FLAIRBIT
DATA CENTRIC SOLUTIONS

DataScienceSeed

Coffee Machine Predictive Maintenance Challenge

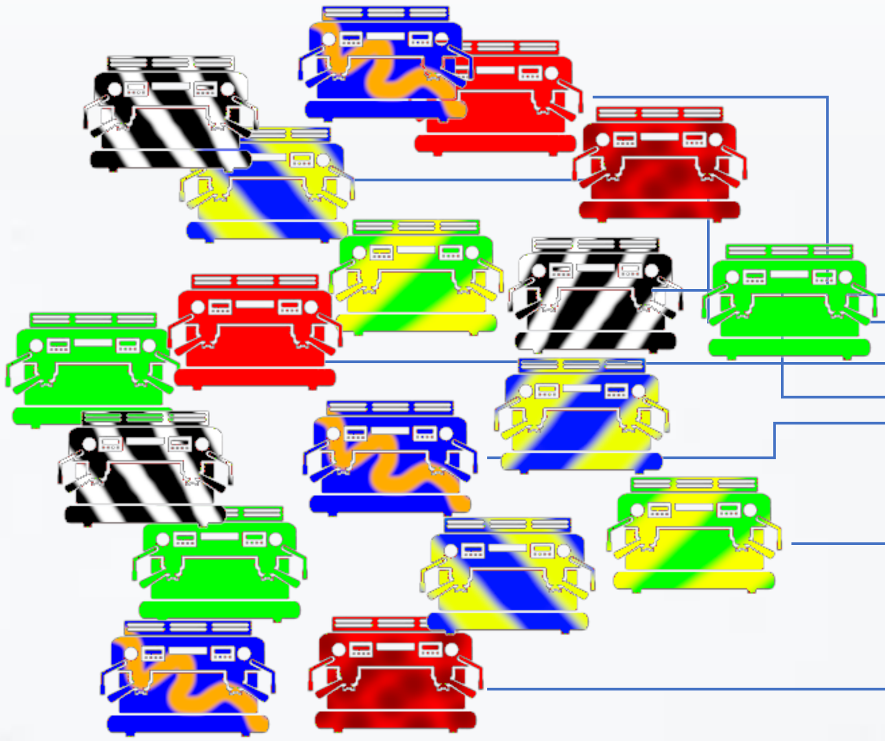
Marcello Morchio – Andrea Boero

May 23rd 2019 – www.datascienceseed.com

FLAIRBIT DATA CENTRIC SOLUTIONS

DataScienceSeed

The Challenge



counters

Serial	YYYY	MM	dd	hh:mm:ss	Week	Model	LabelCounter	AbsoluteCounter	RelativeCounter
1535632	2016	11	29	04:00:07	49	model 31	numcaffegenerale	179112	0
1535632	2016	11	29	05:00:07	49	model 31	numcaffegenerale	179120	8
1535632	2016	11	29	06:00:07	49	model 31	numcaffegenerale	179158	38

cleanings

Serial	YYYY	MM	dd	hh:mm:ss	Week	Model	Errorcode
1535632	2016	11	29	04:00:07	49	model 10	1
1535632	2016	11	29	05:00:07	49	model 9	1
1535632	2016	11	29	06:00:07	49	model 31	1

faults

Serial	YYYY	MM	dd	hh:mm:ss	Week	Model	Errorcode	Critical
1535632	2016	11	29	04:00:07	49	model 62	185	WARNING
1535632	2016	11	29	05:00:07	49	model 20	185	WARNING
1535632	2016	11	29	06:00:07	49	model 20	185	WARNING

Quality

Faults Forecast

Faults Root causes

Counters

Serial	YYYY	MM	dd	hh:mm:ss	Week	Model	LabelCounter	AbsoluteCounter	RelativeCounter
1535632	2016	11	29	04:00:07	49	model 31	numcaffegenerale	179112	0
1535632	2016	11	29	05:00:07	49	model 31	numcaffegenerale	179120	8
1535632	2016	11	29	06:00:07	49	model 31	numcaffegenerale	179158	38

45.330.415 entries
395 csv files
2.5 GB

Faults

Serial	YYYY	MM	dd	hh:mm:ss	Week	Model	Errorcode	Critical
1535632	2016	11	29	04:00:07	49	model 62	185	WARNING
1535632	2016	11	29	05:00:07	49	model 20	185	WARNING
1535632	2016	11	29	06:00:07	49	model 20	185	WARNING

364.695 entries
395 csv files
13 MB

DataScienceSeed

Cleanings

Serial	YYYY	MM	dd	hh:mm:ss	Week	Model	Errorcode
1535632	2016	11	29	04:00:07	49	model 10	1
1535632	2016	11	29	05:00:07	49	model 9	1
1535632	2016	11	29	06:00:07	49	model 31	1

73.861 entries
391 csv files
3 MB

DataScienceSeed

Serial	YYYY	MM	dd	hh:mm:ss	Week	Model	LabelCounter	AbsoluteCounter	RelativeCounter
1535632	2016	11	29	04:00:07	49	model 31	numcaffegenerale	179112	0
1535632	2016	11	29	05:00:07	49	model 31	numcaffegenerale	179120	8
1535632	2016	11	29	06:00:07	49	model 31	numcaffegenerale	179158	38

Serial	YYYY	MM	dd	hh:mm:ss	Week	Model	Errorcode
1535632	2016	11	29	04:00:07	49	model 10	1
1535632	2016	11	29	05:00:07	49	model 9	1
1535632	2016	11	29	06:00:07	49	model 31	1

Serial	YYYY	MM	dd	hh:mm:ss	Week	Model	Errorcode	Critical
1535632	2016	11	29	04:00:07	49	model 62	185	WARNING
1535632	2016	11	29	05:00:07	49	model 20	185	WARNING
1535632	2016	11	29	06:00:07	49	model 20	185	WARNING

Exploration (EDA)

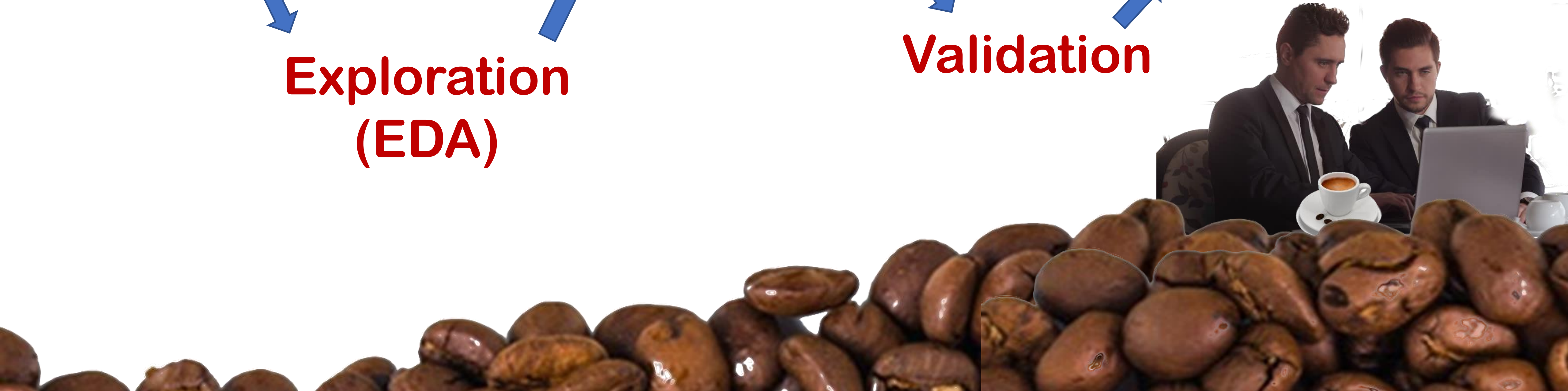
Cleaning

Preparation

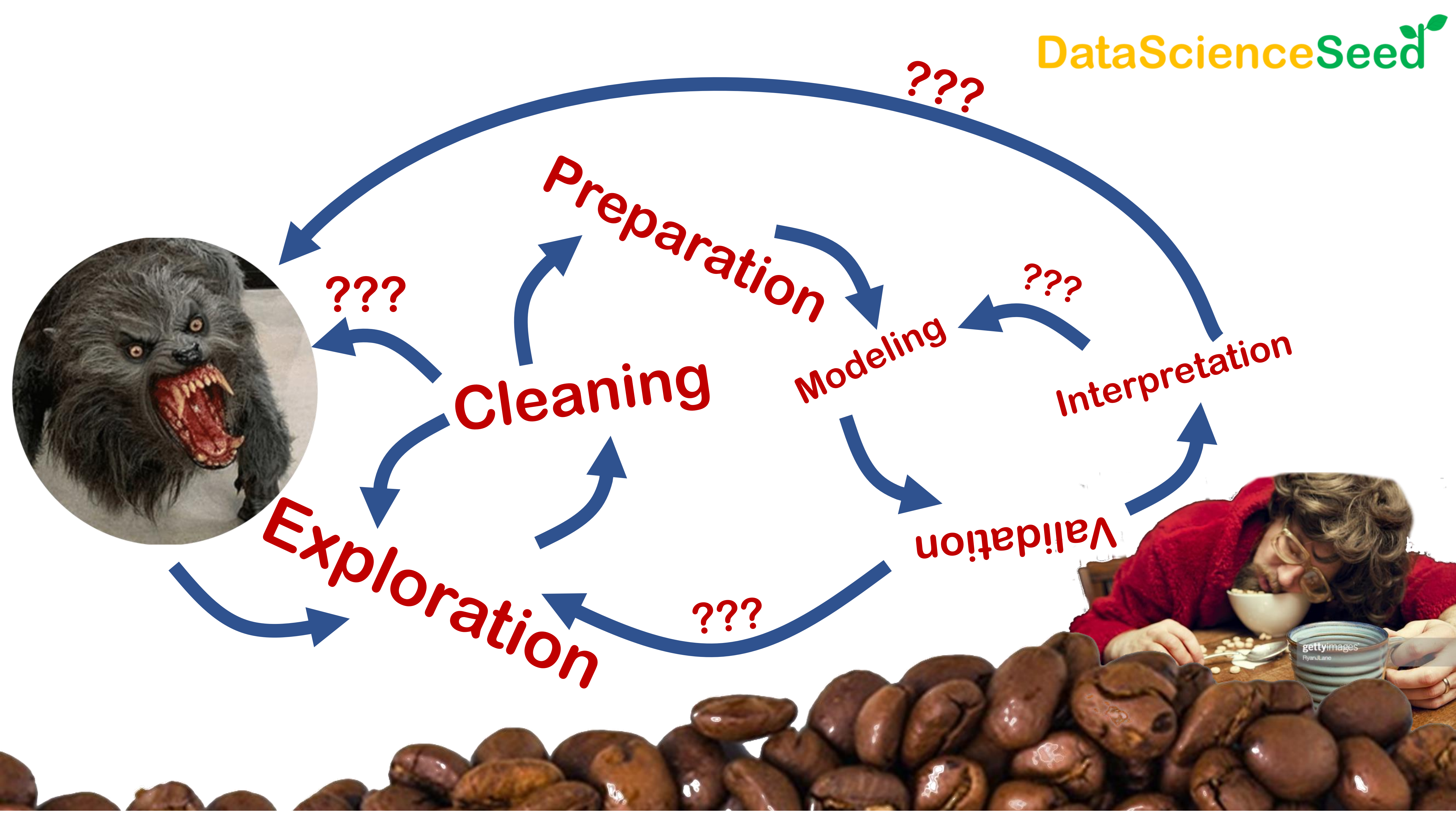
Modeling

Validation

Interpretation




Iris Dataset
Real World Dataset


DataScienceSeed
???
Preparation
???
???
Modeling
Cleaning
Interpretation
Exploration
Validation
???
gettyimages


Preparation
Cleaning
Exploration

Let's see the data in its face!

