“Sheetify” - ML Based Normalization Tool

By Will Judy
For DB Engineering
May 16 - August 17, 2022

GAMEPLAN

01

What's the Problem?

02

Intermediate Steps Towards the Solution

03

Solution, Architecture, Next Steps

04

Demo, Questions

01

The Problem

The Problem: What is it?

Say we have a new building we'd like to analyze...

Start with a "Cache" of points - minimal input labels...

- Cache was configured by one of thousands of Electrical Engineering companies, each with their own naming standards

Must normalize / standardize points by providing specific classifications for various features -> turn a cache into a "Loadsheet"

Currently, somebody has to go through the cache by hand assigning values

“Cache”

controlProgram	ObjectID	DeviceID	ObjectName	units
AC-1 Munters / 1015 Joaquin	AV:1	DEV:2724512	bpsp_1	inches-of-water
AC-1 Munters / 1015 Joaquin	AV:13	DEV:2724512	total_cl_req_1	no-units
AC-1 Munters / 1015 Joaquin	MSV:5	DEV:2724512	m480_1	no-units
AC-1 Munters / 1015 Joaquin	AV:14	DEV:2724512	total_ht_req_1	no-units
OA Conditions - Bldg Broadcast	AV:1301	DEV:2700046	temperature_avo_7	degrees-fahrenheit
AC-1 Munters / 1015 Joaquin	AV:5	DEV:2724512	eff_rat_1	degrees-fahrenheit
AC-1 Munters / 1015 Joaquin	AV:11	DEV:2724512	total_airflow_1	cubic-feet-per-minute
AC-1 Munters / 1015 Joaquin	AV:6	DEV:2724512	stat_press_1	inches-of-water
AC-1 Munters / 1015 Joaquin	AV:17	DEV:2724512	sastat_stpt_read_1	inches-of-water
AC-1 Munters / 1015 Joaquin	AV:9	DEV:2724512	m862_1	degrees-fahrenheit
AC-1 Munters / 1015 Joaquin	AV:7	DEV:2724512	sa_stpt_read_1	degrees-fahrenheit
AC-1 Munters / 1015 Joaquin	BV:19	DEV:2724512	unit_enable_1	no-units

A Cache is the raw data from a building

Each point, contains important information about: PointName, PointType, Device Type, Enumerated Equipment Value, Units

ISSUES: Every cache is configured slightly differently. Might not have access to equivalent input features

“Loadsheet”

location	controlProgram	name
/WebCTRL - BALDAR/Site 2 - Baldar/Joaquin Road/B62 - 1015 Joaquin/Mezzanine	AC-1 Munters / 1015 Joaquin	Bldg Static Pressure Setpt
/WebCTRL - BALDAR/Site 2 - Baldar/Joaquin Road/B62 - 1015 Joaquin/Mezzanine	AC-1 Munters / 1015 Joaquin	Total Cool Request
/WebCTRL - BALDAR/Site 2 - Baldar/Joaquin Road/B62 - 1015 Joaquin/Mezzanine	AC-1 Munters / 1015 Joaquin	Eff run command
/WebCTRL - BALDAR/Site 2 - Baldar/Joaquin Road/B62 - 1015 Joaquin/Mezzanine	AC-1 Munters / 1015 Joaquin	Total Heat Request
/WebCTRL - BALDAR/Site 2 - Baldar/Joaquin Road/B62 - 1015 Joaquin	OA Conditions - Bldg Broadcast	OA Temperature
/WebCTRL - BALDAR/Site 2 - Baldar/Joaquin Road/B62 - 1015 Joaquin/Mezzanine	AC-1 Munters / 1015 Joaquin	Effective RAT
/WebCTRL - BALDAR/Site 2 - Baldar/Joaquin Road/B62 - 1015 Joaquin/Mezzanine	AC-1 Munters / 1015 Joaquin	Total Airflow
/WebCTRL - BALDAR/Site 2 - Baldar/Joaquin Road/B62 - 1015 Joaquin/Mezzanine	AC-1 Munters / 1015 Joaquin	Duct Static Pressure
/WebCTRL - BALDAR/Site 2 - Baldar/Joaquin Road/B62 - 1015 Joaquin/Mezzanine	AC-1 Munters / 1015 Joaquin	SA Static Stpt Read
/WebCTRL - BALDAR/Site 2 - Baldar/Joaquin Road/B62 - 1015 Joaquin/Mezzanine	AC-1 Munters / 1015 Joaquin	supply air temp
/WebCTRL - BALDAR/Site 2 - Baldar/Joaquin Road/B62 - 1015 Joaquin/Mezzanine	AC-1 Munters / 1015 Joaquin	SAT Stpt (Read)
/WebCTRL - BALDAR/Site 2 - Baldar/Joaquin Road/B62 - 1015 Joaquin/Mezzanine	AC-1 Munters / 1015 Joaquin	Unit Enable

