

Supervised Learning

Elizabeth Savochkina | 6th June



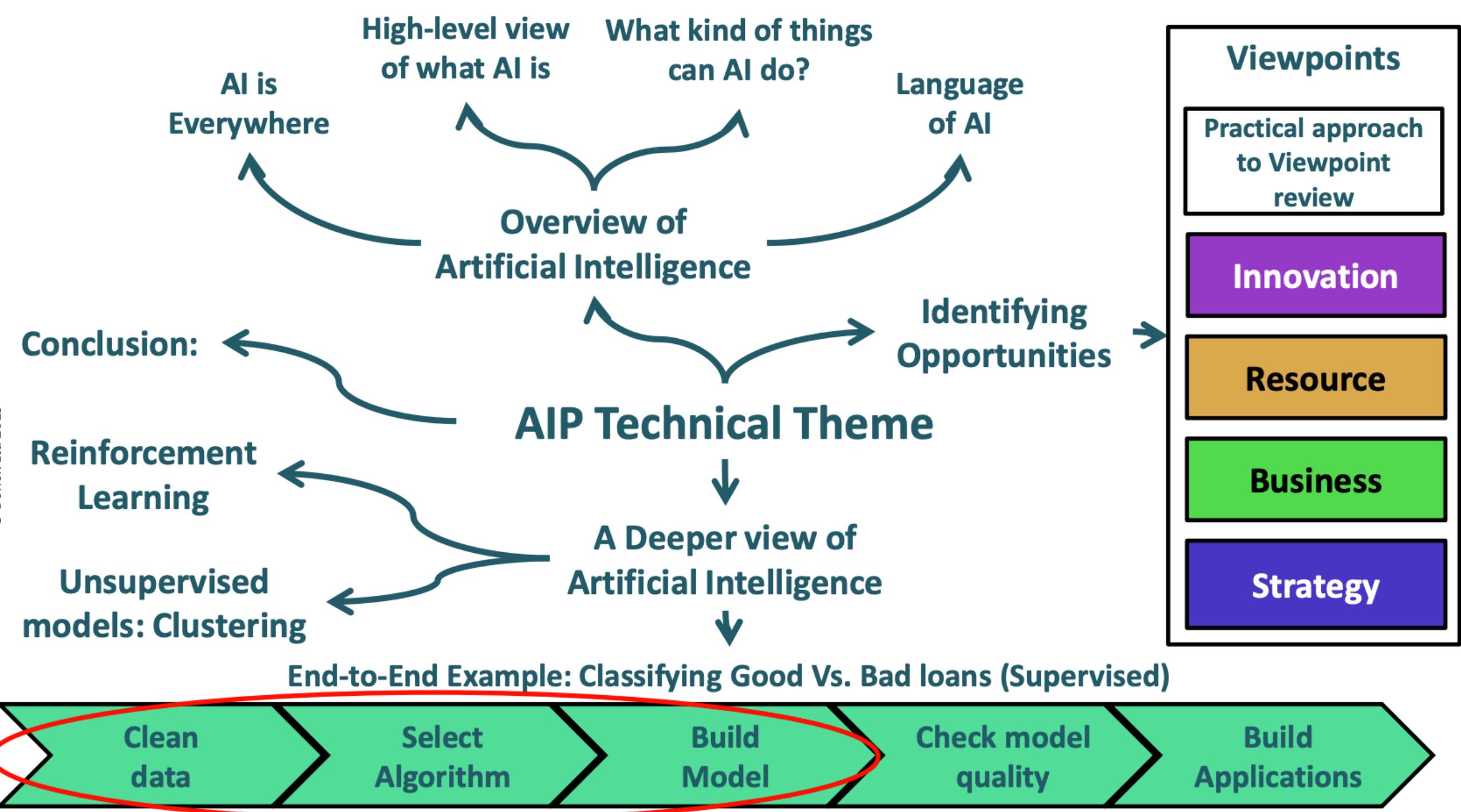
Shape of a Day

Registration
9.00am to 9.30am

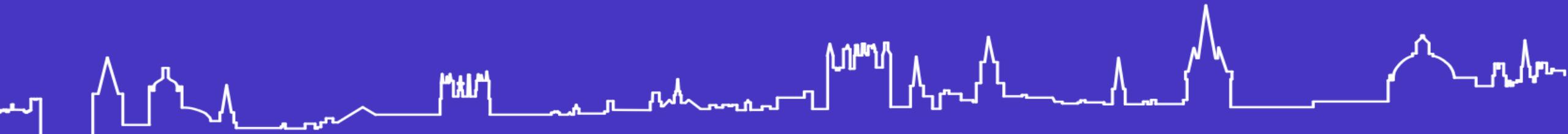
Supervised Learning
9.30am to 11.30am

Break 11.30am to 13.30 pm

**Hands on: RapidMiner,
Python and Jupyter**
13.30am to 15.30pm



End-to-End Example: Classifying Good Vs. Bad loans



Credit Scoring

Why?

It's a high-value problem .. Lending money to people who are likely to default is not fair on lender or borrower

Opportunity?

Significant amount of historic data
Data covers many features
Historic data is labelled

Method?

Build a model that categorises new borrowers based on the probability they will default on their loan



Data Source: The German Credit Data

| Creditability | Account Balance | Duration Of Credit | Payment Status of Previous | Purpose | Credit Amount | Value Savings Stocks | Length of current employment | Instalment per cent | Sex and Marital Status | Guaritors | Duration in Current address | Most valuable available | Age |
|---------------|-----------------|--------------------|----------------------------|---------|---------------|----------------------|------------------------------|---------------------|------------------------|-----------|-----------------------------|-------------------------|-----|
| 1 | 1 | 18 | 4 | 2 | 1049 | 1 | 2 | 4 | 2 | 1 | 4 | 2 | 21 |
| 1 | | 9 | 4 | 0 | 2799 | 1 | 3 | 2 | 3 | 1 | 2 | 1 | 36 |
| 1 | | 12 | 2 | 9 | 841 | 2 | 4 | 2 | 2 | 1 | 4 | 1 | 23 |
| 1 | | | | | 2122 | 1 | 3 | 3 | 3 | 1 | | | 39 |
| 1 | | | | | 2171 | 1 | 3 | 4 | 3 | 1 | | | 38 |
| 1 | | | | | 2241 | 1 | 2 | 1 | 3 | 1 | | | 39 |
| 1 | | | | | 3398 | 1 | 4 | 1 | 3 | 1 | | | 40 |
| 1 | | | | | 1361 | 1 | 2 | 2 | 3 | 1 | | | 65 |
| 1 | | | | | 1098 | | | | | 1 | 4 | 3 | 23 |
| 1 | | | | | 3758 | | | | | 1 | 4 | 4 | 36 |
| 1 | 1 | 11 | 4 | 0 | 3905 | | | | | 1 | 2 | 1 | 36 |
| 1 | 1 | 30 | 4 | 1 | 6187 | | | | | 1 | 4 | 3 | 24 |
| 1 | 1 | 6 | 4 | 3 | 1957 | | | | | 1 | 4 | 3 | 31 |
| 1 | 2 | 48 | 3 | 10 | 7582 | | | | | 1 | 4 | 4 | 31 |
| 1 | 1 | 18 | 2 | 3 | 1936 | | | | | 1 | 4 | 3 | 23 |
| 1 | 1 | 6 | 2 | 3 | 2647 | 3 | 3 | 2 | 3 | 1 | 3 | 1 | 44 |
| 1 | 1 | 11 | 4 | 0 | 3939 | 1 | 1 | 1 | 3 | 1 | 2 | 1 | 40 |
| 1 | 2 | 18 | 2 | 3 | 3213 | 3 | 2 | 1 | 4 | 1 | 3 | 1 | 25 |
| 1 | 2 | 36 | 4 | 3 | 2337 | 1 | 5 | 4 | 3 | 1 | 4 | 1 | 36 |
| 1 | 4 | 11 | 4 | 0 | 7228 | 1 | 3 | 1 | 3 | 1 | 4 | 2 | 39 |

