[9]
```python
##檢查資料表結構
diabetes_data.info(verbose=True)
```

```
<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
```

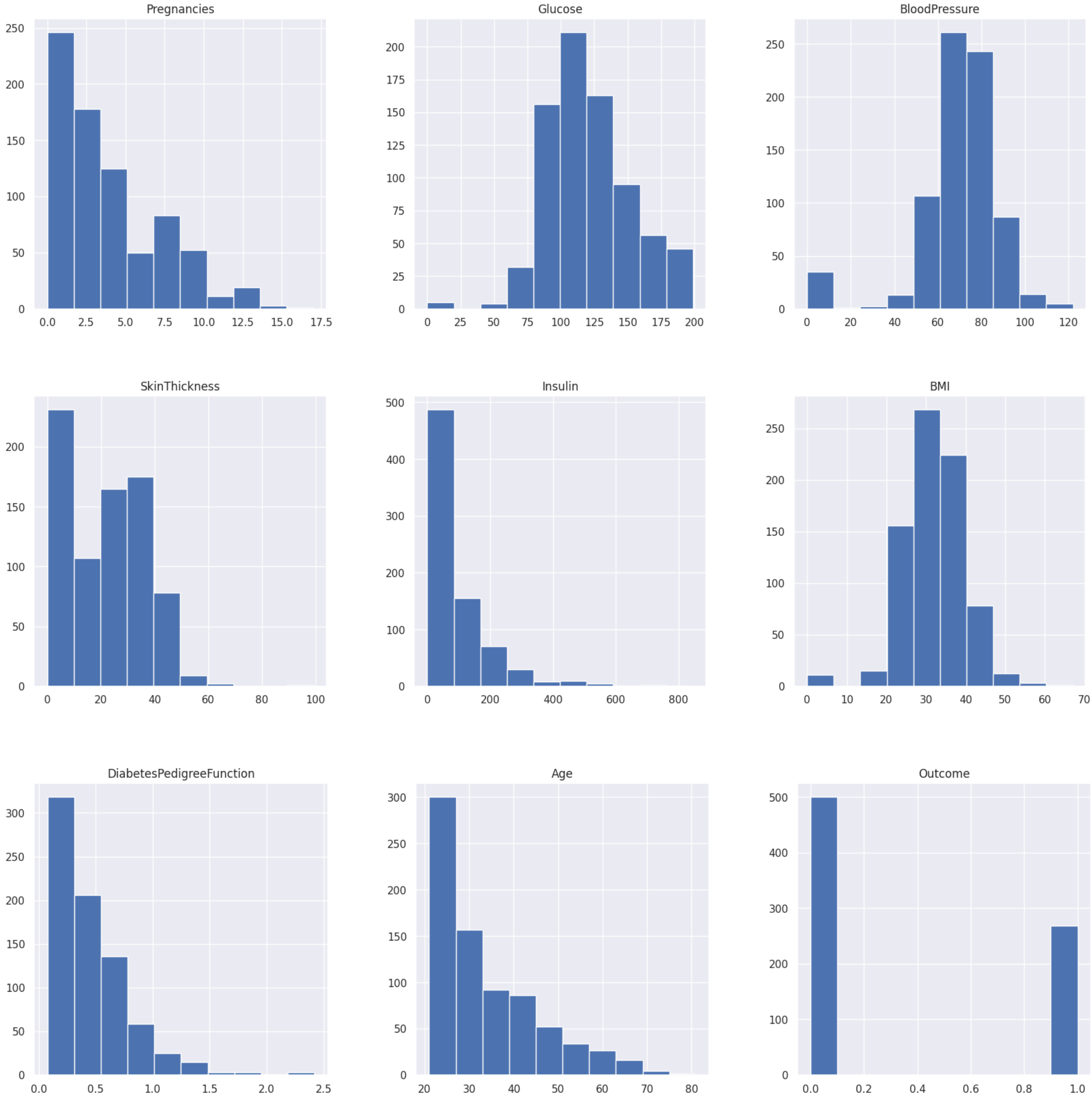
```python
##匯出統計摘要，了解數值型欄位的分布
diabetes_data.describe().T
```

	count	mean	std	min	25%	50%	75%	max
Pregnancies	768.0	3.845052	3.369578	0.000	1.00000	3.0000	6.00000	17.00
Glucose	768.0	120.894531	31.972618	0.000	99.00000	117.0000	140.25000	199.00
BloodPressure	768.0	69.105469	19.355807	0.000	62.00000	72.0000	80.00000	122.00
SkinThickness	768.0	20.536458	15.952218	0.000	0.00000	23.0000	32.00000	99.00
Insulin	768.0	79.799479	115.244002	0.000	0.00000	30.5000	127.25000	846.00
BMI	768.0	31.992578	7.884160	0.000	27.30000	32.0000	36.60000	67.10
DiabetesPedigreeFunction	768.0	0.471876	0.331329	0.078	0.24375	0.3725	0.62625	2.42
Age	768.0	33.240885	11.760232	21.000	24.00000	29.0000	41.00000	81.00
Outcome	768.0	0.348958	0.476951	0.000	0.00000	0.0000	1.00000	1.00

