主要路线:利用 pipe 方法将 PCA 和逻辑斯蒂方法串联起来

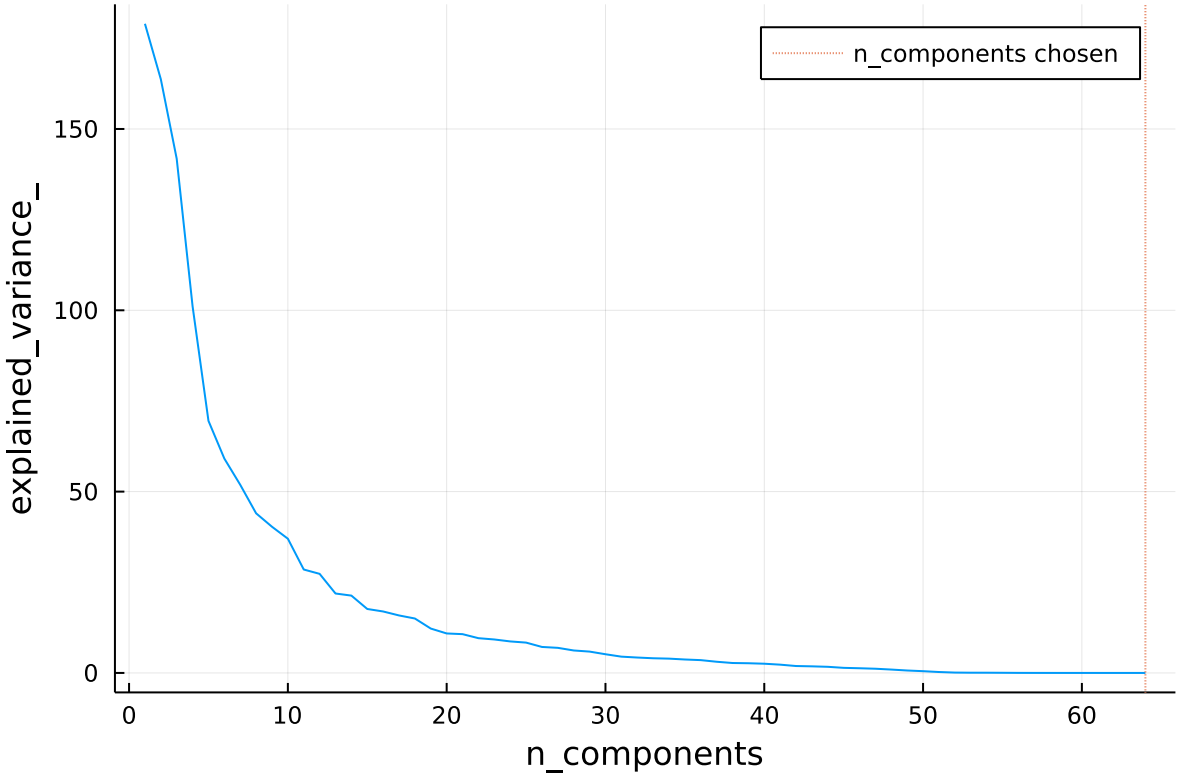
1. 通过 PCA 将高维数据映射为低维度的主成分变量
2. 利用 PCA 主成分进行逻辑斯蒂分类

ScikitLearn.jl/Pipeline_PCA_Logistic.ipynb at master · cstjean/ScikitLearn.jl

```
PyObject <class 'sklearn.linear_model._logistic.LogisticRegression'>
```

```
begin
    using ScikitLearn   , StatsPlots   ,Images   ,StatsBase
    using ScikitLearn.GridSearch: GridSearchCV
    using ScikitLearn.Pipelines: Pipeline, named_steps

    @sk_import decomposition: PCA
    @sk_import datasets: load_digits
    @sk_import linear_model: LogisticRegression
end
```




```
begin
    logistic = LogisticRegression(max_iter=200)

    pca = PCA()
    pipe = Pipeline([("pca", pca), ("logistic", logistic)])

    digits = load_digits() #高维数据
    X_digits = digits["data"]
    y_digits = digits["target"]

    ##############################################################################
    # Plot the PCA spectrum
    ScikitLearn.fit!(pca, X_digits)


    plot(pca.explained_variance_,xlabel="n_components",ylabel="explained_variance_",
    label=false)



    ##############################################################################
    #Prediction

    n_components = [20, 40, 64]
    Cs = 10 .^ range(-4; stop=4, length=3)

    #Parameters of pipelines can be set using '__' separated parameter names:
    estimator = GridSearchCV(pipe,
                             Dict("pca__n_components"=>n_components,
                                  "logistic__C"=>Cs))
    res=ScikitLearn.fit!(estimator, X_digits, y_digits)

    vline!([named_steps(estimator.best_estimator_)["pca"].n_components],
            ls=:dot, label="n_components chosen")
end
```

