

PCA Logistic Regression

Yuhuan Huang

Read Dataset:

```
print('Dataset label split:\n', np.array(np.unique(breast_cancer.target, return_counts=True)))
```

[1]

```
... Number of dataset features: 30
Dataset feature names:
['mean radius' 'mean texture' 'mean perimeter' 'mean area'
 'mean smoothness' 'mean compactness' 'mean concavity'
 'mean concave points' 'mean symmetry' 'mean fractal dimension'
 'radius error' 'texture error' 'perimeter error' 'area error'
 'smoothness error' 'compactness error' 'concavity error'
 'concave points error' 'symmetry error' 'fractal dimension error'
 'worst radius' 'worst texture' 'worst perimeter' 'worst area'
 'worst smoothness' 'worst compactness' 'worst concavity'
 'worst concave points' 'worst symmetry' 'worst fractal dimension']
Dataset data:
[[1.799e+01 1.038e+01 1.228e+02 ... 2.654e-01 4.601e-01 1.189e-01]
 [2.057e+01 1.777e+01 1.329e+02 ... 1.860e-01 2.750e-01 8.902e-02]
 [1.969e+01 2.125e+01 1.300e+02 ... 2.430e-01 3.613e-01 8.758e-02]
 ...
 [1.660e+01 2.808e+01 1.083e+02 ... 1.418e-01 2.218e-01 7.820e-02]
 [2.060e+01 2.933e+01 1.401e+02 ... 2.650e-01 4.087e-01 1.240e-01]
 [7.760e+00 2.454e+01 4.792e+01 ... 0.000e+00 2.871e-01 7.039e-02]]
Dataset label names:
['malignant' 'benign']
Dataset labels:
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0
 1 0 0 0 0 0 0 0 1 0 1 1 1 1 1 0 0 1 0 0 1 1 1 1 0 1 0 0 1 1 1 0 1 0 0
 1 0 1 0 0 1 1 1 0 0 1 0 0 0 1 1 1 0 1 1 0 0 1 1 1 0 0 1 1 1 0 1 1
 ...
 1 1 1 1 1 1 1 0 0 0 0 0 1]
Dataset label split:
[[ 0  1]
 [212 357]]
```

Output is truncated. View as a [scrollable element](#) or open in a [text editor](#). Adjust cell output [settings](#)...

Step 1

```
#For verification purposes – do not change code below this line.
import doctest

"""
    >>> print(df['mean concave points'].iloc[0])
    0.1471
    >>> print(df['worst fractal dimension'].iloc[150])
    0.06435
    >>> print(df['diagnosis'].iloc[200])
    1
    >>> print(df['mean symmetry'].iloc[568])
    0.1587
"""

doctest.testmod()
```

[2]

... TestResults(failed=0, attempted=4)

Step 2

Notes: I changed all True in the tests to np.True_.

```
import doctest

"""
    >>> np.isclose(log_reg_score, 0.9239766081871345, atol=1e-03)
    np.True_
"""

doctest.testmod()
```

[3] ✓ 0.1s

.. Logistic regression model score on UNSCALED dataset (all features): 0.9239766081871345

.. TestResults(failed=0, attempted=1)

Step 3

```
"""
    >>> np.isclose(log_reg_scaled_score, 0.9824561403508771, atol=1e-03)
    np.True_
"""

doctest.testmod()
```

[5] ✓ 0.0s

.. Logistic regression model score on SCALED dataset (all features): 0.9824561403508771

.. TestResults(failed=0, attempted=1)

Step 4

```
#components = None
pca = PCA(n_components=None)

# Perform PCA on the SCALED data, keeping all components
pca.fit(X_scaled)
print('Explained variance:', pca.explained_variance_ratio_)

#For verification purposes – do not change code below this line.
import doctest


"""
>>> np.isclose(pca.explained_variance_ratio_[0], 0.44272025607526366, atol=1e-03)
np.True_
>>> np.isclose(pca.explained_variance_ratio_[14], 0.0031378321676274047, atol=1e-03)
np.True_
>>> np.isclose(pca.explained_variance_ratio_[29], 4.434827427363563e-06, atol=1e-03)
np.True_
"""

doctest.testmod()
```

