

Project Process

**Automated Scales for
real-time weight capture**

Zhiyi Ding
Aug 8, 2017

Requirement

Input parameters:

- Sensor data pulled from api
- A latest batch date
- Standard guide data

Output:

- Average weight of each data point in the last day/hours

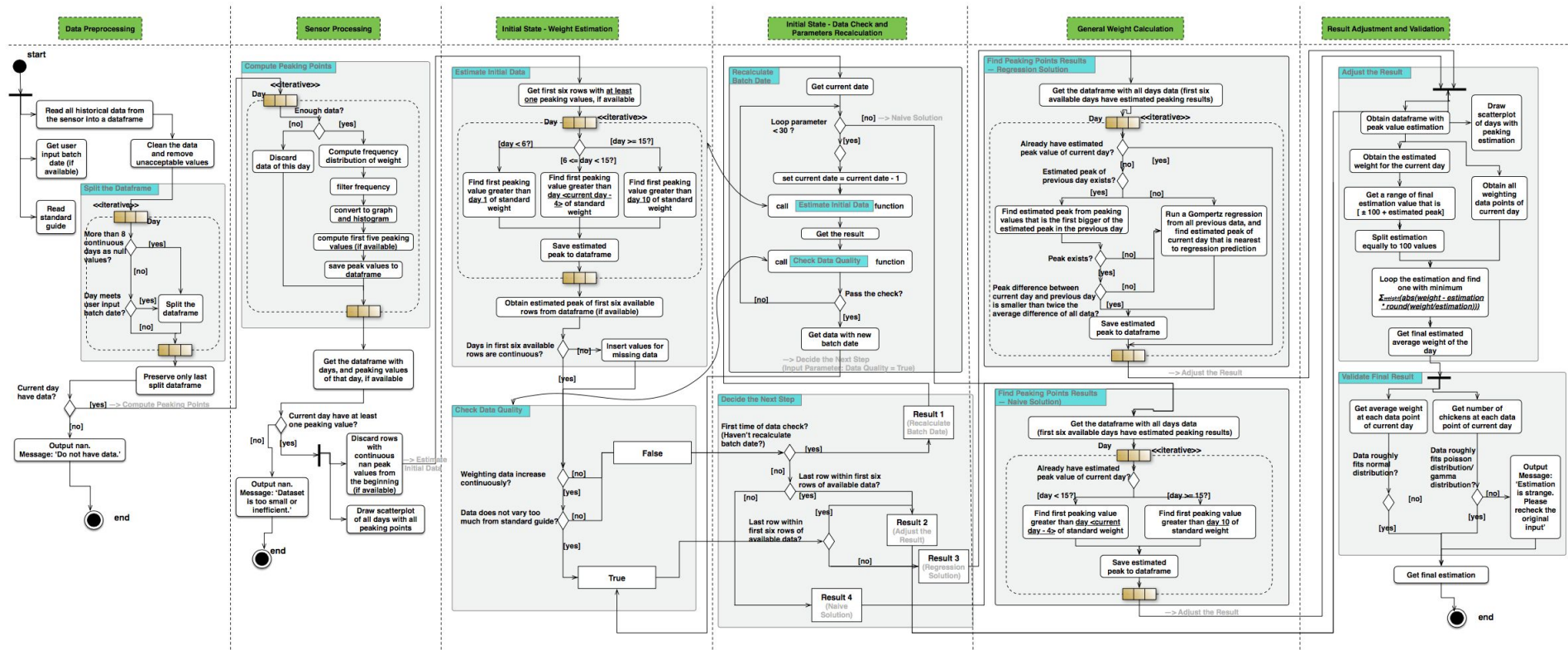


Premex's Farm

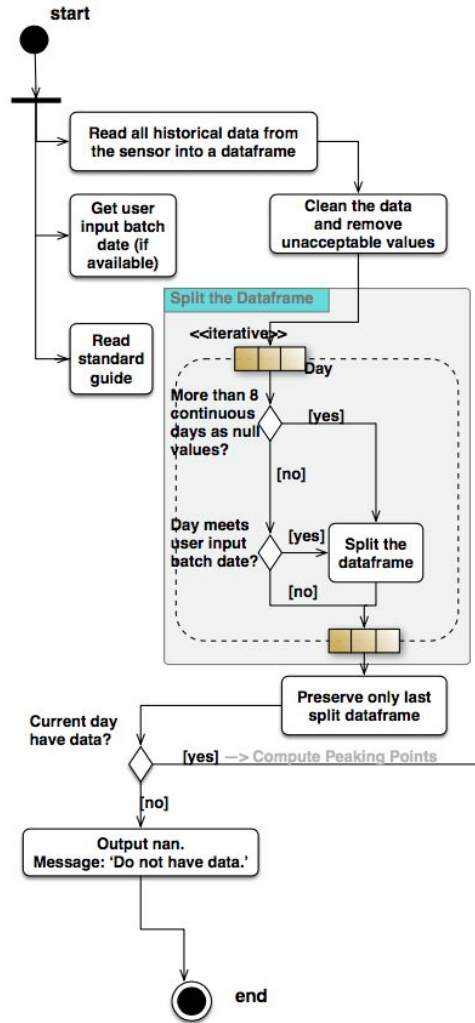
Implementation

1. Data preprocessing
2. Sensor processing
3. Weight estimation in initial state
4. Data check/Parameter recalculation
5. General weight calculation
6. Result adjustment and validation

