**MEASURING ENERGY CONSUMPTION**

**TEAM MEMBER**

**au714221104010: S.DHULASILINKAM**

**PHASE 5 SUBMISSION DOCUMENT**

**Project**: Measuring energy consumption

**Problem Statement:** Design and develop an AI-based Diabetics Prediction System that can accurately predict the risk of diabetes in individuals based on their health and lifestyle data. The system should take into account various risk factors and provide actionable insights for both individuals and healthcare providers.

Design Thinking is a human-centered approach to problem-solving and innovation that focuses on understanding the needs and perspectives of end-users. When applying Design Thinking to the development of an AI-Based Diabetics Prediction System, you can follow a structured process with the following stages:

1. **Empathize: Understand the Needs of Users and Stakeholders**
   * Conduct interviews, surveys, and observations to empathize with potential users, healthcare professionals, and stakeholders.
   * Learn about the challenges, fears, and needs of people at risk of diabetes or those already diagnosed.
   * Understand the workflow of healthcare providers in managing diabetes.
2. **Define: Clearly Articulate the Problem**
   * Define a specific problem statement based on your understanding of users and stakeholders. For example, "How might we accurately predict diabetes risk and provide actionable insights to individuals and healthcare providers?"
   * Identify the goals and key objectives for the Diabetics Prediction System .
3. **Ideate: Brainstorm and Generate Ideas**
   * Brainstorm potential solutions and features for the system that can address the defined problem.
   * Encourage creative thinking and ideation sessions with a diverse group of team members.
4. **Prototype: Create a Low-Fidelity Prototype**
   * Develop a low-fidelity prototype of the AI-Based Diabetics Prediction System. This could be paper sketches or digital wireframes.
   * Focus on creating a user interface that reflects the user experience you envision.
5. **Test: Gather Feedback and Iterate**
   * Share the prototype with potential users and stakeholders to gather feedback.
   * Pay attention to how users interact with the prototype, what they find intuitive, and what needs improvement.
   * Use this feedback to iterate and refine the design.
6. **Build: Develop the AI-Based Diabetics Prediction System**
   * Based on the feedback and insights from the prototype testing, start developing the system's software and AI components.
   * Implement data collection, preprocessing, feature engineering, machine learning models, and a user-friendly interface.
7. **Test Again: Validate the System's Functionality**
   * Conduct testing of the developed system to ensure it works as intended and provides accurate predictions.
   * Perform usability testing to check the user interface's effectiveness and accessibility.
8. **Launch and Deploy: Release the System**
   * Deploy the AI-Based Diabetics Prediction System to a limited user group for initial use.
   * Monitor the system's performance, security, and data privacy.
9. **Collect User Feedback and Data: Continuous Improvement**
   * Continuously gather user feedback and data on system usage and outcomes.
   * Use this information to make improvements, update models, and refine the user interface.
10. **Scale and Integrate: Expand Usage and Integration**
    * Scale the system to reach a larger user base while maintaining performance.
    * Explore integration with healthcare systems, electronic health records, and other relevant platforms.
11. **Measure Impact: Evaluate the System's Success**
    * Continuously assess the impact of the Diabetics Prediction System on users and healthcare providers.
    * Evaluate its accuracy in predicting diabetes and the extent to which it provides actionable insights.
12. **Iterate and Innovate: Keep Improving**
    * Use a feedback loop to iterate on the system's design, features, and functionality.
    * Stay informed about advancements in AI and healthcare to adapt the system accordingly.

Throughout the Design Thinking process, it's essential to maintain a user-centered focus, collaborate with healthcare professionals, ensure data privacy and security, and remain open to feedback and changes to enhance the AI-Based Diabetics Prediction System continually.

**The development of an AI-Based Diabetics Prediction System involves several distinct phases. These phases are essential to ensure a systematic and successful creation of the system. Below are the key phases of development for such a system:**

**1. Project Inception and Planning:**

- Define the scope, objectives, and goals of the project.

- Identify the target audience, including individuals at risk of diabetes and healthcare professionals.

- Establish a project team with roles and responsibilities.

- Develop a project plan with timelines, milestones, and resource allocation.

**2. Data Collection and Integration:**

- Gather relevant health and lifestyle data from various sources, including medical records, surveys, wearable devices, and health apps.

- Clean and preprocess the data, handling missing values, outliers, and data quality issues.

- Ensure data privacy and compliance with applicable regulations (e.g., GDPR, HIPAA).

