**AI Based Diabetes Prediction System**

**NAME: Natanasabapathy. R**

**REG NO: 720421104037**

**AI\_PHASE2 DOCUMENT SUBMISSION**

**PROJECT : AI Based Diabetes Prediction System**

**PROBLEM DEFINITION :**

**The problem is to build an AI-powered diabetes prediction system that uses machine learning algorithms to analyze medical data and predict the likelihood of an individual developing diabetes. The system aims to provide early risk assessment and personalized preventive measures, allowing individuals to take proactive actions to manage their health.** **Early prediction of diabetes and prediabetes can reduce treatment cost and improve intervention. The development of (pre)diabetes is associated with various health conditions that can be monitored by routine health checkups. This study aimed to develop amachine learning-based model for predicting (pre)diabetes.**

**DESIGN THINKING :**

1. **Data Collection: We need a dataset containing medical features such as glucose levels, blood pressure, BMI, etc., along with information about whether the individual has diabetes or not.**
2. **Data Preprocessing: The medical data needs to be cleaned, normalized, and prepared for training machine learning models.**
3. **Feature Selection: We will select relevant features that can impact diabetes risk prediction.**
4. **Model Selection: We can experiment with various machine learning algorithms like Logistic Regression, Random Forest, and Gradient Boosting.**
5. **Evaluation: We will evaluate the model's performance using metrics like accuracy, precision, recall, F1-score, and ROC-AUC.**
6. **Iterative Improvement: We will fine-tune the model parameters and explore techniques like feature engineering to enhance prediction accuracy.**

**Data Source:**

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

**PROGRAM :**

**import pandas as pd**

**import numpy as np**

**from sklearn.preprocessing import StandardScaler**

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

**from sklearn.tree import DecisionTreeClassifier**

**from sklearn.svm import SVC**

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

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

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

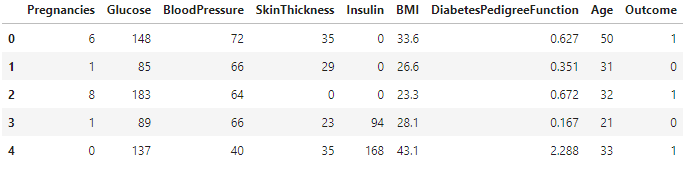
**import matplotlib.pyplot as plt**

**import seaborn as sns**

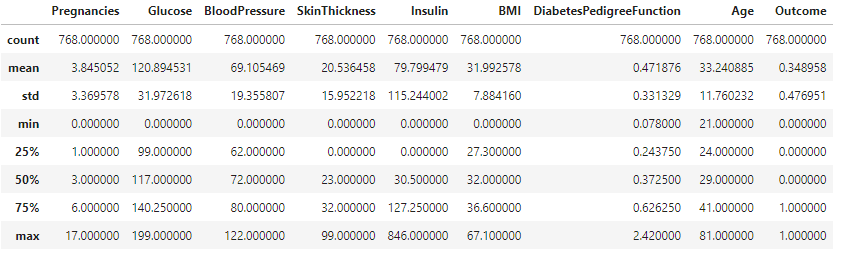
**import pandas as pd**

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

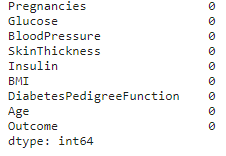
**data.head()**

****

**data.describe()**

****

**data.isnull().sum()**

****

**#here few misconception is there lke BMI can not be zero, BP can't be zero, glucose, insuline can't be zero so lets try to fix it**

