# **PROBLEM DEFINITION FOR AN AI-BASED DIABETES PREDICTION SYSTEM:**

**INTRODUCTION**:

**DIABETES – “A GLOBAL HEALTH CRISIS”**

* Diabetes is a widespread chronic health condition, posing a significant challenge to healthcare systems worldwide. The prevalence of diabetes has reached epidemic proportions, affecting millions of people and leading to severe health and economic consequences. Timely detection and risk assessment are pivotal in managing and preventing diabetes and its associated complications. This presentation delves into the comprehensive problem definition for developing an AI-Based Diabetes Prediction System.

**THE DIABETES CHALLENGE:**

**UNDERSTANDING THE SPECTRUM OF DIABETES:**

* Diabetes encompasses various metabolic disorders characterized by elevated blood sugar levels (hyperglycaemia).
* Two primary types: Type 1 (autoimmune, often diagnosed in childhood) and Type 2 (associated with lifestyle factors).
* Focus on the increasing incidence of preventable Type 2 diabetes.

**OBJECTIVE**:

**EMPOWERING HEALTHCARE PROFESSIONALS AND INDIVIDUALS:**

* Primary objective: Develop an AI-Based Diabetes Prediction System.
* Empower both healthcare professionals and individuals.
* Assess the likelihood of developing Type 2 diabetes.
* Utilize health parameters: blood glucose levels, BMI, family history, diet, and more.

**SCOPE:**

**COMPREHENSIVE SCOPE FOR EFFECTIVE PREDICTIONS:**

* Understanding Diabetes: In-depth research to grasp diabetes nuances, including risk factors and complications.
* Early Detection: Core focus on early detection to reduce complications.
* Healthcare and Individual Empowerment: Personalized guidance for healthcare providers, informed choices for individuals.
* Predictive Metrics: Specific metrics for evaluating system performance.

**DATA COLLECTION AND PRIVACY:**

**ENSURING DATA SECURITY AND PRIVACY:**

* Gather diverse datasets: medical records, genetic data, lifestyle information, real-time health monitoring.
* Address data privacy concerns: compliance with regulations like HIPAA.

**STAKEHOLDER :**

**HARNESSING MULTIDISCIPLINARY EXPERTISE:**

* Collaborative efforts: Engage data scientists, healthcare professionals, domain experts, and designers.
* Diverse team with expertise in machine learning, medical science, and user interface design.

**CHALLENGES:**

**NAVIGATING COMPLEXITIES AND CHALLENGES:**

* Challenges include the complexity of diabetes, dataset diversity, algorithmic robustness, user-friendliness, adaptability to evolving health conditions, and updates based on medical research.

# **DESIGN THINKING FOR AN AI-BASED DIABETES PREDICTION SYSTEM:**

**INTRODUCTION:**

**INNOVATION IN HEALTHCARE : DESIGN THINKING APPROACH:**

* Design thinking is a human-centred problem-solving approach that fosters innovation. Its application to the development of an AI-Based Diabetes Prediction System offers the potential to create a solution that truly addresses the needs of healthcare professionals and individuals at risk of diabetes. This presentation explores the design thinking process and its relevance to our project.

**EMPATHIZE:**

**UNDERSTANDING USER NEEDS:**

* In the “Empathize” phase, we delve into the needs and pain points of our primary users: healthcare providers and individuals at risk of diabetes.
* Research methods include interviews, surveys, and observation to gain deep insights into their concerns and requirements.
* Empathy ensures that the system we develop caters to their specific needs.

**DEFINE:**

**PROBLEM FRAMING AND METRICS FOR SUCCESS:**

* The “Define” phase is about precisely framing the problem. We will set clear objectives for our AI-Based Diabetes Prediction System.
* Specific metrics for success will be established, including prediction accuracy, usability, scalability, and cost-effectiveness.
* Defining the problem is crucial for staying focused on the project’s goals.

**IDEATE:**

**ENCOURAGING CREATIVE COLLABORATION:**

* The “Ideate” is the brainstorming phase, where interdisciplinary collaboration among data scientists, healthcare professionals, and designers takes centre stage.
* We will foster a creative environment where diverse perspectives generate innovative ideas.
* Creative brainstorming will ensure that our system is both effective and user-friendly.

**PROTOTYPE:**

**BUILDING VISUAL REPRESENTATIONS:**

* Prototyping involves creating mock-ups and a minimal viable product (MVP) of the AI system.
* These prototypes will serve as visual representations of the system’s user interface and functionality.
* Prototyping helps us visualize and refine the system’s design.

**TEST:**

**USER-CENTRIC EVALUATION:**

* In the “Test” phase, usability testing becomes paramount. Potential users will engage with our prototypes to evaluate their functionality.
* Gathering user feedback is essential for making iterative improvements.
* User-centred design ensures that the system meets user expectations.

**IMPLEMENT:**

**TECHNOLOGY AND SECURITY:**

* The “Implement” phase involves the development of our AI-Based Diabetes Prediction System.
* State-of-the-art machine learning algorithms and relevant data sources will be employed.
* Robust data privacy and security measures will be implemented to safeguard sensitive health data.

**EVALUATE:**

**CONTINUOUS IMPROVEMENT:**

* Continuous monitoring and evaluation are crucial throughout the system’s lifecycle.
* We will measure its accuracy, reliability, and adherence to predefined metrics.
* Evaluation ensures that the system performs effectively and remains relevant.

**DEPLOY:**

**REAL-WORLD APPLICATION:**

* Upon successful development, we will roll out the system for use by healthcare providers and individuals.
* Comprehensive training and support will be provided to maximize its impact.
* Deployment brings our solution into the real world, where it can make a difference.

**ITERATE:**

**STAYING CUTTING-EDGE:**

* In the “Iterate” phase, we commit to an ongoing process of enhancement.
* User feedback, changing healthcare needs, and advancements in AI and diabetes research will inform our updates.
* Iteration ensures that our system remains cutting-edge and effective.

# **MODEL FOR INNOVATION FOR AN AI DIABETES PREDICTION SYSTEM:**

IMAGINE A HEALTHCARE TECHNOLOGY COMPANY, HEALTHTECH INNOVATORS, EMBARKING ON THE DEVELOPMENT OF AN AI-BASED DIABETES PREDICTION SYSTEM. HERE’S HOW THEY APPLY THE MODEL FOR INNOVATION:

**OBTAINING THE DATASET:**

* The first step in any machine learning project is to obtain a dataset to work with. There are many online resources available to find the datasets suitable for our project. One such popular platform that hosts datasets is Kaggle. Since the goal of our project is to predict Diabetes, I search for diabetes patient’s dataset, which is readily available as were already many such similar projects.
* Go to the Kaggle dataset page you mentioned: <https://www.kaggle.com/datasets/mathchi/diabetes-data-set>
* Click the “Download” button to get the dataset files.
* Unzip the downloaded files to a directory on your local machine into your project repository
* This dataset has the following attributes : Pregnancies, Glucose, Blood Pressure, Skin Thickness, Insulin, BMI, Diabetes Pedigree Function and Age, It also contains the Outcome label which tells whether the patient has diabetes or not. Since there is a class label present by default, this dataset can be studied using Classification based algorithms.

**PROBLEM IDENTIFICATION:**

**Recognize the global diabetes epidemic as a pressing issue affecting millions.**

* Identify the need for an AI system that can predict diabetes risk and empower healthcare professionals and individuals.

**MULTIDISCIPLINARY TEAM:**

* Form a team consisting of data scientists, endocrinologists, nutritionists, user experience designers, and cybersecurity experts.

**CONTINUOUS RESEARCH:**

* Conduct extensive research to understand the complexities of diabetes, its risk factors, and potential complications.

**ETHICAL DATA HANDLING:**

* Develop strict data privacy protocols to ensure compliance with healthcare data regulations.
* Implement robust encryption and access control mechanisms.

**MACHINE LEARNING EXPERTISE:**

* Utilize cutting-edge machine learning algorithms and collaborate with AI researchers to enhance prediction accuracy.

**DESIGN THINKING APPROACH:**

* Empathize with healthcare providers and individuals through interviews and surveys to identify their specific needs and concerns.