**3. Feature Engineering and Selection:**

- Identify and select the most relevant features (predictors) that influence diabetes risk.

- Create new features if necessary, such as BMI calculations or composite risk scores.

- Explore the use of domain knowledge to enhance feature selection.

**4. Machine Learning Model Development:**

- Develop and implement machine learning models to predict diabetes risk. Common models include logistic regression, decision trees, random forests, support vector machines, or neural networks.

- Experiment with different algorithms to find the most accurate model.

**5. Training and Validation:**

- Train the selected machine learning model on labeled data (historical health data).

- Use techniques like cross-validation to validate the model's performance.

- Evaluate the model using appropriate metrics (e.g., accuracy, precision, recall, F1-score, AUC-ROC).

**6. User Interface (UI) Design and Development:**

- Create a user-friendly interface for individuals to input their health and lifestyle data.

- Design the UI to be intuitive and accessible.

- Ensure a seamless user experience for both technical and non-technical users.

**7. Integration with Data Sources:**

- Integrate the AI system with data sources and healthcare systems to fetch real-time or updated data, if applicable.

- Ensure data synchronization and reliability.

**8. Privacy and Security Measures:**

- Implement robust data privacy and security measures to protect personal and health data.

- Encrypt data in transit and at rest.

- Define user access controls and authentication mechanisms.

**9. Testing and Quality Assurance:**

- Conduct thorough testing of the system, including unit testing, integration testing, and system testing.

- Verify that the system meets accuracy and performance requirements.

- Ensure that the system handles various input scenarios.

**10. Deployment:**

- Deploy the AI-Based Diabetics Prediction System to a production environment.

- Monitor system performance, scalability, and data accuracy in a real-world setting.

**11. User Training and Adoption:**

- Provide training and support for end-users and healthcare professionals.

- Promote adoption and proper utilization of the system.

**12. Continuous Improvement and Monitoring:**

- Implement a feedback loop for continuous model improvement based on user outcomes and feedback.

- Monitor system performance and security and apply updates and enhancements as needed.

**13. Scale and Integration:**

- Expand the system's user base while maintaining performance.

- Explore integration with healthcare systems, electronic health records, and other relevant platforms.

**14. Compliance and Regulations:**

- Ensure that the system complies with relevant healthcare and data protection regulations.

- Stay up-to-date with evolving regulations and adjust the system as necessary.

**15. User Support and Maintenance:**

- Provide ongoing user support and maintenance to address issues, bugs, and user inquiries.

- Keep the system up to date with the latest AI and healthcare advancements.

Each of these development phases plays a critical role in the creation of an effective AI-Based Diabetics Prediction System. It's essential to maintain a systematic approach, involve domain experts and healthcare professionals, and adhere to ethical and regulatory guidelines throughout the development process.

To develop an AI-Based Diabetics Prediction System, the choice of the dataset, data preprocessing steps, and feature extraction techniques are crucial. Here's an overview of what you would typically consider for each of these components:

**Dataset:**

The dataset you choose plays a pivotal role in the performance of the AI-based Diabetics Prediction System. The dataset should ideally include health and lifestyle data related to individuals, both those with and without diabetes. Some common features in such datasets may include:

1. Age

2. Gender

3. Body Mass Index (BMI)

4. Blood Pressure (Systolic and Diastolic)

5. Glucose levels (fasting and post-meal)

6. Family history of diabetes

7. Physical activity level

8. Dietary habits (e.g., consumption of sugary beverages, fruits, vegetables)

9. Smoking and alcohol consumption

10. Medication history (e.g., use of insulin)

11. Existing medical conditions (e.g., hypertension, heart disease)

It's essential to ensure that the dataset is balanced, meaning that it contains a roughly equal number of positive cases (individuals with diabetes) and negative cases (individuals without diabetes) to prevent bias in model training.

**Data Preprocessing:**

Data preprocessing is a critical step in preparing the dataset for machine learning. Here are the typical data preprocessing steps:

**1. Data Cleaning:**

- Handle missing values by imputation (mean, median, mode) or removal of rows with missing data.

- Identify and handle outliers that may negatively impact model training.

**2. Data Normalization/Scaling:**

- Scale numeric features to a common range, often between 0 and 1, to ensure that no feature dominates the model due to its scale.

**3. Categorical Variable Encoding:**

- Encode categorical variables (e.g., gender, medication history) into numerical values using techniques like one-hot encoding.