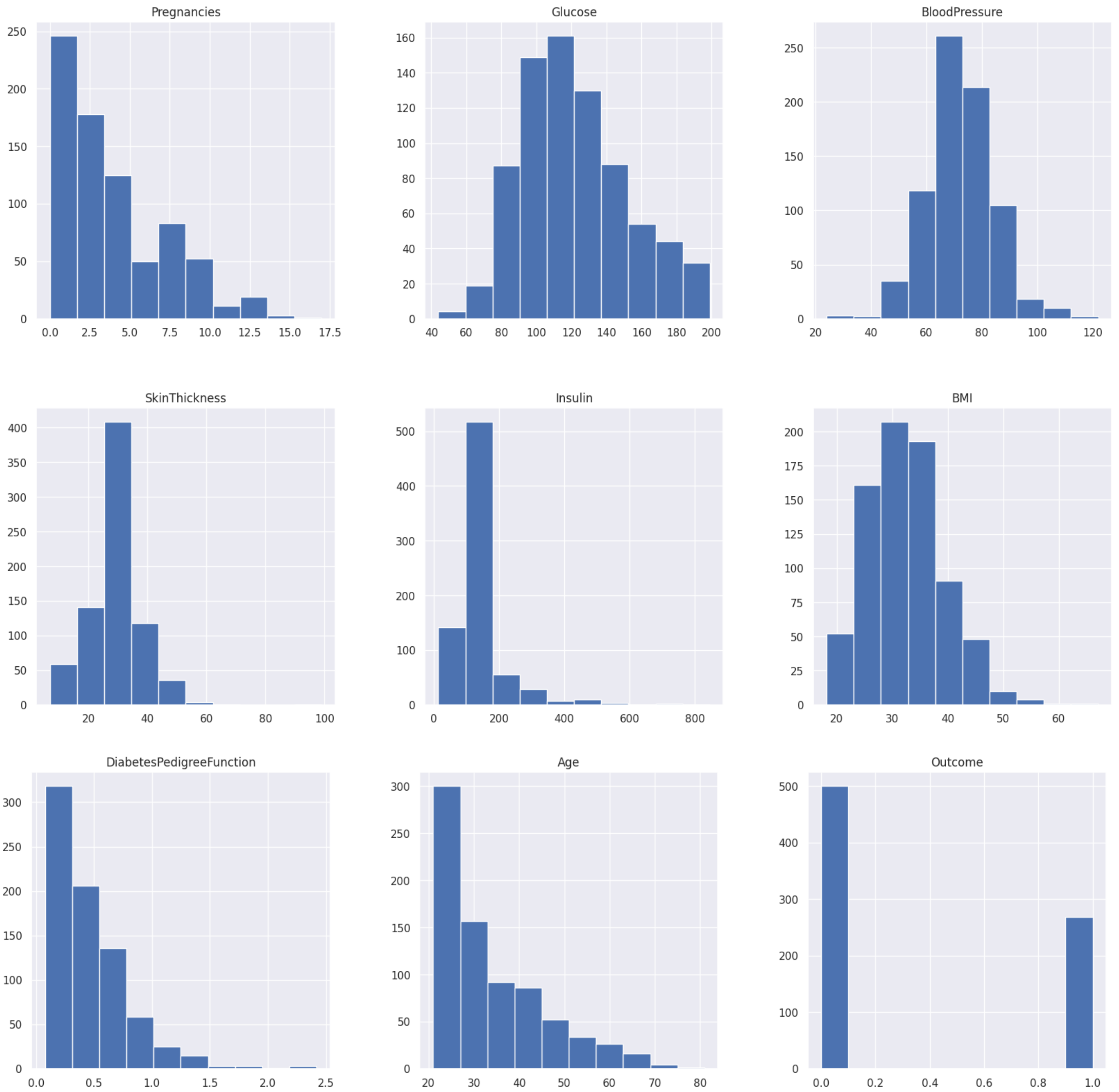
```python
## showing the count of Nans
print(diabetes_data_copy.isnull().sum())
```

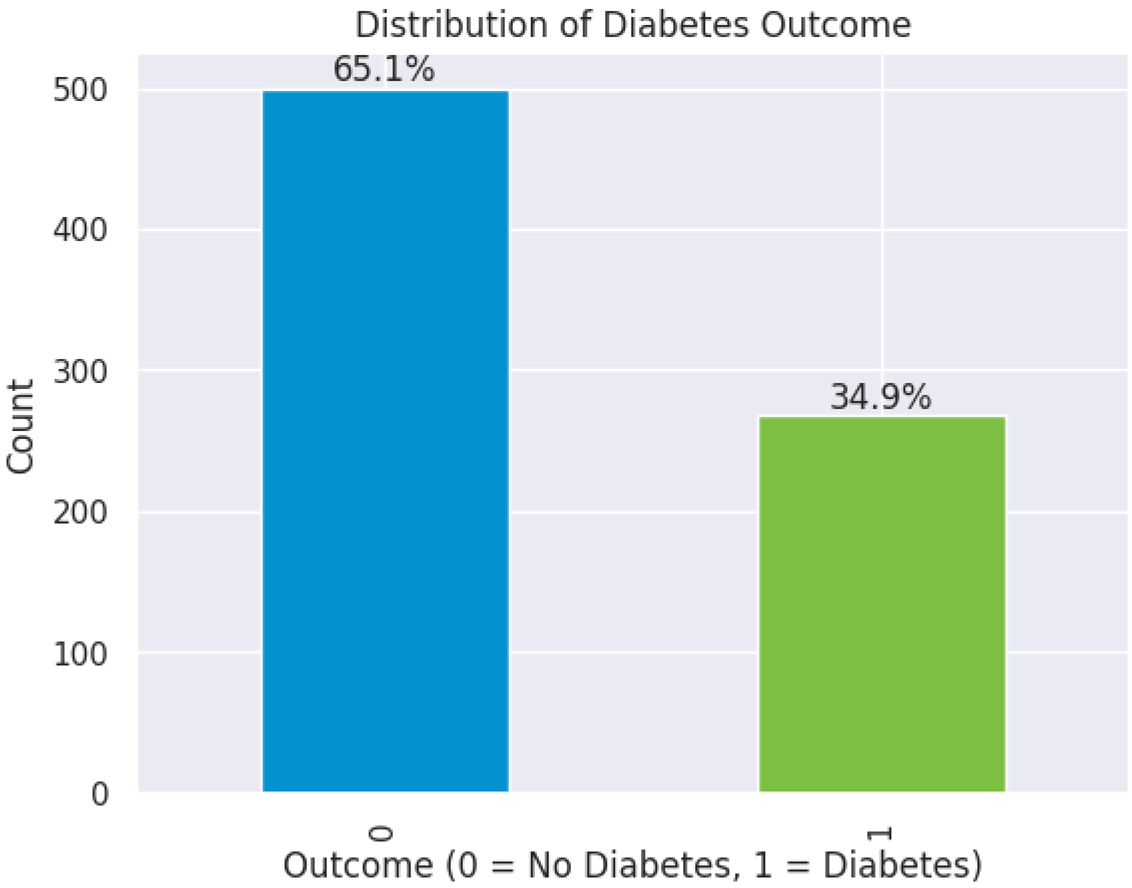
```
Pregnancies                   0
Glucose                       5
BloodPressure                35
SkinThickness               227
Insulin                     374
BMI                          11
DiabetesPedigreeFunction      0
Age                           0
Outcome                       0
dtype: int64
```

```
##了解資料分佈以利估算nan值
p = diabetes_data.hist(figsize = (20,20))
```