ObjectType	ObjectID	DeviceID	ObjectName	units	required	bacnetAvailable
BAV	AV:1	DEV:2724512	bpsp_1	inches-of-water	YES	MAYBE
BAV	AV:13	DEV:2724512	total_cl_req_1	no-units	YES	MAYBE
BMSV	MSV:5	DEV:2724512	m480_1	no-units	YES	MAYBE
BAV	AV:14	DEV:2724512	total_ht_req_1	no-units	YES	MAYBE
BAV	AV:1301	DEV:2700046	temperature_avo_7	degrees-fahrenheit	YES	MAYBE
BAV	AV:5	DEV:2724512	eff_rat_1	degrees-fahrenheit	YES	MAYBE
BAV	AV:11	DEV:2724512	total_airflow_1	cubic-feet-per-minute	YES	MAYBE
BAV	AV:6	DEV:2724512	stat_press_1	inches-of-water	YES	MAYBE
BAV	AV:17	DEV:2724512	sastat_stpt_read_1	inches-of-water	YES	MAYBE
BAV	AV:9	DEV:2724512	m862_1	degrees-fahrenheit	YES	MAYBE
BAV	AV:7	DEV:2724512	sa_stpt_read_1	degrees-fahrenheit	YES	MAYBE
BBV	BV:19	DEV:2724512	unit_enable_1	no-units	YES	MAYBE

building	generaltype	phredentitycod	globalphredentitycode	FieldName
US-MTV-1015	AHU	AC 1	US-MTV-1015:AHU:AC 1	building_air_static_pressure_setpoint
US-MTV-1015	AHU	AC 1	US-MTV-1015:AHU:AC 1	cooling_request_count
US-MTV-1015	AHU	AC 1	US-MTV-1015:AHU:AC 1	exhaust_fan_run_command_1
US-MTV-1015	AHU	AC 1	US-MTV-1015:AHU:AC 1	heating_request_count
US-MTV-1015	AHU	AC 1	US-MTV-1015:AHU:AC 1	outside_air_temperature_sensor
US-MTV-1015	AHU	AC 1	US-MTV-1015:AHU:AC 1	return_air_temperature_sensor
US-MTV-1015	AHU	AC 1	US-MTV-1015:AHU:AC 1	supply_air_flowrate_sensor
US-MTV-1015	AHU	AC 1	US-MTV-1015:AHU:AC 1	supply_air_static_pressure_sensor
US-MTV-1015	AHU	AC 1	US-MTV-1015:AHU:AC 1	supply_air_static_pressure_setpoint
US-MTV-1015	AHU	AC 1	US-MTV-1015:AHU:AC 1	supply_air_temperature_sensor
US-MTV-1015	AHU	AC 1	US-MTV-1015:AHU:AC 1	supply_air_temperature_setpoint
US-MTV-1015	AHU	AC 1	US-MTV-1015:AHU:AC 1	supply_fan_run_command

- A Loadsheet is the data from a building after it’s been labeled and processed.

Cache + Additional Information:

- Equipment Label
  - Ex: ‘AC 1’, ‘VAVRH 2-1-28’
- FieldName
  - Ex: ‘zone_air_temperature_sensor’, ‘heating_water_valve_percentage_command’
- GeneralType
  - Ex: ‘AHU’, ‘FCU’, ‘VAV’, ‘EF’, ‘BLR’
- Required
  - Ex: ‘YES’, ‘NO’, ‘MISSING’

☆ We need to **normalize** a loadsheet before we can perform any analysis on the building. Consider:

Executing a search for [“zone_air_temperature_sensor”] where the value < 65°F

Executing a search for [“zn_t_1.1”, “zone_temp”, “rm_temp_sens”, “airTemp_sensor_3a”] where the value < 65°F

Why are we trying to fix this problem?

- The current labeling process is slow and tedious
- We plan to continue adding more buildings
- We have a stockpile of properly labeled data points from past projects that we can leverage
- Other companies sell software that does very similar labeling...
- We'd like to make Sheetify open source so anyone can use it on their data, saving time & money

What is “Sheetify”?

- Sheetify is a tool that intelligently produces labels and classifications for unlabeled sheets of building data
- Sheetify can generate labels for:
  - Equipment Label
  - Field Name
  - General Type
  - Required Status
- Sheetify learns from the past, analyzing previously labeled examples, using Machine Learning to predict the unknown
- Turns WEEKS of manual labeling work into MINUTES of mathematical predictions

What were the Project Objectives?