**4. Feature Selection:**

- Select the most relevant features using techniques like feature importance scores, correlation analysis, or domain expertise.

- Remove redundant or irrelevant features to improve model performance and reduce dimensionality.

**5. Data Split:**

- Split the dataset into training and testing sets to assess the model's performance on unseen data. Common splits include 70% for training and 30% for testing.

**6. Imbalanced Data Handling (if applicable):**

- If the dataset is imbalanced, use techniques like oversampling (creating more instances of the minority class) or undersampling (reducing instances of the majority class) to balance the classes.

**Feature Extraction:**

Feature extraction involves deriving new features from the existing ones or transforming them to capture more relevant information for predicting diabetes risk. Feature extraction techniques might include:

1. BMI Calculation: Calculate the Body Mass Index (BMI) for each individual using the height and weight features.

2. Composite Risk Scores: Create composite risk scores by combining multiple features, such as BMI, glucose levels, and blood pressure, to represent overall health.

3. Age Categories: Convert age into categories (e.g., young, middle-aged, senior) to capture age-related risk variations.

4. Dietary Patterns: Analyze dietary habits to extract features related to the consumption of high-sugar or low-nutrient foods.

5. Physical Activity Level: Quantify physical activity as a feature, e.g., sedentary, moderate, or active lifestyle.

6. Temporal Features: Create features related to time trends, such as changes in glucose levels over time.

7. Interaction Features: Include interactions between features, for example, the product of glucose levels and BMI.

Feature extraction techniques should be informed by domain expertise and a thorough understanding of the factors contributing to diabetes risk. Additionally, feature extraction can be an iterative process, and different combinations of features may be tested to determine which provide the most valuable information for the prediction model.

By carefully selecting a relevant dataset, conducting effective data preprocessing, and employing appropriate feature extraction techniques, you can build a strong foundation for developing an AI-Based Diabetics Prediction System with the potential for accurate and actionable predictions.

The choice of machine learning algorithm, model training, and evaluation metrics is crucial in the development of an AI-Based Diabetics Prediction System. Here's an explanation of how you might make these decisions:

**Choice of Machine Learning Algorithm:**

Selecting the right machine learning algorithm depends on the nature of the problem, the dataset, and the desired trade-offs between accuracy and interpretability. In the context of a Diabetics Prediction System, several algorithms can be considered:

1. Logistic Regression: A simple yet interpretable algorithm that works well for binary classification problems. It can provide insights into the relationship between features and diabetes risk.

2. Decision Trees and Random Forests: These are useful for capturing non-linear relationships in the data and can handle both categorical and numeric features.

3. Support Vector Machines (SVM): SVM can be effective in separating classes in high-dimensional feature spaces. They are particularly useful when feature engineering has created complex feature combinations.

4. Neural Networks: Deep learning models, such as feedforward neural networks, can capture intricate patterns in the data but may require more data and computational resources. They are suitable for complex, large-scale prediction tasks.

The choice should be guided by experimentation, as different algorithms may perform differently on your specific dataset. You may try multiple algorithms and use cross-validation to assess their performance. Ensemble methods like Random Forests can be considered for combining the strengths of different algorithms.

**Model Training:**

Once you've chosen an algorithm, model training involves the following steps:

1. Data Split: Split your dataset into training and testing sets. A common split is 70% for training and 30% for testing.

2. Hyperparameter Tuning: Tune the hyperparameters of your chosen algorithm. Techniques like grid search or random search can be used to find the best hyperparameter settings.

3. Training: Train the model on the training data using the selected algorithm and hyperparameters. Ensure that the model is robust and not overfitting the training data.

4. Cross-Validation: If needed, perform cross-validation to assess the model's generalization ability. This helps ensure that the model performs consistently across different subsets of the data.

**Evaluation Metrics:**

The choice of evaluation metrics is essential for assessing the performance of your Diabetics Prediction System. Given the nature of a binary classification problem (diabetes or no diabetes), you can consider the following metrics:

1. Accuracy: The proportion of correctly classified instances. It's a general metric for overall model performance but can be misleading in the presence of imbalanced data.

2. Precision: The proportion of true positive predictions among all positive predictions. It's useful when minimizing false positives is crucial, such as in healthcare.

3. Recall (Sensitivity): The proportion of true positive predictions among all actual positive instances. It's essential when identifying all positive cases is critical, like in diagnosing diabetes.

4. F1-Score: The harmonic mean of precision and recall. It's a balanced metric that considers both false positives and false negatives.

