DIABETES PREDICTION

A PROJECT REPORT

In partial fulfilment of the requirements for the award of the degree

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE & ENGINEERING

Under the guidance of

Mr. MAHENDRA DATTA

BY

SRISHTI RANI

SMRITI RANI

KOEL GOSWAMI

team FUTURE
Education | Healthcare

FUTURE INSTITUTE OF ENGINEERING AND MANAGEMENT

In association with

ARDENT®
COMPUTECH PVT. LTD.

(ISO9001:2015)

A-1/20, Ramgarh, Ganguly Bagan, Kolkata, West Bengal 700047

(Note: All entries of the proforma of approval should be filled up with appropriate and complete information. Incomplete proforma of approval in any respect will be summarily rejected.)

1. Title of the Project: **DIABETES PREDICTION**

2. Project Members: **SRISHTI RANI, SMRITI RANI, KOEL GOSWAMI**

3. Name of the guide: **Mr. MAHENDRA DATTA**

4. 4. Address: Ardent Computech Pvt. Ltd
(An ISO 9001:2015 Certified)
A-1/20, Ramgarh, Ganguly Bagan, Kolkata,
West Bengal 700047

Project Version Control History

Version	Members	Description of Version	Date Completed
Final	SRISHTI RANI SMRITI RANI KOEL GOSWAMI	Project Report	23th January,2020

Signature of Team Member

Date:

Signature of Approver

Date:

For Office Use Only

Approved

MR. MAHENDRA DATTA

Not Approved

Project Proposal Evaluator

DECLARATION

We hereby declare that the project work being presented in the project proposal entitled **"DIABETES PREDICTION"** in partial fulfilment of the requirements for the award of the degree of BACHELOR **OF TECHNOLOGY** at **ARDENT COMPUTECH PVT. LTD, JADAVPUR, KOLKATA, WEST BENGAL,** is an authentic work carried out under the guidance of **MR. MAHENDRA DATTA**. The matter embodied in this project work has not been submitted elsewhere for the award of any degree of our knowledge and belief.

Date:

Name of the Student: Srishti Rani
Smriti Rani
Koel Goswami

Signature of the students:

Ardent Computech Pvt. Ltd (An ISO 9001:2015 Certified)

SDF Building, Module #132, Ground Floor, Salt Lake City, GP Block, Sector V, Kolkata, West Bengal 700091

CERTIFICATE

This is to certify that this proposal of minor project entitled **"DIABETES PREDICTION"** is a record of bonafide work, carried out by ***KOEL GOSWAMI, SMRITI RANI and SRISHTI RANI*** under my guidance at **ARDENT COMPUTECH PVT. LTD**. In my opinion, the report in its present form is in partial fulfilment of the requirements for the award of the degree of **BACHELOR OF TECHNOLOGY** and as per regulations of the **ARDENT®.**

To the best of my knowledge, the results embodied in this report, are original in nature and worthy of incorporation in the present version of the report.

Guide / Supervisor

MR. MAHENDRA DATTA

Project Engineer

Ardent Computech Pvt. Ltd (An ISO 9001:2015 Certified)

A-1/20, Ramgarh, Ganguly Bagan, Kolkata, West Bengal 700047

ACKNOWLEDGEMENT

Success of any project depends largely on the encouragement and guidelines of many others. I take this sincere opportunity to express my gratitude to the people who have been instrumental in the successful completion of this project work.

I would like to show our greatest appreciation to ***Mr. Mahendra Datta***, Project Engineer at Ardent, Kolkata. I always feel motivated and encouraged every time by his valuable advice and constant inspiration; without his encouragement and guidance this project would not have materialized.

Words are inadequate in offering our thanks to the other trainees, project assistants and other members at Ardent Computech Pvt. Ltd. for their encouragement and cooperation in carrying out this project work. The guidance and support received from all the members and who are contributing to this project, was vital for the success of this project.

CONTENTS

OVERVIEW

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 has fewer syntactical constructions than other languages.

Python is interpreted: Python is processed at runtime by the interpreter. You do not need to compileyour 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 interpreterdirectly to write your programs.

Python is Object-Oriented: Python supports Object-Oriented style or technique of programmingthat encapsulates code within objects.

Python is a Beginner's Language: Python is a great language for the beginner-level programmersand supports the development of a wide range of applications from simple text processing to WWW browsers to games.

HISTORY OF PYTHON

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. Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, Small Talk, UNIX shell, and other scripting languages. Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL). Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

FEATURES OF PYTHON

Easy-to-learn: Python has few Keywords, simple structure and clearly defined syntax. This allows a student to pick up the language quickly.

Easy-to-Read: Python code is more clearly defined and visible to the eyes.

Easy -to-Maintain: Python's source code is fairly easy-to-maintain.

A broad standard library: Python's bulk of the library is very portable and cross platform compatible on UNIX, Windows, and Macintosh.

Interactive Mode: Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.

Portable: Python can run on the wide variety of hardware platforms and has the same interface on all platforms.

Extendable: You can add low level modules to the python interpreter. These modules enables programmers to add to or customize their tools to be more efficient.

Databases: Python provides interfaces to all major commercial databases.

GUI Programming: Python supports GUI applications that can be created and ported to many system calls, libraries, and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

Scalable: Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below:

- It support 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 datatypes and supports dynamic type checking.

- It supports automatic garbage collections.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA and JAVA.

ENVIRONMENT SETUP

Open a terminal window and type "python" to find out if it is already installed and which version is installed.

- UNIX (Solaris, Linux, FreeBSD, AIX, HP/UX, SunOS, IRIX, etc.)
- Win 9x/NT/2000
- Macintosh (Intel, PPC, 68K)
- OS/2
- DOS (multiple versions)
- PalmOS
- Nokia mobile phones
- Windows CE
- Acorn/RISC OS

BASIC SYNTAX OF PYTHON PROGRAM

Type the following text at the Python prompt and press the Enter –

>>> print "Hello, Python!"

