AI-Based Diabetes Prediction System

****

**Introduction:**

Diabetes is a health condition that affects how your body turns food into energy. Most of the food you eat is broken down into sugar (also called glucose) and released into your bloodstream. When your blood sugar goes up, it signals your pancreas to release insulin.

**Installing Libraries**

In this first step I have imported most common libraries used in python for machine learning such as Pandas, Seaborn, Matplitlib etc.

I am using Python because if very flexible and effective programming language i ever used. I used Python in software development field too.

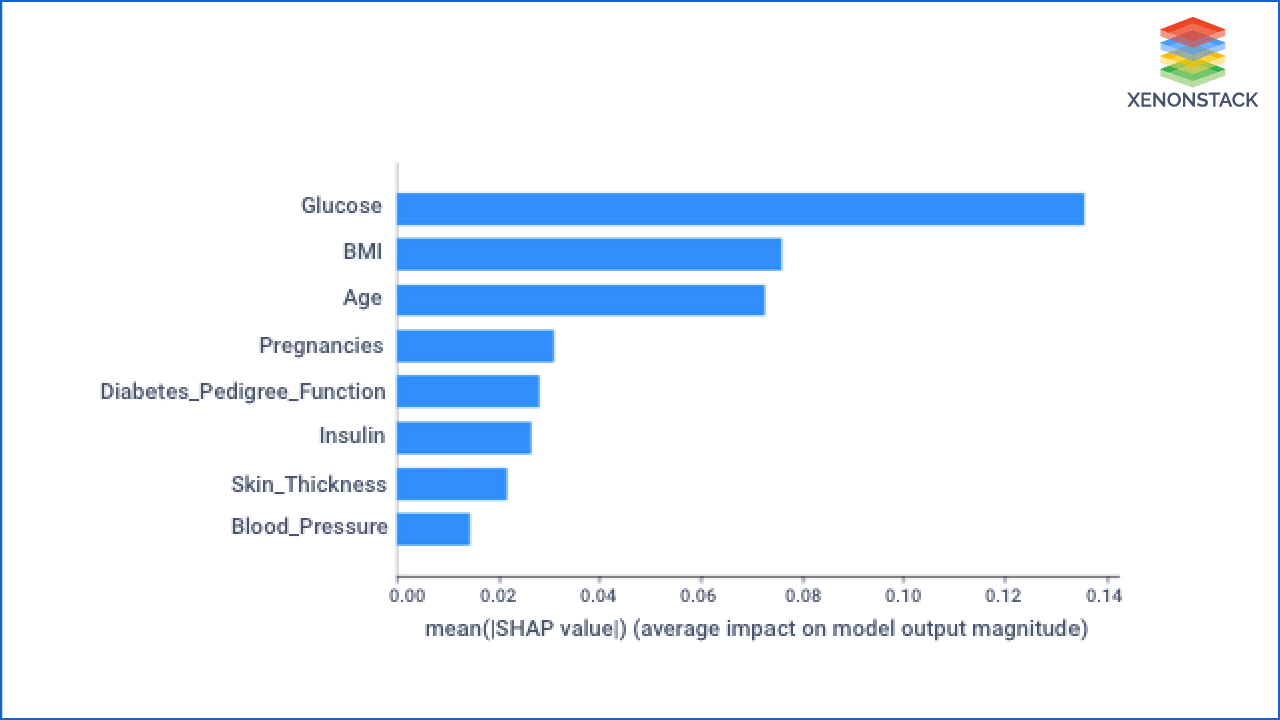
# Import libraries  
import numpy as np *# linear algebra*  
import pandas as pd *# data processing, CSV file I/O (e.g. pd.read\_csv)*  
import seaborn as sns *# for data visualization*  
import matplotlib.pyplot as plt *# to plot charts*  
from collections import Counter  
import os  
  
*# Modeling Libraries*  
from sklearn.preprocessing import QuantileTransformer  
from sklearn.metrics import confusion\_matrix, accuracy\_score, precision\_score  
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier, GradientBoostingClassifier, VotingClassifier  
from sklearn.linear\_model import LogisticRegression  
from sklearn.neighbors import KNeighborsClassifier  
from sklearn.tree import DecisionTreeClassifier  
from sklearn.svm import SVC  
from sklearn.model\_selection import GridSearchCV, cross\_val\_score, StratifiedKFold, learning\_curve, train\_test\_split

The sklearn library is very versatile and handy and serves real-world purposes. It provides wide range of ML algorithms and Models.