“Develop a series of Machine Learning models to categorize contextual building data”

Requirements:

- Accurately labels fields - “FieldName”
- Accurately categorize equipment by type - “GeneralType”
- Accurately extract equipment labels - “EquipmentLabel”
- Abstract away the underlying system in such a way that it can scale to novel input schemas

Bonus! Classify whether the point is “Required” or not

02

Intermediate Steps

Where did I begin?

Step 1: Initial research & planning

Step 2: Playing with training data

Step 3: Experiment with “tokenizing”

Step 4: Test out some ML models

Step 5: Construct independent data pipelines for each required prediction

Step 6: Combine pipelines into master model workflow

Step 1: Initial Research & Planning

Read up on:
- “DBO” - Digital Buildings Ontology
- Tokenizing

Brainstormed with:
- Trevor Sodorff, Nick Lima, Shane Spencer, Julien Ragbeer, Claire Stirdivant, Tom Skoczylas, to name a few

Familiarized myself with the problem:
- What is the useful information I should try to extract and return?
  - EquipmentLabel, FieldName, GeneralType, Required

Step 2: Playing with Data

I started by exploring the training data in Python

- Looked at specific data columns, analyzed possible values for each feature, for each dataset

- Realized... This is going to be difficult

PointNames have an infinite number of possible configurations!

Some FieldNames and GeneralTypes are more frequent than others

Step 3: Tokenizing

Challenge: Machine Learning models only accept numeric values as inputs

Tokenization: create numerical representations for letters and words

We break down “strings” into subcomponents, then represent those with unique numeric identifiers

Tokenization inherently means subtle data loss

Our method involves Letter Counts & “One-Hot Encoding”

Letter Counts

0	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z		_
zn_temp_sensor	0	0	0	0	2	0	0	0	0	0	0	0	1	2	1	1	0	1	2	1	0	0	0	0	0	1	0	2
zone_temp_sens	0	0	0	0	3	0	0	0	0	0	0	0	1	2	1	1	0	0	2	1	0	0	0	0	0	1	0	2
zn_temperature	1	0	0	0	3	0	0	0	0	0	0	0	1	1	0	1	0	2	0	2	1	0	0	0	0	1	0	1
zn_t	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	1	0	1
rm_temp	0	0	0	0	1	0	0	0	0	0	0	0	2	0	0	1	0	1	0	1	0	0	0	0	0	0	0	1
zone_t_sensor	0	0	0	0	2	0	0	0	0	0	0	0	0	2	2	0	0	1	2	1	0	0	0	0	0	1	0	2
air_temp	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	1
air_temperature_sensor	2	0	0	0	4	0	0	0	1	0	0	0	1	1	1	1	0	4	2	2	1	0	0	0	0	0	0	2
zone_air_temp	1	0	0	0	2	0	0	0	1	0	0	0	1	1	1	1	0	1	0	1	0	0	0	0	0	1	0	2
zn_air_sensor	1	0	0	0	1	0	0	0	1	0	0	0	0	2	1	0	0	2	2	0	0	0	0	0	0	1	0	2
Room_temp_sensor	0	0	0	0	2	0	0	0	0	0	0	0	2	1	3	1	0	2	2	1	0	0	0	0	0	0	0	2

Substring Component Presence

0	air	rm	room	sens	sensor	t	temp	emperatur	zn	zone
zn_temp_sensor	0	0	0	0	1	0	1	0	1	0
zone_temp_sens	0	0	0	1	0	0	1	0	0	1
zn_temperature	0	0	0	0	0	0	0	1	1	0
zn_t	0	0	0	0	0	1	0	0	1	0
rm_temp	0	1	0	0	0	0	1	0	0	0
zone_t_sensor	0	0	0	0	1	1	0	0	0	1
air_temp	1	0	0	0	0	0	1	0	0	0
air_temperature_sensor	1	0	0	0	1	0	0	1	0	0
zone_air_temp	1	0	0	0	0	0	1	0	0	1
zn_air_sensor	1	0	0	0	1	0	0	0	1	0
Room_temp_sensor	0	0	1	0	1	0	1	0	0	0

What is Supervised Machine Learning?

- Data needs two primary components:
  - Recognizable features: X's / inputs
  - Known class labels: y's / outcomes

- Behind the scenes, models discover underlying connections between the X's and y's, establishing 'weights'

- When we have data without "y" values, we send these features into a model to make a prediction

Why is Supervised ML fit for this problem?