Design



UML, Activity Diagram

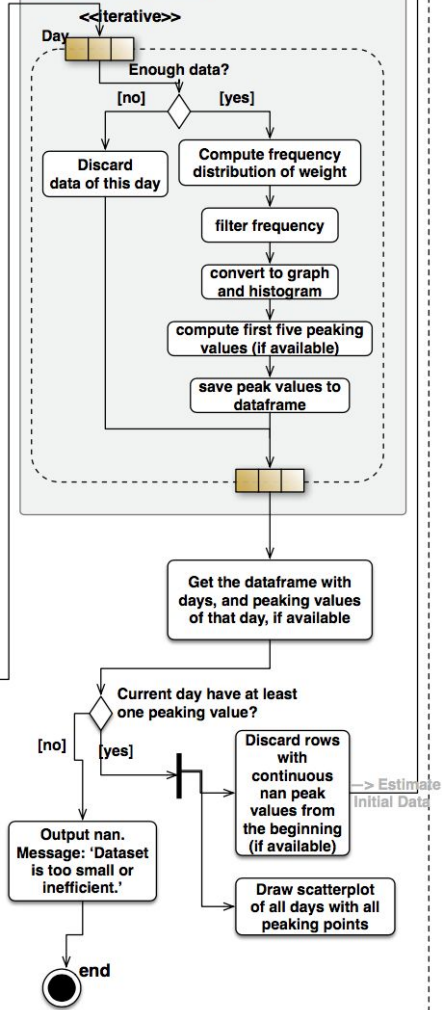


1. Data Preprocessing

- 1) Read in all parameters
- 2) Split the dataframe
- 3) Preserve latest dataframe

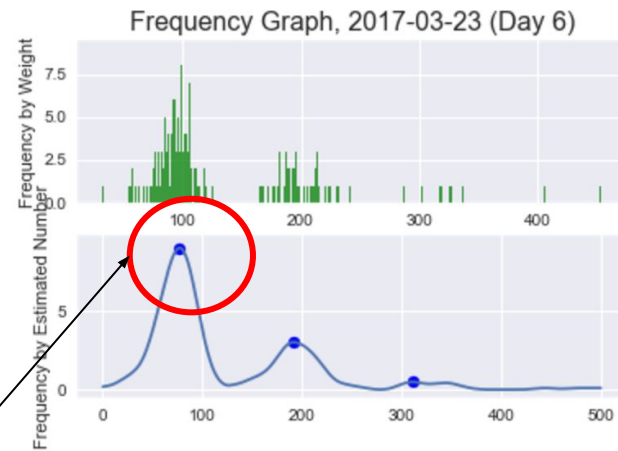
Function: `df_split()`:
- split the dataframe

Compute Peaking Points



2. Sensor Processing

- 1) Process data day by day
- 2) For each day, compute frequency distribution and draw histogram
- 3) Find all peaking values

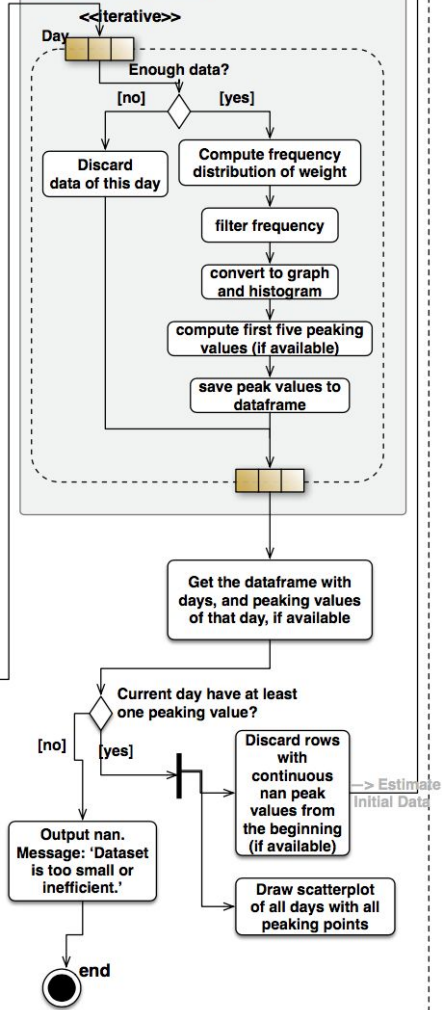


Biggest Problem 1:
How to find this peaking point???

