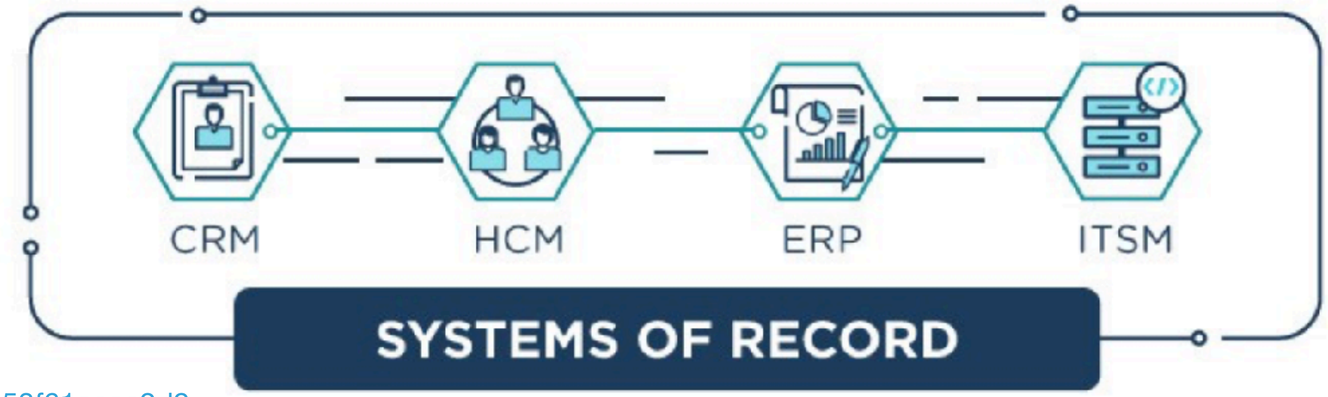
INTELLIGENT APPLICATION NETWORKS

WITH MULE AND TENSORFLOW

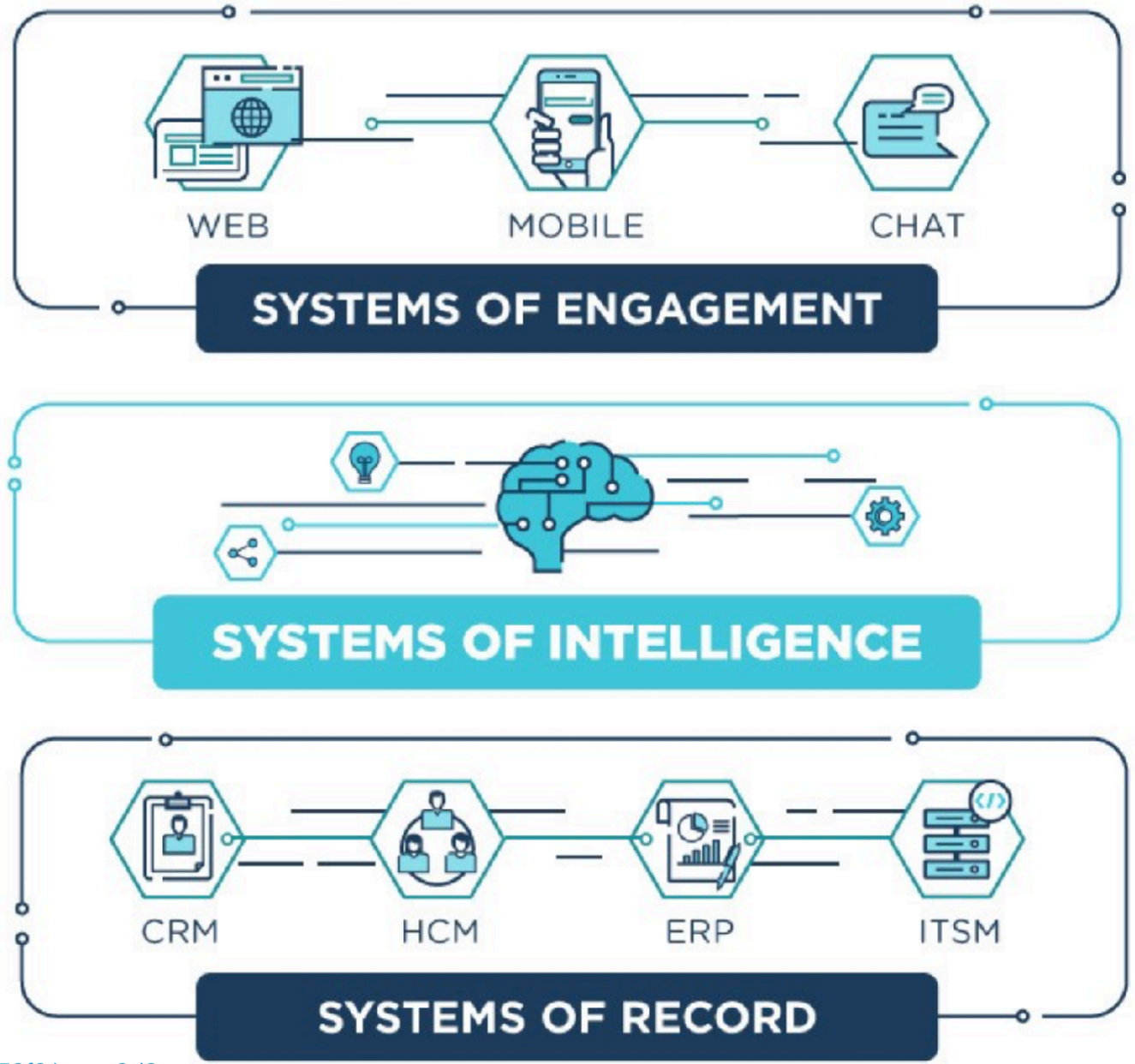
MACHINE LEARNING BASED AI WILL TRANSFORM 80% OF BUSINESSES IN FIVE YEARS



https://news.greylock.com/the-new-moats-53f61aeac2d9

https://sloanreview.mit.edu/projects/reshaping-business-with-artificial-intelligence/

MACHINE LEARNING BASED AI WILL TRANSFORM 80% OF BUSINESSES IN FIVE YEARS



https://news.greylock.com/the-new-moats-53f61aeac2d9

https://sloanreview.mit.edu/projects/reshaping-business-with-artificial-intelligence/

