# Data Science for Business

## *Using pandas for Better (and Worse) Data Science*

Asst. Prof. Teerapong Leelanupab (Ph.D.)
Faculty of Information Technology
King Mongkut's Institute of Technology Ladkrabang (KMITL)

Week 5.1

# Using pandas for Better (and Worse) Data Science

Good Talk By Kevin Markham at PyCon Cleveland 2018

## This tutorial is for intermediate pandas users

- Github: https://github.com/justmarkham/pycon-2018-tutorial
- Youtube playlist: http://tiny.cc/t5i8az

## Datasets:

- police.csv is the Rhode Island dataset from the Stanford Open Policing Project, made available under the Open Data Commons Attribution License.
- ted.csv is the TED Talks dataset from Kaggle Datasets, made available under the CC BY-NC-SA 4.0 license.

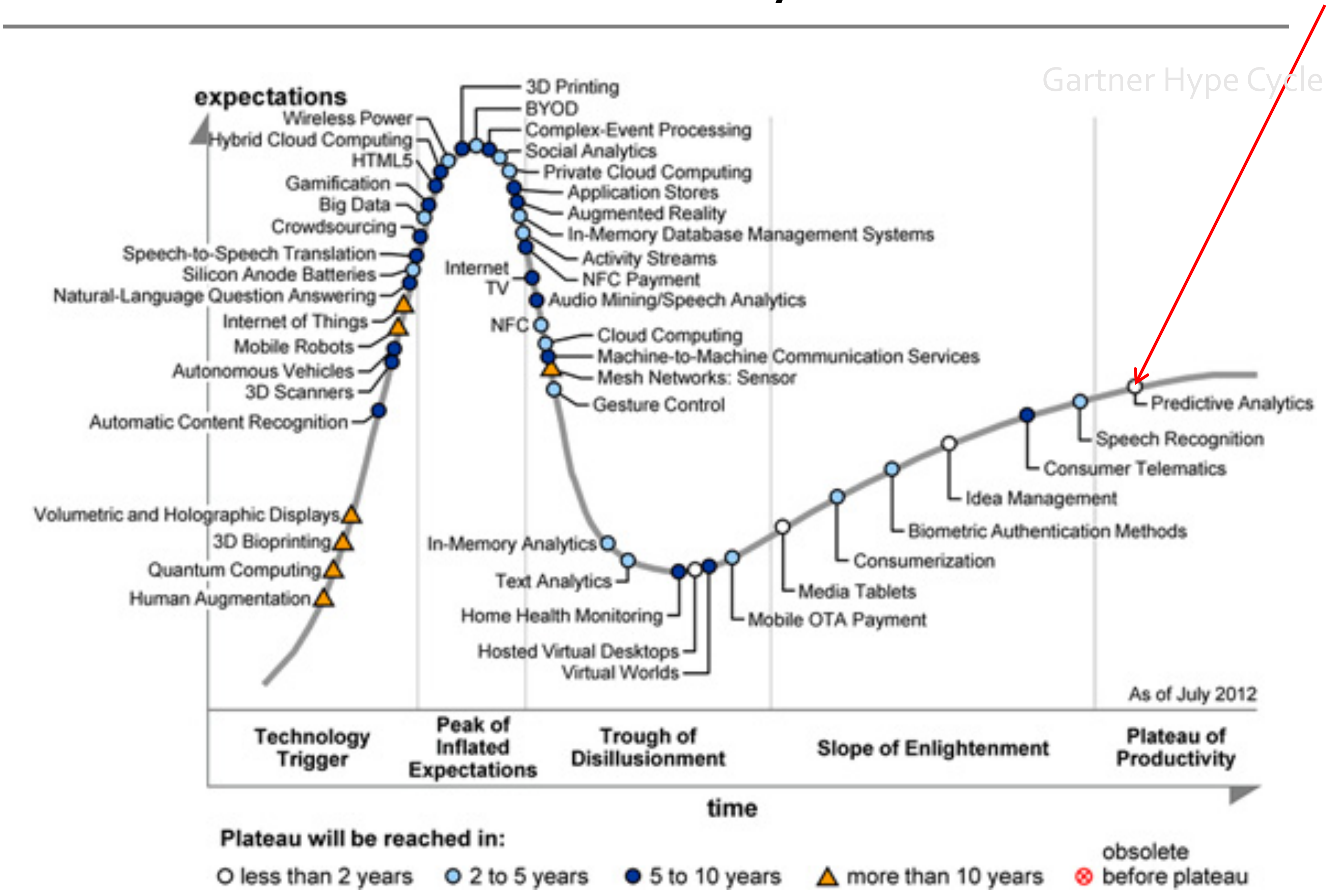
# Data Science for Business

## *Predictive Modeling*

Asst. Prof. Teerapong Leelanupab (Ph.D.)
Faculty of Information Technology
King Mongkut's Institute of Technology Ladkrabang (KMITL)

Week 5.2

# Toward Predictive Analytics


Gartner Hype Cycle

# VodafoneTelCo: Predicting Customer Churn

- You just landed a great analytical job with VodafoneTelCo, one of the largest telecommunication firms in the US
- They are having a major problem with customer retention in their wireless business
- In the mid-Atlantic region, 20% of cell phone customers leave when their contracts expire. Communications companies are now engaged in battles to attract each other’s customers while retaining their own
- Marketing has already designed a special retention offer
- Your task is to devise a precise, step-by-step plan for how the data science team should use VodafoneTelCo vast data resources to solve the problem

# VodafoneTelCo : Predicting Customer Churn

- What data you might use?
- How would they be used?
- How should VodafoneTelCo choose a set of customers to receive their offer in order to best reduce churn for a particular incentive budget?

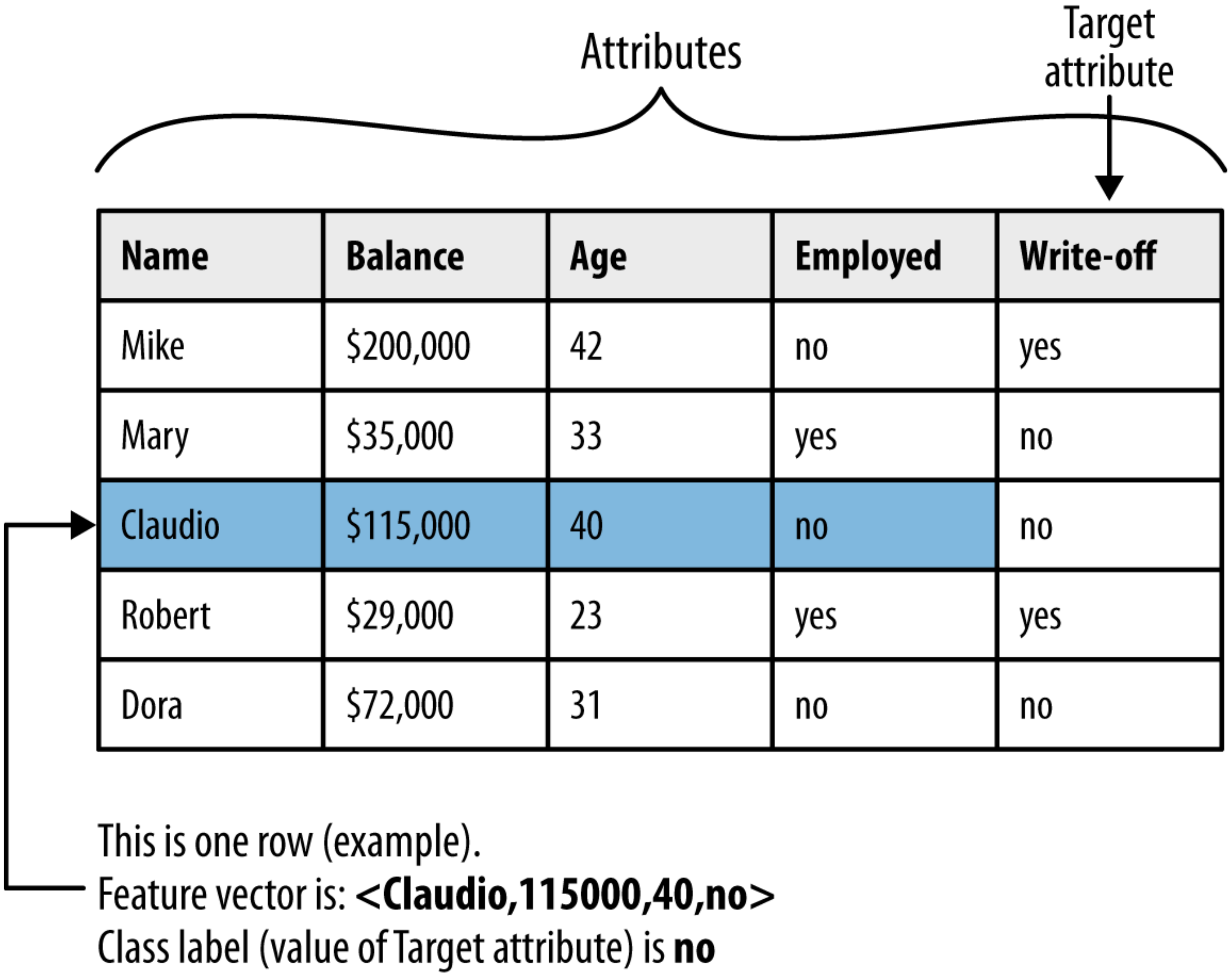
# Terminology

- Model:
  - A simplified representation of reality crated to serve a purpose
- Predictive Model:
  - A formula for estimating the unknown value of interest: **the target**
    - The formula can be mathematical, logical statement (e.g., rule), etc.
- Prediction:
  - Estimate an unknown value (i.e. the target)
- Instance / example:
  - Represents a fact or a data point
  - Described by a set of **attributes** (fields, columns, variables, or features)

# Terminology

- Model induction:
  - The creation of **models** from data
- Training data:
  - The input data for the induction algorithm

# Terminology



| Name | Balance | Age | Employed | Write-off |
|---|---|---|---|---|
| Mike | $200,000 | 42 | no | yes |
| Mary | $35,000 | 33 | yes | no |
| Claudio | $115,000 | 40 | no | no |
| Robert | $29,000 | 23 | yes | yes |
| Dora | $72,000 | 31 | no | no |

This is one row (example).
Feature vector is: **<Claudio,115000,40,no>**
Class label (value of Target attribute) is **no**

# What is a model?

A _simplified_* _representation_ of reality created for a _specific purpose_

*based on some assumptions*

- *Examples: map, prototype, Black-Scholes model, etc.*

- *Data Mining example:*

  *"formula" for predicting probability of customer attrition at contract expiration*

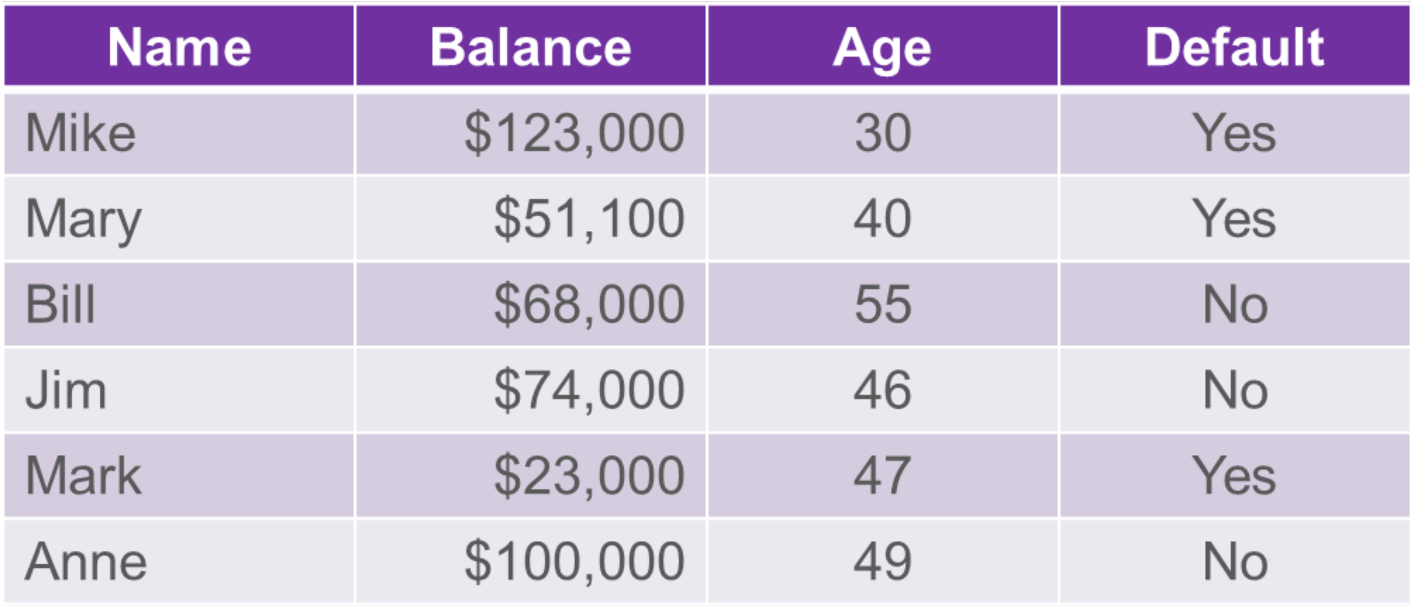
  → *"classification model" or "class-probability estimation model"*

