MONASH University

MONASH
INFORMATION
TECHNOLOGY



# Introduction to Machine Learning



Developed by Prajwol Sangat.
Updated by Chee-Ming Ting (3 April 2021)



GROUP OF EIGHT AUSTRALIA

# Last week

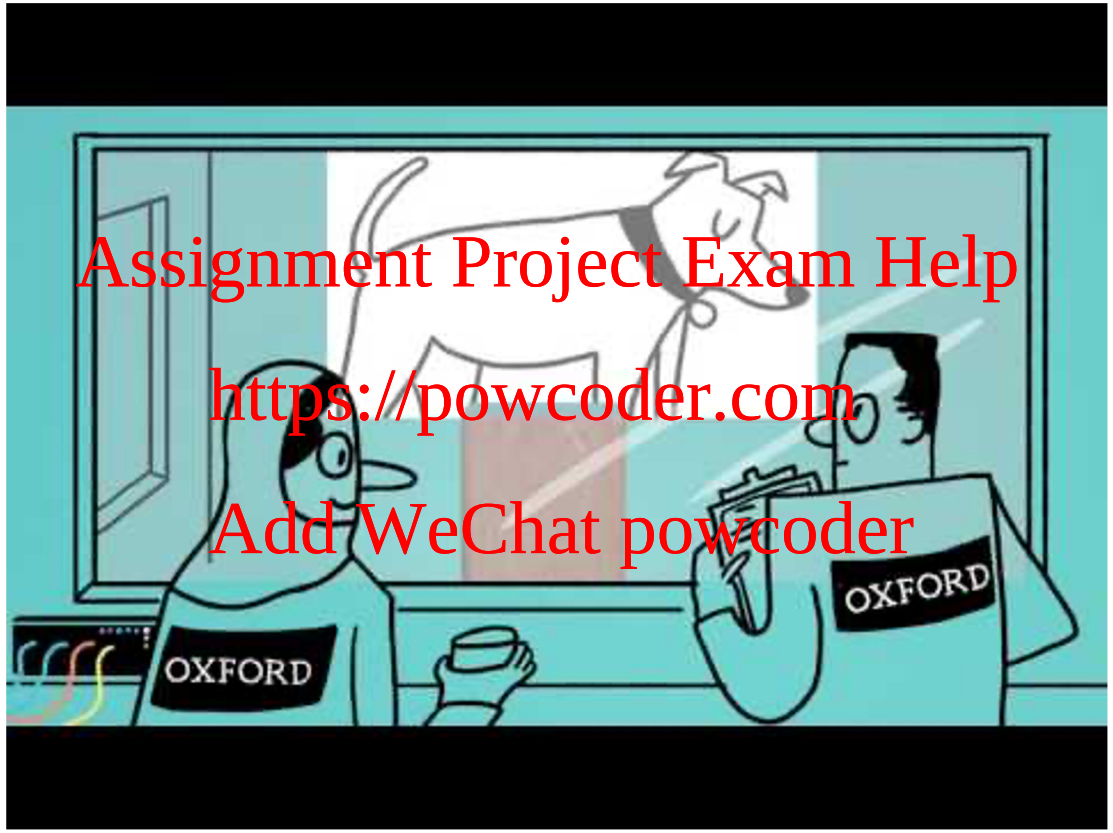
Parallel Aggregation
- Parallel Sort
- Parallel Group-By

# This week

What is Machine Learning?
Machine Learning Basics
Types of Machine Learning
Feature Engineering

According to McKinsey study, 35% of what consumers purchase on Amazon and 75% of what they watch on Netflix is driven by machine learning–based product recommendations.

# What is Machine Learning?



# What is Machine Learning?

*A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E*", (Tom Mitchell, 1997)




| Experience E | Task T | Performance P |
|---|---|---|
| databases of thousands of known faces | given a new photo, recognise the name of the face | how accurate the recognition is |

# Examples


Watson, Ken Jennings, and Brad Rutter in their *Jeopardy!* exhibition match

| Experience E | Task T | Performance P |
|---|---|---|
| databases of millions of question-answer pairs | given an question, find the best answer | how accurate the answer is |






| | | |
|---|---|---|
| Examples of spam emails and not-spam email | To assign a label "spam" or "not-spam" to an email | how accurate spam email can be detected |



Detecting Spam Emails


| | | |
|---|---|---|
| Data collected for credit-card transactions deemed as fraud and not-fraud | To assign a label 'fraud' or "not fraud" to a given credit-card transaction | how accurate a credit-card fraud transaction can be detected. |

Detect credit card fraud

# Elements of machine learning

## 1 Data

feature $x \in \mathbb{R}^d$

label $y \in Y$

Dataset $\mathcal{D} = \{x_i, y_i\}_{i=1}^{n}$

## 2 Model

Supervised: $f_\theta : X \rightarrow Y$

$X$ is data space

$Y$ is label space

$\theta$: model parameter

## 3 Assessment

How well is $f_\theta$ doing w.r.t data $\mathcal{D}$ ?



### Data processing

feature extraction, feature selection, feature transformation, feature reduction, feature scaling, feature normalization

### Predictive Model

$\hat{y} = f_\theta(x)$

### Model Learning (Training)

- Find an optimal model $f_\theta$ (by estimating model parameters $\theta$) using training data
- Based on loss function (e.g., minimize error between true and predicted labels)

### Model Testing

$\hat{y} = f_\theta(x_{test})$

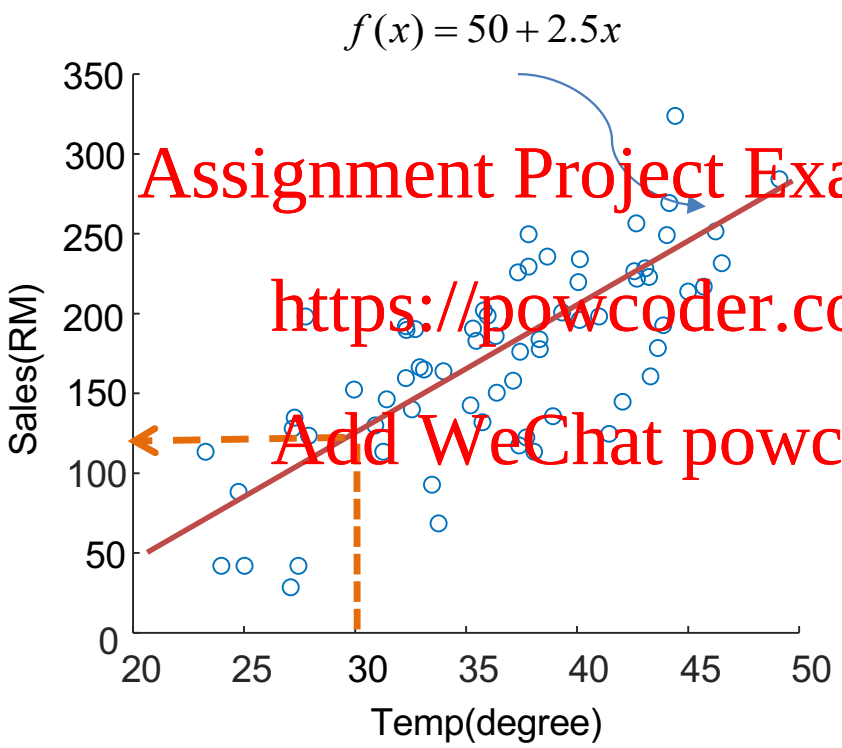
- Test the learned model in predicting unseen test data
- Performance metrics to assess model accuracy

# Illustration: Linear Regression model

**Problem:** Predict ice cream sales given temperature

Data

| Day | Temp | Sales |
|---|---|---|
| $i$ | $x_i$ | $y_i$ |
| 1 | 36 | 200 |
| 2 | 31 | 100 |
| 3 | 24 | 50 |
| ⋮ | ⋮ | |
| 100 | 38 | 250 |






Predictive Model:

- What is good model *f*(.) to maps *x* to *y*?

$$f(x) = \theta_o + \theta_1 x$$

Model Learning/Estimation:

- How to choose parameters $\theta_o, \theta_1$ ?
  - ❑ Define loss function
  - ❑ Estimate using learning algorithm