Accelerating hyperspecialization of data, applications, and devices

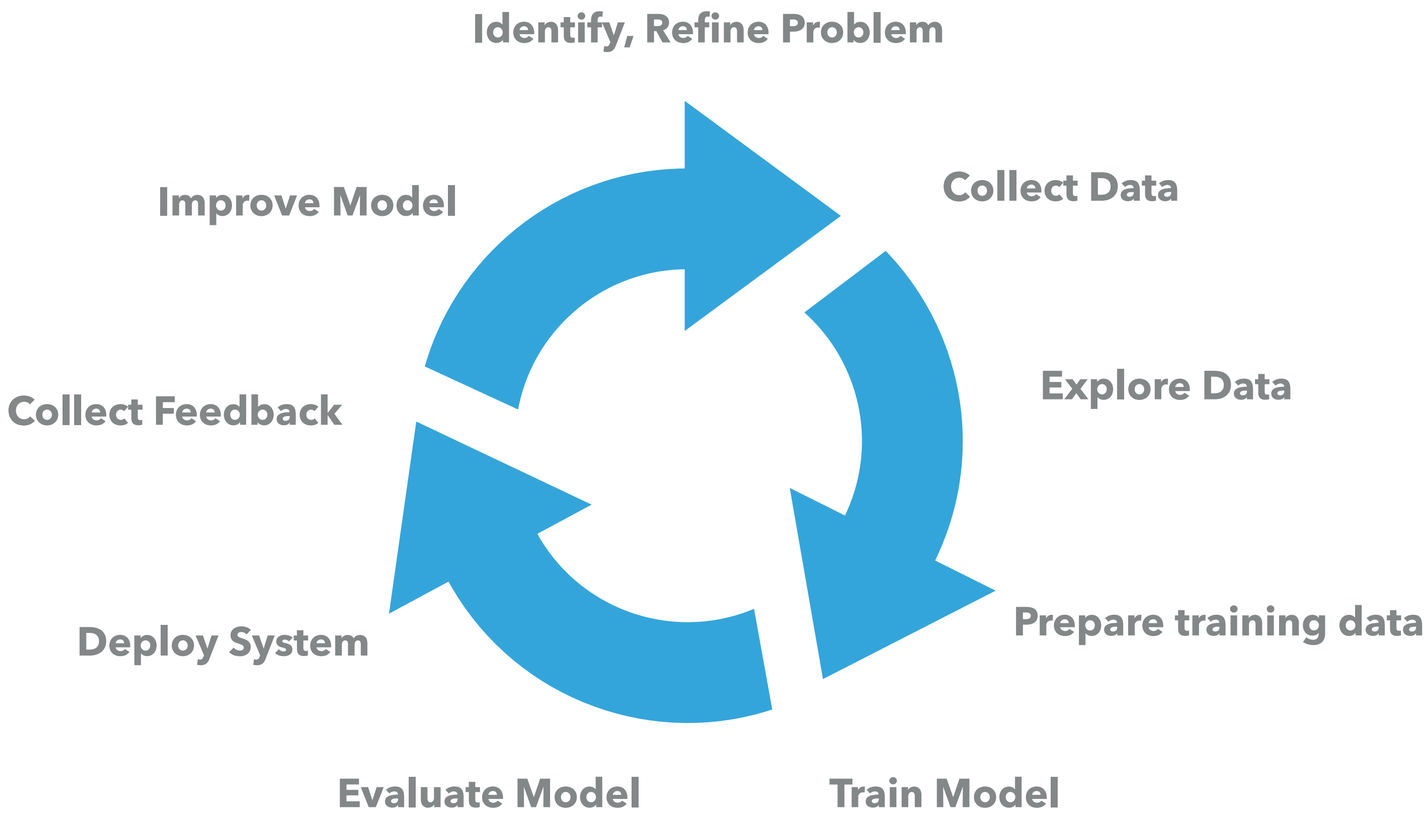
HYPER SPECIALIZATION IN MARKETING

https://chiefmartec.com/2018/04/marketing-technology-landscape-supergraphic-2018/

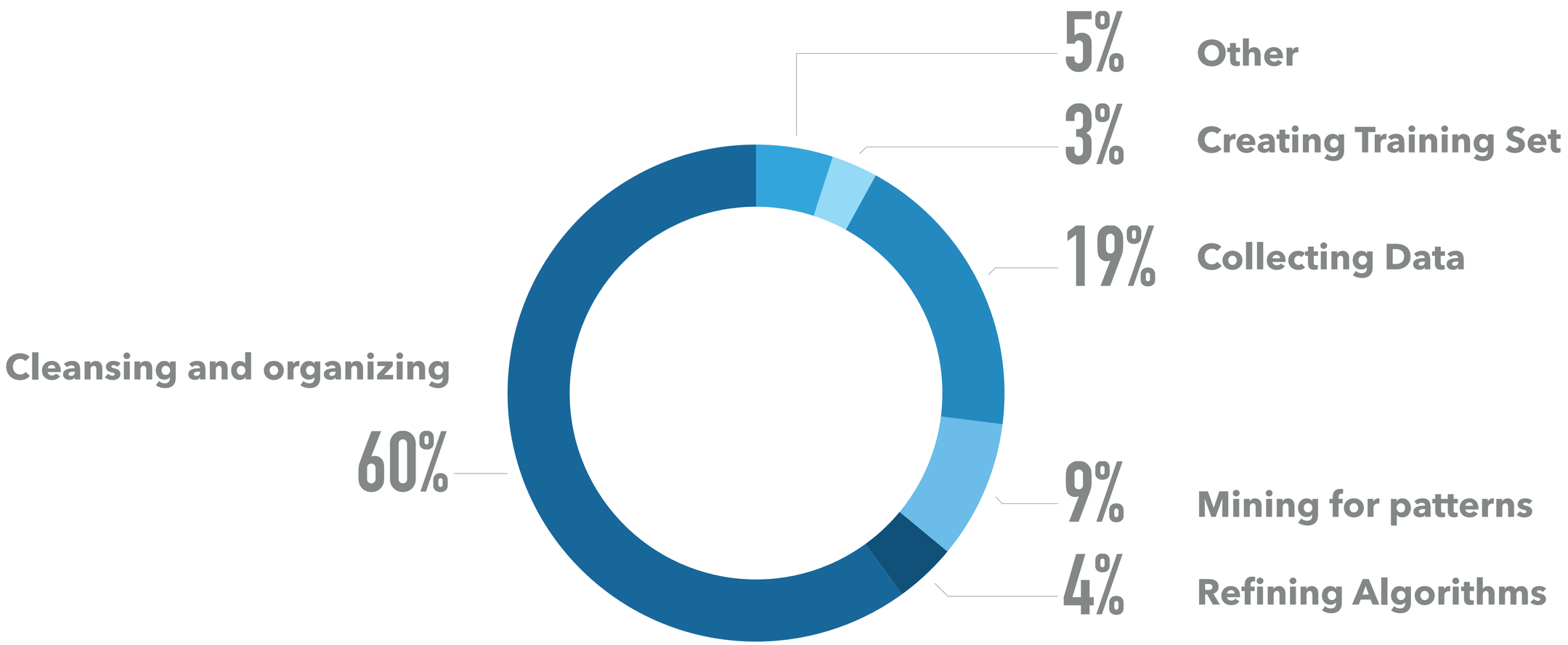
COMPANIES DOING "INITIAL COIN OFFERINGS "

https://www.cbinsights.com/reports/CB-Insights_Blockchain-In-Review.pdf

DATA TEAMS MUST ITERATE QUICKLY



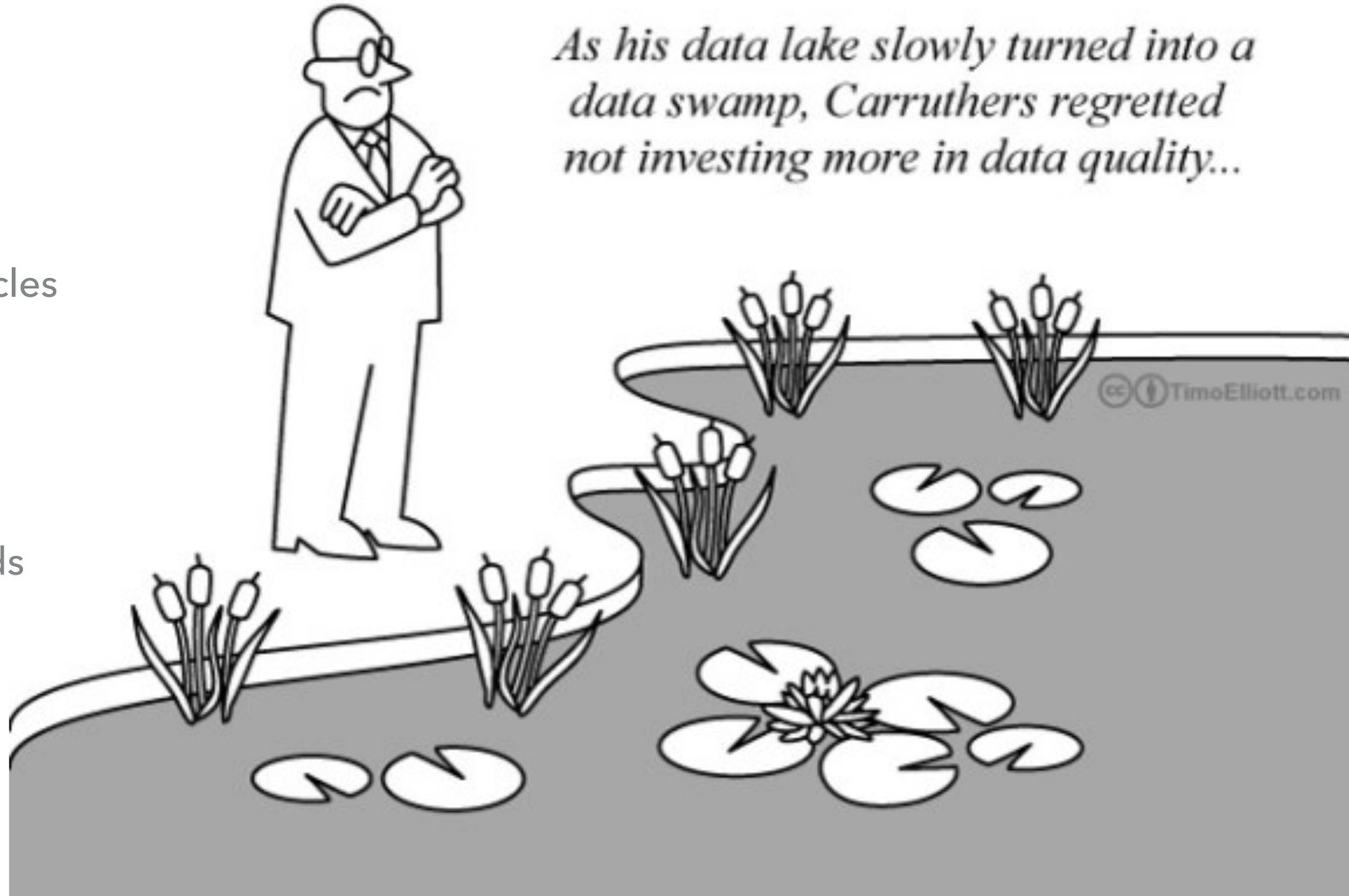
COLLECTING AND CLEANING DATA IS 80%



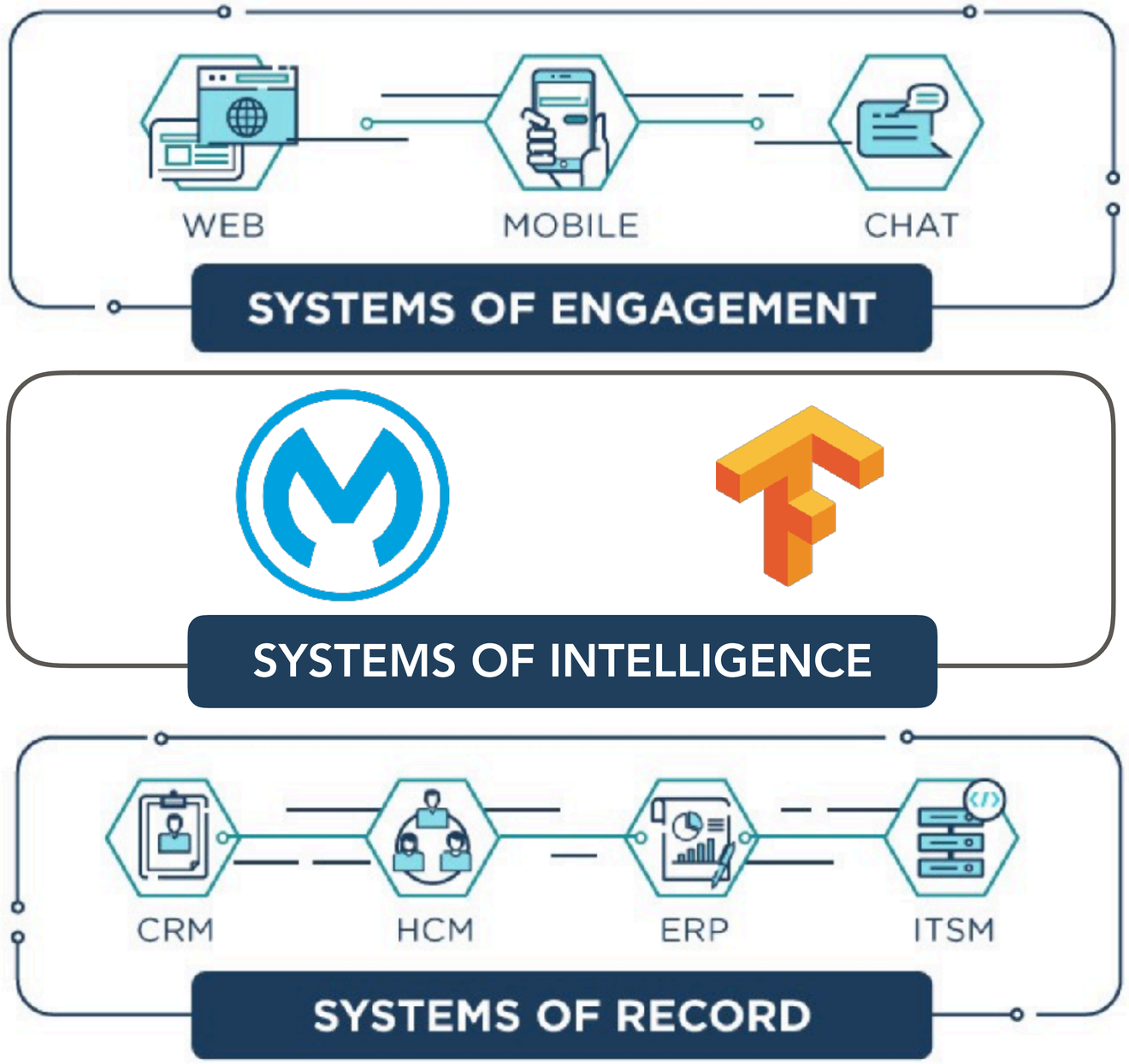
https://www.forbes.com/sites/gilpress/2016/03/23/data-preparation-most-time-consuming-least-enjoyable-data-science-task-survey-says

