A REPORT

ON

# (DIABETES PREDICTION)

BY

Name: Akiti Sri Kalyan Reddy ID.No.:19STUCHH010130

AT

(ICFAI TECH ,INDIA, ,HYDERABAD))

An assignment of data warehousing and mining, station of Icfai tech

FACULTY OF SCIENCE AND TECHNOLOGY, IFHE UNIVERSITY

(December, 2021)

A REPORT

ON

(DIABETES PREDICTION)

BY

Name:Akiti Sri kalyan Reddy ID.No:19STUCHH010130 Discipline:DSAI

Data Warehousing And Mining Course

AT

(ICFAI TECH , INDIA, HYDERABAD)

An assignment of data warehousing and mining, station of Icfai tech

FACULTY OF SCIENCE AND TECHNOLOGY, IFHE UNIVERSITY

ACKNOWLEDGEMENT

First of all I would like to express my heartfelt gratitude to Mr.Deevena Raju, faculty of Icfai Tech as he played a crucial role in guiding us throughout this program. This is my report on Diabetes prediction at IcfaiTech in 2021. I wish to express my sincere thanks to my course incharge . I also would like to thank Icfai Tech for giving me a golden opportunity for providing the practical internship training and giving the constant guidance, necessary support, co-operation, encouragement, and the fullest effort of them to the success of this opportunity, all are grateful and unforgettable. My special thanks are to the Icfai tech and the community staff for their fullest support which contributed immensely for successful completion of this internship training. I wish to express my profound gratitude to all of them.

Thank You

## ABSTRACT

Administrations are the foundation of current economies and are progressively upheld by innovation. In the interim, there is a speed up development of new advancements that can gain from themselves, giving to an ever increasing extent significant outcomes, for example Data analysis (data service).While there have been critical advances in these services, the effects of this innovation on administration arrangement at this point are unclear. Calculated exploration asserts that the data gathering industry offers an approach to increase digital utilization or position it as an advantage to human kind. The objective of this investigation is to get an overview of how data will be useful in particular by understanding things in the digital world, and how they are, and will, be beneficial to users.You'll need to carry out tasks related to data gathering ,data analysis, bookkeeping, making contracts, documents and records, data visualization, data preprocessing, data modeling ,model evaluation, and ongoing self-learning, and more .



KEY WORDS :data service, data visualization and data analysis

**Faculty of Science & Technology, IFHE University**

**Station:** ICFAI TECH, INDIA, HYDERABAD **Centre: Hyderabad**

**Duration:** 10 days **Date of Start:** 17/11/2021 **Date of Submission:** 29/11/2021

**Title of the Project:** DIABETES PREDICTION

**Name of the Student:** AKITI SRI KALYAN REDDY **ID. No:**19STUCHH010130

**Discipline:** DS&AI

**Name of the faculty mentor:** Mr. Deevena Raju

**Project Areas:** data analysis, data preprocessing and data molding

Signature of Student Signature of faculty

27-11-2021 27-11-2021

# TABLE OF CONTENTS

# Chapter -1
# INTRODUCTION

In this task, our goal is to foresee whether or not the patient has diabetes dependent on different highlights like Glucose level, Insulin, Age, BMI. We will play out every one of the means from Data social event to Model organization. During Model assessment, we analyze different AI calculations based on accuracy_score metric and track down the best one. Then, at that point, we make a web application utilizing Flask which is a python miniature structure.

# Chapter-2
# COLLECTING DATA

The information was gathered and made accessible by "Public Institute of Diabetes and Digestive and Kidney Diseases" as a component of the pima Indians diabetes database.Several limitations were put on the determination of these examples from a bigger data set. Specifically, all patients here have a place within the Pima Indian legacy.

In this, the first task we should assign ourselves is to gather information.which is a crucial step in any data collection.For this project we are going to rely on different types of sources.For example to find a few information whose topic would be on diabetes prediction. That information would be useful in our next process. We can't ignore that it should be based on facts for accurate data, here the task is to find information on diabetes prediction and also the information suggested by famous personalities on the same topic. After a brief research over the internet, we might find useful information .We should also provide some links of that information online. which are to be documented.

Information assortment is characterized as the method of gathering, estimating and investigating precise bits of knowledge for research utilizing standard approved strategies. A scientist can assess their theory based on gathered information. Much of the time, information assortment is the essential and most significant stage for research, independent of the field of exploration. The methodology of information assortment is diverse for various fields of study, contingent upon the necessary data.

The most basic goal of information assortment is guaranteeing that data rich and dependable information is gathered for measurable investigation so information driven choices can be made for research.

In-person meets consistently are better, however the enormous disadvantage is the snare you may fall into assuming you don't do them routinely. It is costly to consistently lead interviews and not directing enough meetings may give you bogus up-sides. Approving your exploration is nearly just about as significant as

planning and leading it. We've seen many cases where after the exploration is directed – assuming the outcomes don't coordinate with the "stomach feel" of upper administration, it has been excused off as narrative and a "once" peculiarity. To keep away from such snares, we emphatically suggest that information assortment be done on an "continuous and ordinary" premise. This will help you in looking at and dissecting the adjustment of discernments as per advertising accomplished for your items/administrations. The other issue here is test size. To be certain with your examination you need to talk with sufficient individuals to remove the periphery components.

A few years prior there was a considerable amount of conversation about online overviews and their measurable legitimacy. The way that only one out of every odd client had web availability was one of the fundamental worries. Albeit a portion of the conversations are as yet legitimate, the scope of the web as a method for correspondence has become crucial in most of client corporations. As indicated by the US Census Bureau, the quantity of families with PCs has multiplied somewhere in the range of 1997 and 2001.

**Example of an data collection:**

Information assortment is a significant part of exploration. We should think about an illustration of a portable maker, organization X, which is dispatching another item variation. To direct research about highlights, value range, target market, contender examination and so on information must be gathered from suitable sources. The advertising group can direct different information assortment exercises, for example, online reviews or center gatherings.

The overview ought to have the appropriate inquiries concerning highlights and evaluating, for example, "What are the main 3 elements anticipated from an impending item?" or "What amount are your liable to spend on this item?" or "Which contenders give comparable items?" and so on

For leading a center gathering, the showcasing group ought to choose the members just as the middle person. The subject of conversation and objective behind directing a center gathering ought to be clarified ahead of time so a convincing conversation can be led.

Information assortment strategies are picked relying upon the accessible assets. For instance, leading polls and reviews would require the least assets while center gatherings require reasonably high assets.

# Chapter-3
# ANACONDA NAVIGATOR

We will be using Jupyter Notebook to carry out the executions and obtain the required output.
In the jupyter notebook we will be using the following the libraries:

1)pandas
2)numpy
3)seaborn
4)matplotlib
5)sklearn
6)joblib

Anaconda Navigator is a work area graphical UI (GUI) remembered for Anaconda appropriation that permits you to dispatch applications and effectively oversee conda bundles, conditions, and channels without utilizing order line orders. Pilot can look for bundles on Anaconda.org or in a nearby Anaconda Repository. It is accessible for Windows, macOS, and Linux.

To run, numerous logical bundles rely upon explicit renditions of different bundles. Information researchers regularly utilize various variants of many bundles and utilize numerous conditions to isolate these various forms.

The order line program conda is both a bundle administrator and a climate supervisor. This aides information researchers guarantee that every rendition of each bundle has every one of the conditions it requires and works effectively.

Pilot is a simple, point-and-snap method for working with bundles and conditions without expecting to type conda orders in a terminal window. You can utilize it to find the bundles you need, introduce them in a climate, run the bundles, and update them – all inside Navigator.

# Chapter -4
# DATA ANALYSIS

**Importing libraries :**

As of now we'll utilize Python and a portion of its well known information science related bundles. As a matter of first importance, we will import panda's to peruse our information from a CSV record and control it for additional utilization. We will likewise utilize numpy to change our information into an organization reasonable to take care of our grouping model. We'll utilize ocean conceived and Matplotlib for representations. We will then, at that point, import Logistic Regression calculation from sklearn This calculation will assist us with building our order model. Ultimately, we will utilize joblib accessible in sklearn to save our model for sometime later.

```
In [56]: # Step 0: Import libraries and Dataset 19STUCHH010130

In [57]: # Importing libraries
         import pandas as pd
         import numpy as np
         import matplotlib.pyplot as plt
         import seaborn as sns

         import warnings

In [58]: # Importing dataset
         dataset = pd.read_csv('D:/assingment/Diabetes-Prediction-master/diabetes.csv')
```

**Descriptive Analysis:**

We have our information saved in a CSV document called diabetes.csv. We originally read our dataset into a pandas dataframe called diabetesDF, and afterward utilize the head() capacity to show the initial five records from our dataset.

```
In [58]: # Importing dataset
         dataset = pd.read_csv('D:/assingment/Diabetes-Prediction-master/diabetes.csv')

In [59]: # Step 1: Descriptive Statistics 19STUCHH010130

In [60]: # Preview data
         dataset.head()
```

Out[60]:

| | 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 |

The accompanying highlights have been given to assist us with anticipating whether or not an individual is diabetic:

Pregnancies: Number of times pregnant

Glucose: Plasma glucose fixation north of 2 hours in an oral glucose resistance test

BloodPressure: Diastolic circulatory strain (mm Hg)

SkinThickness: Triceps skin overlap thickness (mm)

Insulin: 2-Hour serum insulin (mu U/ml)

BMI: Body mass record (weight in kg/(stature in m)2)

DiabetesPedigreeFunction: Diabetes family work (a capacity which scores probability of diabetes dependent on family ancestry)

Age: Age (a long time)

Result: Class variable (0 if non-diabetic, 1 if diabetic)

We should likewise ensure that our information is perfect (has no invalid qualities, and so on)

```
In [61]: # Dataset dimensions - (rows, columns)
         dataset.shape

Out[61]: (768, 9)

In [62]: # Features data-type
         dataset.info()

         <class 'pandas.core.frame.DataFrame'>
         RangeIndex: 768 entries, 0 to 767
         Data columns (total 9 columns):
          #   Column                    Non-Null Count  Dtype
         ---  ------                    --------------  -----
          0   Pregnancies               768 non-null    int64
          1   Glucose                   768 non-null    int64
          2   BloodPressure             768 non-null    int64
          3   SkinThickness             768 non-null    int64
          4   Insulin                   768 non-null    int64
          5   BMI                       768 non-null    float64
          6   DiabetesPedigreeFunction  768 non-null    float64
          7   Age                       768 non-null    int64
          8   Outcome                   768 non-null    int64
         dtypes: float64(2), int64(7)
         memory usage: 54.1 KB

In [63]: # Statistical summary
         dataset.describe().T
```

Data exploration

Allow us presently to investigate our informational index to get a vibe of what it resembles and get a few experiences about it.

