DATA70121 COURSEWORK

EDA & REGRESSION

ID-11349153

PREDICTING THE CHANCES OF A PERSON DEVELOPING DIABETES USING REGRESSION

ABSTRACT

Diabetes is a dangerous medical condition that has long caused concern among the masses. It can lead to death or cause blindness. We can now combat this illness more effectively than before, sparing lives, thanks to recent developments in medicine combined with technology.

INTRODUCTION

In this analysis, we will be dealing with the PimaDiabetes dataset collected from the USA's National Institute of Diabetes and Digestive and Kidney Diseases[1]. It comprises diagnostic measures of 750 women along with a variable that determines if they tested positive for diabetes eventually. Measures taken into consideration are:-

Glucose-Plasma glucose levels (mg/dl)

SkinThickness-Skin thickness (mm)

Pregnancies: Frequency of pregnancy

Blood Pressure- Diastolic blood pressure (mm Hg)

Serum Insulin- insulin concentration2 (μ U/ml) at 2 hours in an OGTT

BMI- body mass index (weight in kg)/(height in m)2

Age-Age in years

Diabetes Pedigree: Numerical score to evaluate genetic influence of both the woman's diabetic and nondiabetic relatives on diabetes risk: higher scores imply close relatives with diabetes.

Outcome-1 if the person tests positive for diabetes, 0 otherwise

DATA

While working with datasets we need to consider their quality too. Quality is a subjective term but there are some factors that help us in determining it, namely consistency(No data discrepancy), accuracy, completeness(ensuring adequate data is present), timeliness(check for timely records), reliability(if data can be trusted), relevance, validity. We can rate this dataset a 3* based on these criteria; but, as it is clinical data, we cannot be totally certain of its accuracy because it contains minute details that could have an impact on the analysis even though the source is reliable.

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigree	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
...	...	...	...	...	...	...	...	...	...
745	12	100	84	33	105	30.0	0.488	46	0
746	1	147	94	41	0	49.3	0.358	27	1
747	1	81	74	41	57	46.3	1.096	32	0
748	3	187	70	22	200	36.4	0.408	36	1
749	6	162	62	0	0	24.3	0.178	50	1

750 rows × 9 columns

Fig.1 *PimaDiabetes Dataframe*

From the figure it's clearly visible that there are some '0's in some columns which doesn't make sense. "Skinthickness" and "Insulin" for a person can't be 0. Same goes with other variables too except "outcome" which only has binary values and pregnancies which can be 0.

DATA CLEANING

To handle such issues we require data cleaning. There are 3 options in this -either we can remove that row containing 0 or NAN, or, impute those values with mean/median, or we can do transformation and capping. Here I have imputed with mean to retain the original dataset size. Also, since this is a medical dataset removing columns/rows containing 0s would have impacted the analysis because in medical cases extreme values(outliers) are quite significant other than the normal values.

```
zero = data.eq(0)

# Check for 0
zero_check = zero.any()

# Count the number of 0
zero_count = zero.sum()
```

```
zero_count
```

```
Pregnancies          109
Glucose                5
BloodPressure         35
SkinThickness        221
Insulin              362
BMI                   11
DiabetesPedigree       0
Age                    0
Outcome              490
dtype: int64
```

Fig2. Count of 0s in each column

```
data.isnull()=="TRUE"
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigree	Age	Outcome
0	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...
745	False	False	False	False	False	False	False	False	False
746	False	False	False	False	False	False	False	False	False
747	False	False	False	False	False	False	False	False	False
748	False	False	False	False	False	False	False	False	False
749	False	False	False	False	False	False	False	False	False

750 rows × 9 columns

Fig3.No null values present

Performing Imputation

```
mean = data.iloc[:, :-1].mean()
data.iloc[:, :-1] = data.iloc[:, :-1].apply(lambda col: col.replace(0, mean[col.name]))
```

```
mean
```

```
Pregnancies              3.844000
Glucose                120.737333
BloodPressure           68.982667
SkinThickness           20.489333
Insulin                 80.378667
BMI                     31.959067
DiabetesPedigree         0.473544
Age                     33.166667
dtype: float64
```

```
data
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigree	Age	Outcome
0	6.000	148.0	72.0	35.000000	80.378667	33.6	0.627	50	1
1	1.000	85.0	66.0	29.000000	80.378667	26.6	0.351	31	0
2	8.000	183.0	64.0	20.489333	80.378667	23.3	0.672	32	1
3	1.000	89.0	66.0	23.000000	94.000000	28.1	0.167	21	0
4	3.844	137.0	40.0	35.000000	168.000000	43.1	2.288	33	1
...	...	...	...	...	...	...	...	...	...
745	12.000	100.0	84.0	33.000000	105.000000	30.0	0.488	46	0
746	1.000	147.0	94.0	41.000000	80.378667	49.3	0.358	27	1
747	1.000	81.0	74.0	41.000000	57.000000	46.3	1.096	32	0
748	3.000	187.0	70.0	22.000000	200.000000	36.4	0.408	36	1
749	6.000	162.0	62.0	20.489333	80.378667	24.3	0.178	50	1

750 rows × 9 columns

Fig4.Dataset after imputation

No missing or 0 values in independent variables now. Thus, we can go ahead with the analysis.