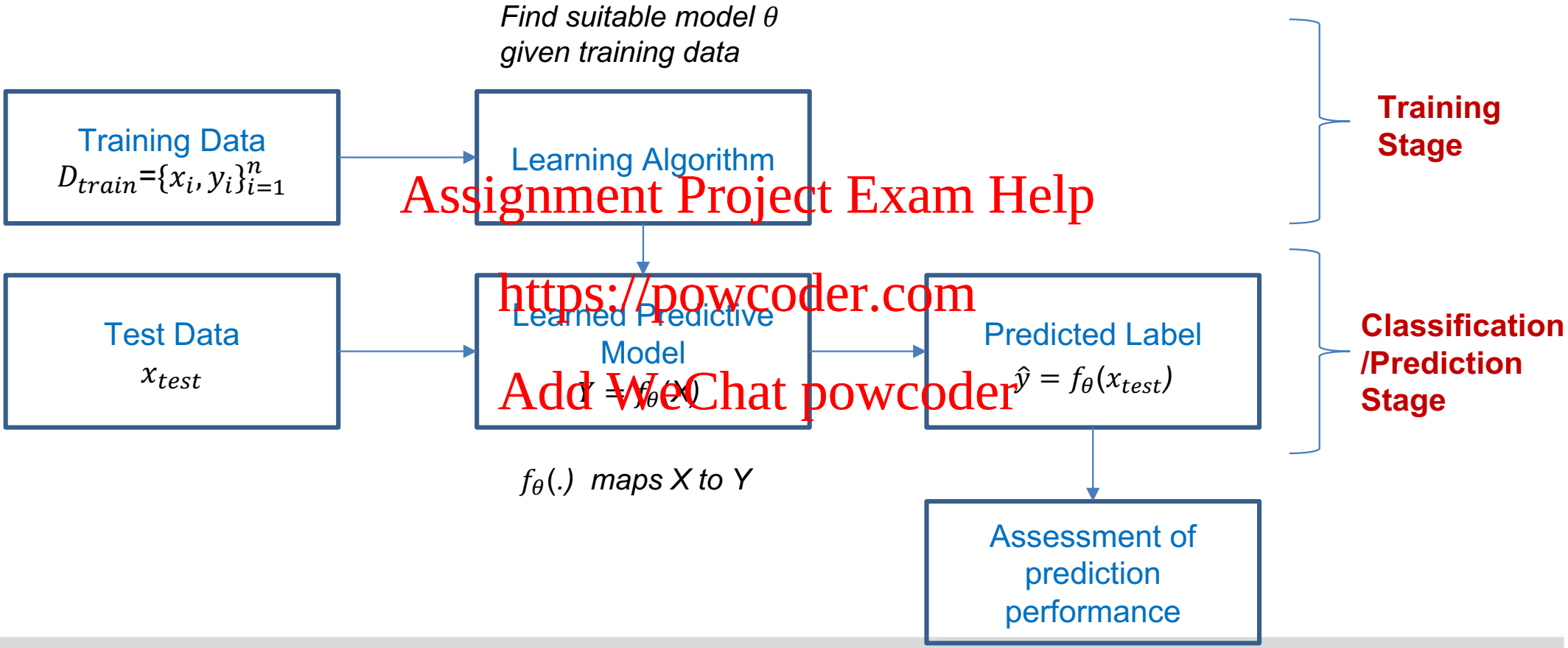
Estimated parameters: $\theta_o = 50,\ \theta_1 = 2.5$

Prediction:

- Given new input, predict *y* with learned model $\hat{y} = f(x_{new}) = \theta_o + \theta_1 x_{new}$

Predicted output $\hat{y} = 50 + 2.5(30) = 125$

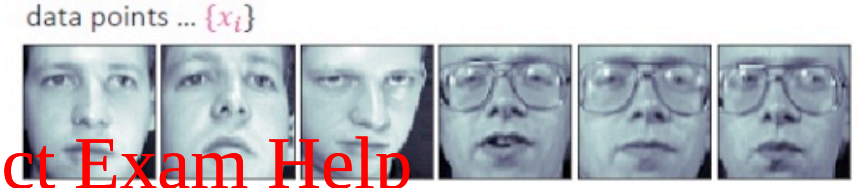
# Overview of machine learning



# Data

Features: $x_i$

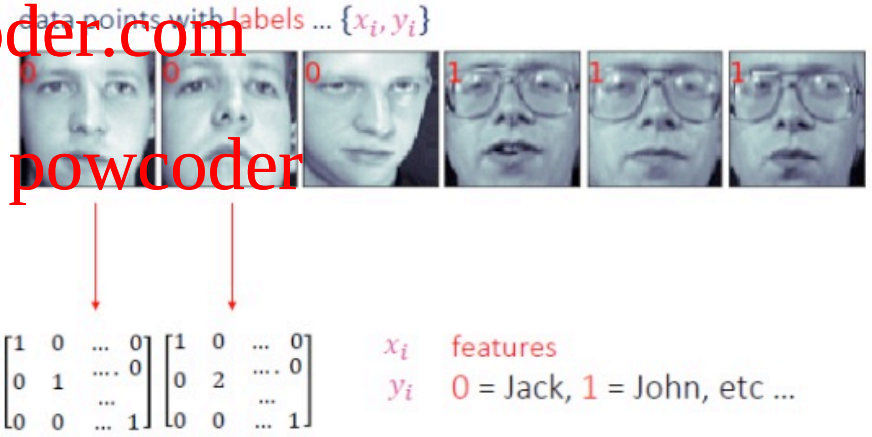
- a set of attributes, each is usually in form of a vector or matrix.
- E.g., represent each email (data point) into a bag-of-word vector (feature); or a face photo into a real-valued matrix.






Labels: $y_i$

- values, categories, classes assigned to data points.
- E.g., 0 = non-spam, 1 = spam.




Data points (aka instances, samples) $\{x_i\}$ or $\{x_i, y_i\}$

- these are items or instances of data used for training and evaluating ML models.
- E.g., labelled emails in spam detection; transaction data in credit card fraud detection; a photo in face recognition.

Dataset with $n$ samples: $\mathcal{D} = \{x_i, y_i\}_{i=1}^{n}$

# Machine Learning: Data Types

**Vector**

- A mathematical vector.
- *dense vectors*, where every entry is stored, and
- *sparse vectors*, where only the nonzero entries are stored to save space.



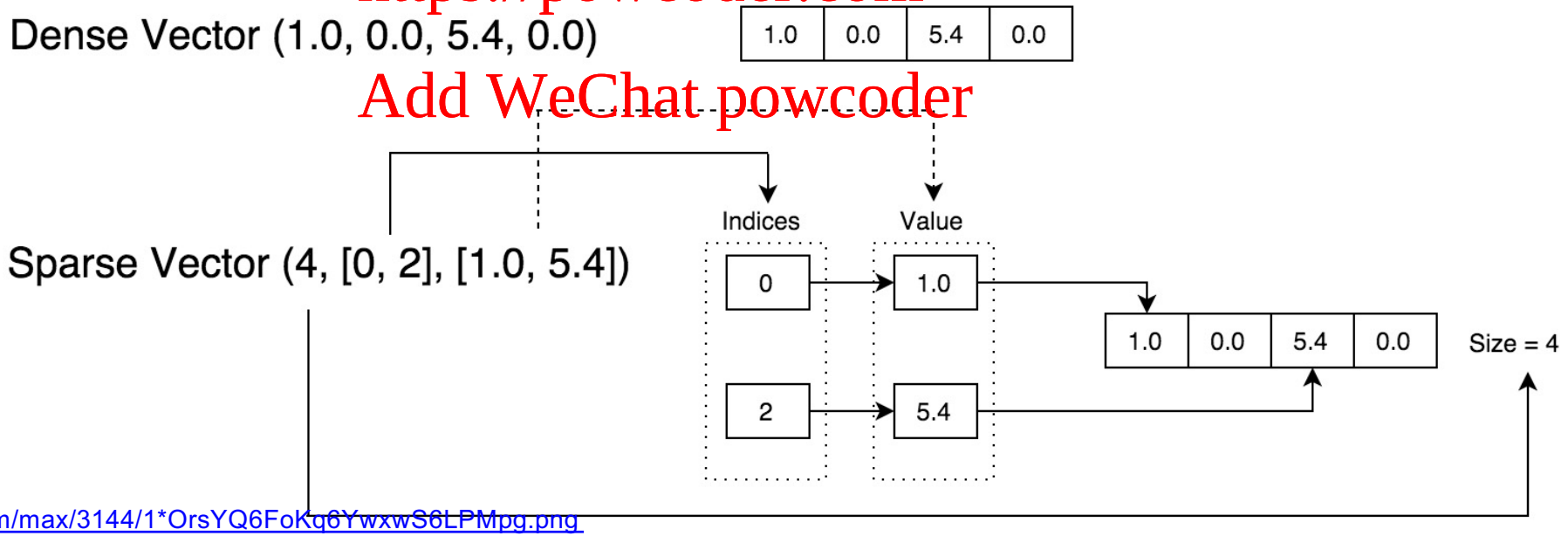
- **Labeled Point**
  - A labeled data point for supervised learning algorithms such as classification and regression.
  - Includes a feature vector and a label (which is a floating point value).