If you are running new version of Python, then you would need to use print statement with parenthesis as in **print ("Hello, Python!");**.
However in Python version 2.4.3, this produces the following result –

Hello, Python!

Python Identifiers

A Python identifier is a name used to identify a variable, function, class, module or other object. An identifier starts with a letter A to Z or a to z or an underscore (_) followed by zero or more letters, underscores and digits (0 to 9).
Python does not allow punctuation characters such as @, $, and % within identifiers. Python is a case sensitive programming language.

Python Keywords

The following list shows the Python keywords. These are reserved words and you cannot use them as constant or variable or any other identifier names. All the Python keywords contain lowercase letters only.

And, exec, not
Assert, finally, or
Break, for, pass
Class, from, print
continue, global, raise
def, if, return del,
import, try elif, in,
while else, is, with
except, lambda, yield

Lines & Indentation

Python provides no braces to indicate blocks of code for class and function definitions or flow control. Blocks of code are denoted by line indentation, which is rigidly enforced. The number of spaces in the indentation is variable, but all statements within the block must be indented the same amount. For example – if True:
print "True"
else: print
"False"

Command Line Arguments

Many programs can be run to provide you with some basic information about how they should be run. Python enables you to do this with -h −

$ python-h usage: python [option]...[-c cmd|-m mod |
file |-][arg]...

Options and arguments (and corresponding environment variables):

-c cmd: program passed in as string(terminates option list)

-d : debug output from parser (also PYTHONDEBUG=x)

-E : ignore environment variables (such as PYTHONPATH)

-h : print this help message and exit [etc.]

VARIABLE TYPES

Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory.

Assigning Values to Variables

Python variables do not need explicit declaration to reserve memory space. The declaration happens automatically when you assign a value to a variable. The equal sign (=) is used to assign values to variables.

```
counter=10          # An integer assignment
weight=10.60        # A floating point
name="Ardent"       # A string
```

Multiple Assignment

Python allows you to assign a single value to several variables simultaneously. For example − a = b = c = 1 a,b,c = 1,2,"hello"

Standard Data Types

The data stored in memory can be of many types. For example, a person's age is stored as a numeric value and his or her address is stored as alphanumeric characters. Python has five standard data types −

- String
- List

- Tuple
- Dictionary
- Number

Data Type Conversion

Sometimes, you may need to perform conversions between the built-in types. To convert between types, you simply use the type name as a function.

There are several built-in functions to perform conversion from one data type to another.

Sr.No.	Function & Description
1	**int(x [,base])** Converts x to an integer. base specifies the base if x is a string
2	**long(x [,base])** Converts x to a long integer. base specifies the base if x is a string.
3	**float(x)** Converts x to a floating-point number.
4	**complex(real [,imag])** Creates a complex number.
5	**str(x)** Converts object x to a string representation.
6	**repr(x)** Converts object x to an expression string.
7	**eval(str)** Evaluates a string and returns an object.
8	**tuple(s)** Converts s to a tuple.
9	**list(s)** Converts s to a list.

FUNCTIONS

Defining a Function

- deffunctionname(parameters):
  "function_docstring"
  function_suite return
  [expression]

Pass by reference vs Pass by value

All parameters (arguments) in the Python language are passed by reference. It means if you change what a parameter refers to within a function, the change also reflects back in the calling function. For example –

 # Function definition is here

```
defchangeme(mylist):
    "This changes a passed list into this function"
    mylist.append([1,2,3,4]);
    print"Values inside the function: ",mylist
    return

# Now you can call changeme function

mylist=[10,20,30];  changeme(mylist);
print"Values outside the function: ",mylist
```

Here, we are maintaining reference of the passed object and appending values in the same object. So, this would produce the following result −

```
Values inside the function: [10, 20, 30, [1, 2, 3, 4]]
Values outside the function: [10, 20, 30, [1, 2, 3, 4]]
```

Global vs. Local variables

Variables that are defined inside a function body have a local scope, and those defined outside have a global scope . For Example-

```
total=0;                # This is global variable.

# Function definition is here
```

```
def sum( arg1, arg2 ):

# Add both the parameters and return them."

total= arg1 + arg2;# Here total is local
variable.  print"Inside the function local total :
", total  return total;

# Now you can call sum function

sum(10,20);
print"Outside the function global total : ", total
```

When the above code is executed, it produces the following result −

```
Inside the function local total : 30
Outside the function global total : 0
```

MODULES

A module allows you to logically organize your Python code. Grouping related code into a module makes the code easier to understand and use. A module is a Python object with arbitrarily named attributes that you can bind and reference .

The Python code for a module named *aname* normally resides in a file named *aname.py*. Here's an example of a simple module, support.py

```
def print_func( par ):
        print"Hello : ", par  return
```

The *import* Statement

You can use any Python source file as a module by executing an import statement in some other Python source file. The *import* has the following syntax –

```
import module1[, module2[,… moduleN]
```

PACKAGES

A package is a hierarchical file directory structure that defines a single Python application environment that consists of modules and sub packages and sub-subpackages, and so on.

