hdm_intro_jl

October 6, 2022

1 Hight-Dimmensional Metrics in Julia

1.1 Introduction

1.2 How to Get Started

[15]:
```
# ]add HDMjl
using CSV, DataFrames
function r_data(n = 1)
    n_m = "r_" * string(n) * ".csv"
    dta = CSV.read(n_m, DataFrame)
    return dta
end
```

[15]:
```
r_data (generic function with 2 methods)
```

1.3 Prediction Using Approximate Sparsity

[16]:
```
using Random, Distributions
include("..\\src\\HDMjl.jl")
# pwd()
```

```
WARNING: replacing module HDMjl.
```

[16]:
```
Main.HDMjl
```

[17]:
```
## 32 A Joint Significance test for Lasso Regression
# Random.seed!(12345)
# n = 100
# #sample size
# p = 100
# # number of variables
s = 3
# # nubmer of variables with non-zero coefficients
# X = rand(Normal(), (n, p))
# Y = X * beta + randn(n);
dta = r_data(1)
n, p = size(dta)
beta = vcat(fill(3, s), zeros(p - s));
```

```
p = p-1
X = dta[:, Not(1)]
Y = dta[:, 1];
```

[18]:
```
lasso_reg = HDMjl.rlasso(X, Y, post = false)
# use lasso, not-Post-lasso
# lassoreg = rlasso(X, Y, post=false)
sum_lasso = HDMjl.r_summary(lasso_reg, all = false)
# can also do print(lassoreg, all=false)
```

```
    Post-Lasso Estimation: false
    Total number of variables: 50
    Number of selected variables: 3
    ---

============ ============
  Variable     Estimate
============ ============
  Intercept    0.0307194
  V2           4.40362
  V3           4.33222
  V4           4.39125
============ ============

    ----
    Multiple R-squared: 0.969344147763692
    Adjusted R-squared: 0.9683861523813074
```

[23]:
```
yhat_lasso = HDMjl.r_predict(lasso_reg)
#in-sample prediction
# Xnew = rand(Normal(), (n, p))
# new X
# Ynew = Xnew * beta + randn(n)
#new Y

dta11 = r_data(1.1)
Xnew = Matrix(dta11[:, Not(1)])
Ynew = dta11[:, 1]
# HDMjl.r_predict()
yhat_lasso_new = HDMjl.r_predict(lasso_reg, xnew = Xnew)
#out-of-sample prediction
post_lasso_reg = HDMjl.rlasso(X, Y, post = true);
#now use post-lasso
HDMjl.r_summary(post_lasso_reg, all = false)
# lasso_reg
```

```
Post-Lasso Estimation: true
Total number of variables: 50
Number of selected variables: 4
---

============ =============
  Variable     Estimate
============ =============
  Intercept    0.00223374
  V2           4.98173
  V3           5.01485
  V4           5.02564
  V22          -0.443961
============ =============

----
Multiple R-squared: 0.9871727371309015
Adjusted R-squared: 0.9866326418522026
```

[24]:
```
yhat_post_lasso = HDMjl.r_predict(post_lasso_reg)
#in-sample prediction
yhat_post_lasso_new = HDMjl.r_predict(post_lasso_reg, xnew = Xnew)
#out-of-sample prediction
MAE = hcat(abs.(Ynew - yhat_lasso_new), abs.(Ynew - yhat_post_lasso_new))
mean.(eachcol(MAE))
# names(MAE) = c("lasso MAE", "Post-lasso MAE")
# print(MAE, digits = 2)
```

[24]:
```
2-element Vector{Float64}:
 1.4760345675461333
 1.0345207789641953
```

1.4 Inference on Target Regression Coefficients

[26]:
```
#41 Intuition for the Orthogonality Principle in Linear Models via Partialling␣
↪Out
using DataFrames, Pipe
Random.seed!(1)
dta2 = r_data(2)
X = dta2[:, Not(1)]
y = dta2[:, 1]
d = dta2[:, 2]
n, p = size(X)
px = p - 2
# n = 5000
# p = 20
# X = rand(Normal(), (n, p))
```

```
# d = X[:, 1] #|> rename(_, :x1 => :d)
X1 = X[:, 2:p]

beta = ones(p)
# y = X * beta + randn(n);
```