# Machine Learning: Data Types

## Vector

- A mathematical vector.
- *dense vectors*, where every entry is stored, and
- *sparse vectors*, where only the nonzero entries are stored to save space.






https://miro.medium.com/max/3144/1*OrsYQ6FoKq6YwxwS6LPMpg.png

# Features

All learning algorithms require defining a set of *features* for each item, which will be fed into the learning function.

- *For example, for an email, some features might include the server it comes from, or the number of mentions of the word free, or the color of the text.*





In many cases, defining the right features is the most challenging part of using machine learning.



- *For example, in a product recommendation task, simply adding another feature (e.g., realizing that which book you should recommend to a user might also depend on which movies she's watched) could give a large improvement in results.*

# Machine Learning Fundamentals

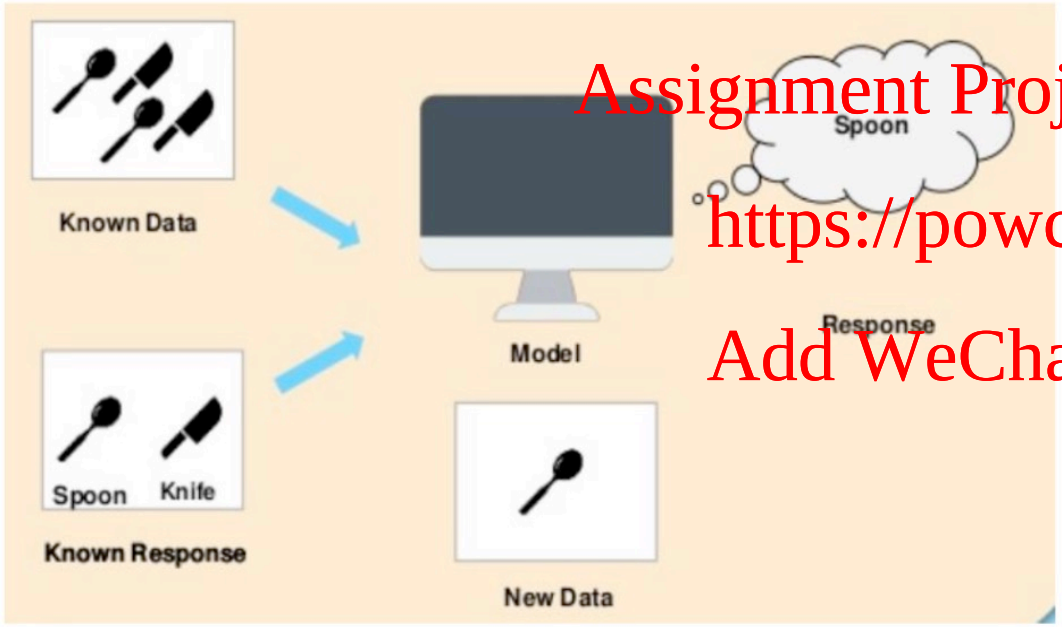
Supervised and Unsupervised Models

Bias and Variance

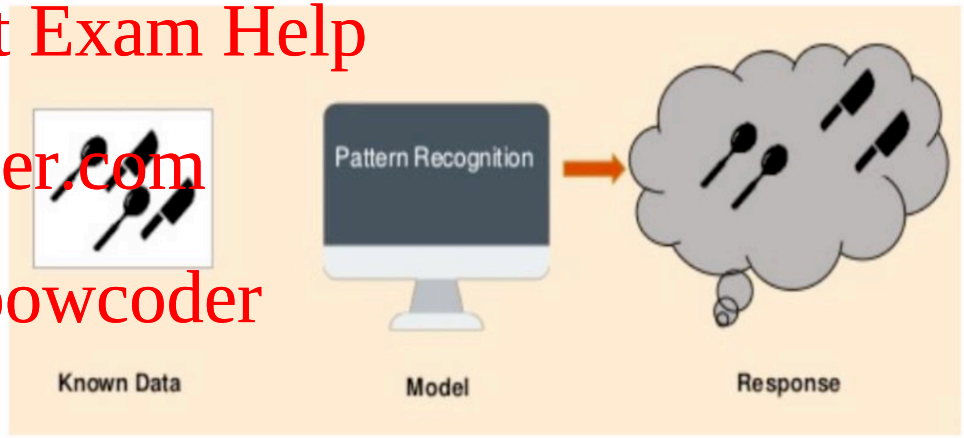
Underfitting and Overfitting

# Model: Types of Model Learning

Supervised




Unsupervised








https://www.simplilearn.com/tutorials/machine-learning-tutorial/supervised-and-unsupervised-learning

# Types of Model Learning: Supervised

- Goal: Learn a function from labelled training data to predict the output label(s) given a new unlabeled input.



- Training data consists of input features and output information (labels)



- Two types of supervised learning



  - ❑ Classification
  - ❑ Regression

Data: $(x_1, y_1), \ldots, (x_n, y_n)$

Function: $f: X \rightarrow Y$

$x$ = feature
$y$ = a discrete label (classification),
$y$ = a continuous value (regression)

# Supervised Machine Learning: Classification

Classification problem: To separate inputs into a discrete set of classes or labels.

- Binary classification
- Multinomial (Multi-class) classification



*Binary classification example: dog or not dog*

# Supervised Machine Learning: Classification



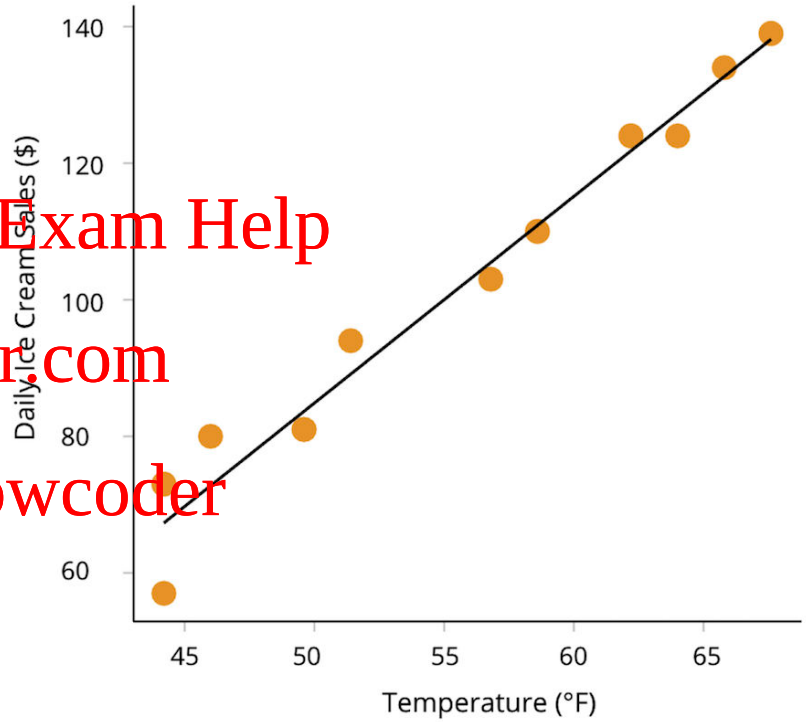


*Multinomial classification example: Australian shepherd, golden retriever, or poodle*

# Supervised Machine Learning: Regression

- A regression problem is when the output variable is a real value, such as "dollars" or "weight".






