KINGSTON ENGINEERING COLLEGE

COLLEGE CODE - 5113

**S. UDAYA KUMAR - (511321106023)[- udayakumar190704@gmail.com](mailto:-%20udayakumar190704@gmail.com) (team head)**

**S. CHANDRU - (511321106006)[- cc7442120@gmail.com](mailto:-%20cc7442120@gmail.com)**

**R. JEEVAN - (511321106013)[- rjeevan.ramachandran@gmail.com](mailto:-%20rjeevan.ramachandran@gmail.com)**

**S. VIGNESH - (511321106025)[- vigneshs4104@gmail.com](mailto:-%20vigneshs4104@gmail.com)**

AI BASED DIABETES PREDICITION SYSTEM

**Phase 3 Project:Development**



Introduction:

AI in Diabetes helps to predict or Detect Diabetes. Any neglect in health can have a high cost for the patients and the medical practitioner. It becomes challenging for the patient to trust that this decision is taken by the machine that does not explain how it reaches a particular conclusion

Purpose of diabetes prediction:

Early detection,Preventive Healthcare, Improved Patient Outcomes, Resource Optimization,Reducing the Burden of Disease, Reducing the Burden of Disease, Data-Driven Insight, Long-Term Health Monitoring and Telemedicine and Remote Monitoring.

Preprocessing the dataset:

Here is an overwiew of dataset preprocessing in diabetes prediction system:

Data Collection:

Gather a comprehensive dataset of patient information, including factors like age, BMI, family history, diet, physical activity, and previous medical records.

Data Preprocessing:

Clean, preprocess, and normalize the data to ensure consistency and remove any outliers or missing values.

Feature Selection:

Determine which features are most relevant for diabetes prediction. Feature engineering may also be necessary to create new informative variables.

Model Selection:

Choose an appropriate machine learning or deep learning algorithm for your prediction task. Common choices include logistic regression, decision trees, random forests, or neural networks.

Training the Model:

Split your dataset into training and testing sets to train and evaluate the model's performance. Fine-tune hyperparameters and optimize the model.

Validation:

Use k-fold cross-validation to validate the model's performance. This helps ensure that the model generalizes well to new data.

Evaluation Metrics:

Select appropriate evaluation metrics like accuracy, precision, recall, F1-score, and ROC-AUC to assess the model's performance.

Deployment:

Develop a user-friendly interface for the prediction system, whether it's a web application or a mobile app. Make sure the model can be easily integrated into the system.

Privacy and Security:

Ensure that patient data is handled securely and in compliance with data protection regulations.

Continuous Improvement:

The model should be periodically updated with new data to improve its accuracy over time.

Interpretability:

Consider using techniques to make the model's predictions interpretable, especially in healthcare applications where transparency is crucial.

Ethical Considerations:

Be mindful of the ethical implications of your AI system, especially when dealing with sensitive medical data.

Regulatory Compliance:

Ensure that your system complies with relevant healthcare and data privacy regulations, such as HIPAA in the United States.

Collaboration with Healthcare Professionals:

Collaborate with healthcare experts to validate the model's predictions and gain insights from their domain knowledge.

Documentation and Reporting:

Document the entire development process, model architecture, and results. This is essential for future reference and publication.

User Education:

Educate users, such as healthcare providers, on how to interpret and use the system's predictions effectively.

Feedback Mechanism:

Implement a feedback mechanism to collect user feedback and continually improve the system

Requirements:

\*Machine learning framework

\*Algorithm selection

\*Pandas

\*Numpy

\*Scalability

Step1:Preprocessing the data:

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

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

diabetes\_df.head()

Output:

Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',

'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],

dtype='object')

Information about the dataset

diabetes\_df.info()

Output:

RangeIndex: 768 entries, 0 to 767

Data columns (total 9 columns):