**PROTOTYPING AND USABILITY TESTING:**

* Create interactive prototypes of the AI system’s user interface.
* Conduct usability testing with healthcare professionals and individuals to refine the design.

**REAL-WORLD DEPLOYMENT:**

* Roll out the AI system in partnership with healthcare institutions and clinics.
* Provide comprehensive training to healthcare staff and user-friendly interfaces for individuals.

**CONTINUOUS MONITORING:**

* Set up a monitoring system to track prediction accuracy and system performance.
* Regularly gather user feedback and iterate on system improvements.

**SCALABILITY AND ACCESSIBILITY:**

* Design the system to handle a growing volume of health data.
* Ensure the system is accessible to individuals of diverse backgrounds and regions.

**COLLABORATION WITH HEALTHCARE INSTITUTIONS:**

* Partner with leading healthcare institutions to integrate the AI system into their electronic health record systems.
* Collaborate on research initiatives to advance diabetes care.

**CODE:**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

dataset = pd.read\_csv('Documents/Roshan/diabetes.csv') dataset.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 |

dataset.shape

(768, 9)

dataset.info()

‹class 'pandas.core.frame.DataFrame'> RangeIndex: 768 entries, 0 to 767 Data columns (total 9 columns):

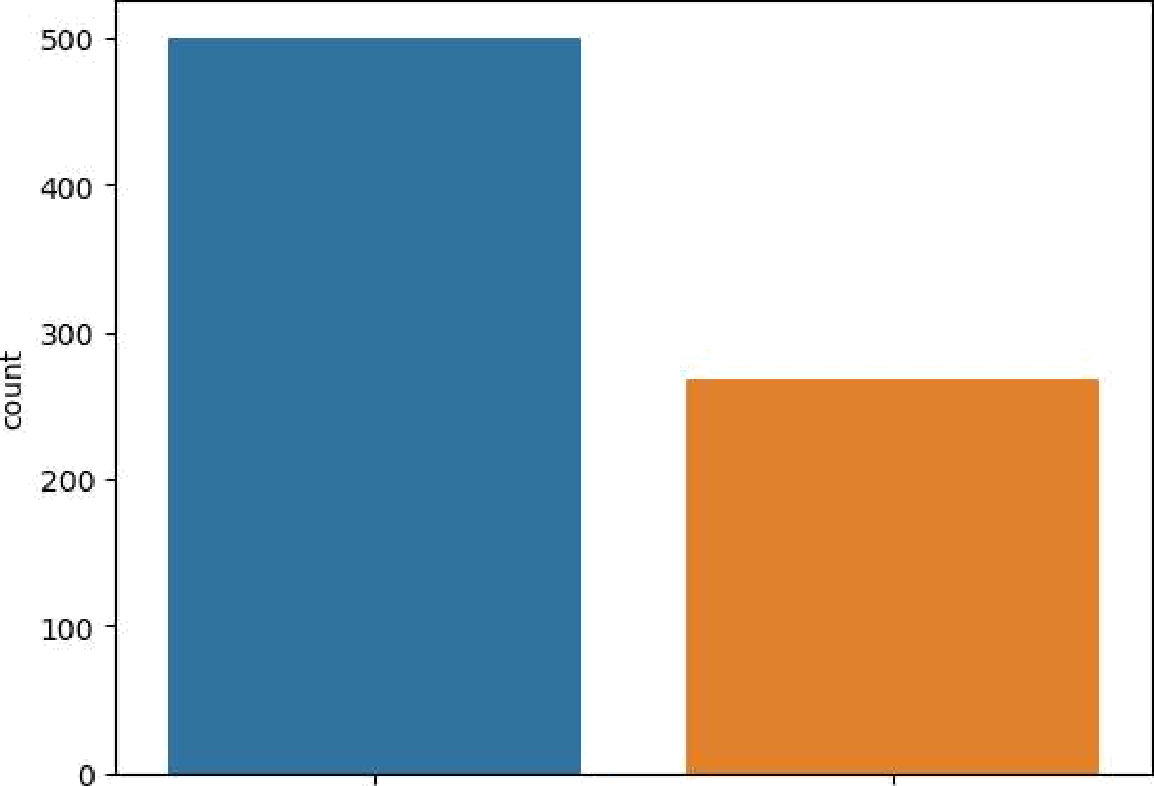
Non-Null Count Dtype

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | Pregnancies | 768 non-null | iut64 |
| 1 | Glucose | 768 non-null | int64 |
| 2 | BloodPressure | 768 non-null | iut64 |
| 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 | iut64 |

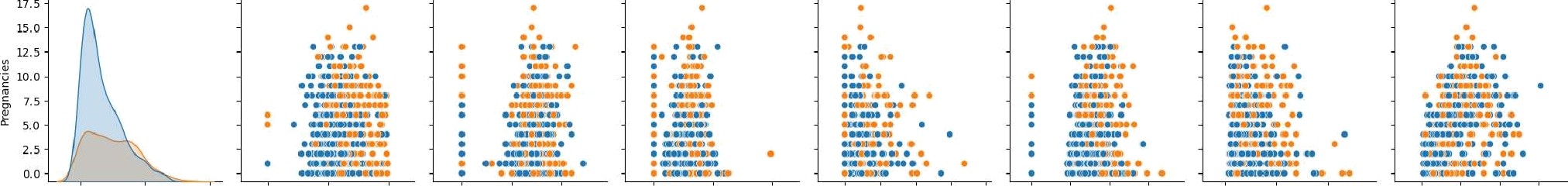
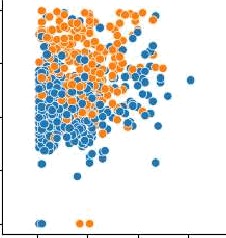
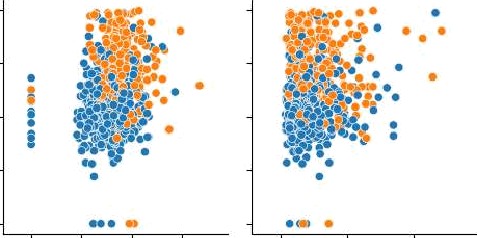
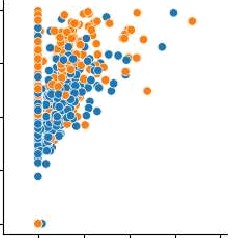
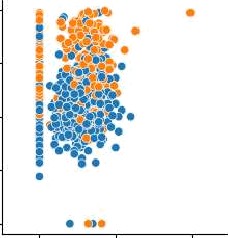
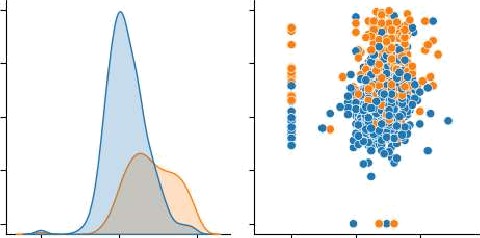
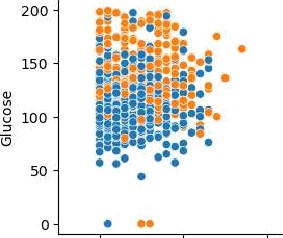
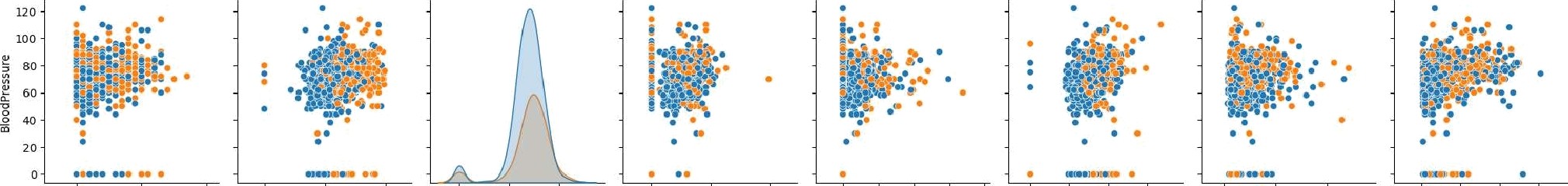
dtypes: float64(2), int64(7)

memory usage: 54.1 KB

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| dataset.describe().T | count | mean | std | m1n | 25% | 5e% | 75% | max |
| Pregnancies | 768.0 | 3.845052 | 3.369578 | 0.000 | 1.00000 | 3.0000 | 6.00000 |  |
| 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 |  |
| **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 |  |
| 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 |  |
| 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 |  |
| dataset . lsnuII ( ) . sum( ) |  |  |  |  |  |  |  |  |
| Pregnant ies | 0 |  |  |  |  |  |  |  |
| G1ucose | 0 |  |  |  |  |  |  |  |
| BloodPressure | 0 |  |  |  |  |  |  |  |
| Sk1nTh1c kness | 0 |  |  |  |  |  |  |  |
| Insulin | 0 |  |  |  |  |  |  |  |
| BMI | 0 |  |  |  |  |  |  |  |
| DiabetesPedigreeFunction | 0 |  |  |  |  |  |  |  |
| Age | 0 |  |  |  |  |  |  |  |



