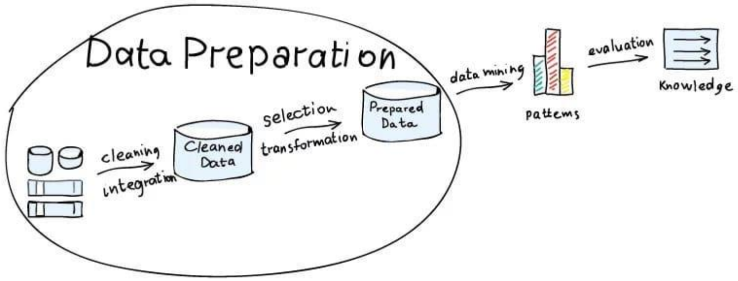
2110446
Data Science
and Data
Engineering

CHULA ENGINEERING
Foundation toward Innovation
COMPUTER



Data Preparation with Python

Peerapon Vateekul, Ph.D.
Department of Computer Engineering,
Faculty of Engineering, Chulalongkorn University
Peerapon.v@chula.ac.th

Data Preparation

+ Terminology: Data table

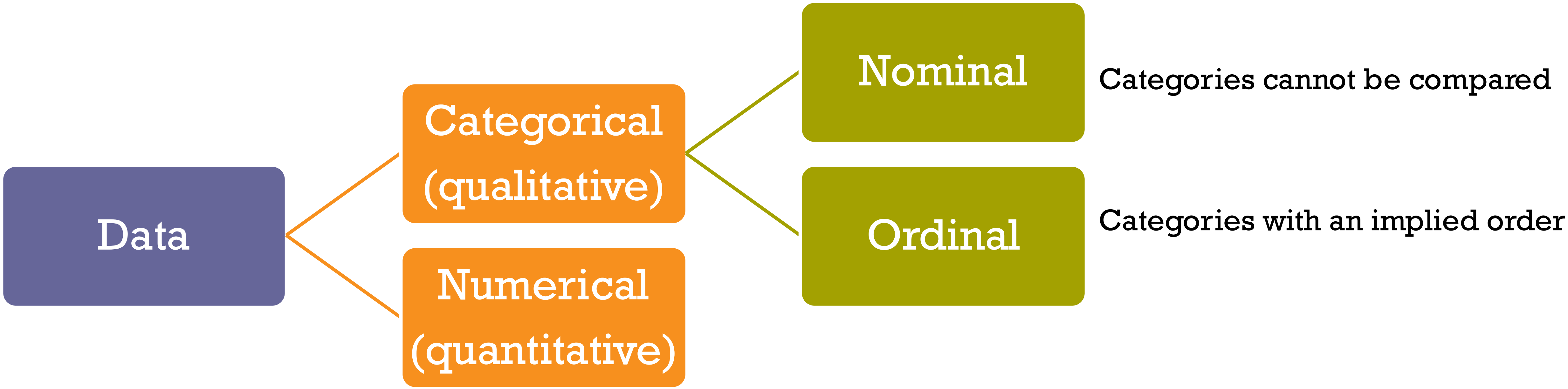
	inputs			target
Age	**Income**	**Gender**	**Province**	**Purchase**
25	25,000	Female	Bangkok	Yes
35	50,000	Female	Nontaburi	Yes
32	35,000	Male	Bangkok	No

- Row
  - Example, instance, case, observation, subject

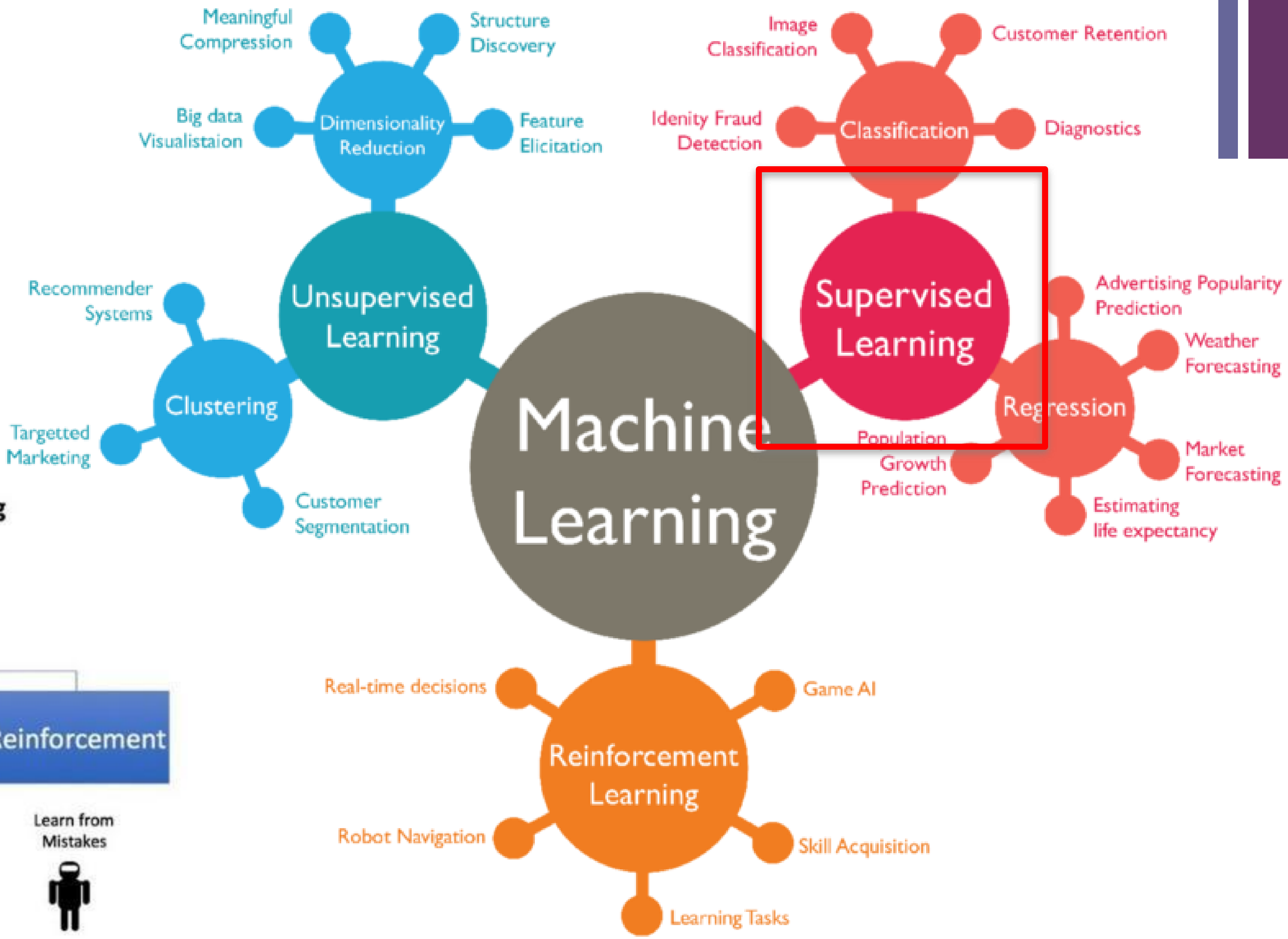
Pandas

- Column
  - Feature, variable, attribute
- Input
  - Predictor, independent, explanatory variable
- Target
  - Output, outcome, response, dependent variable

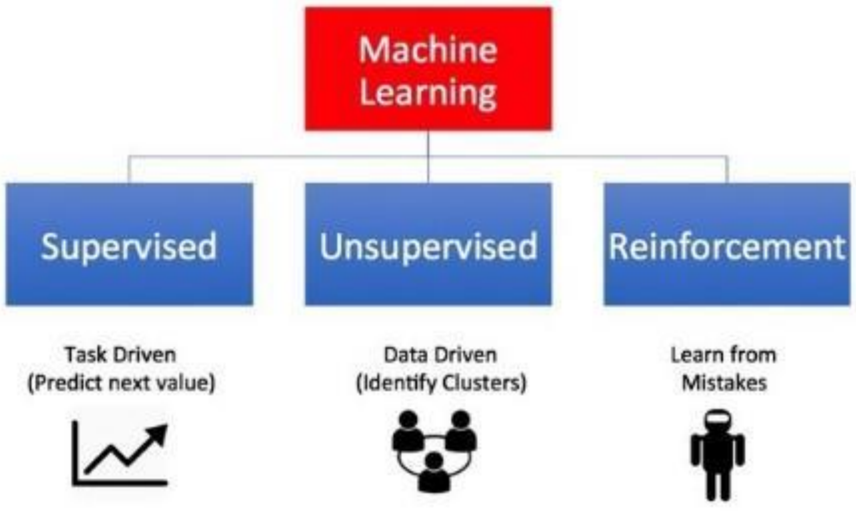
Terminology: Kinds of data



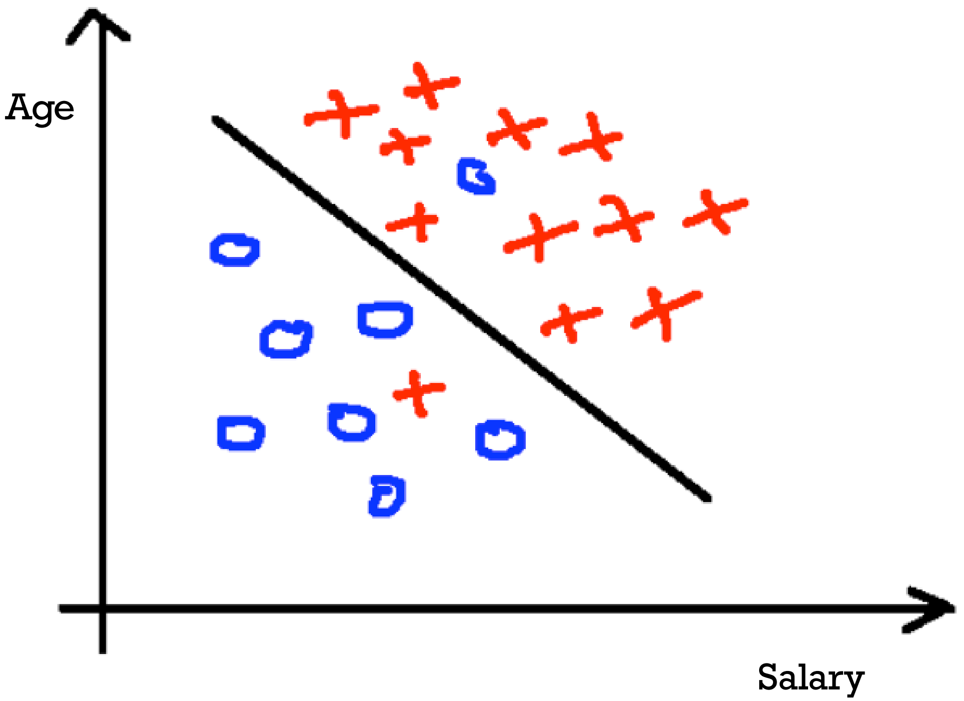
+ Machine Learning (cont.)



