Diabetes

September 22, 2021

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 numpy as np,pandas as pd
```

[2]:
```
#Import the diabetes dataset
data=pd.read_csv('G:/Simplilearn/Data Science with Python/Practice Project/
↪1574412946_lesson8/Lesson 8/1574413358_lesson82/Lesson 8-2/
↪pima-indians-diabetes.data')
```

2: Analyze the dataset

[3]:
```
#View the first five observations of the dataset
data.head(5)
```

[3]:

	6	148	72	35	0	33.6	0.627	50	1
0	1	85	66	29	0	26.6	0.351	31	0
1	8	183	64	0	0	23.3	0.672	32	1
2	1	89	66	23	94	28.1	0.167	21	0
3	0	137	40	35	168	43.1	2.288	33	1
4	5	116	74	0	0	25.6	0.201	30	0

3: Find the features of the dataset

[8]:
```
#Use the .NAMES file to view and set the features of the dataset
features=['Pregnant','glucose','bp mm Hg','skin fold thickness','serum␣
↪insulin','bmi',
          'pedigree function','age','label']
```

[9]:
```
#Use the feature names set earlier and fix it as the column headers of the␣
↪dataset
data.columns=features
```

[10]:
```
#Verify if the dataset is updated with the new headers
data.head(5)
```

[10]:

	Pregnant	glucose	bp mm Hg	skin fold thickness	serum insulin	bmi	\
0	1	85	66	29	0	26.6	
1	8	183	64	0	0	23.3	
2	1	89	66	23	94	28.1	
3	0	137	40	35	168	43.1	
4	5	116	74	0	0	25.6	

	pedigree function	age	label
0	0.351	31	0
1	0.672	32	1
2	0.167	21	0
3	2.288	33	1
4	0.201	30	0

[11]:
```
#View the number of observations and features of the dataset
data.shape
```

[11]: (767, 9)

4: Find the response of the dataset

[20]:
```
#Select features from the dataset to create the model
data.iloc[:,[0,1,2,3,4,5,6,7]]
```

[20]:

	Pregnant	glucose	bp mm Hg	skin fold thickness	serum insulin	bmi	\
0	1	85	66	29	0	26.6	
1	8	183	64	0	0	23.3	
2	1	89	66	23	94	28.1	
3	0	137	40	35	168	43.1	
4	5	116	74	0	0	25.6	
..	…	…	…	…	…	…	
762	10	101	76	48	180	32.9	
763	2	122	70	27	0	36.8	
764	5	121	72	23	112	26.2	
765	1	126	60	0	0	30.1	

```
766         1       93        70                    31              0  30.4

     pedigree function  age
0                0.351   31
1                0.672   32
2                0.167   21
3                2.288   33
4                0.201   30
..                 …   …
762              0.171   63
763              0.340   27
764              0.245   30
765              0.349   47
766              0.315   23

[767 rows x 8 columns]
```

[21]:
```
#Create the feature object
x=data.iloc[:,[0,1,2,3,4,5,6,7]]
```

[25]:
```
#Create the reponse object
y=data.label.values
```

[23]:
```
#View the shape of the feature object
x.shape
```

[23]:
```
(767, 8)
```

[26]:
```
#View the shape of the target object
y.shape
```

[26]:
```
(767,)
```

5: Use training and testing datasets to train the model

[31]:
```
#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=0,test_size=0.
↪25)
```

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
model=LogisticRegression(max_iter=200)
model.fit(x_train,y_train)
```

[45]:
```
LogisticRegression(max_iter=200)
```

[46]:
```
#Make predictions using the testing set
predict=model.predict(x_test)
predict
```

[46]:
```
array([1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1,
       0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0,
       1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0,
       1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0,
       1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], dtype=int64)
```

7: Check the accuracy of the model

[47]:
```
#Evaluate the accuracy of your model
print (model.score(x_train,y_train))
print (model.score(x_test,y_test))
```

```
0.7704347826086957
0.7916666666666666
```

[62]:
```
#Print the first 30 actual and predicted responses
print ('actual:',y_test[0:30])
print ('predicted:',predict[0:30])
```

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

[ ]:
```

```