

Introduction to Data Science

**DATA PREP FOR MODELING: FEATURE
ENGINEERING**

BRIAN D'ALESSANDRO

Fine Print: these slides are, and always will be a work in progress. The material presented herein is original, inspired, or borrowed from others' work. Where possible, attribution and acknowledgement will be made to content's original source. Do not distribute, except for as needed as a pedagogical tool in the subject of Data Science.

FROM DATA TO FEATURES

Feature Engineering is one of the most important skills to develop as a data scientist (both the ability to hypothesize what is predictive, as well as the data engineering skills to build them).

Types of Feature Engineering we'll cover

- Binning
- Non-linear transformations
- Domain knowledge feature extraction

Quick aside: One of the amazing contributions of Deep Learning is that DL does what they call “implicit feature engineering.” But as not all problems will be DL problems, as a DS you’ll still likely need to do “explicit” feature engineering for the foreseeable future.

BINNING (ONE HOT ENCODING)

Taking categorical data with K values (also called a factor) and projecting them to K-1 binary features (also called 'dummy indicators').

Data can be numeric or categorical



If there are 4 categories/ranges, we have 3 binary variables.

User	Income	IncomeGrp
1	\$65,500	[61k,80k]
2	\$81,041	[81k,100k]
3	\$38,346	[21k,40k]
4	\$47,072	[41,60k]
5	\$30,812	[21k,40k]
6	\$21,618	[21k,40k]
7	\$97,872	[81k,100k]



User	inc_41_60	inc_61_80	inc_81_100
1	0	1	0
2	0	0	1
3	0	0	0
4	1	0	0
5	0	0	0
6	0	0	0
7	0	0	1

1. Numeric data can be binned based on a pre-defined range.
2. Categorical data can be binned based on category value.
3. If you make K indicators and not K-1, your X matrix will be degenerate/singular because because the Kth indicator will be a linear combination of the K-1 previous indicators. Many algorithms don't like non-invertible matrices.

NON-LINEAR TRANSFORMATIONS

Let's say you want/have to build a linear model, so you need to express your dependency as:

$$E[Y|X] = \alpha^0 + \beta^0 * X$$

But the phenomenon/data you are modeling is actually non-linear, so to get a better fit you want:

$$E[Y|X] = \alpha^1 + \beta^1 * g(X)$$

In lieu of using more sophisticated non-linear algorithms (i.e., Random Forest, Neural Network), you can make clever feature transformations to get a non-linear fit out of a linear algorithm. This is similar to the kernel trick used in SVMs, but here we are explicitly transforming data into new features.

EXAMPLE – NON-LINEAR TRANSFORM

In this example we have a regression problem with the following model specification: $E[Y|X] = \alpha + \beta * X$

The R^2 is pretty good, but visually, we can see a potentially better fit. What transformation might fit this data better?



EXAMPLE – NON-LINEAR TRANSFORM

Lets use the transformation $g(X) = \langle X^2, X \rangle$

We can then fit the model: $E[Y|X] = \alpha + \beta^*g(X) = \alpha + \beta_1X^2 + \beta_2X$

We can often realize a non-trivial increase in performance by making such transformations prior to modeling.