[27]:
```
using GLM

function intercept(mtrx)
    mtrx = Matrix(mtrx)
    return hcat(ones(size(mtrx, 1)), mtrx)
end

full_fit = GLM.lm(intercept(X), y)

est = round(coeftable(full_fit).cols[1][2], digits = 3)
s_td = round(coeftable(full_fit).cols[2][2], digits = 3)

print("Estimate: $est ($s_td)")
```

```
Estimate: 0.978 (0.014)
```

[28]:
```
lm_y = lm(intercept(X1), y)
lm_d = lm(intercept(X1), d)
# lm_y
rY = GLM.residuals(lm_y)
rd = GLM.residuals(lm_d)

partial_fit_ls = lm(hcat(ones(n), rd), rY)

est = round(coeftable(partial_fit_ls).cols[1][2], digits = 3)
s_td = round(coeftable(partial_fit_ls).cols[2][2], digits = 3)

print("Estimate: $est ($s_td)")
```

```
Estimate: 0.978 (0.014)
```

[29]:
```
rY = HDMjl.rlasso(X1, y)["residuals"]
rd = HDMjl.rlasso(X1, d)["residuals"]
# intercept(rd)
# rY
partial_fit_ls = GLM.lm(intercept(rd), rY[:, 1])


est = round(coeftable(partial_fit_ls).cols[1][2], digits = 3)
s_td = round(coeftable(partial_fit_ls).cols[2][2], digits = 3)

print("Estimate: $est ($s_td)")
```

Estimate: 0.973 (0.014)

1.5 Instrumental Variable Esimation in a High-Dimensional Setting

[30]:
```
Eff = HDMjl.rlassoEffect(X[:, Not(1)], y, X[:, 1], method = "partialling out")
HDMjl.r_summary(Eff);
```

```
Estimates and significance testing of the effect of target variables
  Row   Estimate.   Std. Error   t value

Pr(>|t|)

    1    0.972739    0.0136868   71.0715        0.0
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

1.5.1 Error rlassoEffect

methd = "double selection"

[ ]:
```
# Eff = HDMjl.rlassoEffect(X[:, Not(1)], y, X[:, 1], method = "double␣
↪selection")
# HDMjl.r_summary(Eff);
```

[31]:
```
##42 Inference confidence Intervals and Significance Testing

n = 100
#sample size
p = 100
# number of variables
s = 3
# nubmer of non-zero variables
X = rand(Normal(), (n, p))

dta3 = r_data(3)
y = dta3[:, 1]
X = dta3[:, Not(1)]



# beta = vcat(fill(3, s), zeros(p - s))
# y = 1 .+ X * beta + randn(n);
```

[32]:
```
lassoeffect = HDMjl.rlassoEffects(X, y, index = [1, 2, 3, 8])
HDMjl.r_print(lassoeffect)
```

```
Coefficients:


     X1      X2      X3       X8
```

```
2.944    3.041    2.975    -0.096
```

[33]:
```
HDMjl.r_summary(lassoeffect)
```

```
Estimates and significance testing of the effect of target variables
        Estimate.    Std. Error     t value

Pr(>|t|)

  X1       2.94448    0.0881468    33.4043    1.1892e-244
  X2       3.04127     0.083891    36.2527    9.0141e-288
  X3        2.9754    0.0780394     38.127   4.58085e-318
  X8    -0.0961403    0.0790902   -1.21558       0.224146
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

[34]:
```
HDMjl.r_confint(lassoeffect)
```

	2.5%	97.5%
X1	2.77171	3.11724
X2	2.87685	3.2057
X3	2.82245	3.12836
X8	-0.251154	0.0588737

[35]:
```
HDMjl.r_confint(lassoeffect, 0.99)
```

	0.5%	99.5%
X1	2.71743	3.17153
X2	2.82519	3.25736
X3	2.77439	3.17642
X8	-0.299863	0.107583

plot_cof

[ ]:
```
# plot(lassoeffect, main = "Canfidence Intervals")
```

[53]:
```
using RData, CodecXz, StatsModels, DataFrames
url = "https://github.com/cran/hdm/raw/master/data/cps2012.rda";
cps2012 = load(download(url))["cps2012"][1:500, :];

x_formula = @formula(lnw ~ -1 + widowed + divorced + separated + nevermarried +
hsd08 + hsd911 + hsg + cg + ad + mw + so + we + exp1 + exp2 + exp3)

x_dframe = ModelFrame( x_formula, cps2012)
x1 = ModelMatrix(x_dframe)
x = x1.m
```

```
y = cps2012[:,"lnw"];
# rlassoEffects(x,y)
```

