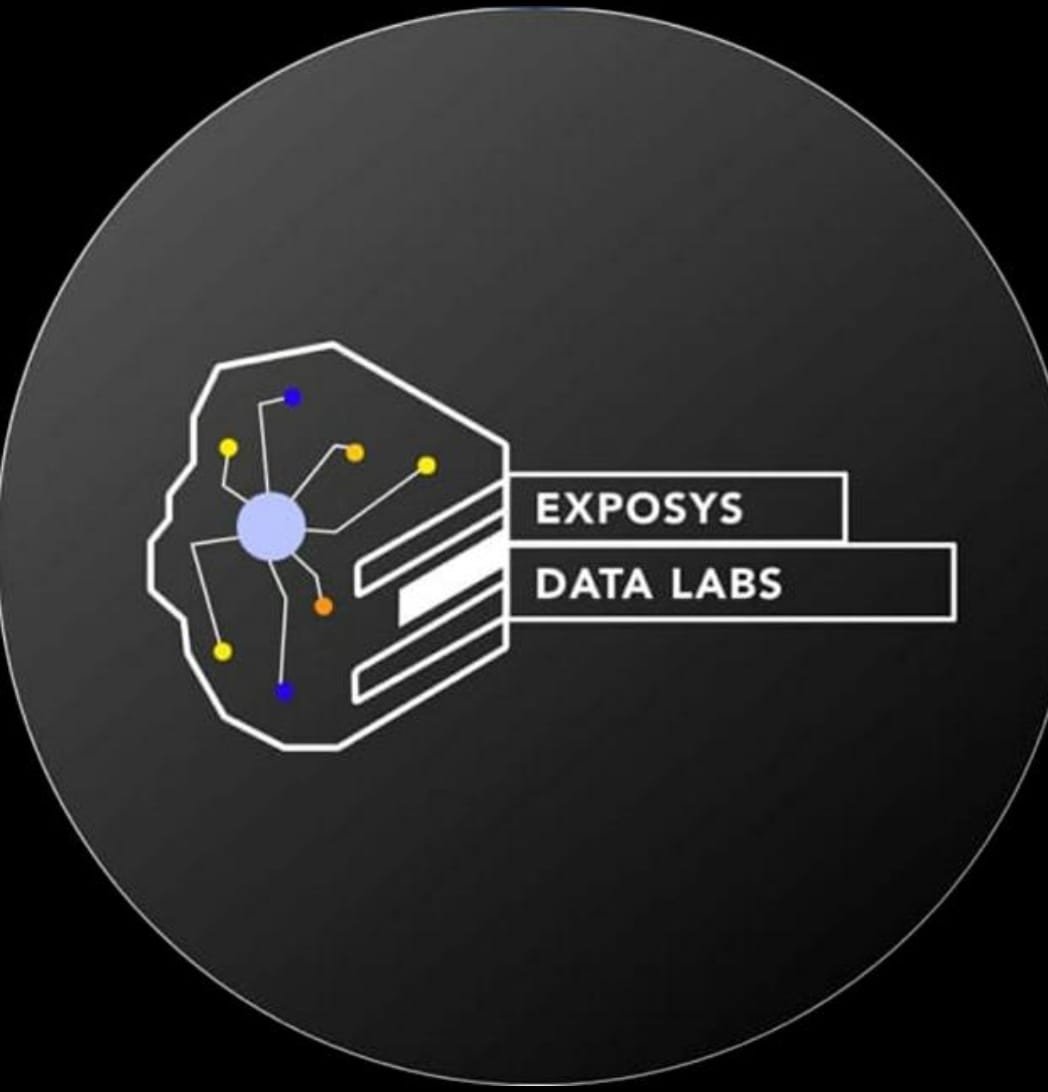
**A**

**Project Report**

**on**

**Diabetes Prediction Using Data Science**



EXPOSYS DATA LABS

Supervised by: Submitted by:

