DATA SCIENCE

(Prediction of Diabetes)

Summer Internship Report Submitted in partial fulfillment

of the requirement for undergraduate degree of

Bachelor of Technology

In

" Computer Science Engineering "

By K SATYA SAI RAMANI

221710309021

Department Of Computer Science Engineering

GITAM (Deemed to be University)

Hyderabad-502329

DECLARATION

I submit this industrial training work entitled **"PREDICTION OF DIABETES"** to GITAM (Deemed To Be University), Hyderabad in partial fulfillment of the requirements for the award of the degree **of "Bachelor of Technology"** in **"Computer Science Engineering"** .I declare that it was carried out independently by me under the guidance.

The results embodied in this report have not been submitted to any other University or Institute for the award of any degree or diploma.

Place: HYDERABAD

K SATYA SAI RAMANI
221710309021

CERTIFICATE

-

CERTIFICATE

-

ACKNOWLEDGEMENT

Apart from my effort, the success of this internship largely depends on the encouragement and guidance of many others. I take this opportunity to express my gratitude to the people who have helped me in the successful competition of this internship.

I would like to thank respected Dr. N. Siva Prasad, Pro Vice Chancellor, GITAM Hyderabad and Prof. N. Seetha Ramaiah, Principal, GITAM Hyderabad

I would like to thank respected Prof. S. Phani Kumar, Head of the Department of Computer Science Engineering for giving me such a wonderful opportunity to expand my knowledge for my own branch and giving me guidelines to present a internship report. It helped me a lot to realize of what we study for.

I would like to thank the respected faculties who helped me to make this internship a successful accomplishment.

I would also like to thank my friends who helped me to make my work more organized and well-stacked till the end.

K SATYA SAI RAMANI

221710309021

ABSTRACT

Data Science and analysis is playing the most significant role today covering every industry in the market.For e.g finance,e-commerce,business,education,government. Now organizations play a 360 degree role to analyzed the behavior and interest of their customers to take decisions in favour of them.Data is analyzed through programming language such as python which is one of the most versatile language and helps in doing a lot of things through it. Netflix is a pure data science project that reached at the top through analyzing every single interest of their customers. Keyword:Data Visualization, AnacondaJupyter Notebook,Exploratory Data Anlaysis,Machine Learning,Deep learning.

INDEX

Chapter 1:

1.1 What is Data Science----------→10

1.2 Need of Data Science----------→11

1.3 Uses of Data Science----------→13

Chapter 2:

2.1 Introduction to Python----------→14

2.2 History of python----------→15

2.3 Features of python----------→15

2.4 How to setup python----------→15

2.4.1 Installation using python IDLE----------→15

2.4.2 Installation using anaconda----------→16

2.5 Python variable types----------→16

2.5.1 Python Numbers----------→16

2.5.2 Python Strings----------→17

2.5.3 Python Lists----------→17

2.5.4 Python Tuples----------→17

2.5.5 Python Dictionary----------→17

2.6 Python Function----------→18

2.6.1 Defining a function----------→18

2.6.2 Calling a function----------→18

2.7 Python using Oop's concepts----------→18

2.7.1 Class----------→18

2.7.2 __init__ method in class----------→19

Chapter 3:

3.1 Problem Statement----------→19

3.2 Data Set----------→19

3.3 Objective of the case study----------→19

Chapter 4:

4.1 Preprocessing of the data----------→20

4.1.1 Getting the data set----------→20

4.1.2 Importing the libraries----------→20

4.1.3 Importing / reading the data set----------→20

4.1.4 Checking the dataset dimensions----------→21

4.1.5 To check the features of the dataset----------→21

4.1.6 Statistical summary of the data----------→22

4.1.7 Checking the null values----------→22

4.2 Data Visualisation----------→**23**

4.2.1 Countplot of the outcome parameter----------→23

4.2.2 Histogram of each parameter----------→23

4.2.3 Scatterplot matrix----------→24

4.2.4 Pair plot visualisation----------→26

4.2.5 Heatmap----------→27

4.3 Data Processing ----------→**27**

4.3.1 Replacing zeros with NaN values----------→27

4.3.2 Count of the NaN values----------→28

4.3.3 Replacing NaN with mean values----------→28

4.3.4 Feature Scaling----------→29

4.3.5 Spliting the data into training and testing----------→29

4.4 Data Modelling----------→**30**

4.4.1 Using Logistic Regression algorithm----------→30

4.4.2 Using K-nearest Neighbours algorithm----------→30

4.4.3 Using Support Vector Classifier algorithm----------→30

4.4.4 Using Naive Baye's algorithm----------→31

4.4.5 Using Decision Tree algorithm----------→31

4.4.6 Using Random Forest algorithm----------→32
4.4.7 Making prediction on the test data----------→32
4.5 Model Evaluation----------**→32**
4.6 Accuracy on the test data----------**→33**
4.7 Confusion Matrix----------**→33**
4.7.1 Confusion matrix----------→33
4.7.2 Heatmap of the Confusion matrix----------→33
4.8 Final Classification Report----------**→34**

CONCLUSION----------**→34**

REFERENCES----------**→35**

CHAPTER-1

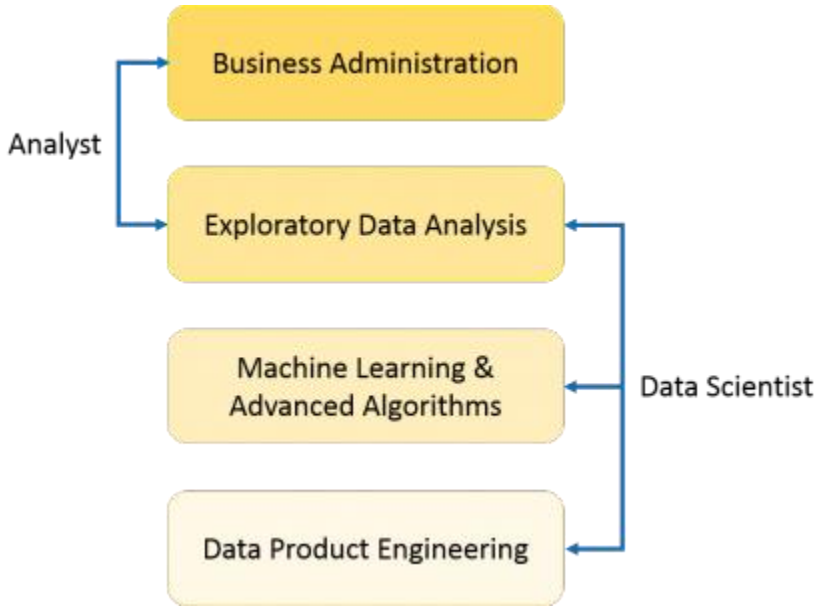
DATA SCIENCE

1.1 What is Data Science

Use of the term Data Science is increasingly common, but what does it exactly mean? What skills do you need to become Data Scientist? What is the difference between BI and Data Science? How are decisions and predictions made in Data Science? These are some of the questions that will be answered further.