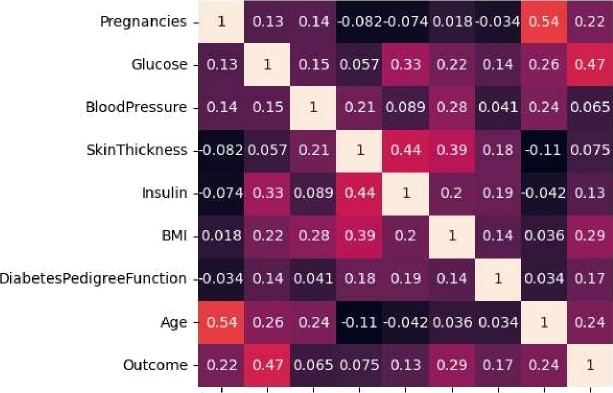
Outcome



# Heatmap

sns.heatmap(dataset.corr(), annot = True) p1t.show()

— 1.0



@ E



# Replacing zero values with Nan

SkinThic k ness

Insulin

DiabetesPedigreeF unction

dataset\_new = dataset

dataset\_new[[“Glucose“, "BloodPressure“, “SkinThickness“, “Insulin", “BMI“]] = dataset\_new[[“Glucose“, “BloodPressure“, “SkinThickness“, "Iss

# Count of NaN dataset\_new.isnull().sum()

|  |  |
| --- | --- |
| Pregnancies | 0 |
| G1ucose | 5 |
| BloodPressure | 35 |
| SkinThickness | 227 |
| Insulin | 374 |
| BMI | 11 |
| DiabetesPedigreeFunction | 0 |
| Age | 0 |
| Outcome dtype: int64 | 0 |

# 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“].fi11na(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 = Tnue)

dataset\_new.isnull().sum()

Pregnancies 0

Glucose 0

BloodPressure 0

SkinThickness 0

Insulin 0

BMI 0

DiabetesPedigreeFunction 0

Age 0

Outcome 0

dtype: int64

**PREDICTION MODEL:**

In this work, we design a prediction model, that predicts whether a patient has diabetes, based on certain diagnostic measurements included in the dataset,  
and explore various techniques to boost performance and accuracy.

**ABOUT THIS PROJECT:**

* The objective of this project is to classify whether someone has diabetes or not.
* Dataset consists of several Medical Variables(Independent) and one Outcome Variable(Dependent)
* The independent variables in this data set are :-'Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin','BMI', 'DiabetesPedigreeFunction', 'Age'
* The outcome variable value is either 1 or 0 indicating whether a person has diabetes(1) or not(0).

In [1]:

CODE:

import os

for dirname, \_, filenames **in** os.walk('/kaggle/input'):

for filename **in** filenames:

print(os.path.join(dirname, filename))

/kaggle/input/diabetes-data-set/diabetes.csv

In [2]:

import warnings

warnings.filterwarnings("ignore")

In [3]:

*# Importing required libraries*

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

In [4]:

from sklearn.metrics import confusion\_matrix

from sklearn.metrics import accuracy\_score, recall\_score, f1\_score

from sklearn.metrics import roc\_auc\_score, classification\_report

In [5]:

*# reading dataset*

df = pd.read\_csv("/kaggle/input/diabetes-data-set/diabetes.csv")

df.sample(5)

Out[5]:

|  | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | DiabetesPedigreeFunction | Age | Outcome |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 646 | 1 | 167 | 74 | 17 | 144 | 23.4 | 0.447 | 33 | 1 |
| 649 | 0 | 107 | 60 | 25 | 0 | 26.4 | 0.133 | 23 | 0 |
| 445 | 0 | 180 | 78 | 63 | 14 | 59.4 | 2.420 | 25 | 1 |
| 566 | 1 | 99 | 72 | 30 | 18 | 38.6 | 0.412 | 21 | 0 |
| 538 | 0 | 127 | 80 | 37 | 210 | 36.3 | 0.804 | 23 | 0 |

The datasets consist of several medical predictor (independent) variables and one target (dependent) variable, Outcome. Independent variables include the number of pregnancies the patient has had, their BMI, insulin level, age, and so on.

# DATA PREPROCESSING:

EDA is the process of investingating the dataset to discover hidden patterns, anomalies(outliers), relationship and form hypotheses based on our understanding of the dataset.

## 1. UNDERSTANDING THE DATASET:

* Head of the dataset
* Shape of the data set
* Types of columns
* Information about data set
* Summary of the data set

## 2. DATA CLEANING:

* Dropping duplicate values
* Checking NULL values
* Checking for 0 value and replacing it :- It isn't medically possible for some data record to have 0 value such as Blood Pressure or Glucose levels. Hence we replace them with the mean value of that particular column.

In [6]:

*# find the shape of datset*

df.shape

Out[6]:

(768, 9)

In [7]:

*# getting information about dataset*

df.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 [8]:

*# getting statical information about our dataset*

df.describe().T

Out[8]:

|  | 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 |

### MISSING VALUES :

In [9]:

*# checking for missing values*

df.isnull().sum()

Out[9]:

Pregnancies 0

Glucose 0

BloodPressure 0

SkinThickness 0

Insulin 0

BMI 0

DiabetesPedigreeFunction 0

Age 0

Outcome 0

dtype: int64

In [10]:

*# checking duplicate rows in dataset*

df.duplicated().sum()

Out[10]:

0

### DATA TYPES:

Checking for datatypes of our all features

In [11]:

*# getting numerical and categorical features form our dataset*

numerical\_features = [feature for feature **in** df.columns if df[feature].dtypes **not** **in** ['O', 'o', 'object']]

categorical\_fatures = [feature for feature **in** df.columns if df[feature].dtypes **in** ['O', 'o', 'object']]

In [12]:

Numerical\_features

Out[12]:

['Pregnancies',

'Glucose',

'BloodPressure',

'SkinThickness',

'Insulin',

'BMI',

'DiabetesPedigreeFunction',

'Age',

'Outcome']

In [13]:

categorical\_fatures

Out[13]:

[]

## DATA VISUALIZATION:

Here we are going to plot :-

* **Count Plot** :- to see if the dataset is balanced or not
* **Histograms** :- to see if data is normally distributed or skewed
* **Box Plot** :- to analyse the distribution and see the outliers
* **Scatter plots** :- to understand relationship between any two variables
* **Pair plot** :- to create scatter plot between all the variables

# UNIVERIANT ANALYSIS:

Analysing each variables seperatly

In [14]:

*# First of all analysing target feature i.e Outcome*

*# find the unique values count in our target feature*

df["Outcome"].value\_counts()

Out[14]:

0 500

1 268

Name: Outcome, dtype: int64

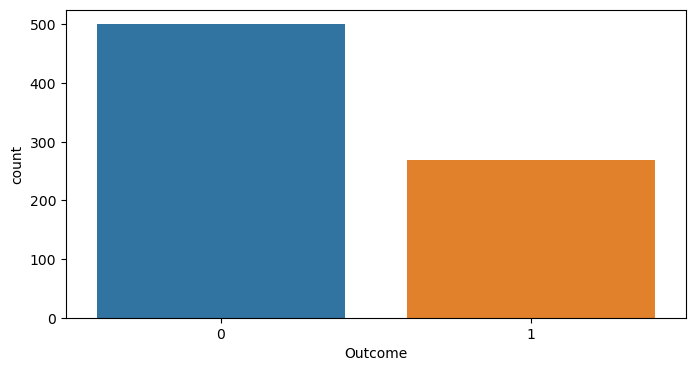
In [15]:

*# plot a counter plot to better understand our target feature*

plt.figure(figsize=(8,4))

sns.countplot(x = 'Outcome',data = df)

plt.show()



**Inference:**

* Our target varaible have only two class. 0 and 1. Here 0 and 1 represend not having diabetes and having diabetes respectilvey.
* We can see that we have inblacne dataset.
* We have to keep same ratio while we spliting our dataset into training and testing set.