Function: `find_peaking_point()`:
- For each day, find peaking points from histogram



Compute Peaking Points

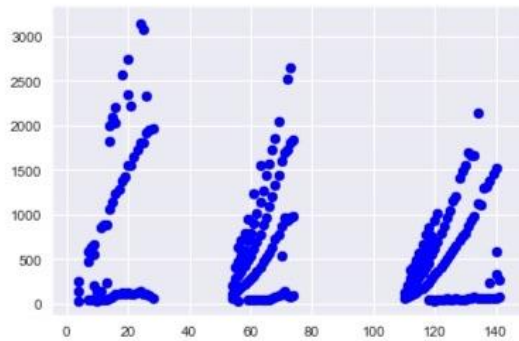


2. Sensor Processing (cont.)

4) Get a scatterplot of all days' values

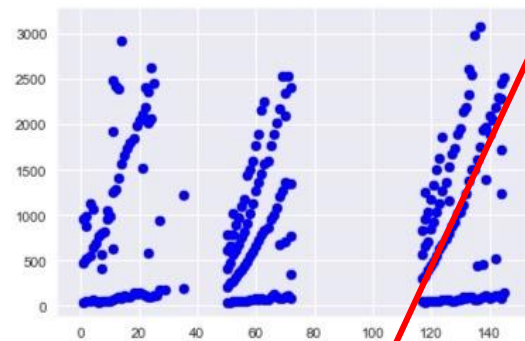
Biggest Problem 2:
How to find the points in this line???

Begin processing Sensor1.
SUCCESS 584302380c51e0353c9d8e9c
Finished pulling data from api.



Finished peak finding and data prediction.

Begin processing Sensor2.
SUCCESS 584493520c51e008d3e88909
Finished pulling data from api.



Finished peak finding and data prediction.

How to find the peaking point???

Discarded Approach 1:

Using standard data as compassion.

Discarded because:

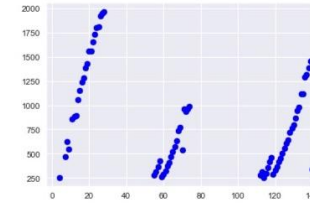
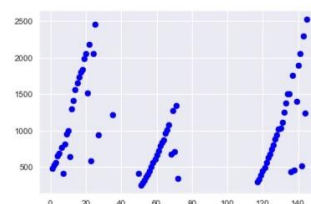
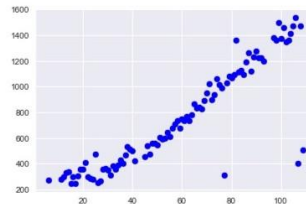
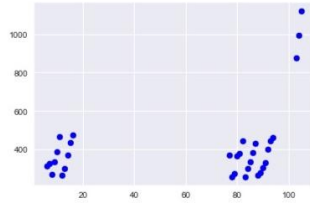
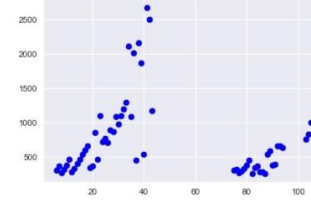
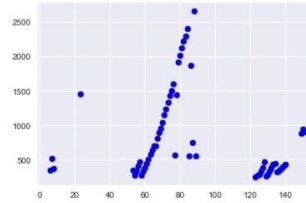
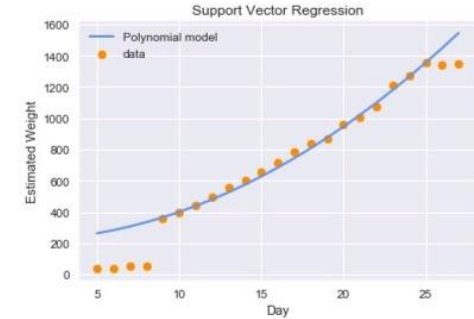
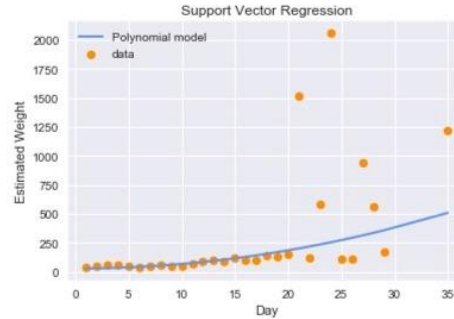
- 1) Logic itself is dubious
- 2) Batch date may be unknown or inaccurate

Discarded Approach 2:

Do regression backwards.