# Feature Types

- Numeric: anything that has some order
  - Numbers (that mean numbers)
  - Dates (that look like numbers …)
  - Dimension of 1
- Categorical: stuff that does not have an order
  - Binary
  - Text
  - Dimension = number of possible values (-1)
- Food for thought: Names, Ratings, Standard Industrial Classification (SIC)

# Dimensionality of the data?

Attributes / Features



| Name | Balance | Age | Default |
|---|---|---|---|
| Mike | $123,000 | 30 | Yes |
| Mary | $51,100 | 40 | Yes |
| Bill | $68,000 | 55 | No |
| Jim | $74,000 | 46 | No |
| Mark | $23,000 | 47 | Yes |
| Anne | $100,000 | 49 | No |

- **Dimensionality of a dataset** is the sum of the dimensions of the features
  - The sum of the number of numeric features and ~ the number of values of categorical features

# Supervised versus Unsupervised Methods

- “Do our customers naturally fall into different groups?”
  - No guarantee that the results are meaningful or will be useful for any particular purpose
- “Can we find groups of customers who have particularly high likelihoods of canceling their service soon after contracts expire?”
  - **A specific purpose**
  - Much more useful results (usually)
  - Different techniques
  - **Requires data on the target**
    - The individual’s label

# Supervised Learning & Predictive Modeling

- Is there a specific, quantifiable **target** that we are interested in or trying to predict?
    - Think about the decision
- Do we have data on this target?
    - Do we have enough data on this target?
        - Need a min ~500 of each type of classification
  - Do we have relevant data prior to decision?
    - Think timing of decision and action
- The result of supervised data mining is a model that predicts some quantity
- A model can either be used to predict or to understand

# Subclasses of Supervised Learning Technique

- Classification
    - Categorical target
        - Often binary
    - Includes “class probability estimation”
- Regression
    - Numeric target

# Subclasses of Supervised Learning Technique

- “Will this customer purchase service $S1$ if given incentive $I1$?”
    - Binary Classification problem
        - Dichotomous target (the customer either purchases or does not)
- “Which service package ($S1$, $S2$, or none) will a customer likely purchase if given incentive $I1$?”
    - Multiclass Classification problem
        - Three-valued target
- “How much will this customer use the service?”
    - Regression problem
        - Numeric target
        - Target variable: amount of usage per customer

# Supervised Learning/Predictive Modeling

*Key (part 1): is there a specific, quantifiable target that we are interested in or trying to predict?*

Examples:

- What will the IBM stock price be tomorrow? (e.g., $200)

  What would you do if you could predict this?

- Will this prospect default her loan? (e.g. yes/no)

  What would you do if you could predict this?

- Do my customers naturally fall into different groups?

  **[Unsupervised: no objective target stated.]**

  What would you do if you could predict this?

# Supervised Learning/Predictive Modeling

*Key (part 2): do we have data on this target?*

*supervised learning requires both parts 1 & 2*

We don’t need the exact data (e.g., whether the current customers will leave) BUT we need data for the same or a related phenomonon (e.g., from customers from last quarter)

→ *we will use these data to build a model to predict the phenomenon of interest*

- Think: Is the phenomenon of “who left the company” last quarter the same as the phenomenon of “who will leave the company” next quarter?
- Think: Who might buy this completely new product I have never sold before?

# Supervised Learning/Predictive Modeling

*Key (part 3): the result of supervised learning is a MODEL that given data predicts some quantity*

- if (income <50K)

  then no Life Insurance

  else Life Insurance

  [Result of Supervised: you can apply this rule to any customer and it gives you prediction ]

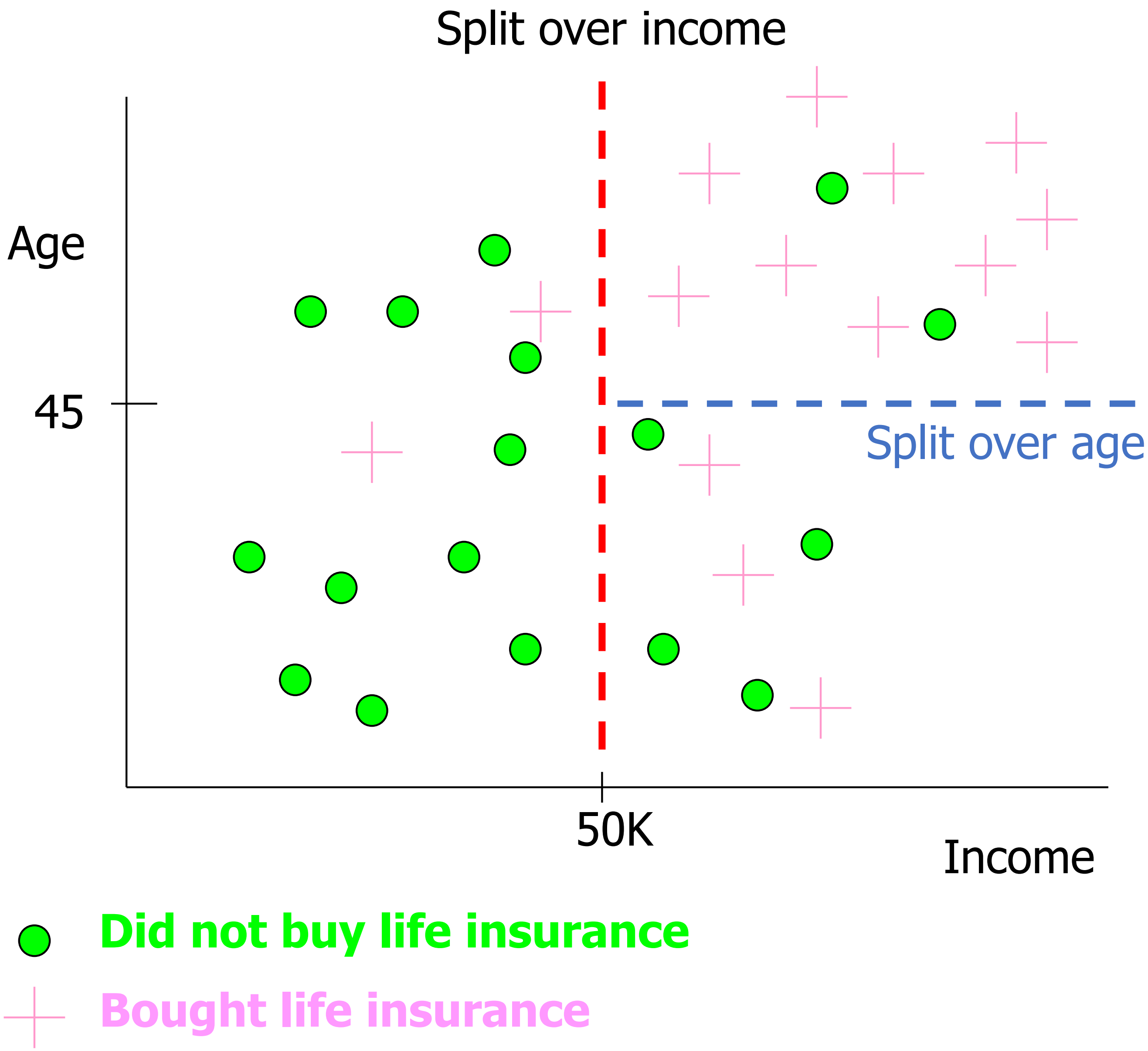
# What might a predictive model look like?

- There are different sorts of model.  Here are just two examples:

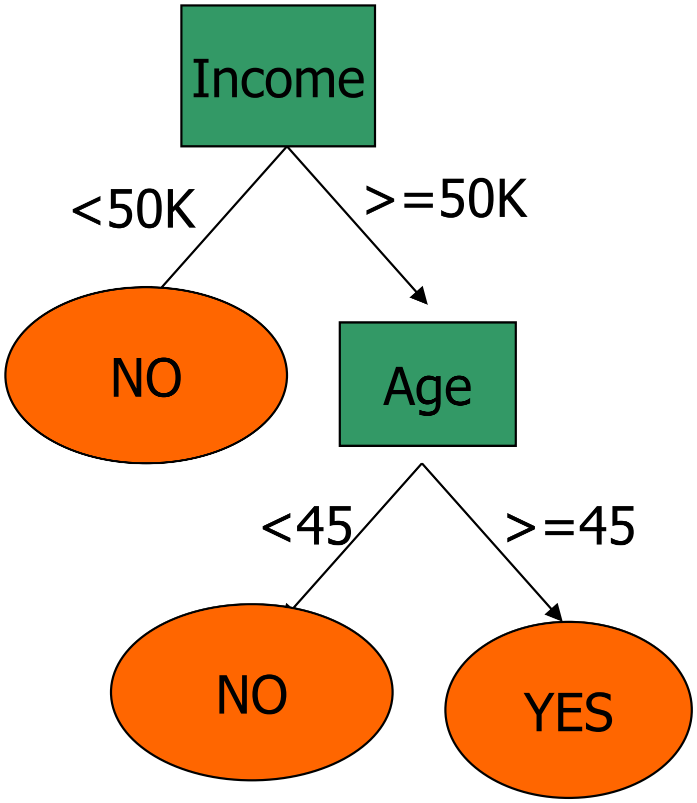
- Tree/Rule: (a supervised segmentation)
- `If (income > $50K) & (age > 45) then LI = YES`
- `If … then …`

- Numeric function:
- $\mathtt{P(LI) = f(x_1, x_2, \ldots, x_k)}$

# What is the model?



## Classification tree

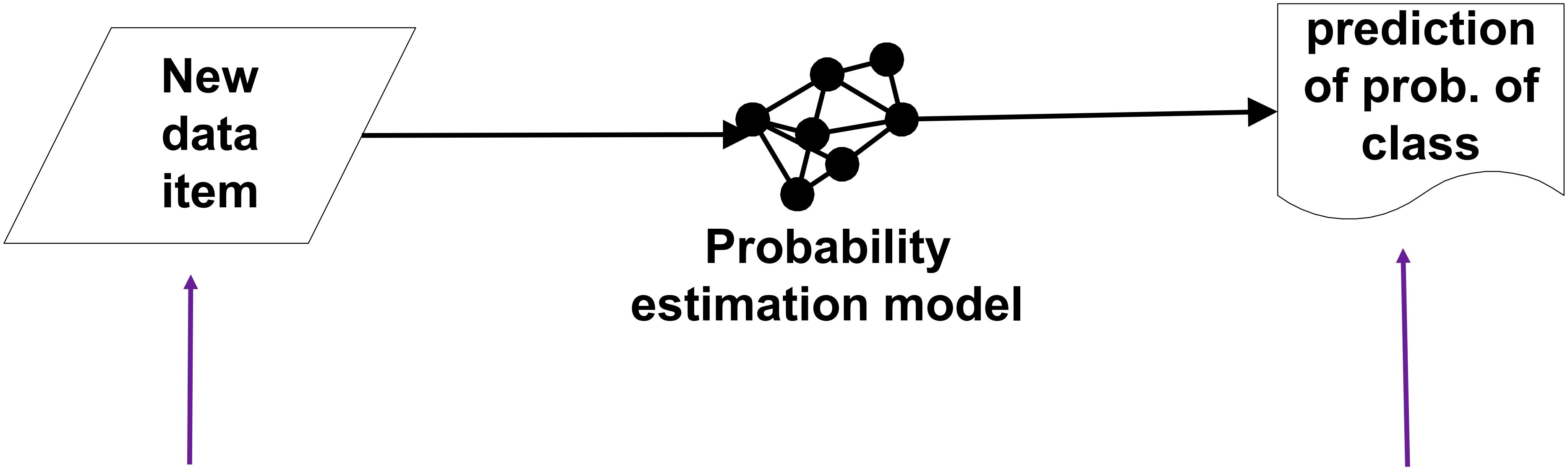


# Supervised Data Mining/Predictive Modeling

*Key (part 4): a data-driven model can either be used to predict or to understand**