In [16]:

*# Pregnancies*

plt.figure(figsize=(20,6))

plt.subplot(1,3,1)

plt.title("Counter Plot")

sns.countplot(x = 'Pregnancies',data = df)

plt.subplot(1,3,2)

plt.title('Distribution Plot')

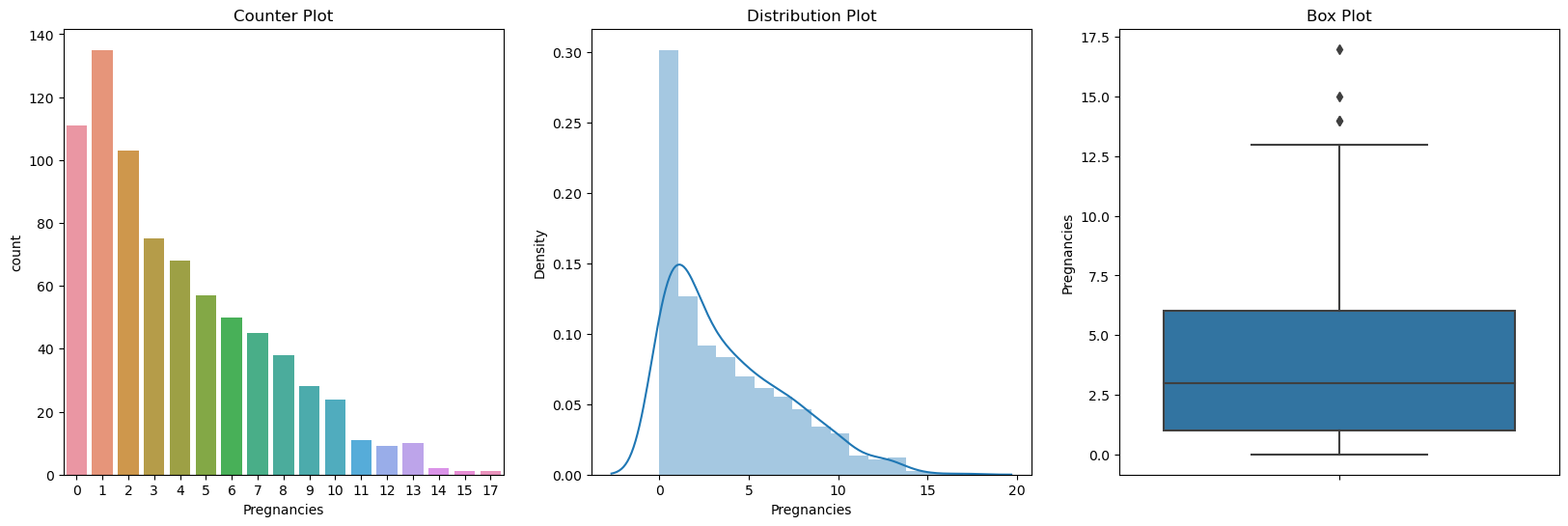
sns.distplot(df["Pregnancies"])

plt.subplot(1,3,3)

plt.title('Box Plot')

sns.boxplot(y=df["Pregnancies"])

plt.show()



In [17]:

*# Glucose*

plt.figure(figsize=(20,6))

plt.subplot(1,3,1)

plt.title("Counter Plot")

sns.countplot(x = 'Glucose',data = df)

plt.subplot(1,3,2)

plt.title('Distribution Plot')

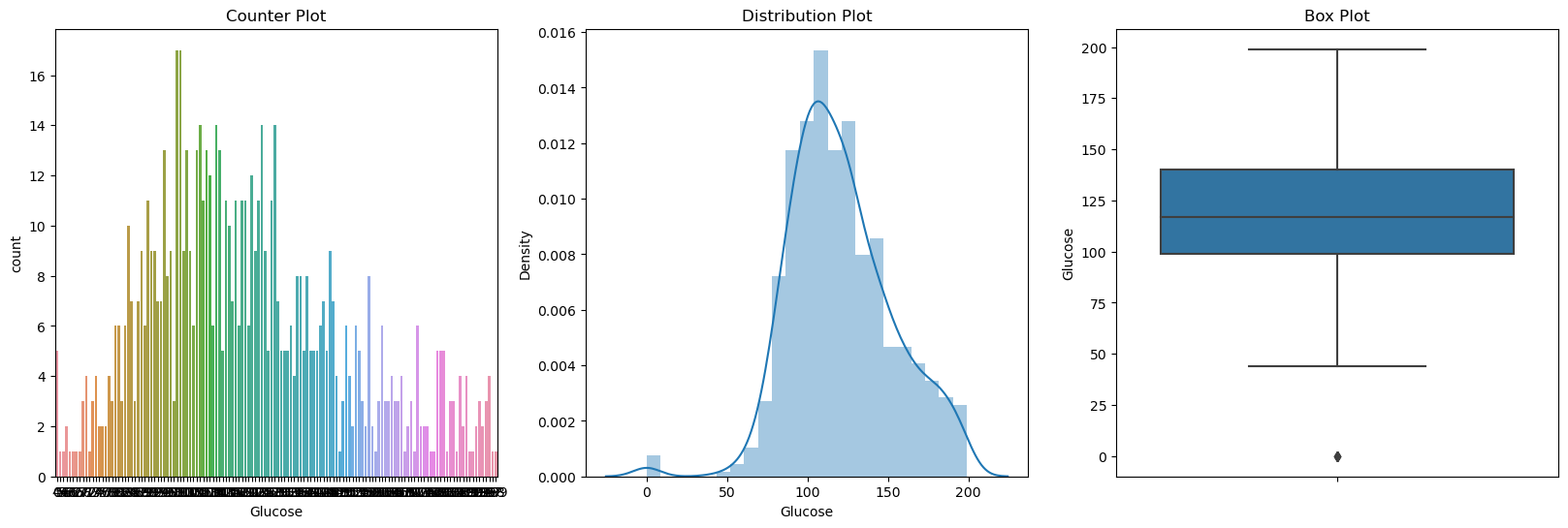
sns.distplot(df["Glucose"])

plt.subplot(1,3,3)

plt.title('Box Plot')

sns.boxplot(y=df["Glucose"])

plt.show()



In [18]:

*# BloodPressure*

plt.figure(figsize=(20,6))

plt.subplot(1,3,1)

plt.title("Counter Plot")

sns.countplot(x = 'BloodPressure',data = df)

plt.subplot(1,3,2)

plt.title('Distribution Plot')

