Import libraries

In [ ]:

```
import numpy as np
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report
```

In [ ]:

```
from google.colab import files
uploaded = files.upload()
```

Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving diabetes.csv to diabetes (4).csv
```

In [ ]:

```
diabetes = pd.read_csv('diabetes.csv')
print(diabetes)
     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
..           ...      ...            ...            ...      ...   ...
763           10      101             76             48      180  32.9
764            2      122             70             27        0  36.8
765            5      121             72             23      112  26.2
766            1      126             60              0        0  30.1
767            1       93             70             31        0  30.4

     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
..                        ...  ...      ...
763                     0.171   63        0
764                     0.340   27        0
765                     0.245   30        0
766                     0.349   47        1
767                     0.315   23        0

[768 rows x 9 columns]
```

key variables of the data set

In [ ]:

```
print("Keys of Diabets_dataset: \n{}".format(diabetes.keys()))
Keys of Diabets_dataset:
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

Cleaning Empty Cells

In [ ]:

```
new_diabetes = diabetes.dropna()
```

Plot each Variables

In [ ]:

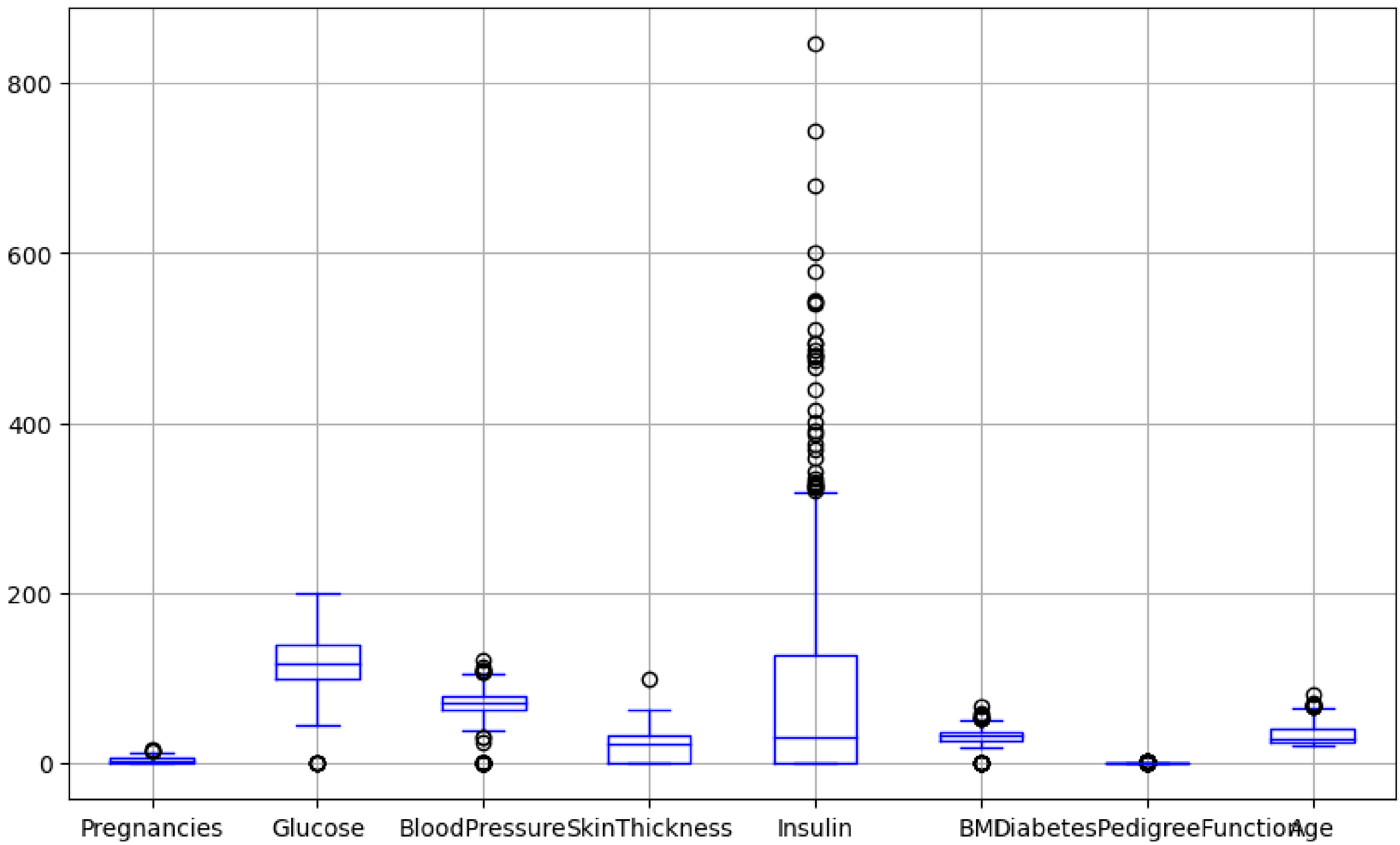
```
import seaborn as sns
import matplotlib.pyplot as plt
```