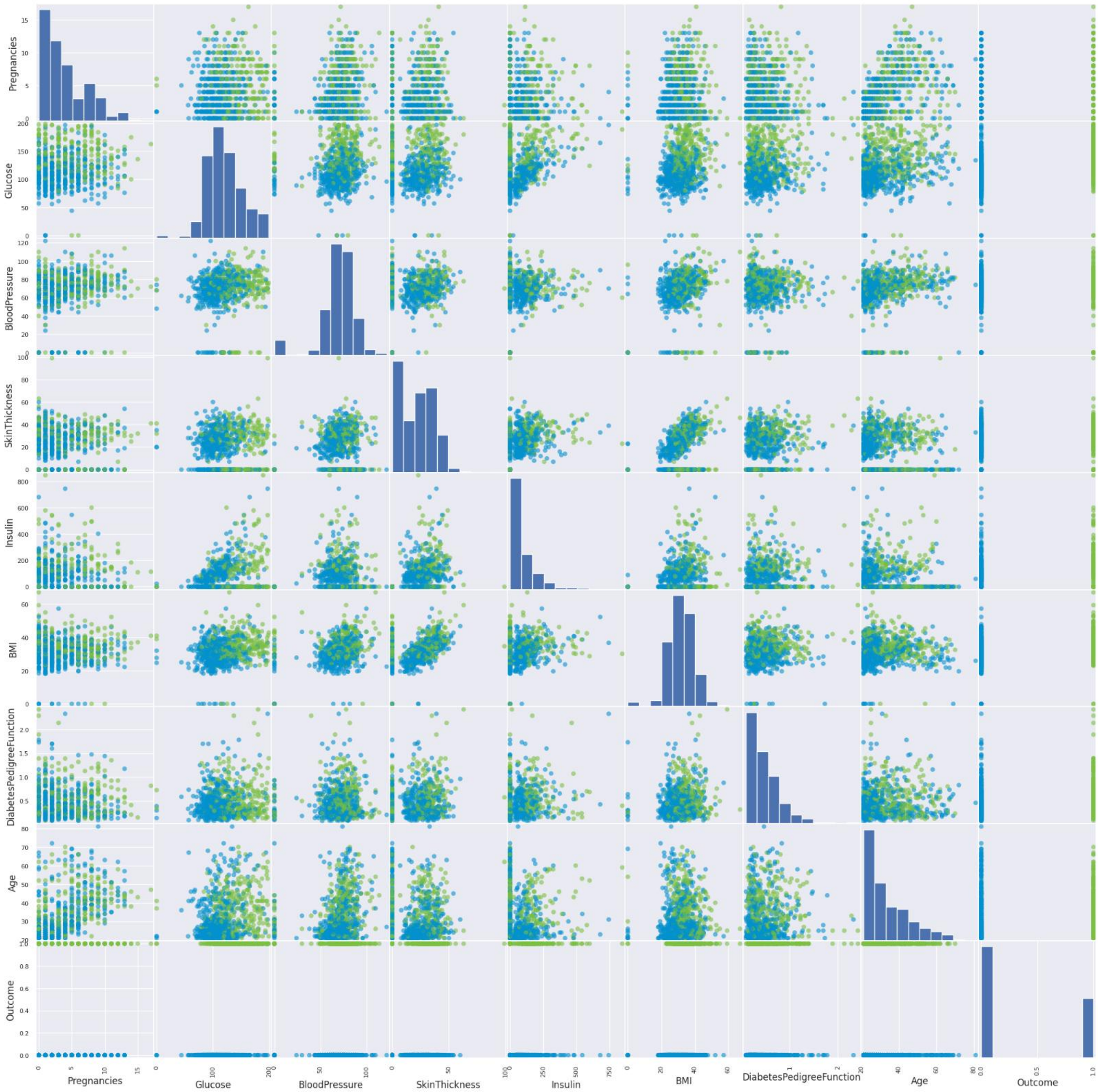
```python
##缺失值處理:以平均數,中位數替換成nan的值
#平均數
diabetes_data_copy['Glucose'].fillna(diabetes_data_copy['Glucose'].mean(), inplace = True)
diabetes_data_copy['BloodPressure'].fillna(diabetes_data_copy['BloodPressure'].mean(), inplace = True)
#中位數
diabetes_data_copy['SkinThickness'].fillna(diabetes_data_copy['SkinThickness'].median(), inplace = True)
diabetes_data_copy['Insulin'].fillna(diabetes_data_copy['Insulin'].median(), inplace = True)
diabetes_data_copy['BMI'].fillna(diabetes_data_copy['BMI'].median(), inplace = True)

##替換後圖形分佈
p = diabetes_data_copy.hist(figsize = (20,20))
```