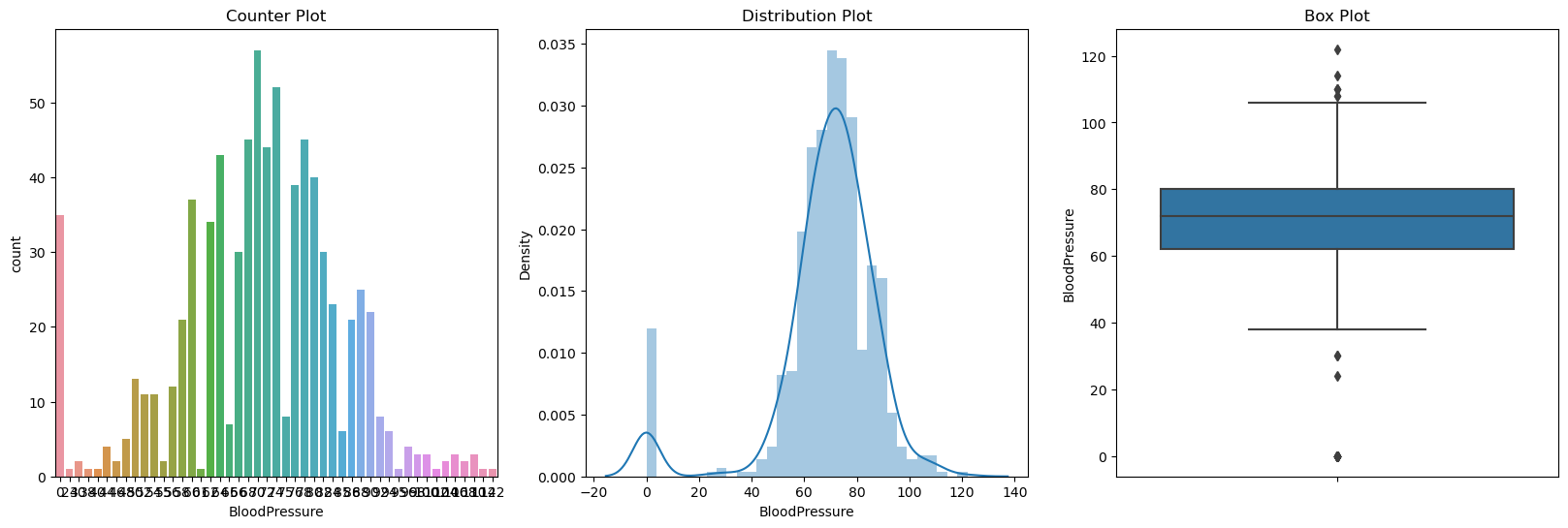
sns.distplot(df["BloodPressure"])

plt.subplot(1,3,3)

plt.title('Box Plot')

sns.boxplot(y=df["BloodPressure"])

plt.show()



In [19]:

*# BloodPressure*

plt.figure(figsize=(20,6))

plt.subplot(1,3,1)

plt.title("Counter Plot")

sns.countplot(x = 'BloodPressure',data = df)

plt.subplot(1,3,2)

plt.title('Distribution Plot')

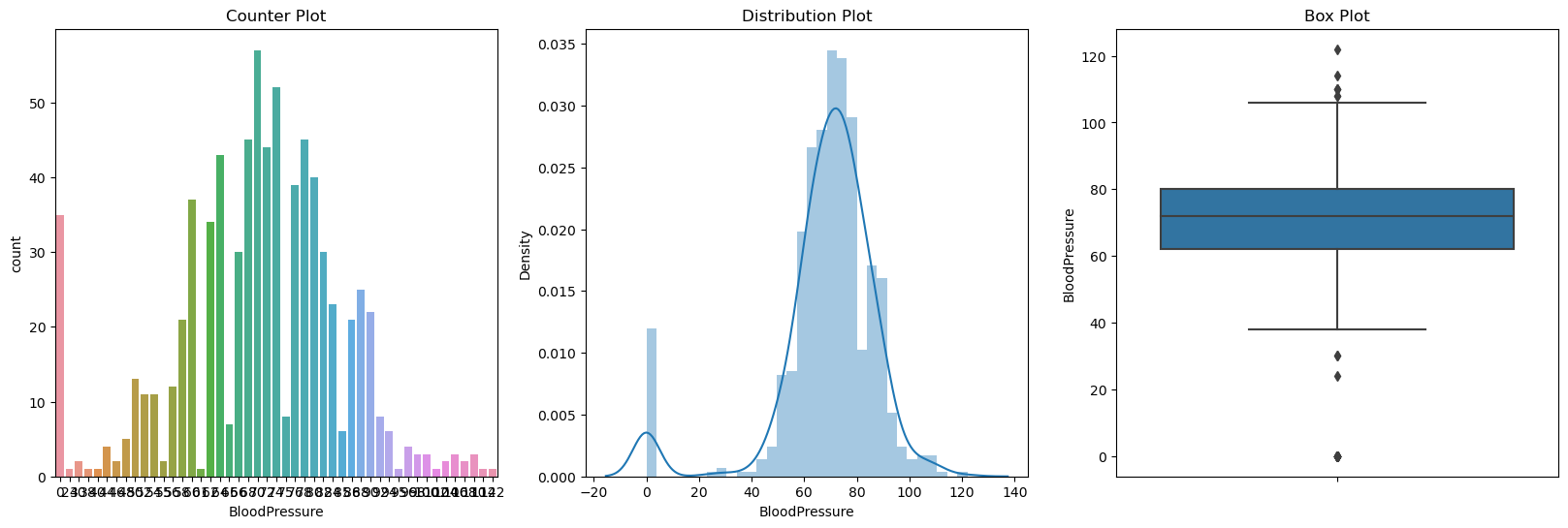
sns.distplot(df["BloodPressure"])

plt.subplot(1,3,3)

plt.title('Box Plot')

sns.boxplot(y=df["BloodPressure"])

plt.show()



In [20]:

*# SkinThickness*

plt.figure(figsize=(20,6))

plt.subplot(1,3,1)

plt.title("Counter Plot")

sns.countplot(x = 'SkinThickness',data = df)

plt.subplot(1,3,2)

plt.title('Distribution Plot')

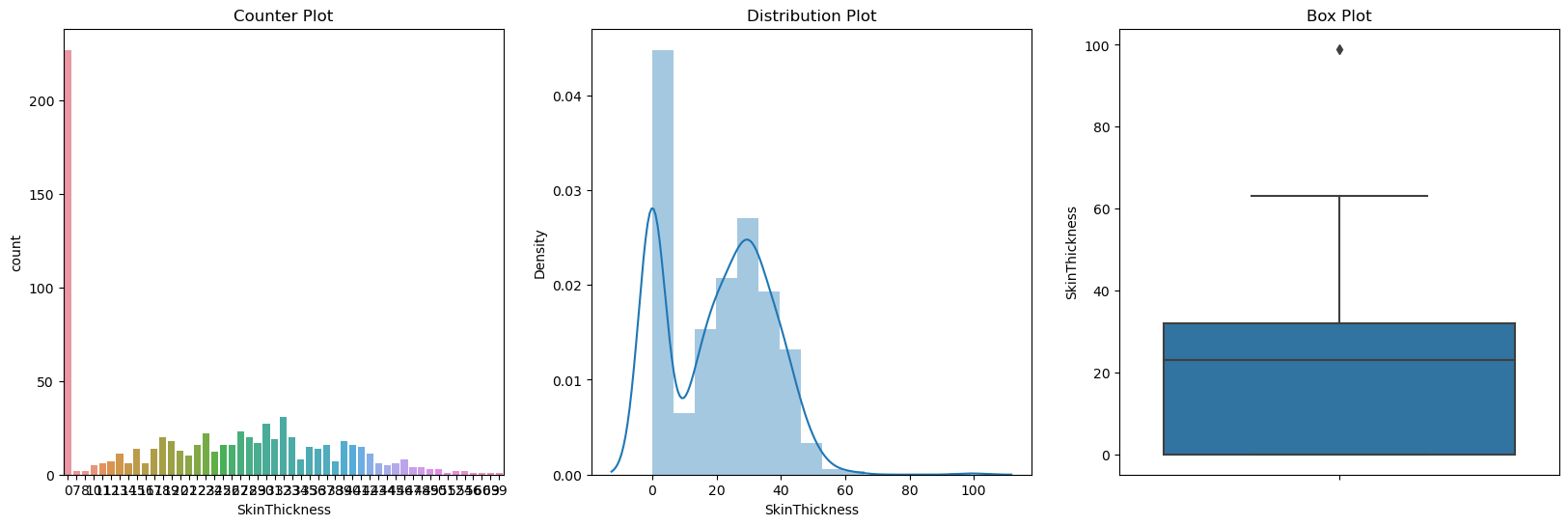
sns.distplot(df["SkinThickness"])

plt.subplot(1,3,3)

plt.title('Box Plot')

sns.boxplot(y=df["SkinThickness"])

plt.show()



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

In [21]:

*# Insulin*

plt.figure(figsize=(20,6))

plt.subplot(1,3,1)

plt.title("Counter Plot")

sns.countplot(x = 'Insulin',data = df)

plt.subplot(1,3,2)

plt.title('Distribution Plot')