FAILURE TO ITERATE RAPIDLY

- Custom code to get models into production grade applications
  - Lack tight integration with user
- Long ETL cycles, different update cycles
  - Poor discoverability
  - Lack of labeled data
- Data lakes become dumping grounds
  - Governance
  - Poor metadata



CONNECTIVITY AND ORCHESTRATION MEETS MACHINE LEARNING TO ENABLE RAPID ITERATION

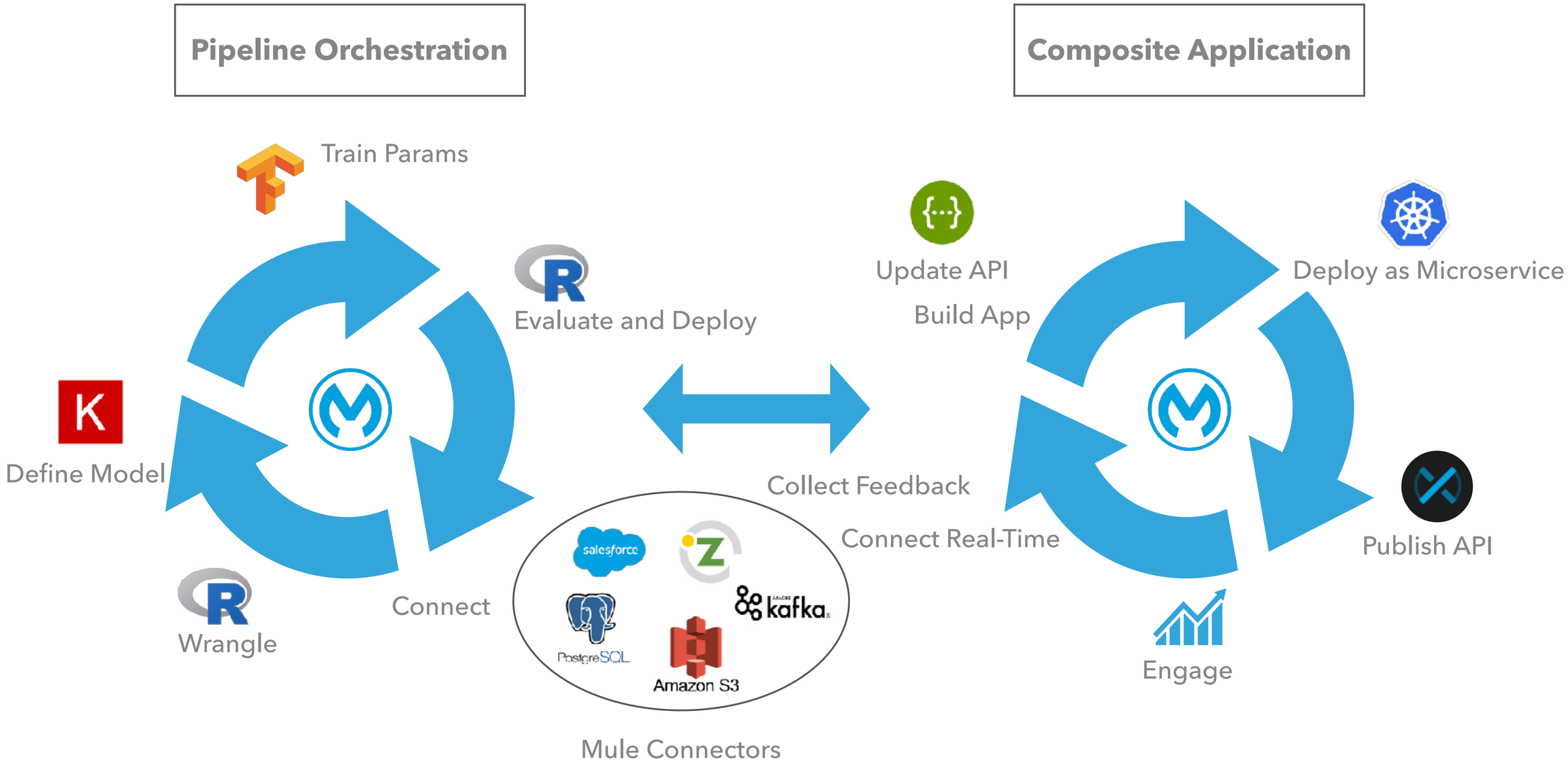


APPLICATION NETWORKS

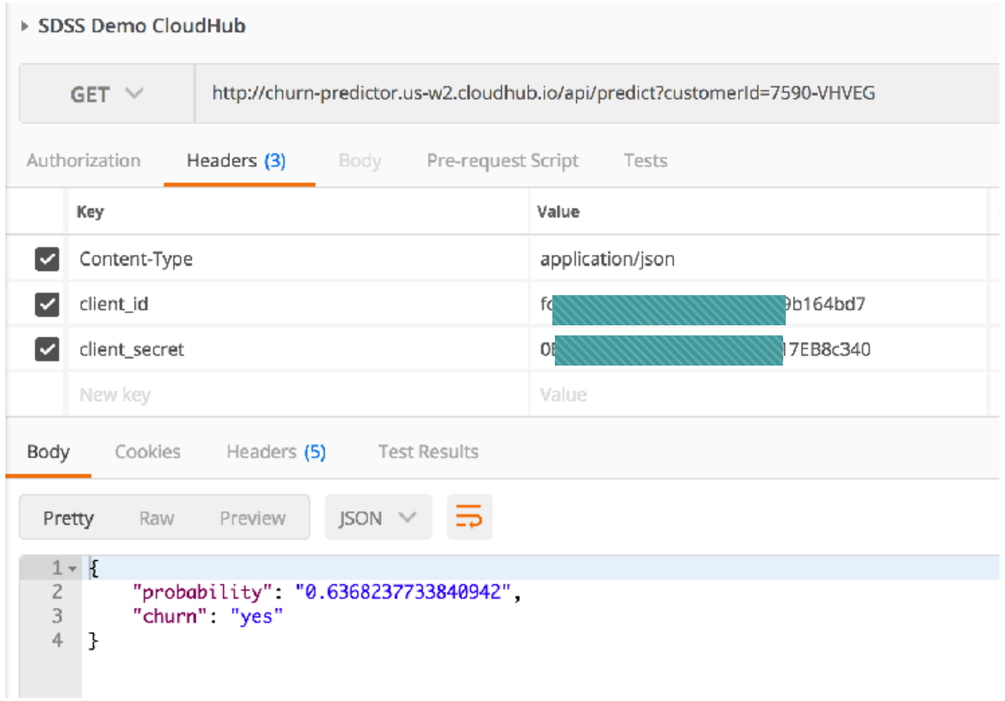
- APIs and metadata
  - Data types
  - Capabilities
- Orchestration
  - Composite applications

