LAB REPORT

Submitted by

Syed Imam Ali [RA2011003010944]

Under the Guidance of

Ms. S. Nagadevi

Assistant Professor, Department of Computing Technologies

In partial satisfaction of the requirements for the degree of

BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE ENGINEERING

SCHOOL OF COMPUTING

COLLEGE OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR - 603203

JUNE 2022

SRM INSTITUTION OF SCIENCE AND TECHNOLOGY
KATTANKULATHUR-603203

BONAFIDE CERTIFICATE

Certified that this lab report titled **"PyDiabetes"** is the bonafide work done by Syed Imam Ali (RA2011003010944) who carried out the lab exercises under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other work.

SIGNATURE

Ms. S. Nagadevi

SEPM – Course Faculty

Assistant Professor

Department of Computing Technologies

TABLE OF CONTENTS

ABSTRACT

Diabetes is a chronic disease with the potential to cause a worldwide health care crisis. According to the International Diabetes Federation 382 million people are living with diabetes across the whole world. By 2035, this will be doubled as 592 million. Diabetes is a disease caused due to the increase level of blood glucose. This high blood glucose produces the symptoms of frequent urination, increased thirst, and increased hunger. Diabetes is a one of the leading cause of blindness, kidney failure, amputations, heart failure and stroke. When we eat, our body turns food into sugars, or glucose. At that point, our pancreas is supposed to release insulin. Insulin serves as a key to open our cells, to allow the glucose to enter and allow us to use the glucose for energy. But with diabetes, this system does not work. Type 1 and type 2 diabetes are the most common forms of the disease, but there are also other kinds, such as gestational diabetes, which occurs during pregnancy, as well as other forms. Machine learning is an emerging scientific field in data science dealing with the ways in which machines learn from experience. The aim of this project is to develop a system which can perform early prediction of diabetes for a patient with a higher accuracy by combining the results of different machine learning techniques. The algorithm Naïve Bayes is used. The accuracy of the model using each of the algorithms is calculated. Then the one with a good accuracy is taken as the model for predicting the diabetes.

PROBLEM STATEMENT

To predict that whether the patient has diabetes or not on the basis of the features we will provide to our machine learning model, and for that, we will be using the famous Pima Indians Diabetes Database.

Project Title:

PyDiabetes

Project Description:

The PIMA Indians dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases. The objective is to predict based on diagnostic measurements whether a patient has diabetes.

Several constraints were placed on the selection of these instances from a larger database. In particular, all patients here are females at least 21 years old of Pima Indian heritage.

- Pregnancies: Number of times pregnant
- Glucose: Plasma glucose concentration a 2 hours in an oral glucose tolerance test
- BloodPressure: Diastolic blood pressure (mm Hg)
- SkinThickness: Triceps skin fold thickness (mm)
- Insulin: 2-Hour serum insulin (mu U/ml)
- BMI: Body mass index (weight in kg/(height in m)^2)
- DiabetesPedigreeFunction: Diabetes pedigree function
- Age: Age (years)
- Outcome: Class variable (0 or 1)

STAKEHOLDERS & PROCESS MODELS

1.1 Project Methodology:

Agile Methodology is a people-focused, results-focused approach to software development that respects our rapidly changing world.

It's centred around adaptive planning, self-organization, and short delivery times.

It's flexible, fast, and aims for continuous improvements in quality, to put in simple terms, Agile helps teams in delivering value to customers quickly and effortlessly.

Thus, developing such software management project, Agile methodology brings out the most effective growth out of it.

1.2 Stakeholder Identification:

Internal stakeholders:

The internal stakeholders include the team members, managers, executives who are all internally related.

Project Role	Responsibilities	Team members assigned to
Project Manager	Plan and develop project ideas	Syed Imam Ali
Technical Lead	Analyzing user needs and then finding applications to serve them	Syed Imam Ali
Business Analyst	Budgeting and forecasting	Syed Imam Ali
Developer	Developing the machine learning model	Syed Imam Ali
Tester	Document bugs and troubleshoot issues	Syed Imam Ali

External stakeholders:

Project Role	Responsibilities	Team members assigned to
Dataset Provider	Providing datasets	Syed Imam Ali

2.Stakeholder Management

2.1 Interest and Influence matrix

Interest	Influence
High	High
Low	Low
Low	High
High	Low

2.2 STAKEHOLDER INTEREST, INFLUENCE, PRIORITY IDENTIFICATION

Stakeholder	Responsibility	Interest	Influence	Estimated Priority
Owner		High	Low	High
Sponsor		Low	Low	Low
Team members		High	High	Low
Project Manager		High	High	High
Investors		High	High	High
Resource Manager		Low	Low	High
Suppliers		Low	High	Low
End Users		High	High	Low

IDENTIFYING REQUIREMENTS

User Requirements :

Some Hospital or some private organisation will be incorporating the model in their app and website, to predict the onset of diabetes.

System Requirements :

- Python
- Numpy
- Pandas
- Matplotlib
- Jupyter notebook
- Prima Indian Dataset

Functional Requirements :

It needs to predict the onset of diabetes accurately. Otherwise, some patients who might have diabetes or can have diabetes in the future.

Accuracy Score : It should be close to 1 on 0 to 1 scale, where 1 being the perfect model.

Precision Score : It should be close to 1 on 0 to 1 scale, where 1 being the perfect model.

Recall Score : It should be close to 1 on 0 to 1 scale, where 1 being the perfect model.

F1 Score : It should be close to 1 on 0 to 1 scale, where 1 being the perfect model.

Non-Functional Requirements :

The dataset have to perfectly cleaned.
We would expect that simply changing the name of the subject doesn't affect the model predictions.

PROJECT PLAN & EFFORT

1. Project Management Plan

Describe the key issues driving the project. [Min 3 Focus Areas]

Focus Area	Details
Integration Management	Governance Framework Project Team Structure Roles & Responsibilities of Team Change Management (Change Control, Issue Management) Project Closure
Scope Management	Scope Statement Requirement Management (Gathering, Control, Assumption, Constraint Stakeholder) Define Deliverable Requirement Change Control Activities and Sub-Tasks
Schedule Management	Define Milestones Schedule Control
Cost Management	Estimate Effort Assign Team Budget Control
Quality Management	Quality Assurance: Quality assurance will be managed including governance, roles and responsibilities, tools and techniques and reporting Quality Control: Specify the mechanisms to be used to measure and control the quality of the work products
Resource Management	Estimate and Manage the need People: People & Skills Required Finance: Budget Required Physical: Facilities, IT Infrastructure
Stakeholder	Identifying, Analyzing, Engaging Stakeholders

Communication Management	**Determine communication requirements, roles and responsibilities, tools and techniques. [Type of Communication, Schedule, Mechanism Recipient]**
Risk Management	**Identifying, analysing, and prioritizing project risks**
Procurement Management	**Adhering to organization procurement process**

2. Estimation

2.1.Effort and Cost Estimation

Activity Description	Sub-Task	Sub-Task Description	Effort (in hours)	Cost in INR
Train the model	Data Preprocessing	Task for cleaning the data, which contains noises, and missing values	1	500
	Getting the hypothesis	Feeding the data to train the model	20	10000
	Visualising the output	Plotting the output	1	500
Test the Model	Test the hypothesis	Seeing how well it performs on the test data set	3	1500
	Adjusting	Making necessary adjustment to fit the test dataset also	7	3500

Effort (hr)	Cost (INR)
1	500

2.2 Infrastructure/Resource Cost [CapEx]

Infrastructure Requirement	Qty	Cost per qty	Cost per item
Dataset	1	1000	0.5
Jupyter Notebook	1	0	0

3. Maintenance and Support Cost [OpEx]

Category	Details	Qty	Cost per qty per annum	Cost per item
License	Dataset	1	1000	1000
Infrastructures	Google Colab and Jupyter Notebook	1	0	0

3. Project Team Formation

3.1.Identification Team members

Name	Role	Responsibilities
Syed Imam Ali	Key Business User (Product Owner)	Provide clear business and user requirements
Syed Imam Ali	Project Manager	Manage the project
Syed Imam Ali	Business Analyst	Discuss and Document Requirements
Syed Imam Ali	Technical Lead	Design and create the model
Syed Imam Ali	Tester	Define Test Cases and Perform Testing