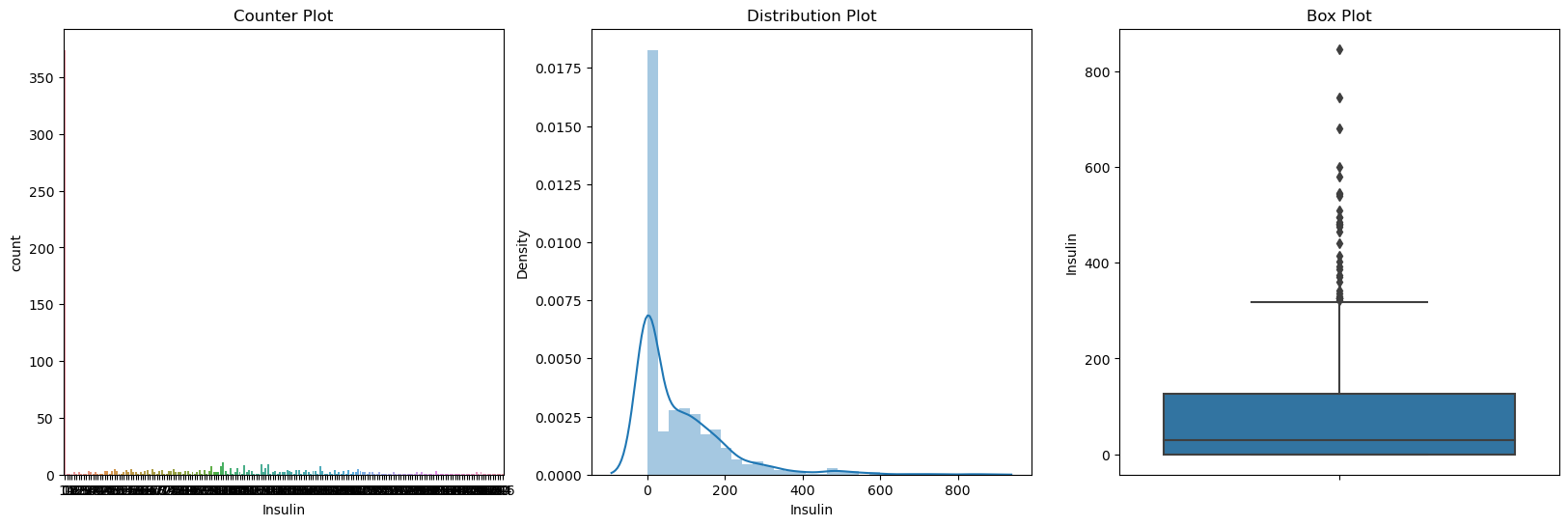
sns.distplot(df["Insulin"])

plt.subplot(1,3,3)

plt.title('Box Plot')

sns.boxplot(y=df["Insulin"])

plt.show()



In [22]:

*# BMI*

plt.figure(figsize=(20,6))

plt.subplot(1,3,1)

plt.title("Counter Plot")

sns.countplot(x = 'BMI',data = df)

plt.subplot(1,3,2)

plt.title('Distribution Plot')

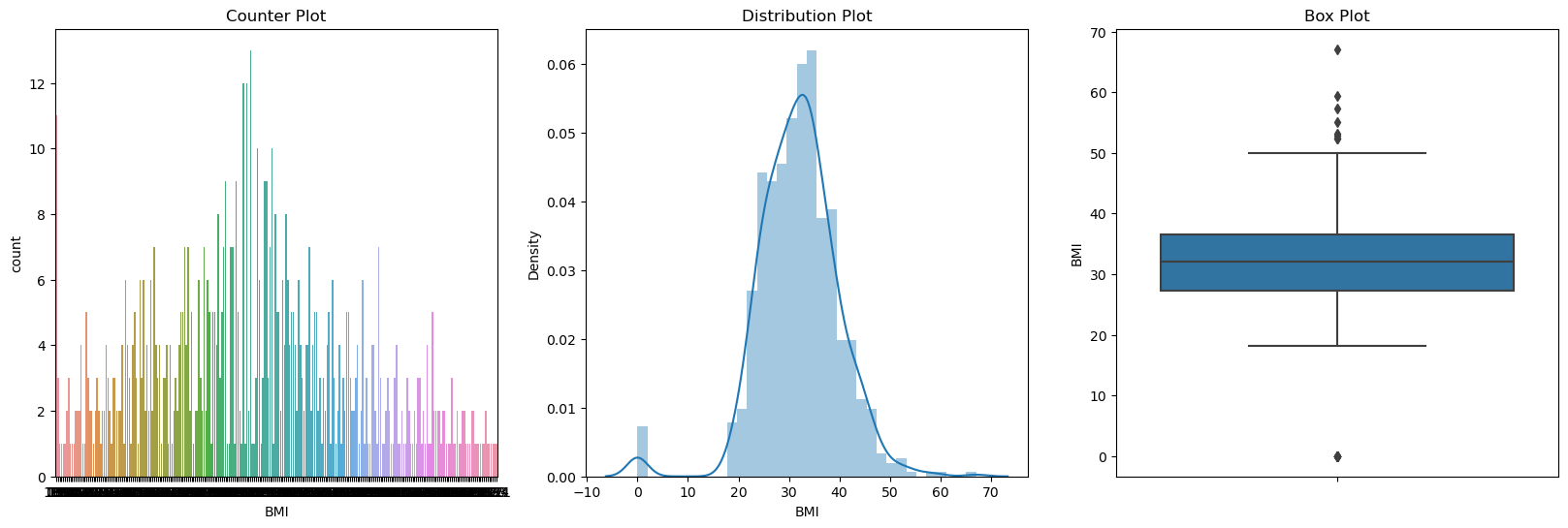
sns.distplot(df["BMI"])

plt.subplot(1,3,3)

plt.title('Box Plot')

sns.boxplot(y=df["BMI"])

plt.show()



# BIVERIANT ANALYSIS:

Now let's find the relationship between independent and dependent numerical features

In [23]:

*# Plotting the Relationship between target and features variables using scatter plot*

for feature **in** numerical\_features:

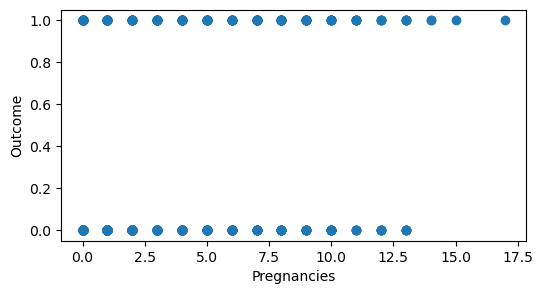
plt.figure(figsize=(6,3))

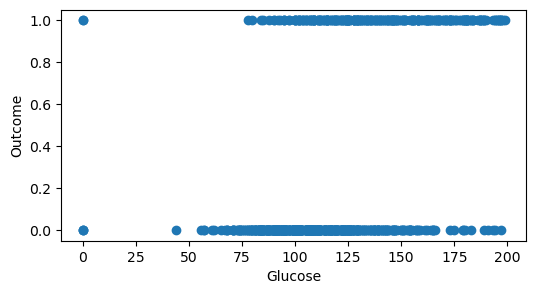
plt.scatter(y=df["Outcome"], x=df[feature])

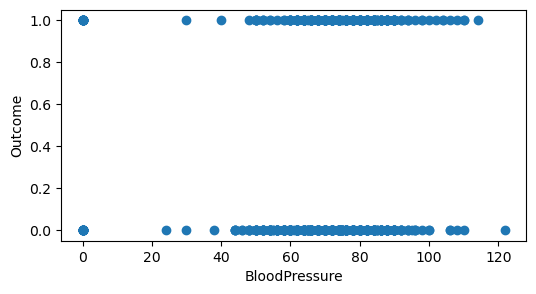
plt.ylabel("Outcome")