**Vishnuvardhan Y Aparna Raha**

**Mentor @Exposys data lab 5th Semester**

CONTENT

1. Abstract…………………………………………………………………………………..03
2. Introduction………………………………………………………………………………04
3. Project gap………………………………………………………………………………..05
4. Motivation………………………………………………………………………………..06
5. Problem Statement……………...………………………………………………………..07
6. Project Requirement……………………………………………………………………...08
7. Project Implementation……………………………………………………………….09-19
8. Project Outcome…….………………………..…………………………………………..20
9. Conclusion………………………………………………………………………………..21

Abstract

Diabetes is a critical disease and lots of people are suffering from this disease. Age, obesity, lack of exercise, heredity, diabetes, living style, bad diet, high blood pressure, etc., can cause diabetes. It is an illness caused to every human kind. Due to high glucose level in a human body. And diabetes should not be ignored if it is untreated then diabetes may cause some major issues in a person like heart related problems, kidney problem, blood pressure and eye damage and it can also affect other organs of human body. data science focuses on the development of computer system that can access data and use it to find hidden insights without being explicitly programmed where to look. In existing method classification and predict accuracy is not so high. A diabetes prediction model for better classification of diabetes which includes few external factors responsible for diabetes along with regular factors like Glucose, BMI, Age, insulin, etc. classification accuracy is boosted with new dataset compared to existing dataset.

INTRODUCTION

Diabetes is a common disease which can pose great threat to human health.it is caused because of obesity or high blood glucose level, and so forth. It affect the hormone insulin, resulting in abnormal metabolism of crabs and improves level of sugar in the blood. Early prediction of disease like diabetes can be controlled and save the human life. Data can be useful to predict diabetes.

Diabetes prediction using data science aims to predict diabetes via three different supervised learning methods including SVM, Logistic, Regression, ANN.

Diabetes can be divided into two categories, type 1 diabetes are normally younger with an age less than age 30 years old. The clinical symptoms are increase thirst and frequent urination this type of diabetes cannot be cleared by medication as it requires therapy. type 2 diabetes occurs more commonly on middle-aged and old people, which can show hypertension, obesity and other disease with our living standard diabetes has increased commonly in people’s daily life.

Hence a system is required as diabetes prediction is important as diabetes prediction is important area in computer, to handle the issues identified based on previous research.

PROJECT GAP

I have taken 4-5 weeks for making of the project, while making the projects I was facing many difficulties for that I required 3-4 more days for project completion. And I have taken 1 weeks and some extra days for making the presentation for the project making and detailing. Before going through the project I have taken time of 2 weeks and 2-3 days extra for learning the basic concept and basic terminologies while thinking more about project details where I have discussed about data science concept and used to learn more about data science and I come to know more about designing and patterning of the projects and how to deal with the mini projects. In this project I have taken help of mentor.

Our different queries I was facing while making the project and I have taken some resource in google also, I have used the references for understanding the concepts behind data science and for various queries comes to my mind.

MOTIVATION

I decided to grow my data science skills by engaging in diabetes prediction. I did this not only for fun and to learn but also to appreciate the essence of data science in solving some of the problems that plague humanity. Therefore this is an interesting project. Current human lifestyle is the main reason behind growth in diabetes. There can be three different types of errors-

1. The false-negative type in which a patient n reality is already a diabetic patient but test results tell that the person is not having diabetes.
2. The false-positive type which a patient in reality is not a diabetic patient but test report say that the person is a diabetic patient.

The third type is unclassified type in which a system cannot diagnose a given case. Therefore, there is need to create a system using data science which will provide accurate results and reduce human efforts.

Problem statement

Diabetes is a chronic disease which is more common among the people of all age group. predicting this disease at an early stage can help a person to take the necessary precaution and change his/her lifestyle accordingly to either prevent the occurrence of this disease or control the disease.

I prepared the data set using the data science and several method to train the model.

And by building a model which can give high accuracy of predicting the disease. This is the classification problem of supervised data science to predict whether or not a patient has diabetes, based on certain diagnosis measurement included in the dataset.