Label in the first column
1 : Good loan
0: Default

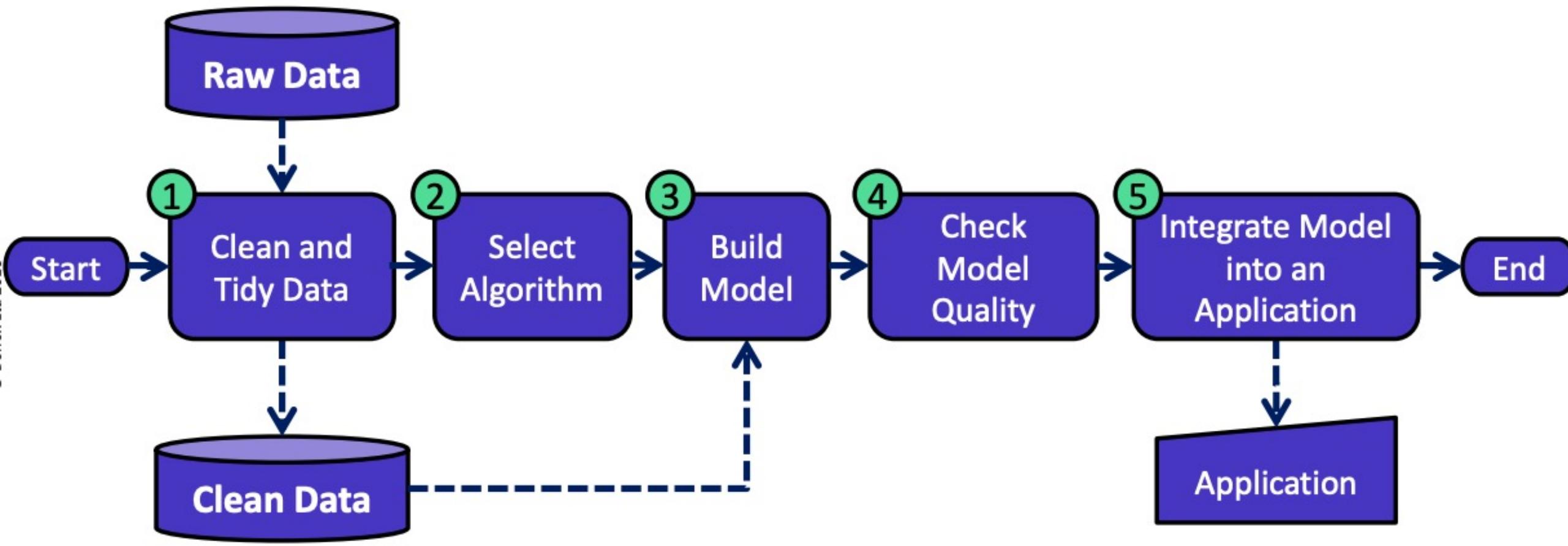
1000 Observations
700 good loans
300 default

20 Features

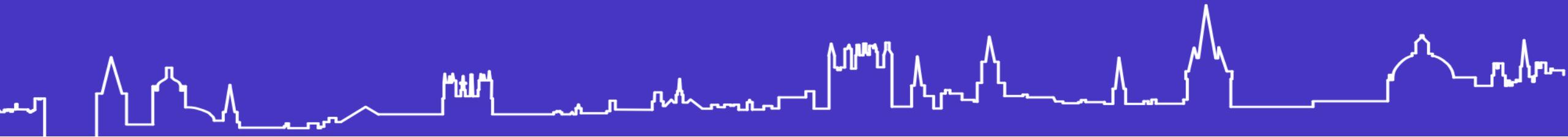
[https://archive.ics.uci.edu/ml/datasets/Statlog+\(German+Credit+Data\)](https://archive.ics.uci.edu/ml/datasets/Statlog+(German+Credit+Data))



Typical ML Workflow



Step 1 : Data Cleansing



Step 1 : Data Cleansing

- Most data will be incomplete and have errors of various types
- We could just delete rows with missing data or errors ..
- But this is very wasteful in data .. You tend to throw away a lot of useful information
- Deleting data can also give biased results

| ID | Name | Age | Credit Score | Loan Amount | Marriage Status |
|----|--------------|-----|--------------|-------------|-----------------|
| 1 | Rob Collins | 28 | 920 | 27,228 | Single |
| 2 | Dave Smith | 43 | 634 | 45,673 | Married |
| 3 | Sue James | 32 | 0 | 41,813 | Divorced |
| 4 | Ankit Gupta | | 720 | 23,645 | Single |
| 5 | Kurt Godel | 51 | 890 | 33,437 | Single |
| 6 | Isaac Newton | 420 | 786 | 39,739 | Bereaved |

Data Cleansing : Many Strategies ..

1

Clean and
Tidy Data

Replace missing value with 'worst case' value

Replace missing value average of other legal values

Alert analyst with a warning message to enable source data to be checked

| ID | Name | Age | Credit Score | Loan Amount | Marriage Status |
|----|--------------|------|--------------|-------------|-----------------|
| 1 | Rob Collins | 28 | 920 | 27,228 | Single |
| 2 | Dave Smith | 43 | 634 | 45,673 | Married |
| 3 | Sue James | 32 | 500 | 41,813 | Divorced |
| 4 | Ankit Gupta | 38.5 | 720 | 23,645 | Single |
| 5 | Kurt Godel | 51 | 890 | 33,437 | Single |
| | Isaac Newton | 420 | 786 | 39,739 | Bereaved |

Turning Classifications into Numbers

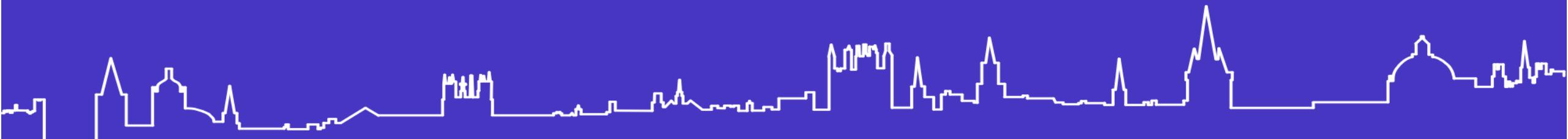
1 Clean and
Tidy Data

- Many of the ML models depend on having numerical data rather than classifications such as “Single”
- Classifiers can be translated into ‘1-hot encoding’

| Marriage Status | Single | Married | Divorced | Bereaved |
|-----------------|--------|---------|----------|----------|
| Single | 1 | 0 | 0 | 0 |
| Married | 0 | 1 | 0 | 0 |
| Divorced | 0 | 0 | 1 | 0 |
| Single | 1 | 0 | 0 | 0 |
| Single | 1 | 0 | 0 | 0 |
| Bereaved | 0 | 0 | 0 | 1 |

*Binary Classification

Data Cleaning: Practical Example



German Credit Data : Data Source

1 Clean and
Tidy Data

- The data in this section is publically available at the UCI Machine Learning Repository

[https://archive.ics.uci.edu/ml/datasets/Statlog+\(German+Credit+Data\)](https://archive.ics.uci.edu/ml/datasets/Statlog+(German+Credit+Data))

<https://bit.ly/1K3bcku>

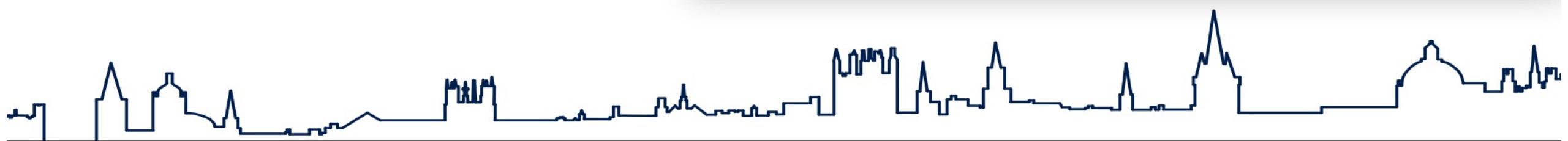
The screenshot shows a web browser window with the UCI Machine Learning Repository logo and navigation menu. The main content area displays the "Statlog (German Credit Data) Data Set". It includes a brief abstract, download links for "Data Folder" and "Data Set Description", and a summary table of dataset characteristics. Below the table, there is a "Source" section with contact information for Professor Dr. Hans-Martin Groß at the University of Hamburg.

| Data Set Characteristics | Multivariate | Number of Instances | 1000 | Area | Financial |
|---------------------------|----------------------|----------------------|------|---------------------|-----------|
| Attribute Characteristics | Categorical, Integer | Number of Attributes | 26 | Data Donor(s) | FATIGUE |
| Assessment Task | Classification | Missing Values? | No | Number of Web Links | 000000 |

Source:
Professor Dr. Hans-Martin Groß
Institut für Statistik und Ökonometrie
Universität Hamburg
FIR Wirtschaftsinformatik
Von-Wedel-Platz 3
20146 Hamburg, Germany

Raw Data

- The data comes as a text file, with coded values
- This needs some tidying before it can be used to build models



