Unsupervised Anomaly Detection with Advanced Analytics: your next steps

London Meetup 24/1/2018

Speaker Bio: Harizo Rajaona

Data Scientist @Dataiku

Harizo has been working at Dataiku for one year, working with customers from financial services and CPG. Before joining Dataiku, he worked at the French Alternative Energies and Atomic Energy Commission, where he focused on designing statistical methods to reconstruct air pollution sources with operational applications in accidental cases and in emission monitoring contexts. He holds a PhD in Mathematics from University Lille 1.

Unsupervised anomaly detection

Jan.24th 2018 ● Harizo Rajaona (Data Scientist)

Analytics & Data science meetup - London

Outline

Summary & conclusion

Which algorithms? How to benchmark them?

How to properly report outliers ?

Introduction: context, supervised vs. unsupervised

About Dataiku

- Founded in 2013 (5y. birthday party 1 week ago!)
- 110+ employees, 120+ clients
- Paris, NYC, London, Munich

DESIGN	PREPARE Load and prepare your data	ANALYSE Visualize and share your work	MODEL Build your models
PRODUCTION	AUTOMATE Re-execute your workflow at ease	MONITOR Follow your production environment	SCORE Get predictions in real time

data iku

Introduction: context, supervised vs. unsupervised

Anomaly detection (AD) in the wild

Some examples

- **Fraud detection:**
  - Detecting fraudulent credit card transactions
  - Flagging suspicious warranty/service claims

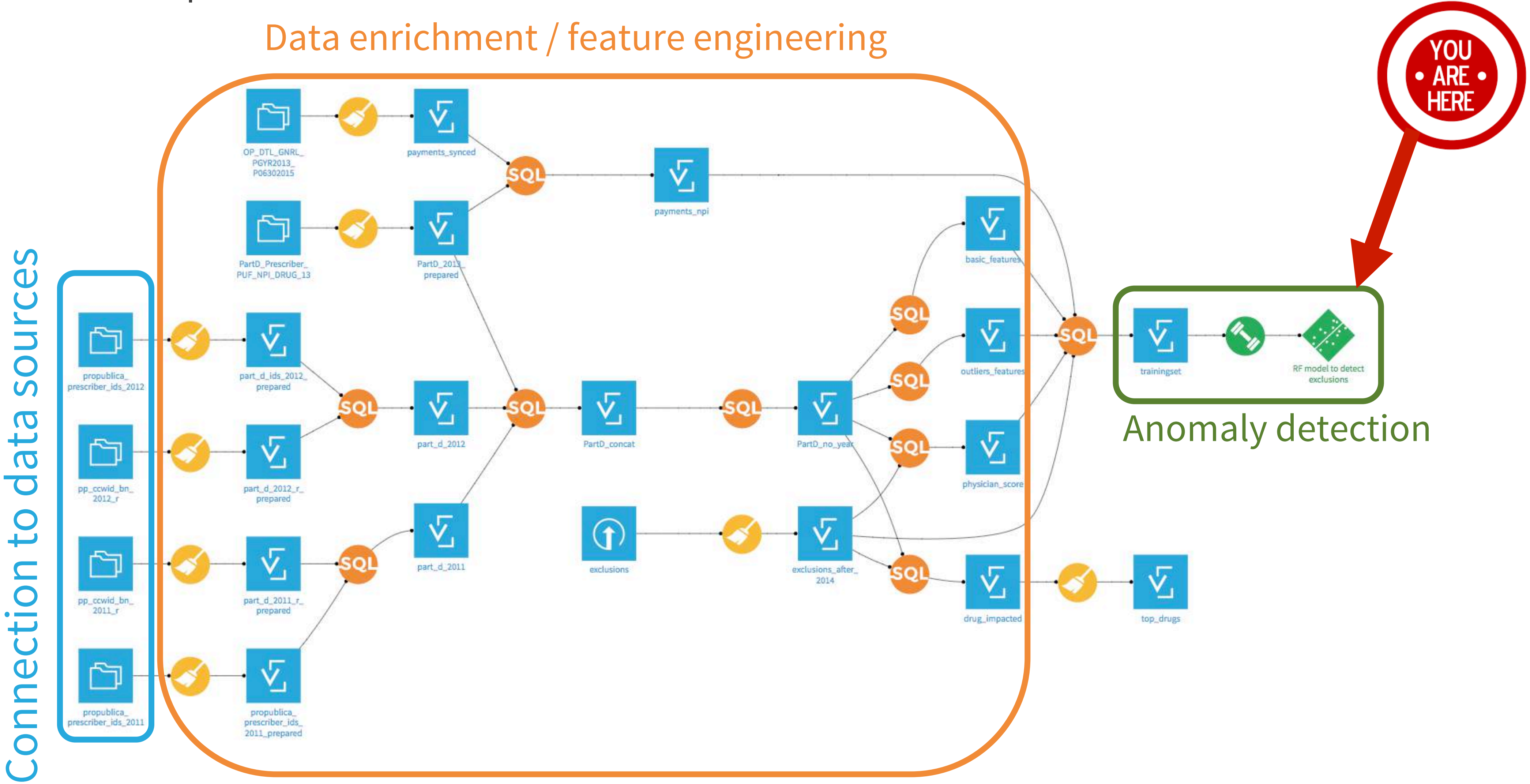
- **Predictive maintenance:**
  - Tracking a machine's abnormal behaviour in a manufacturing plant

- **Healthcare:**
  - Looking for unusual/unobserved symptoms in a group of patients

- **Network monitoring:**
  - Identify non-idle traffic surges/drops
  - Trigger alerts in case of breachs/intrusions

data iku

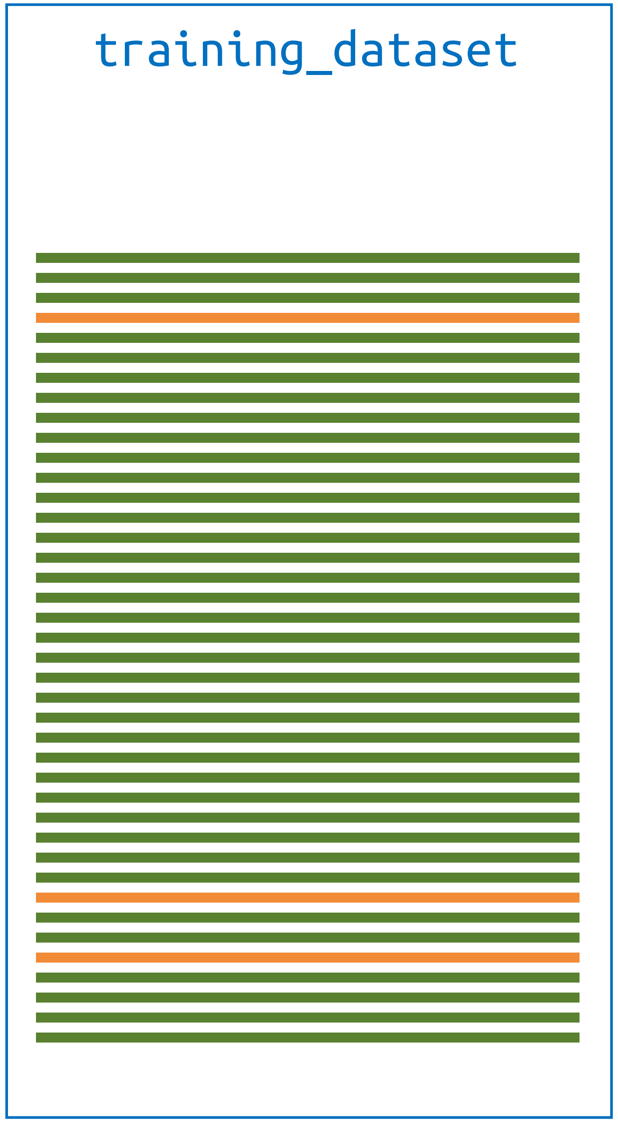
AD: one step in the data science workflow



(Example of a flow in Dataiku DSS for Medicare fraud detection)

dataiku

Definitions & the supervised case



Inliers:

- Vast majority of the data
- Represent an "expected behaviour"

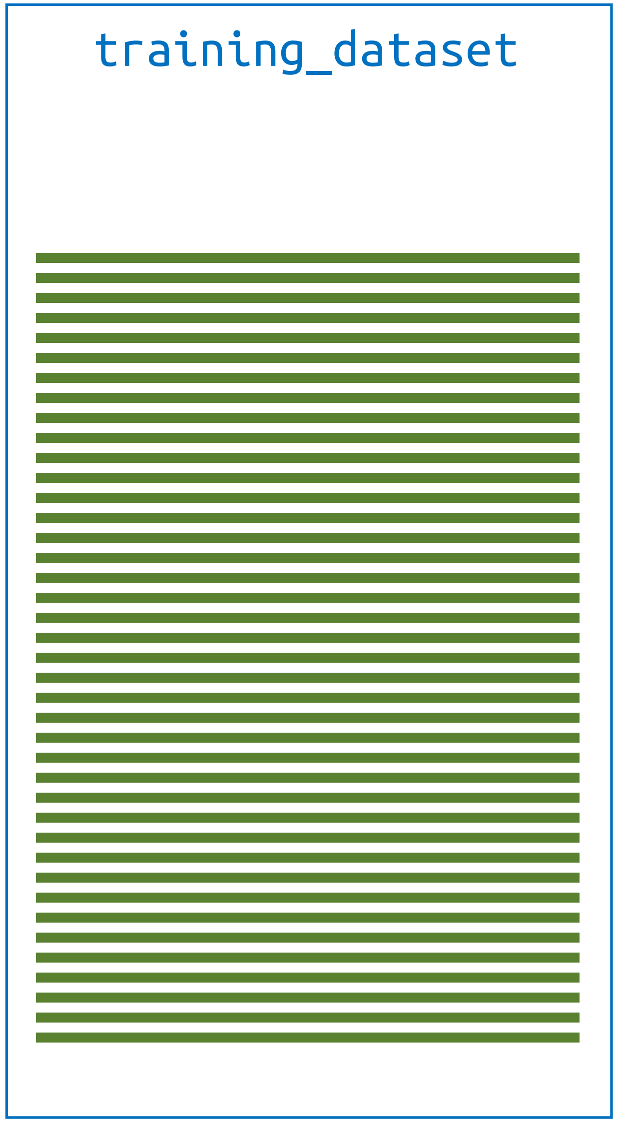
Outliers:

- (Very!) small portion of the data
- Abnormal cases

Supervised AD: data is labelled (inlier/outlier)

- Binary classification problem
- Account for high class imbalance
- Cross-validate carefully (stratified splits)

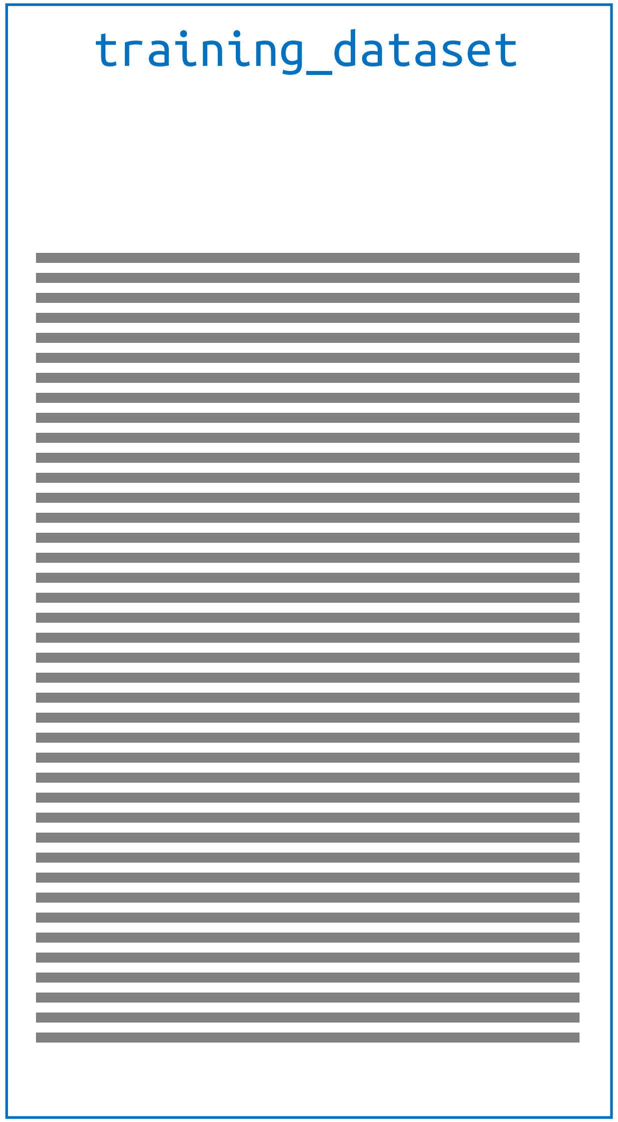
The semi-supervised case



Semi-supervised AD: data contains only inliers

- "Learning normality"(one-class)
- Outliers that appear *at test time* are flagged (novelty detection).

The unsupervised case



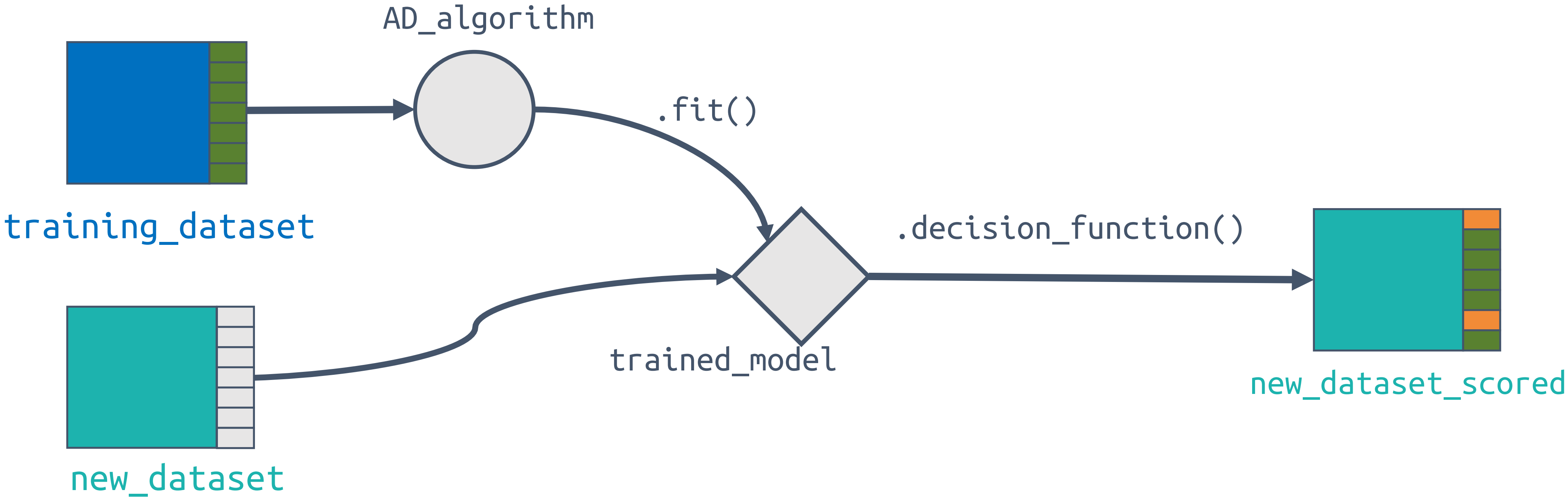
Unsupervised AD (UAD) / outlier detection:

- No labels at all
- Data contains a **sufficiently small amount of outliers**

- Happens quite often !
  - Costly label processes
  - Unreliable labels
  - Potential overfitting on existing sample

