# Introduction to Data Science

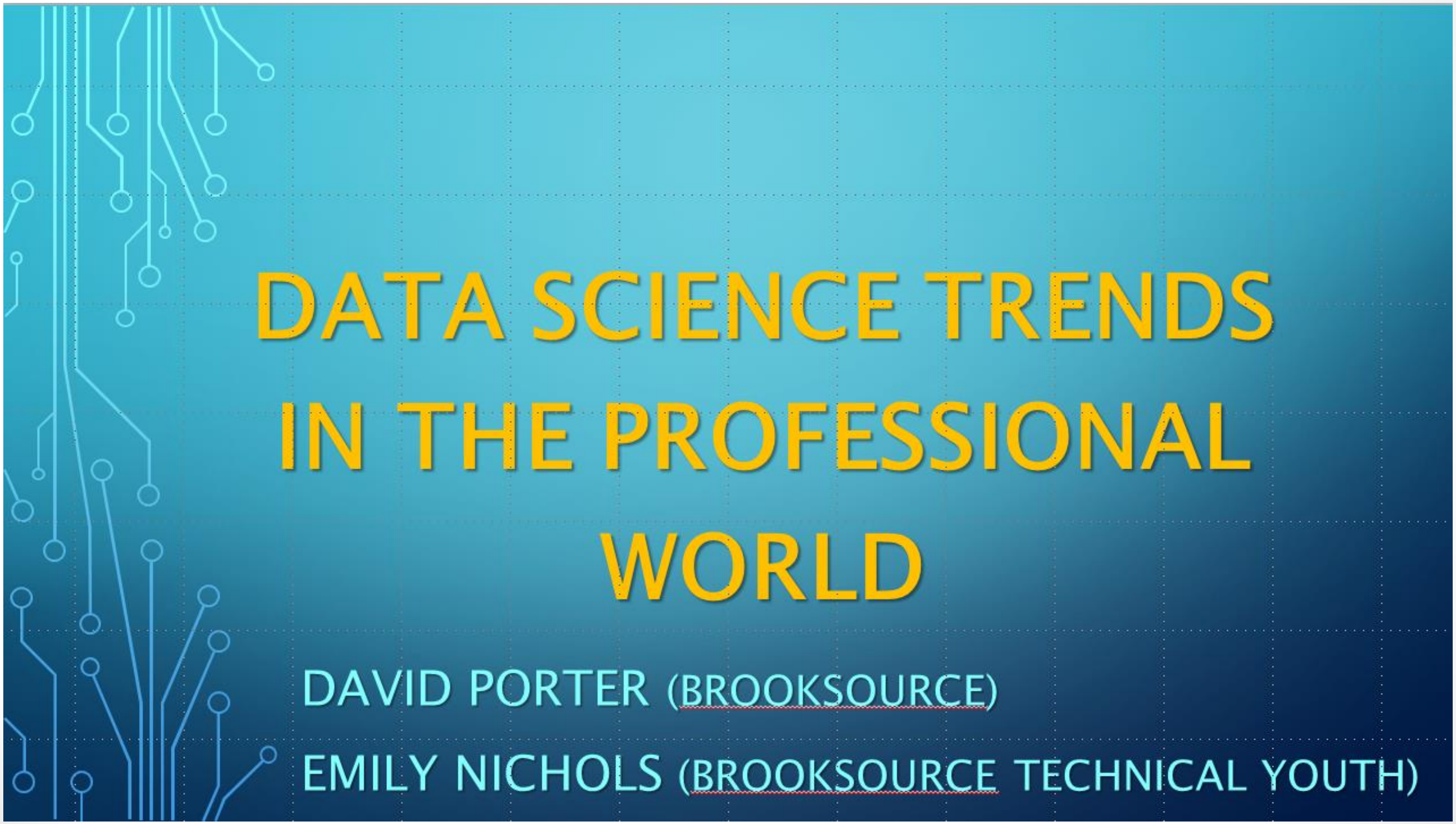
Lecture 3; October 19th, 2016

Ernst Henle
ErnstHe@UW.edu
Skype: ernst-henle

# Agenda

- Announcements
  - The social component is a course requirement:
    - On LinkedIn, start a discussion, make a comment on an existing discussion, or ask questions about homework.
    - Please  collaborate on homework!
  - For all assignments:  Label answers with the pertinent question number and letter (e.g. 3b)
  - I have been requested to say:  UW has a strict policy of removing students from the program on the grounds of plagiarism
- Data Science Trends in the Professional World by David Porter and Emily Nichols
- Review
  - Homework review of K-Means Normalization
- Data and Models in Supervised Learning
- Schema for Classification
- Partition Modeling Data
- Break
- Partition Modeling Data (In-class and Homework Assignment)
- Predictive Analytics Iteration Trap (Time Permitting)
- Assignment. See assignment slides at the end of each deck. (Complete all assignments items from all assignment slides.  Submit by Saturday 11:57 PM)

# Guest Lecture



# Break

- **Big Data Humor: Top 10 Ways You Know You're a Data Scientist**
  - http://inside-bigdata.com/2013/10/28/big-data-humor-top-10-ways-know-youre-data-scientist/

# K-means Review

- See KmeansNorm_completed.R

# K-means Review

- Some Points:
  - Normalizations are important to put data on equal terms
  - Initial centroid number and placement is an art.
  - Categorical Data must be binarized
  - K-means is unsupervised because we do not tell the algorithm what outcome was observed or what outcome is desired.

# Review: Normalization in K-Means

- See the posted KMeansNorm_complete.R
  - Normalize Observations
  - Normalize Initial Cluster Centers
  - Use K-Means to find cluster centers of normalized observations
  - De-normalize the cluster centers

# Review: Normalization in K-Means

- Z-Normalization applies standard deviation and mean of the observations to the observations and to the initial cluster centers

```
# Determine mean and standard deviation of 1st dimension in observations
mu1 <- mean(observations[,1])
sigma1 <- sd(observations[,1])
# normalize 1st dimension of observations
observations[,1] <- (observations[,1] - mu1)/sigma1
# normalize 1st dimension of clusterCenters
clusterCenters[,1] <- (clusterCenters[,1] - mu1)/sigma1
```

# Review: De-Normalization in K-Means

- De-normalization uses the same standard deviation and mean as normalization.

- # denormalize in first dimension
- clusterCenters[,1] <- clusterCenters[,1] * sigma1 + mu1

# Review: Homework Questions

- Why is normalization important in K-means clustering?
  **Answer: So that the dimensions (data attributes) have similar scales.**
- How do you encode categorical data in a K-means clustering?
  **Answer: Category attributes are binarized**
- Why is clustering un-supervised learning as opposed to supervised learning?
  **Answer: The algorithm is not told what is observed or what the "goal" is. There is no expert label.**

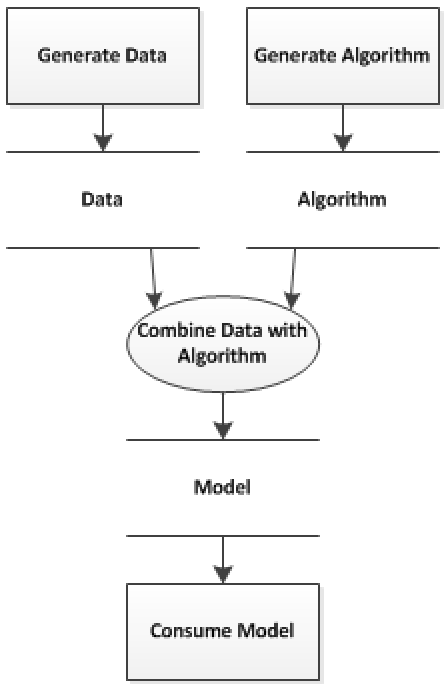
# Quiz:  K-Means Normalization

- Use R if you want (I would not)

# Data and Models in Supervised Learning

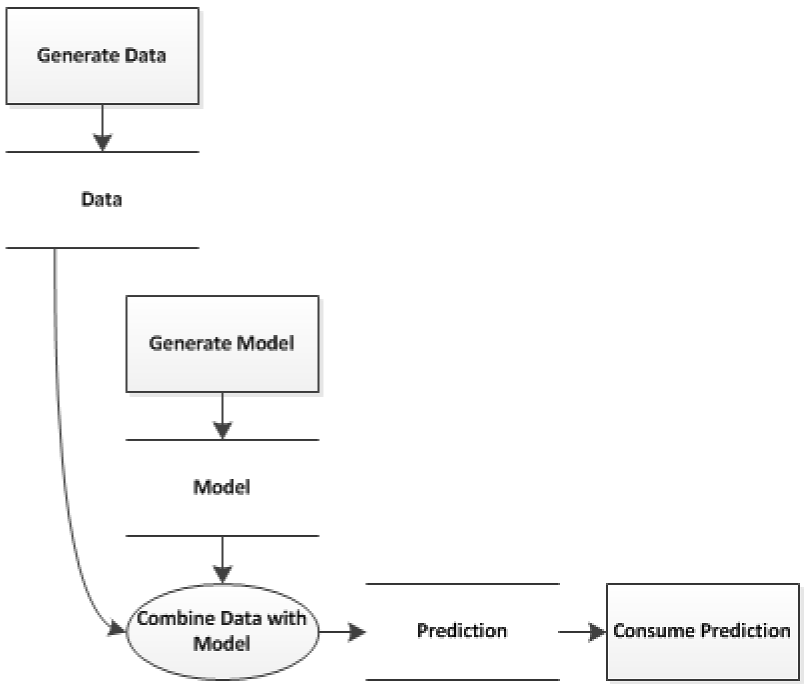
# From Data to Predictions (0)

