OPEN DATA SCIENCE CONFERENCE

#ODSC

@ODSC

Boston | May 1 - 4 2018

Experimental Reproducibility in Data Science with Sacred

Us

Jason: Recommender systems / Personalization @ HBC ; Washed up Kaggler

Karthik: Demand Prediction & CLTV @ HBC

HBC: we sell clothes and stuff

(348 years)

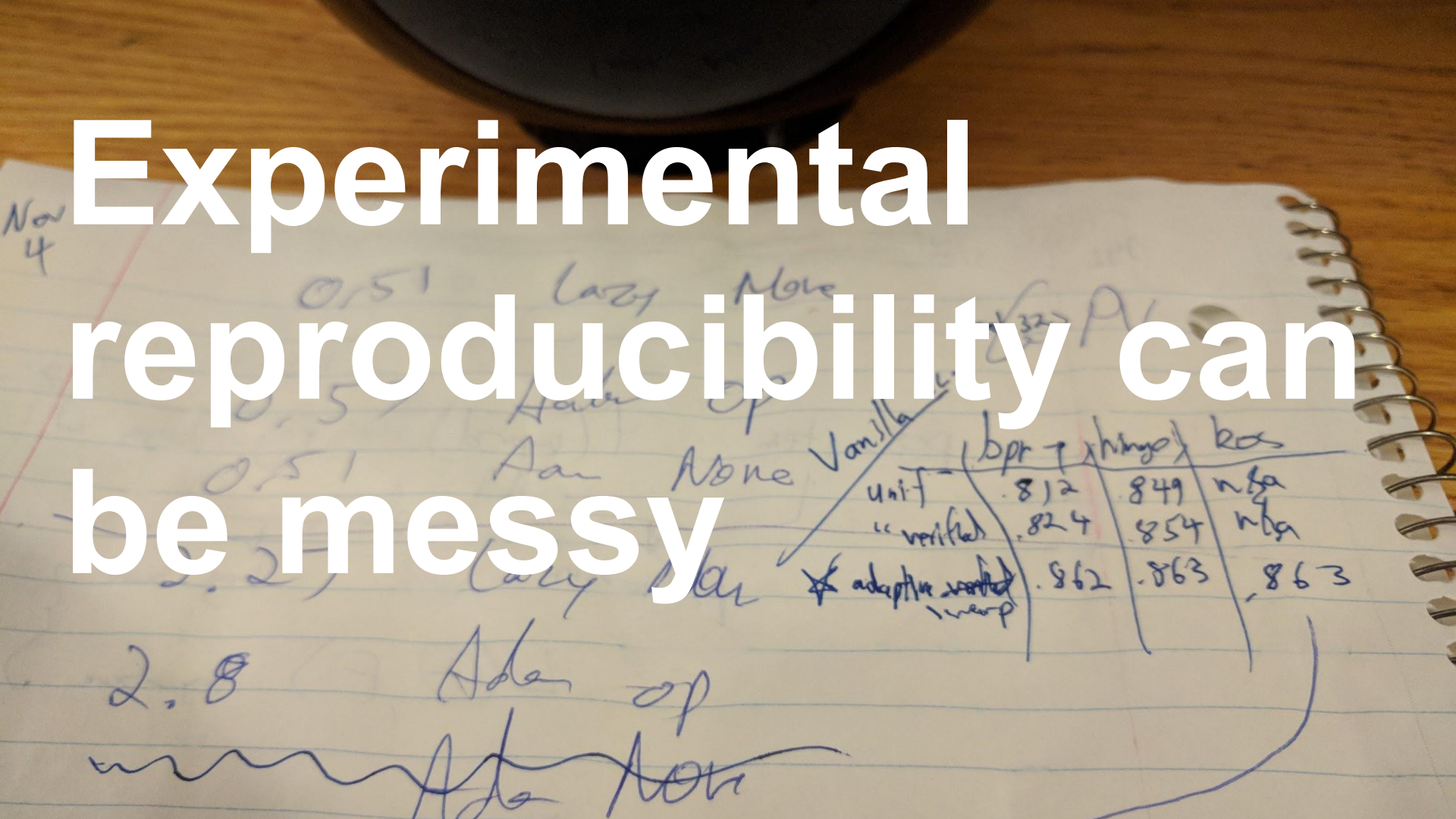
Materials

github.com/gilt/odsc-2018

Outline

1. **Experimental Reproducibility**
    a. **What is it?**
    b. **Current Trends**
2. Sacred - A framework for reproducibility
3. Basic Machine Learning with Sacred
4. Advanced use cases
5. Final Thoughts

Experimental reproducibility can be messy



r/MachineLearning

Discussion [D] What are you using these days for hyperparameter optimization?

(self.MachineLearning)
submitted 4 days ago by ballsandbutts

[–] Gusfoo 107 points 4 days ago

Angrily editing the file and forgetting why I did it immediately.

Facebook FBLearner Flow [1]

FBLearner | Workflow Library | Models/Features | Help

Search for people, tasks, tools...

FBLearner

Launch New Run | Compare

Search by workflow, tag, owner, name, or ID

Advanced search

My Runs | My Test Runs | My Recurring Runs | All Runs | Custom +

ID Owner Workflow Name Progress Start Tags Log Loss AUC

Parameter Sweep Example

Learning Rate: 0.1

ID	Owner	Workflow	Name	Progress	Start Time	Tags	Log Loss	AUC
1047165	Mahaveer				9/9, 9:06pm	london-demo	-	-
1047298	Mahaveer Jain	Gradient Boosted Decision Tree Training	Learning Rate: 0.35		9/9, 9:19pm	-	0.00105	0.95524
1047297	Mahaveer Jain	Gradient Boosted Decision Tree Training	Learning Rate: 0.25		9/9, 9:19pm	-	0.00107	0.95776
1047296	Mahaveer Jain	Gradient Boosted Decision Tree Training	Learning Rate: 0.3		9/9, 9:19pm	-	0.00104	0.95719
1047295	Mahaveer Jain	Gradient Boosted Decision Tree Training			9/9, 9:19pm	-	0.00122	0.95871
1047294	Mahaveer Jain	Gradient Boosted Decision Tree Training	Learning Rate: 0.2		9/9, 9:19pm	-	0.00109	0.95796
1047293	Mahaveer Jain	Gradient Boosted Decision Tree						
1047292	Mahaveer Jain	Gradient Boosted Decision Tree						
1047291	Mahaveer Jain	Gradient Boosted Decision Tree						
1037778	Jason Briceno	Parameter Sweep Example						
950428	Li Zhang	Parameter Sweep Example						
900673	Jiawei Chen	Parameter Sweep Example						
832281	Giri Rajaram	Parameter Sweep Example						
832027	Giri Rajaram	Parameter Sweep Example						

Displaying results 1 - 8 out of 8 matches.

2017-06-02-12-35-47-065-UTC

DEPLOY | RETRAIN

DEPLOY

PERFORMANCE | MODEL VIS | FEATURES

Performance ModelVis Features

Features

Feature histogram & Partial dependence

feature_31

importance	0.2362
coeff	85.28
nulls	0
zeros	0
mean	2756
std	2.35e+5
p01	547.6
p99	2360

feature_36

importance	0.1774
coeff	0.449
nulls	0
zeros	0
mean	535.3
std	240.4
p01	147
p99	1224

feature_12

importance	0.1501
coeff	158
nulls	9.206e+4
zeros	9.045e+4
mean	2761
std	4.362e+5
p01	-10
p99	2434

