5_LogisticRegression

August 27, 2024

[80]:
```
import pandas as pd
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.model_selection import cross_val_score

# Creating a (non-medical) tool used for predicting if a patient has a␣
 ↪vertebral abnormality (either disk hernia or spondylolisthesis)

# Read data (replace with your own path)
df = pd.read_csv('column_2C.dat', delim_whitespace=True, header=None)
```

[82]:
```
# Add the labels to the columns of the data
labels = ['pelvic incidence', 'pelvic tilt', 'lumbar lordosis angle', 'sacral␣
 ↪slope', 'pelvic radius', 'grade of spondylolisthesis', 'result']
df.columns = labels
df.head(10)
```

[82]:

	pelvic incidence	pelvic tilt	lumbar lordosis angle	sacral slope	pelvic radius	grade of spondylolisthesis	result
0	63.03	22.55	39.61	40.48	98.67	-0.25	AB
1	39.06	10.06	25.02	29.00	114.41	4.56	AB
2	68.83	22.22	50.09	46.61	105.99	-3.53	AB
3	69.30	24.65	44.31	44.64	101.87	11.21	AB
4	49.71	9.65	28.32	40.06			
5	40.25	13.92	25.12	26.33			
6	53.43	15.86	37.17	37.57			
7	45.37	10.76	29.04	34.61			
8	43.79	13.53	42.69	30.26			
9	36.69	5.01	41.95	31.68			

```
4          108.17                      7.92     AB
5          130.33                      2.23     AB
6          120.57                      5.99     AB
7          117.27                    -10.68     AB
8          125.00                     13.29     AB
9           84.24                      0.66     AB
```

[84]:
```
# Re-encode result column using value 0 for healthy person and 1 abnormal
df['result'].replace(['NO','AB'], [0,1], inplace=True)
df.head(10)

# Confirm datatypes
print(df.dtypes)
```

```
pelvic incidence             float64
pelvic tilt                  float64
lumbar lordosis angle        float64
sacral slope                 float64
pelvic radius                float64
grade of spondylolisthesis   float64
result                         int64
dtype: object
```

[86]:
```
# Print the basic statistics
df.describe()
```

[86]:

	pelvic incidence	pelvic tilt	lumbar lordosis angle	sacral slope	\
count	310.000000	310.000000	310.000000	310.000000	
mean	60.496484	17.542903	51.930710	42.953871	
std	17.236109	10.008140	18.553766	13.422748	
min	26.150000	-6.550000	14.000000	13.370000	
25%	46.432500	10.667500	37.000000	33.347500	
50%	58.690000	16.360000	49.565000	42.405000	
75%	72.880000	22.120000	63.000000	52.692500	
max	129.830000	49.430000	125.740000	121.430000	

	pelvic radius	grade of spondylolisthesis	result
count	310.000000	310.000000	310.000000
mean	117.920548	26.296742	0.677419
std	13.317629	37.558883	0.468220
min	70.080000	-11.060000	0.000000
25%	110.710000	1.600000	0.000000
50%	118.265000	11.765000	1.000000
75%	125.467500	41.285000	1.000000
max	163.070000	418.540000	1.000000

[88]:
```
# Stardardize the X variables
X = df.iloc[:,:6]
```

```
scaler = StandardScaler(with_mean=True, with_std=True)
X_scaled = pd.DataFrame(scaler.fit_transform(X), columns = list(X.columns.
 ↪values))
X_scaled.head(10)
```

[88]:

	pelvic incidence	pelvic tilt	lumbar lordosis angle	sacral slope	pelvic radius	grade of spondylolisthesis
0	0.147227	0.501111	-0.665128	-0.184602	-1.447831	-0.707946
1	-1.245707	-0.748891	-1.452763	-1.041250	-0.264028	-0.579673
2	0.484273	0.468085	-0.099370	0.272823	-0.897295	-0.795417
3	0.511586	0.711280	-0.411401	0.125820	-1.207159	-0.402332
4	-0.626819	-0.789923	-1.274614	-0.215943	-0.733337	-0.490069
5	-1.176554	-0.362581	-1.447364	-1.240487	0.933313	-0.641810
6	-0.410644	-0.168425	-0.796850	-0.401749	0.199265	-0.541538
7	-0.879023	-0.678834	-1.235745	-0.622627	-0.048928	-0.986092
8	-0.970839	-0.401612	-0.498856	-0.947227	0.532444	-0.346863
9	-1.383431	-1.254296	-0.538804	-0.841266	-2.533109	-0.683678