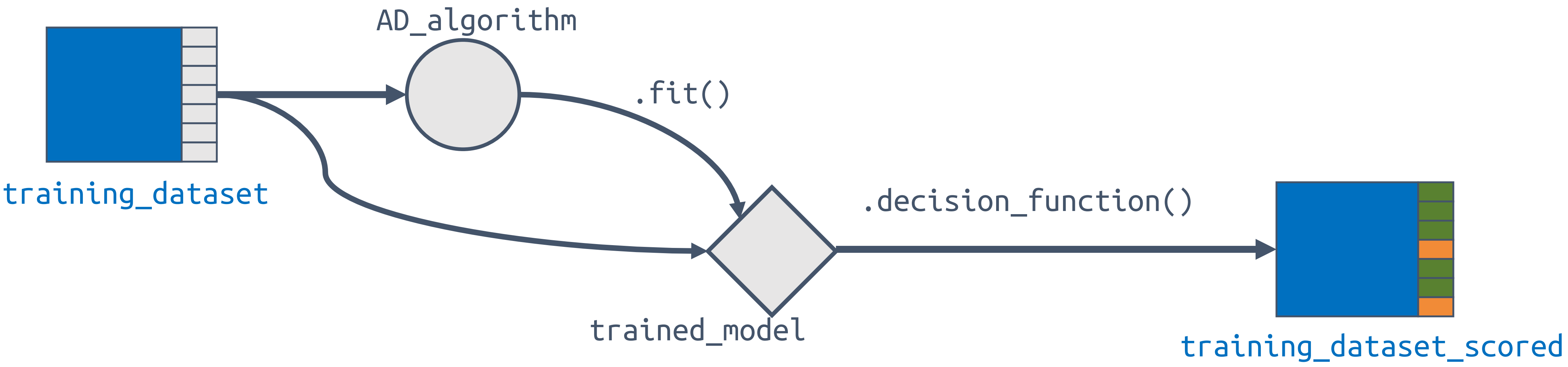
Anomaly detection workflows

Semi-supervised



Anomaly detection workflows

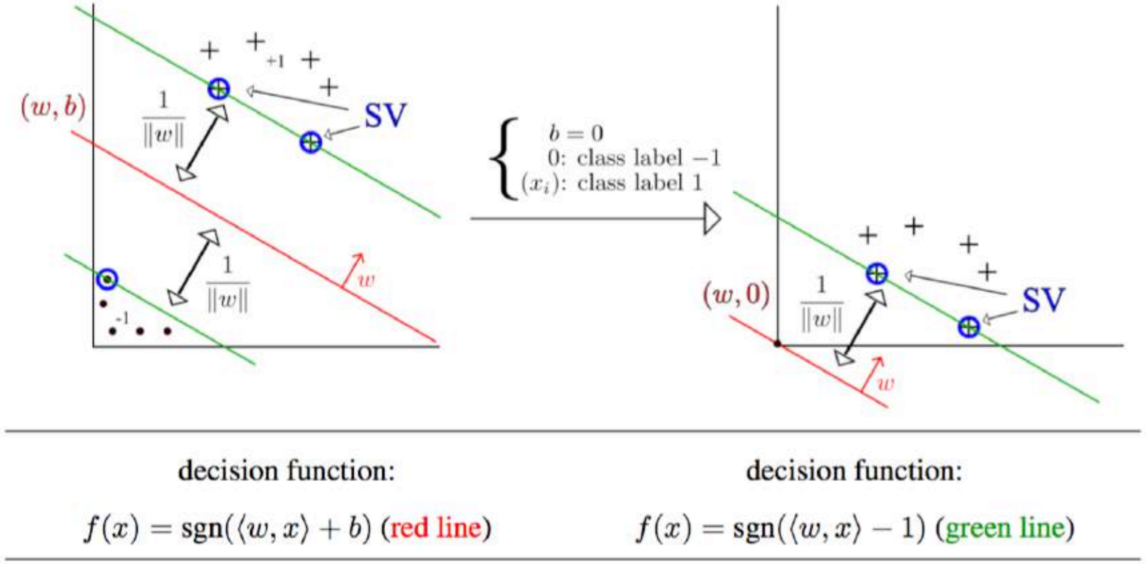
Fully unsupervised



Which algorithms? How to benchmark them?

One-class support vector machines (OCSVM)

A kernel-based method

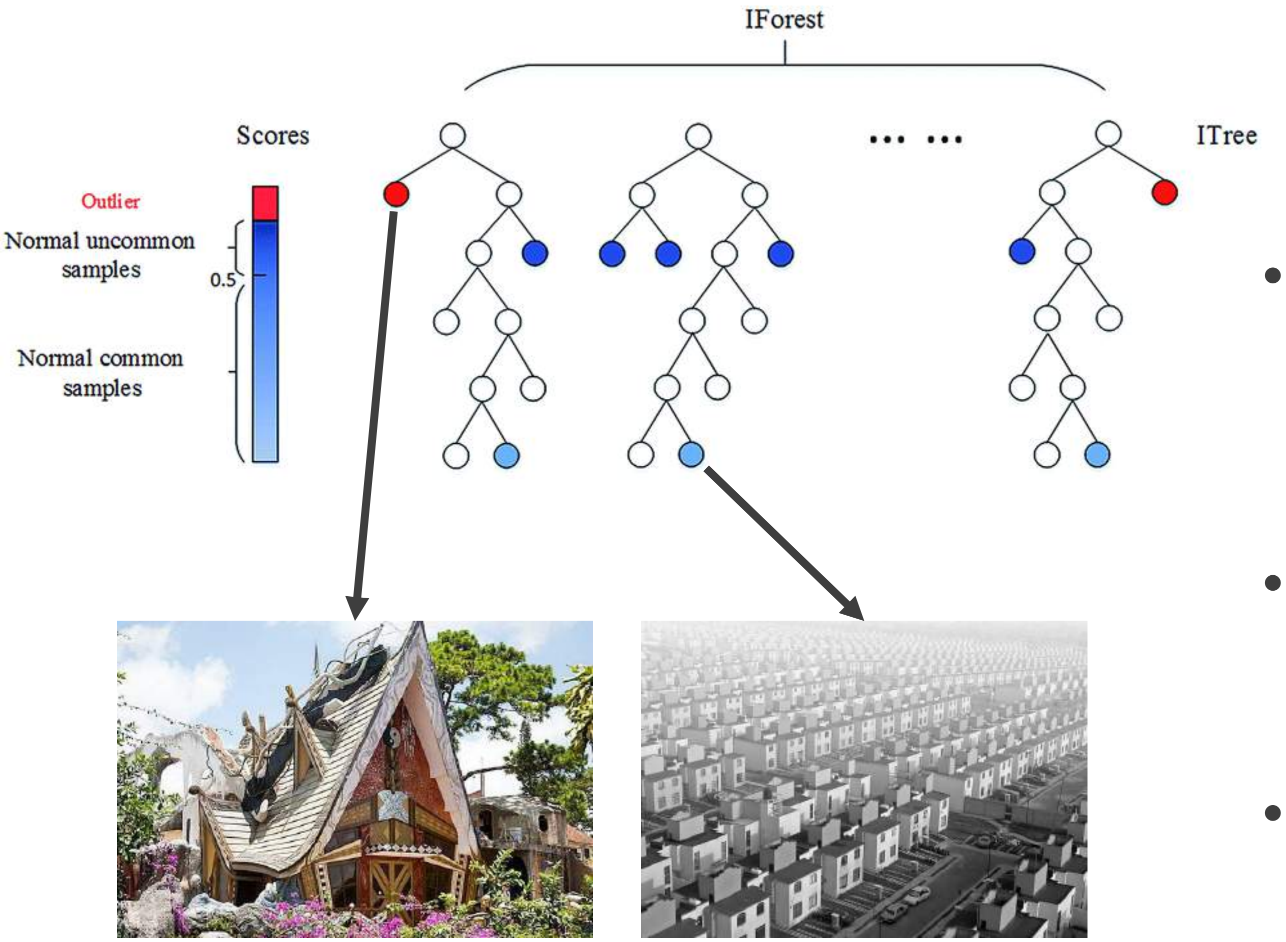


(Machine learning and extremes for anomaly detection, N. Goix, 2016)

- SVM classification problem: find the optimal separation between classes with maximum margin rule
- One-class SVM: find the optimal separation **between the inlier class and the origin** with maximum margin rule
- Can handle highly non-linear cases
- Non-trivial settings (kernel, hyperparams), struggles with high dimensionality

Isolation forest

A tree-based method

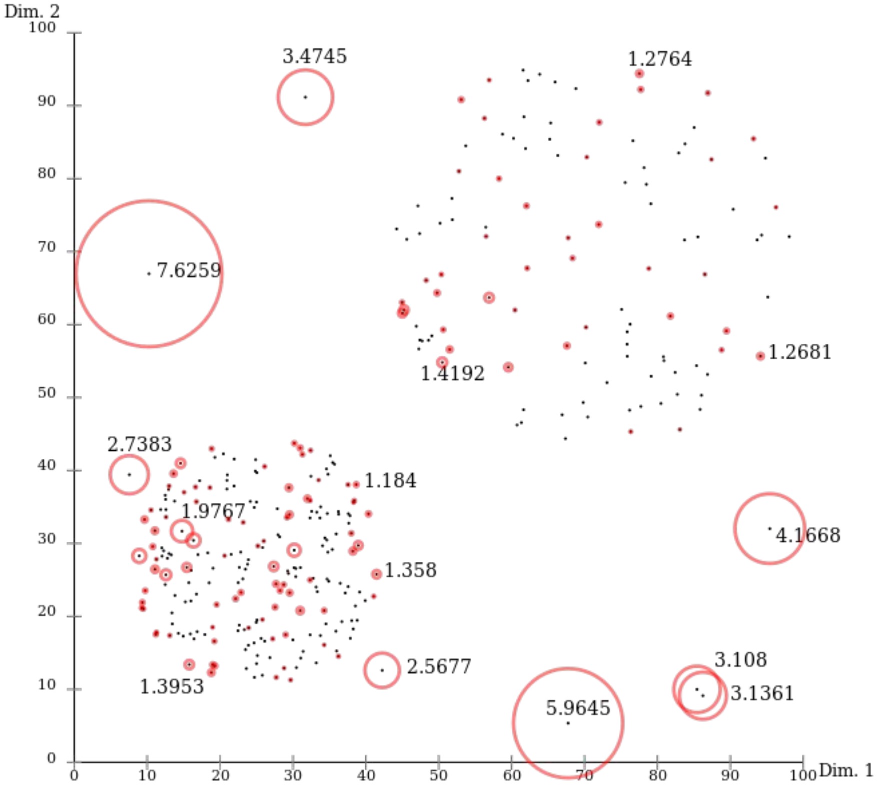


(Representative subset selection and outlier detection via isolation forest, Chen et al, 2016)

- Ensemble of randomized decision trees (random features and splits)
- Outliers have the shortest average path length over the whole set of trees
- Intuitive interpretation
- Scalable implementation (parallelizable)

Local outlier factor (LOF)

A distance-based method



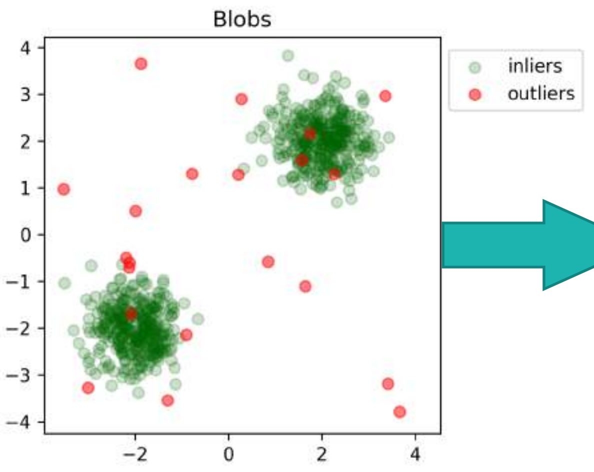
(https://en.wikipedia.org/wiki/Local_outlier_factor)

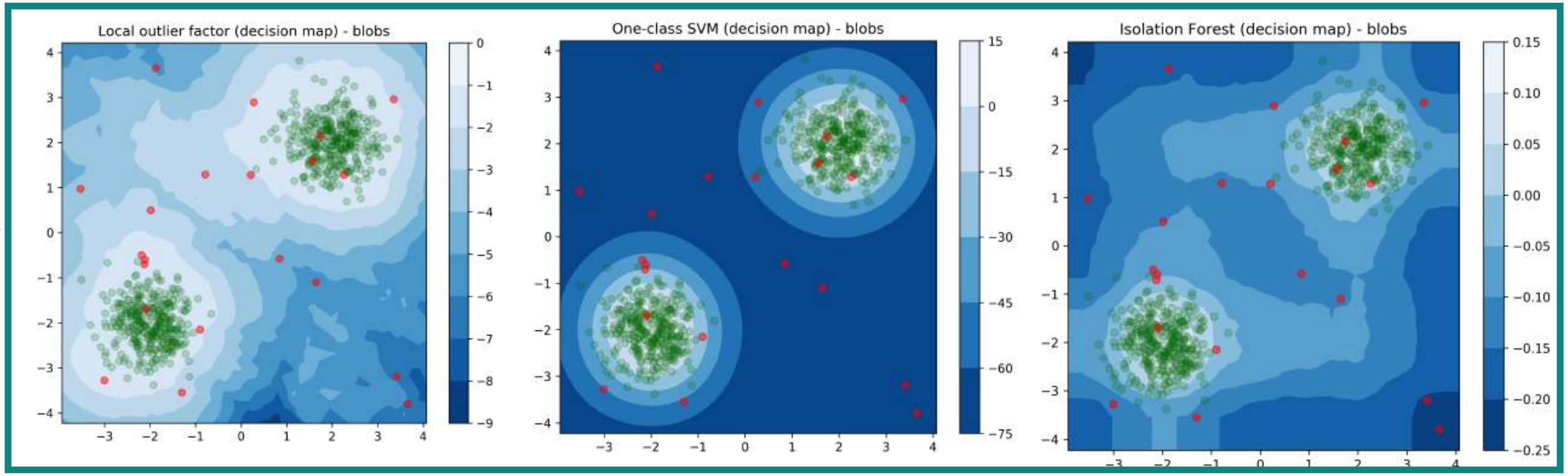
- Compares the local density of a point with the densities of its neighbors.
- « If I'm lonely and my neighbors are too, then I'm normal. »
- Allows to define relative outliers which **may not reflect a totally abnormal state**
- Distance-based -> computation load increases with dimensionality
- Full pass over dataset at evaluation time (costly)

Performance evaluation (simple dataset)

How well are we doing ?

- To evaluate the performance, we actually need labels (**that are NOT used at train time)**
- Decision maps (only for simple 2D cases): apply `.decision_function()` to a grid over the feature space





- Not appropriate for more complex datasets !

Performance evaluation (real datasets)

How well are we doing ?