Consider a file *Pots.py* available in *Phone* directory. This file has following line of source code − def Pots():
print "I'm Pots Phone"

Similar way, we have another two files having different functions with the same name as above –

- *Phone/Isdn.py* file having function Isdn()
- *Phone/G3.py* file having function G3()

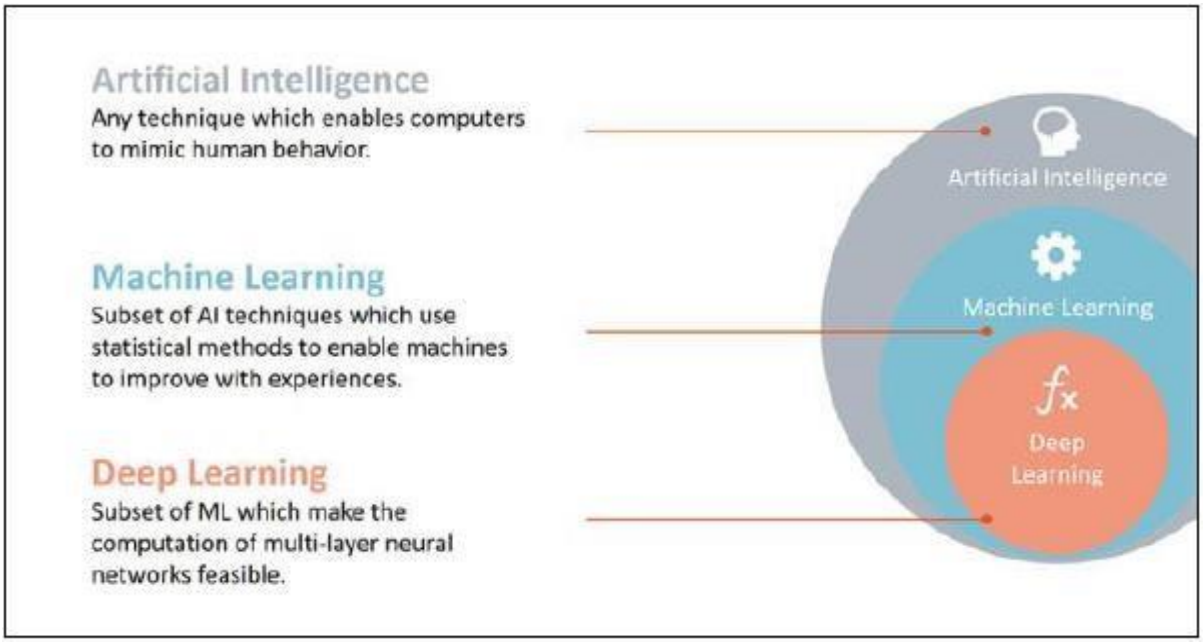
Now, create one more file __init__.py in *Phone* directory −

- Phone/__init__.py

To make all of your functions available when you've imported Phone, you need to put explicit import statements in __init__.py as follows − from Pots import Pots from Isdn import Isdn from G3 import

ARTIFICIAL INTELLIGENCE

Introduction



According to the father of Artificial Intelligence, John McCarthy, it is *"The science and engineering of making intelligent machines, especially intelligent computer programs".*

Artificial Intelligence is a way of **making a computer, a computer-controlled robot, or a software think intelligently**, in the similar manner the intelligent humans think.

AI is accomplished by studying how human brain thinks, and how humans learn, decide, and work while trying to solve a problem, and then using the outcomes of this study as a basis of developing intelligent software and systems.

The development of AI started with the intention of creating similar intelligence in machines that we find and regard high in humans.

Goals of AI

To Create Expert Systems − The systems which exhibit intelligent behaviour, learn, demonstrate, explain, and advice its users.
To Implement Human Intelligence in Machines − Creating systems that understand, think, learn, and behave like humans.

Applications of AI

AI has been dominant in various fields such as :-

Gaming − AI plays crucial role in strategic games such as chess, poker, tic-tac-toe, etc., where machine can think of large number of possible positions based on heuristic knowledge.

Natural Language Processing − It is possible to interact with the computer that understands natural language spoken by humans.

Expert Systems − There are some applications which integrate machine, software, and special information to impart reasoning and advising. They provide explanation and advice to the users.

Vision Systems − These systems understand, interpret, and comprehend visual input on the computer.

For example: A spying aeroplane takes photographs, which are used to figure out spatial information or map of the areas.

Doctors use clinical expert system to diagnose the patient.

Police use computer software that can recognize the face of criminal with the stored portrait made by forensic artist.

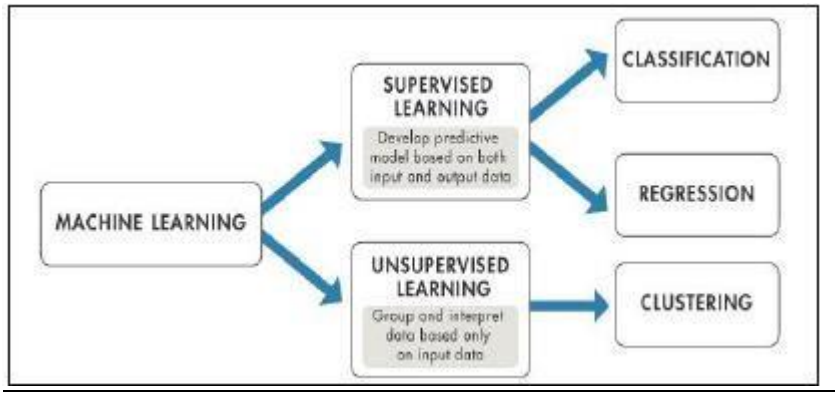
Speech Recognition − Some intelligent systems are capable of hearing and comprehending the language in terms of sentences and their meanings while a human talks to it. It can handle different accents, slang words, noise in the background, change in human's noise due to cold, etc.

Handwriting Recognition − The handwriting recognition software reads the text written on paper by a pen or on screen by a stylus. It can recognize the shapes of the letters and convert it into editable text.

Intelligent Robots − Robots are able to perform the tasks given by a human. They have sensors to detect physical data from the real world such as light, heat, temperature, movement, sound, bump, and pressure. They have efficient processors, multiple sensors and huge memory, to exhibit intelligence. In addition, they are capable of learning from their mistakes and they can adapt to the new environment.

Application of AI

MACHINE LEARNING



Machine learning is a field of computer science that gives computers the ability to learn without being explicitly programmed.
Evolved from the study of pattern recognition and computational learning theory in artificial intelligence, machine learning explores the study and construction of algorithms that can learn from and make predictions on data.

INTRODUCTION TO MACHINE LEARNING

Machine learning is a field of computer science that gives computers the ability to learn without being explicitly programmed.