A screenshot of a Windows Notepad window titled "germancredit.txt - Notepad". The window displays a single column of text data. The data consists of approximately 1000 rows, each containing a sequence of characters. The characters are mostly lowercase letters (a-z) and numbers (0-9), separated by spaces. Some rows begin with a letter like 'A' or 'B'. The text is monospaced and appears to be a tabular dataset where columns are separated by spaces.

```
B11 6 A34 A43 1169 A65 A75 4 A93 A101 4 A121 67 A143 A152 2 A173 1 A192 A201 1  
A12 48 A32 A43 5951 A61 A73 2 A92 A101 2 A121 22 A143 A152 1 A173 1 A191 A201 2  
A14 12 A34 A46 2096 A61 A74 2 A93 A101 3 A121 49 A143 A152 1 A172 2 A191 A201 1  
A11 42 A32 A42 7882 A61 A74 2 A93 A103 4 A122 45 A143 A153 1 A173 2 A191 A201 1  
A11 24 A33 A48 4878 A61 A73 3 A93 A101 4 A124 53 A143 A153 2 A173 2 A191 A201 2  
A14 36 A32 A46 8855 A65 A73 2 A93 A101 4 A124 35 A143 A153 1 A172 2 A192 A201 1  
A14 24 A32 A42 2835 A63 A75 3 A93 A101 4 A122 53 A143 A152 1 A173 1 A191 A201 1  
A12 36 A32 A41 6948 A61 A73 2 A93 A101 2 A123 35 A143 A151 1 A174 1 A192 A201 1  
A14 12 A32 A43 3859 A66 A74 2 A91 A101 4 A121 61 A143 A152 1 A172 1 A191 A201 1  
A12 38 A34 A40 5234 A61 A71 4 A94 A101 2 A123 28 A143 A152 2 A174 1 A191 A201 2  
A12 12 A32 A48 1295 A61 A72 3 A92 A101 1 A123 25 A143 A151 1 A173 1 A191 A201 2  
A11 48 A32 A49 4388 A61 A72 3 A92 A101 4 A122 24 A143 A151 1 A173 1 A191 A201 2  
A12 12 A32 A43 1567 A61 A73 1 A92 A101 1 A123 22 A143 A152 1 A173 1 A192 A201 1  
A11 24 A34 A48 1199 A61 A75 4 A93 A101 4 A123 60 A143 A152 2 A172 1 A191 A201 2  
A11 15 A32 A48 1483 A61 A73 2 A92 A101 4 A123 28 A143 A151 1 A173 1 A191 A201 1  
A11 24 A32 A43 1282 A62 A73 4 A92 A101 2 A123 32 A143 A152 1 A172 1 A191 A201 2  
A14 24 A34 A43 2424 A65 A75 4 A93 A101 4 A122 53 A143 A152 2 A173 1 A191 A201 1  
A11 30 A30 A49 8872 A65 A72 2 A93 A101 3 A123 25 A141 A152 3 A173 1 A191 A201 1  
A12 24 A32 A41 12579 A61 A75 4 A92 A101 2 A124 44 A143 A153 1 A174 1 A192 A201 2
```

Data Description

1

Clean and
Tidy Data

- The data also comes with a descriptive file to help you interpret each column

Attribute description for German credit data

Attribute 1: (qualitative)
status of existing checking account
A1_1 -> <= 0: 0=NO
A1_2 >= 0 <= 3: 1=NO
A1_3 >= 3 <= 5: 2=YES
A1_4 >= 5: 3=CHECKING account

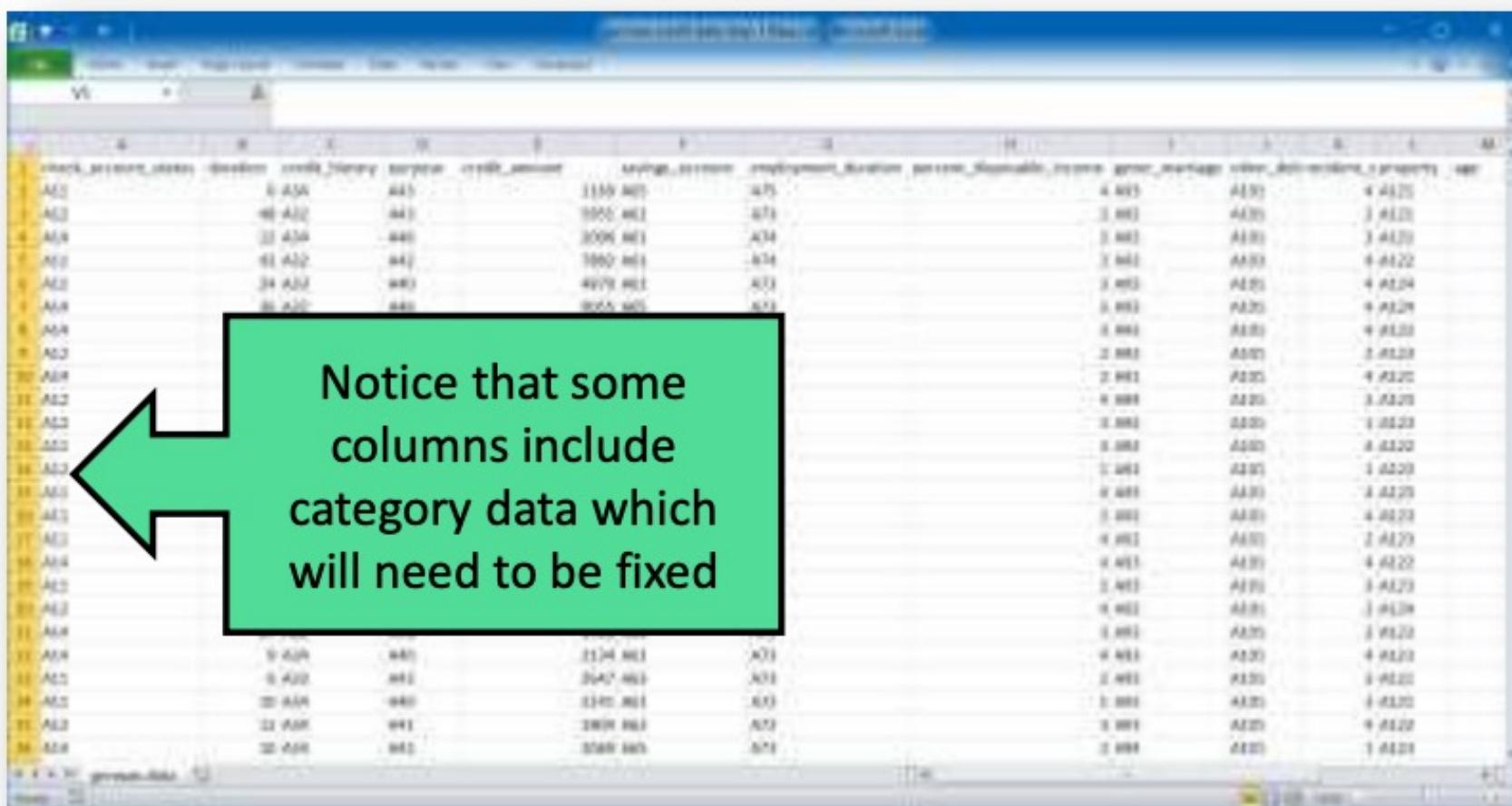
Attribute 2: (numerical)
duration in month

Attribute 3: (qualitative)
credit history
A3_1 no credits taken
A3_2 all credits at this firm paid back duly
A3_3 all credits at this firm paid back duly
A3_4 existing credits paid back duly till now
A3_5 delay in paying off all the past
A3_6 critical amount/
other credits existing not at this bank

Attribute 4: (qualitative)
Purpose
A4_1 car (auto)
A4_2 used goods
A4_3 Domestic equipment
A4_4 business/income
A4_5 domestic appliances
A4_6 repairs
A4_7 television
A4_8 vacation - non durables
A4_9 furnishings
A4_10 forename
A4_11 others

Column (Feature) Names

- I first used Excel to add a row at the top of the file containing names for each column (feature)
- Then I saved as a comma-separated variable file



