# Regression Methods

**Dr Maria Anastasiadi**
**(m.anastasiadi@cranfield.ac.uk)**

16th January 2025

www.cranfield.ac.uk

# Regression

- Regression in data analysis is used for estimating relationships between data and for future numeric predictions.
- The target when doing regression is to predict a target value.
- Regression defines the relationship between a **single numeric dependent** variable (the value to be predicted) and **one or more numeric independent variables** (the predictors).
- Useful in cases when we want to predict the value of a difficult to quantify variable (e.g. a metabolite, microbial counts) using rapid non-destructive techniques.

# A bit of historical background

- Regression was invented by the cousin of Charles Darwin, Sir Francis Galton (1822-1911).
- Galton performed regression on a number of things, including the heights of humans.

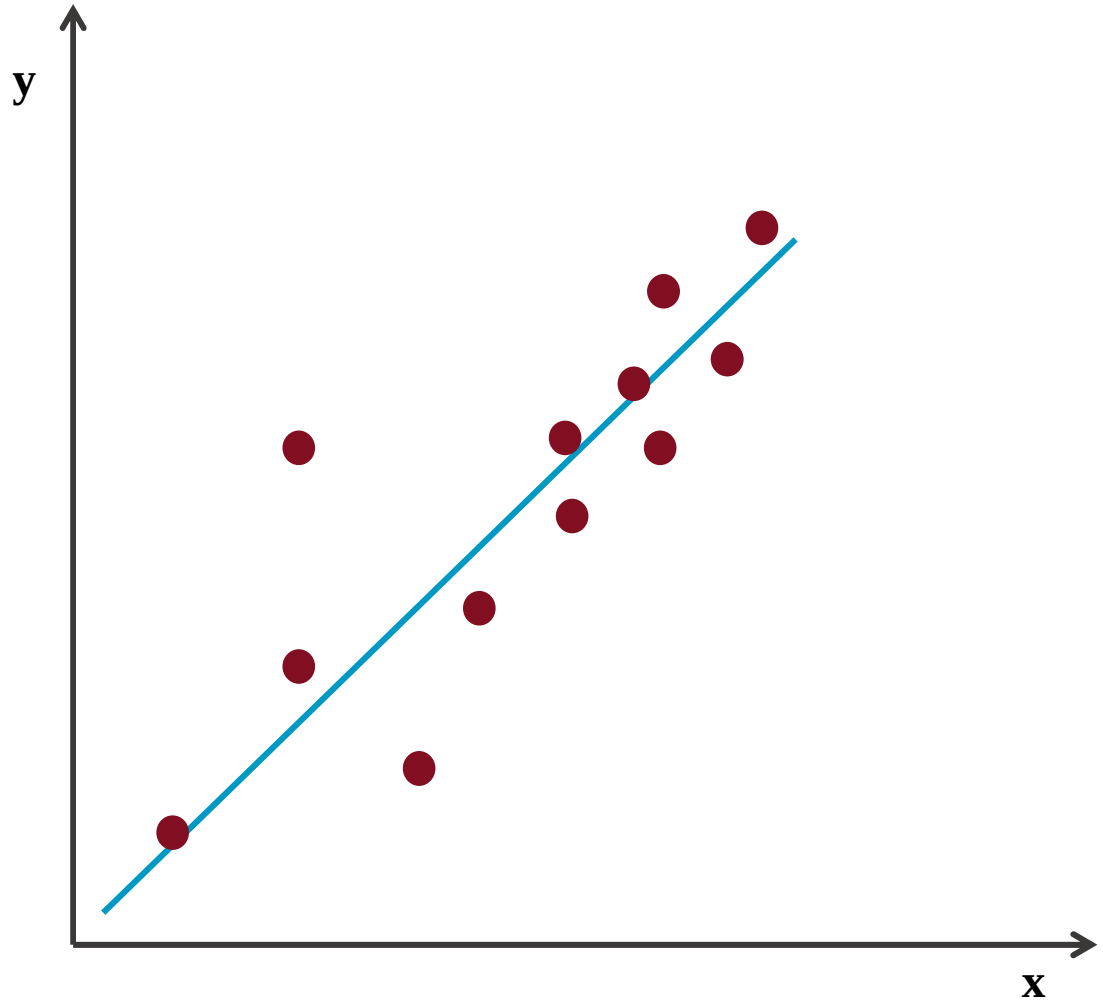
# Simple Linear Regression

## Simple Linear Regression

A simple linear regression model defines the relationship between a dependent variable and a single independent predictor variable using a line defined by an equation in the following form:

$$y = a + bx$$

Where: $a$ = the intercept, $b$ = the slope



# Simple Linear Regression

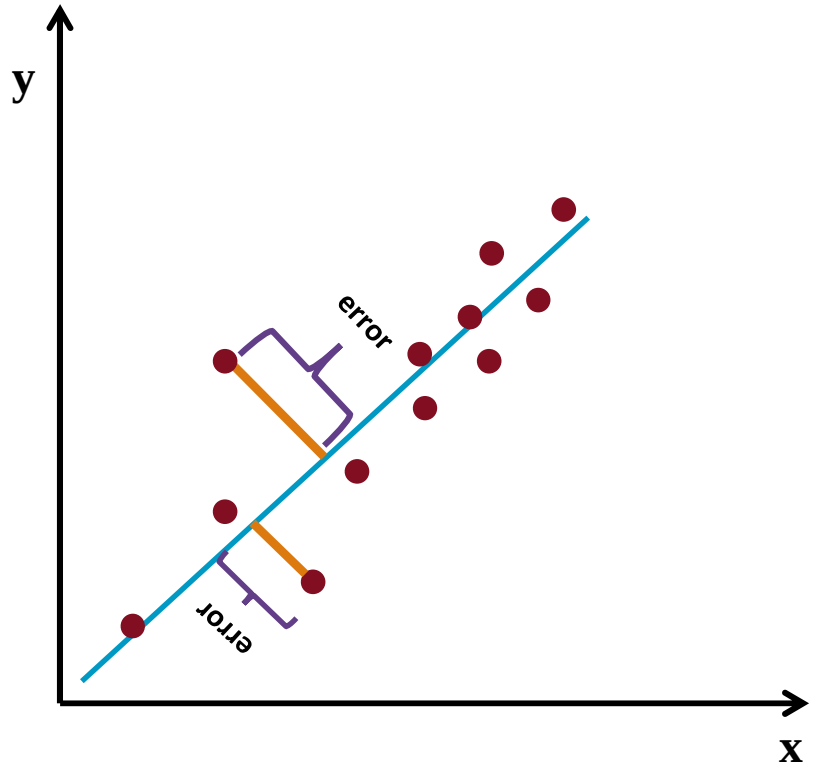
## Ordinary Least Squares Regression (OLS-R)

$$y = a + bx$$

The value of $b$ that results to the min sum of squared errors is:

$$b = \frac{Cov(x, y)}{Var(x)}$$

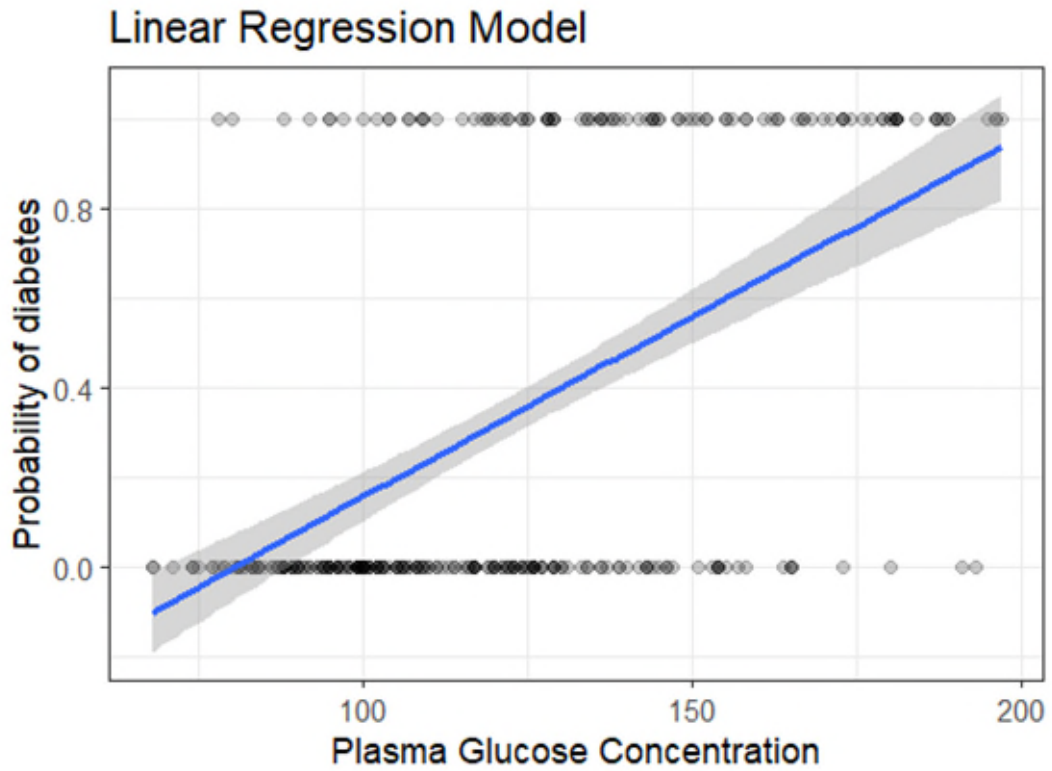
Finally, **a** is calculated by: $a = \bar{y} - b\bar{x}$

