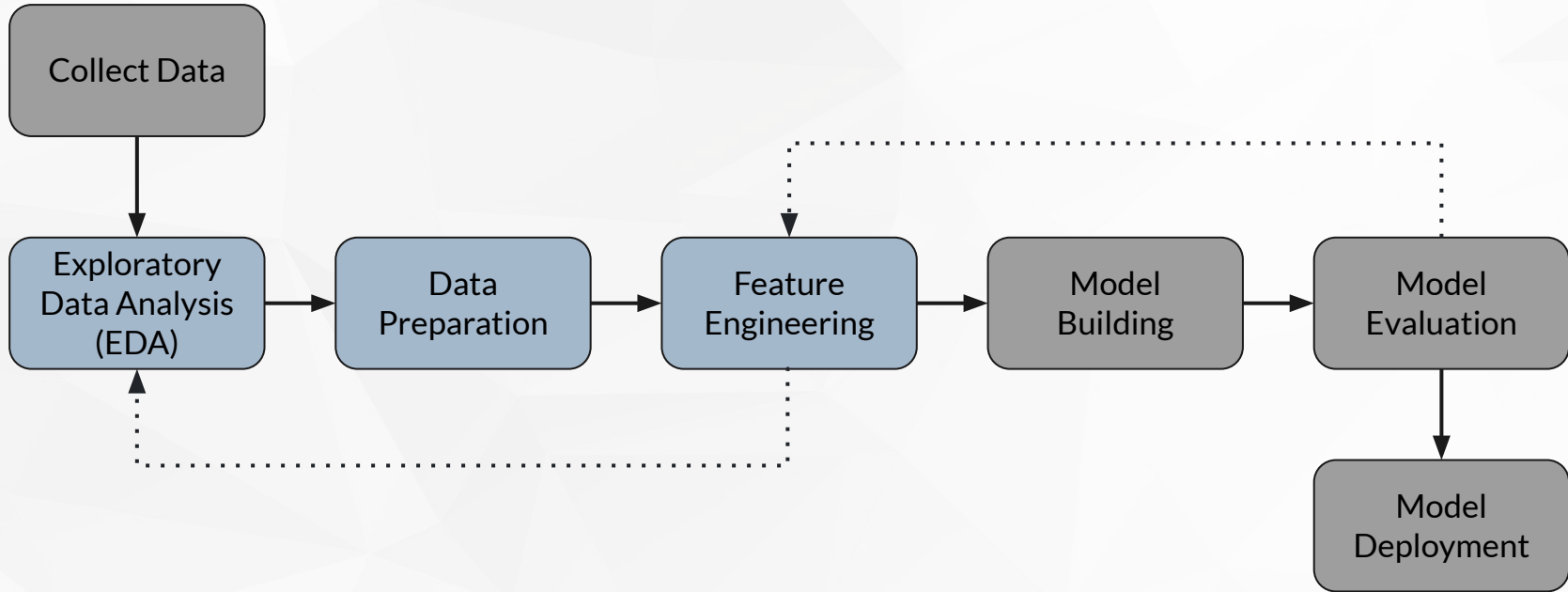


# Getting and Preparing the Right Data

# Data Preparation in the Model Building Process



---

# Exploratory Data Analysis (EDA)



# Exploratory Data Analysis (EDA)

Performing initial investigations on data, with the help of descriptive statistics and graphical representations, so as to

- Describe the data
- Verify data quality
- Spot anomalies
- Discover patterns and derive initial insights
- Test hypothesis
- Check Assumptions



# EDA : Describe the Data

Dimensionality	<code>df.shape</code> <code>df.info()</code>
Feature Names	<code>df.info()</code> <code>df.columns</code>
Data Types	<code>df.info()</code> <code>df.dtypes()</code>
Feature Values & Distribution <i>Note: for classification take particular note of Label distribution</i>	<code>df.describe()</code> <code>df.column.value_counts()</code> <code>df.column.unique()</code> <code>df.column.nunique()</code> Various charts



# EDA : Verify Data Quality

Look out for:

- Incorrect or unexpected data type & format (ALL)
- Duplicate Samples (ALL)
- Unexpected dimensions (i.e. missing rows or columns) (ALL)
- Incorrect Spelling (CAT)
- Mixed cases for strings (CAT)
- Unexpected outliers or anomalous values (NUM)
- Inconsistent or incorrect units of measurement (NUM)



# EDA : Find Patterns, Spot Anomalies, Test Hypothesis using Descriptive Statistics

- Univariate Plots
  - Histograms
  - Density Plots
  - Box Plots
- Multivariate Plots
  - Scatter Plot
  - Crosstabs and Pivot tables
- Plotting on different scales - i.e. log scale
- Time-series charts



# EDA : Check Assumptions

EDA allows you to look at the data before assuming anything about it

- Technical - Algorithm requirements like -
  - Multi-collinearity
  - Variables are independent
- Business - Check assumptions that have been made to frame the problem like
  - identify potential causes for behavior
  - identify potentially problematic or spurious data points
  - develop hypotheses to test that will inform their analysis and model development strategy



# LAB: Exploratory Data Analysis

---

# Data Preparation



# Data Understanding & Preparation

- Describe the Data
- Verify Data Quality
  - Proper Labels
  - Missing Values
  - Format of Values
  - Spelling & Case
  - Deviations / Outliers
  - Duplicates
- Explore the Data
- Feature Engineering
  - Combining Features
  - Dropping Features



# Data Preparation

## Overview

The data preparation phase covers all activities to construct the final dataset (data that will be fed into the modeling tool(s)) from the initial raw data. Data preparation tasks are likely to be performed multiple times, and not in any prescribed order.

### TASKS:

- Table, record, and attribute selection and transformation
- Cleaning of data for modeling tools.



# Data Preparation

- Ensure Columns / Features have Proper Labels
- Remove Duplicates
- Handle Missing Values and Blank Fields
- Correct Spelling & Case
- Ensure Correct Format of Values
  - Dates are in ISO 8601 format
  - Numbers use Float format
  - Remove currency sign and comma for numbers corresponding to money/amount
- Remove Outliers



# DP: Ensure Columns / Features have Proper Labels

All columns in the dataset should be properly labeled to describe its values.

Example:

Correct	Incorrect	Reason
Class_Column	Class Column	Includes newline which may cause error in reading the data
Transaction_Date	Col 0	Non-descriptive Label
Barcode	Bar,code	Contains Erroneous Characters



## DP: Remove Duplicates

If there are rows that contain the same set of values for all columns, these are considered redundant and it is suggested that such columns are removed. The cause of duplicate data errors range from human error or system generated errors.

