HTU Upskilling Program: Data Science Track

Capstone Project: Prediction and diagnosis of future diabetes risk.
Student Name: Abed-alnasser Othman Jaber Al-hmouz.
Date: Feb 2nd 2023

- **Introduction**

  The aim of the project is to predict whether a person will have diabetes or not. I have to analyze the data collected by existing sources, such as electronic health records or other clinical data, the collected data was then used to develop a model about whether a person will have diabetes or not.

  The data will be preprocessed and the classification model K-Nearest Neighbors (KNN) will be applied to the dataset and their accuracy will be evaluated to select the best model.

  The results of this project can have practical applications; data pre-processing is an important step in order to build a better and more reliable model for the process of prediction. In the future, similar approaches can be applied to other disease datasets like cardiovascular disease or oncology-based diseases for the purpose of prediction. Moreover, the same techniques can be used for pathological and rare disease prediction in order to enhance overall healthcare.

- **Problem**

  We live in an era where data generation is exponential with time but if the generated data is not put to work or not converted to knowledge data, its generation is of no use. Similarly, in Healthcare also, data availability is high, and so is the need to extract the information from it for better prognosis, diagnosis, treatment, drug development, and overall healthcare.

  In this research, we have tried to focus more on the diagnosis of Diabetes disease, which is one of the fastest-growing chronic diseases all over the world as declared by the World Health Organization in the year 2014. We have also tried to show the different techniques like Neural Network Classifier, Support Vector Machine, Gradient Boosting Classifier, and K Nearest Neighbors Classifier, which can be used for the diagnosis of diabetes disease with attained accuracy as 80.5% for Neural Network Classifier, 81.2% for Gradient Boosting Classifier, 79.9% for Support Vector Machine, and 81.8% for K Nearest Neighbors classifier.

- **Data set**

diabetes data set is available in the UCI machine learning repository and this set has 768 instances and 8 attributes [1].

The objective of this data set is to diagnose diabetes in people. Based on the various attributes provided in this data set, this paper shows whether a person is diabetes-positive or not using KNN classification, Neural Network Classifier, Gradient Boosting Classifier, and Support Vector Machine. In this database, all patients are females and are of age at least 21 years. The total numbers of instances in the database are 768 out of which 268 are diabetic and 500 are non-diabetic and are described as 1 and 0 respectively in the class attribute.

Till 2011 the diabetes data set present in the UCI machine learning repository was considered to have no missing values but later it is found that in place of missing values, there are zeros, and having values as zeros at these places is biologically impossible such as in age or blood pressure attributes. Also, zero body mass index, zero plasma glucose, and 2-h serum insulin contain almost 50% impossible values. Attributes in this database are either integer or real [2].

The characteristics of diabetes data:

Attribute Number	Attribute Name	Attribute description	Attribute type and measurement
1	Pregnancy	Number of times the female is pregnant	Numeric
2	Plasma glucose	measured using a 2-hour oral glucose tolerance test	Numeric
3	Blood Pressure	diastolic blood pressure	Numeric (mm Hg)
4	Skin Thickness	Triceps skinfold thickness	Numeric (mm)
5	Insulin	serum insulin	Numeric (μ U/ml)
6	BMI	Body mass index	[weight in kg/(height in m)2]
7	Diabetes Pedigree Function	scores likelihood of diabetes based on family history	Numeric
8	Age	Age of patients	Numeric (years)

The description of the data in the data frame; describe() method generates descriptive statistics that summarize the central tendency, dispersion, and shape of a dataset's distribution, excluding NaN values. This method tells us a lot of things about a dataset. One important thing is that the describe() method deals only with numeric values. It doesn't work with any categorical values. So if there are any categorical values in a column the describe() method will ignore it and display a summary for the other columns.

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

Now, let's understand the statistics that are generated by the describe() method:

(count) tells us the number of NoN-empty rows in a feature.

(mean) tells us the mean value of that feature.

(std) tells us the Standard Deviation Value of that feature.

(min) tells us the minimum value of that feature.

(25%, 50%, and 75%) are the percentile/quartile of each feature; this quartile information helps us to detect Outliers.