# From Data to Predictions (1)



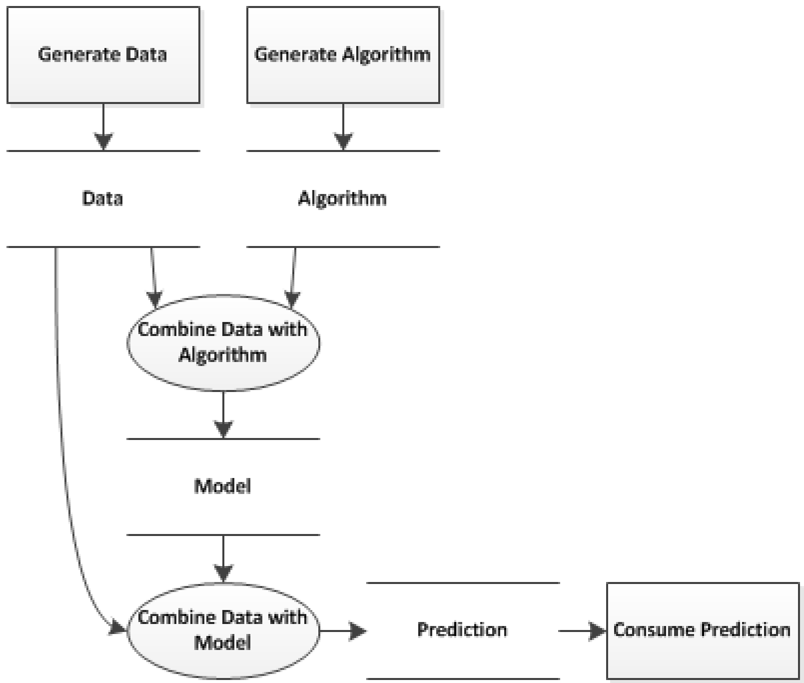

Data + Algorithm → Model

# From Data to Predictions (2)




Model + Data → Prediction

# From Data to Predictions (3)




Data + Algorithm → Model
Model + Data → Prediction

# From Data to Predictions (4)

- Pseudo Assignments (Derivations):
  - Data + Algorithm → Model
  - Model + Data → Prediction

- Create Model from Algorithm and Data
  - Example Algorithm: Logistic Regression
    - Create Model: model <- glm(formula, data=trainSet, family="binomial")
- Predict from Model and Data
    - Predict: prediction <- predict(model, newdata=testSet, type="response")



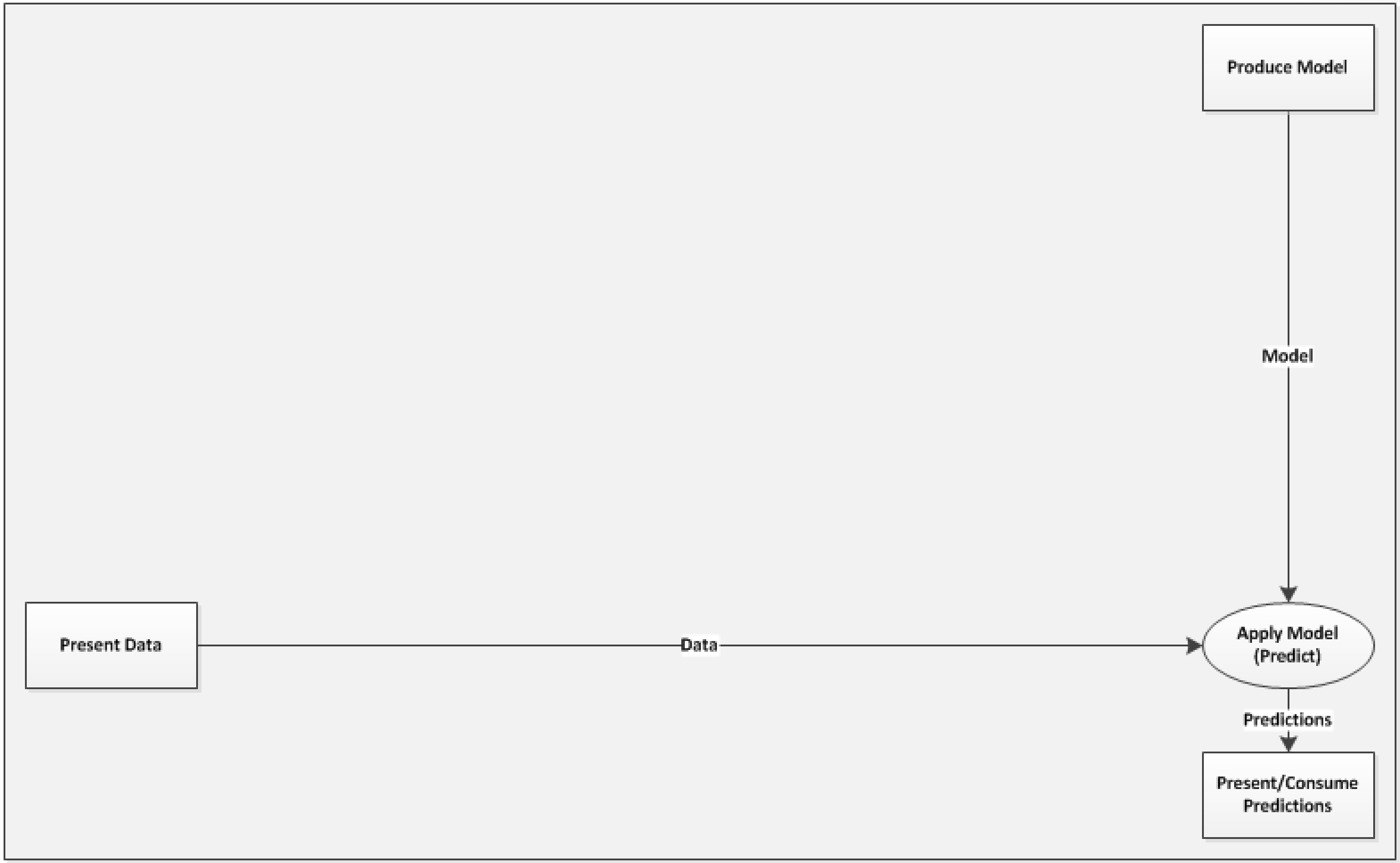
Data + Algorithm → Model
Model + Data → Prediction

# From Data to Predictions (5) Review

- A model or hypothesis is (best response)
  - a combination of test data and training data
  - a predictor based on data and algorithm
  - a falsification of a theory
  - a verified theory as long as the model was not falsified
- A model applied to new data leads to a (best response)
  - Prediction
  - Falsification / Verification
  - Hypothesis
  - errors
- A model applied to test data leads to a (best response)
  - Prediction
  - Falsification / Verification
  - Hypothesis
  - errors
- A hypothesis that cannot be tested
  - is a law if the data are consistent
  - is an untested hypothesis
  - is not a hypothesis
  - is a theory