### **Which feature influences the result more?**

The contribution of features in making decisions can help doctors and patients to trust results.

The figure below depicts the importance of the features in predicting the output. Features are sorted by decreasing the importance from top to bottom to generate the output.

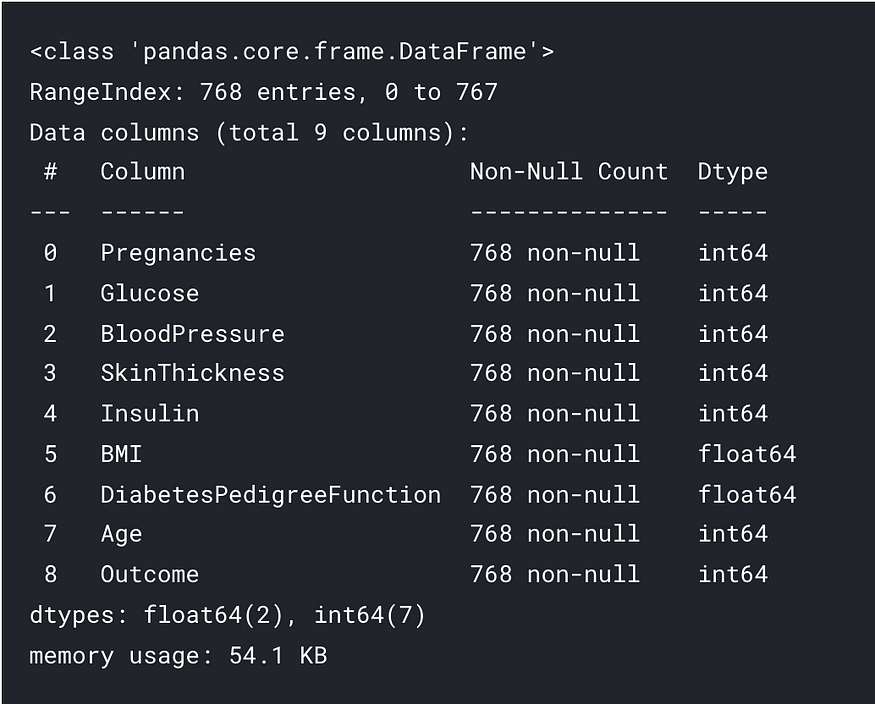


As it is showing, the Glucose value of a person influences the result more while predicting whether a person can have diabetes or not.

**Importing Data**

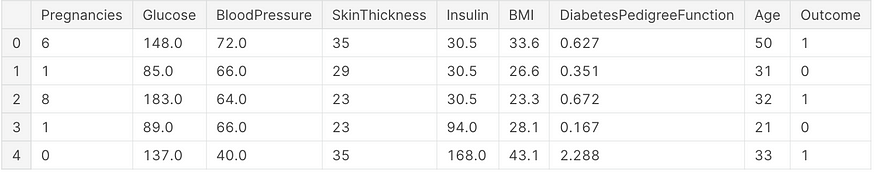
In this project i used [Pima Indians Diabetes Database](https://www.kaggle.com/gopalj/diabetes-prediction-using-python/data) from [Kaggle](https://www.kaggle.com/). This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases.

# Import dataset  
df = pd.read\_csv("../input/pima-indians-diabetes-database/diabetes.csv")*# Get familier with dataset structure*  
df.info()



Excepting BMI and DiabetesPedigreeFunction all the columns are integer. Outcome is the label containing 1 and 0 values. 1 means person has diabetes and 0 mean person is not diabetic.