In [5]:
```
df_counters.head()
```

Out[5]:

	Serial	YYYY	MM	dd	hh:mm:ss	Week	Model	LabelCounter	AbsoluteCounter	RelativeCounter
0	1535632	2016	11	29	00:00:38	49	model 31	numcaffegenerale	179112.0	0.0
1	1535632	2016	11	29	01:00:07	49	model 31	numcaffegenerale	179112.0	0.0
2	1535632	2016	11	29	02:00:07	49	model 31	numcaffegenerale	179112.0	0.0
3	1535632	2016	11	29	03:00:07	49	model 31	numcaffegenerale	179112.0	0.0
4	1535632	2016	11	29	04:00:07	49	model 31	numcaffegenerale	179112.0	0.0

In [6]:
```
df_counters.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 45330415 entries, 0 to 192929
Data columns (total 10 columns):
Serial             int64
YYYY               int64
MM                 int64
dd                 int64
hh:mm:ss           object
Week               int64
Model              object
LabelCounter       object
AbsoluteCounter    float64
RelativeCounter    float64
dtypes: float64(2), int64(5), object(3)
memory usage: 3.7+ GB
```

Explore counters

DataScienceSeed

```
Out[11]: LabelCounter
numcaffegr1          78
numcaffegenerale     77
numacqua             75
nummac1gr1           50
nummac2gr1           50
numcicligr1          50
numlattegr1          48
numvaporeariats      41
numvapore            38
numlattefr           36
numcaffegr2          32
numvaporets          31
ngr2                 25
ngr1                 25
tempogr2             25
                     ..
numcaffegr3          19
ngr3                 17
portatagr3           17
tempogr3             16
nummac3gr1            8
numcaffegr4           4
numsolubile           4
portatagr4            3
ngr4                  3
tempogr4              3
numcicligr2           2
nummac3gr2            2
numlattegr2           2
nummac1gr2            2
nummac2gr2            2
Name: Model, Length: 34, dtype: int64
```

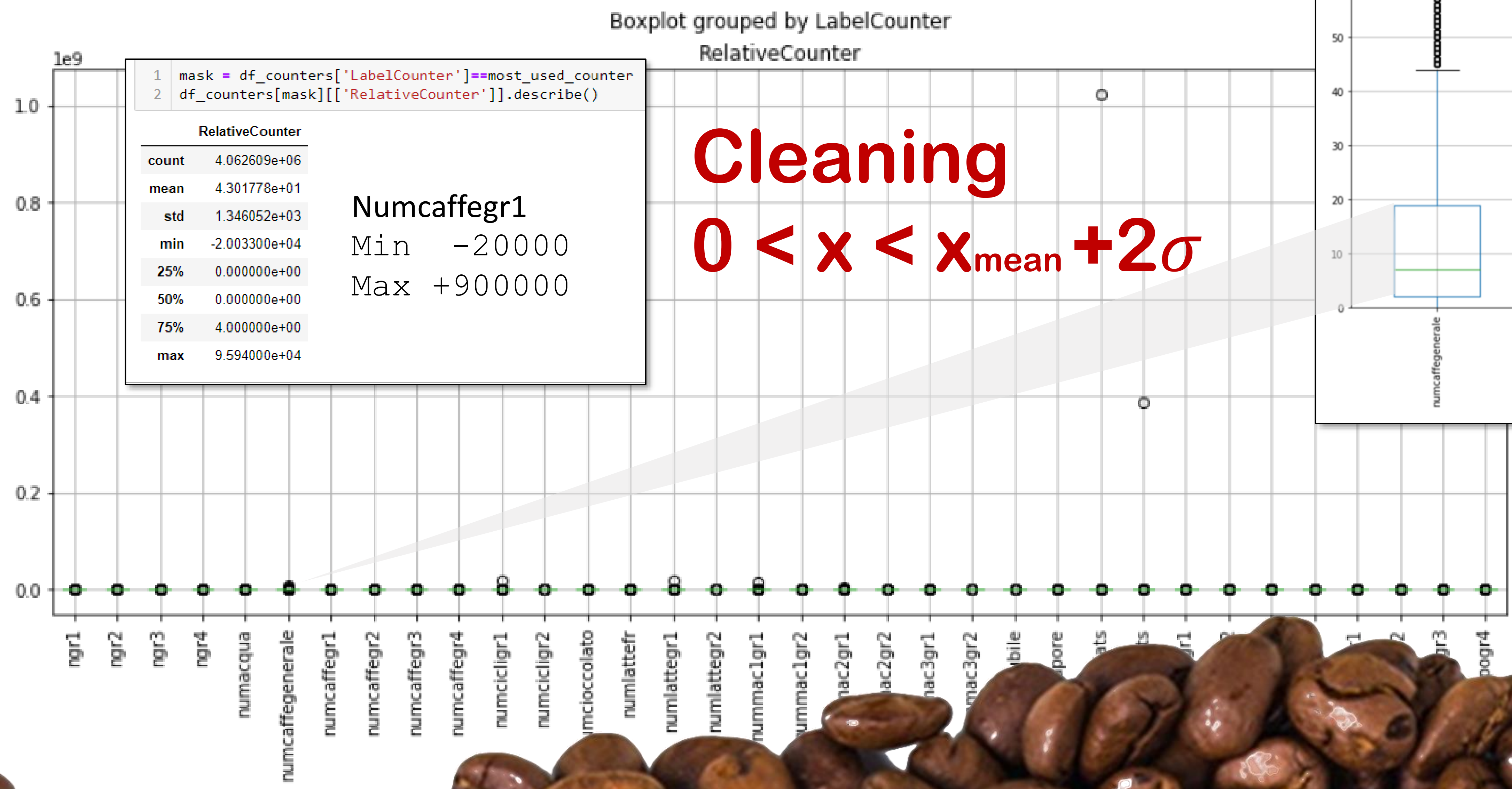
	Serial	YYYY	MM	dd	hh:mm:ss	Week	Model	LabelCounter	AbsoluteCounter	RelativeCounter
0	1535632	2016	11	29	00:00:38	49	model 31	numcaffegenerale	179112.0	0.0
1	1535632	2016	11	29	01:00:07	49	model 31	numcaffegenerale	179112.0	0.0
2	1535632	2016	11	29	02:00:07	49	model 31	numcaffegenerale	179112.0	0.0
3	1535632	2016	11	29	03:00:07	49	model 31	numcaffegenerale	179112.0	0.0
4	1535632	2016	11	29	04:00:07	49	model 31	numcaffegenerale	179112.0	0.0

```
df_counters[['Model','LabelCounter']].drop_duplicates()\
                    .groupby('LabelCounter')['Model'].count().sort_values(ascending=False)
```

Cleaning?

Counter names are not homogenous - numcaffegr1

```python
df_counters[['LabelCounter','RelativeCounter']].boxplot(by='LabelCounter',figsize=(15,6))
plt.xticks(rotation=90)
plt.show()
```




```python
mask = df_counters['LabelCounter']==most_used_counter
df_counters[mask][['RelativeCounter']].describe()
```

	RelativeCounter
count	4.062609e+06
mean	4.301778e+01
std	1.346052e+03
min	-2.003300e+04
25%	0.000000e+00
50%	0.000000e+00
75%	4.000000e+00
max	9.594000e+04

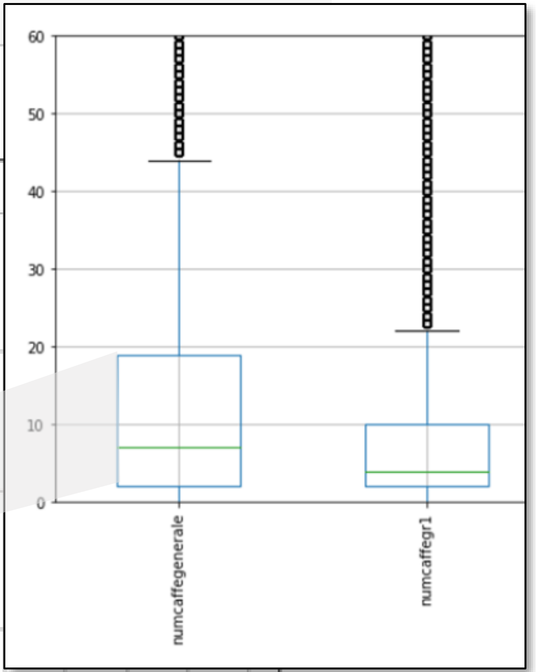
Numcaffegr1

Min -20000

Max +900000

Cleaning

$$0 < x < x_{mean} + 2\sigma$$



Explore machines population

Let's see how many different machines we have in the datasets

In [29]:
```
set_counters   = set(df_counters['Serial'].unique())
set_faults     = set(df_faults['Serial'].unique())
set_cleanings  = set(df_cleanings['Serial'].unique())
print("SN in counters  : %5d"%len(set_counters))
print("SN in faults    : %5d"%len(set_faults))
print("SN in cleanings : %5d"%len(set_cleanings))
```

```
SN in counters  :  1258
SN in faults    :  1109
SN in cleanings :   718
```

In [30]:
```
print("SN in cleaning & faults : %5d"%len(set_cleanings & set_faults))
print("SN in counters & faults : %5d"%len(set_counters & set_faults))
print("SN in all sets          : %5d"%len(set_cleanings & set_counters & set_faults))
```

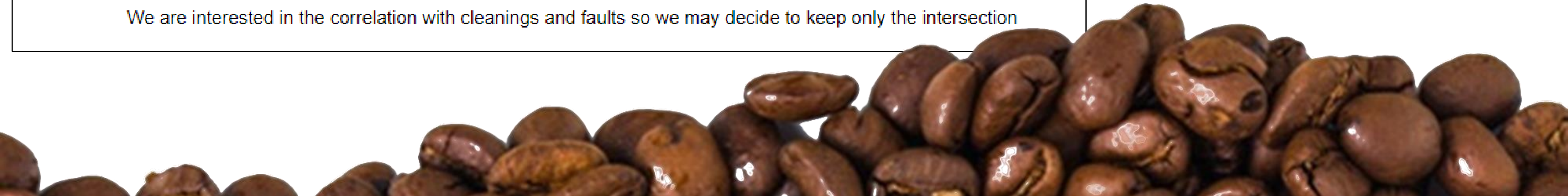
```
SN in cleaning & faults :   688
SN in counters & faults :  1102
SN in all sets          :   684
```

What machines to keep in the dataset to build the model?