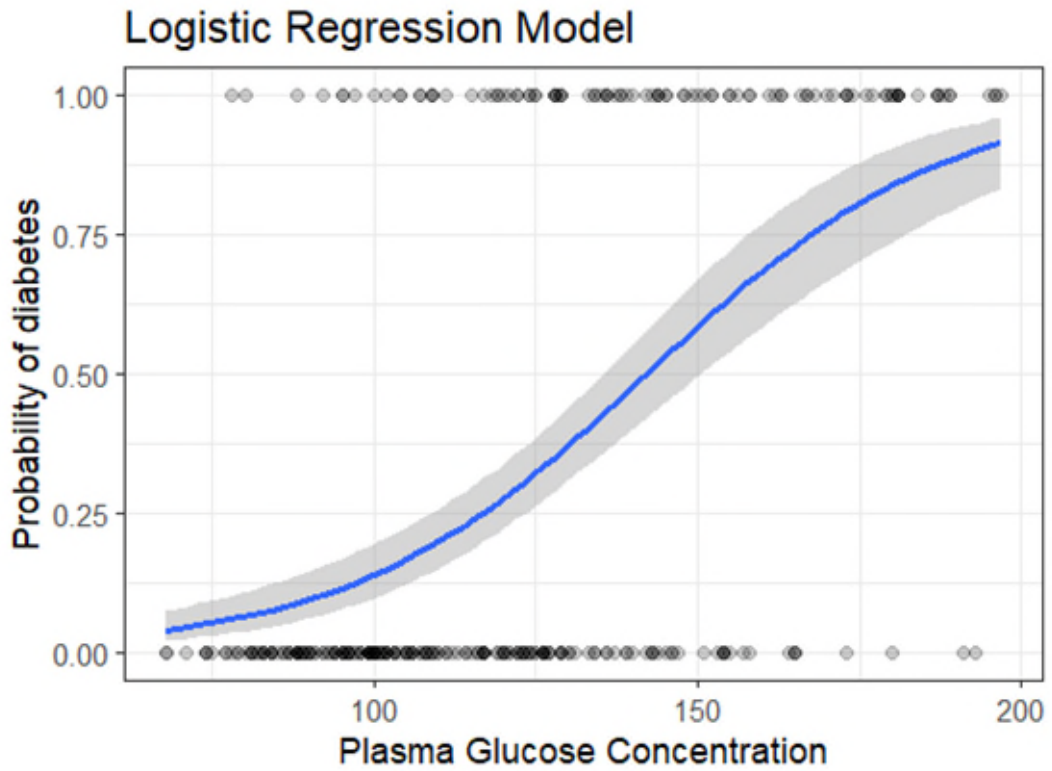


# Generalized linear models (GLMs) – Logistic Regression

Linear regression is used to approximate the linear relationship between a continuous response variable and a predictor variable.

But what happens when the response variable is not continuous but binary?



# Generalized linear models (GLMs) – Logistic Regression

We can use the same function to fit both linear and logistic regression:

**For continuous response variable**

```
model1 <- glm(response.var ~ explan.var, data = data1, family = gaussian)
```

**For binomial response variable**

```
model2 <- glm(response.var ~ explan.var, data = data2, family = binomial)
```

```
model <- glm( diabetes ~ glucose, data = train.data, family = binomial)
➢ summary(model)$coef

               Estimate  Std. Error   z value     Pr(>|z|)
(Intercept) -6.15882009 0.700096646 -8.797100 1.403974e-18
glucose      0.04327234 0.005341133  8.101716 5.418949e-16
```

# Regression - Multivariate Calibration

In practice, we often have more than one predictor variables associated with a response variable.

**Multivariate calibration** is used to establish a regression model relating a response y and a set of M predictor variables $\{x_1, x_2, \ldots, x_M\}$.

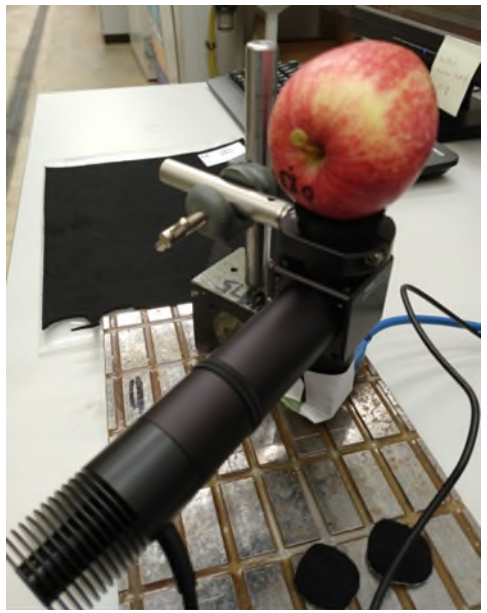
Algorithmically, multivariate calibration can involve two approaches:

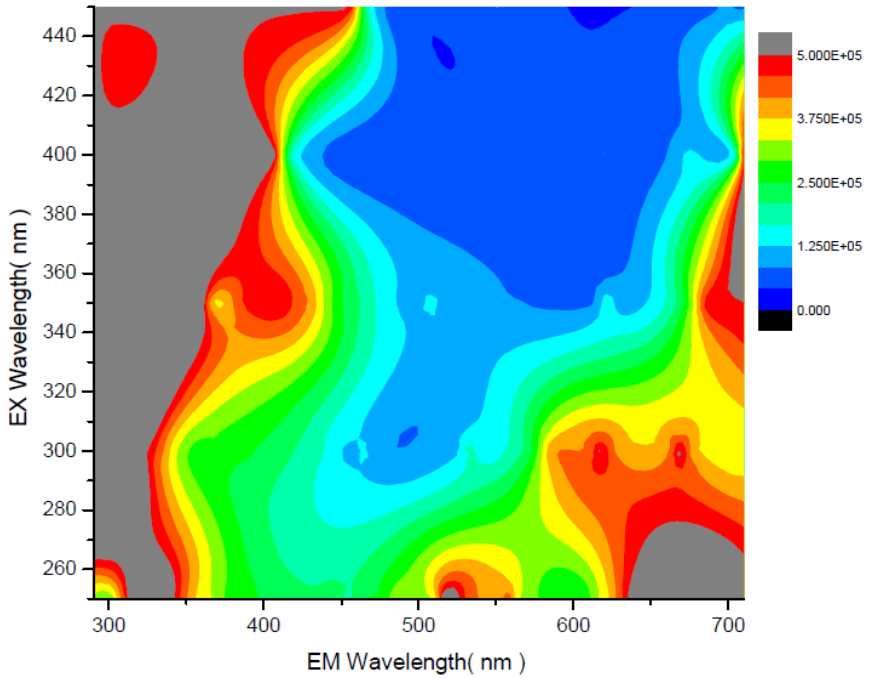
- **Statistical methods;**
- **Machine learning methods.**

# Regression - Multivariate Calibration

Multivariate calibration is particularly suited to spectroscopic techniques, due to the ease of obtaining multiparameter signals.

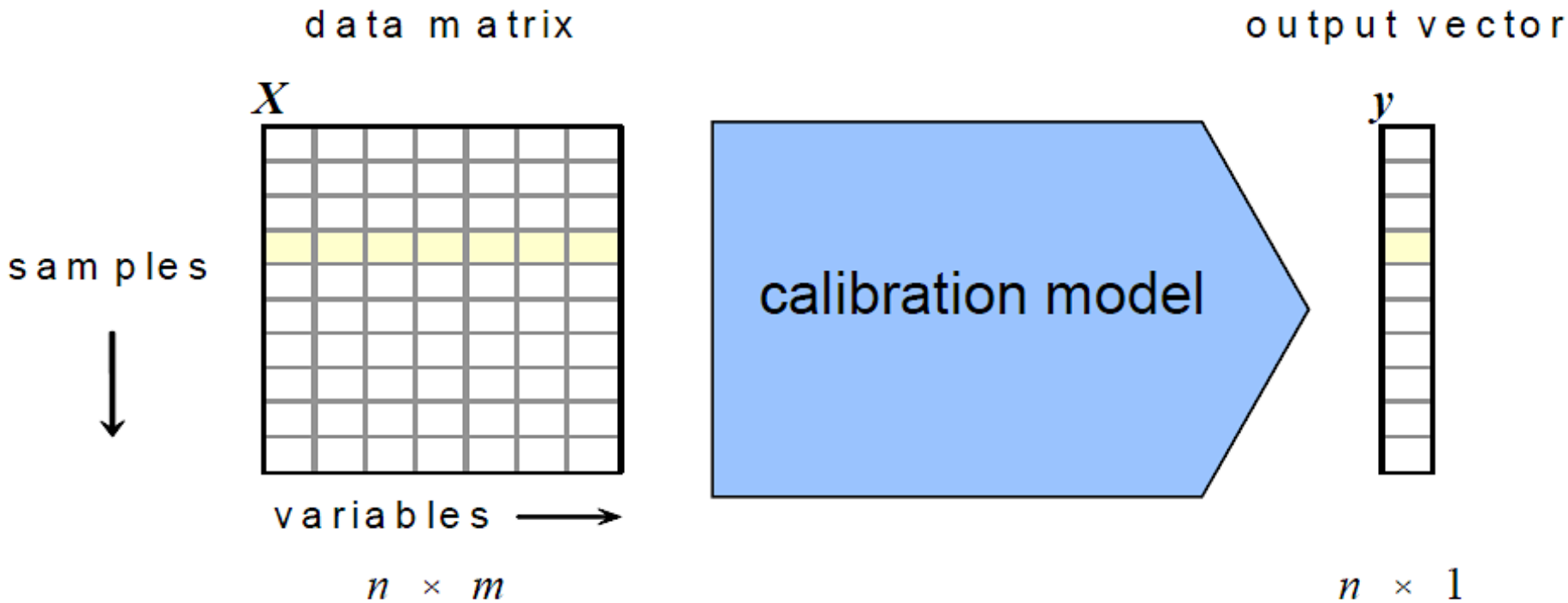
- Near-infrared (NIR)
- UV-Vis
- Raman
- Mid-IR
- Fluorescence
- NMR
- Mass spectra





# Regression - Multivariate Calibration

**Step 1**: We start off with a data matrix, and a corresponding output vector which indicates the value associated with each sample.