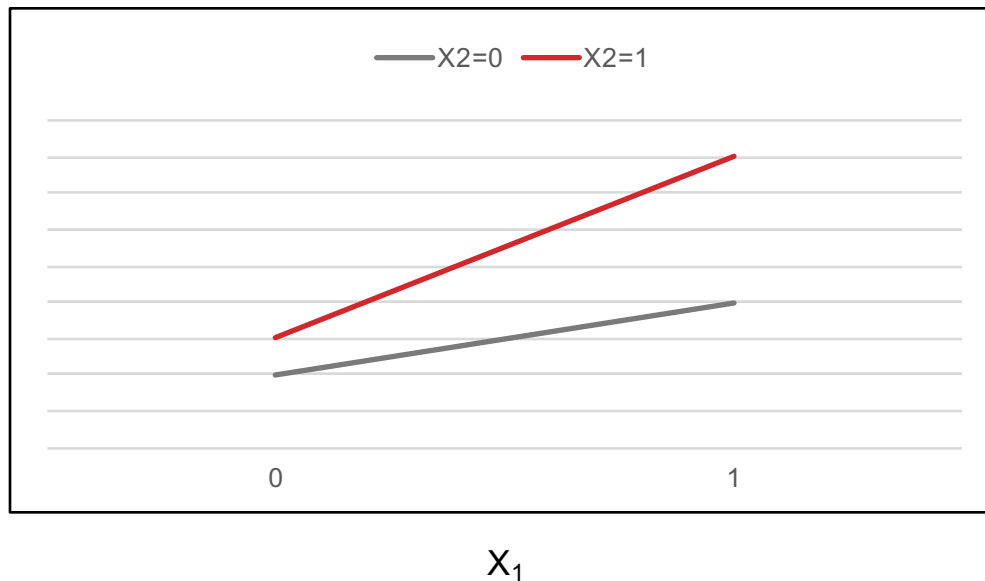


Copyright: Brian d'Alessandro, all rights reserved

INTERACTION TERMS

When $dE[Y|X_1] / dX_1$ depends on X_2 , we say X_1 and X_2 show an interaction effect. In many cases we have to construct explicit features to capture them.

Sample model: $Y = a + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1 X_2$



Logical: $X_1_X2_int = (X_1 = 1 \text{ and } X_2 = 1)$

Mathematical: $X_1_X2_int = X_1 X_2$

Practical note: If you have k features there are roughly $O(k^2)$ possible 2nd order interaction terms, making it infeasible to build and test them all. A good technique is to run your features through a DT and make interactions from the first few split variables. Or use feature importance and make interactions from the more important features.

NON-LINEAR TRANSFORMATIONS IN D-DIMENSIONS

A polynomial transformation in multiple dimensions captures both the interaction effects and the polynomial transformations of each feature.

$$X = \langle X_1, X_2, \dots, X_k \rangle$$

$$E[Y|X] = \alpha + \beta * g_d(X); g_d(X) \Rightarrow (X+1)^d$$

$$\text{e.g., } g_2(X) \Rightarrow \langle 1, X_1, X_2, X_1 * X_2, X_1^2, X_2^2 \rangle$$

When the algorithms do this for you

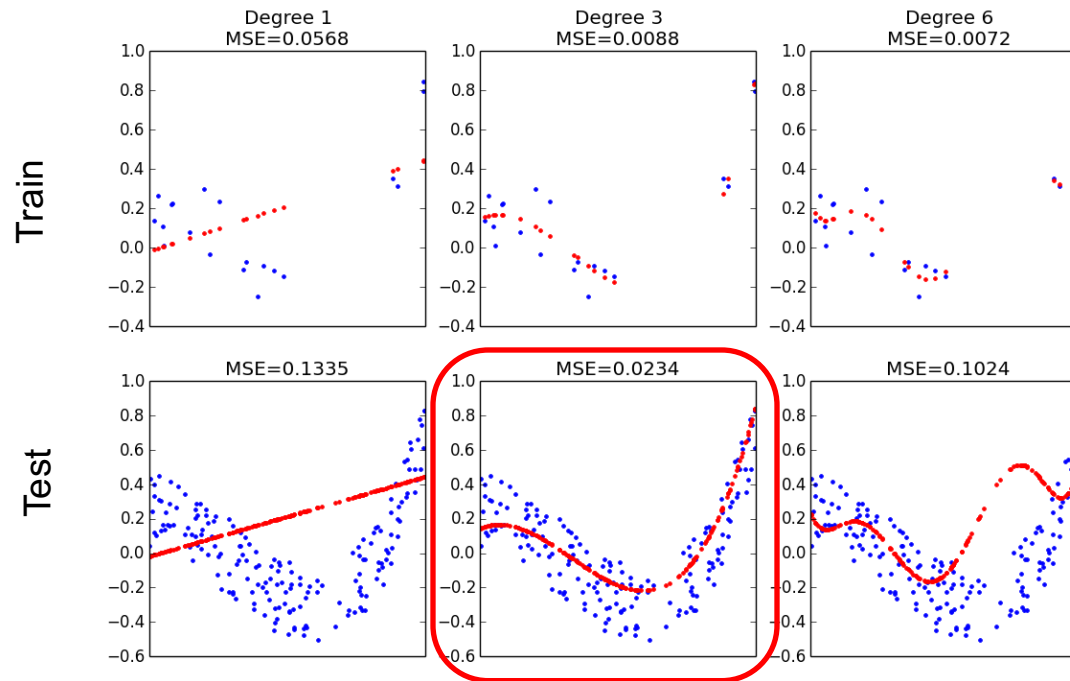
SVMs: The use of Kernels in SVM's can generate these transformations implicitly, so no new explicit feature generation would be needed. The most common kernel is the Radial Basis Function, which effectively maps the data into an infinite space