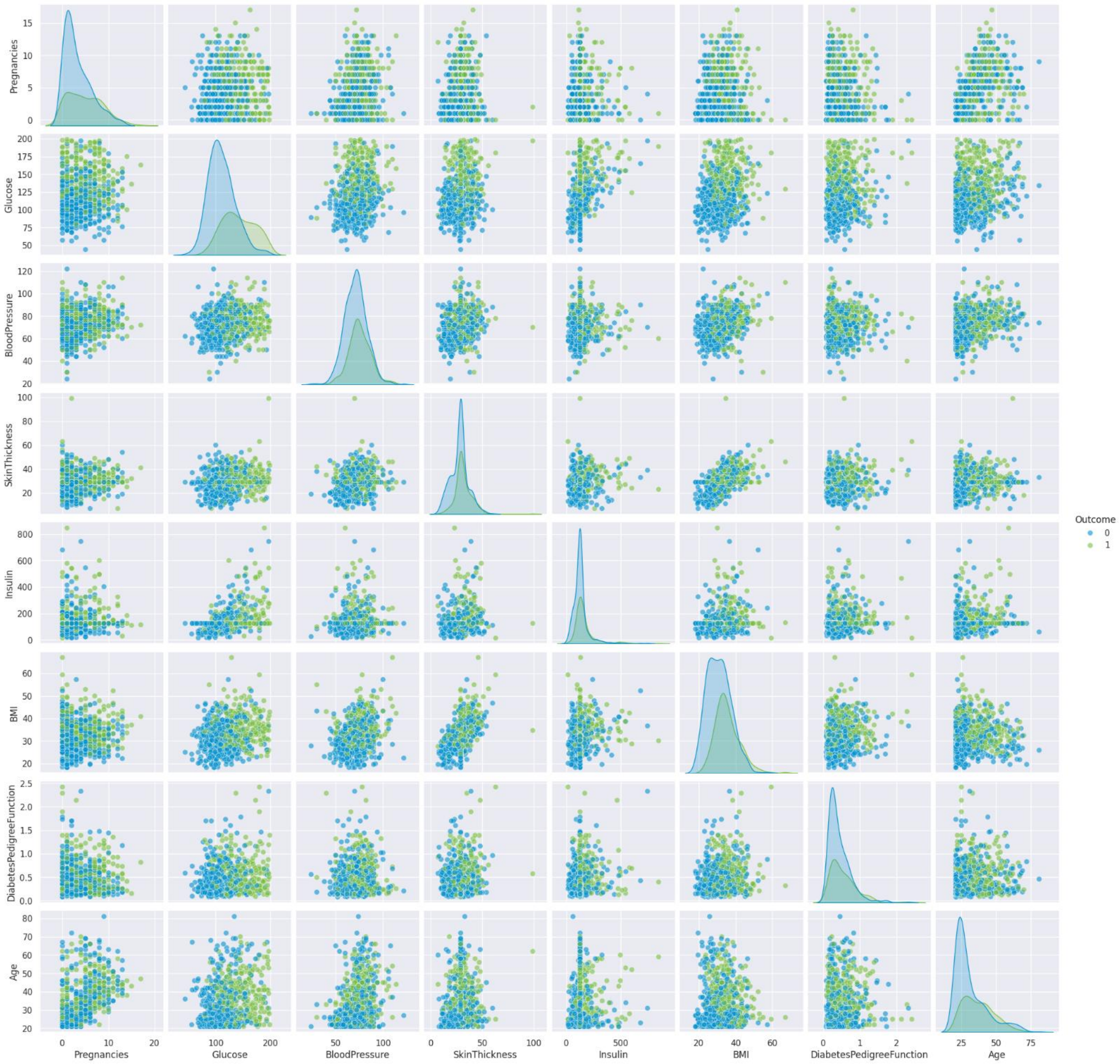


Distribution of Diabetes Outcome
65.1%
34.9%
500
400
300
200
100
0
Count
0
1
Outcome (0 = No Diabetes, 1 = Diabetes)

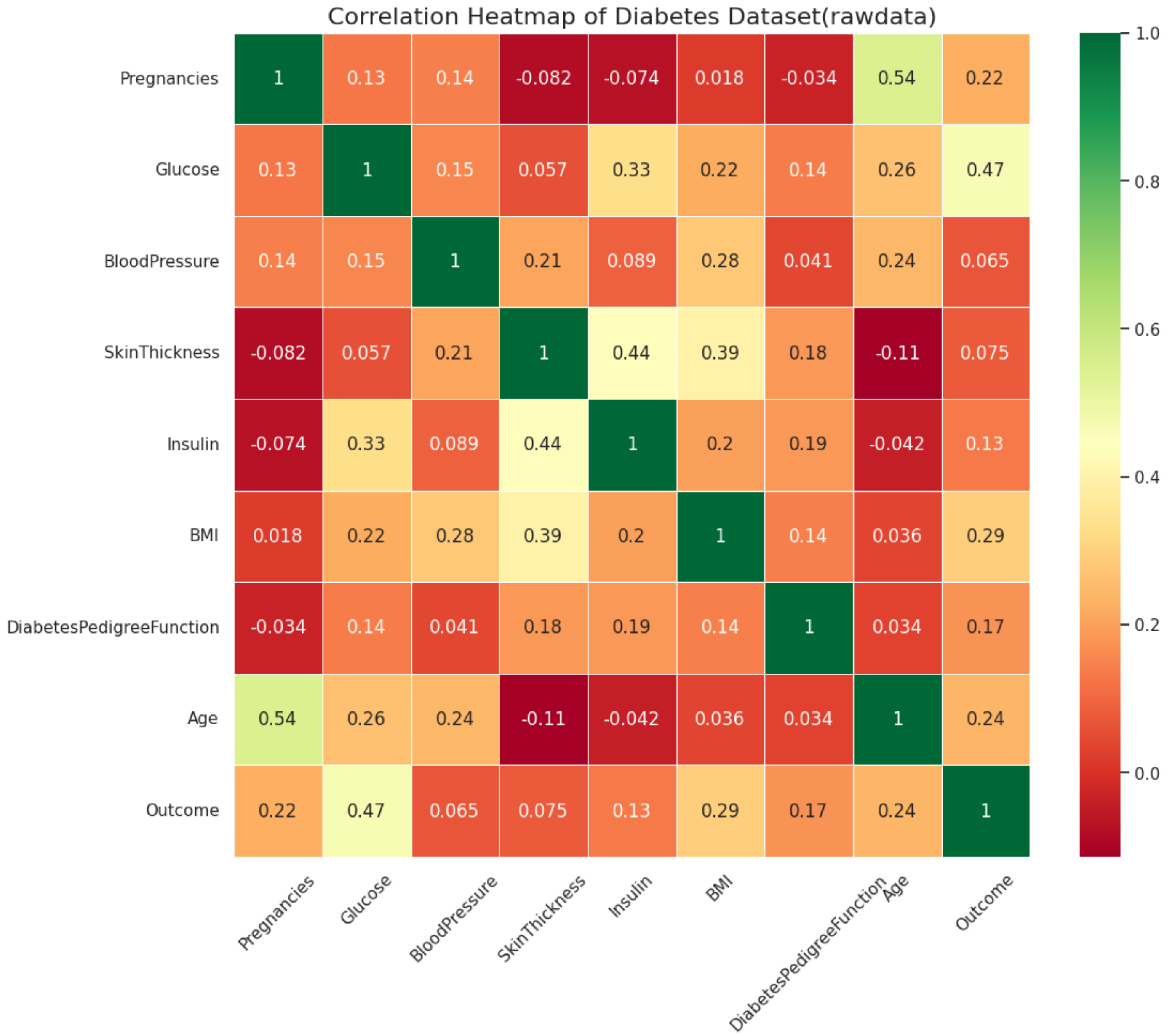
Scatter Matrix of Diabetes Dataset with Outcome Colors