Discarded because:

The system is automatic



How to find the peaking point???

Using a simple one-point method.

If date < 10:

- Discard weighting date less than standard (day 1)
- Choose the smallest weight from remaining peaking point

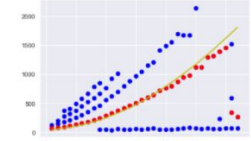
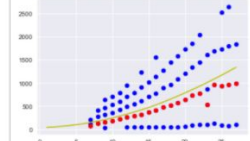
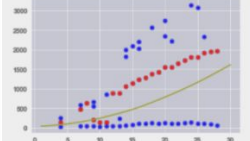
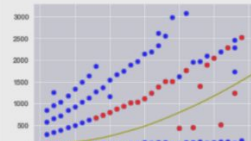
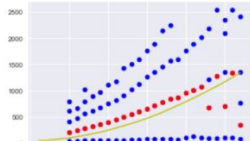
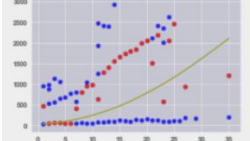
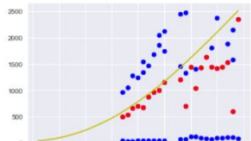
Elif date > 10:

- Discard weighting date less than standard (day 10)
- Choose the smallest weight from remaining peaking point

Discarded because:

Estimation is not accurate.

Discarded Approach 3:

Name			
'Sensor 1'	<p>sensor name: Sensor1 predicted batch date: 2017-03-22 <matplotlib.collections.PathCollection at 0x186e769d></p> 	<p>sensor name: Sensor1 predicted batch date: 2017-01-19 <matplotlib.collections.PathCollection at 0x189faa1d></p>  <p>Given Batch Date</p>	<p>sensor name: Sensor1 predicted batch date: 2016-12-03 <matplotlib.collections.PathCollection at 0x12163c7dd></p> 
'Sensor 2'	<p>sensor name: Sensor2 predicted batch date: 2017-03-31 <matplotlib.collections.PathCollection at 0x1218a811d></p> 	<p>sensor name: Sensor2 predicted batch date: 2017-01-19 <matplotlib.collections.PathCollection at 0x1ee339d0></p>  <p>Given Batch Date</p>	<p>sensor name: Sensor2 predicted batch date: 2016-12-05 <matplotlib.collections.PathCollection at 0x1c98d51d></p> 
'Sensor 3'	<p>sensor name: Sensor3 predicted batch date: 2016-12-13 <matplotlib.collections.PathCollection at 0x1d4b811d></p>  <p>Given Batch Date</p>		

How to find the peaking point???

Discarded Approach 4:

Cross validate several methods.

Method 1: One-point Method (Without assuming batch date)

Method 2: Standard Comparison Method (Have to assume batch date)

Method 3: MSE Method (Without assuming batch date)

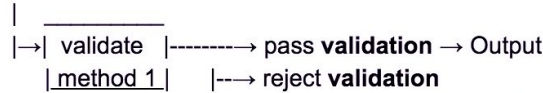
Validation: if the output is compellent.

Regression: Regression Method (for day > 4)

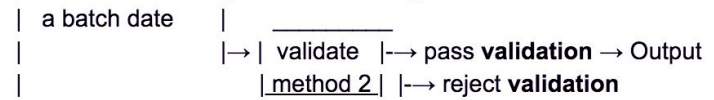
Discarded because:

Sometimes cannot find right estimation anyway using all methods.

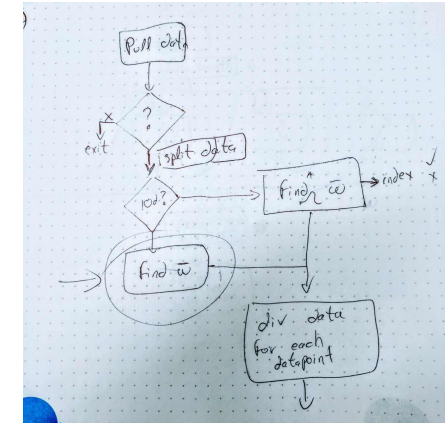
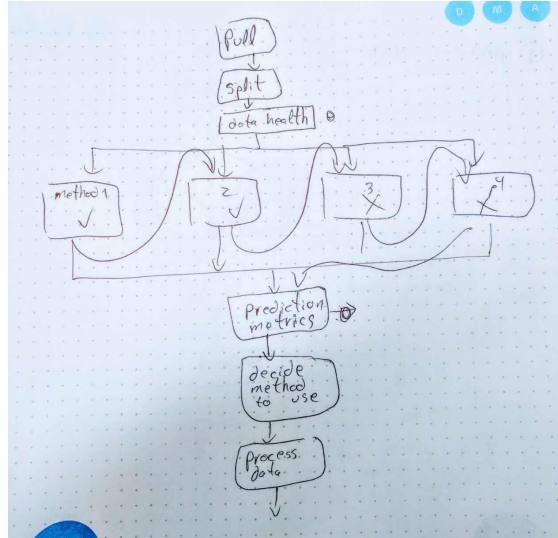
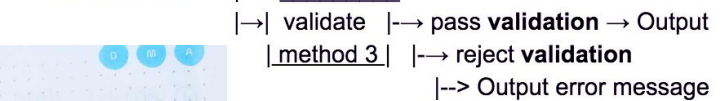
Try **method 1 + Regression**



