**Predicting Occurrence of Diabetes in Pregnant Women**

by

R.Harini 18BCE1010

Aanya Jain 18BCE1067

Shraddha Nair 18BCE1070

A project report submitted to

**Dr. PATTABIRAMAN V**

**SCHOOL OF COMPUTING SCIENCE AND ENGINEERING**

in partial fulfilment of the requirements for the course of

**CSE3020 –DATA VISUALISATION**

in

**B.Tech. (Computer Science Engineering)**

****

**VIT UNIVERSITY, CHENNAI**

**Vandalur – Kelambakkam Road**

**Chennai – 600127**

**NOVEMBER 2018**

**CERTIFICATE**

Certified that this project report entitled “**Predicting the Occurrence of Diabetes in Pregnant Women”** is a bonafide work of R. Harini 18BCE1010, Aanya Jain 18BCE1067 and Shraddha Nair 18BCE1070 who carried out the “J”-Project work under my supervision and guidance for CSE3020-Data Visualisation .

**Dr. Pattabiraman V**

**SCHOOL OF COMPUTING SCIENCE AND ENGINEERING**

VIT University, Chennai

Chennai – 600 127.

**ABSTRACT**

## Diabetes is a problem that lot of pregnant women are facing so we are collecting the data based on previous occurrences and applying different prediction models on it. There are many factors that may cause the occurrence of diabetes like no. of pregnancies, blood pressure, glucose, BMI and age. A comparison of the accuracies of the different prediction models is also done to find the most apt model for the dataset.

## We tried to apply various different models like Decision Trees, Naïve Bayes, Logistic Regression, Random Forest and some more to correctly predict the output as 0 (no diabetes) or 1(has diabetes). Then plotted and visually analysed all the models with their plots to find correlation and dependence of different factors on the output variable.

**ACKNOWLEDGEMENT**

We wish to express our sincere thanks and deep sense of gratitude to our project faculty, **Dr. Pattabiraman V,** for his consistent encouragement and valuable guidance offered to us in a pleasant manner throughout the course of the project work.

Finally, we would like to thank our deemed university, VIT Chennai, for providing us with the opportunity and facilities which ensured this project’s completion.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **TOPIC** | **PAGE** |
| **1** | **ABSTRACT** | **3** |
| **2** | **INTRODUCTION** | **6** |
| **3** | **DATASET USED** | **7** |
| **4** | **IMPLEMENTATION CODE** | **8** |
| **5** | **VISUAL OUTPUT** | **17** |
| **6** | **CONCLUSION** | **20** |
| **7** | **REFERENCES** | **21** |
|  |  |  |
|  |  |  |
|  |  |  |

1. **INTRODUCTION**

Diabetes mellitus: More commonly referred to as "diabetes" -- a chronic disease associated with abnormally high levels of the sugar glucose in the blood. Although it has no cure but can still be predicted using the technology emerging today. Our project tries to predict the occurrence of diabetes in pregnant women by using various prediction and classification models like Logistic Regression, Decision Tree, Naïve Bayes, K Nearest Neighbours and Random Forest Model. In this, various factors are present in the dataset which are independent input variables like frequency of pregnancy, glucose level, blood pressure, skin thickness, insulin, BMI, Diabetes pedigree function and age. The output dependent variable is outcome which has values either 0 or 1,0 is for not having diabetes and 1 for having diabetes according to the given input parameters.

The project tries to compare the accuracy and predictions of different models by using accuracy score which tells the accuracy of every model used. We also plotted the graph of each input variable with the outcome variable and the plot of accuracy of each model. These accuracies and their plots helped to find the best and most apt model for our dataset.

This project helped a lot in developing Data Analysis and Machine Learning skills. With the help of those skills we tried to implement our project successfully and efficiently.

1. **DATASET USED**

## The data was collected and made available by “National Institute of Diabetes and Digestive and Kidney Diseases” as part of the [Pima Indians Diabetes Database](https://www.kaggle.com/uciml/pima-indians-diabetes-database/data). Several constraints were placed on the selection of these instances from a larger database. In particular, all patients here belong to the Pima Indian heritage and are females of ages 21 and above.