feature_15

importance	0.05788
nulls	0
unique categories	7

Overview

All features by importance

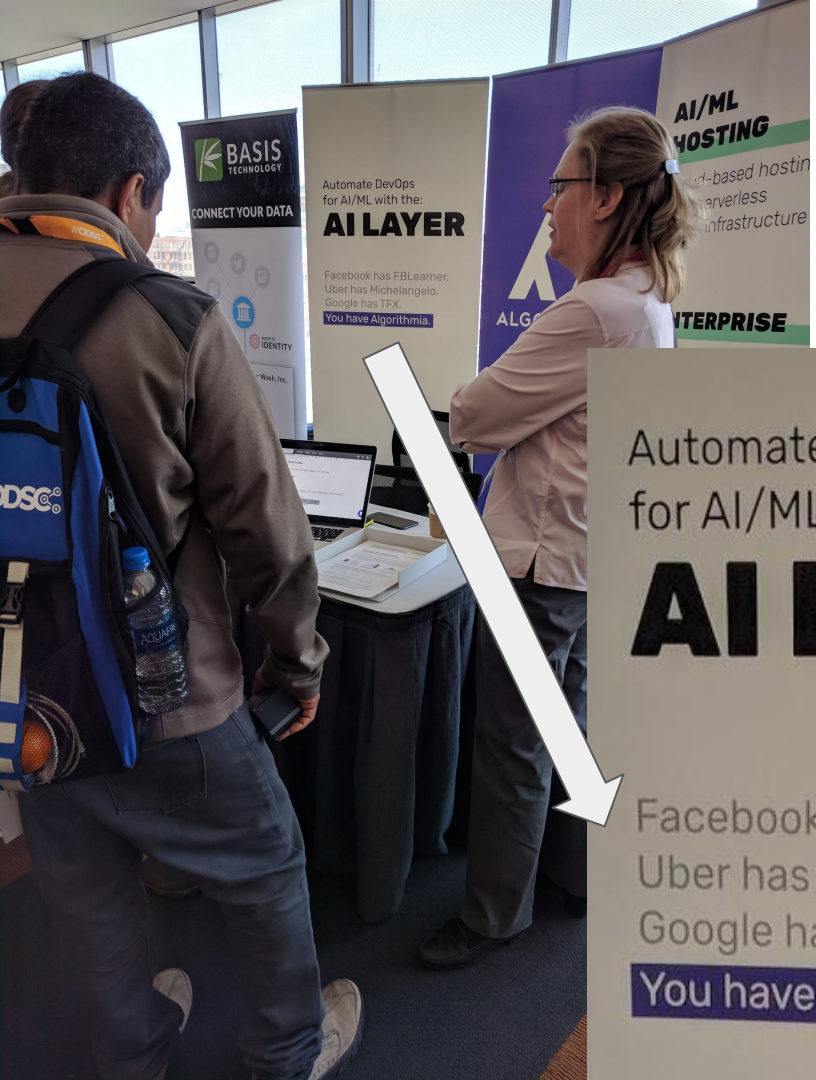
Feature interaction

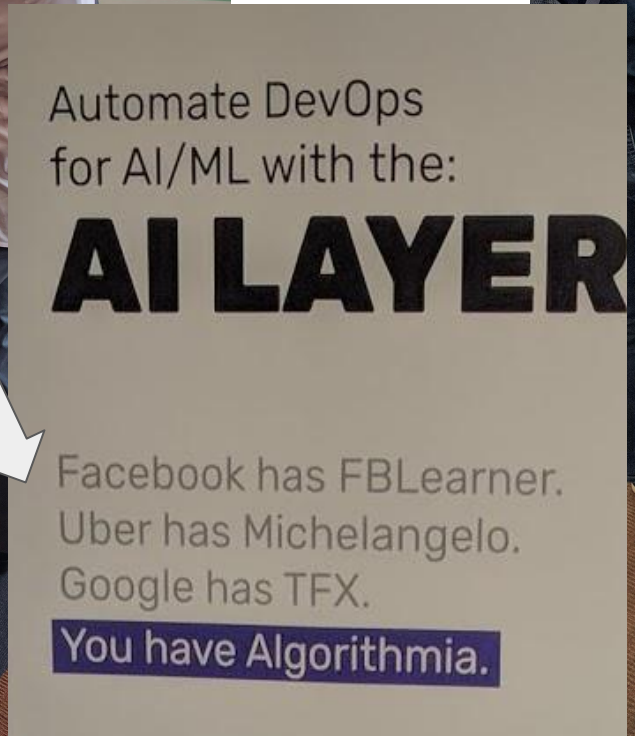
2-way partial dependence

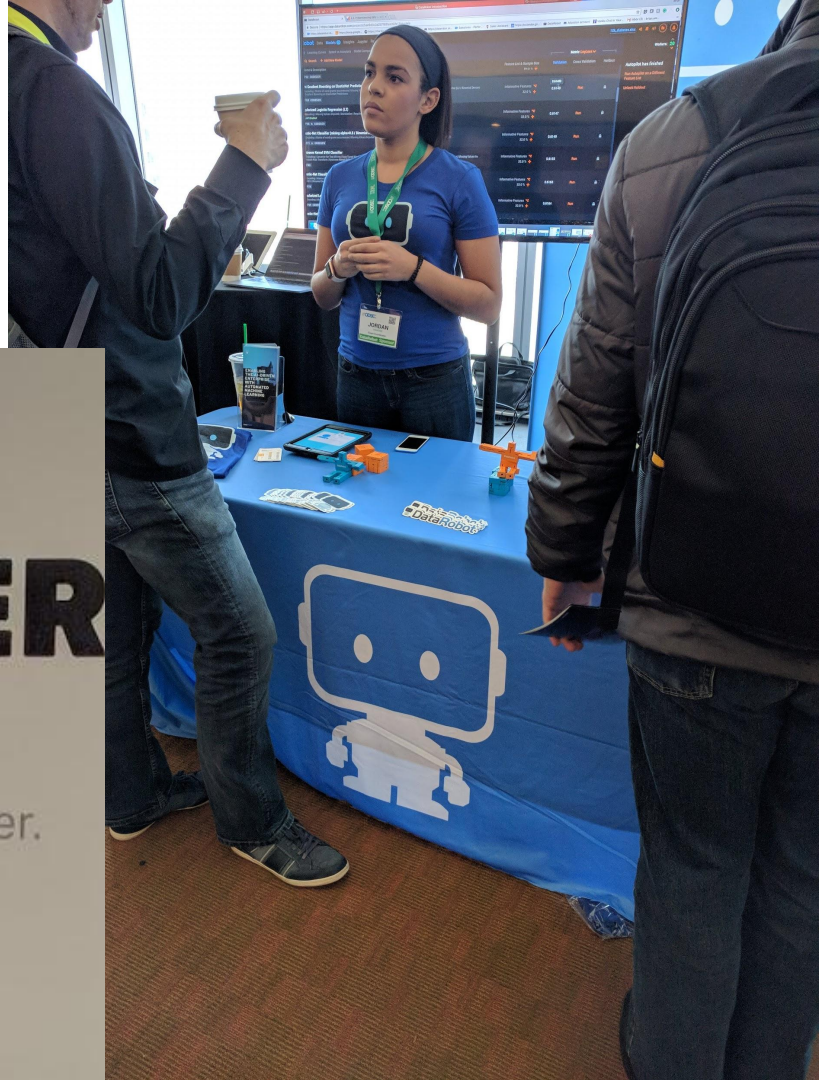
X Axis: feature_31

Y Axis: feature_36

Uber Michelangelo [2]


BASIS TECHNOLOGY
CONNECT YOUR DATA
Automate DevOps
for AI/ML with the:
AI LAYER
Facebook has FBLearner.
Uber has Michelangelo.
Google has TFX.
You have Algorithmia.
AI/ML
HOSTING


Automate DevOps
for AI/ML with the:
AI LAYER
Facebook has FBLearner.
Uber has Michelangelo.
Google has TFX.
You have Algorithmia.