We are interested in the correlation with cleanings and faults so we may decide to keep only the intersection

Cleaning?

DataScienceSeed



DataScienceSeed

In [31]:
```python
models_population = pd.concat([df_counters[['Model','Serial']],
                               df_faults[['Model','Serial']],
                               df_cleanings[['Model','Serial']]]).drop_duplicates()['Model'].value_counts()

models_population
```

Out[31]:
```
model 20    263
model 10    183
model 13    168
model 9     150
model 11     47
model 12     41
model 8      41
model 60     40
model 63     36
model 50     21
model 14     20
model 31     17
model 5      15
model 19     15
model 22     14
            ...
model 68      1
model 69      1
model 30      1
model 45      1
model 43      1
model 28      1
model 29      1
model 4       1
model 58      1
model 46      1
model 75      1
model 36      1
model 65      1
model 49      1
model 67      1
Name: Model, Length: 78, dtype: int64
```

Serial	YYYY	MM	dd	hh:mm:ss	Week	Model	LabelCounter	AbsoluteCounter	RelativeCounter
1535632	2016	11	29	04:00:07	49	model 31	numcaffegenerale	179112	0
1535632	2016	11	29	05:00:07	49	model 31	numcaffegenerale	179120	8
1535632	2016	11	29	06:00:07	49	model 31	numcaffegenerale	179158	38

Serial	YYYY	MM	dd	hh:mm:ss	Week	Model	Errorcode	Critical
1535632	2016	11	29	04:00:07	49	model 62	185	WARNING
1535632	2016	11	29	05:00:07	49	model 20	185	WARNING
1535632	2016	11	29	06:00:07	49	model 20	185	WARNING

Serial	YYYY	MM	dd	hh:mm:ss	Week	Model	Errorcode
1535632	2016	11	29	04:00:07	49	model 10	1
1535632	2016	11	29	05:00:07	49	model 9	1
1535632	2016	11	29	06:00:07	49	model 31	1

Explore models

DataScienceSeed

Explore Faults

How many "critical" values and how many records per each one?

In [33]:
```
df_faults['Critical'].value_counts()
```

Out[33]:
```
1           192014
CRITICAL     53931
WARNING      44889
Name: Critical, dtype: int64
```

The alarm type "1" is numeric an not homogeneous with CRITICAL and WARNING Let's convert it to a string

In [34]:
```
# if-then on one column, see
# http://pandas-docs.github.io/pandas-docs-travis/user_guide/cookbook.html
df_faults.loc[df_faults['Critical']==1,'Critical'] = 'ONE'
```

In [35]:
```
df_faults['Critical'].value_counts()

```

Out[35]:
```
ONE         192014
CRITICAL     53931
WARNING      44889
Name: Critical, dtype: int64
```

Serial	YYYY	MM	dd	hh:mm:ss	Week	Model	Errorcode	Critical
1535632	2016	11	29	04:00:07	49	model 62	185	WARNING
1535632	2016	11	29	05:00:07	49	model 20	185	WARNING
1535632	2016	11	29	06:00:07	49	model 20	185	WARNING

How many error codes per each category?

In [36]:
```
df_faults[['Critical','ErrorCode']].drop_duplicates()['Critical'].value_counts()
```

Out[36]:
```
ONE         80
WARNING     53
CRITICAL    38
Name: Critical, dtype: int64
```

Explore Faults Critical vs ErrorCode

DataScienceSeed

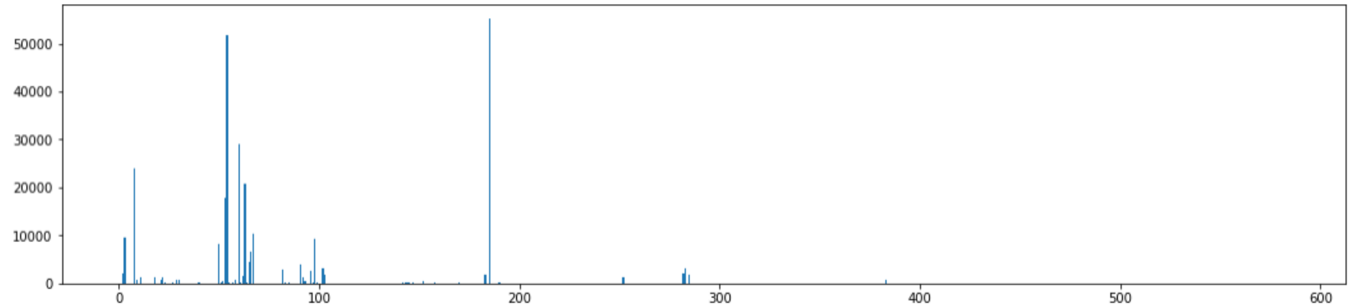
How many error records per error code

```
In [37]:  df_faults['ErrorCode'].value_counts().sort_index()
```

```
Out[37]: 1        2
         2     2143
         3     9600
         7        2
         8    24091
         9      738
         11    1223
         18    1299
         20      14
         21     817
         22    1200
         23     107
         24      24
         25      16
         26      15
                ...
         251     18
         252   1334
         266     11
         270     16
         282   2100
         283   3197
         285   1935
         351      6
         352     50
         366      8
         383    682
         421      2
         451      1
         483      6
         583      1
         Name: ErrorCode, Length: 91, dtype: int64
```

Some error codes are much more freqent than others. Let's see this in a bar diagram

```
In [38]:
plt.figure(figsize=(18,4))
plt.bar(df_faults['ErrorCode'].value_counts().sort_index().index,
        df_faults['ErrorCode'].value_counts().sort_index().values)
plt.show()
```



What is the error code with the maximum number of occurrencies?

```
In [39]:  df_faults['ErrorCode'].value_counts().idxmax()
Out[39]:  185
```

What tipe of error is it?

```
In [40]:  df_faults[df_faults['ErrorCode']==185]['Critical'].value_counts()
Out[40]:  ONE        44448
          WARNING    10763
          CRITICAL      98
          Name: Critical, dtype: int64
```

Cleaning?

DataScienceSeed

Faults distribution across models

To explore this, let's summarize into a dataset indexed by machine model, starting with how many machines there are for each model and how many faults for each model

In [48]:

```python
# Merge models_population with the number of faults recorded
df_issues_summary = pd.concat([models_population,
                               df_faults['Model'].value_counts()], axis=1,sort=False)
# Rename the columns
df_issues_summary.columns = ['Population','Faults']
df_issues_summary.head()
```

Out[48]:

	Population	Faults
model 20	263	90743.0
model 10	183	48821.0
model 13	168	79880.0
model 9	150	15048.0
model 11	47	6007.0

Let's split the faults per type (or severity)

In [49]:

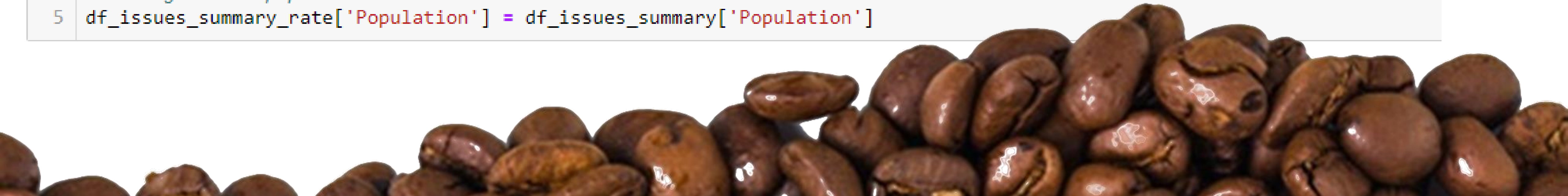
```python
df_faults.groupby(['Critical','Model'])['Serial'].count()\
         .unstack(0).sort_values('CRITICAL',ascending=False).head()
```

Out[49]:

Critical Model	CLEANING	CRITICAL	ONE	WARNING
model 13	11729.0	36755.0	19616.0	11780.0
model 10	6549.0	12185.0	24904.0	5183.0
model 9	3642.0	1345.0	8785.0	1276.0
model 12	714.0	626.0	3080.0	420.0
model 17	554.0	579.0	1290.0	309.0

Let's compute the *ratio* of faults per machine type

```python
# create a dataframe as df_issues_summary with absolute values divided by the number of machines of a certain model
df_issues_summary_rate = df_issues_summary[df_issues_summary.columns[1:]]\
                    .div(df_issues_summary['Population'], axis=0)
# add again the population column
df_issues_summary_rate['Population'] = df_issues_summary['Population']
```



DataScienceSeed

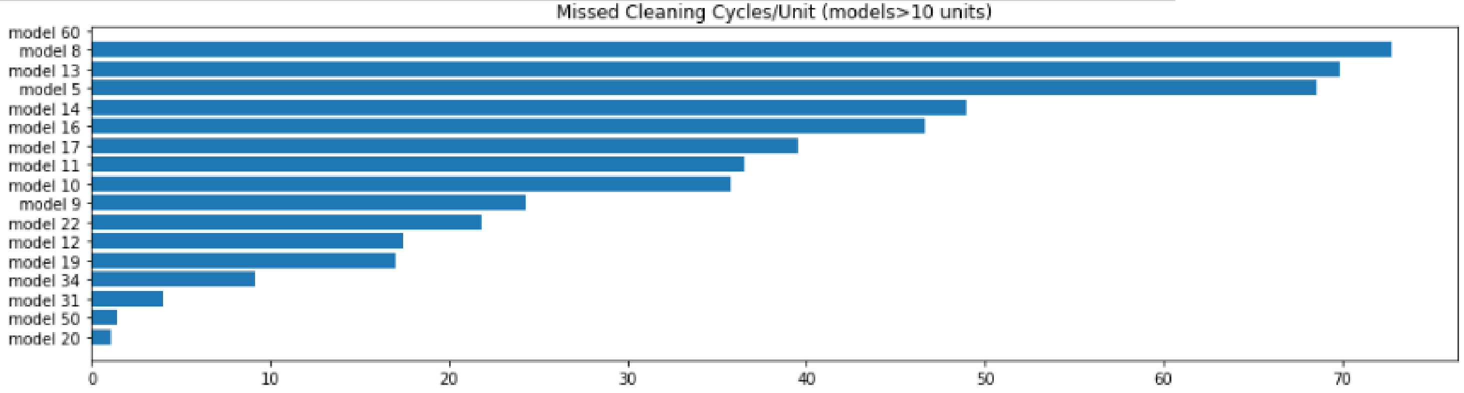
```python
minimum_number_of_machines = 10
```