# Column Non-Null Count Dtype

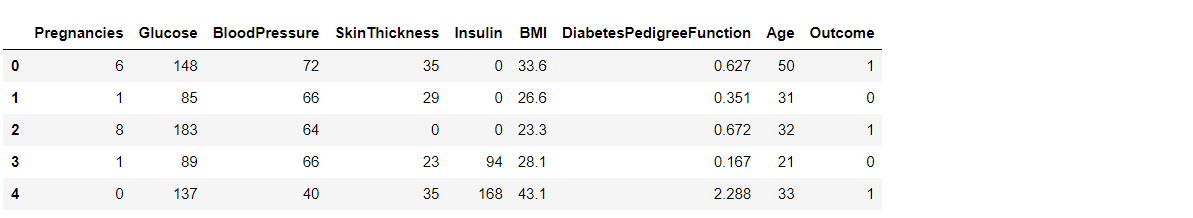
0 Pregnancies 768 non-null int64

1 Glucose 768 non-null int64

2 BloodPressure 768 non-null int64

3 SkinThickness 768 non-null int64

4 Insulin 768



Here from the above code we first checked that is there any null values from the **IsNull()** function then we are going to take the sum of all those missing values from the **sum()** function and the inference we now get is that there are no missing values but that is actually not a true story as in **this particular dataset all the missing values were given the 0 as a value which is not good for the authenticity of the dataset.** Hence we will first **replace the 0 value with the NAN** value then start the imputation process.

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())