If there's a batch date → Try **method 2 + Regression**



If there's not a batch date → Try **method 3 + Regression**



Estimate Initial Data

Get first six rows with at least one peaking values, if available

Day  <<iterative>>

[day < 6?]

[day >= 15?]

[6 <= day < 15?]

Find first peaking value greater than day 1 of standard weight

Find first peaking value greater than day <current day - 4> of standard weight

Find first peaking value greater than day 10 of standard weight

Save estimated peak to dataframe



Obtain estimated peak of first six available rows from dataframe (if available)

Days in first six available rows are continuous?

[no]

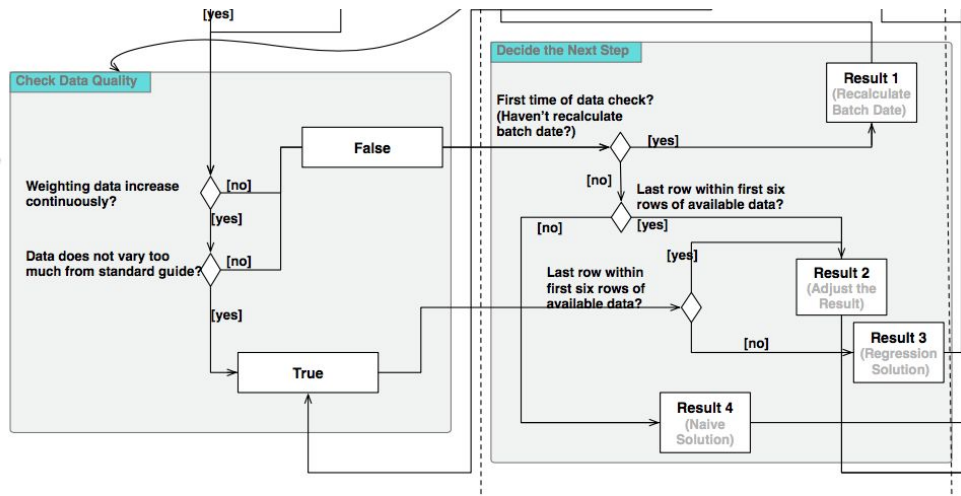
Insert values for missing data

[yes]

3. Weight Estimation in Initial State

Get first six days of estimation, if these data exists.

Function: `estimate_initial()`:
- Estimate the peaking point of first six days of data, if it exists.



4. Data Check/ Parameters Recalculation

1) Check the quality of first six days of data estimation

2) Get four result choices:

- Result 1: Recalculate batch date → call `recalculate_batch_date()`

- Result 2: Adjust the result → call `adjust_result()`

- Result 3: Regression solution → call `find_peaking_point_regression()`

- Result 4: Naive solution → call `find_peaking_point_naive()`

Function:

`check_data_quality():`

- Check if first six days of data estimation is acceptable

Function:

`decide_next_step():`

- Decide which route the program should go on

Recalculate Batch Date

Get current date

Loop parameter
< 30 ?

[no] → Naive Solution

[yes]

set current date = current date - 1

call **Estimate Initial Data** function

Get the result

call **Check Data Quality** function

Pass the check?

[no]

[yes]

Get data with new
batch date

→ Decide the Next Step
(Input Parameter: Data Quality = True)