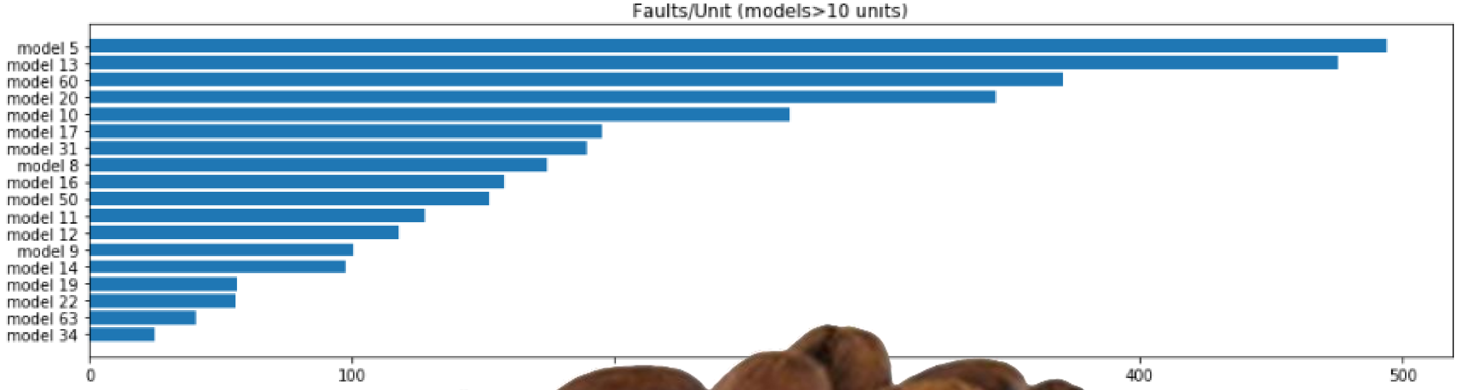
```python
_df = df_issues_summary_rate[df_issues_summary_rate['Population']>=minimum_number_of_machines]
```

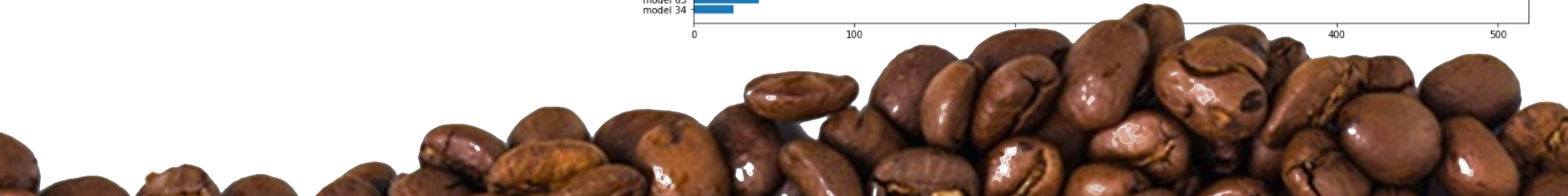
```python
_df[['Population','Faults','ONE','CRITICAL','WARNING','CLEANING']]\
                                .sort_values('Faults',ascending=False)
```

	Population	Faults	ONE	CRITICAL	WARNING	CLEANING
model 5	15	494.200000	356.666667	1.400000	67.600000	68.533333
model 13	168	475.476190	116.761905	218.779762	70.119048	69.815476
model 60	40	370.750000	354.900000	0.025000	15.825000	NaN
model 20	263	345.030418	268.768061	0.547529	74.615970	1.098859
model 10	183	266.781421	136.087432	66.584699	28.322404	35.786885
model 17	14	195.142857	92.142857	41.357143	22.071429	39.571429
model 31	17	189.411765	171.294118	0.058824	14.000000	4.058824
model 8	41	174.463415	83.219512	13.975610	4.487805	72.780488
model 16	14	157.785714	110.928571	0.142857	0.071429	46.642857
model 50	21	152.095238	130.095238	NaN	20.571429	1.428571
model 11	47	127.808511	66.808511	12.297872	12.148936	36.553191
model 12	41	118.048780	75.121951	15.268293	10.243902	17.414634
model 9	150	100.320000	58.566667	8.966667	8.506667	24.280000
model 14	20	97.850000	32.050000	9.050000	7.800000	48.950000
model 19	15	56.466667	35.400000	1.466667	2.600000	17.000000
model 22	14	55.857143	32.500000	0.428571	1.071429	21.857143
model 63	36	40.750000	39.111111	0.444444	1.194444	NaN
model 34	11	25.000000	14.636364	0.818182	0.363636	9.181818

```python
plt.figure(figsize=(16,4))
plt.barh(_df['CLEANING'].sort_values().index,
         _df['CLEANING'].sort_values())
plt.title("Missed Cleaning Cycles/Unit (models>10 units)")
plt.show()
```





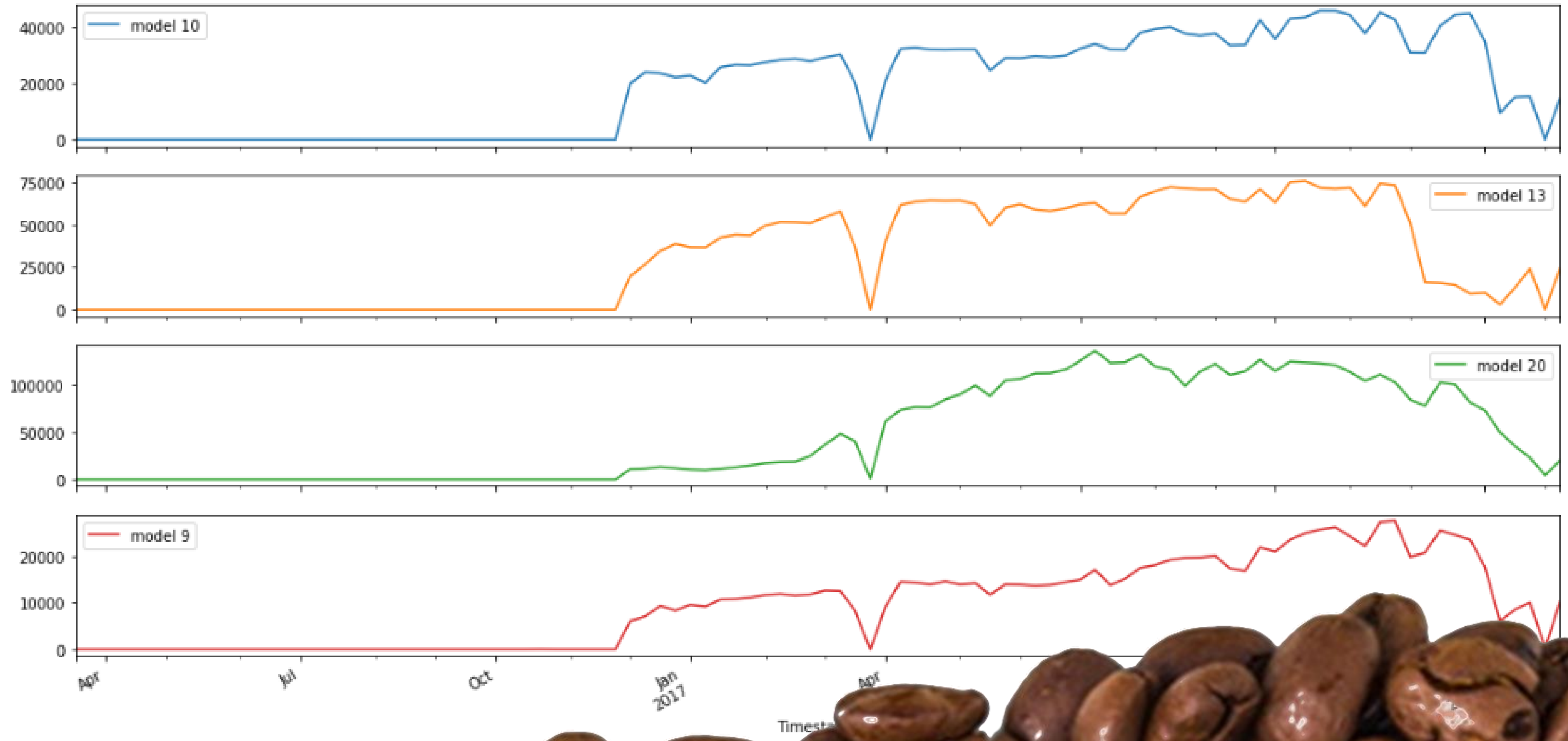


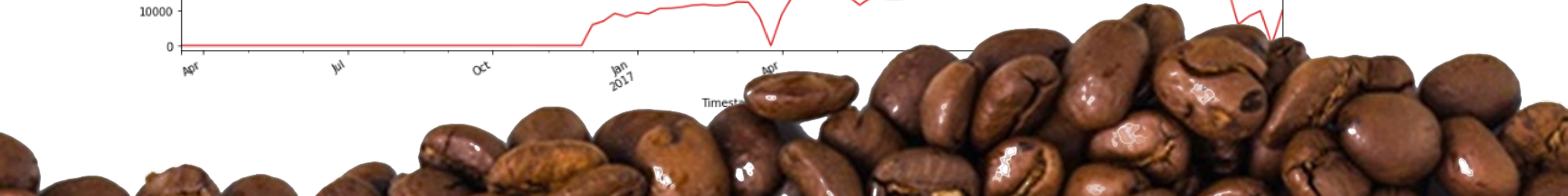
DataScienceSeed

Timeline on counters (per machine type)