- We have so many examples of labeled data!
- Difficult problem, but it's worth trying to save those human hours

Step 4: Testing ML Models

Used models from SciKit-Learn (Sklearn)

Tried:
- Naive Bayes
- Neural Networks
- Support Vector Machine
- K-nearest Neighbors
- Decision Tree
- Random Forest

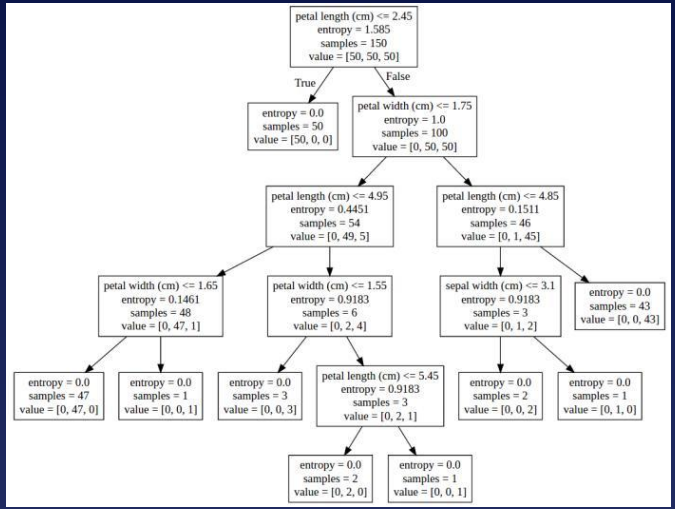


Image from KDNuggets.com: https://www.kdnuggets.com/2017/05/simplifying-decision-tree-interpretation-decision-rules-python.html

5 examples of labeled data - Delta Controls (ECT & EYT), Kamloops, Small Google & Large Google

Tested models with different training & Testing data

Decision Tree & Random Forest performed best across all tests,
Decided to use Random Forest Classifier

Step 5: Independent Pipelines

Initially created 3 tools:

- FieldNamePredict
- GeneralTypePredict
- EquipmentLabelPredict
- (Required predictor came later)

Process: Read in training files, trim out insignificant rows, set up input features, train models, test prediction accuracy

Mistake (Learning Experience): Set up different input configurations to each model

- ObjectName -> FieldName
- FieldNames (grouped by device) -> GeneralType
- General Type + Parsing for enumeration -> EquipmentLabel

Works best to send all potentially useful info into our models

Step 6: Put Everything Together

Previous steps confirmed ML's ability to accurately make predictions

Already know how to clean data and format to send into model, just needed to upscale & make more robust

Issue! Not all data files have the same input columns

- Ex: Some cache files don't have "Units" or "Device" info
- Others don't have Enumerated values for building Equipment Label

How do we work around this?

Solution: Generalized Column Categories & Input Data "Schemes" ->

INPUT ATTRIBUTES:
- Device
- PointType
- PointName
- Units
- Enumeration (only helps generate EQ label #)