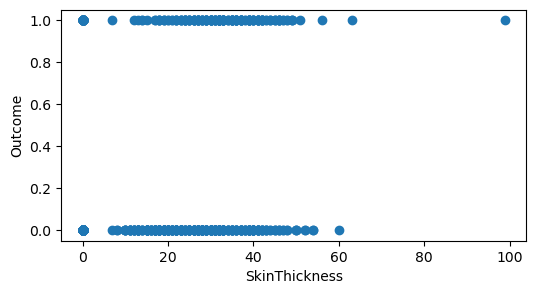
plt.xlabel(feature)

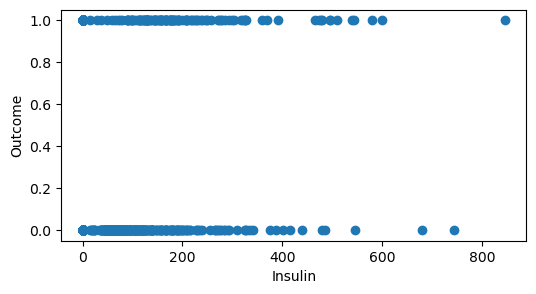
plt.show()

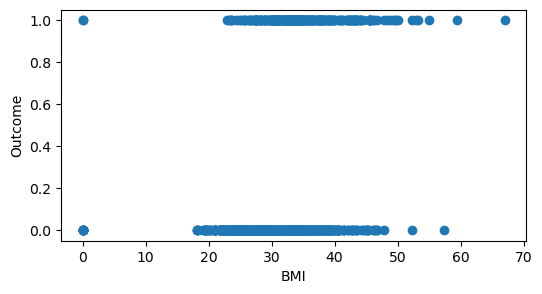


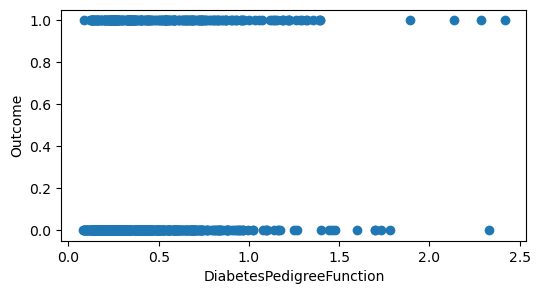


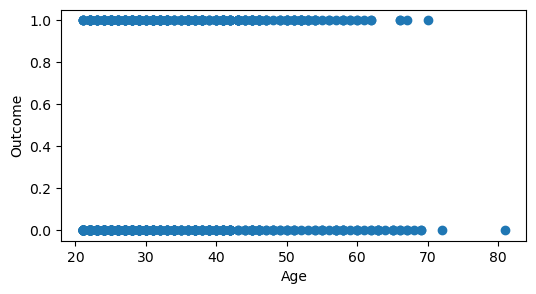


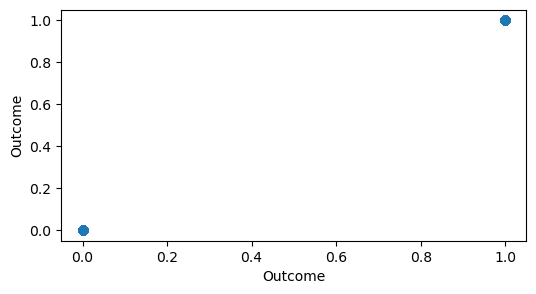










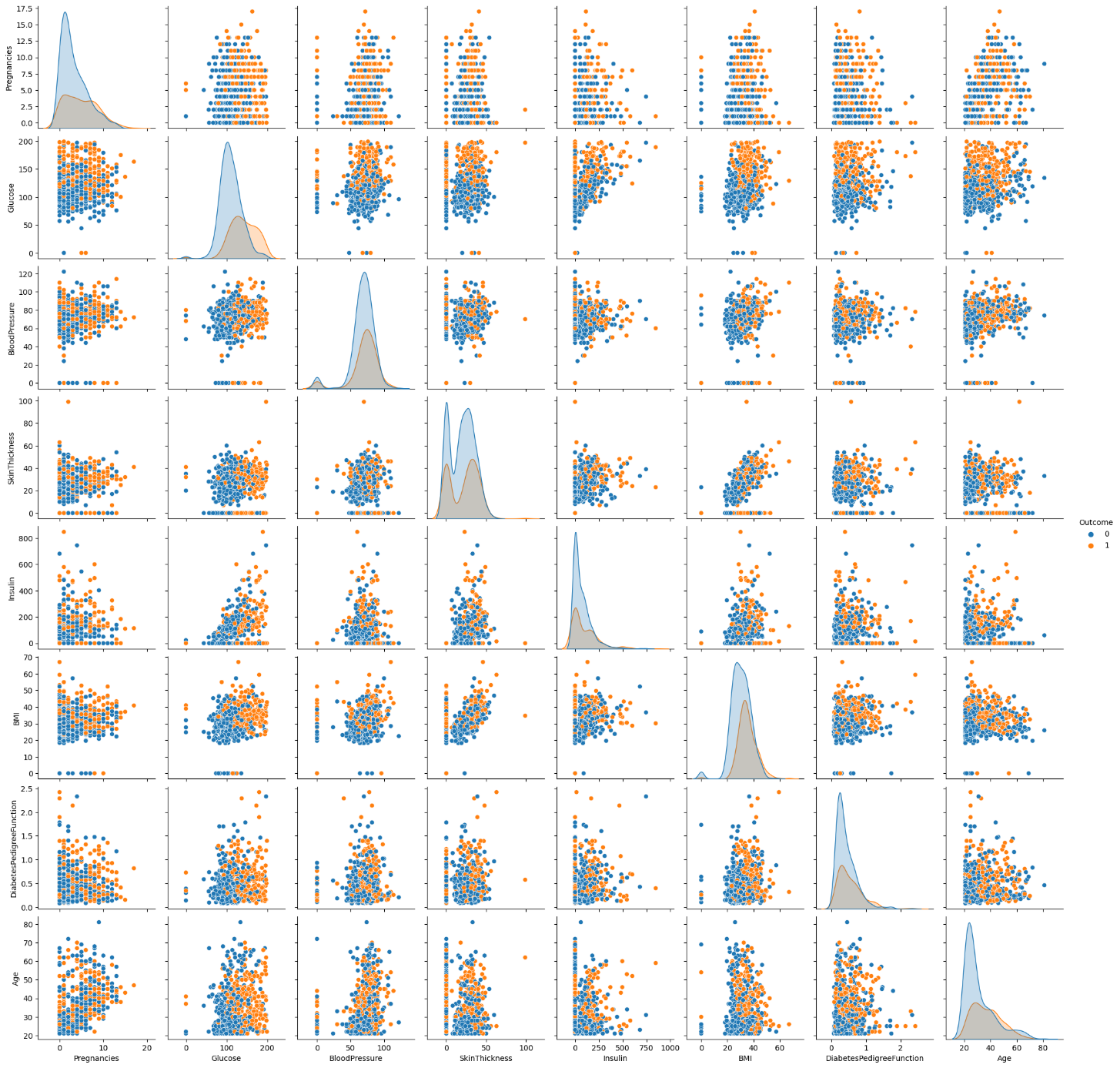


In [24]:

*# Finding the relationship of each fature with others*

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

plt.show()



A correlation matrix is a table showing correlation coefficients between sets of variables. Each random variable (Xi) in the table is correlated with each of the other values in the table (Xj). This allows you to see which pairs have the highest correlation.