Let's check when exactly machines start genereting counters

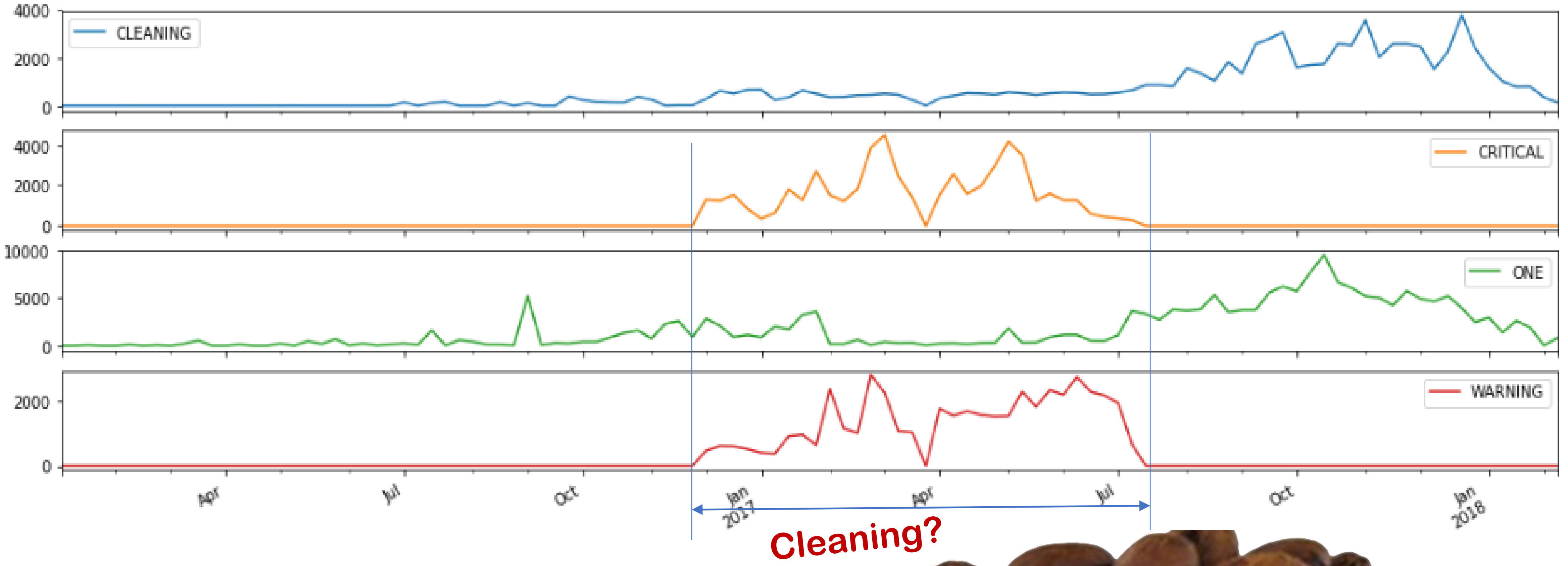
```
df_countersf[df_countersf['Model'].isin(models_to_look)]\
            .groupby(['Model','Timestamp'])['Serial'].count().unstack('Model')\
            .resample('W').sum().plot(figsize=(18,9),subplots=True)
plt.show()
```





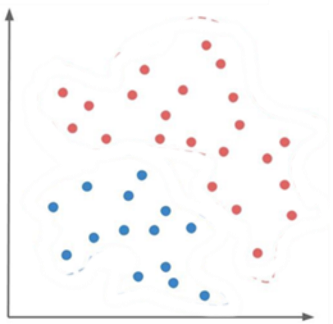
DataScienceSeed

```
df_faultsf.groupby(['Critical','Timestamp'])['Serial'].count()\
            .unstack('Critical').resample('W').sum().plot(figsize=(18,6),subplots=True)
plt.show()
```