## We’ll be using Python and some of its popular data science related packages. First of all, we will import pandas to read our data from a CSV file and manipulate it for further use. We will also use numpy to convert out data into a format suitable to feed our classification model.

## The following features have been provided to help us predict whether a person is diabetic or not:

## Pregnancies: Number of times pregnant

## Glucose: Plasma glucose concentration over 2 hours in an oral glucose tolerance test

## BloodPressure: Diastolic blood pressure (mm Hg)

## SkinThickness: Triceps skin fold thickness (mm)

## Insulin: 2-Hour serum insulin (mu U/ml)

## BMI: Body mass index (weight in kg/(height in m)2)

## DiabetesPedigreeFunction: Diabetes pedigree function (a function which scores likelihood of diabetes based on family history)

## Age: Age (years)

## Outcome: Class variable (0 if non-diabetic, 1 if diabetic)

## The data set consists of record of 767 patients in total. To train our model we will be using 650 records. We will be using 100 records for testing, and the last 17 records to cross check our model.

**3. IMPLEMENTATION CODE**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

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

diabetes.columns

**Out[3]:**

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

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

dtype='object')

diabetes.head

**Out[4]:**

<bound method NDFrame.head of Pregnancies Glucose ... Age Outcome

0 6 148 ... 50 1

1 1 85 ... 31 0

2 8 183 ... 32 1

3 1 89 ... 21 0

4 0 137 ... 33 1

.. ... ... ... ... ...

763 10 101 ... 63 0

764 2 122 ... 27 0

765 5 121 ... 30 0

766 1 126 ... 47 1

767 1 93 ... 23 0

[768 rows x 9 columns]>

diabetes.describe()

**Out[5]:**

Pregnancies Glucose ... Age Outcome

count 768.000000 768.000000 ... 768.000000 768.000000

mean 3.845052 120.894531 ... 33.240885 0.348958

std 3.369578 31.972618 ... 11.760232 0.476951

min 0.000000 0.000000 ... 21.000000 0.000000

25% 1.000000 99.000000 ... 24.000000 0.000000

50% 3.000000 117.000000 ... 29.000000 0.000000

75% 6.000000 140.250000 ... 41.000000 1.000000

max 17.000000 199.000000 ... 81.000000 1.000000

[8 rows x 9 columns]

feature\_names=['Pregnancies','Glucose','BloodPressure','SkinThickness','Insulin','BMI','DiabetesPedigreeFunction','Age']

X=diabetes[feature\_names]

y=diabetes.Outcome

print("Diabetes data set dimensions:{}".format(diabetes.shape))

Diabetes data set dimensions:(768, 9)

corr=diabetes.corr()

print(corr)

Pregnancies Glucose ... Age Outcome

Pregnancies 1.000000 0.129459 ... 0.544341 0.221898

Glucose 0.129459 1.000000 ... 0.263514 0.466581

BloodPressure 0.141282 0.152590 ... 0.239528 0.065068

SkinThickness -0.081672 0.057328 ... -0.113970 0.074752

Insulin -0.073535 0.331357 ... -0.042163 0.130548

BMI 0.017683 0.221071 ... 0.036242 0.292695

DiabetesPedigreeFunction -0.033523 0.137337 ... 0.033561 0.173844

Age 0.544341 0.263514 ... 1.000000 0.238356

Outcome 0.221898 0.466581 ... 0.238356 1.000000

[9 rows x 9 columns]

sns.heatmap(corr, xticklabels=corr.columns, yticklabels=corr.columns)

**Out[11]:** <matplotlib.axes.\_subplots.AxesSubplot at 0x26664662438>

sns.countplot(x=diabetes['Age'])

**Out[12]:** <matplotlib.axes.\_subplots.AxesSubplot at 0x26664662438>

sns.countplot(x=diabetes['Age'], hue=diabetes['Outcome'])