For 0- Absence of diabetes

1. Presence of diabetes

* Prepare the data set using several method to train the model.
* Build a model which can give high accuracy of predicting the disease.

PROJECT REQUIREMENTS

**Software Requirements Specifications**

Operating System Front End Back End Server Documentation : Windows 10.

Software: Python IDE 3.8, Jupyter Notebook, Visual Studio Code.

**Hardware Requirement Specifications**

Computer Processor Core i3 Processor Speed 2.3 GHz Processor Hard Disk 400 GB or more RAM Min 2GB.

PROJECT IMPLEMENTATION

**## Key Motivation for the Project**

I decided to grow my Datascience skills by engaging in diabetes prediction. I did this not only for fun and to learn but also to appreciate the essence of datascience in solving some of the problems that plague humanity.

Let's start by importing the libraries that I will be using.

# importing libraries

import numpy as np

import pandas as pd

from datetime import datetime

from datetime import date

import calendar

import matplotlib.pyplot as plt

import seaborn as sn

%matplotlib inline

Loading and reading the datasets

#Reading the dataset

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

# printing first fifteen rows

dataset.head(15)

Pregnancies Glucose BloodPressure SkinThickness Insulin BMI \

0 6 148 72 35 0 33.6

1 1 85 66 29 0 26.6

2 8 183 64 0 0 23.3

3 1 89 66 23 94 28.1

4 0 137 40 35 168 43.1

5 5 116 74 0 0 25.6

6 3 78 50 32 88 31.0

7 10 115 0 0 0 35.3

8 2 197 70 45 543 30.5

9 8 125 96 0 0 0.0

10 4 110 92 0 0 37.6

11 10 168 74 0 0 38.0

12 10 139 80 0 0 27.1

13 1 189 60 23 846 30.1

14 5 166 72 19 175 25.8

DiabetesPedigreeFunction Age Outcome

0 0.627 50 1

1 0.351 31 0

2 0.672 32 1

3 0.167 21 0

4 2.288 33 1

5 0.201 30 0

6 0.248 26 1

[show more (open the raw output data in a text editor) ...](vscode-file://vscode-app/c:/Users/ASUS/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)

9 0.232 54 1 10 0.191 30 0 11 0.537 34 1 12 1.441 57 0 13 0.398 59 1 14 0.587 51 1

#Printing the shape of dataset

dataset.shape

(768, 9)

#Printing the description of dataset

dataset.describe()

Pregnancies Glucose BloodPressure SkinThickness Insulin \ count 768.000000 768.000000 768.000000 768.000000 768.000000 mean 3.845052 120.894531 69.105469 20.536458 79.799479 std 3.369578 31.972618 19.355807 15.952218 115.244002 min 0.000000 0.000000 0.000000 0.000000 0.000000 25% 1.000000 99.000000 62.000000 0.000000 0.000000 50% 3.000000 117.000000 72.000000 23.000000 30.500000 75% 6.000000 140.250000 80.000000 32.000000 127.250000 max 17.000000 199.000000 122.000000 99.000000 846.000000 BMI DiabetesPedigreeFunction Age Outcome count 768.000000 768.000000 768.000000 768.000000 mean 31.992578 0.471876 33.240885 0.348958 std 7.884160 0.331329 11.760232 0.476951 min 0.000000 0.078000 21.000000 0.000000 25% 27.300000 0.243750 24.000000 0.000000 50% 32.000000 0.372500 29.000000 0.000000 75% 36.600000 0.626250 41.000000 1.000000 max 67.100000 2.420000 81.000000 1.000000

#Printing the Information related to dataset

dataset.info()

<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

#Printing the columns in the datasets

dataset.columns

Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'], dtype='object')

#Printing the datatypes of columns in the datasets

dataset.dtypes

Pregnancies int64

Glucose int64

BloodPressure int64

SkinThickness int64

Insulin int64

BMI float64

DiabetesPedigreeFunction float64

Age int64

Outcome int64

dtype: object

#Check missing values

dataset.isnull().sum()