**Explanatory modeling can be quite complex. It turns out that you need to understand the fundamentals of predictive modeling first.*

# Predictive Model in Use



New data item

Probability estimation model

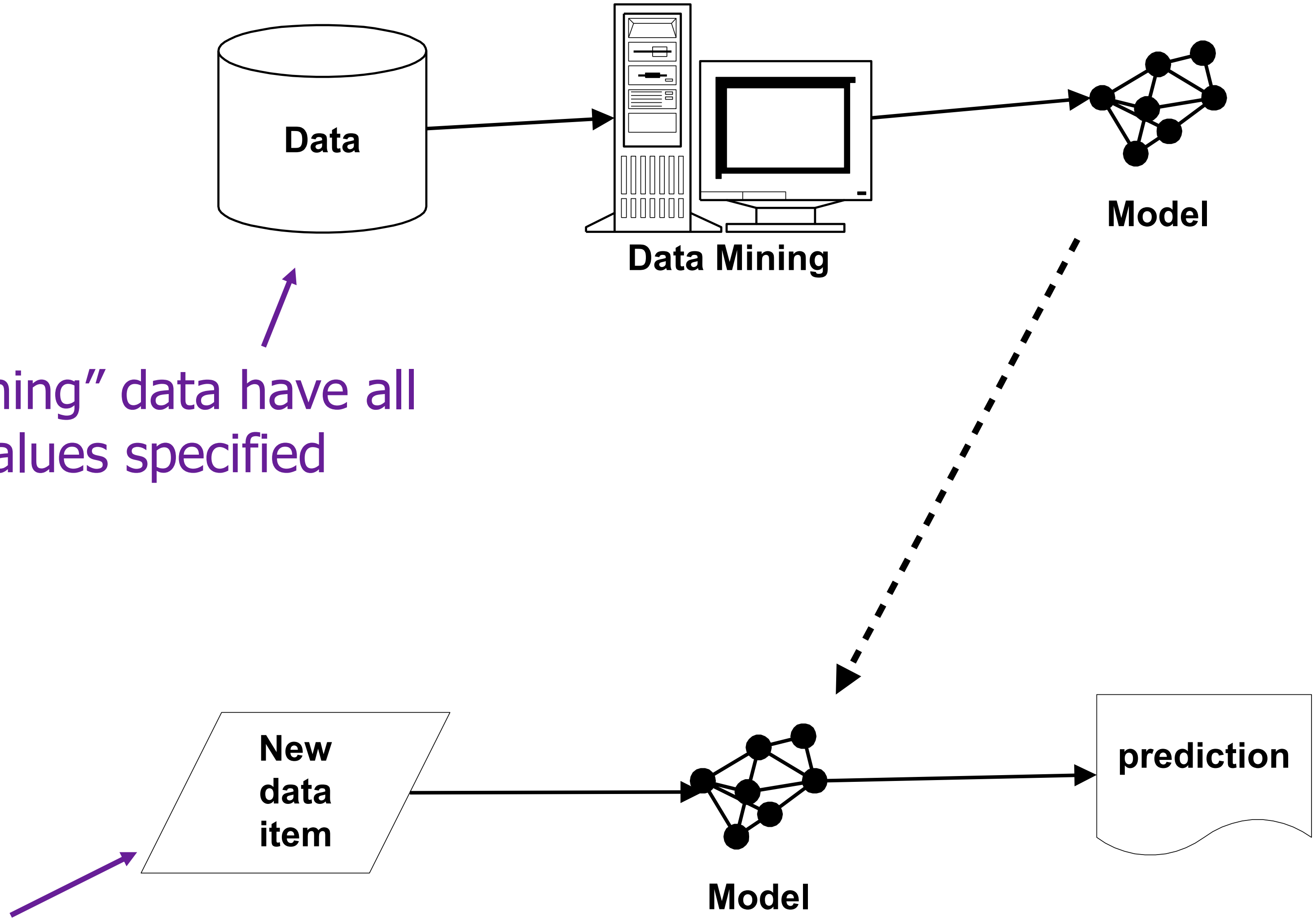
prediction of prob. of class

What is probability of attrition of this customer with characteristics $X, Y, Z$?

$p(C|X, Y, Z) = 0.85$

# Data Mining versus Use of the Model

**"Supervised" modeling:**



Data

Data Mining

Model

"Training" data have all values specified

New data item

Model

prediction

New data item has some value unknown (e.g., will she leave?)

# Classical Pitfalls in modeling setup

- The training data is NOT consistent with the use

- Bad sample
- Bad features

# Sample: “Looking under the streetlight”

- Target Proxy
  - I do not see if a person after seeing an *ad* bought the book, so lets model clicks …
- Sample Proxy
  - I want to run a campaign in Spain but only have data on US customers

https://en.wikipedia.org/wiki/Streetlight_effect

# Sample: “Survivorship issues”

- Lending club wants to have a model to take over the screening (คัดกรอง) process that selects applications and deny those that are likely to default (ผิดชำระหนี้)
- Data of past loans and the outcomes are provided
- Bad:
  - Use only the data (without defaults on the loan), they currently have to predict default (credit default risk)

Recall Key 2: We don’t need the exact data (e.g., whether the application of current customers who will repay their loans) BUT we need data for the same or a related phenomonon (e.g., from customers from last quarter)
→ *we will use these data to build a model to predict the phenomenon of interest*

https://rstudio-pubs-static.s3.amazonaws.com/203258_d20c1a34bc094151a0a1e4f4180c5f6f.html

# Sample: Different Sources

- Things go really bad if the positive and negative are treated differently
- Looking for drivers of diabetes: how do you assemble the training data?

- Bad:
    - Go to a specialized hospital and get records from people treated for diabetes
    - Go somewhere else to get records for healthy people

# Summary on bad habits

- You are missing all the applications that were turned down already
- The sick people came from a very artificial subset
- Your target is NOT really your target
- No way of telling how the model will perform
    - No way of testing either

- The training sample should be as similar as possible to the USE data

# Data Science for Business

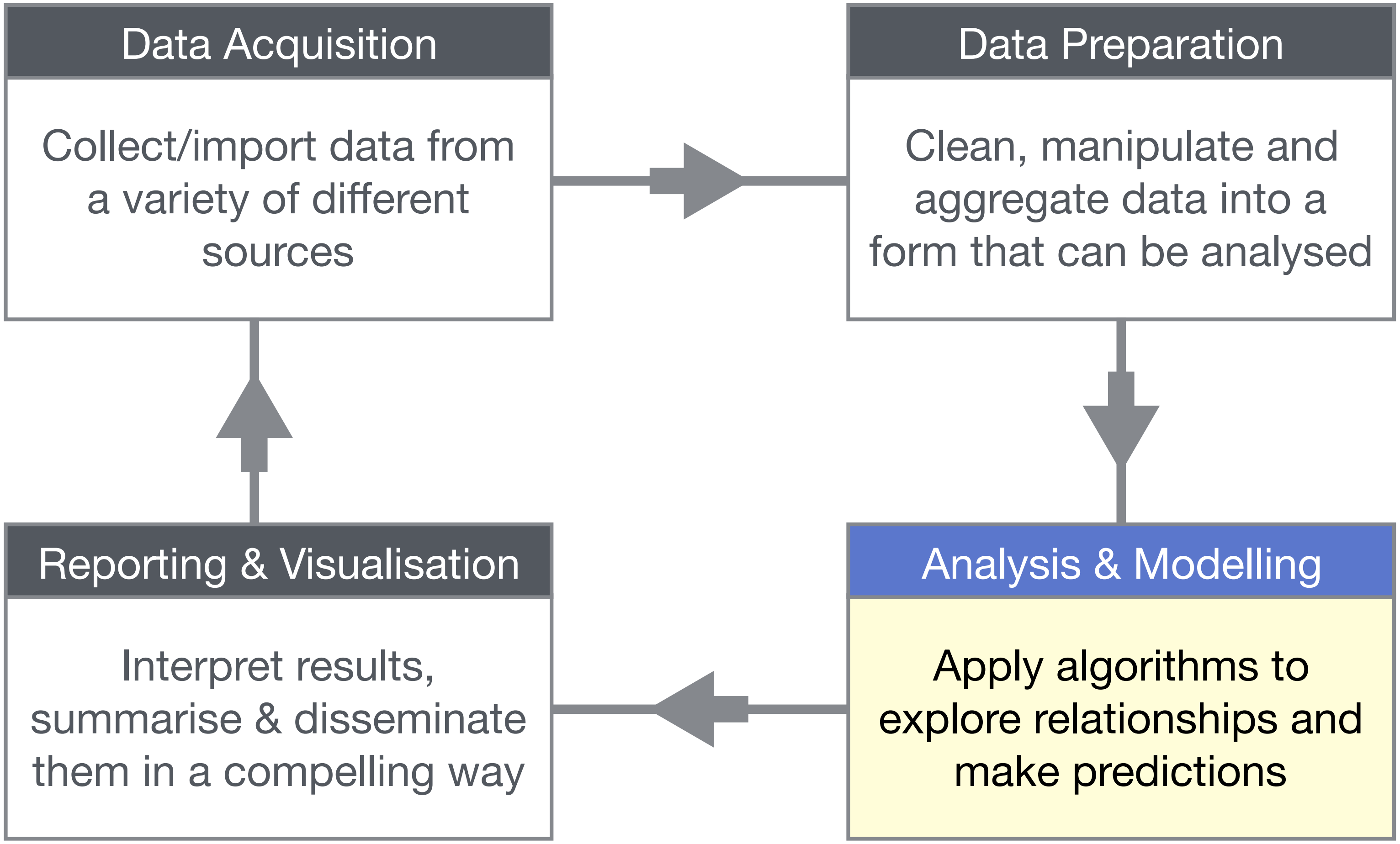
## *Modeling and Prediction in scikit-learn*

Asst. Prof. Teerapong Leelanupab (Ph.D.)
Faculty of Information Technology
King Mongkut's Institute of Technology Ladkrabang (KMITL)

Week 5.3

# Reminder: Data Science Pipeline

- Recall the stages of the basic data science pipeline…
- Most complex component relates to data mining and modelling.



Data Acquisition

Collect/import data from a variety of different sources

Data Preparation

Clean, manipulate and aggregate data into a form that can be analysed

Analysis & Modelling

Apply algorithms to explore relationships and make predictions

Reporting & Visualisation

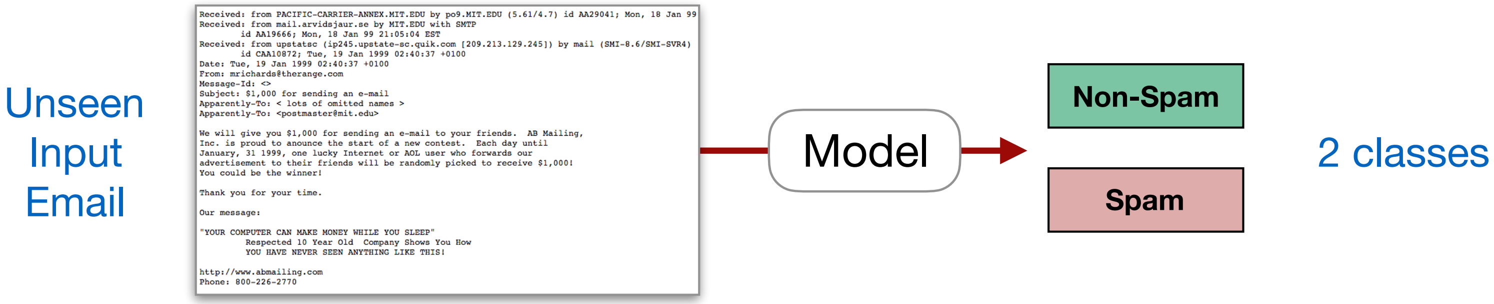
Interpret results, summarise & disseminate them in a compelling way

# Modelling and Prediction Tasks

- Predictive modelling uses statistics to predict outcomes, based on historic data. Also referred to as supervised machine learning.
- **Examples:**
  - Spam filtering: predict if a new email is spam or non-spam, based on annotated examples of past spam / non-spam.
  - Car insurance: assign risk of accidents to policy holders and potential customers.
  - Healthcare: predict disease which a patient has, based on their symptoms.
  - Algorithmic trading: predictive models can be built for different assets like stocks, futures, currencies, etc, based on historic data and company information.

