**AI BASED DIABETES PREDICTION SYSTEM**

**PHASE-5**

**Description:**

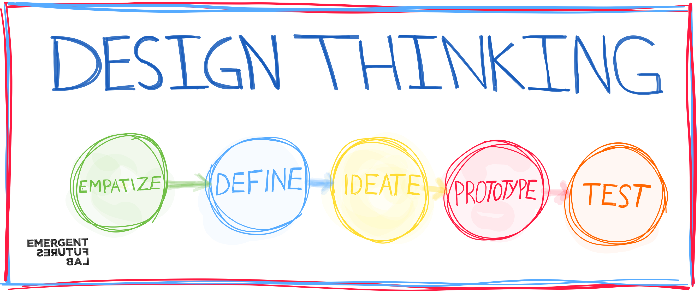
This document outlines the design and approach for creating an AI powered diabetes prediction system. The system’s primary objective is to analyse medical data and predict the likelihood of an individual developing diabetes using a machine learning algorithm.

**Problem Statement:**

The problem is to build an AI-powered diabetes prediction system that uses machine learning algorithms to analyse medical data and predict the likelihood of an individual developing diabetes. The system aims to provide early risk assessment and personalised preventive measures, allowing individuals to take proactive actions to manage their health.

**Design Thinking:**

The problem at hand is to develop an AI-powered diabetes prediction system that leverages machine learning algorithms to analyze extensive medical data. The primary objective of this system is to accurately assess an individual's risk of developing diabetes at an early stage, thus enabling the provision of personalized preventive measures. This early risk assessment will empower individuals to take proactive actions in managing their health and making informed decisions about their lifestyle choices.



* Design thinking is a problem-solving and innovation framework that emphasizes a human-centered approach to developing solutions. It involves a structured process that encourages creativity, empathy, and collaboration. Here are the key stages of the design thinking process:
* **1. Empathize:**
* - Understand the problem from the perspective of the people who are experiencing it.
* - Engage with users and stakeholders to gather insights, stories, and experiences.
* - Develop a deep sense of empathy for the target audience to understand their needs and concerns.
* **2. Define:**
* - Reframe the problem by synthesizing the information gathered during the empathy stage.
* - Identify specific needs and pain points of the users.
* - Create a clear and concise problem statement that guides the design process.
* **3. Ideate:**
* - Generate a wide range of creative ideas and potential solutions to address the defined problem.
* - Encourage brainstorming sessions and free-thinking to explore different possibilities.
* - Avoid judgment or critique of ideas during this phase to foster creativity.
* **4. Prototype:**
* - Create low-fidelity prototypes of the most promising ideas and solutions.
* - These prototypes can be sketches, wireframes, physical models, or other representations of the concepts.
* - Prototyping helps to visualize and test ideas quickly and affordably.
* **5. Test:**
* - Gather feedback by testing the prototypes with real users or stakeholders.
* - Evaluate the effectiveness of the solutions by observing user interactions and receiving feedback.
* - Use the insights from testing to refine and iterate on the prototypes.
* **6. Implement:**
* - Once a solution is validated through testing, develop a plan to implement it.
* - Collaborate with various stakeholders to bring the solution to life.
* - Ensure the solution is practical, feasible, and aligned with the initial problem statement.
* Design thinking is not always a linear process, and it often involves revisiting and iterating on previous stages as new insights and ideas emerge. This iterative approach helps to develop innovative, user-centric solutions and is commonly used in fields like product design, user experience design, and problem-solving in various domains.
* By focusing on empathy and understanding the needs of the end-users, design thinking aims to create solutions that are not only functional but also resonate with people on a meaningful level. It encourages a holistic and human-centric approach to problem-solving and innovation.

