immerse
EDUCATION

# Polynomial Fitting

Elizabeth Savochkina | 11th June

# Shape of a Day

| |
|---|
| **Registration**<br>9.00am to 9.30am |
| **Polynomial Fitting**<br>**+ RapidMiner**<br>9.30am to 11.30am |
| **Break 11.30am to 13.30 pm** |
| **Regression Workshop**<br>13.30pm to 15.00pm |
| **Work on Capstone Project**<br>15.00pm to 15.30pm |

# Machine Learning Fundamentals

Tuning your Machine

Shape matters

Making it easier to get fit

## How to get fit

How do algorithms actually fit models to data?

Let's ROC

Cakeology

## How to get Fat

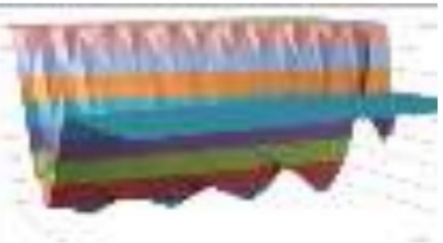
Working in higher-dimensions

## Time to choose

More about classification

Welcome to the Matrix

Some hills are too tough

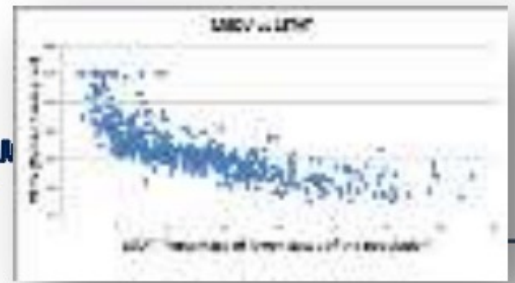


## Concerned about curves

Beyond straight-lines

Reducing others

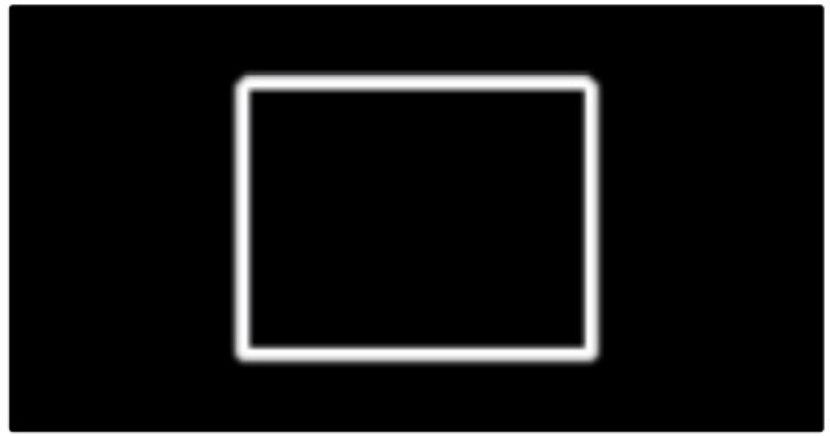
Some curves are useful



# Thinking in Higher Dimensions

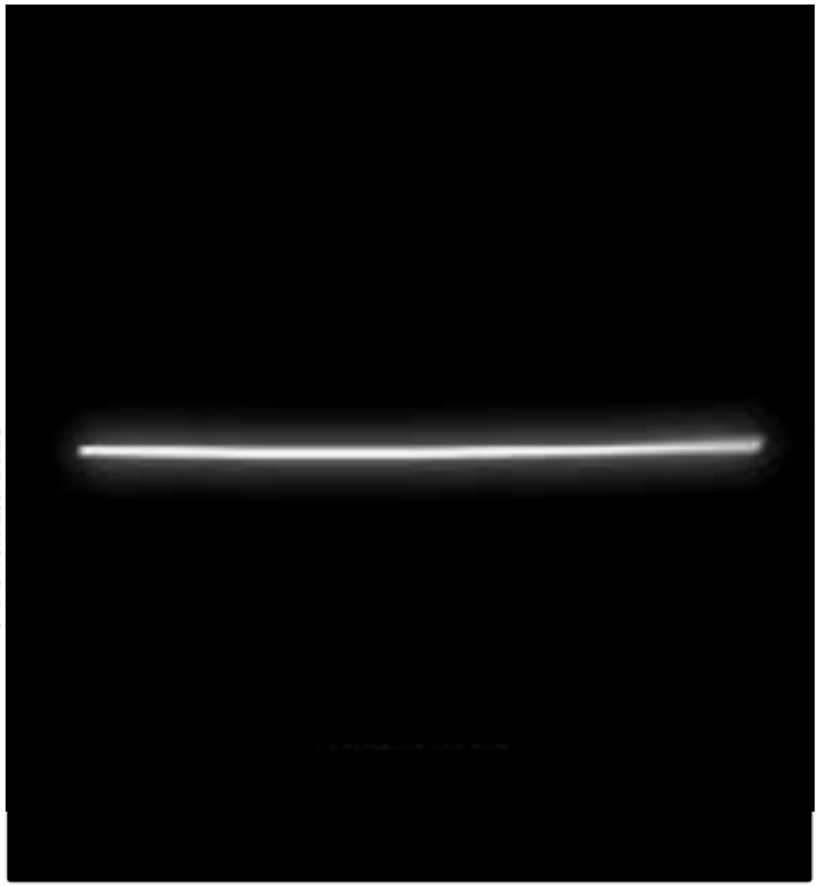
## 1-Dimension
Length of a line

## 2-Dimensions
Area of a Square



## 3-Dimensions
Volume of a Cube

# 4 Dimensional Hyper-Cube?



**I find it hard to imagine anything above 3 dimensions as an object**

**…so I don't try to!**

I can, however,
imagine cake ..

And I frequently do..

# The Science of Cake

# Collins Cakeology Index

**I have developed a new scoring system for cake – the 'Collins Cakeology Index' (CCI)**

**An experimental measure of the impact on body mass when consuming cake**

| Observation | Flour (g) | Eggs (g) | Sugar (g) | Butter (g) | Chocolate (g) | Bicarbonate (g) | Collins Cakeology Index |
|---|---|---|---|---|---|---|---|
| 1 | 1310 | 4 | 385 | 95 | 41 | 4 | 5998 |
| 2 | 1263 | 10 | 1182 | 174 | 71 | 2 | 3385 |
| 3 | 261 | 5 | 675 | 151 | 100 | 8 | 1038 |
| 4 | 890 | 16 | 938 | 111 | 85 | 2 | 3133 |
| 5 | 1229 | 16 | 759 | 263 | 52 | 8 | 5002 |
| 6 | 421 | 3 | 516 | 124 | 10 | 8 | 2458 |
| 7 | 1828 | 13 | 270 | 61 | 94 | 10 | 8421 |
| 8 | 1822 | 7 | 1122 | 79 | 10 | 9 | 6145 |
| 9 | 1443 | 4 | 804 | 260 | 17 | 8 | 5827 |
| 10 | 1324 | 1 | 1273 | 65 | 44 | 10 | 2692 |
| 11 | 1023 | 15 | 1984 | 239 | 22 | 4 | 249 |
| 12 | 1728 | 11 | 1221 | 284 | 19 | 9 | 5190 |
| 13 | 1267 | 12 | 1374 | 216 | 73 | 6 | 3473 |
| 14 | 556 | 9 | 739 | 105 | 56 | 8 | 2166 |
| 15 | 1362 | 5 | 1053 | 283 | 50 | 3 | 4493 |
| 16 | 1239 | 3 | 1118 | 115 | 60 | 1 | 3603 |
| 17 | 934 | 17 | 177 | 287 | 87 | 1 | 5821 |

# Developing a predictive formula for CCI

$$CCI = K + A.Flour + B.Eggs + C.Sugar + D.Butter + E.Chocolate + F.Bicarbonate$$

$$CCI = \theta_0 + \theta_1.Flour + \theta_2.Eggs + \theta_3.Sugar + \theta_4.Butter + \theta_5.Chocolate + \theta_6.Bicarbonate$$

$$CCI = \theta_0.x_0 + \theta_1.x_1 + \theta_2.x_2 + \theta_3.x_2 + \theta_4.x_4 + \theta_5.x_5 + \theta_6.x_6$$

$$CCI = \sum_{0}^{n} \theta_n.x_n$$

This is a linear regression model

Need to set $x_0 = 1$ to make this work

# You could solve this by trial and error ….?

| $\theta_0$ | $\theta_1$ | $\theta_2$ | $\theta_3$ | $\theta_4$ | $\theta_5$ | $\theta_6$ |
|---|---|---|---|---|---|---|
| 50.0 | 10.0 | 960.0 | -20.0 | -70.0 | 30.0 | -60.0 |
| +10 | 10 | 10 | 10 | 10 | 10 | 10 |
| -10 | 10 | 10 | 10 | 10 | 10 | 10 |

Mean Square Error
237,054,800

Collected Data

| Observation | Flour (g) | Eggs (g) | Sugar (g) | Butter (g) | Chocolate (g) | Bicarbonate (g) | Collins Cakeology Index (Y) | Prediction | ε Error | Error^2 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1310 | 4 | 385 | 95 | 41 | 4 | 5998 | 4910 | -1088 | 1184532 |
| 2 | 1263 | 10 | 1182 | 174 | 71 | 2 | 3385 | -10890 | -14275 | 203783936 |
| 3 | 261 | 5 | 675 | 151 | 100 | 8 | 1038 | -11530 | -12568 | 157949701 |
| 4 | 890 | 16 | 938 | 111 | 85 | 2 | 3133 | 850 | -2283 | 5210401 |
| 5 | 1229 | 16 | 759 | 263 | 52 | 8 | 5002 | -2250 | -7252 | 52595683 |
| 6 | 421 | 3 | 516 | 124 | 10 | 8 | 2458 | -9480 | -11938 | 142520201 |
| 7 | 1828 | 13 | 270 | 61 | 94 | 10 | 8421 | 26560 | 18139 | 329030495 |
| 8 | 1822 | 7 | 1122 | 79 | 10 | 9 | 6145 | -340 | -6485 | 42056804 |
| 9 | 1443 | 4 | 804 | 260 | 17 | 8 | 5827 | -13370 | -19197 | 368528099 |
| 10 | 1324 | 1 | 1273 | 65 | 44 | 10 | 2692 | -11840 | -14532 | 211186604 |
| 11 | 1023 | 15 | 1984 | 239 | 22 | 4 | 249 | -30030 | -30279 | 916818077 |
| 12 | 1728 | 11 | 1221 | 284 | 19 | 9 | 5190 | -13500 | -18690 | 349329229 |
| 13 | 1267 | 12 | 1374 | 216 | 73 | 6 | 3473 | -14610 | -18083 | 326998035 |
| 14 | 556 | 9 | 739 | 105 | 56 | 8 | 2166 | -4120 | -6286 | 39519774 |
| 15 | 1362 | 5 | 1053 | 283 | 50 | 3 | 4493 | -20120 | -24613 | 605791955 |
| 16 | 1239 | 3 | 1118 | 115 | 60 | 1 | 3603 | -13030 | -16633 | 276669618 |
| 17 | 934 | 17 | 177 | 287 | 87 | 1 | 5821 | 4950 | -871 | 758464 |

If you had enough time and energy!

