DATA PREPARATION: FEATURE ENGINEERING AND SELECTION

Yvonne Phillips

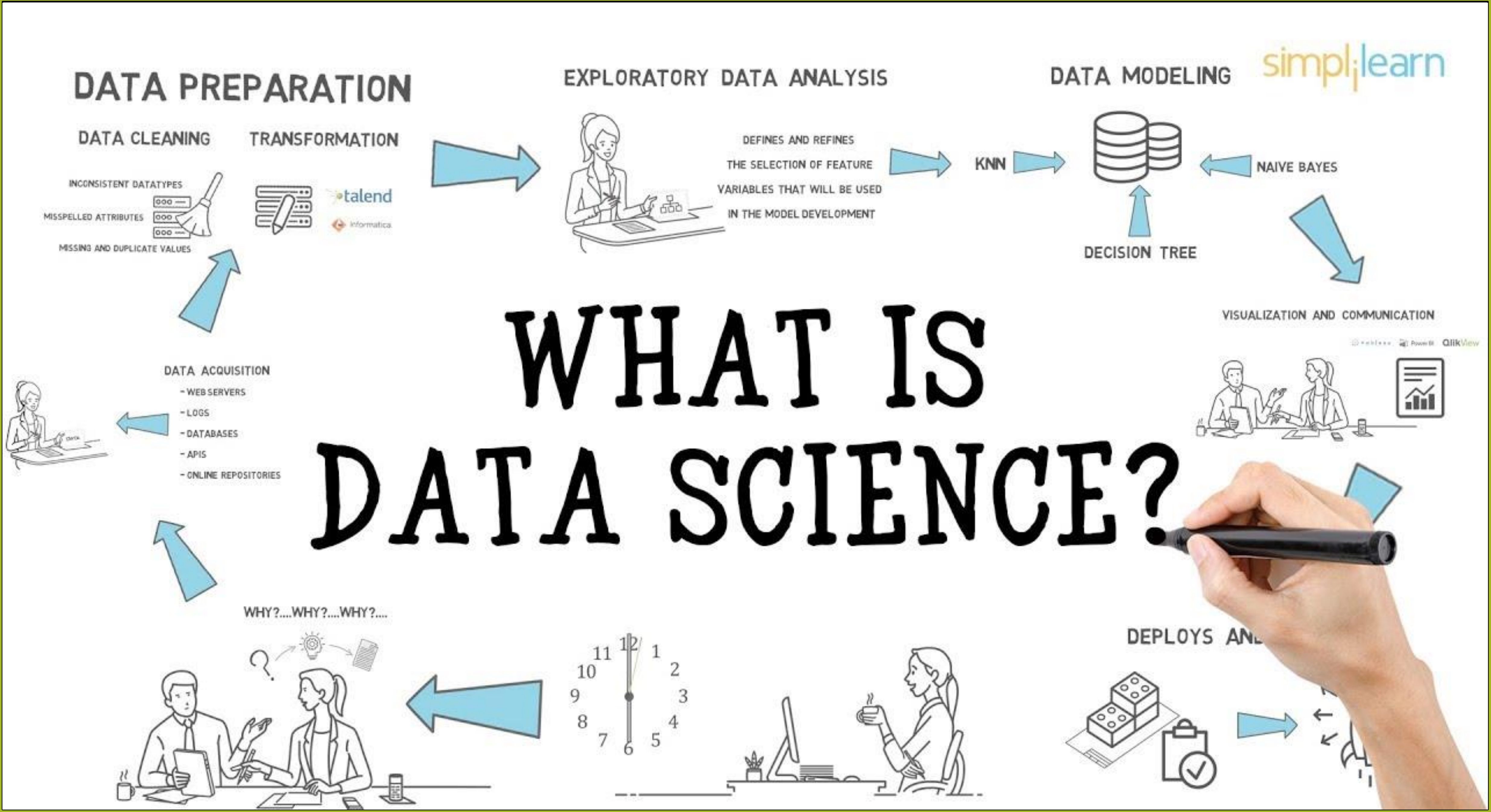
Principal, Analytics Enlightened Consulting LLC

Agenda

- RStudio is now Posit

- Revisit the Data Science Lifecycle
- Data Preparation
- What is Feature Engineering?
- Tips and Best Practices for Feature Engineering
- What is Feature Selection?
- Tips and Best Practices for Feature Selection
- Walking through the steps of a CDC Machine Learning project

WHAT IS DATA SCIENCE?