https://www.mulesoft.com/resources/api/what-is-an-application-network

RAPIDLY ITERATE PRODUCTION MODELS AS SERVICES



A CHURN PREDICTION: DESIGN AND IMPLEMENT AN API



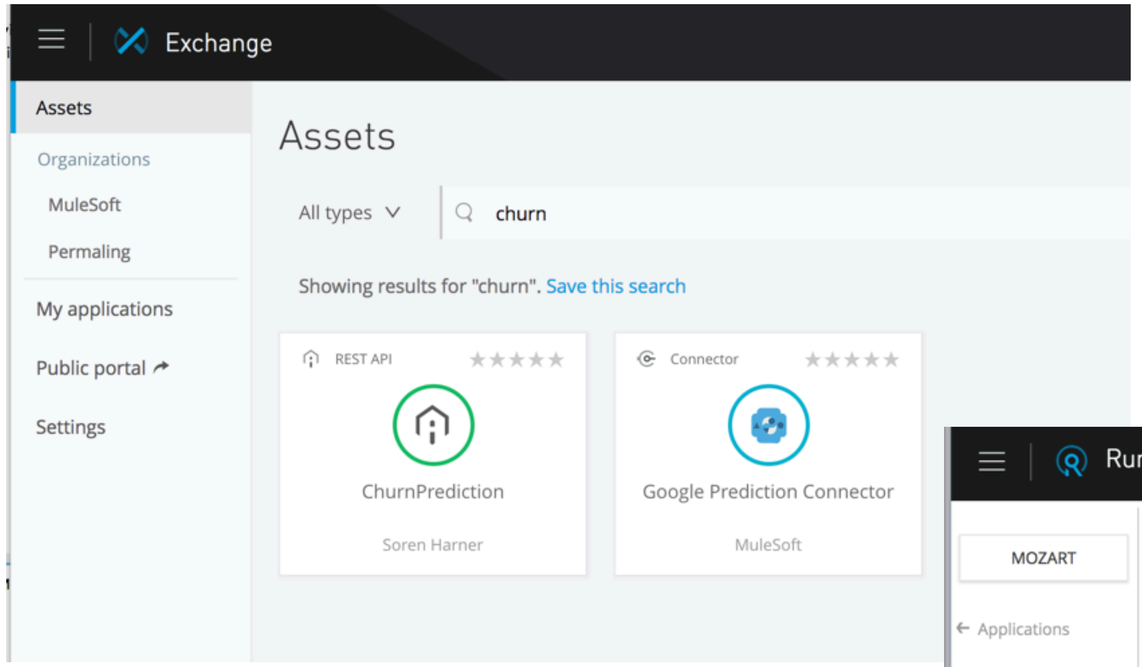
Querying the production server in Postman

churn_prediction.raml

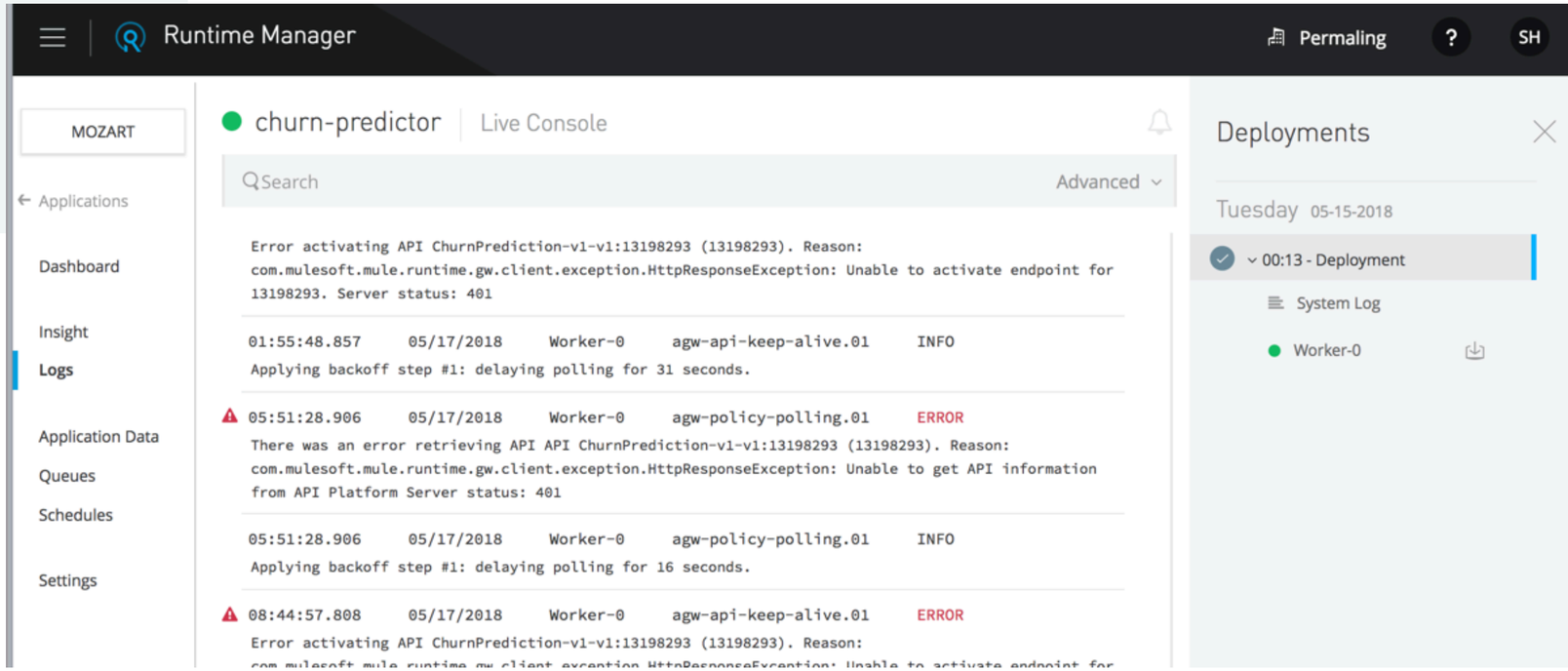
```
#%RAML 1.0
baseUri: https://mocksvc.mulesoft.com/mocks/d0de9712-8a54-4180-9e44-d4359ad4484b #
title: ChurnPrediction
version: v1

traits:
  client-id-required:
    headers:
      client_id:
        type: string
      client_secret:
        type: string

/predict:
  get:
    is: [client-id-required]
    description: |
      given a customer id, predict whether the customer will leave in the next month
    queryParameters:
      customerId:
        required: true
        type: string
        example: "7590-VHVEG"
    responses:
      200:
        body:
          application/json:
            example: |
              {"churn": "yes", "probability": 0.6}
      404:
        body:
          application/json:
            example: |
              {"message": "Customer not found"}