- Datasets from Kaggle (credit card fraud detection) and academic benchmarks (internetads, pageblock)

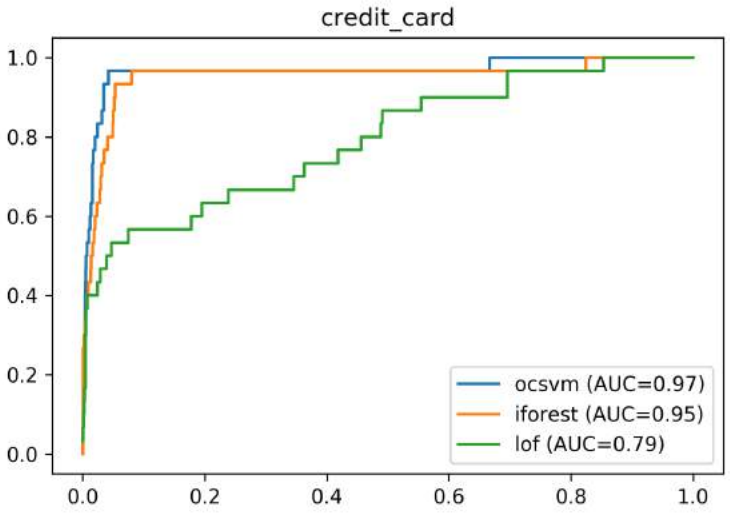
- Cover some configurations often met in « real-life » datasets (categorical features, high dimensionality)

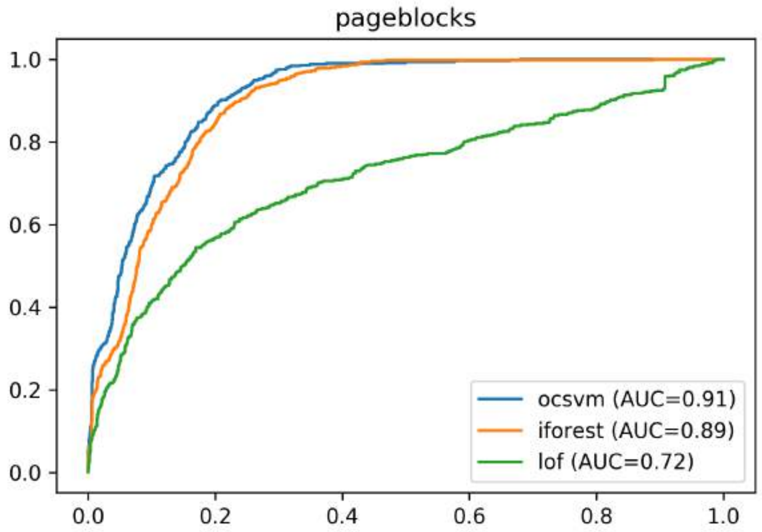
Dataset	#rows	#columns	Outlier ratio	Feature types
Credit card (Kaggle), 6% sample	17088	29	0.002	Numerical
pageblocks	4883	510	0.095	Numerical
internetads	1966	1554	0.187	Binary

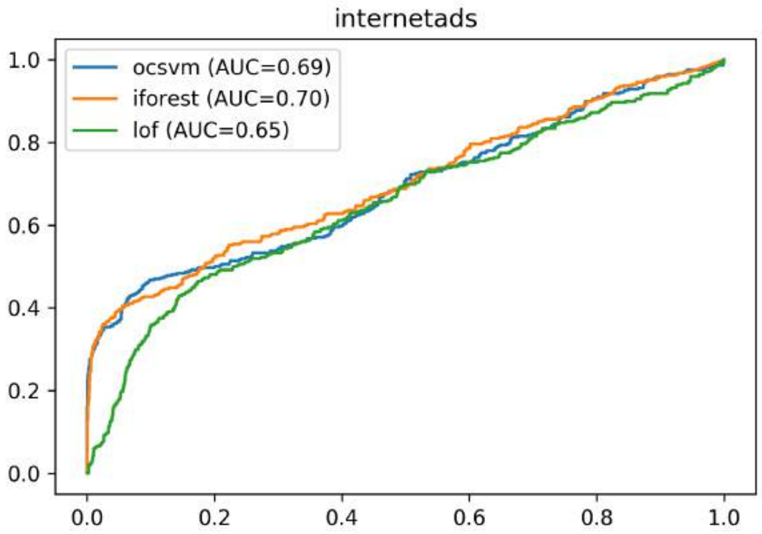
Performance evaluation (real datasets)

How well are we doing ?

- <u>Receiver Operator Characteristic (ROC) curve:</u>
  - Same logic as a supervised binary classification problem.
  - Labels are available but NOT **used at training time!**
  - Area Under the Curve (AUC) provides a numerical metric of performance







Performance evaluation (real datasets)

How well are we doing?

- Computation times matter as well !
- Fit / score times in seconds:

Dataset (shape)	IsolationForest(n_jobs=-1)	OneClassSVM	LocalOutlierFactor
Credit_card (17088, 29)	**1.33** **0.65**	19.31 6.92	**32.56** **32.99**
Pageblocks (4883, 510)	0.51 0.16	0.90 0.41	0.27 0.24
Internetads (1966, 1554)	3.26 1.80	5.64 3.64	**10.38** **10.30**

dataiku

How to properly report outliers?

Anomaly scoring

One does not simply **`predict_proba()`**

- Anomaly scores are purely mathematical and not easily interpretable!

$$s(x, \psi) = 2^{\frac{-\mathbb{E}[h(x)]}{c(\psi)}}$$

(isolation forest)

$$\mathrm{lof}_k(x) = \frac{\sum_{o \in N_k(x)} \left(\frac{\mathrm{lrd}_k(o)}{lrd_k(x)} \right)}{|N_k(x)|}$$

(local outlier factor)

- Scores are algorithm-specific -> not the same scale !

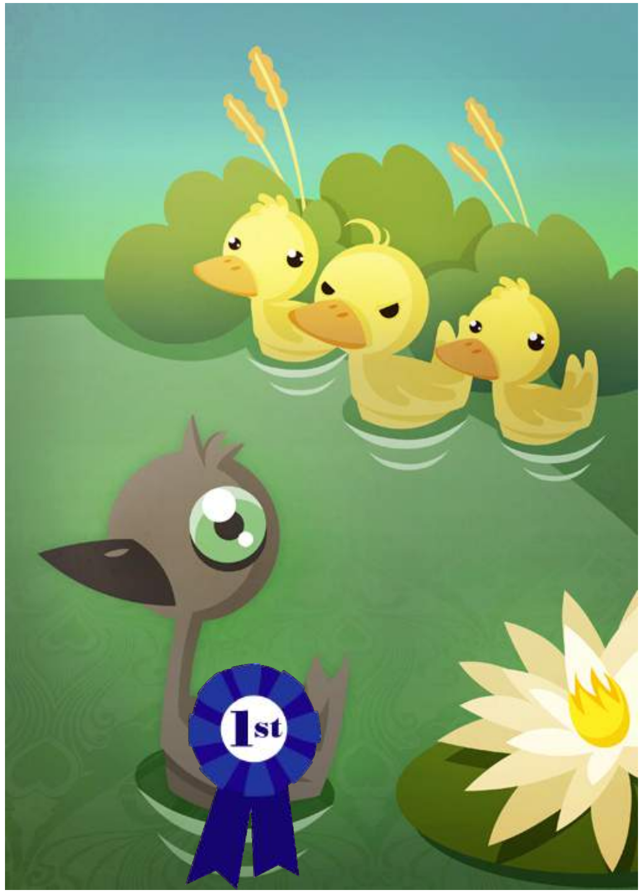
- … So how do we compare predictions across multiple algorithms ?

It’s actually a ranking problem !

An interpretable prediction system

One possible solution: run multiple algorithms and rank data points by anomaly score :

1. Choose a reference algorithm
2. Compute dense ranks for each algorithm score column
3. Sort by increasing rank **for the reference algorithm**
4. Compare top N points and **see if other algorithms « agree » on ranking values**



It's actually a ranking problem !

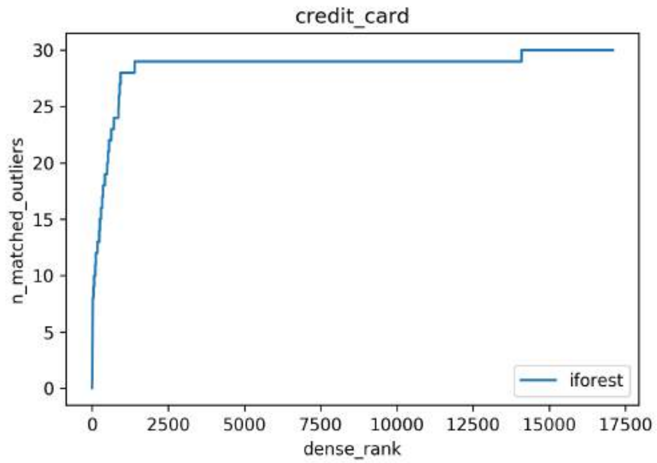
An interpretable prediction system

- Example on isolation forest's top 10 on the credit_card dataset:

	y_true	iforest_score	ocsvm_score	lof_score
3948	0	0.192481	1763.174401	3.699082
7723	1	0.189296	1763.174400	3.035509
1903	1	0.187674	1763.174401	4.620851
14861	1	0.172651	1763.174399	2.526187
1269	1	0.167669	1763.162092	2.201619
16129	0	0.161433	1763.174362	3.475972
9752	0	0.161160	1763.174401	3.970376
4548	1	0.159525	1763.174401	2.765399
2947	0	0.154045	1762.652671	1.782176
6966	0	0.152313	1763.174401	2.683288

dense_rank + sort

	y_true	iforest_rank	ocsvm_rank	lof_rank
3948	0	1.0	5.0	5.0
7723	1	2.0	7.0	13.0
1903	1	3.0	1.0	1.0
14861	1	4.0	9.0	39.0
1269	1	5.0	16.0	85.0
16129	0	6.0	10.0	9.0
9752	0	7.0	1.0	2.0
4548	1	8.0	1.0	27.0
2947	0	9.0	61.0	285.0
6966	0	10.0	4.0	32.0



~ ROC curve !

dataiku

Prediction errors

Dealing with false positives/negatives

Types of errors and their business impact:

- **False positives** (high rank but NOT actual outliers)
  - Bad customer experience
  - Costly investigation (time + money)

- **False negatives** (low rank but actual outliers)
  - Potential (high) financial damage

	y_true	iforest_rank	ocsvm_rank	lof_rank	
3948	0	1.0	5.0	5.0	☹ FP
7723	1	2.0	7.0	13.0	
1903	1	3.0	1.0	1.0	☺
14861	1	4.0	9.0	39.0	
1269	1	5.0	16.0	85.0	😐
16129	0	6.0	10.0	9.0	
9752	0	7.0	1.0	2.0	☹ FP
4548	1	8.0	1.0	27.0	
2947	0	9.0	61.0	285.0	
6966	0	10.0	4.0	32.0	

	y_true	iforest_rank	ocsvm_rank	lof_rank
839	1	12263.0	11356.0	14544.0
12708	1	1338.0	744.0	6213.0
6103	1	573.0	559.0	102.0
3187	1	571.0	434.0	424.0
14770	1	523.0	615.0	101.0

☹ FN

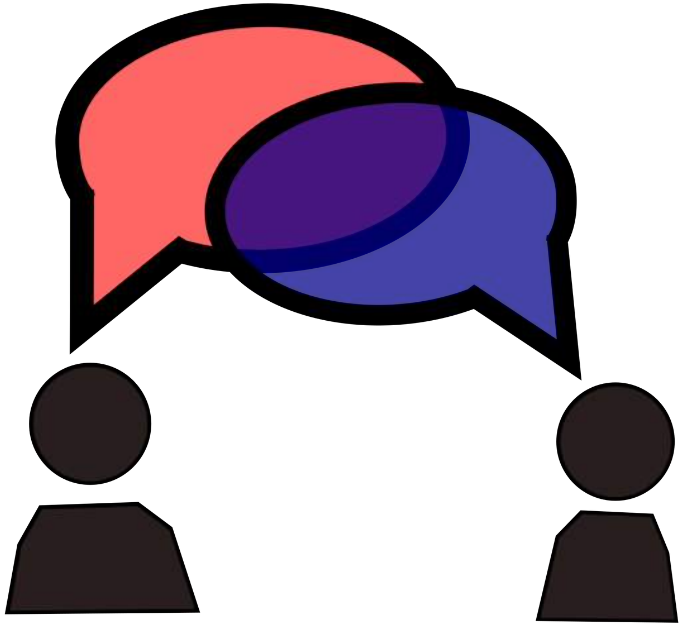
data
iku

Prediction errors

Dealing with false positives/negatives

Solutions:

- **Average rankings** -> more robust estimation
- Build **custom filters/rules** (e.g. amount threshold, projected damage…)
- Rely heavily on **domain knowledge experts** !



Communication between data scientists and domain experts is paramount !

Summary & conclusion

Summary & conclusion

- Outlier detection is a first step to take when looking for suspicious points in an unlabelled dataset.
  - Provides a « **short list** » that can be further investigated
  - Several algorithms and implementations available (Python, R)

- Evaluating outlier detection algorithms is difficult:
  - Often requires **actual label availability**
  - Choice of hyperparameters
  - Area of **active research** (check out AD workshop @ICML)

- Reporting anomalies should be done in a transparent way:
  - Actions taken on outliers are **business-driven**
  - Algorithms are meant to **help domain experts, not replace them**

dataiku

Summary & conclusion

Unsupervised anomaly detection is evolving fast:

- Other types of algorithms available:
  - **Distance-based**: variations of the LOF (e.g. COF, ODIN…)
  - **Statistical methods** (e.g. elliptic envelope fitting)
  - **Clustering-based:** flag points that don't fit with inferred clusters (e.g. DBSCAN)

- Novel approaches using deep learning for anomaly detection:
  - Images: *Learning discriminative reconstructions for unsupervised outlier removal*, Xia et al., 2015
  - Time series: *Engineering extreme event forecasting at Uber with recurrent neural networks*, Uber, 2017

Thank you for your attention !

Speaker Bio: Silviu Tofan

Data Scientist @Dataiku

Silviu is working as a data scientist at Dataiku. Coming from a business-oriented background, with an MSc in Business Analytics from the University of Manchester, he transitioned to a more technical focus while working on an optimization problem together with ARM. Before coming to Dataiku, Silviu was involved with social housing organizations to develop their data science capabilities.

Anomaly Detection in Aviation

Silviu Tofan - Data Scientist

London Meetup - 24 January 2018

Outline

Problem Introduction

What exactly is our airplane problem?

Methodology

What are we trying to solve? How do we actually do it?

Deployment

How do we deploy this into a production environment?

Monitoring

How do we showcase the work and track performance?

Next Steps

Where do we go from here?