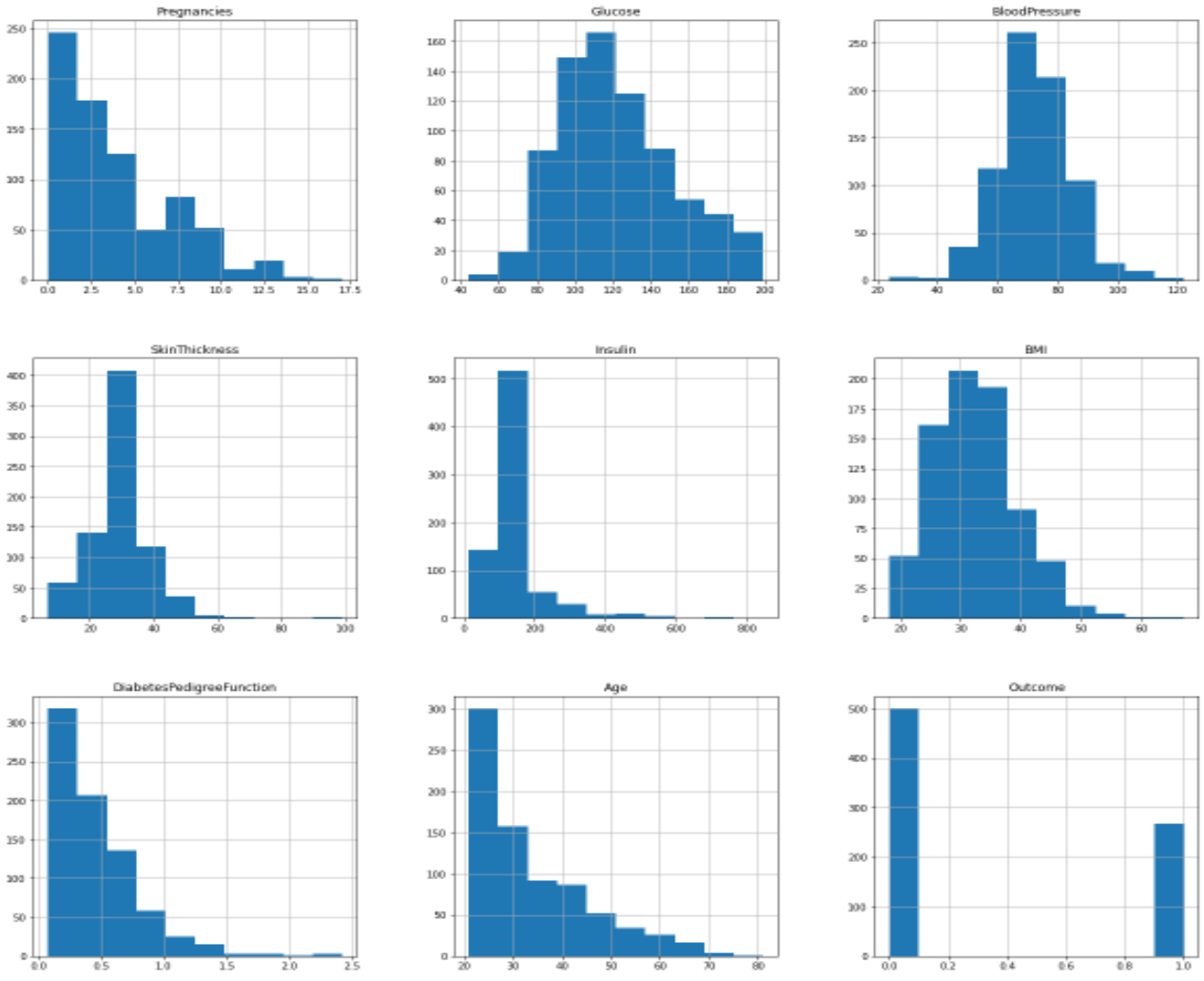
(max) tells us the maximum value of that feature.

- **Source:**

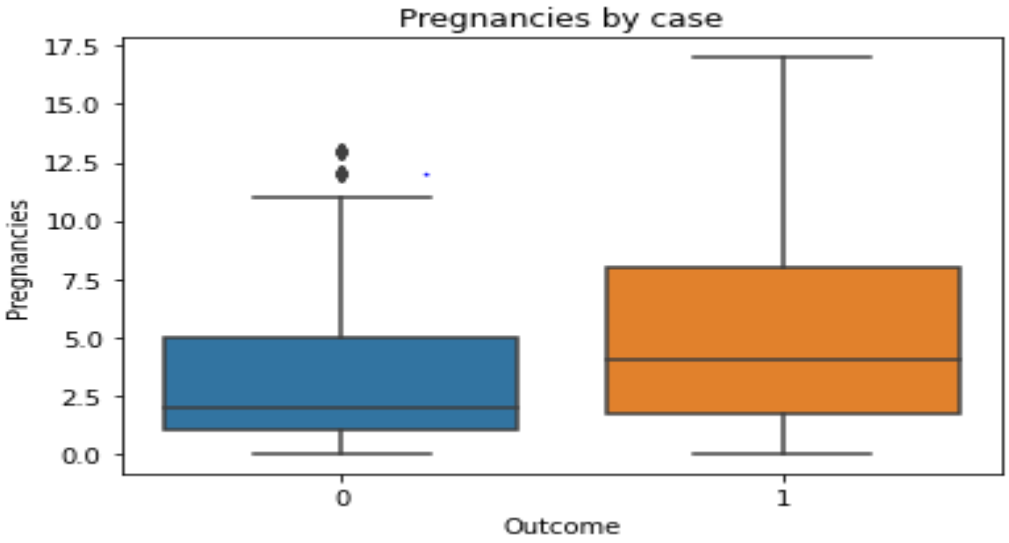
  The diabetes data set is available in the unique client identifier machine learning repository and this set has 768 instances and 8 attributes. The UCI page mentions the following 2 publications as the original source of the data set:

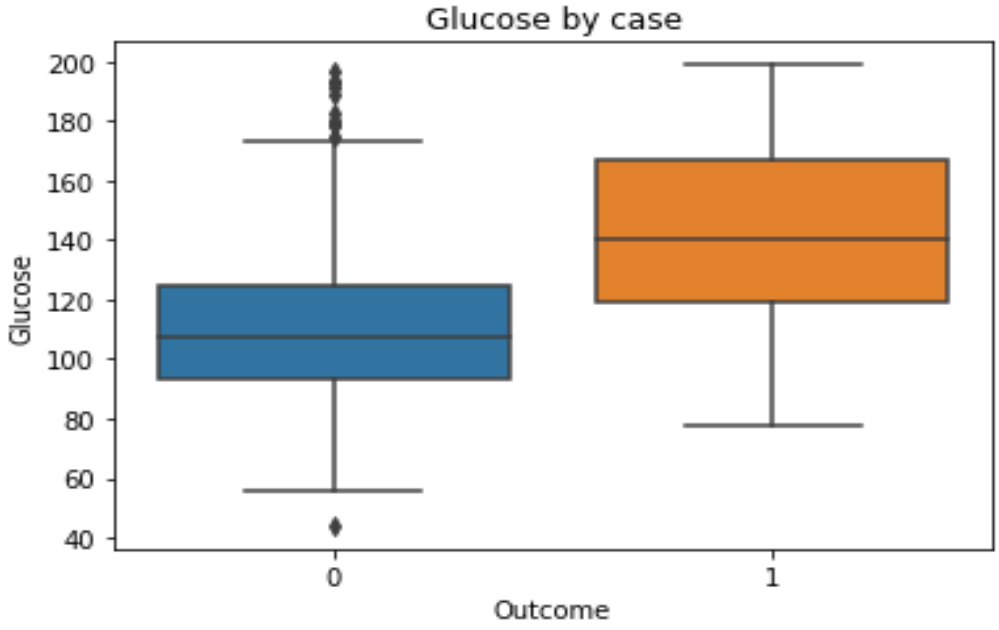
  1. https://archive.ics.uci.edu/ml/datasets/pima%2bindians%2bdiabetes
  2. http://archive.ics.uci.edu/ml/index.php
  3. https://link.springer.com/article/10.1007/s42452-019-1117-9