Trees: tree based algorithms can fit non-linear curves and interactions naturally by partitioning on X and estimating expectations separately for each partition (similar to binning).

In both of these cases, domain driven feature engineering is still necessary

NL TRANSFORMATION: NOISY ENVIRONMENT

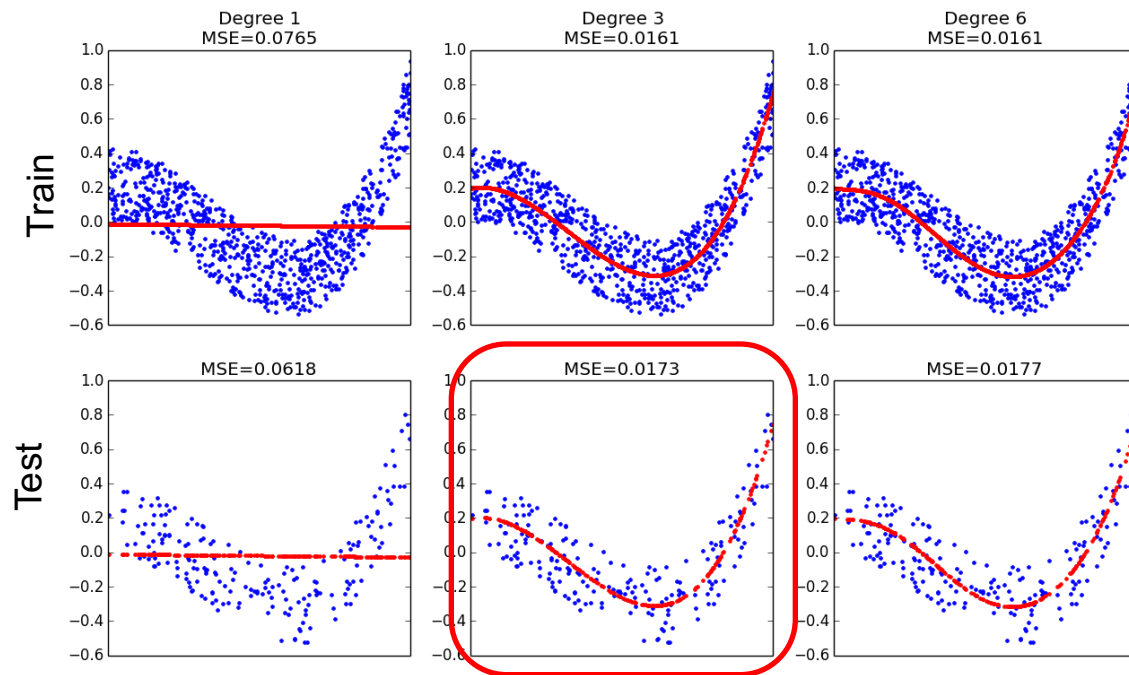
If $g(X)$ is a polynomial expansion of X (i.e., $g^d(x) = \langle X^d, \dots, X^2, X \rangle$), we can get a great fit by choosing the right degree of expansion. Too low and we get generally bad MSE everywhere...too high and we overfit.