Types of Machine Learning



Classification: predicting a category
Logistic Regression

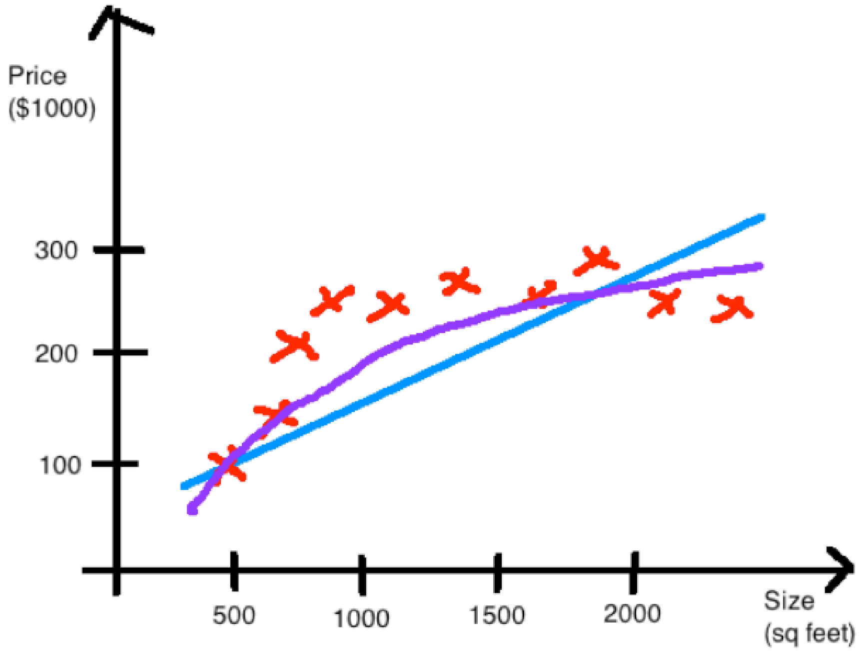


Predict targeted customers who tend to buy our product (yes/no)

- **Some techniques:**
  - Naïve Bayes
  - Decision Tree
  - Logistic Regression
  - Support Vector Machines
  - Neural Network
  - Ensembles
- **Sample Applications**
  - Database marketing
  - Fraud detection
  - Pattern detection
  - Churn customer detection

+

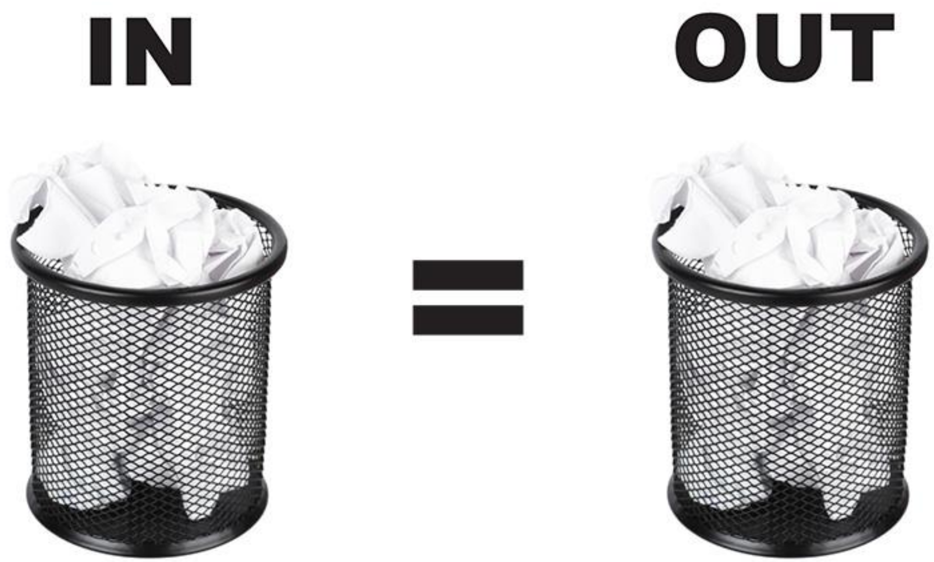
Regression: predict a continuous value
Linear Regression

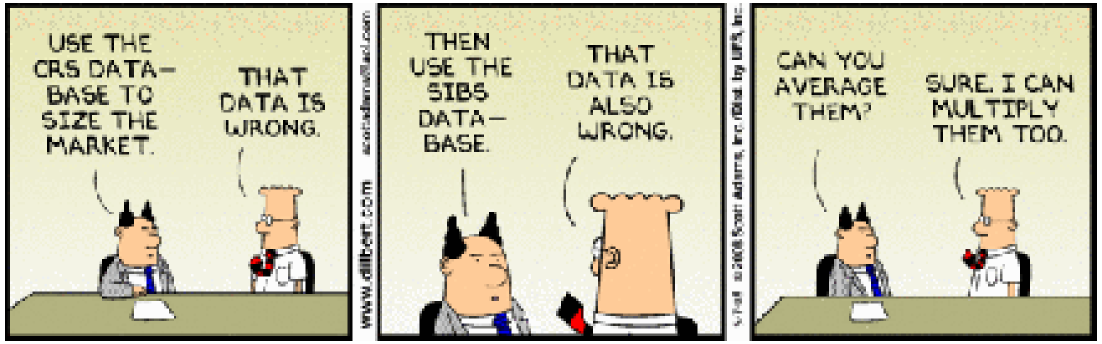


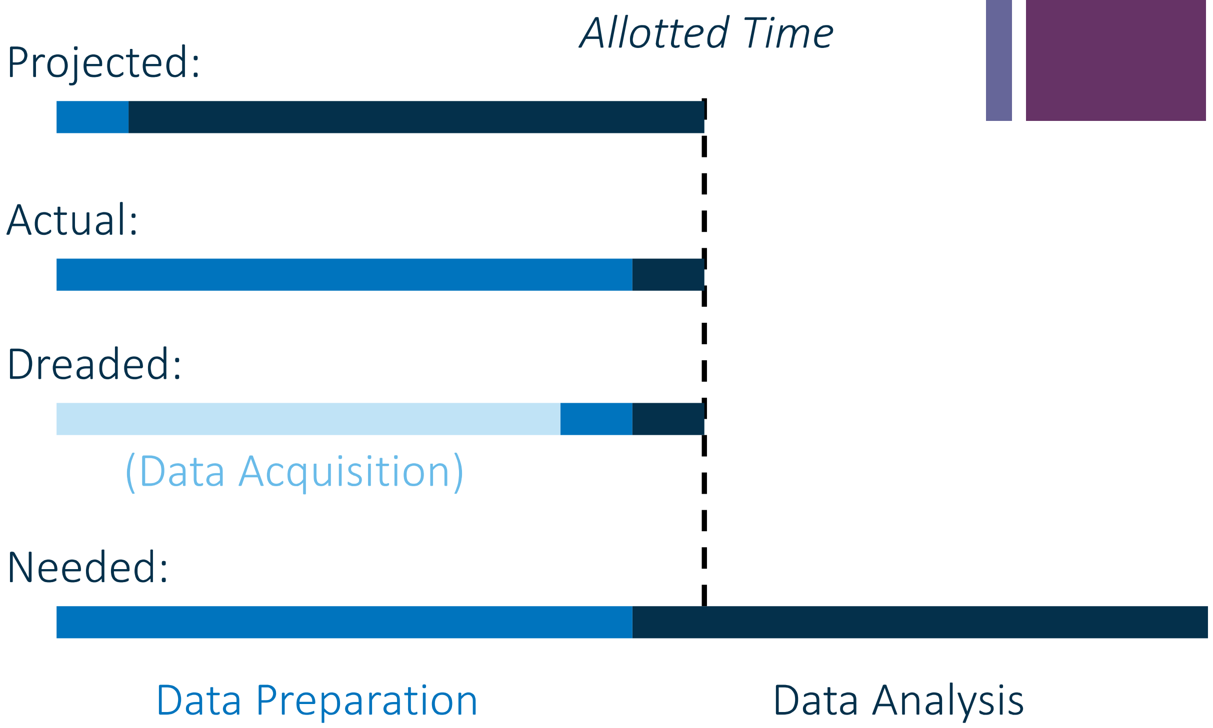
Predict a sale price of each house

- **Some techniques:**
  - Linear Regression / GLM
  - Decision Trees
  - Support vector regression
  - Neural Network
  - Ensembles
- **Sample Applications**
  - Financial risk management
  - Revenue forecasting

Data preparation is very important!







Analytics workflow

Data preparation challenges

- Massive data sets
- Temporal infidelity
- Transaction and event data
- Non-numeric data
- Exceptional, extreme, and missing values
- Stationarity

$$\hat{y} = \hat{w}_0 + \hat{w}_1 x_1 + \hat{w}_2 x_2$$

$$Spend = 500 + 2 \times Age + 3 \times Province$$


Practice is everything.
Periander
quotefancy

DATAQUEST BLOG

28 DECEMBER 2016 / DATA CLEANING

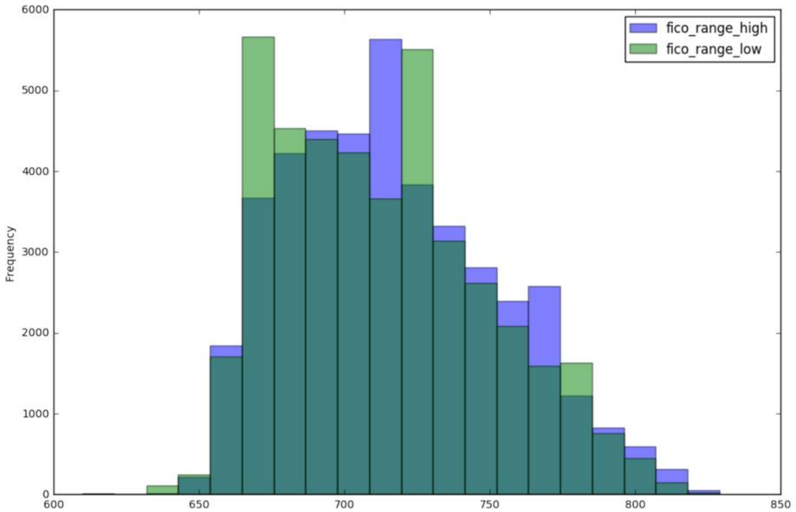
Preparing and Cleaning Data for Machine Learning

- 1) Examining the Data Set
- 2) Narrowing down columns manually
  - Remove Id's
  - Irrelevant variables
  - Remove zipcode & date
  - Temporal infidelity (data from future)
  - Calculated variables
  - Decide target
    - Select studied cases
    - Distribution of target variables
  - Remove flat values
- 3) Preparing features for ML
  - Preview data
  - Handling missing values
    - Drop unqualified features
  - Investigate categorical features
    - Drop too many unique values (treat as Id)
    - Convert ordinal to numeric
    - Convert categorical to numeric
  - Check all numeric variables
- 4) Other preprocessing steps:
  - Train/Test/Validate