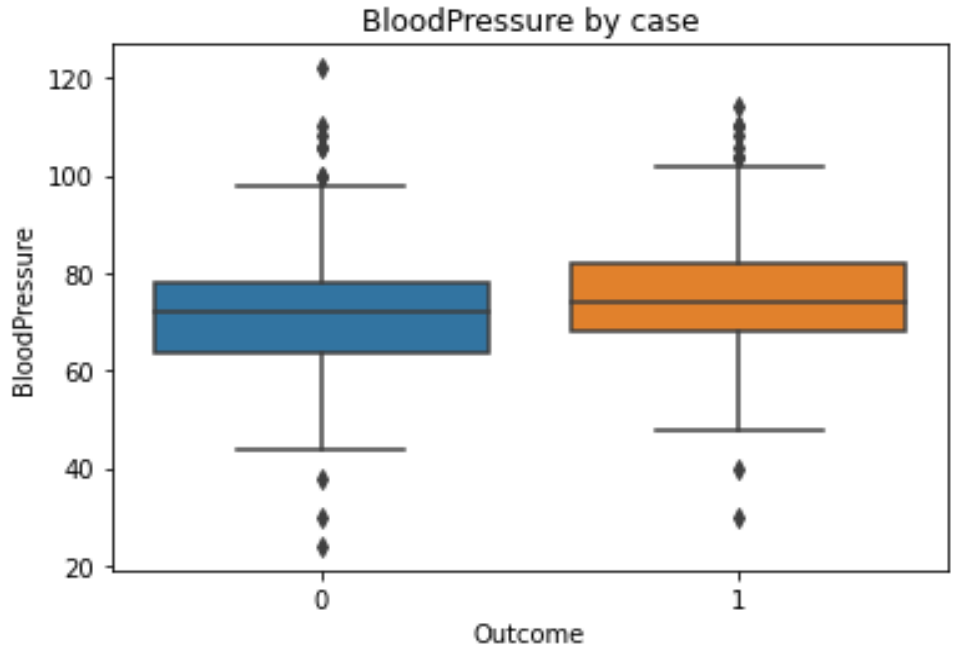
- **Data visualization:**
  - The data distribution needs to be understood

