_train_scaled))  
print("type(X_test_scaled) : ",type(X_test_scaled))
```

```
type(X_train_scaled) : <class 'numpy.ndarray'>  
type(X_test_scaled) : <class 'numpy.ndarray'>
```

**Scaled Values:**

```
#create a pandas dataframe to display the scaled values  
sv=pd.DataFrame(data=X_train_scaled)  
sv.head()
```

	0	1	2
0	0.200000	0.509202	0.163265
1	0.050000	0.259714	0.306122
2	0.483333	0.204499	0.591837
3	0.000000	0.089980	0.387755
4	0.233333	0.374233	0.673469

## Model Fitting

#create instance for SVC

```
svc=SVC()
```

```
svc.fit(X_train_scaled,y_train)
```

```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,  
    decision_function_shape='ovr', degree=3, gamma='auto', kernel='rbf',  
    max_iter=-1, probability=False, random_state=None, shrinking=True,  
    tol=0.001, verbose=False)
```

# use score of SVC() to find Accuracy

```
train_accuracy=svc.score(X_train_scaled,y_train)
```

```
test_accuracy=svc.score(X_test_scaled,y_test)
```

```
print("Accuracy on training set: ",train_accuracy)
```

```
print("Accuracy on testing set: ",test_accuracy)
```

Accuracy on training set: 0.6638078902229846  
Accuracy on testing set: 0.5958904109589042

## Model Tuning

Find the best parameters for SVC

```
param_grid = {  
    'C': [1.0, 10.0, 50.0],  
    'kernel': ['linear', 'rbf', 'poly', 'sigmoid'],  
    'shrinking': [True, False],  
    'gamma': ['auto', 1, 0.1],  
    'coef0': [0.0, 0.1, 0.5]  
}  
  
model_svc = SVC()  
  
grid_search = GridSearchCV(model_svc, param_grid, cv=10, scoring='accuracy')  
  
grid_search.fit(X_train_scaled, y_train)  
  
GridSearchCV(cv=10, error_score='raise',  
estimator=SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,  
decision_function_shape='ovr', degree=3, gamma='auto', kernel='rbf',  
max_iter=-1, probability=False, random_state=None, shrinking=True,  
tol=0.001, verbose=False),  
fit_params=None, iid=True, n_jobs=1,  
param_grid={'C': [1.0, 10.0, 50.0], 'kernel': ['linear', 'rbf', 'poly', 'sigmoid'], 'shrinking': [True,  
False], 'gamma': ['auto', 1, 0.1], 'coef0': [0.0, 0.1, 0.5]},  
pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',  
scoring='accuracy', verbose=0)
```

Print the best score found by GridSearch\_CV

```
best_score=grid_search.best_score_print("Best score = ",best_score)
```

Best score = 0.7066895368782161



### Apply the Parameters to the Model and train it

```
# Create an instance of the SVC algorithm using parameters
```

```
# from best_estimator_ property
```

```
best_svc = grid_search.best_estimator_
```

```
#train the model
```

```
best_svc.fit(X_train_scaled,y_train)
```

```
<class 'sklearn.svm.classes.SVC'>  
SVC(C=50.0, cache_size=200, class_weight=None, coef0=0.5,  
    decision_function_shape='ovr', degree=3, gamma=1, kernel='poly',  
    max_iter=-1, probability=False, random_state=None, shrinking=True,  
    tol=0.001, verbose=False)
```

### Check Accuracy

```
# use score of SVC() to find Accuracy
```

```
best_train_accuracy=best_svc.score(X_train_scaled,y_train)
```

```
best_test_accuracy=best_svc.score(X_test_scaled,y_test)
```

```
print("Best Accuracy on training set: ",best_train_accuracy)
```

```
print("Best Accuracy on testing set: ",best_test_accuracy)
```

```
Best Accuracy on training set: 0.7186963979416809
```

```
Best Accuracy on testing set: 0.6438356164383562
```

### Make a Prediction

```
# create a new (fake) person by taking the values of Age,BMI and BloodPressure
```