4. Data Check/ Parameters Recalculation (cont.)

Function:

```
recalculate_batch_date():  
- Recalculate batch date
```

5. General Weight Calculation (1)

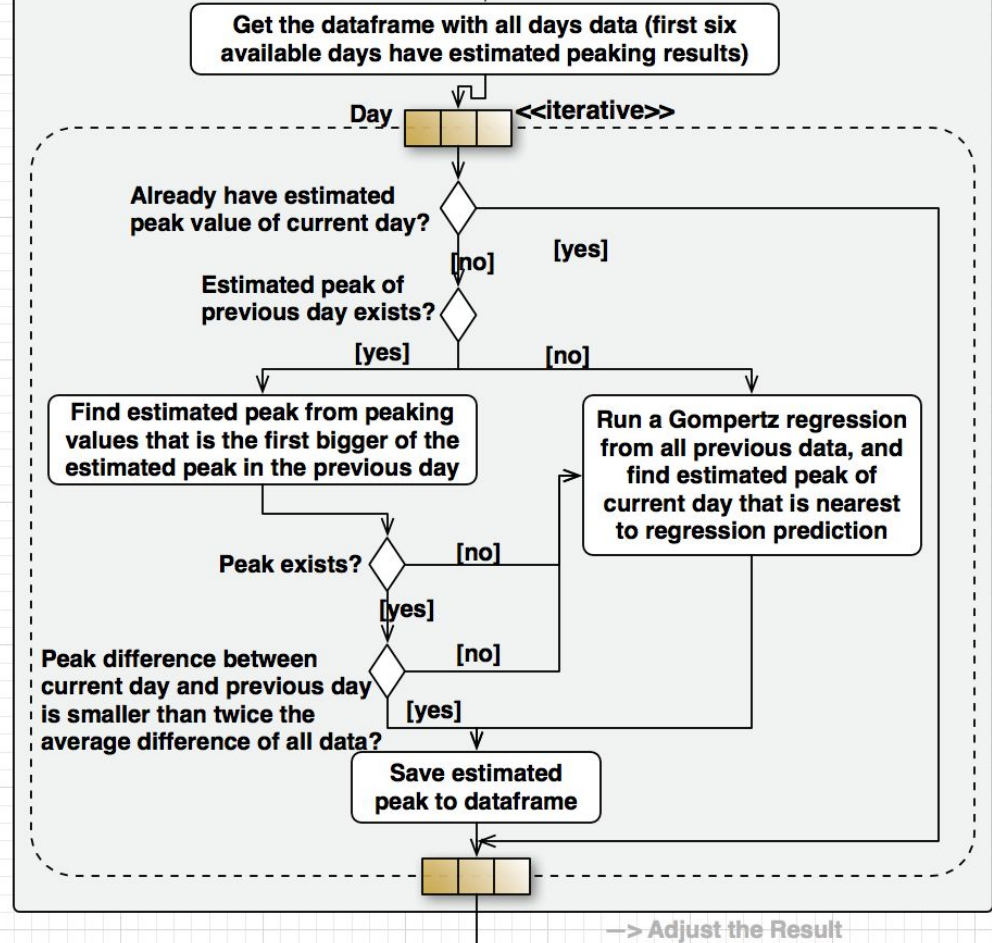
Using regression approach

Function:

`find_peaking_point_regression`
`on ()`:

- Find peaking points using a regression approach

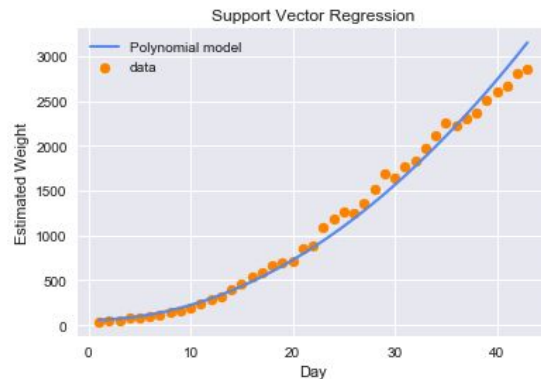
Find Peaking Points Results — Regression Solution



Which Regression??

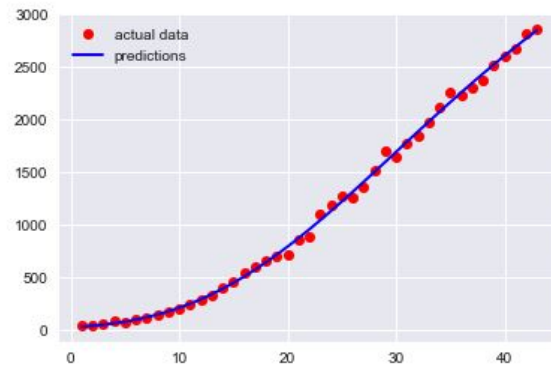
Approach 1: SVR Regression

Pros: Accurate
Cons: Very slow



Approach 2: Gomperts Regression

Pros: Fast; accurate in most times
Cons: Overfitting problem; Can not call the function for many times