The following is an example of a dataset containing duplicate rows (particularly the 1st and 3rd rows):

ID	Date_and_Time	Item_ID	Amount	Discount
1	2007-04-05T14:30:55Z	199	\$10.61	\$0.00
2	2007-04-05T14:31:05Z	135	\$15.05	\$15.05
1	2007-04-05T14:30:55Z	199	\$10.61	\$0.00



# DP: Handle Missing Values and Blank Fields

If there are particular rows that contain missing values for some columns, there are options for removing them:

- Replace missing values with default values.
  - For example, if the “Occupation” is empty for certain rows, replace all such instances with a default value like “No Work” or “N/A”.
- Remove all rows with missing values. This, however, is not recommended because we will lose data by performing this step.
- Remove columns with a high percentage of null values
- Use attribute mean or mode (either overall mean or mean for the same class the row belongs in) or impute
- Leave the implementation to handle missing values



# DP: Correct Spelling & Case

Make sure that string values are consistent throughout the dataset. That is, there should only be one string representation for a particular entity/object.

Example:

- If “personal computer”, “Personal Computer”, “pc” and “PC” all refer to a single product type, then one can replace all instances of these values with “personal computer”.
- (Instead of choosing “personal computer”, you may opt to replace all with another value like “pc”. The point is that all of them should have a single string representation.)

ID	Student_Name	Computer_Type	Brand
1	John	PC	APPLE
2	Sarah	Personal Computer	lenovo
3	Jennifer	pc	Apple

## DP: Ensure Correct Format of Values

- Dates are in ISO 8601 format
- Numbers used for Identification (i.e. UserID, Item\_No, SKU\_Number) should be **object** dtype
- Numbers use **Float** format
- Remove currency sign and comma for numbers corresponding to money/amount

Field Name	dtype	X	✓
Transaction_Date	datetime	September 15, 2018	2018-09-15T15:53:00
SKU_No	object	2,000,550	2000550
Amount	float	\$1,000.50	1000.50

# DP: Remove Outliers

- An observation point that is distant from other observations.
- An observation that appears far away and diverges from an overall pattern in a sample.

Example 1:



Example 2:

- Given the following scores: 25, 29, 3, 32, 85, 33, 27, 28
  - both 3 and 85 are "outliers"



# Data Preparation

- Handle missing values
- Handle outliers
- Handle duplicates
- Transforming to Numeric - Category to Number / Vectorization
- Rescaling
- Normalization / Standardization

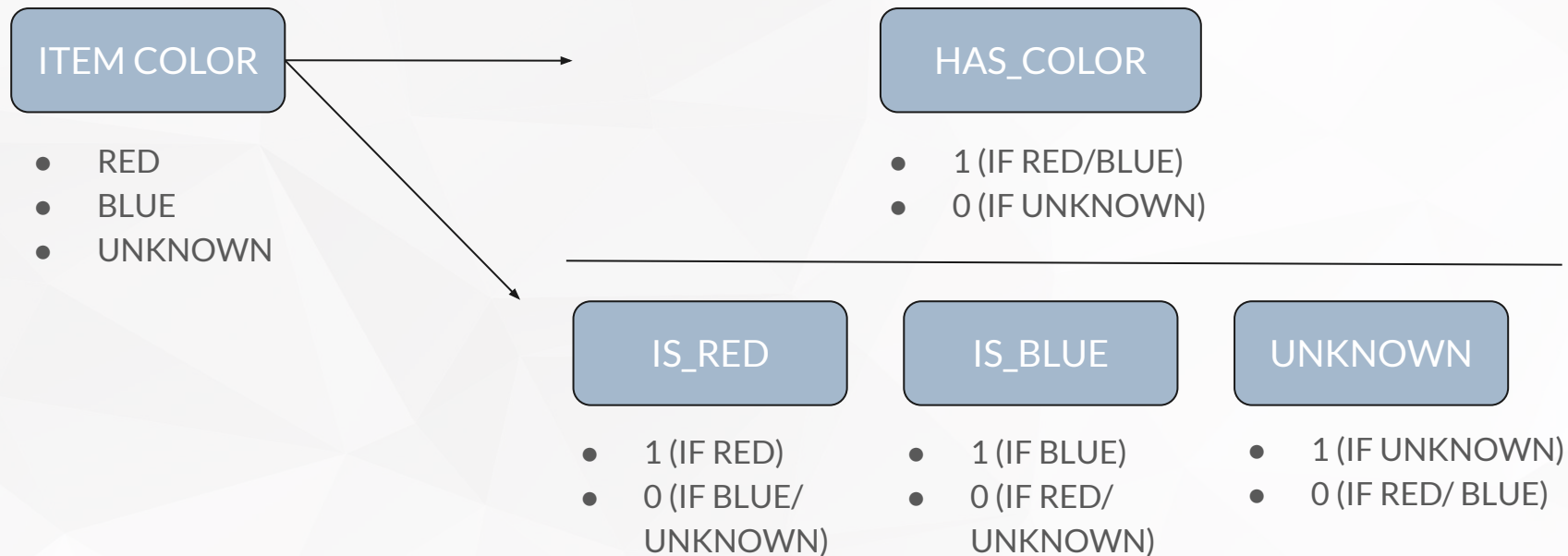


## DP: Categorical Attributes

- Imagine you have a categorical attribute, like “Item\_Color” that can be Red, Blue or Unknown.
- Unknown may be special, but to a model, it looks like just another colour choice. It might be beneficial to better expose this information.
- You could create a new binary feature called “Has\_Color” and assign it a value of “1” when an item has a color and “0” when the color is unknown.
- Going a step further, you could create a binary feature for each value that Item\_Color has. This would be three binary attributes: Is\_Red, Is\_Blue and Is\_Unknown.
- These additional features could be used instead of the Item\_Color feature (if you wanted to try a simpler linear model) or in addition to it (if you wanted to get more out of something like a decision tree).

# Category to Number

- Also known as Vectorization or One-to-Many





# Data Preparation

- Handle missing values
- Remove outliers
- Handle duplicates
- Feature Scaling
- Feature Engineering
- Feature Selection



# Data Preparation: Feature Scaling

- Feature scaling is a method used to standardize the range of independent variables or features of data. In data processing, it is also known as data normalization and is generally performed during the data pre-processing step.
- Methods
  - Standard Scaler
  - MinMax Scaler
  - Robust Scaler
  - Normalizer