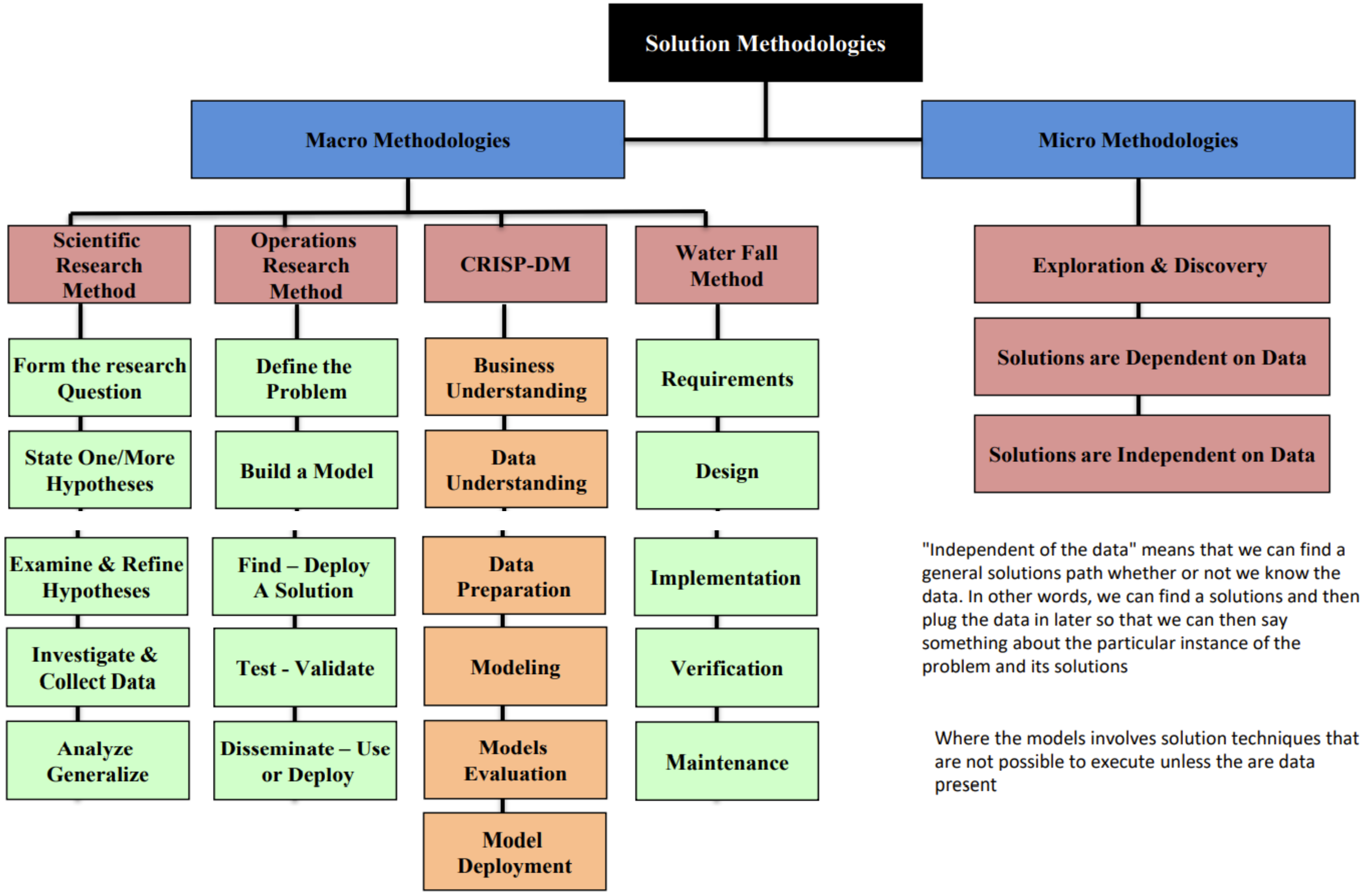
Statistics, Machine Learning and Data Mining