5. Area Under the Receiver Operating Characteristic Curve (AUC-ROC): It assesses the model's ability to distinguish between positive and negative cases. An AUC value close to 1 indicates a strong model.

6. Confusion Matrix: A matrix that provides a detailed breakdown of true positives, true negatives, false positives, and false negatives. It's helpful for understanding model performance.

7. Specificity: The proportion of true negative predictions among all actual negative instances.

The choice of evaluation metrics depends on the specific goals of the Diabetics Prediction System and the relative importance of minimizing false positives or false negatives. It's often recommended to consider multiple metrics to gain a comprehensive understanding of the model's performance.

In summary, the choice of machine learning algorithm, model training, and evaluation metrics should be guided by the specific characteristics of your dataset and the objectives of the Diabetics Prediction System. Experimentation, iterative model development, and a deep understanding of the healthcare domain are essential for building an effective system.

**The development of an AI-Based Diabetics Prediction System can benefit from innovative techniques and approaches that leverage the latest advancements in artificial intelligence, machine learning, and healthcare research. While the field is constantly evolving, here are some innovative techniques and approaches that could be integrated into the system:**

1. Deep Learning and Neural Networks:

- Utilize deep neural networks, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), to capture complex patterns in medical imaging data (e.g., retinal scans for diabetic retinopathy) or time series data (e.g., continuous glucose monitoring).

2. Transfer Learning:

- Apply transfer learning from pre-trained models, such as large-scale language models (e.g., GPT-3) or medical image analysis models, to leverage their knowledge for diabetes prediction. Fine-tune these models with your dataset for improved accuracy.

3. Explainable AI (XAI):

- Implement XAI techniques to provide transparent and interpretable predictions. Techniques like LIME (Local Interpretable Model-agnostic Explanations) and SHAP (SHapley Additive exPlanations) can help users understand why a particular prediction was made.

4. Federated Learning:

- In cases where privacy is a primary concern, employ federated learning techniques to build a global prediction model without centralizing sensitive patient data. This allows for privacy-preserving predictions.

5. Genomic Data Integration:

- Include genetic and genomic data in the prediction process. Genetic markers associated with diabetes risk can be used as features in the model, enhancing prediction accuracy.

6. Continuous Monitoring and IoT Integration:

- Integrate IoT devices and wearables for real-time data collection. This can provide continuous monitoring of physiological parameters and lifestyle habits, improving the accuracy of predictions.

7. Personalized Medicine:

- Develop an approach that tailors predictions and recommendations to individual patient profiles, considering factors like genetic predisposition, past medical history, and lifestyle choices.

8. Natural Language Processing (NLP):

- Analyze textual data from electronic health records, clinical notes, and patient histories to extract valuable information for diabetes prediction and management.

9. Quantum Machine Learning:

- As quantum computing advances, explore its potential for solving complex optimization problems involved in feature selection, model training, and hyperparameter tuning.

10. Blockchain for Data Security:

- Implement blockchain technology to ensure the integrity and security of patient data, especially in scenarios where multiple parties are involved in data sharing.

11. Experiential Learning:

- Incorporate reinforcement learning techniques to optimize recommendations over time, adapting to individual patient responses and behaviors.

12. Robotic Process Automation (RPA):

- Integrate RPA to automate administrative tasks, appointment scheduling, and data entry, allowing healthcare professionals to focus more on patient care.

13. Augmented Reality (AR) and Virtual Reality (VR):

- Use AR and VR for patient education and behavior modification. Visualizing the effects of lifestyle changes or medication adherence can be motivating for patients.

14. Blockchain for Incentive Programs:

- Implement blockchain-based incentive programs to encourage healthy behaviors and compliance with medical advice. Patients can earn rewards for achieving diabetes management goals.

15. AI-Enabled Telemedicine:

- Integrate AI into telemedicine platforms to provide real-time decision support to healthcare providers during remote patient consultations, enhancing the quality of care.

These innovative techniques and approaches demonstrate the potential for AI-Based Diabetics Prediction Systems to be at the forefront of healthcare technology. However, it's essential to carefully evaluate the feasibility and ethical implications of each technique and tailor the system to the specific needs of patients and healthcare providers. Collaboration with medical professionals and adherence to regulatory guidelines are crucial in the development process.