# Excel calculates the coefficients to minimise MSE

Excel solves to give the coefficients with the lowest Mean Square Error (MSE)

| | *Coefficients* |
|---|---|
| Intercept | 1939.21 |
| Flour (g) | 4.05 |
| Eggs (g) | 8.26 |
| Sugar (g) | -3.31 |
| Butter (g) | 2.89 |
| Chocolate (g) | -4.69 |
| Bicarbonate (g | 14.36 |

| $\theta_0$ | $\theta_1$ | $\theta_2$ | $\theta_3$ | $\theta_4$ | $\theta_5$ | $\theta_6$ | Mean Square Err |
|---|---|---|---|---|---|---|---|
| 1939.2 | 4.05 | 8.3 | -3.3 | 2.9 | -4.7 | 14.4 | 3,879 |

Collected Data

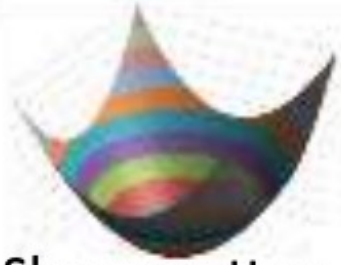
| Observation | Flour (g) | Eggs (g) | Sugar (g) | Butter (g) | Chocolate (g) | Bicarbonate (g) | Collins Cakeology Index (Y) |
|---|---|---|---|---|---|---|---|
| 1 | 283 | 14 | 1590 | 189 | 5 | 9 | -1498 |
| 2 | 697 | 9 | 699 | 121 | 32 | 10 | 2820 |
| 3 | 333 | 20 | 1146 | 95 | 46 | 0 | -285 |
| 4 | 125 | 6 | 1709 | 127 | 99 | 1 | -3197 |
| 5 | 778 | 17 | 225 | 160 | 60 | 4 | 4740 |
| 6 | 1531 | 7 | 1573 | 185 | 75 | 10 | 3252 |
| 7 | 1231 | 10 | 1496 | 220 | 75 | 6 | 2390 |
| 8 | 860 | 19 | 1483 | 188 | 68 | 3 | 996 |
| 9 | 239 | 14 | 1729 | 211 | 92 | 4 | -2543 |
| 10 | 886 | 4 | 366 | 98 | 71 | 5 | 4382 |
| 11 | 1932 | 6 | 1368 | 139 | 26 | 4 | 5689 |
| 12 | 584 | 1 | 928 | 259 | 42 | 7 | 1901 |
| 13 | 1294 | 14 | 628 | 224 | 23 | 2 | 5790 |
| 14 | 484 | 9 | 1371 | 113 | 81 | 10 | -343 |
| 15 | 1533 | 1 | 927 | 230 | 67 | 9 | 5615 |
| 16 | 1552 | 20 | 1153 | 286 | 18 | 2 | 5395 |
| 17 | 1725 | 6 | 714 | 104 | 89 | 0 | 6386 |

# .. Giving me a formula to predict CCI !

| | *Coefficients* |
|---|---|
| Intercept | 1939.21 |
| Flour (g) | 4.05 |
| Eggs (g) | 8.26 |
| Sugar (g) | -3.31 |
| Butter (g) | 2.89 |
| Chocolate (g) | -4.69 |
| Bicarbonate (g | 14.36 |

$$CCI = 1931.21 + 4.05.Flour + 8.26.Eggs - 3.31.Sugar + 2.89.Butter - 4.69.Chocolate + 14.36.Bicarbonate$$

Tuning your Machine



Shape matters

Making it easier to get fit

Let's ROC

## How to get fit

How do algorithms actually fit models to data?

Cakeology

# Machine Learning Fundamentals

## Time to choose

More about classification

## How to get Fat

Working in higher-dimensions

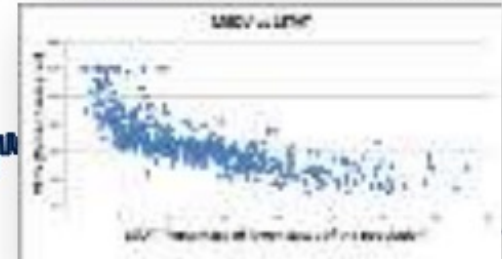
Some hills are too tough

## Concerned about curves

Beyond straight-lines

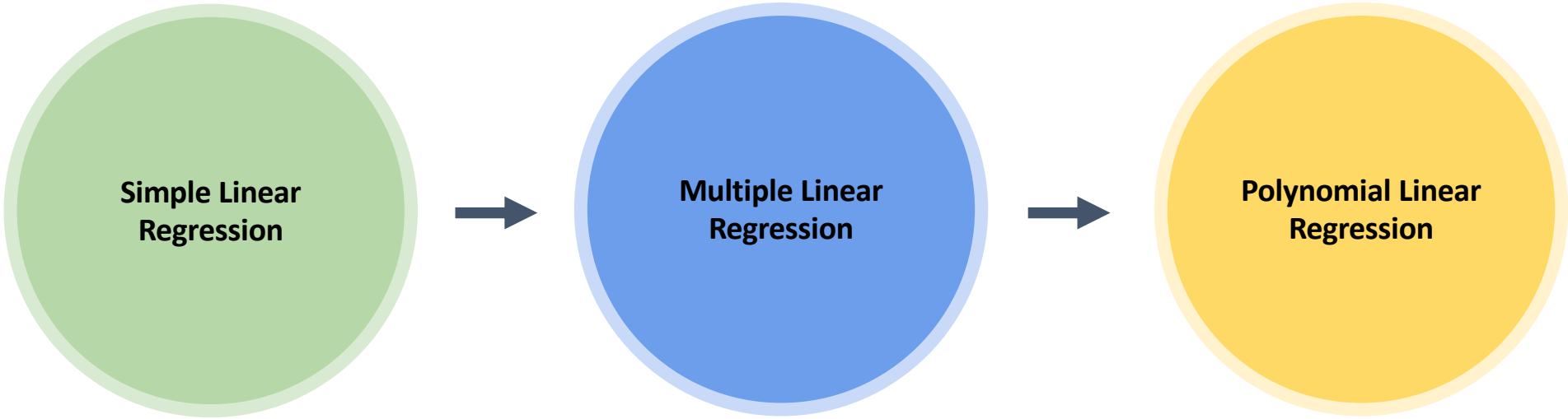
Reducing others


Some curves are useful



Welcome to the Matrix

# Linear Regression Session Outline



# Purpose

Regressions attempts to determine ***the strength*** and ***character of the relationship*** between one dependent variable (usually denoted by Y) and a series of other variables (known as independent variables).

Examples:

- Forecasting sales
- Cash forecasting
- Analysing survey data
- Stock predictions
- Predicting consumer behaviour
- Analysis of relationship between variables

# Linear Regression

**Simple Linear Regression**

$$y = b_0 + b_1 * x_1$$

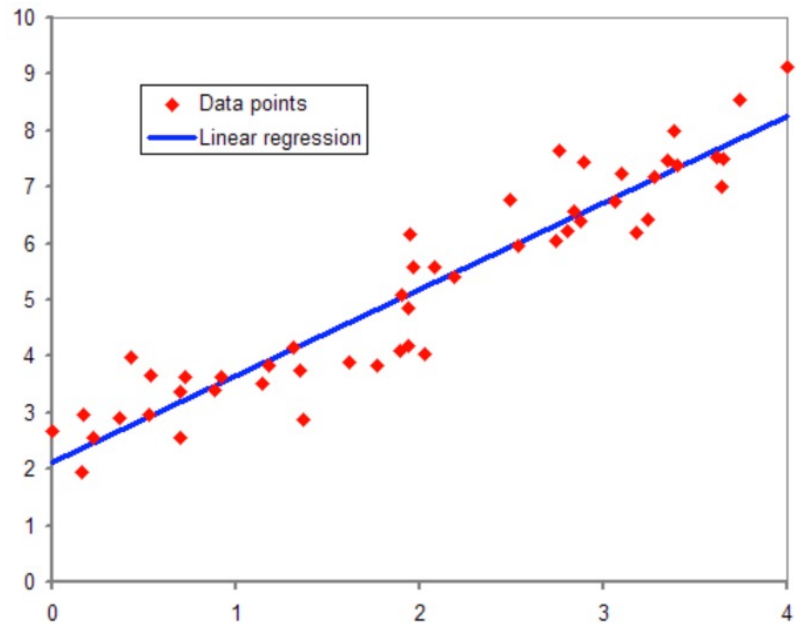

- Predict the value of *y* based on another variable *x*
- *x* - independent variable, *y* - dependent variable

Why do we call it simple?