Arthur Samuel, an American pioneer in the field of computer gaming and artificial intelligence, coined the term "Machine Learning" in 1959 while at IBM. Evolved from the study of pattern recognition and computational learning theory in artificial intelligence, machine learning explores the study and construction of algorithms that can learn from and make predictions on data

Machine learning tasks are typically classified into two broad categories, depending on whether there is a learning "signal" or "feedback" available to a learning system:-

SUPERVISED LEARNING

Supervised learning is the machine learning task of inferring a function from *labeled training data*.[1] The training data consist of a set of *training examples*. In supervised learning, each example is a *pair* consisting of an input object (typically a vector) and a desired output value.

A supervised learning algorithm analyses the training data and produces an inferred function, which can be used for mapping new examples. An optimal scenario will allow for the algorithm to correctly determine the class labels for unseen instances. This requires the learning algorithm to generalize from the training data to unseen situations in a "reasonable" way.

UNSUPERVISED LEARNING

Unsupervised learning is the machine learning task of inferring a function to describe hidden structure from "unlabelled" data (a classification or categorization is not included in the observations). Since the examples given to the learner are unlabelled, there is no evaluation of the accuracy of the structure that is output by the relevant algorithm—which is one way of distinguishing unsupervised learning from supervised learning and reinforcement learning.

A central case of unsupervised learning is the problem of density estimation in statistics, though unsupervised learning encompasses many other problems (and solutions) involving summarizing and explaining key features of the data.

NUMPY

NumPy is a library for the Python programming language, adding support for large, multidimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. The ancestor of NumPy, Numeric, was originally created by Jim Hugunin.

NumPy targets the CPython reference implementation of Python, which is a non-optimizing bytecode interpreter. Mathematical algorithms written for this version of Python often run much slower than compiled equivalents.

Using NumPy in Python gives functionality comparable to MATLAB since they are both interpreted, and they both allow the user to write fast programs as long as most operations work on arrays or matrices instead of scalars.

NUMPY ARRAY

NumPy's main object is the homogeneous multidimensional array. It is a table of elements (usually numbers), all of the same type, indexed by a tuple of positive integers. In NumPy dimensions are called *axes*. The number of axes is *rank.*

For example, the coordinates of a point in 3D space [1, 2, 1] is an array of rank 1, because it has one axis. That axis has a length of 3. In the example pictured below, the array has rank 2 (it is 2dimensional). The first dimension (axis) has a length of 2, the second dimension has a length of 3.

[[1., 0., 0.],
[0., 1., 2.]]

NumPy's array class is called *ndarray*. It is also known by the alias.

SLICING NUMPY ARRAY

```
importnumpy as np

a = np.array([[1,2,3],[3,4,5],[4,5,6]])

print 'Our array is:'   print
a
print '\n'


print 'The items in the second column are:'    print
a[...,1]
print '\n'


print 'The items in the second row are:'   print
a[1,...]
print '\n'


print 'The items column 1 onwards are:'   print
a[...,1:]
```

OUTPUT

```
Our array is:
[[1 2 3]
[3 4 5]
[4 5 6]]

The items in the second column are:
[2 4 5]

The items in the second row are:
[3 4 5]

The items column 1 onwards are:
```

[[2 3]
[4 5]
[5 6]]

SCIPY

modules for optimization, linear algebra, integration, interpolation, special functions, FFT, signal and image processing, ODE solvers and other tasks common in science and engineering.

SciPy builds on the NumPy array object and is part of the NumPy stack which includes tools like Matplotlib, pandas and SymPy, and an expanding set of scientific computing libraries. This NumPy stack has similar users to other applications such as MATLAB, GNU Octave, and Scilab. The NumPy stack is also sometimes referred to as the SciPy stack.

The SciPy Library/Package

The SciPy package of key algorithms and functions core to Python's scientific computing capabilities. Available sub-packages include:

- **constants:** physical constants and conversion factors (since version 0.7.0)
- **cluster:** hierarchical clustering, vector quantization, K-means
- **fftpack:** Discrete Fourier Transform algorithms
- **integrate:** numerical integration routines
- **interpolate:** interpolation tools
- **io:** data input and output
- **lib:** Python wrappers to external libraries
- **linalg:** linear algebra routines
- **misc:** miscellaneous utilities (e.g. image reading/writing)
- **ndimage:** various functions for multi-dimensional image processing
- **optimize:** optimization algorithms including linear programming
- **signal:** signal processing tools
- **sparse:** sparse matrix and related algorithms
- **spatial:** KD-trees, nearest neighbours, distance functions
- **special:** special functions
- **stats:** statistical functions
- **weave:** tool for writing C/C++ code as Python multiline strings