DataScienceSeed

Modeling

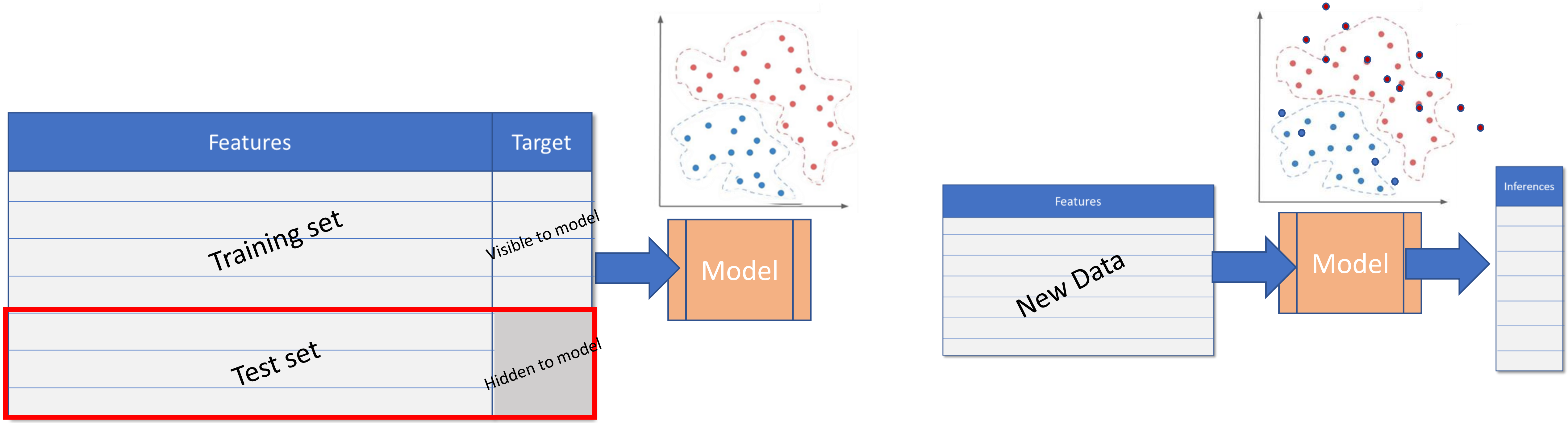


DataScienceSeed

Modeling

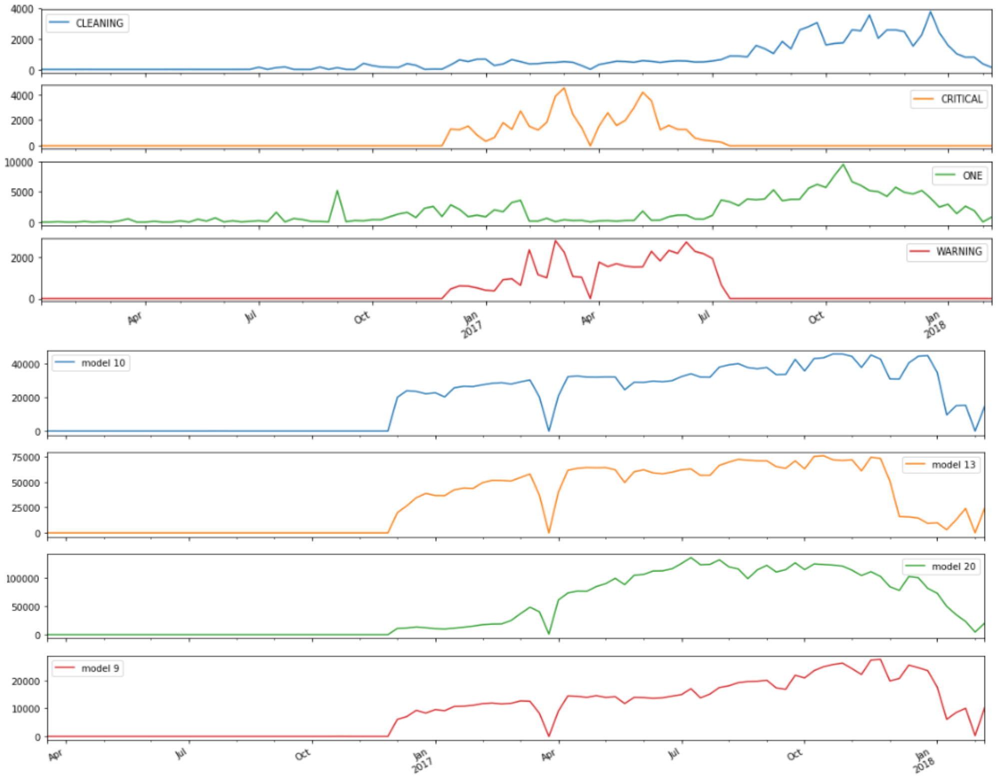
DataScienceSeed

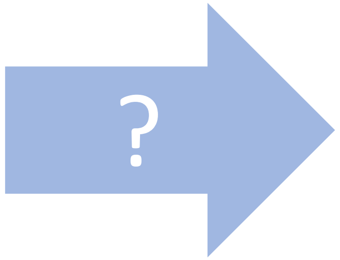
Modeling



DataScienceSeed

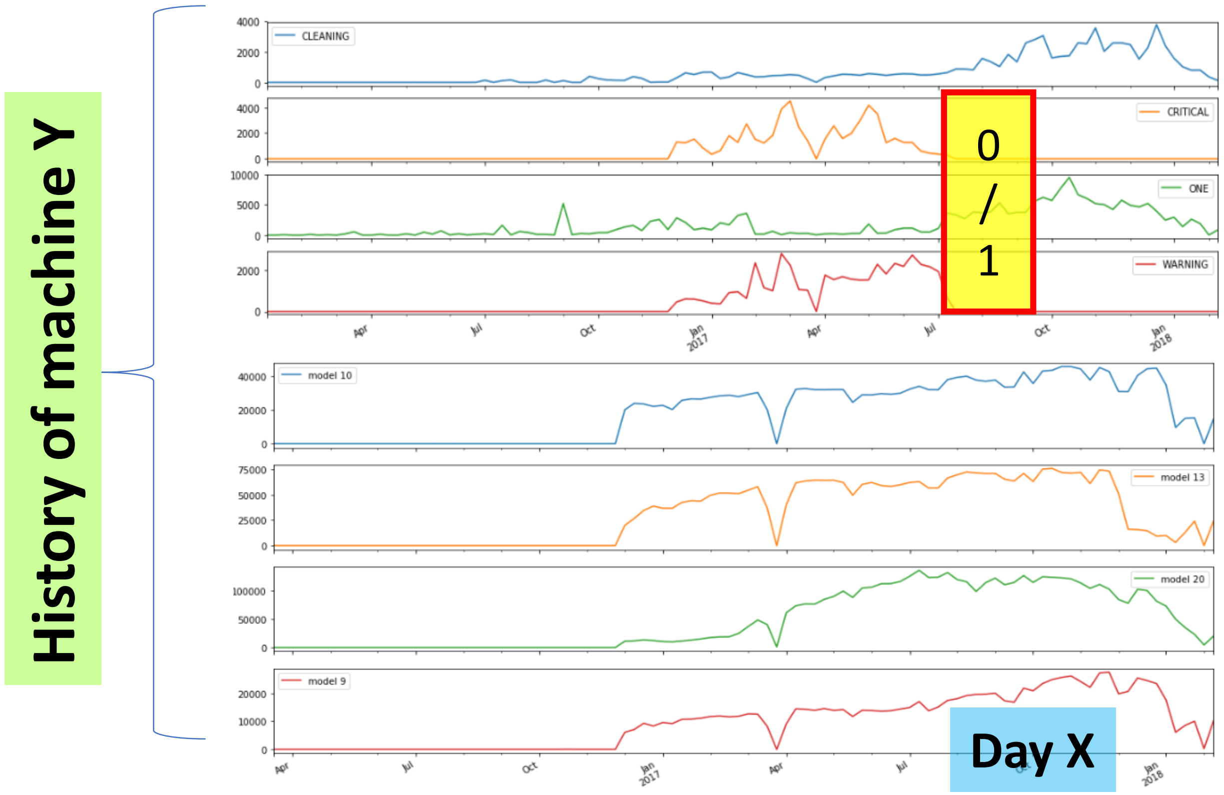
Preparation – Feature Engineering

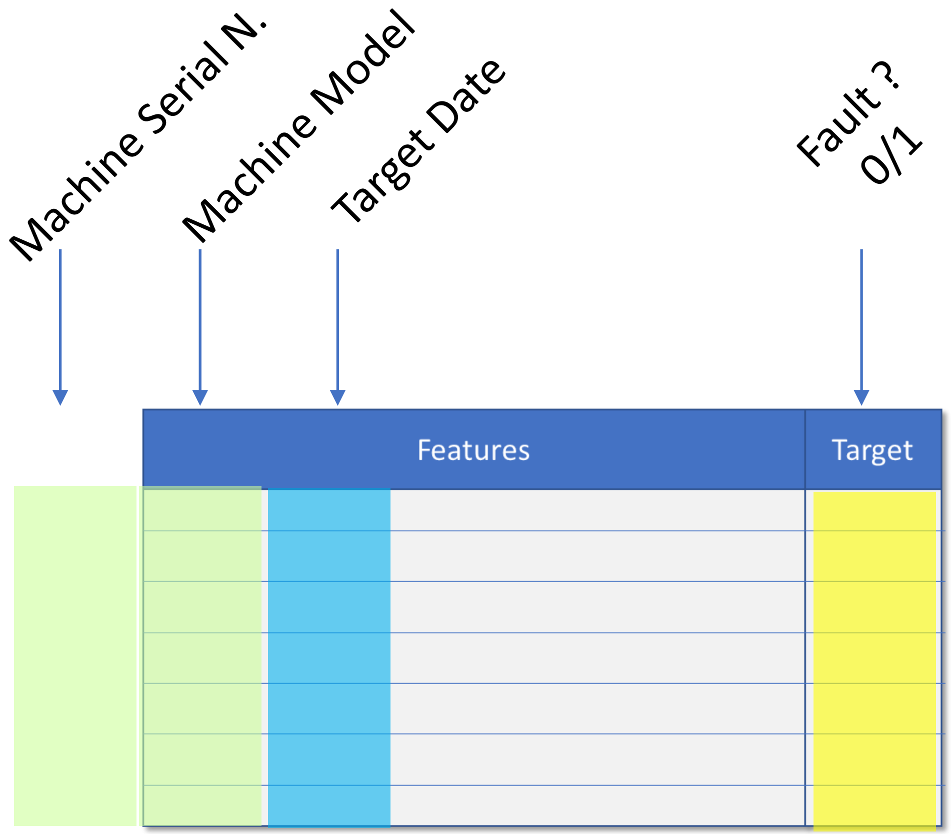


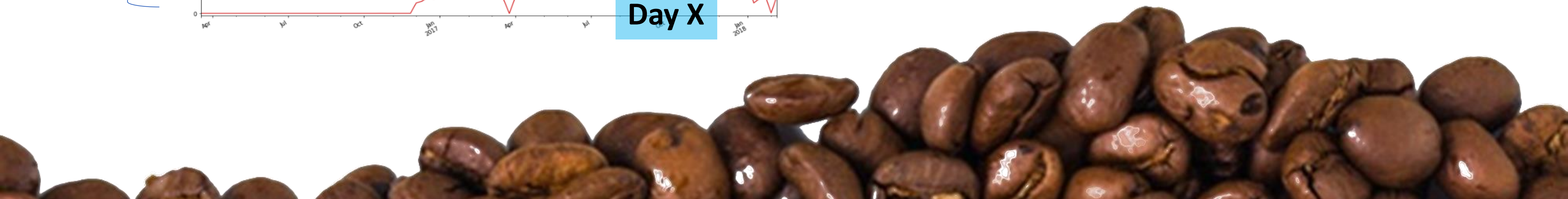


Features	Target

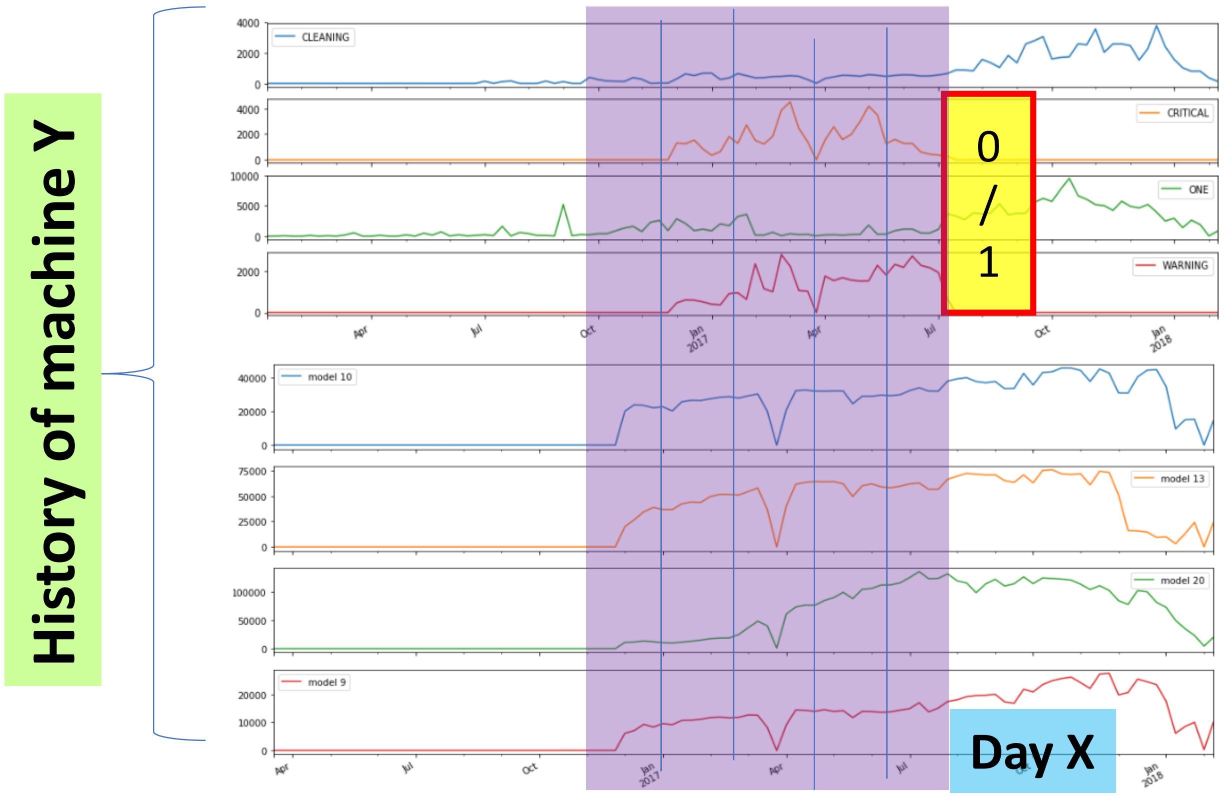
TARGET: there will be a Fault on day X for machine Y?

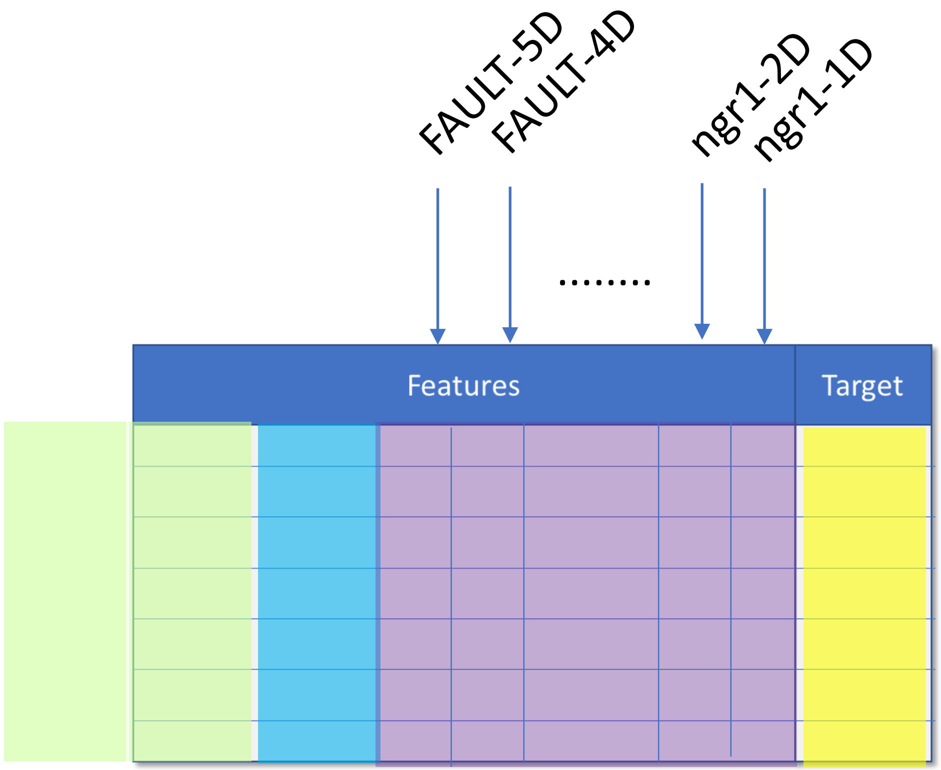






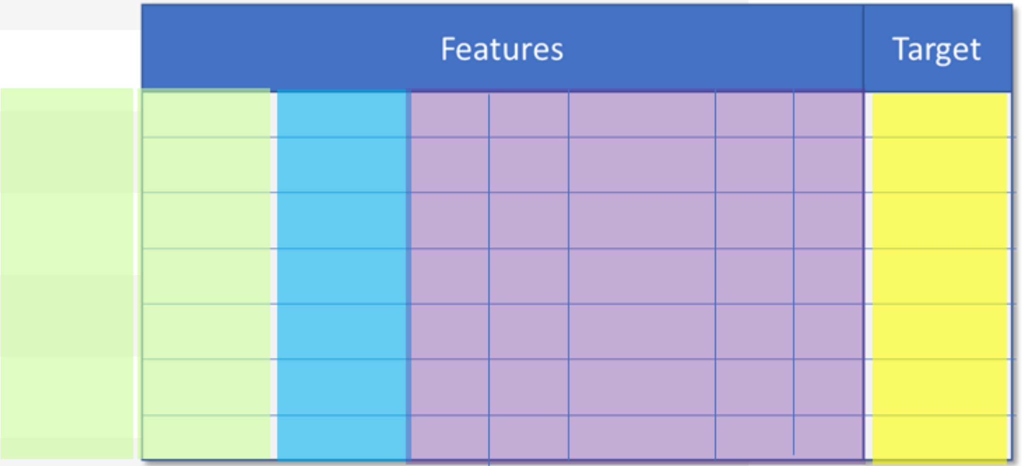
Features: Faults, Cleanings, Counters
In «some» days before day X





