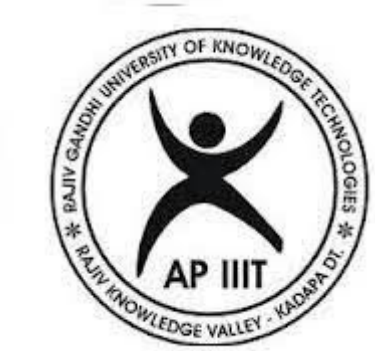
DIABETES PREDICTIVE SYSTEM

BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE AND ENGINEERING



Rajiv Gandhi University of Knowledge Technologies
R.K.VALLEY

Submitted by:-

P.Baby(R170968)
R.Sai Nandini(R171135)

Under the Esteemed guidance of Mr. M. Muni Babu

Department of Computer Science and Engineering

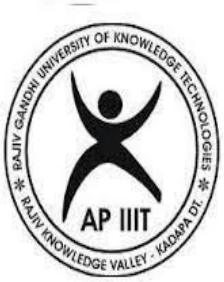
RGUKT RK Valley

DECLARATION

We here by declare that the report of the B.Tech Minor Project Work entitled as **“DIABETES PREDECTIVE SYSTEM ”** which is being submitted to Rajiv Gandhi University of Knowledge Technologies, R. K. Valley, in partial fulfillment of the requirements for the award of Degree of Bachelor of Technology in Computer Science and Engineering, is a bonafide report of the work carried out by us. The material contained in this report has not been submitted to any university or institution for award of any degree.

P.Baby(R170968)
R.Sai Nandini(R171135)
Dept. of Computer Science and Engineering.

RAJIV GANDHI UNIVERSITY OF KNOWLEDGE TECHNOLOGIES



(A.P.Government Act 18 of 2008)
IIIT RK VALLEY, RGUKT-AP
Department of Computer Science and Engineering

CERTIFICATE FOR PROJECT COMPLETION

This is to certify that the project entitled as **"DIABETES PREDICTIVE SYSTEM"** is submitted by P.Baby(R170968) R.Sai Nandini (R171135) ,under our guidance and supervision for the partial fulfillment for the degree of Bachelor of Technology in Computer Science and Engineering during the academic semester -2 in the academic year of 2021-2022 at IIIT, RK VALLEY RGUKT-AP. To the best of my knowledge, the result embodied in this dissertation work have not been submitted to any University or Institute for the award of any degree or diploma.

Project Internal Guide	**Head of the Department**
Mr. M. Muni Babu	Mr. Harinadh
Assistant Professor	HoD of C.S.E.

INDEX

Abstract

The diabetes is one of lethal diseases in the world. In additional, it is an inventor of various varieties of disorders for example: coronary failure, blindness, urinary organ diseases etc. In such case the patient is required to visit a diagnostic center, to get their reports after consultation. Due to this every time they have to spend their time and currency. But with the growth of Machine Learning methods we have got the flexibility to search out an answer to the current issue, we have got advanced system mistreatment information processing that has the ability to forecast whether the patient has polygenic illness or not. Furthermore, forecasting the sickness initially ends up in providing the patients before it becomes vital. Information withdrawal has the flexibility to remove unseen data from a large quantity of diabetes associated information. The aim of this analysis is to develop a system which might predict the diabetic risk level of a patient with a better accuracy. Model development is based on categorization methods such as KnearestNeighbor and SVM algorithms. For KNearestNeighbor, the models give precisions of 79% and 77% for Support Vector Machine. Outcomes show a significant accuracy of the methods.

INTRODUCTION

Diabetes is a situation which causes deficiency due to less amount of insulin in the blood. Warning sign of high blood sugar results in frequent urination, feeling thirsty, increased hunger. If it is not medicated, it will lead to many difficulties. This difficulty may leads to death. When there is a rise in sugar level within the blood, it is referred to as prior diabetes. The effectiveness of the decision support system is recognized by its accuracy. Therefore, the objective is to build a decision support system to predict and diagnose a certain disease with extreme amount of precision. The AI consists of ML, which is its subfield that resolves the real world difficulties by providing learning capability to workstation without supplementary program writing.