| id | age | workclass | fnlwgt | education | education-num | marital-status | occupation | relationship | race | sex | capital-gain | capital-loss | hours-per-week | census-income |
|-----|-----|-----------|--------|-----------|---------------|-----------------------|--------------|---------------|-------|------|--------------|--------------|----------------|---------------|
| A01 | 35 | Private | 16070 | HS-grad | 9 | Married-spouse-absent | Adm-clerical | Not-in-family | White | Male | 0 | 0 | 40 | <=50K |
| A02 | 38 | Private | 21000 | HS-grad | 9 | Married-spouse-absent | Adm-clerical | Not-in-family | White | Male | 0 | 0 | 40 | <=50K |
| A03 | 33 | Private | 26600 | HS-grad | 9 | Married-spouse-absent | Adm-clerical | Not-in-family | White | Male | 0 | 0 | 40 | <=50K |
| A04 | 34 | Private | 26600 | HS-grad | 9 | Married-spouse-absent | Adm-clerical | Not-in-family | White | Male | 0 | 0 | 40 | <=50K |
| A05 | 34 | Private | 40700 | HS-grad | 9 | Married-spouse-absent | Adm-clerical | Not-in-family | White | Male | 0 | 0 | 40 | <=50K |
| A06 | 36 | Private | 20500 | HS-grad | 9 | Married-spouse-absent | Adm-clerical | Not-in-family | White | Male | 0 | 0 | 40 | <=50K |
| A07 | 36 | Private | 20500 | HS-grad | 9 | Married-spouse-absent | Adm-clerical | Not-in-family | White | Male | 0 | 0 | 40 | <=50K |
| A08 | 36 | Private | 20500 | HS-grad | 9 | Married-spouse-absent | Adm-clerical | Not-in-family | White | Male | 0 | 0 | 40 | <=50K |
| A09 | 36 | Private | 20500 | HS-grad | 9 | Married-spouse-absent | Adm-clerical | Not-in-family | White | Male | 0 | 0 | 40 | <=50K |
| A10 | 36 | Private | 20500 | HS-grad | 9 | Married-spouse-absent | Adm-clerical | Not-in-family | White | Male | 0 | 0 | 40 | <=50K |
| A11 | 36 | Private | 20500 | HS-grad | 9 | Married-spouse-absent | Adm-clerical | Not-in-family | White | Male | 0 | 0 | 40 | <=50K |
| A12 | 36 | Private | 20500 | HS-grad | 9 | Married-spouse-absent | Adm-clerical | Not-in-family | White | Male | 0 | 0 | 40 | <=50K |
| A13 | 36 | Private | 20500 | HS-grad | 9 | Married-spouse-absent | Adm-clerical | Not-in-family | White | Male | 0 | 0 | 40 | <=50K |
| A14 | 36 | Private | 20500 | HS-grad | 9 | Married-spouse-absent | Adm-clerical | Not-in-family | White | Male | 0 | 0 | 40 | <=50K |
| A15 | 36 | Private | 20500 | HS-grad | 9 | Married-spouse-absent | Adm-clerical | Not-in-family | White | Male | 0 | 0 | 40 | <=50K |
| A16 | 36 | Private | 20500 | HS-grad | 9 | Married-spouse-absent | Adm-clerical | Not-in-family | White | Male | 0 | 0 | 40 | <=50K |
| A17 | 36 | Private | 20500 | HS-grad | 9 | Married-spouse-absent | Adm-clerical | Not-in-family | White | Male | 0 | 0 | 40 | <=50K |
| A18 | 36 | Private | 20500 | HS-grad | 9 | Married-spouse-absent | Adm-clerical | Not-in-family | White | Male | 0 | 0 | 40 | <=50K |
| A19 | 36 | Private | 20500 | HS-grad | 9 | Married-spouse-absent | Adm-clerical | Not-in-family | White | Male | 0 | 0 | 40 | <=50K |
| A20 | 36 | Private | 20500 | HS-grad | 9 | Married-spouse-absent | Adm-clerical | Not-in-family | White | Male | 0 | 0 | 40 | <=50K |

Notice that some columns include category data which will need to be fixed

Data Cleansing in Python

1

Clean and
Tidy Data

- Python and the various libraries include many functions that support data cleansing,
- For example, detecting and repairing
 - Missing values
 - Zeros
 - Out-of-range values

The screenshot shows a Jupyter Notebook interface with the title "jupyter Simple Data Cleaning Example Last Checkpoint 15 minutes ago (untrusted)". The notebook has a toolbar with File, Edit, View, Insert, Cell, Kernel, Widgets, Help, and a Run button. Below the toolbar is a toolbar with icons for file operations like Open, Save, and Print, and a Run button. The main area contains a cell titled "Detecting and fixing missing values in data". The cell content is as follows:

```
In [3]: import pandas as pd
import numpy as np

Create a set of data with a few missing (NaN) values

In [14]: df = pd.DataFrame(np.random.randn(5,5))
df[df > 0.9] = pd.np.nan
print(df)
```

| | 0 | 1 | 2 | 3 | 4 |
|---|-----------|-----------|-----------|-----------|-----------|
| 0 | -0.251761 | -0.690815 | -1.335982 | 0.448518 | -0.067843 |
| 1 | -0.712657 | 0.870196 | 0.756888 | 0.167461 | -0.683550 |
| 2 | -0.551241 | -0.765534 | -0.755561 | -0.155567 | -0.555555 |
| 3 | -0.551241 | -0.765534 | -0.755561 | -0.155567 | -0.555555 |
| 4 | -0.551241 | -0.765534 | -0.755561 | -0.155567 | -0.555555 |

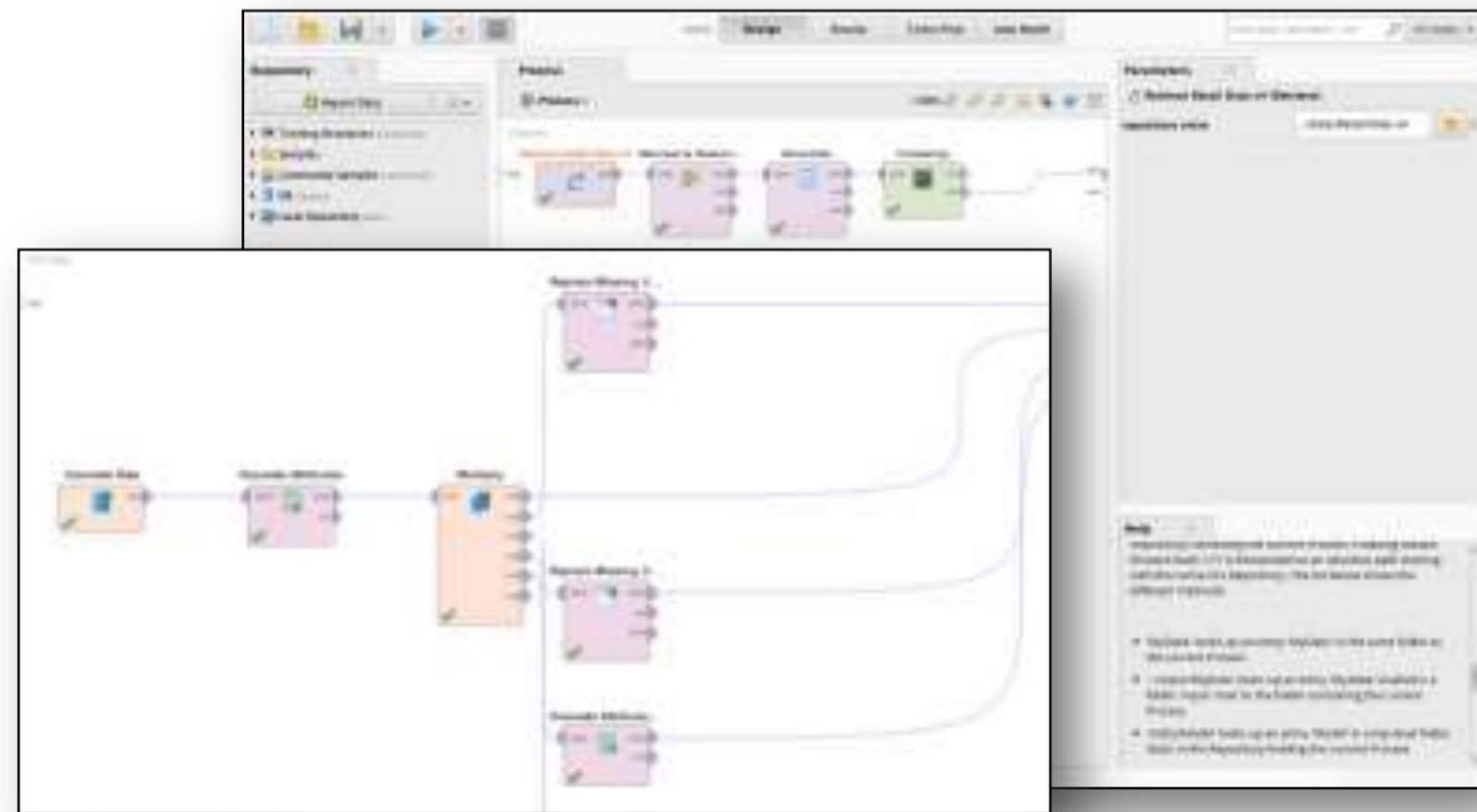
Demonstration: 03-01 – Loading and Cleaning Data”

Fix Missing Data in RapidMiner

1

Clean and
Tidy Data

- RapidMiner also provides these functions ..
- ..but presented as graphic blocks rather than textual computer code