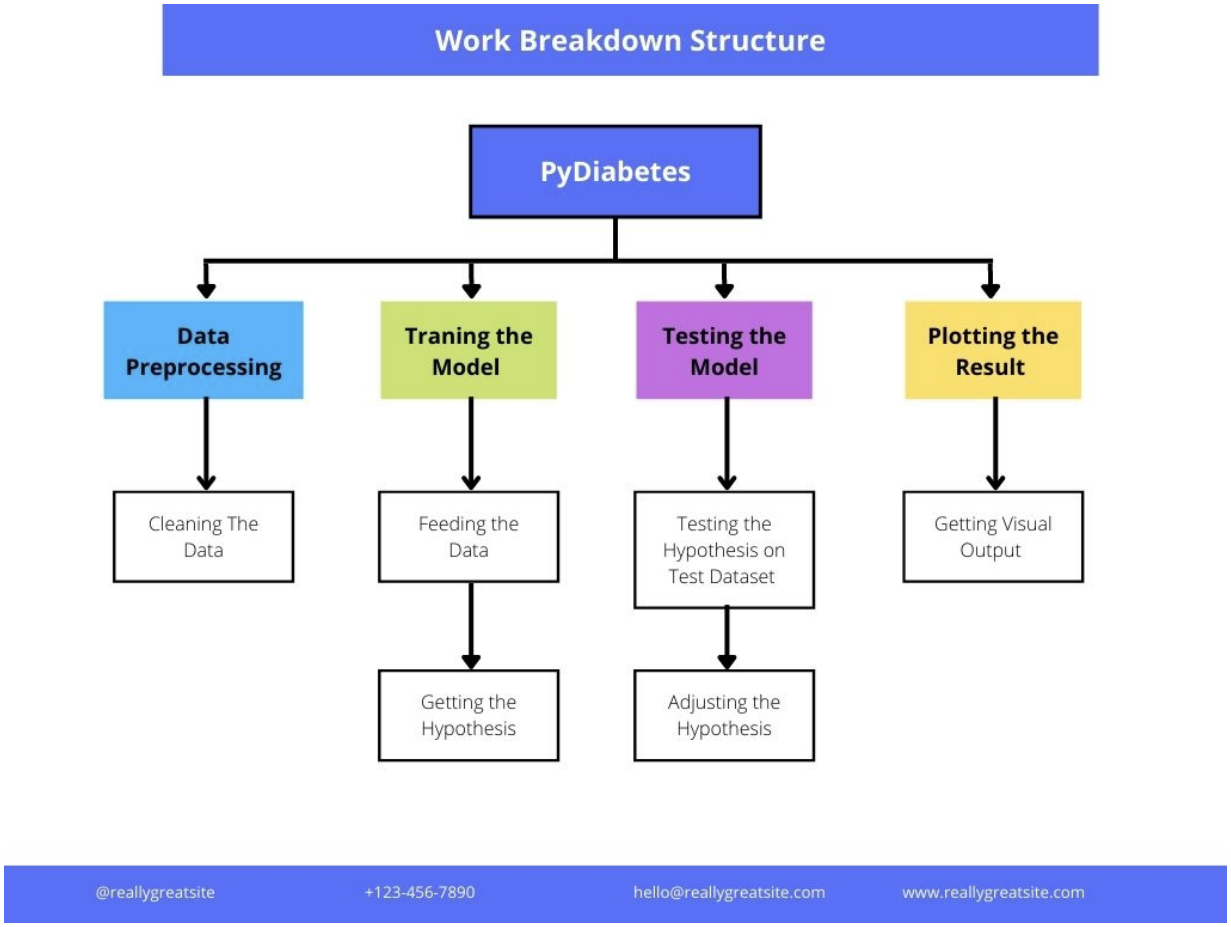
2. Responsibility Assignment Matrix

RACI Matrix	Team Members			
Activity	**Syed Imam Ali (BA)**	**Syed Imam Ali (Developer)**	**Syed Imam Ali (Project Manager)**	**Syed Imam Ali**
User Requirement Documentation	**A/R/C/I**	**A/R/C/I**	**A/R/C/I**	**A/R/C/I**
Developing the model	**A/R/C/I**	**A/R/C/I**	**A/R/C/I**	**A/R/C/I**
Project Management	**A/R/C/I**	**A/R/C/I**	**A/R/C/I**	**A/R/C/I**

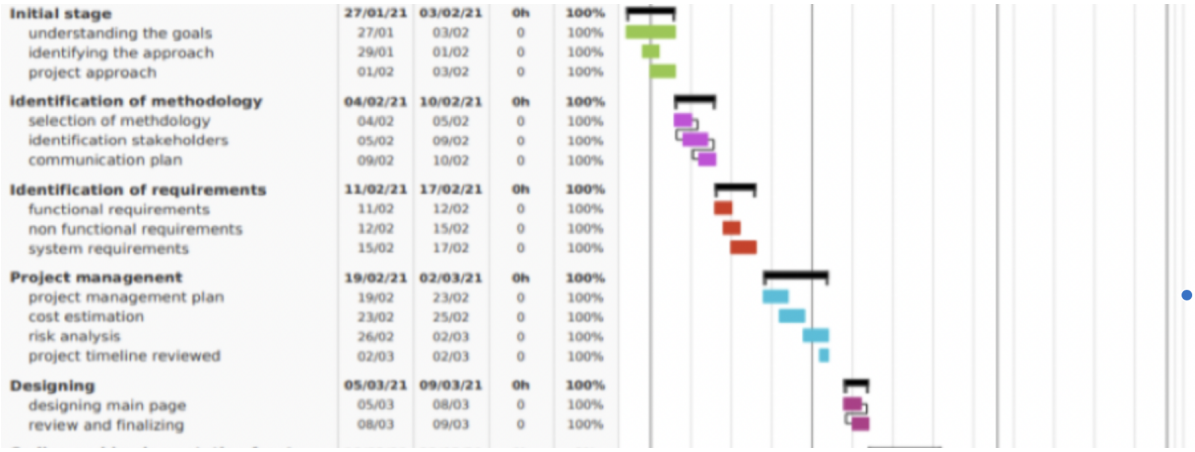
A	**Accountable**
R	**Responsible**
C	**Consult**
I	**Inform**