- We examine the relationship between 2 variable only

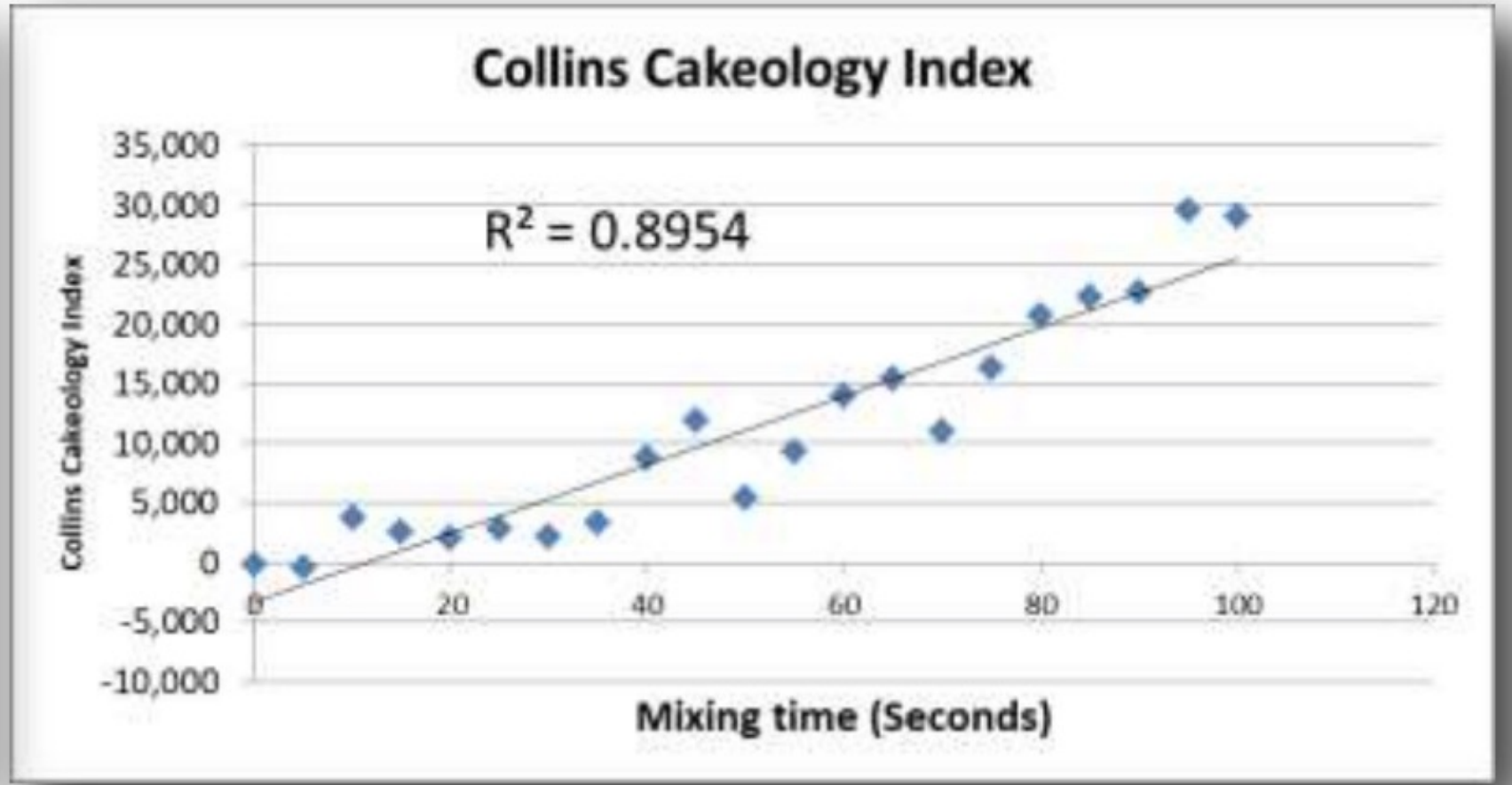
Why is it linear?

- As the independent variable increases/decreases, the dependent variable is increasing/or decreasing in a linear fashion.



# More cake experiments!

| Mixing Time | Collins Cakeology Index (CCI) |
|---|---|
| 0 | -173 |
| 5 | -447 |
| 10 | 3734 |
| 15 | 2548 |
| 20 | 2072 |
| 25 | 2783 |
| 30 | 2164 |
| 35 | 3257 |
| 40 | 8704 |




...

# Multiple Linear Regression

Multiple Linear Regression

$$y = b_0 + b_1 x_1 + b_2 x_2 + \ldots + b_n x_n$$

- Examine the relationships between more than 2 variables
- We aim to fit some weights (coefficients $b_1$, $b_2$) to some input data

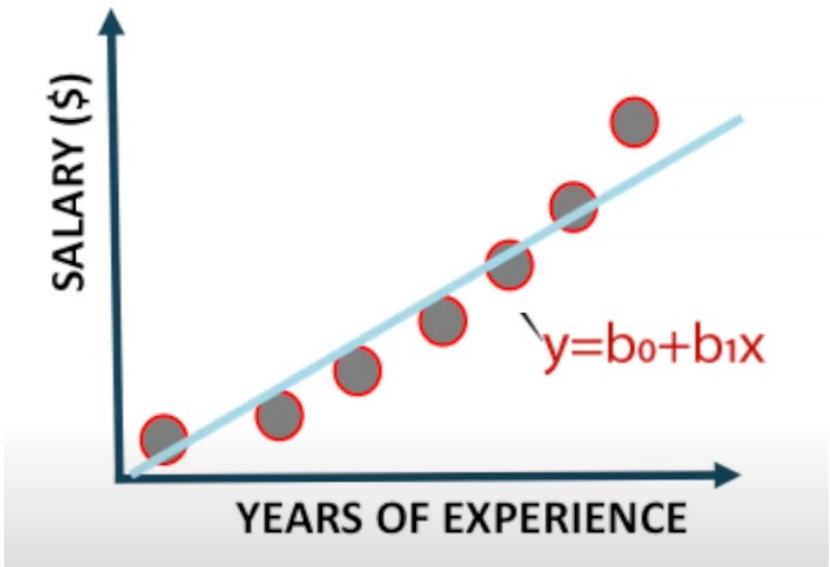
*Salary = b + $m_1$*(number of years of experience) + $m_2$*(number of years of education)*

# Polynomial Linear Regression

Polynomial Linear Regression

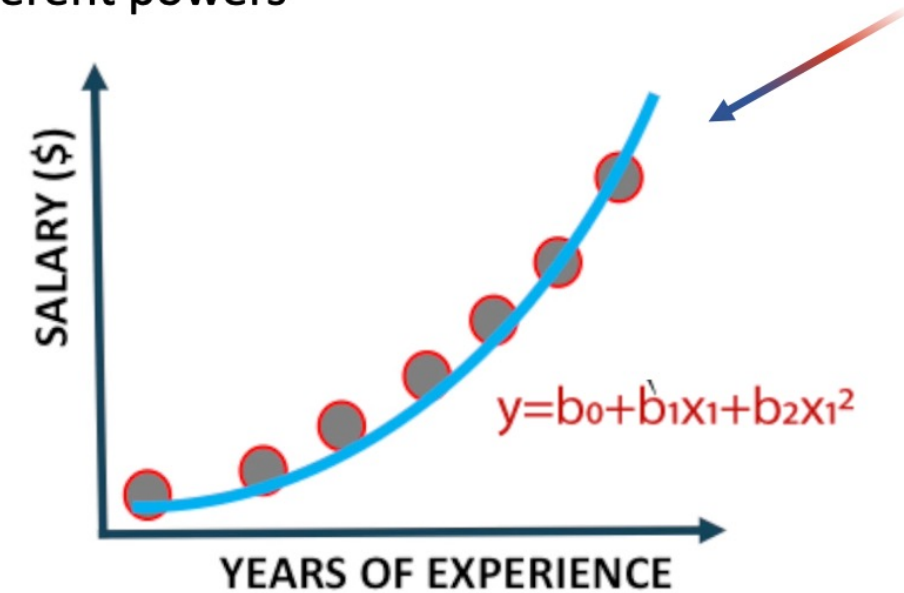
$$y = b_0 + b_1 x_1 + b_2 x_1^2 + \ldots + b_n x_1^n$$

- The relationship between between the *Independent variable (X)* and the *dependent variable (Y),* as an Nth degree polynomial in x.
- We have the same variables $x_1$ but it is raised to different powers



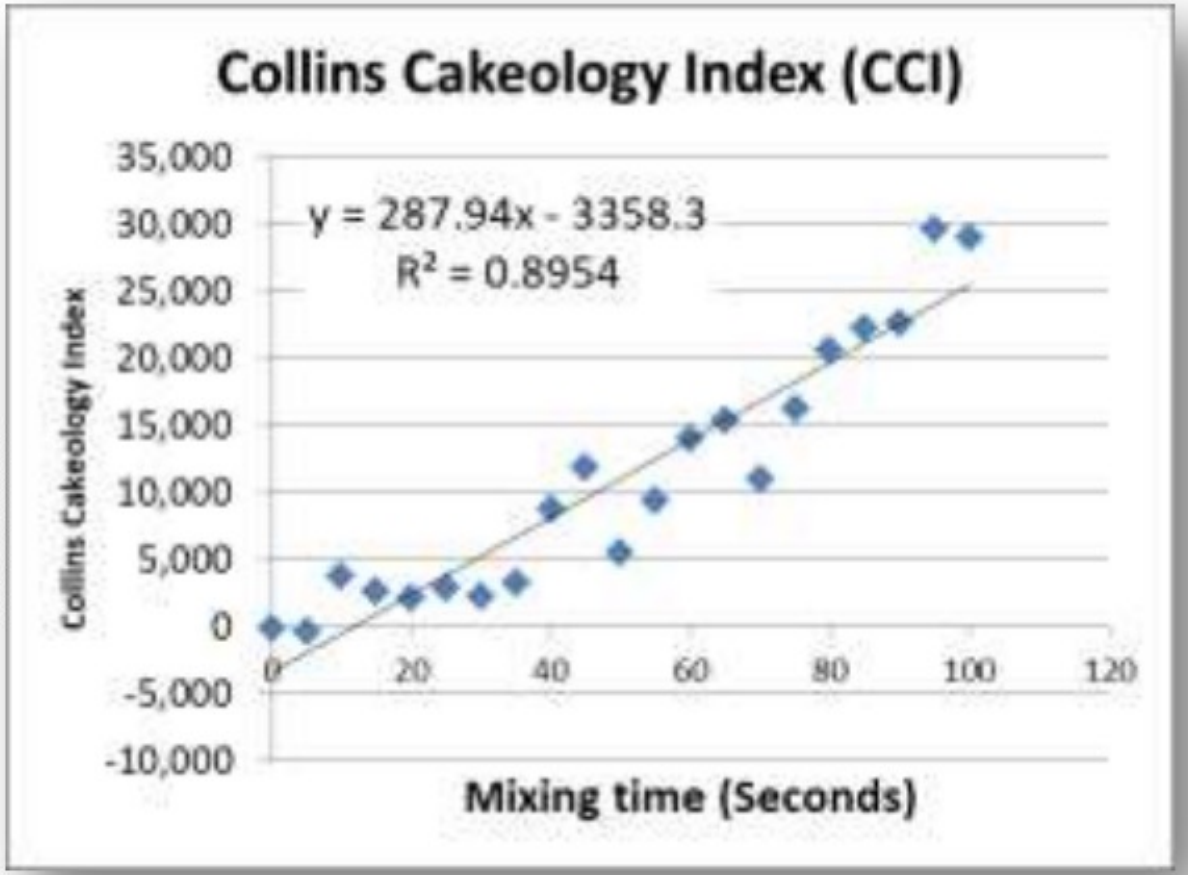

Linear Regression

A much better fit!