Data Structures

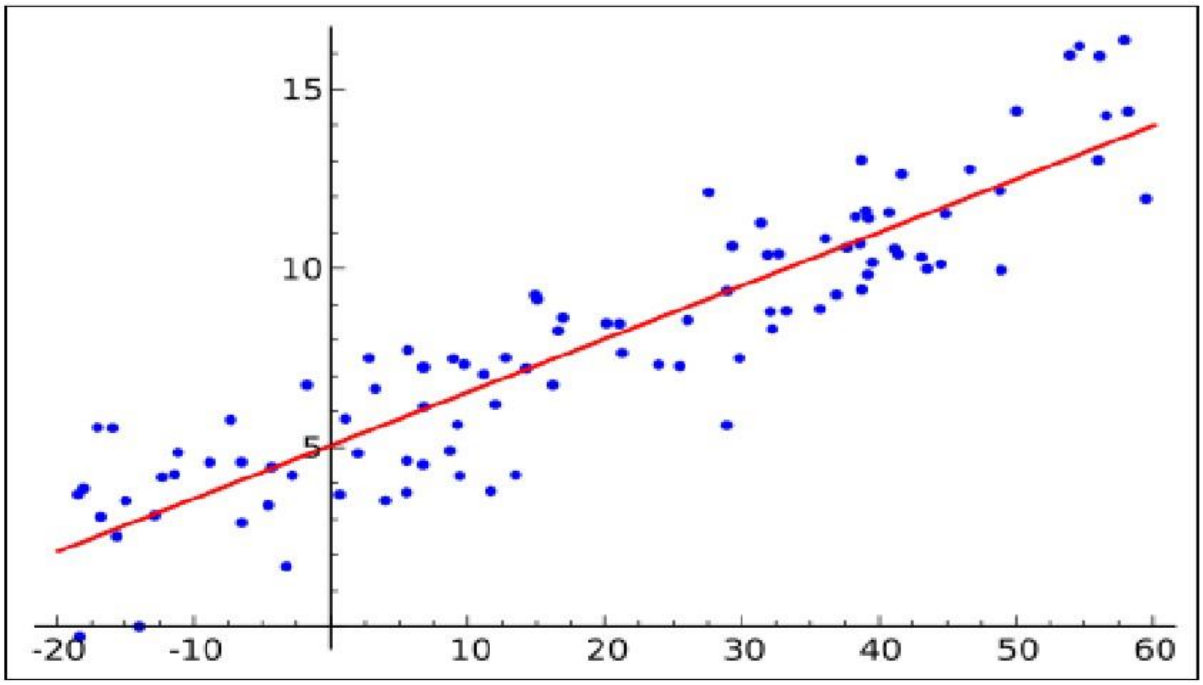
The basic data structure used by SciPy is a multidimensional array provided by the NumPy module. NumPy provides some functions for linear algebra, Fourier transforms and random number generation, but not with the generality of the equivalent functions in SciPy. NumPy can also be used as an efficient multi-dimensional container of data with arbitrary data-types. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases. Older versions of SciPy used Numeric as an array type, which is now deprecated in favour of the newer NumPy array code.

SCIKIT-LEARN

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.

The scikit-learn project started as scikits.learn, a Google Summer of Code project by David Cournapeau. Its name stems from the notion that it is a "SciKit" (SciPy Toolkit), a separatelydeveloped and distributed third-party extension to SciPy.[4] The original codebase was later rewritten by other developers. In 2010 Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort and Vincent Michel, all from INRIA took leadership of the project and made the first public release on February the 1st 2010[5]. Of the various scikits, scikit-learn as well as scikit-image were described as "wellmaintained and popular" in November 2012.

REGRESSION ANALYSIS



In statistical modelling, **regression analysis** is a set of statistical processes for estimating the relationships among variables. It includes many techniques for modelling and analysing several variables, when the focus is on the relationship between a dependent variable and one or more independent variables (or 'predictors'). More specifically, regression analysis helps one understand how the typical value of the dependent variable (or 'criterion variable') changes when any one of the independent variables is varied, while the other independent variables are held fixed.

Regression analysis is widely used for prediction and forecasting, where its use has substantial overlap with the field of machine learning. Regression analysis is also used to understand which among the independent variables are related to the dependent variable, and to explore the forms of these relationships. In restricted circumstances, regression analysis can be used to infer casual relationships between the independent and dependent variables. However this can lead to illusions or false relationships, so caution is advisable

LINEAR REGRESSION

Linear regression is a linear approach for modelling the relationship between a scalar dependent variable y and one or more explanatory variables (or independent variables) denoted X. The case of one explanatory variable is called *simple linear regression*. For more than one explanatory variable, the process is called *multiple linear regression*.

In linear regression, the relationships are modelled using linear predictor functions whose unknown model parameters are estimated from the data. Such models are called *linear models*.

LOGISTIC REGRESSION

Logistic regression, or logit regression, or logit model [1] is a regression model where the dependent variable (DV) is categorical. This article covers the case of a binary dependent variable—that is, where the output can take only two values, "0" and "1", which represent outcomes such as pass/fail, win/lose, alive/dead or healthy/sick. Cases where the dependent variable has more than two outcome categories may be analysed in multinomial logistic regression, or, if the multiple categories are ordered, in ordinal logistic regression. In the terminology of economics, logistic regression is an example of a qualitative response/discrete choice model.

LOGISTIC REGRESSION

Logistic regression is a form of regression analysis in which the relationship between the independent variable x and the dependent variable y is modelled as an n^{th} degree polynomial in x.

Logistic regression fits a nonlinear relationship between the value of x and the corresponding conditional mean of y , denoted E(y | x), and has been used to describe nonlinear phenomena such as the growth rate of tissues, the distribution of carbon isotopes in lake sediments, and the progression of disease epidemics.

Although Logistic regression fits a nonlinear model to the data, as a statistical estimation problem it is linear, in the sense that the regression function E(y | x) is linear in the unknown parameters that are estimated from the data.

DECISION TREE

Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. A Decision tree is a flowchart like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label.

RANDOM FOREST

Random forest is a Supervised Learning algorithm which uses ensemble learning method for classification and regression. It operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.

A random forest is a meta-estimator (i.e. it combines the result of multiple predictions) which aggregates many decision trees, with some helpful modifications:

1. The number of features that can be split on at each node is limited to some percentage of the total (which is known as the hyperparameter). This ensures that the ensemble model does not rely too heavily on any individual feature, and makes fair use of all potentially predictive features.

2. Each tree draws a random sample from the original data set when generating its splits, adding a further element of randomness that prevents overfitting.

The above modifications help prevent the trees from being too highly correlated.

