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!

Building a model to predict Diabetes

DESCRIPTION

Problem:

The given dataset lists the glucose level readings of several pregnant women taken either during a survey examination or routine medical care. It specifies if the 2-hour post-load plasma glucose was at least 200 mg/dl. Analyze the dataset to:

1. Find the features of the dataset,
2. Find the response label of the dataset,
3. Create a model to predict the diabetes outcome,
4. Use training and testing datasets to train the model, and
5. Check the accuracy of the model.

1: Import the dataset

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

In [2]:
```
#Import the diabetes dataset
df_diabetes_data = pd.read_csv('D:\\NIPUN_SC_REC\\3_Practice_Project\\Course_5_Data Science with Python\\Practi
```

2: Analyze the dataset

In [3]:
```
#View the first five observations of the dataset
df_diabetes_data.head()
```

Out[3]:

	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 [4]:
```
#check the column name is present or not
print(df_diabetes_data.columns)
```

```
Int64Index([0, 1, 2, 3, 4, 5, 6, 7, 8], dtype='int64')
```

In [5]:
```
#Since header/Column names are not available in dataset
#Create a variable 'feature_name' to assign the features of the dataset
feature_names = ['pregnant_count','glucose_con','Blood_Pressure','skin_thickness','insulin','BMI','Diabetes_pec
```

In [6]:
```
#read the csv file once again, but now use the feature_names set earlier and fix it as the column headers of tl
df_diabetes_data = pd.read_csv('D:\\NIPUN_SC_REC\\3_Practice_Project\\Course_5_Data Science with Python\\Practi
```

In [7]:
```
#Verify if the dataset is updated with the new headers
df_diabetes_data.head()
```

Out[7]:

	pregnant_count	glucose_con	Blood_Pressure	skin_thickness	insulin	BMI	Diabetes_pedigree	Age	Class_variable
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

In [8]:
```
#View the number of observations and features of the dataset
df_diabetes_data.shape
```

Out[8]:
```
(768, 9)
```

In [9]:
```
df_diabetes_data.size
```

Out[9]:
```
6912
```

4: Find the response of the dataset

In [10]:
```
#Select required features from the dataset to create the model
new_features = ['pregnant_count','Blood_Pressure','insulin','BMI','Age']
```

In [11]:
```
#Create the feature object with the new_features
x_feature = df_diabetes_data[new_features]
```

In [12]:
```
#Create the reponse object
y_target = df_diabetes_data['Class_variable']
```

In [13]:
```
#View the shape of the feature object
x_feature.shape
```

Out[13]:
```
(768, 5)
```

In [14]:
```
#View the shape of the target object
y_target.shape
```

Out[14]:
```
(768,)
```

5: Use training and testing datasets to train the model

In [15]:
```
#Split the dataset to test and train the model
from sklearn import model_selection
x_test,x_train,y_test,y_train= model_selection.train_test_split(x_feature,y_target,random_state=1)
```

6: Create a model to predict the diabetes outcome

In [16]:
```
# Create a logistic regression model using the training set
from sklearn.linear_model import LogisticRegression
logReg=LogisticRegression()
logReg.fit(x_train,y_train)
```

Out[16]:
```
LogisticRegression()
```

In [17]:
```
#Make predictions using the testing set
y_predict = logReg.predict(x_test)
y_predict
```

Out[17]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0,
       0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0,
       0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0,
       1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0,
       1, 0, 0, 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, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0,
       0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0,
       0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1,
       0, 1, 0, 1], dtype=int64)
```

7: Check the accuracy of the model

In [18]:
```
#Evaluate the accuracy of your model
from sklearn import metrics
```

In [19]:
```
print (metrics.accuracy_score(y_test,y_predict))
```

```
0.6944444444444444
```

In [20]:
```
#Print the first 30 actual responses
print ('actual: ' ,y_test[0:30])
```

```
actual:  118    0
205    0
506    1
587    0
34     0
257    0
216    1
598    1
761    1
160    0
714    0
372    0
14     1
612    1
119    0
82     0
373    0
375    1
353    0
107    0
17     1
323    1
172    0
402    1
366    1
103    0
422    0
301    1
396    0
697    0
Name: Class_variable, dtype: int64
```

In [21]:
```
#Print the first 30 pridicted responses
print ('predicted: ', y_predict[0:30])
```

```
predicted:  [0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0]
```

In [22]:
```
#Print the Mean-Squared-Error(MSE) value
print (metrics.mean_squared_error(y_test,y_predict))
```

```
0.3055555555555556
```

In [23]:
```
#print the Root-Mean_Squared-Error (RMSE) value
print (np.sqrt(metrics.mean_squared_error(y_test,y_predict)))
```

```
0.5527707983925667
```

In [24]:
```
#print the Variance score
print (logReg.score(x_test,y_test))
```

```
0.6944444444444444
```