Pregnancies 0

Glucose 0

BloodPressure 0

SkinThickness 0

Insulin 0

BMI 0

DiabetesPedigreeFunction 0

Age 0

Outcome 0

dtype: int64

#check correlations

dataset.corr()

Pregnancies Glucose BloodPressure SkinThickness \

Pregnancies 1.000000 0.129459 0.141282 -0.081672

Glucose 0.129459 1.000000 0.152590 0.057328

BloodPressure 0.141282 0.152590 1.000000 0.207371

SkinThickness -0.081672 0.057328 0.207371 1.000000

Insulin -0.073535 0.331357 0.088933 0.436783

BMI 0.017683 0.221071 0.281805 0.392573

DiabetesPedigreeFunction -0.033523 0.137337 0.041265 0.183928

Age 0.544341 0.263514 0.239528 -0.113970

Outcome 0.221898 0.466581 0.065068 0.074752

Insulin BMI DiabetesPedigreeFunction \

Pregnancies -0.073535 0.017683 -0.033523

Glucose 0.331357 0.221071 0.137337

BloodPressure 0.088933 0.281805 0.041265

SkinThickness 0.436783 0.392573 0.183928

Insulin 1.000000 0.197859 0.185071

BMI 0.197859 1.000000 0.140647

DiabetesPedigreeFunction 0.185071 0.140647 1.000000

Age -0.042163 0.036242 0.033561

Outcome 0.130548 0.292695 0.173844

Age Outcome

Pregnancies 0.544341 0.221898

Glucose 0.263514 0.466581

[show more (open the raw output data in a text editor) ...](vscode-file://vscode-app/c:/Users/ASUS/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)

SkinThickness -0.113970 0.074752

-0.042163 0.130548

BMI 0.036242 0.292695

DiabetesPedigreeFunction 0.033561 0.173844

Age 1.000000 0.238356

Outcome 0.238356 1.000000

**# Training and test or validation dataset**

Training set- A subset to train a model.

Test set- A subset to test the trained model.

Now I will Load the training and test or validation dataset

# loading the data

train = pd.read\_csv('train.csv')

test = pd.read\_csv('test.csv')

# shape of training and testing data

train.shape, test.shape

((768, 9), (768, 8))

# printing first five rows in training dataset

train.head()

Pregnancies Glucose BloodPressure SkinThickness Insulin BMI \ 0 6 148 72 35 0 33.6 1 1 85 66 29 0 26.6 2 8 183 64 0 0 23.3 3 1 89 66 23 94 28.1 4 0 137 40 35 168 43.1

Pregnancies Glucose BloodPressure SkinThickness Insulin BMI \ 0 6 148 72 35 0 33.6 1 1 85 66 29 0 26.6 2 8 183 64 0 0 23.3 3 1 89 66 23 94 28.1 4 0 137 40 35 168 43.1

# printing first five rows in test dataset

test.head()

Pregnancies Glucose BloodPressure SkinThickness Insulin BMI \ 0 6 148 72 35 0 33.6 1 1 85 66 29 0 26.6 2 8 183 64 0 0 23.3 3 1 89 66 23 94 28.1 4 0 137 40 35 168 43.1

DiabetesPedigreeFunction Age 0 0.627 50 1 0.351 31 2 0.672 32 3 0.167 21 4 2.288 33

# columns in the training dataset

train.columns

Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'], dtype='object')

# columns in the test dataset

test.columns

Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age'], dtype='object')

# Data type of the columns

train.dtypes

Pregnancies int64 Glucose int64 BloodPressure int64 SkinThickness int64 Insulin int64 BMI float64 DiabetesPedigreeFunction float64 Age int64 Outcome int64 dtype: object