Exploratory Data Analysis

To obtain a statistical summary of the dataset we can use the describe function.

```
data.describe()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigree	Age	Outcome
count	750.000000	750.000000	750.000000	750.000000	750.000000	750.000000	750.000000	750.000000	750.000000
mean	4.402661	121.542249	72.201858	26.526857	119.174770	32.427800	0.473544	33.166667	0.346667
std	2.982381	30.451560	12.159438	9.645660	92.734709	6.900969	0.332119	11.708872	0.476226
min	1.000000	44.000000	24.000000	7.000000	14.000000	18.200000	0.078000	21.000000	0.000000
25%	2.000000	99.000000	64.000000	20.489333	80.378667	27.500000	0.244000	24.000000	0.000000
50%	3.844000	117.000000	72.000000	23.000000	80.378667	32.000000	0.377000	29.000000	0.000000
75%	6.000000	140.750000	80.000000	32.000000	129.750000	36.575000	0.628500	40.750000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

Fig5.Dataset summary

From here we get to know the five-point summary of dataset comprising of mean, standard deviation, max, min, and the 3 quartile for each variable.

//Distribution of the dependent variable “Outcome” using count plot.

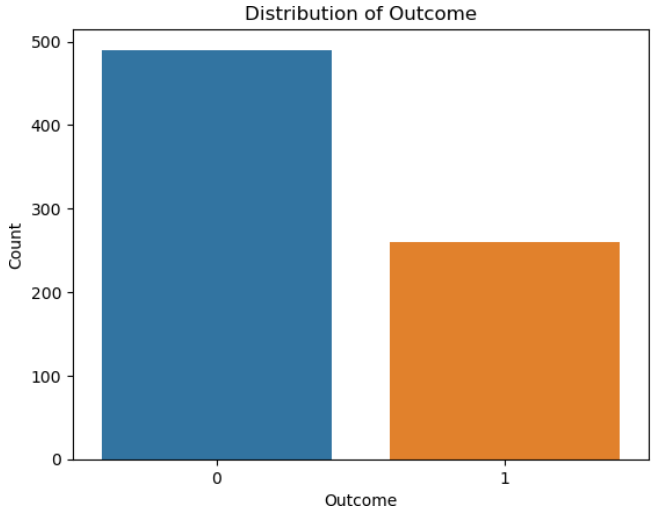


Fig6.Outcome Distribution

The orange bar signifies the count of true values whereas blue signifies the false values under outcome. So, the chances of developing diabetes are relatively low.

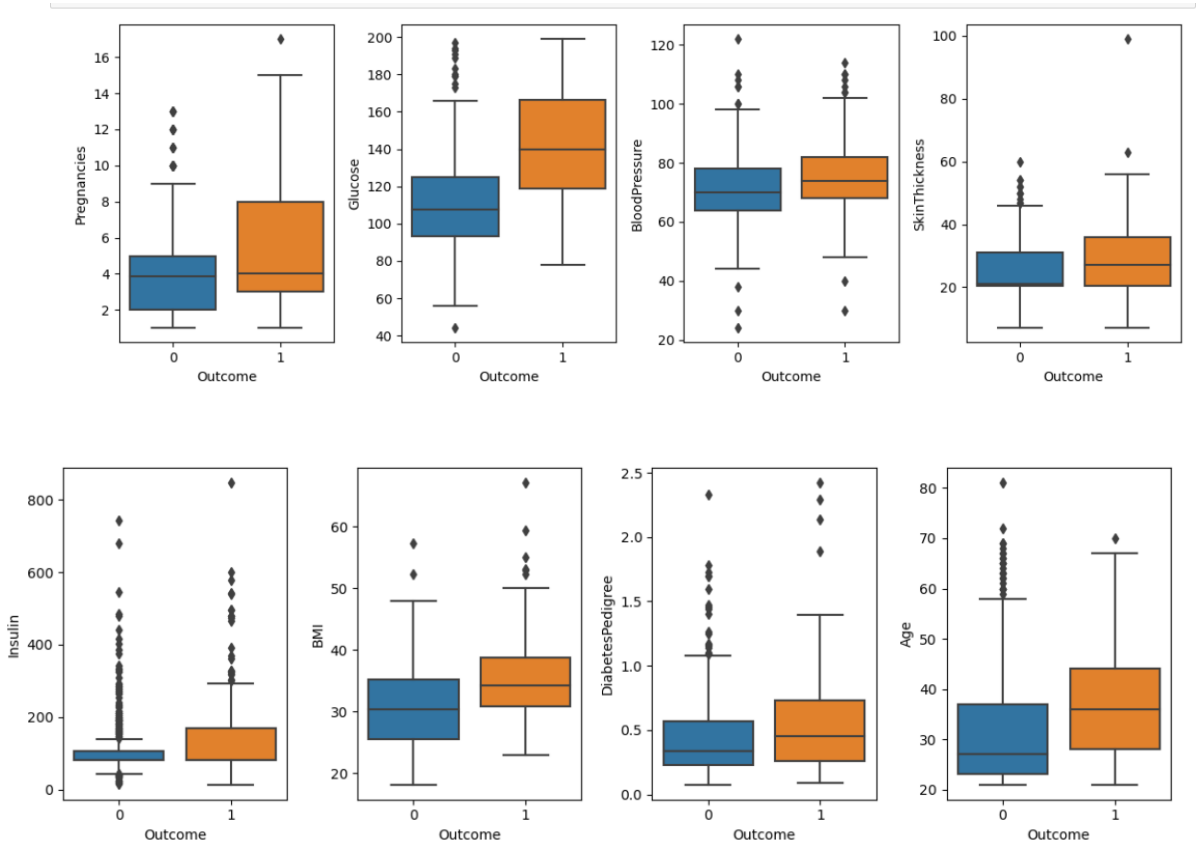


Fig7.Box & Whisker plot

Using a box and whisker plot we can evaluate the relationships of each variable with the dependent variable outcome. Also, we can detect the outliers using this graph i.e. all the points outside the whisker. Based on the graph we can also infer that glucose, pregnancies, and diabetes pedigree are the variables that have affected the outcome variable mostly.

Now to dive deep into the relations between variables we can check the correlation matrix.

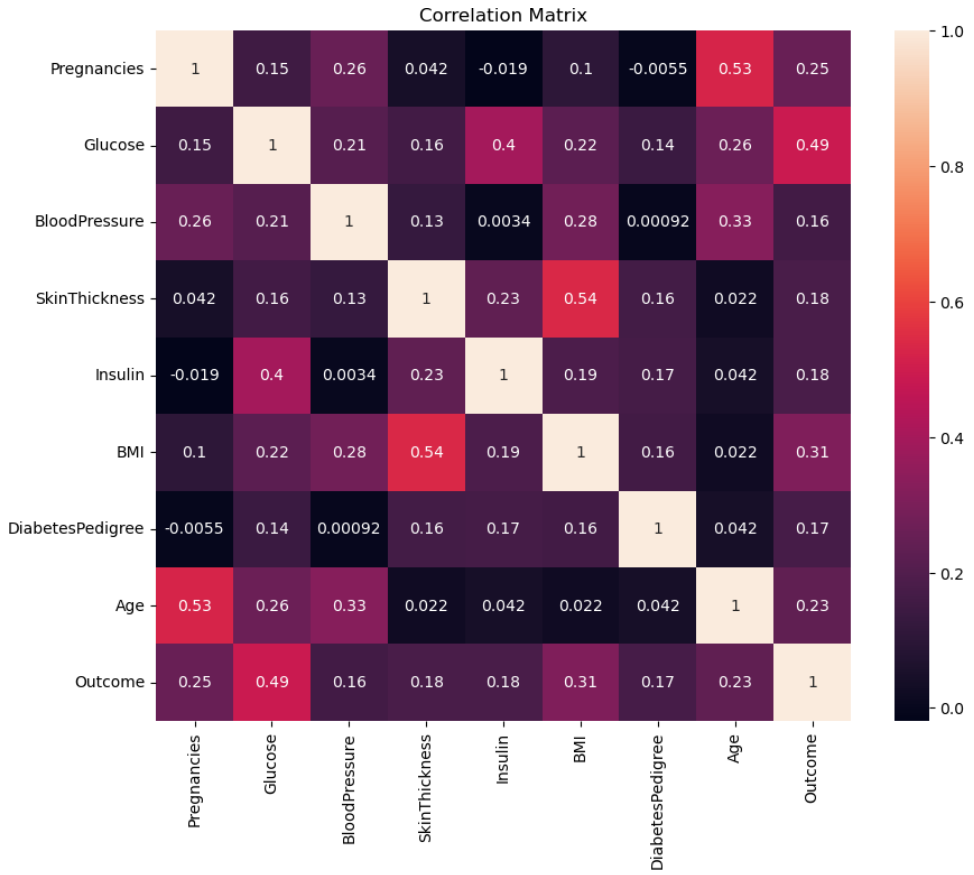


Fig8.Correlation Matrix

Here the diagonal values are 1 because those are cross-products of the same variable. Other than that as per the hue index we can say that darker shades signify the least correlation and the lighter the shade higher will be the relation between them. Here Glucose has highest level of correlation with outcome while blood pressure has lowest which means that it hardly has any impact on the chances of person being diabetic. Similarly, as per shades, we can determine relationships between other variables as well.

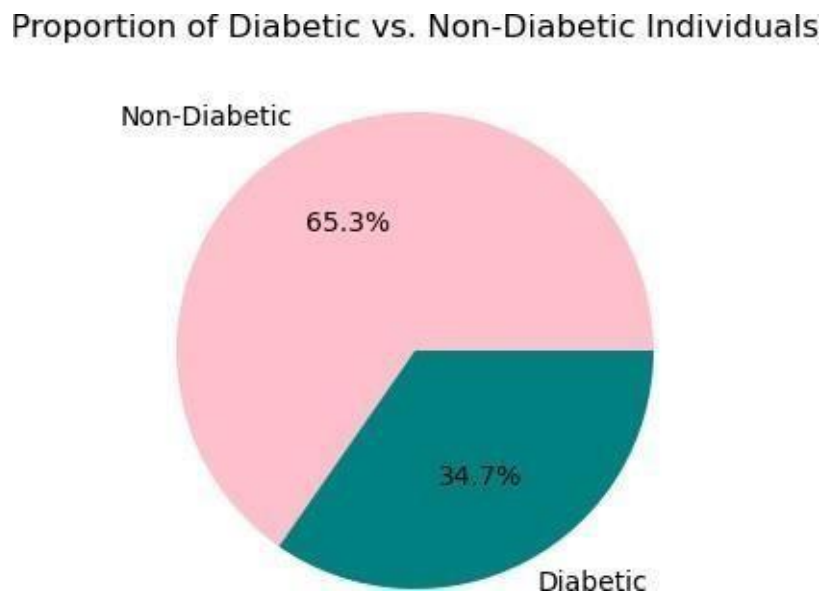


Fig9.Ratio of diabetic vs non-diabetic individuals in this training dataset

Adding New Column-Seven or more pregnancies

