DIABETES ANALYSIS AND PREDICTION PROJECT REPORT

Submitted in fulfillment of the requirements for the award of certificate of

Training

In

Python with Machine Learning

to

National Institute of Electronics and Information Technology
(**NIELIT**), **Chandigarh**

Submitted to :

Dr. Anita Budhiraja

Dr. Sarwan Singh

Submitted by :

Aashish Kumar (Roll No.- 25)

Anurag Verma (Roll No.- 24)

ABOUT PROJECT

Machine learning in medicine has recently made headlines. Google has developed a machine learning algorithm to help identify cancerous tumors on mammograms. Stanford is using a deep learning algorithm to identify skin cancer. A recent JAMA article reported the results of a deep machine-learning algorithm that was able to diagnose diabetic retinopathy in retinal images. It's clear that machine learning puts another arrow in the quiver of clinical decision making.

Still, machine learning lends itself to some processes better than others. Algorithms can provide immediate benefit to disciplines with processes that are reproducible or standardized. Also, those with large image datasets, such as radiology, cardiology, and pathology, are strong candidates. Machine learning can be trained to look at images, identify abnormalities, and point to areas that need attention, thus improving the accuracy of all these processes. Long term, machine learning will benefit the family practitioner or internist at the bedside. Machine learning can offer an objective opinion to improve efficiency, reliability, and accuracy.

We have applied machine learning to predict whether the patient has diabetes or not.

PYTHON

INTRODUCTION

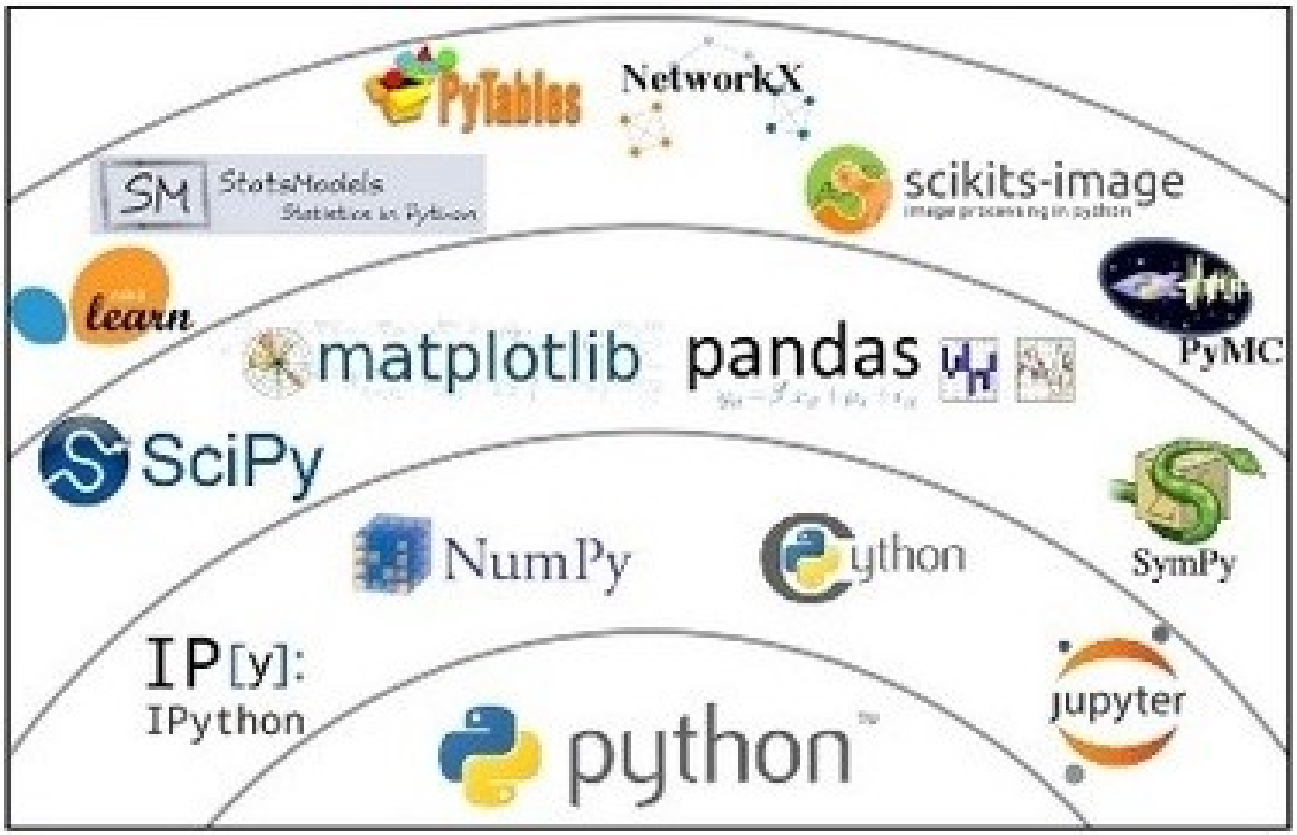
Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

- **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- **Python is Interactive** − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- **Python is a Beginner's Language** − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands. It is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.