```

API Spec and Type Definitions (RAML)

ONCE YOU DEPLOY AND PUBLISH, YOU'VE ADDED A NOTE TO THE APPLICATION NETWORK



Publish the API for other applications to use in MuleSoft's Anypoint Platform



Deploy as a micro service to MuleSoft's PaaS (based on Kubernetes)

LOAD KERAS MODEL AND WEIGHTS INTO SPARK AND CALL FROM MULE

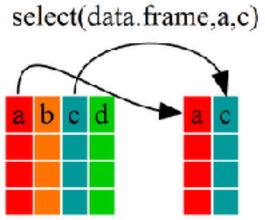
```
var keras_model: MultiLayerNetwork = null

// Initialize Fusion Model
def init_churn_model(): Unit = {
  val in = this.getClass.getClassLoader
    .getResourceAsStream("churn_model.h5")
  FileUtils.copyInputStreamToFile(in, new File(_tmp_model_file))
  keras_model = KerasModelImport.importKerasSequentialModelAndWeights(_tmp_model_file)
}

def predict(predictors: Array[Double]) : java.util.HashMap[String, String] = {
  // Load the model if needed
  if (keras_model == null) init_churn_model

  // Run the model
  val d: INDArray = Nd4j.create(predictors)
  val prob = keras_model.labelProbabilities(d).getDouble(0)

  // Return java-friendly output for Mule
  val jmap = new util.HashMap[String, String]
  jmap.put("churn", if (prob > 0.5) "yes" else "no")
  jmap.put("probability", prob.toString)
  jmap
}
```

ORCHESTRATING THE ML PIPELINE WITH MULE FLOWS

Fetch

Mule
Connectors

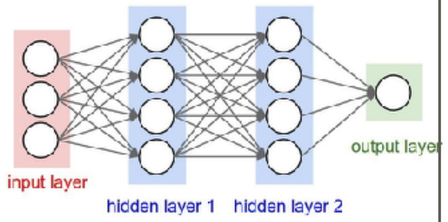
Preprocess



Mule
- Scheduler
- DataWeave

R Packages
- Recipe
- Dplyr
- Farcats

Train



Tensorflow
- 35 inputs
- 2 hidden layers
- Binary output

R Packages
- Keras
- Rsampler

Evaluate

n=165	Predicted: NO	Predicted: YES	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