WORK BREAKDOWN STRUCTURE & RISK ANALYSIS

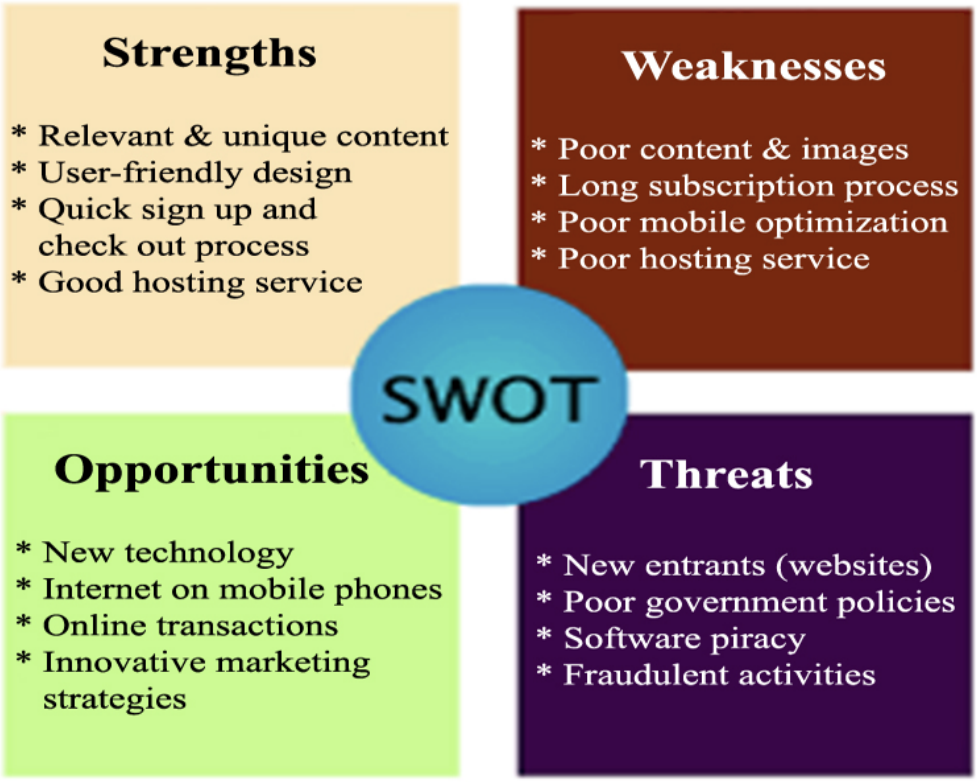
WBS

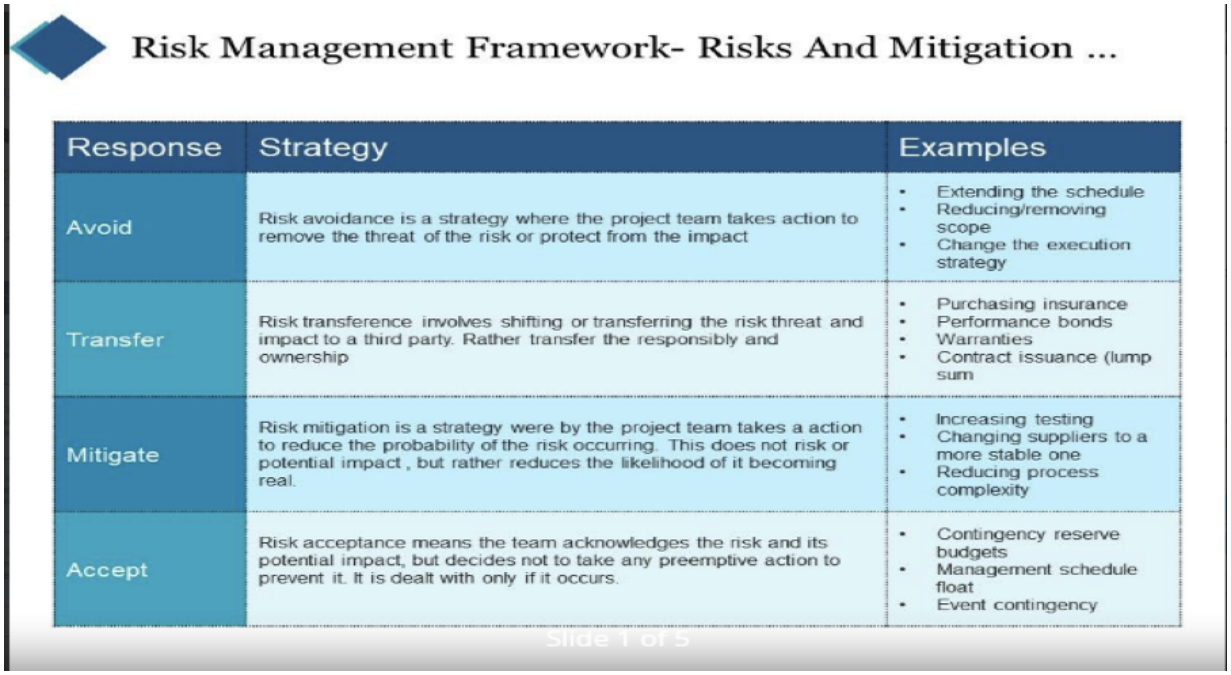


TIMELINE - GANNT CHART



Initial stage	**27/01/21**	**03/02/21**	**0h**	**100%**
understanding the goals	27/01	03/02	0	100%
identifying the approach	29/01	01/02	0	100%
project approach	01/02	03/02	0	100%
identification of methodology	**04/02/21**	**10/02/21**	**0h**	**100%**
selection of methdology	04/02	05/02	0	100%
identification stakeholders	05/02	09/02	0	100%
communication plan	09/02	10/02	0	100%
identification of requirements	**11/02/21**	**17/02/21**	**0h**	**100%**
functional requirements	11/02	12/02	0	100%
non functional requirements	12/02	15/02	0	100%
system requirements	15/02	17/02	0	100%
Project managenent	**19/02/21**	**02/03/21**	**0h**	**100%**
project management plan	19/02	23/02	0	100%
cost estimation	23/02	25/02	0	100%
risk analysis	26/02	02/03	0	100%
project timeline reviewed	02/03	02/03	0	100%
Designing	**05/03/21**	**09/03/21**	**0h**	**100%**
designing main page	05/03	08/03	0	100%
review and finalizing	08/03	09/03	0	100%





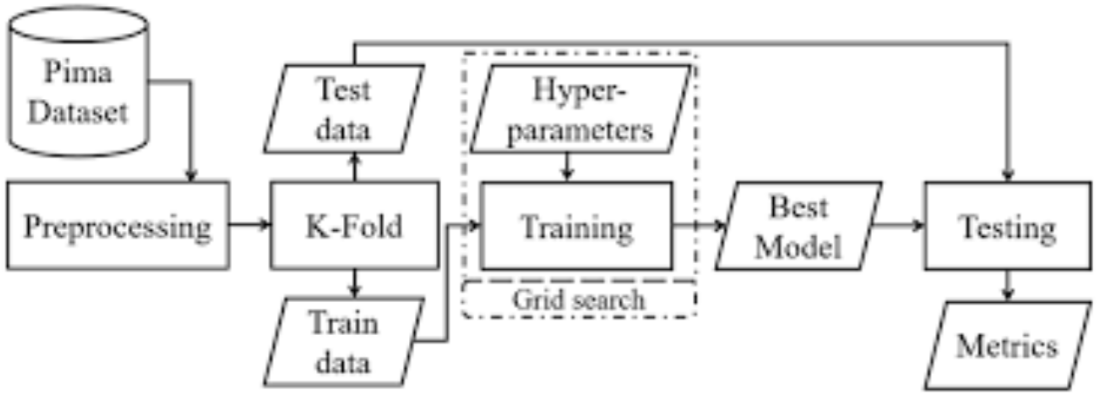
Risk Management Framework- Risks And Mitigation ...

Response	Strategy	Examples
Avoid	Risk avoidance is a strategy where the project team takes action to remove the threat of the risk or protect from the impact	• Extending the schedule • Reducing/removing scope • Change the execution strategy
Transfer	Risk transference involves shifting or transferring the risk threat and impact to a third party. Rather transfer the responsibly and ownership	• Purchasing insurance • Performance bonds • Warranties • Contract issuance (lump sum
Mitigate	Risk mitigation is a strategy were by the project team takes a action to reduce the probability of the risk occurring. This does not risk or potential impact , but rather reduces the likelihood of it becoming real.	• Increasing testing • Changing suppliers to a more stable one • Reducing process complexity
Accept	Risk acceptance means the team acknowledges the risk and its potential impact, but decides not to take any preemptive action to prevent it. It is dealt with only if it occurs.	• Contingency reserve budgets • Management schedule float • Event contingency

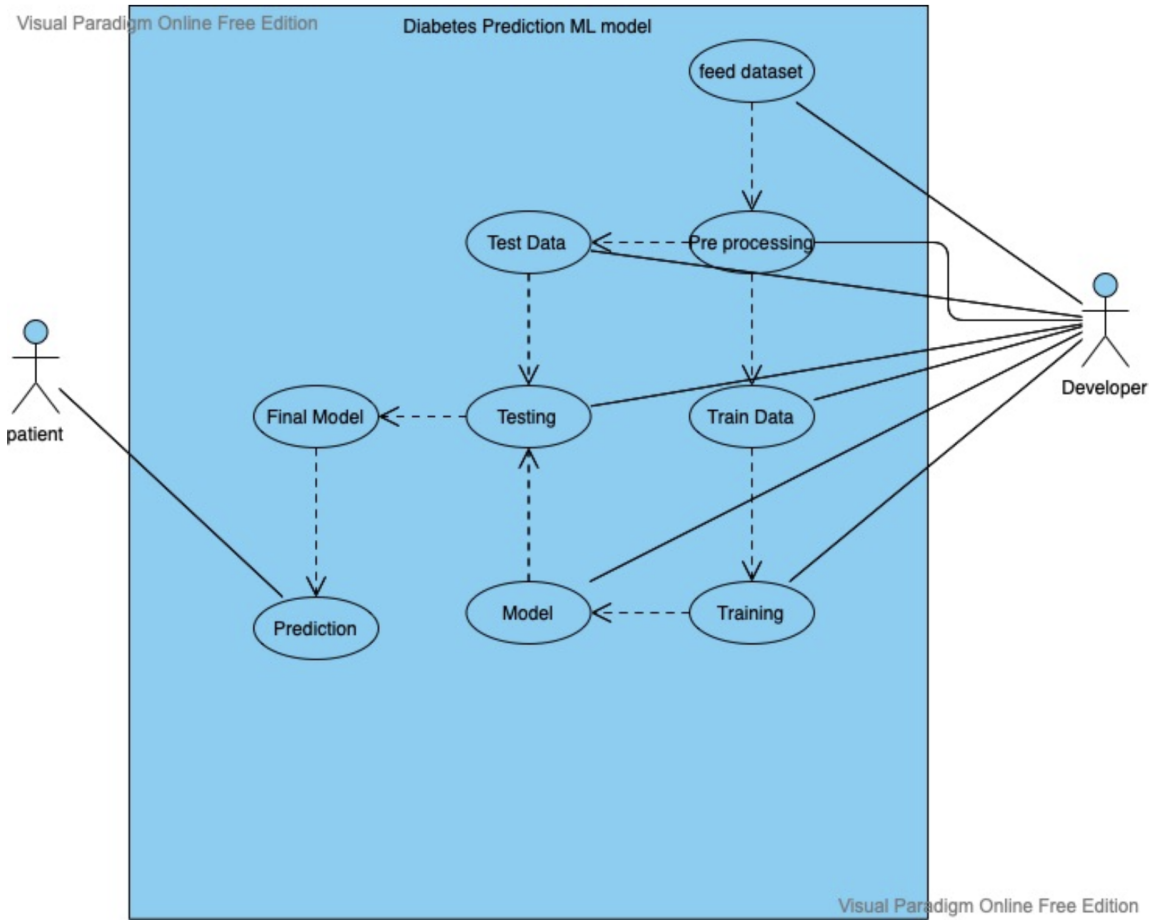
Slide 1 of 5

SYSTEM ARCHITECTURE, USE CASE & CLASS DIAGRAM

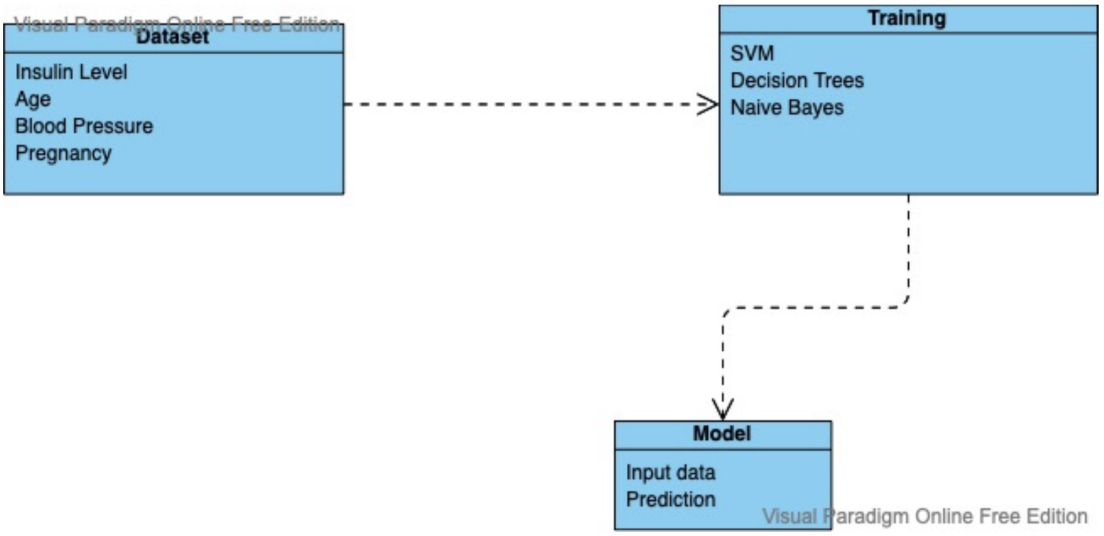
System Architecture :



Use Case Diagram :