OUTPUT ATTRIBUTES:
GeneralType
FieldName
EquipmentLabel
Required

9 possible categories for data

Column Categories & Input Data Schemes

Likelihood of scheme

11 Only Device

12 Only PointType

13 Only PointName

14 Only Units

5 Device & PointType

6 Device & PointName

7 Device & Units

8 PointType & PointName

9 PointType & Units

10 PointName & Units

1 PointType, PointName, Units

2 Device, PointName, Units

3 Device, PointType, Units

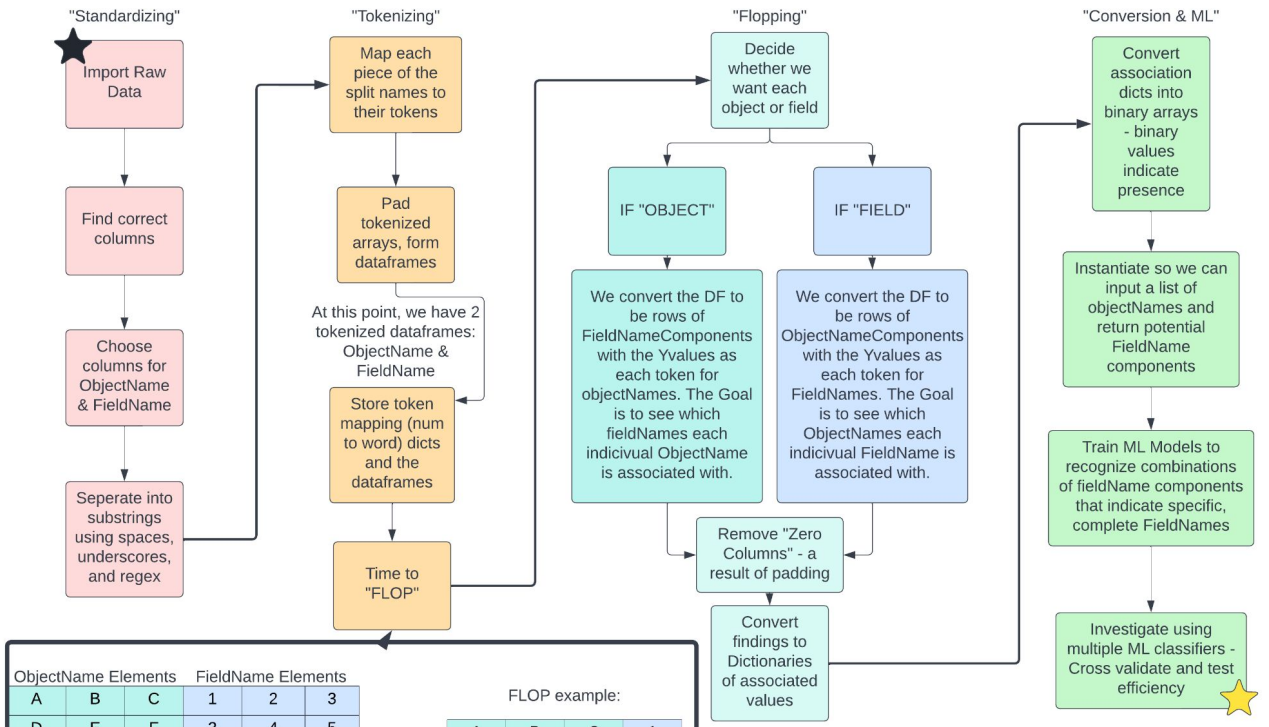
4 Device, PointType, PointName

Model 0 Device. PointType, PointName, Units

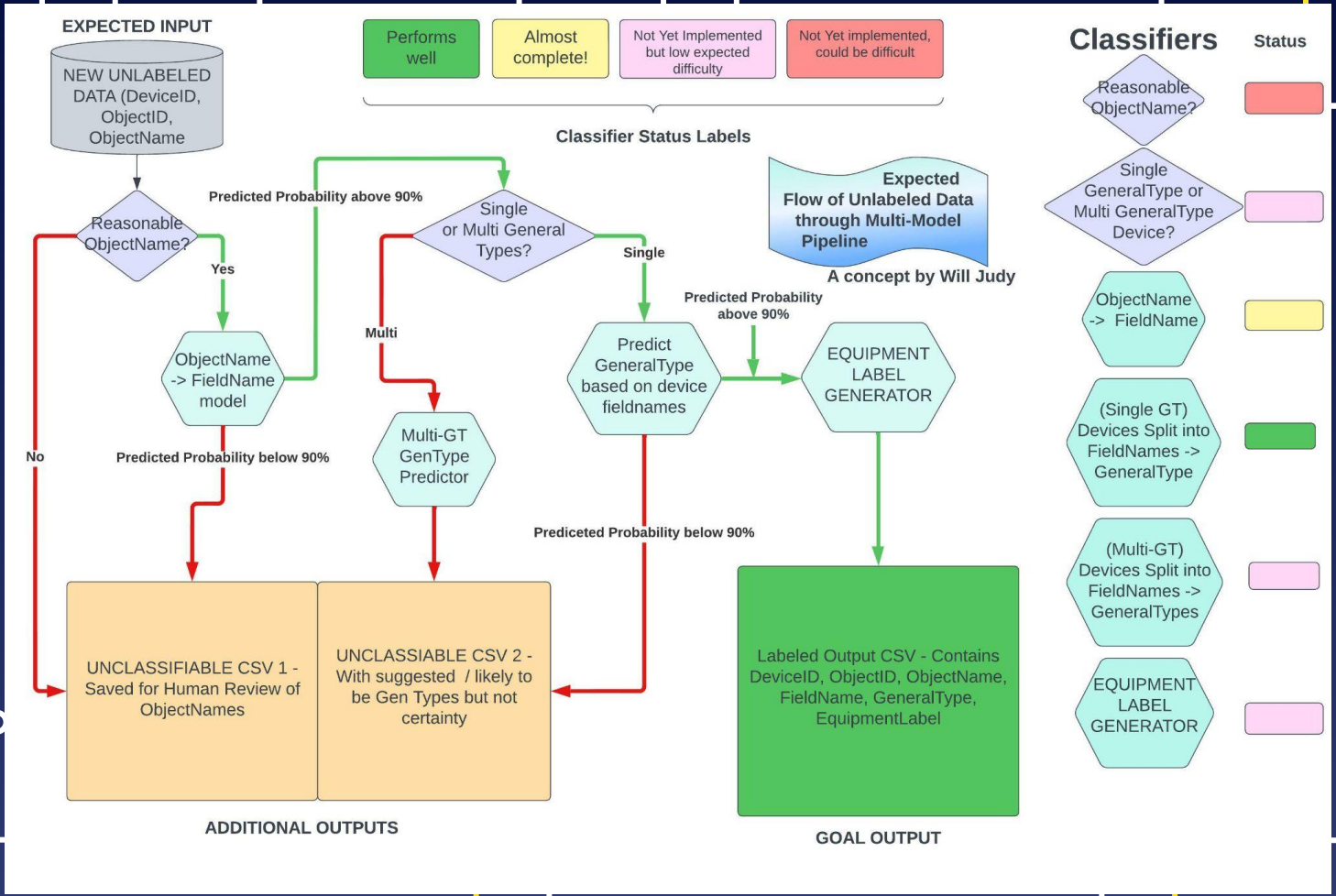
03

Solution, Architecture, Next Steps

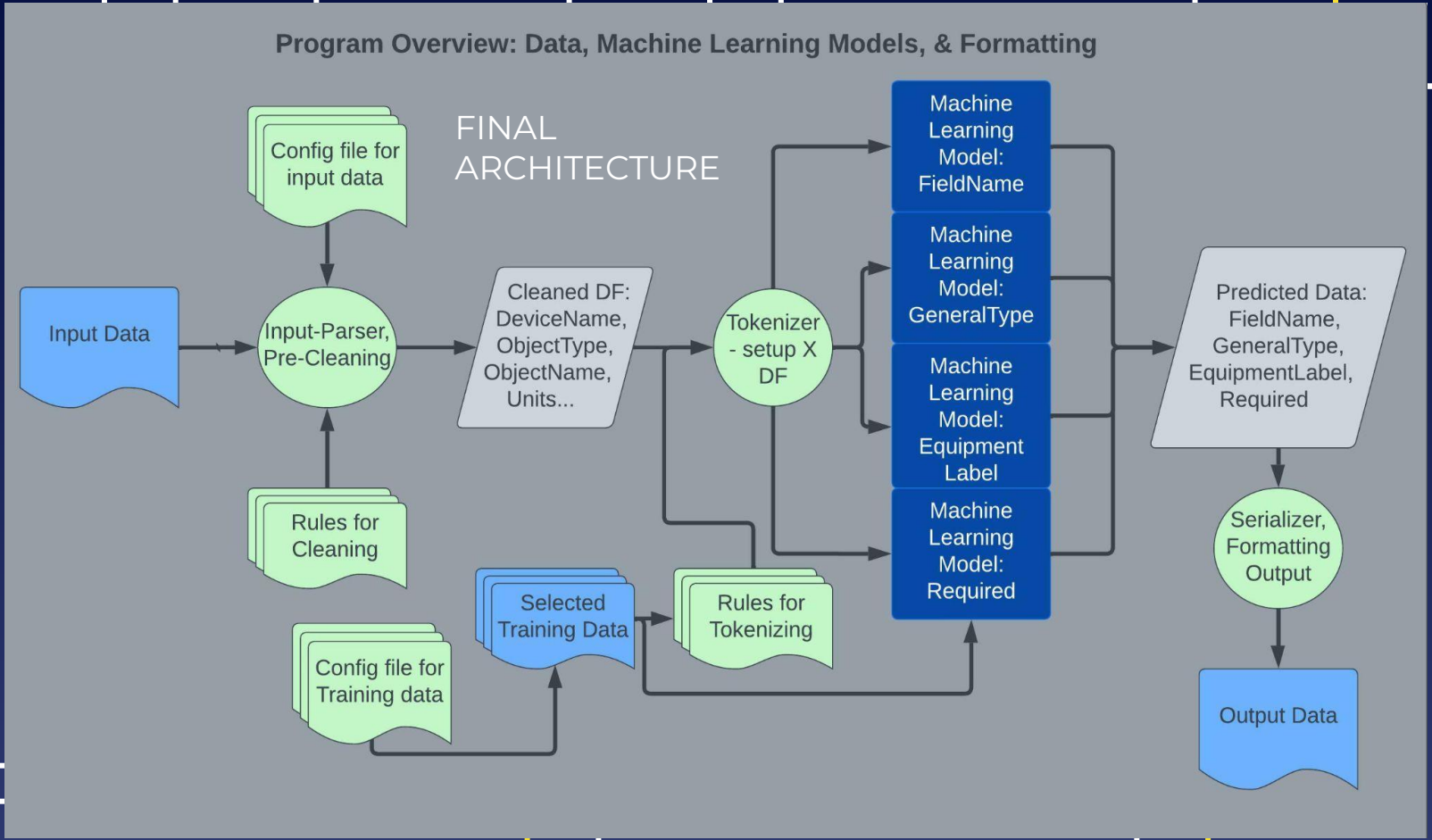
Concept 00 (June 2)