Copyright © 2019, All rights reserved.

NL TRANSFORMATION: INFORMATION RICH ENVIRONMENT

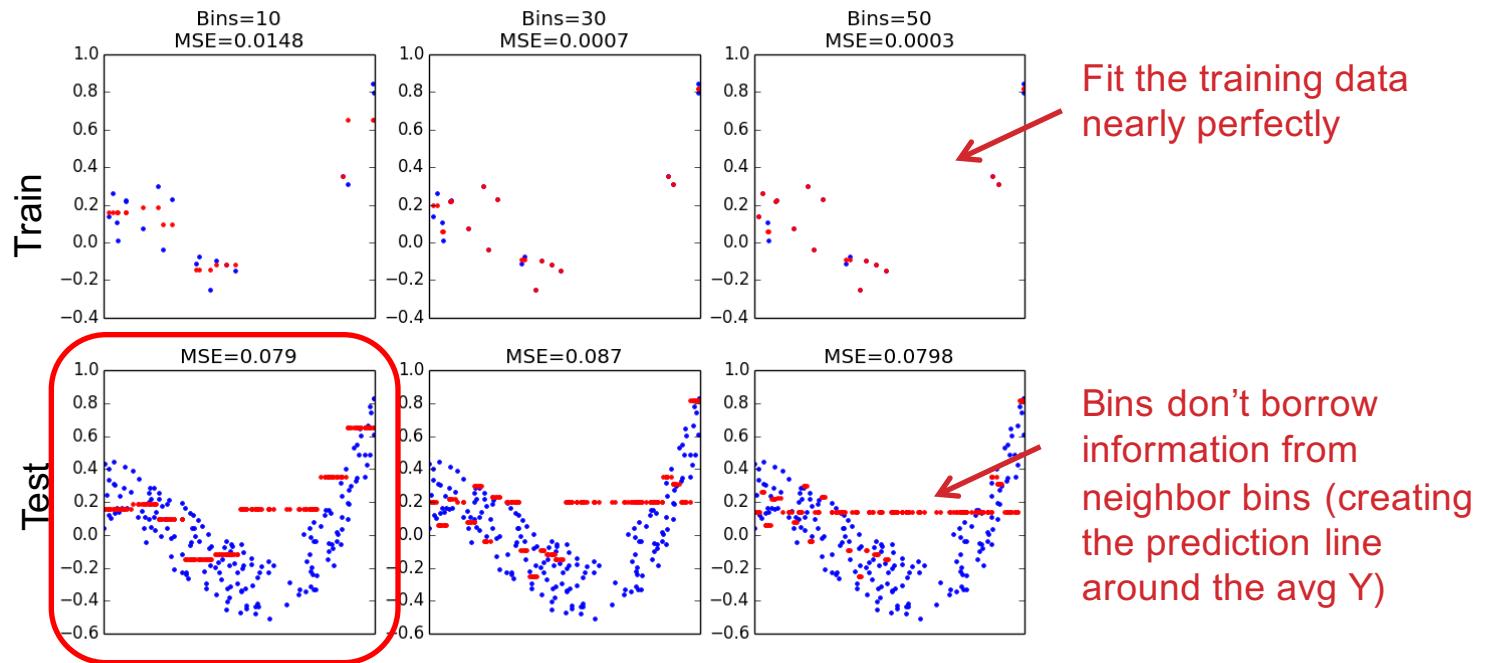
When we have much more training data to fit, the highest order polynomial doesn't help much, but is less likely to overfit. The data was generated using a degree 3 polynomial, and our regression has effectively learned that.



