Assignment 02

October 16, 2020

1 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

[1]:

```
#Import the required libraries
import pandas as pd
```

[9]:

```
#Import the diabetes dataset
df_indian_data = pd.read_csv("pima-indians-diabetes.data", header=None)
```

2: Analyze the dataset

[10]:

```
#View the first five observations of the dataset
df_indian_data.head()
```

[10]:

```
   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

[17]:

```
#Use the .NAMES file to view and set the features of the dataset
# 7. For Each Attribute: (all numeric-valued)
#    1. Number of times pregnant
#    2. Plasma glucose concentration a 2 hours in an oral glucose tolerance test
#    3. Diastolic blood pressure (mm Hg)
#    4. Triceps skin fold thickness (mm)
#    5. 2-Hour serum insulin (mu U/ml)
#    6. Body mass index (weight in kg/(height in m)^2)
#    7. Diabetes pedigree function
#    8. Age (years)
#    9. Class variable (0 or 1)
df_indian_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   0       768 non-null    int64
 1   1       768 non-null    int64
 2   2       768 non-null    int64
 3   3       768 non-null    int64
 4   4       768 non-null    int64
 5   5       768 non-null    float64
 6   6       768 non-null    float64
 7   7       768 non-null    int64
 8   8       768 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

[22]:

```
#Use the feature names set earlier and fix it as the column headers of the␣
 ↪dataset
df_indian_data = pd.DataFrame({
    "TimesPregnant":df_indian_data[0],
    "PlasmaConcentration":df_indian_data[1],
    "BloodPressure":df_indian_data[2],
    "Triceps":df_indian_data[3],
    "Insulin":df_indian_data[4],
    "BodyMass":df_indian_data[5],
    "DiabetesPedigree":df_indian_data[6],
    "Age":df_indian_data[7],
    "Class":df_indian_data[8]
})
```

[23]:

```
#Verify if the dataset is updated with the new headers
df_indian_data.head()
```

[23]:

```
   TimesPregnant  PlasmaConcentration  BloodPressure  Triceps  Insulin  \
0              6                  148             72       35        0
1              1                   85             66       29        0
2              8                  183             64        0        0
```

```
3              1                  89             66        23        94
4              0                 137             40        35       168

   BodyMass  DiabetesPedigree  Age  Class
0      33.6             0.627   50      1
1      26.6             0.351   31      0
2      23.3             0.672   32      1
3      28.1             0.167   21      0
4      43.1             2.288   33      1
```

[24]:
```python
#View the number of observations and features of the dataset
df_indian_data.shape
```

[24]:
```
(768, 9)
```

4: Find the response of the dataset

[26]:
```python
#Select features from the dataset to create the model
feature_select_cols = ['TimesPregnant','Insulin','BodyMass','Age']
```

[27]:
```python
#Create the feature object
X_feature = df_indian_data[feature_select_cols]
X_feature.head()
```

[27]:
```
   TimesPregnant  Insulin  BodyMass  Age
0              6        0      33.6   50
1              1        0      26.6   31
2              8        0      23.3   32
3              1       94      28.1   21
4              0      168      43.1   33
```

[29]:
```python
#Create the reponse object
Y_target = df_indian_data['Class']
```

[31]:
```python
#View the shape of the feature object
X_feature.shape
```

[31]:
```
(768, 4)
```

[12]:
```python
#View the shape of the target object
Y_target.shape
```

5: Use training and testing datasets to train the model

[43]:
```python
#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_feature,Y_target,random_state=1)
```

[44]:

```
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(576, 4)
(192, 4)
(576,)
(192,)
```

6: Create a model to predict the diabetes outcome

[45]:

```
# Create a logistic regression model using the training set
from sklearn.linear_model import LogisticRegression
```

[46]:

```
#Make predictions using the testing set
linreg = LogisticRegression()
linreg.fit(x_train,y_train)
```

[46]:

```
LogisticRegression()
```

[50]:

```
print(linreg.intercept_)
print(linreg.coef_)
```

```
[-5.37141475]
[[0.0850801  0.00210143 0.09515772 0.03309059]]
```

[51]:

```
y_pred = linreg.predict(x_test)
```

7: Check the accuracy of the model

[55]:

```
#Evaluate the accuracy of your model
from sklearn import metrics
import numpy as np
print(np.sqrt(metrics.accuracy_score(y_test,y_pred)))
```

```
0.8322910148099242
```

[62]:

```
#Print the first 30 actual and predicted responses
print('actual:     ', y_test.values[0:30])
print('predicted:  ', y_pred[0:30])
```

```
actual:      [0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1 0 0 0 1 1 1 1 0 0 0 1 0 1]
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]
```

[ ]: