03_MachineLearning

June 26, 2019

1 Import Libraries

```
In [1]: library( tidyverse )
```

```
Registered S3 methods overwritten by 'ggplot2':
  method         from
  [.quosures     rlang
  c.quosures     rlang
  print.quosures rlang
Registered S3 method overwritten by 'rvest':
  method            from
  read_xml.response xml2
 Attaching packages  tidyverse 1.2.1
 ggplot2 3.1.1      purrr   0.3.2
 tibble  2.1.1      dplyr   0.8.1
 tidyr   0.8.3      stringr 1.4.0
 readr   1.3.1      forcats 0.4.0
 Conflicts  tidyverse_conflicts()
 dplyr::filter() masks stats::filter()
 dplyr::lag()    masks stats::lag()
```

```
In [2]: library( tidymodels )
```

```
Registered S3 method overwritten by 'xts':
  method     from
  as.zoo.xts zoo
 Attaching packages  tidymodels 0.0.2
 broom      0.5.2        recipes    0.1.5
 dials      0.0.2        rsample    0.0.4
 infer      0.4.0.1      yardstick  0.0.3
 parsnip    0.0.2
 Conflicts  tidymodels_conflicts()
 scales::discard() masks purrr::discard()
 dplyr::filter()   masks stats::filter()
 recipes::fixed()  masks stringr::fixed()
 dplyr::lag()      masks stats::lag()
```

```
 yardstick::spec() masks readr::spec()
 recipes::step()   masks stats::step()
```

```
In [3]: library( GGally )
```

```
Registered S3 method overwritten by 'GGally':
  method from
  +.gg   ggplot2

Attaching package: GGally

The following object is masked from package:dplyr:

    nasa
```

```
In [4]: library( ggfortify )
```