# Regression - Multivariate Calibration

**Step 2**: We build a ***calibration model*** that relates the matrix to the vector

$$y = Xb + e$$

**Key Goal:** To Estimate the regression vector ***b***

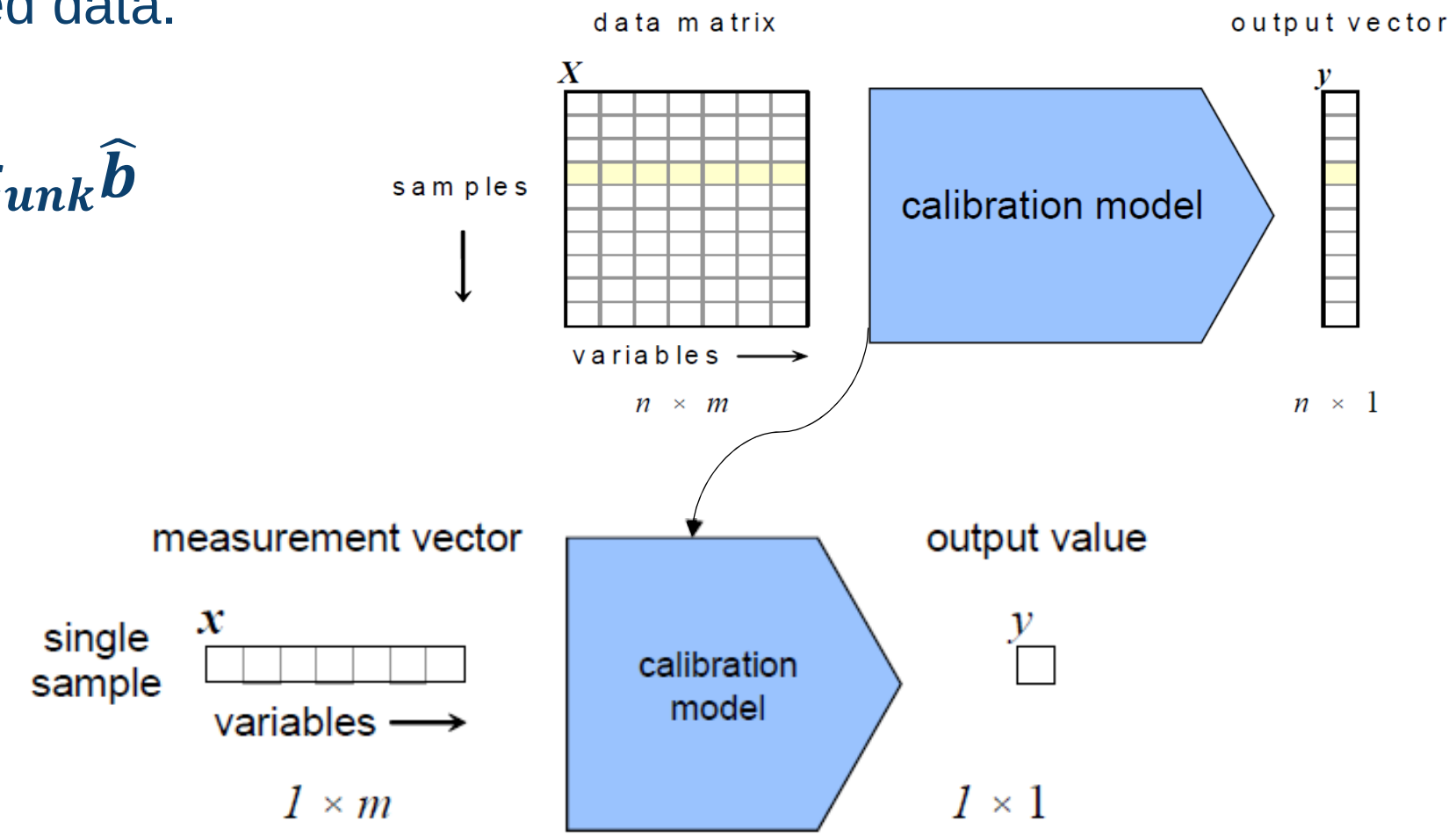
**Case 1:** If the matrix X is of full rank (n>m) we apply the standard least squares solution to find ***b.***

**Case 2:** If the matrix X is not of full rank (m>n) we need to find an approximation.

# Regression - Multivariate Calibration

**Step 3**: Once we've built a calibration model from known data, we can apply it to newly acquired data.

$$\hat{y}_{unk} = x_{unk}\hat{b}$$



# Regression - Multivariate Calibration

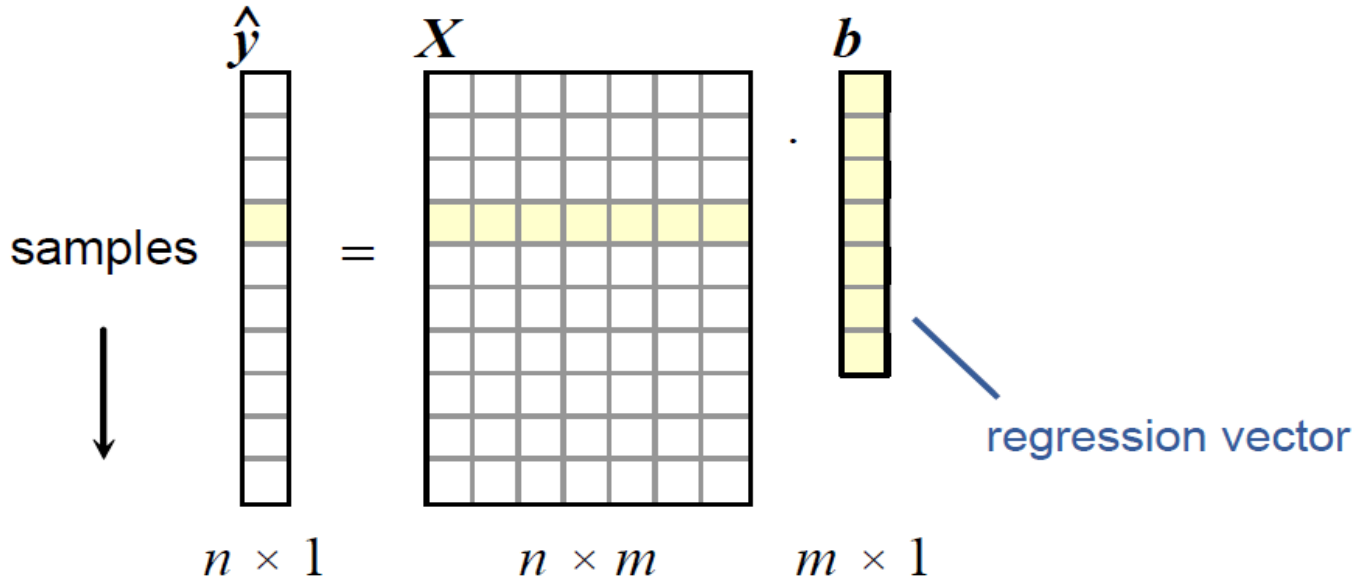
Statistical methods for generating a calibration model split into:

- Multiple linear regression (MLR);
- Logistic Regression (Logit);
- Principal component regression (PCR);
- Partial least squares (PLS);
- Generalized ridge regression (GRR);
- Reduced rank regression (RRR);
- Continuum regression (CR).