BINNING: NOISY ENVIRONMENT

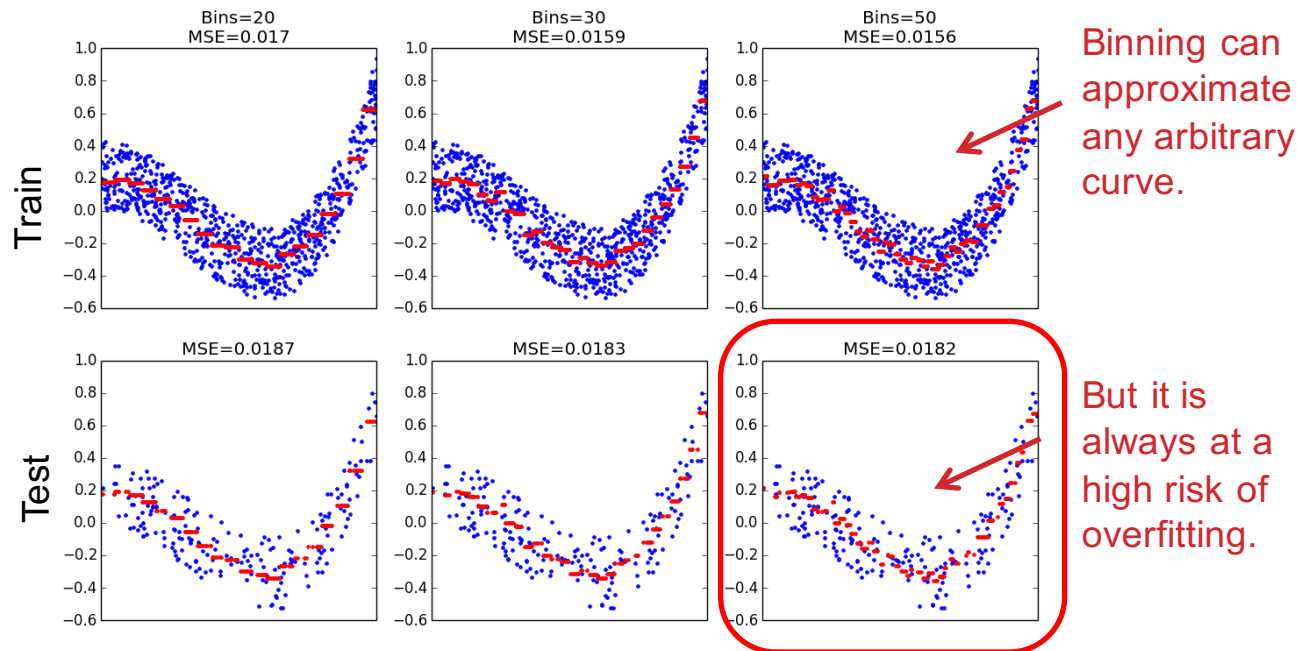
Binning a numeric feature also needs to be done carefully.

Too few bins and you get a poor fit...too many bins and you overfit terribly.



BINNING: INFO RICH ENVIRONMENT

With more data we can support a higher number of bins. This method can be competitive with the polynomial fit, but it is still easy to over fit. Careful consideration and testing must be done to choose the appropriate bin size.