# Supervised Learning

- Classification:
  Learn from a labelled training set to make a prediction to assign a new "unseen" example to one of a fixed number of classes.



Unseen Input Email

```
Received: from PACIFIC-CARRIER-ANNEX.MIT.EDU by po9.MIT.EDU (5.61/4.7) id AA29041; Mon, 18 Jan 99
Received: from mail.arvidsjaur.se by MIT.EDU with SMTP
        id AA19666; Mon, 18 Jan 99 21:05:04 EST
Received: from upstatsc (ip245.upstate-sc.quik.com [209.213.129.245]) by mail (SMI-8.6/SMI-SVR4)
        id CAA10872; Tue, 19 Jan 1999 02:40:37 +0100
Date: Tue, 19 Jan 1999 02:40:37 +0100
From: mrichards@therange.com
Message-Id: <>
Subject: $1,000 for sending an e-mail
Apparently-To: < lots of omitted names >
Apparently-To: <postmaster@mit.edu>

We will give you $1,000 for sending an e-mail to your friends.  AB Mailing,
Inc. is proud to anounce the start of a new contest.  Each day until
January, 31 1999, one lucky Internet or AOL user who forwards our
advertisement to their friends will be randomly picked to receive $1,000!
You could be the winner!

Thank you for your time.

Our message:

"YOUR COMPUTER CAN MAKE MONEY WHILE YOU SLEEP"
         Respected 10 Year Old  Company Shows You How
         YOU HAVE NEVER SEEN ANYTHING LIKE THIS!

http://www.abmailing.com
Phone: 800-226-2770
```

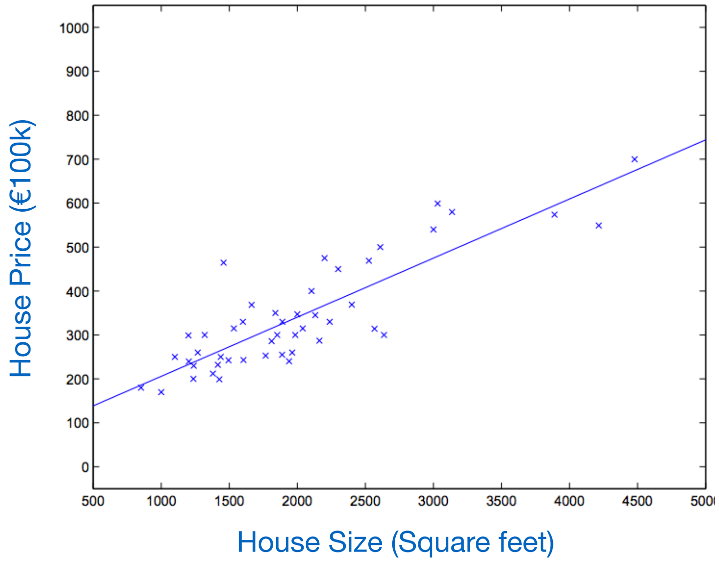
Model

Non-Spam

Spam

2 classes

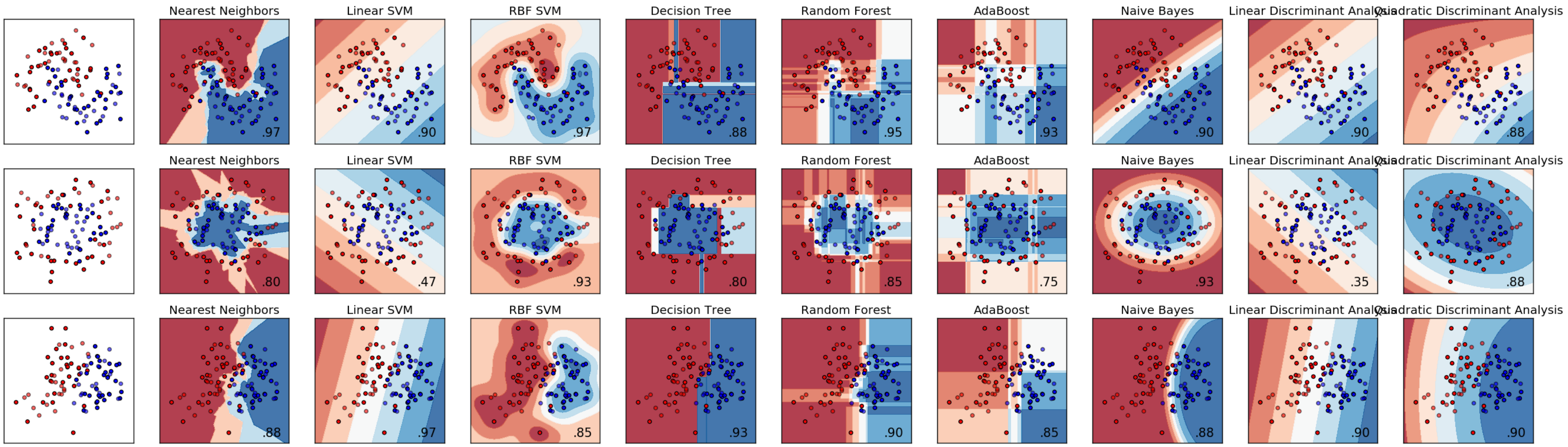
- Regression:
  Learn from a labelled training set to decide the value of a continuous output variable (i.e. the output is a number).



# Scikit-Learn Package

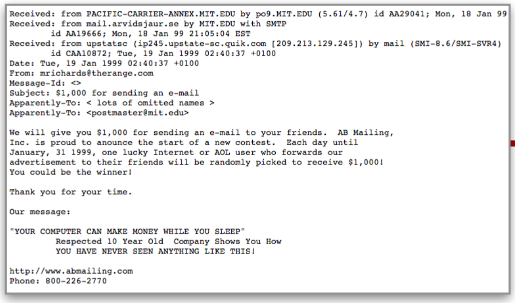
- Scikit-learn is a comprehensive open source Python package for machine learning and data analysis: [http://scikit-learn.org](http://scikit-learn.org)
- Anaconda includes Scikit-learn as part of its free distribution.
- Scikit-learn algorithm inputs and outputs are usually represented as NumPy arrays, although we can also work with input data as Pandas Data Frames.

```
import sklearn
```



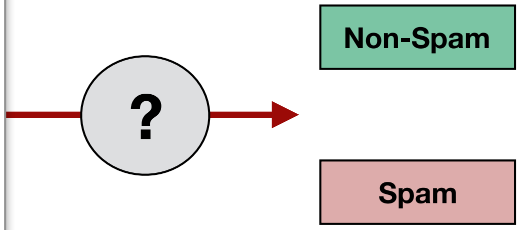
# Types of Classification Tasks

- Binary Classification:
  Assign an input to one of two possible target class labels.



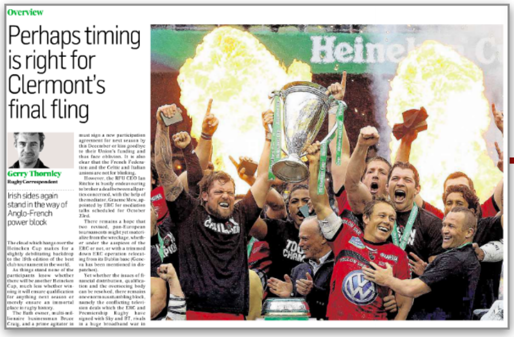
```
Received: from PACIFIC-CARRIER-ANNEX.MIT.EDU by po9.MIT.EDU (5.61/4.7) id AA29041; Mon, 18 Jan 99
Received: from mail.arvidsjaur.se by MIT.EDU with SMTP
        id AA19666; Mon, 18 Jan 99 21:05:04 EST
Received: from upstatsc (ip245.upstate-sc.quik.com [209.213.129.245]) by mail (SMI-8.6/SMI-SVR4)
        id CAA10872; Tue, 19 Jan 1999 02:40:37 +0100
Date: Tue, 19 Jan 1999 02:40:37 +0100
From: mrichards@therange.com
Message-Id: <>
Subject: $1,000 for sending an e-mail
Apparently-To: < lots of omitted names >
Apparently-To: <postmaster@mit.edu>

We will give you $1,000 for sending an e-mail to your friends.  AB Mailing,
Inc. is proud to anounce the start of a new contest.  Each day until
January, 31 1999, one lucky Internet or AOL user who forwards our
advertisement to their friends will be randomly picked to receive $1,000!
You could be the winner!

Thank you for your time.

Our message:

"YOUR COMPUTER CAN MAKE MONEY WHILE YOU SLEEP"
         Respected 10 Year Old  Company Shows You How
         YOU HAVE NEVER SEEN ANYTHING LIKE THIS!

http://www.abmailing.com
Phone: 800-226-2770
```

Input



2 classes

- Multiclass Classification:
  Assign an input to one of *M* different target class labels.



Overview

Perhaps timing is right for Clermont's final fling

Gerry Thornley
Rugby Correspondent

Irish sides again stand in the way of Anglo-French power block

The cloud which hangs over the Heineken Cup makes for a slightly debilitating backdrop to the 19th edition of the best club tournament in the world.

As things stand none of the participants know whether there will be another Heineken Cup, much less whether winning it will ensure qualification for anything next season or merely ensure an immortal place in rugby history.

The Bath owner, multi-millionaire businessman Bruce Craig, and a prime agitator in

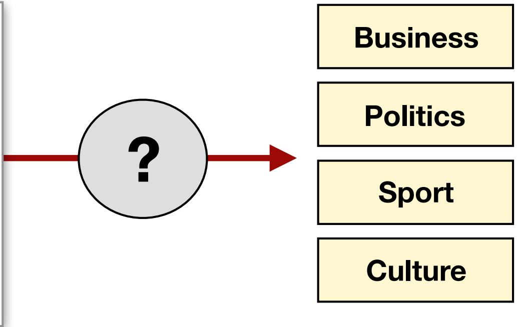
must sign a new participation agreement for next season by this December or kiss goodbye to their Union's funding and thus face oblivion. It is also clear that the French Federation and the Celtic and Italian unions are not for blinking.

However, the RFU CEO Ian Ritchie is busily endeavouring to broker a deal between all parties concerned, with the help of the mediator, Graeme Mew, appointed by ERC for mediation talks scheduled for October 23rd.

There remains a hope that two revised, pan-European tournaments might yet materialise from the wreckage, whether under the auspices of the ERC or not, or with a trimmed down ERC operation relocating from its Dublin base (Geneva has been mentioned in dispatches).

Yet whether the issues of financial distribution, qualification and the overseeing body can be resolved, there remains one enormous stumbling block, namely the conflicting television deals which the ERC and Premiership Rugby have signed with Sky and BT, rivals in a huge broadband war in