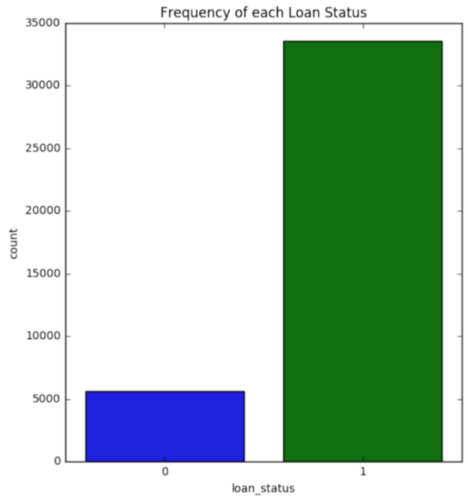
https://www.dataquest.io/blog/machine-learning-preparing-data/

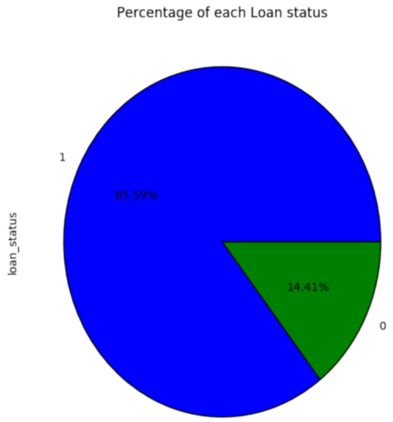
1) Examining the Data Set

- Numerical variables
  - Out of ranges
  - Distribution: histogram



- Categorical variables
  - Miscodes
  - Distribution: frequency table, bar chart
- Target variable
  - Understand class distribution (proportion of each class): bar chart, pie chart

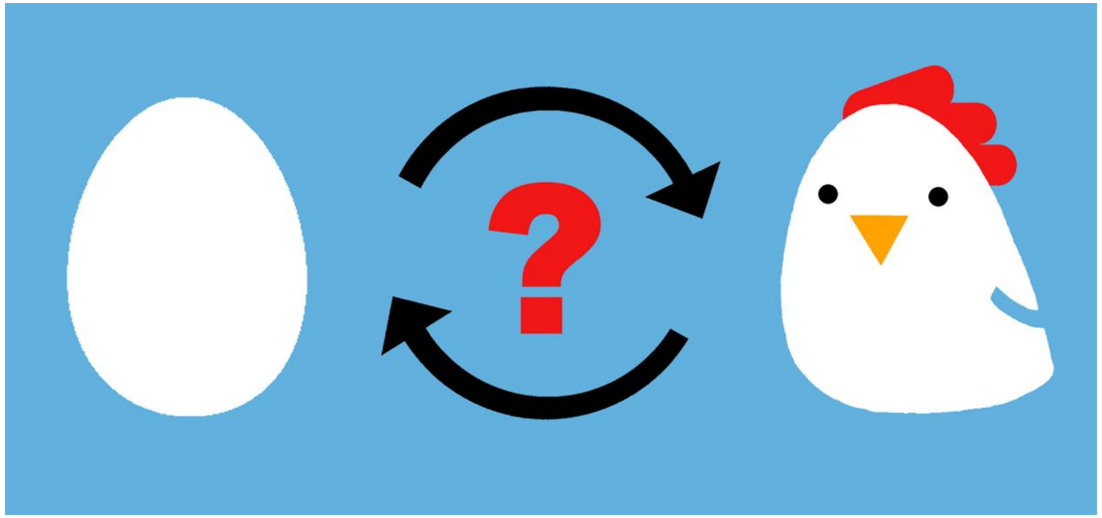




2) Narrowing down columns: Feature understanding is extremely important!

Remove irrelevant features <u>manually</u>

Domain expert



2) Narrowing down columns (cont.): Remove unqualified features

- Remove unqualified features
- ID's (lack of generalization; overfit)
- Variables with missing values > 50%
- Categorical variables
  - Too many unique values (treat as ID's)
  - Flat values (underfit)

- Special ways to treat these data
- High cardinalities of categorical inputs
  - Recode, consolidation (grouping)
- Zip code
  - Distance to closet branch
- Date/time
  - Recency
  - Month, day of week, year
  - Hours, period of days

2) Narrowing down columns (cont.): Remove temporal Infidelity features

- Occurs when the input variables contain information that will be unavailable at the time that the prediction model is deployed.
- Assume that the model will be deployed in July-2017
  - Should we include a variable called "FICO2017", which is calculated at the end of the year?

2) Narrowing down columns (cont.): Remove leaking-target features

Target variables quantify account responses during the current campaign season.

Name	Label	Description
B_TGT	Tgt Binary New Product	A binary target variable. Accounts coded with a 1 contracted for at least one product in the previous campaign season. Accounts coded with a zero did not contract for a product in the previous campaign season.
INT_TGT	Tgt Interval New Sales	The amount of the financial services products (sum of sales) per account in the previous campaign season, denominated in US dollars.
CNT_TGT	Tgt Count Number New Products	The number of the financial services products (count) per account in the previous campaign season.

3) Preparing features for ML (cont.): 3.1 Impute missing values

? != 0

$$\hat{y} = \hat{w}_0 + \hat{w}_1 x_1 + \hat{w}_2 x_2$$

$$Spend = 500 + 10 \times IncomeK - 20 \times Age$$

- For missing in target, those examples must be removed.
- For missing in inputs, either remove those examples or impute missing values

- 1) Statistical approach
- Numerical variables:
  - Mean
  - Median
- Categorical variables:
  - Mode (most-frequent)
- Stats by group can improve the performance, e.g., income by age group.
- 2) Model-based approach
  - x1 ~ (x2, x3, x4, …)
  - Income ~ Age
  - Tree-based imputation

$$Spend = 500 + 10 \times IncomeK - 20 \times Age + 3 \times Province$$

3) Preparing features for ML (cont.): 3.2 Categorical to numeric variables

- Ordinal variable
  - Enumerate

Grade	GradeNum
A	4.00
B+	3.50
B	3.00
C+	2.50
C	2.00
D+	1.50
D	1.00
F	0.00

Size	SizeNum
XL	5
L	4
M	3
S	2
XS	1

- Nominal variable
  - (1) One-hot vector (dummy codes)
  - (2) Target Averaging (prob)
  - (3) Weight of Evidence (WoE)
  - (4) Smoothed weight of evidence (SWoE)

(1) One-hot vector (dummy codes)

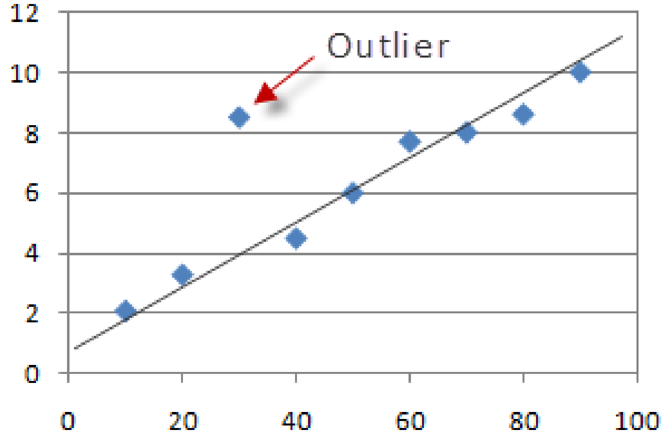
- Dummy coding = (n-1) dummy codes

Branch	BranchNum	D_BKK	B_Patum	D_Non	D_BKK	B_Patum	
BKK	1	1	0	0	1	0	
Patumtani	2	0	1	0	0	1	
Nontaburi	3	0	0	1	0	0	reference

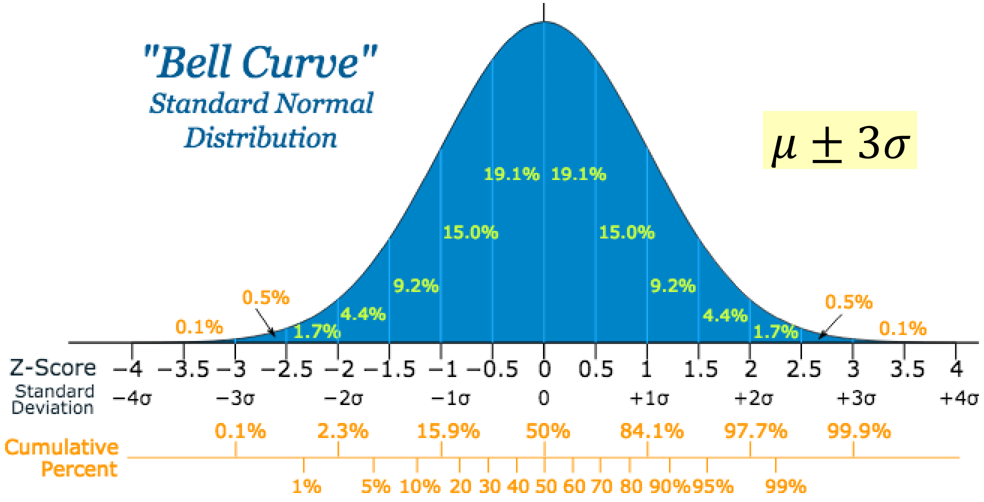
(1) One-hot vector (dummy codes) – Example2

Level	D_A	D_B	D_C	D_D	D_E	D_F	D_G	D_H	D_I
A	1	0	0	0	0	0	0	0	0
B	0	1	0	0	0	0	0	0	0
C	0	0	1	0	0	0	0	0	0
D	0	0	0	1	0	0	0	0	0
E	0	0	0	0	1	0	0	0	0
F	0	0	0	0	0	1	0	0	0
G	0	0	0	0	0	0	1	0	0
H	0	0	0	0	0	0	0	1	0
I	0	0	0	0	0	0	0	0	1

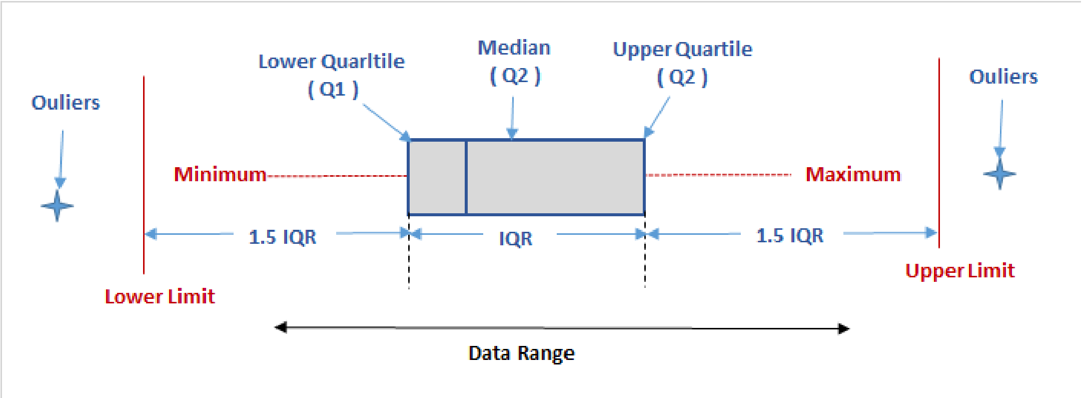
3) Preparing features for ML (cont.): 3.3 Truncate outliers