**Program:**

#Importing Libraries

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

sns.set()

from mlxtend.plotting import plot\_decision\_regions

import missingno as msno

from pandas.plotting import scatter\_matrix

from sklearn.preprocessing import StandardScaler

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

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import confusion\_matrix

from sklearn import metrics

from sklearn.metrics import classification\_report

import warnings

warnings.filterwarnings('ignore')

%matplotlib inline

#Here we will be reading the dataset which is in the CSV format

diabetes\_df = pd.read\_csv('/content/diabetes.csv')

diabetes\_df.head()

#Exploratory Data Analysis (EDA)

#Now let’ see that what are columns available in our dataset.

diabetes\_df.columns

#Information about the dataset

diabetes\_df.info()

#To know more about the dataset

diabetes\_df.describe()

#To know more about the dataset with transpose – here T is for the transpose

diabetes\_df.describe().T

#Now let’s check that if our dataset have null values or not

diabetes\_df.isnull().head(10)

#Now let’s check the number of null values our dataset has.

diabetes\_df.isnull().sum()

diabetes\_df\_copy = diabetes\_df.copy(deep = True)

diabetes\_df\_copy[['Glucose','BloodPressure','SkinThickness','Insulin','BMI']] = diabetes\_df\_copy[['Glucose','BloodPressure','SkinThickness','Insulin','BMI']].replace(0,np.NaN)

# Showing the Count of NANs

print(diabetes\_df\_copy.isnull().sum())

#Data Visualization

#Plotting the data distribution plots before removing null values

p = diabetes\_df.hist(figsize = (20,20))

diabetes\_df\_copy['Glucose'].fillna(diabetes\_df\_copy['Glucose'].mean(), inplace = True)

diabetes\_df\_copy['BloodPressure'].fillna(diabetes\_df\_copy['BloodPressure'].mean(), inplace = True)

diabetes\_df\_copy['SkinThickness'].fillna(diabetes\_df\_copy['SkinThickness'].median(), inplace = True)

diabetes\_df\_copy['Insulin'].fillna(diabetes\_df\_copy['Insulin'].median(), inplace = True)

diabetes\_df\_copy['BMI'].fillna(diabetes\_df\_copy['BMI'].median(), inplace = True)

#Plotting the distributions after removing the NAN values.

p = diabetes\_df\_copy.hist(figsize = (20,20))

#Plotting Null Count Analysis Plot

p = msno.bar(diabetes\_df)

#Now, let’s check that how well our outcome column is balanced

color\_wheel = {1: "#0392cf", 2: "#7bc043"}

colors = diabetes\_df["Outcome"].map(lambda x: color\_wheel.get(x + 1))

print(diabetes\_df.Outcome.value\_counts())

p=diabetes\_df.Outcome.value\_counts().plot(kind="bar")

plt.subplot(121), sns.distplot(diabetes\_df['Insulin'])

plt.subplot(122), diabetes\_df['Insulin'].plot.box(figsize=(16,5))

plt.show()

#Correlation between all the features

#Correlation between all the features before cleaning

plt.figure(figsize=(12,10))

# seaborn has an easy method to showcase heatmap

p = sns.heatmap(diabetes\_df.corr(), annot=True,cmap ='RdYlGn')

#After Standard scaling

sc\_X = StandardScaler()

X =  pd.DataFrame(sc\_X.fit\_transform(diabetes\_df\_copy.drop(["Outcome"],axis = 1),), columns=['Pregnancies',

'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age'])

X.head()

#Model Building

#Splitting the dataset

X = diabetes\_df.drop('Outcome', axis=1)

y = diabetes\_df['Outcome']

#Now we will split the data into training and testing data using the train\_test\_split function

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=7)

#Random Forest

#Building the model using RandomForest

from sklearn.ensemble import RandomForestClassifier

rfc = RandomForestClassifier(n\_estimators=200)

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

#Now after building the model let’s check the accuracy of the model on the training dataset.

rfc\_train = rfc.predict(X\_train)

from sklearn import metrics

print("Accuracy\_Score =", format(metrics.accuracy\_score(y\_train, rfc\_train)))

#Getting the accuracy score for Random Forest

from sklearn import metrics

predictions = rfc.predict(X\_test)

print("Accuracy\_Score =", format(metrics.accuracy\_score(y\_test, predictions)))

#Decision Tree

#Building the model using DecisionTree

from sklearn.tree import DecisionTreeClassifier

dtree = DecisionTreeClassifier()