Input

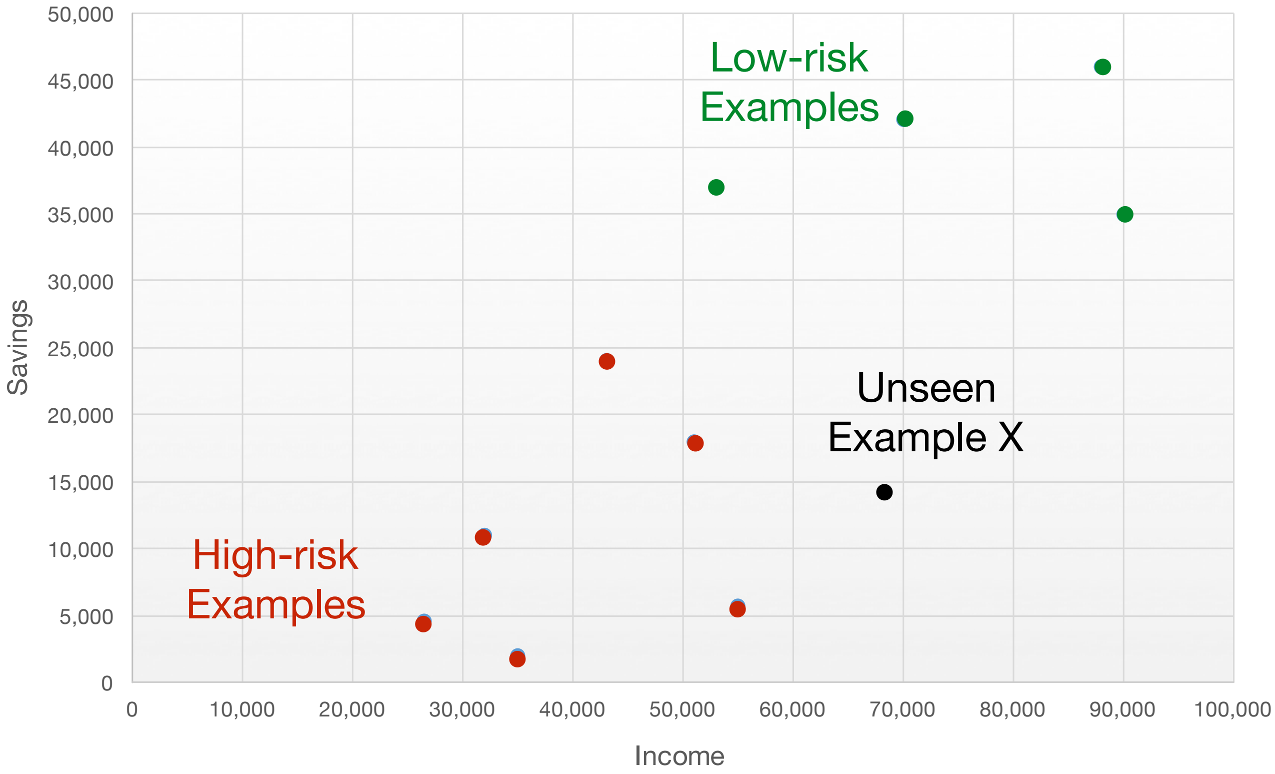


4 classes

# Typical Binary Classification Task

**Example:**
**Mortgage Approval**

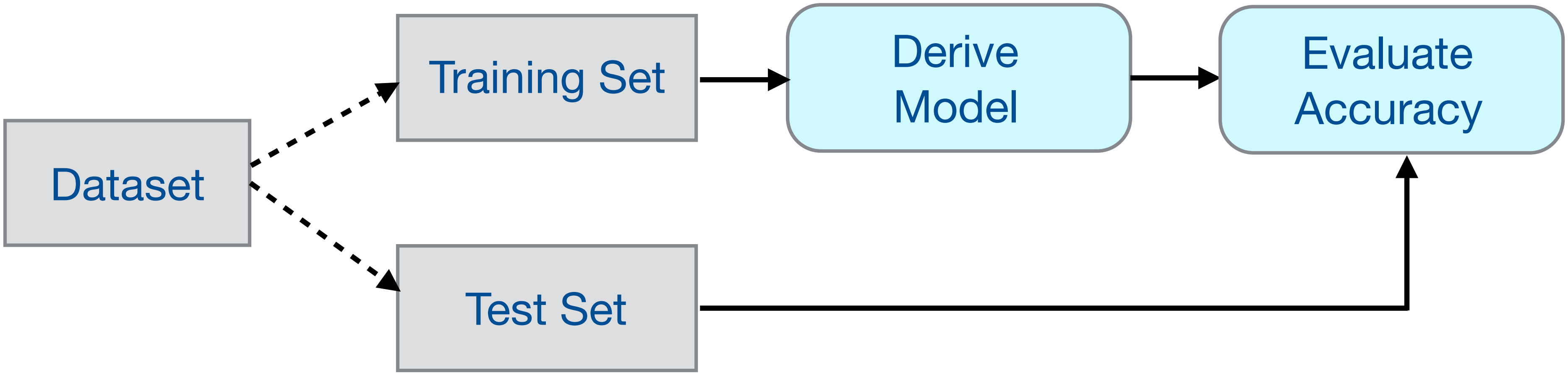
Manually classify mortgage applicants into two categories (low-risk & high-risk) based on savings and income data.



Q. Can we train an algorithm to learn to automatically classify a new application X as either low-risk or high-risk, based on the manually-classified examples of each type?

# Training v Test Data

- Standard evaluation approach for classification tasks is to split the set of examples into a *training set* and the *test set.*
- Training set: Examples provided to the classifier to build a model of the data. Each has been manually assigned a class label.
- Test set:  Examples held back from the classifier, which are used to evaluate the accuracy of the classifier. Test examples are completely separate from the training set, so we can evaluate how well the model built by the classifier will generalise to new input examples.



# Classification Algorithms

- Many different learning algorithms exist for classification (e.g. k-nearest neighbour, decision tree, neural network, support vector machine).
- Problem dimensions will often determine which classification algorithm will be practically applicable, due to processing, memory, and storage requirements.

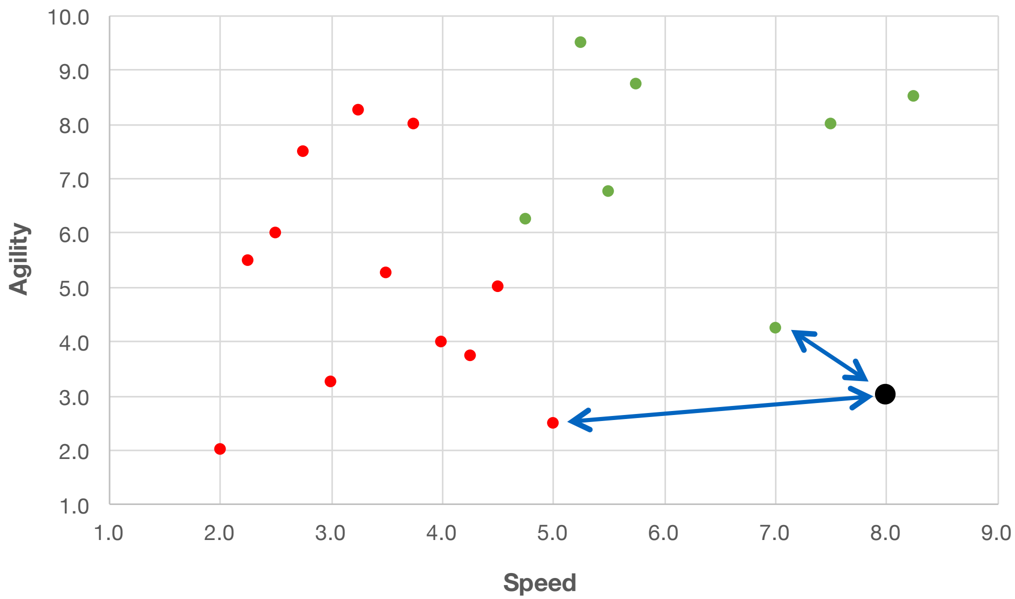
1. Number of examples *N*.
   → Sometimes millions of input examples.
2. Number of features (dimensions) *D* representing each example.
   → Often 10-1000, but sometimes far higher.
3. Number of target classes *M*.
   → Often small (binary), but sometimes far higher.

# Similarity/Distance-based Learning

**Fundamental Strategy:** “Best way to make predictions is to look at past examples and repeat the same process again”.

Feature space:
A *D*-dimensional coordinate space used to represent the input examples for a given problem, with one coordinate per descriptive feature.

Similarity measure:
Some function to measure how similar (or distant) two input examples are from one another are in the *D*-dimensional coordinate space.



2 features describing each example (agility & speed)

➡ 2 coordinate dimensions for measuring similarity

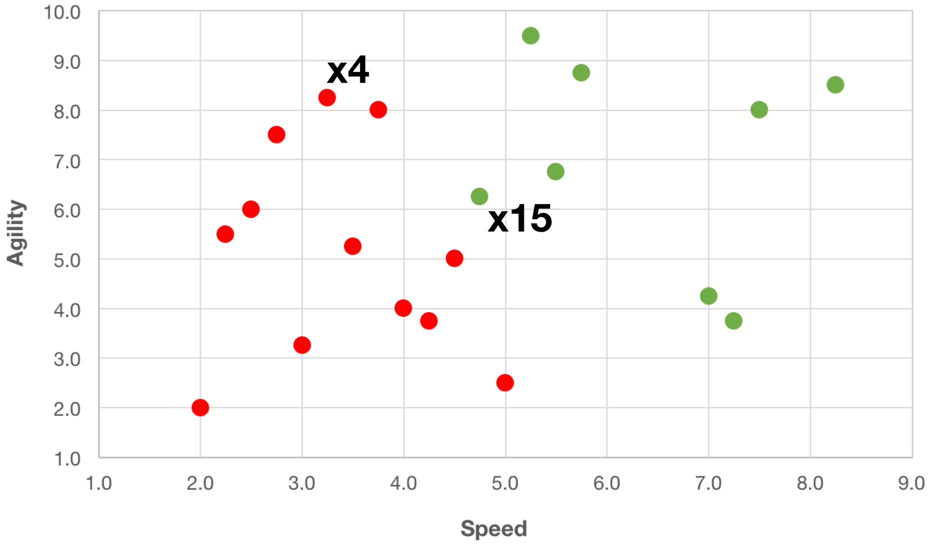
**Typically:** $\text{Distance} = 1/\text{Similarity}$ OR $\text{Distance} = 1 - \text{Similarity}$

# Similarity/Distance-based Learning

- For numeric data, the most common distance function used is the Euclidean distance.
- Calculated as square root of sum of squared differences for each feature *f* representing a pair of examples.

$$\mathrm{ED}(\mathbf{p},\mathbf{q}) = \sqrt{\sum_{f \in F} (q_f - p_f)^2}$$

| *Example* | *Speed* | *Agility* |
|---|---|---|
| **x4** | 3.25 | 8.25 |
| **x15** | 4.75 | 6.25 |