- Outlier, leverage points, extreme values



$$\mu \pm 3\sigma$$

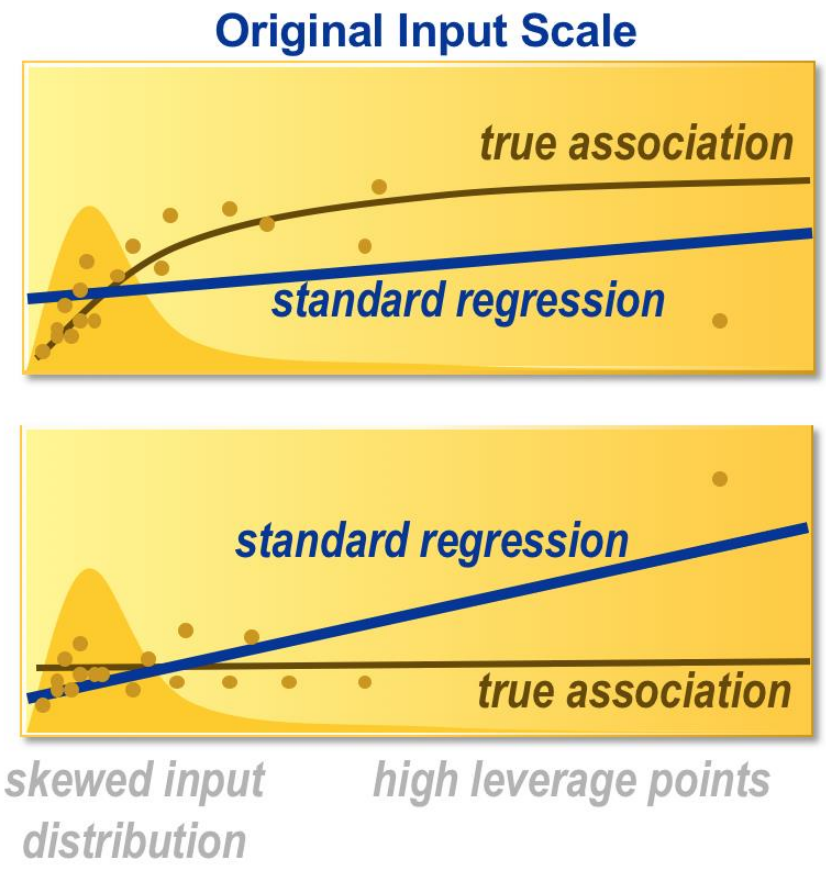


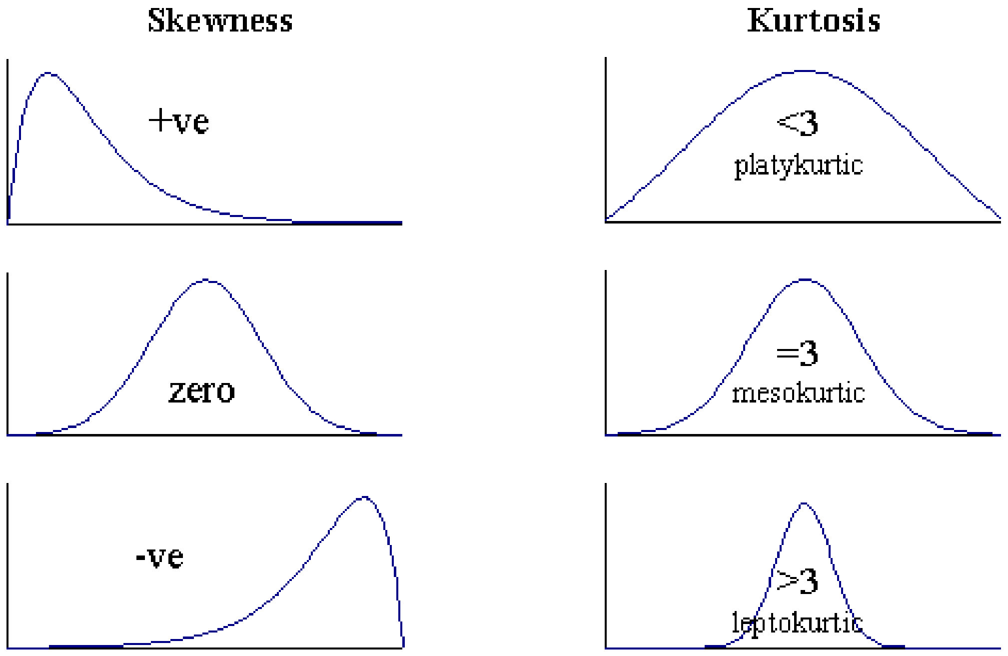
Percentile
1st
2.5th
5th
10th
25th
50th
75th
90th
95th
97.5th
99th

$$\hat{y} = \hat{w}_0 + \hat{w}_1 x_1 + \hat{w}_2 x_2$$

3) Preparing features for ML (cont.): 3.4 Feature transformation

- Skewness
- Example: Salary, Balance in bank account
- **Solutions: Log, Binning**





3) Preparing features for ML (cont.): 3.4 Feature transformation - Log

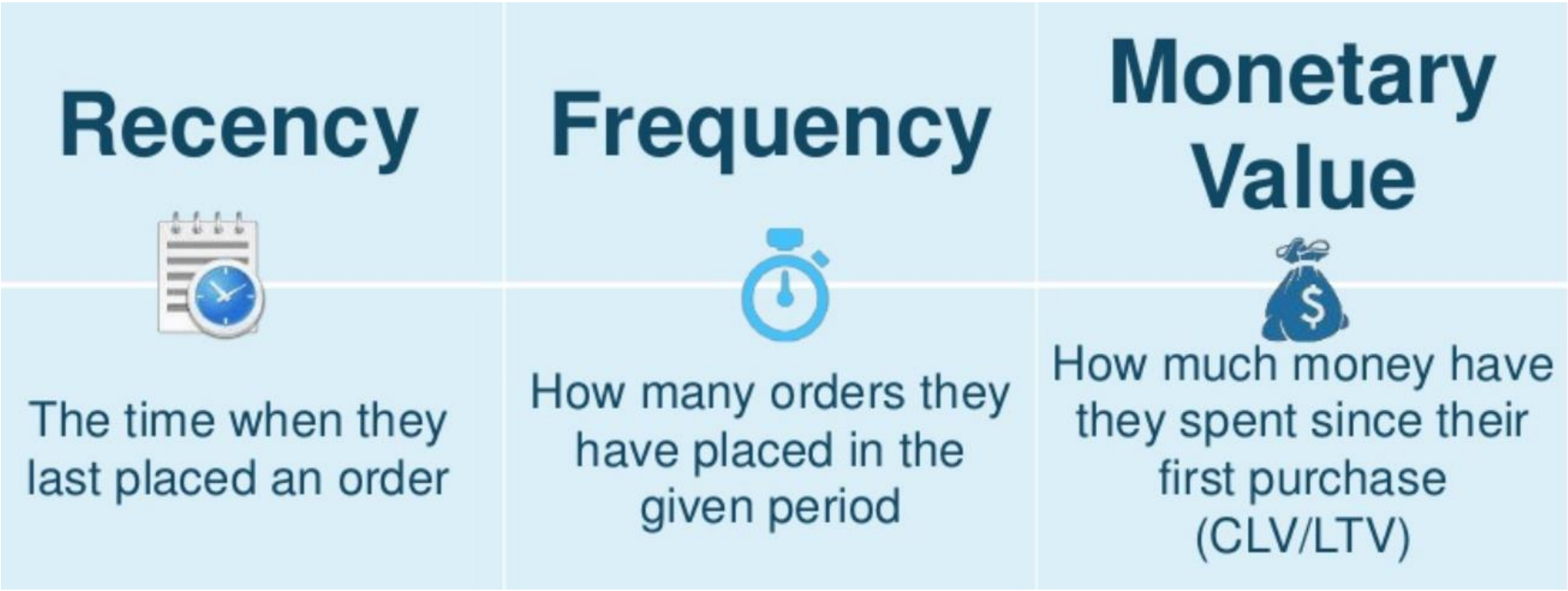
	Spending	Spending with outliers	LOG10(Spending)	LOG10(Spending with outliers)
	2,500.00	2,500.00	3.40	3.40
	2,900.00	2,900.00	3.46	3.46
	3,200.00	3,200.00	3.51	3.51
	4,000.00	4,000.00	3.60	3.60
	4,500.00	4,500.00	3.65	3.65
	6,200.00	6,200.00	3.79	3.79
		10,000,000.00		7.00
mean	**3,883.33**	**1,431,900.00**	**3.57**	**4.06**

3) Preparing features for ML (cont.): 3.4 Feature transformation - Binning

	Spending	Spending with outliers	LOG10(Spending)	LOG10(Spending with outliers)	Binning (Spending)	LOG10(Spending with outliers)
	2,500.00	2,500.00	3.40	3.40	1	1
	2,900.00	2,900.00	3.46	3.46	1	1
	3,200.00	3,200.00	3.51	3.51	2	2
	4,000.00	4,000.00	3.60	3.60	2	2
	4,500.00	4,500.00	3.65	3.65	2	2
	6,200.00	6,200.00	3.79	3.79	3	3
		10,000,000.00		7.00		3
mean	**3,883.33**	**1,431,900.00**	**3.57**	**4.06**	**1.83**	**2.00**

3) Preparing features for ML (cont.): 3.5 Feature engineering

- Feature engineering
  - Calculated variables
  - Behavior from transactional data (RFM/RFA)

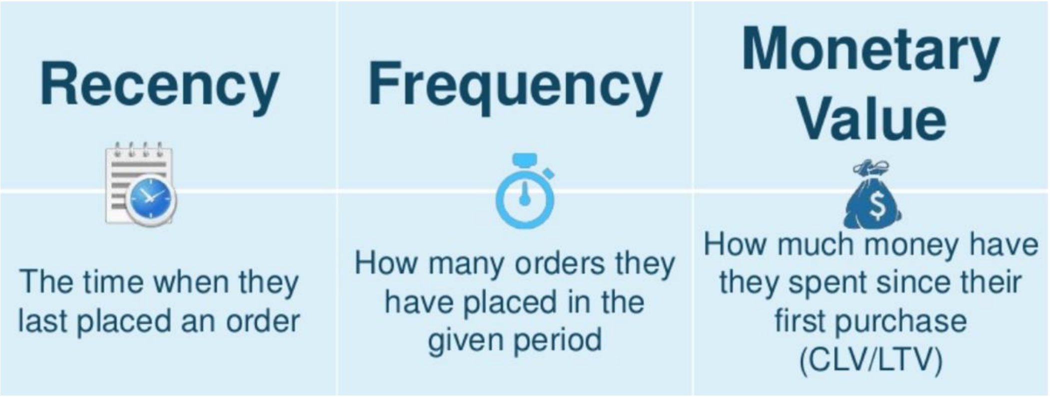


3) Preparing features for ML (cont.): 3.5 Feature engineering (cont.)