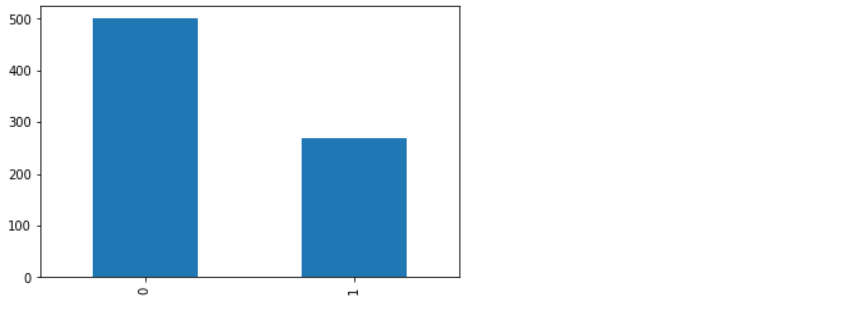
train['Outcome'].value\_counts()

0 500 1 268 Name: Outcome, dtype: int64

# Normalize can be set to True to print proportions instead of number

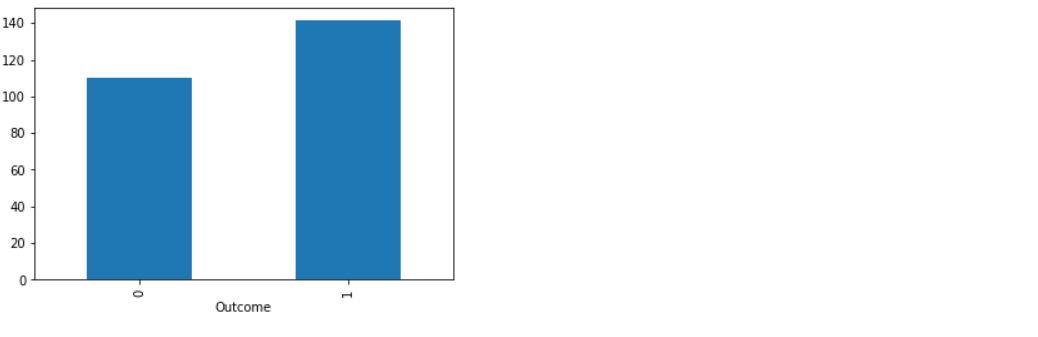
train['Outcome'].value\_counts(normalize=True)

0 0.651042 1 0.348958 Name: Outcome, dtype: float64



train.groupby('Outcome')['Glucose'].mean().plot.bar()

<AxesSubplot:xlabel='Outcome'>

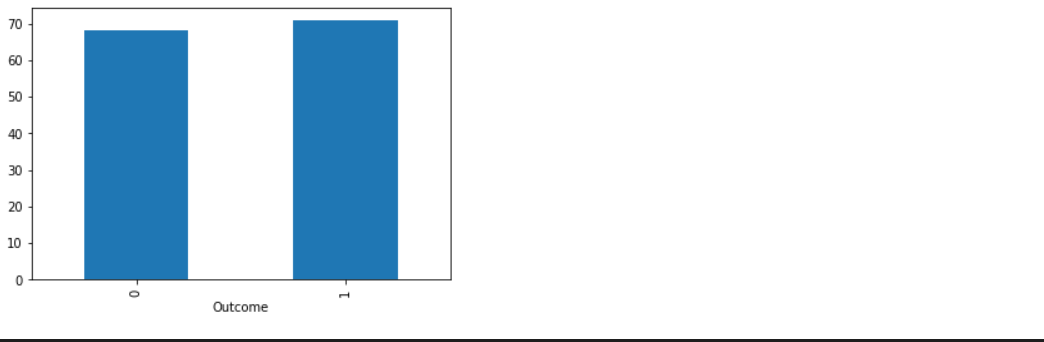


train.groupby('Outcome')['Insulin'].mean().plot.bar()

train.groupby('Outcome')['Insulin'].mean().plot.bar()