Mule
- Scatter-Gather
- Choice

R Packages
- Yardstick

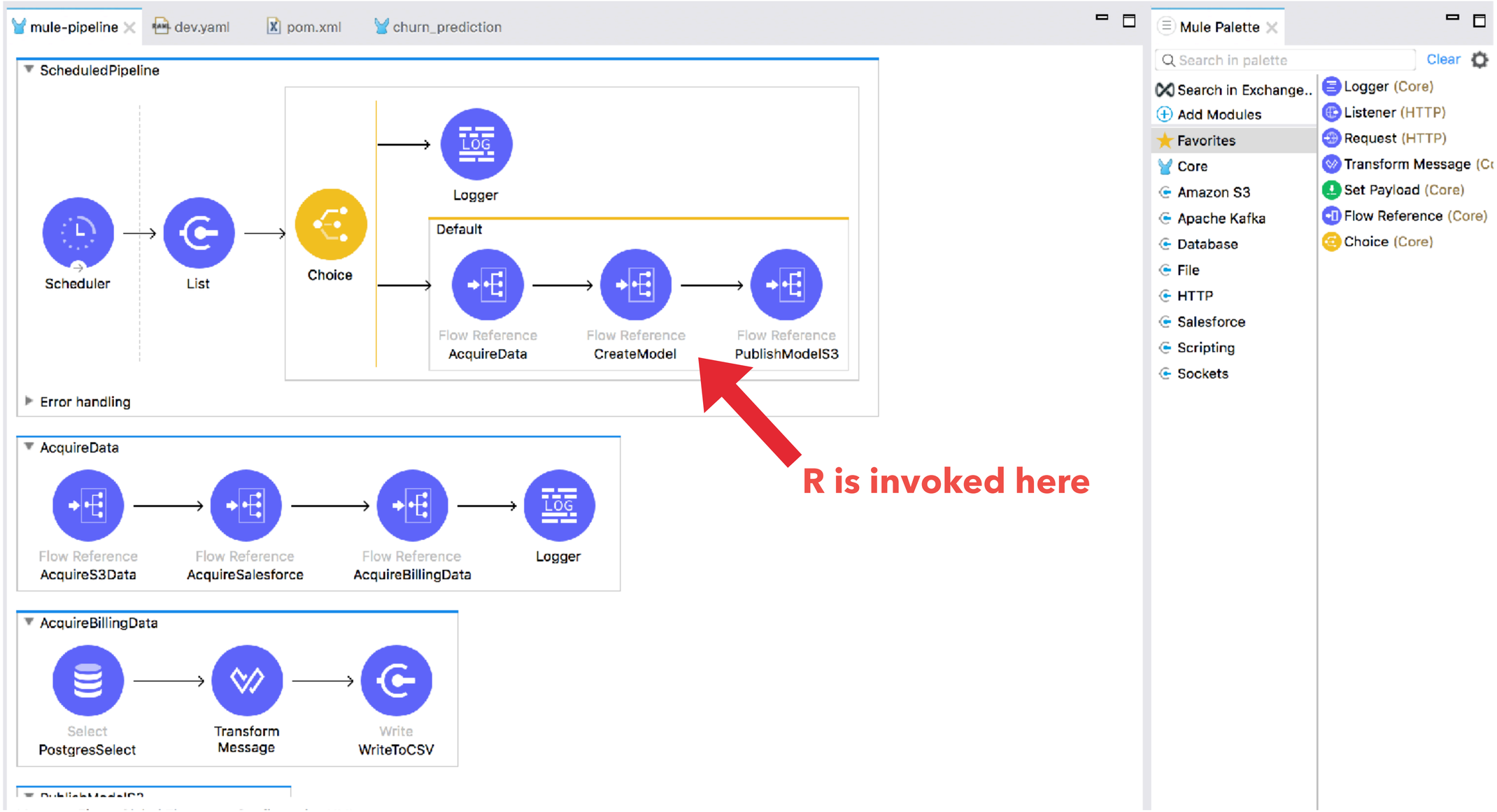
Publish

Amazon S3

Mule
Connector

Persist
- Recipe
- Model
- Weights

BUILDING THE ML PIPELINE IN MULESOFT'S ANYPOINT STUDIO



PROCESSING THE DATA USING RECIPES

```
# A tibble: 7,032 x 20
   Churn gender SeniorCitizen Partner Dependents tenure PhoneService MultipleLines    InternetService
   <chr> <chr>          <int> <chr>   <chr>       <int> <chr>        <chr>           <chr>
 1 No    Female             0 Yes     No              1 No           No phone service DSL
 2 No    Male               0 No      No             34 Yes          No              DSL
 3 Yes   Male               0 No      No              2 Yes          No              DSL
 4 No    Male               0 No      No             45 No           No phone service DSL
 5 Yes   Female             0 No      No              2 Yes          No              Fiber optic
 6 Yes   Female             0 No      No              8 Yes          Yes             Fiber optic
 7 No    Male               0 No      Yes            22 Yes          Yes             Fiber optic
 8 No    Female             0 No      No             10 No           No phone service DSL
 9 Yes   Female             0 Yes     No             28 Yes          Yes             Fiber optic
10 No    Male               0 No      Yes            62 Yes          No              DSL
# ... with 7,022 more rows, and 11 more variables: OnlineSecurity <chr>, OnlineBackup <chr>,
#   DeviceProtection <chr>, TechSupport <chr>, StreamingTV <chr>, StreamingMovies <chr>,
#   Contract <chr>, PaperlessBilling <chr>, PaymentMethod <chr>, MonthlyCharges <dbl>,
#   TotalCharges <dbl>
```

Data Set

Watson dataset on Telco Churn

- 20 predictors of churn
- Demographics from Salesforce
- Products from billing database
- Transactions from S3 flat files

Preparing the training and test with R Recipes

```
rec_obj <- recipe(Churn ~ ., data = train_tbl) %>%
  step_discretize(tenure, options = list(cuts = 6)) %>%
  step_log(TotalCharges) %>%
  step_dummy(all_nominal(), -all_outcomes()) %>%
  step_center(all_predictors(), -all_outcomes()) %>%
  step_scale(all_predictors(), -all_outcomes()) %>%
  prep(data = train_tbl)

# Predictors
x_train_tbl <- bake(rec_obj, newdata = train_tbl) %>% select(-Churn)
x_test_tbl  <- bake(rec_obj, newdata = test_tbl) %>% select(-Churn)

# Response variables for training and testing sets
y_train_vec <- ifelse(pull(train_tbl, Churn) == "Yes", 1, 0)
y_test_vec  <- ifelse(pull(test_tbl, Churn) == "Yes", 1, 0)
```

Adapted from: https://tensorflow.rstudio.com/blog/keras-customer-churn.html

TRAIN THE MODEL WITH KERAS WITH TENSORFLOW BACKEND

Defining the model in R Keras

```
model_keras <- keras_model_sequential() %>%
  # First hidden layer
  layer_dense(
    units              = 16,
    kernel_initializer = "uniform",
    activation         = "relu",
    input_shape        = ncol(x_train_tbl)) %>%
  # Dropout to prevent overfitting
  layer_dropout(rate = 0.1) %>%
  # Second hidden layer
  layer_dense(
    units              = 16,
    kernel_initializer = "uniform",
    activation         = "relu") %>%
  # Dropout to prevent overfitting
  layer_dropout(rate = 0.1) %>%
  # Output layer
  layer_dense(
    units              = 1,
    kernel_initializer = "uniform",
    activation         = "sigmoid") %>%
  # Compile ANN
  compile(
    optimizer = 'adam',
    loss      = 'binary_crossentropy',
    metrics   = c('accuracy')
  )
```

HIDDEN LAYER 1
(20, 16), RELU

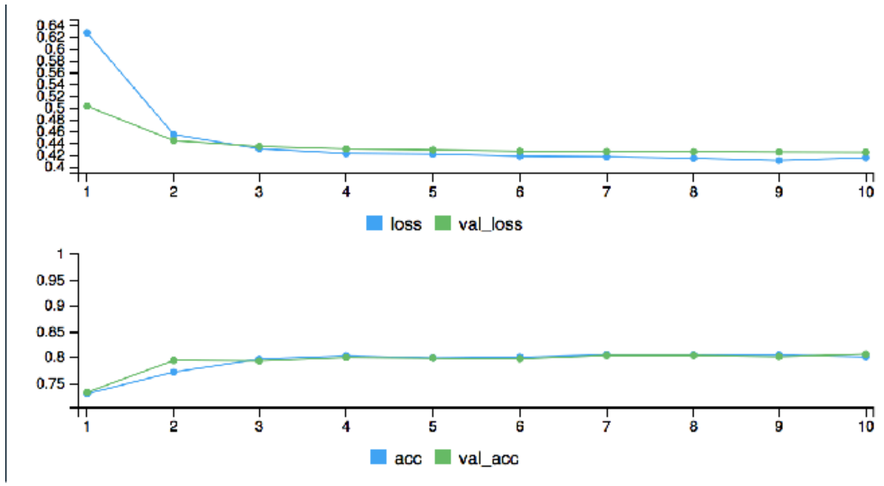
↓

HIDDEN LAYER 2
(16, 16), RELU

↓

HIDDEN LAYER 2
(16, 1), SIGMOID

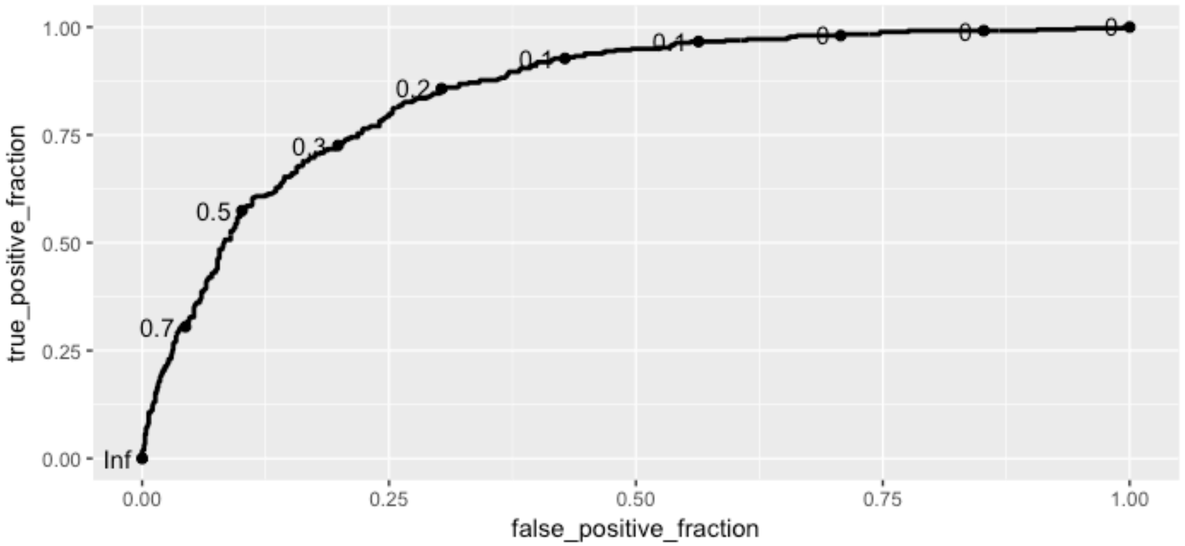
Training the model in R Keras




```
history <- fit(
  object           = model_keras,
  x                = as.matrix(x_train_tbl),
  y                = y_train_vec,
  batch_size       = 50,
  epochs           = 10,
  validation_split = 0.30
)

# Class predictions
yhat_keras_class_vec <- predict_classes(object = model_keras,
                                        x = as.matrix(x_test_tbl)) %>%
  as.vector()

# Predicted class probabilities
yhat_keras_prob_vec  <- predict_proba(object = model_keras,
                                      x = as.matrix(x_test_tbl)) %>%
  as.vector()
```