First, let's see what is Data Science. Data Science is a blend of various tools, algorithms, and machine learning principles with the goal to discover hidden patterns from the raw data. How is this different from what statisticians have been doing for years?

The answer lies in the difference between explaining and predicting.



As you can see from the above image, a Data Analyst usually explains what is going on by processing history of the data. On the other hand, Data Scientist not only does the exploratory analysis to discover insights from it, but also uses various advanced machine learning algorithms to identify the occurrence of a particular event in the future. A Data Scientist will look at the data from many angles, sometimes angles not known earlier.

So, Data Science is primarily used to make decisions and predictions making use of predictive causal analytics, prescriptive analytics (predictive plus decision science) and machine learning.

- **Predictive causal analytics –** If you want a model which can predict the possibilities of a particular event in the future, you need to apply predictive causal analytics. Say, if you are providing money on credit, then the probability of customers making future credit payments on time is a matter of concern for you. Here, you can build a model which can perform predictive analytics on the payment history of the customer to predict if the future payments will be on time or not.

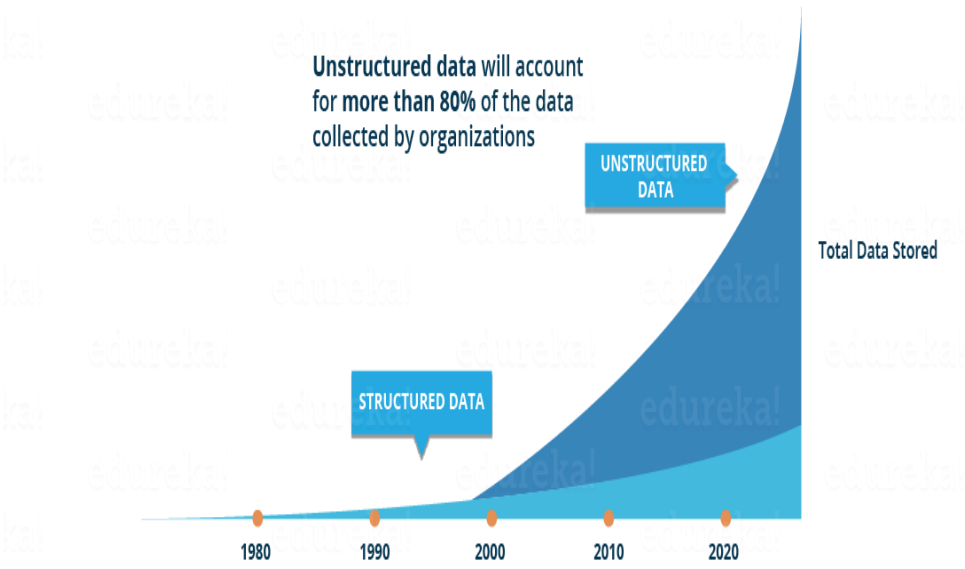
- **Prescriptive analytics:** If you want a model which has the intelligence of taking its own decisions and the ability to modify it with dynamic parameters, you certainly need prescriptive analytics for it. This relatively new field is all about providing advice. In other terms, it not only predicts but suggests a range of prescribed actions and associated outcomes.
  The best example for this is Google's self-driving car which I had discussed earlier too. The data gathered by vehicles can be used to train self-driving cars. You can run algorithms on this data to bring intelligence to it. This will enable your car to take decisions like when to turn, which path to take, when to slow down or speed up.

- **Machine learning for making predictions** — If you have transactional data of a finance company and need to build a model to determine the future trend, then machine learning algorithms are the best bet. This falls under the paradigm of supervised learning. It is called supervised because you already have the data based on which you can train your machines. For example, a fraud detection model can be trained using a historical record of fraudulent purchases.

- **Machine learning for pattern discovery** — If you don't have the parameters based on which you can make predictions, then you need to find out the hidden patterns within the dataset to be able to make meaningful predictions. This is nothing but the unsupervised model as you don't have any predefined labels for grouping. The most common algorithm used for pattern discovery is Clustering.
  Let's say you are working in a telephone company and you need to establish a network by putting towers in a region. Then, you can use the clustering technique to find those tower locations which will ensure that all the users receive optimum signal strength.

1.2 Need of Data Science

- Traditionally, the data that we had was mostly structured and small in size, which could be analyzed by using the simple BI tools. Unlike data in the traditional systems which was mostly structured, today most of the data is unstructured or semi-structured. Let's have a look at the data trends in the image given below which shows that by 2020, more than 80 % of the data will be unstructured.



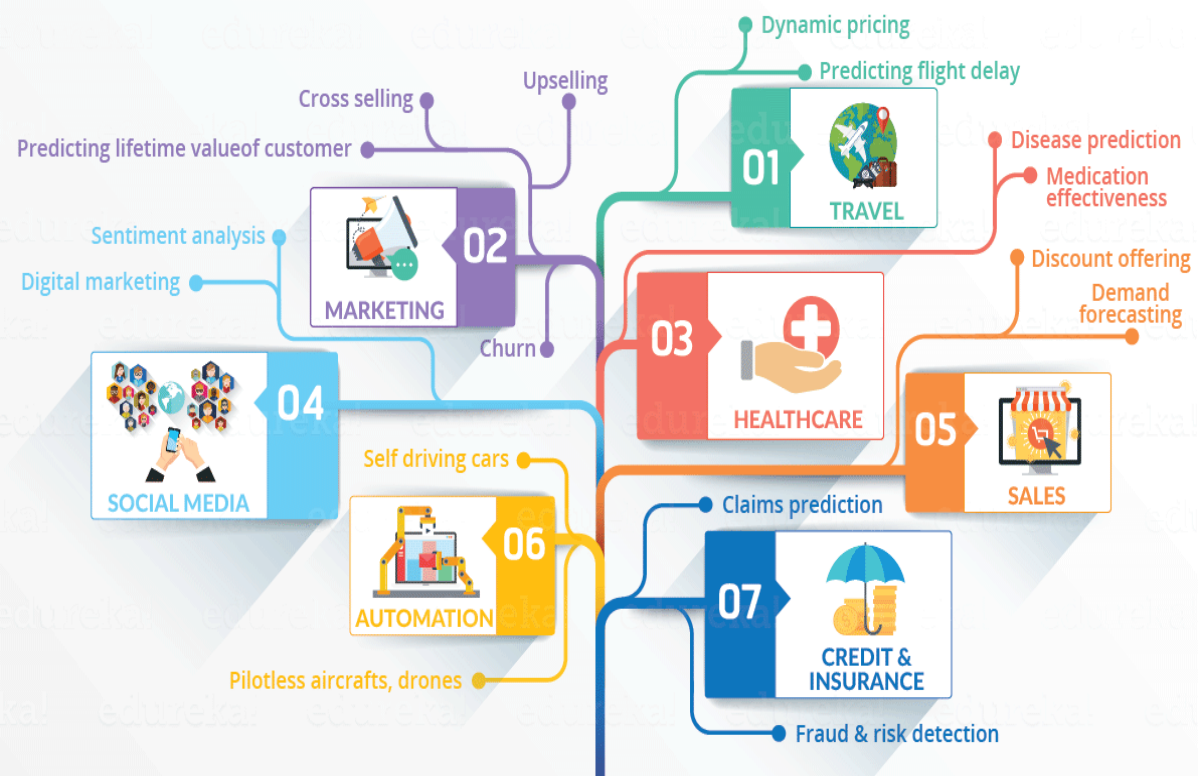
This data is generated from different sources like financial logs, text files, multimedia forms, sensors, and instruments. Simple BI tools are not capable of processing this huge volume and variety of data. This is why we need more complex and advanced analytical tools and algorithms for processing, analyzing and drawing meaningful insights out of it.