[55]:
```
@time effects_female = HDMjl.rlassoEffects(x, y);
```

```
  2.774894 seconds (2.59 M allocations: 8.683 GiB, 16.68% gc time)
```

[56]:
```
HDMjl.r_summary(effects_female)
```

```
Estimates and significance testing of the effect of target variables
         Estimate.   Std. Error     t value
    Pr(>|t|)

   X1    0.0174897       0.2683   0.0651872       0.948025
   X2    -0.230958    0.0791962    -2.91628     0.00354238
   X3     0.194484     0.238039    0.817026       0.413914
   X4    -0.248817    0.0706749    -3.52058      0.0004306
   X5    -0.352518     0.306652    -1.14957       0.250322
   X6    -0.314656     0.153306    -2.05247      0.0401243
   X7   -0.0595341    0.0582492    -1.02206       0.306753
   X8     0.246817    0.0638804     3.86374    0.000111665
   X9     0.595771    0.0960622     6.20193    5.57746e-10
  X10          0.0          NaN         NaN            NaN
  X11          0.0          NaN         NaN            NaN
  X12          0.0          NaN         NaN            NaN
  X13    0.0220862    0.0136327     1.62008       0.105214
  X14    -0.173032     0.214296   -0.807442       0.419412
  X15   -0.0155836    0.0133694    -1.16562       0.243769
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

[60]:
```
jointCI = HDMjl.r_confint(effects_female, 0.95)#, joint = true)
jointCI
```

```
                2.5%        97.5%

   X1     -0.508369     0.543348
   X2      -0.38618   -0.0757362
   X3     -0.272064     0.661031
   X4     -0.387337    -0.110297
   X5     -0.953545      0.24851
   X6      -0.61513   -0.0141812
   X7       -0.1737    0.0546323
   X8      0.121614      0.37202
   X9      0.407493      0.78405
  X10           NaN          NaN
  X11           NaN          NaN
  X12           NaN          NaN
```

X13	-0.00463351	0.0488058
X14	-0.593044	0.246981
X15	-0.0417872	0.0106199

[24]:
```
# Syssleep(7)
# effectsfemale = rlassoEffects(lnw ~ female + female:(widowed + divorced +␣
↪separated +
# nevermarried + hsd08 + hsd911 + hsg + cg + ad + mw + so + we + exp1 + exp2 +
# exp3) + (widowed + divorced + separated + nevermarried + hsd08 + hsd911 + hsg␣
↪+
# cg + ad + mw + so + we + exp1 + exp2 + exp3)^2, data = cps2012, I = ~female +
# female:(widowed + divorced + separated + nevermarried + hsd08 + hsd911 + hsg +
# cg + ad + mw + so + we + exp1 + exp2 + exp3))
```

[70]:
```
## 44
url = "https://github.com/cran/hdm/raw/master/data/GrowthData.rda";
GrowthData = load(download(url))["GrowthData"];
y = GrowthData[:, 1];
# dim(GrowthData)

d = GrowthData[:, 3];
X =  Matrix(GrowthData[:, Not(1, 2, 3)]);
X1 = Matrix(GrowthData[:, Not(1, 2)]);
# GrowthData[1:4, :]
```

[86]:
```
# xnames = varnames[-c(1, 2, 3)]
# names of X variables
# dandxnames = varnames[-c(1, 2)]
# names of D and X variables
# create formulas by pasting names (this saves typing times)
# fmla = asformula(paste("Outcome ~ ", paste(dandxnames, collapse = "+")))
lseffect = lm(intercept(X1), y);
```

[90]:
```
size(X)
```

[90]: (90, 60)

[89]:
```
dX = hcat(d, X)
# dX = asmatrix(cbind(d, X))

lassoeffect = HDMjl.rlassoEffect(X, y, d, method = "partialling out")
HDMjl.r_summary(lassoeffect)
```

Estimates and significance testing of the effect of target variables

Row	Estimate.	Std. Error	t value	Pr(>|t|)
1	-0.05333	0.0143283	-3.722	0.000197655