It supports functional and structured programming methods as well as OOP. It can be used as a scripting language or can be compiled to byte-code for building large applications. It provides very high-level dynamic data types and supports dynamic type checking.IT supports automatic garbage collection.It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

PYTHON ECOSYSTEM



LIBRARIES USED IN THE PROJECT –

- Pandas
- Matplotlib
- Seaborn
- Tkinter
- Scikit – Learn

PANDAS

Pandas is an open-source, BSD-licensed Python library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

In 2008, developer Wes McKinney started developing pandas when in need of high performance, flexible tool for analysis of data.

Prior to Pandas, Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data — load, prepare, manipulate, model, and analyze.

Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

Key Features of Pandas -

- Fast and efficient DataFrame object with default and customized indexing.
- Tools for loading data into in-memory data objects from different file formats.
- Data alignment and integrated handling of missing data.
- Reshaping and pivoting of date sets.
- Label-based slicing, indexing and subsetting of large data sets.
- Columns from a data structure can be deleted or inserted.
- Group by data for aggregation and transformations.
- High performance merging and joining of data.
- Time Series functionality.

MATPLOTLIB

Matplotlib is a plotting library for Python. It is used along with NumPy to provide an environment that is an effective open source alternative for MatLab. It can also be used with graphics toolkits like PyQt and wxPython.

Matplotlib module was first written by John D. Hunter. Since 2012, Michael Droettboom is the principal developer.

Matplotlib is probably the single most used Python package for 2D-graphics. It provides both a very quick way to visualize data from Python and publication-quality figures in many formats. We are going to explore matplotlib in interactive mode covering most common cases.

SEABORN

Seaborn is an open source, BSD-licensed Python library providing high level API for visualizing the data using Python programming language. Seaborn is a visualization library in Python. It is built on top of Matplotlib.

It is summarized that if Matplotlib "tries to make easy things easy and hard things possible", Seaborn tries to make a well-defined set of hard things easy too."

Seaborn helps resolve the two major problems faced by Matplotlib; the problems are−

- Default Matplotlib parameters
- Working with data frames

As Seaborn compliments and extends Matplotlib, the learning curve is quite gradual. If you know Matplotlib, you are already half way through Seaborn.

Important Features of Seaborn

Seaborn is built on top of Python's core visualization library Matplotlib. It is meant to serve as a complement, and not a replacement. However, Seaborn comes with some very important features. Let us see a few of them here. The features help in − built in themes for styling matplotlib graphics, visualizing univariate and bivariate data, fitting in and visualizing linear regression models, plotting statistical time series data, Seaborn works well with NumPy and Pandas data structures, it comes with built in themes for styling Matplotlib graphics.

Tkinter

Tkinter is the standard GUI library for Python. The name *Tkinter* comes from *Tk interface*. Tkinter was written by Fredrik Lundh. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

Creating a GUI application using Tkinter is an easy task. All you need to do is perform the following steps −

- Import the *Tkinter* module.
- Create the GUI application main window.
- Add one or more of the above-mentioned widgets to the GUI application.
- Enter the main event loop to take action against each event triggered by the user.