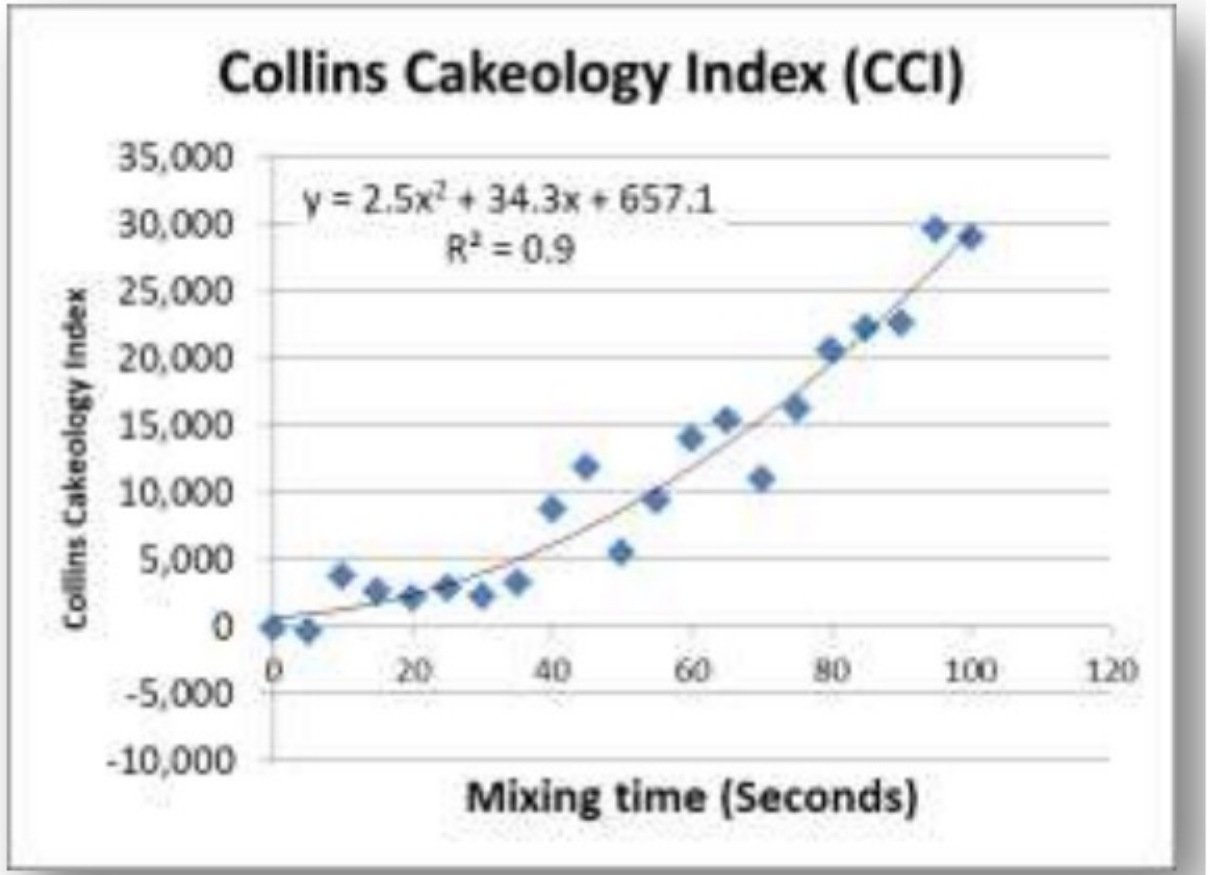



Polynomial Linear Regression

# A more complex model?




**Linear Model**




**Quadratic ($X^2$) Model**

# Summary

Simple Linear Regression

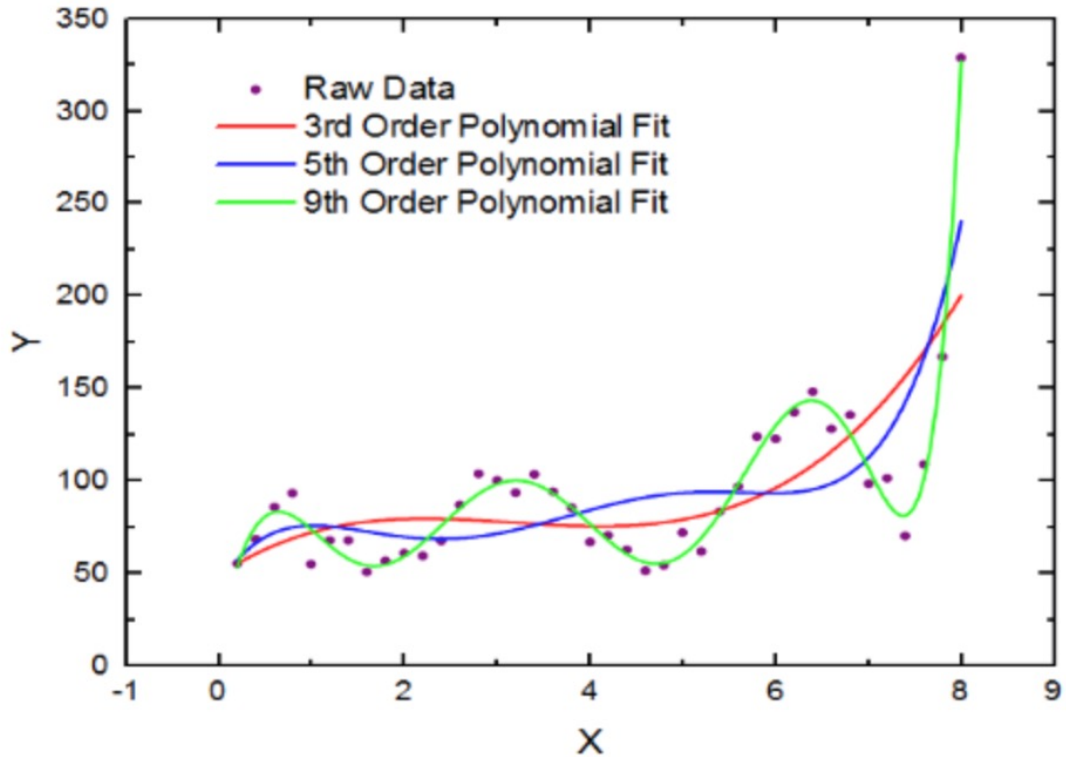
$$y = b_0 + b_1 x_1$$

Multiple Linear Regression

$$y = b_0 + b_1 x_1 + b_2 x_2 + \ldots + b_n x_n$$

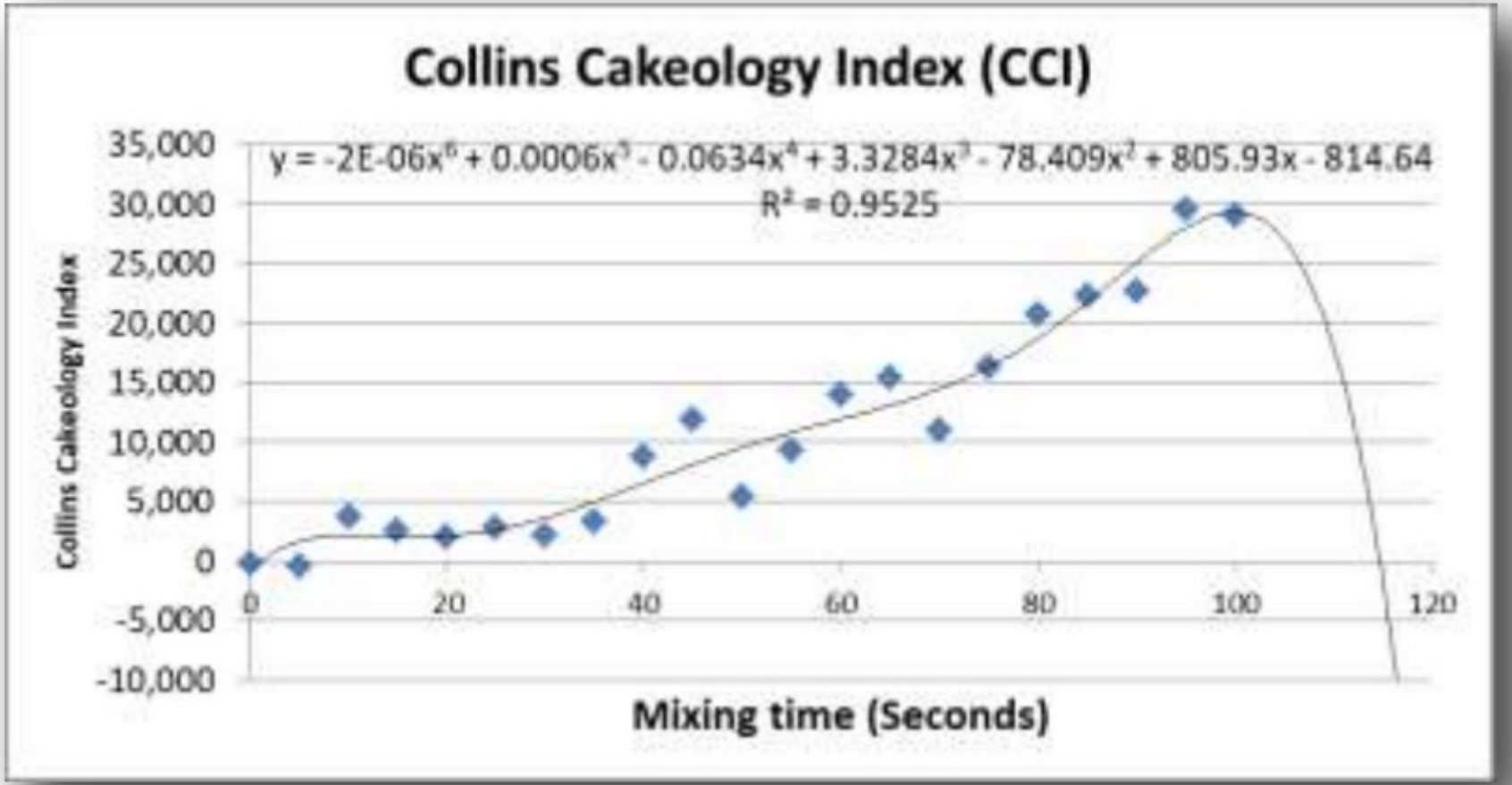
Polynomial Linear Regression

$$y = b_0 + b_1 x_1 + b_2 x_1^2 + \ldots + b_n x_1^n$$




- Polynomial regression can ***reduce your costs returned by the cost function***
- It gives the regression line ***a curvilinear shape*** and makes it more fitting for the underlying data
- By applying a ***higher order polynomial***, you can fit your regression line to your data more precisely BUT watch out for ***overfitting*** *(*next slides)

- We could use a much more complex model
- But the problem here is over-fitting
- The line matches the existing points ..
- But is a poor predictor of new values