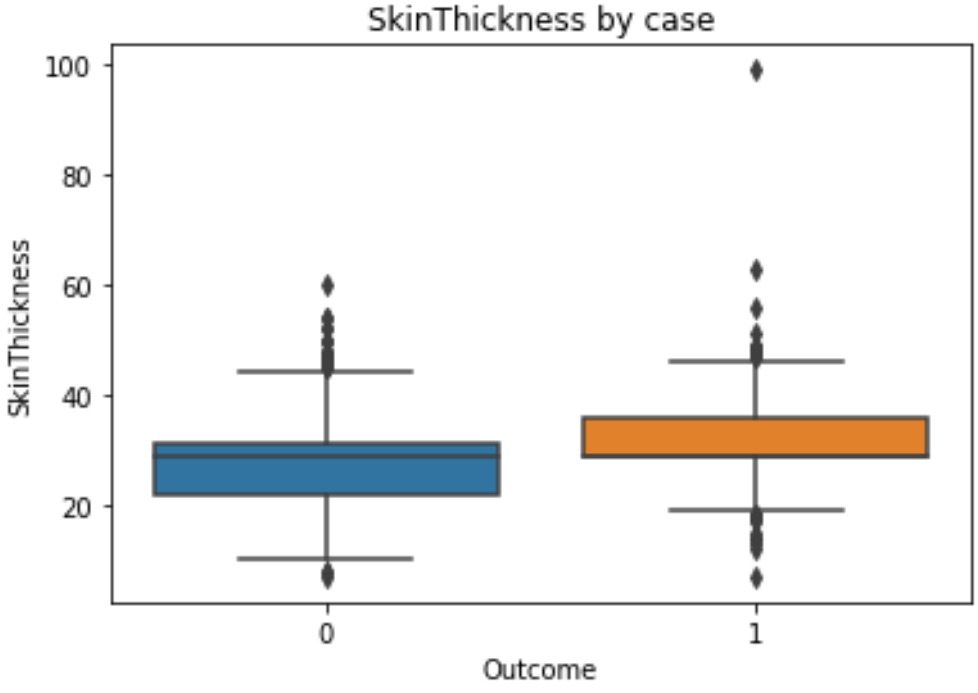


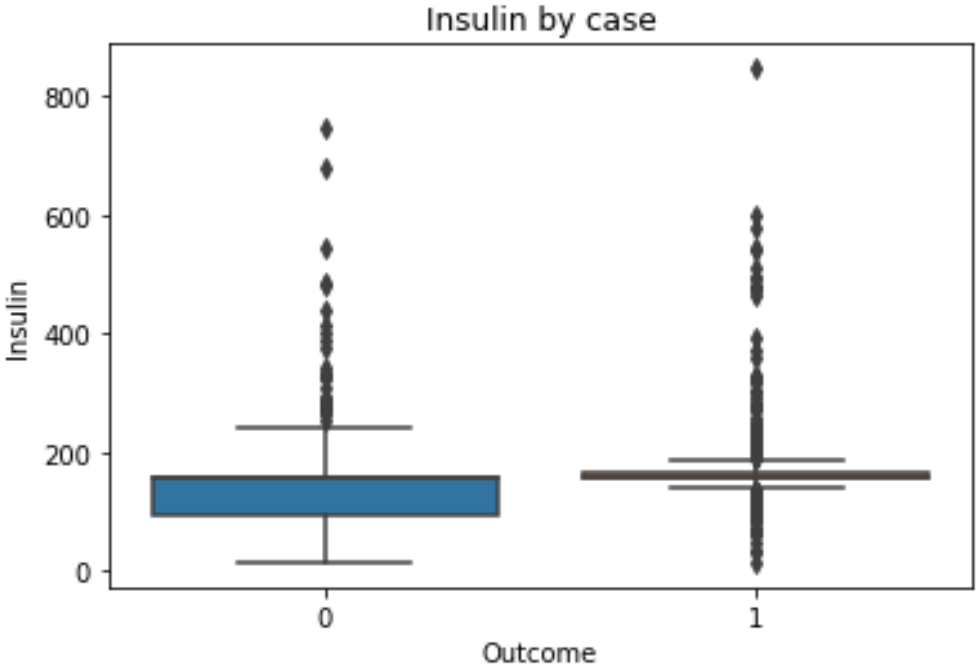
- **Create a box plot for each factor**

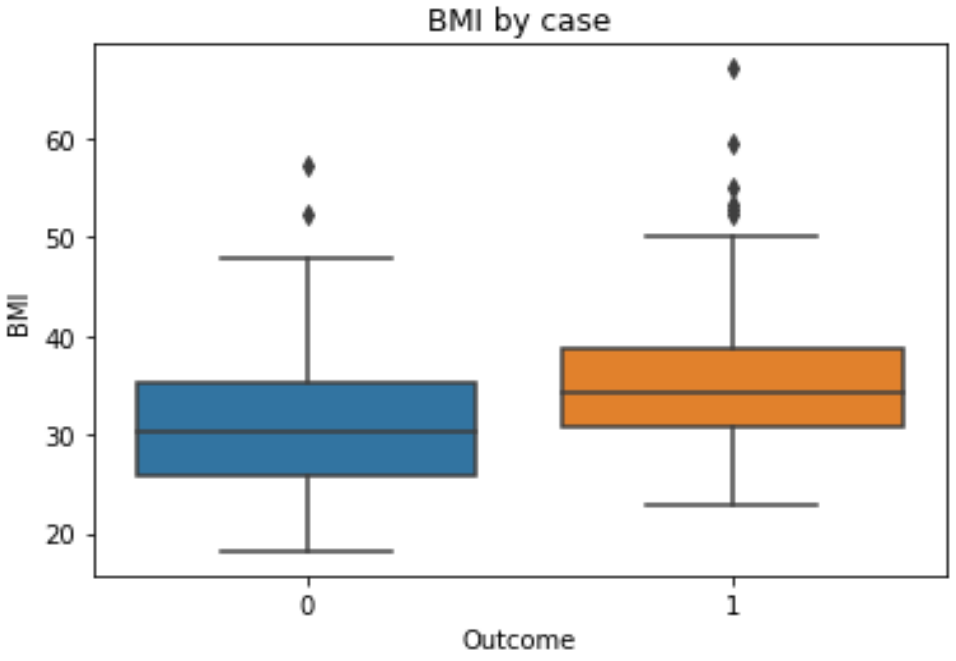


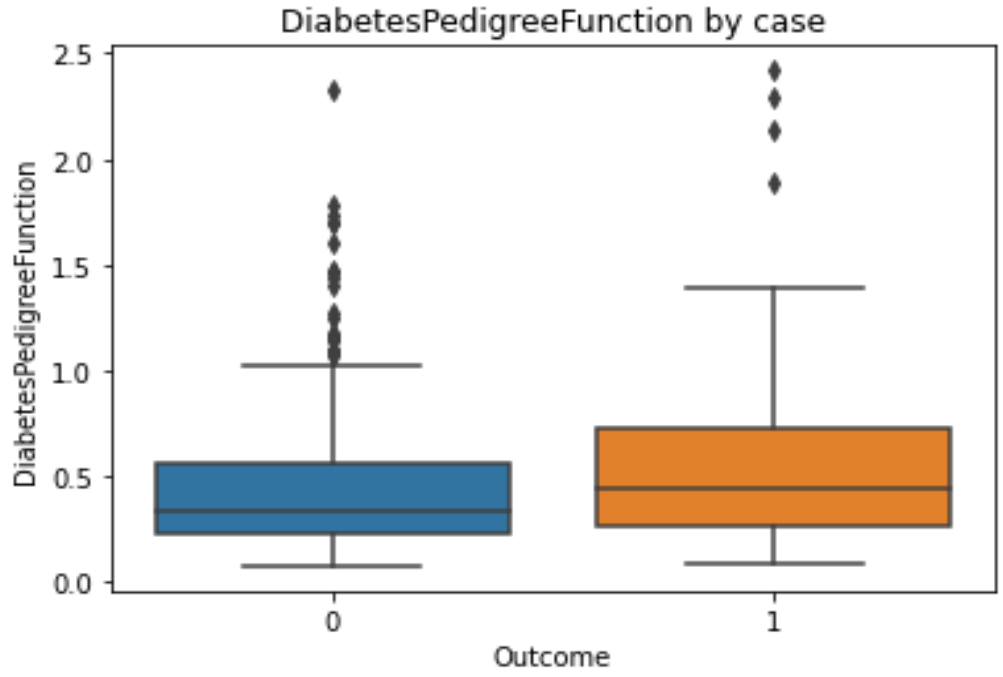


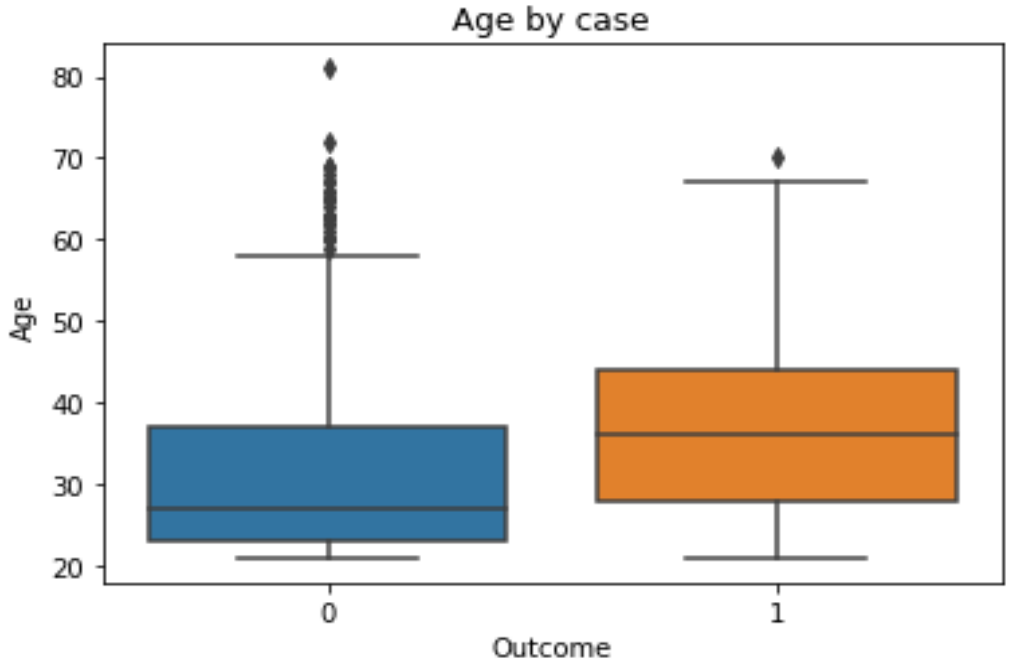