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

#Getting the accuracy score for Decision Tree

from sklearn import metrics

predictions = dtree.predict(X\_test)

print("Accuracy Score =", format(metrics.accuracy\_score(y\_test,predictions)))

#Classification report and confusion matrix of the decision tree model

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

print(confusion\_matrix(y\_test, predictions))

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

#XgBoost classifier

#Building model using XGBoost

from xgboost import XGBClassifier

xgb\_model = XGBClassifier(gamma=0)

xgb\_model.fit(X\_train, y\_train)

#Getting the accuracy score for the XgBoost classifier

from sklearn import metrics

xgb\_pred = xgb\_model.predict(X\_test)

print("Accuracy Score =", format(metrics.accuracy\_score(y\_test, xgb\_pred)))

#Support Vector Machine (SVM)

#Building the model using Support Vector Machine (SVM)

from sklearn.svm import SVC

svc\_model = SVC()

svc\_model.fit(X\_train, y\_train)

#Prediction from support vector machine model on the testing data

svc\_pred = svc\_model.predict(X\_test)

#Accuracy score for SVM

from sklearn import metrics

print("Accuracy Score =", format(metrics.accuracy\_score(y\_test, svc\_pred)))

#Classification report and confusion matrix of the SVM classifier

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

print(confusion\_matrix(y\_test, svc\_pred))

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

#Getting feature importances

rfc.feature\_importances\_

#Plotting feature importances

(pd.Series(rfc.feature\_importances\_, index=X.columns).plot(kind='barh'))

#Saving Model – Random Forest

import pickle

# Firstly we will be using the dump() function to save the model using pickle

saved\_model = pickle.dumps(rfc)

# Then we will be loading that saved model

rfc\_from\_pickle = pickle.loads(saved\_model)

# lastly, after loading that model we will use this to make predictions

rfc\_from\_pickle.predict(X\_test)

diabetes\_df.head()

#putting data points in the model will either return 0 or 1 i.e. person suffering from diabetes or not.

rfc.predict([[0,137,40,35,168,43.1,2.228,33]])

#Another one

rfc.predict([[10,101,76,48,180,32.9,0.171,63]])