Our final product today

127.0.0.1:5000/runs

Sacredboard 0.4.0

Filter runs:

config.? seed or .host.hostname == "string" or 123.4 Add filter

Show 10 entries

Experiments Run Overview

Legend: Running, Completed, Failed, Interrupted, Timeout, ? Probably dead, Queued Refresh

	Id	Experiment name	Command	Start time	Last activity	Hostname	Result
+	283	titanic	run	19:33:26 04/20/18	19:46:16 04/20/18	ML9301.local	
+	282	titanic	run	19:33:26 04/20/18	19:33:26 04/20/18	ML9301.local	0.6681614349775785
+	281	? titanic	run	19:32:56 04/20/18	19:33:06 04/20/18	ML9301.local	
+	280	titanic	run	19:32:56 04/20/18	19:32:56 04/20/18	ML9301.local	0.7085201793721974
+	279	titanic	run	19:32:55 04/20/18	19:32:56 04/20/18	ML9301.local	0.7130044843049327
+	278	titanic	run	19:09:25 04/20/18	19:09:25 04/20/18	ML9301.local	0.43946188340807174
+	277	titanic	run	19:09:25 04/20/18	19:09:25 04/20/18	ML9301.local	0.4484304932735426
+	276	titanic	run	19:09:25 04/20/18	19:09:25 04/20/18	ML9301.local	0.4484304932735426
+	275	titanic	run	19:09:25 04/20/18	19:09:25 04/20/18	ML9301.local	0.43946188340807174
+	274	titanic	run	19:09:25 04/20/18	19:09:25 04/20/18	ML9301.local	0.43946188340807174

Showing 1 to 10 of 283 entries

Previous 1 2 3 4 5 ... 29 Next

Outline

1. Experimental Reproducibility
2. **Sacred - A framework for reproducibility**
   a. **Overview**
   b. **Decorators**
3. Basic Machine Learning with Sacred
4. Advanced use cases
5. Final Thoughts

Sacred [3]

Experiments

Ingredients

Observers

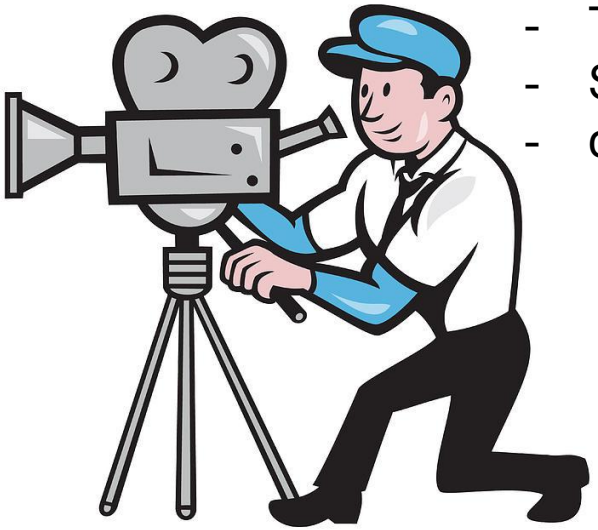
Sacred [3]

Ingredients

- any configurable duplicated code
- Datasets
- Other experiments

Experiment

- your main configurable task
- train/validating a model



Observer

- Files
- Mongo
- TinyDB
- SQL DB
- custom

Refresher on Decorators

```python
def my_decorator(some_function):
    def wrapper():
        print("About to run our function...")
        some_function()
        print("Done!")
    return wrapper
```

```python
def run_something():
    print("ran something")

decorated_function = my_decorator(run_something)

decorated_function()
```

```
About to run our function...
ran something
Done!
```

With syntactic sugar:

```python
@my_decorator
def sweet_decorated_function():
    print("ran something else")

sweet_decorated_function()
```

```
About to run our function...
ran something else
Done!
```

Outline

1. Experimental Reproducibility
2. Sacred - A framework for reproducibility
3. Basic Machine Learning with Sacred
    a. **Predicting Titanic Survivorship with Sacred**
    b. **What gets saved in Mongo?**
    c. **Sacredboard - a front-end for Sacred**
    d. **More examples**
4. Advanced use cases
5. Final Thoughts

kaggle

https://www.kaggle.com/c/titanic

Case: Kaggle Titanic

- starting a project from scratch with sacred in mind

Titanic Overview:

- 1311 data points
  - Train: 891
  - Test: 419
- 11 features
- Target = **Survived** = Binary Classification

Case: Kaggle Titanic

Data Dictionary

Variable	Definition	Key
survival	Survival	0 = No, 1 = Yes
pclass	Ticket class	1 = 1st, 2 = 2nd, 3 = 3rd
sex	Sex	
Age	Age in years	
sibsp	# of siblings / spouses aboard the Titanic	
parch	# of parents / children aboard the Titanic	
ticket	Ticket number	
fare	Passenger fare	
cabin	Cabin number	
embarked	Port of Embarkation	C = Cherbourg, Q = Queenstown, S = Southampton

Barebones Example: Load Data

```
1   import pandas as pd
2   from sklearn.model_selection import train_test_split
3   from sacred import Ingredient
4
5
6   train_data_ingredient = Ingredient('train_dataset')
7
8
9   @train_data_ingredient.config
10  def cfg():
11      filename = 'data/train.csv'
12      target = 'Survived'
13      split_size = .75
14
15
16  @train_data_ingredient.capture
17  def load_data(filename, target, split_size):
18      data = pd.read_csv(filename)
19      features = [i for i in data.columns if i != target]
20      return train_test_split(data[features], data[target],
21                              train_size=split_size)
```

Import

Create an Ingredient

Config: Define and annotate. There can be multiple configs for an ingredient.

Capture: Has access to params defined in config to load data

Barebones Example: Preprocess

```python
from sacred import Ingredient


preprocess_ingredient = Ingredient('preprocess')


@preprocess_ingredient.config
def cfg():
    features = ['Fare', 'SibSp', 'Parch']


@preprocess_ingredient.named_config
def variant_simple():
    features = ['Fare', 'SibSp']


@preprocess_ingredient.capture
def preprocess_data(df, features):
    return df[features]
```

Config: An example of multiple configs.

Capture: This takes a parameter, *df*, that is not defined in the config.

Barebones Example: Experiment

```python
from sklearn.linear_model import LogisticRegression
from sacred import Experiment
from ingredients.data import train_data_ingredient, load_data
from ingredients.preproc import preprocess_ingredient, preprocess_data


ex = Experiment('titanic',
                ingredients=[train_data_ingredient, preprocess_ingredient])


@ex.config
def cfg():
    penalty = 'l2'
    fit_intercept = False



@ex.automain
def run(penalty, fit_intercept):
    X_train, X_val, Y_train, Y_val = load_data()

    clf_lg = LogisticRegression(penalty=penalty, fit_intercept=fit_intercept)
    clf_lg.fit(preprocess_data(X_train), Y_train)

    return clf_lg.score(preprocess_data(X_val), Y_val)
```

Import: Data/Preprocessing Ingredients

Define Experiment: Specify name, and ingredients

Config: Experiment is a kind of Ingredient. It also has a config.

[Auto]main: Does capturing AND indicates this function is the main method of experiment.

Barebones Example: Print Config

```
python experiments/model_accuracy.py print_config

INFO - titanic - Running command 'print_config'
INFO - titanic - Started
Configuration (modified, added, typechanged, doc):
  fit_intercept = False
  penalty = 'l2'
  seed = 460479209                    # the random seed for this experiment
  preprocess:
    features = ['Fare', 'SibSp', 'Parch']
  train_dataset:
    filename = 'data/train.csv'
    split_size = 0.75
    target = 'Survived'
INFO - titanic - Completed after 0:00:00
```

Barebones Example: Modify Config

```
python experiments/model_accuracy.py print_config with
fit_intercept=True preprocess.features="['Fare','Parch']"
```

```
INFO - titanic - Running command 'print_config'
INFO - titanic - Started
Configuration (modified, added, typechanged, doc):
  fit_intercept = True
  penalty = 'l2'
  seed = 592460980                    # the random seed for this experiment
  preprocess:
    features = ['Fare', 'Parch']
  train_dataset:
    filename = 'data/train.csv'
    split_size = 0.75
    target = 'Survived'
INFO - titanic - Completed after 0:00:00
```

Start your [Mongo] Engines!

```
$ mongod

$ python experiments/model_accuracy.py -m sacred

$ python experiments/model_accuracy.py -m sacred with
seed=10

$ python experiments/model_accuracy.py -m sacred with
preprocessing.variant_simple seed=10
```

What’s been saved? - how to view runs on mongo

```
$ python experiments/model_accuracy.py -m sacred
```

```
$ mongo
mongo> use sacred
mongo> db.runs.find().pretty()
```

...

