200301-EDA_and_model-yuqi

Yuqi Miao ym2771

3/1/2020

Read in Data and Manipulation

$$\log(\frac{\pi_i}{1-\pi_i}) = \mathbf{x}_i\boldsymbol{\beta}$$

validation using glm

questions or modify:

1. normalize or standardize?
2. how to standardize easily?

```
# cleaning the above x
library(sjmisc)
y=as.data.frame(x$cv_result)
y_y=rotate_df(y)
names(y_y)=c("Enter","Fold1","Fold2","Fold3","Fold4","Fold5")
knitr::kable(y_y)
```

	Enter	Fold1	Fold2	Fold3	Fold4	Fold5
k	0	1.0000000	2.0000000	3.0000000	4.0000000	5.0000000
best_lambda	0	0.0000000	0.0000000	0.0000000	0.0000000	0.0303030
beta_vec1	1	-0.6663464	-0.8606693	-0.6244876	-0.5355721	-0.7737072
beta_vec2	1	1.4446157	1.7172558	2.1313547	2.2414343	0.2165391
beta_vec3	1	1.1141379	1.0516070	1.0438591	1.0085630	0.3121205
beta_vec4	1	0.1525774	0.0710993	0.3134996	0.1776248	0.0000000
beta_vec5	1	0.3582305	0.2507639	-0.0270073	0.1110899	0.0000000
beta_vec6	1	0.7406111	0.6795368	1.0789549	1.0204367	0.0000000
beta_vec7	1	-0.7592081	-0.5953038	-0.0628564	-0.3732903	0.0000000
beta_vec8	1	0.9333178	0.7112050	1.0380966	0.3601083	0.0000000
beta_vec9	1	1.1557369	1.4566499	0.2751937	1.9059881	0.8541841
beta_vec10	1	0.0243545	0.3555338	0.0343354	0.1857253	0.0000000
beta_vec11	1	-0.2790475	-0.4709215	-0.5075943	-0.5182740	0.0000000
beta_vec12	1	1.4844467	1.1900859	1.1684542	1.1049459	0.0000000
beta_vec13	1	-0.1769061	-0.0938746	-0.0395756	-0.1096598	0.0000000
beta_vec14	1	1.3698537	1.2170041	1.0613571	0.9619370	0.0000000
beta_vec15	1	1.6500516	1.3190013	1.2700602	1.1353677	0.0000000
beta_vec16	1	-0.1667238	-0.1415278	-0.3170986	-0.4389349	0.0000000
beta_vec17	1	-0.5206480	-0.3403789	-0.3967853	-0.3717366	0.0000000
beta_vec18	1	-0.3255053	-0.1431664	-0.2770359	-0.1385741	0.0000000
beta_vec19	1	-0.4716457	-0.2749817	-0.3331025	-0.2050350	0.0000000
beta_vec20	1	-0.6296226	-0.3529685	-0.3045117	-0.3653631	0.0000000
beta_vec21	1	-0.4037133	-0.2866930	-0.3612959	-0.3119301	0.0000000
beta_vec22	1	4.0435322	2.7400581	3.7517734	2.3336327	2.0672393
beta_vec23	1	0.8868274	1.1107406	1.2166616	1.0809459	0.7069871
beta_vec24	1	4.4380960	2.9678262	3.7811780	2.3000294	1.8912901
beta_vec25	1	2.4154698	1.4236061	2.0741866	1.1815997	0.3085923
beta_vec26	1	1.0240182	0.9518878	0.6785581	0.6943120	0.1512226

	Enter	Fold1	Fold2	Fold3	Fold4	Fold5
beta_vec27	1	0.4841746	0.6177355	0.7418958	0.6275855	0.0000000
beta_vec28	1	0.9079662	0.7588550	1.1239071	0.7332618	0.1695023
beta_vec29	1	2.9778771	3.5427933	4.0128312	3.8603744	2.4497217
beta_vec30	1	1.0674250	1.2512243	0.9738290	0.9026693	0.4915037
beta_vec31	1	0.4788639	0.5340397	0.6037901	0.4549507	0.0000000
g.stat_tr	Inf	NaN	NaN	NaN	NaN	59.6412968
auc_te	0	0.9986486	0.9993243	0.9983897	0.9920077	0.9705285
g.stat_te	Inf	NaN	NaN	NaN	NaN	1868.0016983
SSE_test	Inf	2.5853338	2.0644662	3.2451899	4.0137240	3.8903373