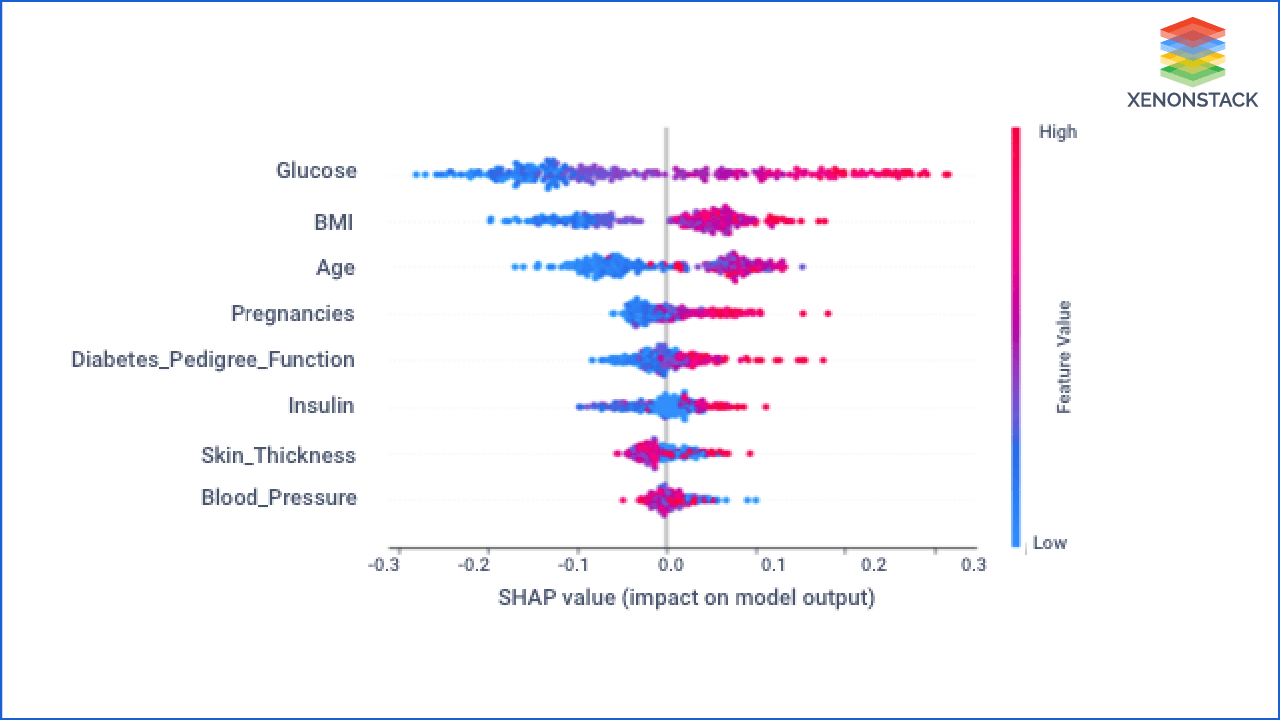
*# Show top 5 rows*  
df.head()



## **How is data contributing to making a decision in Diabetes Detection with AI?**

This is the next version of the previous graph. It also shows the same things with some more information about the value of the feature.

* **Feature importance:** Variables ranked in descending order of importance.
* **Impact:** The horizontal location display whether the effect of that value is associated with a higher or lower prediction.
* **Value:** Color display whether that variable is high or low for that observation. Red color devotes the high value and blue for less value. The variation in color of the dot shows the value of the feature.
* **Correlation:** A high level of "Glucose" content has a high impact on having diabetes. The "high" comes from red color, and the effect is shown on the X-axis.