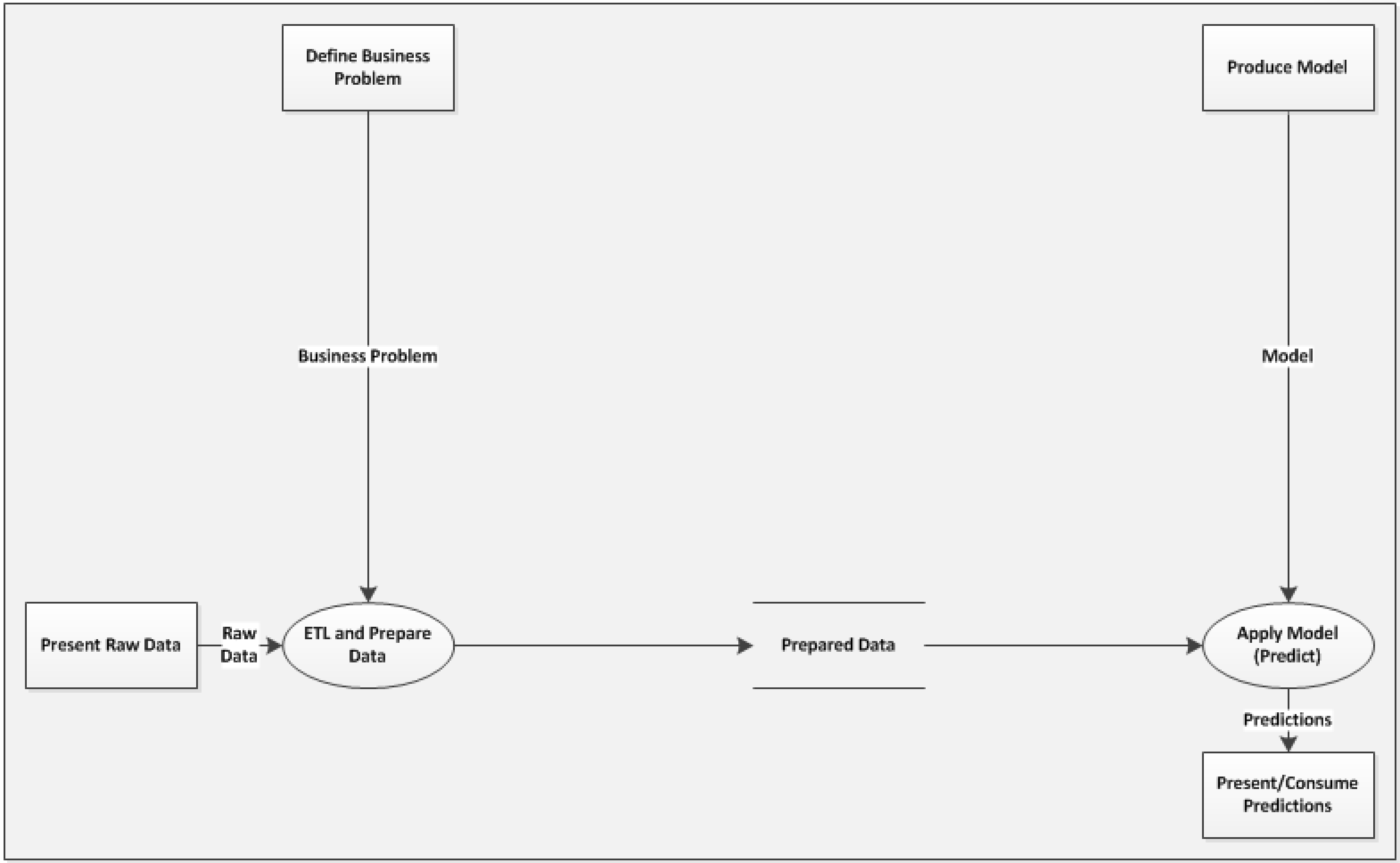
# (0) DFD of Supervised Learning

# (1) Model Acts on Data



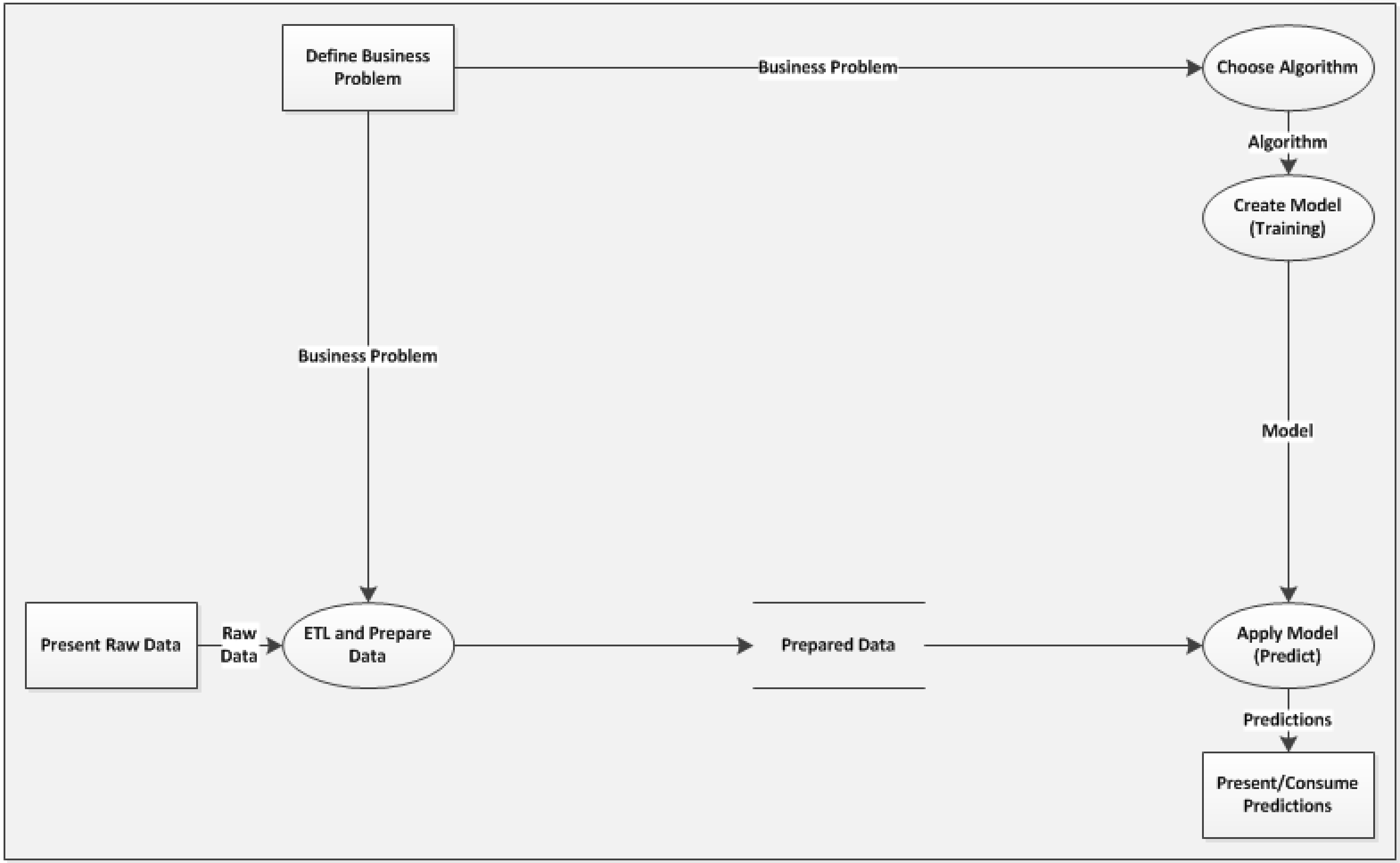

Model + Data → Prediction

# (2) Data ETL and Preparation driven by Business Problem



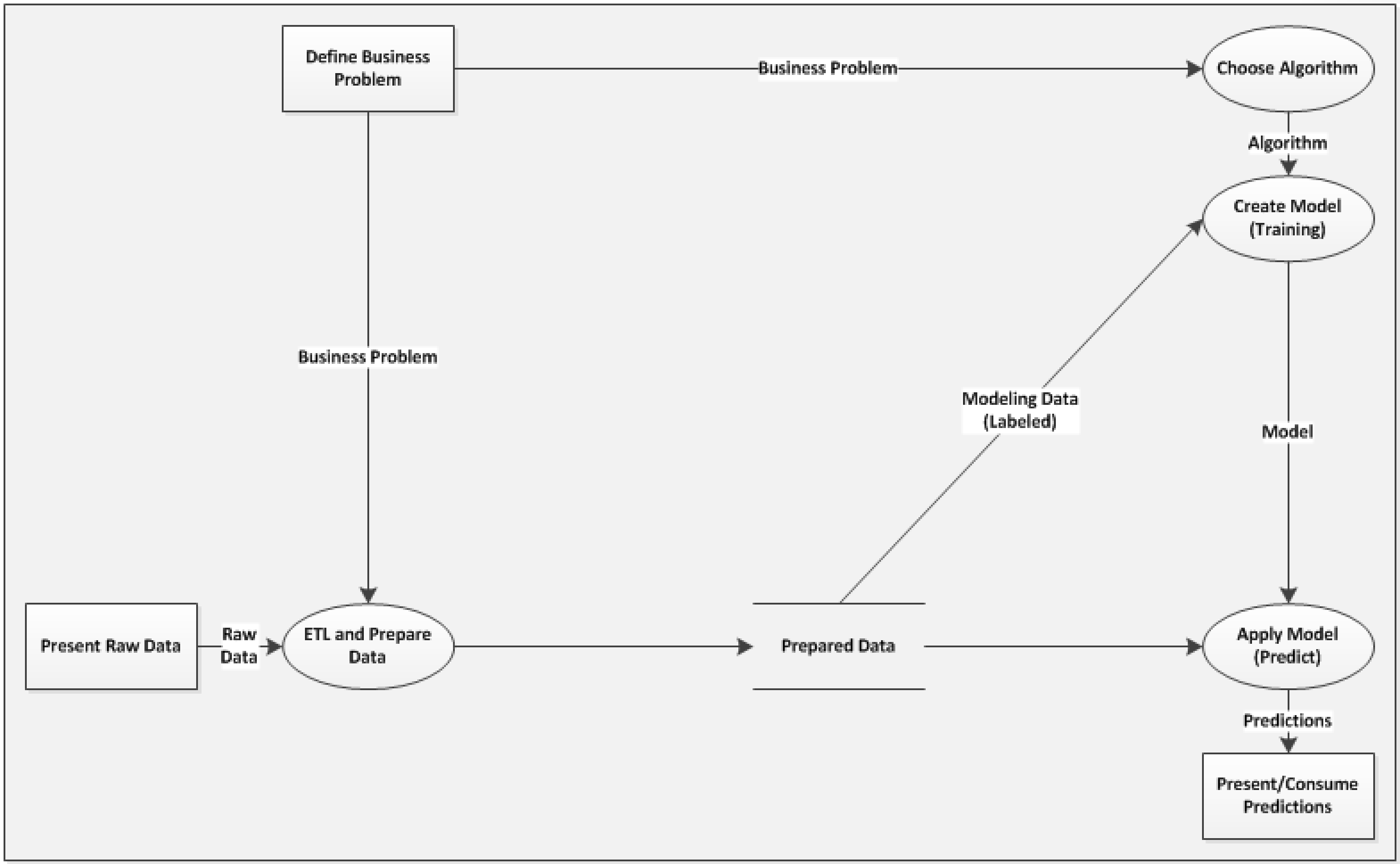

Business Problem determines ETL and Data Prep

# (3) Algorithm choice driven by Business Problem



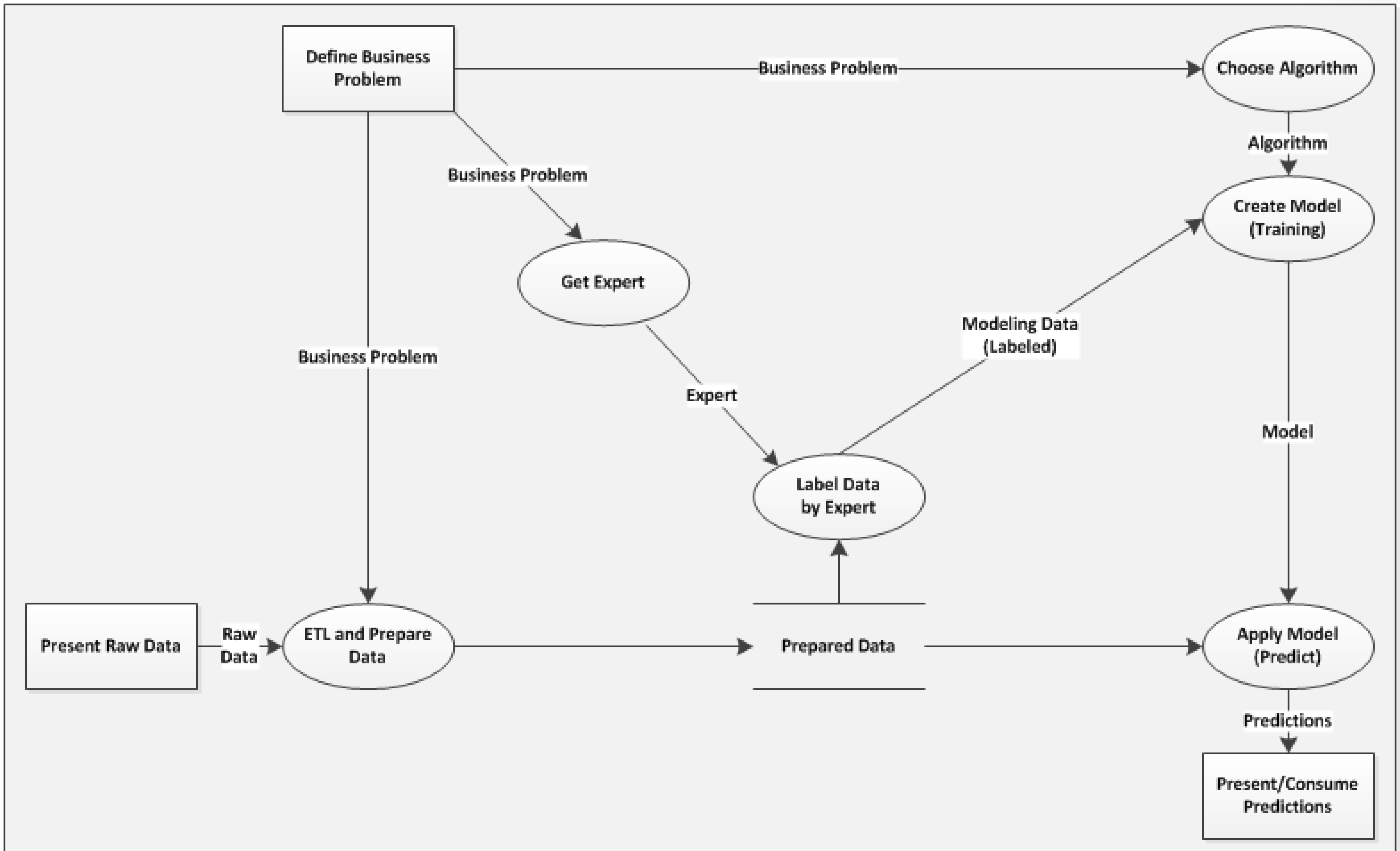

Business Problem determines the choice of Algorithm.