```
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

[89]:

	Estimate.	Std. Error	t value	Pr(>|t|)
	Float64	Float64	Float64	Float64
1	-0.05333	0.0143283	-3.722	0.000197655

[93]:
```
# dX = asmatrix(cbind(d, X))
doubleseleffect = HDMjl.rlassoEffect(X, y, d, method = "double selection")
HDMjl.r_summary(doubleseleffect)
```

```
Estimates and significance testing of the effect of target variables
  Row    Estimate.    Std. Error    t value

Pr(>|t|)


    1   -0.0453558     0.018656   -2.43116   0.0150506
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

[93]:

	Estimate.	Std. Error	t value	Pr(>|t|)
	Float64	Float64	Float64	Float64
1	-0.0453558	0.018656	-2.43116	0.0150506

[29]:
```
# library(xtable)
# table = rbind(summary(lseffect)$coef["gdpsh465", 1:2],␣
 ↪summary(lassoeffect)$coef[,
# 1:2], summary(doubleseleffect)$coef[, 1:2])
# colnames(table) = c("Estimate", "Std Error")
# #names(summary(fullfit)£coef)[1:2]
# rownames(table) = c("full reg via ols", "partial reg
# via post-lasso ", "partial reg via double selection")
# tab = xtable(table, digits = c(2, 2, 5))
# tab
```

A xtable: 3 × 2

	Estimate <dbl>	Std. Error <dbl>
full reg via ols	-0.009377989	0.02988773
partial reg via post-lasso	-0.049811465	0.01393636
partial reg via double selection	-0.050005855	0.01579138

1.6 Inference on Treatment Effects in a Hight-Dimensional Setting

[100]:
```
##51
url = "https://github.com/cran/hdm/raw/master/data/AJR.rda";
AJR = load(download(url))["AJR"];
y = AJR.GDP
d = AJR.Exprop
```

```
z = AJR.logMort
X = AJR[:, ["Latitude", "Latitude2", "Africa", "Asia", "Namer", "Samer"]]
X = Matrix(X)
# # dim(GrowthData)

# d = GrowthData[:, 3];
# X =  Matrix(GrowthData[:, Not(1, 2, 3)]);
# X1 = Matrix(GrowthData[:, Not(1, 2)]);
# data(AJR)
# y = AJR$GDP
# d = AJR$Exprop
# z = AJR$logMort
# x = modelmatrix(~-1 + (Latitude + Latitude2 + Africa + Asia + Namer +␣
↪Samer)^2,
# data = AJR)
# dim(x)
# AJR
```

[ ]:
```
AJRXselect = HDMjl.rlassoIV(X, d, y, z,  select_Z = false)
# summary(AJRXselect)
```

[32]:
```
confint(AJRXselect)
```

```
           2.5 %   97.5 %
Exprop 0.3159812 1.374072
```

[144]:
```
# x_formula = @formula(lnw ~ -1 + widowed + divorced + separated + nevermarried␣
↪+
# hsd08 + hsd911 + hsg + cg + ad + mw + so + we + exp1 + exp2 + exp3)

# x_dframe = ModelFrame( x_formula, cps2012)
# x1 = ModelMatrix(x_dframe)
# x = x1.m
# y = cps2012[:,"lnw"];
AJR = AJR
x_fmla = @formula(GDP ~ (Latitude + Latitude2 + Africa + Asia + Namer + Samer))
x_dframe = ModelMatrix(ModelFrame(x_fmla, AJR))
y = AJR.GDP
d = AJR.Exprop
z = AJR.logMort;
```

[137]:
```
rY = residuals(lm(intercept(X), y))
rd = residuals(lm(intercept(X), d))
rz = residuals(lm(intercept(X), z));
```

[138]:
```
HDMjl.tsls(rd, rY, rz, nothing)
```

[138]:

```
Dict{String, Any} with 5 entries:
  "se"           => [0.442582, 0.125925]
  "sample_size"  => 64
  "vcov"         => [0.195879 -1.33729e-16; -1.18249e-16 0.0158572]
  "residuals"    => [0.34521, 0.880337, 1.02393, -0.582899, -0.318495, 0.911684…
  "coefficients" => [1.08794, 3.70182e-16]
```

[33]:

```
# fmlay = GDP ~ (Latitude + Latitude2 + Africa + Asia + Namer + Samer)^2
# fmlad = Exprop ~ (Latitude + Latitude2 + Africa + Asia + Namer + Samer)^2
# fmlaz = logMort ~ (Latitude + Latitude2 + Africa + Asia + Namer + Samer)^2
# rY = lm(fmlay, data = AJR)$res
# rD = lm(fmlad, data = AJR)$res
# rZ = lm(fmlaz, data = AJR)$res
# # ivfitlm = tsls(y=rY,d=rD, x=NULL, z=rZ, intercept=false)
# ivfitlm = tsls(rY ~ rD | rZ, intercept = false)
# print(cbind(ivfitlm$coef, ivfitlm$se), digits = 3)
```

```
   [,1] [,2]
rD 1.27 1.73
```

[151]:

```
rY = HDMjl.rlasso(X, y)["residuals"]
rD = HDMjl.rlasso(X, d)["residuals"]
rZ = HDMjl.rlasso(X, z)["residuals"]
HDMjl.tsls(rd, rY, rz, nothing)
```

[151]:

```
Dict{String, Any} with 5 entries:
  "se"           => [0.45288, 0.128855]
  "sample_size"  => 64
  "vcov"         => [0.2051 -1.40025e-16; -1.23816e-16 0.0166037]
  "residuals"    => [0.27765; 0.888344; … ; -0.423318; 2.25992;;]
  "coefficients" => [1.08794; -6.79326e-16;;]
```

[35]:

```
data(EminentDomain)
z = asmatrix(EminentDomain$logGDP$z)
x = asmatrix(EminentDomain$logGDP$x)
y = EminentDomain$logGDP$y
d = EminentDomain$logGDP$d
x = x[, apply(x, 2, mean, narm = true) > 005]
#
z = z[, apply(z, 2, mean, narm = true) > 005]
#
```

[36]:

```
EDols = lm(y ~ cbind(d, x))
ED2sls = tsls(y = y, d = d, x = x, z = z[, 1:2], intercept = false)
```

[37]:

```
lassoIVZ = rlassoIV(x = x, d = d, y = y, z = z, selectX = false, selectZ = true)
# or lassoIVZ = rlassoIVselectZt(x=X, d=d, y=y, z=z)
summary(lassoIVZ)
```

```
[1] "Estimates and significance testing of the effect of target variables in the
IV regression model"
   coeff.    se. t-value p-value
d1 0.4146 0.2902   1.428   0.153
```

[38]:
```
confint(lassoIVZ)
```

```
        2.5 %    97.5 %
d1 -0.1542764 0.9834796
```

[39]:
```
lassoIVXZ = rlassoIV(x = x, d = d, y = y, z = z, selectX = true, selectZ = true)
summary(lassoIVXZ)
```

```
Estimates and Significance Testing of the effect of target variables in the IV
regression model
     coeff.      se. t-value p-value
d1 -0.02383  0.12851  -0.185   0.853
```

[40]:
```
confint(lassoIVXZ)
```

```
        2.5 %    97.5 %
d1 -0.2757029 0.2280335
```

[41]:
```
library(xtable)
table = matrix(0, 4, 2)
table[1, ] = summary(EDols)$coef[2, 1:2]
table[2, ] = cbind(ED2sls$coef[1], ED2sls$se[1])
table[3, ] = summary(lassoIVZ)[, 1:2]
```


```
[1] "Estimates and significance testing of the effect of target variables in the
IV regression model"
   coeff.    se. t-value p-value
d1 0.4146 0.2902   1.428   0.153
```


[42]:
```
table[4, ] = summary(lassoIVXZ)[, 1:2]
```