**Functionality:**

* + Define the scope clearly, listing the specific tasks the chatbot can perform. Common tasks include:
    - Answering frequently asked questions.
    - Providing product/service information.
    - Assisting with troubleshooting.
    - Guiding users through processes.
    - Handling basic inquiries and transactions.Implement NLP techniques to understand and process user input naturally. Key components include:
      * Tokenization: Breaking user input into words or phrases.
      * Part-of-speech tagging: Identifying the grammatical structure of sentences.
      * Named entity recognition: Extracting important entities (e.g., dates, locations, product names).
    - Escalating complex issues to human agents.
* **User Interface:**
  + Determine where the chatbot will be integrated (e.g., website, mobile app, messaging platform).
  + Design a user-friendly interface with a clear and accessible chat window.
  + Consider incorporating branding elements to maintain consistency with your organisation's image.
* **Natural Language Processing (NLP):**
  + - Intent recognition: Determining the user's purpose or query.
    - Sentiment analysis: Assessing the emotional tone of user messages.
    - Dialog management: Maintaining context and conversation flow.
* **Responses:**
  + Plan a variety of responses the chatbot will offer based on user queries. These responses should include:
    - Accurate answers to common questions.
    - Relevant suggestions or recommendations.
    - Step-by-step guidance and instructions.
    - Links to relevant resources (e.g., knowledge base articles, FAQs).
    - Polite and natural-sounding interactions.
    - Clear escalation procedures for complex issues.
    - Acknowledgment of user feedback and inquiries.
* **Integration:**
  + Decide how the chatbot will be integrated into your website or app:
    - Embedding a chat widget.
    - Integrating with messaging platforms (e.g., Facebook Messenger, Slack).
    - Providing an API for custom integrations.
  + Ensure seamless communication between the chatbot and your backend systems or databases for retrieving information.
* **Testing and Improvement:**
  + Develop a rigorous testing plan that covers various aspects of chatbot functionality:
    - Functional testing to verify core features work as intended.
    - Usability testing to evaluate the user interface and user experience.
    - NLP testing to assess the chatbot's understanding and response quality.
  + Collect and analyse user feedback to identify areas for improvement.
  + Continuously refine the chatbot's responses and performance based on real user interactions.
  + Implement A/B testing to compare different conversation flows or response strategies and determine which ones are more effective.

**Additionally, consider the following aspects:**

* User Education: Educate users on the chatbot's capabilities and limitations. Provide guidance on how to interact effectively with the chatbot.
* Privacy and Data Security: Ensure that user data is handled securely and that the chatbot complies with relevant data privacy regulations.
* Monitoring and Analytics: Implement monitoring and analytics tools to track user interactions, chatbot performance, and user satisfaction metrics.
* Regular Updates: Keep the chatbot's knowledge base up-to-date, reflecting changes in your products/services and addressing new user queries.

**Success criteria:**

The success of the AI-powered diabetes prediction system can be measured by its ability to accurately predict the chances of an individual developing diabetes using the medical data and machine learning algorithm.

**Constraints:**

* + Collaboration with Healthcare Professionals.
  + Plan for contingencies and redundancies in case of system failures, especially in critical healthcare scenarios.
  + Patient Consent.
  + Cost Constraints.
  + Data Quality and Availability.
  + Real-time Processing.

**Ethical Considerations:**

* + Fairness and Bias.
  + The system should be transparent, and its predictions should be explainable to healthcare professionals and patients.
  + Ensure that patient data is handled with the utmost privacy and security.
  + Informed Consent.

**Innovation:**

The innovation of an AI diabetes prediction system lies in its ability to harness advanced machine learning algorithms and big data analysis to predict the likelihood of an individual developing diabetes with unprecedented accuracy. This system not only considers traditional risk factors but also incorporates real-time health data, genetic information, and lifestyle choices to provide personalised predictions and recommendations. By continuously learning and adapting, it empowers individuals and healthcare professionals to take proactive steps in diabetes prevention, ultimately improving public health outcomes and reducing the burden of this chronic disease.

**Advanced Machine Learning Algorithms:**

Continuous innovation in machine learning algorithms, such as deep learning and reinforcement learning, can lead to more accurate and efficient diabetes prediction models. These algorithms can handle complex, multi-dimensional data and extract meaningful patterns and insights.

**Integration of Multiple Data Sources:**

Innovations in data integration techniques can enable the incorporation of diverse data sources, including genetic, clinical, wearable device, and environmental data, to provide a more comprehensive view of an individual's diabetes risk.

**Real-time Predictive Analytics:**

The development of real-time predictive analytics systems can enable instant risk assessment and proactive interventions. This innovation is crucial for patients with diabetes, as it allows for immediate feedback and timely management of glucose levels.

**Personalised Medicine:**

Innovations in personalised medicine within the context of diabetes prediction involve tailoring recommendations and interventions to an individual's unique genetic and lifestyle factors. AI can play a pivotal role in providing personalised guidance for diabetes prevention and management.

**User-Friendly Interfaces:**

Innovations in user interfaces and user experience design can make diabetes prediction and management tools more accessible to both healthcare providers and patients. Mobile apps, chatbots, and other user-friendly platforms can facilitate engagement and adherence to recommendations.