*Regression example: predicting ice cream sales based on temperature*

# Supervised Machine Learning in Apache Spark

| Algorithm | Typical usage |
|---|---|
| Linear regression | Regression |
| Logistic regression | Classification (we know, it has regression in the name!) |
| Decision trees | Both |
| Gradient boosted trees | Both |
| Random forests | Both |
| Naive Bayes | Classification |
| Support vector machines (SVMs) | Classification |

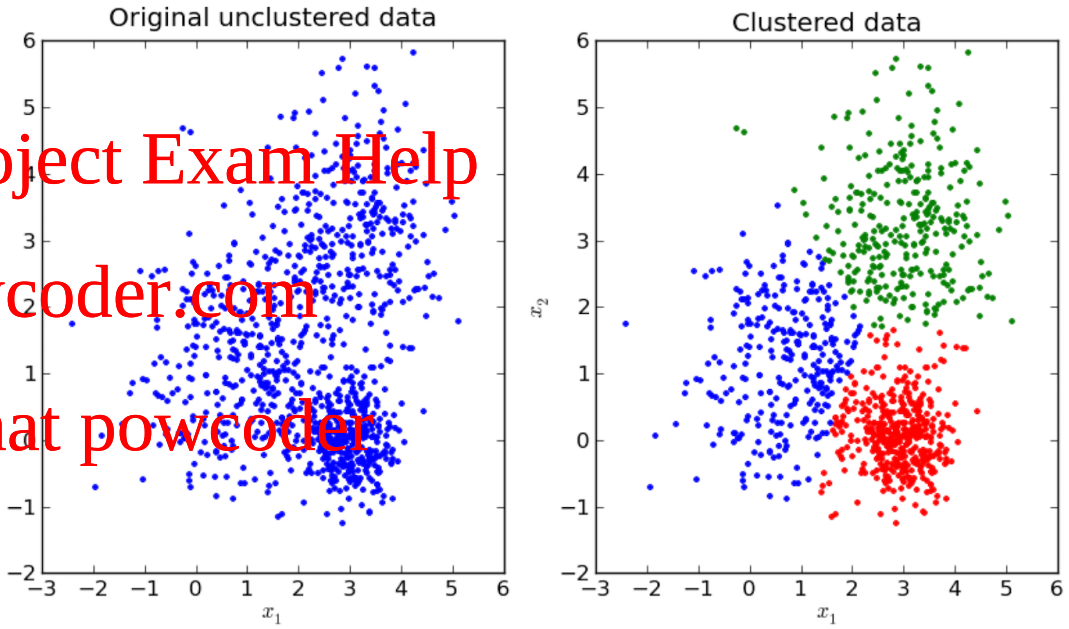
# Types of Model Learning: Unsupervised

- Goal: Explore the underlying structure of the data to extract meaningful information. without guidance of known output info.
- Deals with unlabelled data (no output labels)
- Two types of unsupervised learning:
  - Clustering
  - Association

# Unsupervised Machine Learning: Clustering

- Clustering problem: Divide data into clusters which are similar between them and are dissimilar to the data belonging to another cluster.
- Where you want to discover the inherent groupings in the data, eg. grouping customers by purchasing behaviour

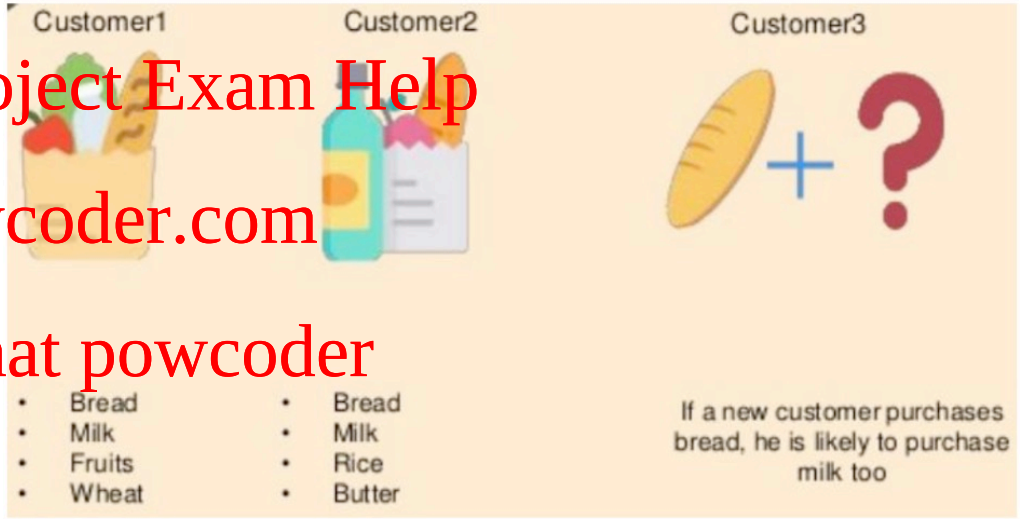





*Clustering example*

# Unsupervised Machine Learning: Association

- Association rule learning problem: Discover the probability of the co-occurrence (association) between items in a large dataset
- Where you want to discover rules that describe large portions of your data, e.g., people who buy X also tend to buy Y.



# Unsupervised Machine Learning in Apache Spark

- *k*-means,
- Latent Dirichlet Allocation (LDA), and
- Gaussian mixture models.

# Machine Learning: Assessment

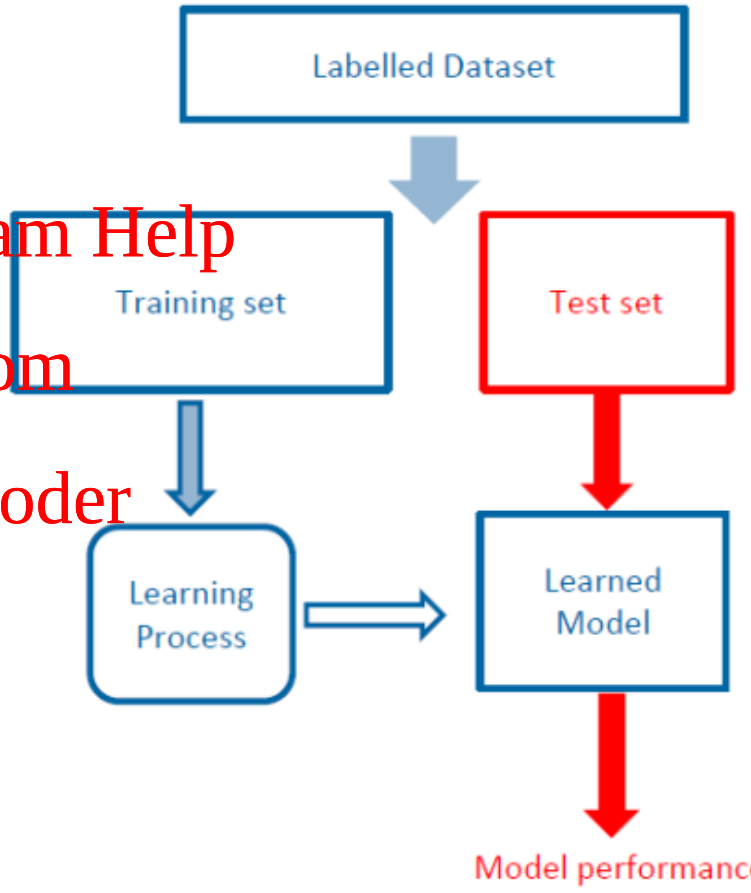
How to prepare the data?

- Train-Test split
- K-fold cross-validation

How to measure performance?