MOTIVATION

There has been a drastic increase in rate of people suffering from diabetes since a decade. Current human lifestyle is the main reason behind growth in diabetes. In current medical diagnosis method, there can be three different types of errors:

1. The false-negative type in which a patient in reality is already a diabetic patient but test results tell that the person is not having diabetes.

2. The false-positive type. In this type, patient in reality is not a diabetic patient but test reports say that he/she is a diabetic patient.

3. The third type is unclassifiable type in which a system cannot diagnose a given case. This happens due to insufficient knowledge extraction from past data, a given patient may get predicted in an unclassified type.

However, in reality, the patient must predict either to be in diabetic category or non-diabetic category. Such errors in diagnosis may lead to unnecessary treatment or no treatment at all when required. In order to avoid or reduce severity of such impact, there is a need to create a system using machine learning algorithm and data mining techniques which will provide accurate results and reduce human efforts.

PROPOSED SYSTEM

1. Data Collection

Before collecting the data.we need to import all the libraries which we are going to use in this model

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn import svm #support vector machine
from sklearn.metrics import accuracy_score
import seaborn as sns
```

Then we collect the data. We have already downloaded the data from Kaggle data set.

data collection

```
diabetes_dataset=pd.read_csv("diabetes.csv")
diabetes_dataset["Outcome"].value_counts()
diabetes_dataset.head(1000)
```

	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
...	...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

768 rows × 9 columns

2. Data Preprocessing

This phase of the model handles the inconsistent data in order to get more accurate and precise results. This dataset contains missing values.

So we inputed missing values for few selected attributes like Glucose level, Blood Pressure, SkinThickness, BMI and Age because these attributes cannot have values zero. Then we scale the dataset to normalize all values.

3. Data analysis

In this step we are going to analyze the data to build a model which predicts whether a person is having diabetes or not.

```
sns.countplot(x="Outcome",data=dataset)
```

```
Out[1]: <AxesSubplot:xlabel='Outcome', ylabel='count'>
```

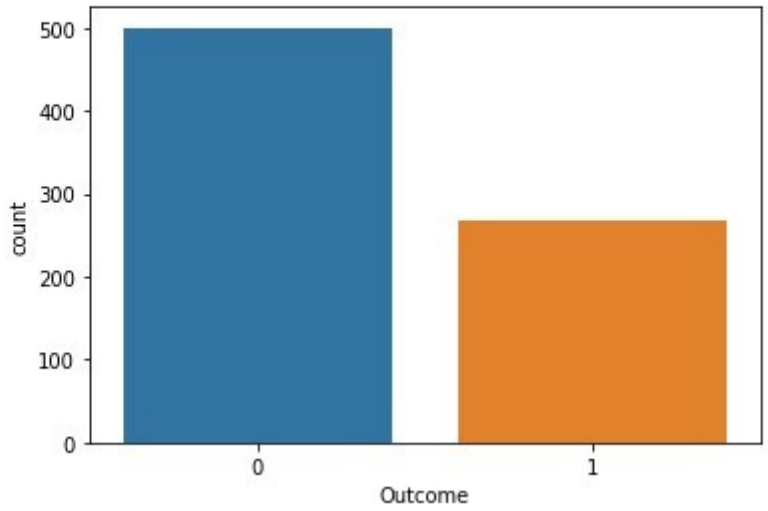


Fig.1: Plotting of Outcome and Count.

Now we plot the correlation Matrix's Heat map to analyze which factor is most correlated to our outcome.