```
import pandas as pd
import statsmodels.api as sm

# Adding column 'SevenOrMorePregnancies'
data['SevenOrMorePregnancies'] = data['Pregnancies'] >= 7

# Typecasting
data['SevenOrMorePregnancies'] = data['SevenOrMorePregnancies'].astype(int)

# predictor and independent variables
X = data['SevenOrMorePregnancies']
y = data['Outcome']

X=sm.add_constant(X)

# logistic regression model
log = sm.Logit(y, X)
reg = log.fit()


print(reg.summary())
```

In this way, we added a new variable by ensuring that the condition is satisfied (pregnancies>7). Since it had Boolean values, we changed the type to int so that it could store 0 and 1. Then we took the new variable as a single predictor for outcome and fitted that into a logistic regression model. Here we are using logistic regression because values are binary and non-continuous. From the regression function, we can view a summary of the model.

```
Optimization terminated successfully.
         Current function value: 0.616518
         Iterations 5
                           Logit Regression Results
==============================================================================
Dep. Variable:                Outcome   No. Observations:                  750
Model:                          Logit   Df Residuals:                      748
Method:                           MLE   Df Model:                            1
Date:                Thu, 16 Nov 2023   Pseudo R-squ.:                 0.04469
Time:                        15:24:05   Log-Likelihood:                -462.39
converged:                       True   LL-Null:                       -484.02
Covariance Type:            nonrobust   LLR p-value:                 4.791e-11
============================================================================================
                               coef    std err          z      P>|z|      [0.025      0.975]
--------------------------------------------------------------------------------------------
const                       -0.9199      0.092    -10.052      0.000      -1.099      -0.741
SevenOrMorePregnancies       1.1898      0.182      6.529      0.000       0.833       1.547
============================================================================================
```

Fig.10 Regression result for model using single predictor

Fig.10 suggests that the odds of the outcome increase by approximately 1.1898(coefficient) times for each unit increase in the "SevenOrMorePregnancies" variable.

```
#  probability of diabetes when  6 or fewer pregnancies
lt6 = reg.params['const'] + reg.params['SevenOrMorePregnancies'] * 0

# probability of diabetes when 7 or more pregnancies
mt7 = reg.params['const'] + reg.params['SevenOrMorePregnancies'] * 1

# logistic function to get probabilities
lt6 = 1 / (1 + 2.71828 ** (-lt6))
mt7 = 1 / (1 + 2.71828 ** (-mt7))

# Printing probabilities
print(f"Probability of diabetes when 6 or fewer pregnancies: {lt6:.3f}")
print(f"Probability of diabetes when 7 or more pregnancies: {mt7:.3f}")
```

Then we calculate the probability with the help of the above model to determine if the person will develop diabetes based on a 6 or 7 pregnancy count.

Predicting Diabetes based on the test dataset

Here we will try to create multiple regression models and see which model works best in predicting the outcome. Things to look for will be the accuracy of the model and how relevant are the explanatory variables with respect to outcome.

Model 1

```
#REGRESSION MODEL 1

import statsmodels.api as sm
import pandas as pd


X_train = data.drop(['Outcome', 'SevenOrMorePregnancies'], axis=1)
X_train = sm.add_constant(X_train)  # constant intercept
y_train = data['Outcome']

# regression model
logit_model = sm.Logit(y_train, X_train)
f = logit_model.fit()

# Test dataset
df2=pd.read_csv("C:/Users/abhin/Downloads/ToPredict.csv")


X_test = sm.add_constant(df2)  # intercept

# Prediction for outcome
prob = f.predict(X_test)

# for statistical summary
print(f.summary())

print("Predictions of outcome:\n", prob)
```

Here I have trained the model with the original dataset excluding “outcome” and “sevenormore pregnancies”. Added the constant intercept to form the equation. Post training I tested the modelwith the “ToPredict” dataset and then found out the probability of outcome using predict function.

Results:-

```
Optimization terminated successfully.
         Current function value: 0.464576
         Iterations 6
                           Logit Regression Results
==============================================================================
Dep. Variable:                Outcome   No. Observations:                  750
Model:                          Logit   Df Residuals:                      741
Method:                           MLE   Df Model:                            8
Date:                Thu, 16 Nov 2023   Pseudo R-squ.:                 0.2801
Time:                        21:04:58   Log-Likelihood:                -348.43
converged:                       True   LL-Null:                       -484.02
Covariance Type:            nonrobust   LLR p-value:                5.539e-54
==============================================================================
                     coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const              -8.9630      0.819    -10.947      0.000     -10.568      -7.358
Pregnancies         0.1503      0.037      4.106      0.000       0.079       0.222
Glucose             0.0378      0.004      9.648      0.000       0.030       0.045
BloodPressure      -0.0116      0.009     -1.330      0.184      -0.029       0.005
SkinThickness       0.0001      0.012      0.009      0.993      -0.023       0.023
Insulin            -0.0013      0.001     -1.237      0.216      -0.003       0.001
BMI                 0.0934      0.018      5.292      0.000       0.059       0.128
DiabetesPedigree    0.8271      0.303      2.730      0.006       0.233       1.421
Age                 0.0123      0.009      1.309      0.191      -0.006       0.031
==============================================================================
Predicted Probabilities:
 0    0.531493
1    0.261701
2    0.097214
3    0.675882
4    0.750536
dtype: float64
```

Fig.11 Regression results for model1

Model 2

```
# REGRESSION MODEL 2

#choosing explanatory variables
X_train2 = data[['Glucose', 'Pregnancies', 'DiabetesPedigree']]
X_train2 = sm.add_constant(X_train2)  # intercept
y_train = data['Outcome']

# logistic regression model
logit_model_2 = sm.Logit(y_train, X_train2)
result_2 = logit_model_2.fit()


df2_2 =df2[['Glucose', 'Pregnancies', 'DiabetesPedigree']]


X_test_2 = sm.add_constant(df2_2)  # intercept


prob2 = result_2.predict(X_test_2)

# statistical summary
print(result_2.summary())

print("Predictions of outcome (Model 2):\n", prob2)
```

Results:

```
Optimization terminated successfully.
         Current function value: 0.490773
         Iterations 6
                           Logit Regression Results
==============================================================================
Dep. Variable:                Outcome   No. Observations:                  750
Model:                          Logit   Df Residuals:                      746
Method:                           MLE   Df Model:                            3
Date:                Sat, 18 Nov 2023   Pseudo R-squ.:                  0.2395
Time:                        11:44:30   Log-Likelihood:                -368.08
converged:                       True   LL-Null:                       -484.02
Covariance Type:            nonrobust   LLR p-value:                 5.427e-50
==============================================================================
                     coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const             -6.6563      0.502    -13.258      0.000      -7.640      -5.672
Glucose            0.0383      0.003     11.051      0.000       0.032       0.045
Pregnancies        0.1673      0.031      5.482      0.000       0.107       0.227
DiabetesPedigree   0.9485      0.286      3.314      0.001       0.388       1.509
==============================================================================
Predicted Probabilities (Model 2):
 0    0.585163
1    0.167152
2    0.141181
3    0.619739
4    0.731394
dtype: float64
```

Fig.12 Regression results for model 2

Model 3