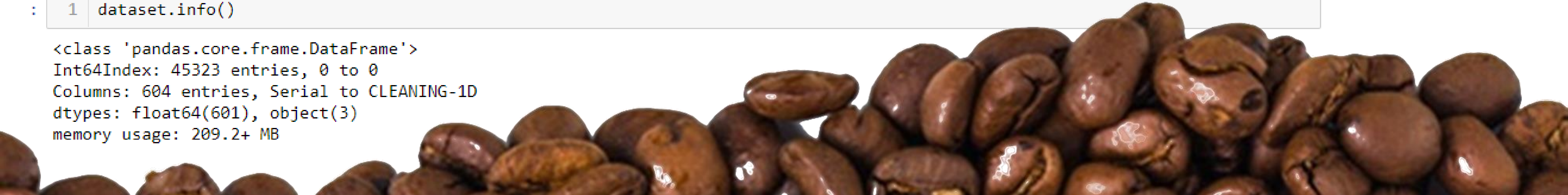
```
dataset.head(10)
```

	Serial	Model	Target Timestamp	TARGET	numcaffegenerale-20D	numcaffegenerale-19D	numcaffegenerale-18D	numcaffegenerale-17D	numcaffegenerale-16D	numcaffegenerale-15D	n
0	1479635	model 20	2016-10-21	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
0	1479635	model 20	2016-10-22	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
0	1479635	model 20	2016-10-23	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
0	1479635	model 20	2016-10-24	0.0	0.0	0.0	0.0				
0	1479635	model 20	2016-10-25	0.0	0.0	0.0	0.0				
0	1479635	model 20	2016-10-26	0.0	0.0	0.0	0.0				
0	1479635	model 20	2016-10-27	0.0	0.0	0.0	0.0				
0	1479635	model 20	2016-10-28	0.0	0.0	0.0	0.0				
0	1479635	model 20	2016-10-29	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
0	1479635	model 20	2016-10-30	0.0	0.0	0.0	0.0	0.0	0.0	0.0	

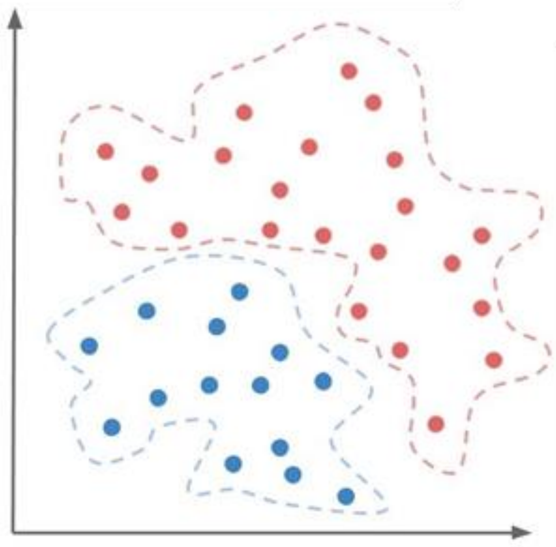



```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 45323 entries, 0 to 0
Columns: 604 entries, Serial to CLEANING-1D
dtypes: float64(601), object(3)
memory usage: 209.2+ MB
```



Models

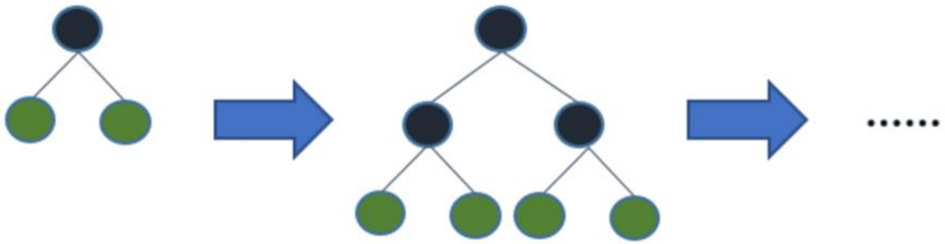


Ligh Gradient Boost Model

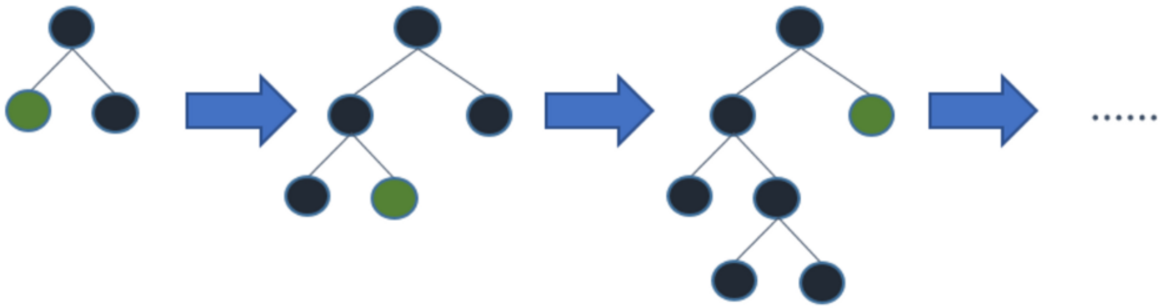
Neural Network

Light-GBM

[Quick overview on Medium]



Level-wise tree growth



Leaf-wise tree growth

E.g. Decision tree – Random Forest

E.g. LGBM

Light-GBM

DataScienceSeed

[Quick overview on Medium]

- Big Dataset (10K+ rows)
- Efficient memory usage
- Light speed
- High Accuracy
- Categorical friendly

- Requires some tuning

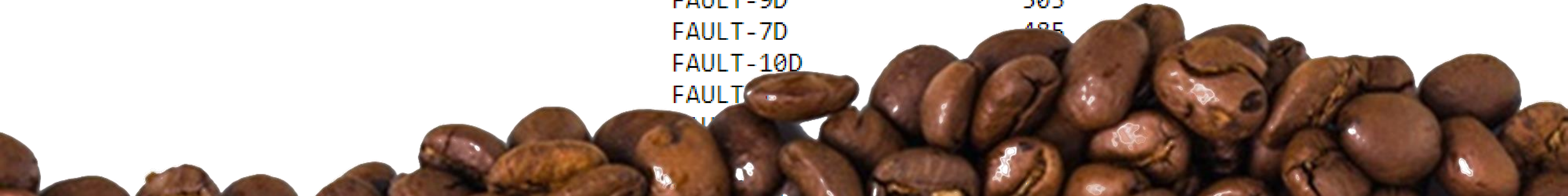
```
Fitting time: 25.623752487267904 s
Classification Report
              precision    recall  f1-score   support

           0       0.87      0.85      0.86      7829
           1       0.83      0.86      0.85      7128

   micro avg       0.85      0.85      0.85     14957
   macro avg       0.85      0.85      0.85     14957
weighted avg       0.85      0.85      0.85     14957

Confusion Matrix
[[6616 1213]
 [ 999 6129]]
F1: 0.8471319972356598
f_name
Week                      979
FAULT-6D                  577
FAULT-9D                  503
FAULT-7D
FAULT-10D
FAULT
```

		True condition	
		Condition positive	Condition negative
Predicted condition	Predicted condition positive	**True positive**	**False positive**, Type I error
	Predicted condition negative	**False negative**, Type II error	**True negative**

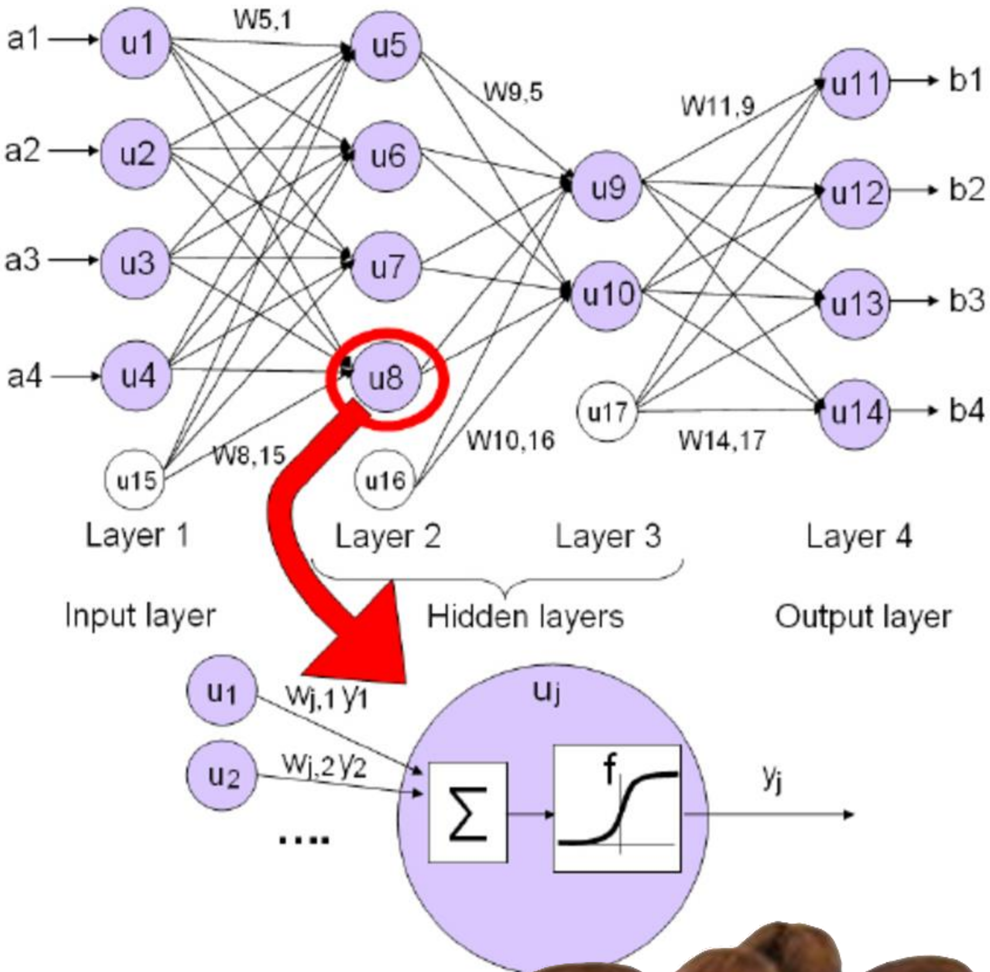


Neural Network

DataScienceSeed

Parameters

- Layers
- Neurons / Layer
- Activation
- Batch size
- Epochs



DataScienceSeed

*keras3.py

File Edit Format Run Options Windows Help

Build inner layers

```
model = Sequential()
for lrs in range(n_of_layers):
        model.add(Dense(int(neurons_per_layer),
                        kernel_initializer='random_uniform',
                        bias_initializer='zeros',
                        kernel_regularizer=regularizers.l2(lambd)))
        model.add(Activation(activation))
```