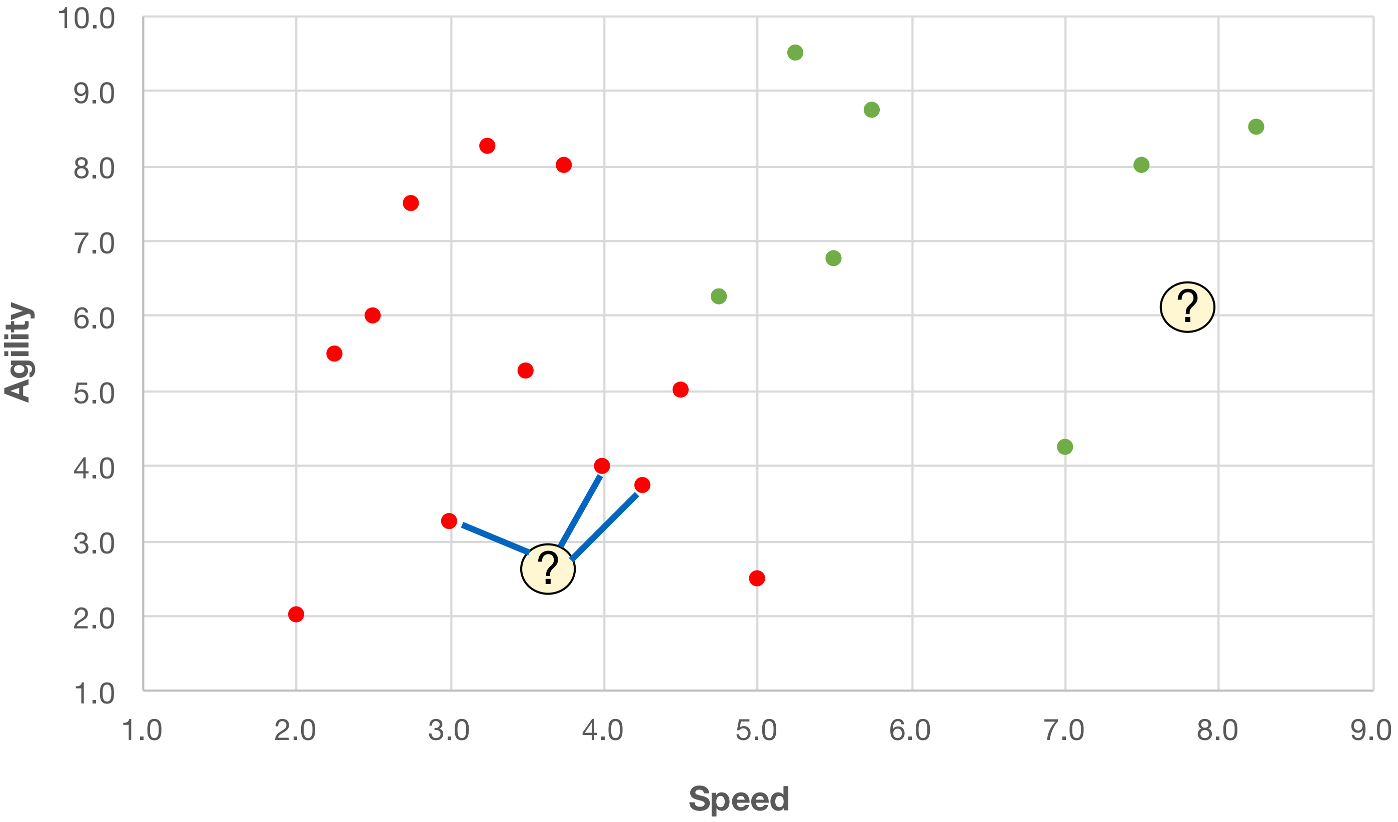


$$ED(x4, x15) = \sqrt{(3.25 - 4.75)^2 + (8.25 - 6.25)^2} = \sqrt{6.25} = 2.5$$

# KNN Classifier

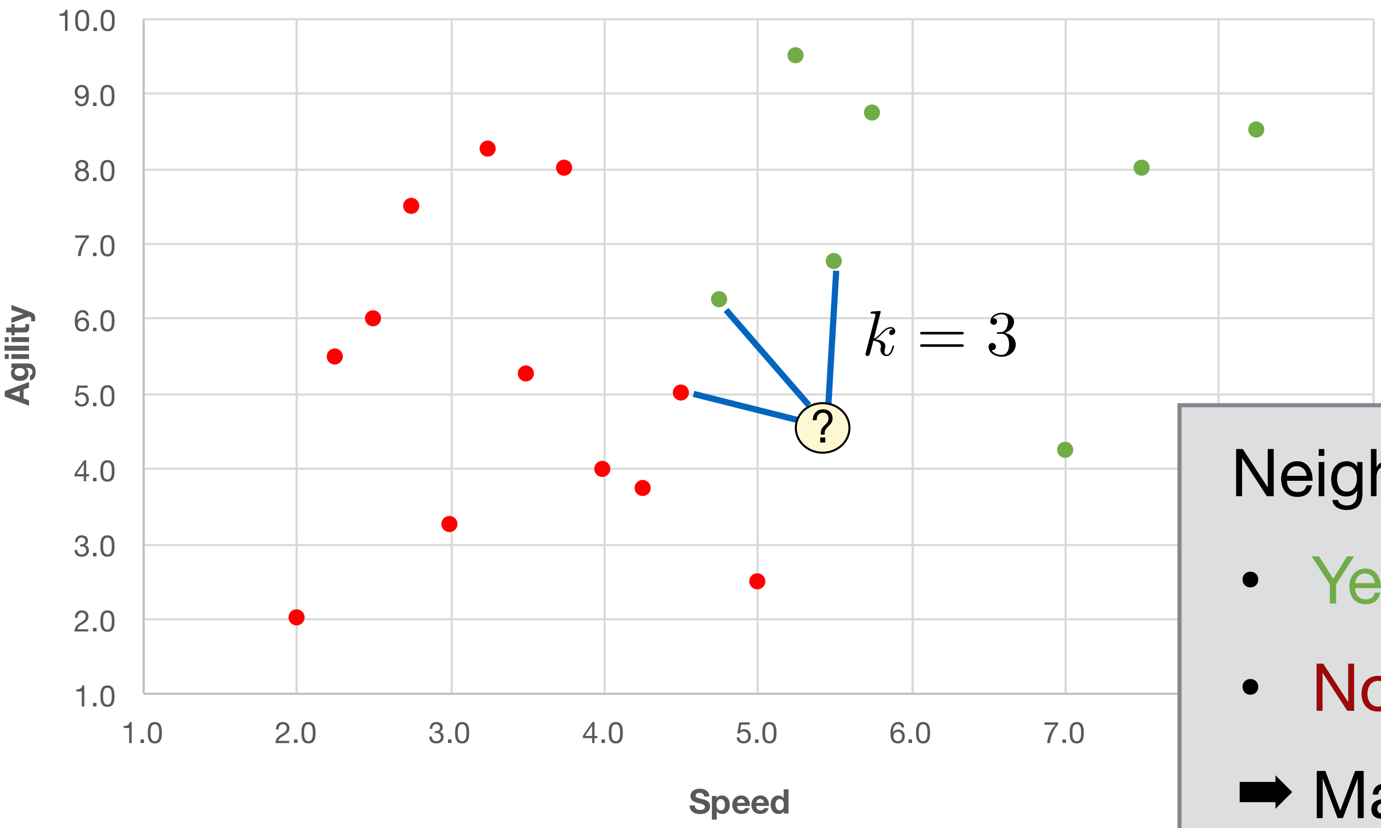
*k*-Nearest Neighbour Classifier: Simple but effective"lazy" classifier. Find most similar previous examples for which a decision has already been made (i.e. their nearest neighbours from the training set).

**Example:** For new unseen examples, look at the label of *k* nearest neighbours under both features (e.g. *k=3* neighbours)



# KNN Classifier

Majority voting: The decision on a label for a new query example is decided based on the “votes” of its *k* nearest neighbours, where the neighbours are selected to minimise the distance $d(q, x_i)$



Neighbour counts

- Yes = 2 votes
- No = 1 vote

➡ Majority says Yes!

# KNN Classifier in Python

- Scikit-learn has a range of different classification algorithms: http://scikit-learn.org/stable/supervised_learning.html
- In all cases, we call the `fit()` function to build a model on training data and then the `predict()` function on unseen data.
- **Example**: Classification using the *Iris* flower dataset, which is built into scikit-learn.

```
from sklearn.datasets import load_iris
iris = load_iris()
```

- When we inspect the dataset, we see that that it has a numpy array of feature values (data) and a set of target labels.

```
print(iris.data)
```

```
[[ 5.1   3.5   1.4   0.2]
 [ 4.9   3.    1.4   0.2]
 [ 4.7   3.2   1.3   0.2]
 [ 4.6   3.1   1.5   0.2]
 [ 5.    3.6   1.4   0.2]
...
```

```
print(iris.target_names)
```

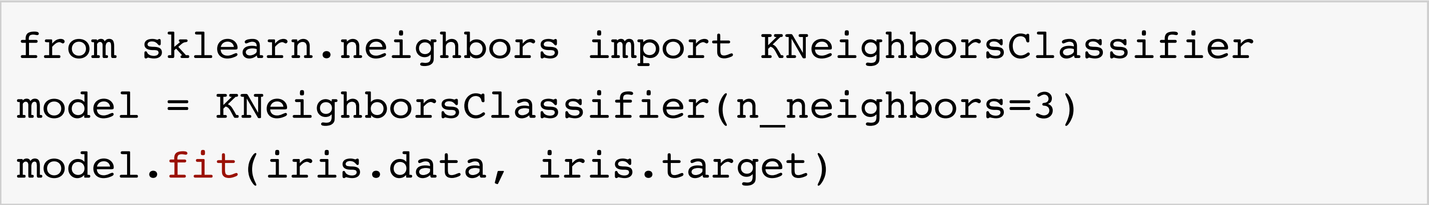
```
['setosa' 'versicolor' 'virginica']
```

```
print(iris.target)
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2]
```

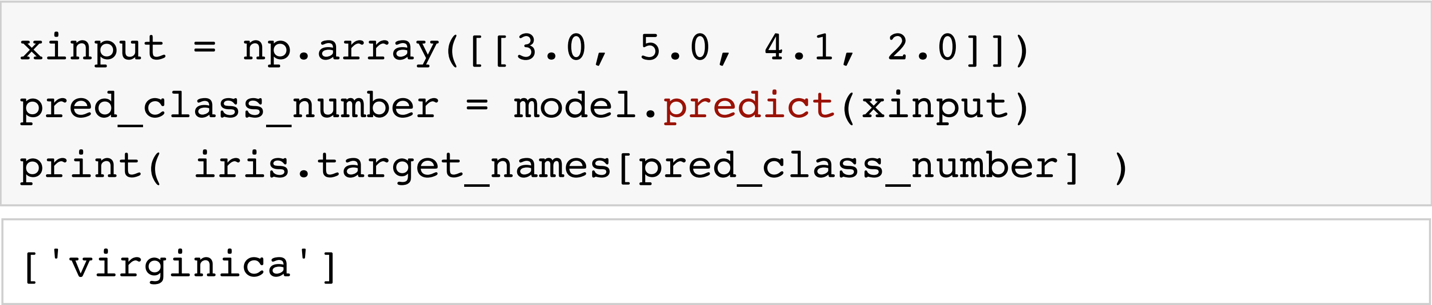
# KNN Classifier in Python

- Goal: Train a classifier to predict the species (class label) for a new numeric record describing a flower.
- Three possible species (class labels), so this is a multi-class problem: 'setosa', 'versicolor', or 'virginica'.
- Step 1: Fit a KNN classifier with K=3 neighbours on the full dataset - this is our training data, using the `fit()` function.



```
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier(n_neighbors=3)
model.fit(iris.data, iris.target)
```

- Step 2:  Make a prediction for a new unseen input example using the trained KNN model, using the `predict()` function.

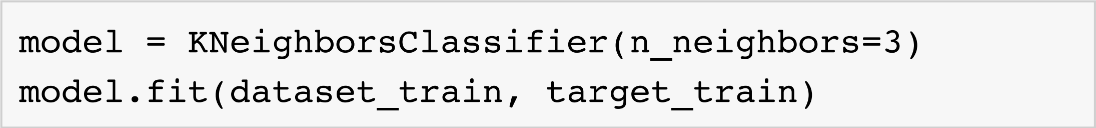


```
xinput = np.array([[3.0, 5.0, 4.1, 2.0]])
pred_class_number = model.predict(xinput)
print( iris.target_names[pred_class_number] )
```

```
['virginica']
```

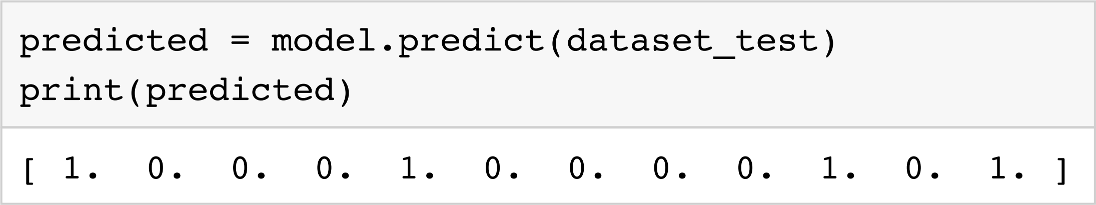
# KNN Classifier in Python

- We can also generate predictions for multiple input examples using a single call to `predict()` - i.e. apply classifier to a new test set.
- In general we will have 4 variables:
  1. `dataset_train`: The set of data on which to build the model.
  2. `target_train`: The true class labels for the training data.
  3. `dataset_test`: The set of data on which to test the model.
  4. `target_test`: The true class labels for the test data (if available).



```
model = KNeighborsClassifier(n_neighbors=3)
model.fit(dataset_train, target_train)
```

Fit a KNN classifier with K=3 neighbours on the training data

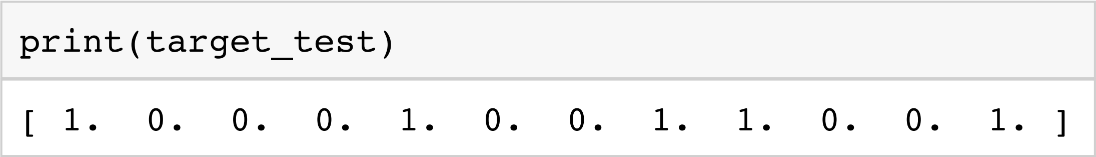


```
predicted = model.predict(dataset_test)
print(predicted)
```

```
[ 1.  0.  0.  0.  1.  0.  0.  0.  0.  1.  0.  1. ]
```

Make a prediction for the test set

A prediction is made for each input in the test set



```
print(target_test)
```