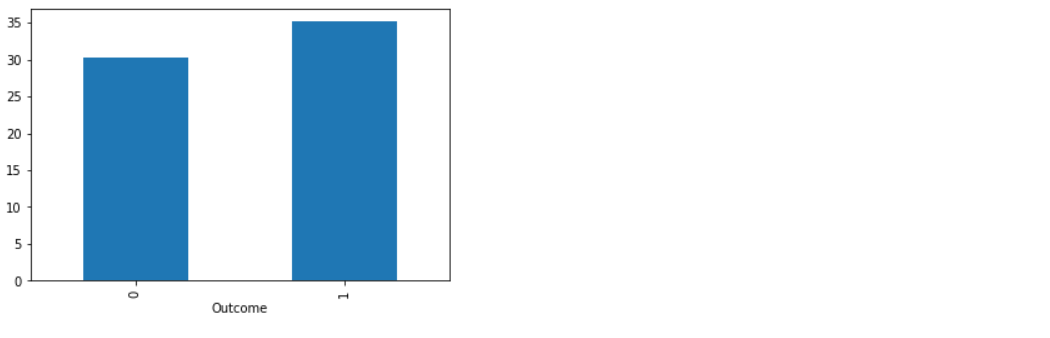


train.groupby('Outcome')['BloodPressure'].mean().plot.bar()



train.groupby('Outcome')['BMI'].mean().plot.bar()

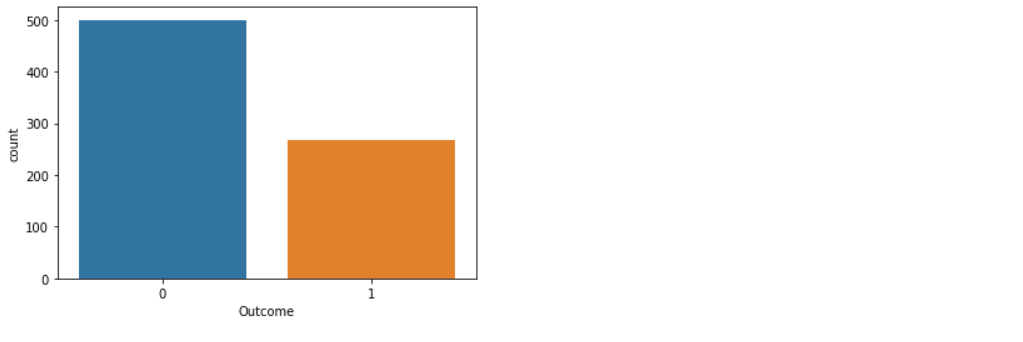
<AxesSubplot:xlabel='Outcome'>



#Plotting the distribution of Outcome

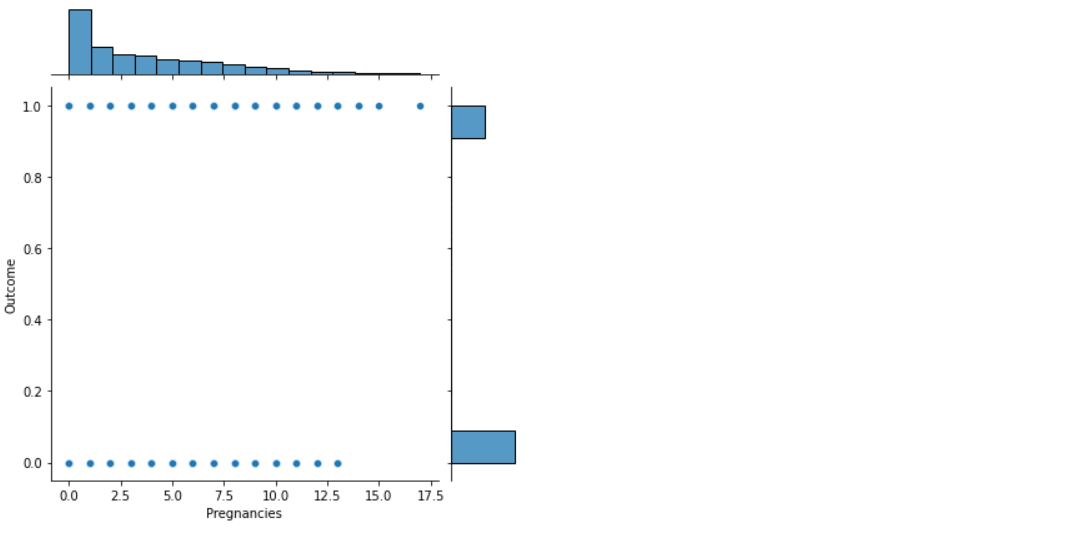
sn.countplot(x='Outcome', data = train)