What’s been saved? - view runs on Sacredboard

```
$ sacredboard -m sacred
```

(Note: port 5000 by default)

Front End: Sacredboard

127.0.0.1:5000/runs

Sacredboard 0.4.0

Filter runs:

config.? seed or .host.hostname == "string" or 123.4 Add filter

Show 10 entries

Experiments Run Overview

Legend: Running, Completed, Failed, Interrupted, Timeout, ? Probably dead, Queued Refresh

	Id	Experiment name	Command	Start time	Last activity	Hostname	Result
+	283	titanic	run	19:33:26 04/20/18	19:46:16 04/20/18	ML9301.local	
+	282	titanic	run	19:33:26 04/20/18	19:33:26 04/20/18	ML9301.local	0.6681614349775785
+	281	? titanic	run	19:32:56 04/20/18	19:33:06 04/20/18	ML9301.local	
+	280	titanic	run	19:32:56 04/20/18	19:32:56 04/20/18	ML9301.local	0.7085201793721974
+	279	titanic	run	19:32:55 04/20/18	19:32:56 04/20/18	ML9301.local	0.7130044843049327
+	278	titanic	run	19:09:25 04/20/18	19:09:25 04/20/18	ML9301.local	0.43946188340807174
+	277	titanic	run	19:09:25 04/20/18	19:09:25 04/20/18	ML9301.local	0.4484304932735426
+	276	titanic	run	19:09:25 04/20/18	19:09:25 04/20/18	ML9301.local	0.4484304932735426
+	275	titanic	run	19:09:25 04/20/18	19:09:25 04/20/18	ML9301.local	0.43946188340807174
+	274	titanic	run	19:09:25 04/20/18	19:09:25 04/20/18	ML9301.local	0.43946188340807174

Showing 1 to 10 of 283 entries

Previous 1 2 3 4 5 ... 29 Next

What’s been saved? - identification


```
{
    "_id" : 1,
    "experiment" : {
        "name" : "titanic",
        "base_dir" : "/home/jason/Documents/gilt/odsc-2018/experiments",
    ...
    "command" : "run",
    ...

    "mainfile" : "model_accuracy.py"

                                    ...
```

...

What’s been saved? - version control

```
"sources" : [
  [
    "model_accuracy.py",
    ObjectId("5ada1ee4ae1a130f14e5d45a")
  ],...
],
```

```
"dependencies" : [
  "ingredients==<unknown>",
  "numpy==1.14.1",
  "pandas==0.22.0",
  "sacred==0.7.1",
  "sklearn==0.19.0"
],
```

```
"repositories" : [
  {
    "url" : "git@github.com:gilt/odsc-2018.git",
    "commit" : "c5174987bb0fbde842e2838284dee8935841ff9c",
    "dirty" : true
  },...
```

...

Experiment

-	293	titanic	run	19:48:20 04/20/18	19:48:20 04/20/18	ML9301.local	0.43946188340807174

Details for: titanic (id: 293)

- Config
- Run info
- Captured output
- **Experiment**
- Meta Info
- Tensorflow logs
- Metrics plots

DELETE

Experiment

base_dir	/web/odsc-2018/experiments
dependencies	[numpy==1.14.2, pandas==0.22.0, sacred==0.7.2, scikit-learn==0.19.1]
mainfile	model_accuracy.py
name	titanic
repositories	[{...}, {...}, {...}]
sources	[[model_accuracy.py, {...}], [../ingredients/data.py, {...}], [../ingredients/preproc.py, {...}]]

...

What’s been saved? - host info

```
"host" : {
  "hostname" : "box",
  "os" : [
    "Linux",
    "Linux-4.13.0-38-generic-x86_64-with-debian-stretch-sid"
  ],
  "python_version" : "3.6.2",
  ...
  },
  "ENV" : {
    ...
  }
},
```

...

...

What’s been saved? - host hardware info


```
"cpu" : "Intel(R) Core(TM) i7-6800K CPU @ 3.40GHz",
  "gpus" : {
    "gpus" : [
      {
        "model" : "GeForce GTX 1080 Ti",
        "total_memory" : 11171,
        "persistence_mode" : false
      },
      {
        "model" : "GeForce GTX 1080 Ti",
        "total_memory" : 11172,
        "persistence_mode" : false
      }
    ],
    "driver_version" : "384.130"
```


...

...

What’s been saved? - experiment config


```
"config" : {
  "fit_intercept" : false,
  "penalty" : "l2",
  "preprocess" : {
    "features" : [
      "Fare",
      "SibSp",
      "Parch"
    ]
  },
  "seed" : 509315819,
  "train_dataset" : {
    "filename" : "data/train.csv",
    "split_size" : 0.75,
    "target" : "Survived"
  }
},
```


...

Config

	Id	Experiment name	Command	Start time	Last activity	Hostname	Result
-	293	titanic	run	19:48:20 04/20/18	19:48:20 04/20/18	ML9301.local	0.43946188340807174

Config
Run info
Captured output
Experiment
Meta Info
Tensorflow logs
Metrics plots
DELETE

Details for: titanic (id: 293)

Run configuration

fit_intercept	false
penalty	l2
preprocess	{...}
features	[Fare, SibSp, Parch]
seed	0
titanic	{...}
C	null
fit_intercept	false
penalty	l1
train_dataset	{...}
filename	data/train.csv

...

What’s been saved? - meta

```
"meta" : {
        "command" : "run",
        "options" : {
            "--enforce_clean" : false,
            "--force" : false,
            "--unobserved" : false,
            "--name" : null,
            "--debug" : false,
            "--priority" : null,
            "--tiny_db" : null,
            "--capture" : null,
            "--loglevel" : null,
            "--queue" : false,
            "--beat_interval" : null,
            ...
            ...
            "--print_config" : false,
            "--sql" : null,
            "--pdb" : false,
            "--mongo_db" : "sacred",
            "--comment" : null,
            "--file_storage" : null,
            "--help" : false,
            "with" : false,
            "UPDATE" : [ ],
            "help" : false,
            "COMMAND" : null
        }
```

...

...

What’s been saved? - files

```
"sources" : [
  [
    "model_accuracy.py",
    ObjectId("5ada1ee4ae1a130f14e5d45a")
  ],...
],


    "resources" : [ ],

    "artifacts" : [ ],
```

...

...

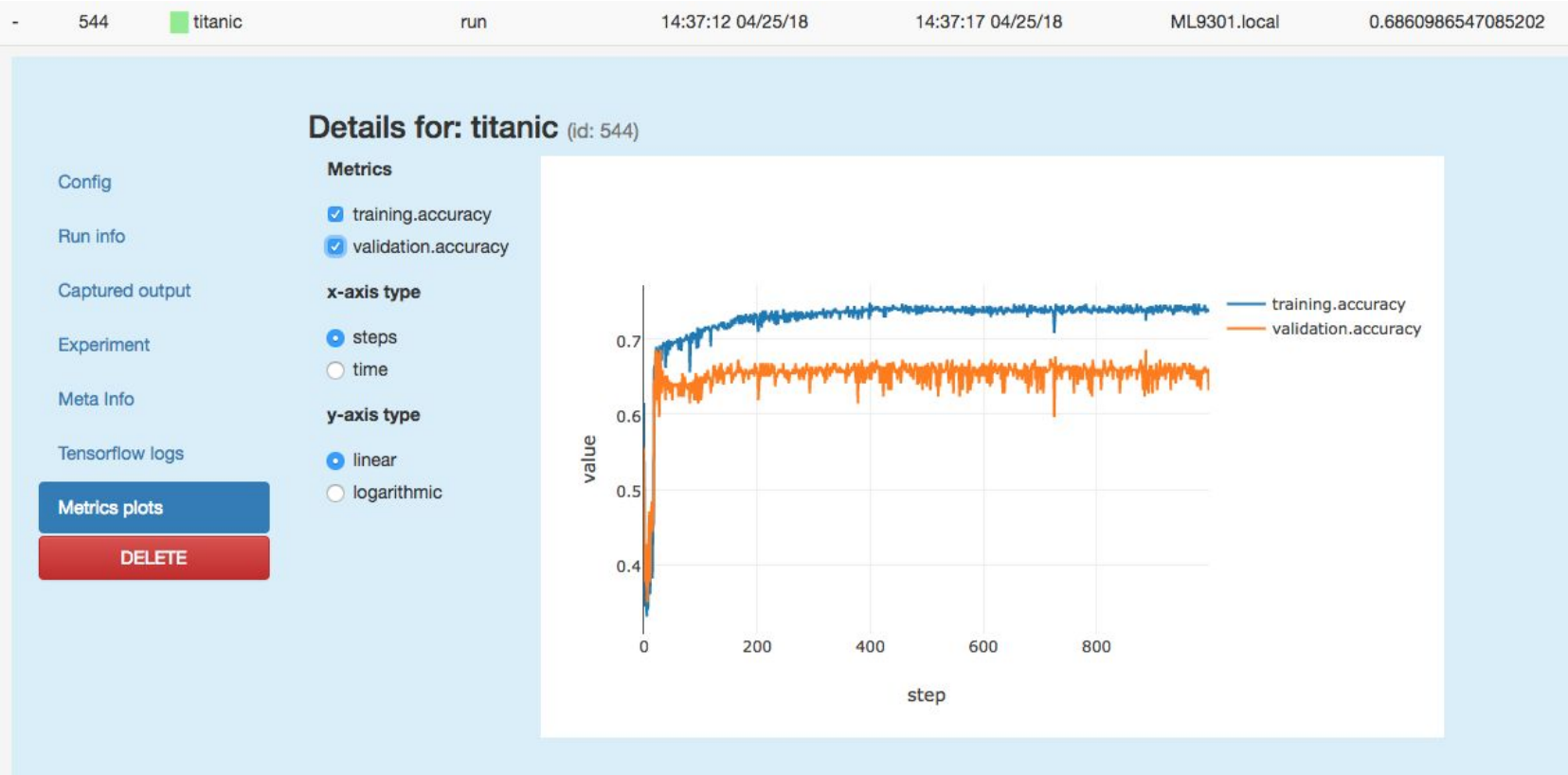
What’s been saved? - status

```
"status" : "COMPLETED",

"start_time" : ISODate("2018-04-20T17:09:56.329Z"),
"heartbeat" : ISODate("2018-04-20T17:09:56.380Z"),
"stop_time" : ISODate("2018-04-20T17:09:56.380Z")

"result" : 0.47533632286995514,
```

