

## **PROJECT DESIGN PHASE - 2**

### **AI BASED DIABETES PREDICTION SYSTEM**

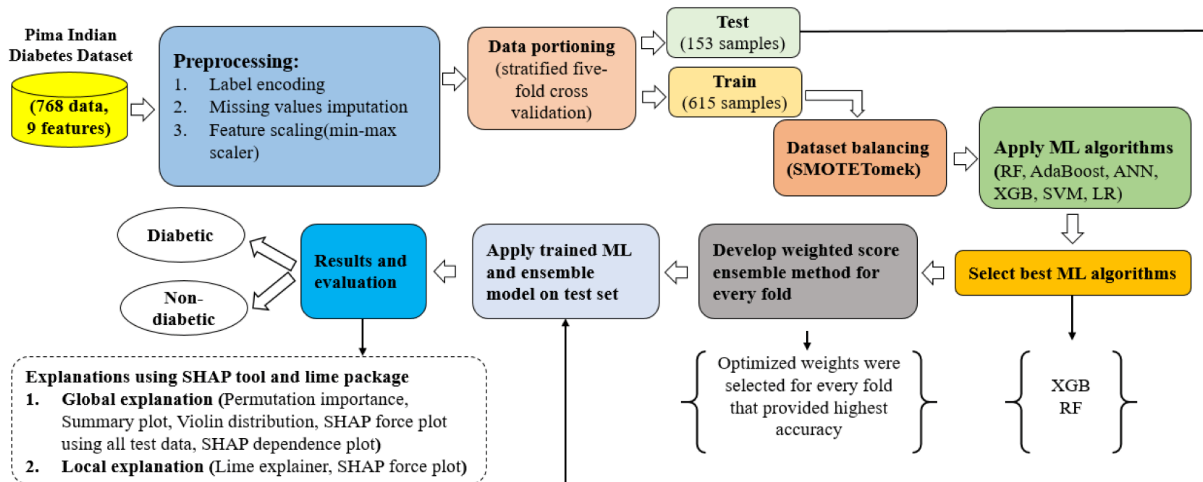
#### **Introduction :**

AI in Healthcare is an industry that always makes it necessary to make a precise decision, whether it is a treatment, test, or discharge. Diabetes is common due to modern food intake, and it is necessary to keep track of the body. AI in Diabetes helps to predict or Detect Diabetes. Any neglect in health can have a high cost for the patients and the medical practitioner. It becomes challenging for the patient to trust that this decision is taken by the machine that does not explain how it reaches a particular conclusion.

Diabetes related diseases have recently become one of the top ten causes of death in developing countries. The government and individuals are funding research projects to find an easier and faster way to detect the disease at an early stage. There are two types of diabetes: type-1 and type-2. Type 2 diabetes is characterized by high blood sugar, insulin resistance, and a relative lack of insulin. Insulin resistance occurs due to excessive fat in the abdomen and around the organs, which is called visceral fat. The majority of obese individuals have elevated plasma levels of free fatty acids (FFA) which are known to cause peripheral (muscle) insulin resistance . Type-1 diabetes is a condition in which blood sugar levels rise due to a shortage of insulin, causing problems with the blood sugar metabolism. Most food people eat is broken down into sugar (glucose) and released into the bloodstream. When blood sugar levels rise, the pancreas cell releases insulin, which provides energy for everyday tasks . Excessive blood sugar stays in the bloodstream when there is not enough insulin or if the cells stop responding to the insulin. This can cause serious health problems such as heart disease, vision loss, and kidney disease in the long run.

## Methodology :

The whole workflow of the proposed approach is demonstrated in Figure 1. The data was downloaded from Kaggle (<https://www.kaggle.com/datasets/mathchi/diabetes-data-set>) then it was cleaned and pre-processed (missing values imputation, class balance, etc.).