In [25]:

*# Findin the correlation between each features*

corr = df.corr()

corr

Out[25]:

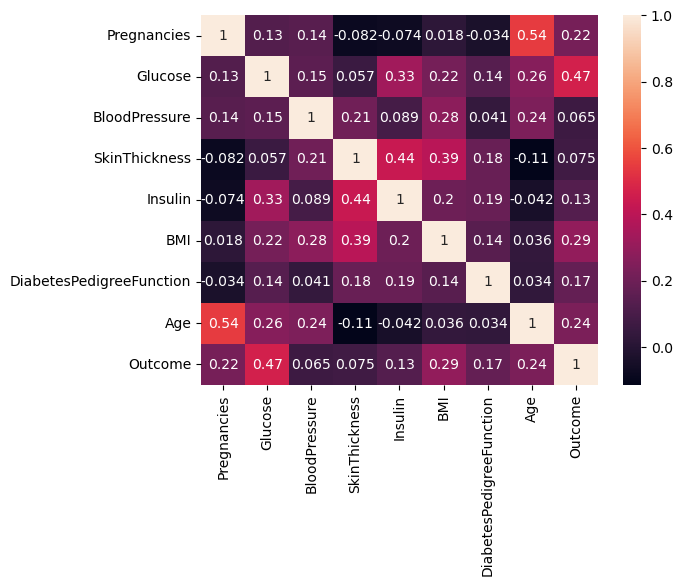
|  | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | DiabetesPedigreeFunction | Age | Outcome |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Pregnancies | 1.000000 | 0.129459 | 0.141282 | -0.081672 | -0.073535 | 0.017683 | -0.033523 | 0.544341 | 0.221898 |
| Glucose | 0.129459 | 1.000000 | 0.152590 | 0.057328 | 0.331357 | 0.221071 | 0.137337 | 0.263514 | 0.466581 |
| BloodPressure | 0.141282 | 0.152590 | 1.000000 | 0.207371 | 0.088933 | 0.281805 | 0.041265 | 0.239528 | 0.065068 |
| SkinThickness | -0.081672 | 0.057328 | 0.207371 | 1.000000 | 0.436783 | 0.392573 | 0.183928 | -0.113970 | 0.074752 |
| Insulin | -0.073535 | 0.331357 | 0.088933 | 0.436783 | 1.000000 | 0.197859 | 0.185071 | -0.042163 | 0.130548 |
| BMI | 0.017683 | 0.221071 | 0.281805 | 0.392573 | 0.197859 | 1.000000 | 0.140647 | 0.036242 | 0.292695 |
| DiabetesPedigreeFunction | -0.033523 | 0.137337 | 0.041265 | 0.183928 | 0.185071 | 0.140647 | 1.000000 | 0.033561 | 0.173844 |
| Age | 0.544341 | 0.263514 | 0.239528 | -0.113970 | -0.042163 | 0.036242 | 0.033561 | 1.000000 | 0.238356 |
| Outcome | 0.221898 | 0.466581 | 0.065068 | 0.074752 | 0.130548 | 0.292695 | 0.173844 | 0.238356 | 1.000000 |

In [26]:

*# Headmap*

sns.heatmap(corr, annot = True)

plt.show()



# FEATURES:

**Feature like Glucose, Blood Pressure, Skin Thickness, Insulin, BMI contains values as 0 which is not correct.  
So we change 0 with NaN and then replace nan value form our datasets.**

In [27]:

*# Replacing zero values with NaN*

dataset = df

dataset[["Glucose", "BloodPressure", "SkinThickness", "Insulin", "BMI"]] = dataset[["Glucose", "BloodPressure", "SkinThickness", "Insulin", "BMI"]].replace(0, np.NaN)

In [28]:

*# Count of NaN*

dataset.isnull().sum()

Out[28]:

Pregnancies 0

Glucose 5

BloodPressure 35

SkinThickness 227

Insulin 374

BMI 11

DiabetesPedigreeFunction 0

Age 0

Outcome 0

dtype: int64

In [29]:

*# Replacing NaN with mean values*

dataset["Glucose"].fillna(dataset["Glucose"].mean(), inplace = True)

dataset["BloodPressure"].fillna(dataset["BloodPressure"].mean(), inplace = True)

dataset["SkinThickness"].fillna(dataset["SkinThickness"].mean(), inplace = True)

dataset["Insulin"].fillna(dataset["Insulin"].mean(), inplace = True)

dataset["BMI"].fillna(dataset["BMI"].mean(), inplace = True)

In [30]:

dataset.isnull().sum()

Out[30]:

Pregnancies 0

Glucose 0

BloodPressure 0

SkinThickness 0

Insulin 0

BMI 0

DiabetesPedigreeFunction 0

Age 0

Outcome 0

dtype: int64

In [31]:

*# divide dataset into target and features varaibles*

y = dataset['Outcome']

X = dataset.drop('Outcome', axis=1)

**SPLIT DATAINOT TRAINING AND TEST SET:**

In [32]:

*# Spliting data into train and test set and also kept target data ration same*

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.2, random\_state = 42, stratify = y )

print("Shape of Training dataset: ", X\_train.shape)

print("ShaXpe of Testing dataset: ", X\_test.shape)

Shape of Training dataset: (614, 8)

ShaXpe of Testing dataset: (154, 8)

**FEATURE SCALING:**

In [33]:

*# Scaling our dataset*

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

scaler.fit(X\_train)

X\_train\_scaled = scaler.transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

# MODEL BUILDING AND PERFORMATION EVALUATION:

# CLASSIFICATION MODEL:

# 1) DECISION TREE

PERFORMANCE EVALUATION:

To measure the performance of a model, we need several elements :

This part is essential

**Confusion matrix** : also known as the error matrix, allows visualization of the performance of an algorithm :

* **true positive (TP)** : Diabetic correctly identified as diabetic
* **true negative (TN)** : Healthy correctly identified as healthy
* **false positive (FP)** : Healthy incorrectly identified as diabetic
* **false negative (FN)** : Diabetic incorrectly identified as healthy

**Metrics :**

* **Accuracy :** (TP +TN) / (TP + TN + FP +FN)
* **Recall :** TP / (TP + FN)
* **F1 score :** 2 x ((Precision x Recall) / (Precision + Recall))

# DECISION TREE:

In [34]:

*# Building Model For Decision Tree classifier*

from sklearn.tree import DecisionTreeClassifier

dt\_model = DecisionTreeClassifier(max\_depth=5, random\_state=42)

*# scalling doest affect decision tree*

dt\_model.fit(X\_test, y\_test)

y\_pred = dt\_model.predict(X\_test)

In [35]:

*# getting all types of accuracy for decision tree*

dt\_accuracy = accuracy\_score(y\_test,y\_pred)

dt\_recall = recall\_score(y\_test,y\_pred)

dt\_f1 = f1\_score(y\_test,y\_pred)

print("Accuracy: ", dt\_accuracy)

print("Recall: ", dt\_recall)

print("F1: ", dt\_f1)

Accuracy: 0.9285714285714286

Recall: 0.9074074074074074

F1: 0.8990825688073394

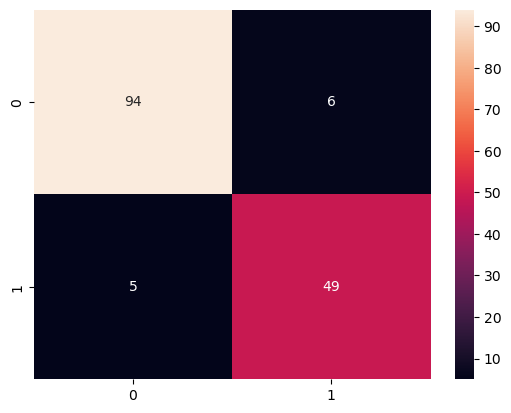
In [36]:

*# plotting counfusion metric for decision tree*

dt\_cm = confusion\_matrix(y\_test, y\_pred)

sns.heatmap(dt\_cm, annot=True)

plt.show()



In [37]:

*# printing overall report for decision tree*

print(classification\_report(y\_test, y\_pred))

precision recall f1-score support

0 0.95 0.94 0.94 100

1 0.89 0.91 0.90 54

accuracy 0.93 154

macro avg 0.92 0.92 0.92 154

weighted avg 0.93 0.93 0.93 154

DECISION TREE OUTPUT:

Here Decision Tree Performs best

Example: Let's check whether the person have diabetes or not using some random values

In [38]:

y\_predict = dt\_model.predict([[1,148,72,35,79.799,33.6,0.627,50]])

print(y\_predict)

if y\_predict==1:

print("Diabetic")

else:

print("Non Diabetic")

[1]

Diabetic

**CONCLUSION:**

* In conclusion, our model for innovation in the AI-based Diabetes Prediction System focuses on user-centric design, multidisciplinary collaboration, data security, continuous improvement, and seamless integration with healthcare institutions.
* By prioritizing user needs, embracing cutting-edge technology, and upholding ethical data practices, we aim to revolutionize diabetes care globally. This model ensures accuracy, usability, and adaptability, ultimately making a positive impact on diabetes prevention and management.