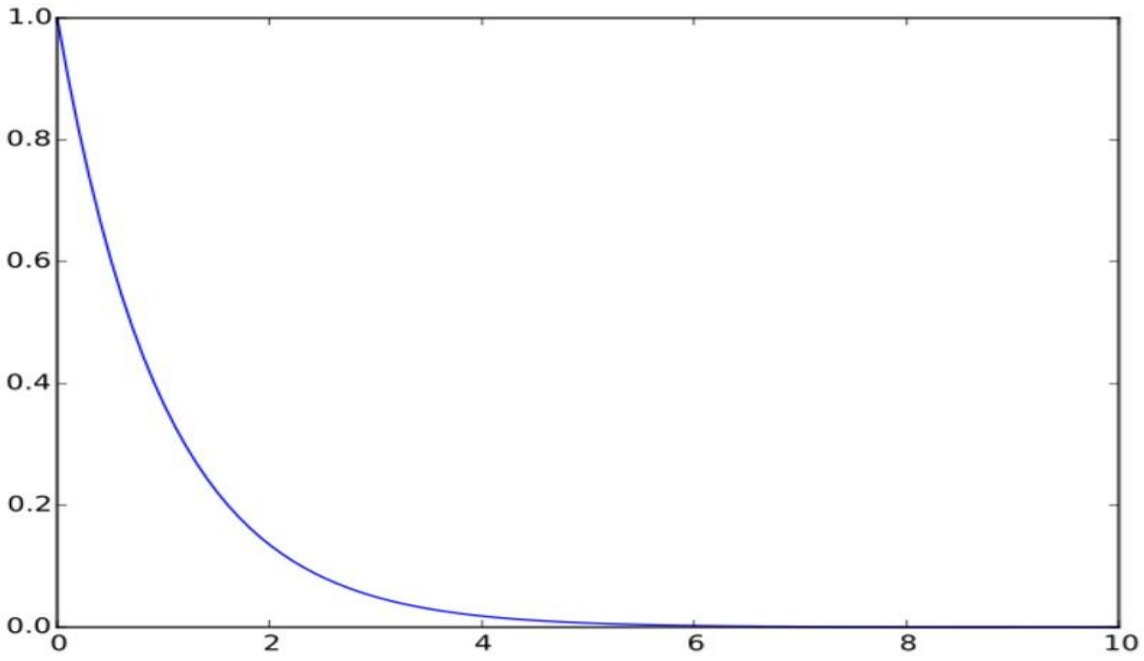
MATPLOTLIB

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter,wxPython, Qt, or GTK+. There is also a procedural "pylab" interface based on a state machine (like OpenGL), designed to closely resemble that of MATLAB, though its use is discouraged .SciPy makes use of matplotlib.

EXAMPLE

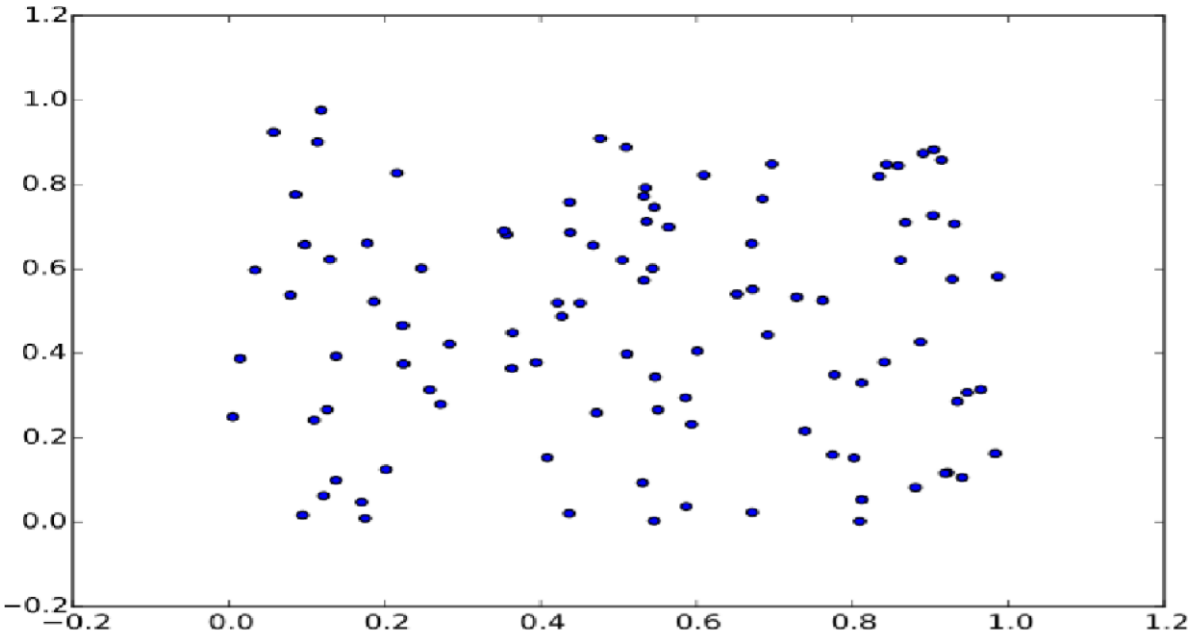
- **LINE PLOT**

```
>>>import matplotlib.pyplotasplt
>>>import numpyasnp
>>> a =np.linspace(0,10,100)
>>> b =np.exp(-a)
>>>plt.plot(a,b)
>>>plt.show()
```



- **SCATTER PLOT**

```
>>>importmatplotlib.pyplotasplt
>>>fromnumpy.randomimport rand
>>> a =rand(100)
>>> b =rand(100)
>>>plt.scatter(a,b)
>>>plt.show()
```



PANDAS

In computer programming, **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. It is free software released under the three-clause BSD license. "Panel data", an econometrics term for multidimensional, structured data sets.

LIBRARY FEATURES

- Data Frame object for data manipulation with integrated indexing.
- Tools for reading and writing data between in-memory data structures and different file formats.
- Data alignment and integrated handling of missing data.
- Reshaping and pivoting of data sets.
- Label-based slicing, fancy indexing, and sub setting of large data sets.
- Data structure column insertion and deletion.
- Group by engine allowing split-apply-combine operations on data sets.
- Data set merging and joining.
- Hierarchical axis indexing to work with high-dimensional data in a lower-dimensional data structure.
- Time series-functionality: Date range generation.