How about we start by tracking down the relationship of each pair of highlights (and the result variable), and envision the connections utilizing a heatmap.

```
In [63]: # Statistical summary
         dataset.describe().T
```

Out[63]:

| | 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 |

```
In [64]: # Count of null values
         dataset.isnull().sum()
```

Out[64]:

```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

In the above heatmap, more brilliant shadings demonstrate more relationships. As we can see from the table and the heatmap, glucose levels, age, BMI and number of pregnancies all have critical relationships with the result variable. Likewise notice the connection between sets of elements, similar to age and pregnancies, or insulin and skin thickness.

We should likewise take a gander at the number of individuals in the dataset who are diabetic and the number who are not. The following is the barplot of the equivalent:

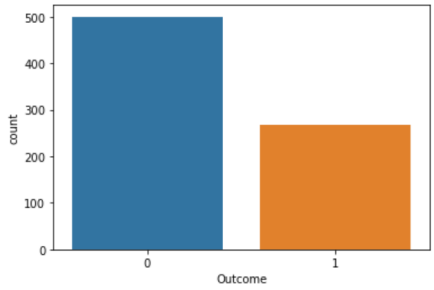
It is additionally useful to envision relations between a solitary variable and the result. Underneath, we'll see the connection among age and result. You can comparably envision other elements. The figure is a plot of the mean age for every one of the result classes. We can see that the mean period of individuals having diabetes is higher.

**Data visualization**

In [65]:
```
# Step 2: Data Visualization  19STUCHH010130
```

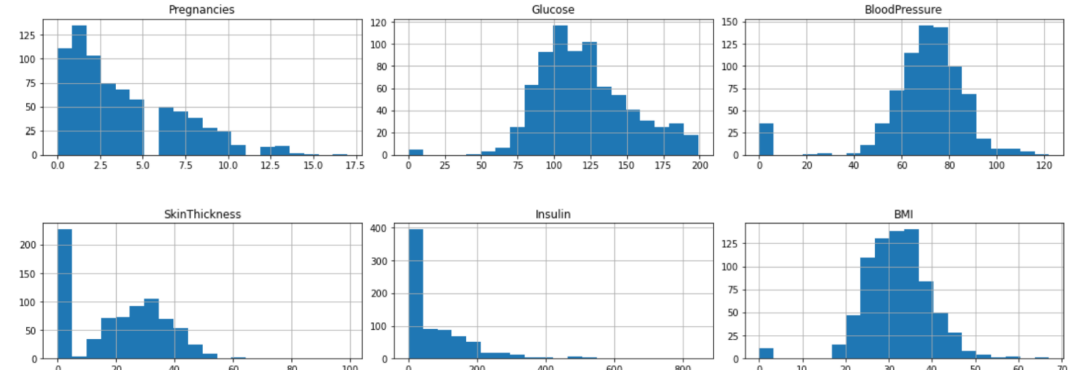
In [66]:
```
# Outcome countplot
sns.countplot(x = 'Outcome',data = dataset)
```

Out[66]: <AxesSubplot:xlabel='Outcome', ylabel='count'>

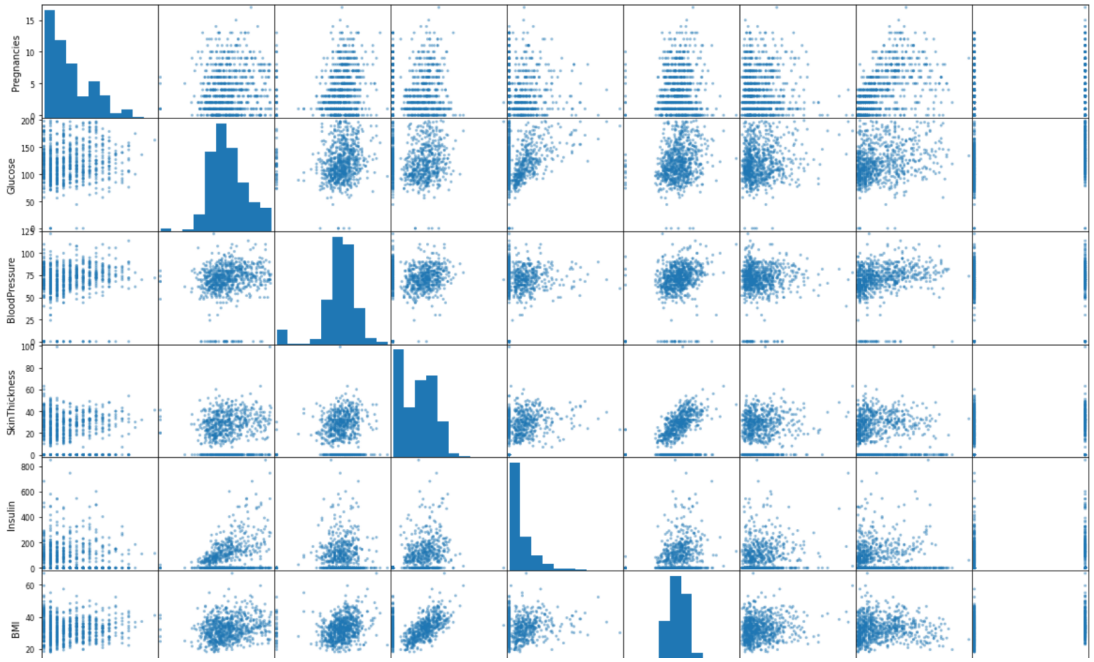



In [67]:
```
# Histogram of each feature
import itertools

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()
```

```
<ipython-input-67-e4c2ff98c5ff>:9: MatplotlibDeprecationWarning: Passing non-integers as three-element position specification i
s deprecated since 3.3 and will be removed two minor releases later.
  plt.subplot((length/2), 3, j + 1)
```

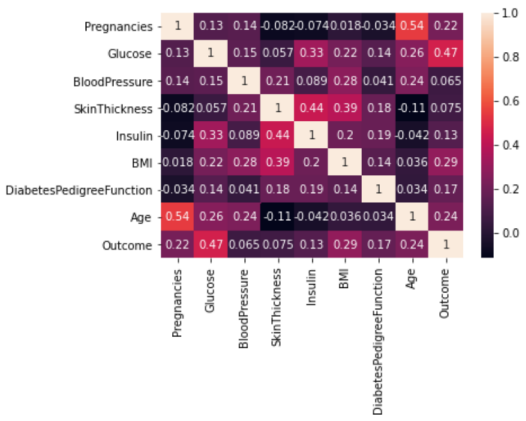


In [68]:
```
# Scatter plot matrix
from pandas.plotting import scatter_matrix
scatter_matrix(dataset, figsize = (20, 20));
```




In [69]:
```
# Pairplot
sns.pairplot(data = dataset, hue = 'Outcome')
plt.show()
```