## **DATA PREPROCESSING :**

Data preprocessing is an important step in the data mining process. It refers to the cleaning, transforming, and integrating of data in order to make it ready for analysis. The goal of data preprocessing is to improve the quality of the data and to make it more suitable for the specific data mining task.

Data preprocessing is an important step in the data mining process that involves cleaning and transforming raw data to make it suitable for analysis.

## **CODE :**

```
import numpy as np

import pandas as pd
```

```
import matplotlib.pyplot as plt
```

```
df=pd.read_csv('/kaggle/input/diabetes-data-set/diabetes.csv')
```

### OUTPUT:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction
0	6	148	72	35	0	33.6	0.627
1	1	85	66	29	0	26.6	0.351
2	8	183	64	0	0	23.3	0.672
3	1	89	66	23	94	28.1	0.167
4	0	137	40	35	168	43.1	2.288
5	5	116	74	0	0	25.6	0.201
6	3	78	50	32	88	31.0	0.248
7	10	115	0	0	0	35.3	0.134
8	2	197	70	45	543	30.5	0.158
9	8	125	96	0	0	0.0	0.232

### CODE:

```
df.describe()
```

### OUTPUT :

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction
count	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	0.472669
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.333114
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243500
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.374700
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.624800
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.425000

**CODE :**

```
df.info()
```

**OUTPUT :**

---

```
<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 :**

```
df=df.drop_duplicates()
```

```
df.shape()
```

**OUTPUT :**

```
(768, 9)
```

**CODE :**

```
df.isnull().sum()
```

```
df.columns()
```

## OUTPUT :

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

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

## DATA VISUALISATION :

### CODE :

```
f, ax = plt.subplots(1, 2, figsize=(10, 5))

df['Outcome'].value_counts().plot.pie(explode=[0, 0.1], autopct='%1.1f%%',
ax=ax[0], shadow=True)

ax[0].set_title('Outcome')

ax[0].set_ylabel('')

# Count plot for Outcome distribution

sns.countplot(x='Outcome', data=df, ax=ax[1]) # Use 'x' instead of 'Outcome'

ax[1].set_title('Outcome')
```

*# Display class distribution*

```
N, P = df['Outcome'].value_counts()
```

```
print('Negative (0):', N)
```

```
print('Positive (1):', P)
```

*# Adding grid and showing plots*

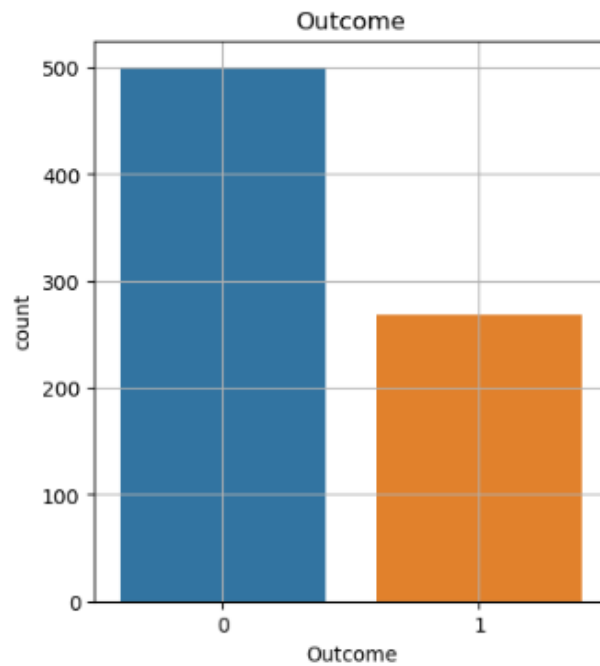
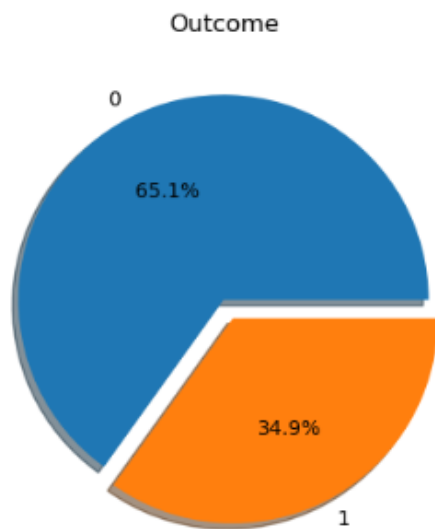
```
plt.grid()
```