- Feature engineering
  - Calculated variables
  - Behavior from transactional data (RFM/RFA)

Transaction

Customer behavior



RFM ANALYSIS

4) Other preprocessing steps: Train/Test – Overfitting issue on train

Training Data

inputs

target

Age	Income	Gender	Province	Purchase
25	25,000	Female	Bangkok	Yes
35	50,000	Female	Nontaburi	Yes
32	35,000	Male	Bangkok	No

Testing Data

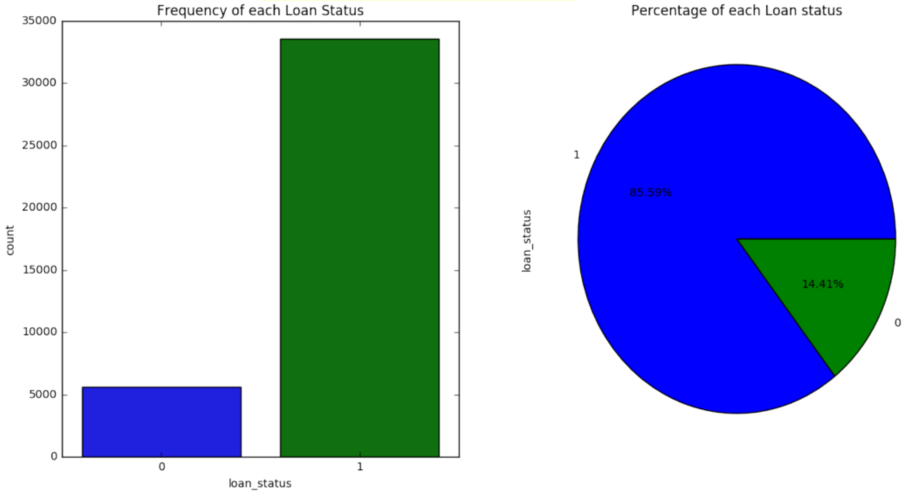
Age	Income	Gender	Province	Purchase
25	25,000	Female	Bangkok	?

+

4) Other preprocessing steps: Train/Test/Validate (cont.)

Simple random sample

Stratification



Remark: Random Seed

- The experiment must be able to reconstruct (replicate).
- All randoms must be assigned a radom seed.
  - random.seed(12345)
  - random_state option

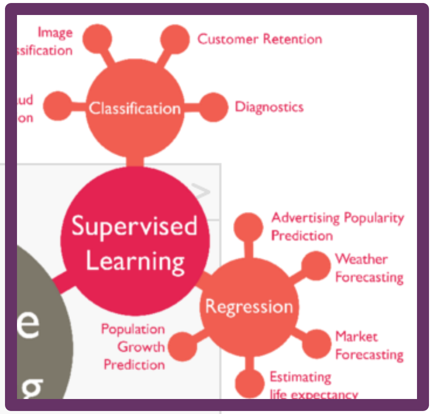
Other data preparation processes

- Impute missing values
- Outlier detections
- Feature transformation
  - Skewness
- Split train/test
  - Simple random sampling
  - Stratification
- Feature clustering
- Feature selection
  - Statistical approach
  - Model-based approach

1.13.2. Univariate feature selection ¶

Univariate feature selection works by selecting the best features based on univariate statistical tests. It can be seen as a preprocessing step to an estimator. Scikit-learn exposes feature selection routines as objects that implement the `transform` method:

- `SelectKBest` removes all but the k highest scoring features
- `SelectPercentile` removes all but a user-specified highest scoring percentage of features
- using common univariate statistical tests for each feature: false positive rate `SelectFpr`, false discovery rate `SelectFdr`, or family wise error `SelectFwe`.
- `GenericUnivariateSelect` allows to perform univariate feature selection with a configurable strategy. This allows to select the best univariate selection strategy with hyper-parameter search estimator.



For instance, we can perform a χ^2 test to the samples to retrieve only the two best features as follows:

```
>>> from sklearn.datasets import load_iris
>>> from sklearn.feature_selection import SelectKBest
>>> from sklearn.feature_selection import chi2
>>> X, y = load_iris(return_X_y=True)
>>> X.shape
(150, 4)
>>> X_new = SelectKBest(chi2, k=2).fit_transform(X, y)
>>> X_new.shape
(150, 2)
```

https://scikit-learn.org/stable/modules/feature_selection.html

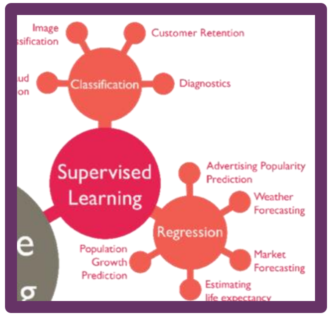
These objects take as input a scoring function that returns univariate scores and p-values (or only scores for `SelectKBest` and `SelectPercentile`):

- For regression: `r_regression`, `f_regression`, `mutual_info_regression`
- For classification: `chi2`, `f_classif`, `mutual_info_classif`

Quick Decision Table

Target **Input**

Task Type	Linear Relationship	Non-linear Relationship	Feature Type	Suggested `score_func`
Classification	✅	❌	Continuous	`f_classif`
Classification	❌	✅	Mixed (encoded)	`mutual_info_classif`
Classification	✅	❌	Non-negative counts	`chi2`
Regression	✅	❌	Continuous	`f_regression`
Regression	❌	✅	Mixed (encoded)	`mutual_info_regression`



Parameters:

score_func : *callable, default=f_classif*

Function taking two arrays X and y, and returning a pair of arrays (scores, pvalues) or a single array with scores. Default is f_classif (see below "See Also"). The default function only works with classification tasks.

Regression

y (cont.) ~ x (cont.)
correlation

These objects take as input a scoring function that returns univariate scores and p-values (or only scores for `SelectKBest` and `SelectPercentile`):

- For regression: `r_regression`, `f_regression`, `mutual_info_regression`
- For classification: `chi2`, `f_classif`, `mutual_info_classif`

y (cont.) ~ x (cont.)
Linear regression

F-test reference
http://facweb.cs.depaul.edu/sjost/csc423/documents/f-test-reg.htm

sklearn.feature_selection.r_regression

`sklearn.feature_selection.r_regression(X, y, *, center=True, force_finite=True)` [source]

Compute Pearson's r for each features and the target.

Pearson's r is also known as the Pearson correlation coefficient.

Linear model for testing the individual effect of each of many regressors. This is a selection procedure, not a free standing feature selection procedure.

The cross correlation between each regressor and the target is computed as:

```
E[(X[:, i] - mean(X[:, i])) * (y - mean(y))] / (std(X[:, i]) * std(y))
```

For more on usage see the User Guide.

New in version 1.0.

Parameters:

X : *{array-like, sparse matrix} of shape (n_samples, n_featur*
The data matrix.

y : *array-like of shape (n_samples,)*
The target vector.

center : *bool, default=True*
Whether or not to center the data matrix `X` and the target vect

force_finite : *bool, default=True*
Whether or not to force the Pearson's R correlation to be finite in `X` or the target `y` are constant, the Pearson's R correlation i correlation of `np.nan` is returned to acknowledge this case. W forced to a minimal correlation of `0.0`.

New in version 1.1.

Returns:

correlation_coefficient : *ndarray of shape (n_features,)*
Pearson's R correlation coefficients of features.

sklearn.feature_selection.f_regression

`sklearn.feature_selection.f_regression(X, y, *, center=True, force_finite=True)` [source]

Univariate linear regression tests returning F-statistic and p-values.

Quick linear model for testing the effect of a single regressor, sequentially for many regressors.

This is done in 2 steps:

1. The cross correlation between each regressor and the target is computed using `r_regression`

```
E[(X[:, i] - mean(X[:, i])) * (y - mean(y))] / (std(X[:, i]) * std(y))
```

2. It is converted to an F score and then to a p-value.

`f_regression` is derived from `r_regression` and will rank features in the same order if all the feature with the target.

Note however that contrary to `f_regression`, `r_regression` values lie in [-1, 1] and can thus be nega therefore recommended as a feature selection criterion to identify potentially predictive feature for a irrespective of the sign of the association with the target variable.

Furthermore `f_regression` returns p-values while `r_regression` does not.

Read more in the User Guide.

Returns:

f_statistic : *ndarray of shape (n_features,)*
F-statistic for each feature.

p_values : *ndarray of shape (n_features,)*
P-values associated with the F-statistic.

y (cont.) ~ x (cont., category)
* x & y must be non-negative.

sklearn.feature_selection.mutual_info_regression

`sklearn.feature_selection.mutual_info_regression(X, y, *, discrete_features='auto', n_neighbors=3, copy=True, random_state=None)` [source]

Estimate mutual information for a continuous target variable.

Mutual information (MI) [1] between two random variables is a non-negative value, which measures the dependency between the variables. It is equal to zero if and only if two random variables are independent, and higher values mean higher dependency.

The function relies on nonparametric methods based on entropy estimation from k-nearest neighbors distances as described in [2] and [3]. Both methods are based on the idea originally proposed in [4].

It can be used for univariate features selection, read more in the User Guide.

Parameters:

X : *array-like or sparse matrix, shape (n_samples, n_features)*
Feature matrix.

y : *array-like of shape (n_samples,)*
Target vector.

discrete_features : *{'auto', bool, array-like}, default='auto'*
If bool, then determines whether to consider all features discrete or continuous. If array, then it should be

Returns:

mi : *ndarray, shape (n_features,)*
Estimated mutual information between each feature and the target.

https://machinelearningmastery.com/feature-selection-for-regression-data/

Classification

y (category) ~ x (category)
Ch-squared

These objects take as input a scoring function that returns univariate scores and p-values (or only scores for `SelectKBest` and `SelectPercentile`):

- For regression: `r_regression`, `f_regression`, `mutual_info_regression`
- For classification: `chi2`, `f_classif`, `mutual_info_classif`

sklearn.feature_selection.chi2

`sklearn.feature_selection.chi2(X, y)`

Compute chi-squared stats between each non-negative feature and cla

This score can be used to select the `n_features` features with the high which must contain only **non-negative features** such as booleans or fr classification), relative to the classes.

