**AI-Based Diabetes Prediction System Project Report**

**Introduction**

The goal of this document is to outline the steps undertaken during the project implmentations.

**Step 1: Obtaining the Dataset**

I firstly created a local repository and initiated a Conda environment within this repository. After activating this environment I downloaded and installed the necessary libraries – pandas, numpy, matplotlib, scikit-learn, seaborn.

Later I opened the Jupyter notebook and created a new Python file. Also I downloaded the dataset which I have mentioned in the previous document from Kaggle and saved it inside my repository.

**Step 2: Data Preprocessing**

I started importing the necessary

# Import necessary libraries

import pandas as pd

import numpy as np

# Load the dataset

data = pd.read\_csv('diabetes\_data.csv')

# Check the first few rows of the dataset

print(data.head())

# Check for missing values

print(data.isnull().sum())

# Split the data into features (X) and target variable (y)

X = data.drop('Outcome’, axis=1)

y = data['Outcome']

**Step 3: Data Exploration (Optional)**

I used seaborn’s pairplot to plot all features (attributes) against one another. Upon doing so, it was found that the “Glucose” attribute plays an important role in predicting the outcome as it splits the data into precisely two clusters when compared to other attributes.

import matplotlib.pyplot as plt

import seaborn as sns

# Pairplot to visualize relationships between variables

sns.pairplot(data, hue='Outcome', diag\_kind='kde')

plt.show()

**Step 4: Data Splitting**

The next step is to split the dataset into training and test sets, I achieved this using the train\_test\_split function that Scikit Learn provides.

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

# Split the data into training and testing sets (80% train, 20% test)

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

**Step 5: Model Building (Gaussian NB)**

In the previous document, I have mentioned that I will be using Logistics Regression as the model. But upon comparing and contrasting different classification model, I found that Gaussian NB which uses Naïve Bayes Classification is better at predicting the result accurately than the Logistic Regression.

from sklearn.preprocessing import StandardScaler

from sklearn.naive\_bayes import GaussianNB

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix

# Create and train the GaussianNB

model = GaussianNB()

model.fit(X\_train, y\_train)

# Make predictions on the test set

y\_pred = model.predict(X\_test)

# Evaluate the model

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

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

report = classification\_report(y\_test, y\_pred)

print(f'Accuracy: {accuracy}')

print(f'Confusion Matrix:\n{confusion}')

print(f'Classification Report:\n{report}')

**Step 6: Model Deployment (Optional)**

To deploy the model, I used Flask since it is a Lightweight Python framework and is suitable for our small model. I created a simple user interface to get input data from the user and make prediction on the data and provide the output to the user.