This is not the only reason why Data Science has become so popular. Let's dig deeper and see how Data Science is being used in various domains.

- How about if you could understand the precise requirements of your customers from the existing data like the customer's past browsing history, purchase history, age and income. No doubt you had all this data earlier too, but now with the vast amount and variety of data, you can train models more effectively and recommend the product to your customers with more precision. Wouldn't it be amazing as it will bring more business to your organization?

- Let's take a different scenario to understand the role of Data Science in decision making. How about if your car had the intelligence to drive you home? The self-driving cars collect live data from sensors, including radars, cameras and lasers to create a map of its surroundings. Based on this data, it takes decisions like when to speed up, when to speed down, when to overtake, where to take a turn – making use of advanced machine learning algorithms.
- Let's see how Data Science can be used in predictive analytics. Let's take weather forecasting as an example. Data from ships, aircrafts, radars, satellites can be collected and analyzed to build models. These models will not only forecast the weather but also help in predicting the occurrence of any natural calamities. It will help you to take appropriate measures beforehand and save many precious lives.

Let's have a look at the below infographic to see all the domains where Data Science is creating its impression.



1.3 Uses of Data Science

Internet Search

When we speak of search, we think 'Google'. Right? But there are many other search engines like Yahoo, Bing, Ask, AOL, Duckduckgo etc. All these search engines (including Google) make use of data science algorithms to deliver the best result for our searched query in fraction of seconds. Considering the fact that, Google processes more than 20 petabytes of data everyday. Had there been no data science, Google wouldn't have been the 'Google' we know today.

Digital Advertisements (Targeted Advertising and re-targeting)

If you thought Search would have been the biggest application of data science and machine learning, here is a challenger – the entire digital marketing spectrum. Starting from the display banners on various websites to the digital bill boards at the airports – almost all of them are decided by using data science algorithms.

This is the reason why digital ads have been able to get a lot higher CTR than traditional advertisements. They can be targeted based on user's past behaviour. This is the reason why I see ads of analytics trainings while my friend sees ad of apparels in the same place at the same time.

Recommender Systems

Who can forget the suggestions about similar products on Amazon? They not only help you find relevant products from billions of products available with them, but also adds a lot to the user experience.

A lot of companies have fervidly used this engine / system to promote their products / suggestions in accordance with user's interest and relevance of information. Internet giants like Amazon, Twitter, Google Play, Netflix, Linkedin, imdb and many more uses this system to improve user experience. The recommendations are made based on previous search results for a user.

Image Recognition

You upload your image with friends on Facebook and you start getting suggestions to tag your friends. This automatic tag suggestion feature uses face recognition algorithm. Similarly, while using whatsapp web, you scan a barcode in your web browser using your mobile phone. In addition, Google provides you the option to search for images by uploading them. It uses image recognition and provides related search results.

Speech Recognition

Some of the best example of speech recognition products are Google Voice, Siri, Cortana etc. Using speech recognition feature, even if you aren't in a position to type a message, your life wouldn't stop. Simply speak out the message and it will be converted to text. However, at times, you would realize, speech recognition doesn't perform accurately.

CHAPTER 2

PYTHON

Basic programming language used for machine learning is : PYTHON

2.1 INTRODUCTION TO PYHTON:

- Python is a high-level, interpreted, interactive and object-oriented scripting language.
- Python is a general purpose programming language that is often applied in scripting roles
- Python is Interpreted: Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is like PERL and PHP.
- Python is Interactive: You can sit at a Python prompt and interact with the interpreter directly to write your programs.

● Python is Object-Oriented: Python supports the Object-Oriented style or technique of programming that encapsulates code within objects.

2.2 HISTORY OF PYTHON:

● Python was developed by GUIDO VAN ROSSUM in early 1990's
● Its latest version is 3.7 , it is generally called as python3

2.3 FEATURES OF PYTHON:

● Easy-to-learn: Python has few keywords, simple structure, and a clearly defined syntax, This allows the 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-maintaining.
● A broad standard library: Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
● Portable: Python can run on a 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 enable 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.

2.4 HOW TO SETUP PYTHON:

● Python is available on a wide variety of platforms including Linux and Mac OS X. Let's understand how to set up our Python environment.
● The most up-to-date and current source code, binaries, documentation, news, etc., is available on the official website of Python.

2.4.1 Installation(using python IDLE):

● Installing python is generally easy, and nowadays many Linux and Mac OS distributions include a recent python.
● Download python from www.python.org
● When the download is completed, double click the file and follow the instructions to install it.
● When python is installed, a program called IDLE is also installed along with it. It provides a graphical user interface to work with python. Figure 4 : Python download

2.4.2 Installation(using Anaconda):

● Python programs are also executed using Anaconda.
● Anaconda is a free open source distribution of python for large scale data processing, predictive analytics and scientific computing.
● Conda is a package manager quickly installs and manages packages.
● In WINDOWS:

- Step 1: Open Anaconda.com/downloads in web browser.
- Step 2: Download python 3.4 version for (32-bitgraphic installer/64 -bit graphic installer)
- Step 3: select installation type(all users)
- Step 4: Select path(i.e. add anaconda to path & register anaconda as default python 3.4) next click install and next click finish
- Step 5: Open jupyter notebook (it opens in default browser)

2.5 PYTHON 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
- Variables are nothing but reserved memory locations to store values.
- Based on the data type of a variable, the interpreter allocates memory and decides what can be stored in the reserved memory
- Python variables do not need explicit declaration to reserve memory space. The declaration happens automatically when you assign a value to a variable.
- Python has various standard data types that are used to define the operations possible on them and the storage method for each of them
- Python has five standard data types –
  - Numbers
  - Strings
  - Lists
  - Tuples
  - Dictionary

2.5.1 Python Numbers:

- Number data types store numeric values. Number objects are created when you assign a value to them.
- Python supports four different numerical types − int (signed integers) long (long integers, they can also be represented in octal and hexadecimal) float (floating point real values) complex (complex numbers).