3

# Controlling over-fitting



# So we can detect over-fitting ..

# But can we control it?

# Maybe dial-down the wiggle?

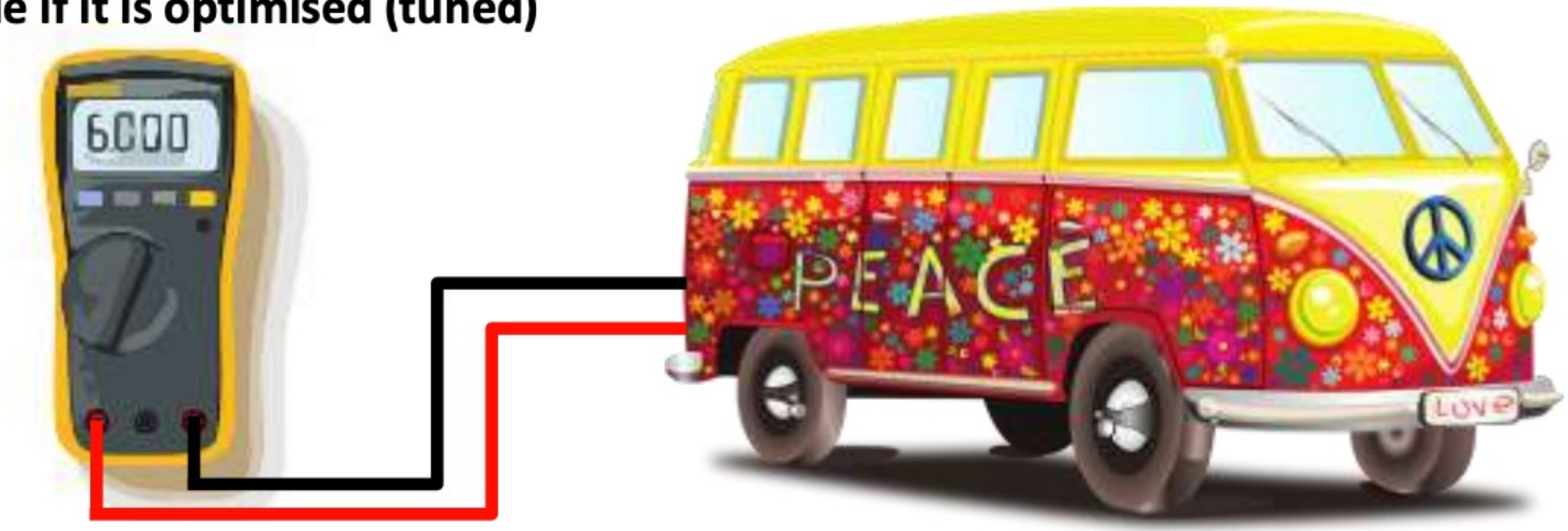

The 'Wiggle' in the curve fit is controlled by the θ values ..
So maybe we could control them to have less impact?

$$MEDV = \theta_0 + \theta_1.LSTAT + \theta_2.RM + \theta_3.LSTAT^2 + \theta_4.RM^2 + \theta_6.LSTAT.RM$$

# Remember: You need measurement to tune a machine

**The measurement you use to decide if it is optimised (tuned)**

**The thing you are optimising (tuning)**



# Regularisation

$$Cost = \sum_{i=1}^{n} (y_i - (k + A.x))^2 + \lambda \sum_{j=0}^{f} |\theta_j|^p$$

- Techniques that are used to calibrate machine learning models in order to minimize the adjusted loss function and prevent overfitting or underfitting.
- Penalization of high-valued regression coefficients/weights.
- Reduces parameters in a model and ***simplifies a model.***

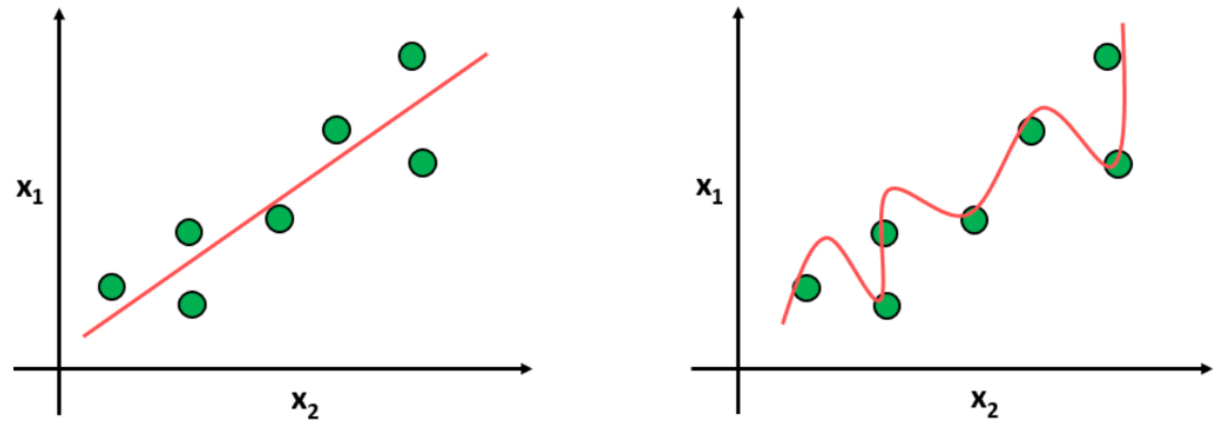
$$y = b_0 + b_1 x_1 + b_2 x_1^2 + \ldots + b_n x_1^n$$

# Regularisation

- Why overfitting occurs?

*Salary = b + $m_1$*(number of years of experience)*

- Simple linear regression and mostly fail.



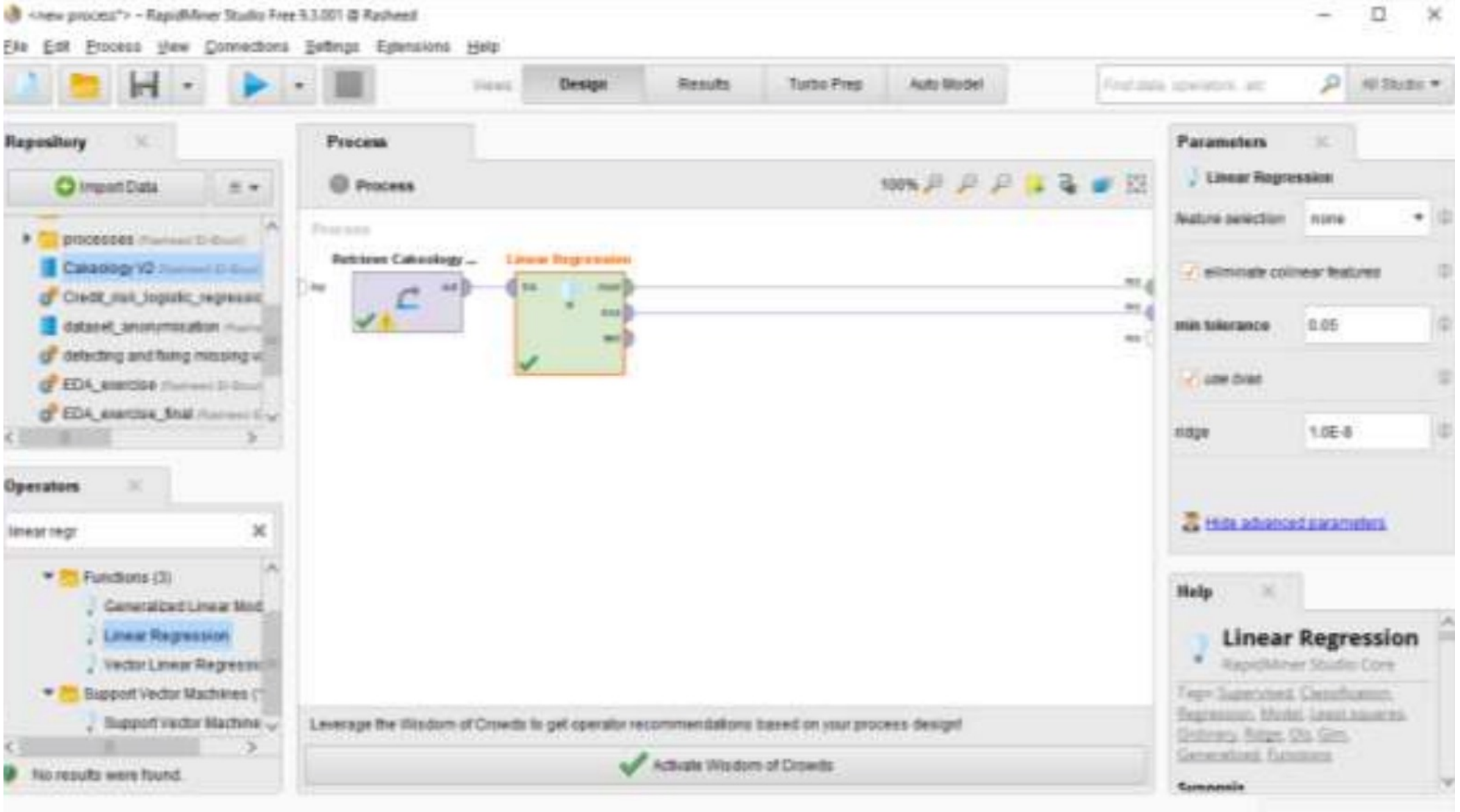

Fitting vs Overfitting

- Add more new variables to improve the model, it will be more complex and make more accurate
- BUT we might add too many variables and make the model OVERFIT (too much noise)
- ***Overfitting*** *means that the model will have poor prediction and generalization problem*

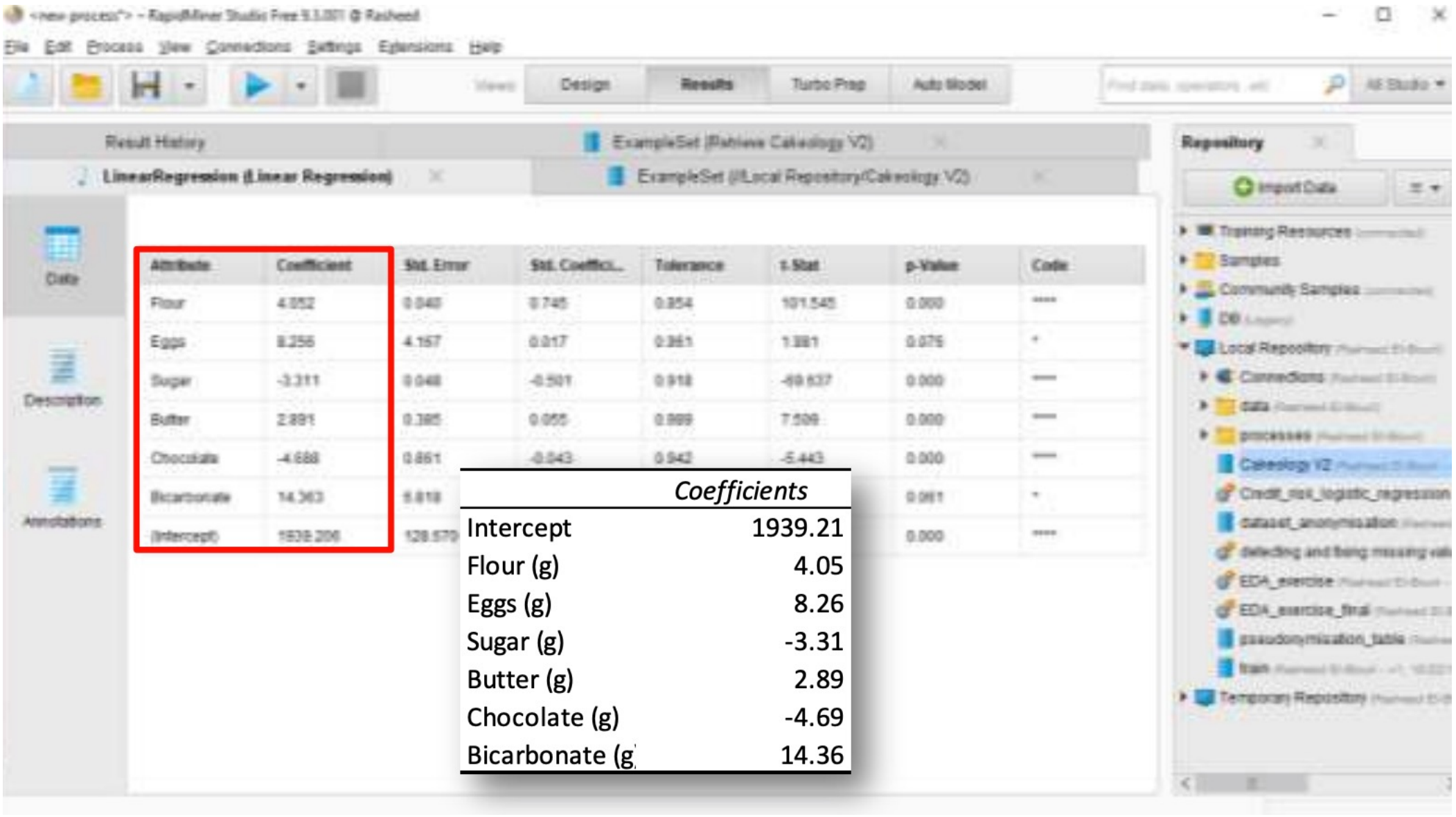
- Regularisation adds a penalty on the different parameters of the model to ***reduce the freedom of the model - less noise***