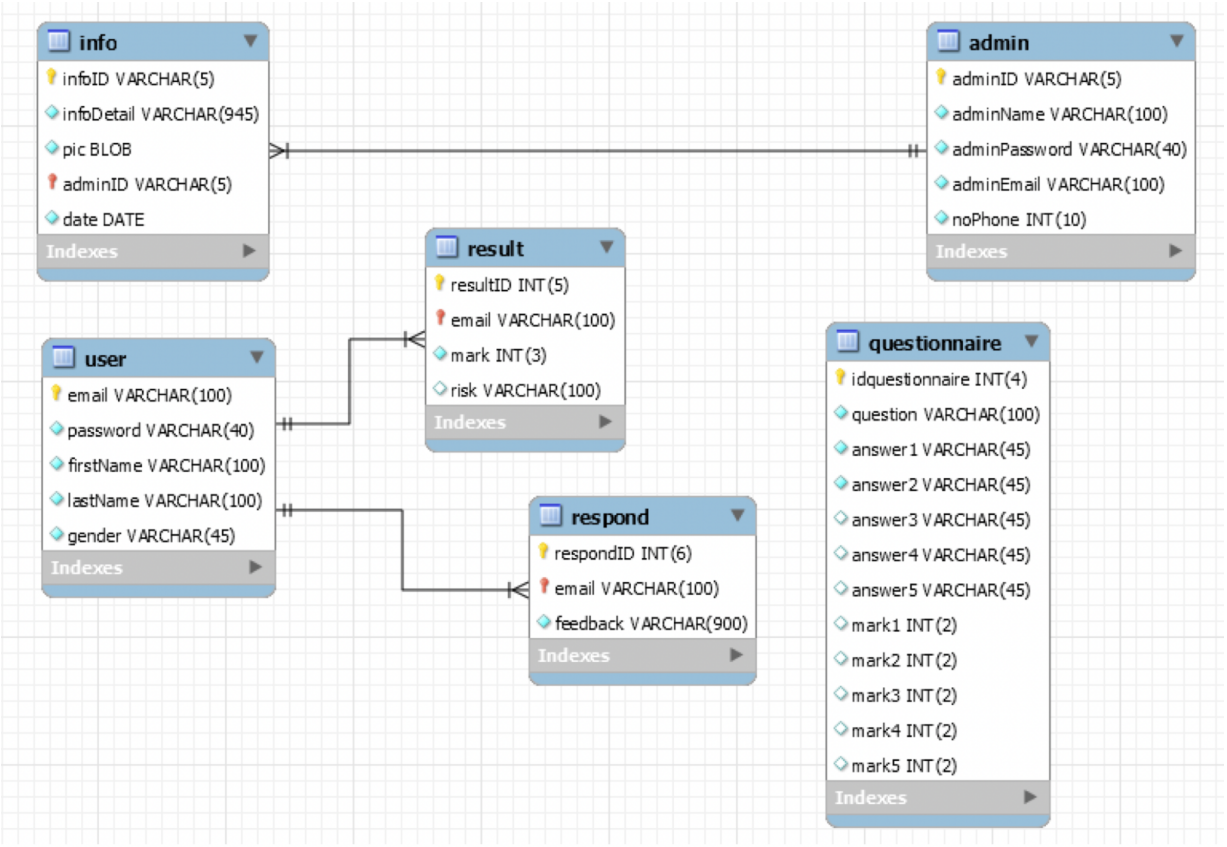


Class Daigram :