- Statistics:
  - more theory-based
  - more focused on testing hypotheses
- Data Mining and Knowledge Discovery
  - integrates theory and heuristics
  - focus on the entire process of knowledge discovery, including data cleaning, learning, and integration and visualization of results
- Machine learning
  - more heuristic
  - focused on improving the performance of a learning agent
  - also looks at real-time learning and robotics – areas not part of data mining
- Distinctions are fuzzy

Statistics, Machine Learning and Data Mining

- Statistics:
  - more theory-based
  - more focused on testing hypotheses
- Data Mining and Knowledge Discovery
  - integrates theory and heuristics
  - focus on the entire process of knowledge discovery, including data cleaning, learning, and integration and visualization of results
- Machine learning
  - more heuristic
  - focused on improving the performance of a learning agent
  - also looks at real-time learning and robotics – areas not part of data mining
- Distinctions are fuzzy

Technical Solution Methodologies

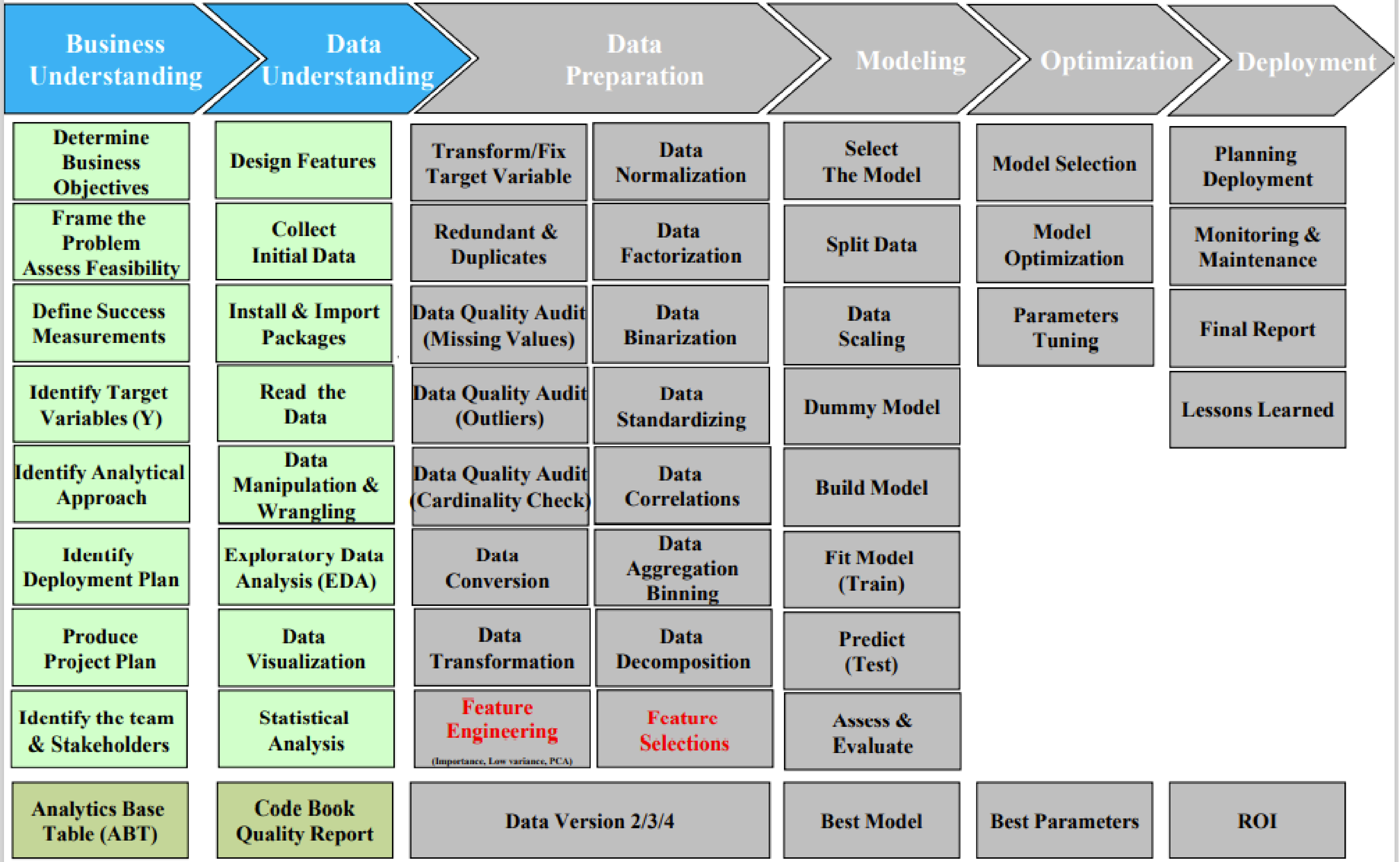


"Independent of the data" means that we can find a general solutions path whether or not we know the data. In other words, we can find a solutions and then plug the data in later so that we can then say something about the particular instance of the problem and its solutions