**Output:**

Pregnancies 0

Glucose 5

BloodPressure 35

SkinThickness 227

Insulin 374

BMI 11

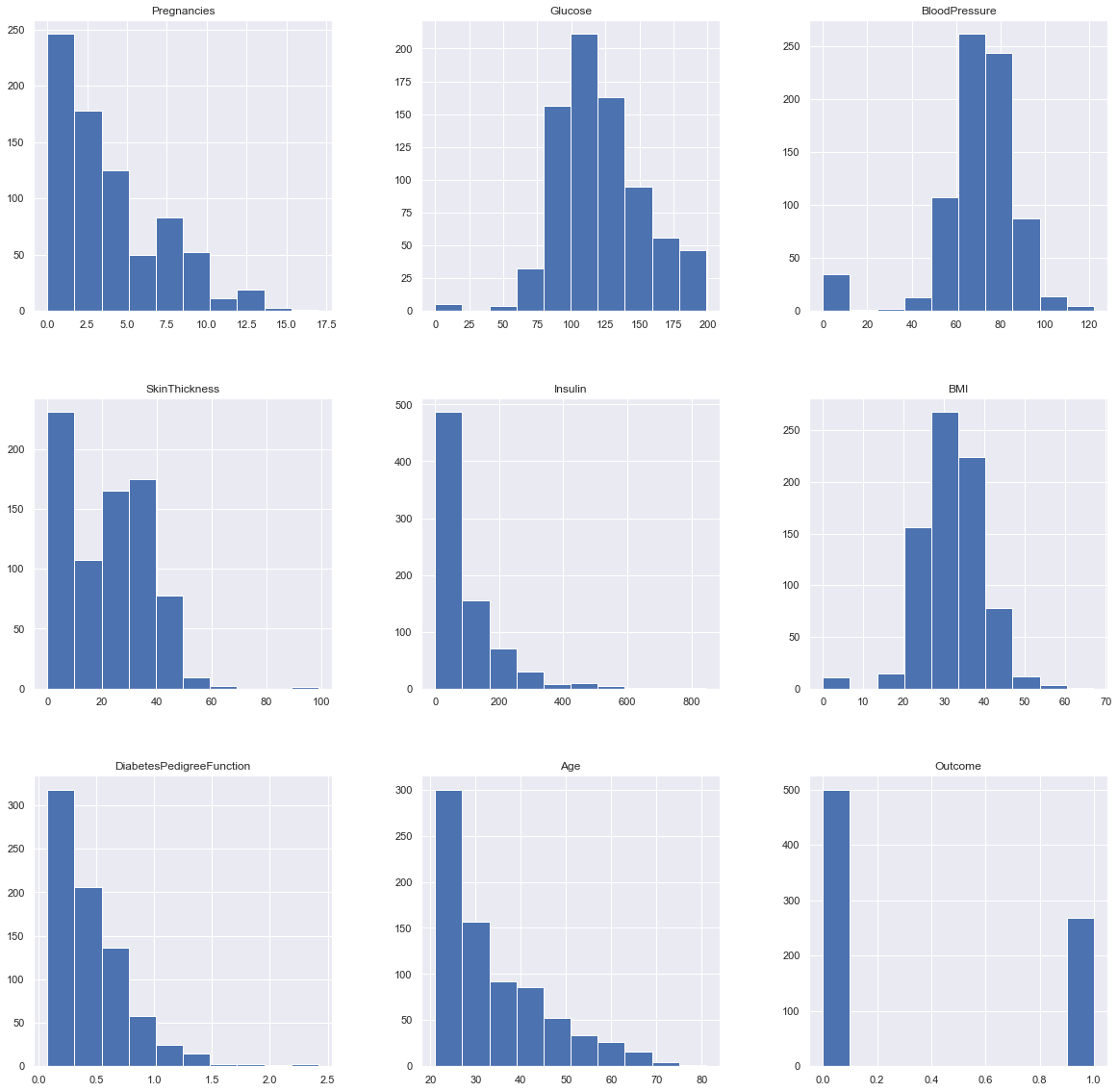
DiabetesPedigreeFunction 0

Data Visualization

**Plotting the data distribution plots before removing null values**

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

**Output:**

So here we have seen the distribution of each features whether it is dependent data or independent data and one thing which could always strike .So the answer is simple it is the best way to start the analysis of the dataset as **it shows the occurrence of every kind of value in the graphical structure which in turn lets us know the range of the data.**

**we will be imputing the mean value of the column to each missing value of that particular column.**

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)

**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))

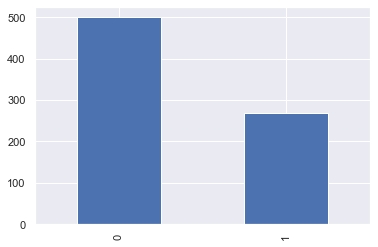
print(diabetes\_df.Outcome.value\_counts(

**Output:**

0 500

1 268

Name: Outcome, dtype: int64



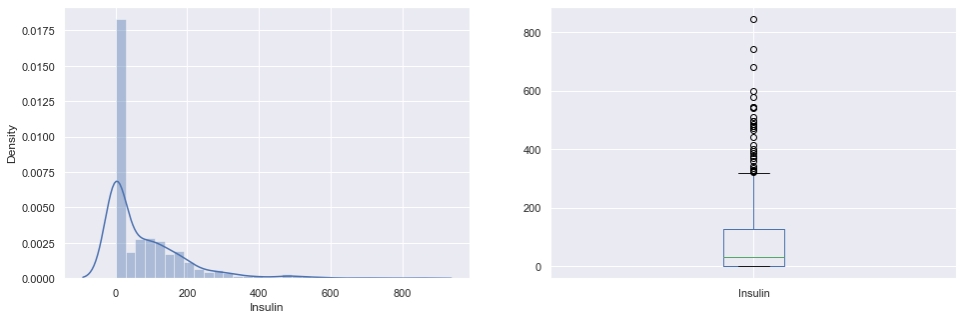
Here from the above visualization it is clearly visible that our **dataset is completely imbalanced** in fact the number of patients who are **diabetic is half of the patients who are non-diabetic.**

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

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

plt.show()

**Output:**



That’s how **Distplot** can be helpful where one will able to see the distribution of the data as well as with the help of **boxplot one can see the outliers in that column** and other information too which can be derived by the **box and whiskers plot.**

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')

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)))

Output: Accuracy = 1.0

**Conclusion:**

In this phase we started to predict a diabetes and summarized and discussed about it’s  preprocessing and dataset diabetes prediction not only used for texts it also used in various purspose like bussines applications,websites, applications, websites, mobile apps, or phone and upcoming phases we are going to build it advance level like featuring and evaluation,modeling.