```
In [70]: # Heatmap
         sns.heatmap(dataset.corr(), annot = True)
         plt.show()
```




The countplot lets us know that the dataset is imbalanced, as the quantity of patients who don't have diabetes is more than the people who do.

From the connection heatmap, we can see that there is a high relationship among Outcome and [Glucose, BMI, Age, Insulin]. We can choose these highlights to acknowledge input from the client and anticipate the result.

**Data processing:**

```
In [71]: # Step 3: Data Preprocessing  19STUCHH010130

In [72]: dataset_new = dataset

In [73]: # Replacing zero values with NaN
         dataset_new[["Glucose", "BloodPressure", "SkinThickness", "Insulin", "BMI"]] = dataset_new[["Glucose", "BloodPressure", "SkinThic

In [74]: # Count of NaN
         dataset_new.isnull().sum()

Out[74]: Pregnancies                 0
         Glucose                     5
         BloodPressure              35
         SkinThickness             227
         Insulin                   374
         BMI                        11
         DiabetesPedigreeFunction    0
         Age                         0
         Outcome                     0
         dtype: int64

In [75]: # 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 this dataset, the missing qualities are addressed by zero qualities that should be supplanted. The zero qualities are supplanted by NaN so that missing qualities can

undoubtedly be credited utilizing the fillna() order.

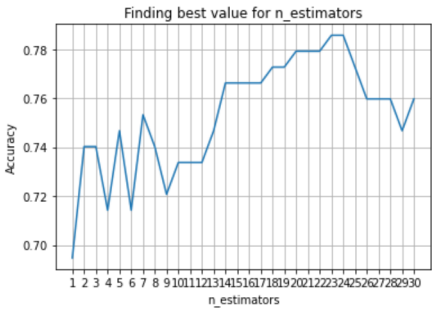
```
In [77]: # Feature scaling using MinMaxScaler
         from sklearn.preprocessing import MinMaxScaler
         sc = MinMaxScaler(feature_range = (0, 1))
         dataset_scaled = sc.fit_transform(dataset_new)

In [78]: dataset_scaled = pd.DataFrame(dataset_scaled)
```

We perform Feature scaling on the dataset utilizing Minmaxscaler() so it scales the whole dataset to such an extent that it lies somewhere in the range of 0 and 1. It is a significant preprocessing venture for some calculations.

```
In [79]: # Selecting features - [Glucose, Insulin, BMI, Age]
         X = dataset_scaled.iloc[:, [1, 4, 5, 7]].values
         Y = dataset_scaled.iloc[:, 8].values

In [80]: # 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 [81]: # 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,)
```

In the element connection heatmap, we can see that Glucose, Insulin, Age and BMI are exceptionally related with the result. Thus, we select these highlights as X and the result as Y. The dataset is then parted utilizing train_test_split with a 80:20 proportion.

```
In [82]: # Step 4: Data Modelling 19STUCHH010130

In [83]: # Logistic Regression Algorithm
         from sklearn.linear_model import LogisticRegression
         logreg = LogisticRegression(random_state = 42)
         logreg.fit(X_train, Y_train)

Out[83]: LogisticRegression(random_state=42)

In [84]: # Plotting a graph for n_neighbors
         from sklearn import metrics
         from sklearn.neighbors import KNeighborsClassifier

         X_axis = list(range(1, 31))
         acc = pd.Series()
         x = range(1,31)

         for i in list(range(1, 31)):
             knn_model = KNeighborsClassifier(n_neighbors = i)
             knn_model.fit(X_train, Y_train)
             prediction = knn_model.predict(X_test)
             acc = acc.append(pd.Series(metrics.accuracy_score(prediction, Y_test)))
         plt.plot(X_axis, acc)
         plt.xticks(x)
         plt.title("Finding best value for n_estimators")
         plt.xlabel("n_estimators")
         plt.ylabel("Accuracy")
         plt.grid()
         plt.show()
         print('Highest value: ',acc.values.max())
```




```
Highest value:  0.7857142857142857
```

In [85]:
```
# 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[85]:
```
KNeighborsClassifier(n_neighbors=24)
```

# Data Modelling:

In [86]:
```
# Support Vector Classifier Algorithm
from sklearn.svm import SVC
svc = SVC(kernel = 'linear', random_state = 42)
svc.fit(X_train, Y_train)
```

Out[86]:
```
SVC(kernel='linear', random_state=42)
```

In [87]:
```
# Naive Bayes Algorithm
from sklearn.naive_bayes import GaussianNB
nb = GaussianNB()
nb.fit(X_train, Y_train)
```

Out[87]:
```
GaussianNB()
```

In [88]:
```
# Decision tree Algorithm
from sklearn.tree import DecisionTreeClassifier
dectree = DecisionTreeClassifier(criterion = 'entropy', random_state = 42)
dectree.fit(X_train, Y_train)
```

Out[88]:
```
DecisionTreeClassifier(criterion='entropy', random_state=42)
```

In [89]:
```
# 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[89]:
```
RandomForestClassifier(criterion='entropy', n_estimators=11, random_state=42)
```

In [90]:
```
# 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)
```

Support Vector Classifier(SVC) is a sort of regulated order model whose goal is to

characterize the information dependent on a maximal edge hyperplane fabricate utilizing support vectors. This hyperplane is a choice limit that orders between different classes. It is constructed utilizing support vectors, which are the anomalies. The hyperplane which has the most noteworthy wiggle room is chosen as the choice limit.