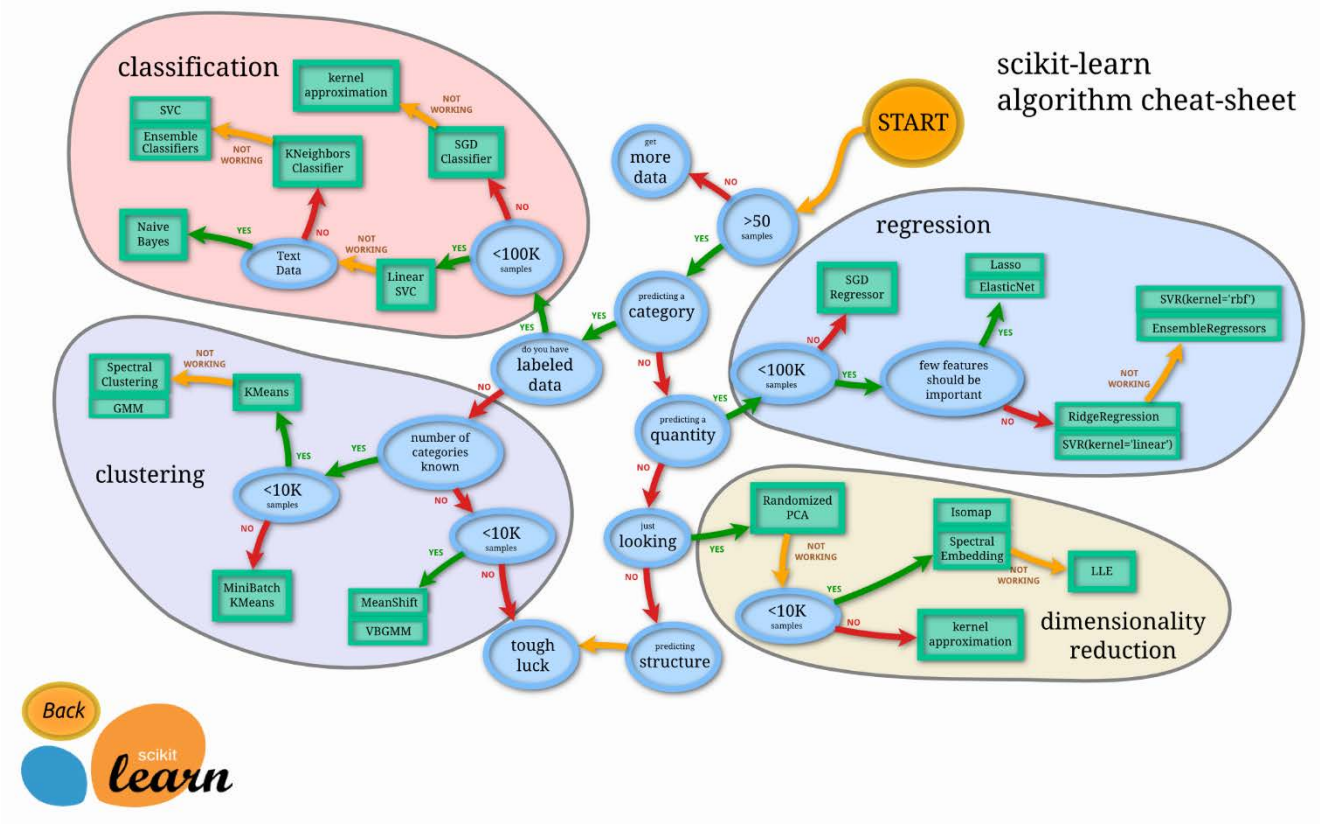
The **Tkinter** module is the standard Python interface to the Tk GUI toolkit from Scriptics (formerly developed by Sun Labs).

Both Tk and Tkinter are available on most Unix platforms, as well as on Windows and Macintosh systems. Starting with the 8.0 release, Tk offers native look and feel on all platforms.

Tkinter consists of a number of modules. The Tk interface is provided by a binary extension module named **_tkinter**. This module contains the low-level interface to Tk, and should never be used directly by application programmers. It is usually a shared library (or DLL), but might in some cases be statically linked with the Python interpreter.

The public interface is provided through a number of Python modules. The most important interface module is the **Tkinter** module itself.

SCIKIT– Learning



Scikit-learn is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, *k* means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

Scikit-learn is probably the most useful library for machine learning in Python. It is on NumPy, SciPy and matplotlib, this library contains a lot of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction. Note that *scikit-learn is used to build models*. It should not be used for reading the data, manipulating and summarizing it. There are better libraries for that (e.g. NumPy, Pandas etc.)

Machine learning is where computational and algorithmic skills of data science meet the statistical thinking of data science,

- The result is a collection of approaches to inference and data exploration that are *not about effective theory* so much as *effective computation*.

- Better to think of machine learning as a *means of building models of Data*

- Machine learning along with entire Data Science ecosystem is trying to make this mathematical, modelbased "learning" as same as "learning" exhibited by the human brain.

Why Machine Learning Matters ?