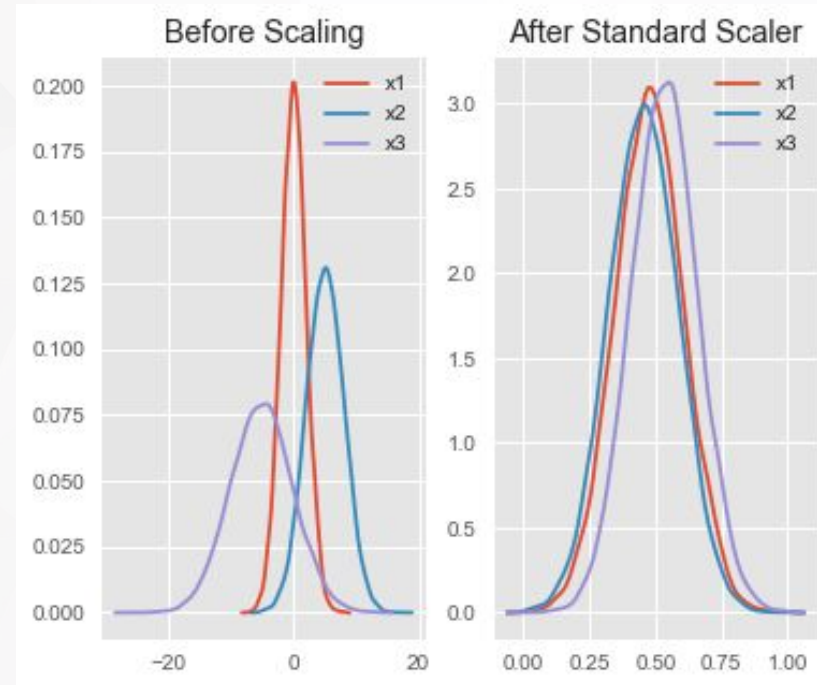


# Feature Scaling: Standard Scaler

- Assumes your data is normally distributed within each feature and will scale them such that the distribution is now centred around 0, with a standard deviation of 1.
- The mean and standard deviation are calculated for the feature and then the feature is scaled based on:

$$\frac{x_i - \text{mean}(x)}{\text{stdev}(x)}$$

- If data is not normally distributed, this is not the best scaler to use.

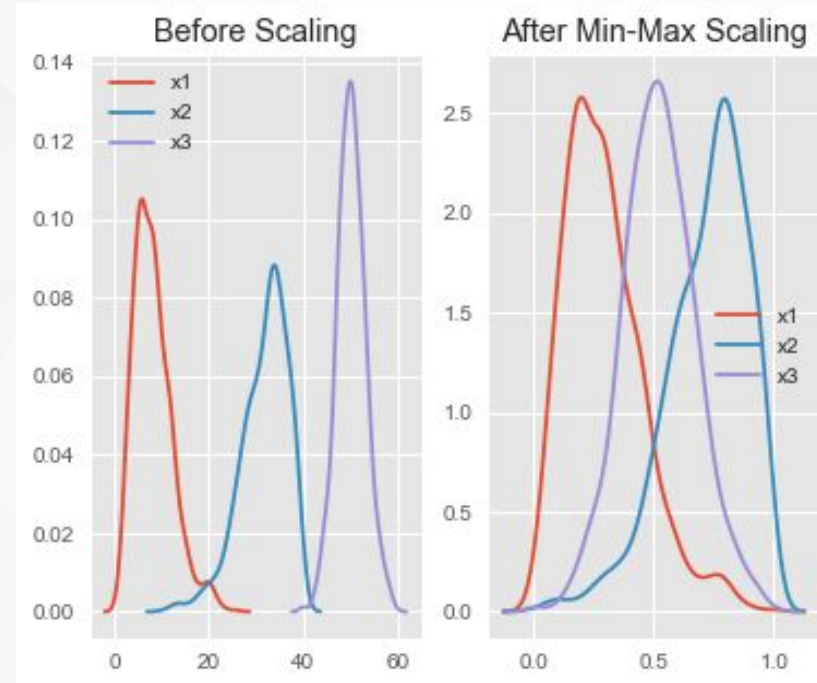


# Feature Scaling: Min-Max Scaler

- One of the most well-known scaling algorithm, and follows the following formula for each feature:

$$\frac{x_i - \min(x)}{\max(x) - \min(x)}$$

- It essentially shrinks the range such that the range is now between 0 and 1 (or -1 to 1 if there are negative values).
- Works better than Standard Scaler for cases in which the distribution is not Gaussian or the standard deviation is very small
- However, it is sensitive to outliers, so if there are outliers in the data, consider the Robust Scaler.

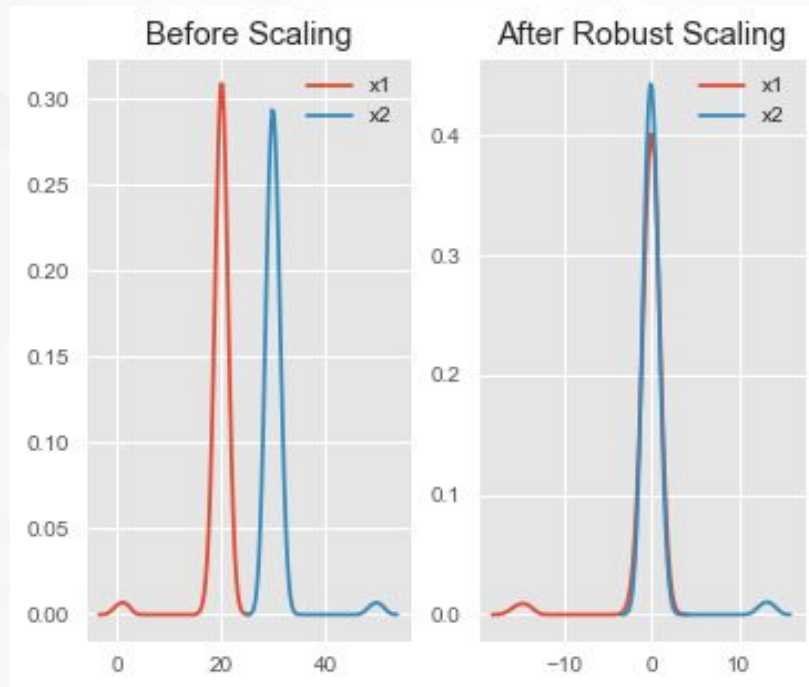


# Feature Scaling: Robust Scaler

- Uses a similar method to the Min-Max scaler but it instead uses the interquartile range, rather than the min-max, so that it is robust to outliers.
- Follows the formula:

$$\frac{x_i - Q_1(x)}{Q_3(x) - Q_1(x)}$$

- Notice that after Robust scaling, the distributions are brought into the same scale and overlap, but the outliers remain outside of bulk of the new distributions.