Concept 01
(July 8)



Concept 02
(August 1)



Config Files

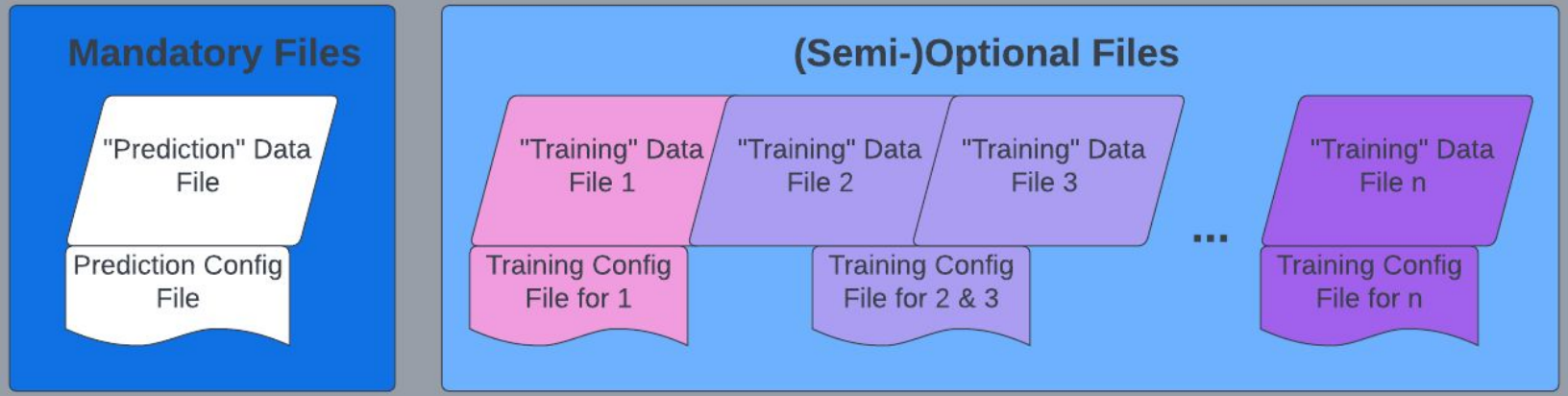
Prediction Config

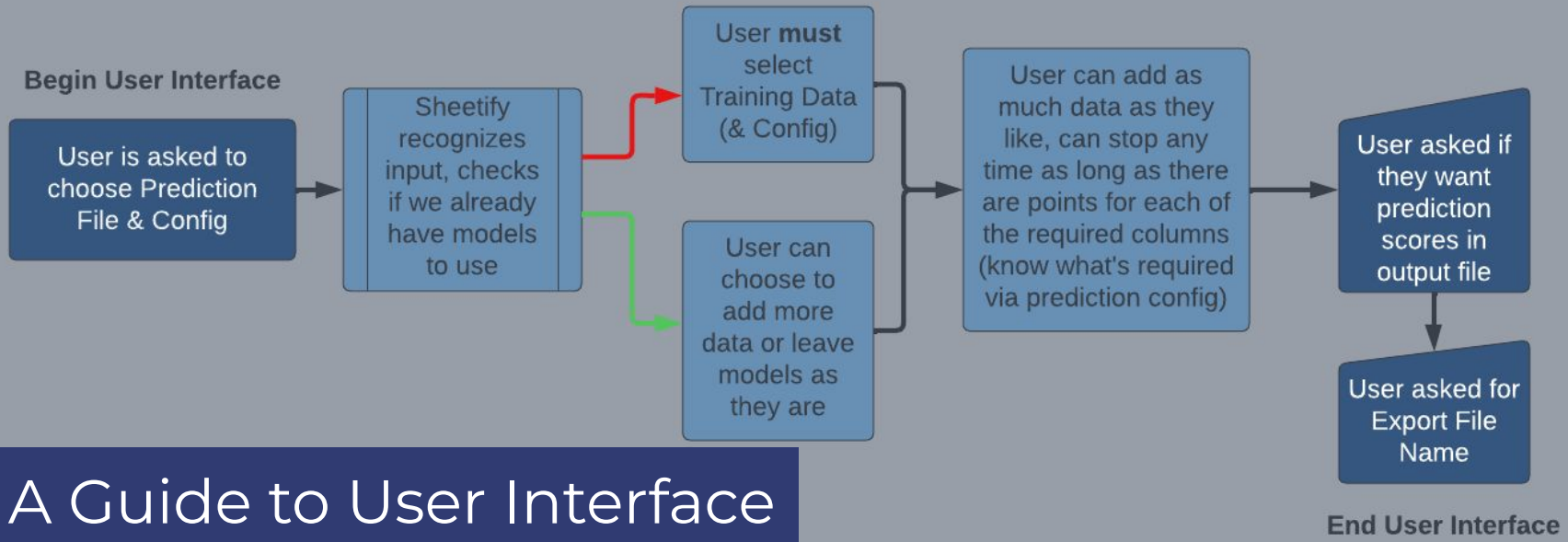
```
1   {
2       "Input" : {
3           "Device" : "controlProgram",
4           "Enumeration": "DeviceID",
5           "PointName" : "ObjectName",
6           "PointType" : "ObjectID",
7           "Units" : "units"
8       },
9       "Predicting" : [
10          "EquipmentLabel",
11          "FieldName",
12          "GeneralType",
13          "Required"
14      ]
15  }
```