# (4) Model Creation needs Data



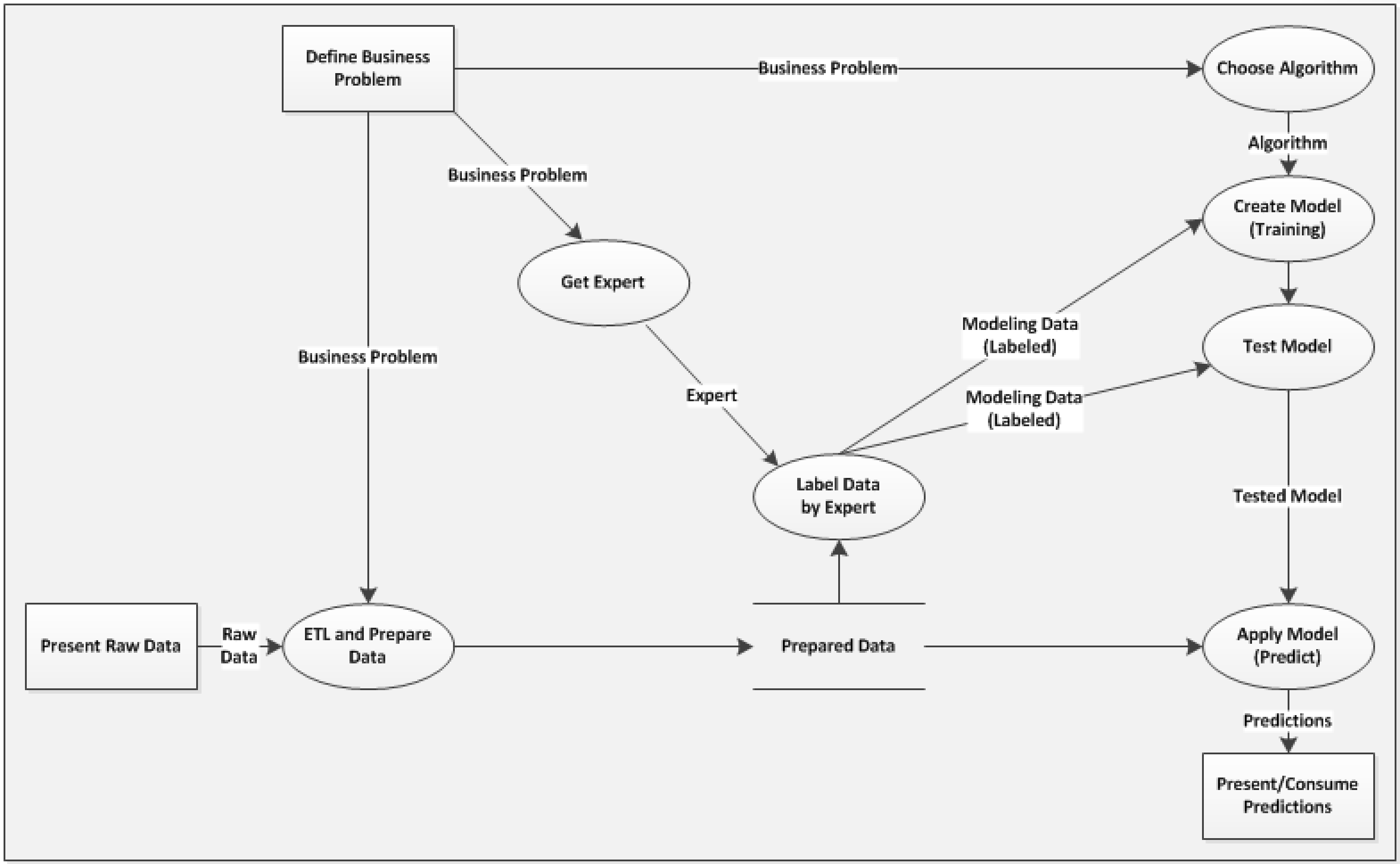

Data + Algorithm → Model

# (5) Supervised Training needs Data Labeled with Outcomes



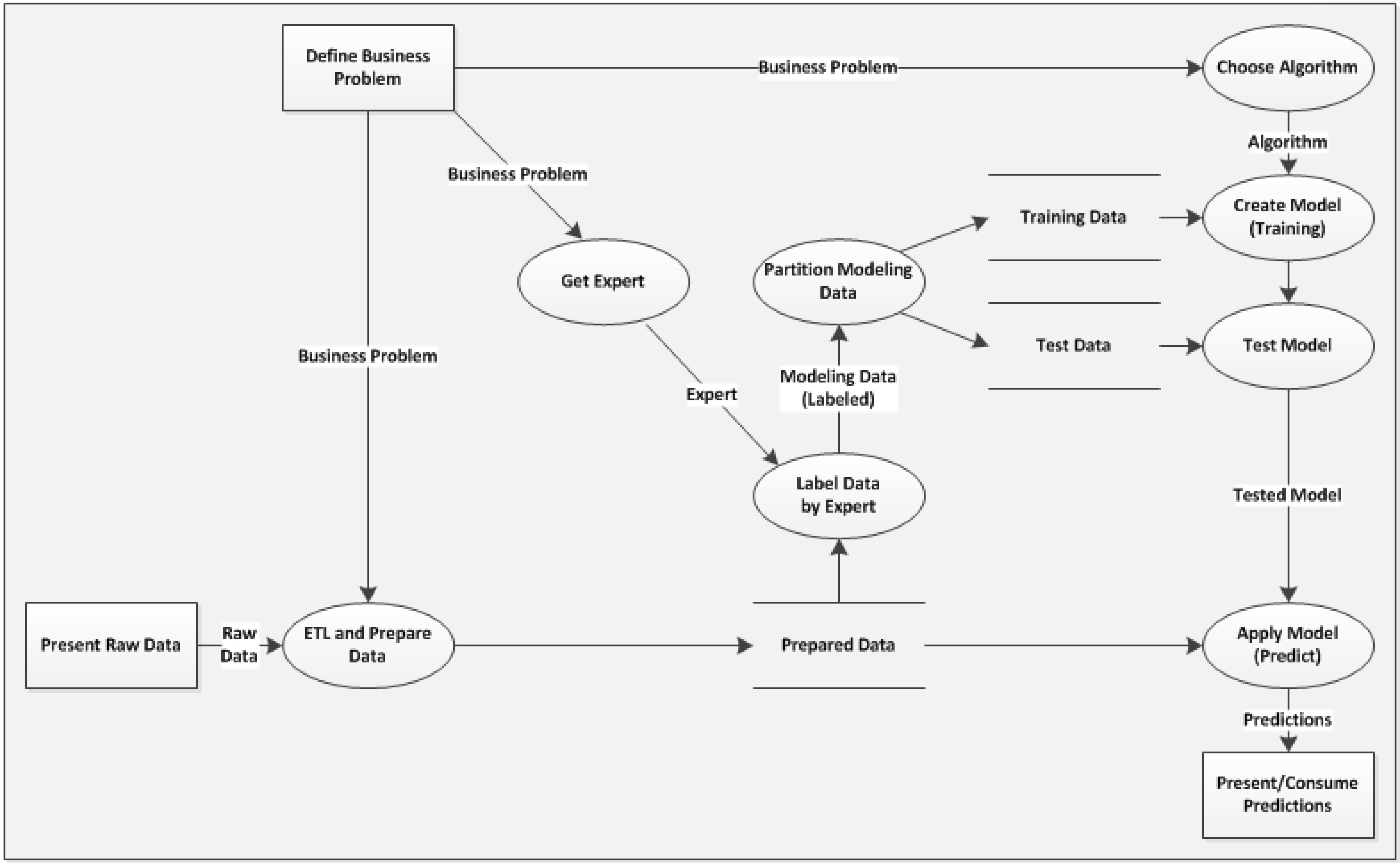

Supervised Learning requires expert labeling of data.

# (6) Models need to be Tested




Do not trust predictions from an un-tested model!

# (8) Training & Testing of Model use different Data




Do not test a model using training data!

# Data and Models in Supervised Learning

# Classification Schema

# Classification Schema (0)

- Rectangular Modeling Dataset
  - Schema
    - Input columns
    - Output column (target column, outcome)
      - Classification:  Category Column
  - Partition For Training and Test Data
  - Incremental Data
- Algorithm
  - Classification
    - Logistic Regression
    - Neural Network
    - Decision Tree
    - Naïve Bayes

# Classification Schema (1)

| Column 1 | Column 2 | Column 3 | Column 4 | Column 5 | Column 6 | Column 7 |
|---|---|---|---|---|---|---|
| 330-3141 | Seaborg | Good | 0.123 | red | T | Yes |
| 330-3150 | Seaborg | No | 0.987 | green | T | No |
| 330-3202 | Seaborg | Yes | 0.245 | blue | F | Yes |
| 415-2008 | Seaborg | Yes | 0.254 | blue | T | Yes |
| 415-2081 | Seaborg | Bad | 0.244 | blue | F | No |
| 415-2796 | Seaborg | | 0.415 | green | F | Maybe |
| 415-2799 | Seaborg | Yes | 0.925 | red | T | Yes |
| 415-2913 | Seaborg | Yes | 0.376 | green | F | Yes |
| 415-3659 | Seaborg | Bad | 0.615 | green | T | No |
| 595-8413 | Seaborg | | 0.321 | blue | F | Maybe |
| 598-1243 | Seaborg | No | 0.098 | green | F | No |
| 598-2454 | Seaborg | Bad | 0.765 | red | T | No |

# Classification Schema (2)

| Column 1 | Column 2 | Column 3 | Column 4 | Column 5 | Column 6 | Column 7 |
|---|---|---|---|---|---|---|
| 330-3141 | Seaborg | Good | 0.123 | red | T | Yes |
| 330-3150 | Seaborg | No | 0.987 | green | T | No |
| 330-3202 | Seaborg | Yes | 0.245 | blue | F | Yes |
| 415-2008 | Seaborg | Yes | 0.254 | blue | T | Yes |
| 415-2081 | Seaborg | Bad | 0.244 | blue | F | No |
| 415-2796 | Seaborg | | 0.415 | green | F | Maybe |
| 415-2799 | Seaborg | Yes | 0.925 | red | T | Yes |
| 415-2913 | Seaborg | Yes | 0.376 | green | F | Yes |
| 415-3659 | Seaborg | Bad | 0.615 | green | T | No |
| 595-8413 | Seaborg | | 0.321 | blue | F | Maybe |
| 598-1243 | Seaborg | No | 0.098 | green | F | No |
| 598-2454 | Seaborg | Bad | 0.765 | red | T | No |

Here is a rectangular dataset. The table has columns with headers and the data in each column have the same datatype. The data have been prepared and are ready for modeling.

# Classification Schema (3)

| Column | Column | Column | Column 4 | Column 5 | Column 6 | Column 7 |
|---|---|---|---|---|---|---|
| | | | 0.123 | red | T | Yes |
| | | | 0.987 | green | T | No |
| | | | 0.245 | blue | F | Yes |
| | | | 0.254 | blue | T | Yes |
| | | | 0.244 | blue | F | No |
| | | | 0.415 | green | F | Maybe |
| | | | 0.925 | red | T | Yes |
| | | | 0.376 | green | F | Yes |
| | | | 0.615 | green | T | No |
| 595-8413 | Seaborg | | 0.321 | blue | F | Maybe |
| 598-1243 | Seaborg | No | 0.098 | green | F | No |
| 598-2454 | Seaborg | Bad | 0.765 | red | T | No |

**Elsewhere, I have new data that do not contain this column. I want to predict categorical values, like these, from this new data. For each row in the new data, I want to use the values from the other columns in the same row to predict the value in the missing column. This predicted value is called the "Target Outcome".**

**Target Outcome**

# Classification Schema (4)

| Column 1 | Column 2 | Column 3 | Column 4 | Column 5 | Column 6 | Column 7 |