Build output layer

```
model.add(Dense(1, kernel_initializer='random_uniform',
                   bias_initializer='zeros',
                   kernel_regularizer=regularizers.l2(lambd)))
model.add(Activation('sigmoid'))

model.compile(optimizer=optimizers.Adam(),
              loss='binary_crossentropy',
              metrics=['accuracy'])
```

Train and evaluate

```
model.fit(train_data, train_labels, epochs=n_of_epochs, verbose=0)
train_loss, train_acc = model.evaluate(train_data, train_labels, batch_size=batch_size)
test_loss, test_acc = model.evaluate(test_data, test_labels, batch_size=batch_size)
```

Ln: 84 Col: 22

DataScienceSeed

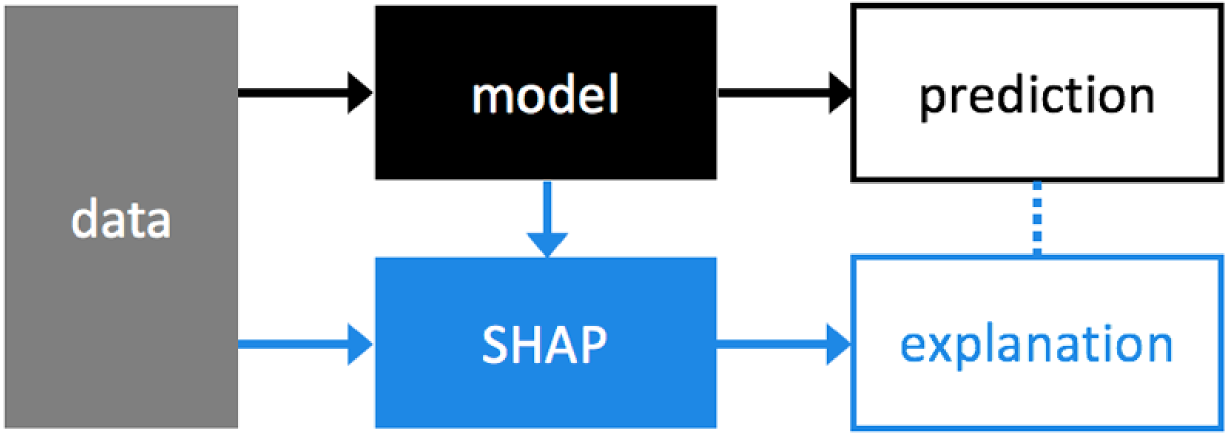
Plot performance vs lambda
Vs parametri

trial	lambda	n_of_epochs	batch_size	n_of_layers	neurons_per_layer	activation	train_acc	test_acc	dataset
1218	0	100	100	3	128	tanh	0.890487369	0.880337079	FlairBit_5d_onehot_scaled.csv
1203	0	100	100	3	128	tanh	0.892935252	0.878089887	FlairBit_5d_onehot_scaled.csv
1225	0	100	100	3	128	tanh	0.890286725	0.877849117	FlairBit_5d_onehot_scaled.csv
246	0	100	100	3	256	tanh	0.876785714	0.877447833	FlairBit_5d_onehot_rm15_scaled.csv
259	0	100	100	3	128	tanh	0.878009631	0.877367578	FlairBit_5d_onehot_rm15_scaled.csv
1229	0	100	100	3	64	tanh	0.888400651	0.876565009	FlairBit_5d_onehot_scaled.csv
239	0	100	100	3	128	tanh	0.881560995	0.875361155	FlairBit_5d_onehot_rm15_scaled.csv
240	0	100	100	3	128	tanh	0.879835473	0.875280898	FlairBit_5d_onehot_rm15_scaled.csv
241	0	100	100	3	64	tanh	0.875441412	0.875040131	FlairBit_5d_onehot_rm15_scaled.csv
1214	0	100	100	3	32	tanh	0.882622042	0.874638844	FlairBit_5d_onehot_scaled.csv
255	0	100	100	3	64	tanh	0.880758428	0.874558585	FlairBit_5d_onehot_rm15_scaled.csv
1211	0	100	100	3	256	tanh	0.893918416	0.874398074	FlairBit_5d_onehot_scaled.csv
235	0	100	100	3	256	tanh	0.879514446	0.874317818	FlairBit_5d_onehot_rm15_scaled.csv
1213	0	100	100	3	32	tanh	0.884146954	0.874077048	FlairBit_5d_onehot_scaled.csv
238	0	100	100	3	16	tanh	0.870505618	0.874077043	FlairBit_5d_onehot_rm15_scaled.csv
323	0	100	100	3	16	relu	0.868398875	0.873675762	FlairBit_5d_onehot_rm15_scaled.csv
257	0	100	100	3	128	tanh	0.880357144	0.873675762	FlairBit_5d_onehot_rm15_scaled.csv
234	0	100	100	3	128	tanh	0.882182986	[illegible]	FlairBit_5d_onehot_rm15_scaled.csv
254	0	100	100	3	128	tanh	[illegible]	[illegible]	[illegible]
1208	0	100	100	3	128	tanh	[illegible]	[illegible]	[illegible]
1215	0	100	100	3	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
245	0	100	100	3	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

Interpretation: SHAP

DataScienceSeed

[Shap Documentation]

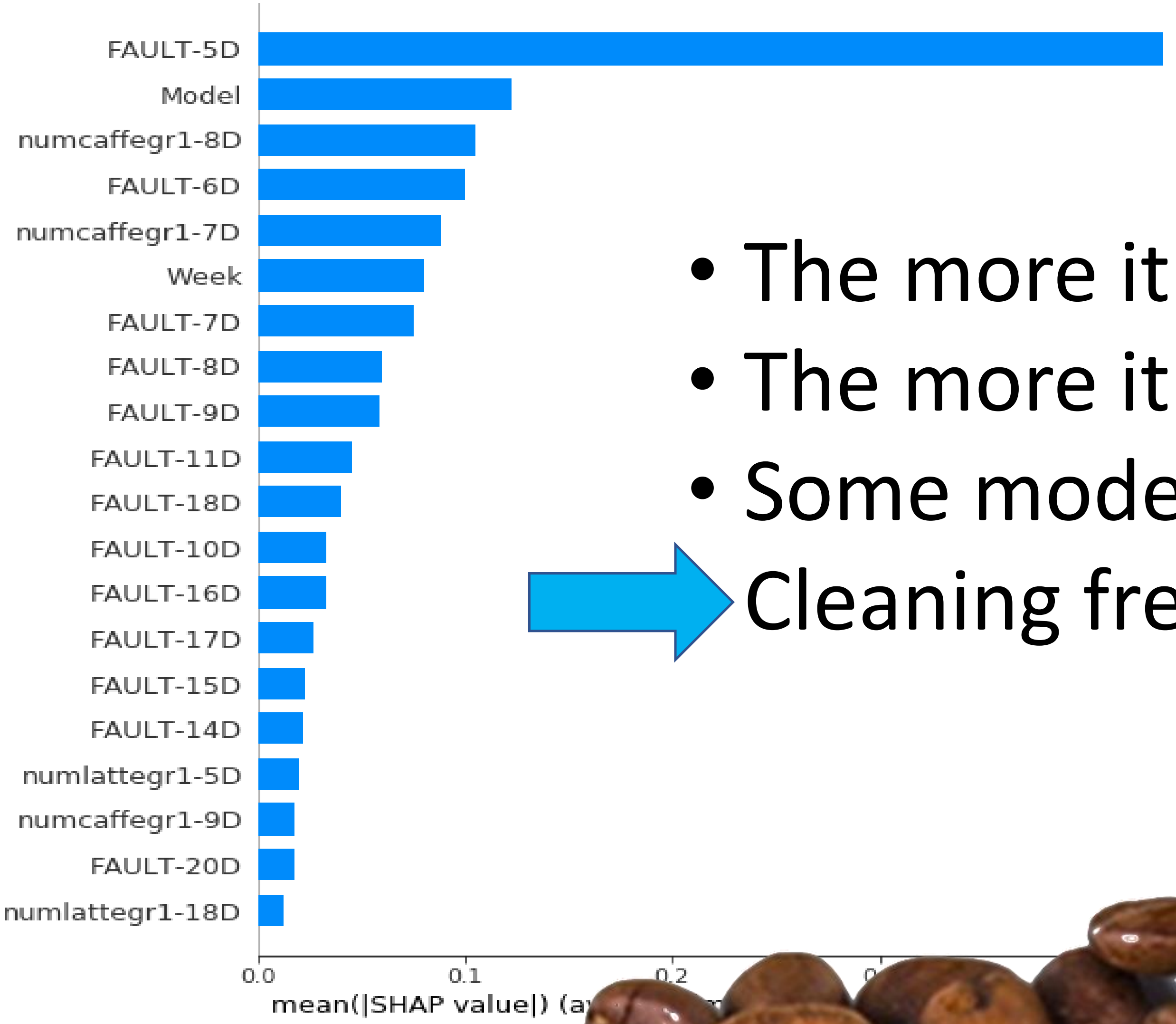


- SHAP (SHapley Additive exPlanations) is a unified approach to explain the output of any machine learning model.
- SHAP connects game theory with local explanations, uniting several previous methods

Interpretation: SHAP (LGBM model)

DataScienceSeed

[Shap Documentation]



- The more it failed, the sooner it will fail
- The more it is used, the sooner it will fail
- Some models are just loosers

Cleaning frequency is not even in the list

DataScienceSeed

Thank you!

Questions?