In [ ]:

```
new_diabetes.boxplot(column=['Pregnancies', 'Glucose', 'BloodPressure',
'SkinThickness', 'Insulin','BMI', 'DiabetesPedigreeFunction',
'Age'],color='blue', figsize=(10, 6))
```

Out[ ]:

```
<Axes: >
```



Remove Outliers

In [ ]:

```
from scipy import stats
new = ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness',
'Insulin','BMI', 'DiabetesPedigreeFunction', 'Age']
z_scores = np.abs(stats.zscore(new_diabetes[new]))
threshold = 3
outliers = (z_scores > threshold).any(axis=1)

data_cleaned = new_diabetes[~outliers]
```

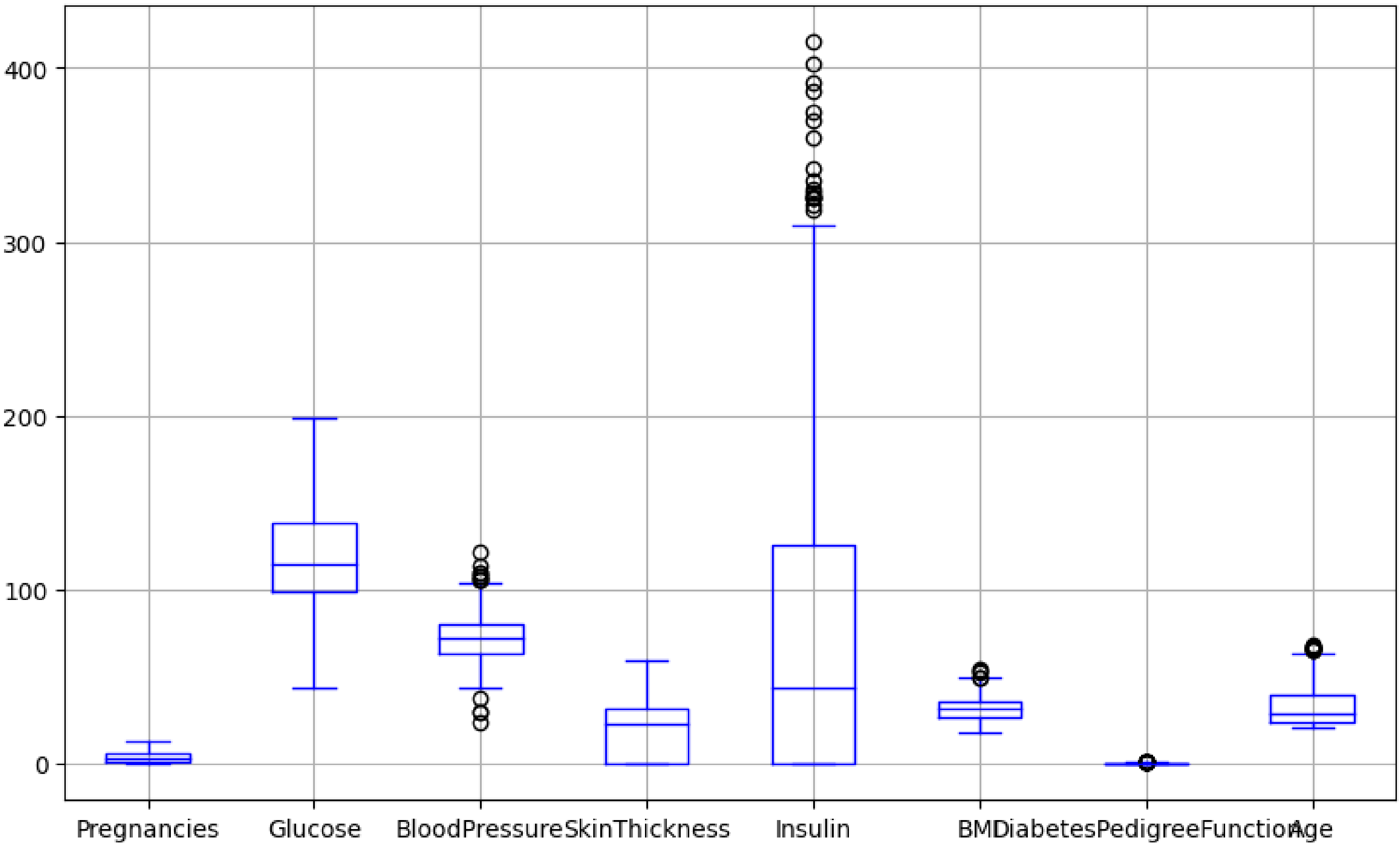
Plot Variables After removing Outliers

In [ ]:

```
data_cleaned.boxplot(column=['Pregnancies', 'Glucose', 'BloodPressure',
'SkinThickness', 'Insulin','BMI', 'DiabetesPedigreeFunction',
'Age'],color='blue', figsize=(10, 6))
```

Out[ ]:

```
<Axes: >
```



When compearing above two boxplot charts, We can conclude that several variable's outliers have been decreased.