Where the models involves solution techniques that are not possible to execute unless the are data present

Source: Big Bang Data Science

Data Scientist Day to Day Activities (CRISP-DM)



Business Understanding	Data Understanding	Data Preparation		Modeling	Optimization	Deployment
Determine Business Objectives	Design Features	Transform/Fix Target Variable	Data Normalization	Select The Model	Model Selection	Planning Deployment
Frame the Problem Assess Feasibility	Collect Initial Data	Redundant & Duplicates	Data Factorization	Split Data	Model Optimization	Monitoring & Maintenance
Define Success Measurements	Install & Import Packages	Data Quality Audit (Missing Values)	Data Binarization	Data Scaling	Parameters Tuning	Final Report
Identify Target Variables (Y)	Read the Data	Data Quality Audit (Outliers)	Data Standardizing	Dummy Model		Lessons Learned
Identify Analytical Approach	Data Manipulation & Wrangling	Data Quality Audit (Cardinality Check)	Data Correlations	Build Model		
Identify Deployment Plan	Exploratory Data Analysis (EDA)	Data Conversion	Data Aggregation Binning	Fit Model (Train)		
Produce Project Plan	Data Visualization	Data Transformation	Data Decomposition	Predict (Test)		
Identify the team & Stakeholders	Statistical Analysis	Feature Engineering (Importance, Low variance, PCA)	Feature Selections	Assess & Evaluate		
Analytics Base Table (ABT)	Code Book Quality Report	Data Version 2/3/4		Best Model	Best Parameters	ROI

Source: Big Bang Data Science

Data Quality Report

Data Understanding

Various types of Data:
- Structured Data
- Unstructured Data

Data from Various Sources
- Point of Sales
- Social Media
- External Vendors

Storage Types:
- MySQL, Oracle
- NOSQL (MongoDB)
- Flat Files

Understanding Data
- Variable type, definition and fill rate
- Possible personal variables
- Descriptive statistics
  - ✓ Uni-variate profiling on the data
  - ✓ Questions to understand more about the data

More about Data & Cleansing

Develop Code Book
- Summary of Data
- Type of variable
- Ranges of variables
- Missing fields
- Identify the primary list of variables to solve the business problem

Develop variable hierarchy
- Numerical
- Factor/Categorical
- Latent variables

Data Transformation
- Converting CustID and Code variables into flag variables
- Grouping variables
- Identify the dependent variables
- Identify the target variables

Descriptive Statistics & Variables creation

Types of Central Tendency
- Mean
- Median
- Mode

Normal Distribution

Skewed Distribution

Measures of Variability
- Standard Deviation
- Range

Measure of Relationships
- Scatter plot
- Correlation

Distribution Transformation:
- Cubic
- Square
- Log

Leverage existing variables to create new variables such as (Velocity, Time, Recency)

Data Scientist Day to Day Activities (CRISP-DM)

Business Understanding	Data Understanding	Data Preparation		Modeling	Optimization	Deployment
Determine Business Objectives	Design Features	Transform/Fix Target Variable	Data Normalization	Select The Model	Model Selection	Planning Deployment
Frame the Problem Assess Feasibility	Collect Initial Data	Redundant & Duplicates	Data Factorization	Split Data	Model Optimization	Monitoring & Maintenance
Define Success Measurements	Install & Import Packages	Data Quality Audit (Missing Values)	Data Binarization	Data Scaling	Parameters Tuning	Final Report
Identify Target Variables (Y)	Read the Data	Data Quality Audit (Outliers)	Data Standardizing	Dummy Model		Lessons Learned
Identify Analytical Approach	Data Manipulation & Wrangling	Data Quality Audit (Cardinality Check)	Data Correlations	Build Model		
Identify Deployment Plan	Exploratory Data Analysis (EDA)	Data Conversion	Data Aggregation Binning	Fit Model (Train)		
Produce Project Plan	Data Visualization	Data Transformation	Data Decomposition	Predict (Test)		
Identify the team & Stakeholders	Statistical Analysis	Feature Engineering (Importance, Low variance, PCA)	Feature Selections	Assess & Evaluate		
Analytics Base Table (ABT)	Code Book Quality Report	Data Version 2/3/4		Best Model	Best Parameters	ROI