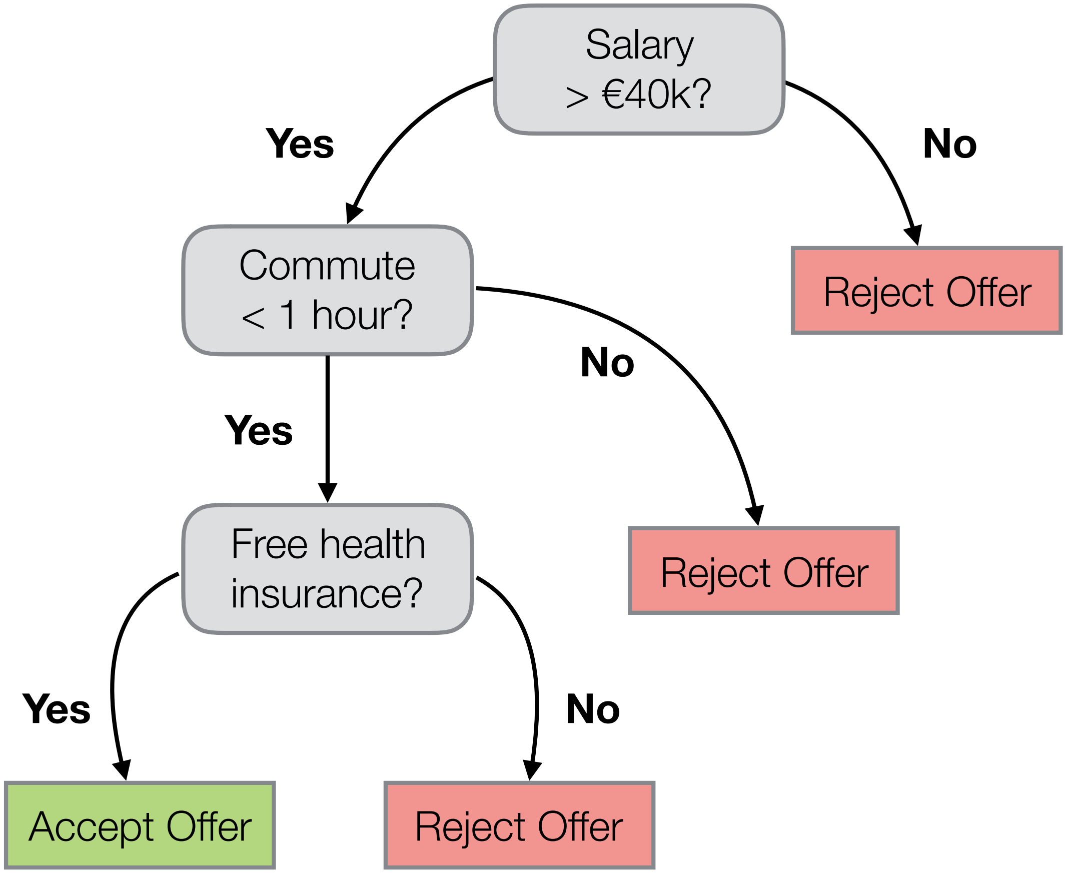
```
[ 1.  0.  0.  0.  1.  0.  0.  1.  1.  0.  0.  1. ]
```

We can compare the predictions to the true labels for the test data

# Decision Tree Classifier

- Create a classification model to predict class labels by learning decision rules learned from training data.

**Decision:**
Should I accept the job offer?



# Decision Tree Classifier

- **Basic Idea:** Build a tree model that splits the training set into subsets in which all examples have the same class.
- Splitting is done using rules inferred from the training set.
- Each rule is based on a feature, and the split corresponds to the values it can take:
  - e.g. insured = {true, false}
  - e.g. wind = {weak, strong}
  - e.g. income = {low, average, high}
  - e.g. height < 6ft, height ≥ 6ft
- If necessary, each subset can be split again using another feature, and so on until all examples have the same class.
- Once the tree is built, we can use it to quickly classify new input examples - this is an eager learning strategy.

# Decision Tree Example

Q. “Will a customer wait for a restaurant table?”

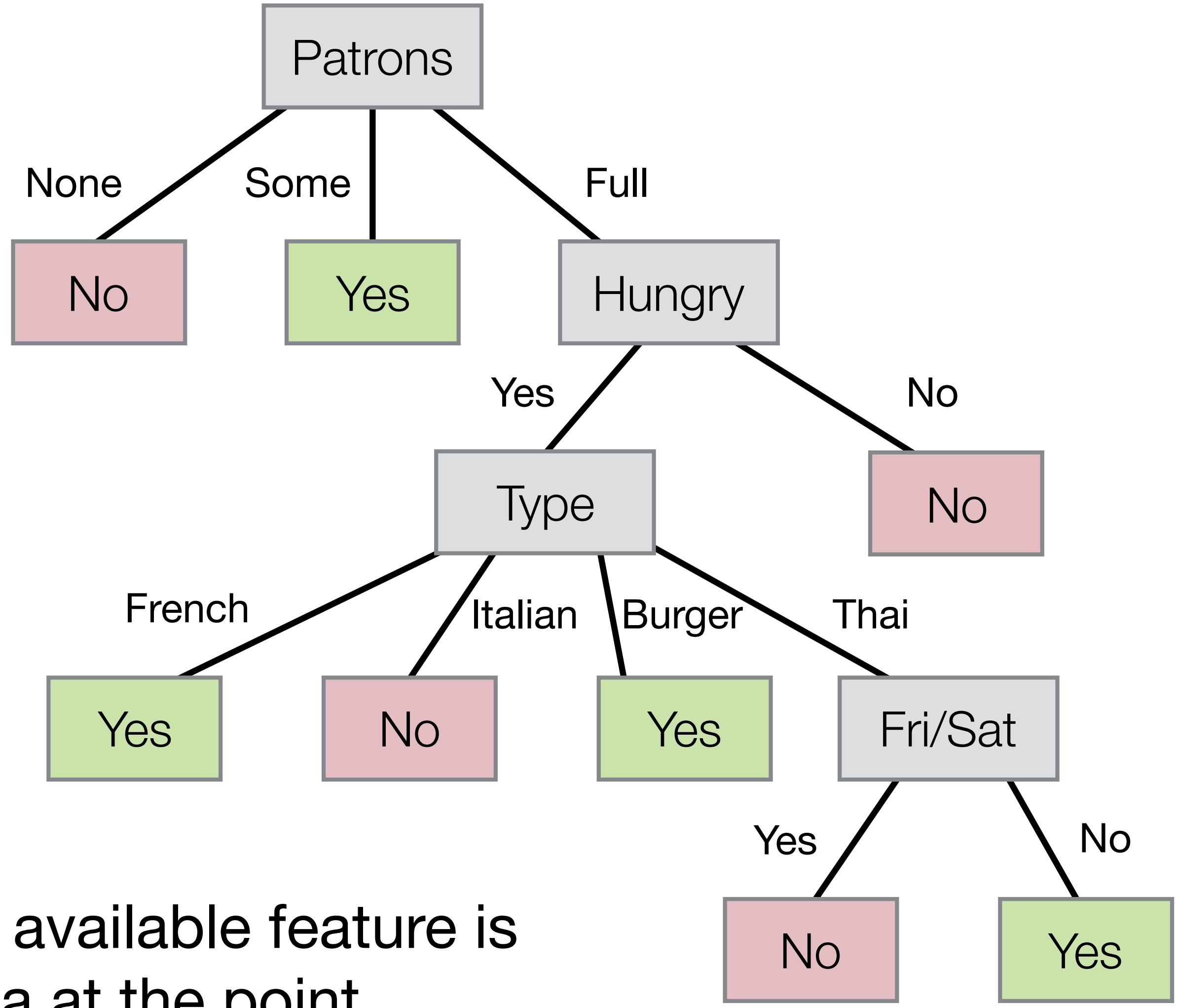
*Russell & Norvig, Artificial Intelligence: A Modern Approach, Prentice Hall, 2009.*

Binary classification task (WillWait = {Yes,No}), with examples described by 10 different features:

| Example | Alternate | Bar | Fri/Sat | Hungry | Patrons | Price | Raining | Reservation | Type | WaitEst | WillWait? |
|---|---|---|---|---|---|---|---|---|---|---|---|
| *1* | Yes | No | No | Yes | Some | €€€ | No | Yes | French | 0-10 | Yes |
| *2* | Yes | No | No | Yes | Full | € | No | No | Thai | 30-60 | No |
| *3* | No | Yes | No | No | Some | € | No | No | Burger | 0-10 | Yes |
| *4* | Yes | No | Yes | Yes | Full | € | No | No | Thai | 10-30 | Yes |
| *5* | Yes | No | Yes | No | Full | €€€ | No | Yes | French | >60 | No |
| *6* | No | Yes | No | Yes | Some | €€ | Yes | Yes | Italian | 0-10 | Yes |
| *7* | No | Yes | No | No | None | € | Yes | No | Burger | 0-10 | No |
| *8* | No | No | No | Yes | Some | €€ | Yes | Yes | Thai | 0-10 | Yes |
| *9* | No | Yes | Yes | No | Full | € | Yes | No | Burger | >60 | No |
| *10* | Yes | Yes | Yes | Yes | Full | €€€ | No | Yes | Italian | 10-30 | No |
| *11* | No | No | No | No | None | € | No | No | Thai | 0-10 | No |
| *12* | Yes | Yes | Yes | Yes | Full | € | No | No | Burger | 30-60 | Yes |

# Decision Tree Example

- A “good” decision tree will classify all examples correctly using as few tree nodes as possible.

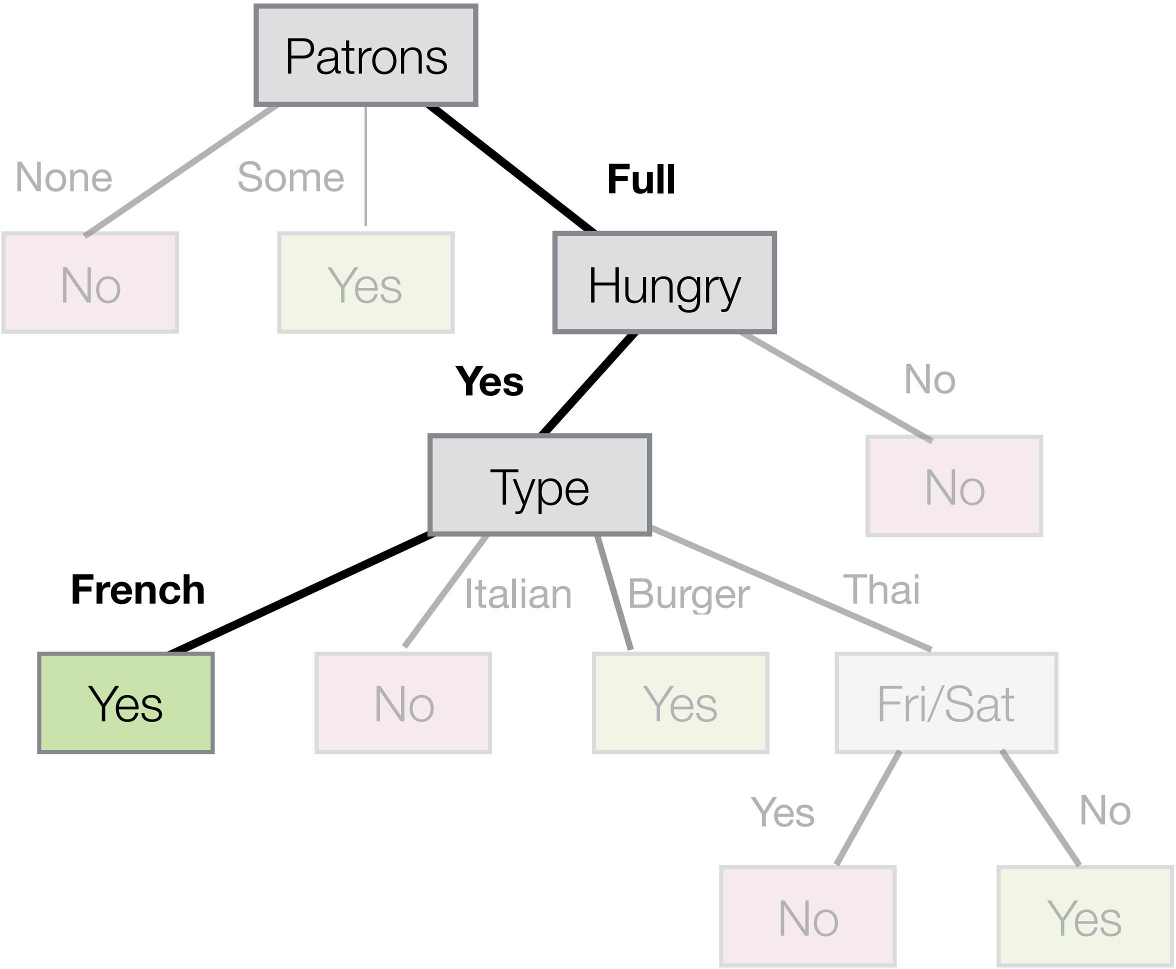


At each level, the best available feature is chosen to split the data at the point.

# Decision Tree Example

- Once we have built a decision tree from the training set, we can use the rules in the tree to quickly classify new input examples.