```
# REGRESSION MODEL 3

#choosing different explanatory variables
X_train3 = data[['Age', 'BMI', 'DiabetesPedigree','Pregnancies']]
X_train3 = sm.add_constant(X_train3)  # Add a constant term for the intercept
y_train = data['Outcome']

# Fit the logistic regression model
logit_model_3 = sm.Logit(y_train, X_train3)
result_3 = logit_model_3.fit()


df3 =df2[['Age', 'BMI', 'DiabetesPedigree','Pregnancies']]


X_test3 = sm.add_constant(df3)


# outcome prediction
prob3 = result_3.predict(X_test3)

# Display the result summary for more information
print(result_3.summary())

# Display predicted probabilities
print("Predictions of outcome (Model 3):\n", prob3)
```

Results:

```
Optimization terminated successfully.
         Current function value: 0.550405
         Iterations 6
                           Logit Regression Results
==============================================================================
Dep. Variable:                Outcome   No. Observations:                  750
Model:                          Logit   Df Residuals:                      745
Method:                           MLE   Df Model:                            4
Date:                Sat, 18 Nov 2023   Pseudo R-squ.:                  0.1471
Time:                        11:48:17   Log-Likelihood:                -412.80
converged:                       True   LL-Null:                       -484.02
Covariance Type:            nonrobust   LLR p-value:                 8.518e-30
==============================================================================
                     coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const             -5.9823      0.565    -10.586      0.000      -7.090      -4.875
Age                0.0301      0.008      3.695      0.000       0.014       0.046
BMI                0.1005      0.013      7.465      0.000       0.074       0.127
DiabetesPedigree   0.9138      0.258      3.542      0.000       0.408       1.420
Pregnancies        0.1207      0.032      3.716      0.000       0.057       0.184
==============================================================================
Predicted Probabilities (Model 3):
 0    0.349188
1    0.296900
2    0.113875
3    0.343743
4    0.499456
dtype: float64
```

Fig.13 Regression results for model 3

Details of 3 models: -

Model 1:

- **Pseudo R-squared:** 0.2801
- **Number of Variables:** 8
- **Log-Likelihood:** -348.43
- **LLR p-value:** 5.539e-54

Model 2:

- **Pseudo R-squared:** 0.2395
- **Number of Variables:** 3
- **Log-Likelihood:** -368.08
- **LLR p-value:** 5.427e-50

Model 3:

- **Pseudo R-squared:** 0.1471
- **Number of Variables:** 4
- **Log-Likelihood:** -412.80
- **LLR p-value:** 8.518e-30

Pseudo R-squared: Measures variation of independent variables on dependent variables.

Log-Likelihood: Evaluates how well a model can fit the variables.