BloodPressure by case
BloodPressure
120
100
80
60
40
20
0
1
Outcome


SkinThickness by case
SkinThickness
100
80
60
40
20
0
1
Outcome

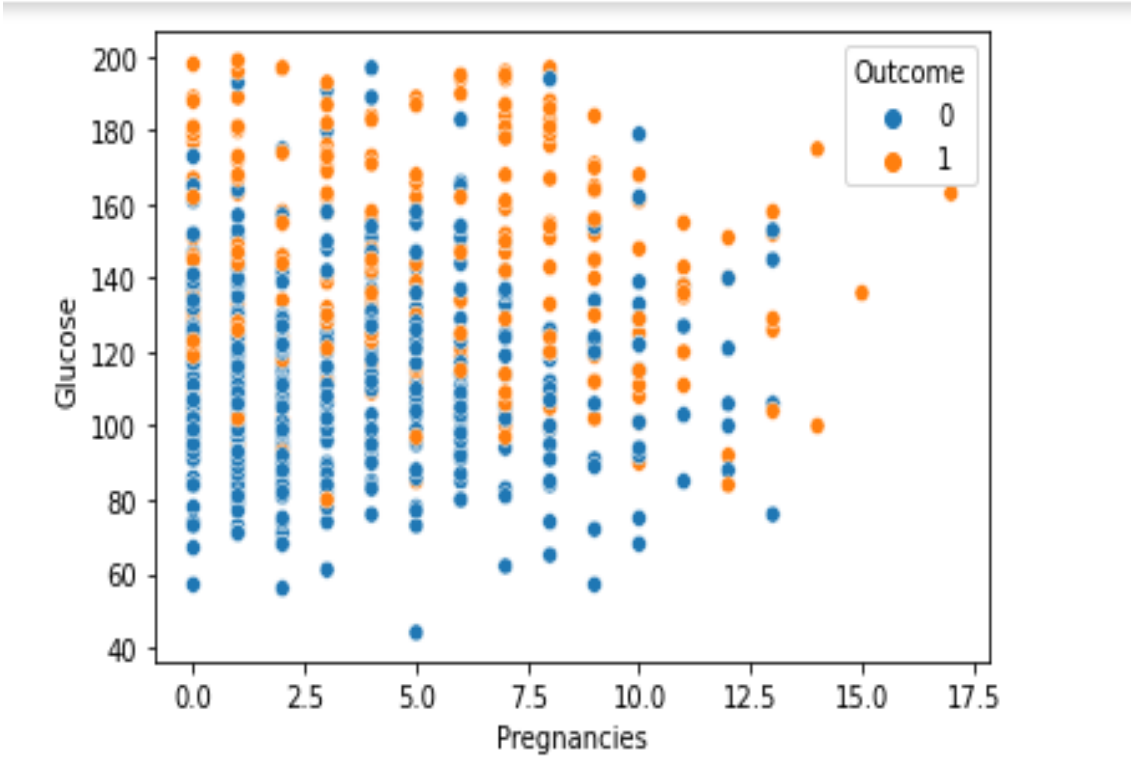

Insulin by case
800
600
400
200
0
Insulin