**# now replacing zero values with the mean of the column**

**data['BMI'] = data['BMI'].replace(0,data['BMI'].mean())**

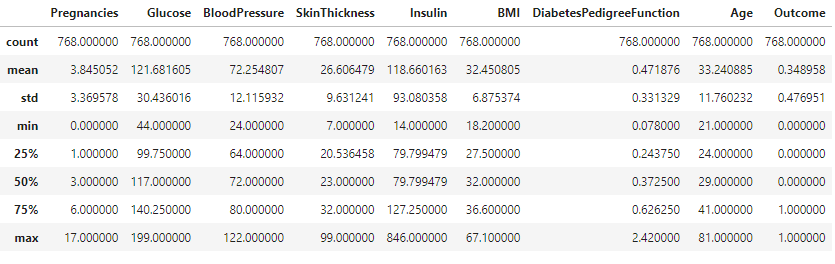
**data['BloodPressure'] = data['BloodPressure'].replace(0,data['BloodPressure'].mean())**

**data['Glucose'] = data['Glucose'].replace(0,data['Glucose'].mean())**

**data['Insulin'] = data['Insulin'].replace(0,data['Insulin'].mean())**

**data['SkinThickness'] = data['SkinThickness'].replace(0,data['SkinThickness'].mean())**

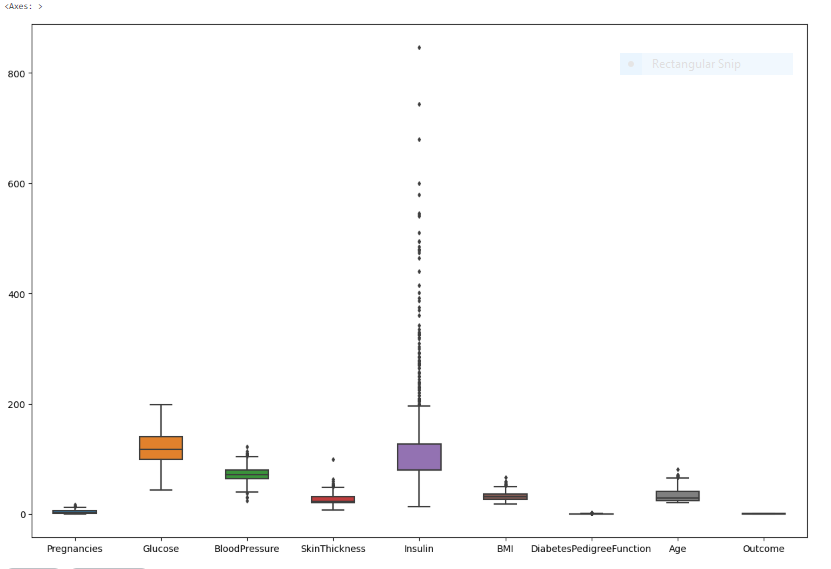
**data.describe()**



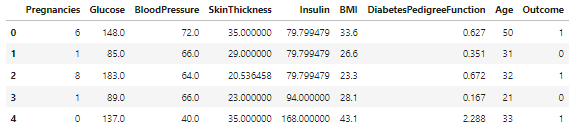
**#now we have dealt with the 0 values and data looks better. But, there still are outliers present in some columns.lets visualize it**

**fig, ax = plt.subplots(figsize=(15,10))**

**sns.boxplot(data=data, width= 0.5,ax=ax, fliersize=3)**

****

**data.head()**

****

**#segregate the dependent and independent variable**

**X = data.drop(columns = ['Outcome'])**

**y = data['Outcome']**

**# separate dataset into train and test**

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(X,y,test\_size=0.25,random\_state=0)**