Source: Big Bang Data Science

Feature Engineering

Conducting feature engineering is a crucial step in enhancing the efficacy of our algorithms and optimizing model performance. The process of feature engineering involves generating novel input features based on pre-existing ones. Broadly speaking, the concept of data cleaning can be understood as a deductive process, while feature engineering can be viewed as an inductive process.

Feature Engineering

1. You can isolate and highlight key information, which helps the algorithms “focus” on what is important.
    - Create any **interaction features** that make sense. These are combinations of two or more features.
    - In some contexts, “interaction terms” must be products between two variables. In other contexts, interaction features can be **products**, **sums**, **differences, or quotients** between two features.

2. Bring in your own domain expertise.
    - Try to think of specific information you might want to **isolate** or **put the focus on**. Here, you have a lot of “creative freedom” and “skill expression” as a data scientist.

3. Most importantly, once you understand the “vocabulary” of feature engineering, you can bring in other people’s domain expertise.

Data Cleansing and Feature Engineering

1. Imputation
2. Handling Outliers
3. Fix mislabeled classes
4. Binning
5. Log Transform
6. One-Hot Encoding
7. Grouping Observations
8. Feature Split
9. Scaling
10. Extracting Date

Data Cleansing: Missing Values: Imputation

- Numeric data: impute with the median or mean

```
> trainData$variablename[is.na(trainData$variablename)] <- median(trainData$variablename, na.rm = TRUE)
> trainData$variablename[is.na(trainData$variablename)] <- mean(trainData$variablename, na.rm = TRUE)
```

- Categorical data: impute with the mode

```
> val <- unique(trainData$variablename[!is.na(trainData$variablename)])
> my_mode <- val[which.max(tabulate(match(trainData$variablename, val)))]
> trainData$variablename <- trainData$variablename
> trainData$variablename[is.na(trainData$variablename)] <- my_mode        #replicate
> table(trainData$variablename)

> testData2 <- predict(preProcess_missingdata_model, testData)  #after preprocessing & model built
> testData2$variablename[is.na(testData2$variablename)] <- my_mode
```

Data Cleansing: Missing Values: Imputation

K Nearest Neighbors

```
> library(RANN)  # required for knnImpute
> preProcess_missingdata_model <- preProcess(trainData, method='knnImpute’)
> trainData <- predict(preProcess_missingdata_model, newdata = trainData)
> anyNA(trainData)
```

- used for centering and scaling. method = "knnImpute" enforces centering and scaling the data because the kNN algorithm makes little sense without those processes.
- use method = "bagImpute" or method = "medianImpute" if you don't want the observations to be transformed.

Data Cleansing: Outlier Analysis

Outliers are data points that deviate significantly from the rest of the data. An outlier is any data point that exhibits an exceptionally high or low value or deviates significantly from the expected pattern within the context of a given project, as determined through observation. As a component of the data cleansing process, a data scientist would commonly detect the outliers and subsequently manage them through an established methodology.

- Delete the outlier values or even the actual variable where the outliers exist
- Transform the values or the variable itself

See “Diabetes EDA.Rmd” from EDA I: Descriptive Statistics Analysis

Feature Engineering: Binning

Binning is a technique that will take a column with continuous numbers and place the numbers in “bins” based on ranges that we determine. This will give a new categorical variable feature.

```
> Data table Information to be sorted %>%
> select(categories) %>%
> table( )   #This will give a table of the sorted information.
> library(Hmisc)
> equal_freq( var, n_bins)
```

**There is an opinion that binning should be avoided at all costs (slight exaggeration to say), but it is certainly the case that binning introduces bin choices that introduce some arbitrariness to the analysis. With modern statistical methods, it is generally not necessary to engage in binning, since anything that can be done on discretized "binned" data can generally be done on the underlying continuous values.