Recall that the chi-square test measures dependence between stochas features that are the most likely to be independent of class and therefor

Read more in the User Guide.

Parameters:
- **X** : *{array-like, sparse matrix} of shape (n_samples* Sample vectors.
- **y** : *array-like of shape (n_samples,)* Target vector (class labels).

Returns:
- **chi2** : *ndarray of shape (n_features,)* Chi2 statistics for each feature.
- **p_values** : *ndarray of shape (n_features,)* P-values for each feature.

y (category) ~ x (cont.)
ANOVA

sklearn.feature_selection.f_classif

`sklearn.feature_selection.f_classif(X, y)` [source]

Compute the ANOVA F-value for the provided sample.

Read more in the User Guide.

Parameters:
- **X** : *{array-like, sparse matrix} of shape (n_samples, n_feature* The set of regressors that will be tested sequentially.
- **y** : *array-like of shape (n_samples,)* The target vector.

Returns:
- **f_statistic** : *ndarray of shape (n_features,)* F-statistic for each feature.
- **p_values** : *ndarray of shape (n_features,)* P-values associated with the F-statistic.

See also:
`chi2`
Chi-squared stats of non-negative features for classification tasks.
`f_regression`
F-value between label/feature for regression tasks.

y (category) ~ x (cont., discrete)
* x & y must be non-negative.

sklearn.feature_selection.mutual_info_classif

`sklearn.feature_selection.mutual_info_classif(X, y, *, discrete_features='auto', n_neighbors=3, copy=True, random_state=None)` [source]

Estimate mutual information for a discrete target variable.

Mutual information (MI) [1] between two random variables is a non-negative value, which measures the dependency between the variables. It is equal to zero if and only if two random variables are independent, and higher values mean higher dependency.

The function relies on nonparametric methods based on entropy estimation from k-nearest neighbors distances as described in [2] and [3]. Both methods are based on the idea originally proposed in [4].

It can be used for univariate features selection, read more in the User Guide.

Parameters:
- **X** : *{array-like, sparse matrix} of shape (n_samples, n_features)* Feature matrix.
- **y** : *array-like of shape (n_samples,)* Target vector.
- **discrete_features** : *'auto', bool or array-like, default='auto'* If bool, then determines whether to consider all features discrete or continuous. If array, then it should be either a boolean mask with shape (n_features,) or array with indices of discrete features. If 'auto', it is assigned to False for dense `X` and to True for sparse `X`.
- **n_neighbors** : *int, default=3* Number of neighbors to use for MI estimation for continuous variables, see [2] and [3]. Higher values

multual_info_classif, multual_info_regression

https://www.blog.trainindata.com/mutual-information-with-python/

Train In Data

Feature Selection | Machine Learning | Python

Mutual information with Python

Mutual information (MI) is a non-negative value that measures the mutual dependence between two random variables. The mutual information measures the amount of information we can know from one variable by observing the values of the second variable.

Mutual information

Utilizing the relative entropy, we can now define the MI. We define the MI as the relative entropy between the joint distribution of the two variables and the product of their marginal distributions.

Thus, the MI is given by:

$$I(X,Y) = \sum_X \sum_Y p(x,y) log(\frac{p(x,y)}{p(x)p(y)})$$

	Female	Male	Total
Not survived	0.09	0.52	0.62
Survived	0.25	0.12	0.37
Total	0.34	0.65	1

The MI for the variables survival and gender is:

$$I(X,Y) = 0.0974 \times log\frac{0.0974}{0.6258 \times 0.3490} + 0.5285 \times log\frac{0.5285}{0.6258 \times 0.6510} +$$

$$0.2516 \times log\frac{0.2516}{0.3742 \times 0.3490} + 0.1225 \times log\frac{0.1225}{0.3742 \times 0.6510}$$

Thus, I(X,Y) = 0.2015.

MI estimation for continuous variables

We can extend the definition of the MI to continuous variables by changing the sum over the values of x and y by the integrals:

$$I(X,Y) = \int_X \int_Y p(x,y) log(\frac{p(x,y)}{p(x)p(y)})$$

With continuous variables, the problem is how to estimate the probability densities for each one of the variable values.

Nearest-neighbours method to estimate the MI

We have a series of data points in our data sets that contain values for the continuous variables x and y, with a joint probability p(x,y) that we do not know but must estimate from the observed data. The nearest neighbour methods estimate the joint probability of these 2 continuous variables, and, as well, the joint probability of a continuous and discrete variable.

The following figure (Figure 1A) illustrates the joint distribution of the discrete variable x, which takes 3 values: red, green, or blue; and the continuous variable y. In this example, we see that the different values of x are associated with different values of y; for example, y is generally lower when x is green or red than when x is blue. Therefore, there is a relation between x and y, implying that MI is some positive number.

1.13.4. Feature selection using SelectFromModel

`SelectFromModel` is a meta-transformer that can be used alongside any estimator that assigns importance to each feature through a specific attribute (such as `coef_`, `feature_importances_`) or via an `importance_getter` callable after fitting. The features are considered unimportant and removed if the corresponding importance of the feature values are below the provided `threshold` parameter. Apart from specifying the threshold numerically, there are built-in heuristics for finding a threshold using a string argument. Available heuristics are "mean", "median" and float multiples of these like "0.1*mean". In combination with the `threshold` criteria, one can use the `max_features` parameter to set a limit on the number of features to select.

For examples on how it is to be used refer to the sections below.

Next topic

Examples

- Model-based and sequential feature selection

1.13.4.1. L1-based feature selection

Linear models penalized with the L1 norm have sparse solutions: many of their estimated coefficients are zero. When the goal is to reduce the dimensionality of the data to use with another classifier, they can be used along with `SelectFromModel` to select the non-zero coefficients. In particular, sparse estimators useful for this purpose are the `Lasso` for regression, and of `LogisticRegression` and `LinearSVC` for classification:

```
>>> from sklearn.svm import LinearSVC
>>> from sklearn.datasets import load_iris
>>> from sklearn.feature_selection import SelectFromModel
>>> X, y = load_iris(return_X_y=True)
>>> X.shape
(150, 4)
>>> lsvc = LinearSVC(C=0.01, penalty="l1", dual=False).fit(X, y)
>>> model = SelectFromModel(lsvc, prefit=True)
>>> X_new = model.transform(X)
>>> X_new.shape
(150, 3)
```

Appendix for statistical calculation

Chi-squared
Correlation

Statistical approach

- Chi-squared
- Correlation

Type of Predictors / Type of Response	Categorical	Continuous	Continuous and Categorical
Continuous	Analysis of Variance (ANOVA)	Ordinary Least Squares (OLS) Regression	Analysis of Covariance (ANCOVA)
Categorical	Contingency Table Analysis or Logistic Regression	Logistic Regression	Logistic Regression

Chi-Square Test (categorical variables)

Type of Response \ Type of Predictors	Categorical	Continuous	Continuous and Categorical
Continuous	Analysis of Variance (ANOVA)	Ordinary Least Squares (OLS) Regression	Analysis of Covariance (ANCOVA)
Categorical	Contingency Table Analysis or Logistic Regression	Logistic Regression	Logistic Regression

Categorical Variables Association

- An **association** exists between **two categorical variables** if the distribution of one variable changes when the value of the other variable changes.
- If there is **no association**, the distribution of the first variable is the same regardless of the level of the other variable.

Categorical Variables Association (cont.)
Confusion Matrix, Contingency Table

		
 	**70%**	**30%**
 	**70%**	**30%**

There seems to be **no association** between your mood and the weather because the row percentages are the **same** in each column.

Categorical Variables Association (cont.)

Confusion Matrix, Contingency Table

		
 	**90%**	**10%**
 	**30%**	**70%**

There seems to be **an association** because the row percentages are the **different** in each column.

Chi-Square Test (cont.)

	Outcome		
	Yes	No	Total
Group A	60	20	80
Group B	90	10	100
Total	150	30	180

- Under the **null hypothesis** that there is **no association between the Row and Column variables**.
  - **The expected percentage** in any R*C cell will be equal to the percent in that cell's row (R/T) times the percent in the cell's column (C/T) = (R/T)*(C/T).
  - **The expected count** is then only that expected percentage times the total sample size = (R/T)*(C/T)*T = (R*C)/T.

$$Exp_{ij} = \frac{T_i \ x \ T_j}{N}$$

- Chi-square tests and the corresponding p-values
  - determine whether an association exists
  - do not measure the strength of an association
  - depend on and reflect the sample size.

$$\chi^2 = \sum_{i=1}^{R} \sum_{j=1}^{C} \frac{(Obs_{ij} - Exp_{ij})^2}{Exp_{ij}}$$

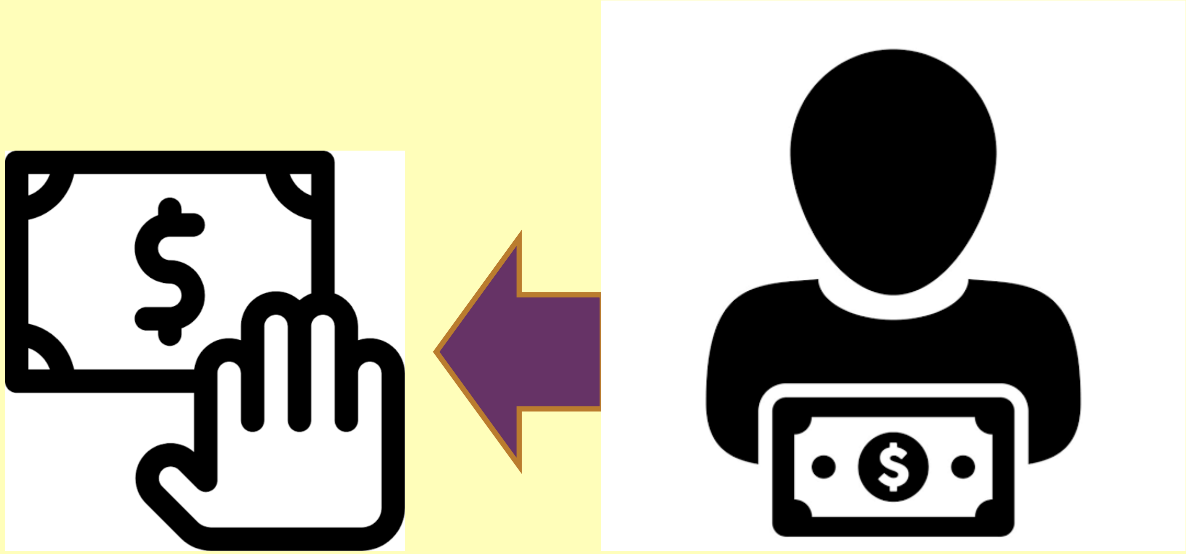
Chi-Square Test

- A commonly used test to check whether there is an association between two categorical variables

- The chi-square test **measures** the difference between the **observed frequencies** and the **expected frequencies**
  - **H0: Observed freq. = expected freq.** → **No Association**
  - **H1: Observed freq. ≠ expected freq.** → **Association**

- If you have a significant chi-square statistic, there is strong evidence that there is **an association** between your variables.