With the rise in big data, machine learning has become a key technique
for solving problems in areas, such as:

• Computational finance, *for credit scoring and algorithmic trading.*

• Image processing and computer vision, *for face recognition, motion detection, and object detection.*

• Computational biology, *for tumor detection, drug discovery, and DNA sequencing.*

• Energy production, *for price and load forecasting.*

• Automotive, aerospace, and manufacturing, *for predictive maintenance.*

• Natural language processing, *for voice recognition applications.*

PROJECT - DIABETES ANALYSIS AND PREDICTION

In this project we will analyse and predict whether the patients have diabetes or not. So first of all we will read the csv file of patients from which we get the data of the patients which contains the following columns-

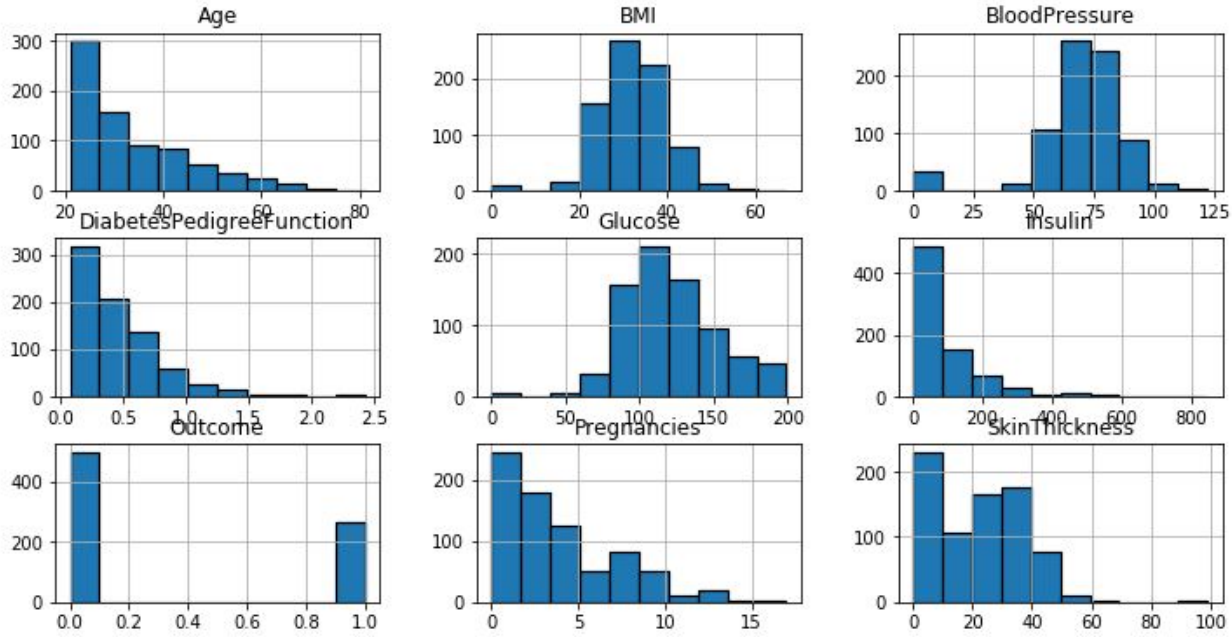
- Pregnancies
- Blood pressure
- Skin thickness
- Insulin
- BMI
- Diabetes Pedigree Function
- Age
- Outcome

Sampled datasheet is given below-

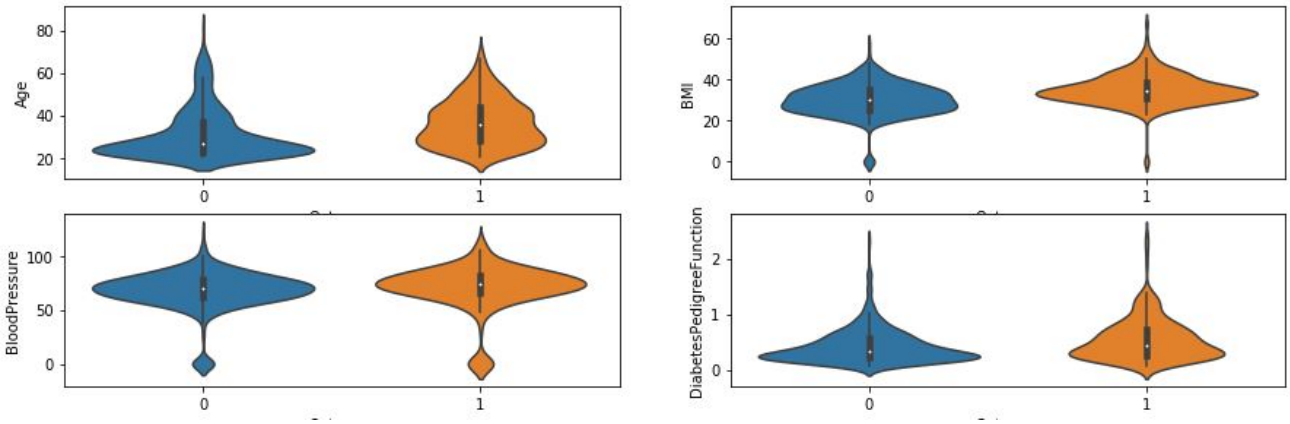
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
5	5	116	74	0	0	25.6	0.201	30	0
6	3	78	50	32	88	31.0	0.248	26	1
7	10	115	0	0	0	35.3	0.134	29	0