```
In [2]: #correlation matrix
        corr_mat=dataset.corr()
        sns.heatmap(corr_mat, annot=True)
        plt.show()
```

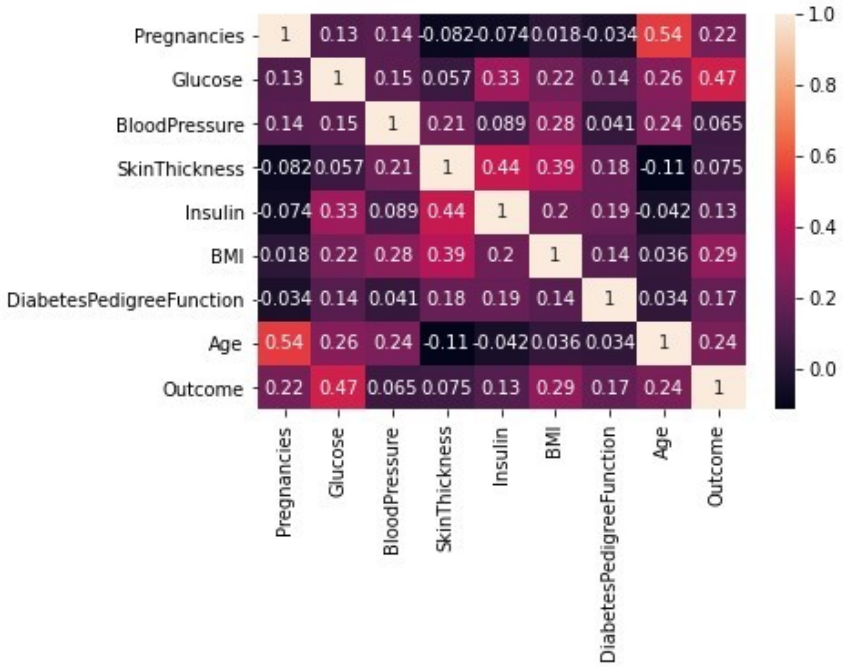


Fig. 2: Correlation Matrix

4. Standardizing the data

In this phase we make the lables as x and y axes.In x axis we take all the values except outcome value which is taken on y axis.

X lable values:

```
x=dataset.iloc[:,:-1].values
y=dataset.iloc[:,-1].values
x
```

```
array([[  6.   , 148.   ,  72.   , ...,  33.6  ,   0.627,  50.   ],
       [  1.   ,  85.   ,  66.   , ...,  26.6  ,   0.351,  31.   ],
       [  8.   , 183.   ,  64.   , ...,  23.3  ,   0.672,  32.   ],
       ...,
       [  5.   , 121.   ,  72.   , ...,  26.2  ,   0.245,  30.   ],
       [  1.   , 126.   ,  60.   , ...,  30.1  ,   0.349,  47.   ],
       [  1.   ,  93.   ,  70.   , ...,  30.4  ,   0.315,  23.   ]])
```

Y lable values:

```
x=dataset.iloc[:,:-1].values
y=dataset.iloc[:,-1].values
y
```

```
array([1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0,
       1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1,
       0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0,
       1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1,
       0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1,
       1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1,
       1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0,
       1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0,
       1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0,
       0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0,
       1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0,
       0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0,
       0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0,
       0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1,
       0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0,
       1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0,
       1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0,
```

Splitting the dataset to test and train

```
In [24]: #splitting the dataset to test and train
         from sklearn.model_selection import train_test_split
         x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
         x_train.shape
```

```
Out[24]: (614, 8)
```

5. Model Building

In this phase, we are going to make a model.
For feature scaling we use standard scaler which predicts the value for new data.