[6] ✓ 0.0s

```
... Explained variance: [4.42720256e-01 1.89711820e-01 9.39316326e-02 6.60213492e-02
5.49576849e-02 4.02452204e-02 2.25073371e-02 1.58872380e-02
1.38964937e-02 1.16897819e-02 9.79718988e-03 8.70537901e-03
8.04524987e-03 5.23365745e-03 3.13783217e-03 2.66209337e-03
1.97996793e-03 1.75395945e-03 1.64925306e-03 1.03864675e-03
9.99096464e-04 9.14646751e-04 8.11361259e-04 6.01833567e-04
5.16042379e-04 2.72587995e-04 2.30015463e-04 5.29779290e-05
2.49601032e-05 4.43482743e-06]
```

```
... TestResults(failed=0, attempted=3)
```

 Rest

Step 5

```

"""
>>> np.isclose(pca_explained_var_ratio_percentage[0], 4.42720256e+01, atol=1e-03)
np.True_
>>> np.isclose(pca_explained_var_ratio_percentage[29], 4.43482743e-04, atol=1e-03)
np.True_
>>> np.isclose(pca_cumulative_var_ratio_percentage[2], 72.63637091, atol=1e-03)
np.True_
>>> np.isclose(pca_cumulative_var_ratio_percentage[25], 99.96876117, atol=1e-03)
np.True_
"""