With the help of this datasheet we analysed the data using various graphs like –

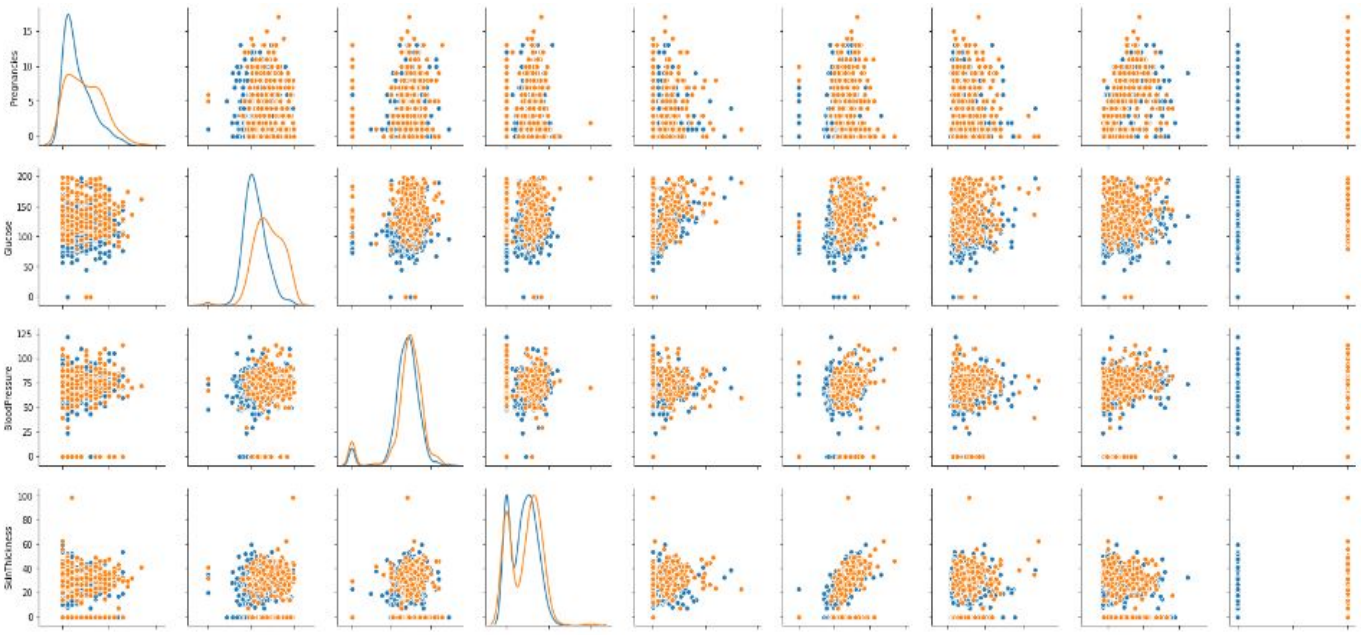
- Histogram plot



- Violin plot



- Scatter plot



The diagonal grouping of all pairs of attributes, suggests a low correlation and poor predictable relationship. That's why it will be difficult to get a high accuracy.

Now we have performed the following standard method for prediction-

1. Choose a class of model by importing the appropriate estimator class from Scikit-Learn.

2. Choose model hyperparameters by instantiating this class with desired values.

3. Arrange data into a features matrix and target vector

4. Fit the model to your data by calling the fit() method of the model instance.

5. Apply the model to new data:
   - For supervised learning, often we predict labels for unknown data using the predict() method.
   - For unsupervised learning, we often transform or infer properties of the data using the transform() or predict() method .

```
#Splitting the data in train and test part
#train = 75%
#test= 25%

train,test=train_test_split(diab,test_size=0.25,random_state=0,stratify=diab['Outcome'])
train_X=train[train.columns[:8]]              # Feature Matrix
train_Y=train['Outcome']                      # Target Matrix
test_X=test[train.columns[:8]]
test_Y=test['Outcome']
```

Linear SVM

```
model=svm.SVC(kernel='linear')
model.fit(train_X,train_Y)
prediction=model.predict(test_X)
print('Accuracy of linear svm =',metrics.accuracy_score(prediction,test_Y))
```

```
Accuracy of linear svm = 0.7708333333333334
```

We have applied the following ML algorithms :-

- Linear SVM
- RBF SVM
- Decision Tree Classifier
- Logistic Regression
- K-Neighbour Classifier

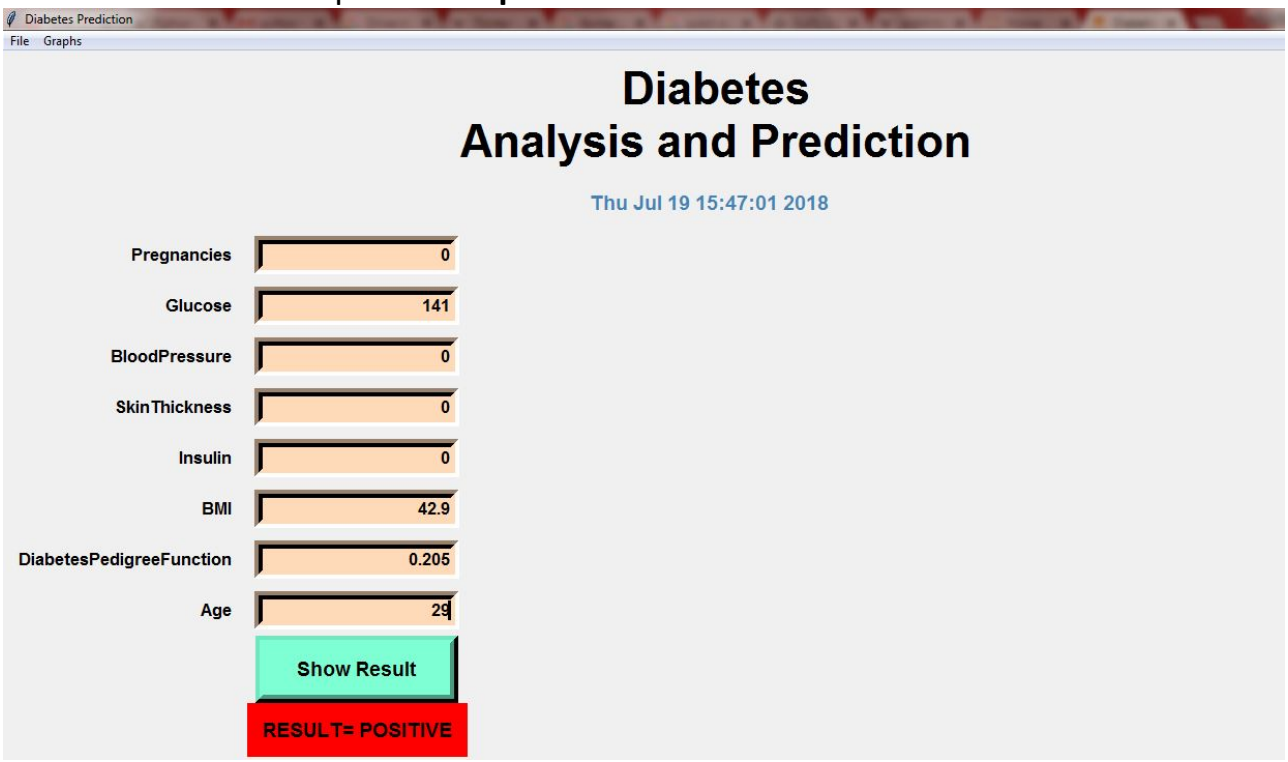
<u>Accuracy of above algorithms</u> :-

- Accuracy of linear SVM = 77%
- Accuracy of RBF SVM = 65%
- Accuracy of Decision Tree Classifier = 77%
- Accuracy of Logistic Regression = 75%
- Accuracy of K-Neighbour Classifier (i=7) = 79%

Therefore we have chosen K-Neighbour Classifier (i=7) for our prediction.

GUI APPLICATION USING TKINTER

Patient with diabetes prediction **positive** :-



Patient with diabetes prediction **negative** :-