```
: #feature scaling
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x_train=sc.fit_transform(x_train)
x_test=sc.transform(x_test)
```

In this model building phase , we use two classification algorithms

1. Kneighborest Algorithm

KNN works by finding the distances between a query and all the examples in the data, selecting the specified number examples (K) closest to the query, then votes for the most frequent label (in the case of classification)

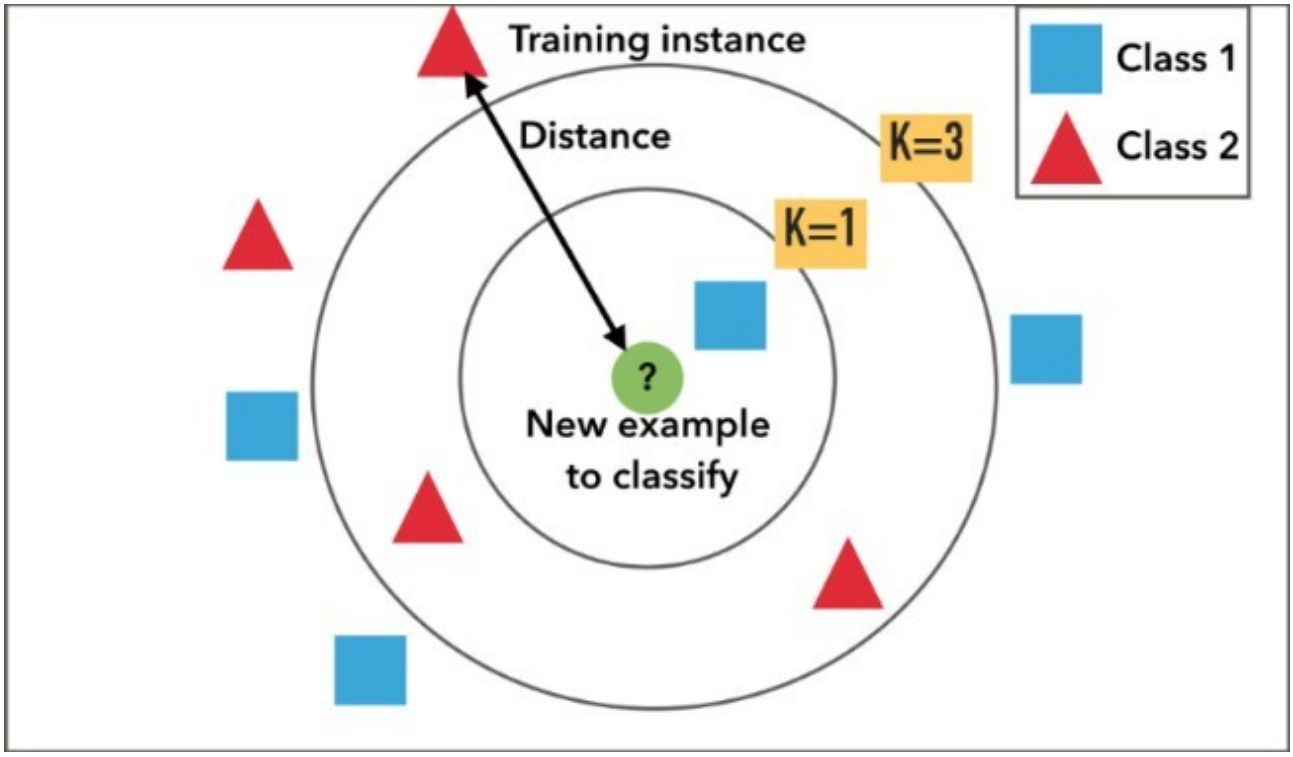


fig.3: KnearestNeighbours

```
#model building
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier(n_neighbors=25,metric="minkowski")
knn.fit(x_train,y_train)
```

```
KNeighborsClassifier(n_neighbors=25)
```

We have to train our data with KNN algorithm with n_neighbors of 25, which was caluculated with least squares.

We Predict the Y values with the Algorithm

```
In [26]: #prediction
         y_pred=knn.predict(x_test)
         y_pred
```

```
Out[26]: array([1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0,
                0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1,
                1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1,
                1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
                0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
                0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

Confusion Matrix

```
In [16]: #confusion matrix
         from sklearn.metrics import confusion_matrix
         cm=confusion_matrix(y_test,y_pred)
         sns.heatmap(cm,annot=True)
```

```
Out[16]: <AxesSubplot:>
```