<AxesSubplot:xlabel='Outcome', ylabel='count'>

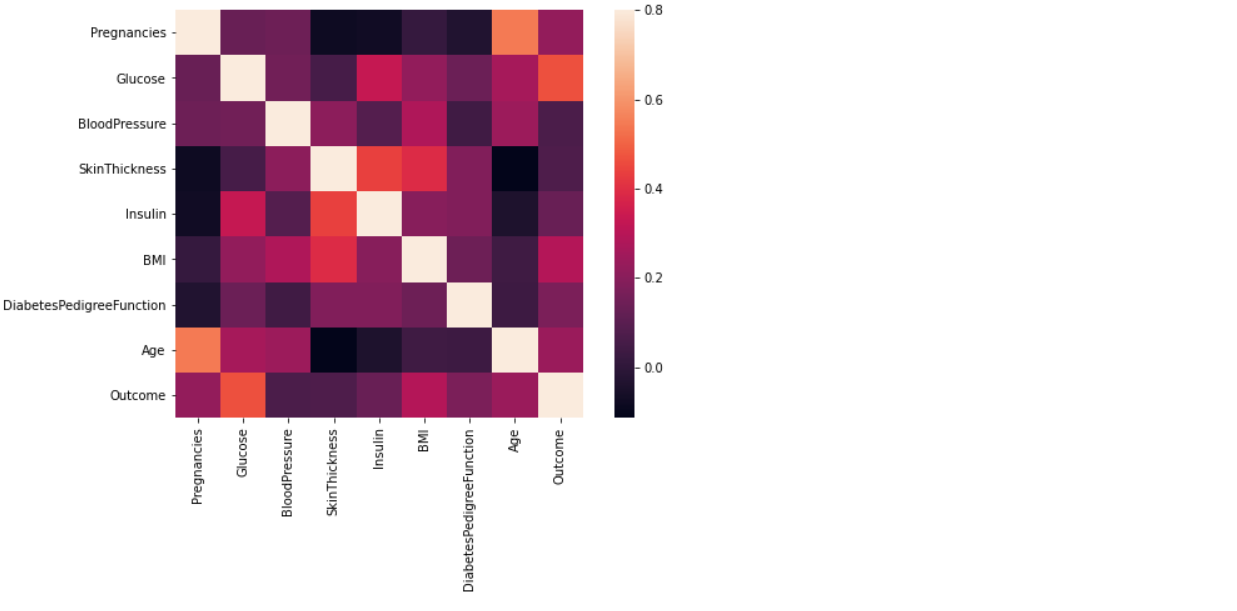


sn.jointplot(x='Pregnancies', y='Outcome', data=train)

<seaborn.axisgrid.JointGrid at 0x1ade92513d0>



sn.jointplot(x='Glucose', y='Outcome', data=train)



#Check missing values

train.isnull().sum()

Pregnancies 0 Glucose 0 BloodPressure 0 SkinThickness 0 Insulin 0 BMI 0 DiabetesPedigreeFunction 0 Age 0 Outcome 0 dtype: int64

**## Model Building**

target = train['Outcome']

train = train.drop('Outcome',1)

# applying dummies on the train dataset

train = pd.get\_dummies(train)