# Multiple Linear Regression (MLR)

**Multiple linear Regression (MLR)** is the simplest type of *multivariate regression*.

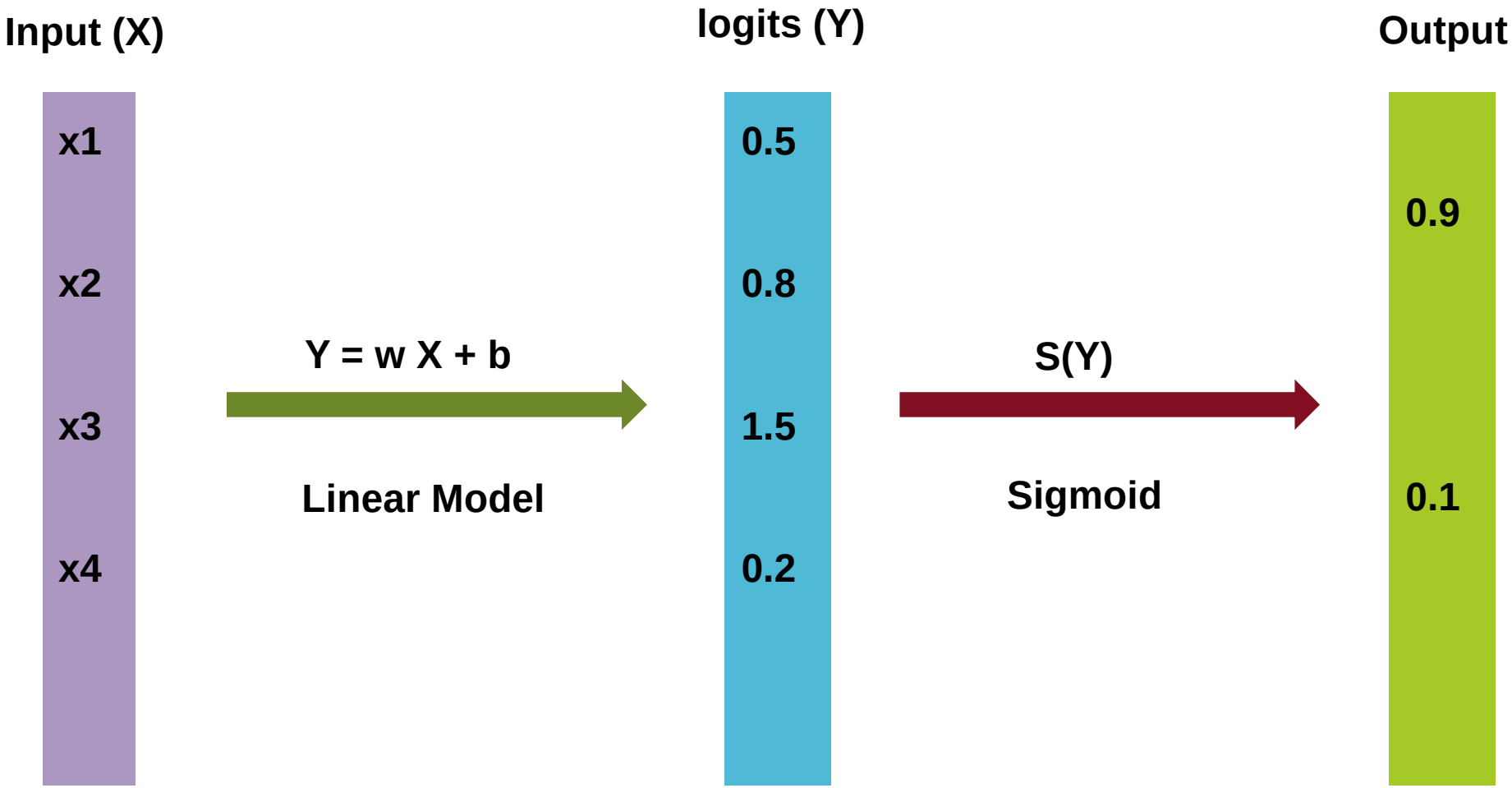
The aim of MLR is to make a direct mathematical link between the ***X*** matrix and the ***y*** vector. Very similar to univariate regression.




$$\hat{y}_4 = x_{4,1}b_1 + x_{4,2}b_2 + x_{4,3}b_3 + x_{4,4}b_4 + x_{4,5}b_5 + x_{4,6}b_6 + x_{4,7}b_7$$

$$\hat{y}_i = x_{i,1}b_1 + x_{i,2}b_2 + x_{i,3}b_3 + x_{i,4}b_4 + x_{i,5}b_5 + x_{i,6}b_6 + x_{i,7}b_7$$

# Generalized linear models (GLMs) – Multiple Logistic Regression



# Generalized linear models (GLMs) – Multiple Logistic Regression

## Use all the variables in the dataset to predict diabetes status

```
model1 <- glm( diabetes ~., data = train.data, family = binomial)
➢ summary(model)$coef

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.053e+01  1.440e+00  -7.317 2.54e-13 ***
pregnant     1.005e-01  6.127e-02   1.640  0.10092
glucose      3.710e-02  6.486e-03   5.719 1.07e-08 ***
pressure    -3.876e-04  1.383e-02  -0.028  0.97764
triceps      1.418e-02  1.998e-02   0.710  0.47800
insulin      5.940e-04  1.508e-03   0.394  0.69371
mass         7.997e-02  3.180e-02   2.515  0.01190 *
pedigree     1.329e+00  4.823e-01   2.756  0.00585 **
age          2.718e-02  2.020e-02   1.346  0.17840
```

# Generalized linear models (GLMs) – Multiple Logistic Regression

## Test the model

```
probabilities <- model1 %>% predict(test.data, type = "response")
predicted.classes <- ifelse(probabilities > 0.5, "pos", "neg")
# Model accuracy
mean(predicted.classes == test.data$diabetes)
0.7564103
```

## Model Optimisation

### 1. Select variables with statistically significant beta values (p<0.05)

```
model2 <- glm( diabetes ~ glucose + mass + pedigree,
               data = train.data, family = binomial)
```

## Test the model

```
# Model accuracy
mean(predicted.classes == test.data$diabetes)
0.7692308
```

# Generalized linear models (GLMs) – Multiple Logistic Regression

## Model Optimisation

### 2. Apply feature selection with stepwise regression

```
library(MASS)
# Fit the model
model3 <- glm(diabetes ~., data = train.data, family = binomial) %>%
              stepAIC(direction = "both", trace = 1)
> summary(model)$coef
                Estimate  Std. Error   z value     Pr(>|z|)
(Intercept) -10.77109289 1.301232729 -8.277607 1.256754e-16
pregnant      0.09903127 0.060971109  1.624233 1.043261e-01
glucose       0.03813593 0.005694031  6.697527 2.119761e-11
mass          0.09512159 0.024175881  3.934566 8.334727e-05
pedigree      1.36220306 0.479358973  2.841718 4.487115e-03
age           0.02956352 0.019146186  1.544094 1.225655e-01
```

## Test the model

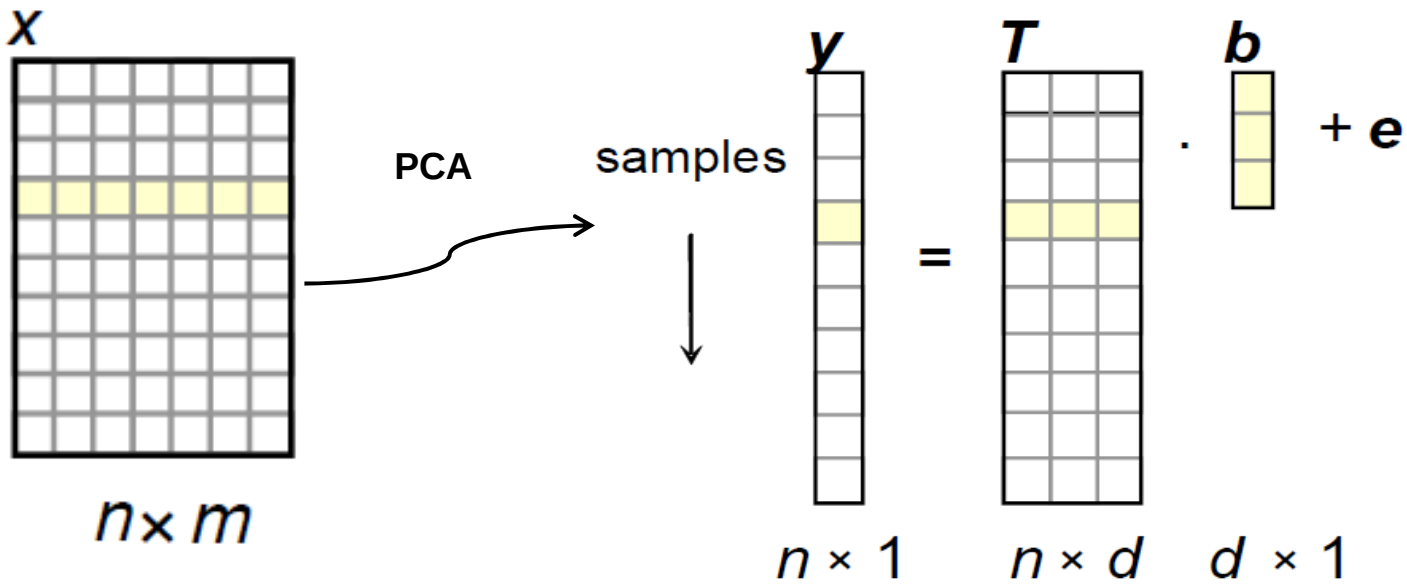
```
# Model accuracy
mean(predicted.classes == test.data$diabetes)
0.7820513
```

# Data Reduction Techniques

- MLR techniques try to build the model using all the data in the data matrix.
- Bad idea due to noise → ***Multi-colinearity***
- **Data reduction** techniques generate a smaller set of predictors that seek to capture the majority of the information in the original variables.
- Also called **signal extraction** or **feature extraction** techniques.

# Principal Component Regression (PCR)

PCR, data is first processed to reduce the data matrix down to a matrix of PC scores, and the regression is carried out on that.




Matrices **T** and **P** are utilised to calculate the regression vector, ***b.***

$$\mathrm{T}=\mathrm{XP}$$

$$\hat{y}=\mathrm{T}\cdot\mathrm{b}$$

# Regression - Partial Least Squares (PLS)

**Partial least squares (PLS)** takes the PCR concept a step further.

It breaks down the data matrix into new combined variables called ***latent variables*** and performing regression on these.

Unlike PCR, PLS takes advantage of the fact that the sample types are known and picks out the latent variables based on the amount of *covariance* between ***X*** and ***y***.