- ***Large weights are indicative of overfitting***! Let's not allow them to get to that stage

# Do it in RapidMiner

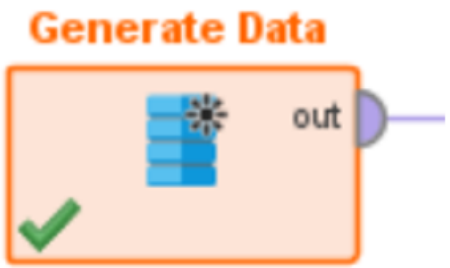


# Do it in RapidMiner



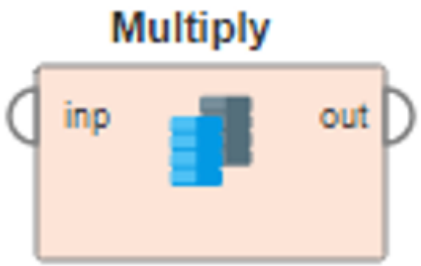
| | *Coefficients* |
|---|---|
| Intercept | 1939.21 |
| Flour (g) | 4.05 |
| Eggs (g) | 8.26 |
| Sugar (g) | -3.31 |
| Butter (g) | 2.89 |
| Chocolate (g) | -4.69 |
| Bicarbonate (g | 14.36 |

# Generate Data




- Generates dataset according to some underlying process
- This can be random or according to some structure
- You may also choose the size of the table and the range of the values included in the entries

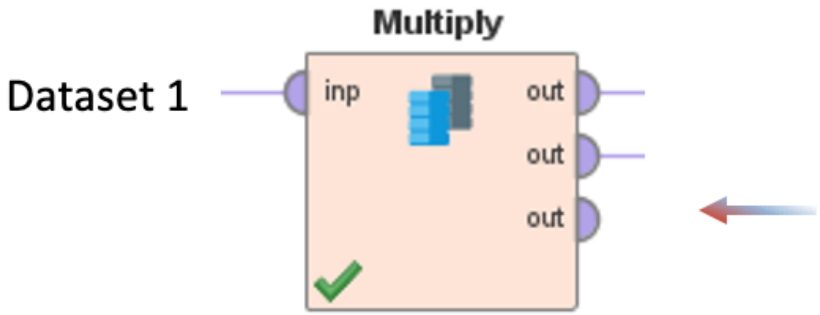
# Multiply




Multiply creates multiple copies of the input

The number of outputs can be as large as you like

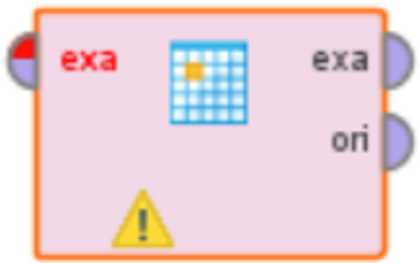
Creates multiple independent copies of the object - no matter how you change one, the other is not affected!




This creates 2 independent copies of Dataset 1.
We could keep creating more if we liked.

# Set Data and Generate Empty Attribute

**Set Data**




Allows you to set a value in a specific attribute to a value of your choosing

**Generate Empty Attribute**


Allows you to create an attribute that is empty and that can be subsequently filled

# Replace Missing Values


Allows you to replace:

- missing values in your dataset with the number 0
- the maximum of the column
- the minimum of the column
- the average of the column

# Normalise Data


Scales your data to a range that is more useful for machine learning

The standard format is to use the Z-transform defined as:

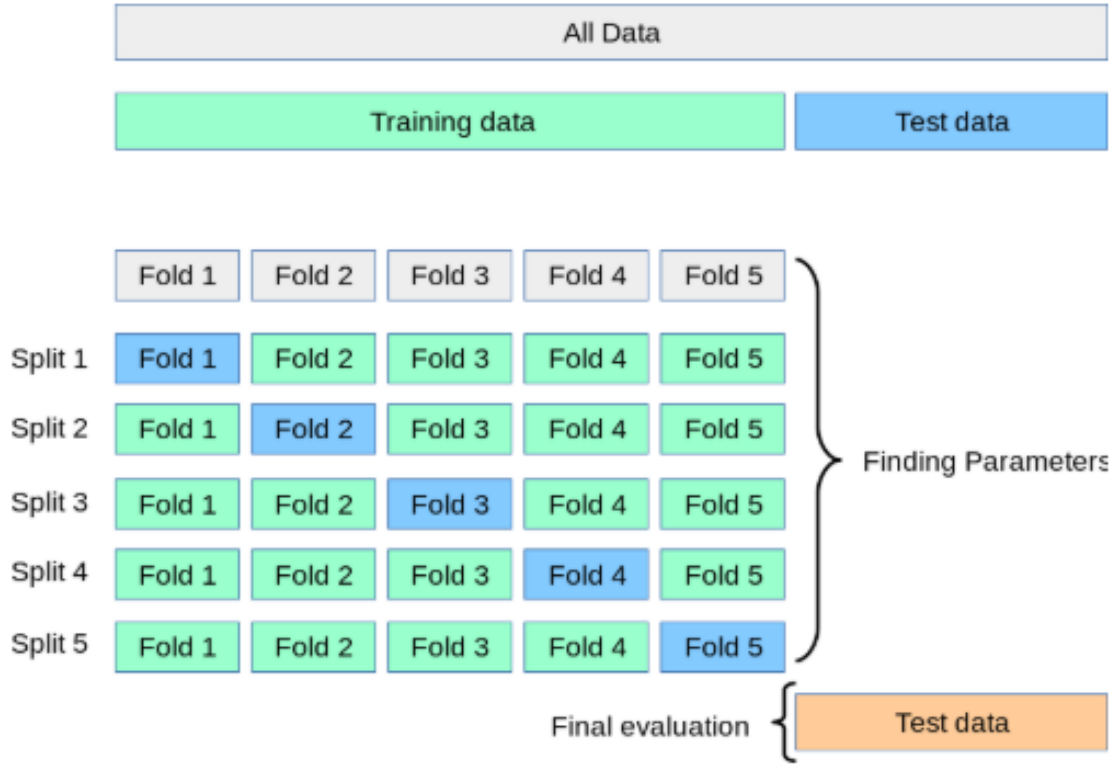
Allows you to set a value in a specific attribute to a value of your choosing

Another type is min-max scaling defined as:

Allows you to create an attribute that is empty and that can be subsequently filled

# K-Fold Cross Validation

- Accuracy estimator
- Resampling method that uses different portions of the data to test and train a model on different iterations
- Make a fixed number of folds (or partitions) of the data
- Train on each fold [Train Fold2 + Fold1],[Train Fold3 + Fold1]
- Average the overall error estimate



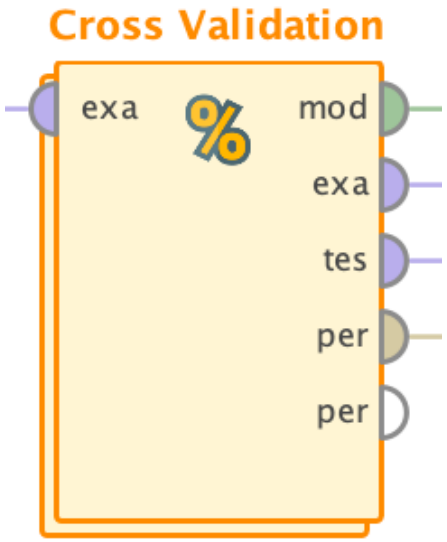

Advantages:

- Protects against overfitting. How?

Note:

- Use max K=10
- The smaller the dataset, we use less of K-folds. Why?