Approach 3: Self-created Simple Regression

Pros: Easy and quick
Cons: Not so accurate

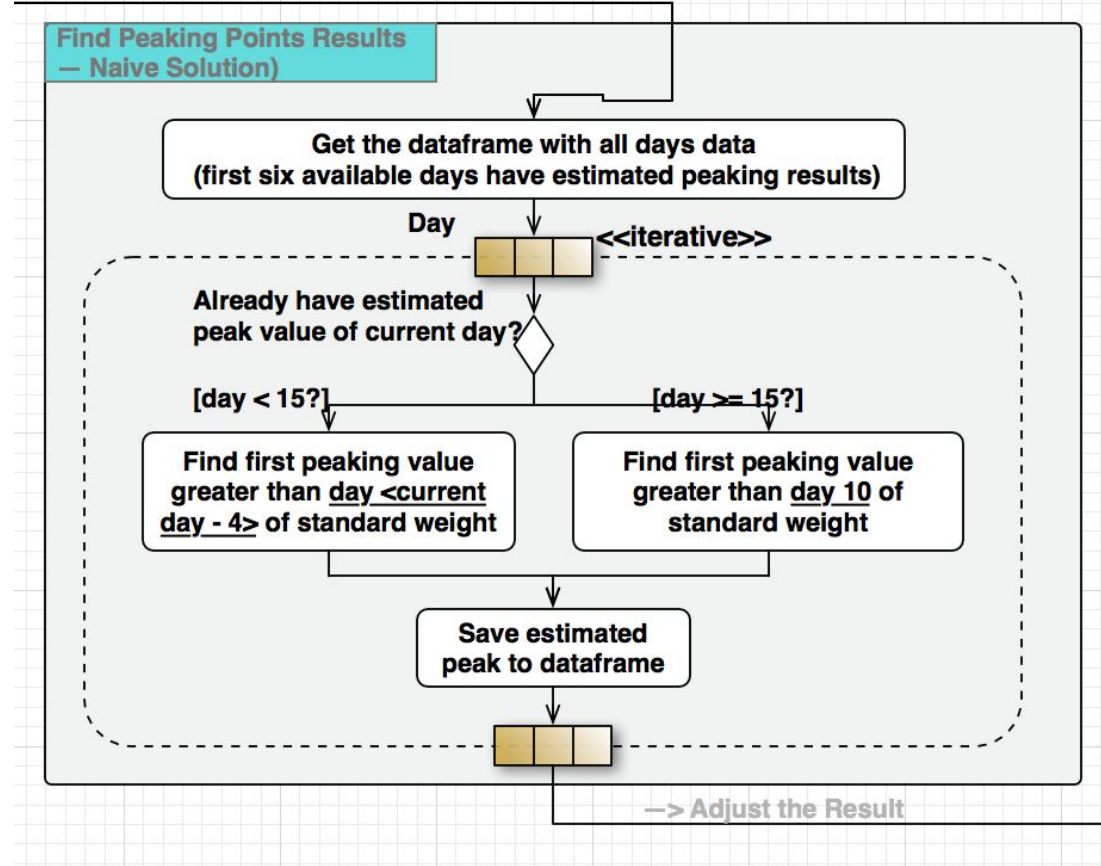
5. General Weight Calculation (2)