Problem Introduction

or what is this guy even talking about

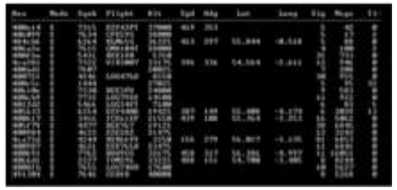
Flight Tracking Data

ETL explained for Aviation Enthusiasts

HPE Edgeline Server

Real-time Data Capture

DUMP1090



Decoding Signals &
Single Stream

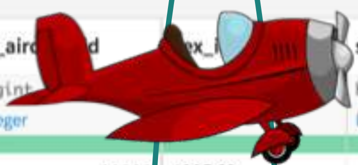
Apache Kafka

Publish Messages
Over Kafka Topics

Microfocus Vertica

Micro-batch Into Tables

Collected Data



Viewing dataset sample Configure sample

10000 rows, 16 cols

10000 matching

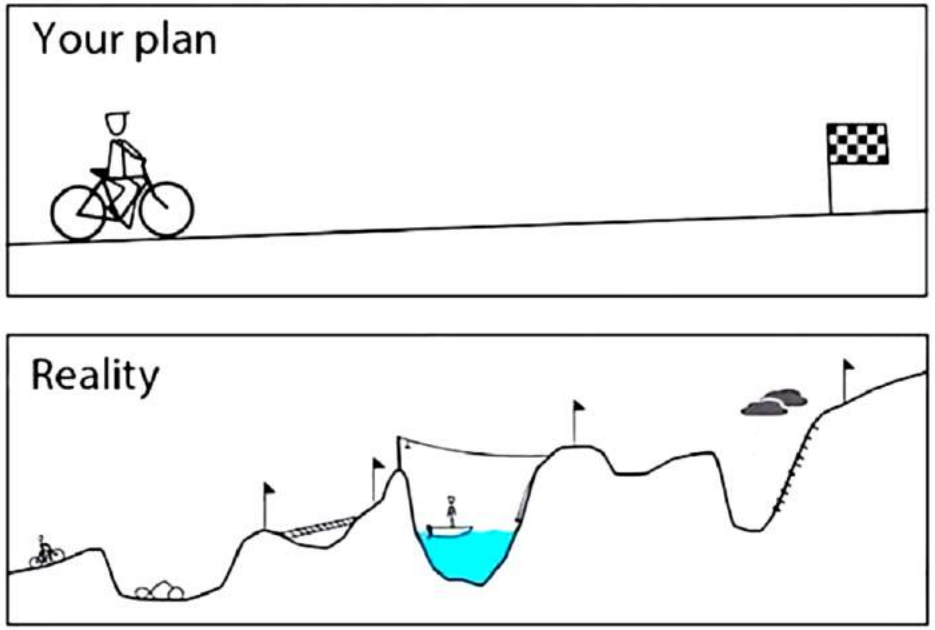
dump1090_sitename	record_type	record_type_number	sg_session_id	sg_airc…d	…ex_i…	sg_flight_id	…	msg_log_ts	altitude	…itude	…gitude
string	string	bigint	bigint	bigint		bigint	str…	string	bigint		
Natural lang.	Text	Integer	Integer	Integer		Integer	Date…	Date (unparsed)	Integer		
pennardel01	MSG	3	111	11111	400D69	111111	201…088	2017-07-20 20:20:04.053	257…	47.24579	7.44598
pennardel01	MSG	3	111	11111	400D69	111111	2017-07-20 20:20:09.138	2017-07-20 20:20:09.108	2557…	47.23879	7.4395
pennardel01	MSG	3	111	11111	400D69	111111	2017-07-20 20:20:10.038	2017-07-20 20:20:10.026	25550	47.23727	7.43809
pennardel01	MSG	3	111	11111	400D69	111111	2017-07-20 21:37:15.083	2017-07-20 21:37:15.072	25400	46.70151	7.15498
pennardel01	MSG	3	111	11111	400D69	111111	2017-07-20 21:37:17.682	2017-07-20 21:37:17.635	25475	46.70474	7.16109
pennardel01	MSG	3	111	11111	4054A7	111111	2017-07-21 00:09:49.756	2017-07-21 00:09:49.749	38025	47.43547	7.29689
pennardel01	MSG	3	111	11111	400D69	111111	2017-07-21 08:16:47.317	2017-07-21 08:16:47.299	25900	47.24162	7.46459
pennardel01	MSG	3	111	11111	400D69	111111	2017-07-21 08:16:48.328	2017-07-21 08:16:48.287	25850	47.24047	7.46324
pennardel01	MSG	3	111	11111	400D69	111111	2017-07-21 08:16:51.088	2017-07-21 08:16:51.039	25750	47.2367	7.45859
pennardel01	MSG	3	111	11111	400D69	111111	2017-07-21 08:16:52.167	2017-07-21 08:16:52.149	25700	47.23526	7.45683
pennardel01	MSG	3	111	11111	400D69	111111	2017-07-21 08:16:52.648	2017-07-21 08:16:52.61	25675	47.23441	7.45577
pennardel01	MSG	3	111	11111	400D69	111111	2017-07-21 08:16:55.589	2017-07-21 08:16:55.558	25575	47.23064	7.45112
pennardel01	MSG	3	111	11111	4054A7	111111	2017-07-21 08:50:54.112	2017-07-21 08:50:54.095	38025	46.33358	5.4672…
pennardel01	MSG	3	111	11111	4054A7	111111	2017-07-21 08:50:55.186	2017-07-21 08:50:55.148	38025	46.33177	5.46975
pennardel01	MSG	3	111	11111	4054A7	111111	2017-07-21 08:50:55.583	2017-07-21 08:50:55.541	38025	46.33116	5.47064
pennardel01	MSG	3	111	11111	4054A7	111111	2017-07-21 08:50:56.083	2017-07-21 08:50:56.064	38025	46.33056	5.47147
pennardel01	MSG	3	111	11111	4054A7	111111	2017-07-21 08:53:14.909	2017-07-21 08:53:14.879	38025	46.12172	5.76645
pennardel01	MSG	3	111	11111	4054A7	111111	2017-07-21 08:53:15.329	2017-07-21 08:53:15.328	38025	46.12112	5.76727
pennardel01	MSG	3	111	11111	4054A7	111111	2017-07-21 08:53:15.878	2017-07-21 08:53:15.855	38025	46.12019	5.76856
pennardel01	MSG	3	111	11111	4054A7	111111	2017-07-21 08:53:16.289	2017-07-21 08:53:16.252	38025	46.1196	5.76943
pennardel01	MSG	3	111	11111	4054A7	111111	2017-07-21 08:53:17.789	2017-07-21 08:53:17.758	38050	46.11717	5.77285

Methodology

or how we came up with all this stuff

Defining a Goal

“Can we say that the current **path** followed by a particular **flight** is **abnormal**?”



Separation of Flights

Some aviation knowledge

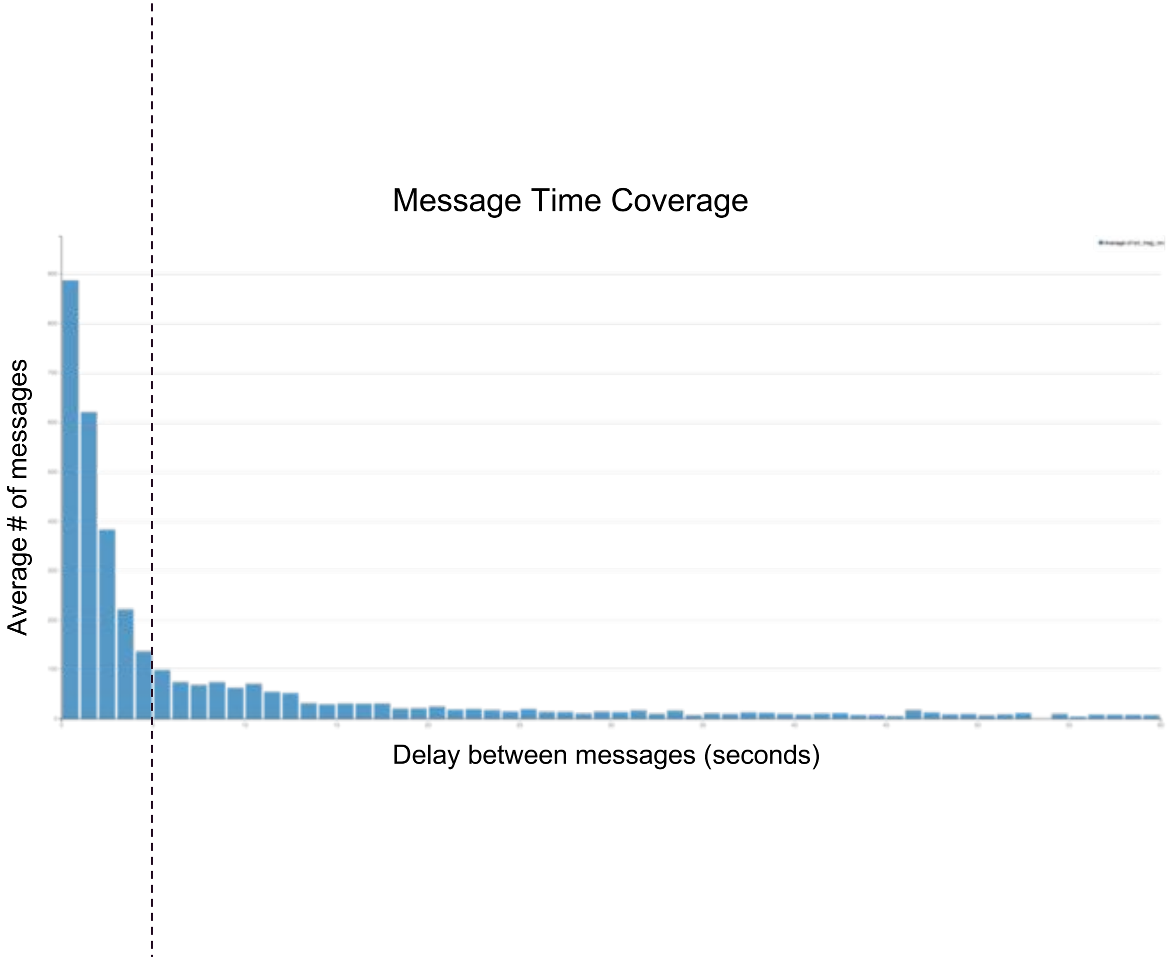
Some data checks

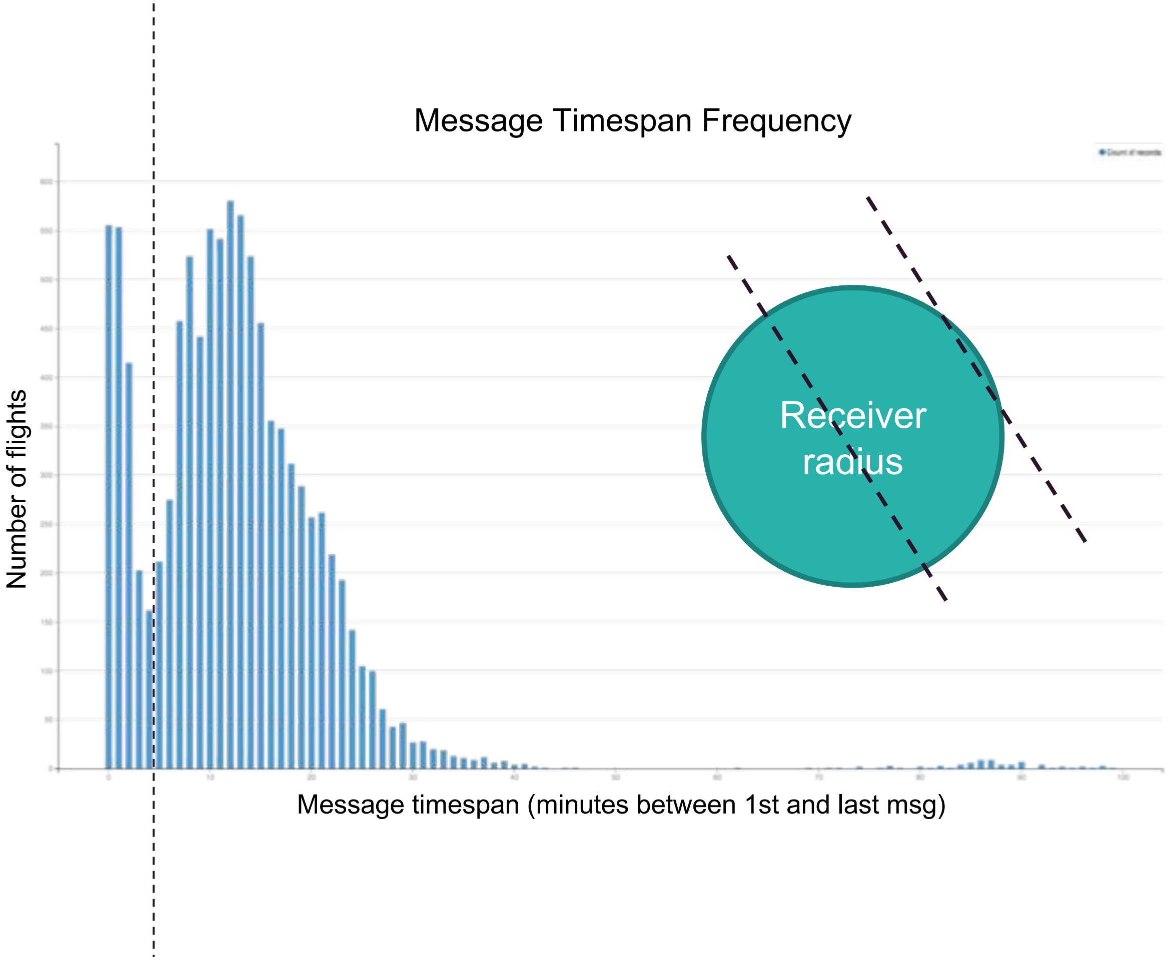
Some subjectivity

Some Vertica SQL

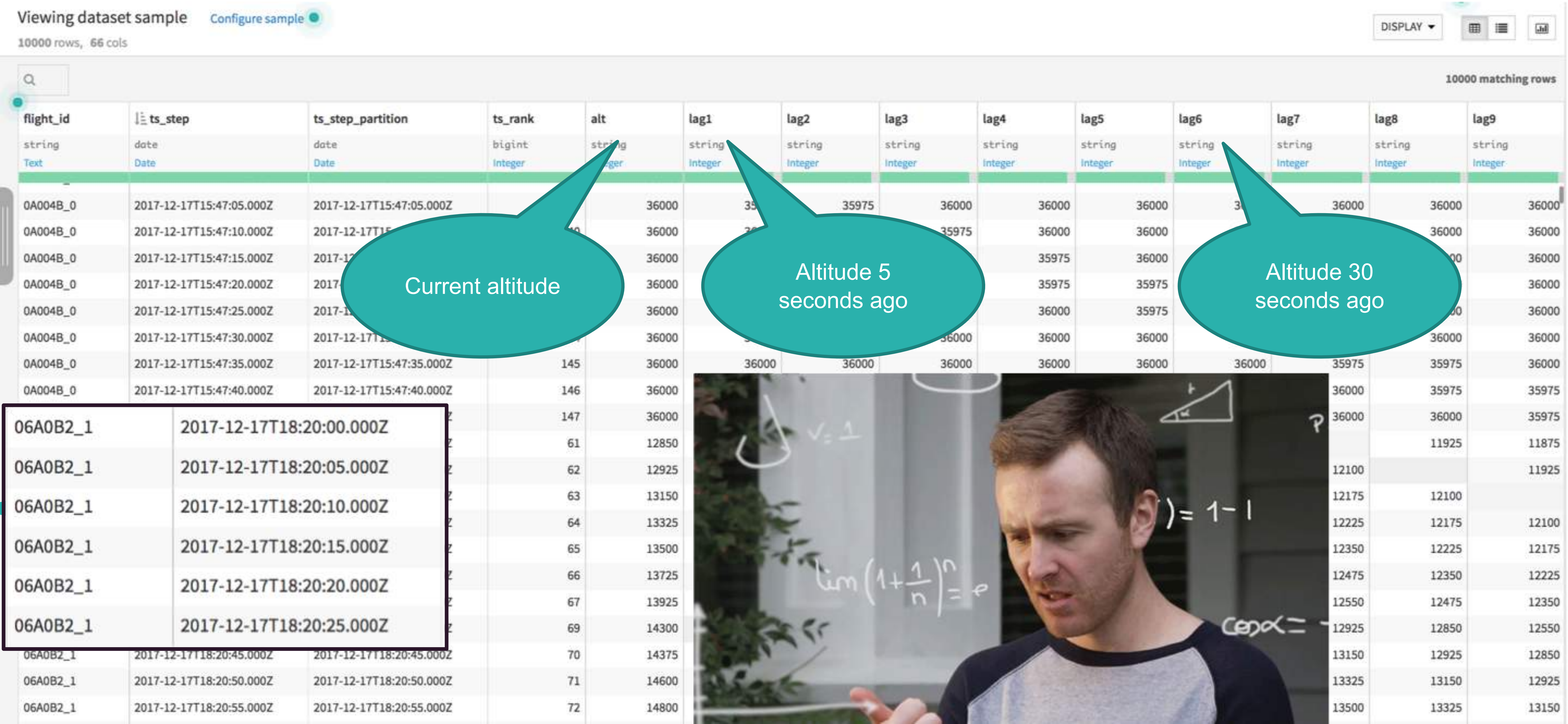
```
SELECT *,
  hex_ident||'_'||conditional_true_event(msg_gen_ts - LAG(msg_gen_ts) > '1 HOUR')
    OVER (PARTITION BY hex_ident ORDER BY msg_gen_ts) AS flight_id
FROM "dump1090_kafka"."dump1090_msg_3"
```