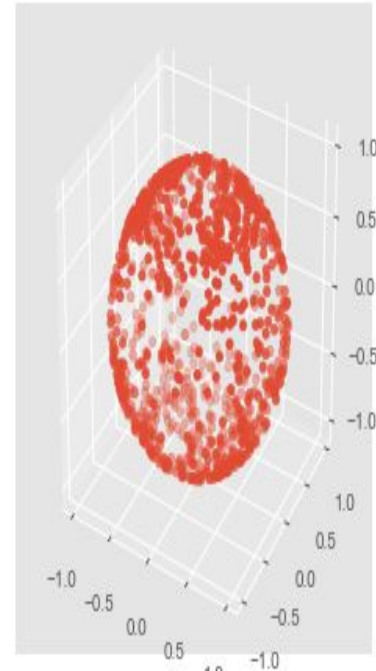
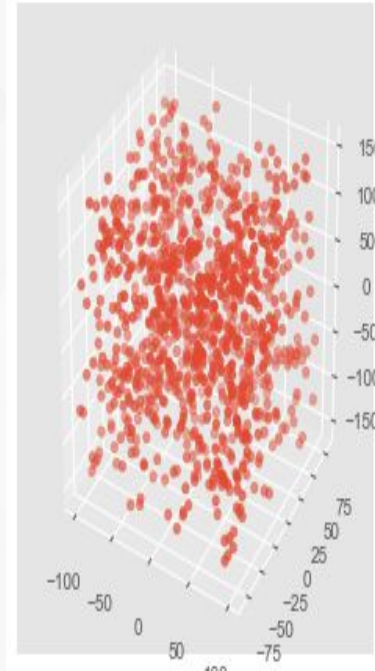


# Feature Scaling: Normalizer


- Scales *per observation* as opposed to the other Scalers which scale *per feature*
- Scales each value by dividing each value by its magnitude in n-dimensional space for n number of features.
- Say your features were x, y and z Cartesian co-ordinates your scaled value for x would be:

$$\frac{X_i}{\sqrt{(x_i^2 + y_i^2 + z_i^2)}}$$

- Each point is now within 1 unit of the origin on this Cartesian co-ordinate system.



# LAB: Data Preparation with Python



# Feature Engineering

## Overview

The process of using domain knowledge of the data to create features that make machine learning algorithms work.

Feature engineering is manually designing what the input  $x$ 's should be

Importance:

- Better features means flexibility.
- Better features means simpler models.
- Better features could mean better results.



# Feature Engineering: Process

## Brainstorm Features

Really get into the problem, look at a lot of data, study feature engineering on other problems and see what you can borrow.

## Devise Features

Depends on your problem, but you may use manual feature construction, automatic feature extraction, or a combination of the two.

## Select Features

Use different feature importance scorings and feature selection methods to filter out unimportant or “extra” features.

## Evaluate Models

Estimate model accuracy on unseen data using the chosen Features.

*You need a well defined problem so that you know when to stop this process and move on to trying other models, other model configurations, ensembles of models, and so on.*

*You need a well considered and designed test harness for objectively estimating model skill on unseen data. It will be the only measure you have of your feature engineering process, and you must trust it not to waste your time.*

# Feature Engineering: Numerical Attributes

- Your data is very likely to contain quantities, which can be reframed to better expose relevant structures.
- You may have a quantity like weight, distance or timing. A linear transform may be useful to regression and other scale dependent methods.

Example: Feature = Item\_Weight\_grams ; Value = 6289

Options	Transformation	Feature_Name	Output
1	Expressed in Kilograms	Item_Weight_kg	6.289
2	Expressed in Kg (Rounded)	Item_Weight_kg	6
3	Split into 2 Features	<ul style="list-style-type: none"><li>• Item_Weight_kg</li><li>• Item_Weight_remainder_grams</li></ul>	<ul style="list-style-type: none"><li>• 6</li><li>• 289</li></ul>





# Feature Engineering: Numerical Attributes

More meaningful discrete features can also be created from numerical attributes.

Examples:

Raw_Feature	Rules/Information	New Feature	Possible Values
Item_Weight_grams	Items above 4kg incur a higher tax rate	Item_Above_4kg	(0, 1); (Yes/No)
Total_Amount	Values > 15,000 indicate a high value customer	High_Value_Customer	(0, 1); (Yes/No)
Average_Monthly_Visits	Values > 20 indicate a high frequency customer	High_Frequency_Customer	(0, 1); (Yes/No)
%_Change_Stock_Price	Any change (+ or -) > 5% indicates movement	Stock_Price_Movement	(Up, No_Change, Down)

# Feature Engineering: Numerical Attributes

- Numerical quantities can also be stored as a rate or an aggregate quantity for an interval.
  - I.e. Total number of customer purchases aggregated over the year
- Existing features can also be combined/computed to create new features
  - I.e. Financial Ratios (Profit Margin, Return on Equity, Earnings Per Share)

Raw_Feature	Transformation	New Feature	Possible Values
<ul style="list-style-type: none"><li>• Transaction_ID</li><li>• User_ID</li></ul>	Count of Unique Transaction ID per User ID ("Groupby")	Num_Cus_Purchases	Integer $\geq 0$
<ul style="list-style-type: none"><li>• Transaction_ID</li><li>• User_ID</li><li>• Season</li></ul>	Count of Unique Transaction ID per User ID per Season ("Groupby")	<ul style="list-style-type: none"><li>• Num_Cus_Purchases_Summer</li><li>• Num_Cus_Purchases_Fall</li><li>• Num_Cus_Purchases_Winter</li><li>• Num_Cus_Purchases_Spring</li></ul>	Integer $\geq 0$
<ul style="list-style-type: none"><li>• Gross Profit</li><li>• Net Sales</li></ul>	Compute for Gross Profit Margin: Gross Profit / Net Sales	Gross_Profit_Margin	Float $\leq 1$

# Feature Engineering: Date-Time

TIMESTAMP

2014-09-20T20:45:40Z

HOUR\_OF\_DAY

*Numeric (Int)*  
1,2,3,4,5,6,7,8...24

DAY\_OF\_WEEK

*Categorical (Str)*  
Monday, Tuesday,  
Wednesday,  
Thursday, Friday,  
Saturday, Sunday

PART\_OF\_DAY

*Categorical (Str) or  
Ordinal (Int)*  
Morning / 1  
Midday / 2  
Afternoon / 3  
Night / 4  
Early Morning / 5

Based on defined logic:  
I.e. if HOUR\_OF\_DAY > 7  
&& HOUR\_OF\_DAY < 12

MONTH

YEAR

WK\_OF\_YEAR

IS\_WEEKEND

IS\_PEAK\_PD



# Feature Engineering: Other Examples

Features on Dataset	Derived Features
<ul style="list-style-type: none"><li>● Customer Info<ul style="list-style-type: none"><li>○ Customer ID</li><li>○ Birthdate</li></ul></li><li>● Transaction Info:<ul style="list-style-type: none"><li>○ Date</li><li>○ Product</li><li>○ Price</li><li>○ Quantity Bought</li></ul></li><li>● Product Info<ul style="list-style-type: none"><li>○ Product Category / Sub-class</li></ul></li></ul>	<ul style="list-style-type: none"><li>● Age</li><li>● Customer Total Spend</li><li>● Customer Spend per Visit</li><li>● Total Number of Visits</li><li>● Frequency of Visits</li><li>● Lifetime (Duration from First to Last)</li><li>● % of Weekend &amp; Weekday Shopping Days</li><li>● Number of Categories Bought per Visit (Variety)</li><li>● High Vs Low Value Shopper: % of High, Medium, and Low Priced Items per Category bought</li></ul>

# LAB: Feature Engineering



# Feature Reduction

## Overview

- The process of reducing the number of variables under consideration, via obtaining a set of principal variables.
- Obtains reduced representation in volume but produces the same or similar analytical results
- Can be divided into :
  - Feature Selection
  - Feature Extraction
- Why? Curse of Dimensionality
  - when the dimensionality increases, the available data becomes sparse.



