PARSIMONY

$A, B \;:\; |corr(A, B)| > 0$

collinearity

multi-collinearity

$f : [x_1, x_2, x_3 \ldots] \longrightarrow \hat{y}$

$|corr(x_i, x_j)| > 0$

$x_i, x_j \in \{x_1, x_2, \ldots\}$

NOT DESIRABLE

collinear

$X \longrightarrow Y$

$X_1 \sim X_2$ or $X_3 \sim X_5$

PARSIMONY: model has the required no: of predictor number, and no less WITH maximum predictive power

> 60, 75%!

(i) avoid collinear variables

(ii) simplest model with highest predictive power

(iii) controls

→ Occam's razor

among multiple competing HAs, choose the one which makes or uses less assumptions / variable,

INFERENCE FOR LR

$$y = m_1 x_1 + m_2 x_2 + \dots + c$$

$H_0 = m_1 = m_2 = m_3 \dots = 0$

H_A = atleast one $m_i \neq 0$

python — OLS — R

statsmodel.OLS — lm, glm

model.fit().summary()
.t_test()

summary(model)

F-statistic = 29.74 on 4 and 429 DF
p-value < 2.2e-16

number of predictors in model = 4
number of observations in the model = 434
degrees of freedom (DF) = n - k - 1
= 434 - 4 - 1
= 429

if $p < \alpha$ (1%, 5%, 10%)
model is actually significant
if $p \geq \alpha$, model is not significant
the specific combination of variables used is NOT a good combo.

T-test $y = m_1(x_1) + m_2 x_2 \cdots + c$

$$t_{x_1} = \frac{m_1 - 0}{\text{standard error}}$$

$$SE = \frac{\sigma}{\sqrt{n}}$$

$df = n - k - 1$

scipy.stats

$pt(t_{x_1}, df, lower.tail = FALSE) * 2$

$= p\text{-value}.$

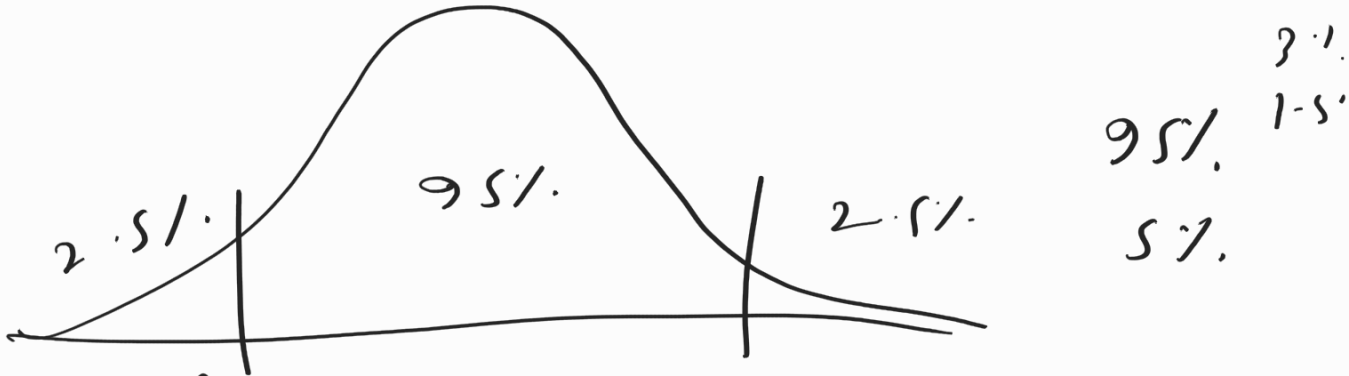
$p < \alpha,$

confidence interval : $m_1 \pm (t_{x_1} * SE)$

$$\Downarrow$$

$$\left[m_1 - (t_{x_1} * SE),\ m_1 + (t_{x_1} * SE) \right]$$

95%.



$qt(0.025, df = 429) = -1.97$

$n - k - 1$

$t_{x_1} = \pm 1.97$

95% CI for X_1 = (−2.09, 7.17)

We are 95% confident that all else being equal, this model predicts that (y w.r.t X_1) scores 2.09 points lower / 7.17 points higher than [opp of X_0 compliment of X_1]

```
>>> import statsmodels.api as sm
>>> import numpy as np
>>> duncan_prestige = sm.datasets.get_rdataset("Duncan", "carData")
>>> Y = duncan_prestige.data['income']
>>> X = duncan_prestige.data['education']
>>> X = sm.add_constant(X)
>>> model = sm.OLS(Y,X)
>>> results = model.fit()
>>> results.params
const        10.603498
education     0.594859
dtype: float64
```

```
>>> results.tvalues
const        2.039813
education    6.892802
dtype: float64
```

```
>>> print(results.t_test([1, 0]))
                             Test for Constraints
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
c0            10.6035      5.198      2.040      0.048       0.120      21.087
==============================================================================
```

```
>>> print(results.f_test(np.identity(2)))
<F test: F=array([[159.63031026]]), p=1.2607168903696672e-20, df_denom=43, df_num=2>
```

```
summary(lmodel)

Call:
lm(formula = sqrt(mpg) ~ sqrt(disp), data = mtcars)

Residuals:
 Min 1Q Median 3Q Max
-0.45591 -0.21505 -0.07875 0.16790 0.71178

Coefficients:
 Estimate Std. Error t value Pr(>|t|)
(Intercept) 6.51921 0.19921 32.73 < 2e-16 ***
sqrt(disp) -0.14246 0.01312 -10.86 6.44e-12 ***
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.3026 on 30 degrees of freedom
Multiple R-squared: 0.7973, Adjusted R-squared: 0.7905
F-statistic: 118 on 1 and 30 DF, p-value: 6.443e-12
```

STEPWISE REGRESSION

(y)

$x_1\ x_2\ x_3\ x_4$

p-value (f-statistic)
R^2, adjusted R^2,
AIC, BIC, Cp
mallow's cp

backwards elimination

(i) $y = m_1x_1 + m_2x_2 + m_3x_3 + m_4x_4 + c$

E_1

(ii) $y = m_1x_1 + m_2x_2 + m_3x_3 + c$

E_2

(iii) $y = m_1x_1 + m_2x_2 + c$

E_3

(iv) $y = m_1x_1 + c$

E_4

(v) $y = c$

E_5

forward selection

(i) $y = c$

E_1

(ii) $y = m_1x_1 + c$

E_2

(iii) $y = m_1x_1 + m_2x_2 + c$

E_3

⋮