一、資料集特徵資料說明、屬性特性說明

1. p_id (Patient ID):
   檢測者編號
2. no_times_pregnant (Number of Pregnant Times):
   檢測者懷孕的次數
3. glucose_concentration (Glucose Concentration):
   檢測者在口服葡萄糖耐量試驗中 2 小時後的血漿葡萄糖濃度，用來評估葡萄糖耐受性
4. blood_pressure (Blood Pressure):
   檢測者的血壓（舒張壓），單位毫米汞柱（mm Hg）
5. skin_fold_thickness (Skin Fold Thickness):
   檢測者的三頭肌皮褶厚度，單位毫米（mm）
6. serum_insulin (Serum Insulin):
   檢測者在口服葡萄糖耐量試驗後 2 小時的血清胰島素水平，單位 μU/ml
7. bmi (Body Mass Index):
   檢測者的身體質量指數，單位公斤/平方公尺（$kg/m^2$）
8. diabetes_pedigree (Diabetes Pedigree Function):
   檢測者的糖尿病家族病史分數
9. age:
   檢測者年齡
10. diabetes:
    檢測者是否患有糖尿病，1 表示患有糖尿病，0 表示沒有。
    此為本資料集建立模型所期望預測的的目標變數。

Nomial: p_id
Ordinal: no_times_pregnant, diabetes
Interval: glucose_concentration, blood_pressure, skin_fold_thickness, serum_insulin, bmi, diabetes_pedigree, age

本資料集收集檢測者的一些生理特徵數據，並且用於預測檢測者是否患有糖尿病。

二、對特徵做甚麼樣的分析？哪些前處理？採用哪些特徵？原因？

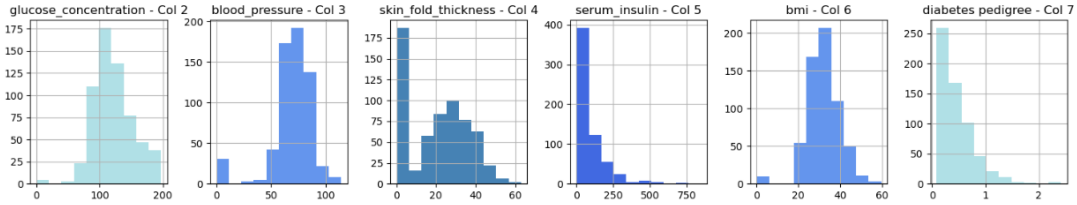
Analysis 透過圖像化數據觀察各特徵分布

繪製各特徵數據分布直條圖

```
fig, axes = plt.subplots(nrows=1, ncols=6, figsize=(15, 3))

train['glucose_concentration'].hist(ax=axes[0], bins=10, color='PowderBlue')
axes[0].set_title('glucose_concentration - Col 2')
train['blood_pressure'].hist(ax=axes[1], bins=10, color='CornflowerBlue')
axes[1].set_title('blood_pressure - Col 3')
train['skin_fold_thickness'].hist(ax=axes[2], bins=10, color='SteelBlue')
axes[2].set_title('skin_fold_thickness - Col 4')
train['serum_insulin'].hist(ax=axes[3], bins=10, color='royalblue')
axes[3].set_title('serum_insulin - Col 5')
train['bmi'].hist(ax=axes[4], bins=10, color='CornflowerBlue')
axes[4].set_title('bmi - Col 6')
train['diabetes pedigree'].hist(ax=axes[5], bins=10, color='PowderBlue')
axes[5].set_title('diabetes pedigree - Col 7')

plt.tight_layout()
plt.show()
```




繪製各特徵與 diabetes 關係圖

```
import seaborn as sns
# draw pairplot for 'diabetes' and other indicators
sns.pairplot(train,hue="diabetes")
plt.show()
```

## Preprocessing

Imputer()：

將資料中缺失項取代為 NaN，並統計若某特徵中有超過 20%的 NaN，表示該特徵可用性低，將該特徵刪除（skin_fold_thickness, serum_insulin），確定最終使用的資料後將各特徵 NaN 項修正。