Continuous ~ Continuous (Correlation, Regression)
One-to-One



Spending

Salary

Type of Response \ Type of Predictors	Categorical	Continuous	Continuous and Categorical
Continuous	Analysis of Variance (ANOVA)	Ordinary Least Squares (OLS) Regression	Analysis of Covariance (ANCOVA)
Categorical	Contingency Table Analysis or Logistic Regression	Logistic Regression	Logistic Regression

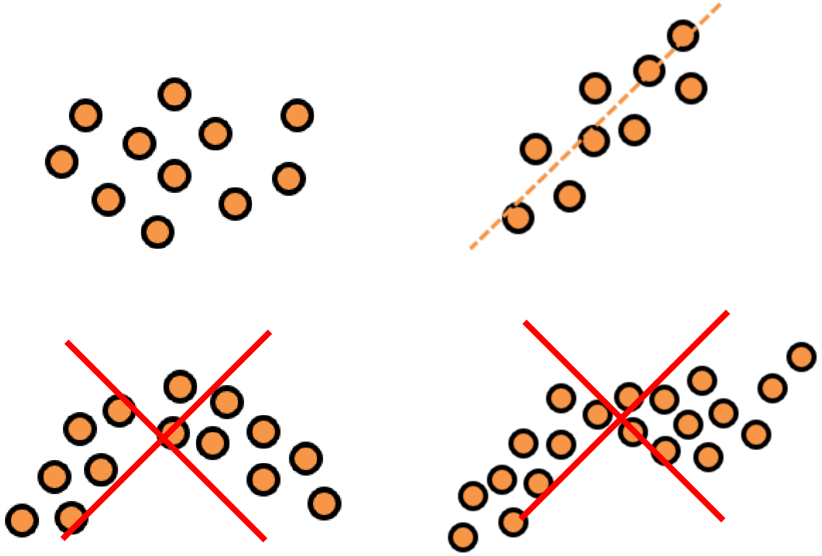
Correlation

- Correlation is a measure that describes the **strength and direction of a relationship** between two variables. It is commonly used in statistics, economics and social sciences for budgets, business plans and etc.
- The method used to understand how closely each variable is related is called **correlation analysis**.

Pearson Correlation

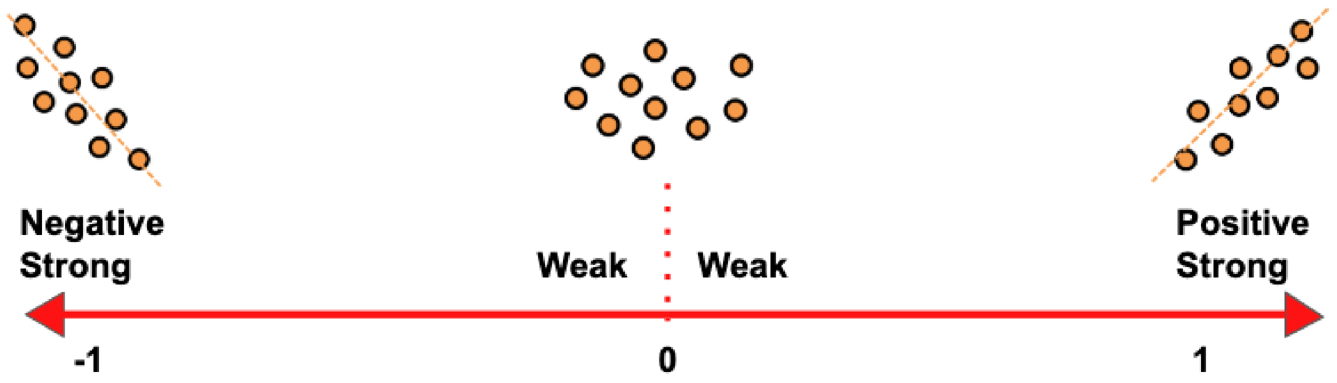
- Pearson Correlation, which is the Pearson Product Moment Correlation (PPMC), is used to evaluate **linear relationships** between two **continuous variable**
- Here's the most commonly used formula to find the Pearson correlation coefficient, which can be called Pearson's R:

$$r = \frac{\sum(x_i - x_{\text{average}})(y_i - y_{\text{average}})}{\sqrt{\sum(x_i - x_{\text{average}})^2 * \sum(y_i - y_{\text{average}})^2}}$$



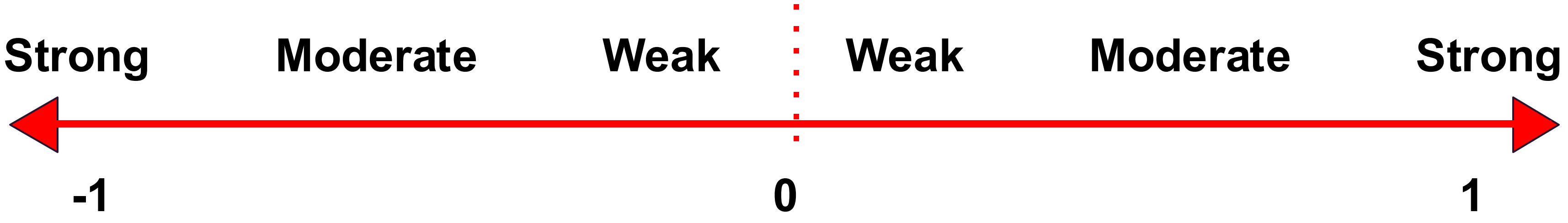
Correlation Coefficient

- The numerical measure of the degree of association between two continuous variables is called the **correlation coefficient (r)**.
- The coefficient value is always between -1 and 1 and it measures both the **strength** and **direction** of the linear relationship between the variables.



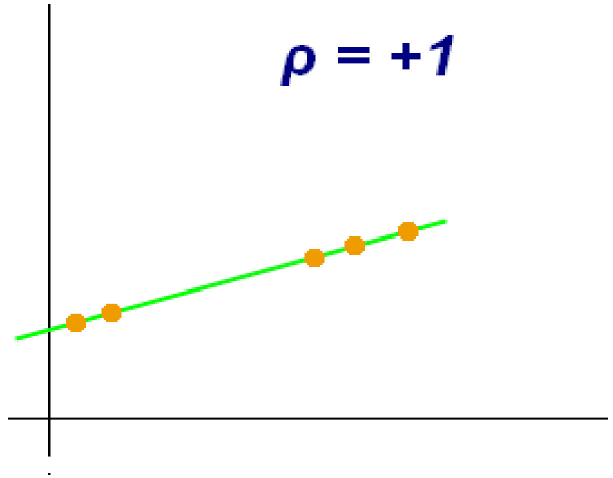
Correlation Coefficient (cont.): Strength

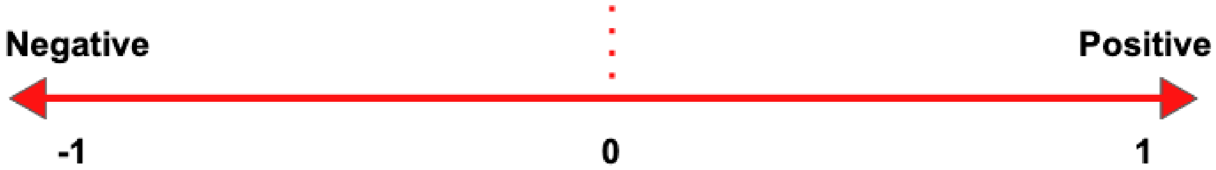
- **Strength**
  - The values of **-1 and 1** indicate a perfect **linear relationship** when all the data points fall on a line. Normally, either positive or negative, is **rarely** found.
  - A coefficient of **0** indicates no linear relationship between the variables. This is what you are likely to get with two sets of random numbers.
  - Values **between 0 and +1/-1** represent a scale of weak, moderate and strong relationships. As the coefficient gets closer to either -1 or 1, the strength of the relationship increases.

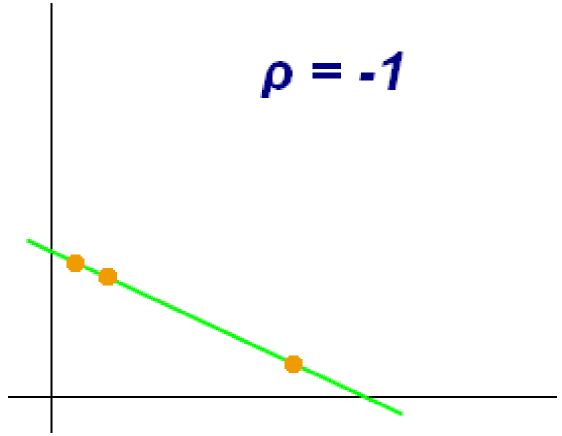


Correlation Coefficient (cont.): Direction

- **Direction**
  - **Positive coefficients** represent **direct** linear association (upward-sloping)
  - **Negative coefficients** represent **inverse** linear association (downward-sloping)