0
1
Outcome

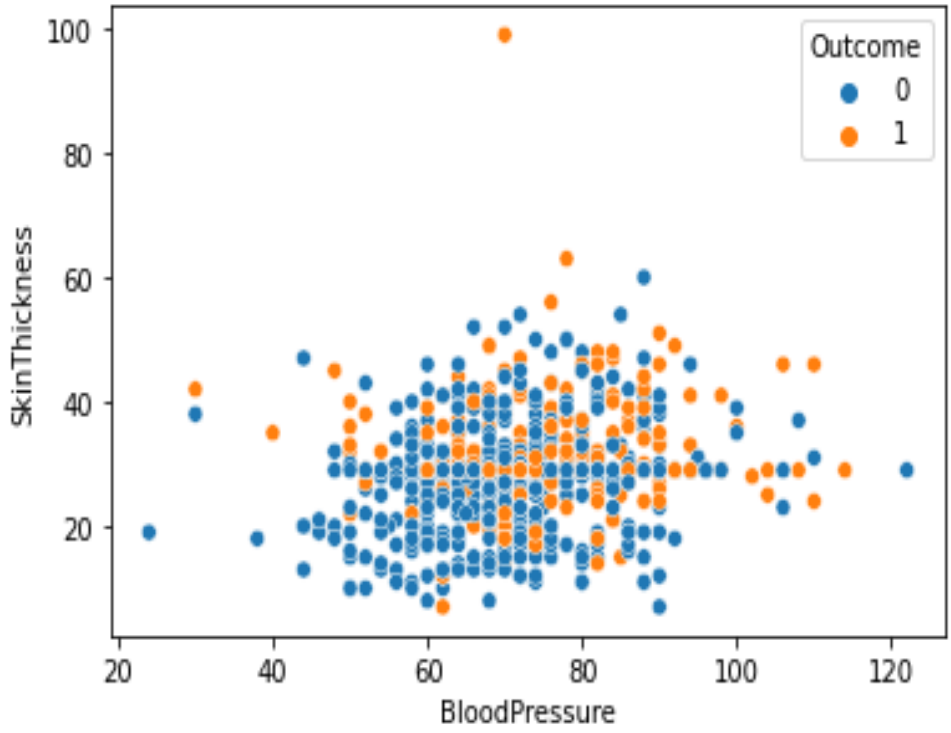

BMI by case
60
50
40
30
20
BMI
0
1
Outcome

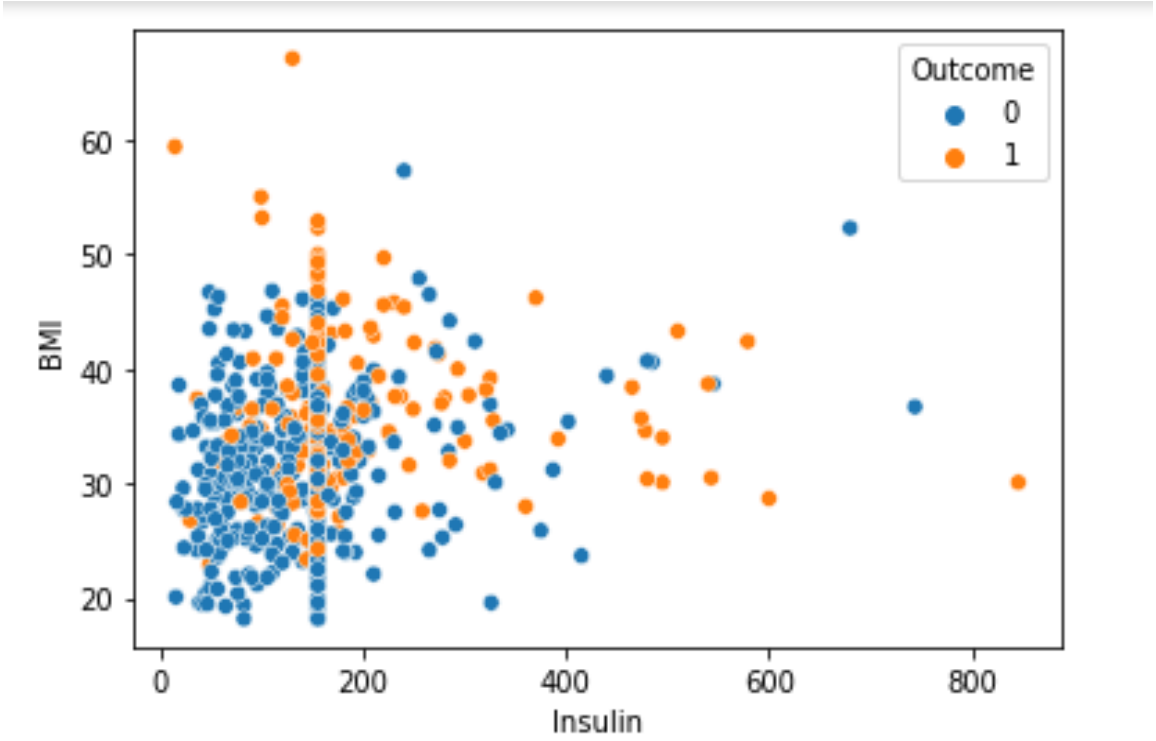

DiabetesPedigreeFunction by case
DiabetesPedigreeFunction
2.5
2.0
1.5
1.0
0.5
0.0
0
1
Outcome