LLR p-value: Used for comparing current model with null model.

CONCLUSION

Comparing the details above I can say that Model 1 with eight variables and a pseudo-R-squared of 0.2801, best explains the variable-outcome relation. The likelihood ratio test (LLR) p-value is low at 5.539e-54, and the log-likelihood is -348.43, showing a considerable improvement over a null model. So chose model 1 as the final regression model. As per this model, the 1st,4th and 5th person have higher chances of developing diabetes, which is evident from their diagnostic measures as well.

CODE

In [102…
```
import pandas as pd
import matplotlib.pyplot as plt
data = pd.read_csv("C:/Users/abhin/Downloads/PimaDiabetes.csv")
```

In [103…
```
data
```

Out[103]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigree	Age	Outcc
0	6	148	72	35	0	33.6	0.627	50	
1	1	85	66	29	0	26.6	0.351	31	
2	8	183	64	0	0	23.3	0.672	32	
3	1	89	66	23	94	28.1	0.167	21	
4	0	137	40	35	168	43.1	2.288	33	
…	…	…	…	…	…	…	…	…	
745	12	100	84	33	105	30.0	0.488	46	
746	1	147	94	41	0	49.3	0.358	27	
747	1	81	74	41	57	46.3	1.096	32	
748	3	187	70	22	200	36.4	0.408	36	
749	6	162	62	0	0	24.3	0.178	50	

750 rows × 9 columns

In [104…
```
mean = data.iloc[:, :-1].mean()
data.iloc[:, :-1] = data.iloc[:, :-1].apply(lambda col: col.replace(0, mean[col.name])
```

In [105…
```
data.describe()
```

Out[105]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigr
count	750.000000	750.000000	750.000000	750.000000	750.000000	750.000000	750.0000
mean	4.402661	121.542249	72.201858	26.526857	119.174770	32.427800	0.4735
std	2.982381	30.451560	12.159438	9.645660	92.734709	6.900969	0.3321
min	1.000000	44.000000	24.000000	7.000000	14.000000	18.200000	0.0780
25%	2.000000	99.000000	64.000000	20.489333	80.378667	27.500000	0.2440
50%	3.844000	117.000000	72.000000	23.000000	80.378667	32.000000	0.3770
75%	6.000000	140.750000	80.000000	32.000000	129.750000	36.575000	0.6285
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.4200

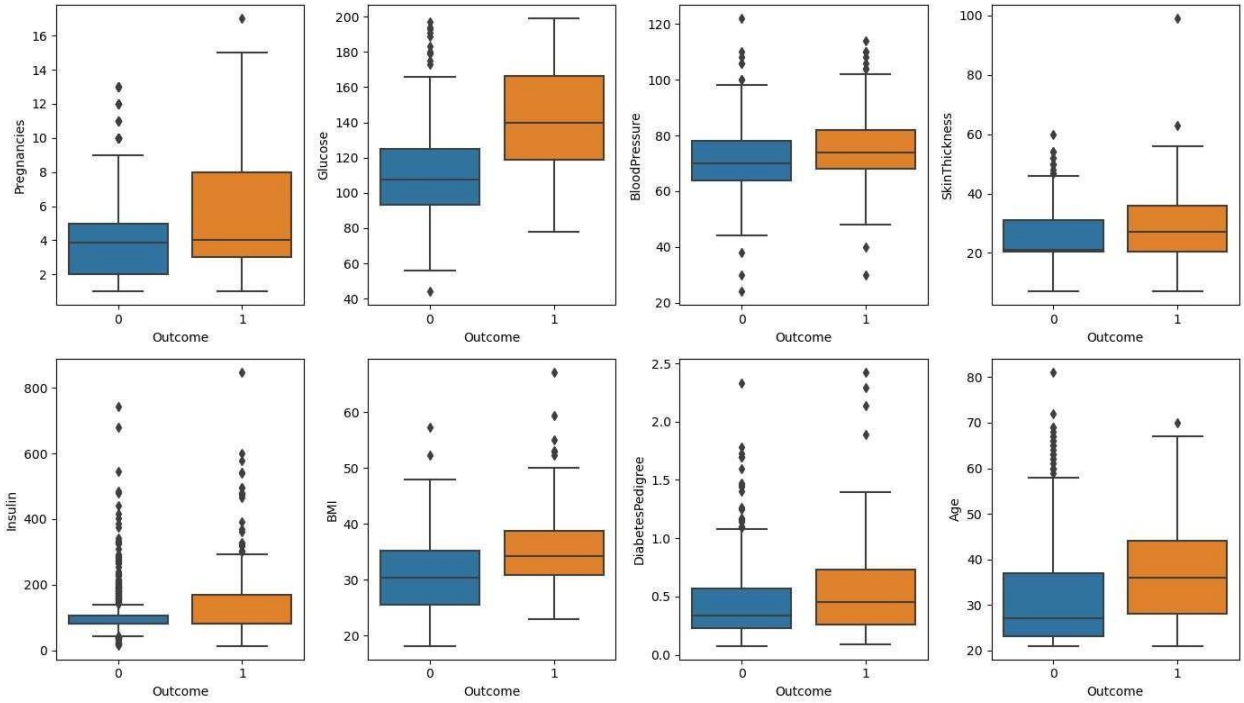
In [58]:
```
data
```

Out[58]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigree	Age	O
0	6.000	148.0	72.0	35.000000	80.378667	33.6	0.627	50	
1	1.000	85.0	66.0	29.000000	80.378667	26.6	0.351	31	
2	8.000	183.0	64.0	20.489333	80.378667	23.3	0.672	32	
3	1.000	89.0	66.0	23.000000	94.000000	28.1	0.167	21	
4	3.844	137.0	40.0	35.000000	168.000000	43.1	2.288	33	
...	...	...	...	...	...	...	...	...	
745	12.000	100.0	84.0	33.000000	105.000000	30.0	0.488	46	
746	1.000	147.0	94.0	41.000000	80.378667	49.3	0.358	27	
747	1.000	81.0	74.0	41.000000	57.000000	46.3	1.096	32	
748	3.000	187.0	70.0	22.000000	200.000000	36.4	0.408	36	
749	6.000	162.0	62.0	20.489333	80.378667	24.3	0.178	50	

750 rows × 9 columns

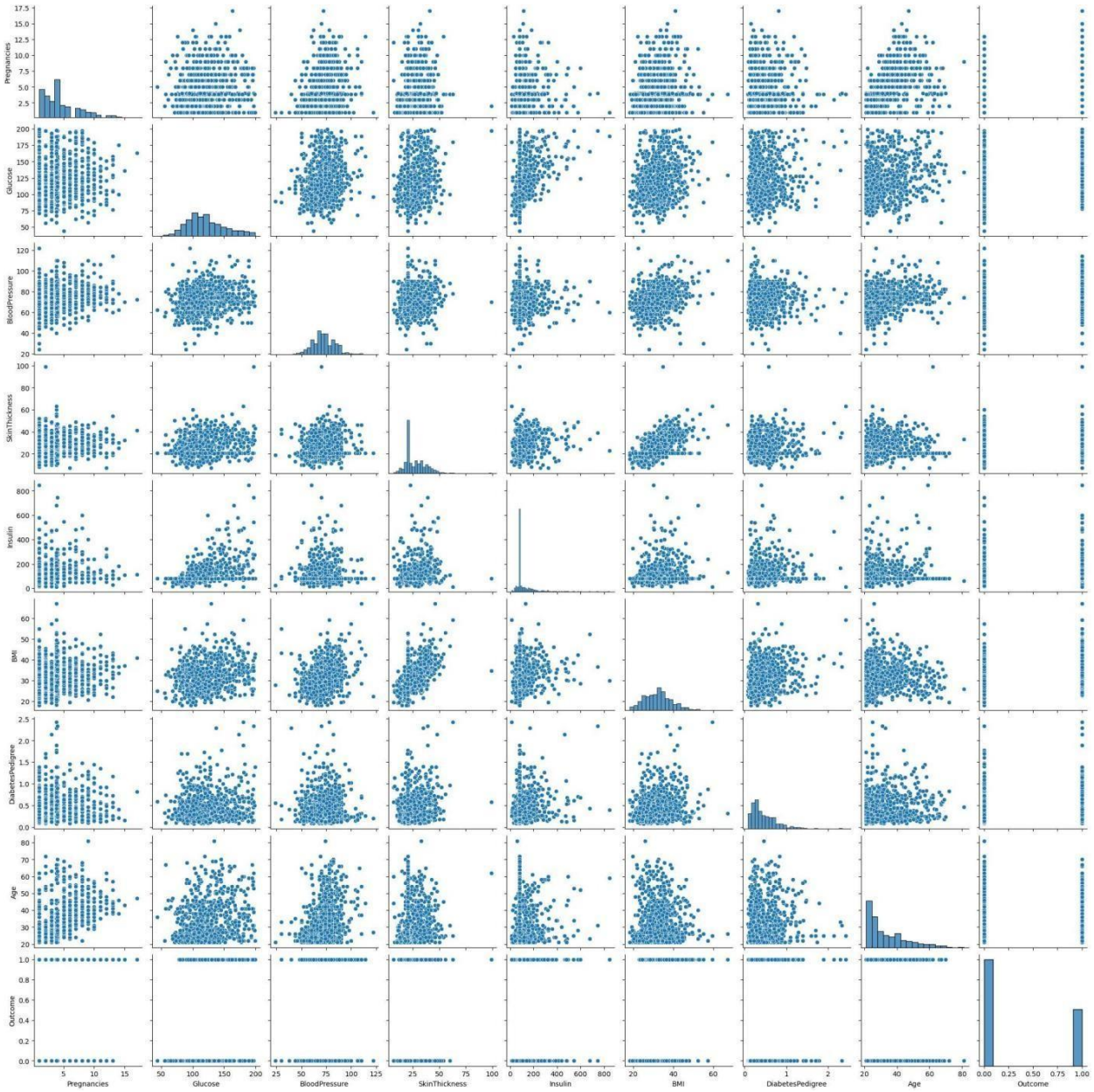
In [91]:
```
var = ['Pregnancies' ,'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', '