2 Non-linearly separable data

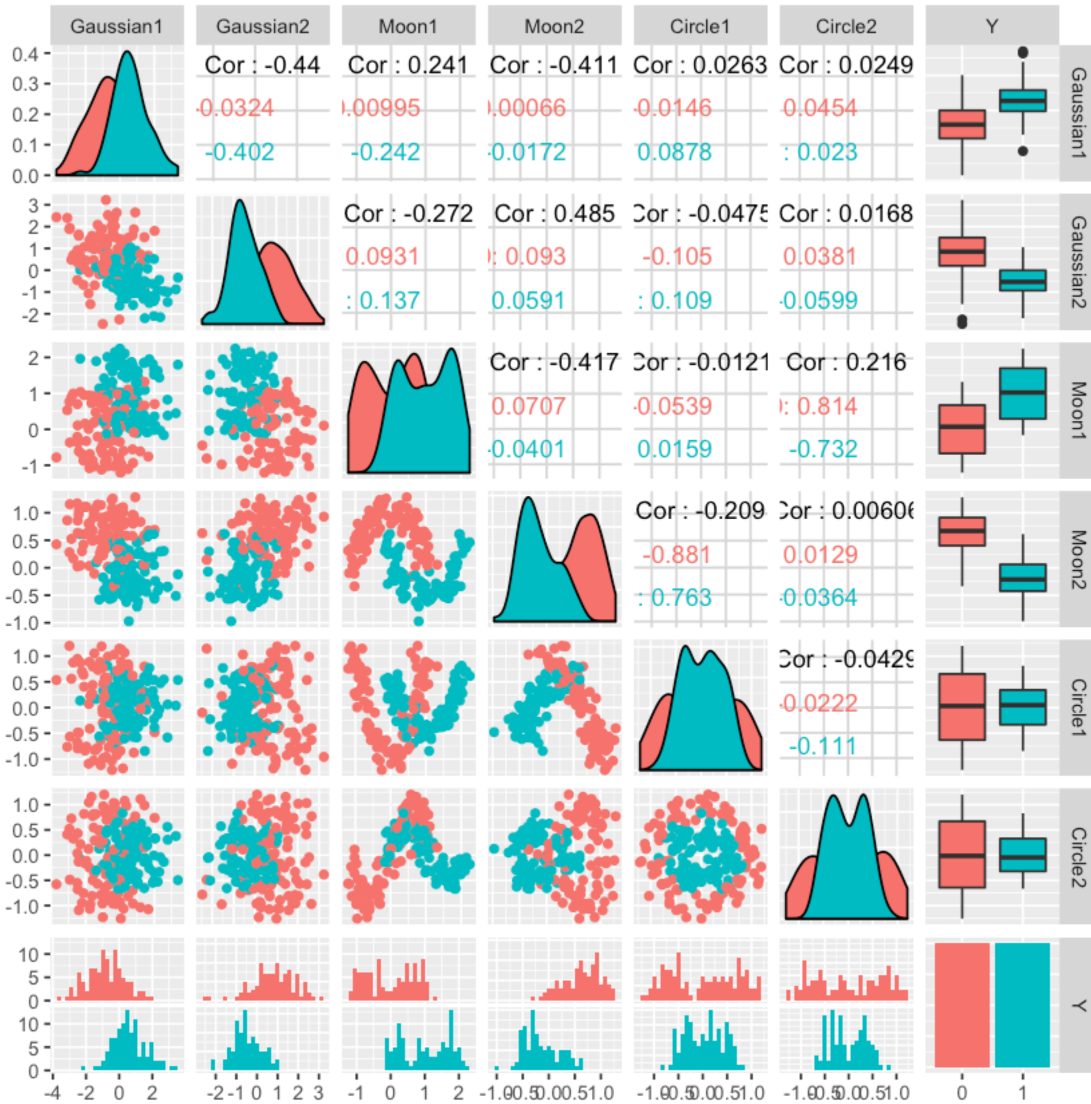
```
In [5]: funkydata <- read_csv( 'funkydata.csv')
```

```
Parsed with column specification:
cols(
  Gaussian1 = col_double(),
  Gaussian2 = col_double(),
  Moon1 = col_double(),
  Moon2 = col_double(),
  Circle1 = col_double(),
  Circle2 = col_double(),
  Y = col_double()
)
```

```
In [6]: funkydata$Y <- factor( funkydata$Y  )

In [7]: funkydata %>% ggpairs( aes( color=Y ) )
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
In [8]: options(repr.plot.width=10, repr.plot.height=10)
```

```
In [9]: read_csv( 'unequal_variance_data.csv' ) %>%
            mutate( Y=factor(Y) ) %>%
            ggpairs( aes( color=Y, alpha=0.1 ) )
```

```
Parsed with column specification:
cols(
  `0` = col_double(),
  `1` = col_double(),
  `2` = col_double(),
  `3` = col_double(),
  `4` = col_double(),
```

```
  `5` = col_double(),
  `6` = col_double(),
  `7` = col_double(),
  `8` = col_double(),
  `9` = col_double(),
  Y = col_double()
)
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

3 Funkydata: Hold some test data in reserve to assess model fit

```
In [10]: set.seed( 42 )
         data_splitter <- initial_split( funkydata, prop=0.8 )
         train_data <- training( data_splitter )
         test_data <- testing( data_splitter )
```

4 Train a Logistic Regression model

- Plain vanilla logistic regression

```
In [11]: model0 <- glm( Y ~ 1, train_data, family='binomial' )
```

```
In [12]: summary( model0 )
```

```
Call:
glm(formula = Y ~ 1, family = "binomial", data = train_data)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-1.109  -1.109  -1.109   1.247   1.247

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -0.1618     0.1581  -1.023    0.306

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 222.14  on 160  degrees of freedom
Residual deviance: 222.14  on 160  degrees of freedom
AIC: 224.14

Number of Fisher Scoring iterations: 3
```

```
In [13]: model1 <- glm( Y ~ Gaussian1 + Gaussian2, train_data, family='binomial' )
```

```
In [14]: summary( model1 )
```

```
Call:
glm(formula = Y ~ Gaussian1 + Gaussian2, family = "binomial",
    data = train_data)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-2.54001  -0.54082  -0.08572   0.50573   1.96573

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -0.05751    0.23391  -0.246    0.806
Gaussian1    1.35899    0.28286   4.804 1.55e-06 ***
Gaussian2   -1.45367    0.29487  -4.930 8.23e-07 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 222.14  on 160  degrees of freedom
Residual deviance: 115.99  on 158  degrees of freedom
```