Feature Engineering: Log Transformation

The application of logarithmic transformation can be utilized to alter the independent and dependent variables and address any skewed data that may potentially affect the linear regression analysis. Other techniques such as arcsine transformation, geometric mean, and negative value can also be employed to establish a linear relationship in the original data. The outcome is an improved normality assumption, rendering the linear model more amenable to conducting statistical tests on transformed data.

The basic way of doing a log in R:

```
> log(df$Y, 10)  # base 10 log
> log_y <- log10(df$Y)   # logarithm of the value in the base
> sqrt_y <- sqrt(df$y)
> cube_y <- df$y^(1/3)
```

A log transformation is a process of applying a logarithm to data to reduce its skew. This is usually done when the numbers are highly skewed to reduce the skew so the data can be understood easier.

```
> df$logvariablename=log(df$variablename) # log in R example - data frame column

> plot(head(df$variablename2),head(df$logvariablename))

> plot(head(df$variablename2),head(df$variablename))
```

Feature Engineering: Dummy variables: fastDummies package

Convert a categorical variable to as many binary variables as there are categories.

```
> library(fastDummies)
> dummy_model = dummy_cols(data, select_columns = c(“variable1“, “variable2”) ,
remove_most_frequent_dummy = TRUE, remove_selected_columns = TRUE)
```

Feature Engineering: Dummy variables: caret package

Convert a categorical variable to as many binary variables as there are categories.

```
> dummies_model <- dummyVars(Y ~ ., data=trainData)
```

Create the dummy variables using predict. The Y variable will not be present in trainData_mat.

```
> trainData$Y<- as.factor(trainData$Y)
> trainData_mat <- predict(dummies_model, newdata = trainData)
```

Convert to dataframe

```
> trainData <- data.frame(trainData_mat)
```

See the structure of the new dataset

```
> str(trainData)
```

Feature Engineering: Grouping Observations

Combine Sparse Classes

- Sparse classes (in categorical features) are those that have very few total observations. They can be problematic for certain machine learning algorithms, causing models to overfit.
  - Group similar classes
- There is no formal rule of how many observations each class needs. It also depends on the size of your dataset and the number of other features you have.

Feature Engineering: Feature Split

In feature splitting, you split a single feature into two or more parts to get the necessary information.

- case_when() function
- If-else() function
- split() function

Feature Engineering: Scaling

Some common methods to handle continuous features:

Min-Max Normalization

- For each value in a feature, Min-Max normalization subtracts the minimum value in the feature and then divides by its range. The range is the difference between the original maximum and the original minimum. It scales all values in a fixed range between **0** and **1.**

Standardization

- The Standardization ensures that each feature has a mean of **0** and a standard deviation of **1**, bringing all features to the same magnitude.
- If the standard deviation of features is ***different***, their range would also differ.

Feature Engineering: Date and time features

Let’s say you have the feature hospital_datetime.

- It might be more useful to extract hospital_day_of_week and hospital_hour_of_day.
- You can also aggregate observations to create features such as hospital_over_last_30_days.

Time series data: The nice thing about time series data is that you only need one feature, some form of date, to layer in features from another dataset.

Feature Selection

The process of feature selection holds significant importance as it involves the elimination of irrelevant and redundant features, thereby reducing the number of variables in the model. As individuals engaged in the practice of modeling, it is imperative that we maintain a level of simplicity in our models. Moreover, the incorporation of variables that hold little to no significance can have a substantial effect on the performance of the model.

Feature Selection: Methods

1. **Embedded**: combine the qualities of filter and wrapper methods. They are implemented by algorithms that have their own built-in feature selection methods.

If some sort of search procedure is required to identify feature subsets that improve predictive performance. There are two general classes of techniques for this purpose: *filters* and *wrappers*.

2. **Filter**: conduct an initial supervised analysis of the predictors to determine which are important and then only provide these to the model.
3. **Wrapper**: use the performance of a learning algorithm to assess the usefulness of a feature set. In order to select a feature subset a learner is trained repeatedly on different feature subsets and the subset which leads to the best learner performance is chosen.

Filter and wrapper methods work to marry feature selection approaches with modeling techniques.

Feature Selection: Remove used or redundant features

Unused features are those that don’t make sense to pass into our machine learning algorithms. Examples include:

- ID columns
- Features that wouldn’t be available at the time of prediction
- Other text descriptions

Redundant features would typically be those that have been replaced by other features that you’ve added during feature engineering. For example, if you group a numeric feature into a categorical one, you can often improve model performance by removing the “distracting” original feature.