In [ ]:

```
print(data_cleaned.info())
<class 'pandas.core.frame.DataFrame'>
Int64Index: 688 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               688 non-null    int64
 1   Glucose                   688 non-null    int64
```

```
 2   BloodPressure             688 non-null    int64
 3   SkinThickness             688 non-null    int64
 4   Insulin                   688 non-null    int64
 5   BMI                       688 non-null    float64
 6   DiabetesPedigreeFunction  688 non-null    float64
 7   Age                       688 non-null    int64
 8   Outcome                   688 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 53.8 KB
None
```

I have removed the outliers above, and this summary shows there are no any null values after removing outliers. and also, before removing outliers data set has 768 raws and after removing outliers that has reduced to 688 raws.

Check Significance of dependent variables

In [ ]:

```
# Define features (X) and binary target variable (y)

X = data_cleaned[new]  # Replace with your independent variables
y = data_cleaned['Outcome']  # Replace with your binary dependent variable

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

#Build and train the Logistic Regression model
model = LogisticRegression()
model.fit(X_train, y_train)
print(model)

# Make predictions
y_pred = model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
confusion = confusion_matrix(y_test, y_pred)
classification = classification_report(y_test, y_pred)

print (model)
print(f"Accuracy: {accuracy}")
print("Confusion Matrix:\n", confusion)
print("Classification Report:\n", classification)
LogisticRegression()
```

```
LogisticRegression()
Accuracy: 0.7608695652173914
Confusion Matrix:
 [[80  5]
 [28 25]]
Classification Report:
              precision    recall  f1-score   support

           0       0.74      0.94      0.83        85
           1       0.83      0.47      0.60        53

    accuracy                           0.76       138
   macro avg       0.79      0.71      0.72       138
weighted avg       0.78      0.76      0.74       138

/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

In [128]:

```
import statsmodels.api as sm

#Fit a logistic regression model using statsmodels
X = sm.add_constant(X)

model = sm.Logit(y, X)
result = model.fit()

p_values = result.pvalues
print(p_values)

#Get the model summary, including p-values
summary = result.summary()
print(summary)
```

```
Optimization terminated successfully.
         Current function value: 0.456687
         Iterations 6
const                      3.105277e-34
```

```
Pregnancies                 9.011362e-06
Glucose                     1.139108e-22
BMI                         1.111667e-08
DiabetesPedigreeFunction    1.885311e-05
dtype: float64
                           Logit Regression Results
==============================================================================
=
Dep. Variable:                Outcome   No. Observations:
688
Model:                          Logit   Df Residuals:
683
Method:                           MLE   Df Model:
4
Date:                Fri, 13 Oct 2023   Pseudo R-squ.:
0.2798
Time:                        16:56:57   Log-Likelihood:                -
314.20
converged:                       True   LL-Null:                       -
436.29
Covariance Type:            nonrobust   LLR p-value:                 1.172e-
51
==============================================================================
===============
                               coef    std err          z      P>|z|
[0.025      0.975]
------------------------------------------------------------------------------
---------------
const                       -9.5412      0.782    -12.200      0.000     -
11.074      -8.008
Pregnancies                  0.1329      0.030      4.440      0.000
0.074       0.192
Glucose                      0.0368      0.004      9.799      0.000
0.029       0.044
BMI                          0.0919      0.016      5.713      0.000
0.060       0.123
DiabetesPedigreeFunction     1.4874      0.348      4.278      0.000
0.806       2.169
==============================================================================
===============
```

Get the variables below the P value

In [129]:

```
alpha = 0.05
```

```
# Identify non-significant variables
non_sig = p_values[p_values > alpha].index