# K-Fold Cross Validation




Resample the same dataset multiple times and pretend they are different

The model is trained from scratch, each time, without reusing the training result from previous attempts

The evaluation of the performance of a model on independent test sets yields a good estimation of the performance on unseen data sets

# Too complex or just right?


- We may need to build a more complex model
  - .. Reflecting the complexity of the thing we are modelling
- But we also need to be concerned about over- fitting..
  - Model testing
  - Controlling over-fitting

1

2

3

# 1 Building more complex models

# Predicting house prices using Polynomial Regression

The “Boston House Price” data-set is a fairly common choice for tutorials in Regression

The task is to model house prices based on a set of features


https://www.kaggle.com/vikrishnan/boston-house-prices

# Data definitions .. What are the features?

| RM | ZN | INDUS | CHAS | NOX | RM | AGE | DIS | RAD | TAX | PTRATIO | B | LSTAT | MEDV |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0.00632 | 18.0 | 2.31 | 0.0 | 0.538 | 6.575 | 65.2 | 4.0900 | 1.0 | 296.0 | 15.3 | 396.90 | 4.98 | 24.0 |
| 0.02731 | 0.0 | 7.07 | 0.0 | 0.469 | 6.421 | 78.9 | 4.9671 | 2.0 | 242.0 | 17.8 | 396.90 | 9.14 | 21.6 |
| 0.02729 | 0.0 | 7.07 | 0.0 | 0.469 | 7.185 | 61.1 | 4.9671 | 2.0 | 242.0 | 17.8 | 392.83 | 4.03 | 34.7 |
| 0.03237 | 0.0 | 2.18 | 0.0 | 0.458 | 6.998 | 45.8 | 6.0622 | 3.0 | 222.0 | 18.7 | 394.63 | 2.94 | 33.4 |
| 0.06905 | 0.0 | 2.18 | 0.0 | 0.458 | 7.147 | 54.2 | 6.0622 | 3.0 | 222.0 | 18.7 | 396.90 | 5.33 | 36.2 |

**RM**: Per capita crime rate by town

**ZN**: Proportion of residential land zoned for lots over 25,000 sq. ft

**INDUS**: Proportion of non-retail business acres per town

**CHAS**: Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)

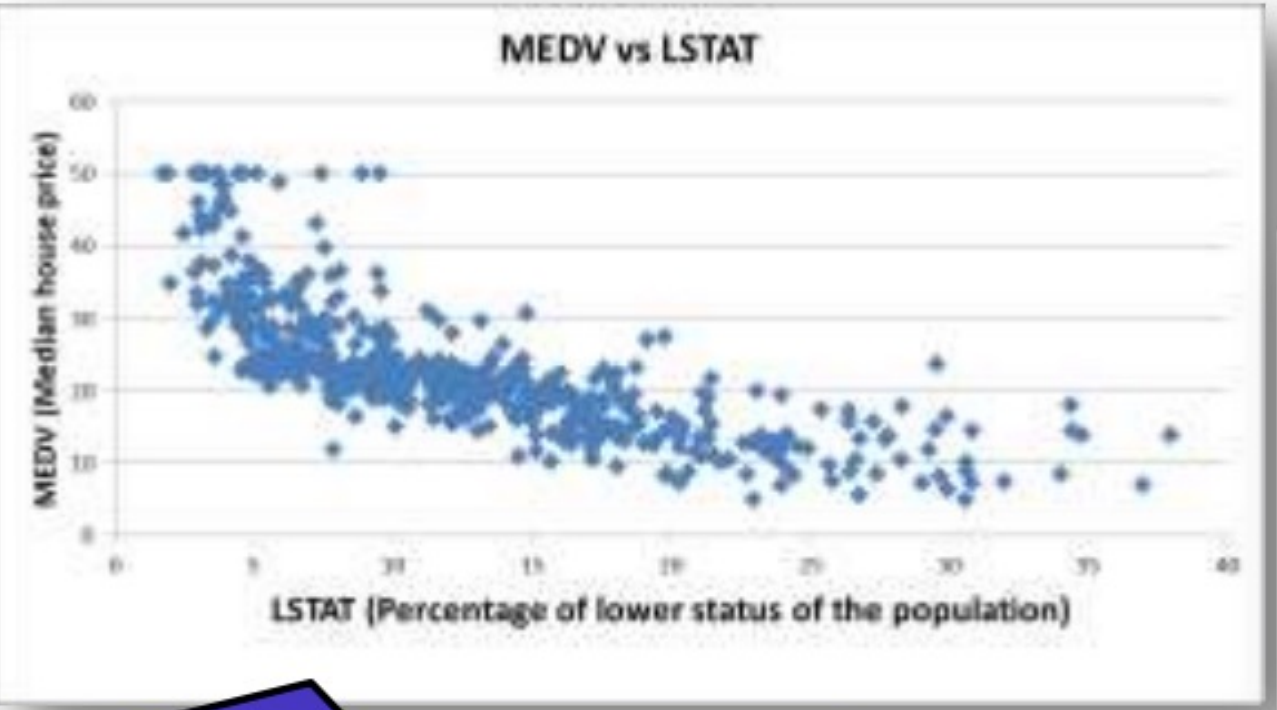
- **NOX**: Nitric oxide concentration (parts per 10 million)
- **RM**: Average number of rooms per dwelling
- **AGE**: Proportion of owner-occupied units built prior to 1940
- **DIS**: Weighted distances to five Boston employment centers
- **RAD**: Index of accessibility to radial highways
- **TAX**: Full-value property tax rate per $10,000
- **B**: $1000(Bk - 0.63)^2$, where Bk is the proportion of people of African American descent by town
- **LSTAT**: Percentage of lower status of the population
- **MEDV: Median value of owner-occupied homes in $1000s**

# Review the data .. Any candidates for Regression?

- This heat-map of correlations identifies highly-correlated items in darker shades of blue
- Both LSTAT and RM have high correlations with house prices (MEDV) and hence are good candidates for regression