[90]:
```
# Create a response variable
y = df.iloc[:,6]

# Build and fit model
reg = LogisticRegression(solver='lbfgs')
reg.fit(X_scaled,y)

print("Coefficients: ",reg.coef_)
print("Intercept: ", reg.intercept_)
```

```
Coefficients:  [[-0.185  0.688 -0.154 -0.75  -1.133  4.067]]
Intercept:  [2.167]
```

[54]:
```
# Print the scaled X values
print(X_scaled)

# Compute predicted values from the training set
y_pred = reg.predict(X_scaled)
```

```
# Create a confusion matrix
cm = confusion_matrix(y, y_pred)
print("Confusion matrix:\n",cm)
```

```
     pelvic incidence  pelvic tilt  lumbar lordosis angle  sacral slope  \
0            0.147227     0.501111              -0.665128     -0.184602
1           -1.245707    -0.748891              -1.452763     -1.041250
2            0.484273     0.468085              -0.099370      0.272823
3            0.511586     0.711280              -0.411401      0.125820
4           -0.626819    -0.789923              -1.274614     -0.215943
..                ...          ...                    ...           ...
305         -0.732001    -0.392605              -0.860012     -0.646505
306         -0.381007     0.317965              -1.226028     -0.726350
307          0.055410     0.515123              -0.310989     -0.313696
308         -0.885997    -0.886000              -0.558778     -0.477116
309         -1.549049    -1.248291              -0.825462     -1.058412

     pelvic radius  grade of spondylolisthesis
0        -1.447831                   -0.707946
1        -0.264028                   -0.579673
2        -0.897295                   -0.795417
3        -1.207159                   -0.402332
4        -0.733337                   -0.490069
..             ...                         ...
305      -0.035390                   -0.814618
306      -0.267036                   -0.712480
307       0.582835                   -0.773549
308       0.047341                   -0.695679
309       0.453474                   -0.706613

[310 rows x 6 columns]
Confusion matrix:
 [[ 80  20]
 [ 24 186]]
```

[92]:
```
# Calculate the accuracy of the model
accuracy = (cm[0][0]+cm[1][1])/(cm[0][0]+cm[1][1]+cm[0][1]+cm[1][0])
print("Accuracy calculated from the training set = %.3f" % (accuracy))

# Print classification report
print(classification_report(y, y_pred, target_names=['normal', 'abnormal']))
```

```
Accuracy calculated from the training set = 0.858
              precision    recall  f1-score   support

      normal       0.77      0.80      0.78       100
    abnormal       0.90      0.89      0.89       210
```

```
    accuracy                           0.86       310
   macro avg       0.84      0.84      0.84       310
weighted avg       0.86      0.86      0.86       310
```

[94]:
```
# Perform a cross-validation

k = 10 # number of folds
scores = cross_val_score(estimator=reg,
                         X=X_scaled,
                         y=y,
                         scoring="accuracy",
                         cv=k)
print("Accuracies from %d individual folds:" % k)
print(scores)
print("Accuracy calculated using %d-fold cross validation = %.3f" % (k, scores.
 ↪mean()))
```

```
Accuracies from 10 individual folds:
[0.581 0.613 0.806 0.774 0.903 0.935 0.935 0.903 0.806 0.935]
Accuracy calculated using 10-fold cross validation = 0.819
```