```
plt.show()
```

**OUTPUT :**

Negative (0): 500

Positive (1): 268



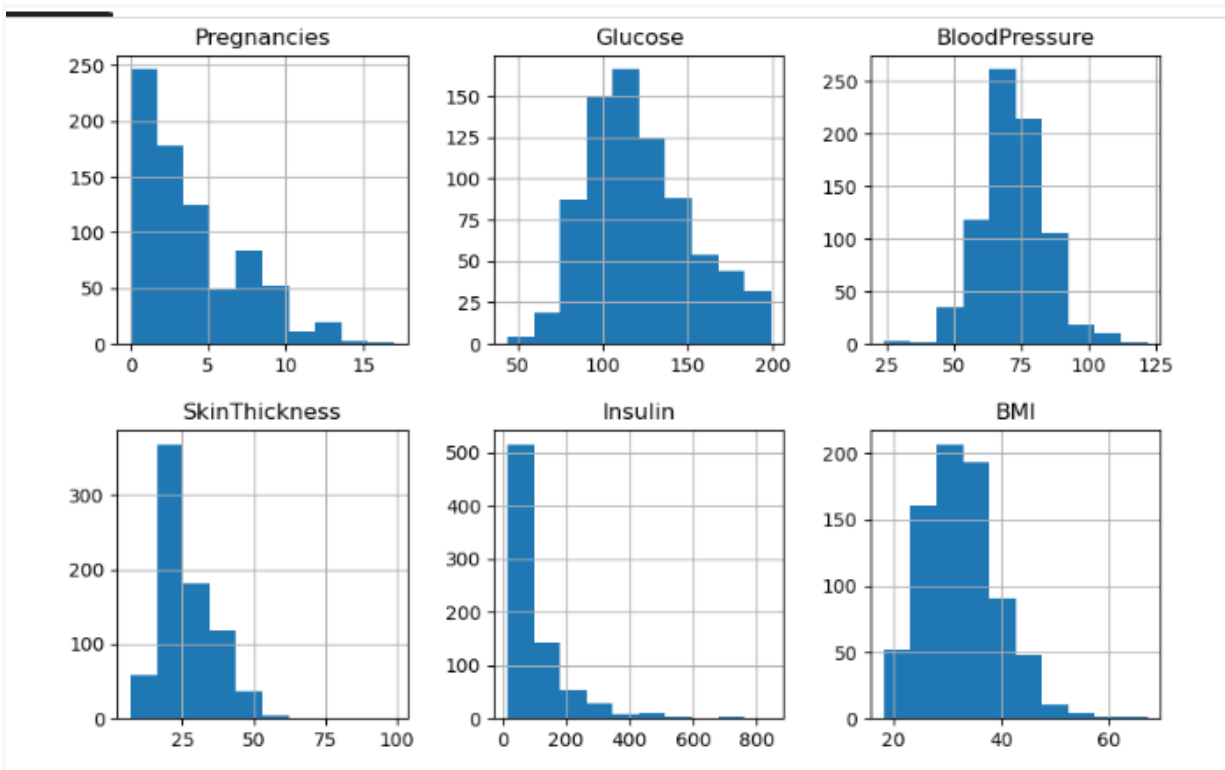
## HISTOGRAM :

### CODE :