Feature Creation





Prepared Dataset

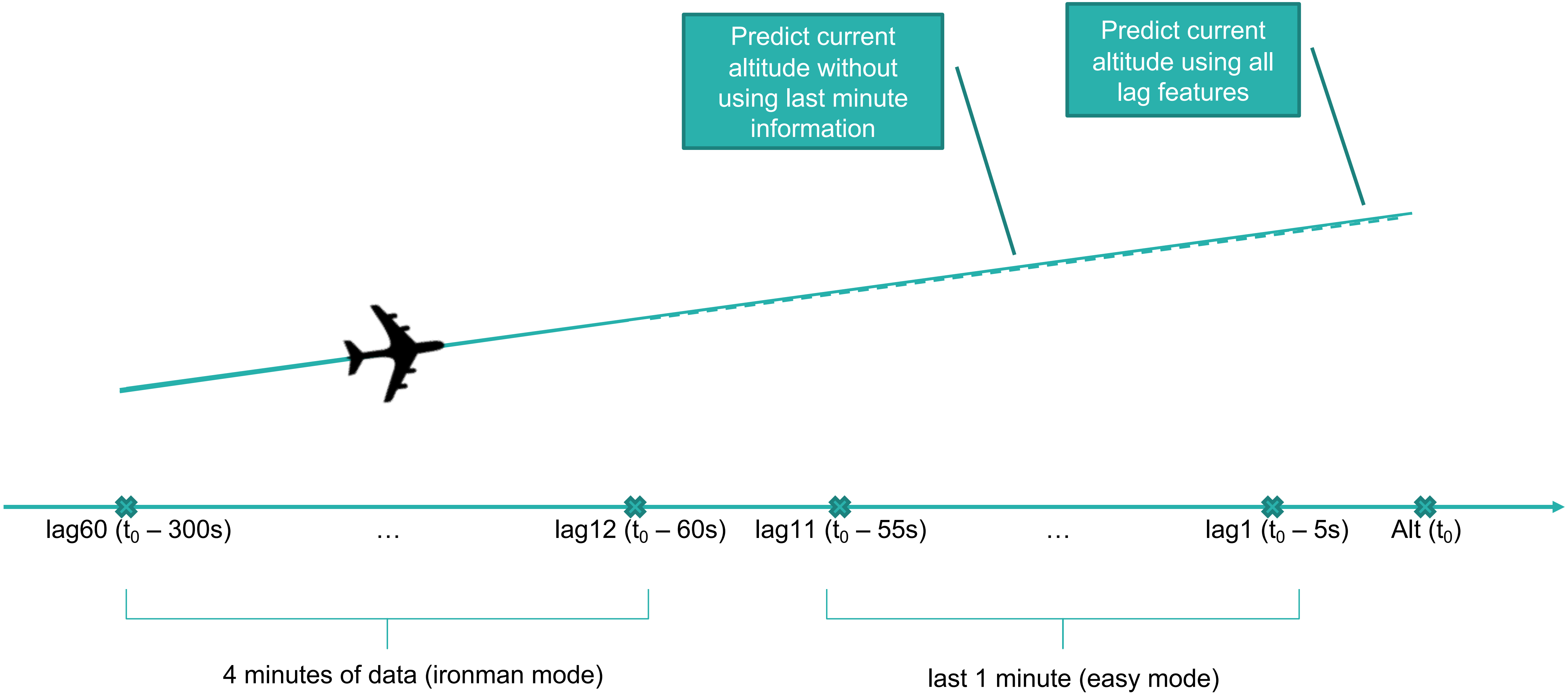


We Have to Adapt

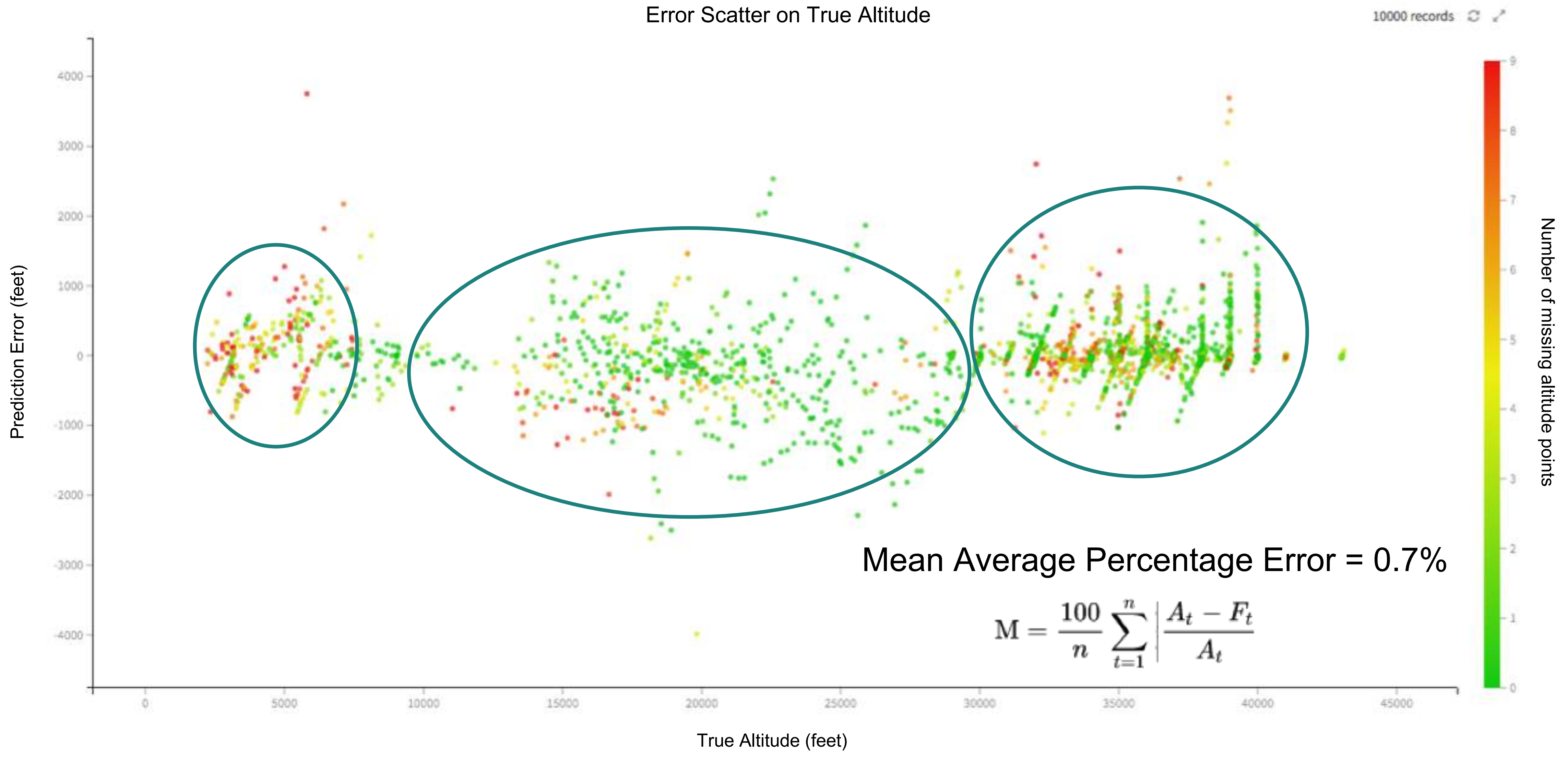
"The difficulty lies, not in the new ideas, but in escaping from the old ones, which ramify, for those brought up as most of us have been, into every corner of our minds."

John Maynard Keynes, Economist

Altitude Timeline



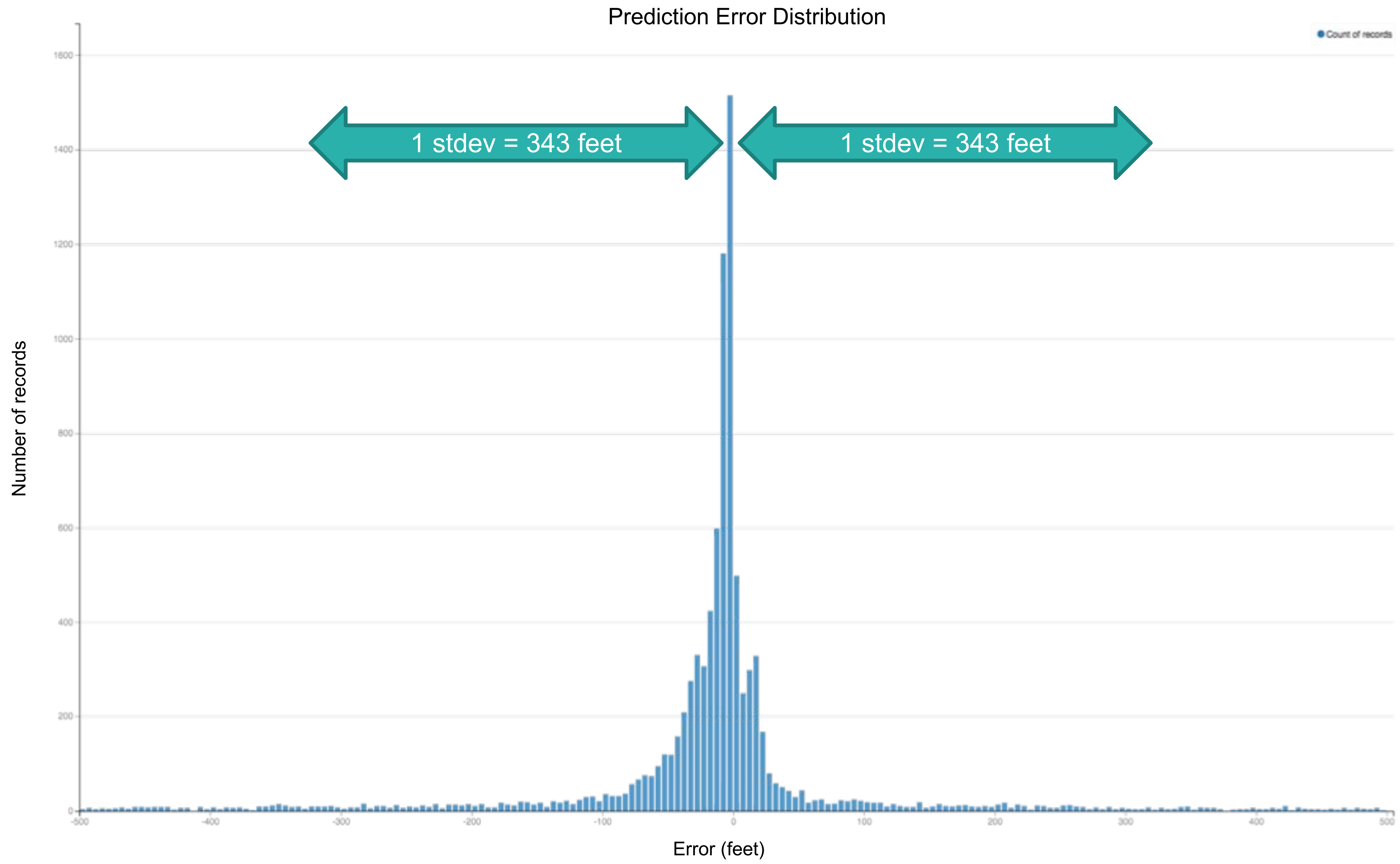
Altitude Predictions



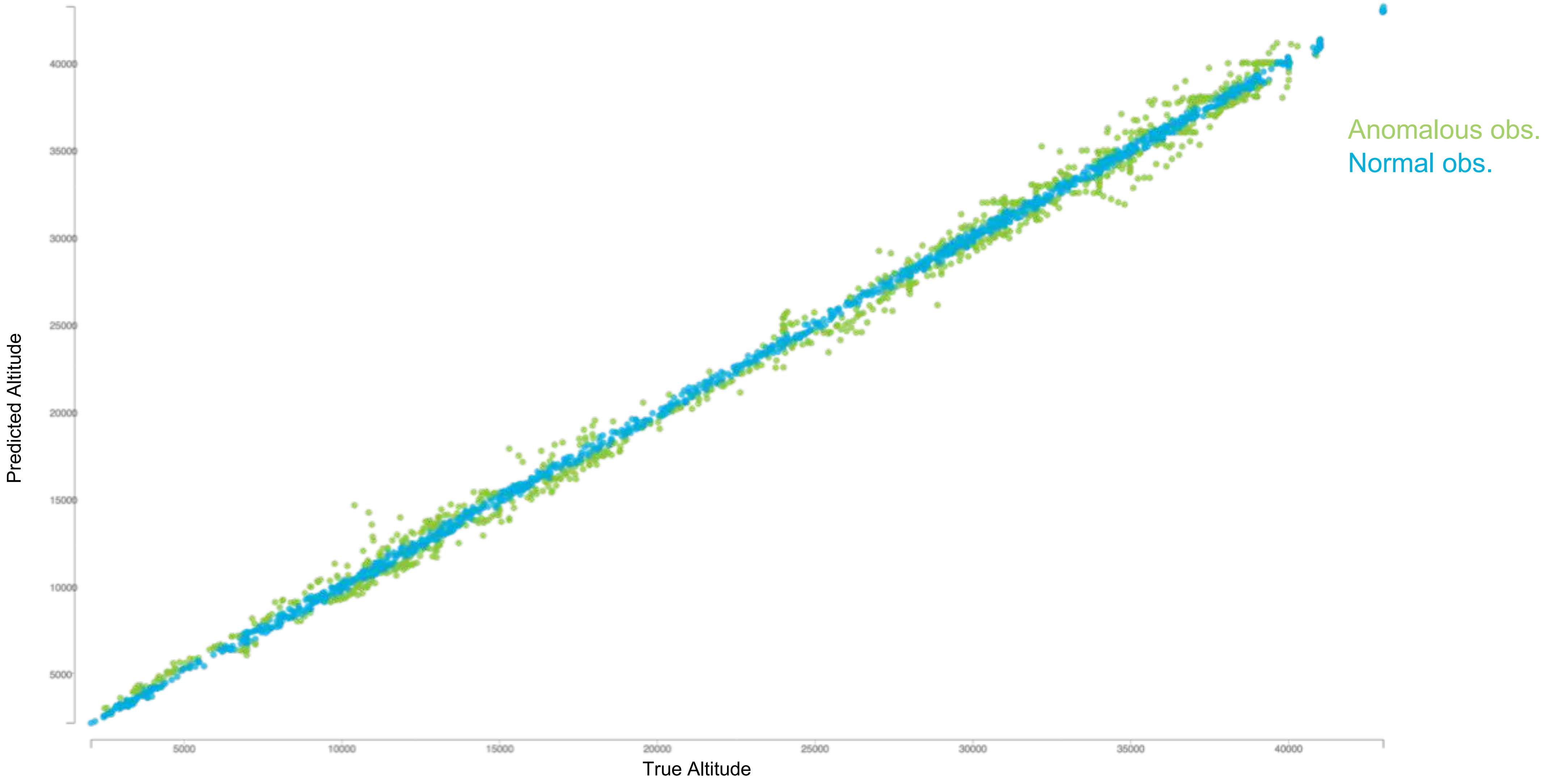
Mean Average Percentage Error = 0.7%

$$M = \frac{100}{n} \sum_{t=1}^{n} \left| \frac{A_t - F_t}{A_t} \right|$$

Prediction Errors



Anomalies


Predicted Altitude
40000
35000
30000
25000
20000
15000
10000
5000
5000
10000
15000
20000
25000
30000
35000
40000
True Altitude
Anomalous obs.
Normal obs.