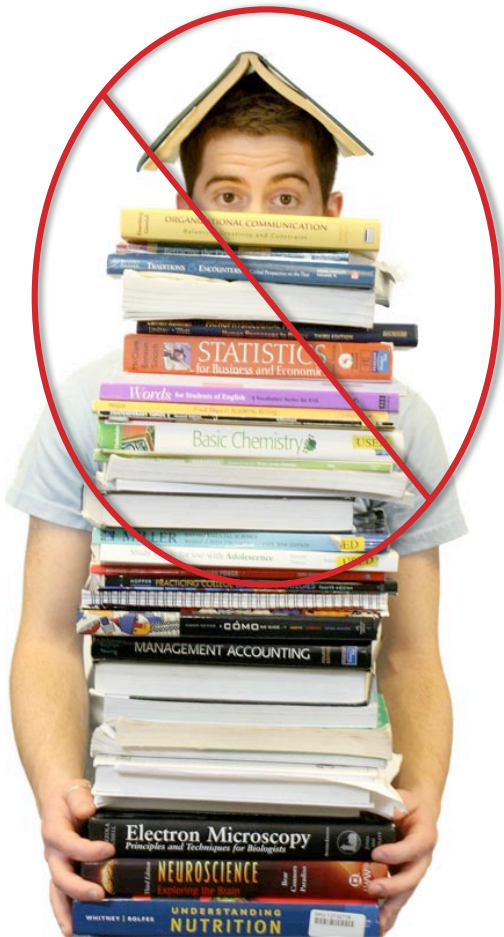
DOMAIN KNOWLEDGE FEATURE EXTRACTION

- Many industrial data collection processes store data in either primitive log formats or in schemas designed to support transactional systems.
- **These systems are usually built for speed, not analytics.**
- As a result, data scientists often need to design their own features to be used in analysis and modeling.



Copyright: Brian d'Alessandro, all rights reserved

HOW TO DESIGN APPROPRIATE FEATURES?



- Unfortunately there are usually no text books or theoretical systems teaching us how to design good features
- Intuition, creativity and knowledge of the application domain are needed.
- Data scientists should consult with business experts when doing this.

Copyright: Brian d'Alessandro, all rights reserved

GUIDELINES FOR DOMAIN KNOWLEDGE FEATURE ENGINEERING

Try to understand the physical mechanism at work

Not all processes you model have theoretical models that define them. Nonetheless, you can put yourself in the user's shoes. What are actions and events that you think might be correlated with the target variable. In this sense, think causally – i.e., what are factors (X) known to cause Y?

Thoroughly review the underlying data system

The data is collected to support the product and transactional systems. These might lead clues as to what can or should be used as features.

Consult with business people

People with extensive knowledge of the business might have solid and experience driven intuition that can help you formulate a feature plan.

Use intuition, then test

It is generally not helpful to debate which features will be the best, as data will answer that question. Nonetheless there is a cost to building each feature, so invest in building the features that are guided by strong knowledge of the problem.

LEAKAGE

1. Target Variable Leakage - Having a feature that is caused by the outcome of the target variable.

Examples:

- using the fact that a user saw a “Thank You” page to predict that the user will purchase (hint: if the feature happens after the outcome, it’s not a legitimate feature)
- Positives and Negatives are stored in different log files, and each log uses a different range for the user id. The user id perfectly predicts the outcome!