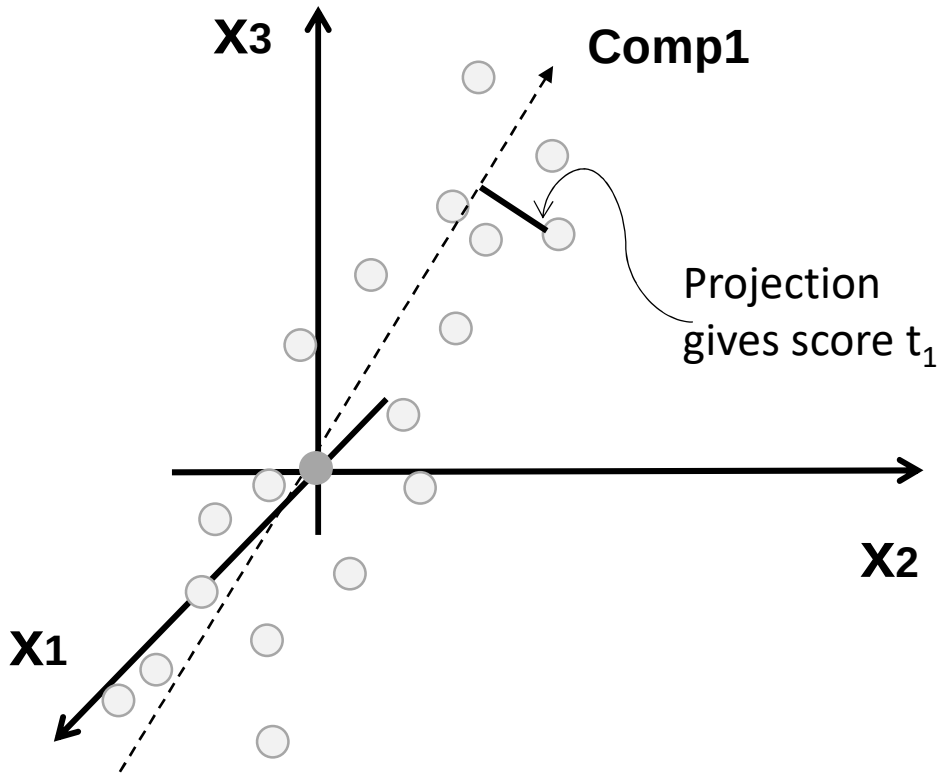
By doing this, we end up with latent variables that have maximum relevance to the application we are working on.

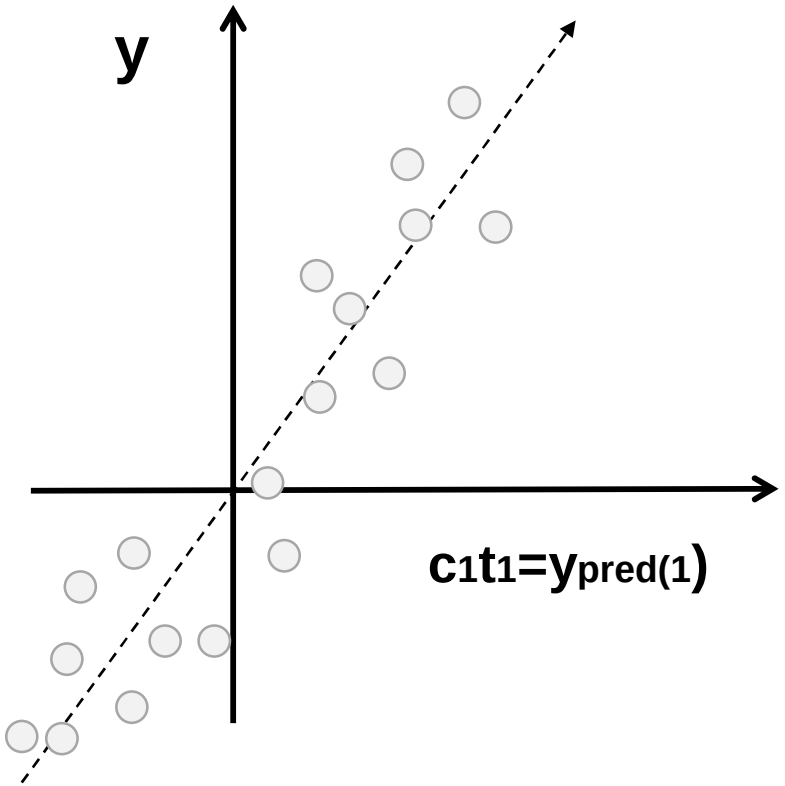
# Regression - PLS

$$X = TP' + E$$
$$Y = UC' + F$$

## Geometrical representation of PLS

$$Y = XB + F$$

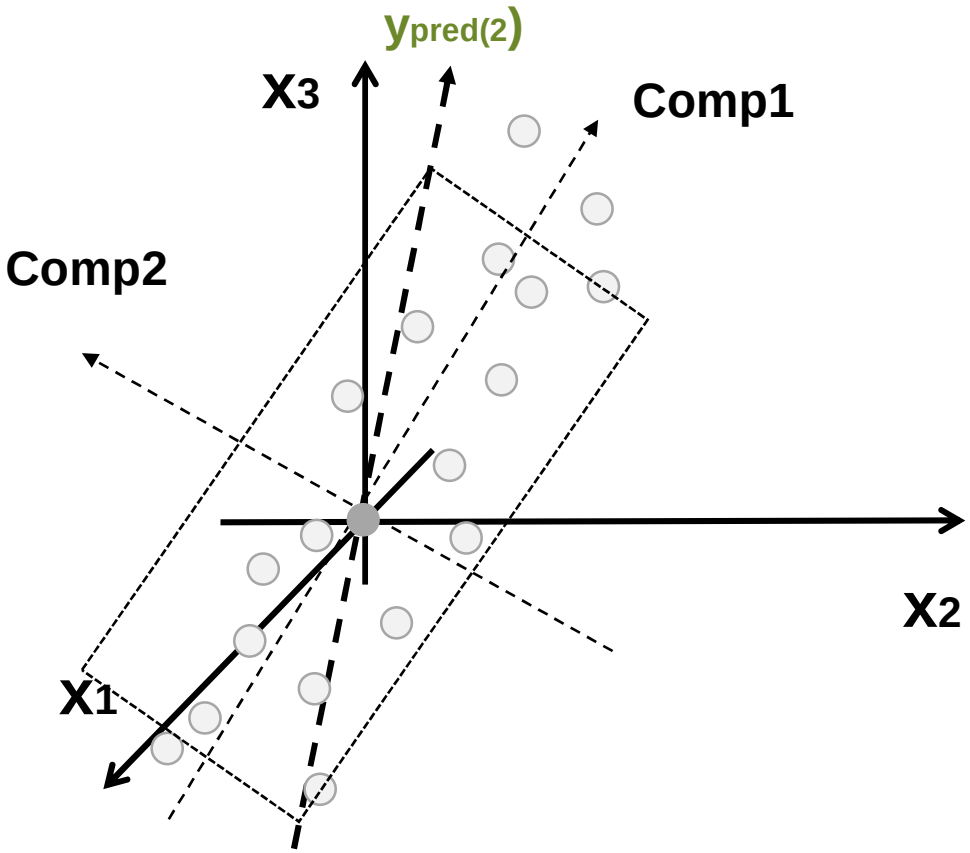


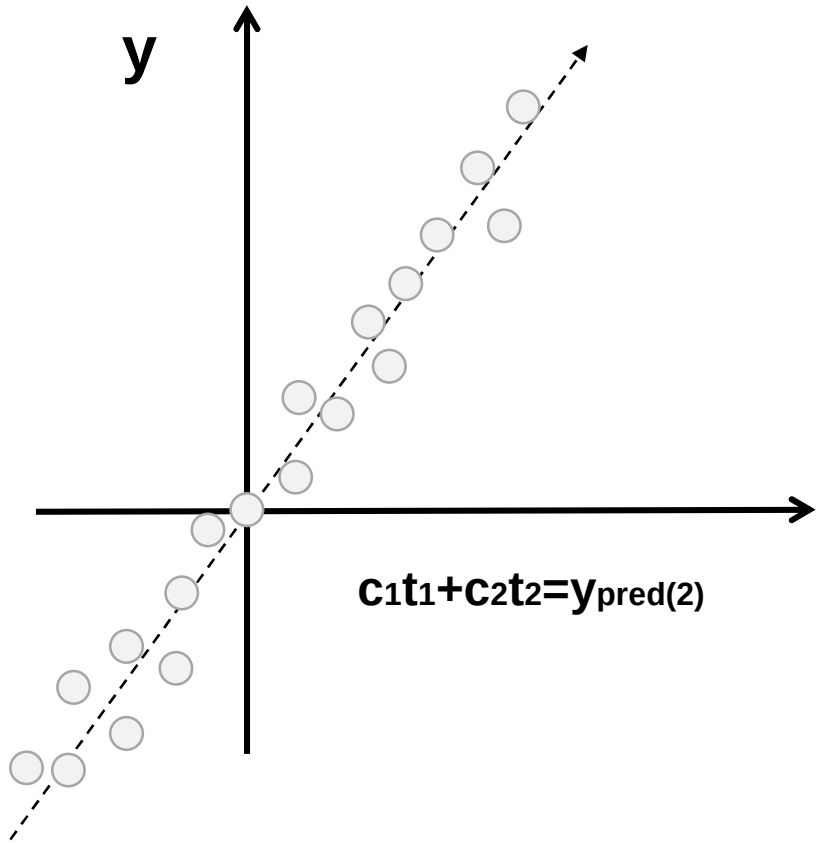




The 1st PLS component will orient itself so that it describes well the point-swarm in the X-space, while also accounting for a good correlation with the y-data.

# Regression - PLS

## Geometrical representation of PLS




With the introduction of the 2nd component the error is minimised.

# Regression - PLS

PLS has become a widely used method for multivariate calibration.

