DS-630 Machine Learning

Project 2: Principal Component Analysis - Logistic Regression and Support Vector Machines

Madhumita Duvvuri

Sravan Kumar B

Problem: Using both Logistic Regression and Support Vector Machines understand the given dataset on heart diseases and based on the risk factors predict if a given patient will have a heart disease or not. Pre-process the input variables using PCA and the number of principal components must account for at least 98% of the variance of the original input variables.

Solution:

Process: ml_prj2_pca_rand_datasets.R

We have categorical variables in our heart dataset i.e. for risk factor Chest Pain we have four levels, for Thal we have three levels and independent variable AHD (angiographic heart disease) with two levels. To perform PCA analysis on the input variables we need to convert these levels to numerical data by assigning them with numbers for each level.

1. Removing NA using na.omit() from our dataset.
2. Assigning values 1 and 0 to Yes and No for our response variable AHD.
3. Assigning values 1, 2, 3 and 4 to levels - nonanginal , nontypical , asymptomatic and typical respectively for risk factor Chest Pain.
4. Assigning values 1, 2 and 3 to levels - fixed, normal and reversible respectively for risk factor Thal.
5. Building a dataset which retains only the numerical values - data_heart1.

ml_prj2_pca_rand_datasets.R | data_heart1 | heart_data1

Filter

	Age	Sex	chestpain1	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	Slope	Ca	thal1	ahd_value1
1	63	1	4	145	233	1	2	150	0	2.3	3	0	1	0
2	67	1	3	160	286	0	2	108	1	1.5	2	3	2	1
3	67	1	3	120	229	0	2	129	1	2.6	2	2	3	1
4	37	1	1	130	250	0	0	187	0	3.5	3	0	2	0
5	41	0	2	130	204	0	2	172	0	1.4	1	0	2	0
6	56	1	2	120	236	0	0	178	0	0.8	1	0	2	0
7	62	0	3	140	268	0	2	160	0	3.6	3	2	2	1
8	57	0	3	120	354	0	0	163	1	0.6	1	0	2	0
9	63	1	3	130	254	0	2	147	0	1.4	2	1	3	1

6. Using princomp() performing PCA on the risk factors alone and print the summary which gives the number of components which account for 98% of variance.
7. Looking at cumulative proportion we observe that first three Principal Components must be considered.

```
> summary(pca_heart)
Importance of components:
                          Comp.1     Comp.2      Comp.3     Comp.4       Comp.5       Comp.6       Comp.7       Comp.8
Standard deviation     52.0044234 23.2915326 17.63226203 7.61491948 1.2056750431 0.9698378305 0.8926861795 0.8276666113
Proportion of Variance  0.7469414  0.1498311  0.08586616 0.01601536 0.0004014824 0.0002597791 0.0002200917 0.0001891981
Cumulative Proportion   0.7469414  0.8967725  0.98263870 0.99865406 0.9990555423 0.9993153214 0.9995354131 0.9997246112
                           Comp.9      Comp.10      Comp.11      Comp.12      Comp.13
Standard deviation     5.802973e-01 4.585557e-01 4.245621e-01 3.981827e-01 3.335909e-01
Proportion of Variance 9.300518e-05 5.807513e-05 4.978384e-05 4.378957e-05 3.073508e-05
Cumulative Proportion  9.998176e-01 9.998757e-01 9.999255e-01 9.999693e-01 1.000000e+00
```

8. Transforming the original heart data using the loadings (coefficients for PC) of each of the three components which retain most of the variance by the following process shown below:

```
##### Principal Components - linear calculation for each PC - PC1, PC2, PC3
comp1 <- loadings(pca_heart)[,1]
pc1 <- t(comp1)
View(pc1)
pc1_heart = vector();
pc1_heart <- pc1[1]*data_heart1$Age + pc1[2]*data_heart1$Sex + pc1[3]*data_heart1$chestpain1 + pc1[4]*data_heart1$RestBP + pc1[5]*data_heart1$Chol +
  pc1[6]*data_heart1$Fbs + pc1[7]*data_heart1$RestECG + pc1[8]*data_heart1$MaxHR + pc1[9]*data_heart1$ExAng + pc1[10]*data_heart1$Oldpeak +
  pc1[11]*data_heart1$Slope + pc1[12]*data_heart1$Ca + pc1[13]*data_heart1$thal1
View(pc1_heart)
```

9. Transformed dataset with three principal components – data_heart2

ml_prj2_pca_rand_datasets.R* | data_heart2 | data_heart1

Filter

	pc1_heart	pc2_heart	pc3_heart	ahd_value1
1	242.0300	121.52242	-158.4634	0
2	295.9283	78.83617	-164.8579	1
3	236.9508	103.10093	-131.7291	1
4	257.1843	164.35322	-144.7163	0
5	211.4611	148.35549	-145.6358	0
6	243.4202	153.12413	-136.7955	0
7	276.6543	132.51474	-152.9302	1
8	361.2501	140.01458	-128.3436	0
9	262.2265	120.65078	-142.1706	1