|---|---|---|---|---|---|---|
| 330-3141 | Seaborg | Good | 0.123 | red | T | Yes |
| 330-3150 | Seaborg | No | 0.987 | green | T | No |
| 330-3202 | Seaborg | Yes | 0.245 | blue | F | Yes |
| 415-2008 | Seaborg | Yes | 0.254 | blue | T | Yes |
| 415-2081 | Seaborg | Bad | 0.244 | blue | F | No |
| 415-2796 | Seaborg | | 0.415 | green | F | Maybe |
| 415-2799 | Seaborg | Yes | 0.925 | red | T | Yes |
| 415-2913 | Seaborg | Yes | 0.376 | green | F | Yes |
| 415-3659 | Seaborg | Bad | 0.615 | green | T | No |
| 595-8413 | Seaborg | | 0.321 | blue | F | Maybe |
| 598-1243 | Seaborg | No | 0.098 | green | F | No |
| 598-2454 | Seaborg | Bad | 0.765 | red | T | No |

Target Outcome

# Classification Schema (5)

| Column 1 | Column 2 | Column 3 | Column | Column | Column | Column |
|---|---|---|---|---|---|---|
| 330-3141 | Seaborg | Good | 0 | | | |
| 330-3150 | Seaborg | No | 0 | | | |
| 330-3202 | Seaborg | Yes | 0 | | | |
| 415-2008 | Seaborg | Yes | 0 | | | |
| 415-2081 | Seaborg | Bad | 0 | | | |
| 415-2796 | Seaborg | | 0 | | | |
| 415-2799 | Seaborg | Yes | 0 | | | |
| 415-2913 | Seaborg | Yes | 0 | | | |
| 415-3659 | Seaborg | Bad | 0 | | | |
| 595-8413 | Seaborg | | 0.321 | blue | F | Maybe |
| 598-1243 | Seaborg | No | 0.098 | green | F | No |
| 598-2454 | Seaborg | Bad | 0.765 | red | T | No |

Keys and random data should not be used as inputs for predictive analytics. Random data may appear to have patterns, but those patterns are fortuitous and will not be available when needed for predictions. Keys may contain patterns, but these patterns are deceptive and may also not be available when needed.

Random or Keys

Target Outcome

# Classification Schema (6)

| Column 1 | Column 2 | Column 3 | Column 4 | Column 5 | Column 6 | Column 7 |
|---|---|---|---|---|---|---|
| 330-3141 | Seaborg | Good | 0.123 | red | T | Yes |
| 330-3150 | Seaborg | No | 0.987 | green | T | No |
| 330-3202 | Seaborg | Yes | 0.245 | blue | F | Yes |
| 415-2008 | Seaborg | Yes | 0.254 | blue | T | Yes |
| 415-2081 | Seaborg | Bad | 0.244 | blue | F | No |
| 415-2796 | Seaborg | | 0.415 | green | F | Maybe |
| 415-2799 | Seaborg | Yes | 0.925 | red | T | Yes |
| 415-2913 | Seaborg | Yes | 0.376 | green | F | Yes |
| 415-3659 | Seaborg | Bad | 0.615 | green | T | No |
| 595-8413 | Seaborg | | 0.321 | blue | F | Maybe |
| 598-1243 | Seaborg | No | 0.098 | green | F | No |
| 598-2454 | Seaborg | Bad | 0.765 | red | T | No |

Random or Keys

Target Outcome

# Classification Schema (7)

| Column 1 | Column 2 | Column 3 | Column | Column | Column | Column |
|---|---|---|---|---|---|---|
| 330-3141 | Seaborg | Good | | | | |
| 330-3150 | Seaborg | No | | | | |
| 330-3202 | Seaborg | Yes | | | | |
| 415-2008 | Seaborg | Yes | | | | |
| 415-2081 | Seaborg | Bad | | | | |
| 415-2796 | Seaborg | | | | | |
| 415-2799 | Seaborg | Yes | | | | |
| 415-2913 | Seaborg | Yes | | | | |
| 415-3659 | Seaborg | Bad | | | | |
| 595-8413 | Seaborg | | 0.321 | blue | F | Maybe |
| 598-1243 | Seaborg | No | 0.098 | green | F | No |
| 598-2454 | Seaborg | Bad | 0.765 | red | T | No |

Columns with constant data are unnecessary. In general, they will not affect the algorithm and therefore the model will be the same. But, they distract from the task. Also, they may increase memory and processing requirements.