CLUSTERING

Cluster analysis or **clustering** is the task of grouping a set of objects in such a way that objects in the same group (called a **cluster**) are more similar (in some sense or another) to each other than to those in other groups (clusters). It is a main task of exploratory data mining, and a common technique for statistical data analysis, used in many fields, including machine learning, pattern recognition, image analysis, information retrieval, bioinformatics, data compression, and computer graphics.

Cluster analysis itself is not one specific algorithm, but the general task to be solved. It can be achieved by various algorithms that differ significantly in their notion of what constitutes a cluster and how to efficiently find them. Popular notions of clusters include groups with small distances among the cluster members, dense areas of the data space, intervals or particular statistical distributions. Clustering can therefore be formulated as a multi-objective optimization problem.

The appropriate clustering algorithm and parameter settings (including values such as the distance function to use, a density threshold or the number of expected clusters) depend on the individual data set and intended use of the results. Cluster analysis as such is not an automatic task, but an iterative process of knowledge discovery or interactive multi-objective optimization that involves trial and failure. It is often necessary to modify data pre-processing and model parameters until the result achieves the desired properties.

ALGORITHM

- Data Collection
- Data Pre-processing
- Model Selection
- Training
- Testing

Data Collection: We have collected data sets of sales from online website. We have downloaded the .csv files in which information was present.

DataPre-processing: The collected data is formatted into suitable data sets. Label Encoder is used for certain columns to convert the values into numeric form.

Model Selection: We have selected different models to minimize the error of the predicted value. The different models used are Linear Regression Linear Model, Ridge Linear model, Decision Tree Regression Model and Random Forest Regression Model .

Training: The data sets was divided such that x_train is used to train the model with corresponding x_test values and some y_train kept reserved for testing.

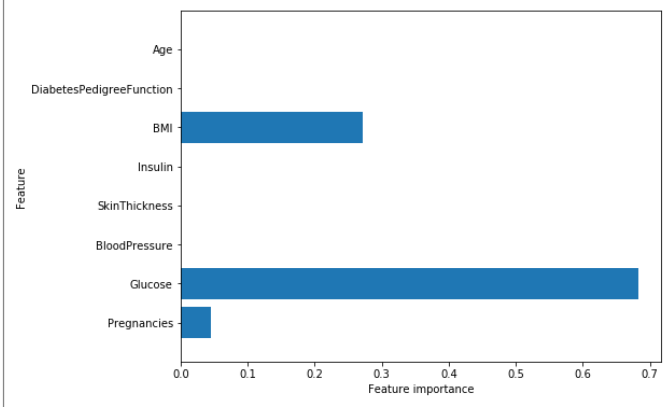
Testing: The model was tested with y_train and stored in y_predict . Both y_train and y_predict was compared.

ACTUAL CODES FOR DIABETES PREDICTION

- ***USING DECISION TREE REGRESSOR***:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
diabetes = pd.read_csv('dia.csv')
diabetesfeatures=diabetes.columns
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(diabetes.loc[:, diabetes.columns != 'Outcome'],
                                                    diabetes['Outcome'], stratify=diabetes['Outcome'], random_state=66)
from sklearn.tree import DecisionTreeClassifier
tree = DecisionTreeClassifier(random_state=0)
tree.fit(X_train, y_train)
print("Accuracy on training set: {:.3f}".format(tree.score(X_train, y_train)))
print("Accuracy on test set: {:.3f}".format(tree.score(X_test, y_test)))
tree = DecisionTreeClassifier(max_depth=3, random_state=0)
tree.fit(X_train, y_train)
print("Accuracy on training set: {:.3f}".format(tree.score(X_train, y_train)))
print("Accuracy on test set: {:.3f}".format(tree.score(X_test, y_test)))
print("Feature importances:\n{}".format(tree.feature_importances_))
def plot_feature_importances_diabetes(model):
    plt.figure(figsize=(8,6))
    n_features = 8
    plt.barh(range(n_features), model.feature_importances_, align='center')
    plt.yticks(np.arange(n_features), diabetesfeatures)
    plt.xlabel("Feature importance")
    plt.ylabel("Feature")
    plt.ylim(-1, n_features)
plot_feature_importances_diabetes(tree)
plt.savefig('feature_importance')
```