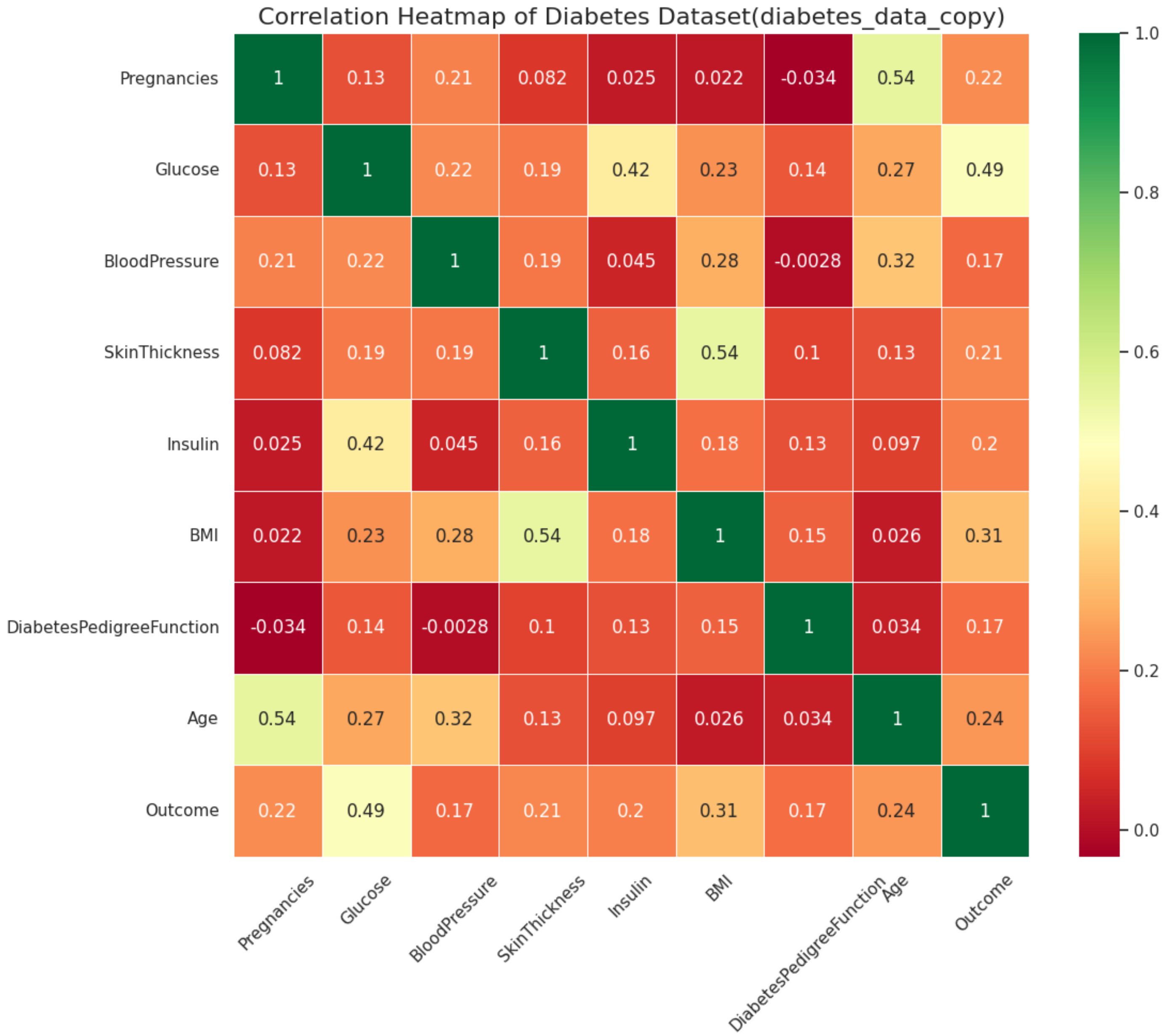
##用Seaborn的pairplot取代傳統scatter_matrix



```
##相關係數熱圖:rawdata
```