**Telemedicine and Remote Monitoring:**

The integration of AI with telemedicine and remote monitoring technologies can empower healthcare providers to remotely monitor and manage diabetes patients. This innovation can extend the reach of diabetes care to underserved areas and improve patient outcomes.

**Ethical and Privacy Considerations:**

Innovations in ethical and privacy considerations are essential, especially when dealing with sensitive health data. Developing robust data anonymization, encryption, and consent management systems is crucial to protect patient information.

**Interoperability and Data Sharing:**

Innovations in interoperability standards and data sharing mechanisms can facilitate the exchange of health data between different healthcare providers and systems. This promotes more comprehensive diabetes prediction and care.

**AI-Enabled Drug Discovery:**

AI-driven drug discovery and development can lead to the creation of more effective medications for diabetes prevention and treatment. Innovations in this area can result in novel therapeutic options.

**Collaborative Research Initiatives:**

Encouraging collaboration among researchers, healthcare providers, technology companies, and patient advocacy groups can foster innovation in diabetes prediction. Multidisciplinary efforts can lead to breakthroughs and the integration of new technologies and insights.Innovation in AI-based diabetes prediction is essential for staying at the forefront of healthcare advancements. These innovations not only enhance the accuracy of predictions but also improve patient outcomes, reduce the economic burden of diabetes care, and promote proactive health management. Additionally, ethical and privacy considerations remain central to the development and deployment of these innovations to ensure the responsible use of health data.

AI-based diabetes prediction is a sophisticated application of artificial intelligence (AI) and machine learning that aims to predict the likelihood of an individual developing diabetes or their risk of complications associated with diabetes. Here is a detailed description:

Artificial Intelligence (AI) for Diabetes Prediction:

In the realm of healthcare, AI has emerged as a powerful tool for early diagnosis and risk assessment. When applied to diabetes, AI-based prediction systems leverage advanced algorithms and data analysis techniques to provide invaluable insights into a patient's susceptibility to diabetes or their potential complications. The following components define AI-based diabetes prediction:

**Data Collection and Integration:**

AI diabetes prediction systems begin by aggregating extensive datasets. These datasets can include patient medical records, lifestyle information, genetic data, and biomarkers like blood glucose levels, HbA1c measurements, and insulin sensitivity.

**Feature Selection:**

The AI system identifies the most relevant features or variables from the collected data, such as family history, age, BMI, dietary habits, physical activity, and more. This step helps in reducing noise and improving the accuracy of predictions.

**Machine Learning Algorithms:**

Various machine learning algorithms are then employed to analyse and model the data. Common algorithms include logistic regression, decision trees, random forests, support vector machines, and deep learning neural networks.

**Training and Validation:**

The AI model is trained on historical data that includes records of both diabetic and non-diabetic patients. It's crucial to validate the model's performance using separate datasets to ensure its reliability.

**Risk Assessment**:

The AI system assigns a risk score to individuals based on the input data. This score indicates their likelihood of developing diabetes in the future or the risk of diabetes-related complications. This information can help doctors and patients take preventive measures.

**Personalised Recommendation:**

AI systems can also provide personalised recommendations for individuals, including diet and lifestyle changes, regular check-ups, and early intervention strategies to reduce the risk of diabetes.

**Real-time Monitoring:**

For patients already diagnosed with diabetes, AI can continuously monitor their health and provide real-time feedback. For instance, it can alert them to fluctuating glucose levels and suggest necessary actions.

**Remote Health Monitoring:**

With the rise of wearable devices and remote monitoring solutions, AI can access real-time data from patients and provide instant feedback or alert healthcare professionals when intervention is needed.

**Research and Drug Development:**

AI can also assist in diabetes research by analysing vast volumes of medical literature and clinical trial data to identify potential treatments or interventions.

**Visualisation Using Matplotlib:**

Matplotlib is a comprehensive library for creating data visualisations in Python. In addition to the basic Matplotlib library, there are several sub-libraries and modules that provide additional functionality and customization for your plots and charts. Here are some commonly used Matplotlib libraries and sub-modules:

Here's an example of how you might use some of these libraries and modules:

***import matplotlib.pyplot as plt***

***import matplotlib.patches as patches***

***import matplotlib.colors as mcolors***

***import matplotlib.legend as mlegend***

***import matplotlib.axes as axes***

***import matplotlib.ticker as ticker***

***import matplotlib.gridspec as gridspec***

***import matplotlib.animation as animation***

***import matplotlib.text as text***

The specific libraries and modules you use will depend on your visualization needs. For basic plots and charts, `matplotlib.pyplot` is usually sufficient, but for more complex and customised visualizations, you may explore the other Matplotlib libraries and modules.

