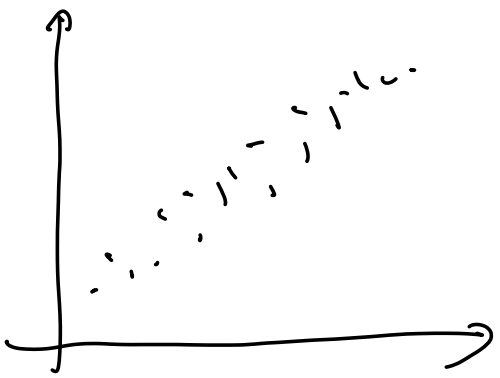
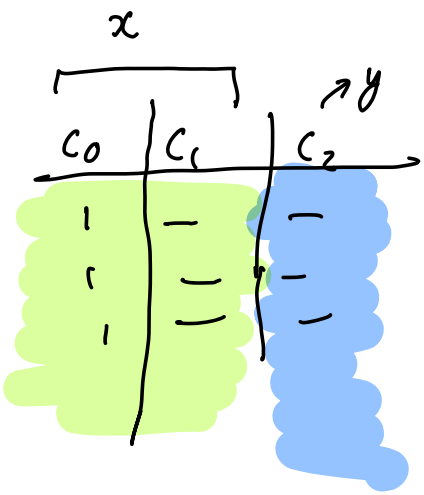
Polynomial Regression

⇒ Multi-col ⇒ VIF → ↑ ↗ → bad col
↓ → good col

⇒ SGD / mini-Batch GD





→ predict(X, weight)
np.dot(x, weight) $w^T \cdot x$

→ gradient(X, y, weight)
y-pred = ____
grad

$x\,(y - y_{pred})$
=
np.dot(x, y-ypred)
↓
x.transpose()

$w^T x$
$x^T w$
np.dot(w, x)
np.dot(x, w)

```
def cost (x, y, weight):
    y_pred
    e =
```

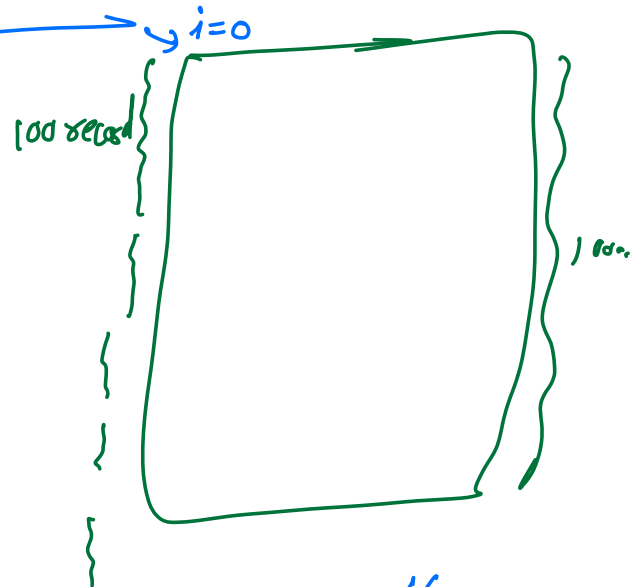
```
def create_mini_batches (x, y, batch_size):
    data = hstack(x, y)
    np.random.shuffle(data)
    count =
```