10. Get the sample_size by using floor() – 95% data in training and 5 % in test.
11. Use a loop condition to generate 50 train datasets (train_heart_pca) and 50 test datasets (test_heart_pca).

```
> ##### The below vectors have all the 50 random training and test datasets
> train_heart_pca
 [1] "pca_heart_train_1"  "pca_heart_train_2"  "pca_heart_train_3"  "pca_heart_train_4"  "pca_heart_train_5"  "pca_heart_train_6"
 [7] "pca_heart_train_7"  "pca_heart_train_8"  "pca_heart_train_9"  "pca_heart_train_10" "pca_heart_train_11" "pca_heart_train_12"
[13] "pca_heart_train_13" "pca_heart_train_14" "pca_heart_train_15" "pca_heart_train_16" "pca_heart_train_17" "pca_heart_train_18"
[19] "pca_heart_train_19" "pca_heart_train_20" "pca_heart_train_21" "pca_heart_train_22" "pca_heart_train_23" "pca_heart_train_24"
[25] "pca_heart_train_25" "pca_heart_train_26" "pca_heart_train_27" "pca_heart_train_28" "pca_heart_train_29" "pca_heart_train_30"
[31] "pca_heart_train_31" "pca_heart_train_32" "pca_heart_train_33" "pca_heart_train_34" "pca_heart_train_35" "pca_heart_train_36"
[37] "pca_heart_train_37" "pca_heart_train_38" "pca_heart_train_39" "pca_heart_train_40" "pca_heart_train_41" "pca_heart_train_42"
[43] "pca_heart_train_43" "pca_heart_train_44" "pca_heart_train_45" "pca_heart_train_46" "pca_heart_train_47" "pca_heart_train_48"
[49] "pca_heart_train_49" "pca_heart_train_50"
> test_heart_pca
 [1] "pca_heart_test_1"  "pca_heart_test_2"  "pca_heart_test_3"  "pca_heart_test_4"  "pca_heart_test_5"  "pca_heart_test_6"  "pca_heart_test_7"
 [8] "pca_heart_test_8"  "pca_heart_test_9"  "pca_heart_test_10" "pca_heart_test_11" "pca_heart_test_12" "pca_heart_test_13" "pca_heart_test_14"
[15] "pca_heart_test_15" "pca_heart_test_16" "pca_heart_test_17" "pca_heart_test_18" "pca_heart_test_19" "pca_heart_test_20" "pca_heart_test_21"
[22] "pca_heart_test_22" "pca_heart_test_23" "pca_heart_test_24" "pca_heart_test_25" "pca_heart_test_26" "pca_heart_test_27" "pca_heart_test_28"
[29] "pca_heart_test_29" "pca_heart_test_30" "pca_heart_test_31" "pca_heart_test_32" "pca_heart_test_33" "pca_heart_test_34" "pca_heart_test_35"
[36] "pca_heart_test_36" "pca_heart_test_37" "pca_heart_test_38" "pca_heart_test_39" "pca_heart_test_40" "pca_heart_test_41" "pca_heart_test_42"
[43] "pca_heart_test_43" "pca_heart_test_44" "pca_heart_test_45" "pca_heart_test_46" "pca_heart_test_47" "pca_heart_test_48" "pca_heart_test_49"
[50] "pca_heart_test_50"
>
```

Task 1: Logistic Regression

As discussed in the first Project, logistic regression is usually done when we have a binary response variable with values 1 or 0. Here we have AHD (diagnosis of heart disease) with Yes or No as responses, so we use logistic regression to predict values as shown in steps below.

Process: ml_prj2_logistic.R

1. Initialize temporary datasets - def, def_test.

2. Initialize a vector to hold the original and fitted values for test dataset.

3. Perform Logistic regression on training dataset (all 50).

4. Use the logit model from train dataset and Predict for test dataset.

5. If predicted values are > 0.5 then assign 1 (Yes) else 0 (No).

6. Reset the temporary datasets and variables.

7. To perform cross validation match the original and fitted values for all 50 iterations. If matched - 'No error' else 'Error' and calculate the percentage of accuracy.

8. Get the total # of errors and no errors from the 50 iterations to get an aggregate % of accuracy of the logistic model.

Results

Cross Validation – In Logistic regression we observe from comparing the 50 sets of predicted values to their original values that the percentage of accuracy of the models is maximum 86.67% (with 2 errors) to a minimum of 33.33% (with 10 errors). In project 1 we have got 100% maximum accuracy and 66.67% minimum accuracy for a set of 50 randomly generated datasets. Overall looking at the values for each random dataset, we can say that the accuracy rate has decreased after performing PCA on the input variables.

ml_prj2_logit.R | logit_result1

Filter

	error	no error	% of accuracy
3	2	13	86.66667
10	2	13	86.66667
19	2	13	86.66667
21	2	13	86.66667
34	2	13	86.66667
35	2	13	86.66667
40	2	13	86.66667
47	2	13	86.66667
1	3	12	80.00000
6	3	12	80.00000
7	3	12	80.00000