- TP, FP, TN, FP, confusion matrix
- Accuracy, Recall, Precision, F1-score





# Machine Learning: Performance Metrics

**Example: Email Spam Detection**
In test set: 10 spam, 20 non-spam
Positive = spam

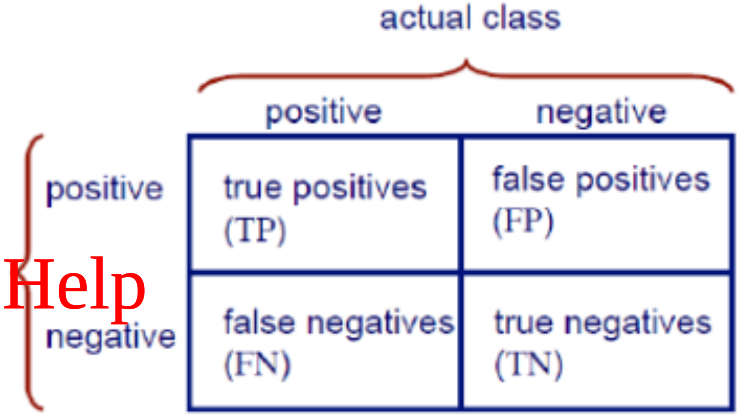


True labels

| Predicted labels | SPAM (1) | NON-SPAM (0) |
|---|---|---|
| SPAM (1) | 7 | 5 |
| NON-SPAM (0) | 3 | 15 |








$$\text{Accuracy} = \frac{tp + tn}{tp + tn + fp + fn}$$

$$\text{Precision} = \frac{tp}{tp + fp}$$

$$\text{Recall} = \frac{tp}{tp + fn}$$

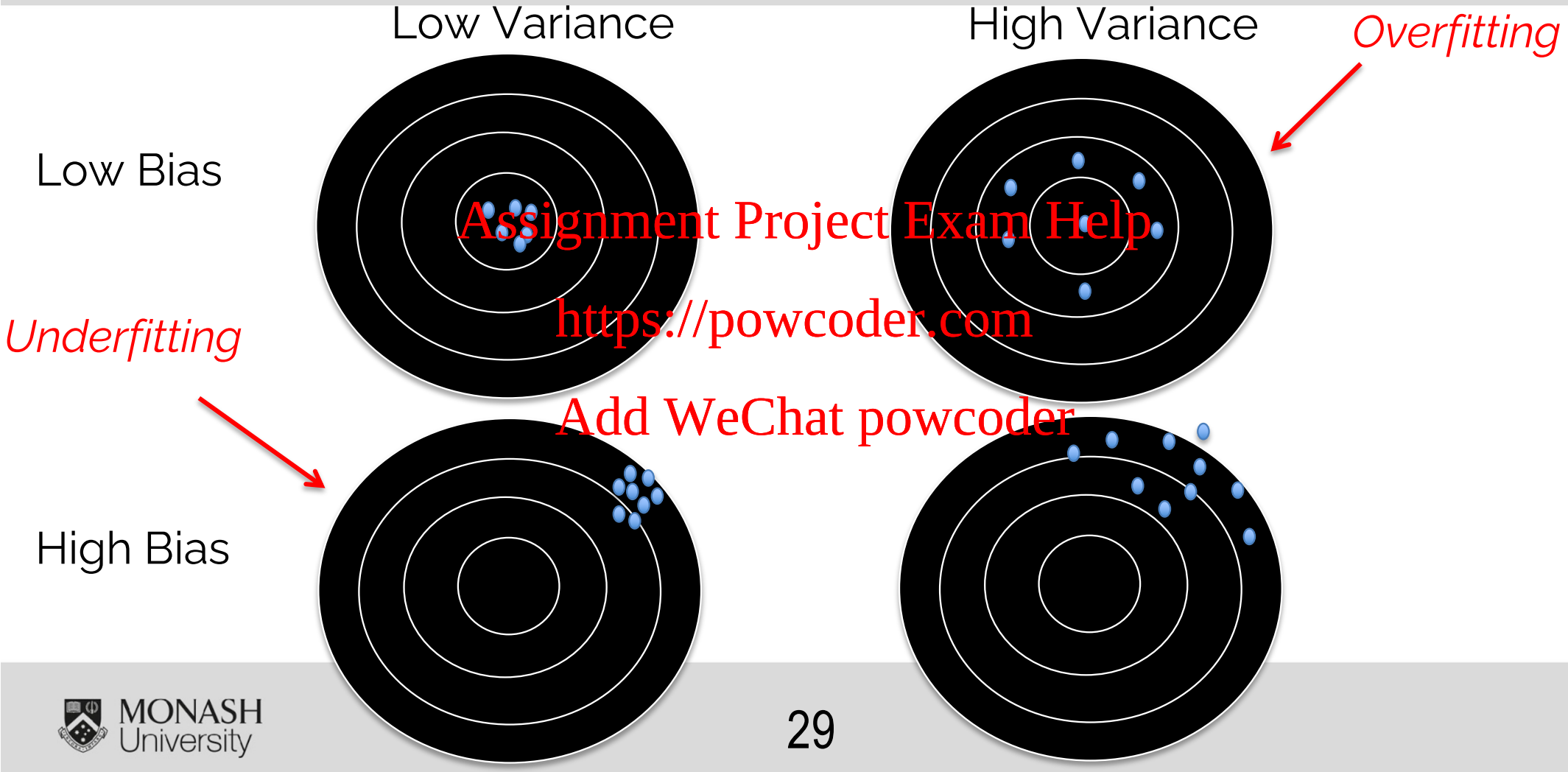
$$F = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

# Machine Learning: Bias and Variance

**Bias** is the gap between the averaged predicted value by the model and the actual value of the data.

**Variance** measures the distance of the predicted values in relation to each other.

# Machine Learning: Bias and Variance



# Machine Learning: Overfitting and Underfitting

**Overfitting** (high variance, low bias) is a model that performs well on the training data but generalizes poorly to any new data.

**Underfitting** (low variance, high bias) is an overly simple model that does not perform well even on the training data.

# Machine Learning: Overfitting and Underfitting



- **Preventing Overfitting**
  - Train with more data

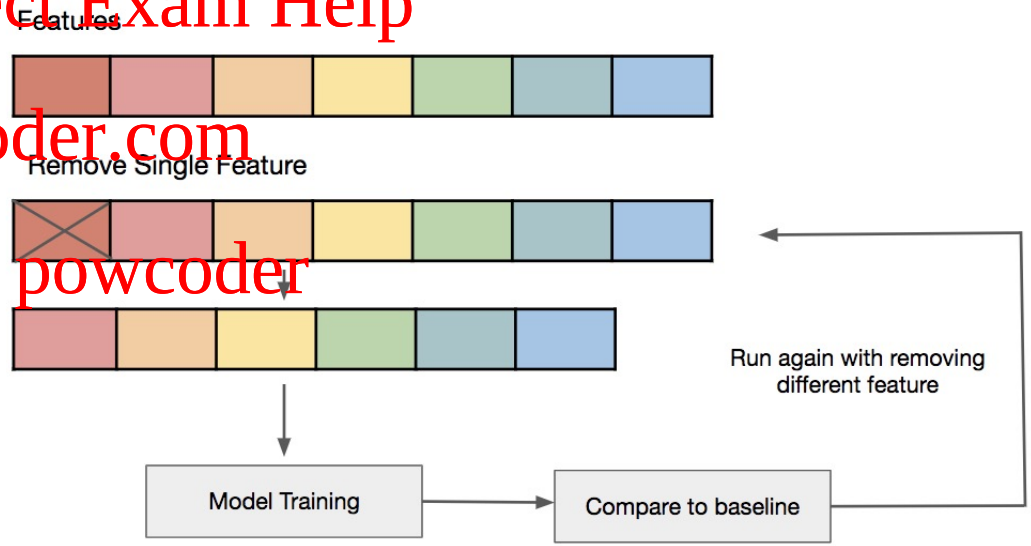




# Machine Learning: Overfitting and Underfitting



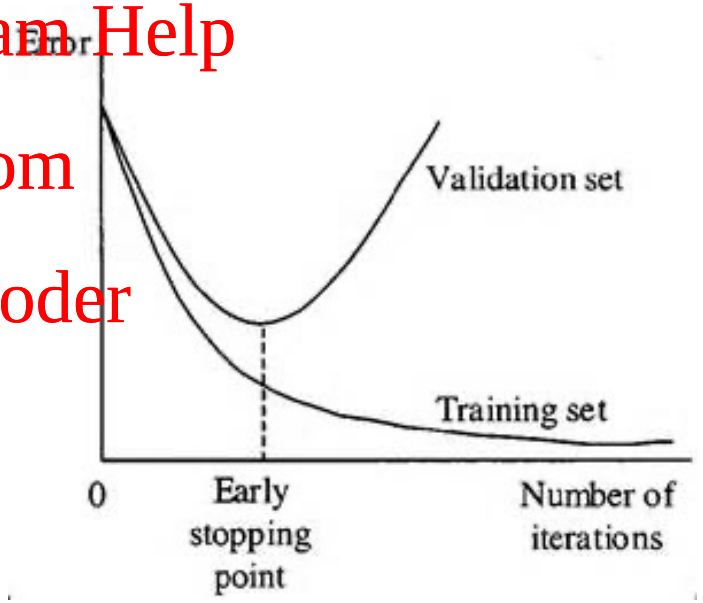
- **Preventing Overfitting**
  - Train with more data
  - Remove features