# Remove non-significant variables from the feature matrix
X = X.drop(non_sig, axis=1)
Y = data_cleaned.Outcome

# Fit the model again with the selected variables
model = sm.Logit(Y, X)
result = model.fit()

summary = result.summary()
print(summary)
Optimization terminated successfully.
         Current function value: 0.456687
         Iterations 6
                           Logit Regression Results
=============================================================================
=
Dep. Variable:                Outcome   No. Observations:
688
Model:                          Logit   Df Residuals:
683
Method:                           MLE   Df Model:
4
Date:                Fri, 13 Oct 2023   Pseudo R-squ.:
0.2798
Time:                        16:57:02   Log-Likelihood:                -
314.20
converged:                       True   LL-Null:                       -
436.29
Covariance Type:            nonrobust   LLR p-value:                 1.172e-
51
=============================================================================
===============
                              coef    std err          z      P>|z|
[0.025      0.975]
-----------------------------------------------------------------------------
---------------
const                      -9.5412      0.782    -12.200      0.000      -
11.074      -8.008
Pregnancies                 0.1329      0.030      4.440      0.000
0.074       0.192
```

```
Glucose                        0.0368      0.004      9.799      0.000
0.029       0.044
BMI                            0.0919      0.016      5.713      0.000
0.060       0.123
DiabetesPedigreeFunction       1.4874      0.348      4.278      0.000
0.806       2.169
==============================================================================
===============
```

Considering the above summary, we can identify P values for each variables. According to the P values for each variables, we can get a conclution about what are the most significance factors for affecting to the diabetes. To do that, we consider the P values of varibales that are lower than the 0.05 (P < 0.05) That variables are,

1. Pregnancies
2. Glucose
3. BMI
4. DiabetesPedigreeFunction

Logistic regression as a binary classifier

In [130]:

```
significance_var = ['Pregnancies',
'Glucose','BMI','DiabetesPedigreeFunction']
# Define features (X) and binary target variable (y)
X = data_cleaned[significance_var]  # Replace with your independent variables
Y = data_cleaned['Outcome']  # Replace with your binary dependent variable

# Split the data into training and testing sets
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2,
random_state=42)

# Build and train the Logistic Regression model
model = LogisticRegression()
model.fit(X_train, Y_train)
print(model)

# Make predictions
Y_pred = model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(Y_test, Y_pred)
confusion = confusion_matrix(Y_test, Y_pred)
classification = classification_report(Y_test, Y_pred)
```

```
print (model)
print(f"Accuracy: {accuracy}")
print("Confusion Matrix:\n", confusion)
print("Classification Report:\n", classification)
LogisticRegression()
LogisticRegression()
Accuracy: 0.782608695652174
Confusion Matrix:
 [[82  3]
 [27 26]]
Classification Report:
               precision    recall  f1-score   support

           0       0.75      0.96      0.85        85
           1       0.90      0.49      0.63        53

    accuracy                           0.78       138
   macro avg       0.82      0.73      0.74       138
weighted avg       0.81      0.78      0.76       138
```

Evaluating the dataset with OVR Model

In [131]:

```
#Evaluating the dataset with OVR Model
#X = data_cleaned[significance_var]  # Replace with your independent
variables
#y = data_cleaned['Outcome']  # Replace with your binary dependent variable
# Create one-vs-rest logistic regression object
logreg = LogisticRegression(multi_class="ovr")

# Train model
model = logreg.fit(X_train, Y_train)

print("Test set predictions: {}".format(model.predict(X_test)))
# We calculate the predictions for y_test.

print("Test set accuracy: {:.2f}".format(logreg.score(X_test, Y_test)))
# To evaluate how well our model generalizes, we call the score method with
the test data together
# with the test labels.
Test set predictions: [0 0 0 0 1 1 0 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 1 0 0 0 1
0 0 1 0 0 1 0 0 0 0
 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 1 0 0 1 0 0
 1 0 0 0 0 1 0 0 1 0 0 1 0 0 1 0 0 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1
```

```
 0 0 0 1 0 1 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 1 0]
Test set accuracy: 0.78
```

Evaluating the dataset with multinomial Model

In [132]:

```
# Create multinomial logistic regression object


logreg = LogisticRegression(random_state=0, multi_class="multinomial")

# Train model
model = logreg.fit(X_train, Y_train)

print("Test set predictions: {}".format(model.predict(X_test)))
# We calculate the predictions for y_test.

print("Test set accuracy: {:.2f}".format(logreg.score(X_test, Y_test)))
# To evaluate how well our model generalizes, we call the score method with
the test data together
# with the test labels.
Test set predictions: [0 0 0 0 1 1 0 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 1 0 0 0 1
0 0 1 0 0 1 0 0 0 0
 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 1 0 0 1 0 0
 1 0 0 0 0 1 0 0 1 0 0 1 0 0 1 0 0 0 0 1 0 1 1 0 0 0 0 0 0 0 0 1 0 0 0 0 1
 0 0 0 1 0 1 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 1 0]
Test set accuracy: 0.78
```

When considering the accuracy results for the above several logistic regression model, that all models results are the same. (0.78) therefore, according to the our diabetes data set, we can get any suitable best fitted logistic regression model for the to predict whether or not a patient has diabetes.