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

# splitting into train and validation with 20% data in validation set and 80% data in train set.

X\_train, X\_val, y\_train, y\_val = train\_test\_split(train, target, test\_size = 0.2, random\_state=12)

**## Logistic Regression**

from sklearn.linear\_model import LogisticRegression

# defining the logistic regression model

lreg = LogisticRegression()

# making prediction on the validation set

prediction = lreg.predict(X\_val)

from sklearn.metrics import accuracy\_score

# calculating the accuracy score

accuracy\_score(y\_val, prediction)

0.8116883116883117

We got an accuracy score of around 81% on the validation dataset. Logistic regression has a linear decision boundary. What if our data have non linearity? We need a model that can capture this non linearity.

Let's try decision tree algorithm now to check if we get better accuracy with that.

**## Decision Tree**

from sklearn.tree import DecisionTreeClassifier

# defining the decision tree model with depth of 4, you can tune it further to improve the accuracy score

clf = DecisionTreeClassifier(max\_depth=4, random\_state=0)

# fitting the decision tree model

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

# fitting the decision tree model

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

# making prediction on the validation set

predict = clf.predict(X\_val)

# calculating the accuracy score

accuracy\_score(y\_val, predict)

**## Random Forest Classifier**

from sklearn.ensemble import RandomForestClassifier

#defining the random forest classifier with estimators 200

rfc = RandomForestClassifier(n\_estimators=200)

RandomForestClassifier(n\_estimators=200)

# making prediction on the validation set

predict = rfc.predict(X\_val)

# calculating the accuracy score

accuracy\_score(y\_val, predict)

0.8116883116883117

**## XGBClassifier**

from xgboost import XGBClassifier

xgb = XGBClassifier(gamma=0)

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

XGBClassifier(base\_score=0.5, booster='gbtree', colsample\_bylevel=1, colsample\_bynode=1, colsample\_bytree=1, gamma=0, gpu\_id=-1, importance\_type='gain', interaction\_constraints='', learning\_rate=0.300000012, max\_delta\_step=0, max\_depth=6, min\_child\_weight=1, missing=nan, monotone\_constraints='()', n\_estimators=100, n\_jobs=0, num\_parallel\_tree=1, random\_state=0, reg\_alpha=0, reg\_lambda=1, scale\_pos\_weight=1, subsample=1, tree\_method='exact', validate\_parameters=1, verbosity=None)

# making prediction on the validation set

predict = xgb.predict(X\_val)

# calculating the accuracy score

accuracy\_score(y\_val, predict)

0.7402597402597403

After applying all these models we found out the best model to be Logistic Regression with the accuracy of 81%. So will apply this model for prediction

PROJECT OUTCOME

On the basis of the accuracy we basically finalize the model for our prediction. Here in my project I have build my model with several classification techniques and choose the logistic regression because the accuracy in this category is more than 80%.

**## Logistic Regression**

from sklearn.linear\_model import LogisticRegression

# defining the logistic regression model

lreg = LogisticRegression()

# making prediction on the validation set

prediction = lreg.predict(X\_val)

from sklearn.metrics import accuracy\_score

# calculating the accuracy score

accuracy\_score(y\_val, prediction)

0.8116883116883117

We got an accuracy score of around 81% on the validation dataset. Logistic regression has a linear decision boundary.

CONCLUSION

In this project I have analysis the early prediction of diabetes by taking all the related factors in its tests and implementing using data science techniques by extracting knowledge from our real health care medical dataset to predict diabetic patients thus I have done our experiment using some of the various data science algorithms namely Random forest, decision tree, logistic regression, to predict dataset.

The main aim of this project was to design and implement diabetes prediction using data science and performance analysis of the methods and it has been achieved successfully.

The proposed approach uses various classification and ensemble learning method in which RANDOM FOREST, DECSION TREE XGBOOST CLASSIFIER and LOGISTIC REGRESSION.

In this project results can be asst health care to take early prediction and make early prediction and make early decision to cure diabetes and save human life.