Metrics Plots



Captured Output

	Id	Experiment name	Command	Start time	Last activity	Hostname	Result
-	293	titanic	run	19:48:20 04/20/18	19:48:20 04/20/18	ML9301.local	0.43946188340807174

Config

Run info

Captured output

Experiment

Meta Info

Tensorflow logs

Metrics plots

DELETE

Details for: titanic (id: 293)

Captured output from the experiment

```
INFO - titanic - Running command 'run'
INFO - titanic - Started run with ID "293"
/web/odsc-2018/sacred-demo/lib/python3.6/site-packages/sklearn/model_selection/_split.py:2026: FutureWarning: From version 0.21, test_size will always compl
  FutureWarning)
INFO - titanic - Result: 0.43946188340807174
INFO - titanic - Completed after 0:00:00
```

Try it out: Track Metrics

```
$ python experiments/track_metrics.py -m sacred with seed=0

$ python experiments/track_metrics.py -m sacred with n_epochs=10 seed=0
```

File Observer

```
$ python experiments/model_accuracy.py -F folder_for_stuff

$ tree folder_for_stuff/
folder_for_stuff/
├── 1
│   ├── config.json
│   ├── cout.txt
│   └── run.json
└── _sources
    ├── data_8b1f7f9747e9f49a016f621b4dd38787.py
    ├── model_accuracy_464cab39647d446c580787735a5ac433.py
    └── preproc_7bfe4e06f5f9beafd9be50791aca9ccf.py

2 directories, 6 files
```

More Experiments! **Config**

experiments/model_accuracy.py	**experiments/model_accuracy2.py**	**experiments/model_accuracy3.py**

ex = Experiment('titanic', ingredients=...)
 	
@ex.config
def cfg():
penalty = 'l2'
fit_intercept = False
# folds = 10 	
	ex = Experiment('titanic', ingredients=...)
	
	@ex.config
	def cfg():
	...
	penalty = 'l2'
	fit_intercept = False
	C = 1.0
	save_probs = True
	save_submission = False
		ex = Experiment('titanic', ingredients=...)
		
		@ex.config
		def cfg():
		model_type = "lr" # LogisticRegression
		
		# LR Params
		lr_penalty = 'l2'
		lr_fit_intercept = False
		lr_c = 1.0
		
		# RF Params
		rf_n_estimators = 10
		rf_max_depth = None
		rf_min_samples_split = 2
		
		save_probs = True
		save_submission = False

1 model	1 model + save flags	2 models

More Experiments! **Variants**

experiments/model_accuracy.py	**experiments/model_accuracy2.py**	**experiments/model_accuracy3.py**

ex = Experiment('titanic', ingredients=...)
 	
@ex.config
def cfg():
...
 	
#
	ex = Experiment('titanic', ingredients=...)
	
	@ex.config
	def cfg():
	...
	
	@ex.named_config
	def variant_rand_params():
	penalty = np.random.choice(['l1', 'l2'])
	fit_intercept = np.random.randint(2,dtype=bool)
	C = np.exp(np.random.randn() * 5)
		ex = Experiment('titanic', ingredients=...)
		
		@ex.config
		def cfg():
		…
		
		@ex.named_config
		def variant_rand_params():
		
		model_type = np.random.choice(['lr', 'rf'])
		
		lr_penalty = np.random.choice(['l1', 'l2'])
		lr_fit_intercept = np.random.randint(2,dtype=bool)
		lr_c = np.exp(np.random.randn() * 5)
		
		rf_n_estimators = np.random.choice([10, ...])
		rf_max_depth = np.random.choice([None, 5, 10])
		rf_min_samples_split = np.random.choice([2, ...])

No variant

Rand Variant
For 1 model

Rand Variant
For 2 models

More Experiments! **Run**

experiments/model_accuracy.py	**experiments/model_accuracy2.py**	**experiments/model_accuracy3.py**
ex = Experiment('titanic', ingredients=...) @ex.config def cfg(): ... @ex.automain def run(penalty, fit_intercept): ...	ex = Experiment('titanic', ingredients=...) @ex.config def cfg(): ... @ex.named_config def variant_rand_params(): ... @ex.automain def run(penalty, fit_intercept, C, save_probs, save_submission): ...	ex = Experiment('titanic', ingredients=...) @ex.config def cfg(): ... @ex.named_config def variant_rand_params(): … @ex.automain def run(lr_penalty, lr_fit_intercept, lr_c, rf_n_estimators, rf_max_depth, rf_min_samples_split, model_type, ...): ...

The more we track, the longer the list of parameters for the *run* function

Try it out!

```
$ mongod

$ python experiments/model_accuracy.py -m sacred
```

```
$ python experiments/model_accuracy2.py -m sacred with
dataset.variant_presplit seed=10

$ python experiments/model_accuracy3.py -m sacred with
dataset.variant_presplit seed=10 model_type='rf'
```

Dataset ingredient from `ingredients/data2.py`

Outline

1. Experimental Reproducibility
2. Sacred - A framework for reproducibility
3. Basic Machine Learning with Sacred
4. Advanced use cases
    a. **Hyperparameter Optimization**
    b. **Blending**
    c. **Recommender Systems**
5. Final Thoughts

Hyper Parameter Optimization - What is it?

Model	**Model Parameters**	**Hyperparameters**
Logistic Regression	- Coefficients of features/intercept	- L1 or L2 regularization - fit_intercept - C
Random Forest	- Variables and values to split on	- # of Trees - Depth - Split Criterion - ...
Neural Network	- weights/biases	- Activation function - Learning Rate - # Hidden Layers - ...

How to track every HPO run?