```
AIC: 121.99

Number of Fisher Scoring iterations: 6
```

```
In [15]: anova( model0, model1 )
```

A df[,4]: 2 Œ 4

Resid. Df	Resid. Dev	Df	Deviance
<dbl>	<dbl>	<dbl>	<dbl>
160	222.1426	NA	NA
158	115.9891	2	106.1534

4.1 Use augment() function to attached fitted values to original data frame

```
In [16]: augmented_funky1 <- augment( model1 )
```

```
In [17]: augmented_funky1 %>% head
```

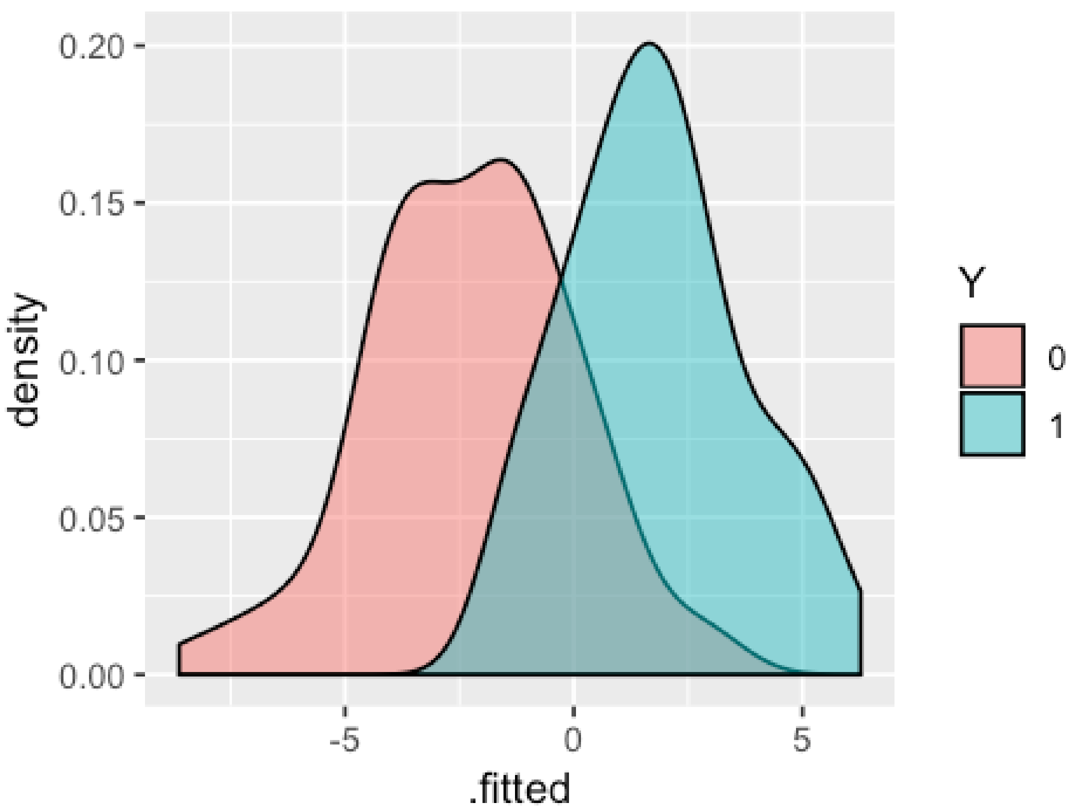
A tibble: 6 Œ 10

Y	Gaussian1	Gaussian2	.fitted	.se.fit	.resid	.hat	.sigma
<fct>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
0	-0.7994022	0.1545466	-1.368548	0.3410897	-0.6733768	0.018813077	0.85781
0	-0.3301211	2.3876841	-3.977038	0.7507004	-0.1927051	0.010176560	0.85938
1	1.4185881	-0.8775885	3.146052	0.5572896	0.2902479	0.012281822	0.85920
0	-1.1174380	1.8842391	-4.315150	0.7157296	-0.1629480	0.006666868	0.85942
0	-0.9947197	0.1794926	-1.670245	0.3845873	-0.5872655	0.019716611	0.85822
0	-0.9994371	-2.4650403	2.167613	0.7913876	-2.1335296	0.057713163	0.84143

4.2 Plot distribution of TRAINING set fitted values colored by class

```
In [18]: options(repr.plot.width=4, repr.plot.height=3)
```

```
In [19]: augmented_funky1 %>% ggplot( aes( x=.fitted, fill=Y) ) + geom_density( alpha=0.5 )
```



4.3 Plot distribution of TEST set fitted values colored by class

```
In [20]: Ypred_test <- predict( model1, test_data )
```

```
In [21]: qplot( Ypred_test, geom='density', fill=test_data$Y, alpha=0.5 )
```

5 Logit link function

- The target variable Y is binary (0/1, loss/win)
- The output is not a 0/1 directly, but the probability of a win
- Linear regression involves solving simultaneous linear equations => linear combinations
- Predicted values of a linear regression MUST also be linear. Consider:
  - $\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + ... + \hat{\beta}_n x_n + \epsilon$
  - Use logit link function
  - $logit = log(Odds) = log(\frac{p}{1-p}) = \hat{\beta}_0 + \hat{\beta}_1 x_1 + ... + \hat{\beta}_n x_n + \epsilon$
  - logit function is a "sigmoid" function
- Exponentiate the coefficient to get the odds ratio - if bigger than 1, a 1 unit change in x is an increase
  - Odds ratio estimates = "times more likely" - probability of a win over probability of a loss

```
In [22]: coef( model1 )
```

(Intercept) -0.0575112921833882 **Gaussian1** 1.35898693828992 **Gaussian2** -1.45366635126567

```
In [23]: exp( coef( model1 ) )
```

(Intercept) 0.944111229249089 **Gaussian1** 3.8922482161911 **Gaussian2** 0.233711845658519

```
In [24]: Ypred_test <- predict( model1, test_data )
```

```
In [25]: head( Ypred_test )
```

1 1.22155896041885 **2** -4.15266485413218 **3** -0.440851379841008 **4** 2.58701268252882 **5** -3.23406378302684 **6** -0.190908032390208

5.1 Use type="response" argument to predict() to get probabilities

```
In [26]: Ypred_test <- predict( model1, test_data, type='response')
```

```
In [27]: head(Ypred_test)
```

1 0.772337781135761 **2** 0.0154790930787615 **3** 0.391538121188741 **4** 0.930021046405469 **5** 0.0379037741712596 **6** 0.452417419932877

5.2 Classification metrics

- Goodness of fit is not adjusted R-squared, but rather accuracy, F1, ROC curve AUC, others...

```
In [28]: # In class activity 1: How to get test prediction accuracy?
```

```
In [29]: # Homework: How to get four-square confusion matrix of TP/FP/FN/TN?
```

5.3 ROC Curve

- How to create a ROC curve

```
In [30]: install.packages( "ROCit" )
```

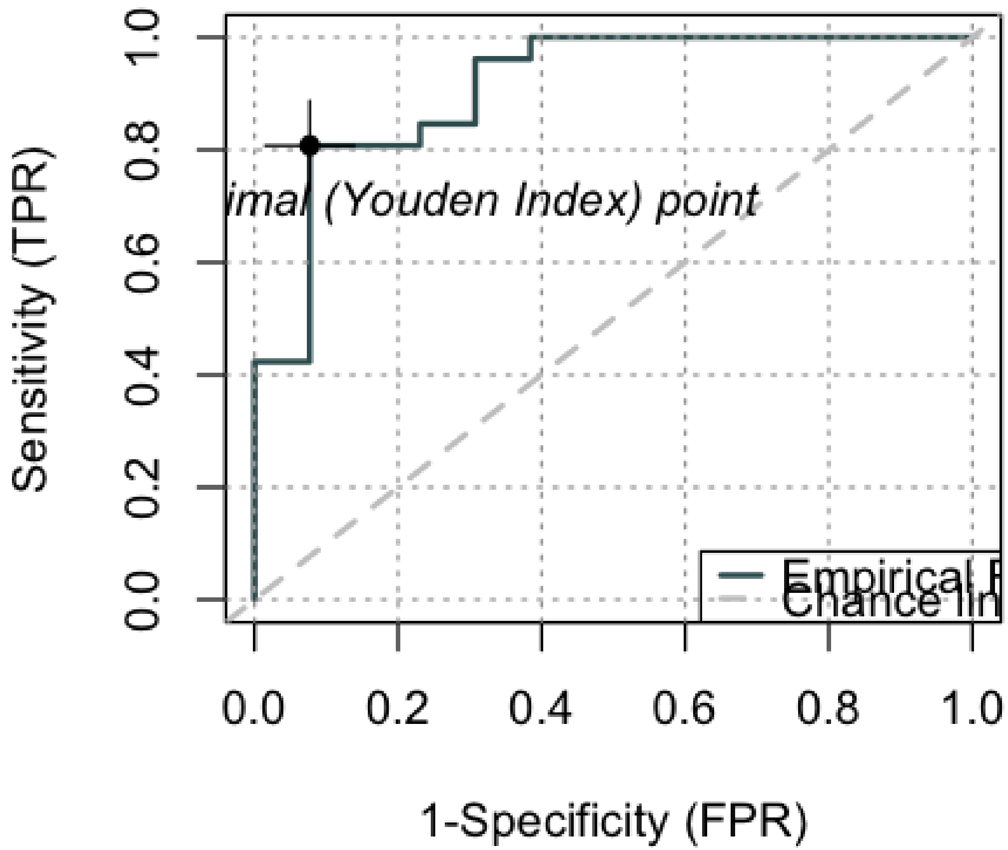
```
Installing package into /usr/local/lib/R/3.6/site-library
(as lib is unspecified)
```

```
In [31]: library(ROCit)
```

```
In [32]: ROCit_obj <- rocit( score = Ypred_test, class = test_data$Y )
```

```
In [33]: options(repr.plot.width=4, repr.plot.height=4)
```

```
In [34]: plot(ROCit_obj)
```



5.4 Other Logistic regression considerations

5.4.1 Categorical/nominal predictor variables

- One hot encoding: the creation of a dummy variable for each level, e.g.,
  - If you have a nominal variable with 12 cases, you just picked up 11 variables. Each one is going to have it's own coefficient.
- Options: change variable type to interval like an ordinal, or bin into "other" category

6 Retrain GLM model with Parsnip interface (Tidyverse for modelling!)

- Parsnip is a unified modelling interface, allowing you to swap in and out classification algorithms easily

```
In [35]: test_predictions <- logistic_reg() %>%
             set_engine( "glm" ) %>%
             fit( Y ~ Gaussian1 + Gaussian2, train_data ) %>%
             predict( test_data ) %>%
             bind_cols( test_data )
```

```
In [36]: head( test_predictions )
```

A tibble: 6 Œ 8

.pred_class <fct>	Gaussian1 <dbl>	Gaussian2 <dbl>	Moon1 <dbl>	Moon2 <dbl>	Circle1 <dbl>	Circle2 <dbl>
1	1.65586765	0.66812598	-0.7572982	0.67647928	0.1200559	-1.07310621
0	-0.18800968	2.64135636	-0.3005245	0.71755671	-0.6291578	-0.96904142
0	0.06262386	0.32225076	2.0400183	0.41237844	0.4888741	-0.28609402
1	1.41794353	-0.49361894	1.8422351	0.00584937	0.5923969	-0.65389154
0	-2.31634471	0.01972274	1.9629823	0.60950815	0.1341592	-0.07741052
0	-0.22614206	-0.11964738	0.6500019	-0.14814799	-0.6925768	0.33714084

```
In [37]: #model_metrics <- metric_set( accuracy )
         #test_predictions %>%
         #    model_metrics
```

7 Random Forest classifier

- RF YouTube explainer

```
In [38]: rand_forest_model <- rand_forest() %>%
             set_engine( "ranger" )
```

```
In [39]: rf_fit <- rand_forest_model %>%
             fit( Y ~ Gaussian1 + Gaussian2, train_data )
```

```
In [40]: rf_fit
```

```
parsnip model object

Ranger result

Call:
 ranger::ranger(formula = formula, data = data, num.threads = 1,      verbose = FALSE, seed = s

Type:                             Classification
Number of trees:                  500
```

```
Sample size:                      161
Number of independent variables:  2
Mtry:                             1
Target node size:                 1
Variable importance mode:         none
Splitrule:                        gini
OOB prediction error:             19.88 %
```

```
In [ ]: #rf_fit %>%    predict( test_data )
```

8 XGBoost classifier

- Gradient-boosted classification trees YouTube explainer