**X\_train.shape, X\_test.shape**

**import pickle**

**##standard Scaling- Standardization**

**def scaler\_standard(X\_train, X\_test):**

**#scaling the data**

**scaler = StandardScaler()**

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

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

**#saving the model**

**file = open('standardScalar.pkl','wb')**

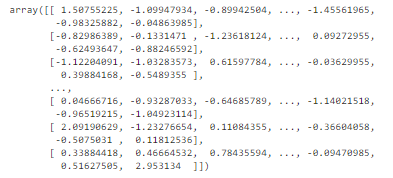
**pickle.dump(scaler,file)**

**file.close()**

**return X\_train\_scaled, X\_test\_scaled**

**X\_train\_scaled, X\_test\_scaled = scaler\_standard(X\_train, X\_test)**

**X\_train\_scaled**



**## Decision Tree Model Training With Hyperparameter Tuning**

**import warnings**

**warnings.filterwarnings('ignore')**

**parameter={**

**'criterion':['gini','entropy','log\_loss'],**

**'splitter':['best','random'],**

**'max\_depth':[1,2,3,4,5],**

**'max\_features':['auto', 'sqrt', 'log2']**

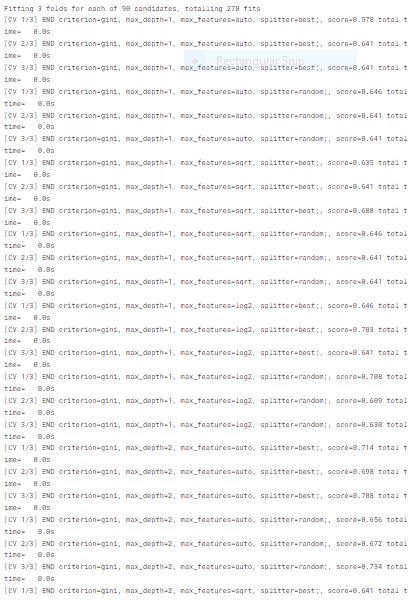
**}**

**from sklearn.model\_selection import GridSearchCV**

**classifier=DecisionTreeClassifier()**

**clf=GridSearchCV(classifier,param\_grid=parameter,cv=3,scoring='accuracy',verbose=3)**

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



clf.best\_params\_

{'criterion': 'gini',

'max\_depth': 5,

'max\_features': 'auto',

'splitter': 'best'}

classifier=DecisionTreeClassifier(criterion='entropy',max\_depth=5,max\_features='auto',splitter='random')

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

*## Support Vector Classifier With Hyperparameter Tuning*

*# defining parameter range*

param\_grid = {'C': [0.1, 1, 10],

'gamma': [1, 0.1, 0.01, 0.001, 0.0001],

'kernel':['linear','rbf','polynomial']

}

grid=GridSearchCV(SVC(),param\_grid=param\_grid,refit=True,cv=3,verbose=3,scoring='accuracy')

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

grid.best\_params\_

{'C': 0.1, 'gamma': 1, 'kernel': 'linear'}

svc\_clf=SVC(C=0.1,gamma=1,kernel='linear')

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

*## Decision Tree prediction*

y\_pred = classifier.predict(X\_test\_scaled)

*## SVC prediction*

y\_pred\_svc = svc\_clf.predict(X\_test\_scaled)

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

conf\_mat

array([[118, 12],

[ 48, 14]])

conf\_mat = confusion\_matrix(y\_test,y\_pred\_svc)

conf\_mat

array([[130, 0],

[ 62, 0]])

true\_positive = conf\_mat[0][0]

false\_positive = conf\_mat[0][1]

false\_negative = conf\_mat[1][0]

true\_negative = conf\_mat[1][1]

Accuracy = (true\_positive + true\_negative) / (true\_positive +false\_positive + false\_negative + true\_negative)

Accuracy

0.6770833333333334

Accuracy = (true\_positive + true\_negative) / (true\_positive +false\_positive + false\_negative + true\_negative)

Accuracy

0.6770833333333334

Precision = true\_positive/(true\_positive+false\_positive)

Precision

1.0

Recall = true\_positive/(true\_positive+false\_negative)

Recall

0.6770833333333334

F1\_Score = 2\*(Recall \* Precision) / (Recall + Precision)

F1\_Score

0.8074534161490683

import pickle

file = open('modelForPrediction.pkl','wb')

pickle.dump(classifier,file)

file.close()

**Conclusion:**

In summary, building an AI-powered diabetes prediction system is a complex endeavor involving data preprocessing, model training, and deployment. The provided code outline is a starting point. However, full development includes user interface integration, personalized preventive measures, data security, and ongoing updates. Ethical considerations, data privacy, and collaboration with healthcare professionals are crucial in real-world healthcare applications.