Random or Keys

Constant

Target Outcome

# Classification Schema (8)

| Column 1 | Column 2 | Column 3 | Column 4 | Column 5 | Column 6 | Column 7 |
|---|---|---|---|---|---|---|
| 330-3141 | Seaborg | Good | 0.123 | red | T | Yes |
| 330-3150 | Seaborg | No | 0.987 | green | T | No |
| 330-3202 | Seaborg | Yes | 0.245 | blue | F | Yes |
| 415-2008 | Seaborg | Yes | 0.254 | blue | T | Yes |
| 415-2081 | Seaborg | Bad | 0.244 | blue | F | No |
| 415-2796 | Seaborg | | 0.415 | green | F | Maybe |
| 415-2799 | Seaborg | Yes | 0.925 | red | T | Yes |
| 415-2913 | Seaborg | Yes | 0.376 | green | F | Yes |
| 415-3659 | Seaborg | Bad | 0.615 | green | T | No |
| 595-8413 | Seaborg | | 0.321 | blue | F | Maybe |
| 598-1243 | Seaborg | No | 0.098 | green | F | No |
| 598-2454 | Seaborg | Bad | 0.765 | red | T | No |

Random or Keys

Constant

Target Outcome

# Classification Schema (9)

| Column 1 | Column 2 | Column 3 | Column | Column | Column | Column |
|---|---|---|---|---|---|---|
| 330-3141 | Seaborg | Good | 0 | | | |
| 330-3150 | Seaborg | No | 0 | | | |
| 330-3202 | Seaborg | Yes | 0 | | | |
| 415-2008 | Seaborg | Yes | 0 | | | |
| 415-2081 | Seaborg | Bad | 0 | | | |
| 415-2796 | Seaborg | | 0 | | | |
| 415-2799 | Seaborg | Yes | 0 | | | |
| 415-2913 | Seaborg | Yes | 0 | | | |
| 415-3659 | Seaborg | Bad | 0 | | | |
| 595-8413 | Seaborg | | 0.321 | blue | F | Maybe |
| 598-1243 | Seaborg | No | 0.098 | green | F | No |
| 598-2454 | Seaborg | Bad | 0.765 | red | T | No |

A proxy column is a column that was created after the "target" was observed. The proxy contains information that would not be available for predictions. The proxy column correlates well with the target .

Random or Keys

Constant

Proxy

Target Outcome

# Classification Schema (10)

| Column 1 | Column 2 | Column 3 | Column 4 | Column 5 | Column 6 | Column 7 |
|---|---|---|---|---|---|---|
| 330-3141 | Seaborg | Good | 0.123 | red | T | Yes |
| 330-3150 | Seaborg | No | 0.987 | green | T | No |
| 330-3202 | Seaborg | Yes | 0.245 | blue | F | Yes |
| 415-2008 | Seaborg | Yes | 0.254 | blue | T | Yes |
| 415-2081 | Seaborg | Bad | 0.244 | blue | F | No |
| 415-2796 | Seaborg | | 0.415 | green | F | Maybe |
| 415-2799 | Seaborg | Yes | 0.925 | red | T | Yes |
| 415-2913 | Seaborg | Yes | 0.376 | green | F | Yes |
| 415-3659 | Seaborg | Bad | 0.615 | green | T | No |
| 595-8413 | Seaborg | | 0.321 | blue | F | Maybe |
| 598-1243 | Seaborg | No | 0.098 | green | F | No |
| 598-2454 | Seaborg | Bad | 0.765 | red | T | No |

Random or Keys

Constant

Proxy

Target Outcome

# Classification Schema (11)

| Column | Column | Column | Column 4 | Column 5 | Column 6 | Column 7 |
|---|---|---|---|---|---|---|
| | | | 0.123 | red | T | Yes |
| | | | 0.987 | green | T | No |
| | | | 0.245 | blue | F | Yes |
| | | | 0.254 | blue | T | Yes |
| | | | 0.244 | blue | F | No |
| | | | 0.415 | green | F | Maybe |
| | | | 0.925 | red | T | Yes |
| | | | 0.376 | green | F | Yes |
| | | | 0.615 | green | T | No |
| 595-8413 | Seaborg | | 0.321 | blue | F | Maybe |
| 598-1243 | Seaborg | No | 0.098 | green | F | No |
| 598-2454 | Seaborg | Bad | 0.765 | red | T | No |

Some inputs to supervised learning are continuous attributes, like integers, floats and time.

Some inputs to supervised learning are categories, like strings, binned numbers, and factors.

Some inputs to supervised learning are binary attributes, like categories with only two states and binarized multi-state categories.

Random or Keys

Constant

Proxy

Continuous Input

Categorical Input

Binary Input

Target Outcome

# Classification Schema (12)

| Column 1 | Column 2 | Column 3 | Column 4 | Column 5 | Column 6 | Column 7 |
|---|---|---|---|---|---|---|
| 330-3141 | Seaborg | Good | 0.123 | red | T | Yes |
| 330-3150 | Seaborg | No | 0.987 | green | T | No |
| 330-3202 | Seaborg | Yes | 0.245 | blue | F | Yes |
| 415-2008 | Seaborg | Yes | 0.254 | blue | T | Yes |
| 415-2081 | Seaborg | Bad | 0.244 | blue | F | No |
| 415-2796 | Seaborg | | 0.415 | green | F | Maybe |
| 415-2799 | Seaborg | Yes | 0.925 | red | T | Yes |
| 415-2913 | Seaborg | Yes | 0.376 | green | F | Yes |
| 415-3659 | Seaborg | Bad | 0.615 | green | T | No |
| 595-8413 | Seaborg | | 0.321 | blue | F | Maybe |
| 598-1243 | Seaborg | No | 0.098 | green | F | No |
| 598-2454 | Seaborg | Bad | 0.765 | red | T | No |

Random or Keys

Constant

Proxy

Continuous Input

Categorical Input

Binary Input

Target Outcome

# Classification Schema (13)

[redacted]

| Column 4 | Column 5 | Column 6 | Column 7 |
|---|---|---|---|
| 0.123 | red | T | Yes |
| 0.987 | green | T | No |
| 0.245 | blue | F | Yes |
| 0.254 | blue | T | Yes |
| 0.244 | blue | F | No |
| 0.415 | green | F | Maybe |
| 0.925 | red | T | Yes |
| 0.376 | green | F | Yes |
| 0.615 | green | T | No |
| 0.321 | blue | F | Maybe |
| 0.098 | green | F | No |
| 0.765 | red | T | No |

Continuous Input

Categorical Input

Binary Input

Target Outcome

# Classification Schema (14)

| Column 4 | Column 5 | Column 6 | Column 7 |
|---|---|---|---|
| 0.123 | red | T | Yes |
| 0.987 | green | T | No |
| 0.245 | blue | F | Yes |
| 0.254 | blue | T | Yes |
| 0.244 | blue | F | No |
| 0.415 | green | F | Maybe |
| 0.925 | red | T | Yes |
| 0.376 | green | F | Yes |
| 0.615 | green | T | No |
| 0.321 | blue | F | Maybe |
| 0.098 | green | F | No |
| 0.765 | red | T | No |

Continuous Input

Categorical Input

Binary Input

Target Outcome

# Classification Schema (15)

Outcome ~ Input 1 + Input 2 + Input 3

| Input 1 | Input 2 | Input 3 | Outcome |
|---|---|---|---|
| 0.123 | red | T | Yes |
| 0.987 | green | T | No |
| 0.245 | blue | F | Yes |
| 0.254 | blue | T | Yes |
| 0.244 | blue | F | No |
| 0.415 | green | F | Maybe |
| 0.925 | red | T | Yes |
| 0.376 | green | F | Yes |
| 0.615 | green | T | No |
| 0.321 | blue | F | Maybe |
| 0.098 | green | F | No |
| 0.765 | red | T | No |

# Classification Schema (16)

Outcome ~ Input 1 + Input 2 + Input 3

Modeling Data (300-100000 rows)

| Input 1 | Input 2 | Input 3 | Outcome |
|---|---|---|---|
| 0.123 | red | T | Yes |
| 0.987 | green | T | No |
| 0.245 | blue | F | Yes |
| 0.254 | blue | T | Yes |
| 0.244 | blue | F | No |
| 0.415 | green | F | Maybe |
| 0.925 | red | T | Yes |
| 0.376 | green | F | Yes |
| 0.615 | green | T | No |
| 0.321 | blue | F | Maybe |
| 0.098 | green | F | No |
| 0.765 | red | T | No |

# Classification Schema (17)

Outcome ~ Input 1 + Input 2 + Input 3

Training Data (200-50000 rows)

Modeling Data (300-100000 rows)

| Input 1 | Input 2 | Input 3 | Outcome |
|---|---|---|---|
| 0.123 | red | T | Yes |
| 0.987 | green | T | No |
| 0.245 | blue | F | Yes |
| 0.254 | blue | T | Yes |
| 0.244 | blue | F | No |
| 0.415 | green | F | Maybe |
| 0.925 | red | T | Yes |
| 0.376 | green | F | Yes |
| 0.615 | green | T | No |
| 0.321 | blue | F | Maybe |
| 0.098 | green | F | No |
| 0.765 | red | T | No |