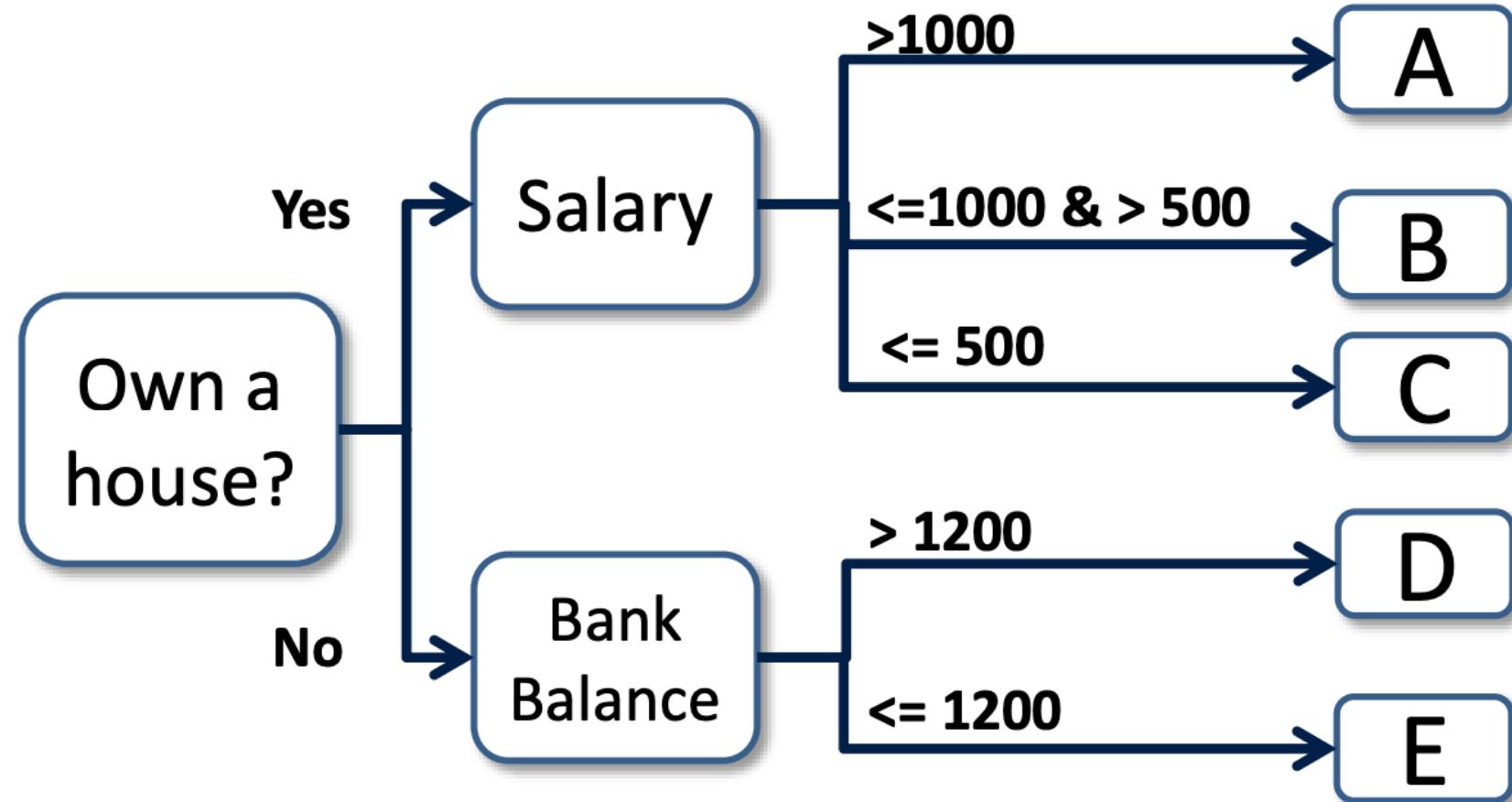


Step 2 : Selecting an Algorithm



Classification using Decision Trees

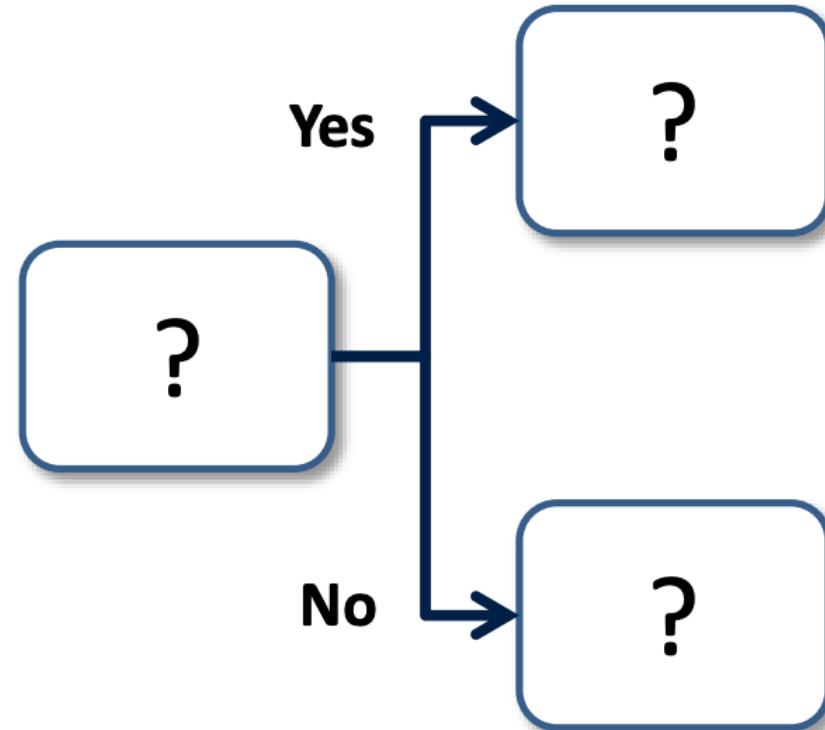
- The Decision Tree algorithm analyses data and builds rules based on previous examples
- Each decision step is selected by the algorithm to generate the most 'pure' set at the next level down



Decision Trees - Good and Bad

Good

- Simple to use
- Easy to explain the resulting logic (transparent)
- Works with 'category' data as well as numbers



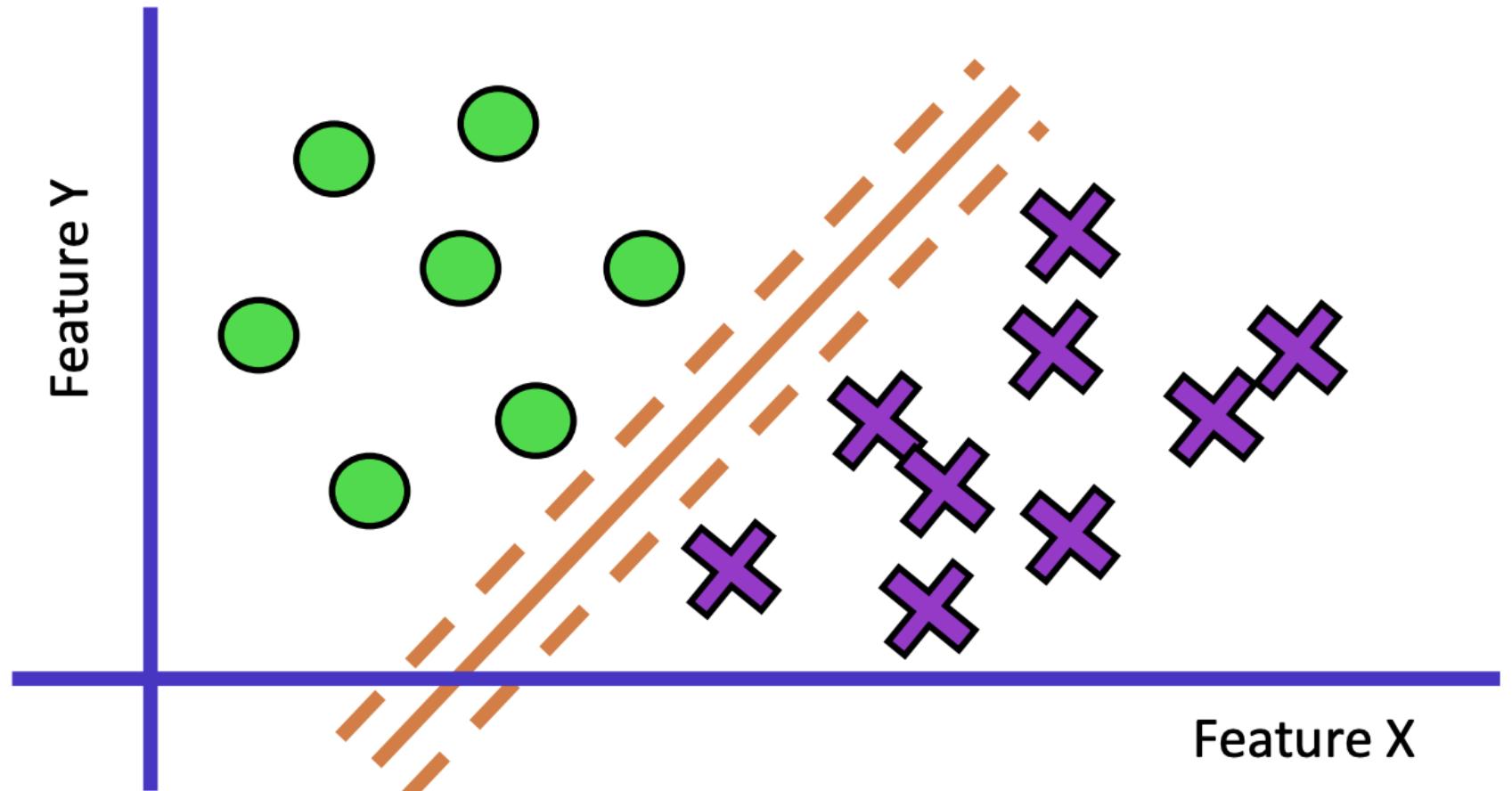
Bad

- Trees can grow very deep
- Tendency to 'over-fit' the data
 - I.e. result is too restricted and does not work well with new data



Support Vector Machine (SVM)

- Wiggle a line around until it neatly divides two classes with the biggest possible gap
- If there is more than two dimensions this will need to be a 'plane'