# Feature Reduction: Feature Selection

- It is the automatic selection of attributes that are most relevant to the predictive modeling problem you are working on.
  - Also called variable selection or attribute selection.
  - It mostly acts as a filter, muting out features that aren't useful in addition to your existing features.
  - Fewer attributes is desirable because it reduces the complexity of the model, and a simpler model is simpler to understand and explain.
  - Less complex models usually also lead to better results
- Some Feature Selection criteria / techniques:
  - High Missing Values
  - Low Variance
  - High Correlation with other Data Columns
  - Random Forest / Tree Ensembles Feature Importance

# Feature Selection: High Missing Values

- Data columns with too many missing values are unlikely to carry much useful information.
- Data columns with number of missing values greater than a given threshold can be removed, the higher the threshold, the more aggressive the reduction
- Need to find the optimal threshold
  - Too aggressive a value reduces dimensionality at the expense of performance
  - A too soft value might not get the best reduction ratio

Feature	Total Samples in Data	Missing Data	% Missing
• User_ID	1,000	100	10%
• City	1,000	800	80%
• Zip_Code	1,000	600	60%





# Feature Selection: Low Variance

- Data columns with little changes in the data carry little useful information.
- Data columns with variance lower than a given threshold are removed
- Variance is range dependent; recommend to normalize before running this function
- Variance should be computed on numerical, scaled data.

Transaction_ID	Location	Amount	Currency
001	Manila	10	USD
002	Manila	10	USD
003	Manila	20	USD
004	Manila	10	USD
005	Ortigas	10	USD

# Feature Selection: Constant Values

- Columns that contain only 1 value for all rows can be discarded.
  - Example: if a dataset contains “location” as one of the columns and the value of this column is “Manila” for all rows, this column can be removed.
  - Example: if a dataset contains “currency” as one of the columns and the value of this column is “USD” for all rows, this column can be removed.

Transaction_ID	Location	Amount	Currency
001	Manila	10	USD
002	Manila	10	USD
003	Manila	20	USD
004	Manila	10	USD
005	Ortigas	10	USD

# Feature Selection: High Correlation

- Refers to High Correlation with other data columns.
- Data columns with very similar trends are also likely to carry very similar information, Any one of them could suffice to feed the machine learning model
- Calculate the correlation coefficient between numerical columns (using Pearson's Product Moment Coefficient) and between nominal columns (using Pearson's chi-square value)
- Pairs of columns with correlation coefficient higher than a threshold are reduced to only one
- Correlation is scale sensitive, normalization/standardization is required

Year_Born	Age	Gross_Amount	Discount	Net_Amount
1990	28	100	0	100
1995	23	80	0	80
2015	3	200	0	200



# Feature Selection: Random Forest and Ensemble Tree

- Decision Tree Ensembles, also referred to as random forests, are useful for feature selection in addition to being effective classifiers.
- Generate a large and carefully constructed set of trees against a target attribute, and then use each attribute usage statistics to find the most informative subset of features.
- Implementation in sklearn (`feature_importances_`). The higher the value, the more “important” the feature.

Feature	Rank	Importance
Income	1	0.76664038
Age	2	0.14205973
Number_of_Dependents	3	0.0282433



# Feature Selection: Other Techniques

- Forward Selection
  - Forward selection is an iterative method in which we start with having a single feature in the model. In each iteration, we keep adding the feature which best improves our model till an addition of a new variable does not improve the performance of the model.
- Backward Elimination
  - In backward elimination, we start with all the features and removes the least significant feature at each iteration which improves the performance of the model. We repeat this until no improvement is observed on removal of features.

## Rarely used

- Time and computationally expensive.
- Are practically only applicable to a data set with an already relatively low number of input columns.



# Feature Extraction: Overview

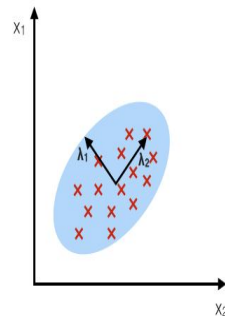
- Transforms data in high dimensionality space to a space of fewer dimensions that preserves (most of) the information and structure of the data
- Linear:
  - Principal Component Analysis (PCA)
  - Linear Discriminant Analysis
- Non Linear:
  - Kernel PCA
  - 1st Layer of Networks

# Feature Extraction

- A new reduced set of variables is created by applying a mapping of the multidimensional space into a space of fewer dimensions. This means that the original feature space is transformed into a reduced, although informative space.
- Signal Representation vs Classification
  - Signal Representation - PCA
    - The goal of feature extraction is to represent the samples accurately in a lower-dimensional space
  - Classification
    - The goal of feature extraction is to enhance the class-discriminatory information in the lower-dimensional space

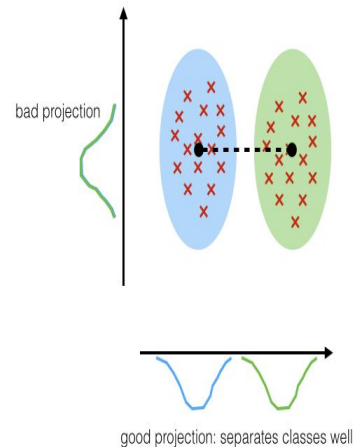
## PCA:

component axes that maximize the variance



## LDA:

maximizing the component axes for class-separation

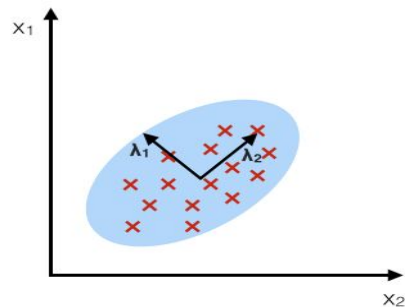


# Feature Extraction: Principal Component Analysis (PCA)

- A method of Feature Extraction
- extracts *low dimensional* set of features *from a high dimensional* data set with a motive to *capture as much information as possible*
- Used for model improvement as well as visualization

## PCA:

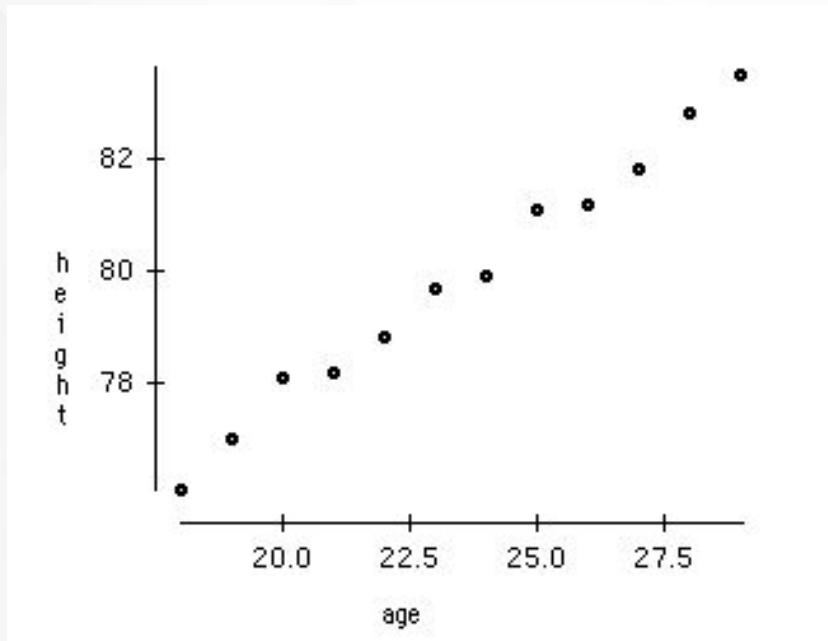
component axes that maximize the variance