Commonly employed variants of PLS include:

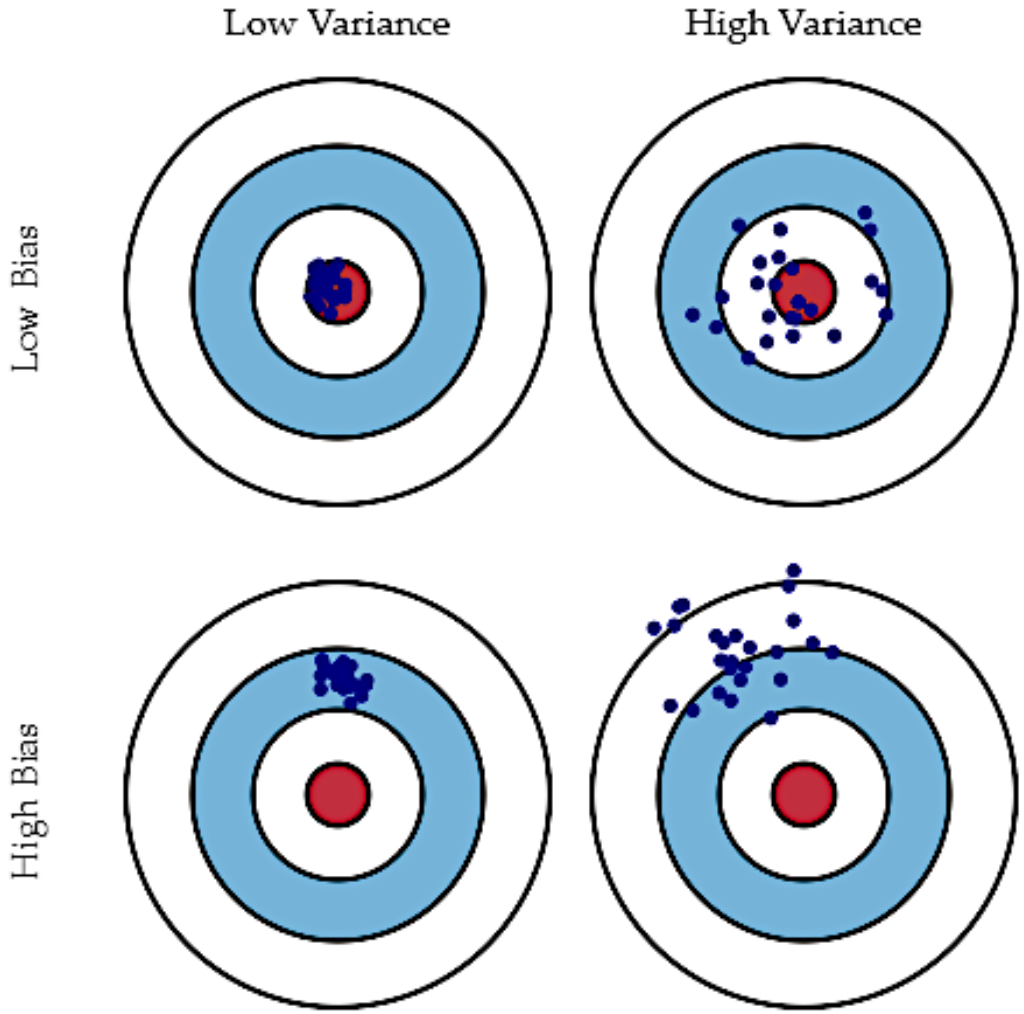
- Non-linear iterative PLS (NIPALS) → Wold, 1966.
- Statistically Inspired Modification of PLS (SIMPLS) → de Jong, 1993.
- SIMPLS is rather extensively employed.

| NIPALS | SIMPLS |
|---|---|
| Calculates weights W, loadings P and scores T | Calculates weights R, loadings P and scores T |
| Transparent | Maximizes covariance for multivariate Y |
| Accurate | Accurate |
| Slow | Very fast |

# Penalised Regression

## L1 and L2 regularisation

- Regularisation improves model performance by ignoring less important features.
- It minimises the validation loss and improves accuracy.
- It avoids overfitting by adding a penalty to non-zero coefficients.
- It shrinks the beta coefficients and regularises the weights around a mid point.



# Penalised Regression

## Ridge Regression (L2 regularisation)

- Used to address the problem of multicollinearity in regression
- We can use a value $\lambda$ to impose a maximum value on the sum of all regression coefficients.

$$\sum_{k=1}^{n} w_k^2 \leq \lambda$$

- By imposing this penalty, we can decrease unimportant parameters (**shrinkage**).

# Penalised Regression

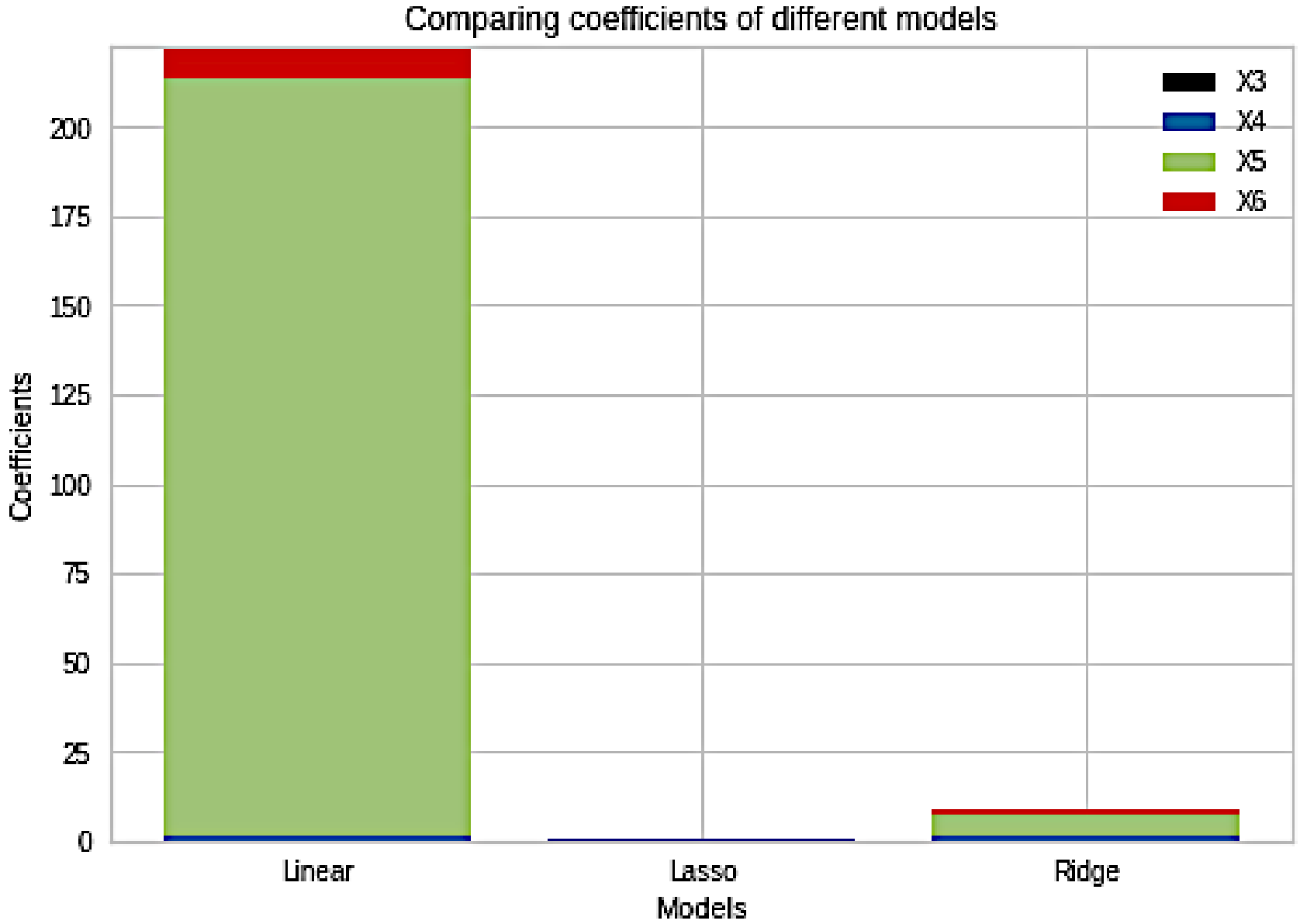
## Lasso (L1 Regularisation)

- **Ridge regression is preferred over lasso when the number of variables is too large and vice versa.**
- The lasso imposes a different constraint on the weights:

$$\sum_{k=1}^{n} |w_k| \leq \lambda$$

- Using a slightly different constraint will give us different results.
- The coefficients of some less important variables are forced to be exactly zero. Only the most significant variables are kept in the final model.

# Penalised Regression



# Penalised Regression - Elastic Net Regression

**Elastic Net:** uses both Lasso and Ridge Regression regularisation to remove all unnecessary coefficients.

ENR = Lasso Regression + Ridge Regression

Elastic Net is better at handling collinearity than ridge and lasso regression.

Elastic Net is more efficient at reducing model complexity by removing more variables.