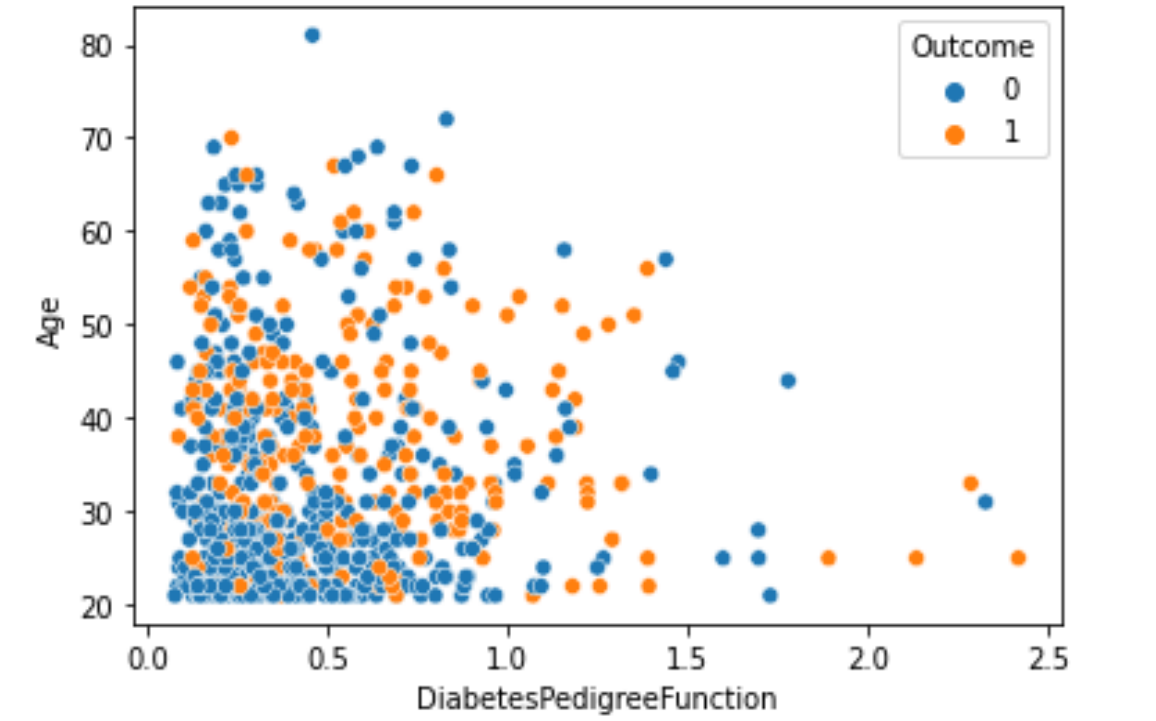

Age by case
Age
80
70
60
50
40
30
20
0
1
Outcome

➢ **Create a scatter plot for two factors**



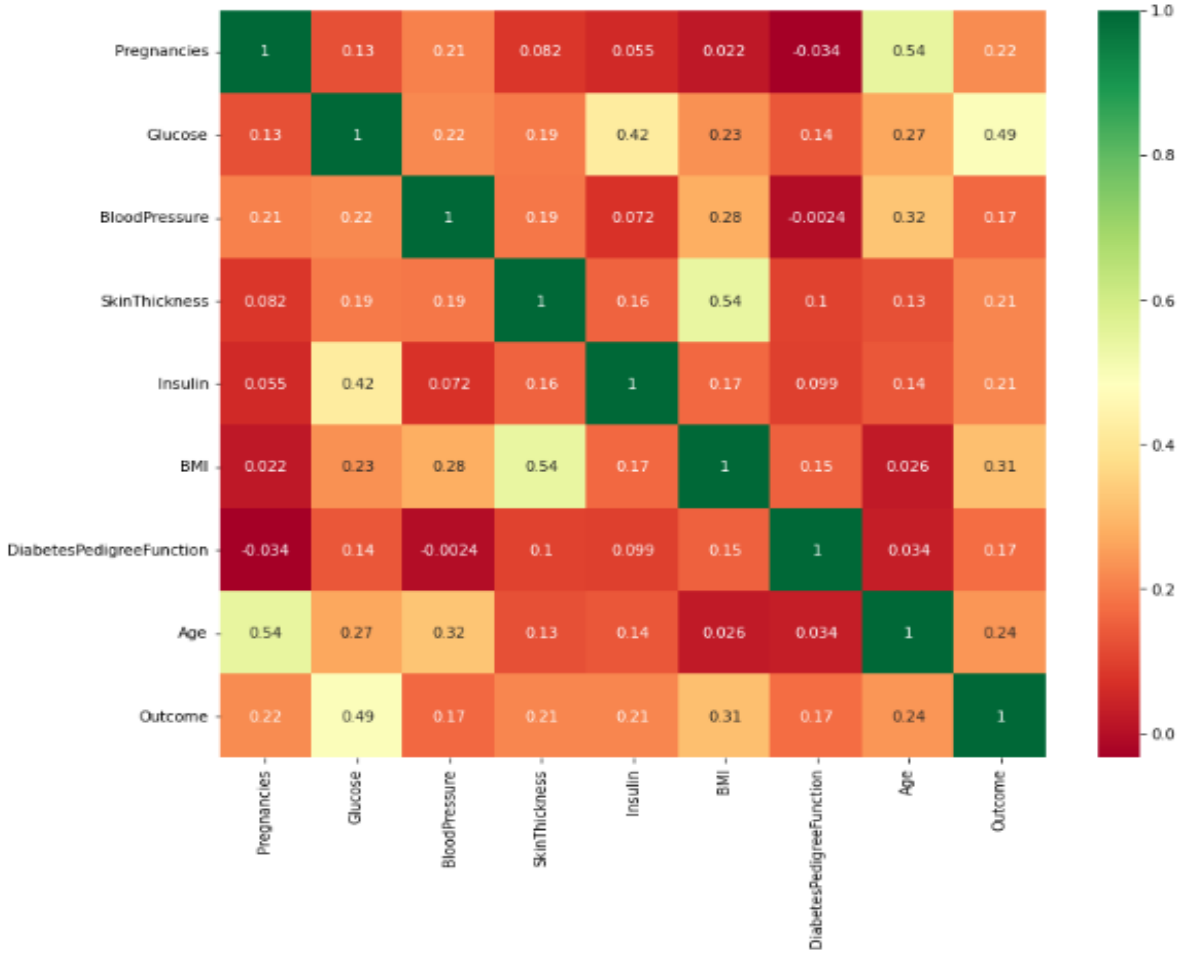



Outcome
0
1
BMI
Insulin
20
30
40
50
60
0
200
400
600
800


Outcome
0
1
Age
DiabetesPedigreeFunction
20
30
40
50
60
70
80
0.0
0.5
1.0
1.5
2.0
2.5

- **Pair plot for clean data:** The pairs plot builds on two basic figures, the histogram and the scatter plot. The histogram on the diagonal allows us to see the distribution of a single variable while the scatter plots on the upper and lower triangles show the relationship (or lack thereof) between two variables.

➢ **Heat map for clean data:**



- **Scale Features:**

  When your data has different values and even different measurement units, it can be difficult to compare them.
  The answer to this problem is scaling. We can scale data into new values that are easier to compare.
  Standard Scaler standardizes a feature by subtracting the mean and then scaling to unit variance. Unit variance means dividing all the values by the standard deviation.
  Standardize features by removing the mean and scaling to unit variance. The standard score of sample x is calculated as z = (x - u) / s.
  So, Standard Scaler removes the mean and scales the data to unit variance.

```
#Feature scaling
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)
```

- **Models:**

  1) The Neural Network classifier with one hidden layer of 100 neurons:

```
# Create the Neural Network classifier with one hidden layer of 100 neurons
nn = MLPClassifier(hidden_layer_sizes=(100,), max_iter=1000)

# Fit the classifier to the training data
nn.fit(X_train, y_train)

# Predict the labels for the test data
y_pred = nn.predict(X_test)

# Calculate the accuracy of the model
acc = accuracy_score(y_test, y_pred)

print("Accuracy-NN:", acc)
```

2) **The GBM classifier with 100 estimators:**

```
# Create the GBM classifier with 100 estimators
gbm = GradientBoostingClassifier(n_estimators=100)

# Fit the classifier to the training data
gbm.fit(X_train, y_train)

# Predict the labels for the test data
y_pred = gbm.predict(X_test)

# Calculate the accuracy of the model
acc = accuracy_score(y_test, y_pred)

print("Accuracy-GBM:", acc)
```

3) **The SVM classifier with a linear kernel:**

```