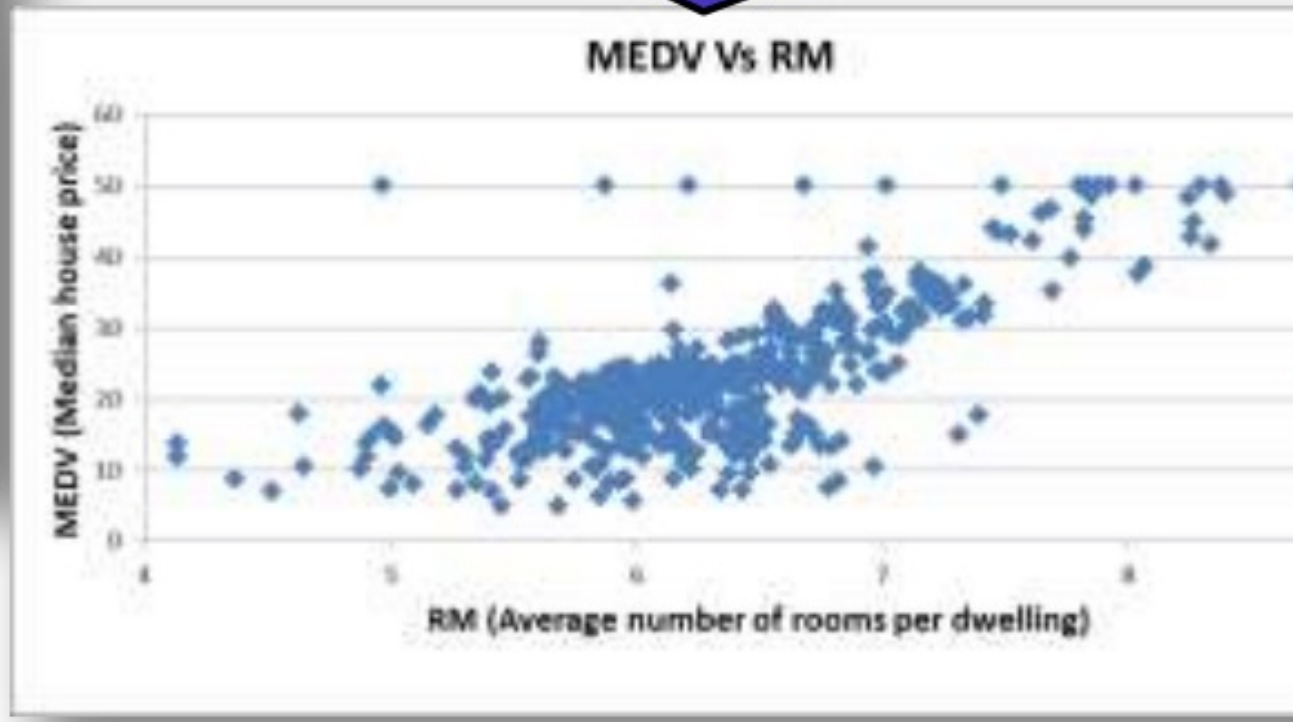

Absolute (non-negative) correlations

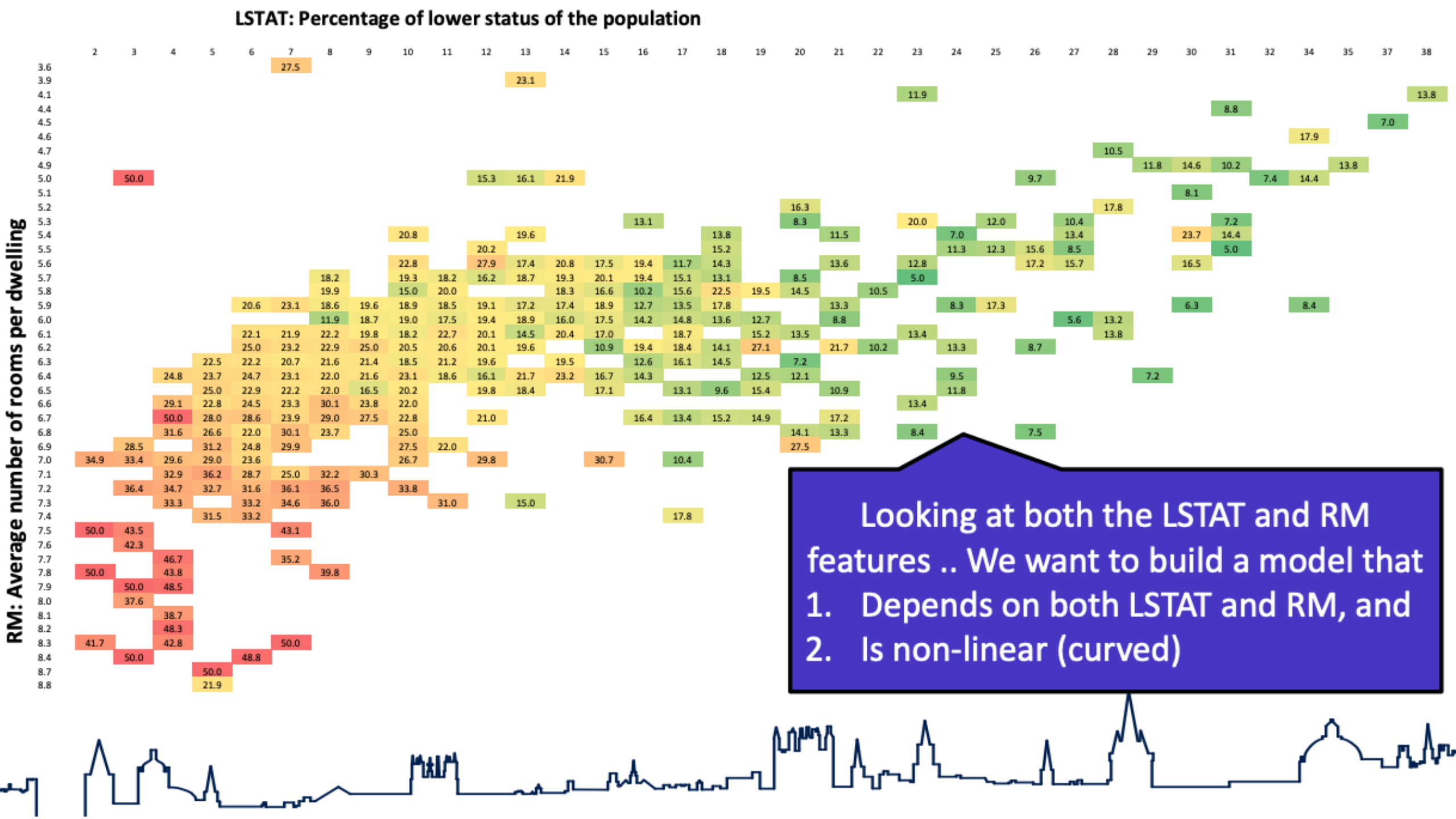
# Looking more closely at LSTAT and RM




Definitely looks non-linear

May or may not be linear

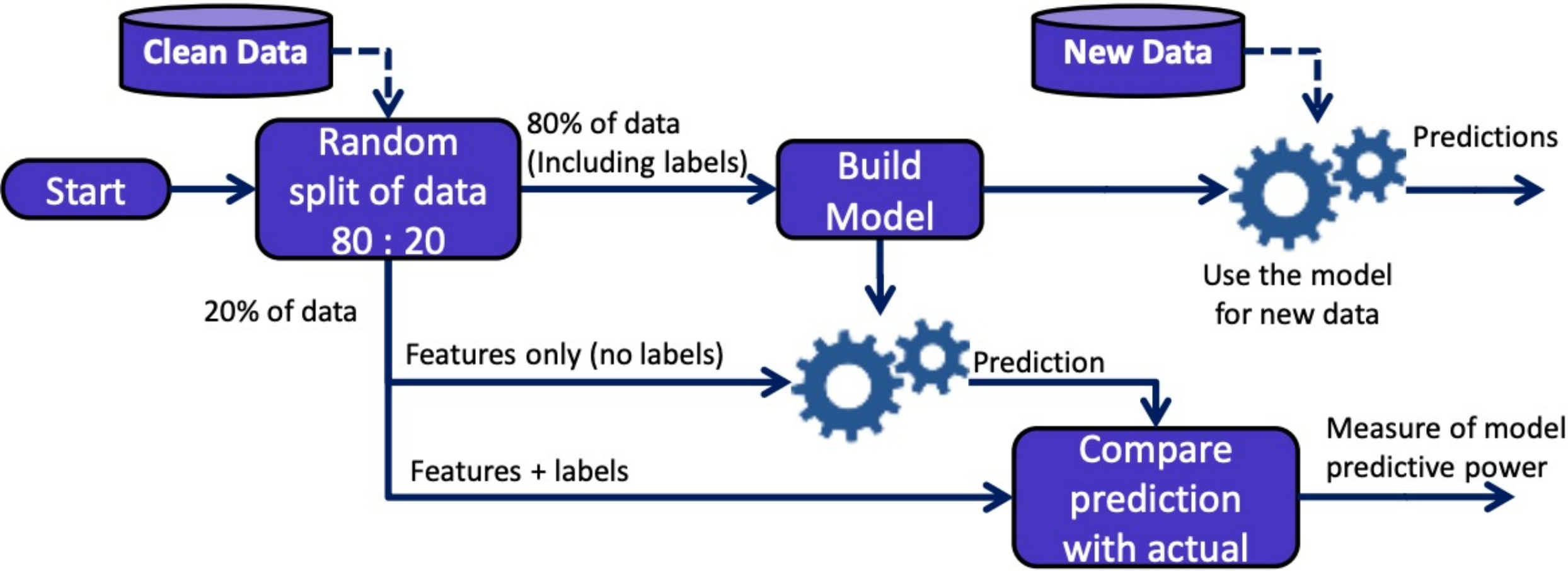



LSTAT: Percentage of lower status of the population
RM: Average number of rooms per dwelling
Looking at both the LSTAT and RM features .. We want to build a model that
1. Depends on both LSTAT and RM, and
2. Is non-linear (curved)

# 2 Fitness testing



# Reminder from week 1 : Model Testing




# Splitting the data into testing and training data-sets

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2)
print(X_train[0:5])
```

```
 LSTAT     RM
26.82  5.403
 3.13  8.040
12.67  5.898
 5.08  6.431
17.64  6.348
```

Sklearn provides a built-in function for randomly dividing a data set into training and testing portions

# Comparing Linear and Higher-order model performance

```
Linear Model Performance
========================

Training set
------------
Root Mean Square is  5.72
R2 score is   0.64


Testing set
-----------
Root Mean Square is  4.73
R2 score is   0.62
```

```
Performance of Polynomial Model of degree 2
===========================================

Training set
------------
Root Mean Square is   4.57
R2 score is   0.77


Test set
--------
Root Mean Square is  4.49
R2 score is   0.65
```

```
Performance of Polynomial Model of degree 5
==========================================
Training set
------------
Root Mean Square is  4.15
R2 score is  0.81


Test set
--------
Root Mean Square is  4.07
R2 score is  0.72
```

```
Performance of Polynomial Model of degree 20
==========================================
Training set
------------
Root Mean Square is  4.94
R2
```

```
Performance of Polynomial Model of degree 25
==========================================
Training set
------------
Root Mean Square is  8.15
R2 score is  0.27


Test set
--------
Root Mean Square is  6.50
R2 score is  0.27
```

# The Impact of Different types of Regularisation

# Different forms of Regularisation

**L1 Regularisation : "LASSO"**

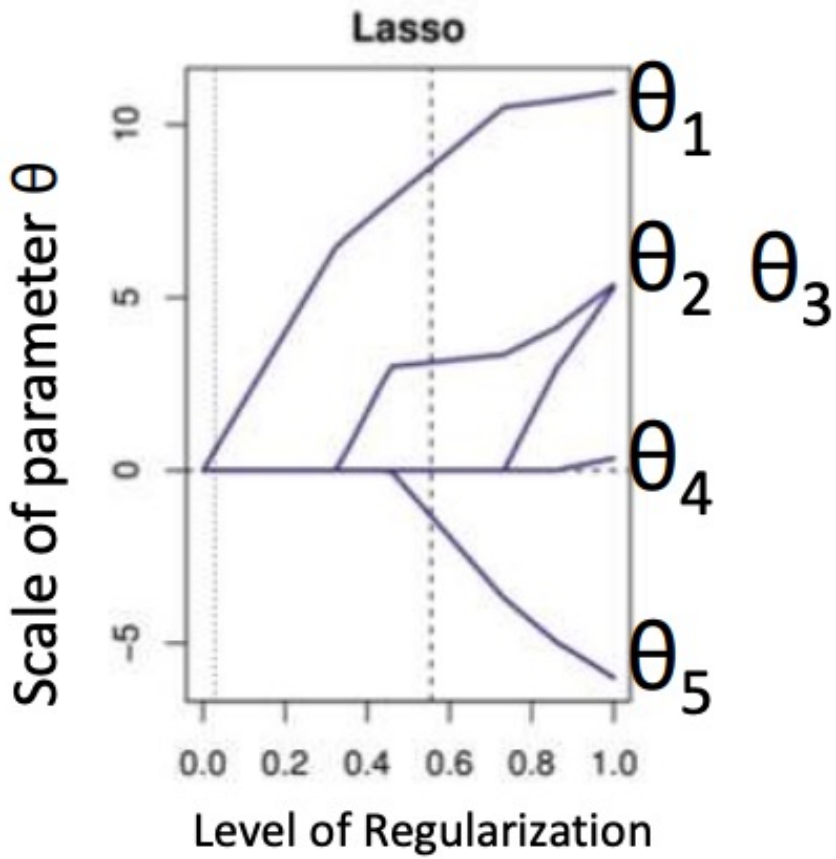
$$Cost = \sum_{i=1}^{n} (y_i - (\theta_0 + \theta_1 . x_i))^2 + \lambda . \sum_{j=1}^{n} |\theta_j|$$

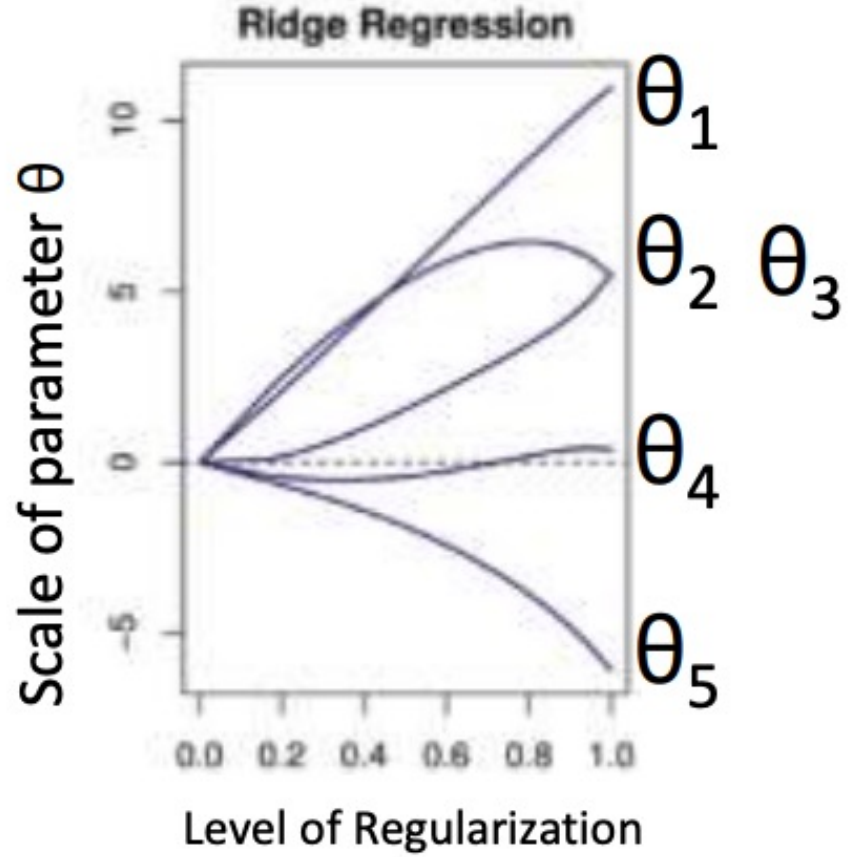
**L2 Regularisation : "Ridge"**

$$Cost = \sum_{i=1}^{n} (y_i - (\theta_0 + \theta_1 . x_i))^2 + \lambda . \sum_{j=1}^{n} \theta_j{}^2$$

# L1 and L2 .. What is the difference?

- L1 and L2 regularisation have different impacts on the parameters θ
- L1 (Lasso) can have the effect of completely 'switching off' some parameters – hence reducing computational effort for large models



# Conclusion

- Regularisation provides a mechanism for controlling over-fitting
- It enables control over the complexity of the model:
  - Complex enough to fit the data well
  - Simple enough to prevent over-fitting
- Tuned as a hyper-parameter


# References

- https://www.youtube.com/watch?v=d6XDOS4btck
  - Quite advanced, rather mathematical .. But excellent lecture on L1 and L2 regression and how they are different
-