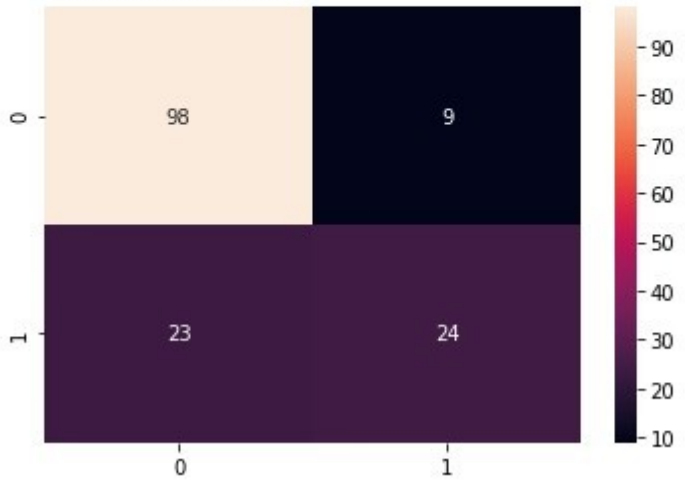


Fig 4: Confusion matrix

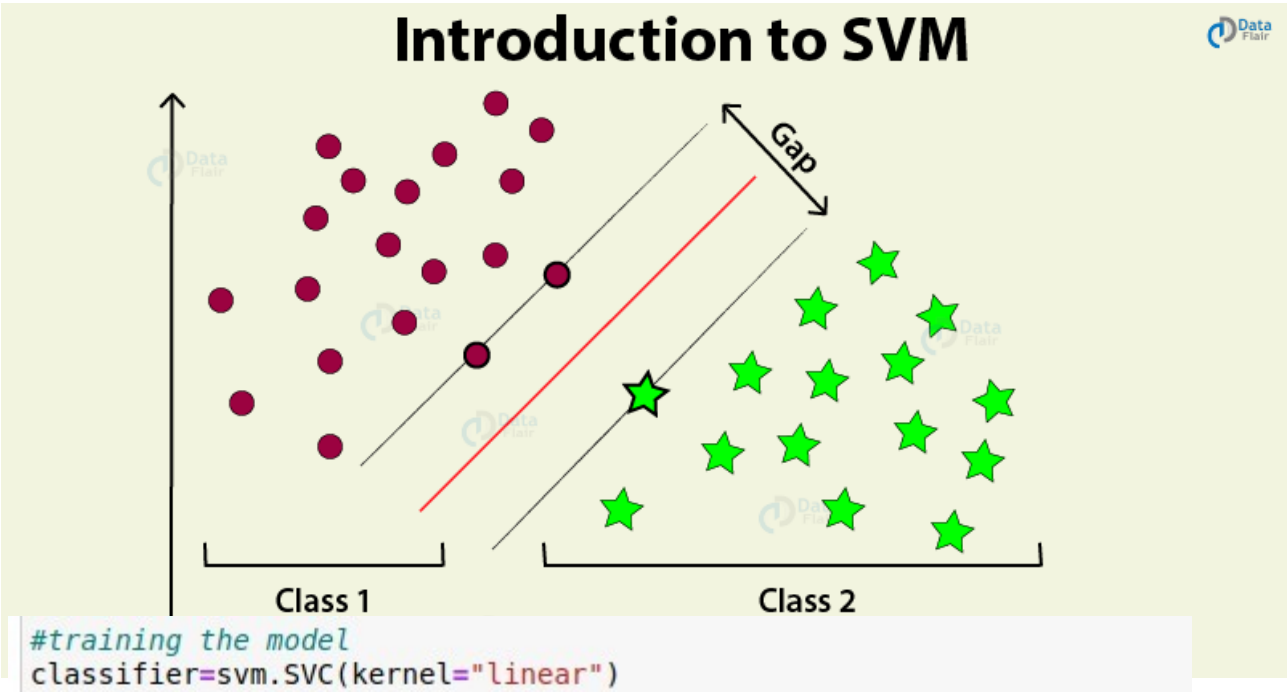
Accuracy of that Algorithm

```
In [17]: #accuracy
         from sklearn.metrics import accuracy_score
         accuracy_score(y_test,y_pred)
```

```
Out[17]: 0.7922077922077922
```

2. Support vector machine algorithm

Support vectors are data points that are closer to the hyperplane and influence the position and orientation of the hyperplane. Using these support vectors, we maximize the margin of the classifier. Deleting the support vectors will change the position of the hyperplane. These are the points that help us build our SVM.




```
#training the model
classifier=svm.SVC(kernel="linear")
```

```
#training the support vector Machine classifier
classifier.fit(X_train,Y_train)
```

```
SVC(kernel='linear')
```

```
#model evaluation
#accuracy score on training data
X_train_prediction=classifier.predict(X_train)
training_data_accuracy=accuracy_score(X_train_prediction,Y_train)
print("accuracy of training data=",training_data_accuracy)
```

```
accuracy of training data= 0.7866449511400652
```

It is the classifier with the algorithm, support vector machine which classifies our data and used to predict with our train data. It gives nearly 0.78 Accuracy score.

