Assignment_08_02_Completed

July 19, 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

[2]:
```
#Import the required libraries
import numpy as np
import pandas as pd
```

[4]:
```
#Import the diabetes dataset
df_diabetes_data = pd.read_csv("pima-indians-diabetes.data", header= None)
```

2: Analyze the dataset

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

[5]:

	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

[7]:
```
#Use the .NAMES file to view and set the features of the dataset
```

```
df_diabetes_data_features = ['Number of times pregnant','Plasma glucose␣
↪concentration a 2 hours in an oral glucose tolerance test','Diastolic blood␣
↪pressure (mm Hg)','Triceps skin fold thickness (mm)','2-Hour serum insulin␣
↪(mu U/ml)','Body mass index (weight in kg/(height in m)^2)','Diabetes␣
↪pedigree function','Age (years)','Class variable (0 or 1)']
```

[8]:
```
#Use the feature names set earlier and fix it as the column headers of the␣
↪dataset
df_diabetes_data = pd.read_csv("pima-indians-diabetes.data", header= None,␣
↪names = df_diabetes_data_features)
```

[9]:
```
#Verify if the dataset is updated with the new headers
df_diabetes_data.head()
```

[9]:

	Number of times pregnant	Plasma glucose concentration a 2 hours in an oral glucose tolerance test	Diastolic blood pressure (mm Hg)	Triceps skin fold thickness (mm)	2-Hour serum insulin (mu U/ml)	Body mass index (weight in kg/(height in m)^2)	Diabetes pedigree function
0	6	148	72	35	0	33.6	0.627
1	1	85	66	29	0	26.6	0.351
2	8	183	64	0	0	23.3	0.672
3	1	89	66	23	94	28.1	0.167
4	0	137	40	35	168		

```
4                                                43.1                                  2.288
```

```
   Age (years)  Class variable (0 or 1)
0           50                        1
1           31                        0
2           32                        1
3           21                        0
4           33                        1
```

[10]:
```
#View the number of observations and features of the dataset
df_diabetes_data.shape
```

[10]:
```
(768, 9)
```

4: Find the response of the dataset

[31]:
```
#Select features from the dataset to create the model
X_features = df_diabetes_data.loc[:,df_diabetes_data.columns != 'Class variable␣
↪(0 or 1)']
```

[32]:
```
#Create the feature object
X_features
```

[32]:
```
     Number of times pregnant  \
0                           6
1                           1
2                           8
3                           1
4                           0
..                        …
763                        10
764                         2
765                         5
766                         1
767                         1

     Plasma glucose concentration a 2 hours in an oral glucose tolerance test  \
0                                                  148
1                                                   85
2                                                  183
3                                                   89
4                                                  137
..                                                 …
763                                                101
764                                                122
765                                                121
766                                                126
```

```
767                                                  93

     Diastolic blood pressure (mm Hg)  Triceps skin fold thickness (mm)  \
0                                  72                                  35
1                                  66                                  29
2                                  64                                   0
3                                  66                                  23
4                                  40                                  35
..                                 …                                   …
763                                76                                  48
764                                70                                  27
765                                72                                  23
766                                60                                   0
767                                70                                  31

     2-Hour serum insulin (mu U/ml)  \
0                                 0
1                                 0
2                                 0
3                                94
4                               168
..                              …
763                             180
764                               0
765                             112
766                               0
767                               0

     Body mass index (weight in kg/(height in m)^2)  \
0                                              33.6
1                                              26.6
2                                              23.3
3                                              28.1
4                                              43.1
..                                              …
763                                            32.9
764                                            36.8
765                                            26.2
766                                            30.1
767                                            30.4

     Diabetes pedigree function  Age (years)
0                         0.627           50
1                         0.351           31
2                         0.672           32
3                         0.167           21
4                         2.288           33
```

```
..                          …          …
763                         0.171         63
764                         0.340         27
765                         0.245         30
766                         0.349         47
767                         0.315         23

[768 rows x 8 columns]
```

[33]:
```
#Create the reponse object
Y_target = df_diabetes_data['Class variable (0 or 1)']
```

[34]:
```
#View the shape of the feature object
X_features.shape
```

[34]: (768, 8)

[36]:
```
#View the shape of the target object
Y_target.shape
```

[36]: (768,)

5: Use training and testing datasets to train the model

[38]:
```
#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_features, Y_target,␣
 ↪random_state = 1)
```

6: Create a model to predict the diabetes outcome

[40]:
```
# Create a logistic regression model using the training set
from sklearn.linear_model import LogisticRegression
lgreg = LogisticRegression()
lgreg.fit(X_train, Y_train)
```

```
/usr/local/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:940:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```

[40]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

[41]:
```
#Make predictions using the testing set
Y_pred = lgreg.predict(X_test)
```

7: Check the accuracy of the model

[44]:
```
#Evaluate the accuracy of your model
from sklearn import metrics
metrics.accuracy_score(Y_test, Y_pred)
```

[44]:
```
0.7760416666666666
```

[53]:
```
#Print the first 30 actual and predicted responses
print('actual, predicated :' + str(list(zip(Y_test.values[0:30], Y_pred[0:
 ↪30]))))
```

```
actual, predicated :[(0, 0), (0, 0), (0, 0), (0, 0), (0, 0), (0, 0), (0, 0), (0,
0), (0, 0), (0, 0), (0, 0), (0, 0), (1, 1), (1, 1), (0, 0), (1, 1), (1, 0), (0,
0), (0, 0), (0, 0), (1, 0), (1, 0), (1, 1), (1, 0), (0, 0), (0, 0), (0, 0), (1,
1), (0, 0), (1, 0)]
```