**Loading and Preprocessing Dataset**

**Development part:**

-To load, preprocess the dataset and perform different analysis

**1.Download dataset:**

-Download the dataset from the provided Kaggle link:

https://www.kaggle.com/datasets/mathchi/diabetes-data-set

**2.Import necessary Libraries:**

-Use Python and import libraries such as numpy,pandas,matplotlib and scikit-learn for data manipulation and visualization.

***import numpy as np***

***import pandas as pd***

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

***from sklearn.preprocessing import StandardScaler***

***from sklearn.linear\_model import LogisticRegression***

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

**3.Load the dataset:**

-Read the dataset into a pandas dataframe.

***from google.colab import files***

***uploaded = files.upload()***

***import pandas as pd***

***import io***

***df = pd.read\_csv(io.BytesIO(uploaded['diabetes.csv']))***

***print(df)***

**4.Explore the dataset:**

-Check the first few rows of the dataset to understand what we are dealing with.

-Examine the column names and data types.

***print(df.head())***

***print(df.info())***

**5.Spilting the data:**

-Split the dataset into training and testing sets:

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

***X = df.drop('target', axis=1)***

***y = df['target']***

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

**6.Model Training:**

-Train a machine learning model to predict diabetes. For example, you can use a linear regression model:

***from sklearn.linear\_model import LinearRegression***

***model = LinearRegression()***

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

**7.Model Evaluation:**

-Evaluate the model's performance:

***from sklearn.metrics import mean\_squared\_error, r2\_score***

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

***mse = mean\_squared\_error(y\_test, y\_pred)***

***r2 = r2\_score(y\_test, y\_pred)***

***print(f"Mean Squared Error: {mse}")***

***print(f"R-squared: {r2}")***

**Data Preprocessing and Exploration:**

Data preprocessing is a critical step in the machine learning and data analysis pipeline. It involves cleaning, transforming, and organising raw data into a format suitable for training machine learning models.

Data preprocessing can be a time-consuming and iterative process, but it is crucial for ensuring that your machine learning model can effectively learn from the data. The specific preprocessing steps you need to perform depend on the nature of your data, the problem you're trying to solve, and the algorithms you plan to use.

Here we use some machine learning algorithms to develop AI based diabetes prediction systems.Before building a machine learning model,we need to prepare the data.

Import the necessary libraries like numpy,pandas,scikit-learn and so on. And handle some missing data,if any.Split the dataset into training and testing sets.

#Importing the given dataset

from google.colab import files

uploaded=files.upload()

import pandas as pd

import io

df=pd.read\_csv(io.BytesIO(uploaded['diabetes.csv']))

# Import necessary libraries

import numpy as np

import pandas as pd

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

from sklearn.preprocessing import StandardScaler

from sklearn.linear\_model import LogisticRegression

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

from sklearn.naive\_bayes import GaussianNB

from sklearn.ensemble import RandomForestClassifier

import matplotlib.pyplot as plt

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

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error, r2\_score

**Feature Selection:**

Feature selection is the process of isolating the most consistent, non-redundant, and relevant features to use in model construction.

X = df[['BloodPressure' ]]

y = df[['Age']]

**Model Selection:**

Artificial Intelligence (AI) and machine learning encompass a wide range of algorithms and techniques. These algorithms can be categorized into several common types based on their functionality and application. Here are some popular AI and machine learning algorithms:

**Linear Regression** :

Used for regression tasks, linear regression models the relationship between a dependent variable and one or more independent variables by fitting a linear equation.

**Logistic Regression:**

Used for classification tasks, logistic regression models the probability of a binary outcome.

**Decision Trees:**

A tree-like structure that makes decisions by splitting data into branches based on features. Decision trees can be used for both classification and regression.

**Random Forest:**

An ensemble learning method that combines multiple decision trees to improve accuracy and reduce overfitting.

**Support Vector Machines:**

Used for classification and regression, SVM finds a hyperplane that best separates data into distinct classes.

**K-Nearest Neighbors (K-NN):**

A simple instance-based learning algorithm for classification and regression that classifies data points based on the majority class of their k-nearest neighbours.

**Naive Bayes:**

A probabilistic algorithm based on Bayes' theorem used for classification tasks, particularly in text and document classification.

**Model Training:**

Model training is a crucial step in the development of machine learning and AI models. It involves using a dataset to teach a model to make predictions or decisions. The trained model learns patterns, relationships, and characteristics within the data, allowing it to generalise and make predictions on new, unseen data.

# Train the model on the training data

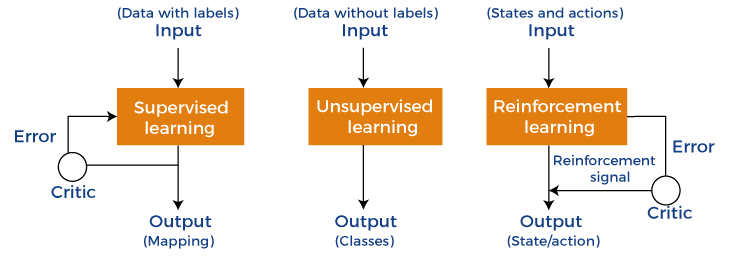
model.fit(X\_train, y\_train)

The process of training a machine learning model is iterative and may involve multiple cycles of data preprocessing, model selection, and tuning. The goal is to develop a model that generalises well to new, unseen data and provides valuable insights or predictions for your specific application.

### **Classification of Model Training:**

### Based on different business goals and data sets, there are three learning models for algorithms. Each machine learning algorithm settles into one of the three models:

* **Supervised Learning**
* **Unsupervised Learning**
* **Reinforcement Learning**

****

**Model Evaluation:**

Model evaluation is the process of using different evaluation metrics to understand a machine learning model's performance, as well as its strengths and weaknesses.

**# Make predictions on the test data**

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

**# Calculate accuracy and other classification metrics**

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

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

**# Print the results**

**print(f'Accuracy: {accuracy}')**

**print('Classification Report:\n', report)**

**Visualisation :**

Data visualisation is a crucial aspect of machine learning that enables analysts to understand and make sense of data patterns, relationships, and trends. Through data visualisation, insights and patterns in data can be easily interpreted and communicated to a wider audience, making it a critical component of machine learning. In this article, we will discuss the significance of data visualisation in machine learning, its various types, and how it is used in the field.

# Plot the training data

plt.scatter(X\_train, y\_train, label='Training Data')

plt.plot(X\_test, y\_pred, color='red', linewidth=3, label='Regression Line')

plt.xlabel('X')

plt.ylabel('y')

plt.legend()

plt.show()