```
df.hist(bins=10, figsize=(10, 10))
```

```
plt.show()
```

### OUTPUT :



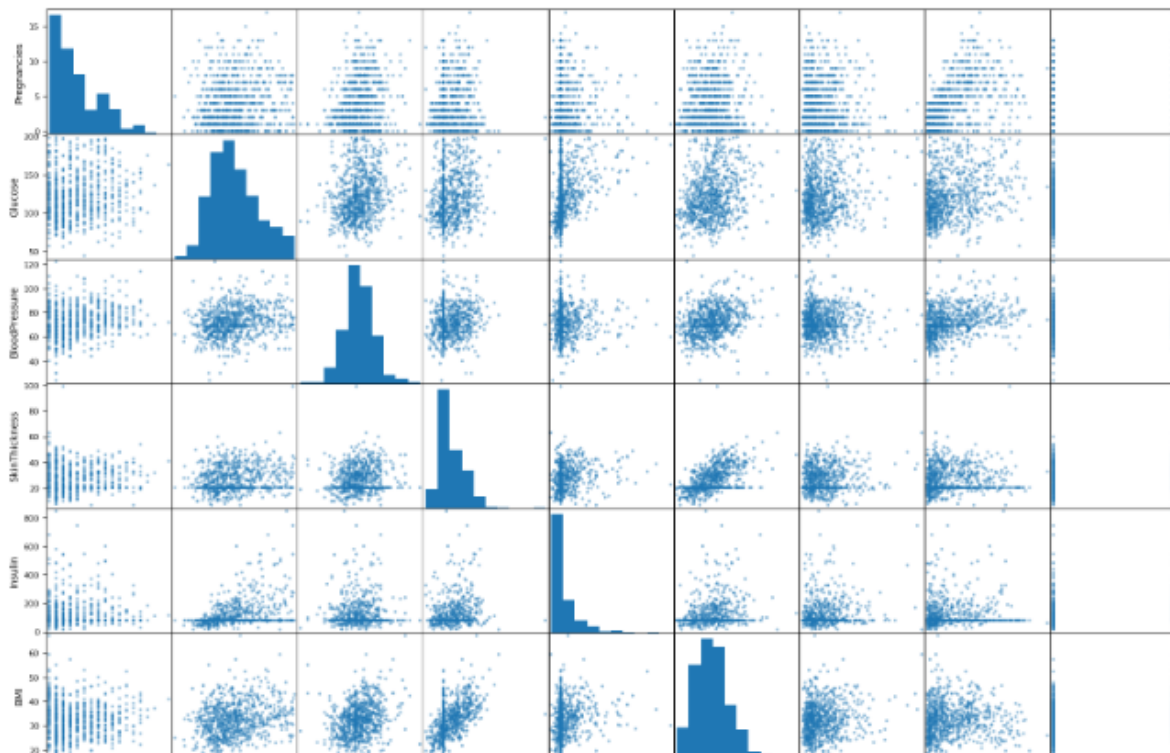
## SCATTER PLOT :

### CODE :

```
from pandas.plotting import scatter_matrix
```

```
scatter_matrix(df, figsize=(20, 20))
```

### OUTPUT :



## **PAIR PLOT :**

You can create a pair plot in a Pandas data frame using the `pairplot()` function from Seaborn. By customizing the arguments in the `pairplot()` function, you can customize the appearance of the pair plot to better suit your needs.

The Seaborn Pairplot allows us to plot pairwise relationships between variables within a dataset. This creates a nice visualisation and helps us understand the data by summarising a large amount of data in a single figure.

## **CODE :**



```
sns.pairplot(data=df, hue='Outcome')
```

```
plt.show()
```

**OUTPUT :**



**HEATMAP :**

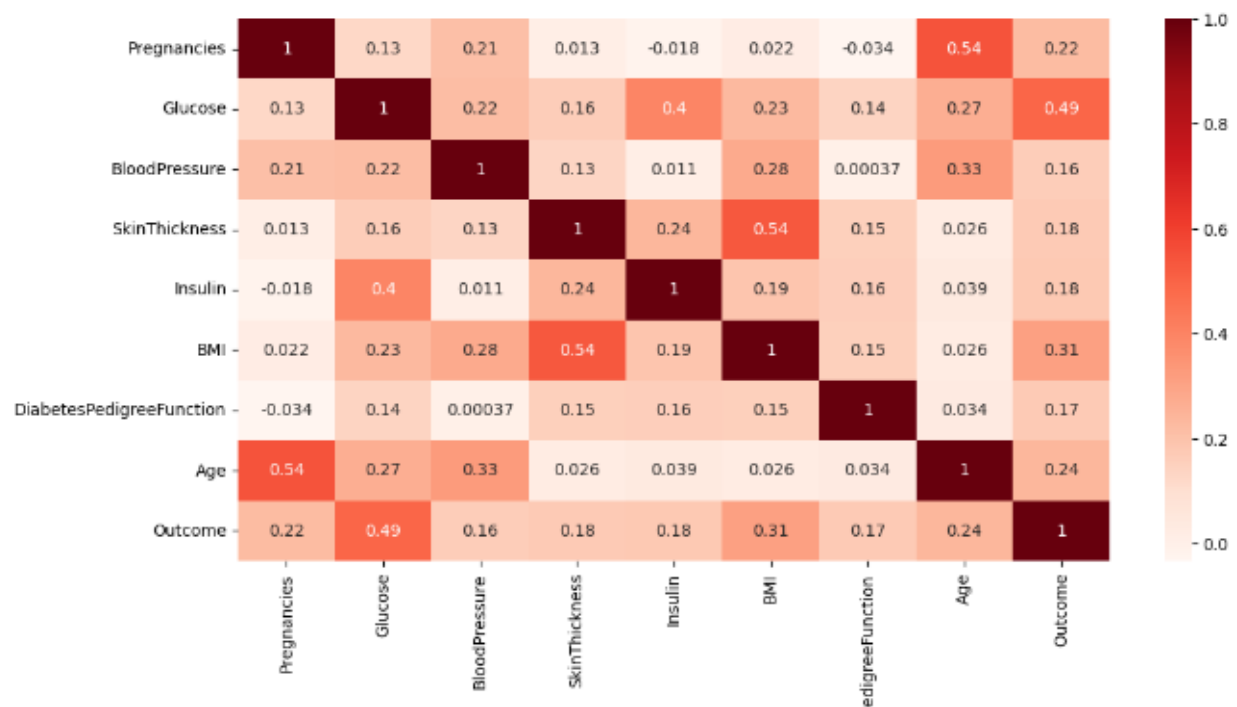
**CODE :**

```
plt.figure(figsize=(12, 6))
```

```
sns.heatmap(df.corr(), annot=True, cmap='Reds')
```

```
plt.plot()
```

## OUTPUT :



## FUTURE SCALING :

### CODE :

*# Standard Scaler:*

```
from sklearn.preprocessing import StandardScaler
```

```
scaler = StandardScaler()
```

```
scaler.fit(X)
```

```
SSX = scaler.transform(X)
```

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(SSX, y, test_size=0.2,  
random_state=7)
```

## Classification Algorithms:

### Logistic Regression:

CODE :

```
from sklearn.linear_model import LogisticRegression
```

```
lr = LogisticRegression(solver='liblinear', multi_class='ovr')
```

```
lr.fit(X_train, y_train)
```

OUTPUT :

```
|:  
└─ LogisticRegression  
   LogisticRegression(multi_class='ovr', solver='liblinear')
```

### Decision Tree:

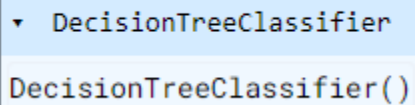
CODE :

```
from sklearn.tree import DecisionTreeClassifier
```

```
dt=DecisionTreeClassifier()
```

```
dt.fit(X_train, y_train)
```