**Out[13]:** <matplotlib.axes.\_subplots.AxesSubplot at 0x26664662438>

sns.countplot(diabetes['Outcome'])

**Out[14]:** <matplotlib.axes.\_subplots.AxesSubplot at 0x26664662438>

plt.scatter(diabetes['Age'], diabetes['BMI'])

plt.xlabel('Age')

plt.ylabel('BMI')

diabetes.groupby('Outcome').size()

diabetes.hist(bins=50, figsize=(20,15))

plt.show()

#Data cleaning

diabetes.isnull().sum()

**Out[20]:**

Pregnancies 0

Glucose 0

BloodPressure 0

SkinThickness 0

Insulin 0

BMI 0

DiabetesPedigreeFunction 0

Age 0

Outcome 0

dtype: int64

diabetes.isna().sum()

Out[21]:

Pregnancies 0

Glucose 0

BloodPressure 0

SkinThickness 0

Insulin 0

BMI 0

DiabetesPedigreeFunction 0

Age 0

Outcome 0

dtype: int64

median\_bmi=diabetes['BMI'].median()

diabetes['BMI']=diabetes['BMI'].replace(to\_replace=0, value=median\_bmi)

median\_bloodp=diabetes['BloodPressure'].median()

diabetes['BloodPressure']=diabetes['BloodPressure'].replace(to\_replace=0, value=median\_bloodp)

median\_gluc=diabetes['Glucose'].median()

diabetes['Glucose']=diabetes['Glucose'].replace(to\_replace=0, value=median\_gluc)

median\_skin=diabetes['SkinThickness'].median()

diabetes['SkinThickness']=diabetes['SkinThickness'].replace(to\_replace=0, value=median\_skin)

median\_ins=diabetes['Insulin'].median()

diabetes['Insulin']=diabetes['Insulin'].replace(to\_replace=0, value=median\_ins)

**#Splitting**

median\_bmi=diabetes['BMI'].median()

diabetes['BMI']=diabetes['BMI'].replace(to\_replace=0, value=median\_bmi)

median\_bloodp=diabetes['BloodPressure'].median()

diabetes['BloodPressure']=diabetes['BloodPressure'].replace(to\_replace=0, value=median\_bloodp)

median\_gluc=diabetes['Glucose'].median()

diabetes['Glucose']=diabetes['Glucose'].replace(to\_replace=0, value=median\_gluc)

median\_skin=diabetes['SkinThickness'].median()

diabetes['SkinThickness']=diabetes['SkinThickness'].replace(to\_replace=0, value=median\_skin)

median\_ins=diabetes['Insulin'].median()

diabetes['Insulin']=diabetes['Insulin'].replace(to\_replace=0, value=median\_ins)

#**Model Selection**

from sklearn.model\_selection import cross\_val\_score

from sklearn.metrics import accuracy\_score

1. **K-Nearest Neighbors Model**

#Applying K-Nearest Neighbors Model

from sklearn.neighbors import KNeighborsClassifier

classifier1=KNeighborsClassifier()

classifier1.fit(X\_train, y\_train)

y\_pred=classifier1.predict(X\_test)

from sklearn.metrics import confusion\_matrix

from sklearn.metrics import f1\_score

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

print(conf\_matrix)

[[104 21]

[ 26 41]]

print(f1\_score(y\_test,y\_pred))

0.6356589147286821

scores.append(accuracy\_score(y\_test, y\_pred))

names.append("K-Nearest Neighbor")

1. **Naïve Bayes Model**

#Applying Naïve Bayes Model

from sklearn.naive\_bayes import GaussianNB

classifier2=GaussianNB()

classifier2.fit(X\_train,y\_train)

y\_pred=classifier2.predict(X\_test)

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

print(conf\_matrix)

[[105 20]

[ 25 42]]

print(f1\_score(y\_test,y\_pred))

0.6511627906976745

scores.append(accuracy\_score(y\_test,y\_pred))

names.append("Naive Bayes")