**CODE:**

**LINEAR REGRESSION MODEL:**

***X = df[['BloodPressure' ]]***

***y = df[['Age']]***

***# Split the data into training and testing sets***

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

***# Create a linear regression model***

***model = LinearRegression()***

***# Train the model on the training data***

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

***# Make predictions on the test data***

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

***# Calculate the mean squared error and R-squared (coefficient of determination)***

***mse = mean\_squared\_error(y\_test, y\_pred)***

***r2 = r2\_score(y\_test, y\_pred)***

***# Print the results***

***print(f'Mean Squared Error: {mse}')***

***print(f'R-squared: {r2}')***

***# Plot the training data and the regression line***

***plt.scatter(X\_train, y\_train, label='Training Data')***

***plt.plot(X\_test, y\_pred, color='red', linewidth=3, label='Regression Line')***

***plt.xlabel('X')***

***plt.ylabel('y')***

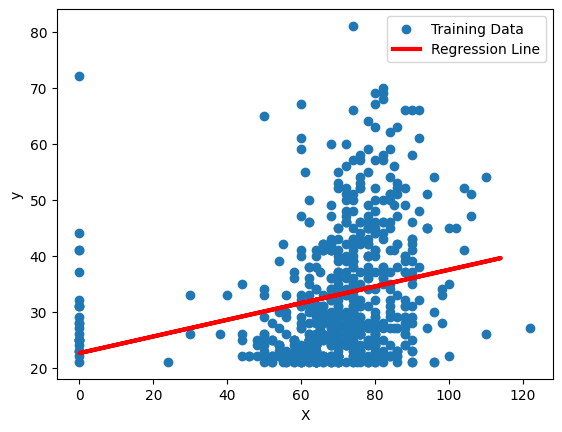
***plt.legend()***

***plt.show()***

**Output:**

Mean Squared Error: 153.53893981827423

R-squared: 0.039590936856064185

****

**NAIVE BAYES CLASSIFIER MODEL:**

***X = df[['Glucose' ]]***

***y = df[['Age']]***

***# Split the data into training and testing sets***

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

***# Create a Gaussian Naive Bayes model***

***model = GaussianNB()***

***# Train the model on the training data***

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

***# Make predictions on the test data***

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

***# Calculate accuracy and other classification metrics***

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

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

***# Print the results***

***print(f'Accuracy: {accuracy}')***

***print('Classification Report:\n', report)***

**Output:**

**Accuracy: 0.07792207792207792**

**Classification Report:**

**precision recall f1-score support**

**21 0.09 0.50 0.15 12**

**22 0.08 0.38 0.13 13**

**23 0.00 0.00 0.00 7**

**24 0.00 0.00 0.00 10**

**25 0.00 0.00 0.00 8**

**26 0.00 0.00 0.00 3**

**27 0.00 0.00 0.00 5**

**28 0.00 0.00 0.00 9**

**29 0.00 0.00 0.00 10**

**30 0.00 0.00 0.00 4**

**31 0.00 0.00 0.00 3**

**32 0.00 0.00 0.00 4**

**33 0.00 0.00 0.00 2**

**34 0.00 0.00 0.00 2**

**36 0.00 0.00 0.00 4**

**37 0.00 0.00 0.00 3**

**38 0.00 0.00 0.00 6**

**39 0.00 0.00 0.00 4**

**40 0.00 0.00 0.00 2**

**41 0.11 0.33 0.17 3**

**42 0.00 0.00 0.00 4**

**43 0.00 0.00 0.00 4**

**44 0.00 0.00 0.00 3**

**45 0.00 0.00 0.00 2**

**48 0.00 0.00 0.00 1**

**49 0.00 0.00 0.00 1**

**50 0.00 0.00 0.00 2**

**51 0.00 0.00 0.00 1**

**53 0.00 0.00 0.00 2**

**54 0.00 0.00 0.00 2**

**55 0.00 0.00 0.00 1**

**56 0.00 0.00 0.00 1**

**57 0.00 0.00 0.00 1**

**58 0.00 0.00 0.00 4**

**60 0.00 0.00 0.00 3**

**61 0.00 0.00 0.00 0**

**62 0.00 0.00 0.00 3**

**63 0.00 0.00 0.00 2**

**64 0.00 0.00 0.00 0**

**65 0.00 0.00 0.00 2**

**67 0.00 0.00 0.00 1**

**68 0.00 0.00 0.00 0**

**70 0.00 0.00 0.00 0**

**72 0.00 0.00 0.00 0**

**81 0.00 0.00 0.00 0**

**accuracy 0.08 154**

**macro avg 0.01 0.03 0.01 154**

**weighted avg 0.02 0.08 0.03 154**

**GAUSSIAN NB MODEL:**

***import numpy as np***

***import pandas as pd***

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

***from sklearn.naive\_bayes import GaussianNB***

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

***# Load the dataset (change the file path to your dataset)***

***data = pd.read\_csv("diabetes.csv")***

***# Split the dataset into features and labels***

***X = data.drop("Outcome", axis=1)***

***y = data["Outcome"]***

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

***nb\_model = GaussianNB()***

***nb\_model.fit(X\_train, y\_train)***

***y\_pred = nb\_model.predict(X\_test)***

***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("Confusion Matrix:")***

***print(confusion)***

***print("Classification Report:")***

***print(report)***

**Output:**

Accuracy: 0.7662337662337663

Confusion Matrix: [[79 20] [16 39]]

Classification Report: precision recall f1-score support 0 0.83 0.80 0.81 99 1 0.66 0.71 0.68 55 accuracy 0.77 154 macro avg 0.75 0.75 0.75 154 weighted avg 0.77 0.77 0.77 154

**BERNOULLI NB MODEL:**

***import numpy as np***

***import pandas as pd***

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

***from sklearn.naive\_bayes import BernoulliNB***

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

***# Load the dataset (change the file path to your dataset)***

***data = pd.read\_csv("diabetes.csv")***

***# Split the dataset into features and labels***

***X = data.drop("Outcome", axis=1)***

***y = data["Outcome"]***

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

***bnb\_model = BernoulliNB()***

***bnb\_model.fit(X\_train, y\_train)***

***y\_pred = bnb\_model.predict(X\_test)***

***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("Confusion Matrix:")***

***print(confusion)***

***print("Classification Report:")***

***print(report)***

**Output:**

Accuracy: 0.6558441558441559 Confusion Matrix: [[98 1] [52 3]] Classification Report: precision recall f1-score support 0 0.65 0.99 0.79 99 1 0.75 0.05 0.10 55 accuracy 0.66 154 macro avg 0.70 0.52 0.44 154 weighted avg 0.69 0.66 0.54 154