ENTITY RELATIONSHIP DIAGRAM

ER Diagram



DATA FLOW DIAGRAM

DFD

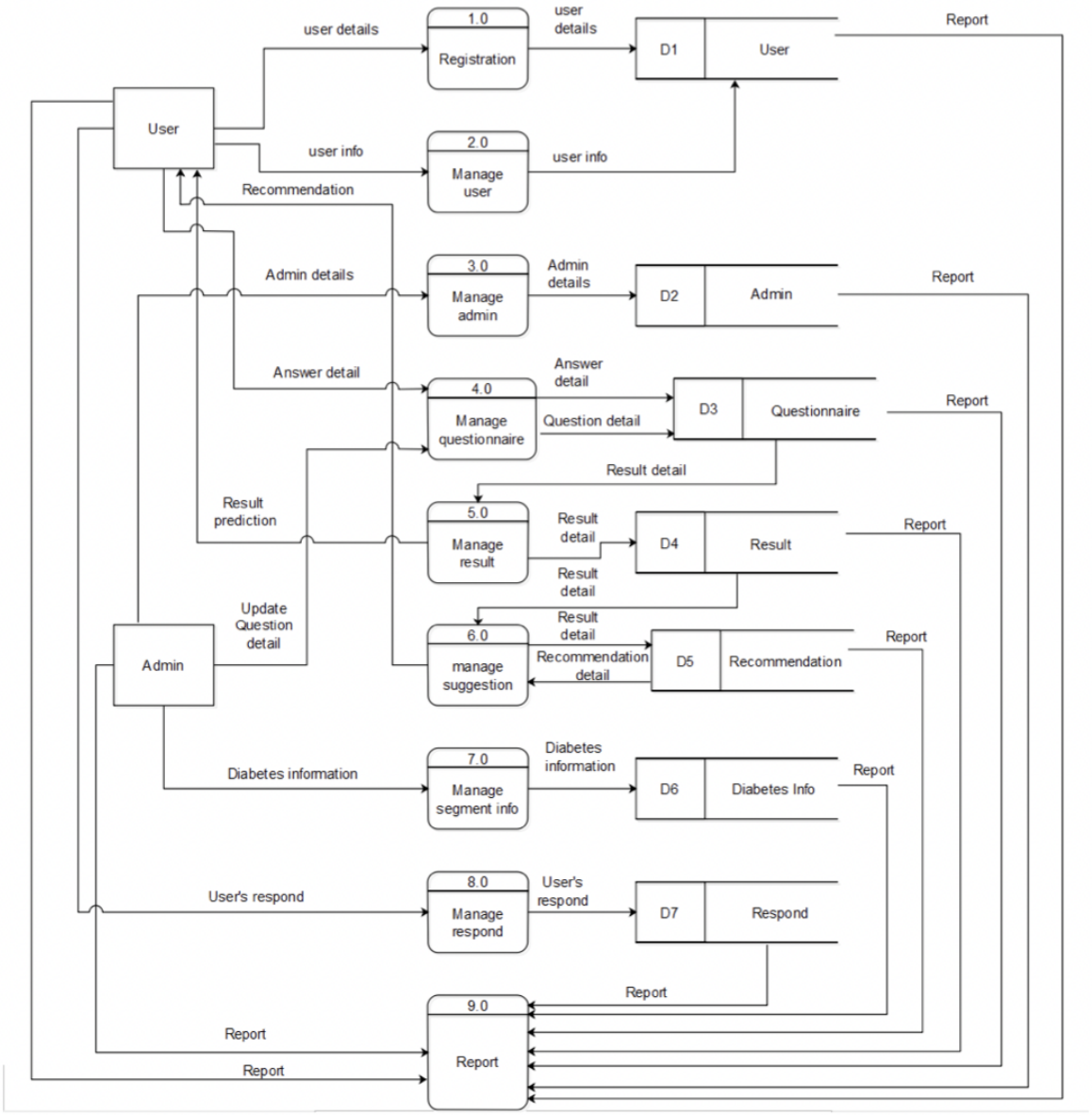


Figure 3.8 Data Flow Diagram Level 0 for Diabetes Prediction System

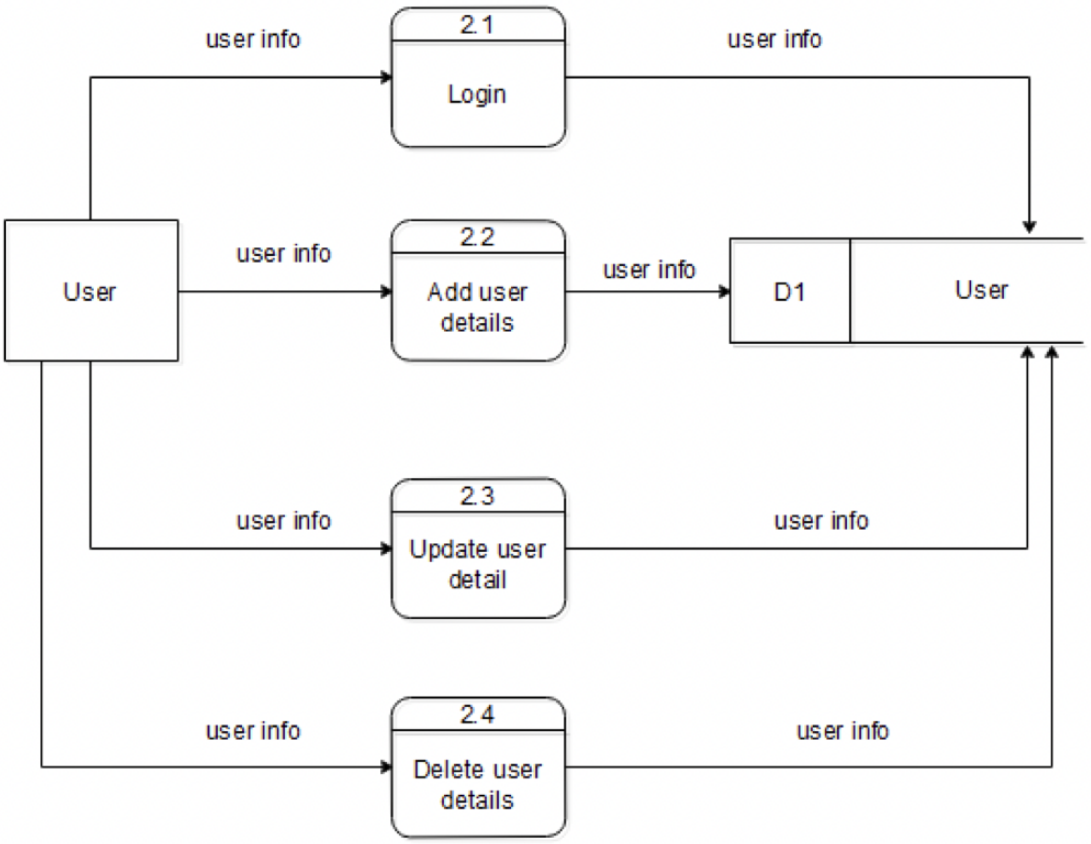


Figure 3.9.1: Data Flow Diagram Level 1 for Manage User

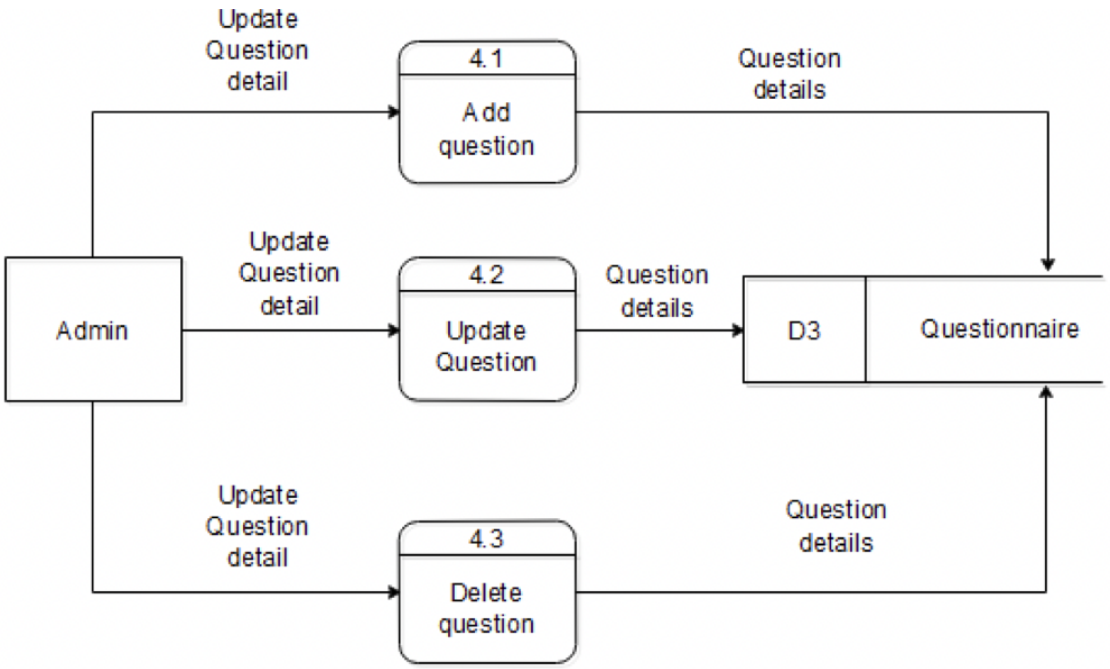


Figure 3.9.2: Data Flow Diagram Level 1 for Manage Prediction

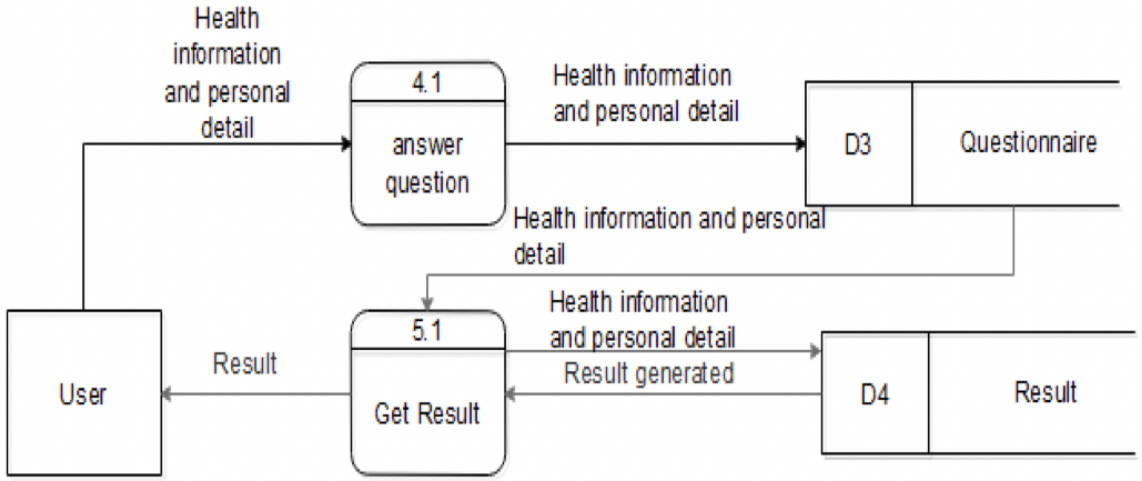


Figure 3.9.3: Data Flow Diagram Level 1 for Manage Result

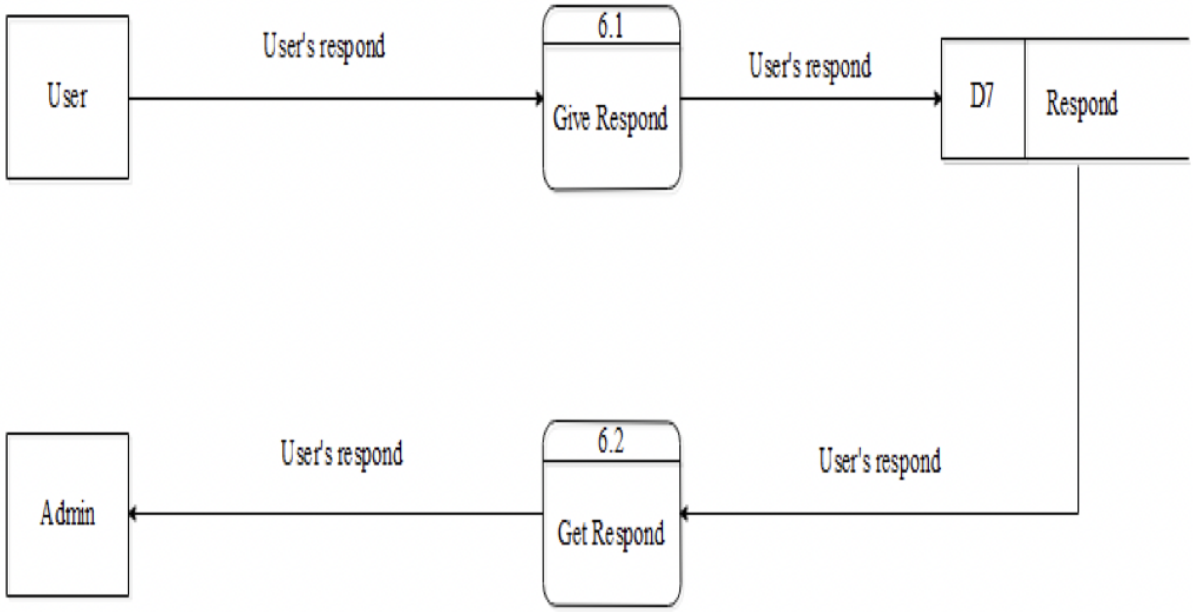


Figure 3.9.4: Data Flow Diagram Level 1 for Manage Respond

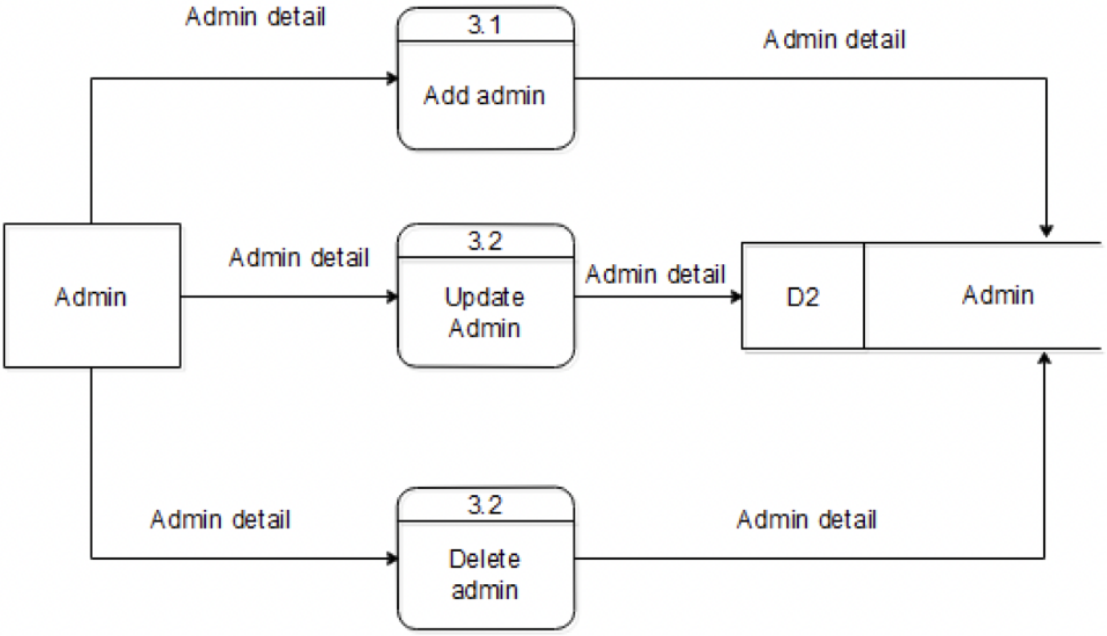


Figure 3.9.5: Data Flow Diagram Level 1 for Manage Admin

SEQUENCE & COLLABORATION DIAGRAM

Sequence Diagram

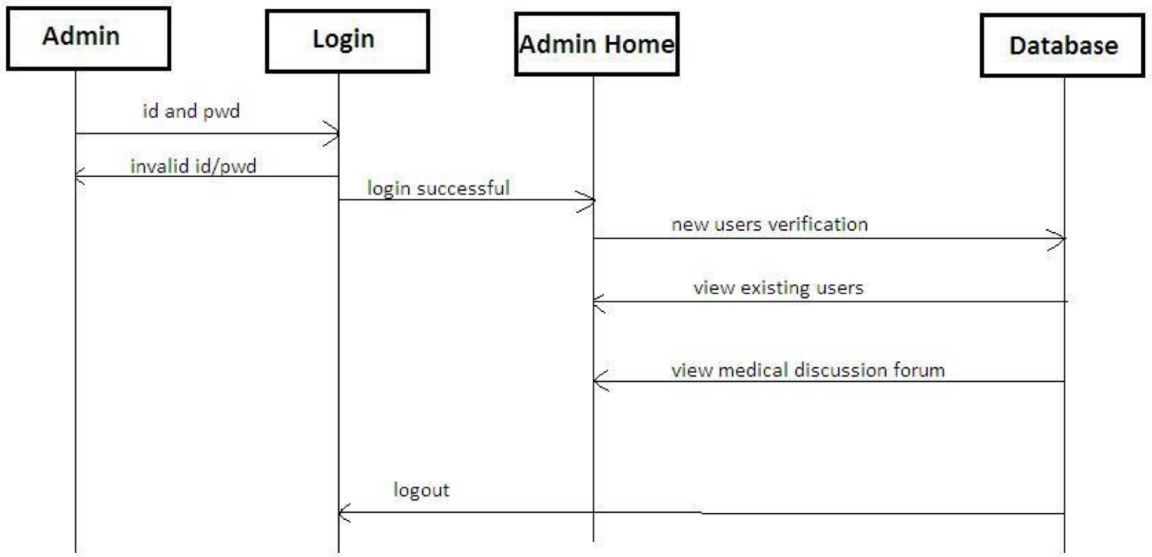


Figure 6: Sequence Diagram for Admin.

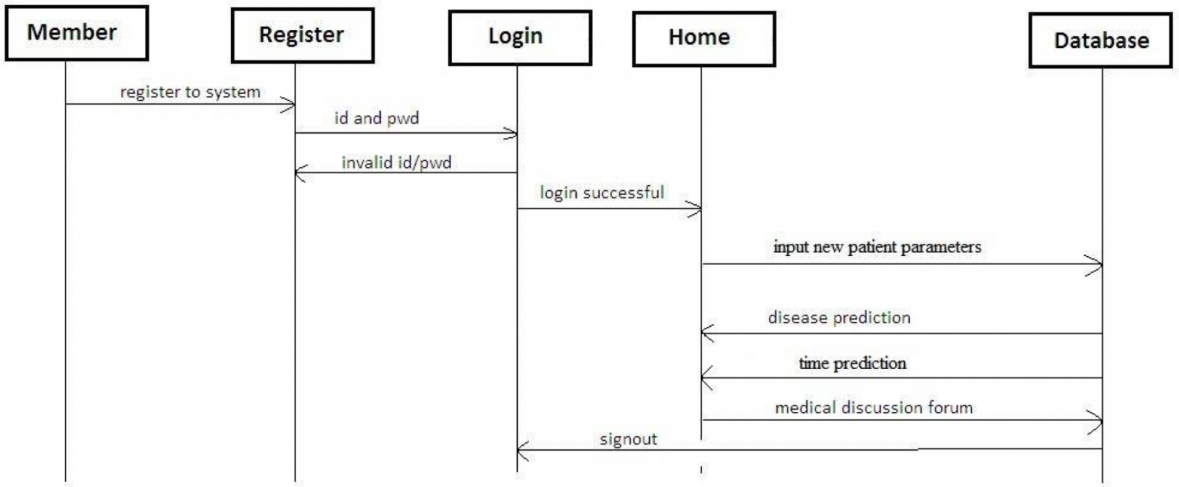
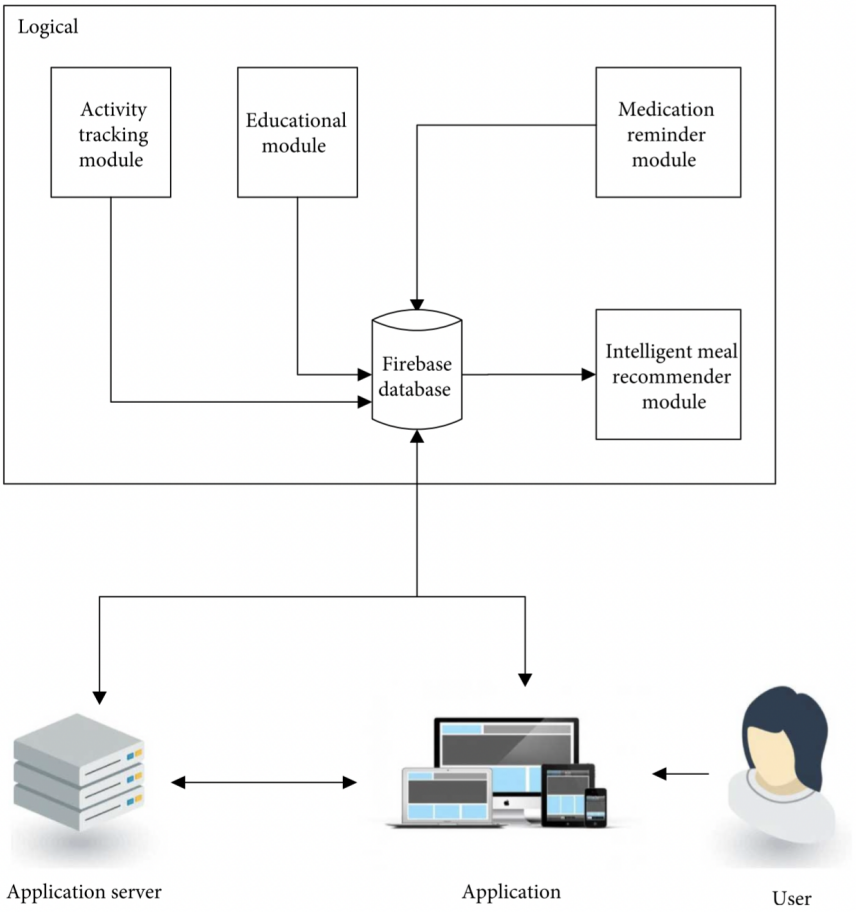


Figure 7: Sequence Diagram for Member/Doctor.

Collaboration Diagram :

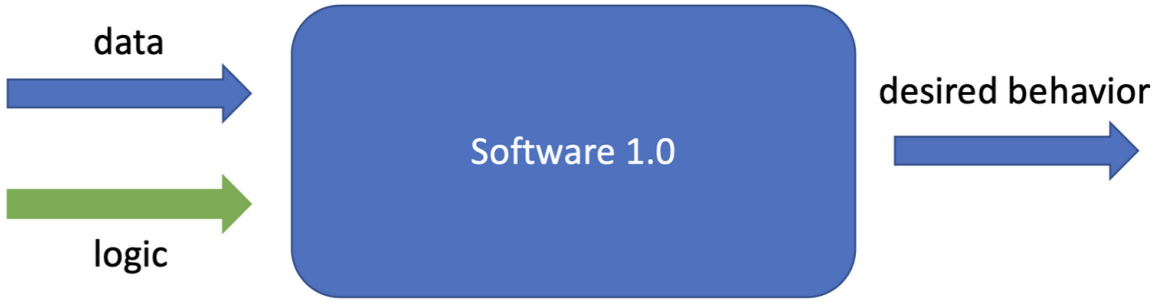


DEVELOPMENT OF TESTING FRAMEWORK/USER INTERFACE

Executive Summary

Scope -