## OUTPUT :



```
▼ DecisionTreeClassifier  
DecisionTreeClassifier()
```

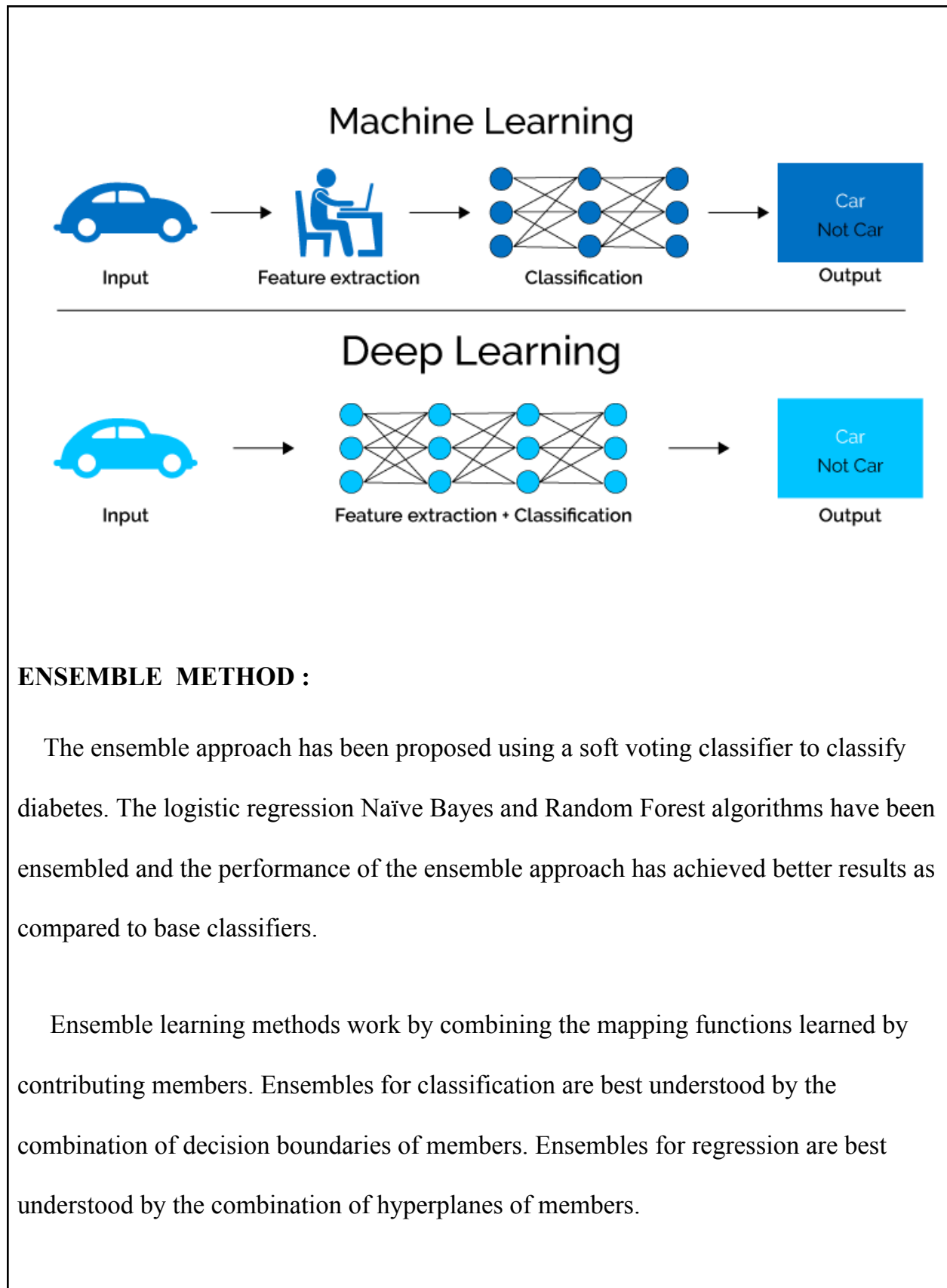
## DEEPLARNING :

***Convolutional neural networks (CNNs)*** : used primarily in computer vision and image classification applications, can detect features and patterns within an image, enabling tasks, like object detection or recognition. In 2015, a CNN bested a human in an object recognition challenge for the first time.