**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

Pregnancies 0

Glucose 5

BloodPressure 35

SkinThickness 227

Insulin 374

BMI 11

DiabetesPedigreeFunction 0

Age 0

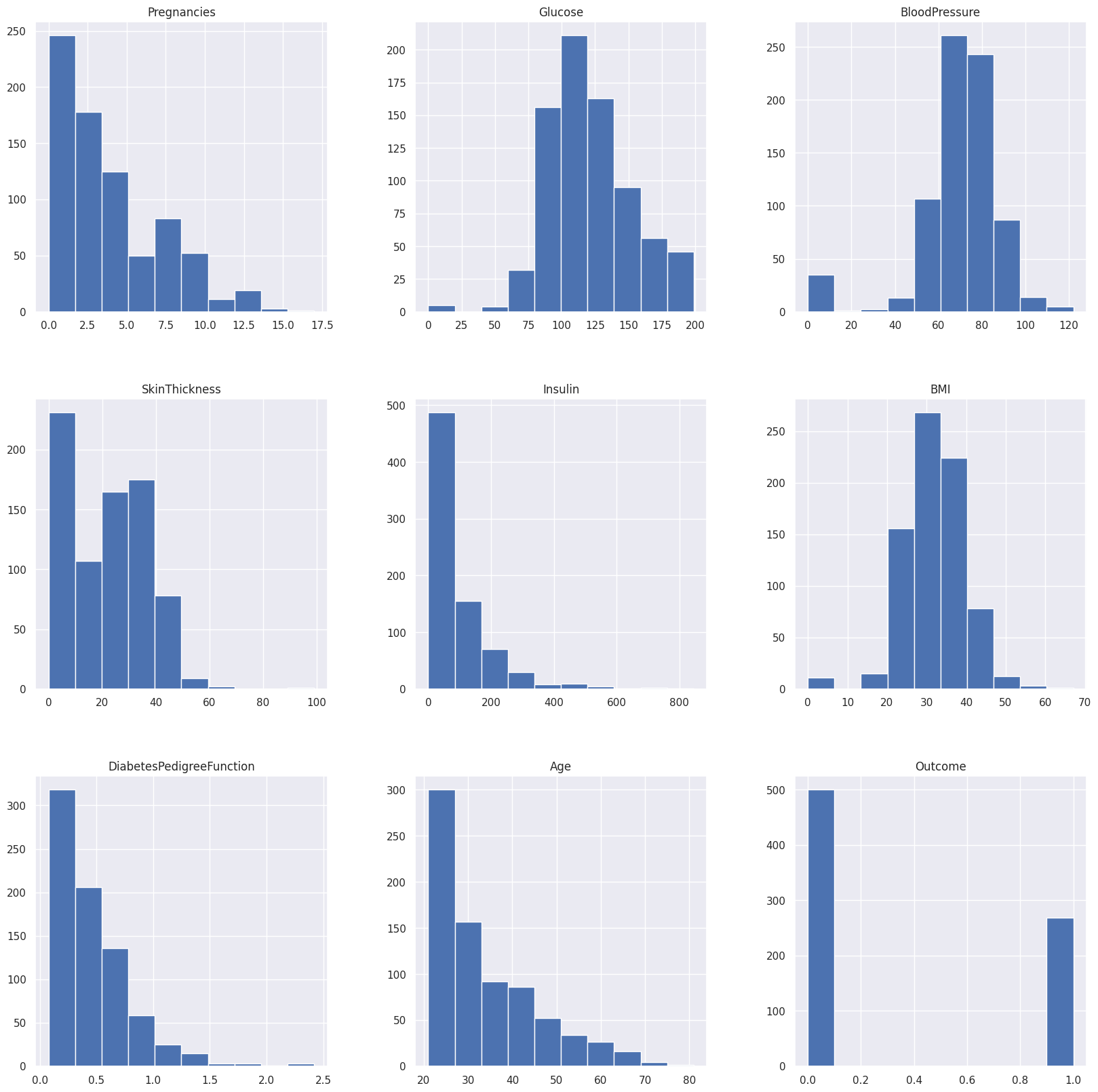
Outcome 0

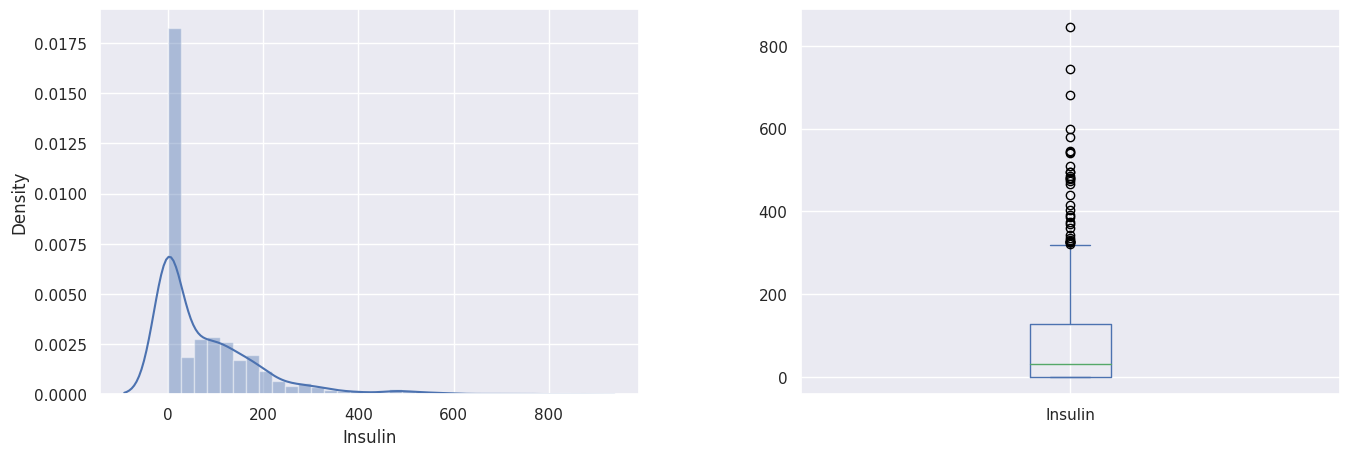
dtype: int64

0 500

1 268

Name: Outcome, dtype: int64





Accuracy\_Score = 1.0

Accuracy\_Score = 0.7834645669291339

Accuracy Score = 0.6968503937007874

[[125 37]

[ 40 52]]

precision recall f1-score support

0 0.76 0.77 0.76 162

1 0.58 0.57 0.57 92

accuracy 0.70 254

macro avg 0.67 0.67 0.67 254

weighted avg 0.69 0.70 0.70 254

Accuracy Score = 0.7283464566929134

Accuracy Score = 0.7480314960629921

[[145 17]