# Classification Schema (18)

Outcome ~ Input 1 + Input 2 + Input 3

Training Data
(200-50000 rows)

Test Data
(100-50000 rows)

Modeling Data
(300-100000 rows)

| Input 1 | Input 2 | Input 3 | Outcome |
|---|---|---|---|
| 0.123 | red | T | Yes |
| 0.987 | green | T | No |
| 0.245 | blue | F | Yes |
| 0.254 | blue | T | Yes |
| 0.244 | blue | F | No |
| 0.415 | green | F | Maybe |
| 0.925 | red | T | Yes |
| **0.376** | **green** | **F** | **Yes** |
| **0.615** | **green** | **T** | **No** |
| **0.321** | **blue** | **F** | **Maybe** |
| **0.098** | **green** | **F** | **No** |
| **0.765** | **red** | **T** | **No** |

# Classification Schema (19)

Outcome ~ Input 1 + Input 2 + Input 3

Train & Test

Training Data (200-50000 rows)

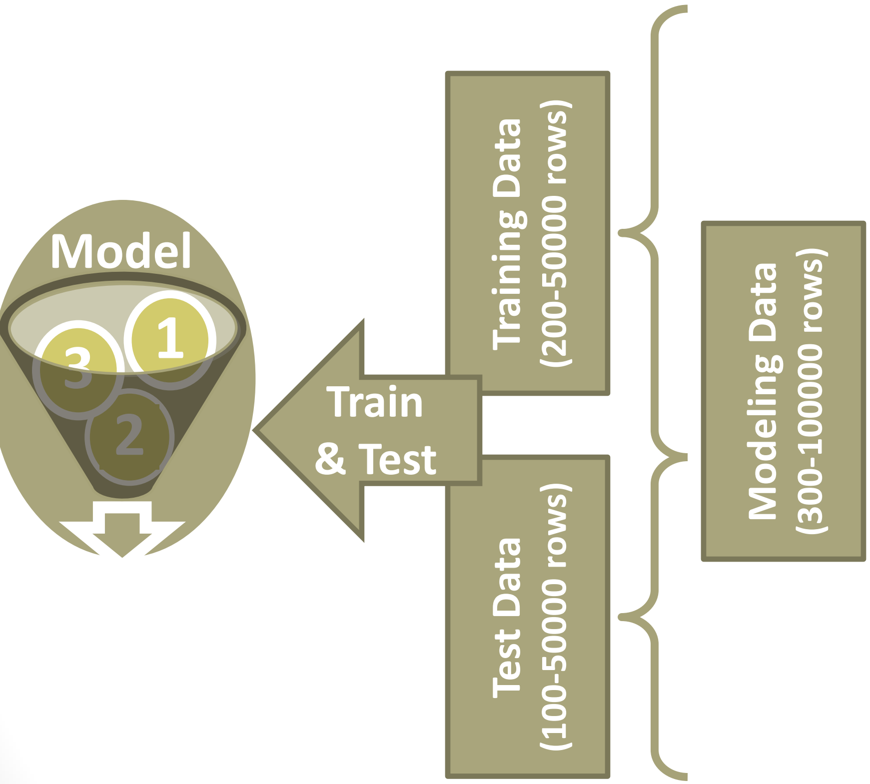
Test Data (100-50000 rows)

Modeling Data (300-100000 rows)

| Input 1 | Input 2 | Input 3 | Outcome |
|---|---|---|---|
| 0.123 | red | T | Yes |
| 0.987 | green | T | No |
| 0.245 | blue | F | Yes |
| 0.254 | blue | T | Yes |
| 0.244 | blue | F | No |
| 0.415 | green | F | Maybe |
| 0.925 | red | T | Yes |
| 0.376 | green | F | Yes |
| 0.615 | green | T | No |
| 0.321 | blue | F | Maybe |
| 0.098 | green | F | No |
| 0.765 | red | T | No |

# Classification Schema (20)

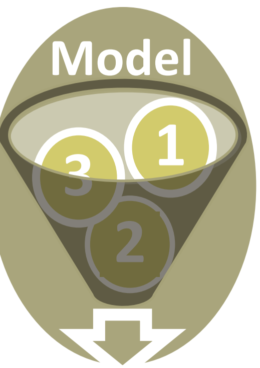
Outcome ~ Input 1 + Input 2 + Input 3




| Input 1 | Input 2 | Input 3 | Outcome |
|---|---|---|---|
| 0.123 | red | T | Yes |
| 0.987 | green | T | No |
| 0.245 | blue | F | Yes |
| 0.254 | blue | T | Yes |
| 0.244 | blue | F | No |
| 0.415 | green | F | Maybe |
| 0.925 | red | T | Yes |
| 0.376 | green | F | Yes |
| 0.615 | green | T | No |
| 0.321 | blue | F | Maybe |
| 0.098 | green | F | No |
| 0.765 | red | T | No |

# Classification Schema (21)

Outcome ~ Input 1 + Input 2 + Input 3




**Elsewhere, I have new data that do not contain the target outcome. I want to predict categorical values, like these, from this new data. For each row in the new data, I want to use the values from the other columns in the same row to predict the value in the missing column. This predicted value is called the "Target Outcome".**

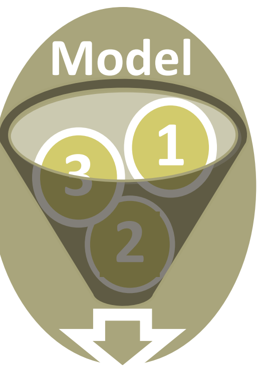
| Input 1 | Input 2 | Input 3 | Outcome |
|---|---|---|---|
| 0.123 | red | T | Yes |
| 0.987 | green | T | No |
| 0.245 | blue | F | Yes |
| 0.254 | blue | T | Yes |
| 0.244 | blue | F | No |
| 0.415 | green | F | Maybe |
| 0.925 | red | T | Yes |
| 0.376 | green | F | Yes |
| 0.615 | green | T | [illegible] |
| 0.321 | blue | F | [illegible] |
| 0.098 | green | F | No |
| 0.765 | red | T | No |
| 0.234 | green | T | |
| 0.567 | blue | F | |
| 0.890 | green | T | |
| 0.314 | red | T | |

**Target Outcome**

**Operational Data** (1- ∞ rows)

# Classification Schema (22)

Outcome ~ Input 1 + Input 2 + Input 3



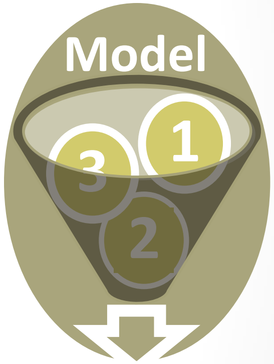

Target Outcome

Operational Data (1- ∞ rows)

| Input 1 | Input 2 | Input 3 | Outcome |
|---|---|---|---|
| 0.234 | green | T | |
| 0.567 | blue | F | |
| 0.890 | green | T | |
| 0.314 | red | T | |

# Classification Schema (23)

Outcome ~ Input 1 + Input 2 + Input 3




Operational Data (1- ∞ rows)

| Input 1 | Input 2 | Input 3 | Outcome |
| --- | --- | --- | --- |
| 0.234 | green | T | |
| 0.567 | blue | F | |
| 0.890 | green | T | |
| 0.314 | red | T | |

# Classification Schema (24)

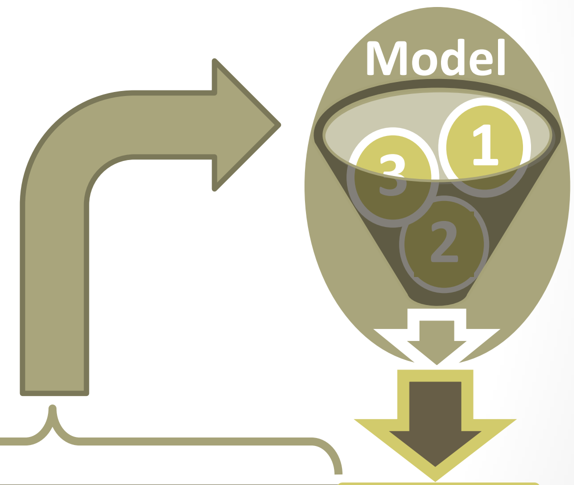
Outcome ~ Input 1 + Input 2 + Input 3


Operational Data (1- ∞ rows)

| Input 1 | Input 2 | Input 3 | Outcome |
|---|---|---|---|
| 0.234 | green | T | |
| 0.567 | blue | F | |
| 0.890 | green | T | |
| 0.314 | red | T | |

# Classification Schema (25)

Outcome ~ Input 1 + Input 2 + Input 3




Operational Data (1- ∞ rows)

| Input 1 | Input 2 | Input 3 | Outcome |
|---|---|---|---|
| 0.234 | green | T | Yes |
| 0.567 | blue | F | No |
| 0.890 | green | T | Maybe |
| 0.314 | red | T | Maybe |

# Classification Schema (26)

- Attributes
  - All the columns are attributes
- Input Column
  - Input columns are columns that can help predict the outcome.  Input columns can be of type binary, ordinal, or category.
- Target Outcome
  - The term "Target Outcome" is redundant.  The outcome is the target and vice versa.  The target or outcome is the output of a predict function.  Providing target or outcome values during modeling makes the process supervised. Creating a model using a outcome is called supervised learning.
- Proxy Column
  - A proxy column is a column that predicts too well.  It is too good to be true.  Something from the target leaked.  This is also called target leakage.  The leaked information is "not fair" to use in modeling.  Values for that attribute will not be available when you want to predict the target outcome from operational data.
- Key Column
  - In principle, a key column should not affect the model's prediction.  The relationship between a key and any other attribute should be random.  In practice, the algorithm will find a pattern in the key column and train on this pattern.  This pattern is likely to be fortuitous, that means:  random.  The pattern will not hold for test data or when the model is applied.  As a consequence, the key column will affect the model in a bad way.
- Constant Column
  - A constant column should have no affect on the model's predictions.  The constant column may increase computation time and cause other problems.  It is standard practice to remove all constant columns prior to modeling.