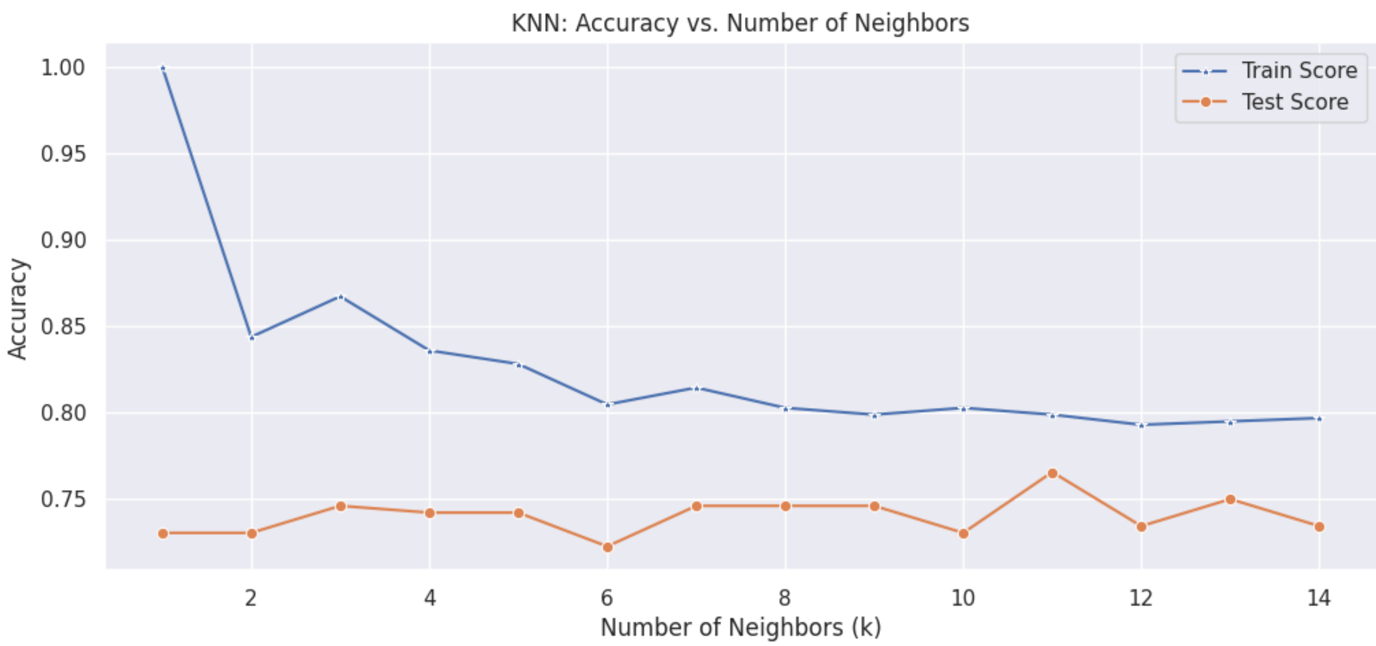
##相關係數熱圖:完成缺失值處理



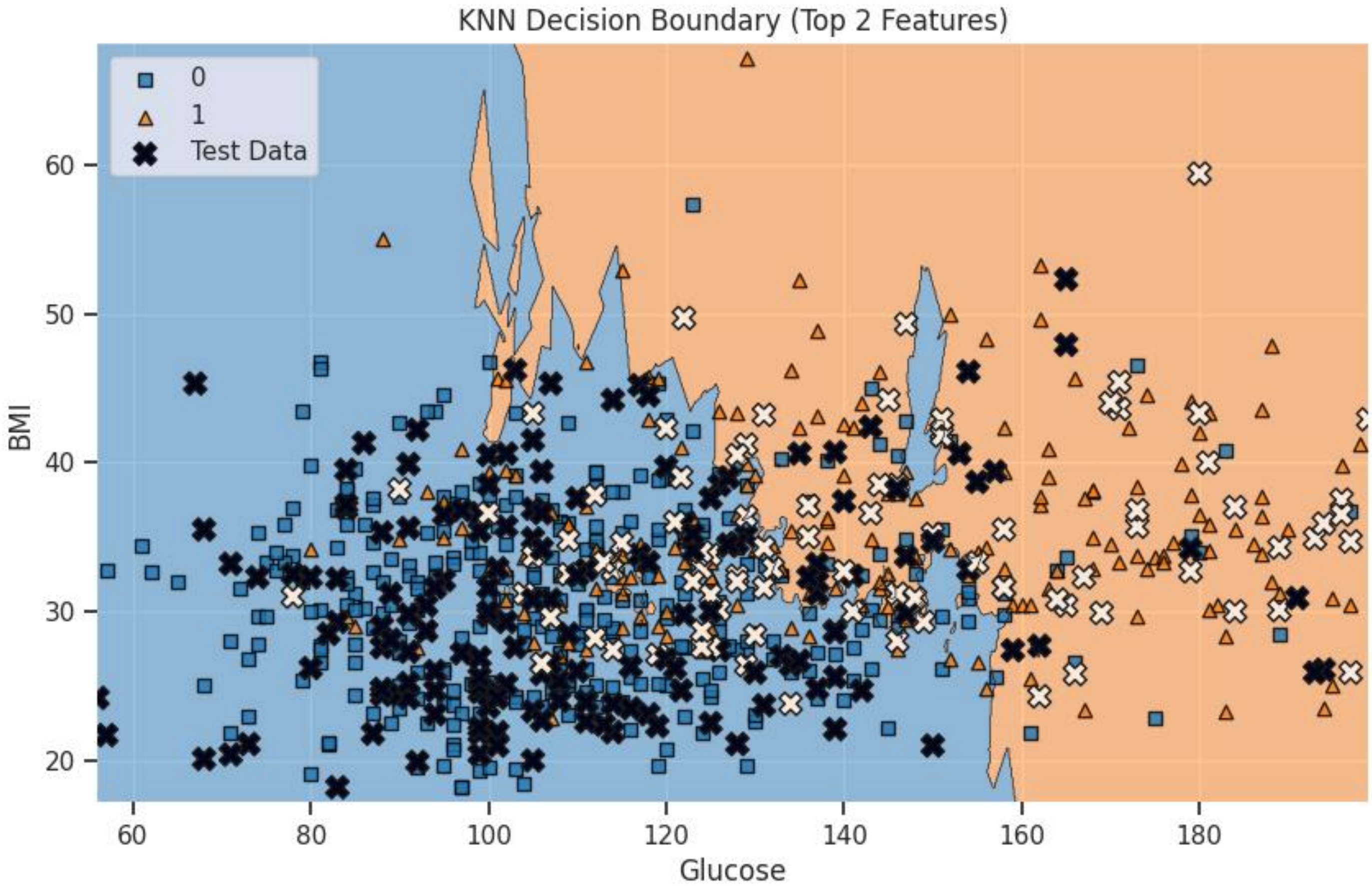
```python
##特徵標準化(Feature  Scaling/Standardization):所有數值特徵做z-score標準化,產生新的DataFrame
#StandardScaler:把數值欄位轉換成平均值為0,標準差為1的標準化資料
from  sklearn.preprocessing  import  StandardScaler
sc_X  =  StandardScaler()
X  =    pd.DataFrame(sc_X.fit_transform(diabetes_data_copy.drop(["Outcome"],axis  =  1),),
              columns=['Pregnancies',  'Glucose',  'BloodPressure',  'SkinThickness',  'Insulin','BMI',  'DiabetesPedigreeFunction',  'Age'])