Logistic Regression

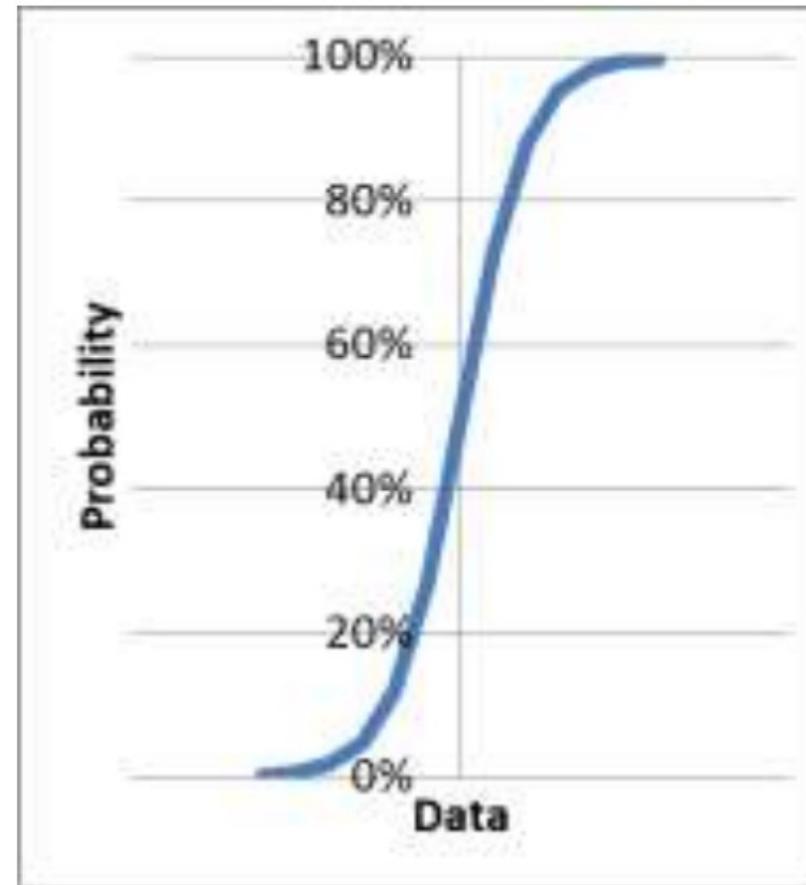
- Create an 'S' shaped curve that fits known data points
- Curve runs from 0% to 100% and represents the probability of the item being in a particular classification
- Estimates the probability of an event occurring (A or not A), based on a given dataset of independent variables



Logistic Regression – Good and Bad

Good

- Gives a probability of the item being in a classification
- Better than simple 'Yes' / No result as we can set our own level of risk of getting the wrong answer

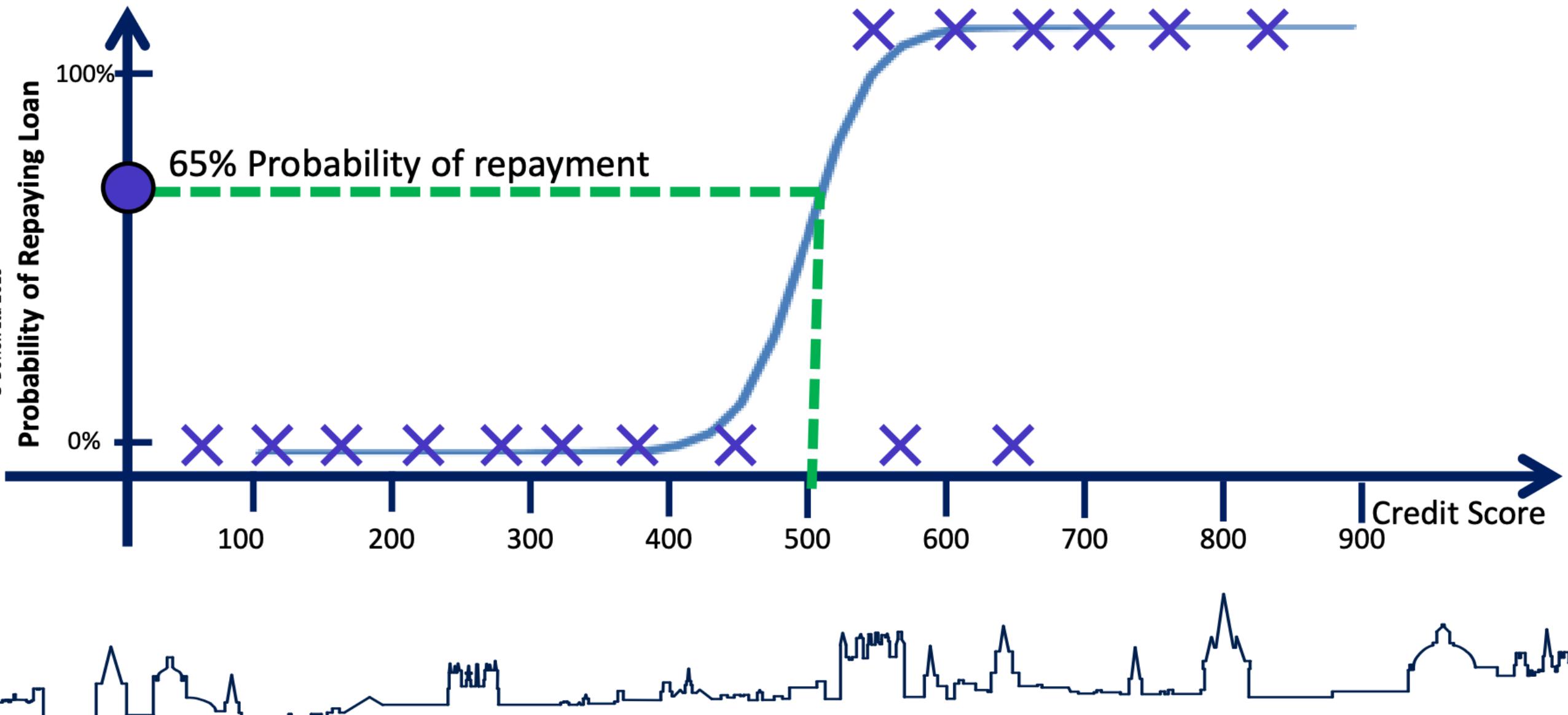


Bad

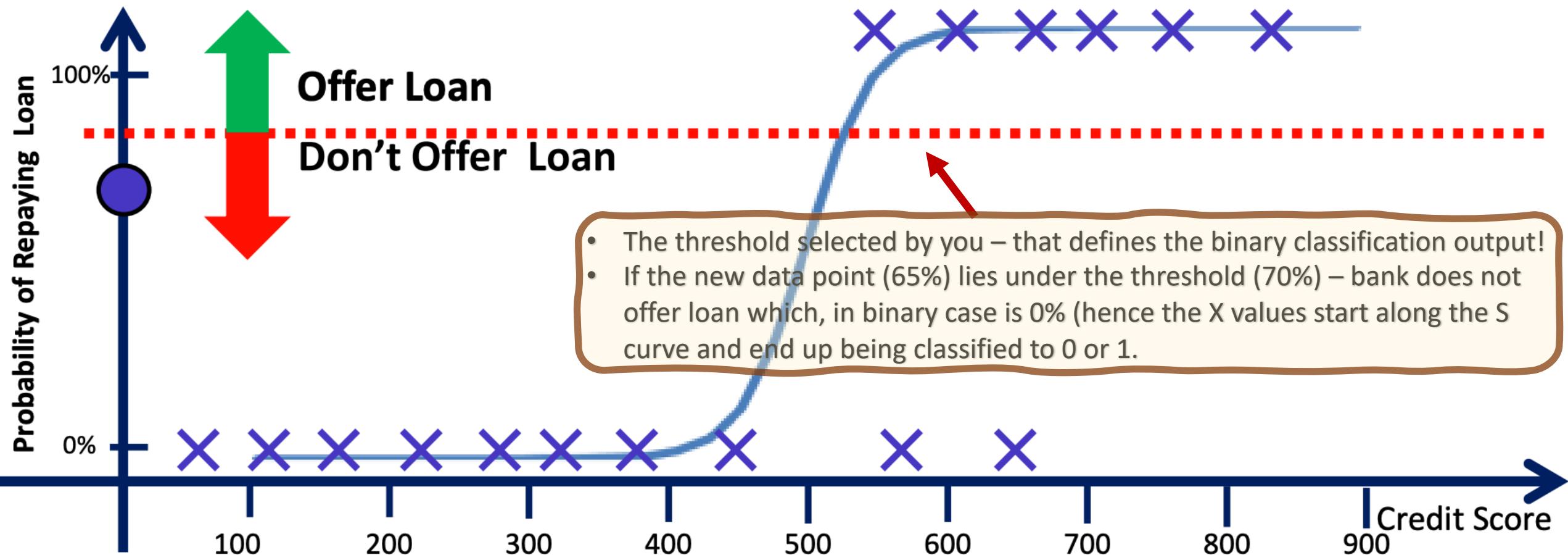
- Does not work well with lots (1000s) of different features (dimensions)
- Not so good with category data inputs

Logistic Regression to Predict Probability

2 Select Algorithm



Setting the Decision Threshold



Step 3 : Building a Model



Building the Model

3

Build
Model

- Building models is made significantly easier as a result of a huge range of pre-built tools and libraries
- You can draw on the resources of companies such as Google and Facebook who have made their software libraries publically available for free



TensorFlow



Keras

pandas
 $y_1 = \beta_1 x_1 + \beta_2 x_2$



NumPy



For Example .. Credit Analysis

Get the data from a file
into your program

```
german_credit_data = pd.read_csv("german_credit.csv")
```

Separate out the 'result' column
(label) indicating if the person
defaulted on their loan

```
X = german_credit_data.loc[:, 'Account_Balance':'Foreign_Worker']
Y = german_credit_data['Creditability']
```

Create the tool that is used to
build the model – in this case,
from an existing library of solvers

```
logModel = LogisticRegression(solver='liblinear')
```

Build the model!

```
logModel = logModel.fit(X, Y)
```



It Can't be that Easy!