plt.figure(figsize=(14, 8))


for i, col in enumerate(var, 1):
    plt.subplot(2, 4, i)
    sns.boxplot(x='Outcome', y=col, data=data)

plt.tight_layout()
```



In [88]:
```
def remove (dataFrame) :
    for column_name in dataFrame.columns:
        Q1=data[column_name].quantile(0.25)
        Q3=data[column_name].quantile(0.75)
        IQR=Q3-Q1
        lower_lim= Q1- 9*IQR
        Upper_LIM= Q3+ 9*IQR
        dataFrame = dataFrame[(dataFrame[column_name]> lower_lim)|(dataFrame[column_na

    return dataFrame
```

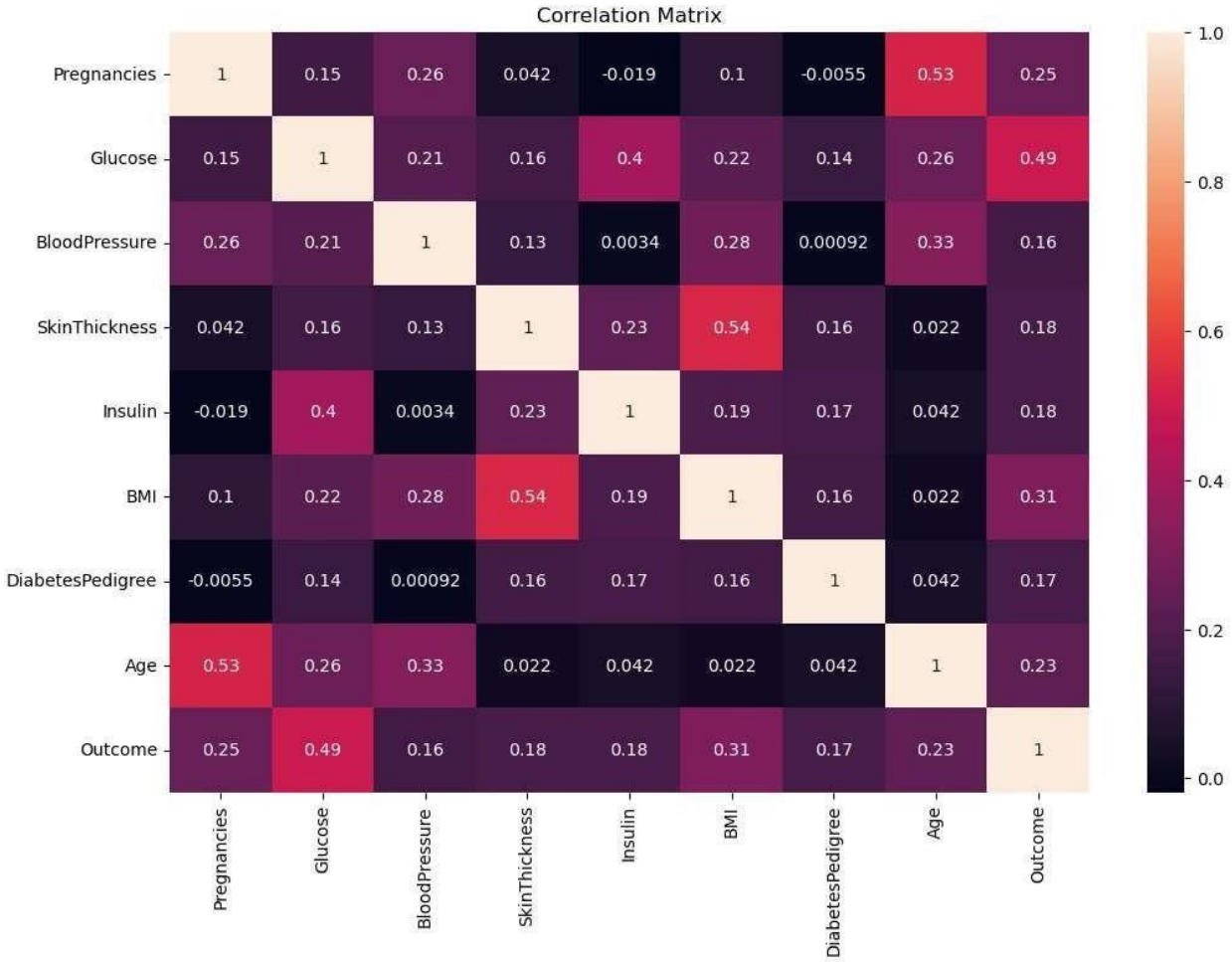
In [112…
```
sns.pairplot(data_no_outliers)
```

Out[112]:
```
<seaborn.axisgrid.PairGrid at 0x228ddddfe50>
```



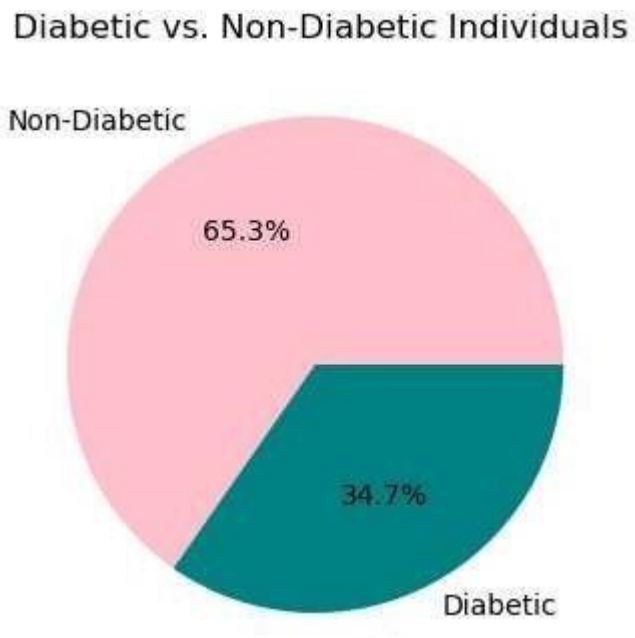
In [17]:
```
corel=data.corr()
plt.figure(figsize=(12, 8))
sns.heatmap(corel, annot=True)
plt.title('Correlation Matrix')
```

Out[17]:
```
Text(0.5, 1.0, 'Correlation Matrix')
```

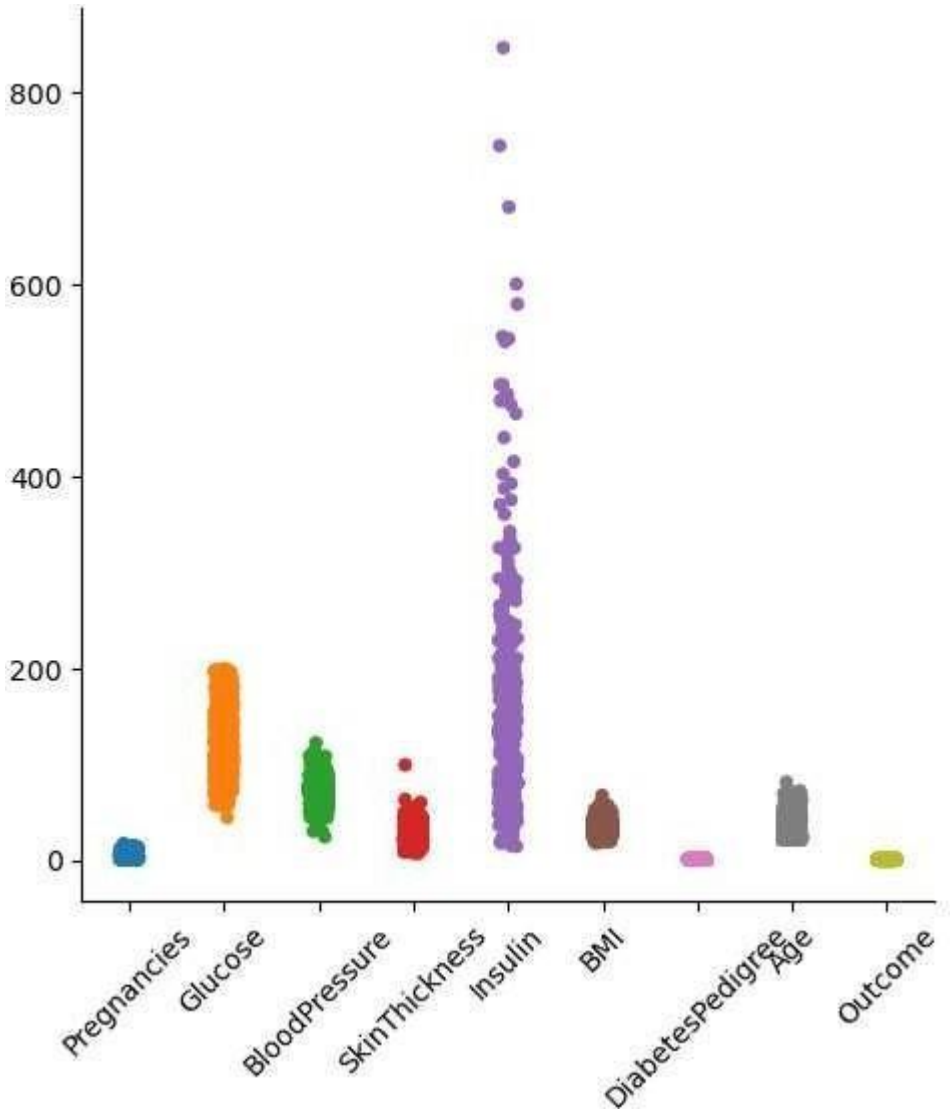


In [20]:
```
# diabetic vs non-diabetic patients in the dataset

count = data['Outcome'].value_counts()

label = ['Non-Diabetic', 'Diabetic']
c = ['pink', 'teal']

plt.figure(figsize=(4, 6))
plt.pie(count, labels=label, colors=c,autopct='%1.1f%%')
plt.title(' Diabetic vs. Non-Diabetic Individuals')
```

Out[20]:
```
Text(0.5, 1.0, ' Diabetic vs. Non-Diabetic Individuals')
```



In [18]:
```
import seaborn as sns
sns.catplot(data=data,height=5)
plt.xticks(rotation=55)
plt.show()
```



In [93]:
```
import pandas as pd
import statsmodels.api as sm