Implemented Improvements

Missing Values

Removed rows with 10 or more missing historical data in the last 5 minutes

Train/Control Split

- Totaling 5 weeks of data
- First 4 weeks train
- Final week control

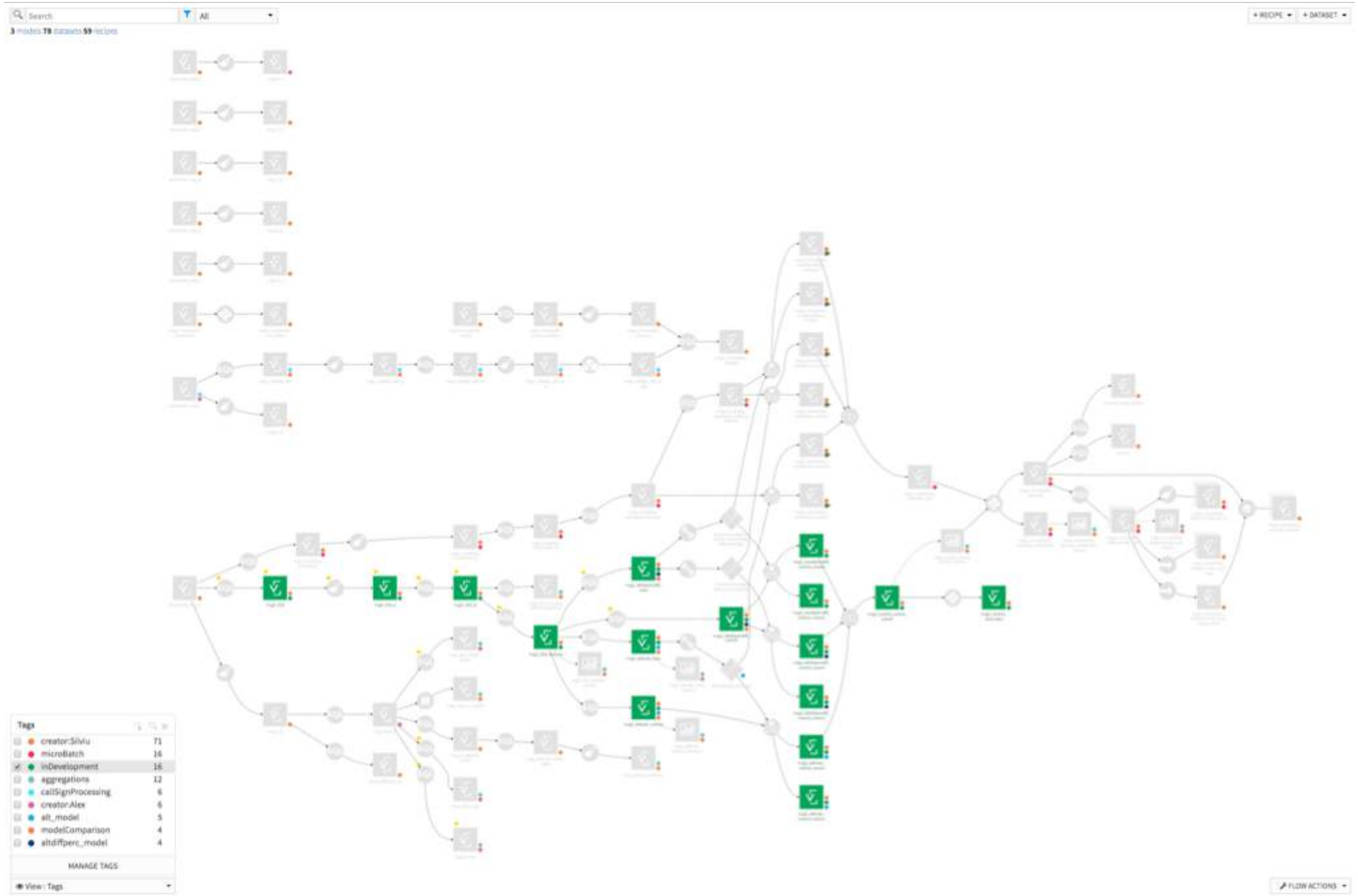
Multiple Feature/Prediction Models

- Raw altitude
- Percentage difference in altitude
- Combination of the two

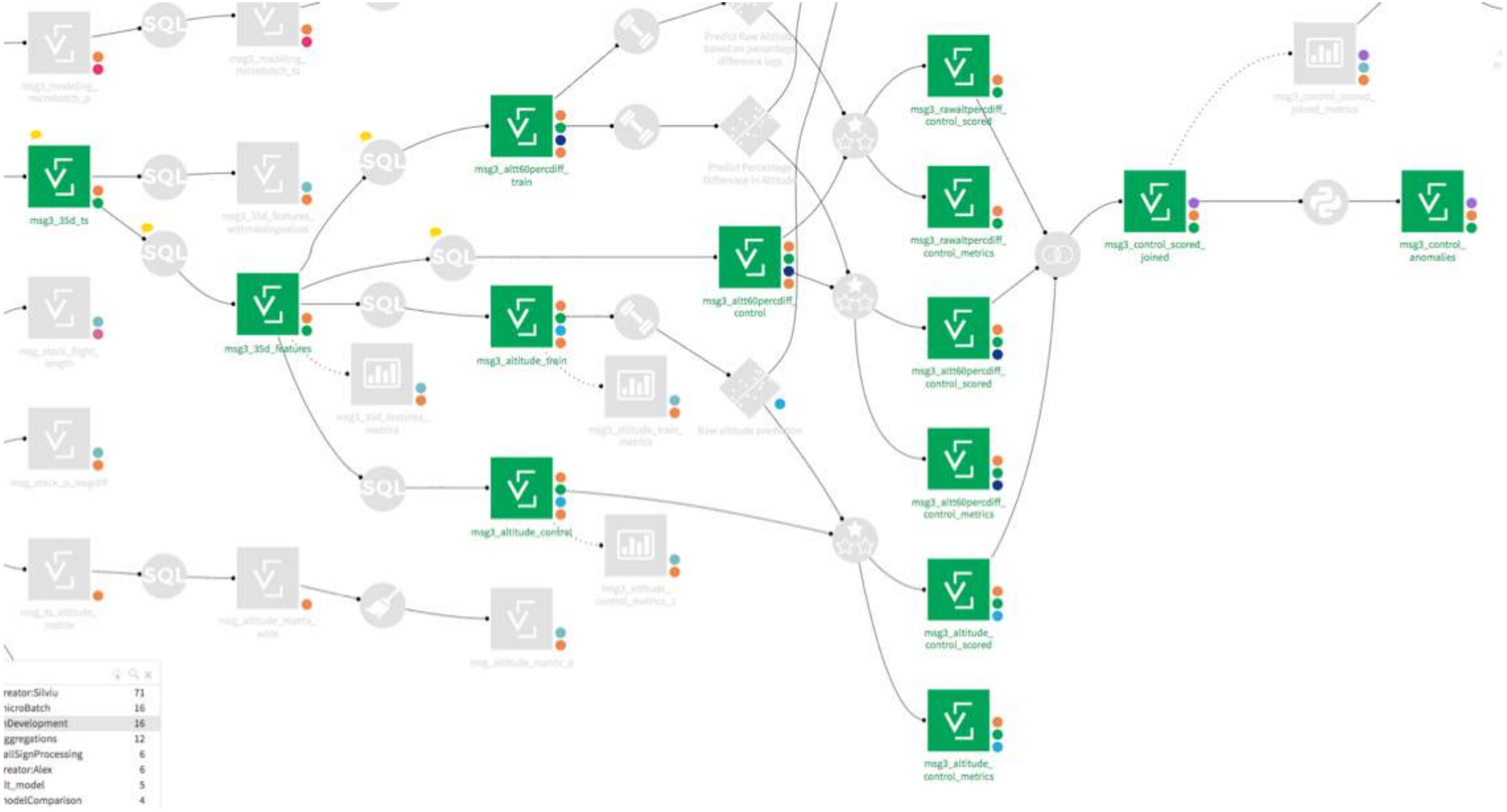
Overall Flow Cleanliness

Continuously reformatted the flow for maintainability

Final Modeling Flow



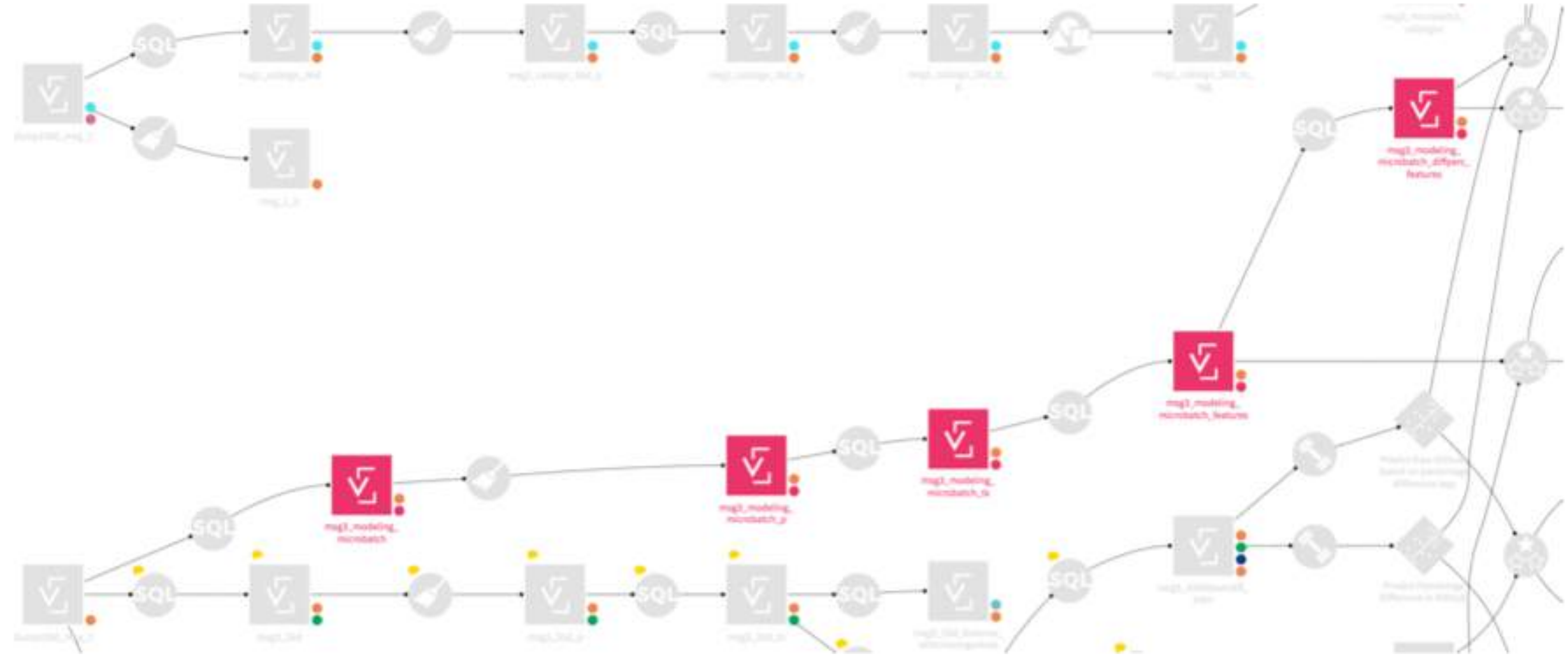
Final Modeling Flow



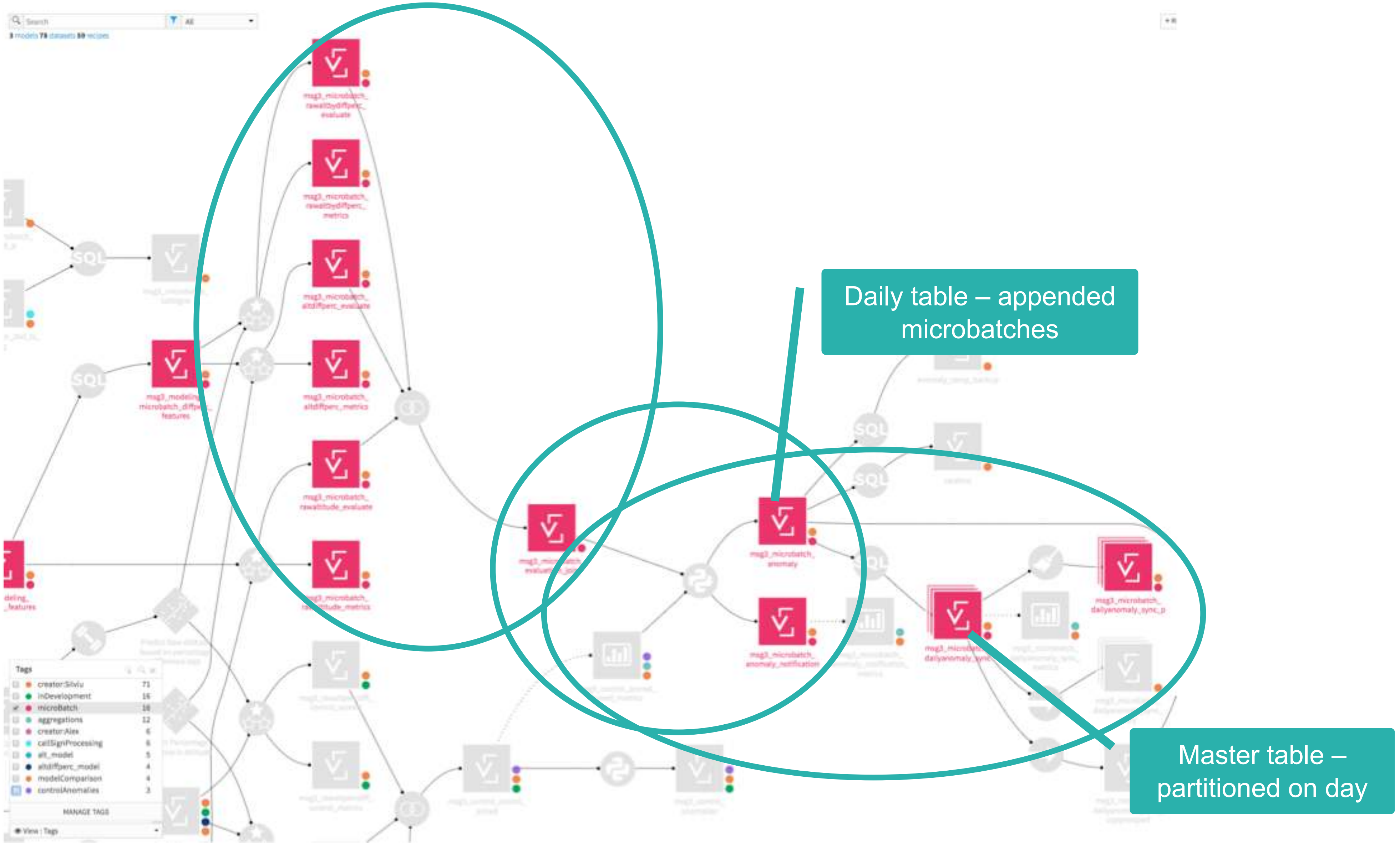
Deployment

or how this is using our partner's IT resources

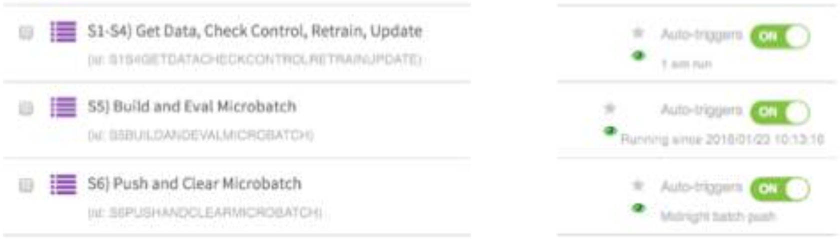
Microbatches

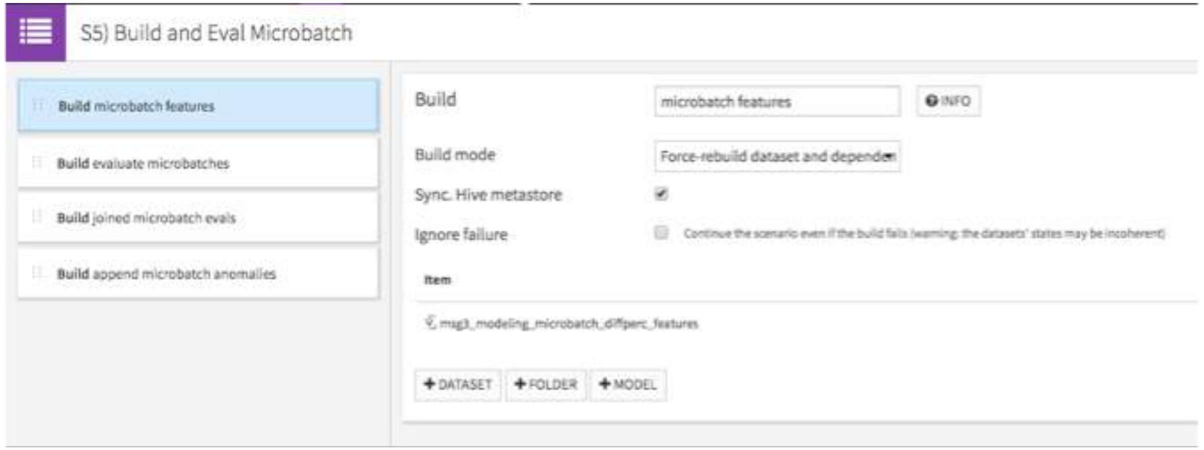


Microbatches



Scenario Management



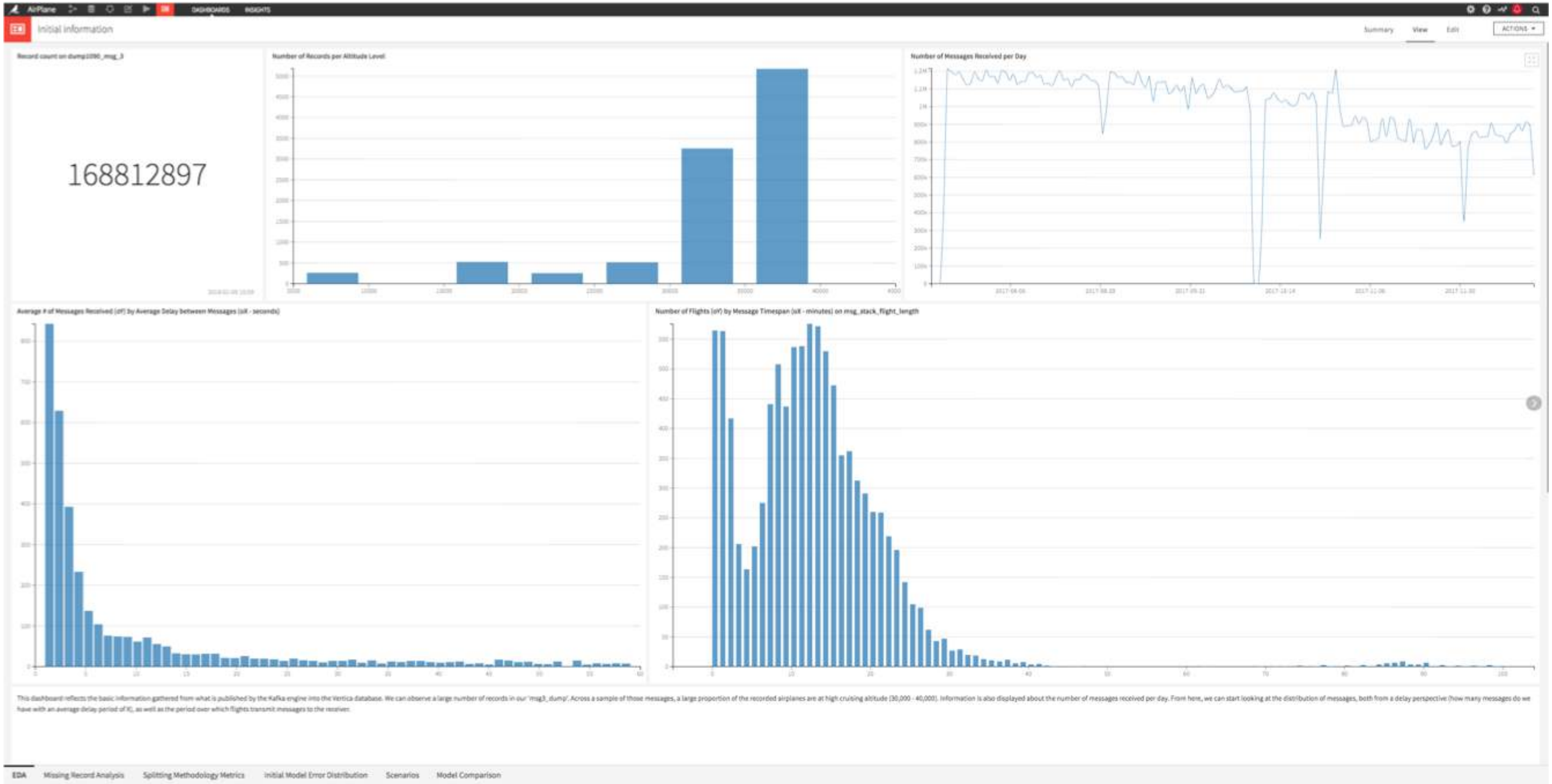



```python
from dataiku.scenario import Scenario
import dataiku
import pandas as pd
scenario = Scenario()

#Function that returns the latest 'metric' from a 'table'
def get_metric(table, metric):
    df = dataiku.Dataset(table).get_dataframe()
    df.sort_values(by='date', ascending=False, inplace=True)
    val = df[metric].iloc[0]
    return val


#Force build "msg3_35d_features" and all dependencies
scenario.build_dataset("msg3_35d_features", build_mode="RECURSIVE_FORCED_BUILD")

#Build equivalent but with missing values (for reporting purposes)
scenario.build_dataset("msg3_35d_features_withmissingvalues", build_mode="NON_RECURSIVE_FORCED_BUILD")

#Get current MAPE / RMSE for the previous control samples
rawalt_mape = get_metric("msg3_altitude_control_metrics", 'mape')
rawaltbydiffperc_mape = get_metric("msg3_rawaltpercdiff_control_metrics", 'mape')
altdiffperc_rmse = get_metric("msg3_altt60percdiff_control_metrics", 'rmse')