```
INPUT: new_person = pd.DataFrame([[50,33.6,72]])
```

```
# Scale those values like the others using MinMaxScaler
```

```
new_person_scaled = scaler.transform(new_person)
```

```
#predict the outcome
```

```
#here "1" means "person is likely to have type-2 diabetes"
```

```
# 0 means "person doesn't have type-2 diabetes"
```

```
prediction = best_svc.predict(new_person_scaled)
```

```
type(prediction)
```

```
numpy.ndarray
```

```
print("Prediction value : ",prediction[0])
```

```
Prediction value : 1
```

```
if(prediction==1):
```

```
    print("You are likely to have type-2 diabetes.")
```

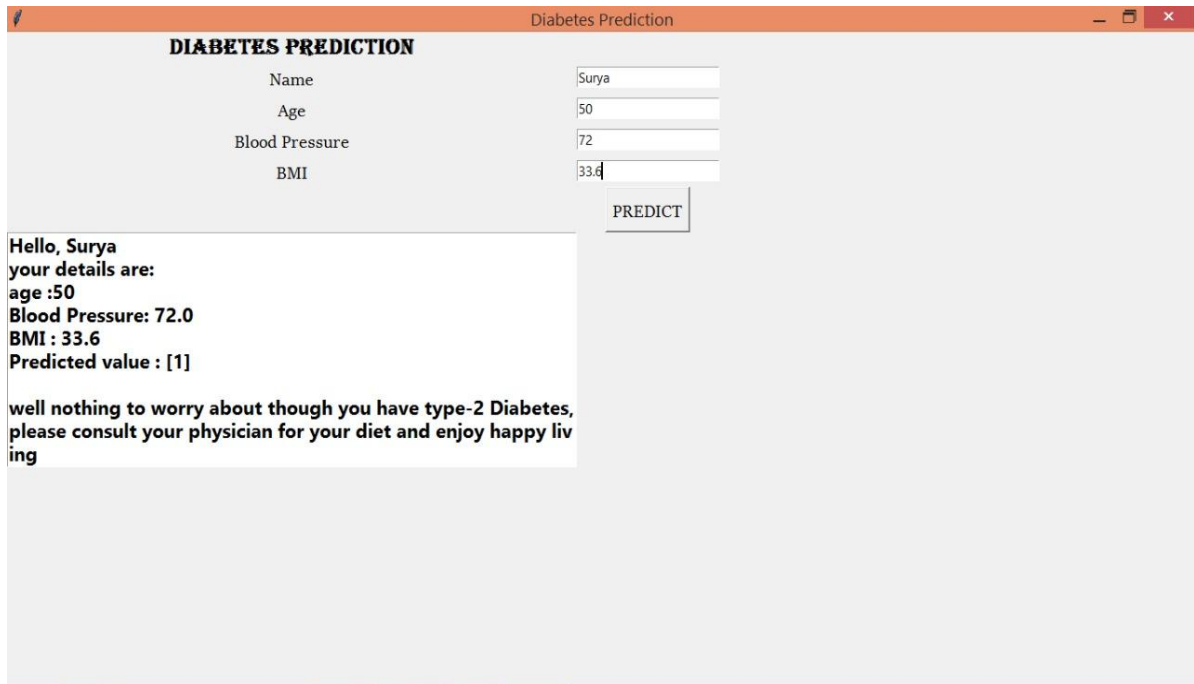
```
else:
```