2.5.2 Python Strings:

- Strings in Python are identified as a contiguous set of characters represented in the quotation marks.
- Python allows for either pairs of single or double quotes.
- Subsets of strings can be taken using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the string and working their way from -1 at the end.
- The plus (+) sign is the string concatenation operator and the asterisk (*) is the repetition operator.

2.5.3 Python Lists:

- Lists are the most versatile of Python's compound data types.
- A list contains items separated by commas and enclosed within square brackets ([]).
- To some extent, lists are similar to arrays in C. One difference between them is that all the items belonging to a list can be of different data type.
- The values stored in a list can be accessed using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the list and working their way to end -1.
- The plus (+) sign is the list concatenation operator, and the asterisk (*) is the repetition operator.

2.5.4 Python Tuples:

- A tuple is another sequence data type that is similar to the list.
- A tuple consists of a number of values separated by commas. Unlike lists, however, tuples are enclosed within parentheses.
- The main differences between lists and tuples are: Lists are enclosed in brackets ([ ]) and their elements and size can be changed, while tuples are enclosed in parentheses (( )) and cannot be updated.
- Tuples can be thought of as read-only lists.
- For example − Tuples are fixed size in nature whereas lists are dynamic. In other words, a tuple is immutable whereas a list is mutable. You can't add elements to a tuple. Tuples have no append or extend method. You can't remove elements from a tuple. Tuples have no remove or pop method.

2.5.5 Python Dictionary:

- Python's dictionaries are kind of hash table type. They work like associative arrays or hashes found in Perl and consist of key-value pairs. A dictionary key can be almost any Python type, but are usually numbers or strings. Values, on the other hand, can be any arbitrary Python object.
- Dictionaries are enclosed by curly braces ({ }) and values can be assigned and accessed using square braces ([]).

- You can use numbers to "index" into a list, meaning you can use numbers to find out what's in lists. You should know this about lists by now, but make sure you understand that you can only use numbers to get items out of a list.
- What a dict does is let you use anything, not just numbers. Yes, a dict associates one thing to another, no matter what it is.

2.6 PYTHON FUNCTION:

2.6.1 Defining a Function:

- You can define functions to provide the required functionality. Here are simple rules to define a function in Python. Function blocks begin with the keyword def followed by the function name and parentheses (i.e.()).
- Any input parameters or arguments should be placed within these parentheses. You can also define parameters inside these parentheses
- The code block within every function starts with a colon (:) and is indented. The statement returns [expression] exits a function, optionally passing back an expression to the caller. A return statement with no arguments is the same as return None.

2.6.2 Calling a Function:

- Defining a function only gives it a name, specifies the parameters that are to be included in the function and structures the blocks of code. Once the basic structure of a function is finalized, you can execute it by calling it from another function or directly from the Python prompt.

2.7 PYTHON USING OOP's CONCEPTS:

2.7.1 Class:

- Class: A user-defined prototype for an object that defines a set of attributes that characterize any object of the class. The attributes are data members (class variables and instance variables) and methods, accessed via dot notation.
- Class variable: A variable that is shared by all instances of a class. Class variables are defined within a class but outside any of the class's methods. Class variables are not used as frequently as instance variables are.
- Data member: A class variable or instance variable that holds data associated with a class and its objects.
- Instance variable: A variable that is defined inside a method and belongs only to the current instance of a class.

Defining a Class:

- We define a class in a very similar way how we define a function.
- Just like a function ,we use parentheses and a colon after the class name(i.e. ():) when we define a class. Similarly, the body of our class is indented like a functions body is.

2.7.2 __init__ method in Class:

- The init method — also called a constructor — is a special method that runs when an instance is created so we can perform any tasks to set up the instance.
- The init method has a special name that starts and ends with two underscores:__init__().

CHAPTER 3

CASE STUDY

3.1 Problem Statement:

To predict whether a patient has diabetes or not

3.2 Data Set:

The given data set consists of the following parameters

- Pregnancies
- Glucose
- BloodPressure
- SkinThickness
- Insulin
- BMI
- DiabetesPedigreeFunction
- Age
- Outcome

3.3 Objective of the case study:

The objective of the dataset is to predict whether a patient has diabetes or not, based on certain diagnostic measurements included in the dataset. The datasets consists of several medical predictor variables and one target variable, Outcome. Predictor variables includes the number of pregnancies the patient has had, their BMI, insulin level, age, and so on.

CHAPTER 4

MODEL BUILDING

4.1 PREPROCESSING OF THE DATA:

Preprocessing of the data actually involves the following steps:

4.1.1 GETTING THE DATASET:

We can get the data set from the database or
We can get the data from client.

4.1.2 IMPORTING THE LIBRARIES:

We have to import the libraries as per the requirement of the algorithm.

Importing libraries

```
In [1]: # Importing libraries
        import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns

        import warnings
        warnings.filterwarnings('ignore')
```

4.1.3 IMPORTING THE DATA-SET:

Pandas in python provide an interesting method read_csv(). The read_csv function reads the entire dataset from a comma separated values file and we can assign it to a DataFrame to which all the operations can be performed. It helps us to access each and every row as well as columns and each and every value can be access using the dataframe. Any missing value or NaN value have to be cleaned.

Reading dataset

```
In [3]: # reading the data set
        dataset = pd.read_csv('diabetes.csv')
        dataset.head()
```

Out[3]:

	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

4.1.4 Dataset dimensions:

```
In [4]: # Dataset dimensions - (rows, columns)
        dataset.shape
```

```
Out[4]: (768, 9)
```

4.1.5 To check the features data type:

```
In [5]: # Features data-type
        dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
Pregnancies                 768 non-null int64
Glucose                     768 non-null int64
BloodPressure               768 non-null int64
SkinThickness               768 non-null int64
Insulin                     768 non-null int64
BMI                         768 non-null float64
DiabetesPedigreeFunction    768 non-null float64
Age                         768 non-null int64
Outcome                     768 non-null int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

4.1.6 Statistical summary of the data:

```
In [6]: # Statistical summary
        dataset.describe().T
```

Out[6]:

	count	mean	std	min	25%	50%	75%	max
Pregnancies	768.0	3.845052	3.369578	0.000	1.00000	3.0000	6.00000	17.00
Glucose	768.0	120.894531	31.972618	0.000	99.00000	117.0000	140.25000	199.00
BloodPressure	768.0	69.105469	19.355807	0.000	62.00000	72.0000	80.00000	122.00
SkinThickness	768.0	20.536458	15.952218	0.000	0.00000	23.0000	32.00000	99.00
Insulin	768.0	79.799479	115.244002	0.000	0.00000	30.5000	127.25000	846.00
BMI	768.0	31.992578	7.884160	0.000	27.30000	32.0000	36.60000	67.10
DiabetesPedigreeFunction	768.0	0.471876	0.331329	0.078	0.24375	0.3725	0.62625	2.42
Age	768.0	33.240885	11.760232	21.000	24.00000	29.0000	41.00000	81.00
Outcome	768.0	0.348958	0.476951	0.000	0.00000	0.0000	1.00000	1.00

4.1.7 To check the number of null values in the given data set:

```
In [7]: # number of null values
        dataset.isnull().sum()
```

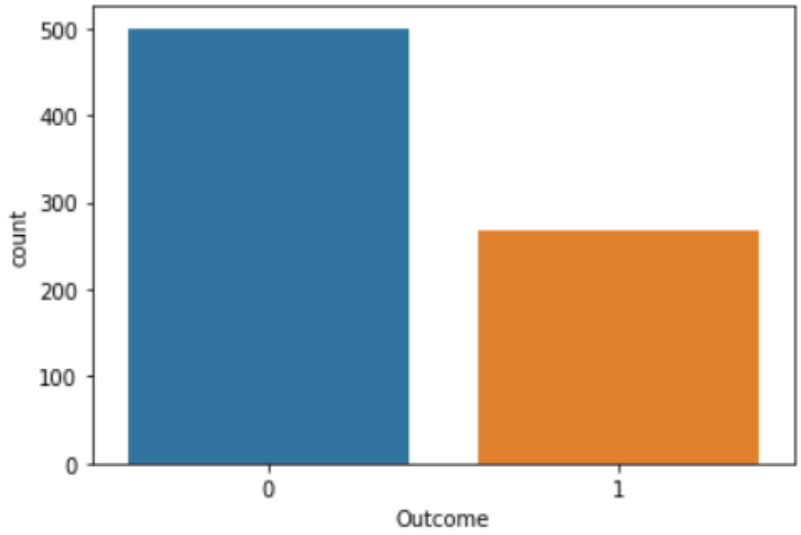
```
Out[7]: Pregnancies                 0
        Glucose                     0
        BloodPressure               0
        SkinThickness               0
        Insulin                     0
        BMI                         0
        DiabetesPedigreeFunction    0
        Age                         0
        Outcome                     0
        dtype: int64
```

4.2 DATA VISUALISATION:

4.2.1 Countplot of the outcome parameter:

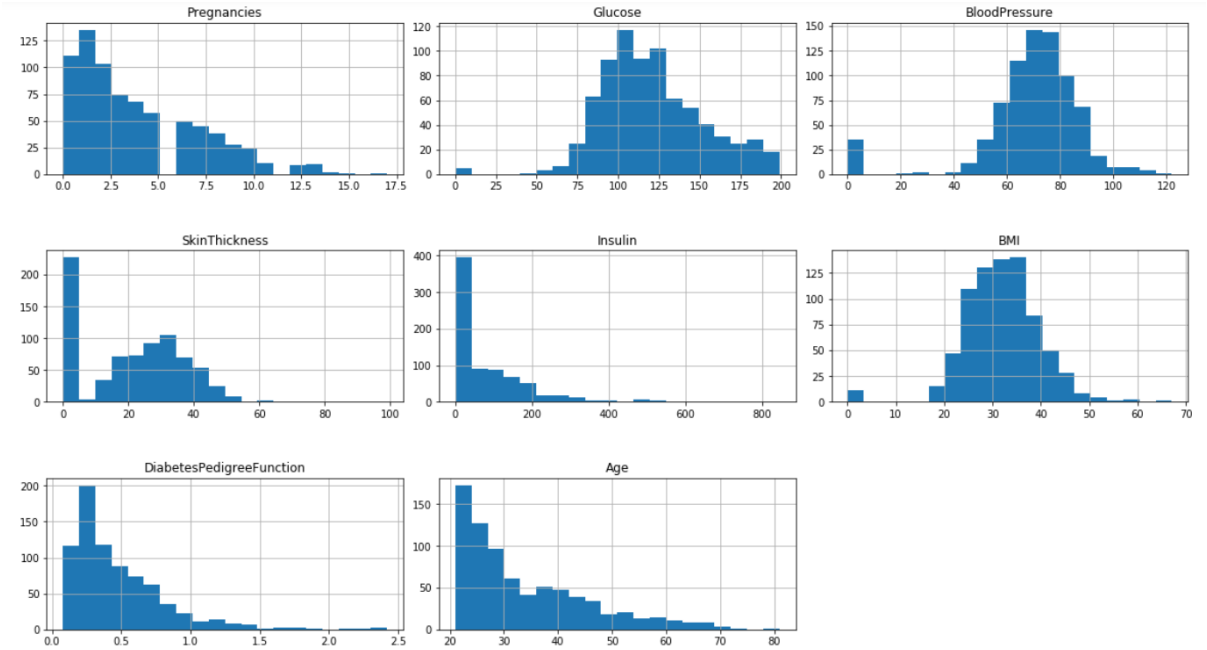
In [8]:
```
# Outcome countplot
sns.countplot(x = 'Outcome',data = dataset)
```

Out[8]: <matplotlib.axes._subplots.AxesSubplot at 0x1817cf17888>



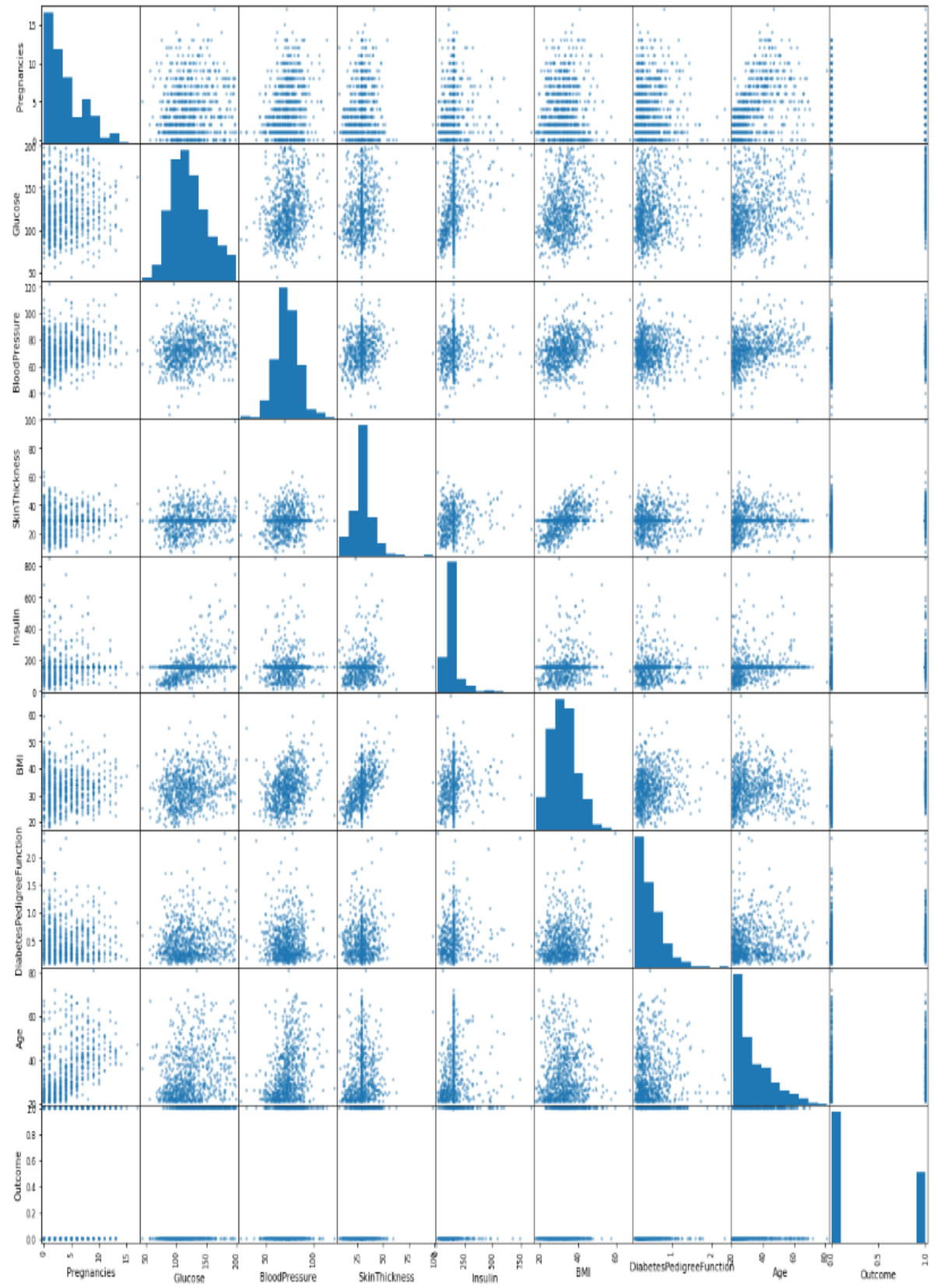
4.2.2 Histogram of each feature in the data set:

In [9]:
```
# Histogram of each feature in the dataset
import itertools