1. Define Hyperparameter Search Space
   a. Hyperopt will use this to generate configs
2. Attach an [Mongo] Observer
   a. To store results
3. Run experiment with configs generated via hyperopt
   a. Running in python (as opposed to running from the command line)

Hyperopt + Sacred: Define Search Space

```python
# from hpo/hyperopt_configs.py
from hyperopt import hp
# HPO params for experiments/model_accuracy.py
vanilla_exp_space = {
  ...
  # Preprocess Ingredient: preprocess
 "preprocess": {
     "features": hp.choice("features", [['Fare', 'SibSp'],
                                        ['Fare', 'SibSp', 'Parch']]),
  },
  # Experiment: titanic
 "fit_intercept": hp.choice('fit_intercept', [True, False]),
 "penalty": hp.choice('penalty', ["l1", "l2"]),
 "C": hp.loguniform('C', -5, 5)

}
```

```python
# from ingredients/data2.py
data_ingredient = Ingredient("dataset")
....
```

```python
# from ingredients/preproc.py
preprocess_ingredient = Ingredient("preprocess")
...
```

```python
# from experiments/model_accuracy.py
ex = Experiment("titanic",
                ingredients=[data_ingredient, preprocess_ingredient])
...
```

Hyperopt + Sacred: Attach Mongo Observer

```
...
from hyperopt import fmin, tpe, hp, Trials
class HyperoptHPO(object):
  ...
  def __init__(self, base_experiment, command_line_args, param_space):

    self.base_experiment = base_experiment
    self.mongo_url = command_line_args.mongo_db_address
    self.mongo_db = command_line_args.mongo_db_name
    ...
    self.base_experiment.observers.append(
                    MongoObserver.create(url=self.mongo_url, db_name=self.mongo_db))
```

Titanic Experiment

Initialize Sacred Observer

Hyperopt + Sacred: Update Configs

```
class HyperoptHPO(object):
  ...
  def __init__(self, base_experiment, ...):
    # initialize experiment, mongo observer, runs, search space
    ...

  def objective(self, experiment_args):
    self.experiment_config = experiment_args
    run_obj = self.base_experiment.run(config_updates=self.experiment_config)

    return - run_obj.result
```

Runs sacred titanic experiment with given config. *This will send the value of that run to Sacred*

Hyperopt + Sacred: Put it all together in fmin

```
class HyperoptHPO(object):
  ...
  def __init__(self, base_experiment, ...):
      ...
  def objective(self, experiment_args):
      ...
  def run_hyperopt(self):
     ...
     trials = Trials()
     # main hyperopt fmin function
     optimal_run = fmin(
        self.objective,
        self.param_space,
        algo=tpe.suggest,
        max_evals=self.num_runs,
        trials=trials)
     return trials, optimal_run
```

Sending result to Sacred in here

Hyperopt + Sacred: Run

```
python hpo/hyperopt_hpo.py --num-runs 3 --experiment-file-name model_accuracy3
```

```
def gather_experiments_and_configs(experiment_file_name):
    from experiments.model_accuracy import ex as vanilla
    from experiments.model_accuracy2 import ex as model_params
    from experiments.model_accuracy3 import ex as multiple_models

    from hpo.hyperopt_hpo_configs import vanilla_exp_space, stage0_space,
stage0_space_multiple_models

    exp_and_configs = {
        'model_accuracy': [vanilla, vanilla_exp_space],
        'model_accuracy2': [model_params, stage0_space],
        'model_accuracy3': [multiple_models, stage0_space_multiple_models],
    }
    return exp_and_configs[experiment_file_name][0],exp_and_configs[experiment_file_name][1]
```

Import Experiments

Import search spaces

Define dict to get experiment, search space based on command line arg

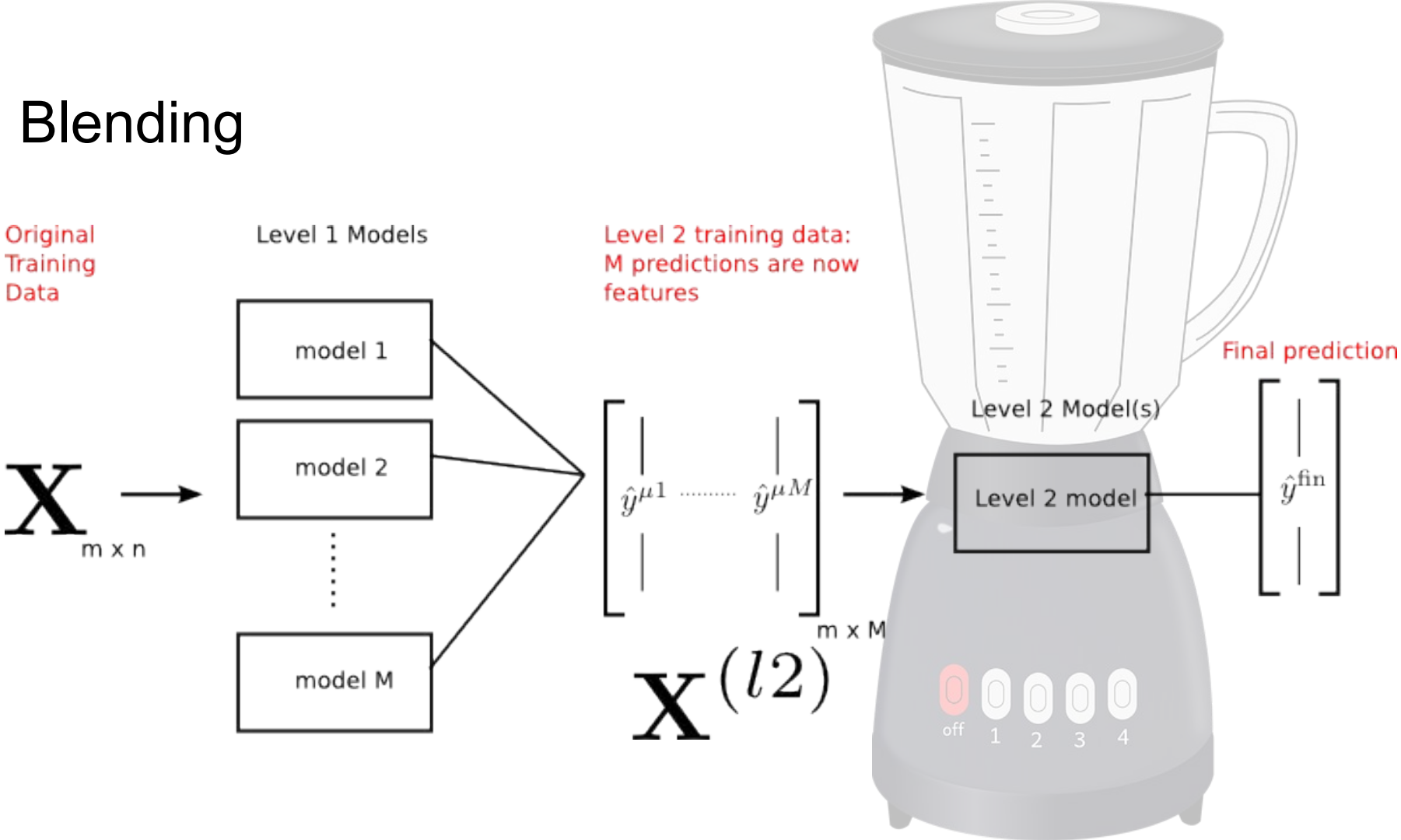
Try it out

```
$ python hpo/hyperopt_hpo.py --num-runs 10 --experiment-file-name model_accuracy

$ python hpo/hyperopt_hpo.py --num-runs 10 --experiment-file-name model_accuracy2

$ python hpo/hyperopt_hpo.py --num-runs 10 --experiment-file-name model_accuracy3
```

Blending


Original Training Data
Level 1 Models
Level 2 training data: M predictions are now features
Final prediction
model 1
model 2
model M
$\mathbf{X}$
m x n
$\hat{y}^{\mu 1}$ ..... $\hat{y}^{\mu M}$
m x M
$\mathbf{X}^{(l2)}$
Level 2 Model(s)
Level 2 model
$\hat{y}^{\text{fin}}$
off
1
2
3
4

Blending - saving holdout predictions as artifacts

```
@ex.automain
def run(...,
        save_probs, save_submission):
    ...
        # Export prob predictions
        if save_probs:
            # Val
            val_prob_df = pd.DataFrame(
                pred_prob_val[:, 1],
                index=pd.Index(x_val.index, name='PassengerId'),
                columns=['pred_proba'])
            prob_df['Survived'] = y_val
            df_artifact(ex, val_prob_df, 'holdout_predictions')
     ...
            test_prob_df = ...
            df_artifact(ex, test_prob_df, 'test_predictions')
```

Blending - saving holdout predictions as artifacts

```
def df_artifact(ex, df, name=None):
    """Writes a DataFrame as an artifact (csv format)"""
    f_tmp = tempfile.NamedTemporaryFile(mode='w', delete=False)
    df.to_csv(f_tmp, header=True, index=True)
    f_tmp.close()
    ex.add_artifact(f_tmp.name, name=name)
    os.remove(f_tmp.name)
    return f_tmp.name
```

a little strange

oh?