```
In [1]: runfile('C:/Users/koel/Desktop/decision.py', wdir='C:/Users/koel/Desktop')
Accuracy on training set: 1.000
Accuracy on test set: 0.714
Accuracy on training set: 0.773
Accuracy on test set: 0.740
Feature importances:
[0.04554275 0.6830362  0.         0.         0.         0.27142106
 0.         0.        ]
```

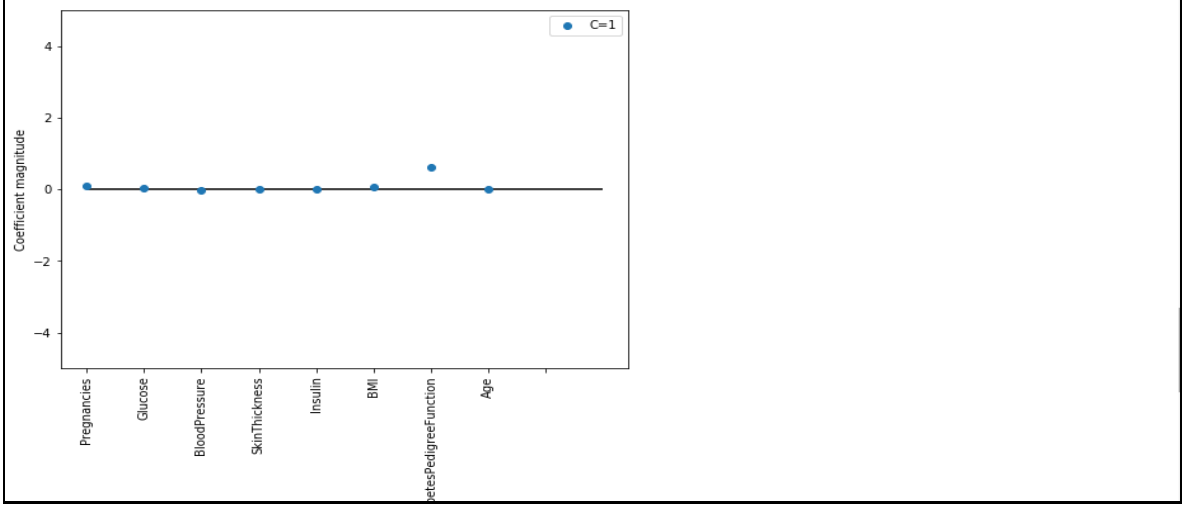


- ***USING LOGICAL REGRESSION***

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
diabetes = pd.read_csv('dia.csv')

print(diabetes.columns)
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(diabetes.loc[:, diabetes.columns != 'Outcome'],
                                                    diabetes['Outcome'], stratify=diabetes['Outcome'], random_state=66)
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression().fit(X_train, y_train)
print("Training set score: {:.3f}".format(logreg.score(X_train, y_train)))
print("Test set score: {:.3f}".format(logreg.score(X_test, y_test)))
diabetes_features = [x for i,x in enumerate(diabetes.columns) if i!=8]
plt.figure(figsize=(8,6))
plt.plot(logreg.coef_.T, 'o', label="C=1")
#plt.plot(logreg100.coef_.T, '^', label="C=100")
##plt.plot(logreg001.coef_.T, 'v', label="C=0.001")
plt.xticks(range(diabetes.shape[1]), diabetes_features, rotation=90)
plt.hlines(0, 0, diabetes.shape[1])
plt.ylim(-5, 5)
plt.xlabel("Feature")
plt.ylabel("Coefficient magnitude")
plt.legend()
plt.savefig('log_coef')
```

```
Training set score: 0.781
Test set score: 0.771
C:\Users\koel\AppData\Local\Continuum\anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:432: FutureWarning: Default
solver will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
```

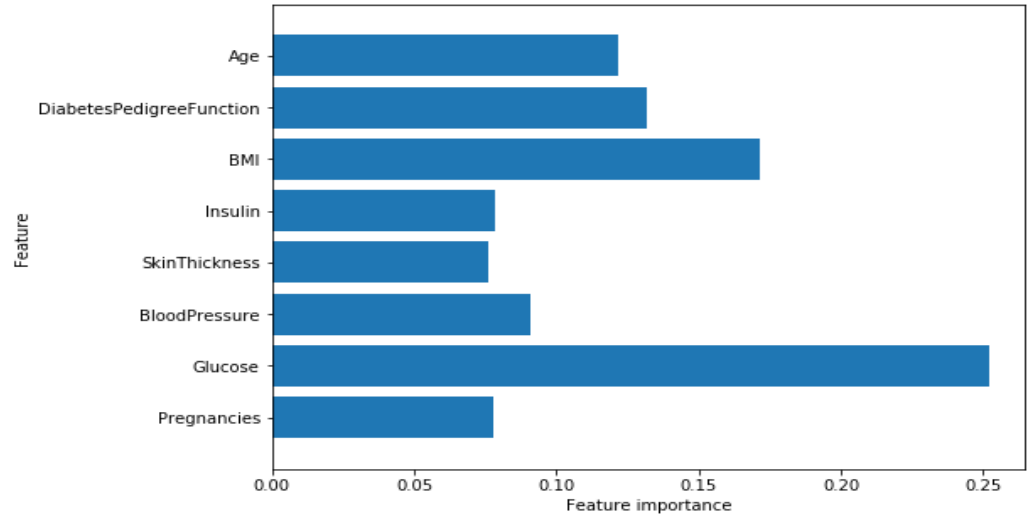


- ***USING RANDOM FOREST CLASSIFICATION***

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
diabetes = pd.read_csv('dia.csv')
diabetesfeatures=diabetes.columns

print(diabetes.columns)
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(diabetes.loc[:, diabetes.columns != 'Outcome'],
                                                    diabetes['Outcome'], stratify=diabetes['Outcome'], random_state=66)
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(n_estimators=100, random_state=0)
rf.fit(X_train, y_train)
print("Accuracy on training set: {:.3f}".format(rf.score(X_train, y_train)))
print("Accuracy on test set: {:.3f}".format(rf.score(X_test, y_test)))
rf1 = RandomForestClassifier(max_depth=3, n_estimators=100, random_state=0)
rf1.fit(X_train, y_train)
print("Accuracy on training set: {:.3f}".format(rf1.score(X_train, y_train)))
print("Accuracy on test set: {:.3f}".format(rf1.score(X_test, y_test)))
def plot_feature_importances_diabetes(model):
    plt.figure(figsize=(8,6))
    n_features = 8
    plt.barh(range(n_features), model.feature_importances_, align='center')
    plt.yticks(np.arange(n_features), diabetesfeatures)
    plt.xlabel("Feature importance")
    plt.ylabel("Feature")
    plt.ylim(-1, n_features)
plot_feature_importances_diabetes(rf)
plt.savefig('feature_importance')
```

```
In [3]: runfile('C:/Users/koel/Desktop/random.py', wdir='C:/Users/koel/Desktop')
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
Accuracy on training set: 1.000
Accuracy on test set: 0.786
Accuracy on training set: 0.800
Accuracy on test set: 0.755
```




```
In [4]:
```

CONCLUSION

We have collected the raw data from online sources. Then we take this raw data and format it.

Now we have selected few models for error detection. We have used four models namely logistic regression, decision tree regression model and random forest regression model.

FUTURE SCOPE

The data taken was limited. The project could be extended to more number of days. Thus limited number of algorithms could only be used.

The error can be minimized as well using other algorithms.

THANK YOU