# Adding column 'SevenOrMorePregnancies'
data['SevenOrMorePregnancies'] = data['Pregnancies'] >= 7

# Typecasting
data['SevenOrMorePregnancies'] = data['SevenOrMorePregnancies'].astype(int)

# predictor and independent variables
X = data['SevenOrMorePregnancies']
y = data['Outcome']

X=sm.add_constant(X)

# logistic regression model
log = sm.Logit(y, X)
reg = log.fit()


print(reg.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.616518
         Iterations 5
                           Logit Regression Results
==============================================================================
Dep. Variable:                Outcome   No. Observations:                  750
Model:                          Logit   Df Residuals:                      748
Method:                           MLE   Df Model:                            1
Date:                Sun, 19 Nov 2023   Pseudo R-squ.:                 0.04469
Time:                        21:14:42   Log-Likelihood:                -462.39
converged:                       True   LL-Null:                       -484.02
Covariance Type:            nonrobust   LLR p-value:                 4.791e-11
==============================================================================
                                 coef    std err          z      P>|z|      [0.025
0.975]
------------------------------------------------------------------------------
const                      -0.9199      0.092    -10.052      0.000      -1.099      -
0.741
SevenOrMorePregnancies      1.1898      0.182      6.529      0.000       0.833
1.547
==============================================================================
```

In [94]:
```
#  probability of diabetes when  6 or fewer pregnancies
lt6 = reg.params['const'] + reg.params['SevenOrMorePregnancies'] * 0

# probability of diabetes when 7 or more pregnancies
mt7 = reg.params['const'] + reg.params['SevenOrMorePregnancies'] * 1

# logistic function to get probabilities
lt6 = 1 / (1 + 2.71828 ** (-lt6))
mt7 = 1 / (1 + 2.71828 ** (-mt7))

# Printing probabilities
print(f"Probability of diabetes when 6 or fewer pregnancies: {lt6:.3f}")
print(f"Probability of diabetes when 7 or more pregnancies: {mt7:.3f}")
```

```
Probability of developing diabetes given 6 or fewer pregnancies: 0.2850
Probability of developing diabetes given 7 or more pregnancies: 0.5671
```

In [95]:
```
#REGRESSION MODEL 1

import statsmodels.api as sm
import pandas as pd


X_train = data.drop(['Outcome', 'SevenOrMorePregnancies'], axis=1)
X_train = sm.add_constant(X_train)  # constant intercept
y_train = data['Outcome']

# regression model
logit_model = sm.Logit(y_train, X_train)
f = logit_model.fit()

# Test dataset
df2=pd.read_csv("C:/Users/abhin/Downloads/ToPredict.csv")
```

```
X_test = sm.add_constant(df2)  # intercept

# Prediction for outcome
prob = f.predict(X_test)

# for statistical summary
print(f.summary())

print("Predictions of outcome:\n", prob)
```

```
Optimization terminated successfully.
         Current function value: 0.464576
         Iterations 6
                           Logit Regression Results
==============================================================================
Dep. Variable:                Outcome   No. Observations:                  750
Model:                          Logit   Df Residuals:                      741
Method:                           MLE   Df Model:                            8
Date:                Sun, 19 Nov 2023   Pseudo R-squ.:                  0.2801
Time:                        21:15:15   Log-Likelihood:                -348.43
converged:                       True   LL-Null:                       -484.02
Covariance Type:            nonrobust   LLR p-value:                 5.539e-54
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          -8.9630      0.819    -10.947      0.000     -10.568      -7.358
Pregnancies     0.1503      0.037      4.106      0.000       0.079       0.222
Glucose         0.0378      0.004      9.648      0.000       0.030       0.045
BloodPressure  -0.0116      0.009     -1.330      0.184      -0.029       0.005
SkinThickness   0.0001      0.012      0.009      0.993      -0.023       0.023
Insulin        -0.0013      0.001     -1.237      0.216      -0.003       0.001
BMI             0.0934      0.018      5.292      0.000       0.059       0.128
DiabetesPedigree  0.8271    0.303      2.730      0.006       0.233       1.421
Age             0.0123      0.009      1.309      0.191      -0.006       0.031
==============================================================================
Predicted Probabilities:
 0    0.531493
1    0.261701
2    0.097214
3    0.675882
4    0.750536
dtype: float64
```

In [99]:

```
# REGRESSION MODEL 2

#choosing explanatory variables
X_train2 = data[['Glucose', 'Pregnancies', 'DiabetesPedigree']]
X_train2 = sm.add_constant(X_train2)  # intercept
y_train = data['Outcome']

# logistic regression model
logit_model_2 = sm.Logit(y_train, X_train2)
result_2 = logit_model_2.fit()


df2_2 =df2[['Glucose', 'Pregnancies', 'DiabetesPedigree']]


X_test_2 = sm.add_constant(df2_2)  # intercept
```

```
prob2 = result_2.predict(X_test_2)

# statistical summary
print(result_2.summary())

print("Predictions of outcome (Model 2):\n", prob2)
```

```
Optimization terminated successfully.
         Current function value: 0.490773
         Iterations 6
                           Logit Regression Results
==============================================================================
Dep. Variable:                Outcome   No. Observations:                  750
Model:                          Logit   Df Residuals:                      746
Method:                           MLE   Df Model:                            3
Date:                Sun, 19 Nov 2023   Pseudo R-squ.:                  0.2395
Time:                        21:16:07   Log-Likelihood:                -368.08
converged:                       True   LL-Null:                       -484.02
Covariance Type:            nonrobust   LLR p-value:                5.427e-50
==============================================================================
                     coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const             -6.6563      0.502    -13.258      0.000      -7.640      -5.672
Glucose            0.0383      0.003     11.051      0.000       0.032       0.045
Pregnancies        0.1673      0.031      5.482      0.000       0.107       0.227
DiabetesPedigree   0.9485      0.286      3.314      0.001       0.388       1.509
==============================================================================
Predicted Probabilities (Model 2):
 0    0.585163
1    0.167152
2    0.141181
3    0.619739
4    0.731394
dtype: float64
```

In [96]:

```
# REGRESSION MODEL 3

#choosing different explanatory variables
X_train3 = data[['Age', 'BMI', 'DiabetesPedigree','Pregnancies']]
X_train3 = sm.add_constant(X_train3)  # Add a constant term for the intercept
y_train = data['Outcome']

# Fit the logistic regression model
logit_model_3 = sm.Logit(y_train, X_train3)
result_3 = logit_model_3.fit()


df3 =df2[['Age', 'BMI', 'DiabetesPedigree','Pregnancies']]


X_test3 = sm.add_constant(df3)


# outcome prediction
prob3 = result_3.predict(X_test3)

# Display the result summary for more information
print(result_3.summary())
```

```
# Display predicted probabilities
print("Predictions of outcome (Model 3):\n", prob3)
```