```

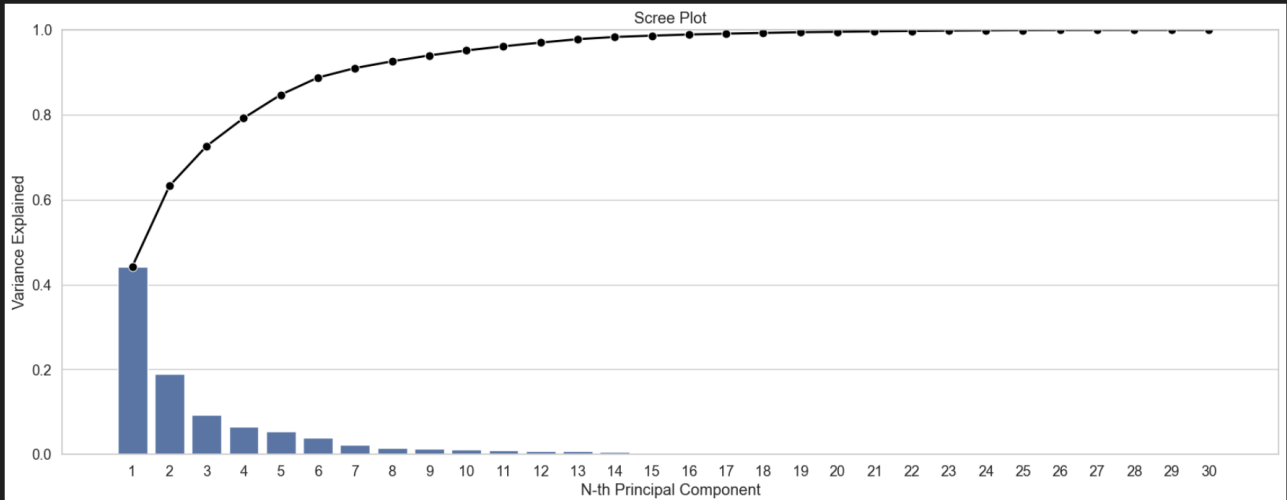
```
doctest.testmod()
```

✓ 0.2s

```

Variances (percentage): [4.42720256e+01 1.89711820e+01 9.39316326e+00 6.60213492e+00
5.49576849e+00 4.02452204e+00 2.25073371e+00 1.58872380e+00
1.38964937e+00 1.16897819e+00 9.79718988e-01 8.70537901e-01
8.04524987e-01 5.23365745e-01 3.13783217e-01 2.66209337e-01
1.97996793e-01 1.75395945e-01 1.64925306e-01 1.03864675e-01
9.99096464e-02 9.14646751e-02 8.11361259e-02 6.01833567e-02
5.16042379e-02 2.72587995e-02 2.30015463e-02 5.29779290e-03
2.49601032e-03 4.43482743e-04]
Cumulative Variances (percentage): [ 44.27202561  63.24320765  72.63637091  79.23850582  84.73427432
 88.75879636  91.00953007  92.59825387  93.98790324  95.15688143
 96.13660042  97.00713832  97.81166331  98.33502905  98.64881227
 98.91502161  99.1130184  99.28841435  99.45333965  99.55720433
 99.65711397  99.74857865  99.82971477  99.88989813  99.94150237
 99.96876117  99.99176271  99.99706051  99.99955652 100.         ]

```



... TestResults(failed=0, attempted=4)

Step 6

```

    """
    >>> print(int(pca_num_components[0]))
    3
    >>> print(int(pca_num_components[1]))
    5
    >>> print(int(pca_num_components[2]))
    7
    """

    doctest.testmod()
[8] ✓ 0.0s
... Number of PCs retained for each explained variance: [np.int64(3), np.int64(5), np.int64(7)]
... TestResults(failed=0, attempted=3)

```

Step 7

```

import doctest

    """
    >>> print(pca_num_components)
    10
    >>> np.isclose(log_reg_pca_score, 0.9766081871345029, atol=1e-03)
    np.True_

    """

    doctest.testmod()
✓ 0.0s

Number of retained PCs: 10
(569, 10)
[[ 8.89142226  2.85709736 -1.59545501 ...  0.29344042  0.16900763
 -1.11056824]
 [ 2.64367714 -3.42934081 -0.77475479 ... -0.29833978  1.01769231
  0.44044061]
 [ 5.75141919 -0.49087436 -0.60506353 ... -0.11451378  0.05409256
  0.42144914]
 ...
 [ 1.40775479 -1.78218653  1.10871546 ... -0.59568962 -0.39216929
  0.73385966]
 [10.23772742  2.5086675  -0.97804441 ...  0.64681721  0.55956386
  0.12672673]
 [-5.48431229 -1.2085518  1.64247356 ... -1.38796123  0.22480948
 -0.47674518]]
Logistic regression model score on transformed dataset: 0.9766081871345029

TestResults(failed=0, attempted=2)

```

My Reflection:

Review your notebook results and answer the following reflection questions:

1. In the notebook, you should have observed that 3 principal components could explain approximately 70% of the variance in the breast cancer dataset which originally had 30 features. Explain why this dimensionality reduction is significant from both a computational and interpretability perspective. How might this impact your approach to similar high-dimensional medical datasets?

A: Reducing the dimension from 30 to 3 would greatly reduce the computational loads. Also, it would be easier for people to interpret, as we only need to focus on the top 3 most significant features. Next time if I see similar dataset, I may also throw-away some less-irrelevant features while preserving the most influential ones.

2. The notebook should have demonstrated that using PCA to retain about 95% of the variance (10 principal components) resulted in a classification accuracy of over 97%, compared to about 98% with all 30 original features. Discuss this trade-off between dimensionality reduction and model performance. What factors would you consider when deciding whether this reduction is worthwhile in a real-world medical diagnosis application?

A: By reducing dimensions, the model becomes more efficient, but more accuracy would be lost. In the real-world problem, I need to know about the threshold that allows the loss in accuracy. Also, I may also need to know how much computing resources I have if it is a massive problem.

3. When examining the results from Steps 5 and 6, you should have observed diminishing returns in the explained variance as more principal components were added. The first principal component should have explained over 44% of the variance, while later components contributed much less. Explain what this pattern tells you about the underlying structure of the breast cancer dataset, and how you might use this information when building predictive models for similar biomedical applications.

A: This tells me that the first component really matters in the breast cancer prediction problem. And only including the first three components would result in a relatively trustworthy prediction. When building predictive models, I should focus more on the top components, and may ignore some of the less-relevant components.