SVCs can order direct just as non-straight information utilizing a part stunt which certainly maps the contribution to high dimensional vector spaces. This part stunt changes over the lower-dimensional component space into higher dimensional element space which is directly detachable. For instance, information in a 2D may not be straightly distinct however when it is changed over into 3D utilizing the piece work it turns out to be directly distinguishable.

SVC has three fundamental boundaries that influence the exhibition of the model which are the portion, gamma, C. The portion boundary connotes the sort of part which can be "Straight" for directly distinguishable information or "rbf", "poly" for non directly divisible information. The Gamma boundary is the part coefficient. As the worth of gamma builds, it attempts to precisely fit the dataset which gives speculation blunder and causes overfitting. C boundary is the expense of misclassification of the model. The high worth of C gives you low inclination and high fluctuation while the low worth of C gives you high predisposition and low change.

**Model Evaluation:**

In [91]:
```
# 5 Model Evaluation
```

In [92]:
```
# 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)
```

In [93]:
```
# 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: 72.07792207792207
K Nearest neighbors: 78.57142857142857
Support Vector Classifier: 73.37662337662337
Naive Bayes: 71.42857142857143
Decision tree: 68.18181818181817
Random Forest: 75.97402597402598
```

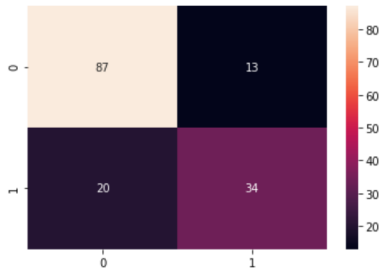
In [94]:
```
#From the above comparison, we can observe that K Nearest neighbors gets the highest accuracy of 78.57 %
```

In [95]:
```
# Confusion matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(Y_test, Y_pred_knn)
cm
```

Out[95]:
```
array([[87, 13],
       [20, 34]], dtype=int64)
```

In [96]:
```
# Heatmap of Confusion matrix
sns.heatmap(pd.DataFrame(cm), annot=True)
```

Out[96]: <AxesSubplot:>




In [97]:
```
# 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
```

We have picked three measurements: accuracy_score, disarray framework and grouping report for assessing our model.

# Chapter - 5
# VSCODE AND PYTHON

**VSCODE:**

Visual Studio Code is a lightweight, however incredible source code supervisor which runs on your work area and is accessible for Windows, macOS and Linux. It accompanies worked in help for JavaScript, TypeScript and Node.js and has a rich biological system of expansions for different dialects (like C++, C#, Java, Python, PHP, Go) and runtimes, (for example, .NET and Unity). Start your excursion with VS Code with these initial

**PYTHON:**

Python is a deciphered, object-arranged, undeniable level programming language with dynamic semantics. Its significant level of implicit information structures, joined with dynamic composing and dynamic restricting, make it exceptionally appealing for Rapid Application Development, just as for use as a pre-arranging or paste language to associate existing parts together. Python's straightforward, simple to learn sentence structure underscores intelligibility and accordingly diminishes the expense of program upkeep. Python upholds modules and bundles, which empowers program measured quality and code reuse. The Python mediator and the broad standard library are accessible in source or parallel structure without charge for every single significant stage, and can be openly dispersed.

# Chapter - 6
# FLASK

Flask is an API of Python that allows us to build up web-applications. It was developed by Armin Ronacher.A Web-Application Framework or Web Framework is the collection of modules and libraries that helps the developer to write applications without writing the low-level codes such as protocols, thread management, etc. Flask is based on the WSGI(Web Server Gateway Interface) toolkit and Jinja2 template engine.

flask is a web system, it's a Python module that allows you to foster web applications without any problem. It has a little and simple to-expand center: it's a microframework that does exclude an ORM (Object Relational Manager) or such highlights.

It has many cool highlights like url steering, format motor. It is a WSGI web application system.

In contrast to the Django system, Flask is very Pythonic. It's not difficult to get everything rolling with Flask, since it doesn't have a colossal expectation to learn and adapt.

On top of that it's extremely expressive, which builds meaningfulness. To make the "Hi World" application, you just need a couple lines of code.

This is a standard code model.

It's a microframework, however that doesn't mean your entire application ought to be inside one single Python record. You can and should utilize many documents for bigger projects, to deal with intricacy.

Miniature implies that the Flask system is basic however extensible. You may have every one of the choices: which information base to utilize, do you need an ORM and so forth, Flask doesn't choose for you.

Flask is quite possibly the most famous web framework, which means it's forward-thinking and current. You can without much of a stretch expand its usefulness. You can increase it for complex applications.