for → 10 times

i * batch : (i+1) * batch

0 * 100 : (1) * batch

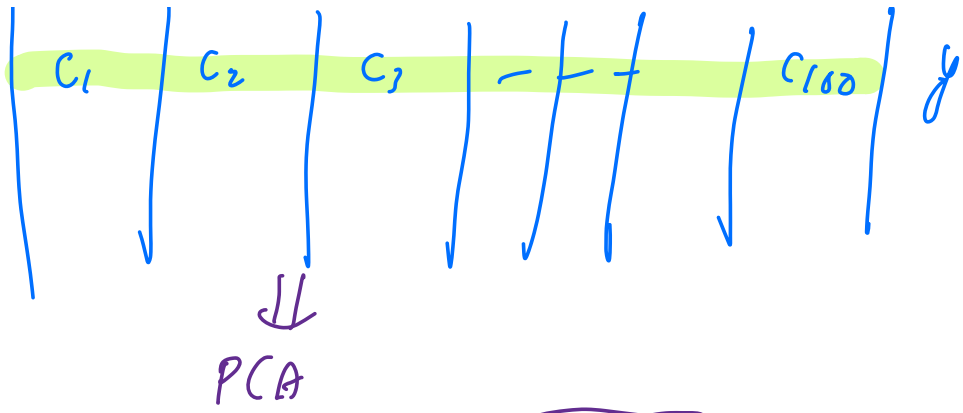
0 : 100



1100 // 100

100 - 199 ⇒

RFE



RFE ⇒

↓

10 columns

↓

p-value

multi

100 columns ⇒ LR

↓

R^2 Adj R^2

↓

15 col.