Blending

```python
@data_ingredient.named_config
def blend():
    """Blend of predictions from our top 3 models
    Example:
        ```
        python experiments/model_accuracy2.py -m sacred with \
        variant_rand_params dataset.blend preprocess.variant_all \
        save_submission=True
        ```
    """
    path_train = None
    path_val = None
    path_test = None
    blended = True
```

Blending

```
@data_ingredient.capture
def load_data(path_train, path_val, path_test,
              index_col, target_col,
              split_size=None,
              blended=False, ):

    if blended:
        return gather_stage1_features(target_col)
```

Blending - pymongo, artifacts

```
import pandas as pd
import pymongo
from pymongo import MongoClient
import gridfs


def gather_stage1_features(target_col):
    """Blend of predictions from our top 3 models"""

    client = MongoClient('localhost', 27017)
    db = client['sacredblender']
    fs = gridfs.GridFS(db)
    collection = db.runs
```

Blending - pymongo, artifacts

```
 ...
query = collection.find() \
     .sort('result', pymongo.DESCENDING) \
     .limit(3)

 train_df_l = []
 test_df_l = []
 for doc in query:
     for df_l, name in [(train_df_l, 'holdout_predictions'),
                        (test_df_l, 'test_predictions')]:

         preds_obj_id = artifact_by_name(doc, name)
         df = pd.read_csv(fs.get(preds_obj_id)) \
             .set_index('PassengerId') \
             .rename({'pred_proba': f"pred_proba-{doc['_id']}"}, axis=1)
         df_l.append(df)
   ...
```

```
def artifact_by_name(doc, name):
    """Convenience fn to grab ObjectId
    by artifact name"""
    obj_id = [d for d in doc['artifacts']
              if d['name'] == name][0]['file_id']
    return obj_id
```

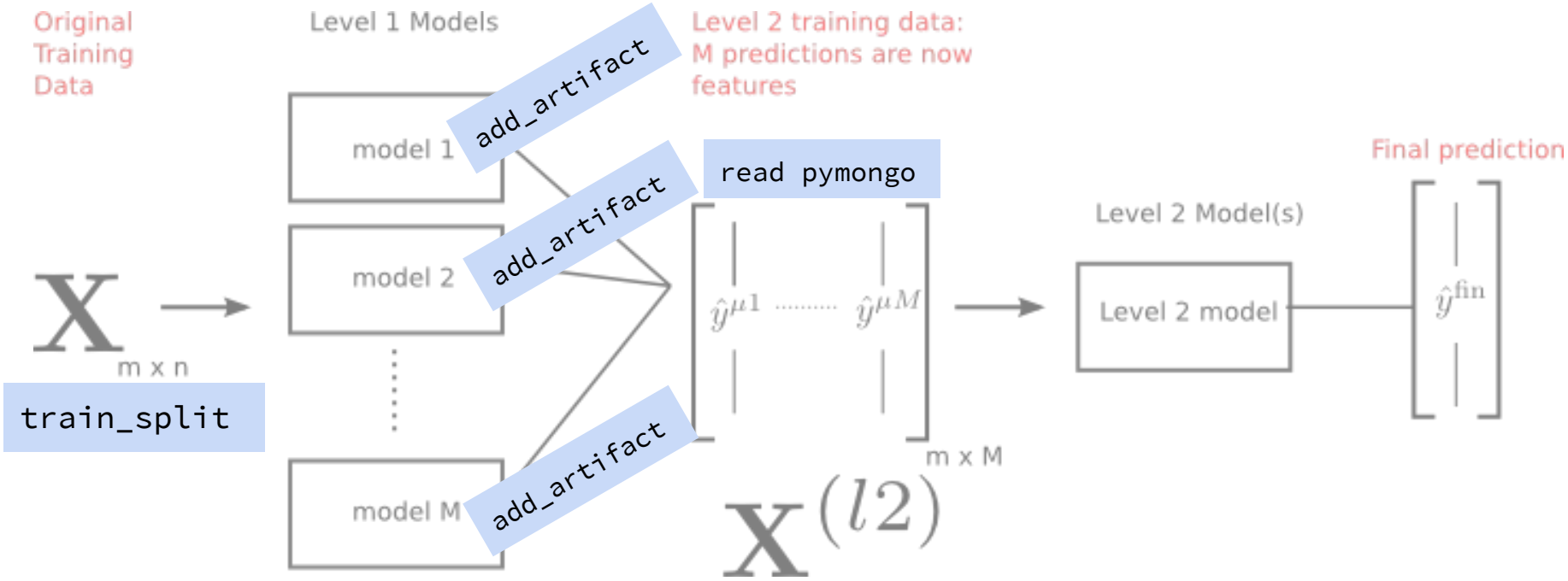
Blending - pymongo, artifacts

```
...
train_df = pd.concat(train_df_l, axis=1)
# De-dupe redundant target columns in training
train_df = train_df.loc[:, ~train_df.columns.duplicated()]
test_df = pd.concat(test_df_l, axis=1)

ret_d = {
    'train': (train_df.drop(target_col, axis=1), train_df[target_col]),
    'test': (test_df.drop(target_col, axis=1), None),
}

return ret_d
```

Blending - overview w.r.t. Sacred



Hyperopt Experiments

gather_stage1_features

experiment

Blending -- running the blender

```
$ python experiments/model_accuracy2.py -m sacredblender \
  with variant_rand_params dataset.variant_presplit
```

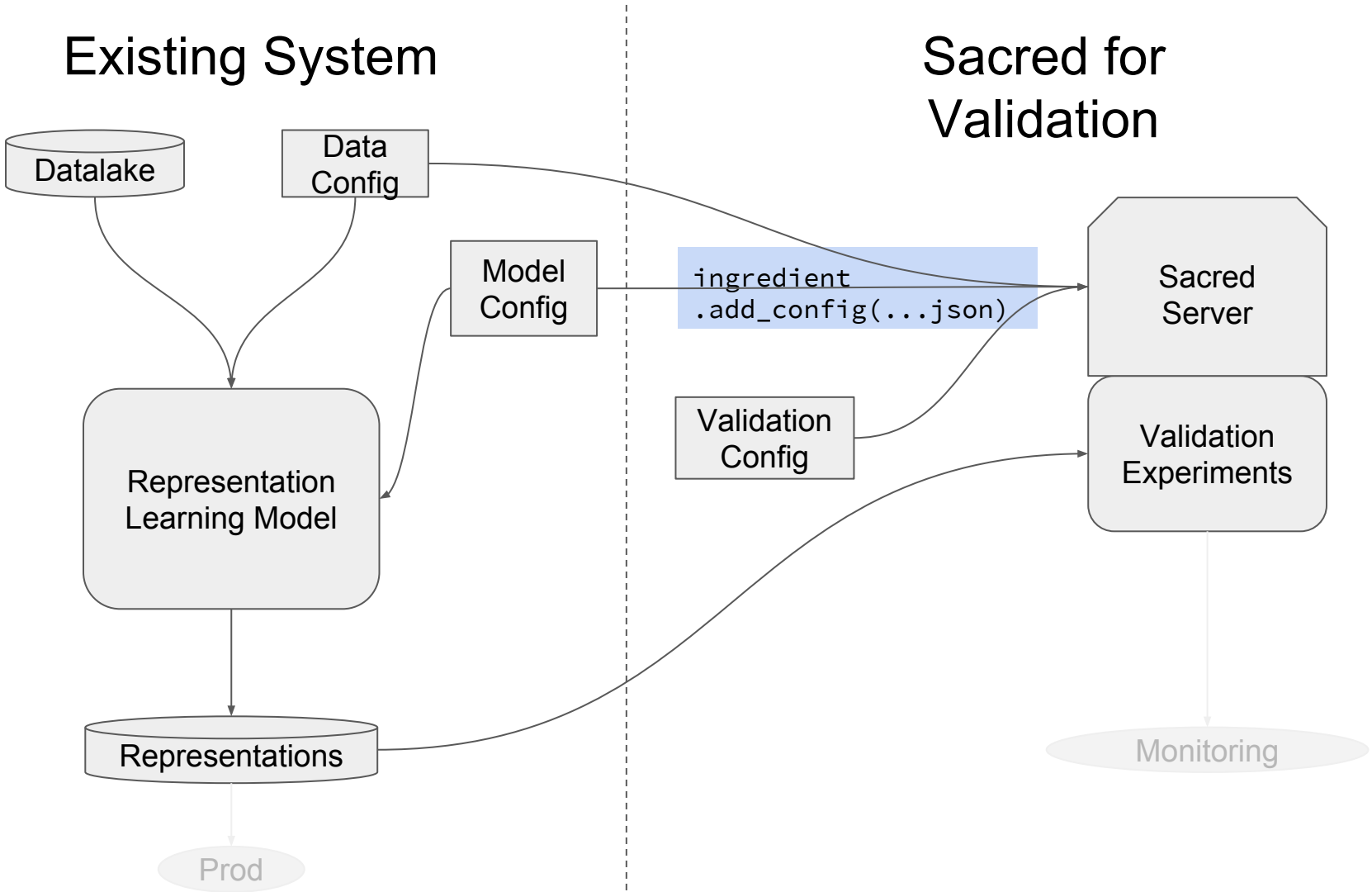
Create lvl 1 predictions

```
$ python experiments/model_accuracy2.py -m sacred \
  with dataset.blend preprocess.variant_all
```