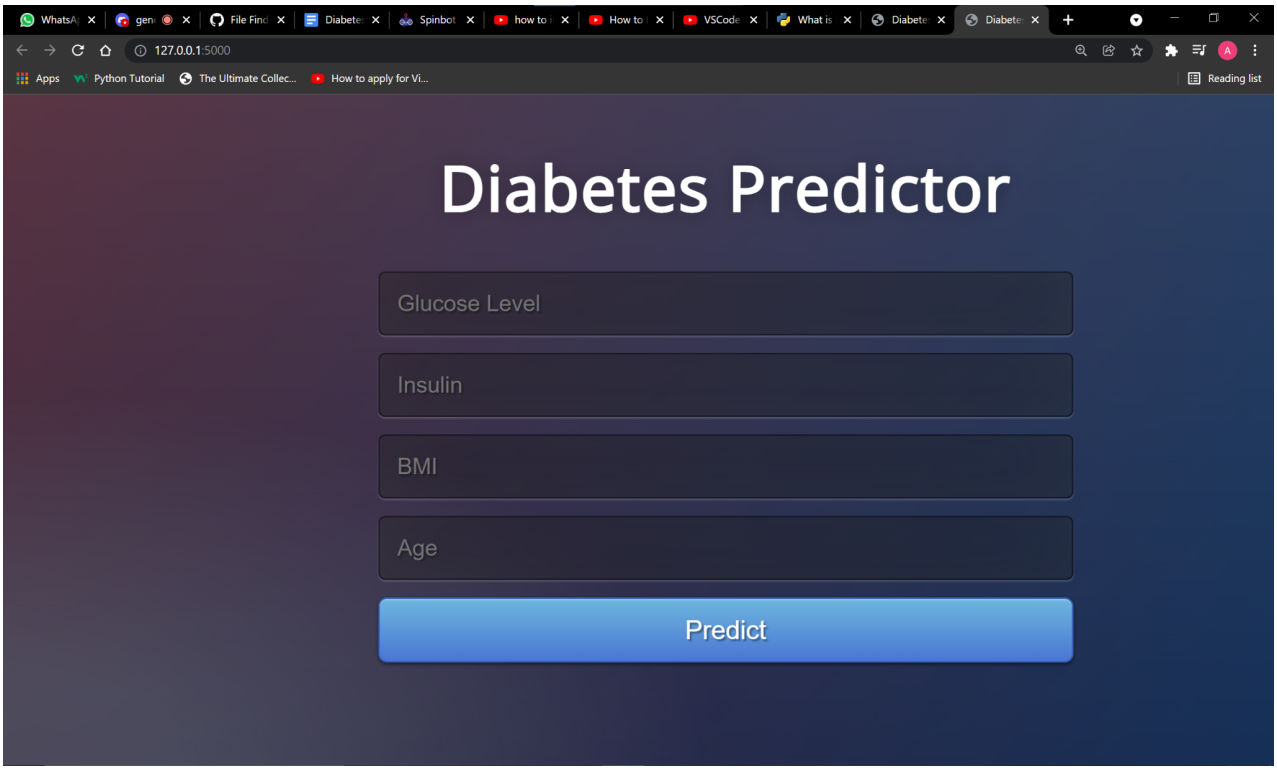
# Chapter - 7
# FLASK CSS AND TEMPLATES

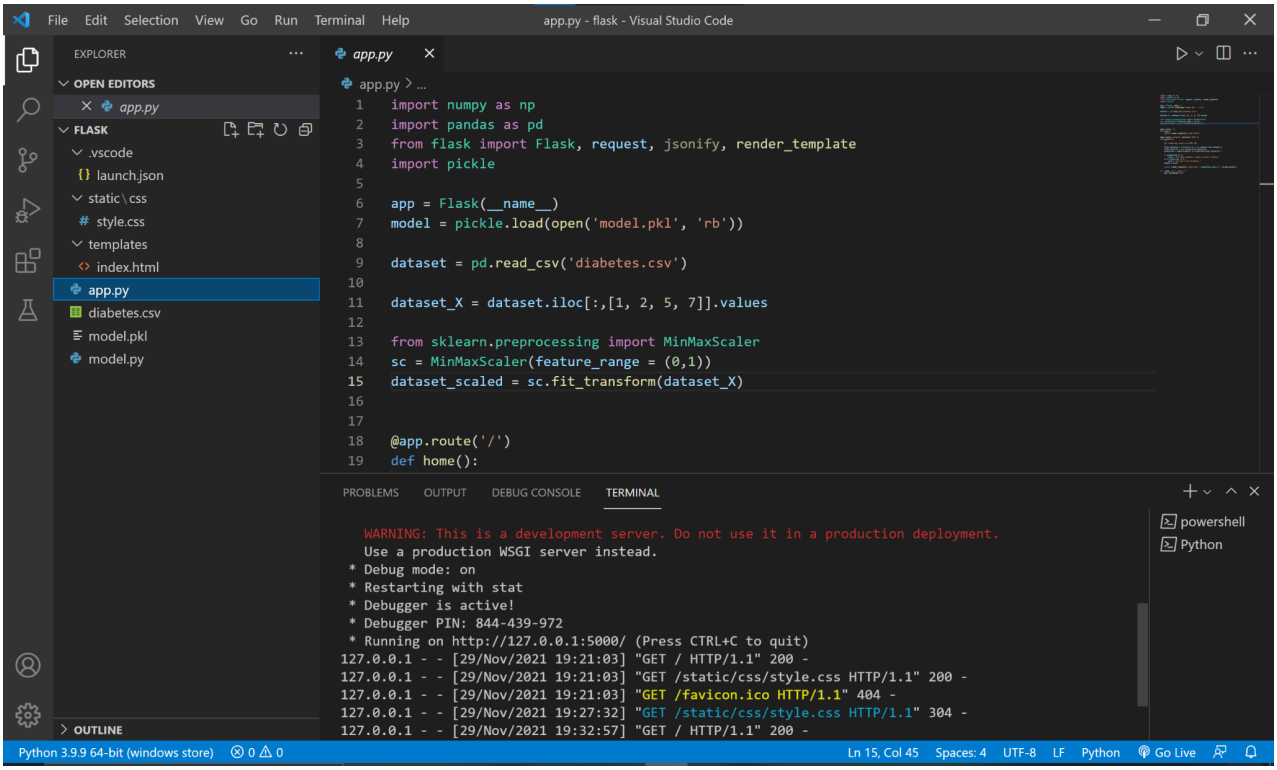
## Model Deployment:

In this progression, we will utilize Flask miniature structure to make a web utilization of our model. Every one of the necessary records can be found in my GitHub storehouse here.