Using a naive approach

Function:

```
find_peaking_point_naive():
```

- Find peaking points using a naive approach



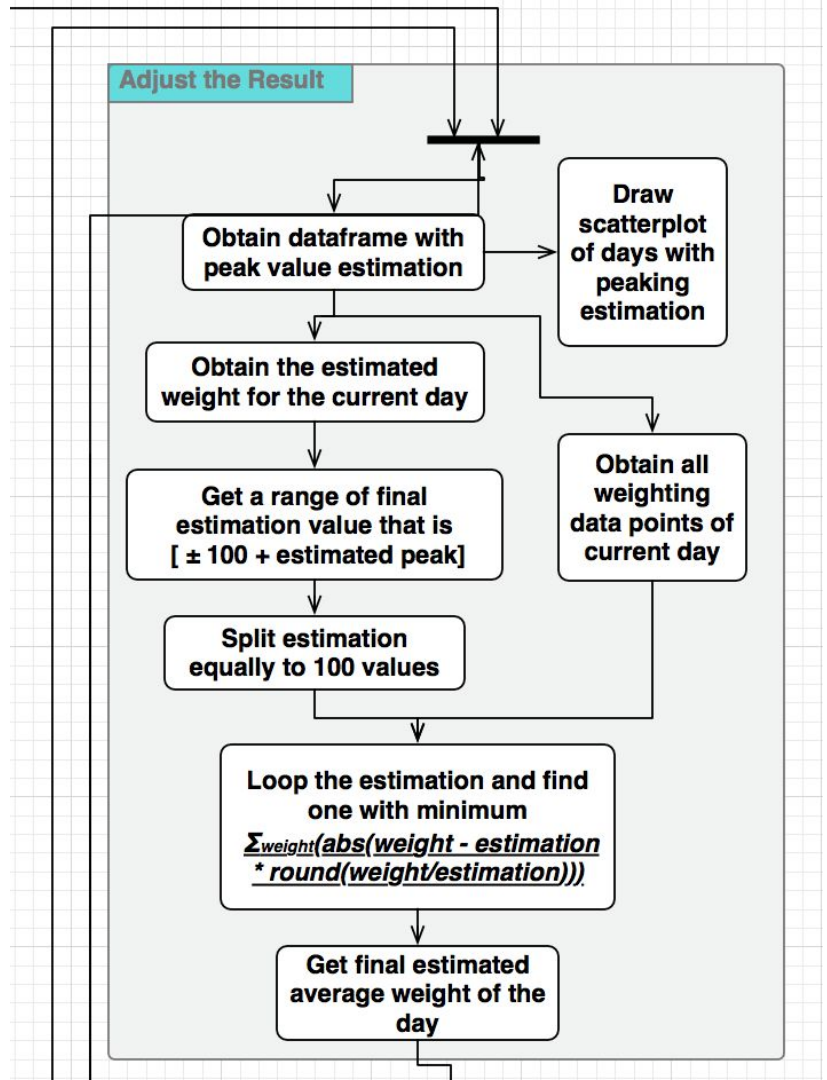
6. Result Adjustment and Validation

Adjust the result using an iterative approach.

Function:

`adjust_result():`

- adjust the result

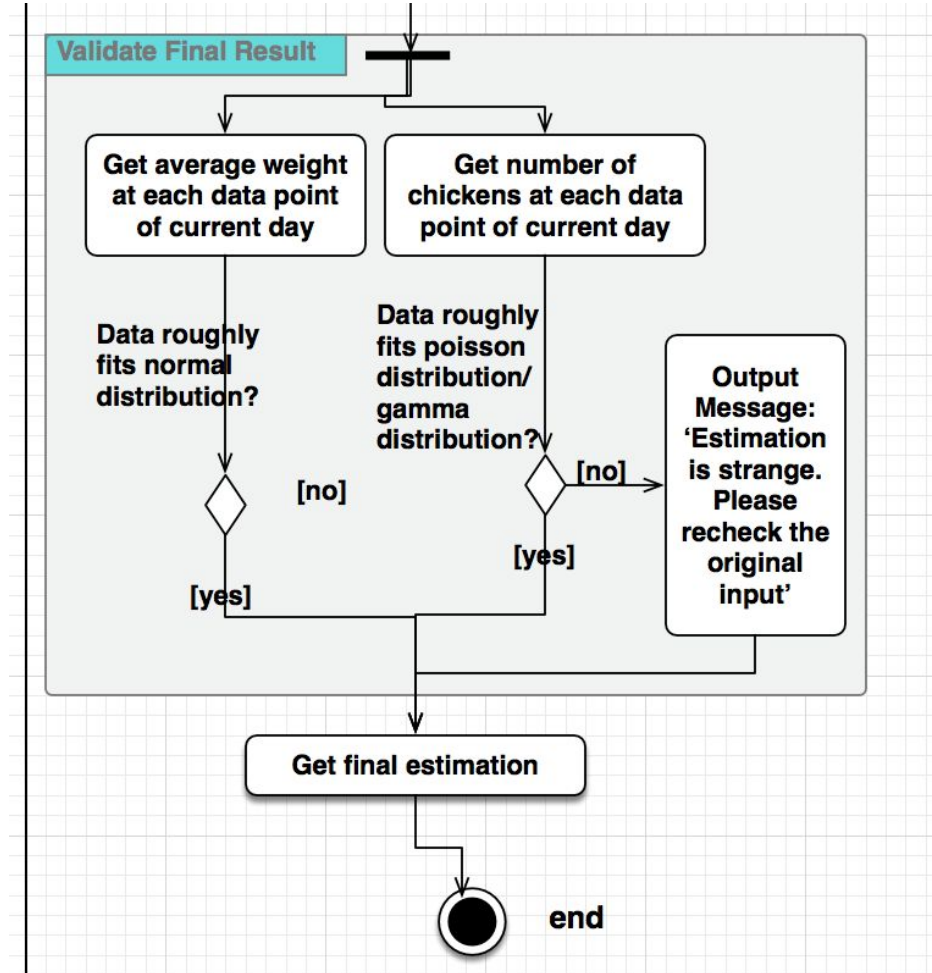


6. Result Adjustment and Validation

Validate final result.

Function:

```
validate_result ():  
- Validate final result
```



TBD

No	Priority	Features	Difficulty	Estimated Time
1	***	Recalculate batch date	***	4- 5 hours
2	***	Find peaking points results (data passed check)	**	2 - 3 hours
3	***	Check data Quality	***	2 - 3 hours
4	**	Insert missing data	**	0.5 - 1 hour
5	**	Find peaking points results (data not passed check)	**	1 - 2 hours
6	**	Adjust the result	***	3 - 4 hours
7	**	Validate final result	***	3 - 4 hours
8	*	Split the dataframe	**	2 - 3 hours