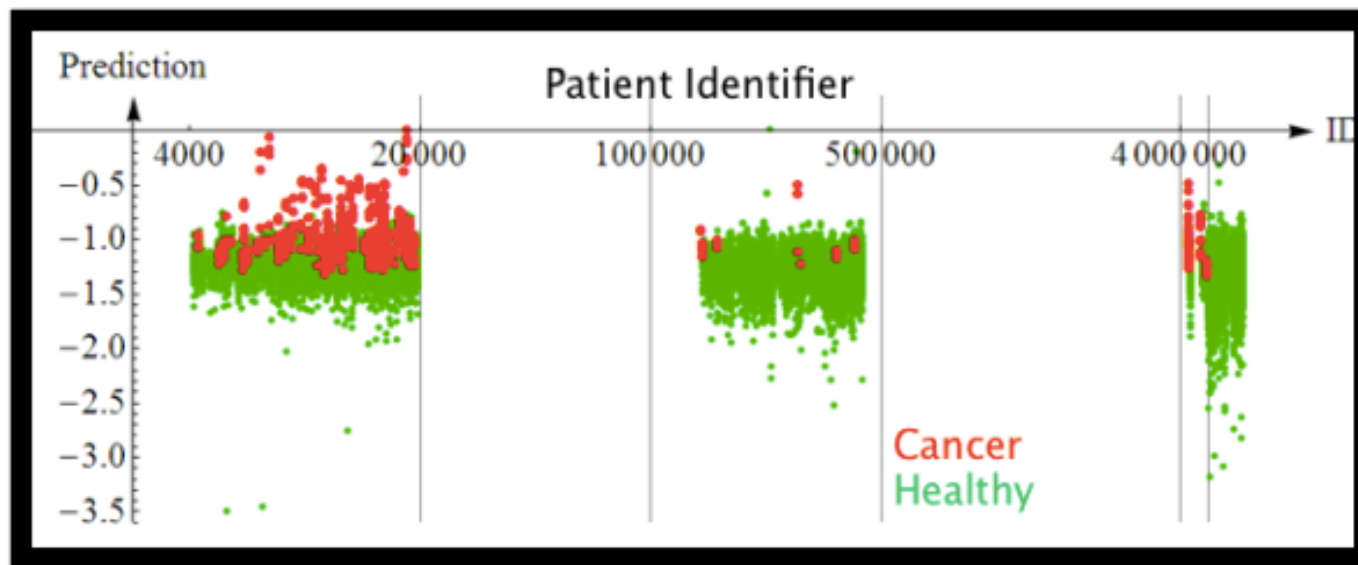
2. Training/Testing Leakage – Having records in the training set also appear in the test set

Model performance that is too good to be true is a good reason to suspect leakage!

LEAKAGE - EXAMPLE

Patients with Cancer were pulled from a different system and generally had a different distribution of IDs. But ID is a worthless artifact of the system, and not a true predictor of cancer.

Leakage is often caused by sloppy data prep!!!



From the KDD 2008 Cup Winning Solution, provided by Claudia Perlich.

Copyright: Brian d'Alessandro, all rights reserved

LETS BUILD A DATASET FOR TWITTER RECOMMENDATIONS

Who to follow

Twitter accounts suggested for you based on who you follow and more.

Search using a person's full name or @username

Search Twitter



Johan Ugander @jugander

I study social graphs and the structure of social data.
Present: MSR Redmond postdoc; Future: Stanford
MS&E faculty, Fall 2015; Past: Cornell Applied Math
PhD

Followed by josh attenber, jake hofman and Blake
Shaw.



+ Follow



Moira Burke @grammarnerd

Data scientist at Facebook. Recent CMU grad. I
study people and the internets.



+ Follow



Michael E. Driscoll @medriscoll

Founder + CEO @Metamarkets. Investor @DCVC. I
♥ data, analytics, & visualization.

Followed by Randy Au, jake hofman and chris wiggins.



+ Follow



Mark Faridani @siah









Just finished a PhD at Berkeley. Silicon Valley
Scientist. Machine Learning, HCI, Data Science and
Statistics. Addicted to coffee and in love with
@marjoo.



+ Follow



Copyright: Brian d'Alessandro, all rights reserved

DATA INSTANCES

<u>User i</u>	<u>User j</u>	<u>Label</u>	<u>Feature</u>
		1	$\langle X_1, X_2, \dots, X_k \rangle$
		0	$\langle X_1, X_2, \dots, X_k \rangle$
		0	$\langle X_1, X_2, \dots, X_k \rangle$
		1	$\langle X_1, X_2, \dots, X_k \rangle$

FEATURE TYPES

In a dataset where instances are pairwise combinations of individuals, we can group features into sets that reflect individual characteristics as well as pairwise characteristics.

<u>User i</u>	<u>User j</u>	<u>Label</u>	<u>Features</u>
		1	<X1, X2, .. Xk>

User_i features = <number of followers, interests, etc.>

User_j features = <number of followers, number of follows, interests, etc.>

Pairwise features = <number of friends in common, degree of separation, number common interests, etc.>