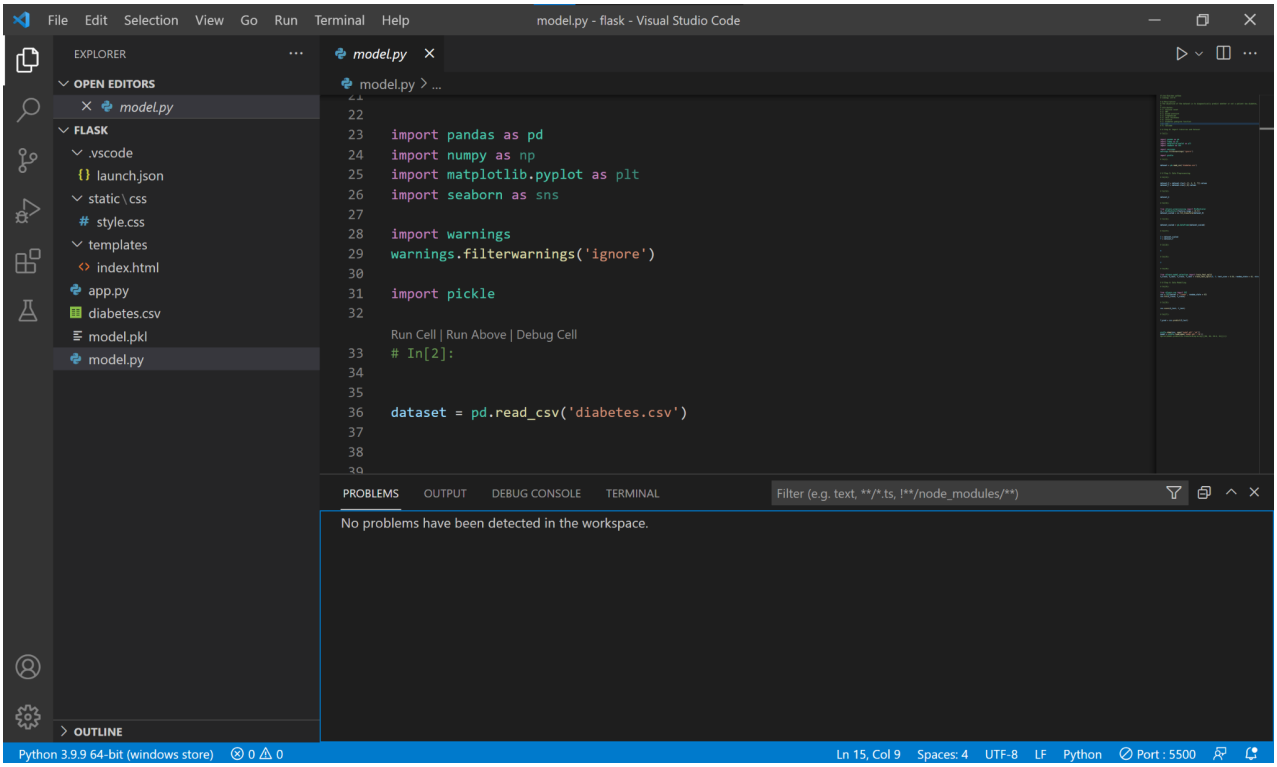
With this progression, we have finished our undertaking from information social event to show organization.