**SUPPORT VECTOR MODEL:**

***from sklearn import datasets***

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

***from sklearn.preprocessing import StandardScaler***

***from sklearn.svm import SVC***

***from sklearn.metrics import accuracy\_score***

***# Load the diabetes dataset***

***diabetes = datasets.load\_diabetes()***

***X = diabetes.data***

***y = diabetes.target***

***# Split the dataset into training and testing sets***

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

***scaler = StandardScaler()***

***X\_train = scaler.fit\_transform(X\_train)***

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

***# Create an SVC classifier***

***clf = SVC(kernel='linear', C=1.0)***

***# Train the classifier***

***clf.fit(X\_train, y\_train)***

***# Make predictions on the test set***

***y\_pred = clf.predict(X\_test)***

***# Calculate the accuracy***

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

***print(f"Accuracy: {accuracy \* 100:.2f}%")***

**Output:**

***Accuracy: 1.12%***

**Visualization:**

**CORRELATION:**

***#Correlation between features***

***import seaborn as sns***

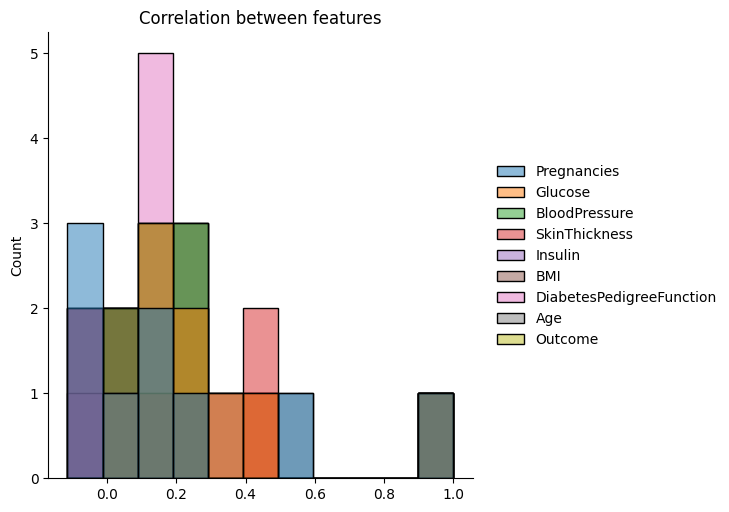
***import matplotlib.pyplot as plt***

***c=data.corr()***

***sns.displot(c)***

***plt.title("Correlation between features")***

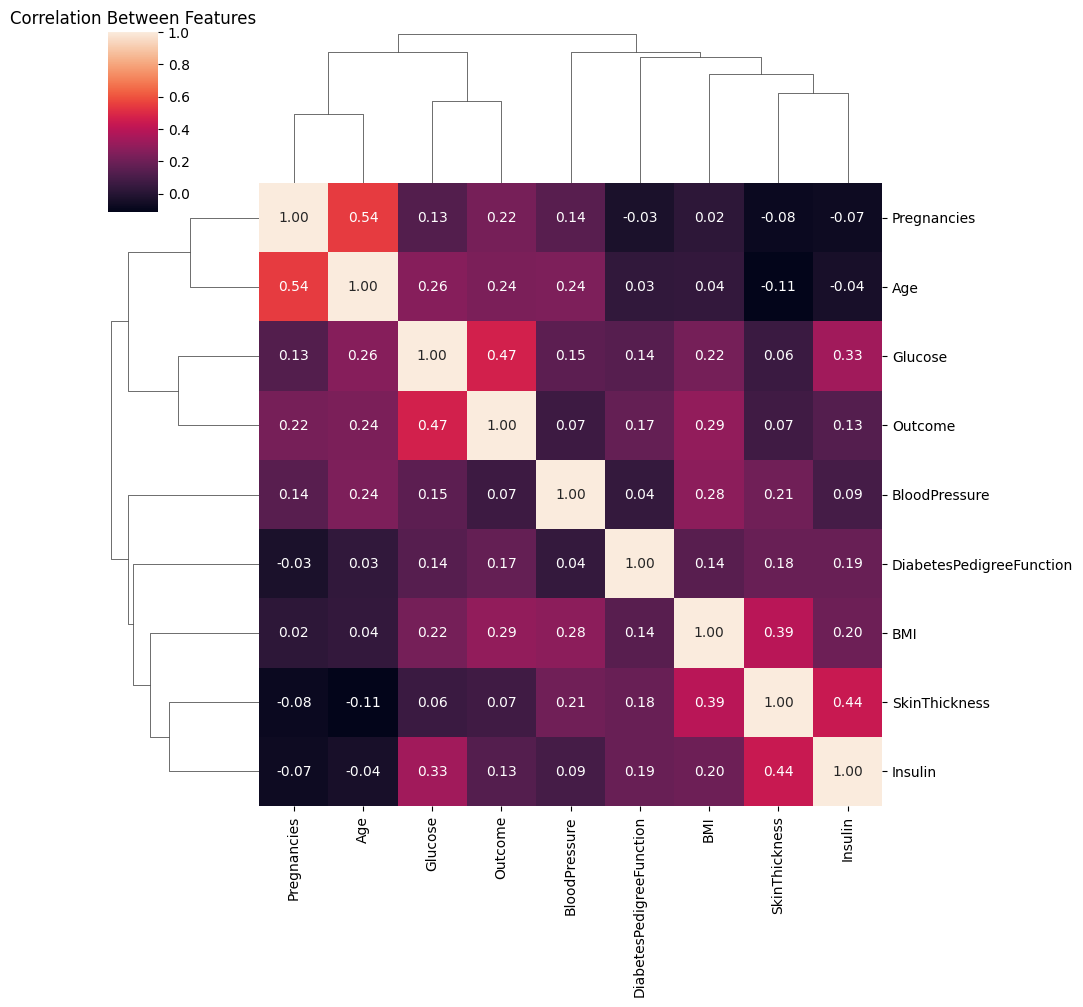
***plt.show()***

**Output:**

***#Correlation between features using clustermap***

***Corr\_matrix = data.corr()***

***sns.clustermap(Corr\_matrix, annot = True, fmt = '.2f')***

***plt.title("Correlation Between Features")plt.show()***

**FEATURE ENGINEERING:**

***import matplotlib.pyplot as plt***

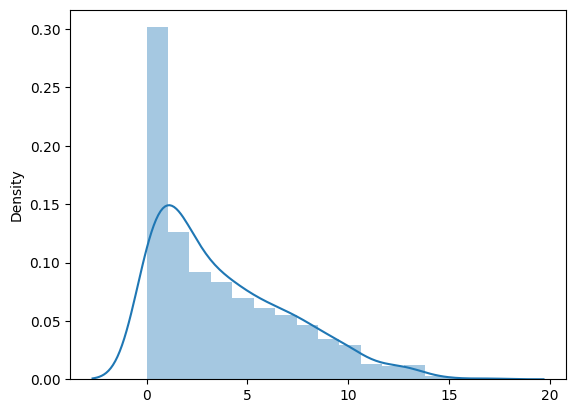
***for i in data.columns:***

***plt.figure()***

***sns.distplot(data[i], fit = 'norm')***

***plt.show()***

**Output:**



**CONCLUSION:**

In conclusion, an AI-based diabetes prediction system holds great promise in revolutionizing the way we approach the diagnosis and management of diabetes. By harnessing the power of machine learning and predictive analytics, such a system can offer several significant advantages.First and