Adj R^2 = 0.8

↓

10 col

Adj R^2 = 0.9

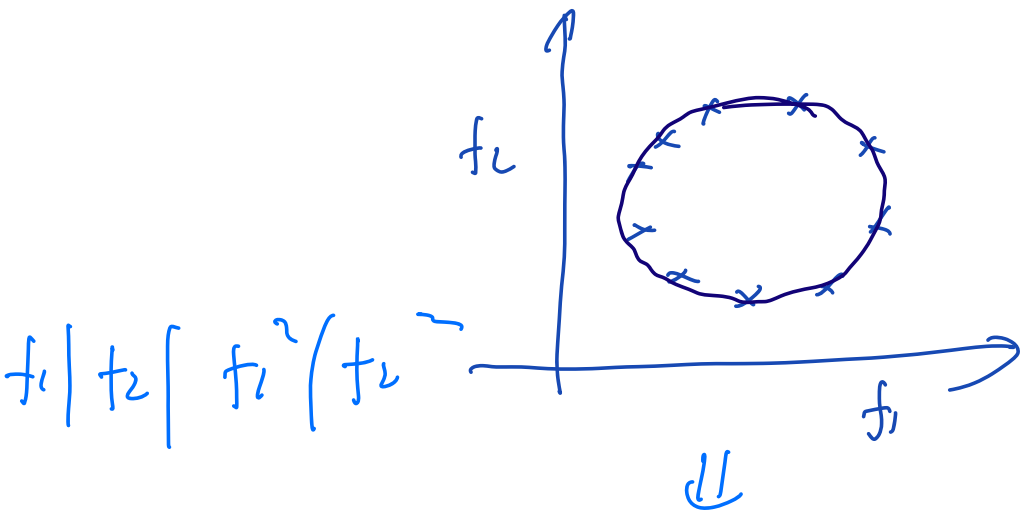
↓

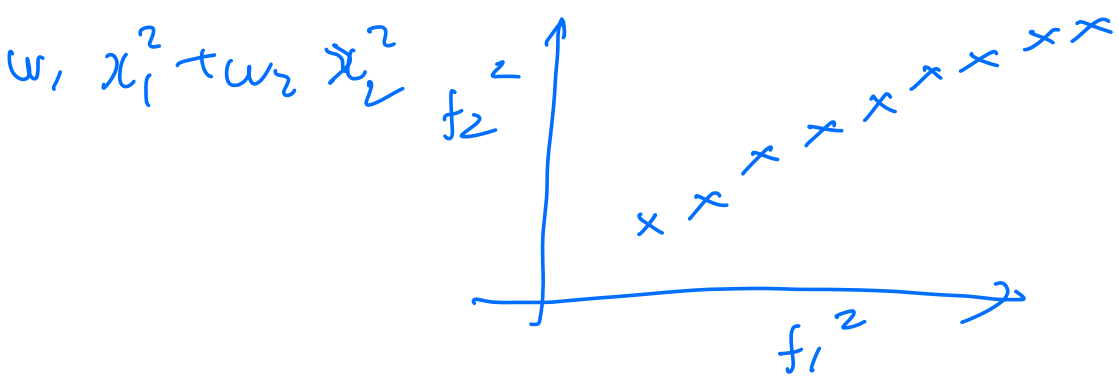
8 col

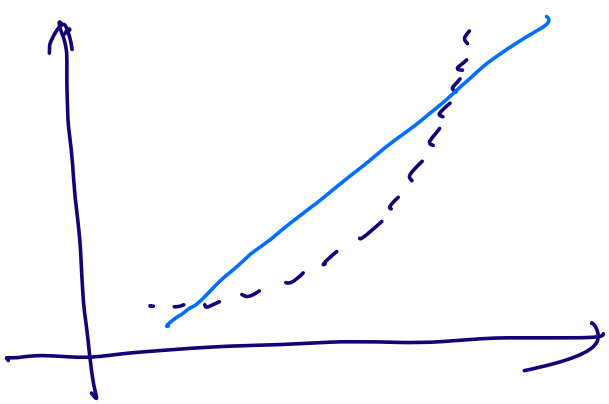
Adj R^2 = 0.86

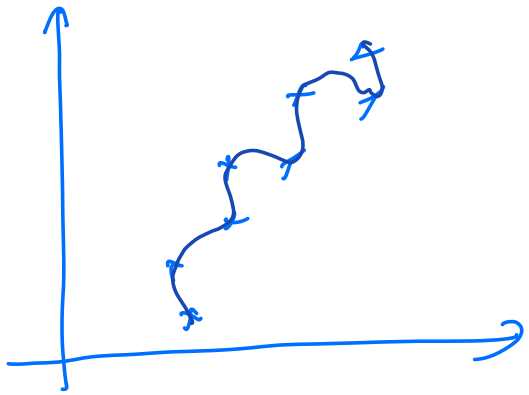
Accuracy: 12:05

Polynomial Regression

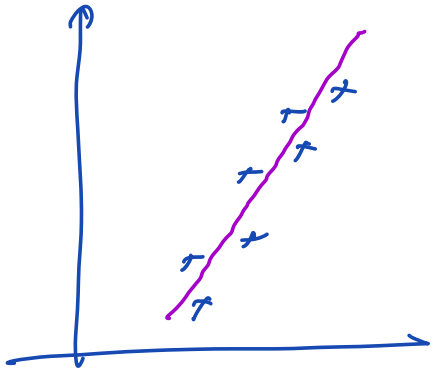








f_1, f_1^2, f_1^3



f_1

$R^2 \to 90\%$

① Generalization

test → R^2

② Occum rajor

Linear : f_1

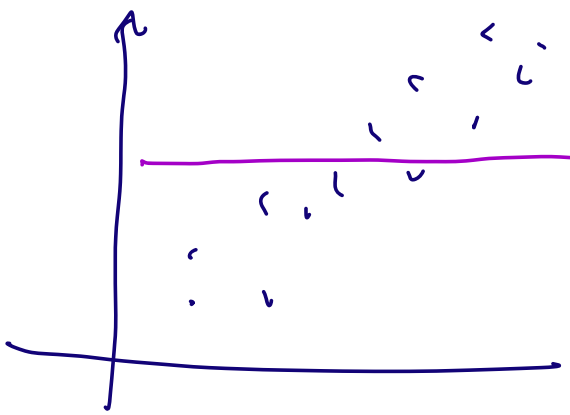
Quard : f_1, f_1^2

cubic : f_1, f_1^2, f_1^3

Occ	L	Q	C
Test err	H	L	L

Train ↑
Test ↓ ⇒ overfitting

Train ↓
Test ↓ ⇒ underfit



90 → 93
78
δ → 91
→ 82

		D_{T_2}	D_{T_1}
underfit	→	100	120
✓	⇒	20	30
overfit	→	0.1	50

$$y_i = w_0 + w_1 x_1 + w_2 x_2^2 + w_3 x_3^3 + w_4 x_4^4$$

$w_1 + w_2 + w_3 + w_4 \neq 0$

overfit

w_2