# Machine Learning: Overfitting and Underfitting



- **Preventing Overfitting**
  - Train with more data
  - Remove features
  - Early stopping



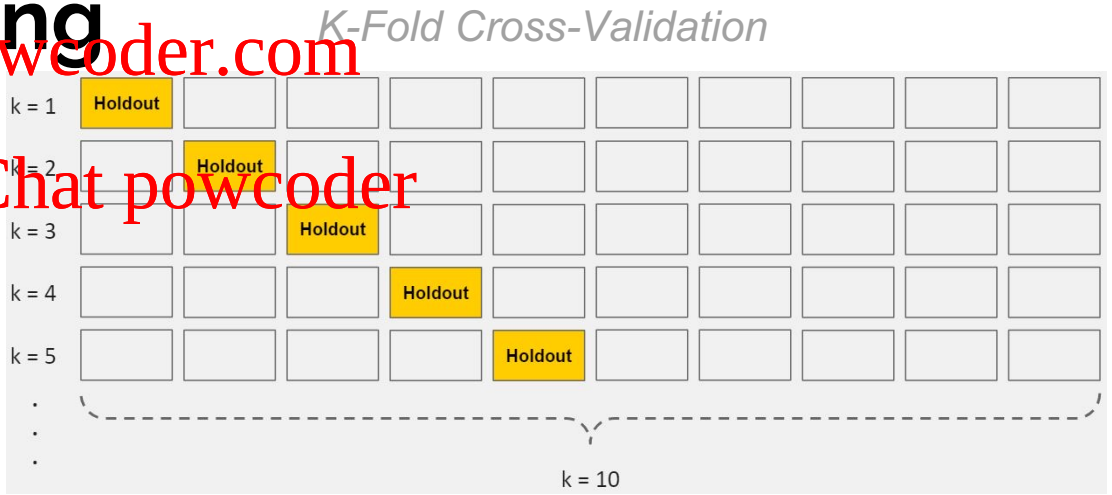
# Machine Learning: Overfitting and Underfitting



- **Preventing Overfitting**
  - Train with more data
  - Remove features
  - Early stopping
  - Cross validation