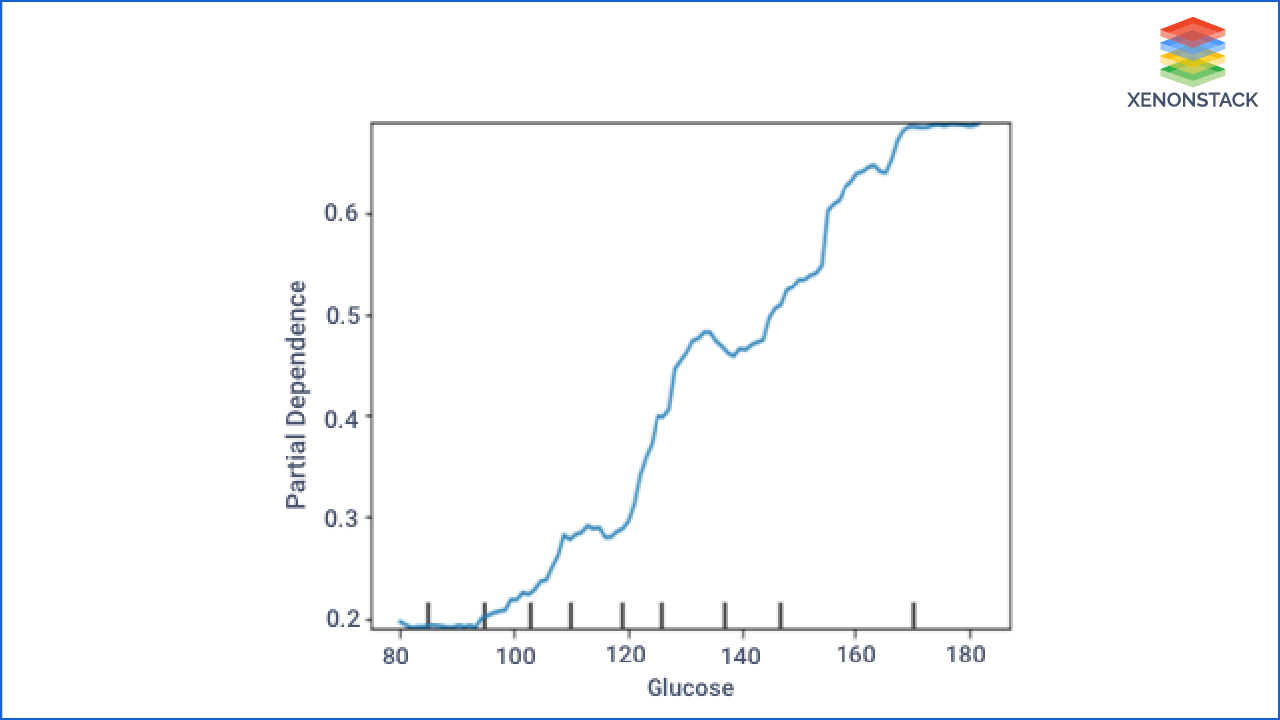
I observed that there is no missing values in dataset however the features like Glucose, BloodPressure, Insulin, SkinThickness has 0 values which is not possible. We have to replace 0 values with either mean or median values of specific column.

df['Glucose'] = df['Glucose'].replace(0, df['Glucose'].mean())# Correcting missing values in blood pressure  
df['BloodPressure'] = df['BloodPressure'].replace(0, df['BloodPressure'].mean()) # There are 35 records with 0 BloodPressure in dataset# Correcting missing values in BMI  
df['BMI'] = df['BMI'].replace(0, df['BMI'].median())# Correct missing values in Insulin and SkinThickness  
  
df['SkinThickness'] = df['SkinThickness'].replace(0, df['SkinThickness'].median())  
df['Insulin'] = df['Insulin'].replace(0, df['Insulin'].median())

After getting the answer to the first question, the customer can ask how the Glucose value change changes the system output when other parameters are not changing?