X.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
0	0.639947	0.865108	-0.033518	0.670643	-0.181541	0.166619	0.468492	1.425995
1	-0.844885	-1.206162	-0.529859	-0.012301	-0.181541	-0.852200	-0.365061	-0.190672
2	1.233880	2.015813	-0.695306	-0.012301	-0.181541	-1.332500	0.604397	-0.105584
3	-0.844885	-1.074652	-0.529859	-0.695245	-0.540642	-0.633881	-0.920763	-1.041549
4	-1.141852	0.503458	-2.680669	0.670643	0.316566	1.549303	5.484909	-0.020496

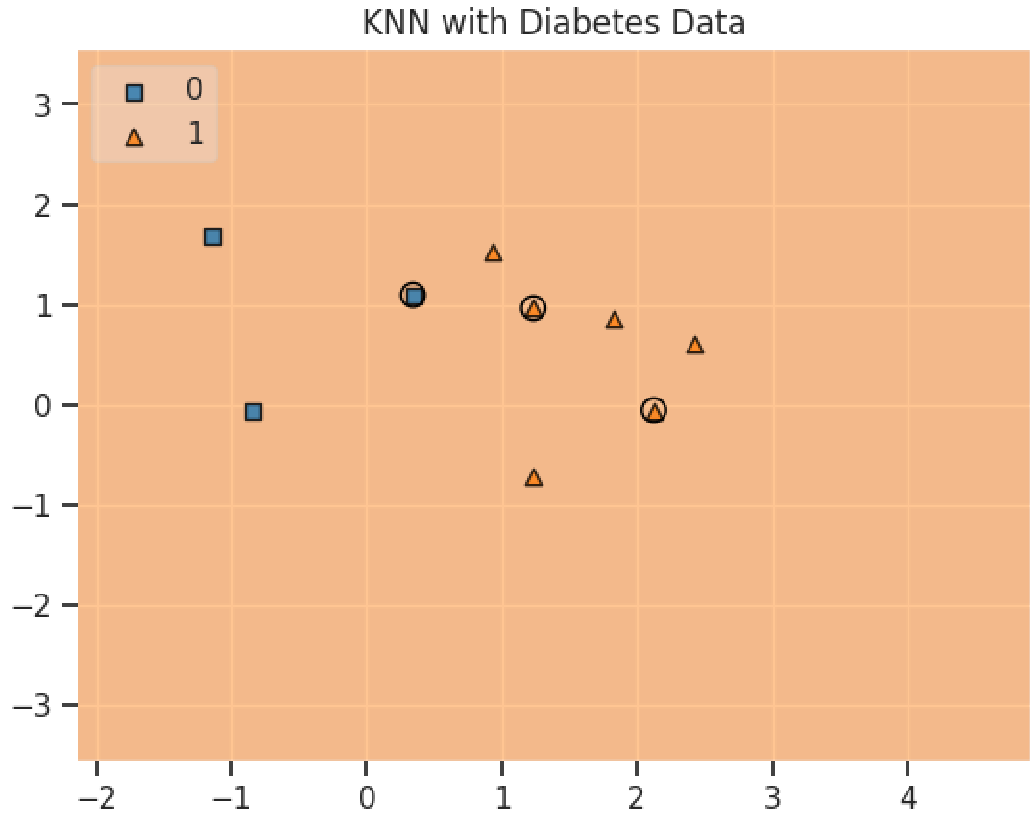
##畫出k值對應的準確率曲線