Showing 1 to 12 of 50 entries

ml_prj2_logit.R | logit_result1

Filter

	error	no error	% of accuracy
49	10	5	33.33333
41	8	7	46.66667
22	7	8	53.33333
30	7	8	53.33333
32	7	8	53.33333
39	7	8	53.33333
16	6	9	60.00000
20	6	9	60.00000
24	6	9	60.00000
29	6	9	60.00000
31	6	9	60.00000

Showing 1 to 12 of 50 entries

Viewing the overall aggregate accuracy rate of the logistic model we have 70.53% accuracy after performing PCA. The performance of this model while compared to the original values is 70.53% accurate.

```
> logit_aggr_accuracy_rate1 <- (sum_noerror1/(sum_error1+sum_noerror1))*100
> logit_aggr_accuracy_rate1
[1] 70.53333
```

Task 2: Support Vector Machines

As discussed in the previous project, Support Vector Machines method is helpful when the data can be linearly separable and the response variable is binary i.e. 1 or 0. Here we have the response variable AHD (diagnosis of heart disease) with Yes or No values and hence SVM method is used in predicting values. R's svm() function uses many parameters like cost, gamma and type of kernel.

- Based on the type of kernel chosen the 'linearly separable' feature of svm models improve. It provides a higher dimensionality of the dataset where the resemblance of linearly separable data is prominent. We have chosen radial as the kernel type.
- Cost in svm model regulates the cost of misclassification of the data. Smaller the cost parameter, lower is the cost of misclassification. (We have considered an optimal cost = 10)
- Gamma – is the kernel parameter. It defines the influence of the data points on generating the decision boundaries for support vectors. If we have a lower gamma then the influence is far and easy to linearly separate the data. Hence, we have considered it to be 0.1.

Process: It is similar to logistic regression except we will perform svm modelling here. (ml_prj2_svm.R)

1. Initialize temporary datasets - pqr, pqr_test.

2. Initialize a vector to hold the original and fitted values for test dataset.

3. Perform Support Vector Modelling (SVM) on training dataset (all 50).

4. Use the svm model from train dataset and Predict for test dataset.

5. If predicted values are > 0.5 then assign 1 (Yes) else 0 (No).

6. Reset the temporary datasets and variables.

7. To perform cross validation match the original and fitted values for all 50 iterations. If matched - 'No error' else 'Error' and calculate the percentage of accuracy.

8. Get the total # of errors and no errors from the 50 iterations to get an aggregate % of accuracy of the svm model.

Results

Cross Validation – From SVM modelling we observe that after comparing 50 random dataset's predicted values to their original values the percentage of accuracy of the models is maximum 86.67% (with 2 errors) to a minimum of 33.33% (with 10 errors). In project 1 we have got 100% maximum accuracy and 53.33% minimum accuracy for a set of 50 randomly

generated datasets. On an average we can say that we are getting a maximum of 10 errors i.e. the predicted values did not match the original values. Overall looking at the values for each random dataset, we can say that the accuracy rate has decreased after performing PCA on the input variables.

ml_prj2_svm.R × svm_result1 ×

Filter

	error	no error	% of accuracy
34	2	13	86.66667
3	3	12	80.00000
7	3	12	80.00000
28	3	12	80.00000
35	3	12	80.00000
10	4	11	73.33333
11	4	11	73.33333
18	4	11	73.33333
19	4	11	73.33333
29	4	11	73.33333
47	4	11	73.33333

Showing 1 to 12 of 50 entries

ml_prj2_svm.R × svm_result1 ×

Filter

	error	no error	% of accuracy
41	10	5	33.33333
49	10	5	33.33333
16	9	6	40.00000
9	8	7	46.66667
22	8	7	46.66667
24	8	7	46.66667
31	8	7	46.66667
32	8	7	46.66667
37	8	7	46.66667
1	7	8	53.33333
4	7	8	53.33333

Showing 1 to 12 of 50 entries

Svm model's aggregate accuracy rate is 62%. The performance of this model while compared to the original values is only 62% accurate.

```
> svm_aggr_accuracy_rate1 <- (sum_noerr1/(sum_err1+sum_noerr1))*100
> svm_aggr_accuracy_rate1
[1] 62
>
```

Conclusion:

Observing the aggregate accuracy rates from logistic model and svm model we can say that for Heart Disease Dataset it is appropriate to use Logistic Regression with pre-processing the risk factors using PCA as it has 70.5% accuracy from 50 randomly generated tests compared to SVM model which has 62% accuracy only (8.5% less than logistic regression). ***Based on the risk factors considered, logistic regression model can generate a better accuracy rate.***

Comparing the results obtained in Project 1 (without principal component analysis) to Project 2 results, we can say that the accuracy rate has decreased in both the models. In Logistic regression the accuracy rate has decreased by 13.33%. And performing SVM modelling, we have the accuracy rate decrease by 15.867%. The performance of the models is better when the dimensions are not changed and dataset is not transformed.

Model Type	**Accuracy Rate – Without PCA**	**Accuracy Rate – With PCA**
Logistic Regression	83.867%	70.53%
SVM Modelling	77.867%	62%