# Classification Schema

# Break

- Colbert on Predictive Analytics
  - http://www.cc.com/video-clips/dv9iqc/the-colbert-report-the-word---surrender-to-a-buyer-power

# Partition Modeling Data

# How to Partition Data (0)

- Test data need to be derived from the same data source as the training data
- Partition of Data between Test and Training must be random

# How to Partition Data (1)

- Test data need to be derived from the same data source as the training data
- Partition of Data between Test and Training must be random

- Open R Studio:
  1. Open ClassifyStudents.R
  2. Open CollegeStudentsDatasets.R
  3. Source ClassifyStudents.R (just a test)
  4. Find the function definition of PartitionWrong() in CollegeStudentsDatasets.R
  5. See how to use the function PartitionWrong() in ClassifyStudents.R

# How to Partition Data (2)
# The Wrong Way

1. Specify test fraction (e.g. split off 30% or 40% for testing)
2. Take the first fraction of the data as test data
3. Take the rest of the data as training data

# How to Partition Data (3) The Wrong Way

1. Get the dataSet and the fractionOfTest (fractionOfTest default is 0.3)
2. numberOfRows <- nrow(dataSet)
3. numberOfTestRows <- fractionOfTest * numberOfRows
4. testFlag <- 1:numberOfRows <= numberOfTestRows
5. testingData <- dataSet[testFlag , ]
6. trainingData <- dataSet[!testFlag , ]

# How to Partition Data (4)
# The Fast Way

1. Specify test fraction (e.g. split off 30% or 40% for testing)
2. Generate random number for each case (row)
3. Create Flag to partition cases at value of test fraction
4. Apply Flag for Test Data selection
5. Apply Flag for Train Data selection

# How to Partition Data (5) Exact Method

1. Specify test fraction (e.g. split off 30% or 40% for testing)
2. Generate a random number for each case (use “runif”)
3. Create Flag to partition cases: find quantile
   a) Determine threshold where quantile of random numbers is at fractionOfTest. Use “sort” or “quantile”
   b) Compare random numbers to that threshold. For the random numbers that are smaller than the threshold, the testFlag is TRUE.
4. Apply Flag for Testing Data selection (testingData <- dataSet[testFlag, ] )
5. Apply Flag for Training Data selection. Training data should contain all observations that are not testing data.

# How to Partition Data (6) Take Home Message

1. For supervised learning, partition the modeling data in a test set and a train set
2. Use the test data to check the accuracy of an algorithm
3. A confusion matrix and an ROC chart can be used to test classifications
4. Classification accuracy can be defined as the number of correct predictions divided by the total number of predictions.

# How to Partition Data (7) Take Home Messages

1. For supervised learning, partition the modeling data in a test set and a train set
2. Use the test data to check the accuracy of an algorithm
3. A confusion matrix and an ROC chart can be used to test classifications
4. Classification accuracy can be defined as the number of correct predictions divided by the total number of predictions.

| | 0 | 1 |
|---|---|---|
| 0 | 227 | 92 |
| 1 | 8 | 73 |

# How to Partition Data (8)
# Take Home Messages

1. For supervised learning, partition the modeling data in a test set and a train set
2. Use the test data to check the accuracy of an algorithm
3. A confusion matrix and an ROC chart can be used to test classifications
4. Classification accuracy can be defined as the number of correct predictions divided by the total number of predictions.

| | 0 | 1 |
|---|---|---|
| 0 | 227 | 92 |
| 1 | 8 | 73 |

$$75\% = (227 + 73) \div (8 + 92 + 227 + 73)$$

# Partition Modeling Data

# In-class Exercise

- Open R Studio:
  1. Open ClassifyStudents.R
  2. Open CollegeStudentsDataset.R
  3. Source ClassifyStudents.R (just a test)
  4. Find the function definition of PartitionWrong() in CollegeStudentsDataset.R and see how it is constructed.
  5. See how PartitionWrong() is used in ClassifyStudents.R
  6. Add code to create a logistic regression model.
  7. Add code to get probabilities from the logistic regression model
  8. Add code to threshold the probabilities and create a confusion matrix

# Review: Terminology

- Algorithm
- Anomaly detection
- Association
- Attribute
- Binarize Categories
- Binary Column
- Case
- Category Column
- Character Column
- Classification
- Clustering
- Coercion
- Column
- Column Header
- Data
- Data Dimensionality
- Data Frame
- Data Type
- DFD
- Dummy Variable
- Estimation
- Feature Scaling
- Field
- Hypothesis
- Key Column
- Machine Learning
- Market-basket analysis
- MATLAB
- Matrix
- Missing Data
- Model
- Multinomial Column
- Normalization
- Numeric Column
- Observation
- Outcome
- Outlier Removal
- Predictive Analytics
- R
- Rectangular Data
- Relabeling
- Row
- Schema
- Shaping Data
- Sparse Multi-Dimensional Matrix
- Standard Deviation
- States
- String
- Supervised Learning
- Support
- Table
- Target Column
- Text Column
- Theory
- Un-structured Data
- Unsupervised Learning
- Z-score

# Assignment (1)

1. Add code to ClassifyStudents.R that creates a Logistic Regression model.
2. In ClassifyStudents.R, add code to predict outcomes based on the Logistic Regression model.
3. Add code to create a confusion matrix to evaluate the logistic regression model
4. Add code to ClassifyStudents.R that creates a Naïve Bayes model. You will have to look up the Naive Bayes package "e1071" to determine the inputs. Get help!
5. In ClassifyStudents.R add code to predict outcomes based on the Naive Bayes model. You will have to read the documentation to determine the “type” parameter. **<u>It is very important that you answer for yourself: How many rows are there in the outcome? How many columns? How many columns are in the output for the logistic regression? Get Help!</u>**
6. Add code to create a confusion matrix to evaluate the naïve Bayes model

# Assignment (2)

7. Partition Functions
   a) Run ClassifyStudents.R using PartitionWrong() with a threshold=0.5 and fractionOfTest=0.4. Verify the resulting confusion matrices of the logistic regression and Naïve Bayes models. The results are already listed as comments in ClassifyStudents.R.
   b) Add a function to CollegeStudentsDataset.R. The name of the function is: PartitionFast. The function works as described in the lecture slides and has the same signature and return type as PartitionWrong. Specifically, the function takes in only a dataframe and a fraction. The function returns a list of two dataframes. The names of the two dataframes are trainingData and testingData. trainingData and testingData are mutually exclusive cases from the input data frame. trainingData, testingData , and the data frame all have the same schema. testingData contains the fraction of cases as specified by the fraction input. trainingData contains the rest.
   c) Add a function to CollegeStudentsDataset.R. The name of the function is: PartitionExact. The function works as described in the lecture slides and has the same basic structure as PartitionWrong and PartitionFast (see above).

# Assignment (3)

8. Classification in R
   a) In ClassifyStudents.R replace the line containing PartitionWrong() with PartitionFast(). Note that the probability threshold is 0.5. Run ClassifyStudents.R and copy the resulting confusion matrix of the logistic regression and naïve Bayes as a comment into ClassifyStudents.R. Add the accuracy calculations as indicated.
   b) In ClassifyStudents.R replace the line containing PartitionWrong() with PartitionExact(). Note that the probability threshold is 0.5. Run ClassifyStudents.R and copy the resulting confusion matrix of the logistic regression and naïve Bayes as a comment into ClassifyStudents.R. Add the accuracy calculations as indicated.
   c) In ClassifyStudents.R use PartitionExact() again but this time change the probability threshold to 0.7. Run ClassifyStudents.R and copy the resulting confusion matrix of the logistic regression and naïve Bayes as a comment into ClassifyStudents.R. Add the accuracy calculations as indicated.
9. Submit your revised ClassifyStudents.R and CollegeStudentsDatasets.R by Saturday 11:57 PM. Late submissions cannot be accepted.

# Assignment (4)

10. Take Quiz 03b Elbow before Tuesday 11:57 PM. You will need Elbow.R and StudentPlans.csv
11. On LinkedIn, start a discussion, make a comment on an existing discussion, or ask questions about homework.
12. Reading Assignments
    - Review terminology at the end of this slide deck
    - Read Quiz Previews (They might not be posted until tomorrow)
    - Classification
        - Read ROC Curve, Lift Chart and Calibration Plot by Vuk and Curk: http://mrvar.fdv.uni-lj.si/pub/mz/mz3.1/vuk.pdf
        - Read about Accuracy: http://en.wikipedia.org/wiki/Precision_and_recall
        - Read about target leakage: http://www.cs.umb.edu/~ding/history/470_670_fall_2011/papers/cs670_Tran_PreferredPaper_LeakingInDataMining.pdf
    - Relational Model, Relational Algebra, and Relational Calculus
        - http://en.wikipedia.org/wiki/Relational_algebra
        - http://sentences.com/docs/amd.pdf (Pages 35 to 48 only)
        - http://en.wikipedia.org/wiki/Relational_model
        - http://www.youtube.com/watch?v=NvrpuBAMddw

# Introduction to Data Science