Training Config

```
1   {
2       "Input" : {
3           "Device" : "controlProgram",
4           "Enumeration": "DeviceID",
5           "PointName" : "ObjectName",
6           "PointType" : "ObjectID",
7           "Units" : "units"
8       },
9       "Targets" : {
10          "EquipmentLabel" : "phredentitycode",
11          "FieldName" : "FieldName",
12          "GeneralType" : "generaltype",
13          "Required" : "required"
14      }
15  }
```

“Category” : “Name of corresponding column in data file”





A Guide to User Interface

Next Steps

Collect more Training Data & develop a better system to filter out noisy data

Make it so the user doesn’t have to use a Config file

Configure system for “Stemming”

Add statistics and metrics as a second sheet of the Excel output

Provide a pre-trained default models for non-DBO users who don’t have access to training data of their own

Issues with Training Data

ObjectName	ActualFieldName	PredictedFieldName
space_pressure_setpoint_bavi_3	building_air_static_pressure_sensor	building_air_static_pressure_setpoint
space_pressure_setpoint_bavi_2	building_air_static_pressure_sensor	building_air_static_pressure_setpoint
building_pressure_setpoint_bavo_1	building_air_static_pressure_sensor	building_air_static_pressure_setpoint
space_pressure_setpoint_bavi_1	building_air_static_pressure_sensor	building_air_static_pressure_setpoint
building_pressure_setpoint_bavo_1	building_air_static_pressure_sensor	building_air_static_pressure_setpoint

Issues with Training Data

Object Name	General Type	Actual FieldName	Predicted FieldName	Model Probability	Correct
bstp_active_bav_3	AHU	building_air_static_pressure_sensor	building_air_static_pressure_sensor	0.94081	TRUE
bstp_active_bav_4	AHU	building_air_static_pressure_sensor	building_air_static_pressure_sensor	0.94023	TRUE
bstp_active_bav_5	AHU	building_air_static_pressure_sensor	building_air_static_pressure_sensor	0.995	TRUE
bstp_active_av_6	AHU	building_air_static_pressure_sensor	building_air_static_pressure_sensor	0.9975	TRUE
bstp_active_av_3	AHU	building_air_static_pressure_setpoint	building_air_static_pressure_sensor	0.78635	FALSE
bstp_active_av_4	AHU	building_air_static_pressure_setpoint	building_air_static_pressure_sensor	0.79367	FALSE
bstp_active_av_5	AHU	building_air_static_pressure_sensor	building_air_static_pressure_sensor	0.99375	TRUE
bstp_active_av_3	AHU	building_air_static_pressure_sensor	building_air_static_pressure_sensor	0.78635	TRUE
bstp_active_av_4	AHU	building_air_static_pressure_sensor	building_air_static_pressure_sensor	0.79367	TRUE

Next Steps

Collect more Training Data & develop a better system to filter out noisy data

Make it so the user doesn’t have to use a Config file

Configure system for “Stemming”

Add statistics and metrics as a second sheet of the Excel output

Provide a pre-trained default models for non-DBO users who don’t have access to training data of their own

04

Demo & Questions

DEMO

QUESTIONS

THANKS

Thank you DB Engineering & Doug Blome for giving me the chance to work on this project!

This project would not have been the same without Trevor Sodorff, Nick Lima, Shane Spencer, Julien Ragbeer, Claire Stirdivant, & Tom Skoczylas

“I have no special talent. I am only passionately curious.”

- Albert Einstein

slidesgo