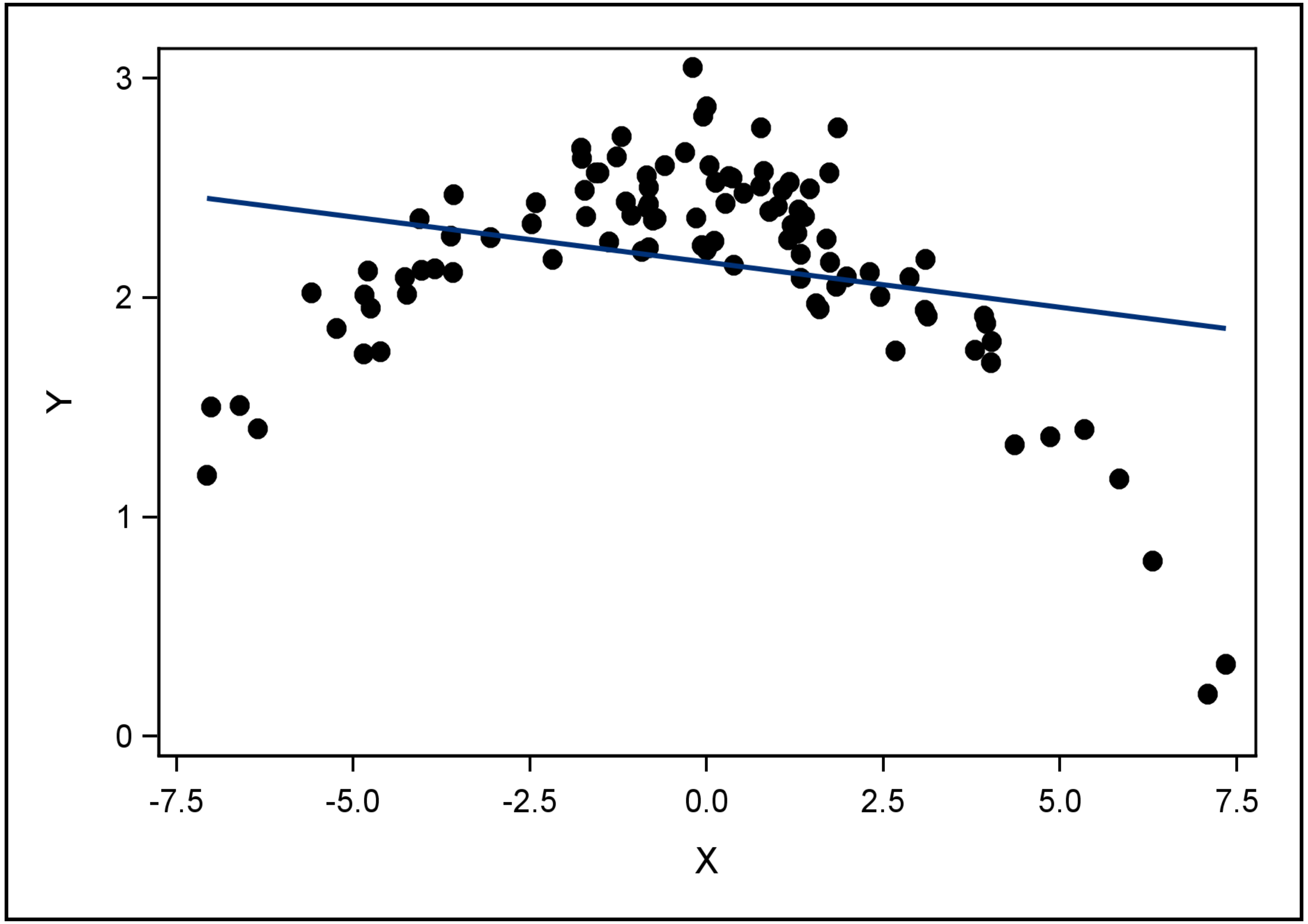


Hypothesis Test for a Correlation

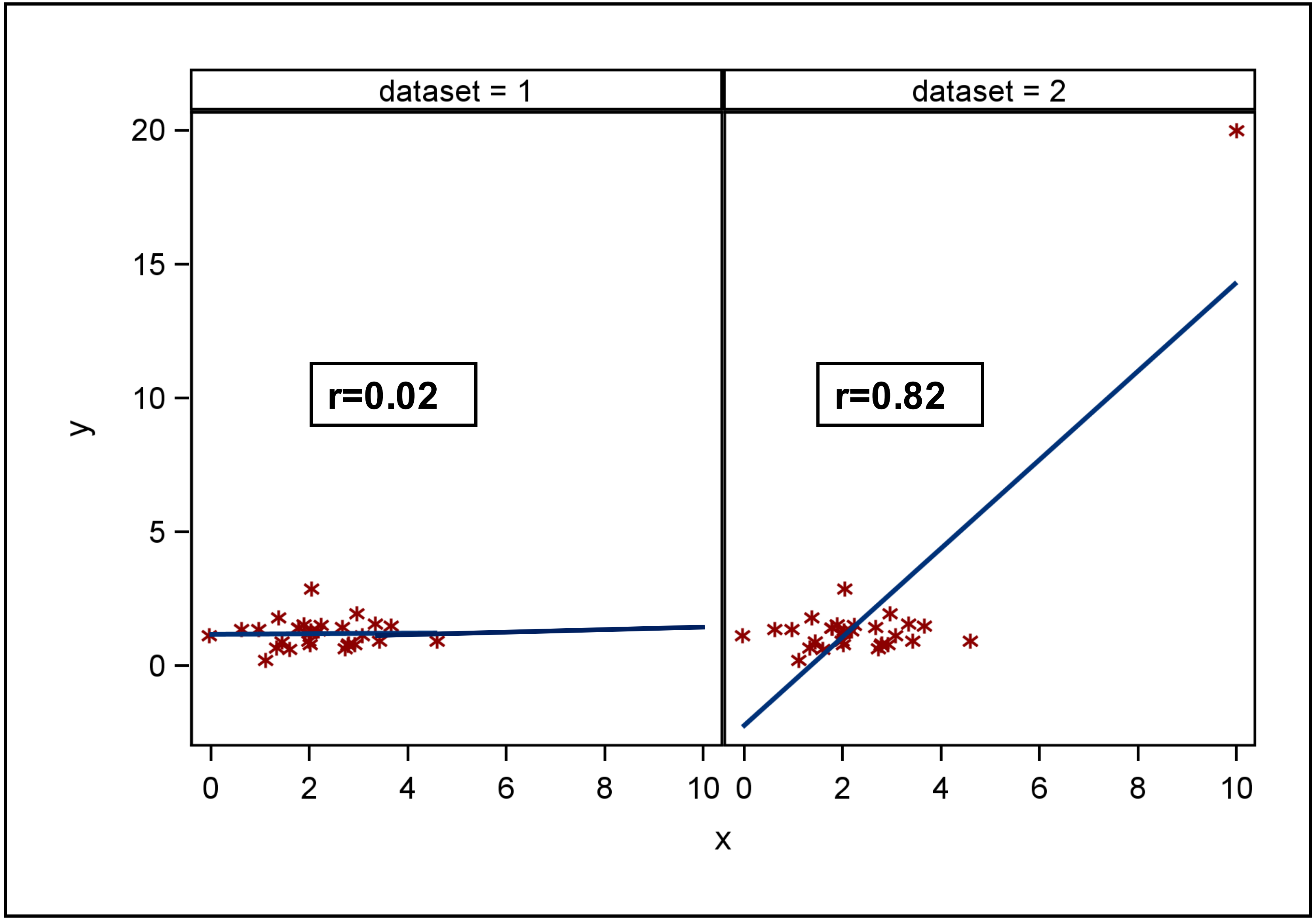
- The parameter representing correlation is ρ.
- ρ is estimated by the sample statistic r.

- H_0: $\rho=0$
- Rejecting H0 indicates only great confidence that ρ is not exactly zero.
- A p-value does not measure the magnitude of the association.
- Sample size affects the p-value.

Remark1: Missing Another Type of Relationship



Remark2: Extreme Data Values



Conclusion

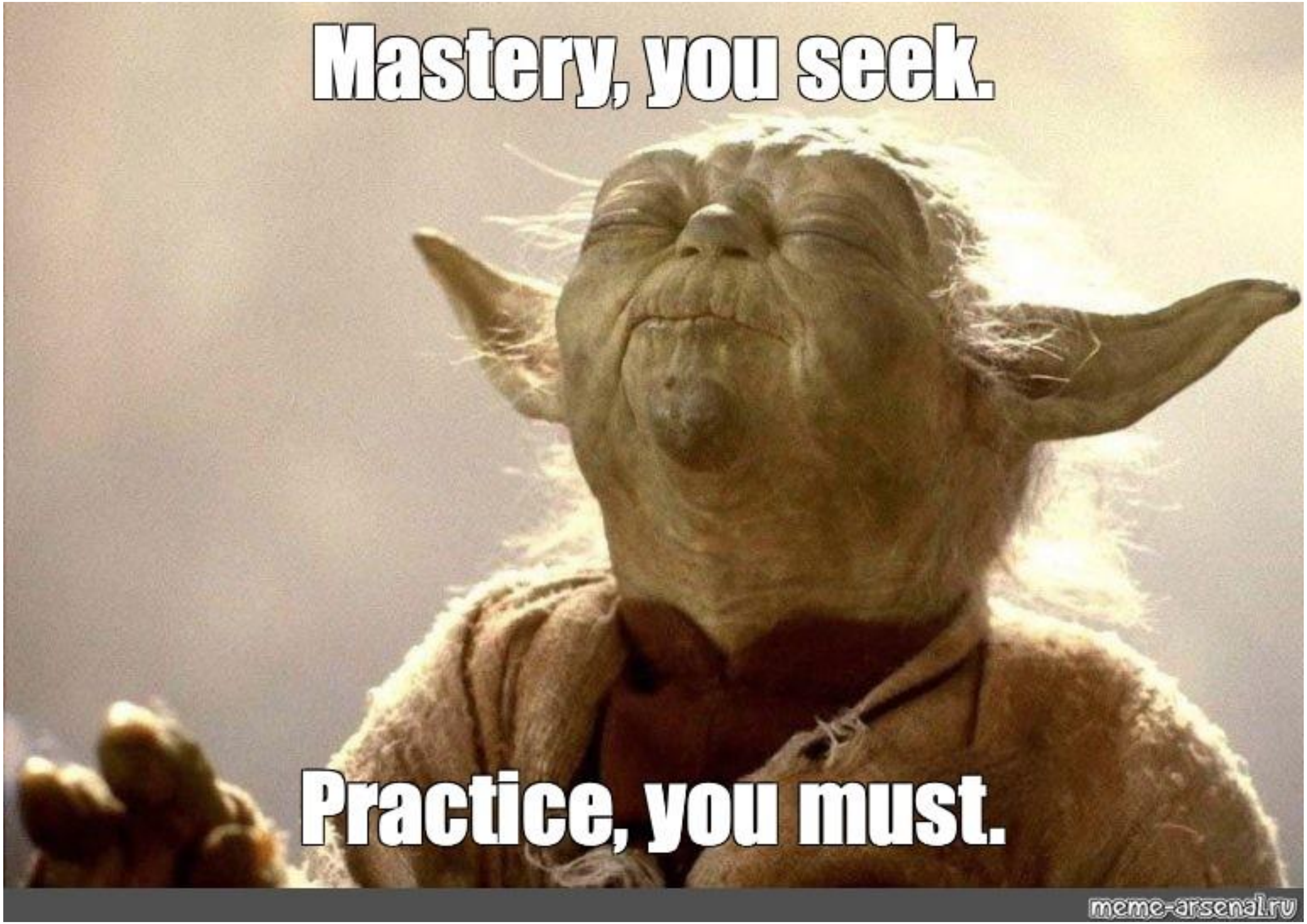
DATAQUEST BLOG

28 DECEMBER 2016 / DATA CLEANING

Preparing and Cleaning Data for Machine Learning

- 1) Examining the Data Set
- 2) Narrowing down columns manually
  - Remove Id's
  - Irrelevant variables
  - Remove zipcode & date
  - Temporal infidelity (data from future)
  - Calculated variables
  - Decide target
    - Select studied cases
    - Distribution of target variables
  - Remove flat values
- 3) Preparing features for ML
  - Preview data
  - Handling missing values
    - Drop unqualified features
  - Investigate categorical features
    - Drop too many unique values (treat as Id)
    - Convert ordinal to numeric
    - Convert categorical to numeric
  - Check all numeric variables
- 4) Other preprocessing steps:
  - Train/Test/Validate

https://www.dataquest.io/blog/machine-learning-preparing-data/


Mastery, you seek.
Practice, you must.
meme-arsenal.ru

+

Any questions? ☺