foremost, an AI-based diabetes prediction system can help in early detection and risk assessment. By analyzing vast amounts of patient data, including medical records, genetics, and lifestyle factors, it can identify individuals at high risk of developing diabetes, enabling healthcare providers to intervene with preventative measures before the condition progresses.Furthermore, AI can enhance the accuracy of diabetes diagnosis. It can assist healthcare professionals in distinguishing between different types of diabetes, such as type 1 and type 2, which have distinct treatment strategies. This precision can lead to more tailored and effective treatment plans for patients.

Additionally, AI can play a crucial role in ongoing disease management. By continuously monitoring patients and providing real-time feedback, it can help individuals with diabetes make informed decisions about their lifestyle choices, medication adherence, and glucose control. This continuous support can significantly improve the quality of life for those living with diabetes.Moreover, AI-based diabetes prediction systems can contribute to more efficient and cost-effective healthcare delivery. By streamlining the diagnostic process and reducing the need for unnecessary tests, they can help alleviate the burden on healthcare systems and resources.However, it's essential to recognize that while AI has tremendous potential in diabetes prediction and management, it is not without challenges. Issues related to data privacy, algorithm bias, and ethical considerations must be addressed to ensure the responsible and equitable implementation of these technologies.

In conclusion, AI-based diabetes prediction systems have the potential to transform the way we approach diabetes diagnosis, risk assessment, and management. Their ability to provide early detection, precise diagnosis, ongoing support, and cost efficiency makes them a valuable tool in the fight against diabetes. With careful consideration of ethical and privacy concerns, AI in diabetes care has the potential to greatly improve the lives of individuals affected by this chronic condition.