# Penalised Regression- glmnet package in R

The R function glmnet() can be used for computing penalised logistic regression.

```
glmnet(x, y, family = "binomial", alpha = 1, lambda = NULL)
```

x = matrix of predictor variables

y = the response or outcome variable, which is a binary variable.

family = the response type. Use “binomial” for a binary outcome variable

alpha = the elastic net mixing parameter. It takes values :

a = 1: for lasso regression

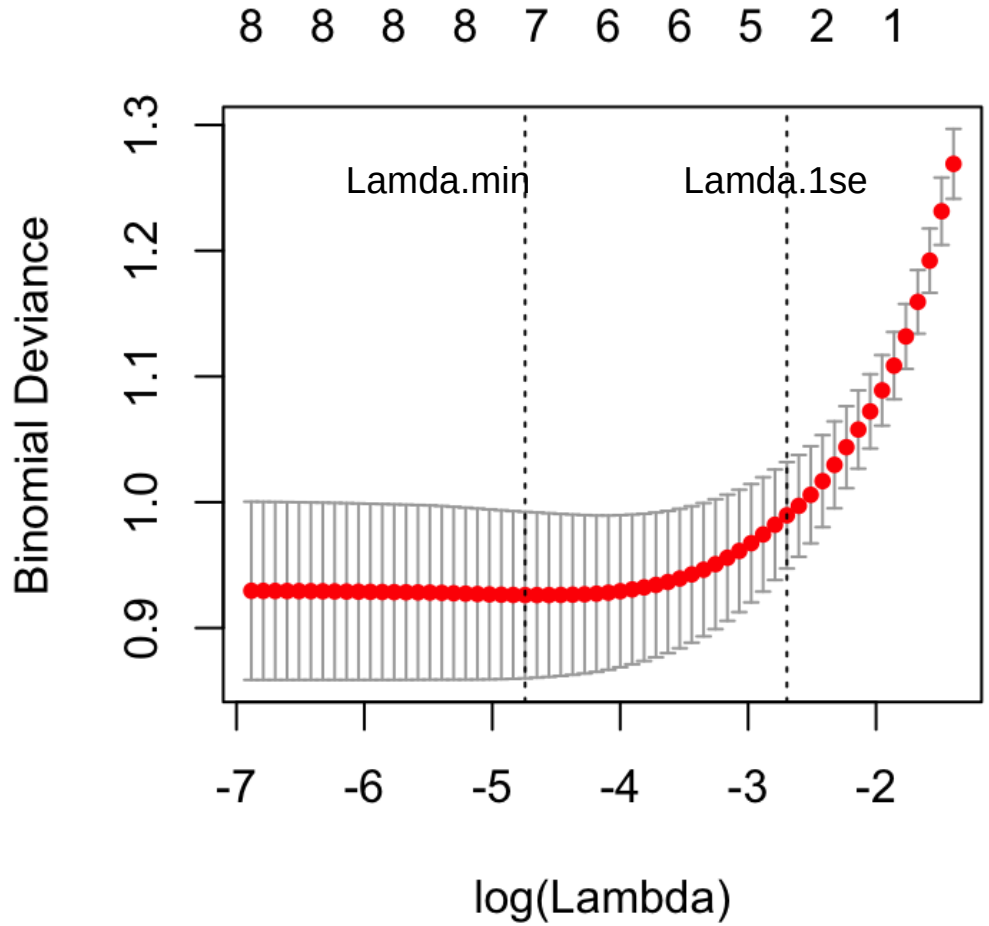
a = 0 : for ridge regression

0< a < 1 for elastic net regression.

lamba: a user defined numeric value defining the amount of shrinkage.

# Penalised Regression - λ parameter optimisation

- High λ values increase the bias and decrease the variance.
- To find the optimal value for λ we can perform cross-validation and select the value of λ that minimizes the cross-validation prediction error rate.
- This can be determined automatically using the function **`cv.glmnet()`**.




```
cv.lasso <- cv.glmnet(x, y,   alpha = 1, family = "binomial")
plot(cv.lasso)
```

# Penalised Regression - λ parameter optimisation

```
# Fit the model with optimal λ
model1 <- glmnet(x, y, alpha = 1, family = "binomial",
lambda = cv.lasso$lambda.min)

# Fit the model with lamda.1se
model2 <- glmnet(x, y, alpha = 1, family = "binomial",
lambda = cv.lasso$lambda.1se)
```

Lamda.min

```
## (Intercept) -8.615615
## pregnant     0.035076
## glucose      0.036916
## pressure     .
## triceps      0.016484
## insulin     -0.000392
## mass         0.030485
## pedigree     0.785506
## age          0.036265
```

Lamda.1se

```
## (Intercept) -4.65750
## pregnant     .
## glucose      0.02628
## pressure     .
## triceps      0.00191
## insulin      .
## mass         .
## pedigree     .
## age          0.01734
```

# Regression - Variable selection

## Stepwise regression

**This is an easier algorithm compared to lasso which gives similar results.**

- It builds a regression model from a set of candidate predictor variables by entering and removing predictors  in a stepwise manner into the model.

# Regression - Variable selection

## Stepwise regression

**The Backward method: Recommended for a modest number of predictors**

- Starts with all available variables, deleting one variable at a time as the regression model progresses.
- At each step the "**F-to-remove**" statistic is calculated for the coefficient of each variable in the model.
- The variable with the **lowest** "**F-to-remove**" statistic is deleted from the model.

# Regression - Variable selection

## Stepwise regression

**The Forward method: Recommended for a large set of predictors**

- Starts the test with no predictor variables adding one at a time as the regression model progresses.
- An "**F-to-add**" statistic is created for each variable *not* in the model.
- The variable with the **highest** "F-to-add" statistic is added to the model.

**The Bidirectional method**: A combination of the above, searching at each step for variables to be included or excluded.

# Regression - Variable selection

## Stepwise regression

| Strengths | Weaknesses |
|---|---|
| Retains only the best predictors (model fine-tuning) | Collinearity is usually a major issue. |
| Easier and faster than other automatic selection methods | If two predictor variables in the model are highly correlated, only one may make it into the model. |
| The order in which variables are removed or added can provide valuable information about the quality of the predictor variables | Adjusted r-squared values might be high, and then dip sharply as the model progresses. If this happens, identify the variables that were added or removed when this happens and adjust the model. |

# Matching ML algorithms to learning tasks

| Machine Learning Algorithms | |
|---|---|
| **Unsupervised Learning Algorithms** | **Learning Task** |
| Association Rules | Pattern detection |
| k-means clustering | Clustering |
| **Supervised learning algorithm** | **Learning Task** |
| Decision Trees | Classification |
| Naïve Bayes | Classification |
| Linear Regression | Numeric prediction |
| Model Trees | Numeric prediction |
| K-NN | Dual use |
| Neural Networks | Dual use |
| Support Vector Machines | Dual use |
| **Meta-Learning Algorithms** | **Learning Task** |
| Bagging | Dual use |
| Boosting | Dual use |
| Random Forests | Dual use |

# Regression - Trees

## Regression Trees and Model Trees

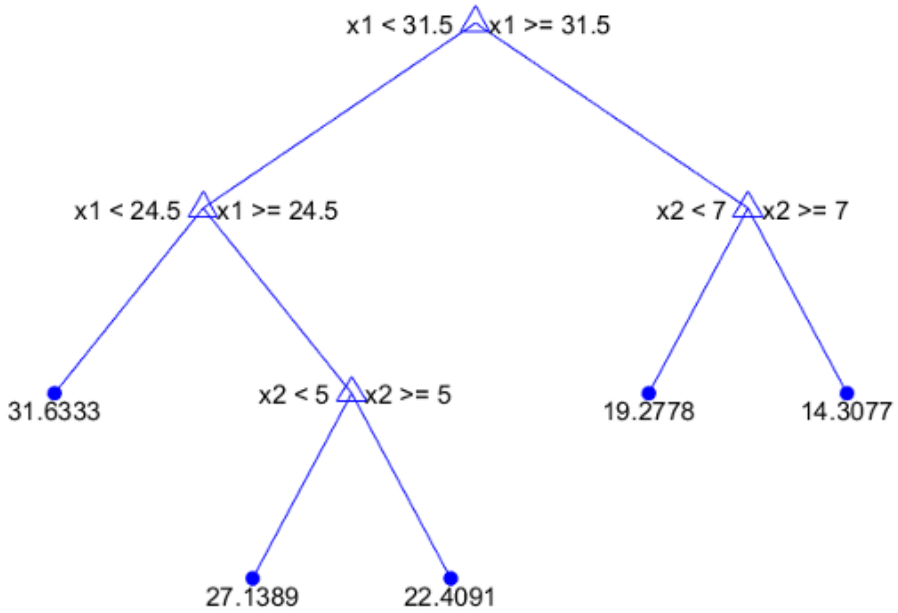
For numeric predictions we can use the same concepts as with classification

Trees for numeric prediction belong to two categories:

- **Regression trees**, which make predictions based on the average value of samples that reach a leaf (No actual regression is involved).
- **Model trees**, which are grown in much the same way as regression trees, but at each leaf, a MLR model is built from the samples reaching that node.

# Regression - Trees

- Trees for numeric prediction are built using the same  divide-and-conquer strategy as with classification.
- Beginning at the root node, the data are partitioned according to the feature that will result in the greatest increase in homogeneity in the outcome after a split is performed.
- For numeric decision trees, homogeneity can be estimated by measuring: variance, standard deviation, or absolute deviation from the mean.

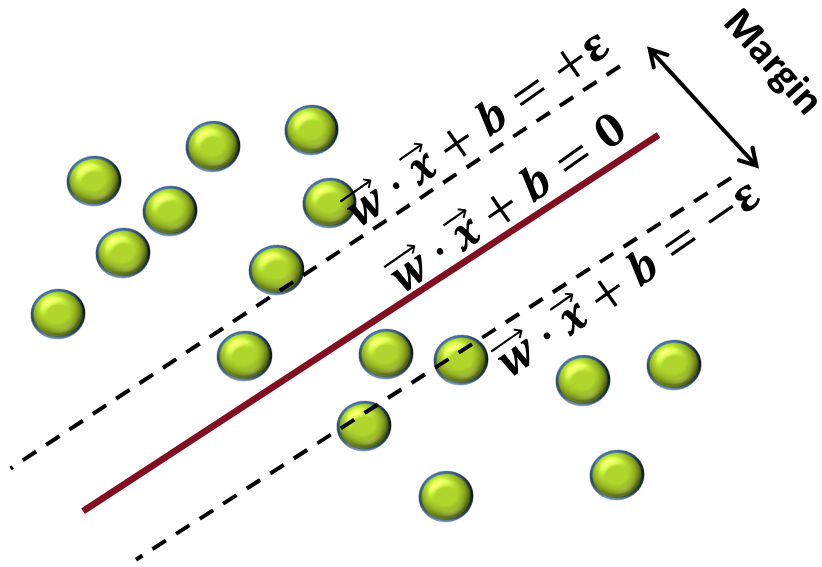


# Regression - SVR

The Support Vector Regression (SVR) uses the same principles as the SVM for classification (maximal margin).