1. **Logistic Regression**

#Applying Logistic Regression Model

from sklearn.linear\_model import LogisticRegression

classifier3=LogisticRegression()

classifier3.fit(X\_train, y\_train)

y\_pred=classifier3.predict(X\_test)

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

print(conf\_matrix)

[[110 15]

[ 27 40]]

print(f1\_score(y\_test,y\_pred))

0.6557377049180327

1. **Decision Tree Model**

#Applying Decision Tree Model

from sklearn.tree import DecisionTreeClassifier

classifier4=DecisionTreeClassifier(random\_state=0)

classifier4.fit(X\_train, y\_train)

y\_pred=classifier4.predict(X\_test)

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

print(conf\_matrix)

[[101 24]

[ 22 45]]

print(f1\_score(y\_test,y\_pred))

0.6617647058823529

scores.append(accuracy\_score(y\_test, y\_pred))

names.append('Decision Tree')

1. **Random Forest Model**

#Applying Random Forest Model

from sklearn.ensemble import RandomForestClassifier

rf=RandomForestClassifier(n\_estimators=100, random\_state=0)

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

y\_pred=rf.predict(X\_test)

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

print(conf\_matrix)

[[110 15]

[ 22 45]]

print(f1\_score(y\_test,y\_pred))

0.7086614173228347

scores.append(accuracy\_score(y\_test, y\_pred))

names.append('Random Forest')

**Comparison**

Final=pd.DataFrame({'Name':names,'Score':scores})

print(Final)

Name Score

0 K-Nearest Neighbor 0.755208

1 Naive Bayes 0.765625

2 Logistic Regression 0.781250

3 Decision Tree 0.760417

4 Random Forest 0.807292

axis=sns.barplot(x='Name', y='Score', data=Final)

axis.set(xlabel='Model',ylabel='Accuracy')

**Out[57]:**

[Text(2548.3916666666664, 0.5, 'Accuracy'),

Text(0.5, 158.1444444444444, 'Model')]