Feature Selection: Correlation

- In Statistics, often use the Pearson correlation coefficient to measure the linear relationship between two variables.
- Magnitude of r tells you the degree of LINEAR relationship between variables
- Sign of r tells you the direction (positive or negative) of the relationship between variables
- Presence of outliers affects the sign and magnitude of r
- Variability of scores within a distribution affects the value of r
- Used when scores are normally distributed

Feature Selection: Hypothesis Testing

To determine if the independent variable has a significant dependent variable or not, use hypothesis testing.

For instance, we want to determine whether the distance traveled correlates with the car's speed or not.

a. The null hypothesis is that there is no correlation between the distance traveled and the vehicle's speed.
b. Another hypothesis is that the amount of distance driven is related to the vehicle's speed.

Since there are two continuous variables in this situation, we would conduct **independent t-tests**.

We reject the null hypothesis if the p-value is less than the alpha value. This indicates that since speed has a substantial link with distance, it should be included in the model. If both variables are categorical, the **chi-square test** can be used.

Feature Selection: Information gain

- Information gain tells us how much information is given by the independent variable about the dependent variable. Information gain is helpful in the case of both categorical and numerical dependent variables.

```
> library(FSelector)
> information.gain(Y~., df)
```

Feature Selection: Wrapper Methods

- Forward Selection – The algorithm starts with an empty model and keeps on adding the significant variables one by one to the model.
- Backward Selection – In this technique, start with all the variables in the model and then keep on deleting the worst features one by one.
- Stepwise Selection – the Stepwise algorithm is a combination of both forward and backward algorithms which essentially means that at each iteration, a variable can be considered for addition or deletion from the model.

```
> library(MASS)
 Using stepAIC function
> stepAIC(model, direction = "both", trace = FALSE)
> stepAIC(model, direction = "backward", trace = FALSE)
> stepAIC(model, direction = "forward", trace = FALSE)
```

These tradition methods can also be achieved by using **selectFeatures() function from {mlr} package**.

Feature Selection: Recursive feature elimination (rfe)

- Determine what features are important to predict the Y. This technique builds a model with all the variables, and then the algorithm removes the weakest features one by one until we reach the specified number of features. While using RFE, we need to specify the number of features to be used. However, this is often not known at the beginning. The optimal number of features can be identified using cross-validation.
- Use the {caret} package to run the rfe algorithm.

Setting the cross-validation parameters

- ctrl_param <- rfeControl(functions = rfFuncs, method = "repeatedcv", repeats = 5, verbose = **FALSE**, returnResamp = "all")
- rfe_lm_profile <- rfe(training[, -5], training[, 5], sizes = c(2,3), rfeControl = ctrl_param, newdata = testing[, -5])
- rfe_lm_profile
- varImp(rfe_lm_profile)

Feature Selection: Lasso Regression

LASSO regression is a classification algorithm is a powerful technique that performs two main tasks; regularization and feature selection. The method shrinks (regularizes) the coefficients of the regression model as part of penalization. For feature selection, the variables which are left after the shrinkage process are used in the model. The goal of the algorithm is to minimize the prediction error.

```
> library(glmnet)

# Using LASSO for variable selection
> feat_mod_select <- cv.glmnet(as.matrix(df[ , 1]) , df[ , 1], standardize = TRUE, alpha =1)

# Checking coefficients with the minimum cross-validation error
> as.matrix(coef(features, features$lambda.min))
```