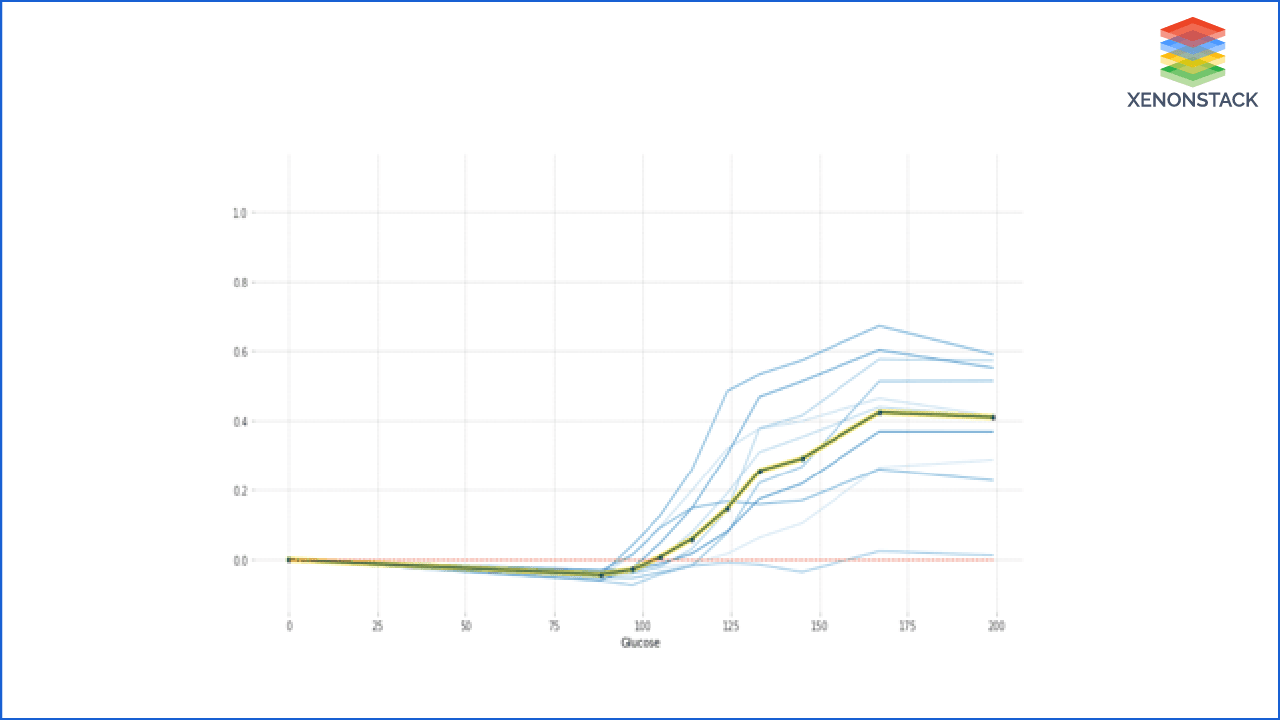
To answer this, let's discuss the [Partial Dependence Plot](https://www.kaggle.com/dansbecker/partial-dependence-plots) (PDP)

PDP shows the relation between the target response and feature. Other features are marginalized. This graph shows how a change in the value of Glucose changes the predicted output. It depicts that the increase in Glucose's value increases the probability of having diabetes to the person.

  
ICE (Individual Conditional Expectation) is a more detailed view of PDP. ICE is used to inspect the model's behavior for a specific instance, where everything except Glucose is held constant, fixed to its observed values. At the same time, Glucose is free to attain different values.

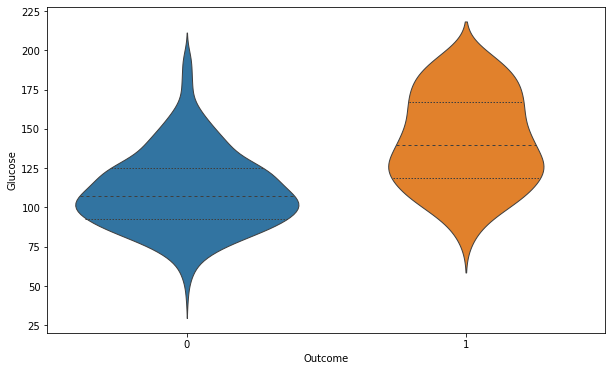
In PDP, other features are averaged out, but it took each case individually and the plot graph by only changing salary and remaining others constant. So this is a broad view of PDP.

In the Figure below, we just selected some of the observations. Users are free to choose any number of observations that they want to explore.



**Glucose**

# Explore Gluecose vs Outcomeplt.figure(figsize=(10,6))  
sns.violinplot(data=df, x="Outcome", y="Glucose",  
 split=True, inner="quart", linewidth=1)



The chances of diabetes is gradually increasing with level of Glucose.

*# Explore Glucose vs Outcome*  
  
plt.figure(figsize=(13,6))  
g = sns.kdeplot(df["Glucose"][df["Outcome"] == 1], color="Red", shade = True)  
g = sns.kdeplot(df["Glucose"][df["Outcome"] == 0], ax =g, color="Green", shade= True)  
g.set\_xlabel("Glucose")  
g.set\_ylabel("Frequency")  
g.legend(["Positive","Negative"])