[96]:
```
# Form the prediction
predicted_probabilities = reg.predict_proba(X_scaled)

# Create a prettier print
np.set_printoptions(precision=3, suppress=True)

# Display the predicted probabilities
# the first column represents the probability of the sample belonging to the␣
 ↪normal class
# the second column represents the probability of the sample belonging to the␣
 ↪abnormal class
print(predicted_probabilities)
```

```
[[0.184 0.816]
 [0.304 0.696]
 [0.502 0.498]
 [0.094 0.906]
 [0.282 0.718]
 [0.593 0.407]
 [0.469 0.531]
 [0.808 0.192]
 [0.301 0.699]
 [0.086 0.914]
 [0.602 0.398]
 [0.188 0.812]
```

```
[0.225 0.775]
[0.161 0.839]
[0.246 0.754]
[0.51  0.49 ]
[0.542 0.458]
[0.716 0.284]
[0.465 0.535]
[0.483 0.517]
[0.643 0.357]
[0.537 0.463]
[0.064 0.936]
[0.648 0.352]
[0.164 0.836]
[0.292 0.708]
[0.654 0.346]
[0.112 0.888]
[0.118 0.882]
[0.503 0.497]
[0.192 0.808]
[0.126 0.874]
[0.242 0.758]
[0.5   0.5  ]
[0.22  0.78 ]
[0.251 0.749]
[0.622 0.378]
[0.558 0.442]
[0.285 0.715]
[0.385 0.615]
[0.247 0.753]
[0.567 0.433]
[0.404 0.596]
[0.576 0.424]
[0.634 0.366]
[0.39  0.61 ]
[0.169 0.831]
[0.364 0.636]
[0.398 0.602]
[0.257 0.743]
[0.324 0.676]
[0.049 0.951]
[0.198 0.802]
[0.078 0.922]
[0.544 0.456]
[0.107 0.893]
[0.387 0.613]
[0.404 0.596]
[0.508 0.492]
[0.428 0.572]
```

```
[0.02  0.98 ]
[0.    1.   ]
[0.251 0.749]
[0.151 0.849]
[0.694 0.306]
[0.023 0.977]
[0.176 0.824]
[0.    1.   ]
[0.7   0.3  ]
[0.165 0.835]
[0.134 0.866]
[0.    1.   ]
[0.001 0.999]
[0.033 0.967]
[0.033 0.967]
[0.    1.   ]
[0.    1.   ]
[0.004 0.996]
[0.05  0.95 ]
[0.121 0.879]
[0.    1.   ]
[0.001 0.999]
[0.015 0.985]
[0.057 0.943]
[0.002 0.998]
[0.902 0.098]
[0.165 0.835]
[0.075 0.925]
[0.202 0.798]
[0.095 0.905]
[0.055 0.945]
[0.074 0.926]
[0.007 0.993]
[0.009 0.991]
[0.063 0.937]
[0.    1.   ]
[0.028 0.972]
[0.061 0.939]
[0.01  0.99 ]
[0.07  0.93 ]
[0.059 0.941]
[0.074 0.926]
[0.103 0.897]
[0.12  0.88 ]
[0.    1.   ]
[0.24  0.76 ]
[0.005 0.995]
[0.001 0.999]
```

```
[0.053 0.947]
[0.101 0.899]
[0.034 0.966]
[0.003 0.997]
[0.287 0.713]
[0.002 0.998]
[0.001 0.999]
[0.    1.   ]
[0.012 0.988]
[0.001 0.999]
[0.486 0.514]
[0.047 0.953]
[0.029 0.971]
[0.002 0.998]
[0.    1.   ]
[0.009 0.991]
[0.045 0.955]
[0.016 0.984]
[0.033 0.967]
[0.015 0.985]
[0.002 0.998]
[0.045 0.955]
[0.027 0.973]
[0.492 0.508]
[0.006 0.994]
[0.019 0.981]
[0.019 0.981]
[0.    1.   ]
[0.002 0.998]
[0.003 0.997]
[0.002 0.998]
[0.065 0.935]
[0.071 0.929]
[0.    1.   ]
[0.013 0.987]
[0.    1.   ]
[0.036 0.964]
[0.002 0.998]
[0.001 0.999]
[0.248 0.752]
[0.031 0.969]
[0.01  0.99 ]
[0.009 0.991]
[0.173 0.827]
[0.064 0.936]
[0.074 0.926]
[0.031 0.969]
[0.003 0.997]
```

```
[0.003 0.997]
[0.02  0.98 ]
[0.013 0.987]
[0.011 0.989]
[0.003 0.997]
[0.021 0.979]
[0.    1.   ]
[0.    1.   ]
[0.276 0.724]
[0.008 0.992]
[0.055 0.945]
[0.024 0.976]
[0.002 0.998]
[0.061 0.939]
[0.178 0.822]
[0.52  0.48 ]
[0.03  0.97 ]
[0.009 0.991]
[0.019 0.981]
[0.004 0.996]
[0.033 0.967]
[0.005 0.995]
[0.011 0.989]
[0.    1.   ]
[0.583 0.417]
[0.012 0.988]
[0.001 0.999]
[0.001 0.999]
[0.179 0.821]
[0.026 0.974]
[0.003 0.997]
[0.146 0.854]
[0.002 0.998]
[0.002 0.998]
[0.002 0.998]
[0.    1.   ]
[0.    1.   ]
[0.27  0.73 ]
[0.201 0.799]
[0.175 0.825]
[0.001 0.999]
[0.    1.   ]
[0.162 0.838]
[0.185 0.815]
[0.094 0.906]
[0.    1.   ]
[0.    1.   ]
[0.127 0.873]
```

```
[0.032 0.968]
[0.    1.   ]
[0.    1.   ]
[0.004 0.996]
[0.004 0.996]
[0.006 0.994]
[0.291 0.709]
[0.643 0.357]
[0.743 0.257]
[0.709 0.291]
[0.869 0.131]
[0.801 0.199]
[0.796 0.204]
[0.745 0.255]
[0.734 0.266]
[0.182 0.818]
[0.197 0.803]
[0.932 0.068]
[0.077 0.923]
[0.534 0.466]
[0.626 0.374]
[0.896 0.104]
[0.93  0.07 ]
[0.51  0.49 ]
[0.551 0.449]
[0.912 0.088]
[0.8   0.2  ]
[0.546 0.454]
[0.666 0.334]
[0.91  0.09 ]
[0.534 0.466]
[0.786 0.214]
[0.825 0.175]
[0.887 0.113]
[0.799 0.201]
[0.893 0.107]
[0.715 0.285]
[0.422 0.578]
[0.857 0.143]
[0.75  0.25 ]
[0.187 0.813]
[0.801 0.199]
[0.272 0.728]
[0.492 0.508]
[0.698 0.302]
[0.426 0.574]
[0.951 0.049]
[0.955 0.045]
```

```
[0.959 0.041]
[0.341 0.659]
[0.664 0.336]
[0.317 0.683]
[0.686 0.314]
[0.99  0.01 ]
[0.721 0.279]
[0.714 0.286]
[0.775 0.225]
[0.685 0.315]
[0.625 0.375]
[0.817 0.183]
[0.823 0.177]
[0.843 0.157]
[0.715 0.285]
[0.8   0.2  ]
[0.407 0.593]
[0.237 0.763]
[0.896 0.104]
[0.25  0.75 ]
[0.562 0.438]
[0.877 0.123]
[0.638 0.362]
[0.915 0.085]
[0.794 0.206]
[0.937 0.063]
[0.91  0.09 ]
[0.627 0.373]
[0.532 0.468]
[0.596 0.404]
[0.504 0.496]
[0.756 0.244]
[0.701 0.299]
[0.7   0.3  ]
[0.668 0.332]
[0.502 0.498]
[0.685 0.315]
[0.808 0.192]
[0.487 0.513]
[0.357 0.643]
[0.814 0.186]
[0.714 0.286]
[0.876 0.124]
[0.867 0.133]
[0.744 0.256]
[0.26  0.74 ]
[0.781 0.219]
[0.358 0.642]
```

```
[0.765 0.235]
[0.613 0.387]
[0.452 0.548]
[0.877 0.123]
[0.977 0.023]
[0.651 0.349]
[0.355 0.645]
[0.733 0.267]
[0.672 0.328]
[0.705 0.295]]
```