```
Estimates and Significance Testing of the effect of target variables in the IV
regression model
     coeff.      se. t-value p-value
d1 -0.02383  0.12851  -0.185   0.853
```

[43]:
```
colnames(table) = c("Estimate", "Std Error")
rownames(table) = c("ols regression", "IV estimation ", "selection on Z",␣
  ↪"selection on X and Z")
tab = xtable(table, digits = c(2, 2, 7))
tab
```

A xtable: 4 × 2

	Estimate <dbl>	Std. Error <dbl>
ols regression	0.007864732	0.009865927
IV estimation	-0.010733269	0.033766362
selection on Z	0.414601641	0.290249208
selection on X and Z	-0.023834697	0.128506538

[44]:
```
data(pension)
y = pension$tw
d = pension$p401
z = pension$e401
X = pension[, c("i2", "i3", "i4", "i5", "i6", "i7", "a2", "a3", "a4", "a5",␣
  ↪"fsize",
"hs", "smcol", "col", "marr", "twoearn", "db", "pira", "hown")]
# simple model
xvar = c("i2", "i3", "i4", "i5", "i6", "i7", "a2", "a3", "a4", "a5", "fsize",␣
  ↪"hs",
"smcol", "col", "marr", "twoearn", "db", "pira", "hown")
xpart = paste(xvar, collapse = "+")
form = asformula(paste("tw ~ ", paste(c("p401", xvar), collapse = "+"), "|",␣
  ↪paste(xvar,
collapse = "+")))
formZ = asformula(paste("tw ~ ", paste(c("p401", xvar), collapse = "+"), "|",␣
  ↪paste(c("e401",
xvar), collapse = "+")))
```

[45]:
```
pensionate = rlassoATE(form, data = pension)
summary(pensionate)
```

```
Estimation and significance testing of the treatment effect
Type: ATE
Bootstrap: not applicable
   coeff.   se. t-value  p-value
TE  10180  1931   5.273 1.34e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

[46]:
```
pensionatet = rlassoATET(form, data = pension)
summary(pensionatet)
```

```
Estimation and significance testing of the treatment effect
Type: ATET
Bootstrap: not applicable
   coeff.   se. t-value p-value
TE  12628  2944   4.289 1.8e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

1.6.1 Error

[53]:
```
# pensionlate = rlassoLATE(X, d, y, z)
# pensionlate = rlassoLATE(formZ, data=pension)
# summary(pensionlate)
```

[48]:
```
# pensionlatet = rlassoLATET(X, d, y, z)
```

[49]:
```
xvar2 = paste("(", xvar, ")^2", sep = "")
formExt = asformula(paste("tw ~ ", paste(c("p401", xvar2), collapse = "+"), "|",
paste(xvar2, collapse = "+")))
formZExt = asformula(paste("tw ~ ", paste(c("p401", xvar2), collapse = "+"),␣
 ↪"|",
paste(c("e401", xvar2), collapse = "+")))
```

[50]:
```
pensionate = rlassoATE(X, z, y)
pensionatet = rlassoATET(X, z, y)
# pensionlate = rlassoLATE(X, d, y, z)
# pensionlatet = rlassoLATET(X, d, y, z)
```

1.7 The Lasso Methods for Discovery of Significant Causes amongst Many Potential Causes, with Many Controls

[54]:
```
setseed(1)
n = 100
p1 = 20
p2 = 20
D = matrix(rnorm(n * p1), n, p1)
# Causes
W = matrix(rnorm(n * p2), n, p2)
X = cbind(D, W)
# Regressors
Y = D[, 1] * 5 + W[, 1] * 5 + rnorm(n)
#Outcome
confint(rlassoEffects(X, Y, index = c(1:p1)), joint = true)
```

A matrix: 20 × 2 of type dbl

	2.5 %	97.5 %
V1	4.5145877	5.21430498
V2	-0.3142909	0.30494650
V3	-0.3524109	0.18678880
V4	-0.2542430	0.28738914
V5	-0.2765802	0.27627177
V6	-0.3214676	0.29422684
V7	-0.2262507	0.30094168
V8	-0.0473541	0.47366372
V9	-0.1865636	0.39023520
V10	-0.2372356	0.26411185
V11	-0.3147091	0.20945872
V12	-0.3091905	0.26572176
V13	-0.1741550	0.37682465
V14	-0.3235734	0.38543162
V15	-0.3219763	0.31312486
V16	-0.2649505	0.33100700
V17	-0.1792080	0.41696169
V18	-0.3693247	0.04695928
V19	-0.1073109	0.39368776
V20	-0.2157182	0.25543839