#Build control samples - only "msg3_altitude_control" and "msg3_altt60percdiff_control"
scenario.build_dataset("msg3_altitude_control", build_mode="NON_RECURSIVE_FORCED_BUILD")
scenario.build_dataset("msg3_altt60percdiff_control", build_mode="NON_RECURSIVE_FORCED_BUILD")

#Evaluate new control samples
scenario.build_dataset("msg3_altitude_control_scored", build_mode="NON_RECURSIVE_FORCED_BUILD")
scenario.build_dataset("msg3_rawaltpercdiff_control_scored", build_mode="NON_RECURSIVE_FORCED_BUILD")
scenario.build_dataset("msg3_altt60percdiff_control_scored", build_mode="NON_RECURSIVE_FORCED_BUILD")

#Get MAPE/RMSE for the newly created controls
rawalt_mape_new = get_metric("msg3_altitude_control_metrics", 'mape')
rawaltbydiffperc_mape_new = get_metric("msg3_rawaltpercdiff_control_metrics", 'mape')
altdiffperc_rmse_new = get_metric("msg3_altt60percdiff_control_metrics", 'rmse')

#Check control performance against current MAPE/RMSE (within 10%) as well as fixed values
#Do we retrain when the mape decreases too?

#Normal zeros now

if rawalt_mape_new >= 0.015 or rawalt_mape_new >= 1.1 * rawalt_mape:

    #Get metric in current active version
    for model in dataiku.Model('GaoJN868').list_versions():
        if model['active'] == True:
            active_mape = model['snippet']['mape']

    train_ret = scenario.train_model("GaoJN868")
    trained_model = train_ret.get_trained_model()
    performance = trained_model.get_new_version_metrics().get_performance_values()
    #performance = get_version_metrics().get_performance_values()
    if performance["MAPE"] < active_mape:
        trained_model.activate_new_version()

    #Rebuild scored/metrics
    scenario.build_dataset("msg3_altitude_control_scored", build_mode="NON_RECURSIVE_FORCED_BUILD")