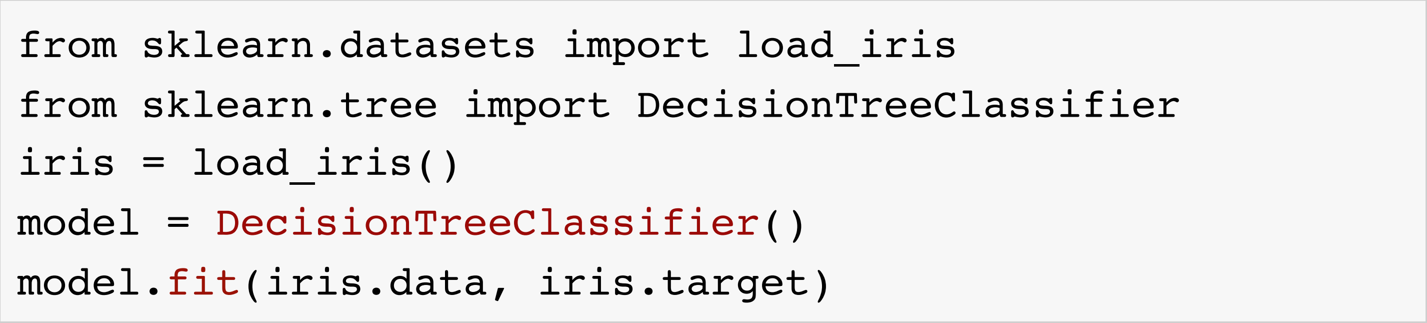
| **Feature** | **Query *x1*** |
|---|---|
| *Alternate* | Yes |
| *Bar* | No |
| *Fri/Sat* | Yes |
| *Hungry* | Yes |
| *Patrons* | Full |
| *Price* | €€ |
| *Raining* | No |
| *Reservation* | No |
| *Type* | French |
| *WaitEstimate* | 10-30 |



⇒ Based on tree, output for *x1* is "Yes"

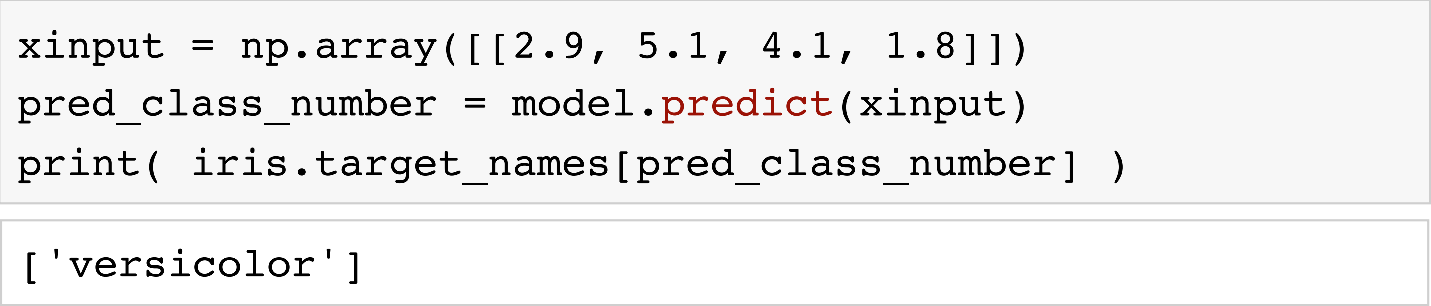
# Decision Tree Classifier in Python

- **Example:** Apply a Decision Tree to the *Iris* dataset.
- Step 1: Apply a Decision Tree classifier on the full dataset, again using the `fit()` function to create the model.



```
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier
iris = load_iris()
model = DecisionTreeClassifier()
model.fit(iris.data, iris.target)
```

- Step 2:  Make a prediction for a new unseen input example using the trained tree model, using the `predict()` function.



```
xinput = np.array([[2.9, 5.1, 4.1, 1.8]])
pred_class_number = model.predict(xinput)
print( iris.target_names[pred_class_number] )
```

```
['versicolor']
```

# Alternative Classifiers

- **Naive Bayes**: Simple classifier based on counts. Requires less training data, assumes all features are independent.

```
from sklearn.naive_bayes import MultinomialNB
model = MultinomialNB()
model.fit(data_train, target_train)
predicted = model.predict(data_test)
```

- **Support Vector Machines**: SVMs use a kernel function to accurately separate different classes. Difficult to interpret, can require parameter tuning.

```
from sklearn.svm import SVC
model = SVC()
model.fit(data_train, target_train)
predicted = model.predict(data_test)
```

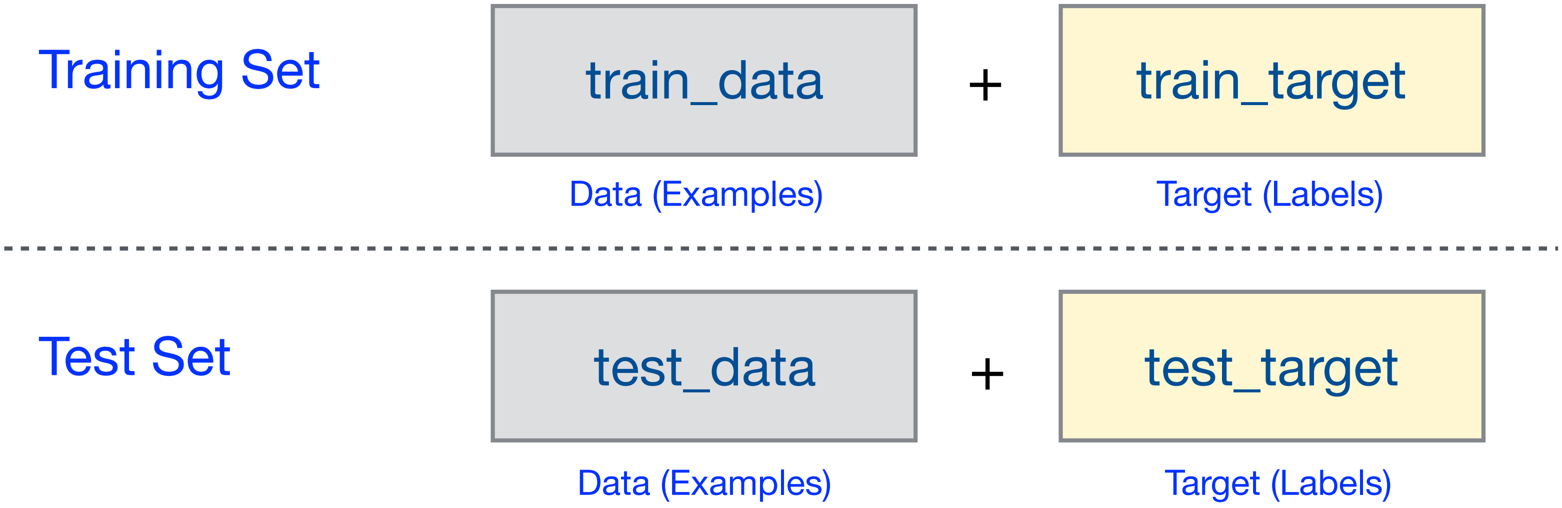
- **Logistic Regression**: Probabilistic model, works well in an online setting.

```
from sklearn import linear_model
model = linear_model.LogisticRegression()
model.fit(data_train, target_train)
predicted = model.predict(data_test)
```

# Evaluating Classifiers

- Reminder - Standard evaluation approach for classification tasks is to split the set of examples into two sets:
  1. Training set: Examples provided to the classifier to build a model of the data. Each has been assigned a class label.
  2. Test set:  Separate set of examples which are used to evaluate the accuracy of the classifier.

- **Hold-out strategy:** Randomly hold back some examples (e.g. 20%) for evaluation. In Scikit-learn, this split involves 4 variables:



Training Set

train_data + train_target

Data (Examples)

Target (Labels)

Test Set

test_data + test_target

Data (Examples)

Target (Labels)

# Evaluation Measures

When making predictions, we need some kind of measure to capture how often the model makes correct or incorrect predictions, and how severe the mistakes are.

Accuracy:
Simplest measure. Fraction of correct predictions made by the classifier.

$$ACC = \frac{\#\ \text{correct predictions}}{\text{total predictions}}$$

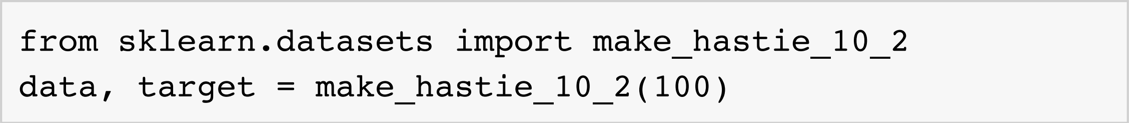
Example: Predictions made for test set of 10 emails

$$ACC = \frac{7}{10} = 0.7$$

| Email | Label | Prediction | Correct? |
|---|---|---|---|
| *1* | spam | non-spam | ✗ |
| *2* | spam | spam | ✓ |
| *3* | non-spam | non-spam | ✓ |
| *4* | spam | spam | ✓ |
| *5* | non-spam | spam | ✗ |
| *6* | non-spam | non-spam | ✓ |
| *7* | spam | spam | ✓ |
| *8* | non-spam | spam | ✗ |
| *9* | non-spam | non-spam | ✓ |
| *10* | spam | spam | ✓ |

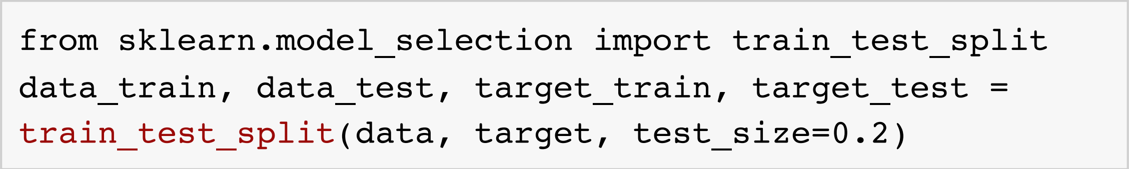
# Measuring Accuracy in Python

- The `sklearn.metrics` package provides evaluation functionality.
- To demonstrate, we will create an artificial dataset, where the examples have 2 binary class labels [-1,1]:



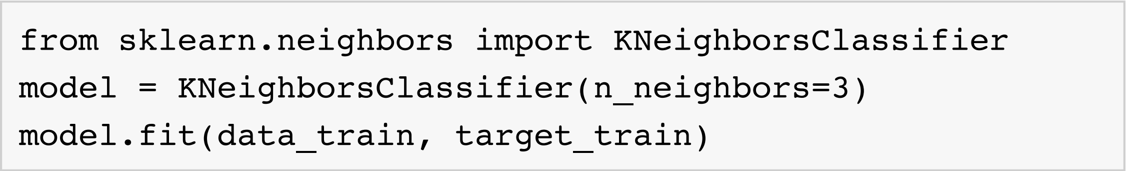
```
from sklearn.datasets import make_hastie_10_2
data, target = make_hastie_10_2(100)
```

100 examples in total, both the data and the target labels



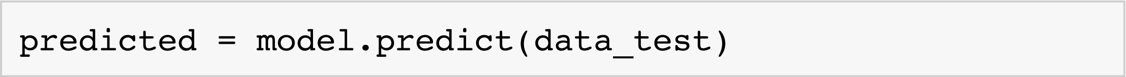
```
from sklearn.model_selection import train_test_split
data_train, data_test, target_train, target_test = 
train_test_split(data, target, test_size=0.2)
```

Randomly split the data into training and test sets. Test set size is 20% (0.2)



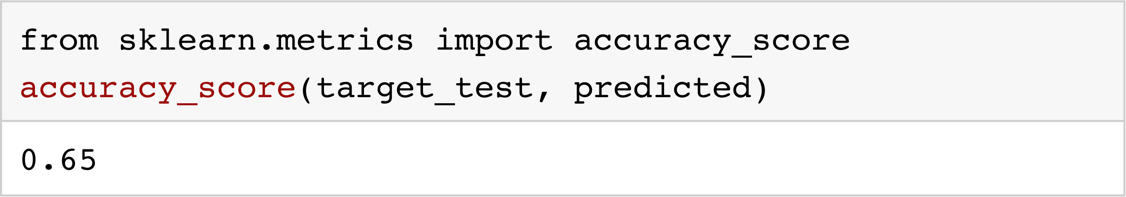
```
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier(n_neighbors=3)
model.fit(data_train, target_train)
```

Apply KNN classifier to training data with target labels



```
predicted = model.predict(data_test)
```

Make predictions for test data



```
from sklearn.metrics import accuracy_score
accuracy_score(target_test, predicted)
```

```
0.65
```

Calculate accuracy score for the predictions, based on actual target labels for test data