In traditional software systems, humans write the logic which interacts with data to produce a desired behavior. Our software tests help ensure that this written logic aligns with the actual expected behavior.



However, in machine learning systems, humans provide desired behavior as examples during training and the model optimization process produces the logic of the system. How do we ensure this learned logic is going to consistently produce our desired behavior?

Objective and Approach:

Let's start by looking at the best practices for testing traditional software systems and developing high-quality software.

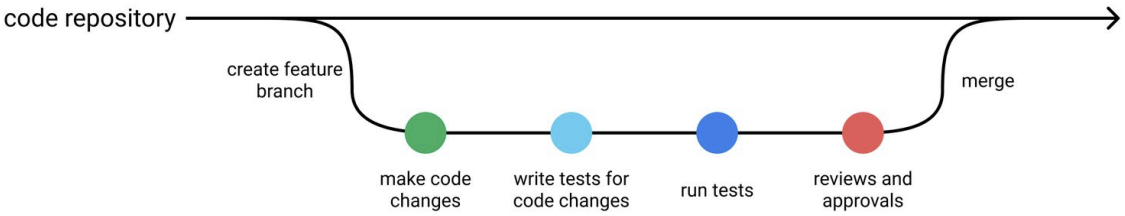
A typical software testing suite will include:

- unit tests which operate on atomic pieces of the codebase and can be run quickly during

development,
- regression tests replicate bugs that we've previously encountered and fixed,
- integration tests which are typically longer-running tests that observe higher-level behaviors that leverage multiple components in the codebase,

and follow conventions such as:

- don't merge code unless all tests are passing,
- always write tests for newly introduced logic when contributing code,
- when contributing a bug fix, be sure to write a test to capture the bug and prevent future regressions.