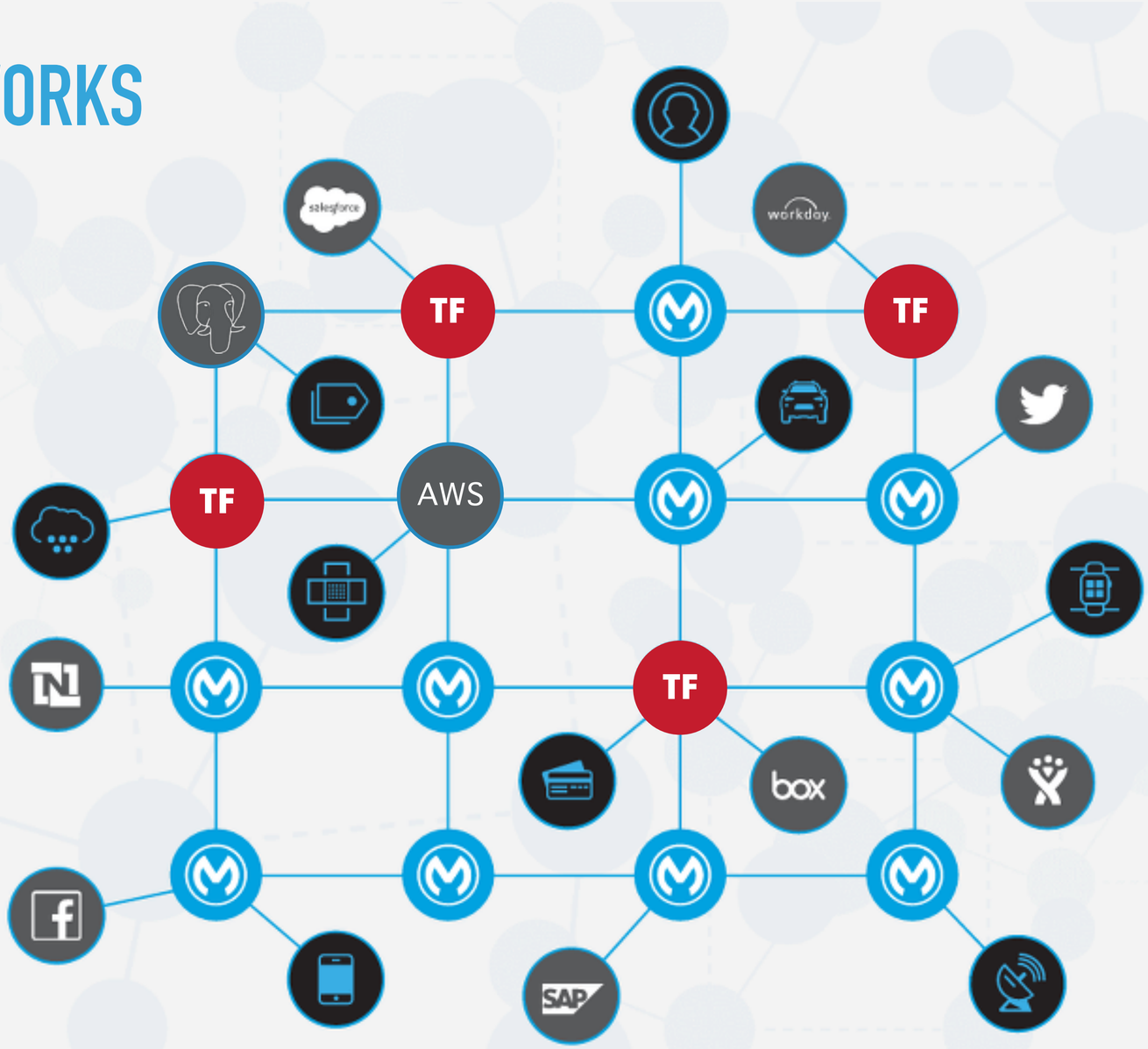
EVALUATE WITH YARDSTICK, "PUBLISH" UPDATE MODEL IF IT IMPROVES

```
estimates_keras_tbl <- tibble(
  truth      = as.factor(y_test_vec) %>% fct_recode(yes = "1", no = "0"),
  estimate   = as.factor(yhat_keras_class_vec) %>% fct_recode(yes = "1", no = "0"),
  class_prob = yhat_keras_prob_vec
)
estimates_keras_tbl
options(yardstick.event_first = FALSE)

# Accuracy
estimates_keras_tbl %>% metrics(truth, estimate)

# Confusion Table
estimates_keras_tbl %>% conf_mat(truth, estimate)

# AUC
estimates_keras_tbl %>% roc_auc(truth, class_prob)

# Plot the ROC
roc_plot <- ggplot(estimates_keras_tbl, aes(d = truth, m = class_prob)) + geom_roc()

# Precision
tibble(
  precision = estimates_keras_tbl %>% precision(truth, estimate),
  recall    = estimates_keras_tbl %>% recall(truth, estimate)
)

# F1-Statistic
estimates_keras_tbl %>% f_meas(truth, estimate, beta = 1)
```



INTELLIGENT APPLICATION NETWORKS

- Specifications
  - Capabilities
  - Ontologies
- Orchestration
  - Composite applications
  - Knowledge Graph
- Systems of Intelligence
  - Computational Graphs



LINK TO PRESENTATION, VIDEO OF DEMO, AND CODE FOR EXAMPLE

https://github.com/sharner/sdss-mule

Soren Harner is a principal data scientist with Permaling.

Permaling provides strategy consulting, implementation, and training for AI in the enterprise.

https://permaling.com

soeren [at] permaling [dot] com

Permaling