blend

Case: Recsys Validation / Monitoring

- Adapting an existing project to sacred (at least in validation)
- Many recsys model variants
- Export json configs to s3
- Read remote configs as part of ingredient


Existing System
Sacred for Validation
Datalake
Data Config
Model Config
ingredient
.add_config(...json)
Sacred Server
Validation Config
Validation Experiments
Representation Learning Model
Representations
Monitoring
Prod

Outline

1. Experimental Reproducibility
2. Sacred - A framework for reproducibility
3. Basic Machine Learning with Sacred
4. Advanced use cases
5. **Final Thoughts**

Nov 4

0.51 Lazy None

Seq 322 PV

0.57 Adam OP

0.51 Adam None

2.27 Lazy Nan

2.8 Adam OP

~~Adam None~~

Vanilla MC:

	bpr ↑	hinge ↓	kos
Unif	.812	.849	n/a
" verified	.824	.854	n/a
* adaptive verified warp	.862	.863	.863

127.0.0.1:5000/runs

Sacredboard 0.4.0

Filter runs:

config.? seed or .host.hostname == "string" or 123.4 Add filter

Show 10 entries

Experiments Run Overview

Legend: Running, Completed, Failed, Interrupted, Timeout, ? Probably dead, Queued Refresh

	Id	Experiment name	Command	Start time	Last activity	Hostname	Result
+	283	titanic	run	19:33:26 04/20/18	19:46:16 04/20/18	ML9301.local	
+	282	titanic	run	19:33:26 04/20/18	19:33:26 04/20/18	ML9301.local	0.6681614349775785
+	281	? titanic	run	19:32:56 04/20/18	19:33:06 04/20/18	ML9301.local	
+	280	titanic	run	19:32:56 04/20/18	19:32:56 04/20/18	ML9301.local	0.7085201793721974
+	279	titanic	run	19:32:55 04/20/18	19:32:56 04/20/18	ML9301.local	0.7130044843049327
+	278	titanic	run	19:09:25 04/20/18	19:09:25 04/20/18	ML9301.local	0.43946188340807174
+	277	titanic	run	19:09:25 04/20/18	19:09:25 04/20/18	ML9301.local	0.4484304932735426
+	276	titanic	run	19:09:25 04/20/18	19:09:25 04/20/18	ML9301.local	0.4484304932735426
+	275	titanic	run	19:09:25 04/20/18	19:09:25 04/20/18	ML9301.local	0.43946188340807174
+	274	titanic	run	19:09:25 04/20/18	19:09:25 04/20/18	ML9301.local	0.43946188340807174

Showing 1 to 10 of 283 entries

Previous 1 2 3 4 5 ... 29 Next

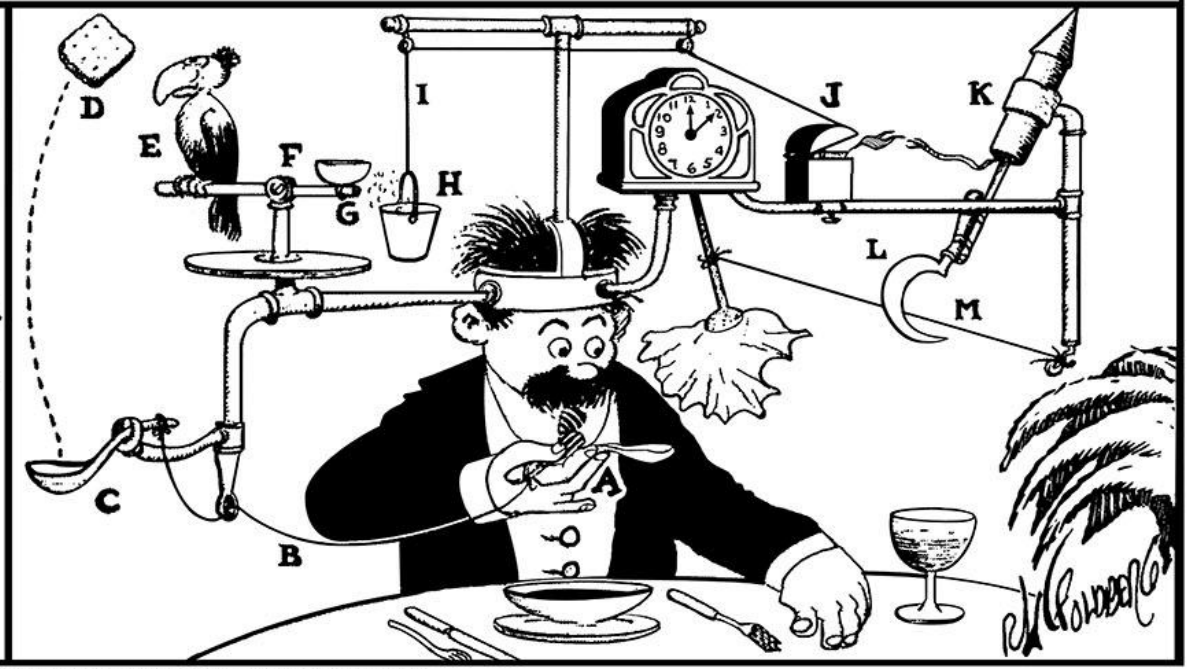
Over / Under - Engineered? … maybe

Self-Operating Napkin by Rube Goldberg

PROFESSOR BUTTS WALKS IN HIS SLEEP, STROLLS THROUGH A CACTUS FIELD IN HIS BARE FEET, AND SCREAMS OUT AN IDEA FOR A SELF-OPERATING NAPKIN.

AS YOU RAISE SPOON OF SOUP (A) TO YOUR MOUTH IT PULLS STRING (B), THEREBY JERKING LADLE (C) WHICH THROWS CRACKER (D) PAST PARROT (E). PARROT JUMPS AFTER CRACKER AND PERCH (F) TILTS, UPSETTING SEEDS (G) INTO PAIL (H). EXTRA WEIGHT IN PAIL PULLS CORD (I) WHICH OPENS AND LIGHTS AUTOMATIC CIGAR LIGHTER (J), SETTING OFF SKY-ROCKET (K) WHICH CAUSES SICKLE (L) TO CUT STRING (M) AND ALLOW PENDULUM WITH ATTACHED NAPKIN TO SWING BACK AND FORTH THEREBY WIPING OFF YOUR CHIN.

AFTER THE MEAL, SUBSTITUTE A HARMONICA FOR THE NAPKIN AND YOU'LL BE ABLE TO ENTERTAIN THE GUESTS WITH A LITTLE MUSIC.



If anything...

- Version Control
- Config
- Seed
- Persist

Please please please check out...

https://github.com/IDSIA/sacred

(^ also contains list of related projects)

https://github.com/chovanecm/sacredboard

Thank You!

jtam@gilt.com

krajasethupathy@gilt.com

http://**tech.hbc**.com/

References

1. https://code.facebook.com/posts/1072626246134461/introducing-fblearner-flow-facebook-s-ai-backbone/
2. https://eng.uber.com/michelangelo/
3. https://github.com/IDSIA/sacred
4. https://github.com/chovanecm/sacredboard
5. https://github.com/hyperopt/hyperopt
6. https://www.kaggle.com/c/titanic/data