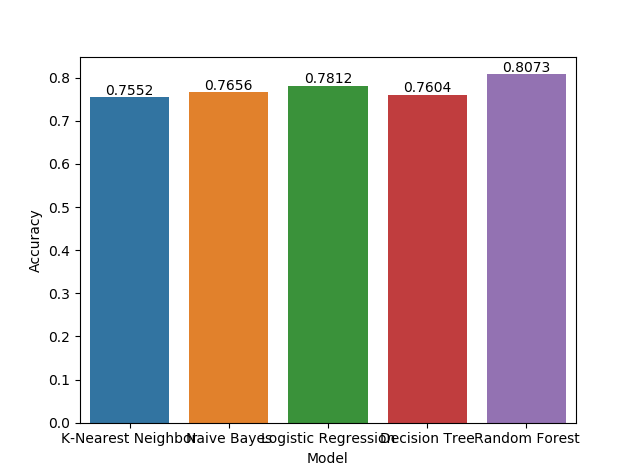
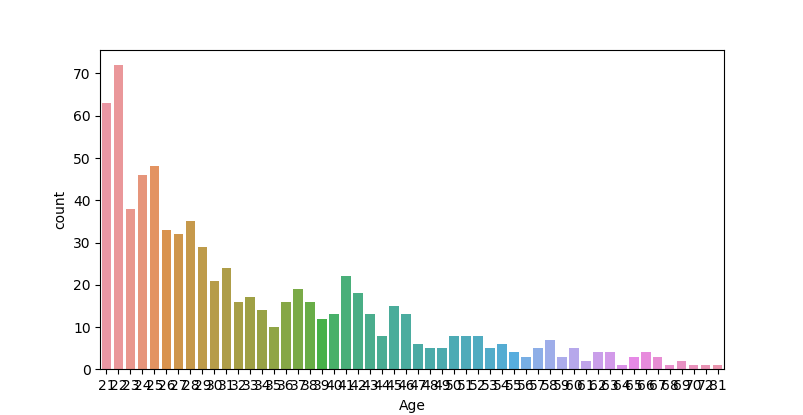
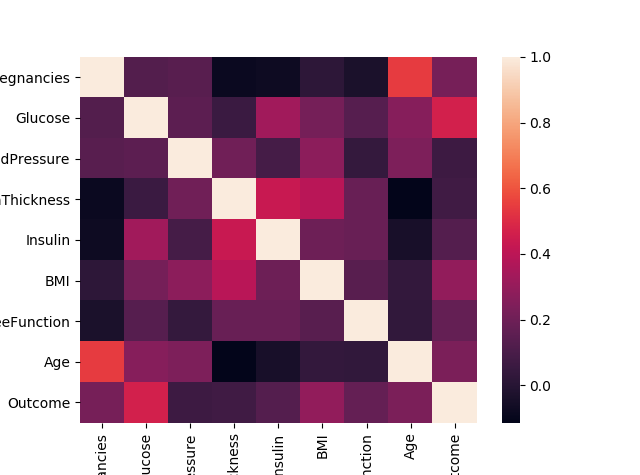
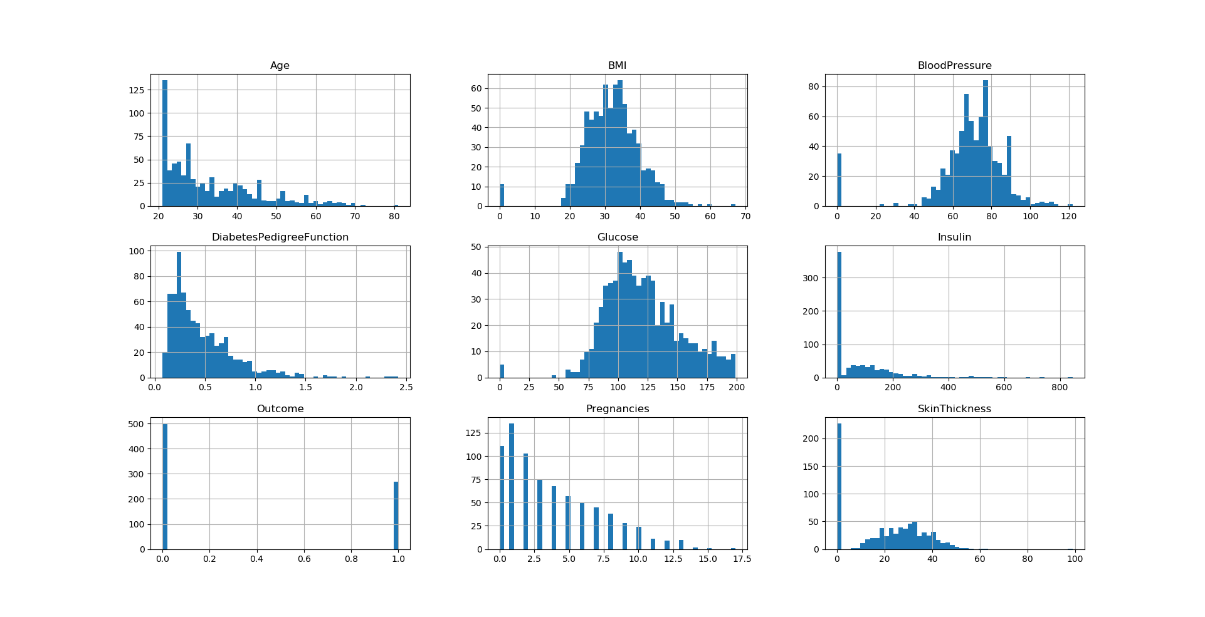
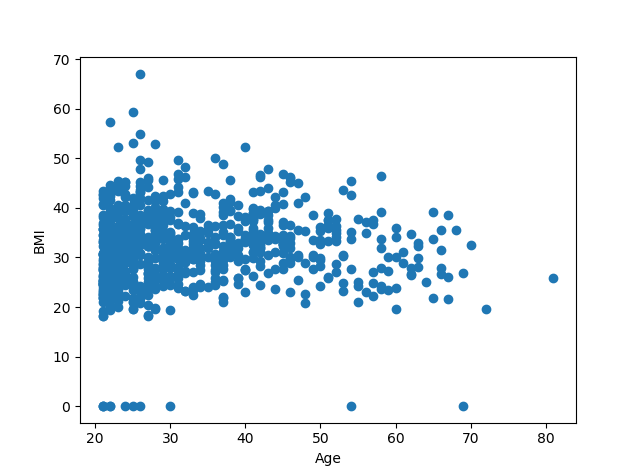
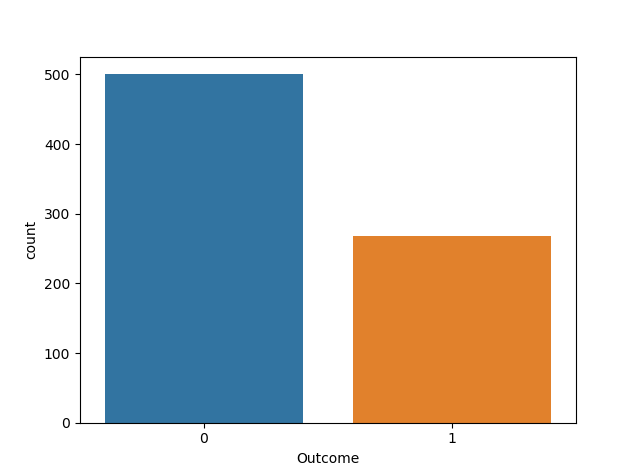
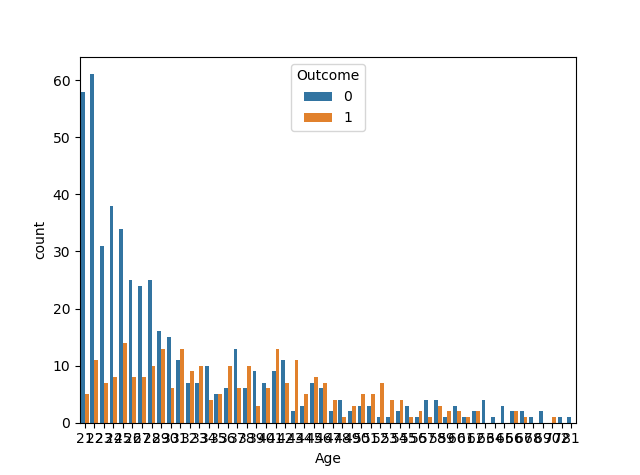
for p in axis.patches:

height=p.get\_height()

axis.text(p.get\_x()+p.get\_width()/2,height+0.005,'{:1.4f}'.format(height), ha="center")

plt.show()

**4. VISUAL OUTPUT**

****

**5. CONCLUSION**

Based on the final graph that was plotted based on the accuracy score of each classifier, we can see that K-Nearest Neighbours algorithm has the lowest accuracy of 75.5%.

However, it can also be seen that Random Forest shows the highest accuracy of 80.73%.  
  
Hence we can conclude that with the help of the visualization of the correlation between various features and comparison of the accuracy scores, Random Forest has the highest accuracy.

**6. REFERENCES**

* <https://towardsdatascience.com/data-pre-processing-techniques-you-should-know-8954662716d6>
* <https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc>
* <https://towardsdatascience.com/random-forest-in-python-24d0893d51c0>
* <https://towardsdatascience.com/data-visualization-for-machine-learning-and-data-science-a45178970be7>
* <https://www.researchgate.net/figure/Techniques-for-visualizing-classification-results-a-an-interactive-confusion-matrix_fig1_270789956>
* <https://www.researchgate.net/publication/270789956_Visual_Methods_for_Analyzing_Probabilistic_Classification_Data>