```
In [41]: test_predictions <- boost_tree() %>%
             set_engine( "xgboost" ) %>%
             fit( Y ~ Gaussian1 + Gaussian2, train_data ) %>%
             predict( test_data ) %>%
             bind_cols( test_data )
```

```
In [42]: test_predictions
```

A tibble: 39 Œ 8

.pred_class	Gaussian1	Gaussian2	Moon1	Moon2	Circle1	Circ
<fct>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<db
0	1.6558676502	0.668125980	-0.75729820	0.67647928	0.120055937	-1.0
0	-0.1880096751	2.641356365	-0.30052452	0.71755671	-0.629157830	-0.9
0	0.0626238564	0.322250763	2.04001830	0.41237844	0.488874131	-0.2
1	1.4179435275	-0.493618939	1.84223510	0.00584937	0.592396893	-0.6
0	-2.3163447117	0.019722740	1.96298234	0.60950815	0.134159196	-0.0
1	-0.2261420600	-0.119647377	0.65000191	-0.14814799	-0.692576774	0.33
1	0.2270759775	-0.507753673	1.77716240	-0.42903955	0.003267762	-0.5
0	-0.5969662960	0.369458824	0.97169175	-0.56735363	-0.562316475	0.07
1	0.8586829764	-0.331131219	0.14044903	-0.21966508	0.162196055	0.23
0	-2.4770264147	0.923472230	0.70305250	0.02496383	0.945482393	0.03
0	-1.8037906851	-0.101255631	0.93810901	0.66298693	0.069959399	1.04
0	-1.9608033259	0.443044737	-0.32638898	1.10822502	-0.756865727	-0.1
0	-2.4625986710	0.846115057	0.10872866	0.85881768	-0.792590975	0.43
1	1.6123753093	0.074948580	-0.08007019	0.56677819	0.514306242	-0.1
0	-0.4342303771	0.163155565	1.60443748	-0.54617186	-0.345529082	-0.5
0	-1.5833100217	1.227229722	0.92649946	0.49189021	0.438684118	0.77
1	0.5401740742	-0.073084328	0.19005626	0.31674494	0.501649694	0.29
0	-0.7353438489	0.826731161	2.01274529	0.49230467	0.812581460	-0.2
1	1.2350916284	-0.769249327	0.11536418	0.05004481	0.184154812	0.62
0	0.5932292715	1.061860497	0.88882760	-0.49659493	-0.344694977	0.39
1	0.0007393842	-0.924164375	2.24426279	0.46433851	0.612628746	-0.1
0	-1.3827299949	0.139191527	1.89423265	0.09442643	0.169541888	-0.3
1	2.7854904947	-1.458262531	0.38183860	-0.34377601	0.059557048	0.30
0	-2.7218274490	0.687199725	0.99326509	0.16822611	0.876884275	0.32
1	0.5482647824	-0.504186671	0.18150563	0.18133960	0.558896594	0.32
1	1.0196412973	-0.749209166	0.28320170	-0.19666803	0.094622159	0.45
0	0.4798458503	1.484237232	1.10688118	0.17218468	0.840630299	0.56
0	-1.7574668241	1.059429793	-0.68699186	0.81902125	-0.411478044	-1.0
0	-0.7949517753	0.404936934	-0.13037966	0.38523097	0.486272751	0.43
1	0.0583034403	-0.907377916	2.03606695	0.25954895	0.278357534	-0.1
1	-0.2906075251	-0.225703143	2.14704453	0.36909832	0.703435515	-0.0
1	0.2566899621	-1.098083214	0.82855720	-0.23845831	-0.601888625	-0.1
0	-0.7473150173	-0.608385809	0.92504046	-0.28293448	-0.441012071	0.49
0	-1.2954867952	2.084281856	-0.20381067	1.15105763	-0.782344224	-0.5
0	0.7379528314	1.599841231	0.93596514	0.57630864	0.619781358	0.65
1	-0.3933948255	0.004419483	1.73905958	0.01263737	0.210900076	-0.4
0	-1.8262258001	-1.065161693	0.01023941	0.79893048	-1.091185914	0.12
1	0.3041291925	-0.536522506	1.64894910	-0.55382701	-0.332578397	-0.6
1	1.9812212804	-1.418520529	-0.02149899	0.61356625	0.539736454	-0.0