## Flask app.py execution:

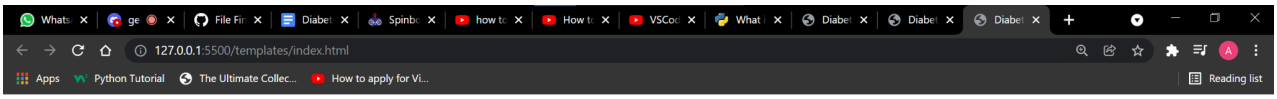





# Model.py execution:



# diabetes.html execution:




# Diabetes Predictor

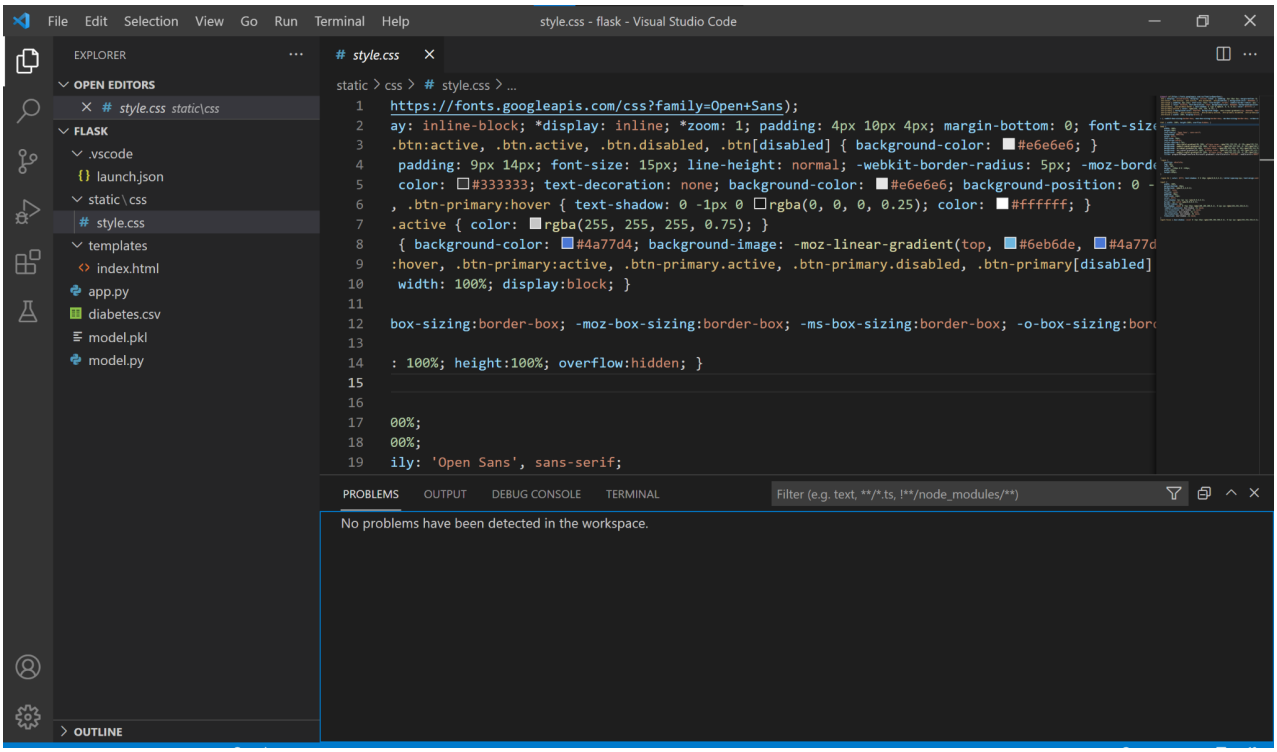
Glucose Level | Insulin | BMI | Age

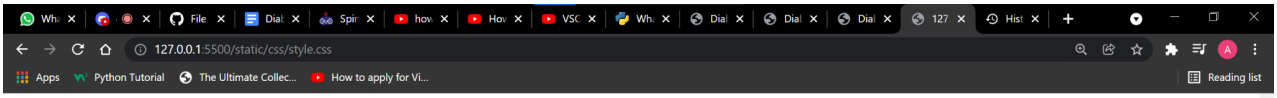
Predict

{{ prediction_text }}

---

# Diabetes.ccs execution:






```
@import url(https://fonts.googleapis.com/css?family=Open+Sans);
.btn { display: inline-block; *display: inline; *zoom: 1; padding: 4px 10px 4px; margin-bottom: 0;
font-size: 13px; line-height: 18px; color: #333333; text-align: center;text-shadow: 0 1px 1px
rgba(255, 255, 255, 0.75); vertical-align: middle; background-color: #f5f5f5; background-image: -
moz-linear-gradient(top, #ffffff, #e6e6e6); background-image: -ms-linear-gradient(top, #ffffff,
#e6e6e6); background-image: -webkit-gradient(linear, 0 0, 0 100%, from(#ffffff), to(#e6e6e6));
background-image: -webkit-linear-gradient(top, #ffffff, #e6e6e6); background-image: -o-linear-
gradient(top, #ffffff, #e6e6e6); background-image: linear-gradient(top, #ffffff, #e6e6e6);
background-repeat: repeat-x; filter:
progid:dximagetransform.microsoft.gradient(startColorstr=#ffffff, endColorstr=#e6e6e6,
GradientType=0); border-color: #e6e6e6 #e6e6e6 #e6e6e6; border-color: rgba(0, 0, 0, 0.1) rgba(0,
0, 0, 0.1) rgba(0, 0, 0, 0.25); border: 1px solid #e6e6e6; -webkit-border-radius: 4px; -moz-
border-radius: 4px; border-radius: 4px; -webkit-box-shadow: inset 0 1px 0 rgba(255, 255, 255,
0.2), 0 1px 2px rgba(0, 0, 0, 0.05); -moz-box-shadow: inset 0 1px 0 rgba(255, 255, 255, 0.2), 0
1px 2px rgba(0, 0, 0, 0.05); box-shadow: inset 0 1px 0 rgba(255, 255, 255, 0.2), 0 1px 2px rgba(0,
0, 0, 0.05); cursor: pointer; *margin-left: .3em; }
.btn:hover, .btn:active, .btn.active, .btn.disabled, .btn[disabled] { background-color: #e6e6e6; }
.btn-large { padding: 9px 14px; font-size: 15px; line-height: normal; -webkit-border-radius: 5px;
-moz-border-radius: 5px; border-radius: 5px; }
.btn:hover { color: #333333; text-decoration: none; background-color: #e6e6e6; background-
position: 0 -15px; -webkit-transition: background-position 0.1s linear; -moz-transition:
background-position 0.1s linear; -ms-transition: background-position 0.1s linear; -o-transition:
background-position 0.1s linear; transition: background-position 0.1s linear; }
.btn-primary, .btn-primary:hover { text-shadow: 0 -1px 0 rgba(0, 0, 0, 0.25); color: #ffffff; }
.btn-primary.active { color: rgba(255, 255, 255, 0.75); }
```

# Chapter - 8
# CONCLUSION

In conclusion I was able to build a useful website for prediction of diabetes and learnt about data modeling, Research about data aspects,creating diabetes prediction website,using flask and other applications and html, css languages and known about.

A similar model can be built for a variety of diseases like breast cancer, malaria, etc in much detail and can be highly reliable once it gets high enough accuracy.

We can create new variables for blood pressure in a particular range, glucose levels in a particular range, and so on.

We can read a bit about what metrics doctors rely on the most to diagnose a diabetic patient, and create new features accordingly.

# REFERENCES:


1)https://towardsdatascience.com/end-to-end-data-science-example-predicting-diabetes-with-logistic-regression-db9bc88b4d16

2)https://medium.com/analytics-vidhya/building-a-diabetes-predictor-4702b99bc7e4

3)https://towardsdatascience.com/build-deploy-diabetes-prediction-app-using-flask-ml-and-heroku-2de07cbd902d

4)https://www.python.org/

5)https://www.questionpro.com/blog/data-collection/