***Recurrent neural network (RNNs)***: are typically used in natural language and speech recognition applications as it leverages sequential or times series data.

Deep learning is a subset of machine learning, which is essentially a neural network with three or more layers. These neural networks attempt to simulate the behavior of the human brain—albeit far from matching its ability—allowing it to “learn” from large amounts of data.

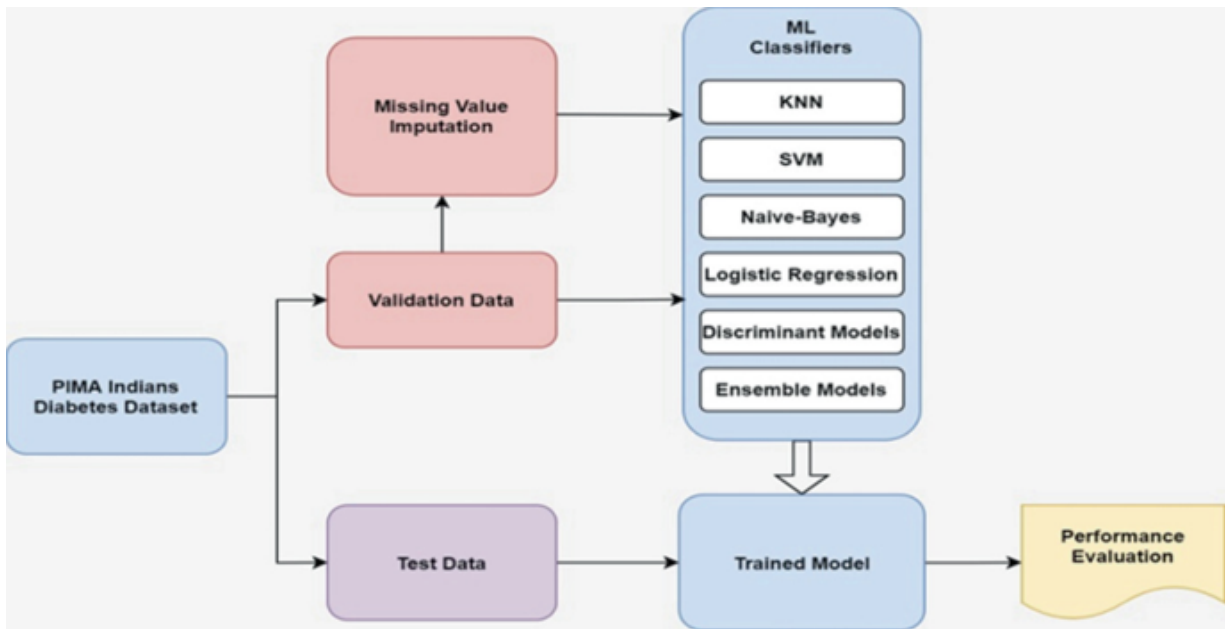
Because deep learning models process information in ways similar to the human brain, they can be applied to many tasks people do. Deep learning is currently used in most common image recognition tools, natural language processing (NLP) and speech recognition software.



### **ENSEMBLE METHOD :**

The ensemble approach has been proposed using a soft voting classifier to classify diabetes. The logistic regression Naïve Bayes and Random Forest algorithms have been ensembled and the performance of the ensemble approach has achieved better results as compared to base classifiers.

Ensemble learning methods work by combining the mapping functions learned by contributing members. Ensembles for classification are best understood by the combination of decision boundaries of members. Ensembles for regression are best understood by the combination of hyperplanes of members.



## 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.