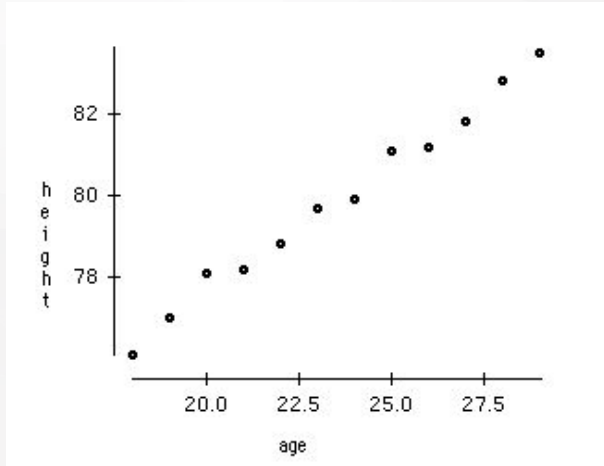


# Feature Extraction: PCA for Visualization Example

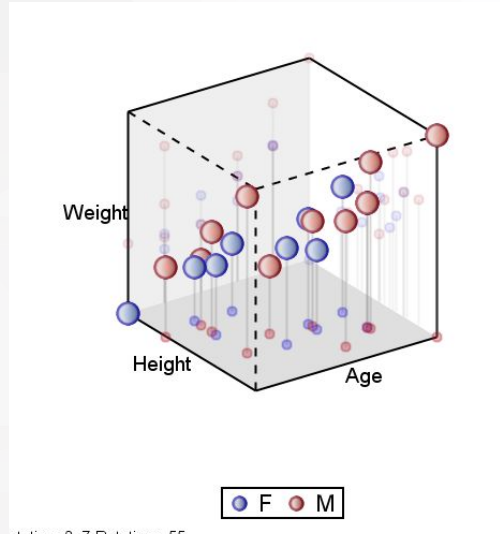
Given 2 variables: Age & Height



# Feature Extraction: PCA for Visualization Example



Plotting 2 variables

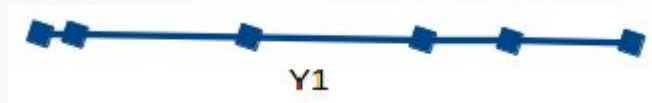
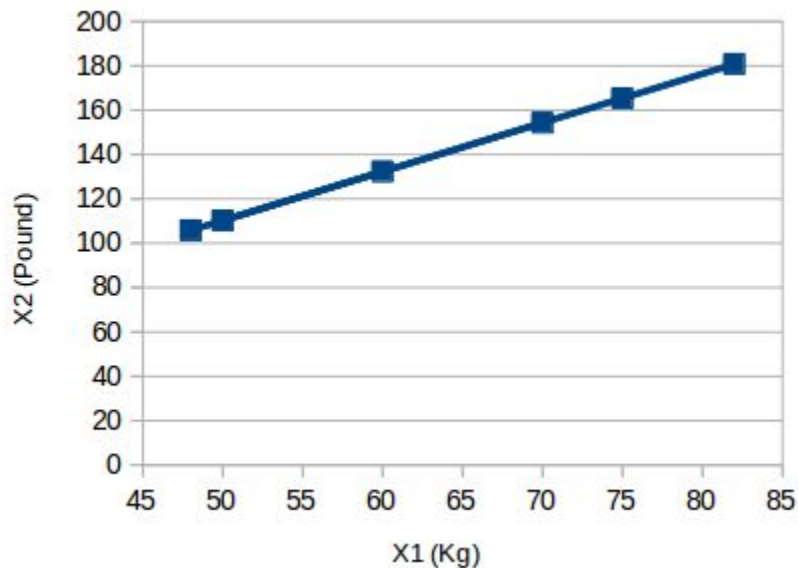


Plotting 3 - 4 variables

How do we plot more variables??

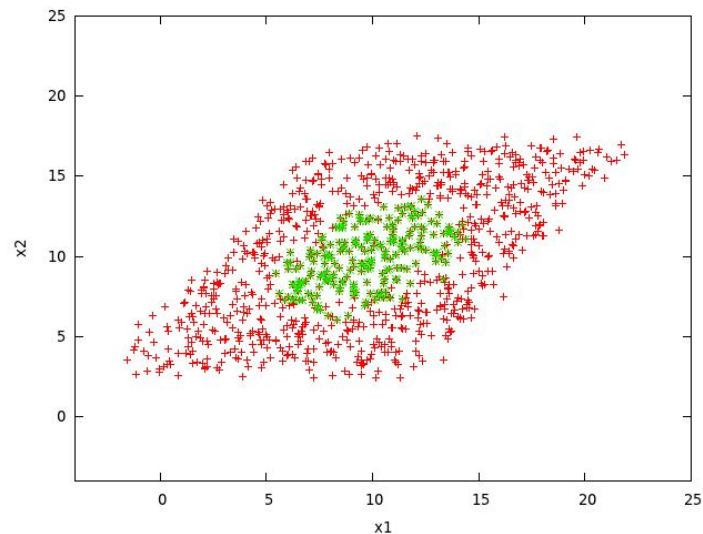
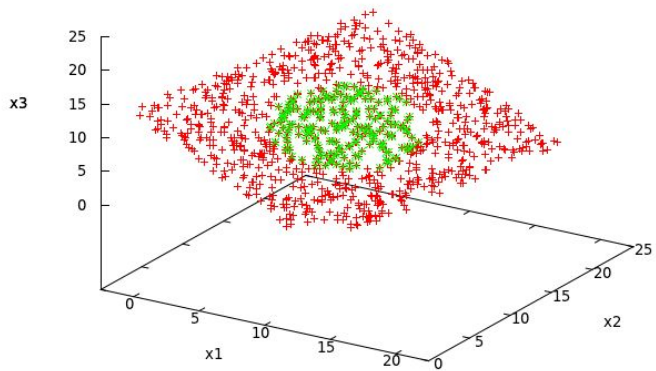
# Feature Extraction: PCA

- Objective: Reduce the number of dimensions in a dataset whilst retaining most information



