Assignment 02: Evaluate the Diabetes Dataset

The comments/sections provided are your cues to perform the assignment. You don't need to limit yourself to the number of rows/cells provided. You can add additional rows in each section to add more lines of code.

If at any point in time you need help on solving this assignment, view our demo video to understand the different steps of the code.

Happy coding!

1: Import the dataset

In [1]:
```
#Import the required libraries
import pandas as pd
import numpy as np
```

In [5]:
```
#Import the diabetes dataset
df_diabetes = pd.read_csv('pima-indians-diabetes.data' ,header =None)
```

2: Analyze the dataset

In [7]:
```
#View the first five observations of the dataset
df_diabetes.head()
```

Out[7]:

	0	1	2	3	4	5	6	7	8
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

3: Find the features of the dataset

In [16]:
```
#Use the .NAMES file to view and set the features of the dataset
df_diabetes.columns= ['pregnant','glucose','bp','skin' ,'insulin', 'bmi','pedigree' ,'age','label']
```

In [17]:
```
#Use the feature names set earlier and fix it as the column headers of the dataset
df_diabetes.columns= ['pregnant','glucose','bp','skin' ,'insulin', 'bmi','pedigree' ,'age','label']
```

In [18]:
```
#Verify if the dataset is updated with the new headers
df_diabetes.columns
```

Out[18]:
```
Index(['pregnant', 'glucose', 'bp', 'skin', 'insulin', 'bmi', 'pedigree',
       'age', 'label'],
      dtype='object')
```

In [19]:
```
#View the number of observations and features of the dataset
df_diabetes.shape
```

Out[19]:
```
(768, 9)
```

4: Find the response of the dataset

In [20]:
```
#Select features from the dataset to create the model
feature_cols = ['pregnant','insulin','bmi','age']
```

In [21]:
```
#Create the feature object
X = df_diabetes[feature_cols]
```

In [23]:
```
#Create the reponse object
Y = df_diabetes[['label']]
```

In [24]:
```
#View the shape of the feature object
X.shape
```

Out[24]:
```
(768, 4)
```

In [25]:
```
#View the shape of the target object
Y.shape
```

Out[25]:
```
(768, 1)
```

5: Use training and testing datasets to train the model

In [27]:
```
#Split the dataset to test and train the model
from sklearn.model_selection import train_test_split
x_train , x_test , y_train ,y_test = train_test_split(X,Y,random_state=1)
```

6: Create a model to predict the diabetes outcome

In [29]:
```
# Create a logistic regression model using the training set
from sklearn.linear_model import LogisticRegression
log = LogisticRegression().fit(x_train,y_train);
```

```
C:\Users\Mohannad\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:432: FutureWarning: De
fault solver will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
C:\Users\Mohannad\Anaconda3\lib\site-packages\sklearn\utils\validation.py:724: DataConversionWarning:
A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_sample
s, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

In [30]:
```
#Make predictions using the testing set
log.predict(x_test)
```

Out[30]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0,
       0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0,
       1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0], dtype=int64)
```

In [34]:
```
y_hat = log.predict(x_test)
```

7: Check the accuracy of the model

In [32]:
```
#Evaluate the accuracy of your model
log.score(x_train,y_train)
```

Out[32]:
```
0.6857638888888888
```

In [33]:
```
log.score(x_test,y_test)
```

Out[33]:
```
0.6979166666666666
```

In [37]:
```
from sklearn import metrics
print( metrics.accuracy_score(y_test,y_hat))
```

```
0.6979166666666666
```

In [63]:
```
#Print the first 30 actual and predicted responses
print('the actual {}'.format(y_test.values[0:30]))
print(' the predicted {}'.format(y_hat[0:30]))
```

```
the actual [0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0]
 the predicted [0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0]
```

In [ ]:
```

```