[102]:
```
# A program for calculating the chance of having a vertebral abnormality
choice = -1
notQuit = True
while(notQuit):
    choice = int(input("1: Enter dimensions 2: Exit "))
    if choice == 1:
        pelvicIncidence = float(input("Give pelvic incidence "))
        pelvicTilt = float(input("Give pelvic tilt "))
        lumbar = float(input("Give lumbar lordosis angle "))
        sacralSlope = float(input("Give sacral slope "))
        pelvicRadius = float(input("Give pelvic radius "))
        spondylolisthesis = float(input("Give grade of spondylolisthesis "))

        user_input = pd.DataFrame({'pelvic incidence': [pelvicIncidence],
                                   'pelvic tilt': [pelvicTilt],
                                   'lumbar lordosis angle': [lumbar],
                                   'sacral slope': [sacralSlope],
                                   'pelvic radius': [pelvicRadius],
                                   'grade of spondylolisthesis': [spondylolisthesis]})

        user_input_scaled = pd.DataFrame(scaler.transform(user_input),␣
 ↪columns=user_input.columns)
        result = reg.predict_proba(user_input_scaled)

        # Printing the result
        print(f"The chance that the patient has a vertebral abnormality is␣
 ↪{round(result[0][1]*100, 2)}%\n")

    elif choice == 2:
        notQuit = False
        print("Bye")
    else:
         print("Give a valid number please")
```

```
1: Enter dimensions 2: Exit  1
Give pelvic incidence  63.03
```

```
Give pelvic tilt  22.55
Give lumbar lordosis angle  39.61
Give sacral slope  40.48
Give pelvic radius  98.67
Give grade of spondylolisthesis  -0.25

The chance that the patient has a vertebral abnormality is 81.55%


1: Enter dimensions 2: Exit  1
Give pelvic incidence  33.84
Give pelvic tilt  5.07
Give lumbar lordosis angle  36.64
Give sacral slope  28.77
Give pelvic radius  123.95
Give grade of spondylolisthesis  -0.2

The chance that the patient has a vertebral abnormality is 29.51%


1: Enter dimensions 2: Exit  2

Bye
```