# Feature Extraction: PCA

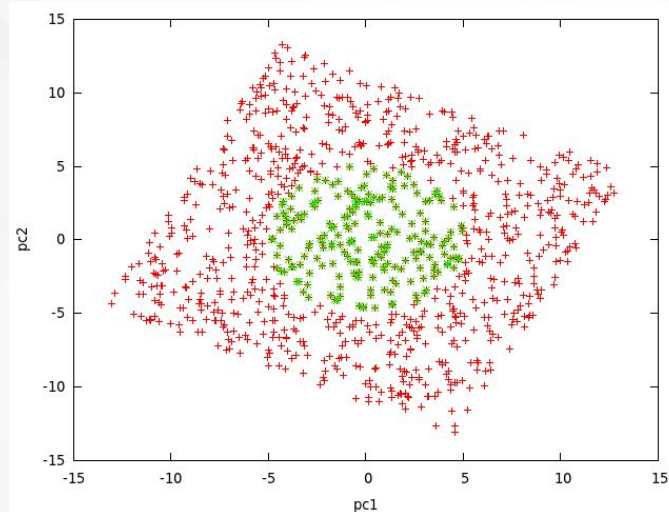
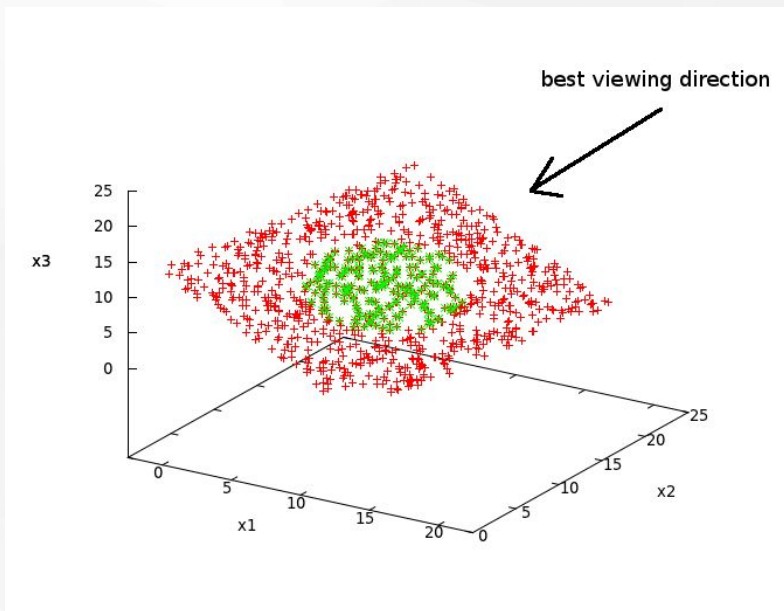
Suppose we have data with 3 features, simulating a square with a circle in the middle



But projecting just using 2 of the variables, it appears skewed

# Feature Extraction: PCA

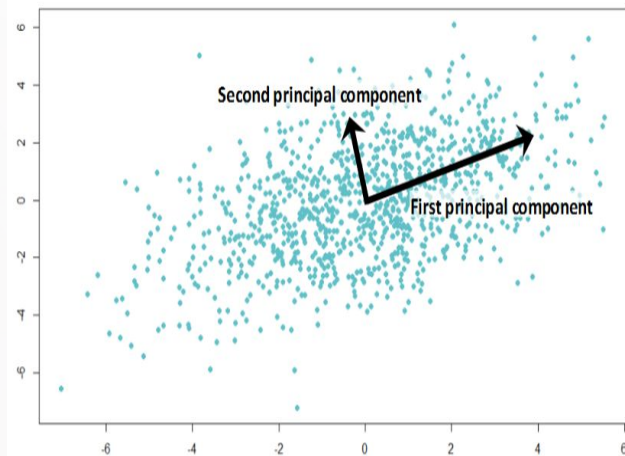
The best projection is when we're looking directly / perpendicular to the plane



- With PCA, you can see the square and the circle clearly
- PCA essentially "rotates" the data, putting it on a new set of axes/principal components : PC 1, PC 2, PC 3...

# Feature Extraction: PCA

- A **principal component** is a *normalized linear combination* of the original predictors in a data set.
- The first principal component is a linear combination of original predictor variables which ***captures the maximum variance*** in the data set → The larger the variability, the larger the information captured by the component
- The second and succeeding principal components capture the remaining variation without being related to the previous component (Correlation = 0)
- **Explained Variance** - how much information (variance) can be attributed to each of the principal components.
- E.g. if PCA1 has Explained variance of 72% and PCA2 has explained variance of 23%, PCA1 and PCA2 capture 95% of the information



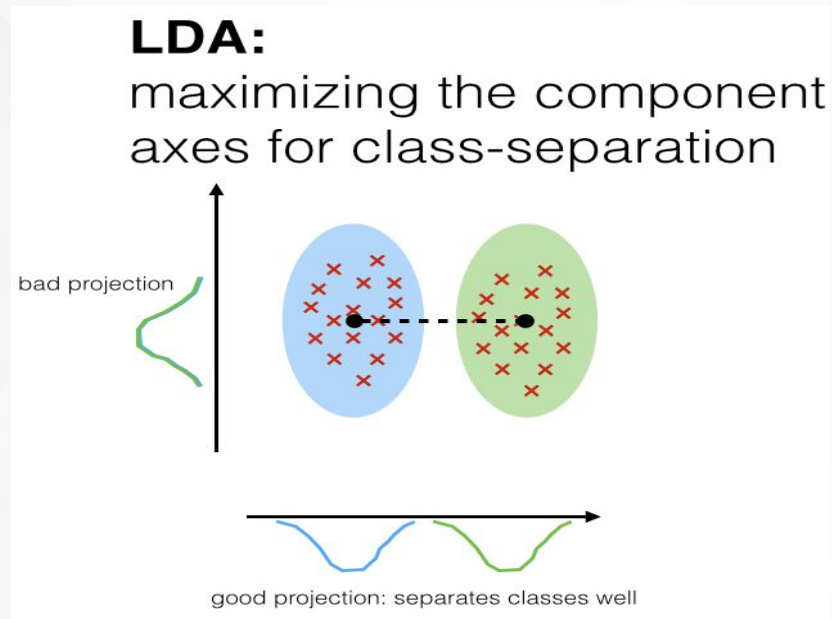


# Feature Extraction: PCA - Reminder to Normalize before PCA

- The original variables might be in different scales
- Performing PCA on un-normalized variables will lead to insanely large loadings for variables with high variance;
- This will lead to dependence of a principal component on the variable with high variance
- Solution: The principal components are supplied with normalized version of original predictors/input variables

# Feature Extraction: Linear Discriminant Analysis (LDA)

- A method of Feature Extraction
- extracts *low dimensional* set of features *from a high dimensional* data set
- Like PCA. but focuses on maximizing the separability among known categories
- Used for model improvement as well as visualization