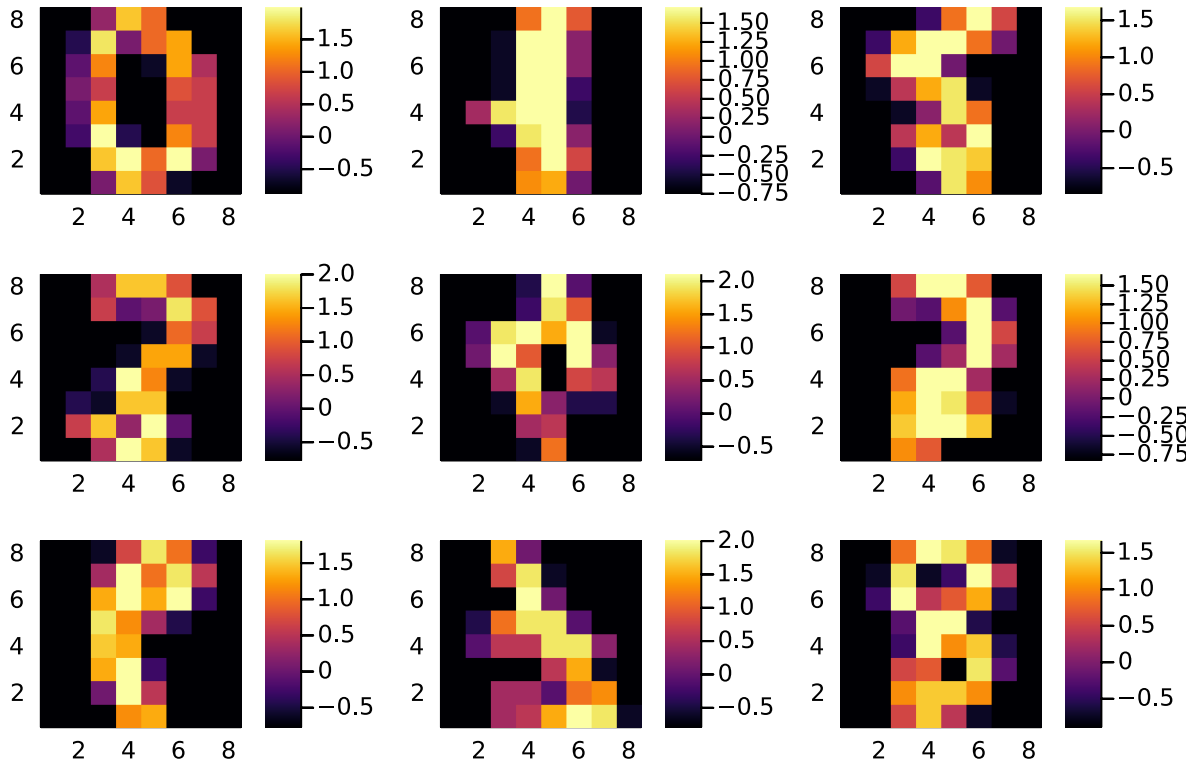
```
score =
  [CVScoreTuple(Dict(:pca__n_components ⇒ 20, :logistic__C ⇒ 0.0001), 0.196439, [0.22757
```

```
score=estimator.grid_scores_
```

```
#[[x.parameters[:pca__n_components],x.mean_validation_score ] for x in score]
```

```
9×64 Matrix{Float64}:
 -0.879315  -0.879315   0.0777625   1.60909   …  -0.879315    -0.879315  -0.879315
 -0.750085  -0.750085  -0.750085    1.09038       0.783635    -0.750085  -0.750085
 -0.846676  -0.846676  -0.846676   -0.216592      1.67366      0.571014  -0.846676
 -0.772143  -0.772143   0.523438    2.0041        0.893604    -0.772143  -0.772143
 -0.709482  -0.709482  -0.709482   -0.533486     -0.00549986  -0.709482  -0.709482
 -0.827624  -0.827624   1.0309      0.721146  …   0.721146    -0.827624  -0.827624
 -0.77034   -0.77034   -0.77034     1.16306       1.00195     -0.286989  -0.77034
 -0.791279  -0.791279   0.431111    0.605738     -0.791279    -0.791279  -0.791279
 -0.889851  -0.889851   0.545875    1.3435        0.864926    -0.730326  -0.889851
```

```
begin
c=digits["data"][1:9,:]
dt = fit(ZScoreTransform, c, dims=2)
#numbers=StatsBase.transform(dt, c)|>d->reshape(d,:,8)
numbers=StatsBase.transform(dt, c)
end
```



```
begin
    function plot_number(d)
        heatmap(1:8,1:8,d')
    end
    res2=[plot_number(numbers[d,:]) for d in 1:9]
    plot()
    plot!(res2...)

end
```