[ 47 45]]

precision recall f1-score support

0 0.76 0.90 0.82 162

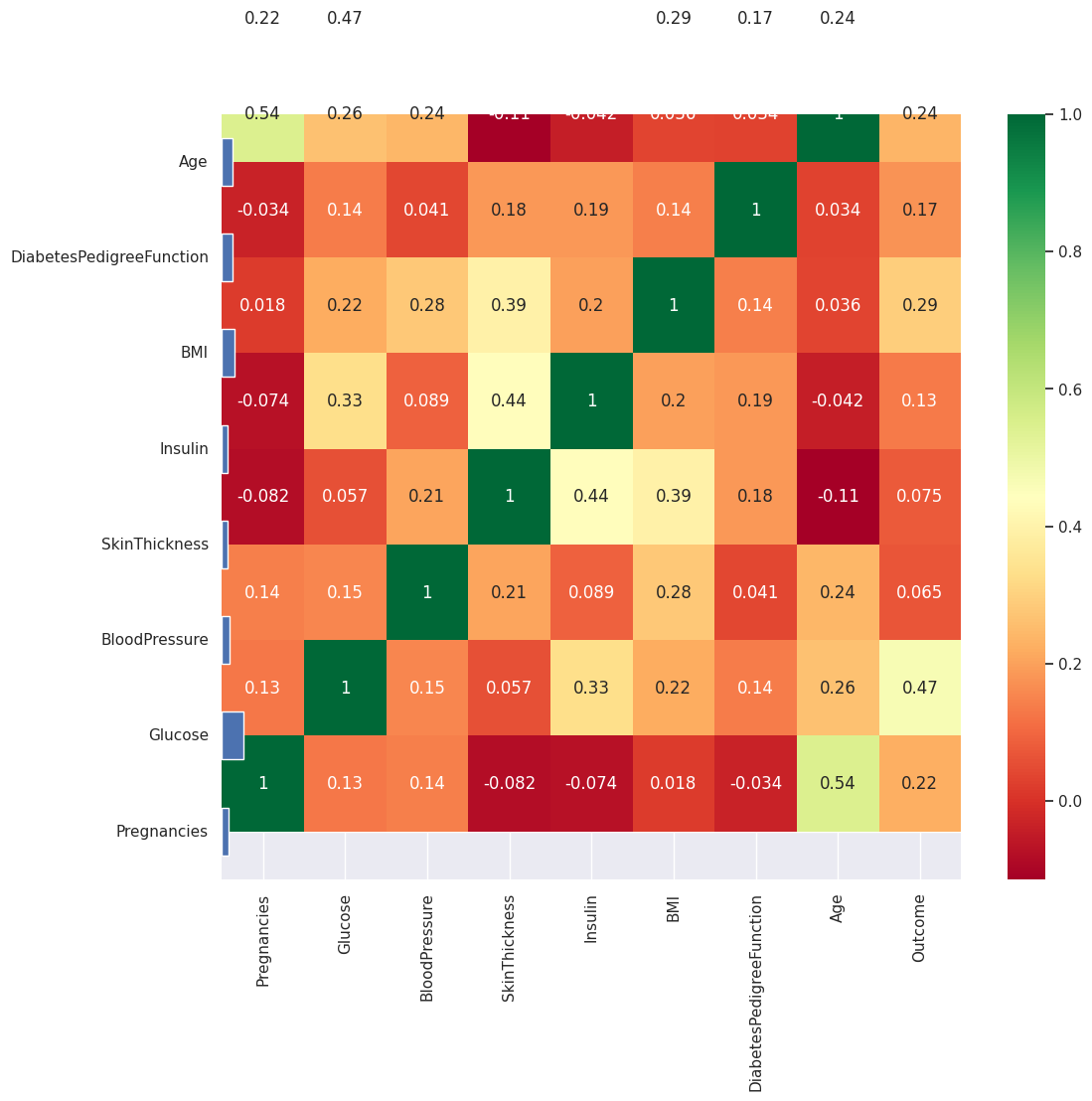
1 0.73 0.49 0.58 92

accuracy 0.75 254

macro avg 0.74 0.69 0.70 254

weighted avg 0.74 0.75 0.73 254

array([0])



**DATASET SOURCE: Kaggle(diabetics-**[**https://www.kaggle.com/datasets/mathchi/diabetes-data-set**](https://www.kaggle.com/datasets/mathchi/diabetes-data-set)**)**