```
Optimization terminated successfully.
         Current function value: 0.550405
         Iterations 6
                           Logit Regression Results
==============================================================================
Dep. Variable:                Outcome   No. Observations:                  750
Model:                          Logit   Df Residuals:                      745
Method:                           MLE   Df Model:                            4
Date:                Sun, 19 Nov 2023   Pseudo R-squ.:                  0.1471
Time:                        21:15:29   Log-Likelihood:                -412.80
converged:                       True   LL-Null:                       -484.02
Covariance Type:            nonrobust   LLR p-value:                 8.518e-30
==============================================================================
                     coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const             -5.9823      0.565    -10.586      0.000      -7.090      -4.875
Age                0.0301      0.008      3.695      0.000       0.014       0.046
BMI                0.1005      0.013      7.465      0.000       0.074       0.127
DiabetesPedigree   0.9138      0.258      3.542      0.000       0.408       1.420
Pregnancies        0.1207      0.032      3.716      0.000       0.057       0.184
==============================================================================
Predicted Probabilities (Model 3):
 0    0.349188
1    0.296900
2    0.113875
3    0.343743
4    0.499456
dtype: float64
```

In [97]:
```
result_3.aic
```

Out[97]:
```
835.6079380745894
```

In [100…
```
result_2.aic
```

Out[100]:
```
744.1592675571069
```

In [101…
```
result.aic
```

Out[101]:
```
714.8637723339812
```

In [1]:
```
pip install nbconvert
```

```
Requirement already satisfied: nbconvert in c:\users\abhin\anaconda3\lib\site-package
s (6.5.4)
Requirement already satisfied: lxml in c:\users\abhin\anaconda3\lib\site-packages (fr
om nbconvert) (4.9.2)
Requirement already satisfied: beautifulsoup4 in c:\users\abhin\anaconda3\lib\site-pa
ckages (from nbconvert) (4.12.2)
Requirement already satisfied: bleach in c:\users\abhin\anaconda3\lib\site-packages
(from nbconvert) (4.1.0)
Requirement already satisfied: defusedxml in c:\users\abhin\anaconda3\lib\site-packag
es (from nbconvert) (0.7.1)
Requirement already satisfied: entrypoints>=0.2.2 in c:\users\abhin\anaconda3\lib\sit
e-packages (from nbconvert) (0.4)
Requirement already satisfied: jinja2>=3.0 in c:\users\abhin\anaconda3\lib\site-packa
ges (from nbconvert) (3.1.2)
Requirement already satisfied: jupyter-core>=4.7 in c:\users\abhin\anaconda3\lib\site
-packages (from nbconvert) (5.3.0)
Requirement already satisfied: jupyterlab-pygments in c:\users\abhin\anaconda3\lib\si
te-packages (from nbconvert) (0.1.2)
Requirement already satisfied: MarkupSafe>=2.0 in c:\users\abhin\anaconda3\lib\site-p
ackages (from nbconvert) (2.1.1)
Requirement already satisfied: mistune<2,>=0.8.1 in c:\users\abhin\anaconda3\lib\site
-packages (from nbconvert) (0.8.4)
Requirement already satisfied: nbclient>=0.5.0 in c:\users\abhin\anaconda3\lib\site-p
ackages (from nbconvert) (0.5.13)
Requirement already satisfied: nbformat>=5.1 in c:\users\abhin\anaconda3\lib\site-pac
kages (from nbconvert) (5.7.0)
Requirement already satisfied: packaging in c:\users\abhin\anaconda3\lib\site-package
s (from nbconvert) (23.0)
Requirement already satisfied: pandocfilters>=1.4.1 in c:\users\abhin\anaconda3\lib\s
ite-packages (from nbconvert) (1.5.0)
Requirement already satisfied: pygments>=2.4.1 in c:\users\abhin\anaconda3\lib\site-p
ackages (from nbconvert) (2.15.1)
Requirement already satisfied: tinycss2 in c:\users\abhin\anaconda3\lib\site-packages
(from nbconvert) (1.2.1)
Requirement already satisfied: traitlets>=5.0 in c:\users\abhin\anaconda3\lib\site-pa
ckages (from nbconvert) (5.7.1)
Requirement already satisfied: platformdirs>=2.5 in c:\users\abhin\anaconda3\lib\site
-packages (from jupyter-core>=4.7->nbconvert) (2.5.2)
Requirement already satisfied: pywin32>=300 in c:\users\abhin\anaconda3\lib\site-pack
ages (from jupyter-core>=4.7->nbconvert) (305.1)
Requirement already satisfied: jupyter-client>=6.1.5 in c:\users\abhin\anaconda3\lib
\site-packages (from nbclient>=0.5.0->nbconvert) (7.4.9)
Requirement already satisfied: nest-asyncio in c:\users\abhin\anaconda3\lib\site-pack
ages (from nbclient>=0.5.0->nbconvert) (1.5.6)
Requirement already satisfied: fastjsonschema in c:\users\abhin\anaconda3\lib\site-pa
ckages (from nbformat>=5.1->nbconvert) (2.16.2)
Requirement already satisfied: jsonschema>=2.6 in c:\users\abhin\anaconda3\lib\site-p
ackages (from nbformat>=5.1->nbconvert) (4.17.3)
Requirement already satisfied: soupsieve>1.2 in c:\users\abhin\anaconda3\lib\site-pac
kages (from beautifulsoup4->nbconvert) (2.4)
Requirement already satisfied: six>=1.9.0 in c:\users\abhin\anaconda3\lib\site-packag
es (from bleach->nbconvert) (1.16.0)
Requirement already satisfied: webencodings in c:\users\abhin\anaconda3\lib\site-pack
ages (from bleach->nbconvert) (0.5.1)
Requirement already satisfied: attrs>=17.4.0 in c:\users\abhin\anaconda3\lib\site-pac
kages (from jsonschema>=2.6->nbformat>=5.1->nbconvert) (22.1.0)
Requirement already satisfied: pyrsistent!=0.17.0,!=0.17.1,!=0.17.2,>=0.14.0 in c:\us
ers\abhin\anaconda3\lib\site-packages (from jsonschema>=2.6->nbformat>=5.1->nbconver
t) (0.18.0)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\abhin\anaconda3\lib
```

```
\site-packages (from jupyter-client>=6.1.5->nbclient>=0.5.0->nbconvert) (2.8.2)
Requirement already satisfied: pyzmq>=23.0 in c:\users\abhin\anaconda3\lib\site-packa
ges (from jupyter-client>=6.1.5->nbclient>=0.5.0->nbconvert) (23.2.0)
Requirement already satisfied: tornado>=6.2 in c:\users\abhin\anaconda3\lib\site-pack
ages (from jupyter-client>=6.1.5->nbclient>=0.5.0->nbconvert) (6.3.2)
Note: you may need to restart the kernel to use updated packages.
```

In [2]:
```
pip install `nbconvert[webpdf]`
```

```
Note: you may need to restart the kernel to use updated packages.
ERROR: Invalid requirement: ``nbconvert[webpdf]`'
```

In [ ]:
```
```

REFERENCES

Knopp, J. L., Holder-Pearson, L., & Chase, J. G. (2018, 2023/01/16). Insulin units and conversion
factors: A story of truth, boots, and faster half-truths. *Journal of Diabetes Science and Tech-
nology*, *13*(3), 597–600. doi: 10.1177/1932296818805074

Melmed, S., Polonsky, K. S., Larsen, P. R., & Kronenberg, H. M. (Eds.). (2016). *Williams textbook of
endocrinology* (Thirteenth Edition ed.). Philadelphia: Elsevier. doi: 10.1016/C2013-0-15980- 6

Smith, J. W., Everhart, J., Dickson, W., Knowler, W., & Johannes, R. (1988). Using the ADAP learning
algorithm to forecast the onset of diabetes mellitus. In *Proceedings of the annual symposium
on computer application in medical care* (pp. 261–265).

Python Lambda, https://www.w3schools.com/python

Anderson, Sweeney, Williams. (2011). Statistics for Business and Economics: Logistic Regression (pp. 665-679).

GeeksforGeeks. https://www.geeksforgeeks.org/logistic-regression-using-statsmodels/