col = dataset.columns[:8]
plt.subplots(figsize = (20, 15))
length = len(col)

for i, j in itertools.zip_longest(col, range(length)):
    plt.subplot((length/2), 3, j + 1)
    plt.subplots_adjust(wspace = 0.1,hspace = 0.5)
    dataset[i].hist(bins = 20)
    plt.title(i)
plt.show()
```



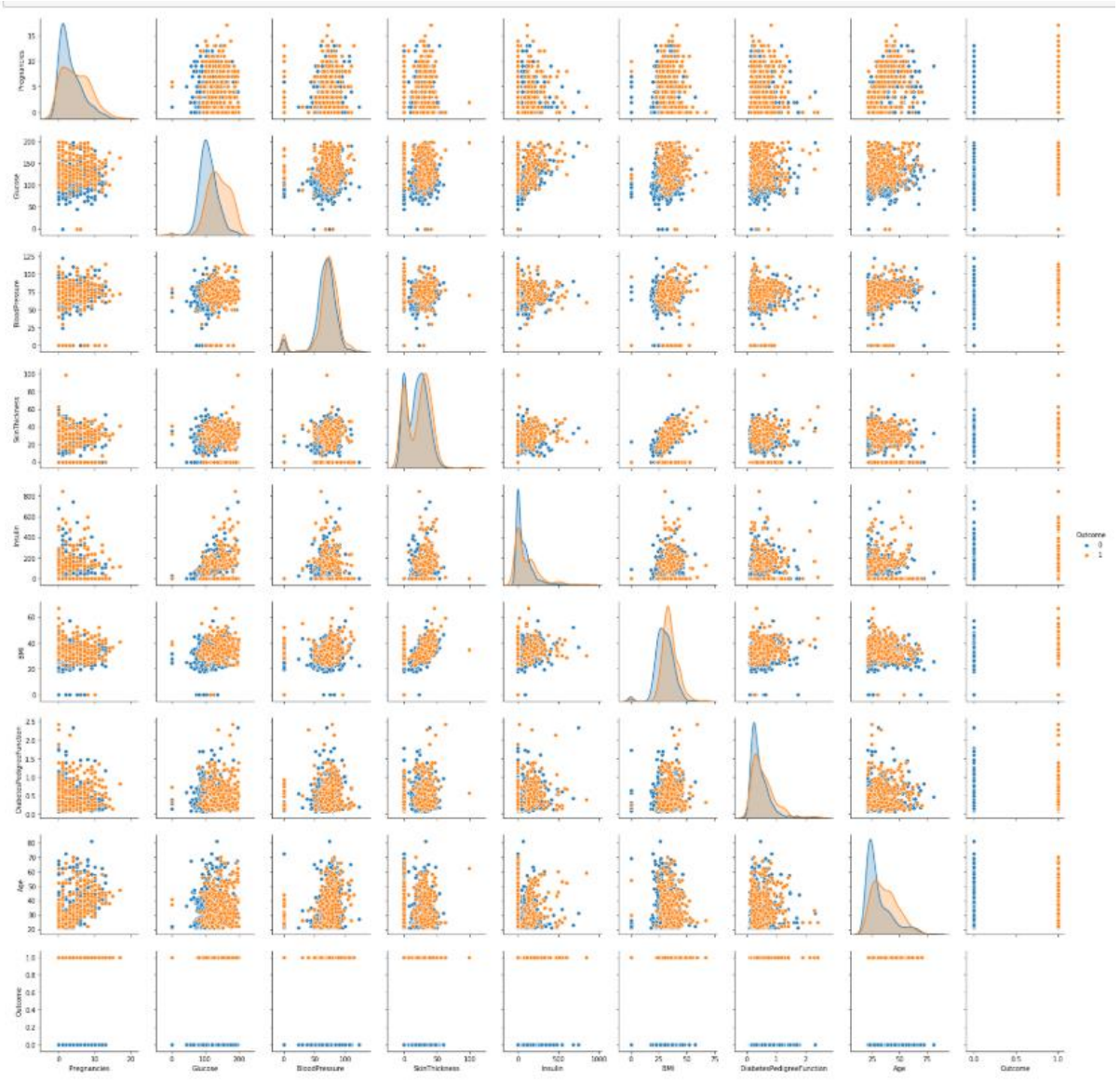
4.2.3 Scatterplot matrix of the given data set:

```
In [39]: # Scatter plot matrix
         from pandas.plotting import scatter_matrix
         scatter_matrix(dataset, figsize = (20, 20));
```


Pregnancies
Glucose
BloodPressure
SkinThickness
Insulin
BMI
DiabetesPedigreeFunction
Age
Outcome

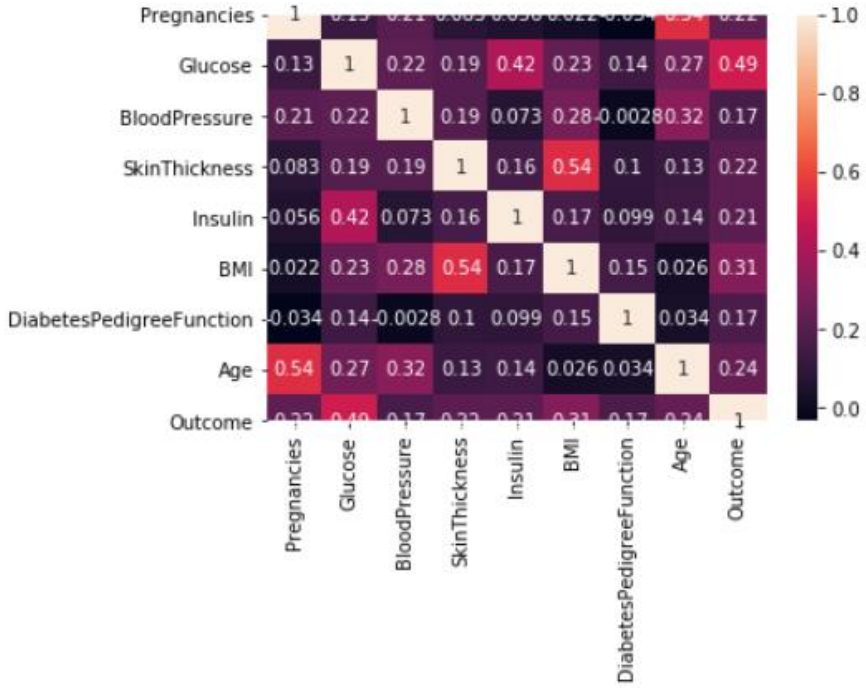
4.2.4 Pairplot visualisation for the given data set:

```
In [13]: # Pairplot
         sns.pairplot(data = dataset, hue = 'Outcome')
         plt.show()
```



4.2.5 heatmap for the given data set:

```
In [40]: # Heatmap
         sns.heatmap(dataset.corr(), annot = True)
         plt.show()
```



4.3 DATA PROCESSING:

Copying into a new data set without changing the original data set:

```
In [15]: dataset_new = dataset
```

4.3.1 Replacing all the zero values with NaN:

```
In [17]: # Replacing zero values with NaN
dataset_new[["Glucose", "BloodPressure", "SkinThickness", "Insulin", "BMI"]] = dataset_new[["Glucose",
                                                    "BloodPressure","SkinThickness", "Insulin", "BMI"]].replace(0, np.NaN)

dataset_new.head()
```

Out[17]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148.0	72.0	35.0	NaN	33.6	0.627	50	1
1	1	85.0	66.0	29.0	NaN	26.6	0.351	31	0
2	8	183.0	64.0	NaN	NaN	23.3	0.672	32	1
3	1	89.0	66.0	23.0	94.0	28.1	0.167	21	0
4	0	137.0	40.0	35.0	168.0	43.1	2.288	33	1

4.3.2 Count of NaN values:

```
In [18]: # Count of NaN
dataset_new.isnull().sum()
```

```
Out[18]: Pregnancies                 0
Glucose                     5
BloodPressure              35
SkinThickness             227
Insulin                   374
BMI                        11
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

4.3.3 Replacing NaN values with mean values:

In [19]:
```
# Replacing NaN with mean values
dataset_new["Glucose"].fillna(dataset_new["Glucose"].mean(), inplace = True)
dataset_new["BloodPressure"].fillna(dataset_new["BloodPressure"].mean(), inplace = True)
dataset_new["SkinThickness"].fillna(dataset_new["SkinThickness"].mean(), inplace = True)
dataset_new["Insulin"].fillna(dataset_new["Insulin"].mean(), inplace = True)
dataset_new["BMI"].fillna(dataset_new["BMI"].mean(), inplace = True)
```

In [20]:
```
# Statistical summary
dataset_new.describe().T
```

Out[20]:

	count	mean	std	min	25%	50%	75%	max
Pregnancies	768.0	3.845052	3.369578	0.000	1.00000	3.000000	6.000000	17.00
Glucose	768.0	121.686763	30.435949	44.000	99.75000	117.000000	140.250000	199.00
BloodPressure	768.0	72.405184	12.096346	24.000	64.00000	72.202592	80.000000	122.00
SkinThickness	768.0	29.153420	8.790942	7.000	25.00000	29.153420	32.000000	99.00
Insulin	768.0	155.548223	85.021108	14.000	121.50000	155.548223	155.548223	846.00
BMI	768.0	32.457464	6.875151	18.200	27.50000	32.400000	36.600000	67.10
DiabetesPedigreeFunction	768.0	0.471876	0.331329	0.078	0.24375	0.372500	0.626250	2.42
Age	768.0	33.240885	11.760232	21.000	24.00000	29.000000	41.000000	81.00
Outcome	768.0	0.348958	0.476951	0.000	0.00000	0.000000	1.000000	1.00

4.3.4 Feature scaling using MinMaxScaler:

```
# Feature scaling using MinMaxScaler
from sklearn.preprocessing import MinMaxScaler
sc = MinMaxScaler(feature_range = (0, 1))
dataset_scaled = sc.fit_transform(dataset_new)
```

```
dataset_scaled = pd.DataFrame(dataset_scaled)
```

```
# Selecting features - [Glucose, Insulin, BMI, Age]
X = dataset_scaled.iloc[:, [1, 4, 5, 7]].values
Y = dataset_scaled.iloc[:, 8].values
```

4.3.5 Spliting the data into training and testing:

In [24]:
```
# Splitting X and Y
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.20, random_state = 42, stratify = dataset_new['Outcome'])
```

In [25]:
```
# Checking dimensions
print("X_train shape:", X_train.shape)
print("X_test shape:", X_test.shape)
print("Y_train shape:", Y_train.shape)
print("Y_test shape:", Y_test.shape)
```

```
X_train shape: (614, 4)
X_test shape: (154, 4)
Y_train shape: (614,)
Y_test shape: (154,)
```

4.4 DATA MODELING:

4.4.1 Using logistic regression algorithm:

In [26]:
```
# Logistic Regression Algorithm
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression(random_state = 42)
logreg.fit(X_train, Y_train)
```

Out[26]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=42, solver='warn', tol=0.0001, verbose=0,
                   warm_start=False)
```

4.4.2 Using K-nearest neighbours algorithm:

```
In [28]: # K nearest neighbors Algorithm
         from sklearn.neighbors import KNeighborsClassifier
         knn = KNeighborsClassifier(n_neighbors = 24, metric = 'minkowski', p = 2)
         knn.fit(X_train, Y_train)

Out[28]: KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                              metric_params=None, n_jobs=None, n_neighbors=24, p=2,
                              weights='uniform')
```

4.4.3 Using support vector classifier algorithm:

```
In [29]: # Support Vector Classifier Algorithm
         from sklearn.svm import SVC
         svc = SVC(kernel = 'linear', random_state = 42)
         svc.fit(X_train, Y_train)

Out[29]: SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
             decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
             kernel='linear', max_iter=-1, probability=False, random_state=42,
             shrinking=True, tol=0.001, verbose=False)
```

4.4.4 Using naive baye's algorithm:

```
In [30]: # Naive Bayes Algorithm
         from sklearn.naive_bayes import GaussianNB
         nb = GaussianNB()
         nb.fit(X_train, Y_train)

Out[30]: GaussianNB(priors=None, var_smoothing=1e-09)
```

4.4.5 Using Decision tree algorithm:

```
In [31]: # Decision tree Algorithm
         from sklearn.tree import DecisionTreeClassifier
         dectree = DecisionTreeClassifier(criterion = 'entropy', random_state = 42)
         dectree.fit(X_train, Y_train)

Out[31]: DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=None,
                                max_features=None, max_leaf_nodes=None,
                                min_impurity_decrease=0.0, min_impurity_split=None,
                                min_samples_leaf=1, min_samples_split=2,
                                min_weight_fraction_leaf=0.0, presort=False,
                                random_state=42, splitter='best')
```

4.4.6 Using Random forest algorithm:

```
In [32]: # Random forest Algorithm
         from sklearn.ensemble import RandomForestClassifier
         ranfor = RandomForestClassifier(n_estimators = 11, criterion = 'entropy', random_state = 42)
         ranfor.fit(X_train, Y_train)

Out[32]: RandomForestClassifier(bootstrap=True, class_weight=None, criterion='entropy',
                               max_depth=None, max_features='auto', max_leaf_nodes=None,
                               min_impurity_decrease=0.0, min_impurity_split=None,
                               min_samples_leaf=1, min_samples_split=2,
                               min_weight_fraction_leaf=0.0, n_estimators=11,
                               n_jobs=None, oob_score=False, random_state=42, verbose=0,
                               warm_start=False)
```

4.4.7 Making predictions on the test data set:

```
In [33]: # Making predictions on test dataset
         Y_pred_logreg = logreg.predict(X_test)
         Y_pred_knn = knn.predict(X_test)
         Y_pred_svc = svc.predict(X_test)
         Y_pred_nb = nb.predict(X_test)
         Y_pred_dectree = dectree.predict(X_test)
         Y_pred_ranfor = ranfor.predict(X_test)
```

4.5 MODEL EVALUATION:

Evaluating using accuracy_score_metric

```
In [34]: # Evaluating using accuracy_score metric
         from sklearn.metrics import accuracy_score
         accuracy_logreg = accuracy_score(Y_test, Y_pred_logreg)
         accuracy_knn = accuracy_score(Y_test, Y_pred_knn)
         accuracy_svc = accuracy_score(Y_test, Y_pred_svc)
         accuracy_nb = accuracy_score(Y_test, Y_pred_nb)
         accuracy_dectree = accuracy_score(Y_test, Y_pred_dectree)
         accuracy_ranfor = accuracy_score(Y_test, Y_pred_ranfor)
```

4.6 ACCURACY ON THE TEST SET:

```
In [35]: # Accuracy on test set
         print("Logistic Regression: " + str(accuracy_logreg * 100))
         print("K Nearest neighbors: " + str(accuracy_knn * 100))
         print("Support Vector Classifier: " + str(accuracy_svc * 100))
         print("Naive Bayes: " + str(accuracy_nb * 100))
         print("Decision tree: " + str(accuracy_dectree * 100))
         print("Random Forest: " + str(accuracy_ranfor * 100))
```

```
Logistic Regression: 71.42857142857143
K Nearest neighbors: 78.57142857142857
Support Vector Classifier: 73.37662337662337
Naive Bayes: 71.42857142857143
Decision tree: 68.18181818181817
Random Forest: 75.97402597402598
```

From the above comparison, we can observe that K Nearest neighbors gets the highest accuracy of 78.57

4.7 CONFUSION MATRIX

4.7.1 Confusion matrix :

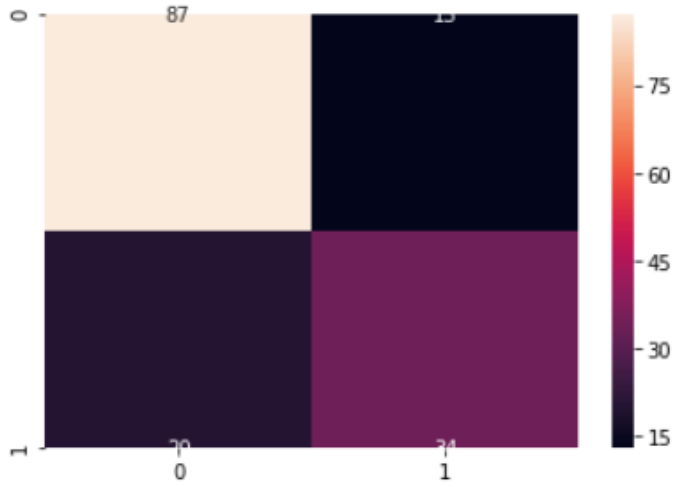
```
In [36]: # Confusion matrix
         from sklearn.metrics import confusion_matrix
         cm = confusion_matrix(Y_test, Y_pred_knn)
         cm
```

```
Out[36]: array([[87, 13],
                [20, 34]], dtype=int64)
```

4.7.2 Heatmap of the confusion matrix:

```
In [37]: # Heatmap of Confusion matrix
         sns.heatmap(pd.DataFrame(cm), annot=True)
```

```
Out[37]: <matplotlib.axes._subplots.AxesSubplot at 0x1817f6dde48>
```



4.8 FINAL CLASSIFICATION REPORT:

```
In [38]: # Classification report
         from sklearn.metrics import classification_report
         print(classification_report(Y_test, Y_pred_knn))
```

```
              precision    recall  f1-score   support

         0.0       0.81      0.87      0.84       100
         1.0       0.72      0.63      0.67        54

    accuracy                           0.79       154
   macro avg       0.77      0.75      0.76       154
weighted avg       0.78      0.79      0.78       154
```

CONCLUSION:

One of the important real-world medical problems is the detection of diabetes at its early stage. In this study, systematic efforts are made in designing a system which results in the prediction of disease like diabetes. During this work, three machine learning classification algorithms are studied and evaluated on various measures. Experiments are performed on Pima Indians Diabetes Database. Experimental results determine the adequacy of the designed system with an achieved accuracy of approx. 78 % . In future, the designed system with the used machine learning classification algorithms can be used to predict or diagnose other diseases. The work can be extended and improved for the automation of diabetes analysis including some other machine learning algorithms.

REFERENCES:

https://www.kaggle.com/uciml/pima-indians-diabetes-database