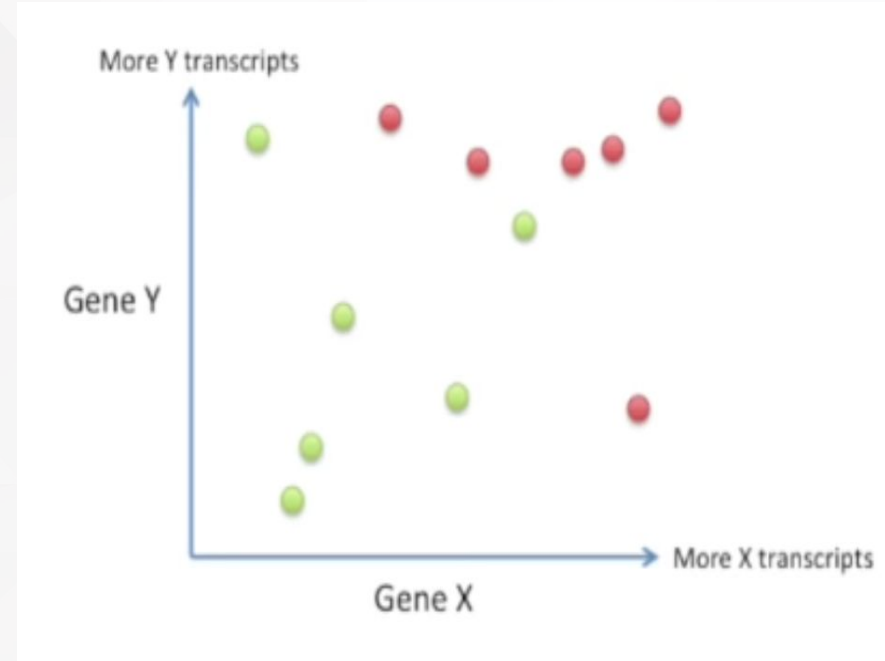




# Feature Extraction: LDA Example

For a certain drug:

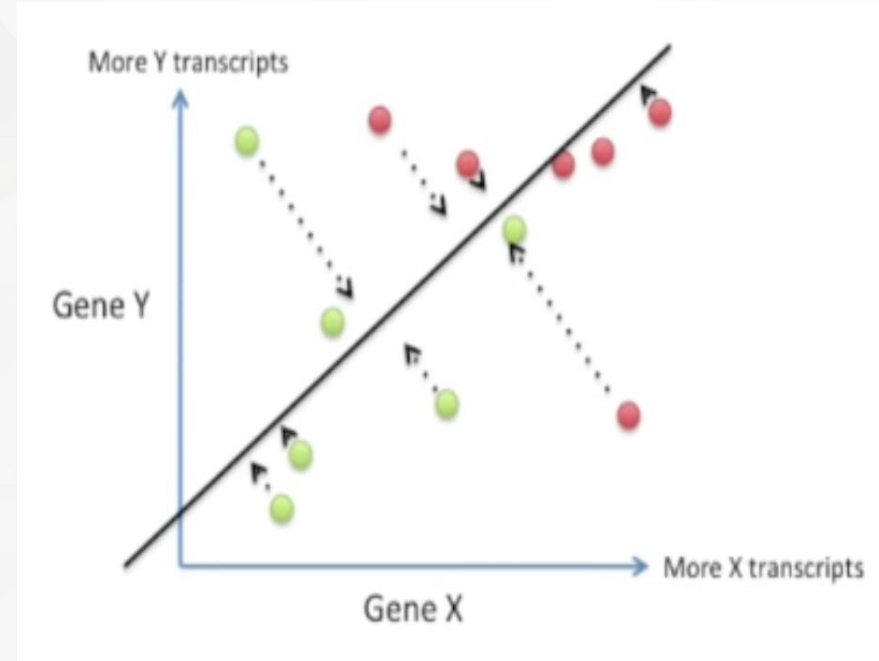
- Given 2 variables:
  - Gene X
  - Gene Y
- Given 2 Categories:
  - Green = Positive Effect
  - Red = Negative Effect



# Feature Extraction: LDA Example

For a certain drug:

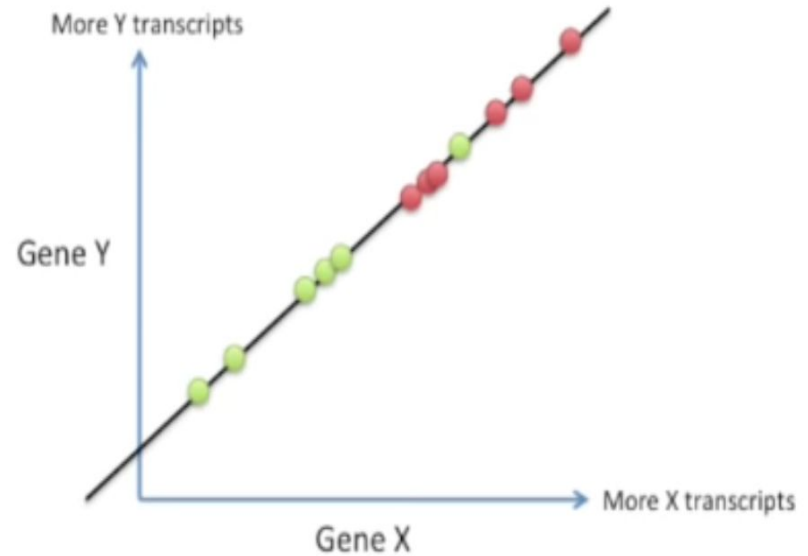
- Given 2 variables:
  - Gene X
  - Gene Y
- Given 2 Categories:
  - Green = Positive Effect
  - Red = Negative Effect



# Feature Extraction: LDA Example

For a certain drug:

- Given 2 variables:
  - Gene X
  - Gene Y
- Given 2 Categories:
  - Green = Positive Effect
  - Red = Negative Effect
- Criteria
  - Maximize the distance between means of the two classes
  - Minimize the variation within each category





# Feature Extraction: PCA and LDA

- Both rank the new axes in order of importance
  - PC1 - accounts for the most variation in the data
  - LD1 - accounts for the most variation between the categories
- Both let you dig deeper and see which features are informative
  - PCA = `pca.components_`
  - LDA = `lda.coef_`
- Use PCA when categories are unknown or general variation in the data is important
- Use LDA when categories are known or variation between categories is important

# LAB: Feature Extraction

---

# Addendum: Introduction to OpenRefine



# OpenRefine

## Overview

OpenRefine is a powerful tool for working with messy data: cleaning it; transforming it from one format into another; and extending it with web services and external data.

### Uses/Functions:

- Explore Data
- Clean and Transform Data
- Reconcile and Match Data
- GREL (Google Regular Expression Language)
- Functions: Designed to resemble JavaScript and used to transform data programmatically.



# OpenRefine: Tutorial

- Facet Counts
- Quick Edit of Values via Facet
- Transformation (Number;String;Date)
- Trim leading and trailing whitespace
- Clustering



# LAB: Introduction to OpenRefine