The main difference is that since the output in regression is a number there are infinite possibilities, which add an additional level of complexity to the algorithm.

However, the main idea is always the same: to minimize error, and find the hyperplane which maximizes the margin.

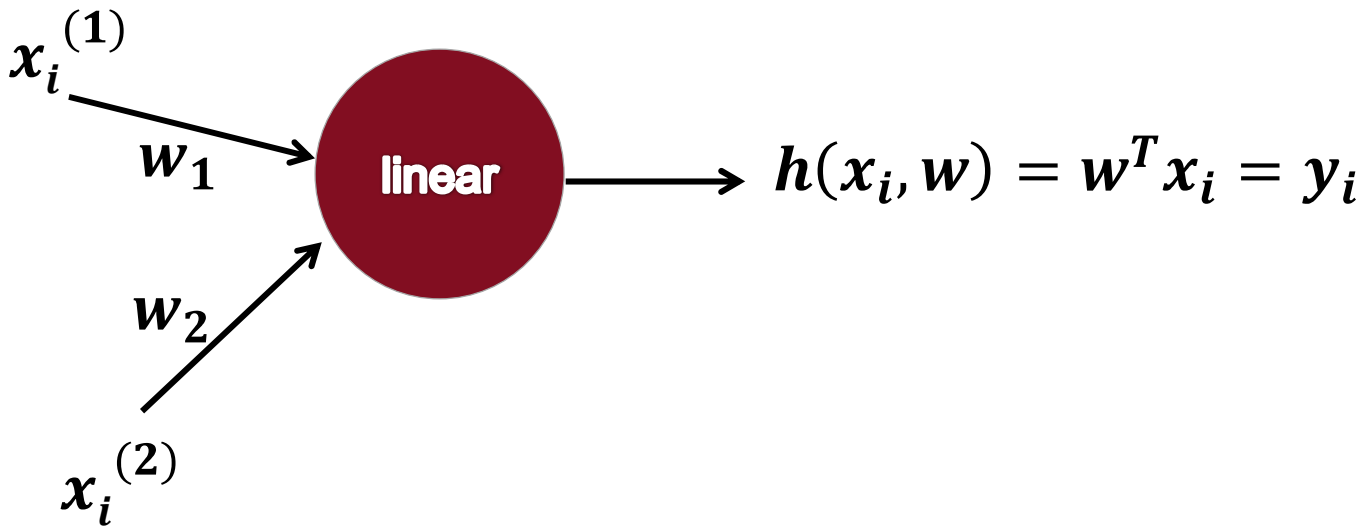


# Regression - NN

Neural networks can be trained to perform regression.

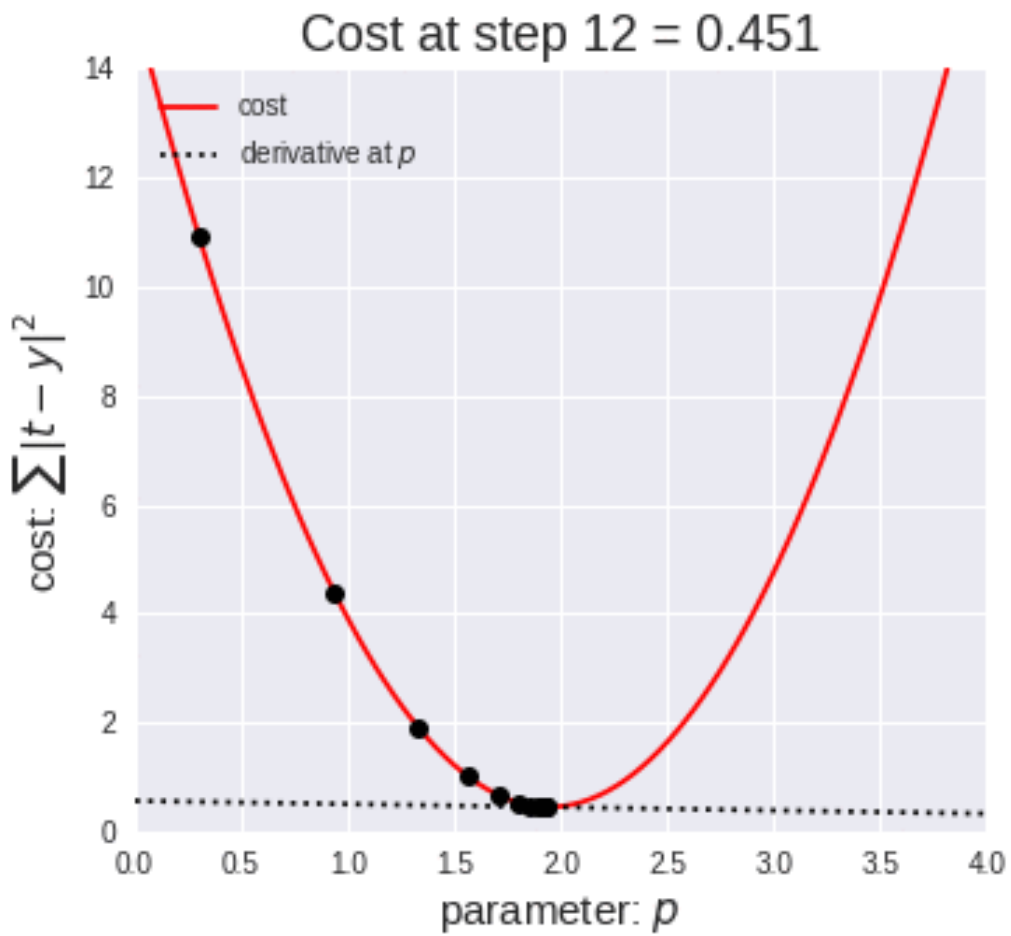
In fact, the simplest neural network performs **least squares regression**.

The following single-layer neural network, with a single node uses a linear activation function:



# Regression - NN

## Gradient Descent in Regression



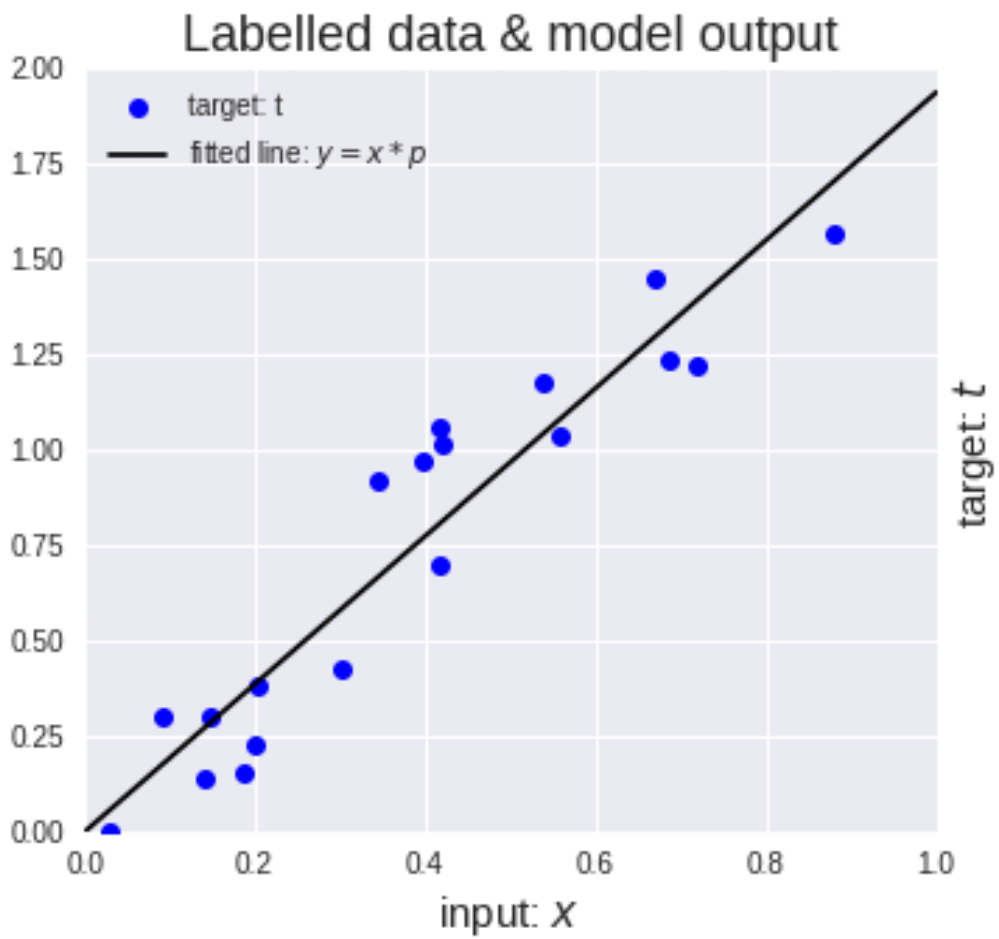




Image Source: https://medium.com/onfido-tech/machine-learning-101-be2e0a86c96a

# Regression: Model building

## Overall Measures of Error

Typically, we combine the individual errors to give an overall model error.

A common way of expressing this is the **root mean square error of prediction (RMSEP)**.

$$RMSEP = \sqrt{\frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{n}}$$

$n$ = number of samples

RMSEP can be used for comparing overall performance between models.

However, we may also want to look carefully at the errors associated with individual samples.

# Summary

- Introduction to linear regression methods for numerical prediction.
- Ordinary Least Squares.
- Multivariate Regression (MLR, PCR, PLS).
- Variable Selection.
- Regression with Machine Learning.
- Regression Trees and Model Trees.
- SVM and NN.


**www.cranfield.ac.uk**

**T: +44 (0)1234 750111**

@cranfielduni

@cranfielduni

/cranfielduni