```
    print(" You don't have type-2 diabetes.")
```

### **Output:**

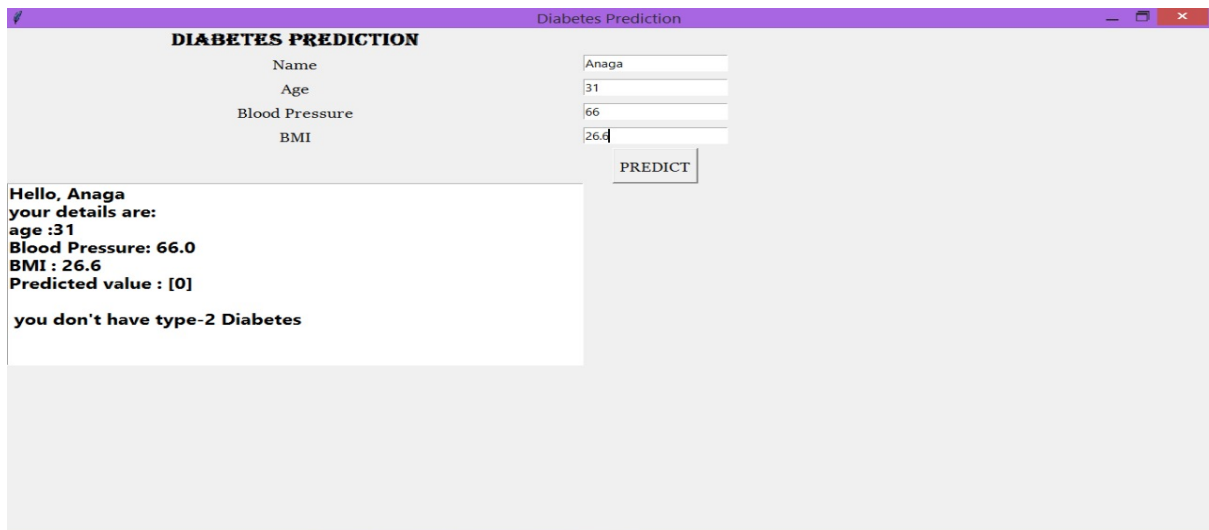
```
You are likely to have type-2 diabetes.
```

## UI Outputs:



The screenshot shows a web application window titled "Diabetes Prediction". The header is "DIABETES PREDICTION". The form contains four input fields: "Name" (Surya), "Age" (50), "Blood Pressure" (72), and "BMI" (33.6). A "PREDICT" button is located below the BMI field. The output area on the left displays the following text:

Hello, Surya  
your details are:  
age :50  
Blood Pressure: 72.0  
BMI : 33.6  
Predicted value : [1]  
  
well nothing to worry about though you have type-2 Diabetes,  
please consult your physician for your diet and enjoy happy living



The screenshot shows the same web application window titled "Diabetes Prediction". The header is "DIABETES PREDICTION". The form contains four input fields: "Name" (Anaga), "Age" (31), "Blood Pressure" (66), and "BMI" (26.6). A "PREDICT" button is located below the BMI field. The output area on the left displays the following text:

Hello, Anaga  
your details are:  
age :31  
Blood Pressure: 66.0  
BMI : 26.6  
Predicted value : [0]  
  
you don't have type-2 Diabetes

## 6. TESTING

### 6.1 Testing

**Software testing** is an investigation conducted to provide stakeholders with information about the quality of the software product or service under test. Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation. Test techniques include the process of executing a program or application with the intent of finding software bugs (errors or other defects), and verifying that the software product is fit for use.

Software testing involves the execution of a software component or system component to evaluate one or more properties of interest. In general, these properties indicate the extent to which the component or system under test

- meets the requirements that guided its design and development,
- responds correctly to all kinds of inputs,
- performs its functions within an acceptable time,
- is sufficiently usable,
- can be installed and run in its intended environments and
- achieves the general result its stakeholders desire.

### 6.2 Validation

Data validation is intended to provide certain well-defined guarantees for fitness, accuracy, and consistency for any of various kinds of user input into an application or automated system. Data validation rules can be defined and designed using any of various methodologies, and be deployed in any of various contexts

#### 6.2.1 Kinds of Validation

1. Data type validation;
2. Range and constraint validation;
3. Code and Cross-reference validation; and
4. Structured validation

### **6.2.1.1 Data-type validation**

Data type validation is customarily carried out on one or more simple data fields. The simplest kind of data type validation verifies that the individual characters provided through user input are consistent with the expected characters of one or more known primitive data types; as defined in a programming language or data storage and retrieval mechanism as well as the specification of the following primitive data types: 1) integer; 2) float (decimal); or 3) string.

### **6.2.1.2 Simple range and constraint validation**

Simple range and constraint validation may examine user input for consistency with a minimum/maximum range, or consistency with a test for evaluating a sequence of characters, such as one or more tests against regular expressions. For example, a US phone number should have 10 digits and no letters or special characters.

### **6.2.1.3 Code and cross-reference validation**

Code and cross-reference validation includes tests for data type validation, combined with one or more operations to verify that the user-supplied data is consistent with one or more external rules, requirements, or validity constraints relevant to a particular organization, context or set of underlying assumptions. These additional validity constraints may involve cross-referencing supplied data with a known look-up table or directory information service.

### **6.2.1.4 Structured validation**

Structured validation allows for the combination of any of various basic data type validation steps, along with more complex processing. Such complex processing may include the testing of conditional constraints for an entire complex data object or set of process operations within a system.

## **6.3 Test Cases**

A **TEST CASE** is a set of conditions or variables under which a tester will determine whether a system under test satisfies requirements or works correctly.

The process of developing test cases can also help find problems in the requirements or design of an application.

Following are the ranges of each attributes:

Age: 21 – 41 years

BP: 24 – 122

BMI: 18.2-67.1

**Test Case 1:**

**Input:** 50, 72, 33.6

**Output:** 1

**Test Case 2:**

**Input:** 31, 66, 26.6

**Output:** 0

## 7. CONCLUSIONS

### 7.1 Limitations

- Accuracy is up-to 64.38%.
- Results are based on Correlation analysis of Pima Indians Diabetic Dataset.
- Works only on windows OS.

### 7.2 Future Work:

- The app would consist of details of the near by diagnostic centres based on one's location along with their price list.
- A clean record of user's history is maintained along with credentials.
- Would associate with medical shops for delivery of ordered medicines from the associated pharmaceutical stores.

### 7.3 Conclusion

In this project, we have used Pima Indian Diabetes Dataset from the machine learning laboratory at University of California, Irvine. All the patients data are trained by using SVM. Choosing the best kernel for SVM plays an important role in the outcome. In the proposed work, SVM with Radial basis function kernel is used for classification. The performance parameter like classification accuracy of the SVM and RBF have found to be high, thus making it a good option for the classification process

## 8. REFERENCES

1. [Type 2 Diabetes: Causes and Symptoms](#)
2. <http://scikit-learn.org/stable/modules/preprocessing.html>
3. <https://towardsdatascience.com/machine-learning-for-diabetes-562dd7df4d42>
4. <https://www.geeksforgeeks.org/ml-handling-missing-values/amp/>
5. <https://www.svm-tutorial.com/2014/11/svm-understanding-math-part-1/>