# To be continued..

**Next topic -> Featurization**

MONASH University

MONASH
INFORMATION
TECHNOLOGY



# Machine Learning- Featurization



GROUP OF EIGHT AUSTRALIA

# Machine Learning: Pipeline

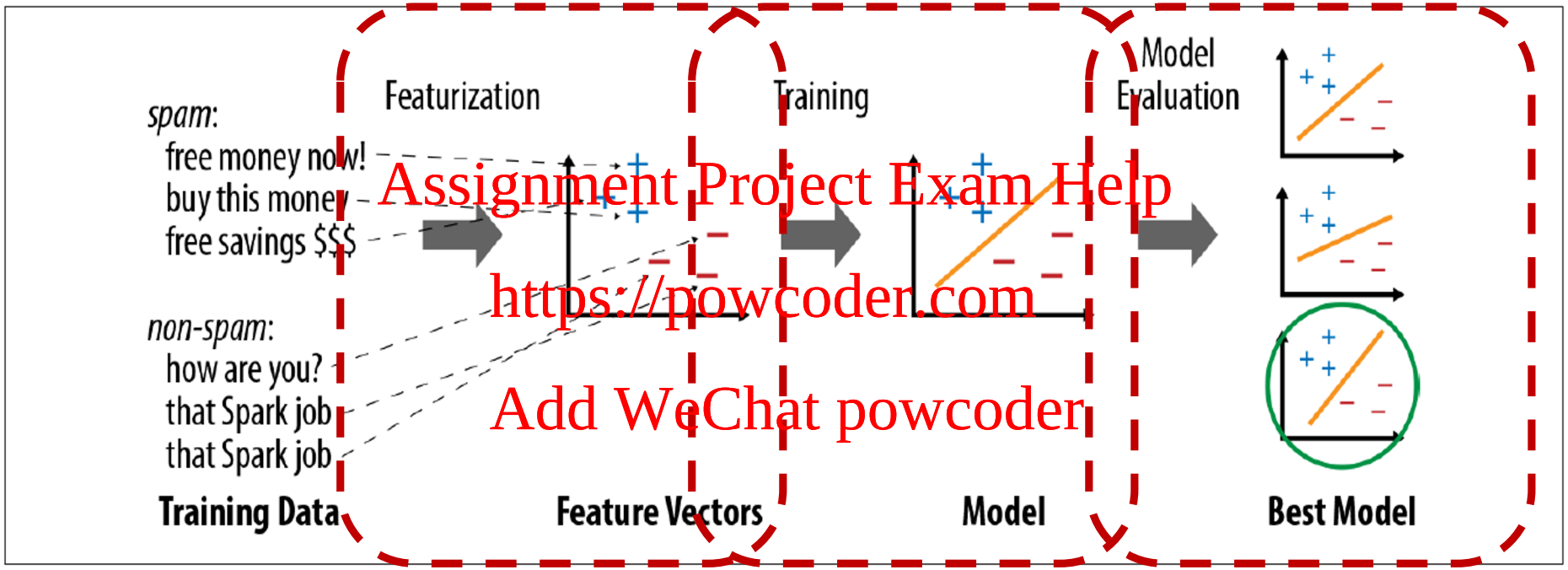





*Figure 11-1. Typical steps in a machine learning pipeline*

# Machine Learning: Pipeline

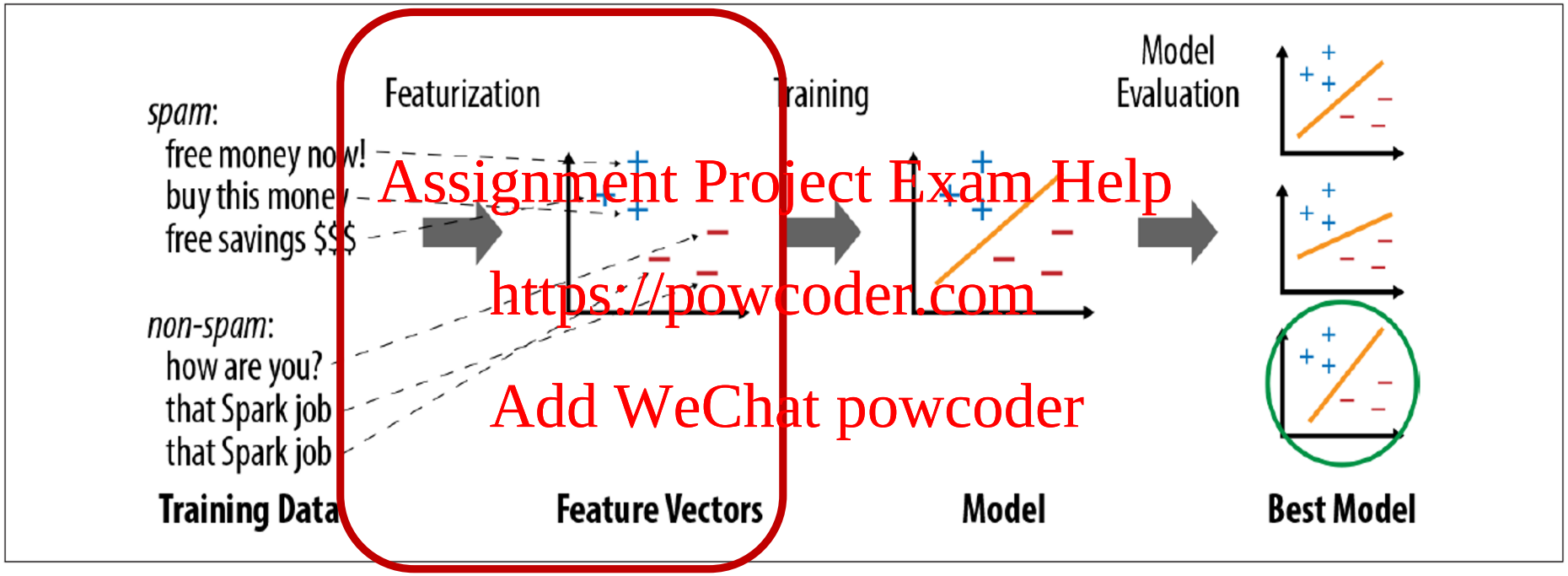





*Figure 11-1. Typical steps in a machine learning pipeline*

# Featurization: Extraction, transformation and selection

**Extraction**
- Extracting features from “raw” data

**Transformation**
- Scaling, converting, or modifying features

**Selection**
- Selecting a subset from a larger set of features

# Featurization: Feature Extraction and Transformation

## Features

- Any machine learning algorithm requires some training data. In training data we have values for all features for all historical records. Consider this simple dataset



| Height | Weight | Age | Class |
|---|---|---|---|
| 165 | 70 | 22 | Male |
| 160 | 58 | 22 | Female |



- We can prepare training data by following two techniques
  - *Feature Extraction*
  - *Feature Selection*

# Featurization: Feature Extraction and Transformation

**Feature extractors**

- CountVectorizer
- TF-IDF
- Word2Vec
- FeatureHasher (In tutorial)

# Featurization: Feature Extractors

## Count Vectorizer

- Convert a collection of text documents to vectors of token counts.
- During the fitting process, *Count Vectorizer* will select the top *vocabSize* words ordered by term frequency across the corpus.



```
 id | texts
----|----------
 0  | Array("a", "b", "c")
 1  | Array("a", "b", "b", "c", "a")
```

# Featurization: Feature Extractors

**Term Frequency–Inverse Document Frequency, or TF-IDF**,

- A simple way to generate feature vectors from text documents (e.g., web pages).
- It computes two statistics for each term in each document:
  - *The term frequency (TF)*, which is the number of times the term occurs in that document, and
  - *The inverse document frequency (IDF)*, which measures how (in)frequently a term occurs across the whole document corpus.

# Featurization: Feature Extractors

**Term Frequency–Inverse Document Frequency, or TF-IDF**,

- Denote a term by ***t,*** a document by ***d,*** and the corpus by ***D***.



- Term frequency ***TF(t,d)*** is the number of times that term ***t*** appears in document ***d,*** while document frequency ***DF(t,D)*** is the number of documents that contains term ***t.***





- Inverse document frequency is a numerical measure of how much information a term provides:

$$IDF(t, D) = \log \frac{|D| + 1}{DF(t, D) + 1},$$

where **|D|** is the total number of documents in the corpus.

# Featurization: Feature Extractors

**Term Frequency–Inverse Document Frequency, or TF-IDF**,

- The product of these values, TF × IDF, shows how relevant a term is to a specific document (i.e., if it is common in that document but rare in the whole corpus).
- The TF-IDF measure is simply the product of TF and IDF:



$$TFIDF(t, d, D) = TF(t, d) \cdot IDF(t, D).$$

# Featurization: Feature Extractors

**Term Frequency–Inverse Document Frequency, or TF-IDF**,

Suppose that we have term count tables of a corpus consisting of only two documents, as listed on the right.

**Calculate TF-IDF for the term "this".**

**Document 1**

| Term | Term Count |
|---|---|
| this | 1 |
| is | 1 |
| a | 2 |
| sample | 1 |

**Document 2**

| Term | Term Count |
|---|---|
| this | 1 |
| is | 1 |
| another | 2 |
| example | 3 |

# Featurization: Feature Extractors

**TF-IDF (Solution),**

**Calculating TF for "this":**

TF ("this", d1) = 1/5 = 0.2
TF ("this", d2) = 1/7 = 0.14
(Approx.)



**Document 1**

| Term | Term Count |
|---|---|
| this | 1 |
| is | 1 |
| a | 2 |
| sample | 1 |

**Document 2**

| Term | Term Count |
|---|---|
| this | 1 |
| is | 1 |
| another | 2 |
| example | 3 |

# Featurization: Feature Extractors

**TF-IDF (Solution),**

**Calculating IDF for "this":**

|D| = 2

DF(t, D) = 2

IDF ("this", D) = log ( 3/ 3) = 0



$$IDF(t, D) = \log \frac{|D| + 1}{DF(t, D) + 1},$$

**Document 1**

| Term | Term Count |
|---|---|
| this | 1 |
| is | 1 |
| a | 2 |
| sample | 1 |

**Document 2**

| Term | Term Count |
|---|---|
| this | 1 |
| is | 1 |
| another | 2 |
| example | 3 |

# Featurization: Feature Extractors

**TF-IDF (Solution),**

**Calculating TF-IDF for "this":**

TF-IDF ("this", d1, D) = 0.2 * 0 = 0

TF-IDF ("this", d2, D) = 0.14 * 0 = 0



**Document 1**

| Term | Term Count |
|---|---|
| this | 1 |
| is | 1 |
| a | 2 |
| sample | 1 |

**Document 2**

| Term | Term Count |
|---|---|
| this | 1 |
| is | 1 |
| another | 2 |
| example | 3 |

$$TFIDF(t,d,D) = TF(t,d) \cdot IDF(t,D).$$

# Featurization: Feature Extractors

**Exercise: Calculate TF-IDF for the term “example”.**



Document 1

| Term | Term Count |
|---|---|
| this | 1 |
| is | 1 |
| a | 2 |
| sample | 1 |

Document 2

| Term | Term Count |
|---|---|
| this | 1 |
| is | 1 |
| another | 2 |
| example | 3 |

# Featurization: Feature Extractors

**Word2Vec**

- maps each word to a unique fixed-size vector.
- transforms each document into a vector using the average of all words in the document.
- this vector can then be used as features for prediction, **document similarity calculations** etc.



**Home Work: Do some research and write a program to find the document similarity using Word2Vec.**

# Featurization: Extraction, transformation and selection

**Extraction**
- Extracting features from "raw" data

**Transformation**
- Scaling, converting, or modifying features

**Selection**
- Selecting a subset from a larger set of features

# Featurization: Feature Extraction and Transformation

**Feature Transformers**

- Tokenization
- Stop Words Remover
- String Indexing
- One Hot Encoding
- Vector Assembler (Implement in tutorial)