instead of using MSE, using pearson chi-square

validation

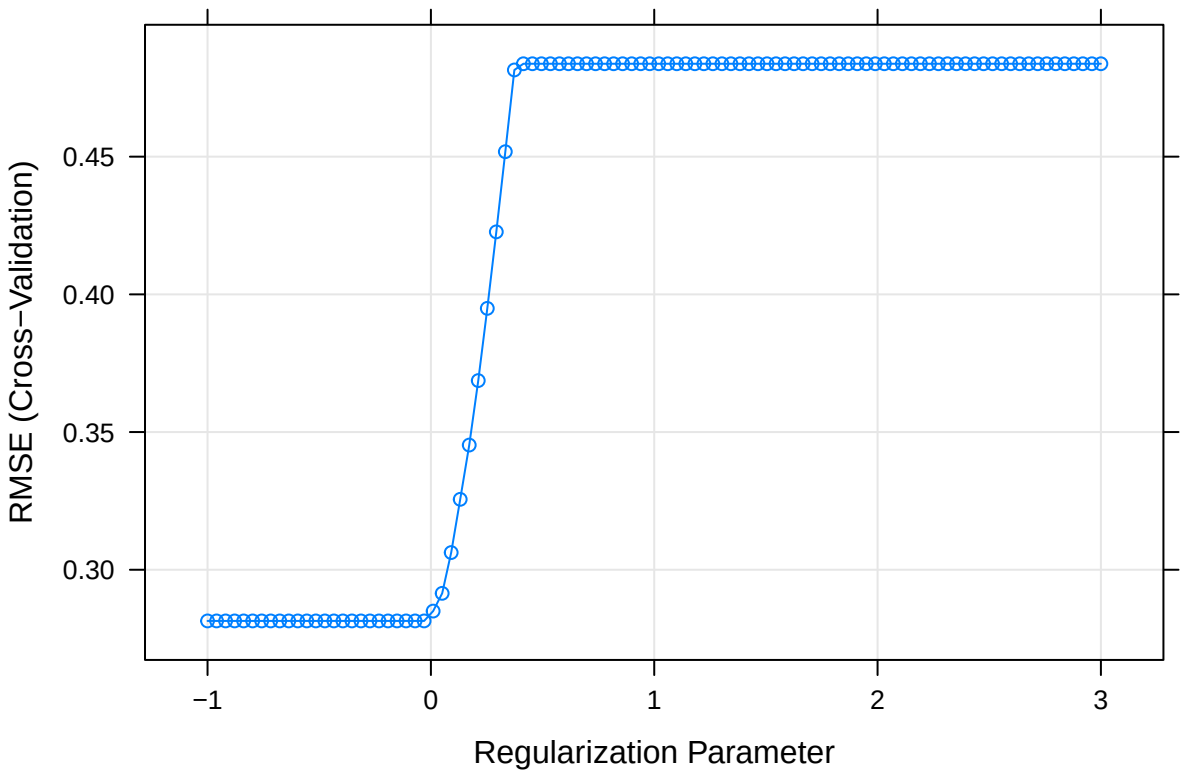
```
x.mat <- model.matrix(diagnosis~., cancer_package[-1])[,-1]
y.class <- cancer_package$diagnosis

ctrl1 <- trainControl(method = "cv", number = 5)
lasso.fit <- train(x.mat, y.class,
                   method = "glmnet",
                   tuneGrid = expand.grid(alpha = 1,
                                          lambda = seq(3, -1,length = 100)),
                   # preProc = c("center", "scale"),
                   trControl = ctrl1)

lasso.fit$bestTune
```

```
##    alpha      lambda
## 25     1 -0.03030303
```

```
plot(lasso.fit)
```



```r
# min(lasso.fit$results$RMSE)
# co=coef(lasso.fit$finalModel,lasso.fit$bestTune$lambda)
# co2=co@x
#
# names(co2)=co@Dimnames[[1]]
# co2 %>% as.data.frame() %>% knitr::kable()
```