採用特徵：no_times_pregnant, glucose_concentration, blood_pressure, bmi, diabetes pedigree, age

```
# replace with NaN
colume = ['glucose_concentration','blood_pressure','skin_fold_thickness','serum_insulin']
train[colume] = train[colume].replace(0,np.nan)
```

```
# delete the row with empty data more than 20%
thresh_count = train.shape[0]*0.8
train = train.dropna(thresh=thresh_count,axis=1)
```

```
train.head()
```

| | p_id | no_times_pregnant | glucose_concentration | blood_pressure | bmi | diabetes pedigree | age | diabetes |
|---|---|---|---|---|---|---|---|---|
| **0** | 316 | 2 | 112.0 | 68.0 | 34.1 | 0.315 | 26 | 0 |
| **1** | 25 | 11 | 143.0 | 94.0 | 36.6 | 0.254 | 51 | 1 |
| **2** | 710 | 2 | 93.0 | 64.0 | 38.0 | 0.674 | 23 | 1 |
| **3** | 658 | 1 | 120.0 | 80.0 | 38.9 | 1.162 | 41 | 0 |
| **4** | 542 | 3 | 128.0 | 72.0 | 32.4 | 0.549 | 27 | 1 |

```
from sklearn.impute import SimpleImputer
# fix the remaining missing data
imr = SimpleImputer()
colume_2 = ['glucose_concentration','blood_pressure']
train[colume_2] = imr.fit_transform(train[colume_2])
```

MinMaxScaler()：

將 train 和 test 兩份資料 Normalize 至 0~1 之間，所有特徵範圍設定相同以便分析。

```
from sklearn.preprocessing import MinMaxScaler
min_max_scaler = MinMaxScaler()
# MinMaxScaler
x_scaled = min_max_scaler.fit_transform(train.values)
train = pd.DataFrame(x_scaled, columns=train.columns)
y_scaled = min_max_scaler.fit_transform(test.values)
test = pd.DataFrame(y_scaled, columns=test.columns)
```

```
train = train.drop('p_id',axis=1)
train.head()
```

| | no_times_pregnant | glucose_concentration | blood_pressure | bmi | diabetes pedigree | age | diabetes |
|---|---|---|---|---|---|---|---|
| **0** | 0.117647 | 0.444444 | 0.488889 | 0.574074 | 0.101196 | 0.083333 | 0.0 |
| **1** | 0.647059 | 0.647059 | 0.777778 | 0.616162 | 0.075149 | 0.500000 | 1.0 |
| **2** | 0.117647 | 0.320261 | 0.444444 | 0.639731 | 0.254483 | 0.033333 | 1.0 |
| **3** | 0.058824 | 0.496732 | 0.622222 | 0.654882 | 0.462852 | 0.333333 | 0.0 |
| **4** | 0.176471 | 0.549020 | 0.533333 | 0.545455 | 0.201110 | 0.100000 | 1.0 |

```
test.head()
```

| | p_id | no_times_pregnant | glucose_concentration | blood_pressure | skin_fold_thickness | serum_insulin | bmi | diabetes pedigree | age |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 0.589918 | 0.857143 | 0.703518 | 0.696721 | 0.333333 | 0.000000 | 0.557377 | 0.077935 | 0.444444 |

train_test_split()：

設定 70%測試資料及 30%訓練資料。

```
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.3,stratify=y)
```

三、基於什麼理由選擇哪個分類器?

Classification

透過測試各分類器（KNeighborsClassifier, GaussianNaiveBayes, DecisionTreeClassifier, MLPClassifier, RandomForestClassifier）調整參數，輸出 Accuracy 分數，決定選用分數最高的 RandomForestClassifier 進行分類操作。

```
classifiers = [
    KNeighborsClassifier(2),
    GaussianNB(),
    DecisionTreeClassifier(max_depth=3,min_samples_split=3),
    MLPClassifier(hidden_layer_sizes=(20,20),max_iter=100),
    RandomForestClassifier(max_depth=5,min_samples_split=3),
    ]

log = []
for clf in classifiers:
    clf.fit(x_train,y_train)
    name = clf.__class__.__name__
    print('*'*30)
    print(name)
    predictions = clf.predict(x_test)
    acc = accuracy_score(y_test,predictions)
    print('Accuracy:{:.4%}'.format(acc))
    log.append([name,acc*100])

print('*'*30)
```

```
******************************
KNeighborsClassifier
Accuracy:76.7568%
******************************
GaussianNB
Accuracy:74.0541%
******************************
DecisionTreeClassifier
Accuracy:74.0541%
******************************
MLPClassifier
Accuracy:71.3514%
******************************
RandomForestClassifier
Accuracy:78.9189%
******************************
```

四、採用的評估指標結果與觀察

輸出實際預測的資料 prediction 的 Accuracy 分數，評估模型對應預測結果的可用程度

```
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.25,stratify=y)
```

```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

accuracy = accuracy_score(y_test, raw_predictions)
print(f'Accuracy: {accuracy:.4f}')
```

```
Accuracy: 0.5065
```

在建立 RandomForest 分類模型的時候，有執行多次程式，調整參數，還有測試同參數多次測試的結果區間，測試分類器時 RandomForest 的 Accuracy 分數較穩定且較高，所以選用 RandomForest，但多次實際輸出預測資料的分數還是會一定程度的浮動。

五、將預測結果上傳至 kaggle 並截圖測試的分數