A typical workflow for software development.

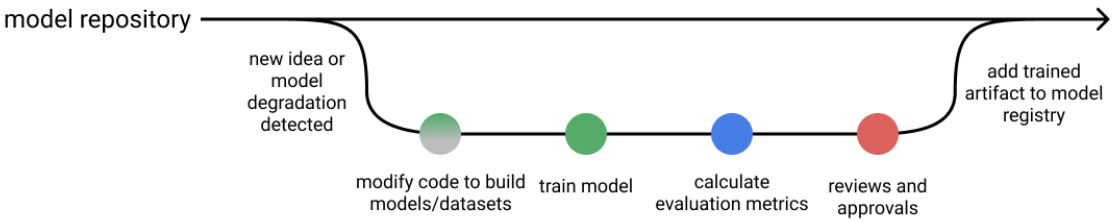
When we run our testing suite against the new code, we'll get a report of the specific behaviors that we've written tests around and verify that our code changes don't affect the expected behavior of the system. If a test fails, we'll know which specific behavior is no longer aligned with our expected output. We can also look at this testing report to get an understanding of how extensive our tests are by looking at metrics such as code coverage.

Let's contrast this with a typical workflow for developing machine learning systems. After training a new model, we'll typically produce an evaluation report including:

- performance of an established metric on a validation dataset,
- plots such as precision-recall curves,
- operational statistics such as inference speed,
- examples where the model was most confidently incorrect,

and follow conventions such as:

- save all of the hyper-parameters used to train the model,
- only promote models which offer an improvement over the existing model (or baseline) when evaluated on the same dataset.



A typical workflow for model development.

Test Plan

Scope of Testing

Functional: Are all modules covered? Any exception for any modules ? Does automation cover all functional test cases or Regression – Critical Path Test Cases ?

Accuracy Score : It should be close to 1 on 0 to 1 scale, where 1 being the perfect model.

Precision Score : It should be close to 1 on 0 to 1 scale, where 1 being the perfect model.

Recall Score : It should be close to 1 on 0 to 1 scale, where 1 being the perfect model.

F1 Score : It should be close to 1 on 0 to 1 scale, where 1 being the perfect model.

Non-Functional: Are all NFR (Non-Functional Requirements) covered?

Pre-train tests

There's some tests that we can run without needing trained parameters. These tests include:

- check the shape of your model output and ensure it aligns with the labels in your dataset
- check the output ranges and ensure it aligns with our expectations (eg. the output of a classification model should be a distribution with class probabilities that sum to 1)
- make sure a single gradient step on a batch of data yields a decrease in your loss
- make assertions about your datasets
- check for label leakage between your training and validation datasets

The main goal here is to identify some errors early so we can avoid a wasted training job.

Post-train tests

However, in order for us to be able to understand model behaviors we'll need to test against trained model artifacts. These tests aim to **interrogate the logic learned during training** and provide us with a behavioral report of model performance.

Invariance Tests

Invariance tests allow us to describe a set of perturbations we should be able to make to the input *without* affecting the model's output. We can use these perturbations to produce pairs of input examples (original and perturbed) and **check for consistency** in the model predictions. This is closely related to the concept of data augmentation, where we apply perturbations to inputs during training and preserve the original label.

For example, imagine running a sentiment analysis model on the following two sentences:

- Mark was a great instructor.
- Samantha was a great instructor.

We would expect that simply changing the name of the subject doesn't affect the model predictions.

Directional Expectation Tests

Directional expectation tests, on the other hand, allow us to define a set of perturbations to the input which *should* have a *predictable* effect on the model output.

For example, if we had a housing price prediction model we might assert:

- Increasing the number of bathrooms (holding all other features constant) should not cause a drop in price.
- Lowering the square footage of the house (holding all other features constant) should not cause an increase in price.

Let's consider a scenario where a model fails the second test - taking a random row from our validation dataset and decreasing the feature house_sq_ft yields a higher predicted price than the original label. This is surprising as it doesn't match our intuition, so we decide to look further into it. We realize that, without having a feature for the house's neighborhood/location, our model has learned that smaller units tend to be more expensive; this is due to the fact that smaller units from our dataset are more prevalent in cities where prices are generally higher. In this case, the selection of our dataset has influenced the model's logic in unintended ways - this isn't something we would have been able to identify simply by examining performance on a validation dataset.

Minimum Functionality Tests (aka data unit tests)

Just as software unit tests aim to isolate and test atomic components in your codebase, data unit tests allow us to quantify model performance for specific cases found in your data.

This allows you to identify critical scenarios where prediction errors lead to high consequences. You may also decide to write data unit tests for failure modes that you uncover during error analysis; this allows you to "automate" searching for such errors in future models.

Snorkel also has introduced a very similar approach through their concept of slicing functions. These are programatic functions which allow us to identify subsets of a dataset which meet certain criteria. For example, you might write a slicing function to identify sentences less than 5 words to evaluate how the model performs on short pieces of text.

TEST CASES & REPORTING

Test Case

Functional Test Cases

Test ID (#)	Test Scenario	Test Case	Execution Steps	Expected Outcome	Actual Outcome	Status	Remarks
1	Precision Score	The number of positive class predictions that actually belong to the positive class	1.Enter input data 2.Predict 3.Train again	Precision should be close to 1	0.606741573033708	Pass / Failure	success
2	Accuracy Score	Number of correct predictions	1.Enter input data 2.Predict 3.Train again	Accuracy should be close to 1	0.736220472440945	Pass/Failure	success
3	Recall Score	Measures the model performance in terms of measuring the count of true positives in a correct manner out of all the actual positive values.	1.Enter input data 2.Predict 3.Train again	Accuracy should be close to 1	0.627906976744186	Pass/Failure	success
4	F1 Score	Harmonic Mean between precision and recall	1.Enter input data 2.Predict 3.Train again	F1 score should be close to 1	0.617142857142857	Pass/Failure	success

Non-Functional Test Cases

Test ID (#)	Test Scenario	Test Case	Execution Steps	Expected Outcome	Actual Outcome	Status	Remarks
1	Pre train Test	To identify some bugs early on and short-circuit a training job.	1.Draw the shape of your model output. 2. Check the shape of your model output	The output of a classification model should be a distribution with class probabilities that sum to 1)	Probability sum is close to 1.	Pass/Failure	success
2	Post Train Test	To inspect behaviors for a variety of important scenarios that we define.	1.check for consistency 2.Update the model	We would expect that simply changing the name of the subject doesn't affect the model predictions.	Probability sum is close to 1.	Pass/Failure	success

Testing:

The testing is done.

Problem:

The data is to be pre processed more.

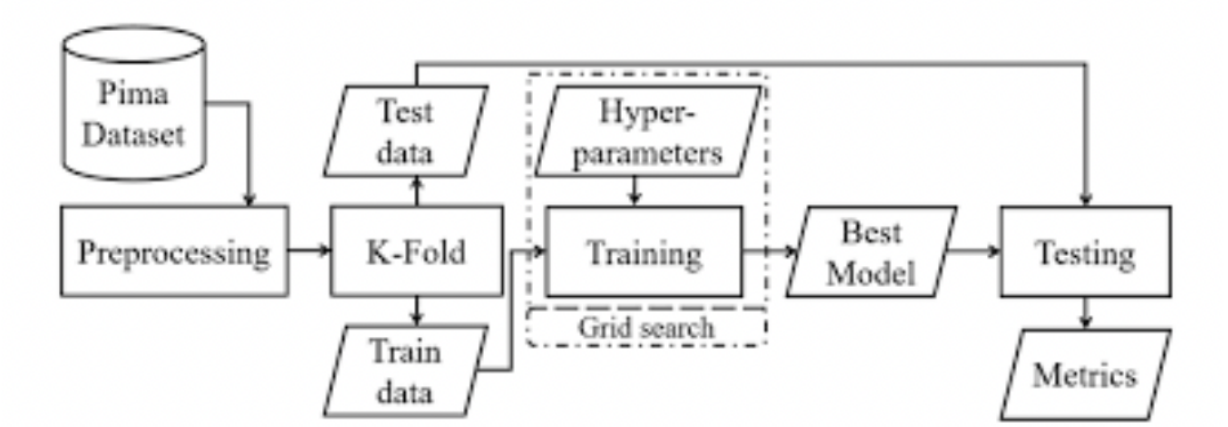
Category	Progress Against Plan	Status
Functional Testing	Green	Completed
Non-Functional Testing	Green	Completed

Functional	Test Case Coverage (%)	Status
Precision Score	100%	Completed
Accuracy Score	100%	Completed
Recall Score	100%	Completed
F1 Score	100%	Completed