```
##選擇最重要的兩個特徵(使用與Outcome相關性最大的兩個)
```

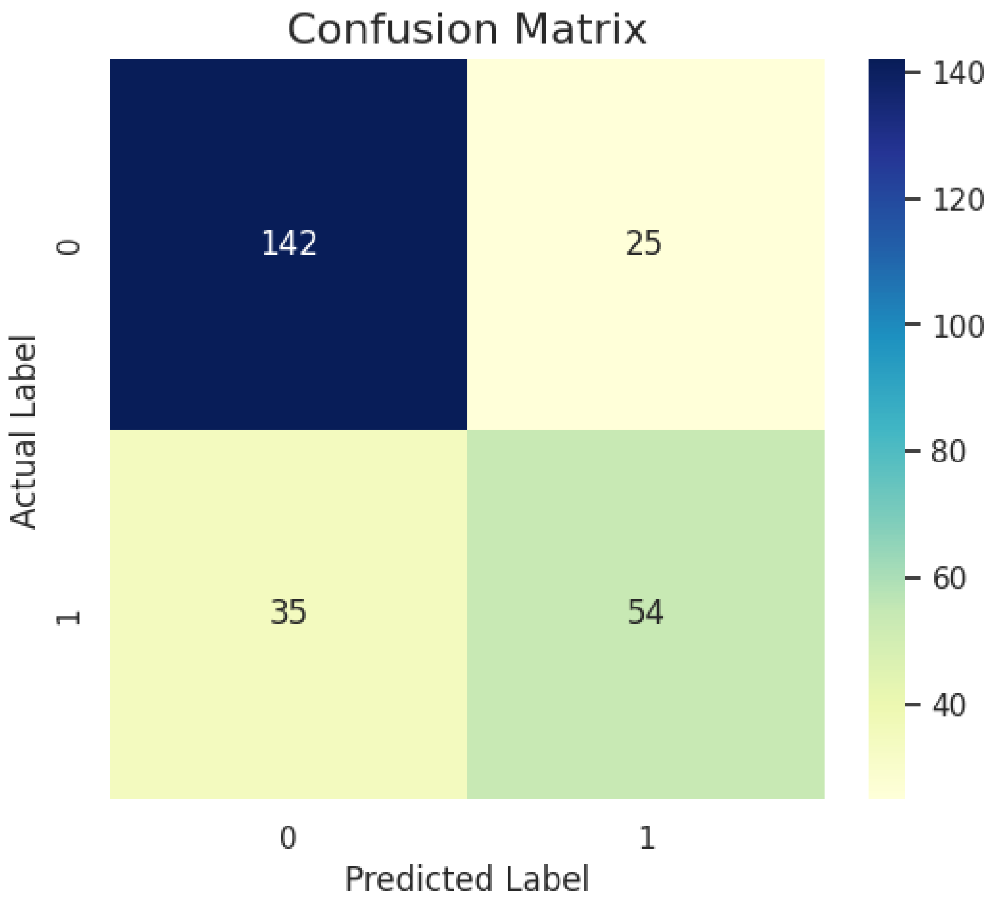



```
#只投影到前兩個特徵的 2D 表示,固定其他維度
```



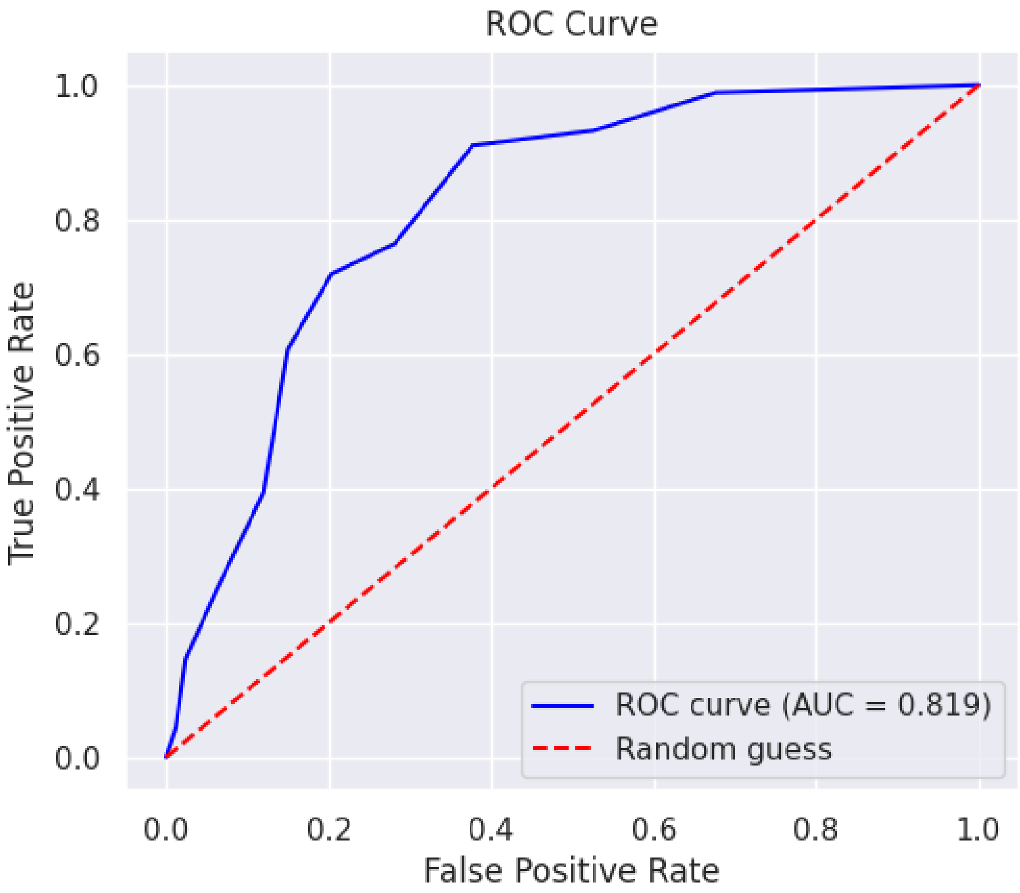
```
#用sklearn的confusion_matrix計算KNN模型的預測結果
```

Predicted True	0	1	All
0	142	25	167
1	35	54	89
All	177	79	256




```
##數據說明:
#1. 類別不平衡：0 類別樣本數多（167 vs 89），模型對0類的表現比1類好。
#2. 召回率偏低：1 類召回率0.61(模型漏掉約39%的實際1類樣本)。
#精確度 vs 召回率:
#類別 0：精確度略低於召回率,偶爾誤把1類預測成 0。
#類別 1：精確度高於召回率,預測為1的結果較可靠，但漏掉許多實際1類。
```

```
              precision    recall  f1-score   support

           0       0.80      0.85      0.83       167
           1       0.68      0.61      0.64        89

    accuracy                           0.77       256
   macro avg       0.74      0.73      0.73       256
weighted avg       0.76      0.77      0.76       256
```




```
##使用GridSearchCV調整KNN的n_neighbors超參數
```

```
Best Score:0.7721840251252015
Best Parameters: {'n_neighbors': np.int64(25)}
```