6. Making Predective System

In [14]:
```
#making predictive system
input_data=(0,66,9,29,0,26.6,0,31)
input_data_as_numpy_array=np.asarray(input_data)
input_data_reshaped=input_data_as_numpy_array.reshape(1,-1)
#standardized the input data
std_data=scaler.transform(input_data_reshaped)
print(std_data)
prediction=classifier.predict(std_data)
print(prediction)
if(prediction[0]==0):
    print("no diabetics")
else:
    print("yes diabetics")
```

```
[[-1.14185152 -1.71804212 -3.10731749  0.53090156 -0.69289057 -0.68442195
  -1.42512243 -0.19067191]]
[0]
no diabetics
```

Deploying the ML model in Web Appication

After making a predictive system in jupyter notebook.we deploy that system in a web application using Flask. Before that we load the entire ML model into a pickle file which is useful to deploy entire project with a single file, we load both classifier and standardized data to deploy.

]:
```
#saving the classifier model
import pickle
pickle.dump(knn,open("classifier.pkl","wb"))
pickle.dump(sc,open("sc.pkl","wb"))
```

After that we design a simple web page.

Technologies Used

Libraries:

Pandas:

pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series.

Sklearn:

Scikit-learn is a free machine learning library for Python. It features various algorithms like support vector machine, random forests, and k-neighbours, and it also supports Python numerical and scientific libraries like NumPy and SciPy.

Matplotlib

matplotlib.Py plot is a collection of functions that make matplotlib work like MATLAB. Each pyplot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc.

Seaborn

Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics. For a brief introduction to the ideas behind the library, you can read the introductory notes.

Pickle

“Pickling” is the process whereby a Python object hierarchy is converted into a byte stream, and “unpickling” is the inverse operation, whereby a byte stream (from a binary file or bytes-like object) is converted back into an object hierarchy.

Framework

Flask:

Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions.

Languages

Python

Python offers concise and readable code. While complex algorithms and versatile workflows stand behind machine learning and AI, Python's simplicity allows developers to write reliable systems. ... Python code is understandable by humans, which makes it easier to build models for machine learning.

HTML

The HyperText Markup Language, or HTML is the standard markup language for documents designed to be displayed in a web browser.

CSS

Cascading Style Sheets is a style sheet language used for describing the presentation of a document written in a markup language such as HTML.

JAVASCRIPT

JavaScript is high-level, often just-in-time compiled, and multi-paradigm. It has curly-bracket syntax, dynamic typing, prototype-based object-orientation, and first class functions

Future Scope of the Project

A multicenter study with more variables can give different results. More variables can be included using Delphi Method. The present methodology used i.e. Logistic Regression can be compared with more advance tools like ANN (Artificial Neural Network) for the results.

Conclusion

Diabetes is a heterogeneous group of diseases. It's characterized by chronic elevation of glucose in the blood. The main motto of the American diabetes association is "To prevent and cure diabetes and to improve the lives of all people affected by diabetes". To support the lives of the people all over the world, Support vector machine and NB techniques give the accuracy of 77.73% and 73.48% respectively from the existing method and the proposed method improves the accuracy of the classification techniques. Improved SVM accuracy is 77% and NB accuracy is 82.30%, hence it is able to map the features effectively from low dimensions to high dimensions. It gives the best fit to the data with respect to the diabetic and non-diabetic patients. The Disease prevalence percentage is measured highest from the SVM is 45.7%.

References

Global Report on Diabetes 2016 by World Health Organisation.
http://www.who.int/diabetes/publications/grd-2016/en/ ,ISBN 978 92 4 156525 7.

Classification and Diagnosis of Diabetes: Standards of Medical Care in Diabetes—2018 American Diabetes Association Diabetes Care 2018; 41(Supplement 1): S13–S27.
https://doi.org/10.2337/dc18-S002