Feature Selection: Random Forest

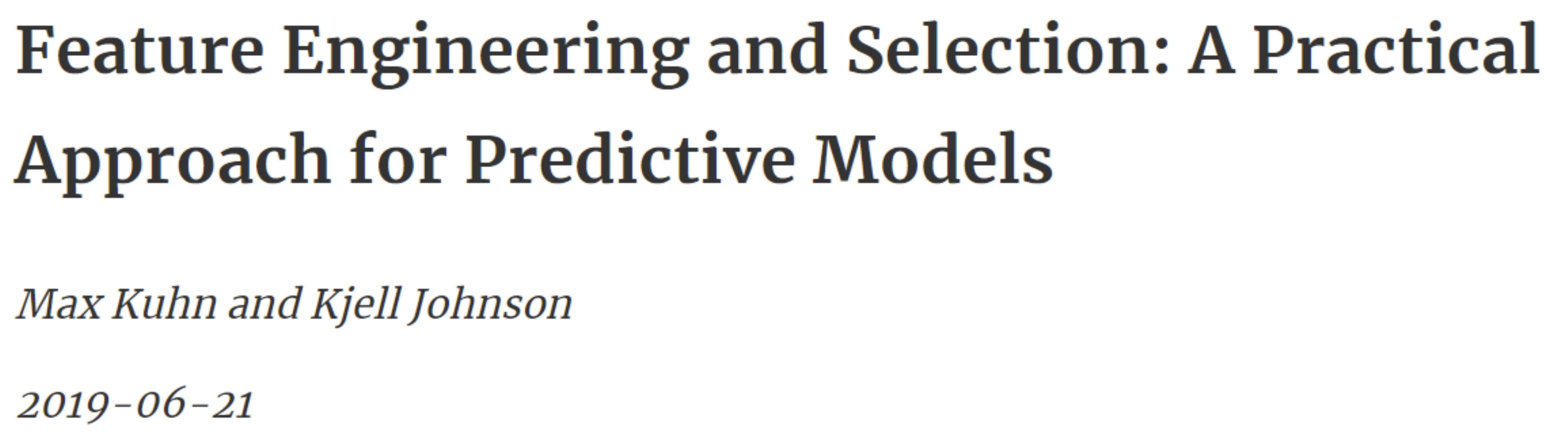
Feature selection using Random forest comes under the category of Embedded methods. When training a tree, it is possible to compute how much each feature decreases the impurity. The more a feature decreases the impurity, the more important the feature is. In random forests, the impurity decrease from each feature can be averaged across trees to determine the final importance of the variable.

```
> library(randomForest)

> rfModel <-randomForest(Y ~ ., data = traindata) # Using random forest for variable selection

> importance(rfModel) # Getting the list of important variables
```

To give a better intuition, features that are selected at the top of the trees are in general more important than features that are selected at the end nodes of the trees, as generally the top splits lead to bigger information gains.



Feature Engineering and Selection: A Practical Approach for Predictive Models

Max Kuhn and Kjell Johnson

2019-06-21

Link to the HTML HERE

TIDYMODELS

The tidymodels framework is a collection of packages for modeling and machine learning using tidyverse principles.

Install tidymodels with:

```
install.packages("tidymodels")
```

Preprocessing data

- https://www.tidyverse.org/blog/2022/02/recipes-0-2-0/
- Introduction to recipes

```
> library(tidyverse)
> library(recipes)
> rec <- recipe(data) %>%
> step_center(everything()) %>%
> step_scale(everything()) %>%
> check_missing(everything())
> rec %>% class()
## [1] "recipe"
```

Data Scientist Day to Day Activities (CRISP-DM)

Business Understanding	Data Understanding	Data Preparation		Modeling	Optimization	Deployment
Determine Business Objectives	Design Features	Transform/Fix Target Variable	Data Normalization	Select The Model	Model Selection	Planning Deployment
Frame the Problem Assess Feasibility	Collect Initial Data	Redundant & Duplicates	Data Factorization	Split Data	Model Optimization	Monitoring & Maintenance
Define Success Measurements	Install & Import Packages	Data Quality Audit (Missing Values)	Data Binarization	Data Scaling	Parameters Tuning	Final Report
Identify Target Variables (Y)	Read the Data	Data Quality Audit (Outliers)	Data Standardizing	Dummy Model		Lessons Learned
Identify Analytical Approach	Data Manipulation & Wrangling	Data Quality Audit (Cardinality Check)	Data Correlations	Build Model		
Identify Deployment Plan	Exploratory Data Analysis (EDA)	Data Conversion	Data Aggregation Binning	Fit Model (Train)		
Produce Project Plan	Data Visualization	Data Transformation	Data Decomposition	Predict (Test)		
Identify the team & Stakeholders	Statistical Analysis	Feature Engineering (Importance, Low variance, PCA)	Feature Selections	Assess & Evaluate		
Analytics Base Table (ABT)	Code Book Quality Report	Data Version 2/3/4		Best Model	Best Parameters	ROI

Source: Big Bang Data Science