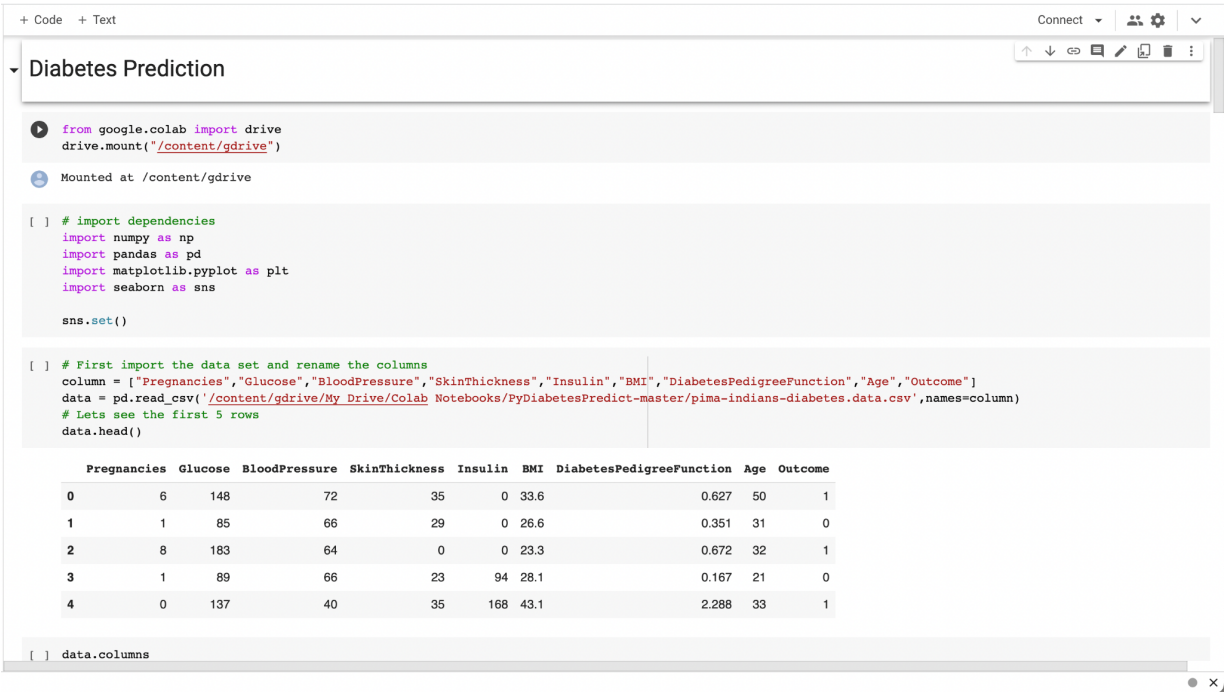
Functional	Test Case Coverage (%)	Status
Pre train Test	100%	Completed
Post Train Test	100%	Completed

ARCHITECTURE/DESIGN/FRAMEWORK/IMPLEMENTATION

Framework :



Implementation :



+ Code + Text — Connect

Diabetes Prediction

```
from google.colab import drive
drive.mount("/content/gdrive")
```

```
Mounted at /content/gdrive
```

```
# import dependencies
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

sns.set()
```

```
# First import the data set and rename the columns
column = ["Pregnancies","Glucose","BloodPressure","SkinThickness","Insulin","BMI","DiabetesPedigreeFunction","Age","Outcome"]
data = pd.read_csv('/content/gdrive/My Drive/Colab Notebooks/PyDiabetesPredict-master/pima-indians-diabetes.data.csv',names=column)
# Lets see the first 5 rows
data.head()
```

	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

```
data.columns
```

+ Code + Text Connect

```
[ ] data.describe()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

```
[ ] data.isnull().head(10)
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False
5	False	False	False	False	False	False	False	False	False
6	False	False	False	False	False	False	False	False	False
7	False	False	False	False	False	False	False	False	False
8	False	False	False	False	False	False	False	False	False
9	False	False	False	False	False	False	False	False	False

+ Code + Text Connect

```
[ ] data.columns
```

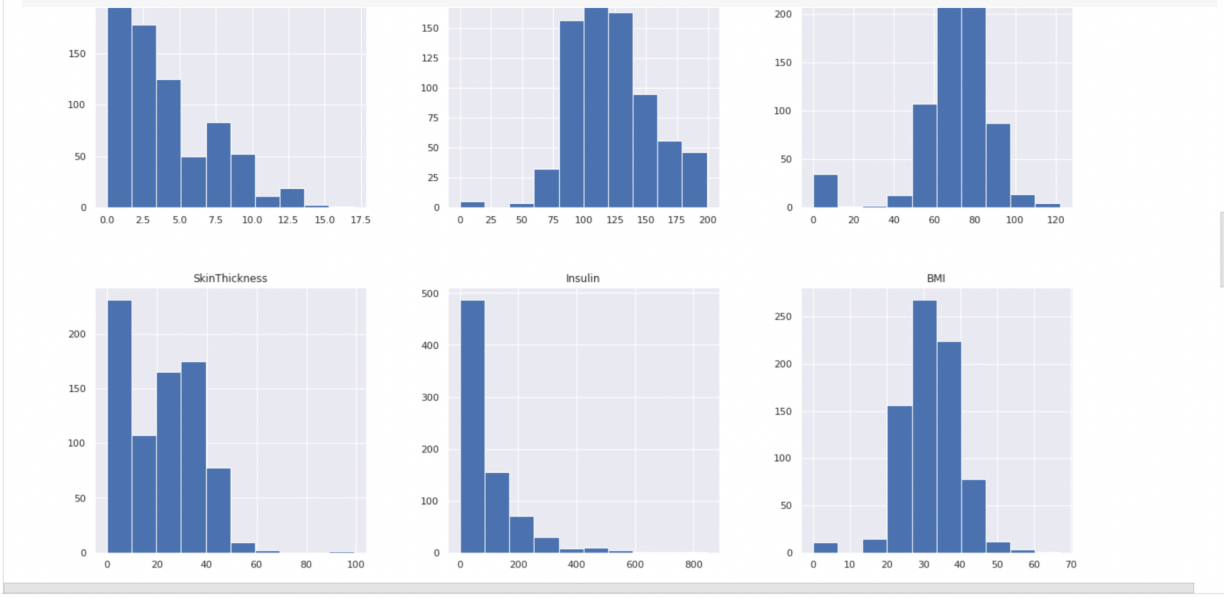
```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

```
[ ] data.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
```

+ Code + Text Connect

```
[ ] p = data.hist(figsize = (20,20))
```



+ Code + Text Connect

```
[ ] data.isnull().sum()
```

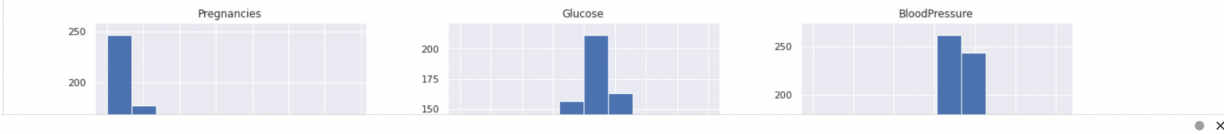
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

```
[ ] data_copy = data.copy(deep = True)
    data_copy[['Glucose','BloodPressure','SkinThickness','Insulin','BMI']] = data_copy[['Glucose','BloodPressure','SkinThickness','Insulin','BMI']].replace(0,np

    # Showing the Count of NANs
    print(data_copy.isnull().sum())
```

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

```
[ ] p = data.hist(figsize = (20,20))
```



+ Code + Text Connect

```
[ ] from sklearn.model_selection import train_test_split

    x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)
    print('Number of rows in the total set: {}'.format(data.shape[0]))
    print('Number of rows in the training set: {}'.format(x_train.shape[0]))
    print('Number of rows in the test set: {}'.format(x_test.shape[0]))
```

```
Number of rows in the total set: 768
Number of rows in the training set: 514
Number of rows in the test set: 254
```

Using Gaussian Naive Bayes to train our model

```
[ ] from sklearn.naive_bayes import GaussianNB
    naive_bayes = GaussianNB()
    naive_bayes.fit(x_train, y_train)
    predictions = naive_bayes.predict(x_test)
```

Model Evaluation

```
[ ] from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
    print('Accuracy score: ', format(accuracy_score(y_test, predictions)))
    print('Precision score: ', format(precision_score(y_test, predictions)))
    print('Recall score: ', format(recall_score(y_test, predictions)))
    print('F1 score: ', format(f1_score(y_test, predictions)))
```

```
Accuracy score:  0.7362204724409449
Precision score:  0.6067415730337079
Recall score:  0.627906976744186
F1 score:  0.6171428571428572
```

```
[ ]
```

+ Code + Text Connect

Now lets split the dataset in Training and Testing Sets

```
[ ] X = data.iloc[:,0:-1] # X is the features in our dataset
    y = data.iloc[:,-1]   # y is the Labels in our dataset
```


```
[ ] from sklearn.model_selection import train_test_split

    x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)
    print('Number of rows in the total set: {}'.format(data.shape[0]))
    print('Number of rows in the training set: {}'.format(x_train.shape[0]))
    print('Number of rows in the test set: {}'.format(x_test.shape[0]))
```

```
Number of rows in the total set: 768
Number of rows in the training set: 514
Number of rows in the test set: 254
```

Using Gaussian Naive Bayes to train our model

```
[ ] from sklearn.naive_bayes import GaussianNB
    naive_bayes = GaussianNB()
    naive_bayes.fit(x_train, y_train)
    predictions = naive_bayes.predict(x_test)
```

Model Evaluation

```
[ ] from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
    print('Accuracy score: ', format(accuracy_score(y_test, predictions)))
    print('Precision score: ', format(precision_score(y_test, predictions)))
    print('Recall score: ', format(recall_score(y_test, predictions)))
    print('F1 score: ', format(f1_score(y_test, predictions)))
```

CONCLUSION

After using all these patient records, we are able to build a machine learning model (random forest – best one) to accurately predict whether or not the patients in the dataset have diabetes or not along with that we were able to draw some insights from the data via data analysis and visualization.

REFERENCES

1. https://www.pmi.org/
2. https://www.projectmanagement.com/
3. https://www.tpsgc-pwgsc.gc.ca/biens-property/sngp-npms/ti-it/ervcpgpm-dsfvpmpt-eng.html