- It isn't – it would be misleading to say it was
- But some powerful models are surprisingly easy, given the available tools
- Bespoke ML and more complex models are, of course, difficult!

PhD+ in
Mathematics Data
Science / ML / AI

MSc in Data Science
/ ML / AI

Experienced
Programmer with
some ML training

Primary school level
use of Python

Creating new types of ML. Building libraries.
Building ML for new and unusual applications.

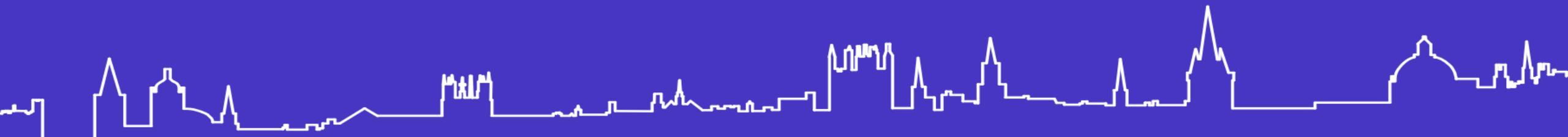
Building sophisticated models from more complex
data sets. Using advanced tools including Neural
Networks. Tuning and optimising models

Building and running 'real' models using existing ML
tools and libraries. Building simpler ML applications

Building and running simple models based on many
tutorials and worked examples on the Internet



Model Building Practical Example



Step 3 : Build the Model

- Model building can be relatively straightforward for many types of ML model due to the powerful libraries and tools that are publically available

jupyter German Credit v2 (Last Checkpoint: an hour ago) (autosave)

File Edit View Insert Cell Kernel Widgets Help

Step 3 Build the Model

In [22]: `from sklearn.linear_model import LogisticRegression`

In [23]: `X = credit_data.drop(['A201','dependent'], axis=1)`
`Y = credit_data['default']`

In [24]: `logModel = LogisticRegression(solver='liblinear')`

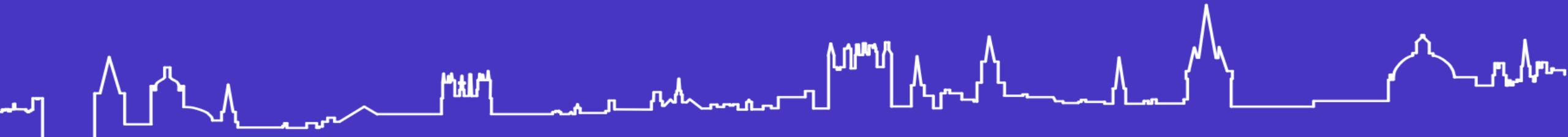
In [25]: `logModel = logModel.fit(X,Y)`

In [26]: `print(logModel.score(X, Y))`

0.762

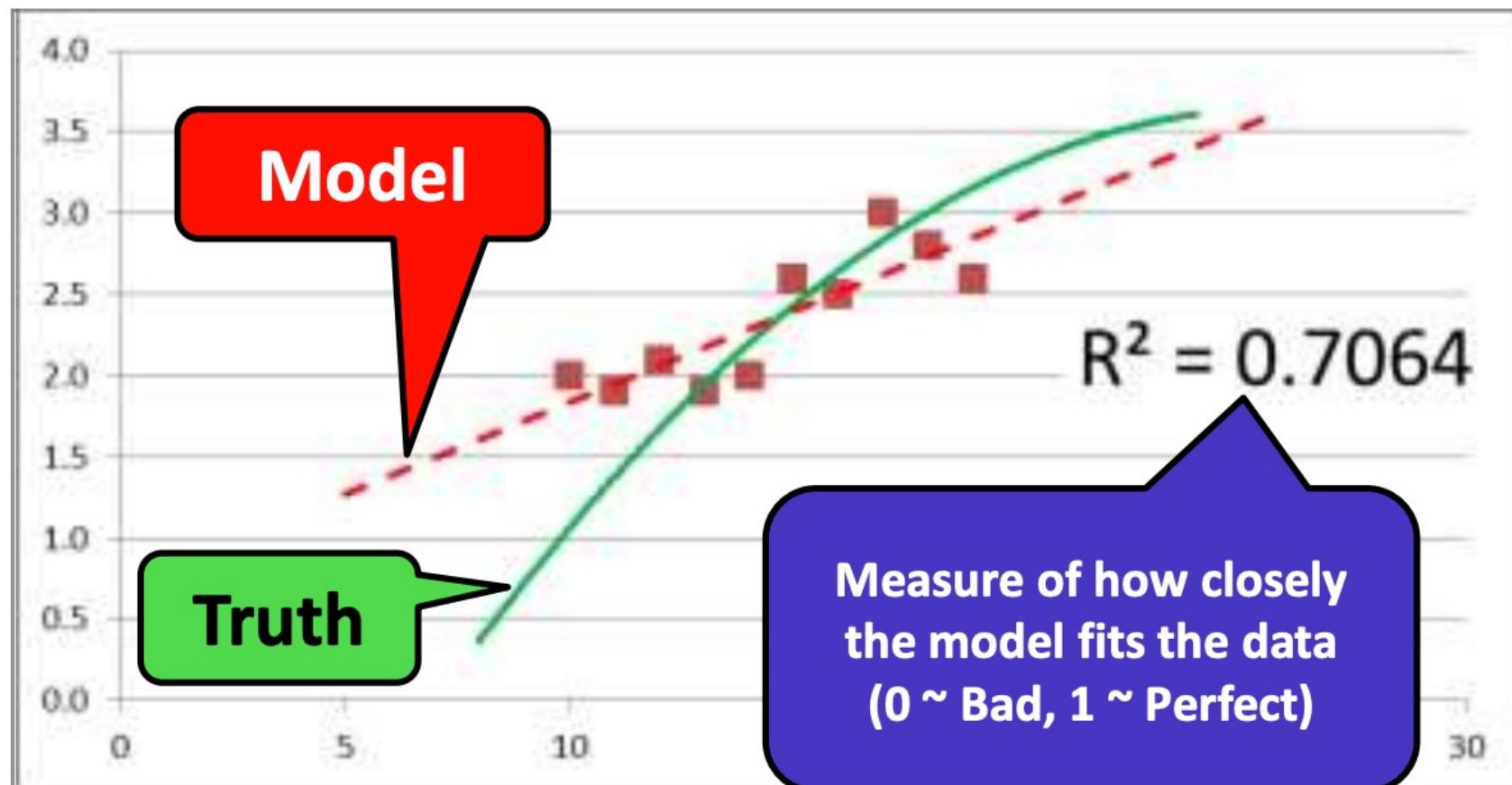
Demonstration: “03-02 – Building the Model”