```

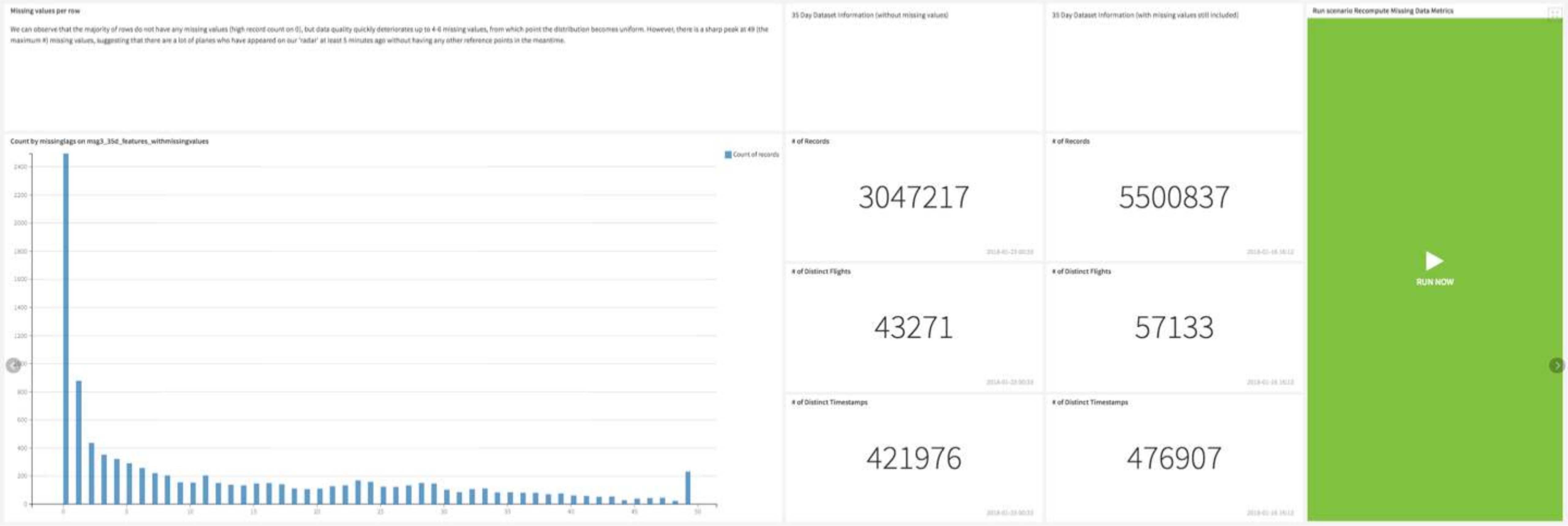
Monitoring

or how we remember we did all of this

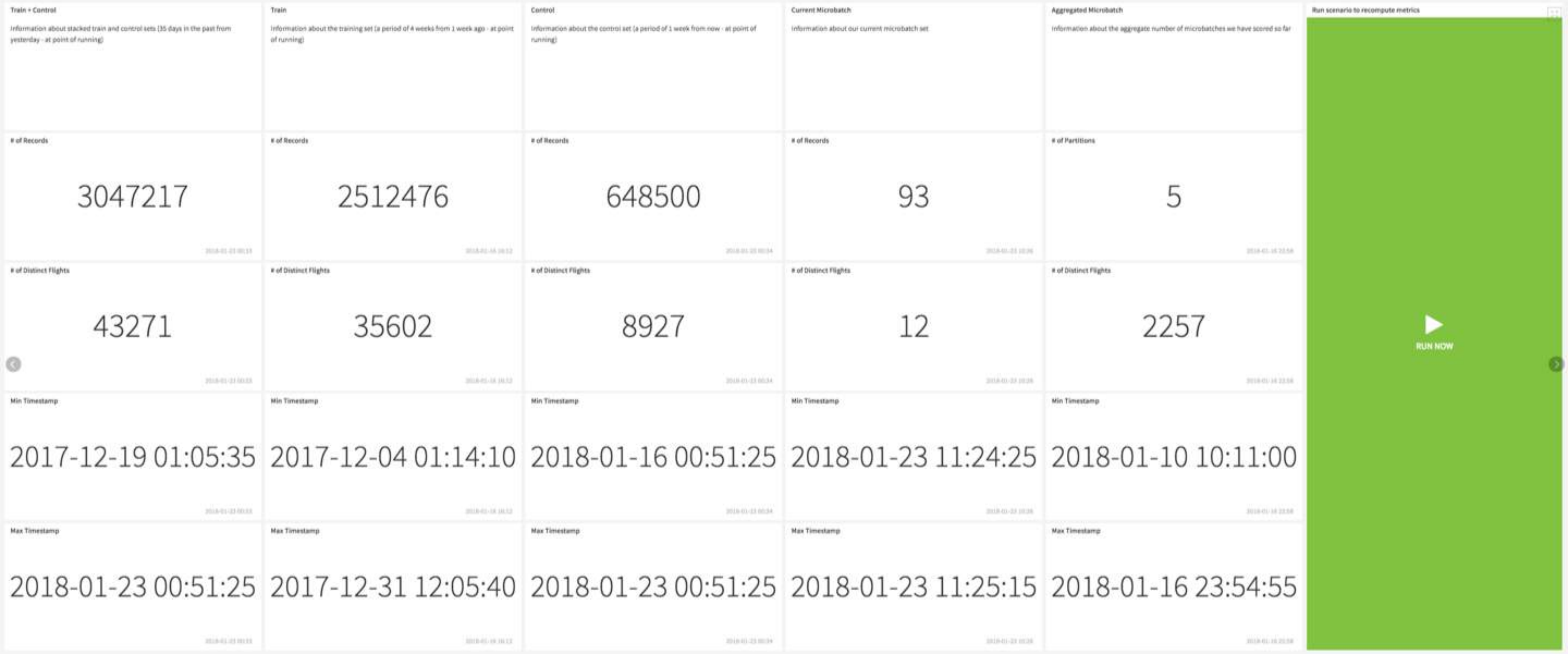
Exploratory Info



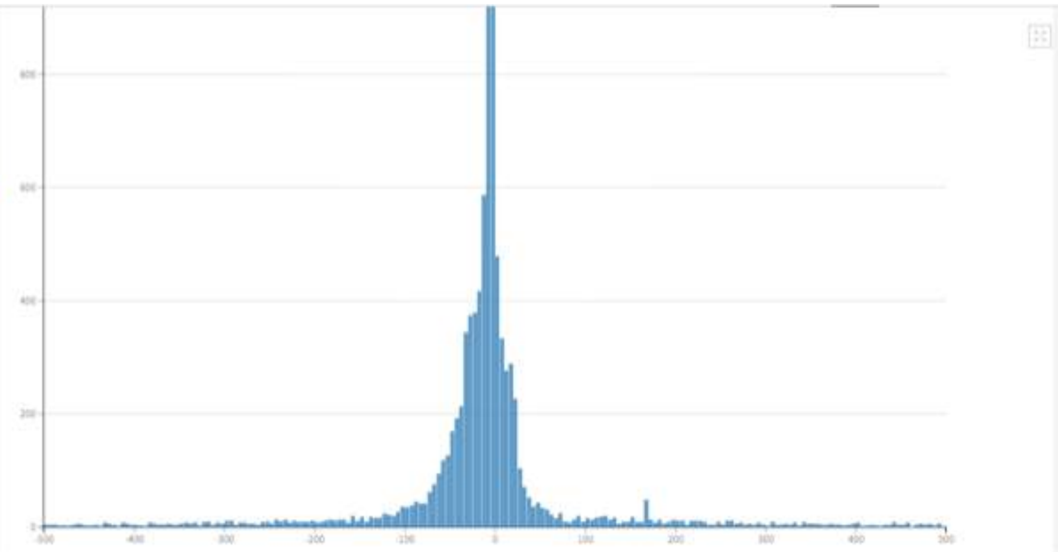
Missing Data Analysis



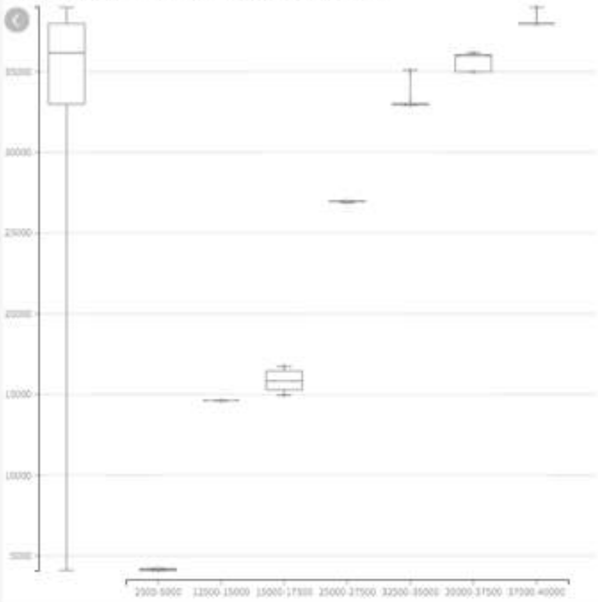
Splitting Methodology Review



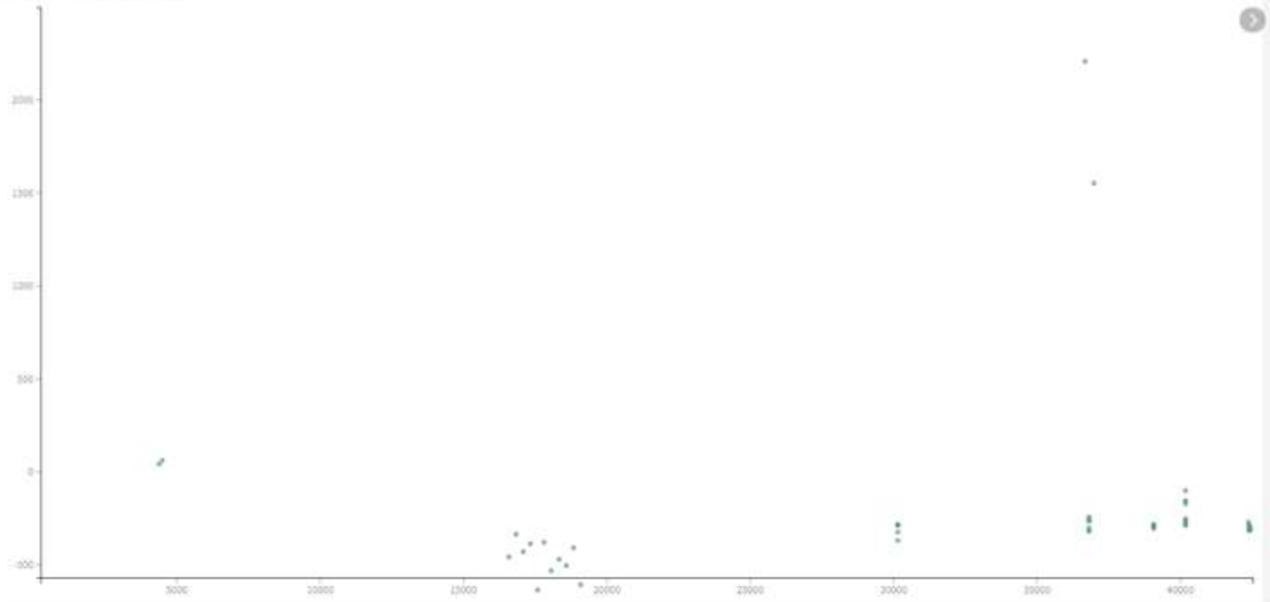
Model Performance



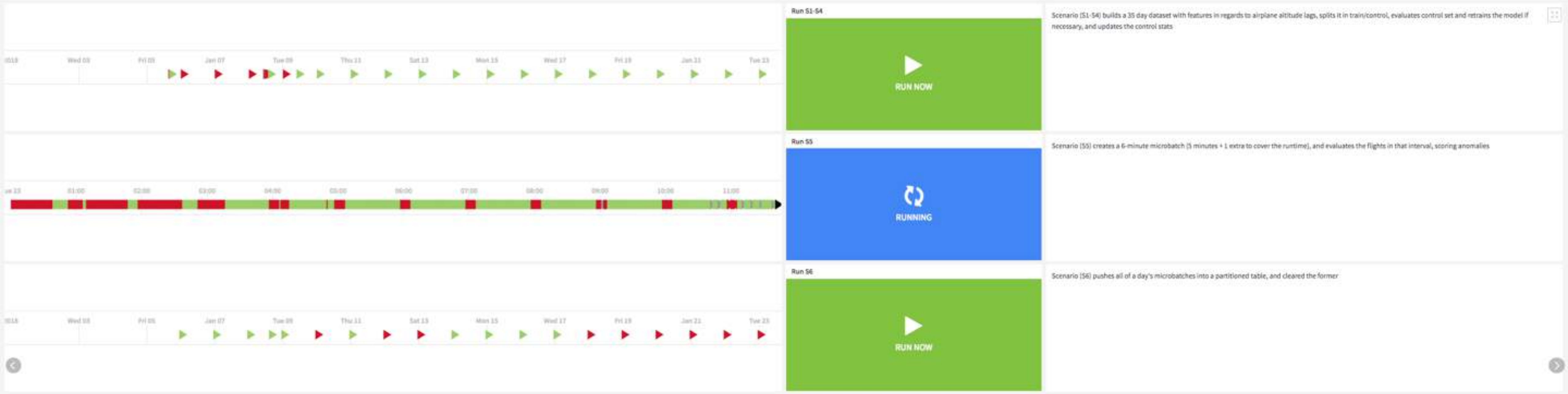
Prediction Distribution (oY) per Level of Altitude (oX) on Current Microbatch



Error (oY) on Altitude Prediction (oX)

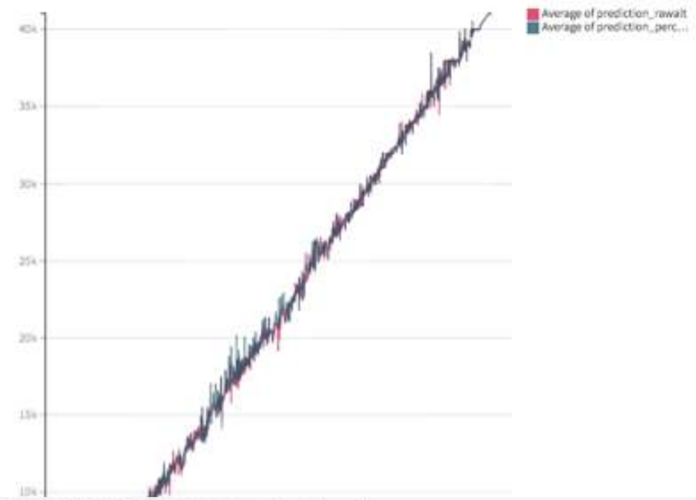


Scenario Tracking

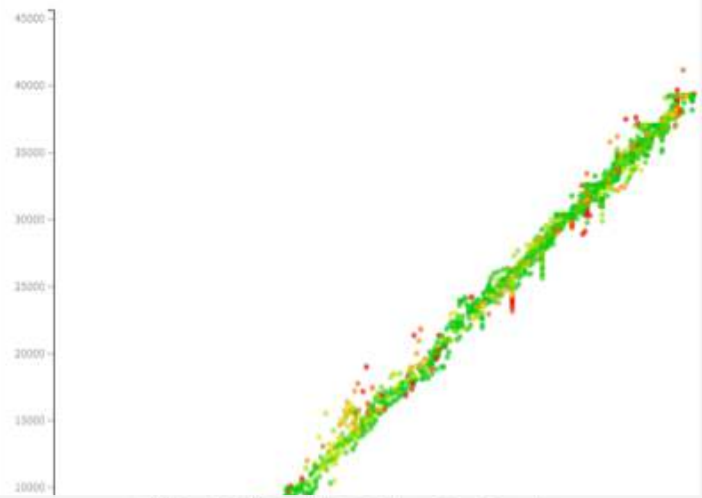


Model Comparison

Altitude Predictions (oY) vs Real Altitude (oX) - Control Sample



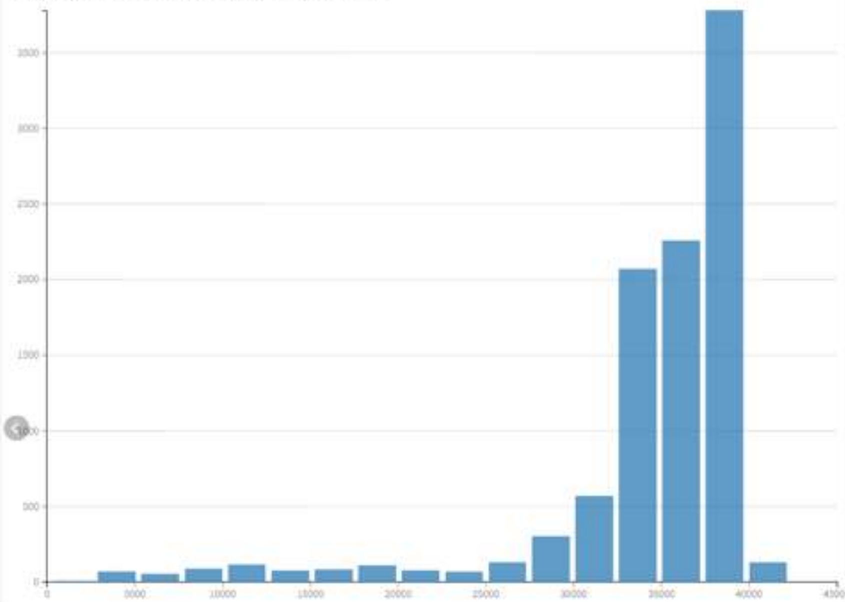
Real Altitude (oX) vs Altdiffperc Predicted Altitude (oY) - Missing (color) on msg3_control_anomalies



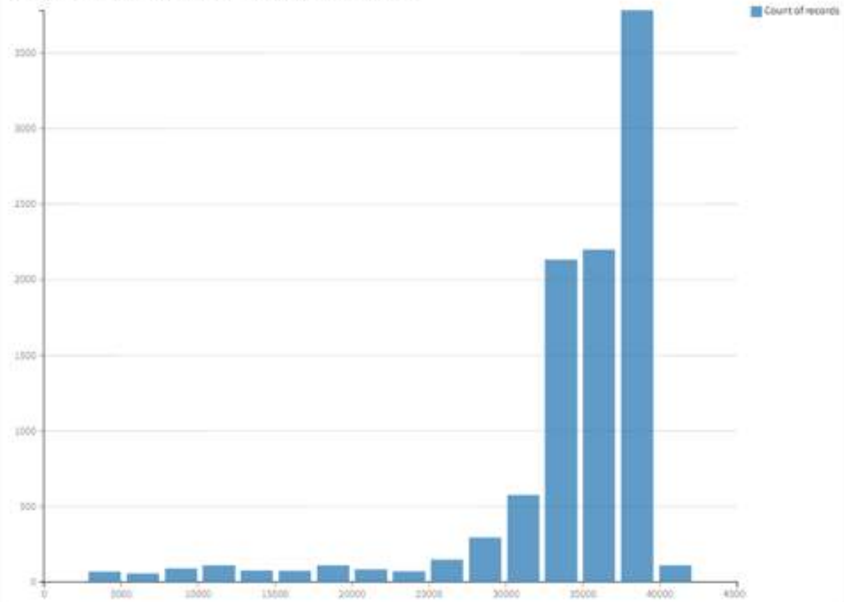
Prediction Count - Altdiffperc Pred(rows) vs Real Altitude (cols) on msg3_control_anomalies

	0-5000	5000-10000	10000-15000	15000-20000	20000-25000	25000-30000	30000-35000	35000-40000	40000-45000	45000-50000	
0-5000	74	0	0	0	0	0	0	0	0	0	74
5000-10000	5	133	9	0	0	0	0	0	0	0	147
10000-15000	0	7	174	6	0	0	0	0	0	0	187
15000-20000	0	0	3	177	5	0	0	0	0	0	185
20000-25000	0	0	0	10	140	5	0	0	0	0	155
25000-30000	0	0	0	0	12	395	35	0	0	0	442
30000-35000	0	0	0	0	0	8	1993	711	0	0	2712
35000-40000	0	0	0	0	0	0	42	4924	1021	0	5987
40000-45000	0	0	0	0	0	0	0	5	106	0	111
45000-50000	0	0	0	0	0	0	0	0	0	0	0
	79	140	186	193	157	408	2070	5640	1127	0	

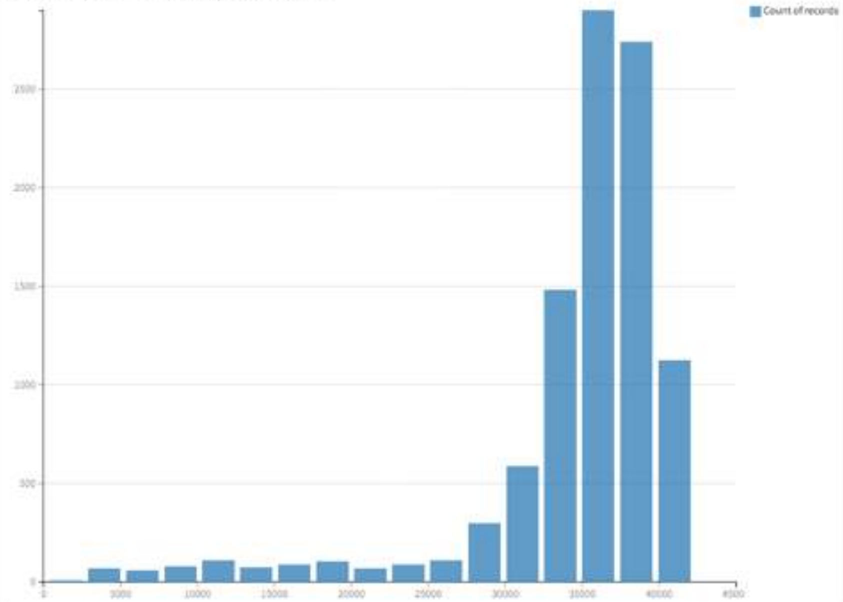
Raw Predicted Altitudes Distributions (in feet) on msg3_control_anomalies



Altdiffperc Predicted Altitudes Distribution (in feet) on msg3_control_anomalies



Real Altitude Distributions (in feet) on msg3_control_anomalies



Anomaly Count - Altdiffperc Error(rows) vs Alt Error (cols) on msg3_control_anomalies

	0	1	
0	9123	487	9610
1	142	248	390
	9265	735	

Anomalies (on altdiffperc_error) versus # of Missing Values on msg3_control_anomalies

	0	1	2	3	4	5	6	7	8	9	
0	4743	1480	799	528	484	402	386	322	239	227	9610
1	153	29	30	22	26	21	40	21	17	31	390
	4896	1509	829	550	510	423	426	343	256	258	

Next Steps

or if we have anything else to do

(I couldn't find a funny gif)

We can do LOADS more

Great story so far, but how about:

- latitude
- longitude
- speed
- verticality

Similar train of thought, different approach

- regression target becomes an array
- deep learning involved (5 features x 60 timestamps x N rows)
- add anomaly definition on top
- make it run in near real-time

Key Takeaways

- Don’t always adhere to the standard
- Start simple
- Keep stakeholders involved
- Think (a lot) and don’t rush