# Create the SVM classifier with a linear kernel
svm = SVC(kernel='linear')

# Fit the classifier to the training data
svm.fit(X_train, y_train)

# Predict the labels for the test data
y_pred = svm.predict(X_test)

# Calculate the accuracy of the model
acc = accuracy_score(y_test, y_pred)

print("Accuracy-SVM:", acc)
```

4) **The K Nearest Neighbors Classifier module:**

```
# Fit Model
classifier.fit(X_train, y_train)

# Predict the test set results
y_pred = classifier.predict(X_test)
y_pred

# Evaluate Model
cm = confusion_matrix(y_test, y_pred)
#print(f1_score(y_test, y_pred))

print("Accuracy-KNN:",accuracy_score(y_test, y_pred))
```

- **The accuracy:**

```
Accuracy-NN: 0.8051948051948052
Accuracy-GBM: 0.8116883116883117
Accuracy-SVM: 0.7987012987012987
Accuracy-KNN: 0.8181818181818182
```

Git-hub link: https://github.com/Abed-alnasser/Prediction-and-diagnosis-of-future-diabetes-risk..git

- **Results:**

  The results of the project revealed that the various models performed well and had high accuracy scores. The K-Nearest Neighbors model had an accuracy of 0.81. The results of the project revealed that the KNN model performed well and had high accuracy score.

  The use of machine learning techniques allowed them to accurately identify the patterns in the data and make predictions about whether a person will have diabetes or not based on their measurements.

Traditionally, Doctors have evaluated whether the person is diabetic with the help of some diagnostic test. First, they checked the serum and plasma glucose rate per hour. Diagnosis of a diabetic person has historically included fasting blood glucose higher than the prescribed rate.

Another factor like Body mass index has also played a very important role during the diagnosis of a diabetic pregnant woman, compared with women with a pre-pregnancy, BMI<29 kg/ m2, women with a BMI>29 kg/m2 have a 10-fold increased risk of developing type 2 diabetes.

Consequently, both of these factors such as BMI and Plasma glucose have also significantly co-related attributes during our study. Although the accuracy of the different models varied, the K-Nearest Neighbors model performed well.

The performance of the models is not always the same across different scenarios and data sets. Therefore, it is important to thoroughly evaluate the models and make sure that they can generalize well to new data.