When Models go Bad

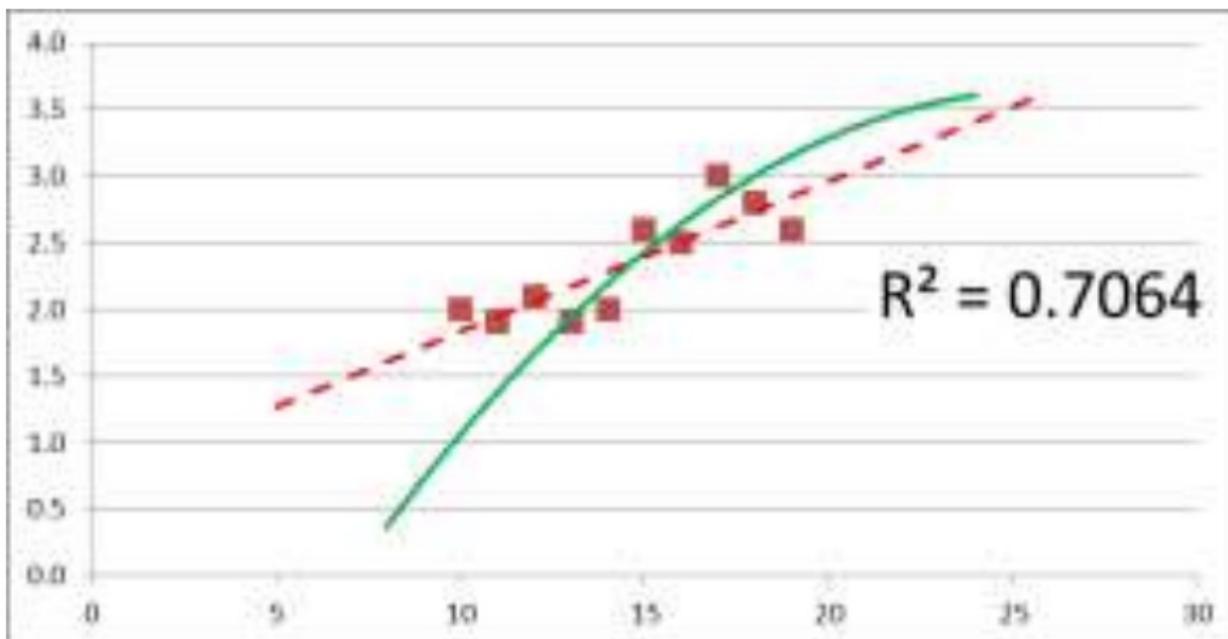


Fitting a Straight Line to the Data

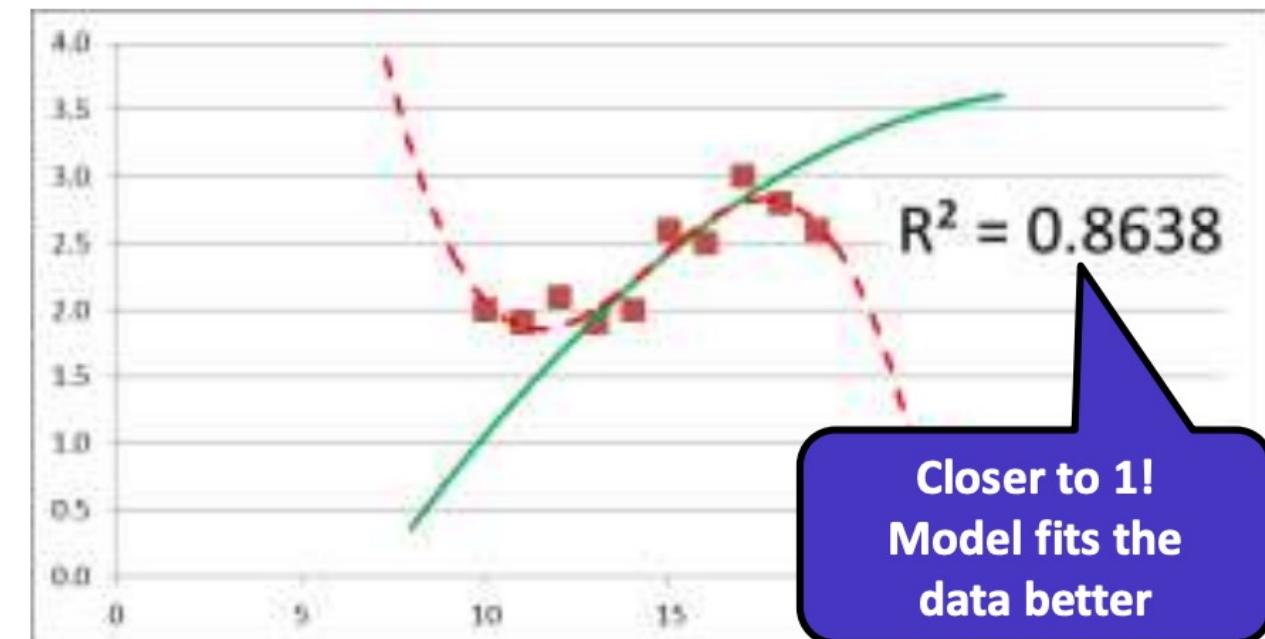
| X | Truth | Measurement |
|----|-------|-------------|
| 10 | 1.0 | 2 |
| 11 | 1.4 | 1.9 |
| 12 | 1.7 | 2.1 |
| 13 | 2.0 | 1.9 |
| 14 | 2.2 | 2 |
| 15 | 2.4 | 2.6 |
| 16 | 2.6 | 2.5 |
| 17 | 2.8 | 3 |
| 18 | 3.0 | 2.8 |
| 19 | 3.2 | 2.6 |



Let's Try a More 'Sophisticated' Model



$$Y = A \cdot x + K$$

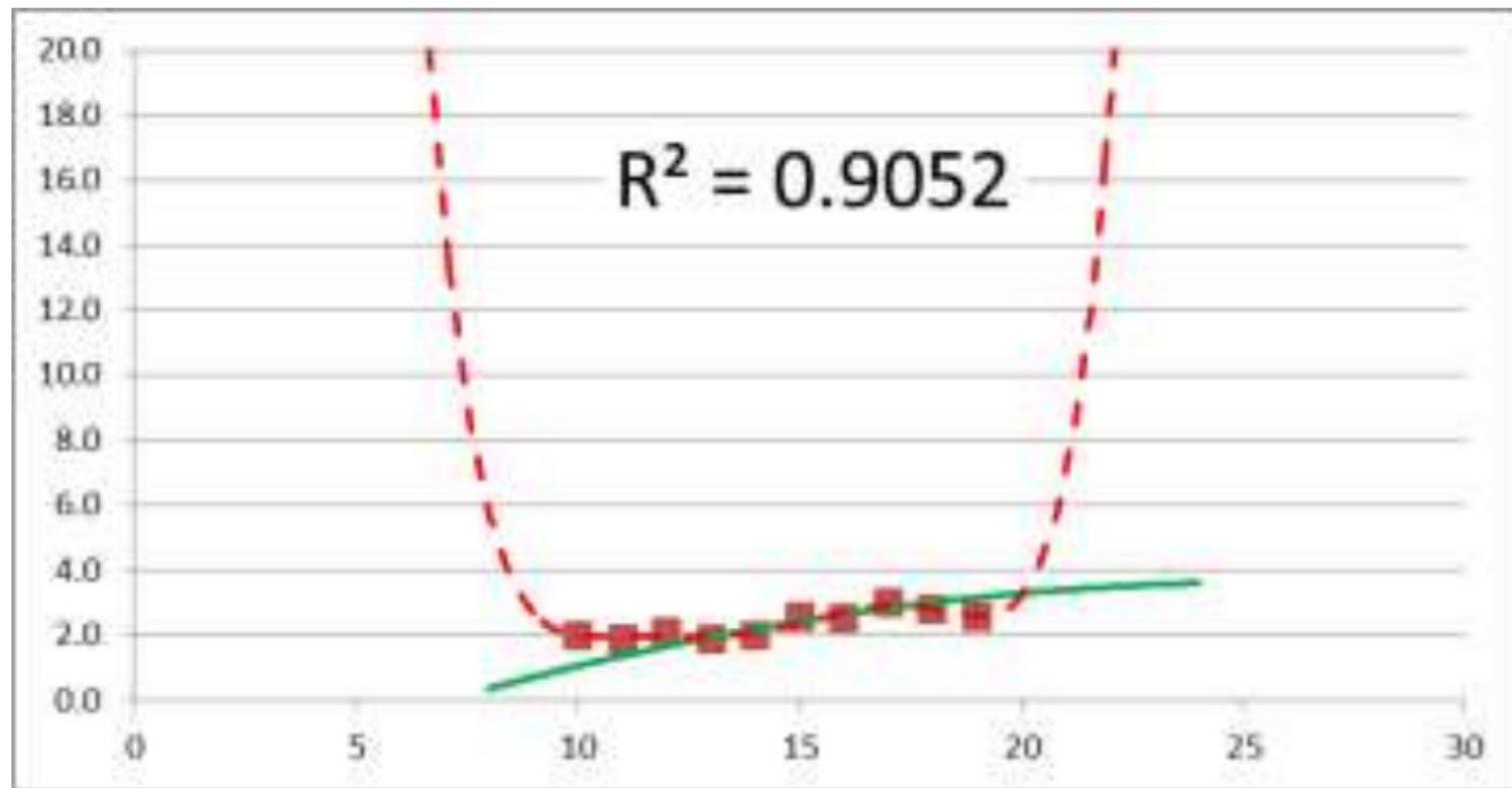


$$Y = A \cdot x + B \cdot x^2 + C \cdot x^3 + K$$



Let's Try a Really 'Sophisticated' Model!

- Our R² value is now close to 1 ...
 - The line very closely matches our data points
- But do we have a 'good' model?
- Not really, .. It does not seem to do a good job of predicting



$$Y = A \cdot x + B \cdot x^2 + C \cdot x^3 + D \cdot x^4 + E \cdot x^5 + G \cdot x^6 + K$$



Over-Fitting

- This phenomena is called ‘over-fitting’ – and is a reoccurring problem in data-driven models
- The model learns specific examples ...
 - .. but fails to generalise to new data ..
 - .. and is therefore not useful for prediction
- We will learn how to detect this by evaluating model quality in the next session.