# Featurization: Feature Transformers

**Tokenization**

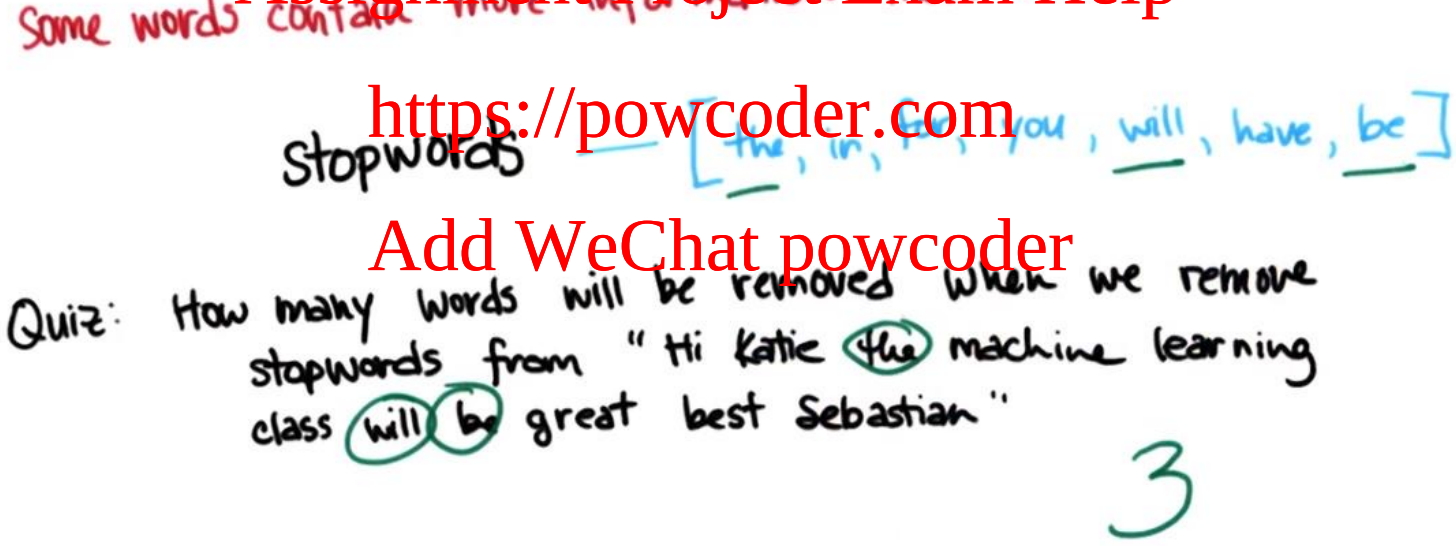
- It is the process of taking text (such as a sentence) and breaking it into individual terms (usually words).



# Featurization: Feature Transformers

**Stop Words** are words which should be excluded from the input, typically because the words appear frequently and don't carry as much meaning.





# Featurization: Feature Transformers

## Stop Words Remover

Takes as input a sequence of strings (e.g. the output of a Tokenizer)

Drops all the stop words from the input sequences.



```
 id | raw                          | filtered
----|------------------------------|--------------------
 0  | [I, saw, the, red, balloon]  |  [saw, red, balloon]
 1  | [Mary, had, a, little, lamb]|[Mary, little, lamb]
```

# Featurization: Feature Transformers

## String Indexing

Encoding a string column of labels to a column of label indices.



| id | category | categoryIndex |
|---|---|---|
| 0 | a | 0.0 |
| 1 | b | 2.0 |
| 2 | c | 1.0 |
| 3 | a | 0.0 |
| 4 | a | 0.0 |
| 5 | c | 1.0 |

# Featurization: Feature Transformers

## One Hot Encoding

Maps a categorical feature represented as a label index to a binary vector..

A single one-value indicates the presence of a specific feature value from among the set of all feature values.

For string type input data, it is common to encode categorical features using String Indexing first.

# Featurization: Feature Transformers

## Why One Hot Encoding?

For categorical variables when there is no ordinal relationship, the string indexing is not enough.

Using this encoding and allowing the model to assume a natural ordering between categories may result in poor performance or unexpected results.

A one-hot encoding can be applied to the integer representation. This is where the integer encoded variable is removed and a new binary variable is added for each unique integer value.

# Featurization: Feature Transformers

## Why One Hot Encoding?

Example: Let's say we have 3 data instances with attributes of **Preferred Programming Language** and **OS of Choice.**



| Preferred Programming Language | OS of Choice |
|---|---|
| Javascript | OSX |
| Python | Linux |
| Scala | OSX |

# Featurization: Feature Transformers

## Why One Hot Encoding?

### String Indexing



| Preferred Programming Language | OS of Choice |
|---|---|
| Javascript | OSX |
| Python | Linux |
| Scala | OSX |

| Preferred Programming Language | OS of Choice |
|---|---|
| 0 | 0 |
| 1 | 1 |
| 2 | 0 |

# Featurization: Feature Transformers

- **Why cant we STOP here?**

**The Problem Of Ordinality**

Machine learning algorithms treat the ordinality of numbers in an attribute with some significance: *a higher number "must be better" than a lower number*.

**String Indexing**



| Preferred Programming Language | OS of Choice |
|---|---|
| 0 | 0 |
| 1 | 1 |
| 2 | 0 |

# Featurization: Feature Transformers

- String Indexing

| Preferred Programming Language | OS of Choice |
|---|---|
| 0 | 0 |
| 1 | 1 |
| 2 | 0 |

- One Hot Encoding

| Java | Python | Scala | OSX | Linux |
|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 |



**Homework**: Have a look at **Principal Component Analysis (PCA)**

# Featurization: Feature Transformers

## Why One Hot Encoding?

For categorical variables when there is no ordinal relationship, the string indexing is not enough.

Using this encoding and allowing the model to assume a natural ordering between categories may result in poor performance or unexpected results.

A one-hot encoding can be applied to the integer representation. This is where the integer encoded variable is removed and a new binary variable is added for each unique integer value.

# Featurization: Extraction, transformation and selection

**Extraction**
- Extracting features from "raw" data

**Transformation**
- Scaling, converting, or modifying features



**Selection**
- Selecting a subset from a larger set of features

# Featurization: Feature Selectors

**Feature selection**

- This process tries to get most important features that are contributing to decide the label.

**Vector Slicer**

- It takes a feature vector and outputs a new feature vector with a sub-array of the original features.
- It is useful for extracting features from a vector column.



```
userFeatures     | features
-----------------|--------------
[0.0, 10.0, 0.5] | [10.0, 0.5]
```

# Lecture Demo

## Spam Classification Example

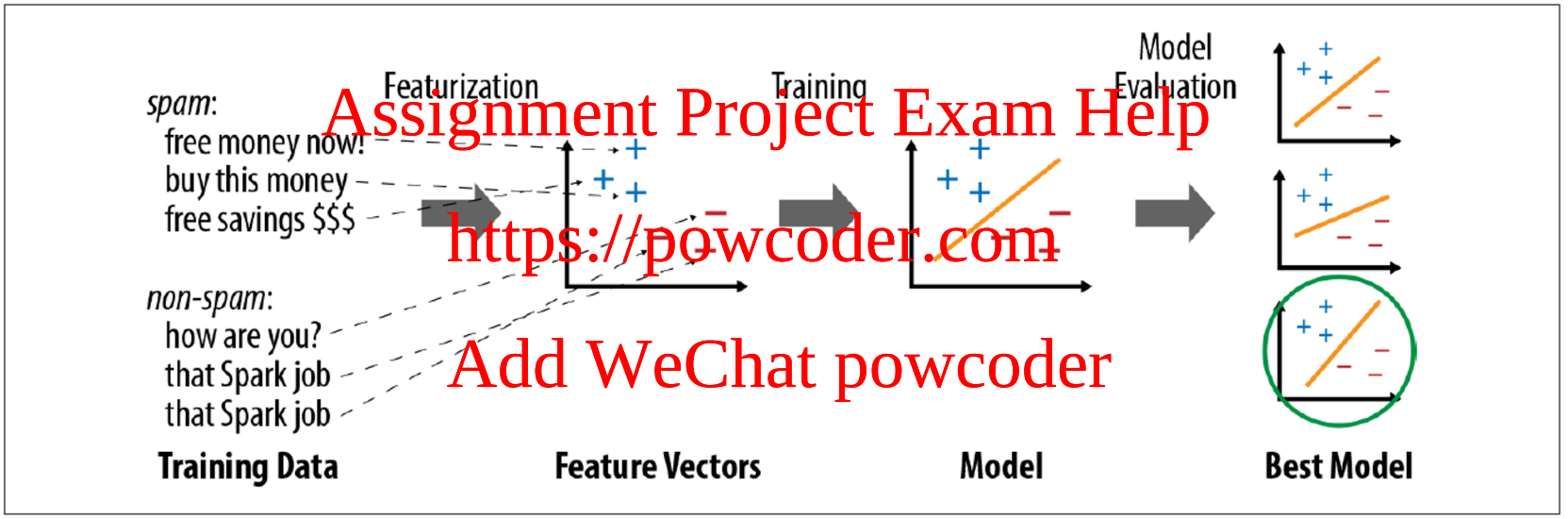





*Figure 11-1. Typical steps in a machine learning pipeline*

# Thank You

**See you next week**