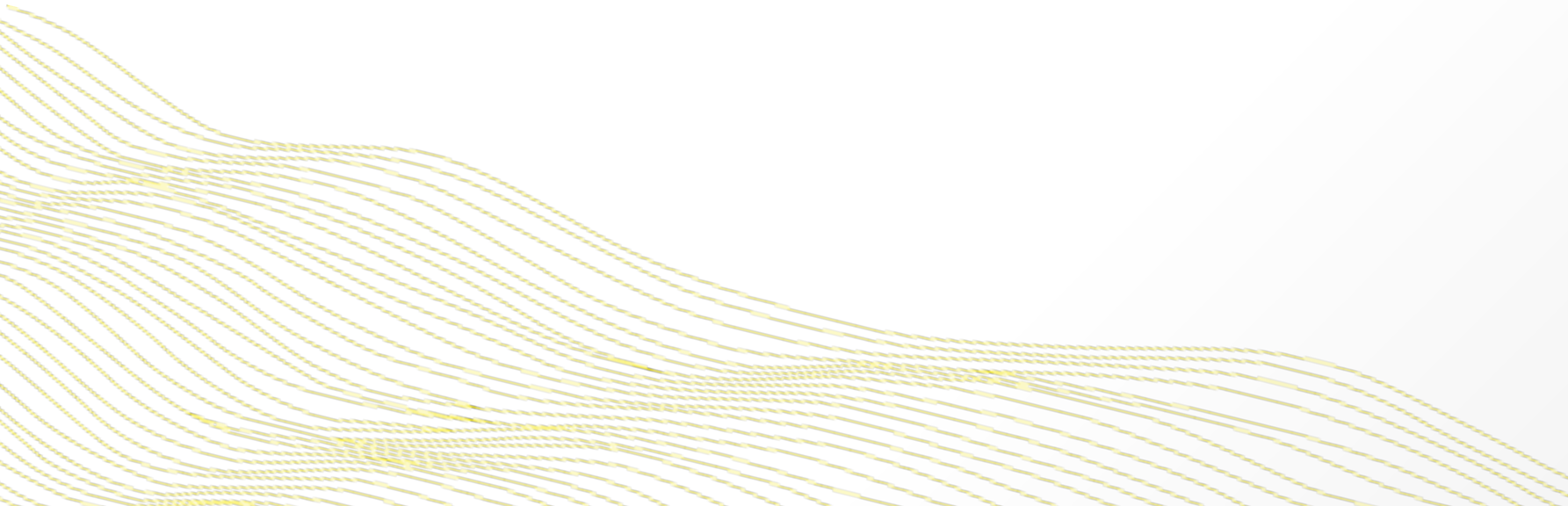
Enhancing Spark Pipeline API with Sparkling Water



Martin Barus

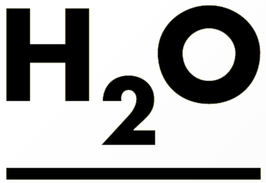
Spark, H2O-3 and Sparkling Water

Apache Spark

- Analytics engine for large-scale data processing

H2O-3

- Distributed in-memory machine learning platform



Sparkling Water

- H2O-3 and Apache Spark Integration

Spark Pipeline API

Transformer

- Implements method `transform()`
- Converts one DataFrame into another
- Tokenizer, Bucketizer, ElementWiseProduct …

Estimator

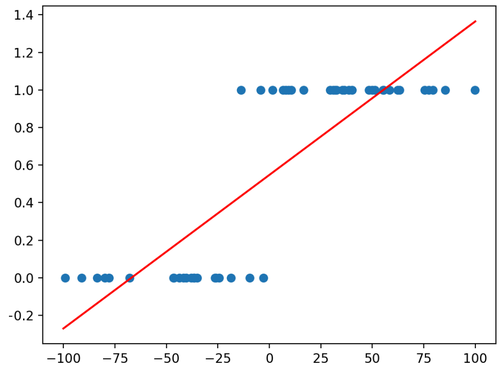
- Implements method `fit()`
- `fit()` accepts DataFrame and produces Model, which is Transformer
- PCA, StandardScaler, Linear Regression, Random Forest …

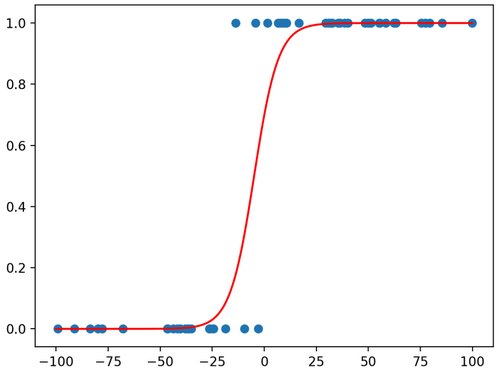
Pipeline

- Chains multiple Transformers and Estimators to create ML Workflow
- Estimator, `fit` returns Transformer

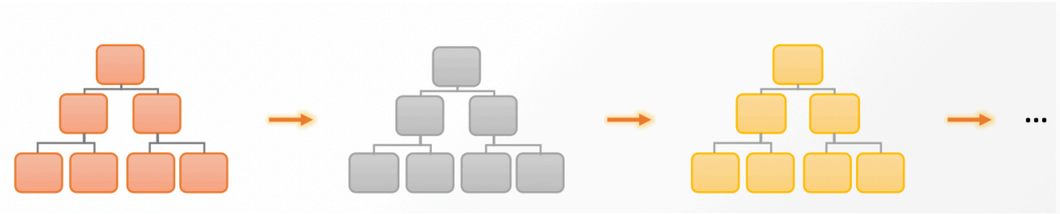
SW Algorithms with Spark Pipeline API

- Generalized Linear Models

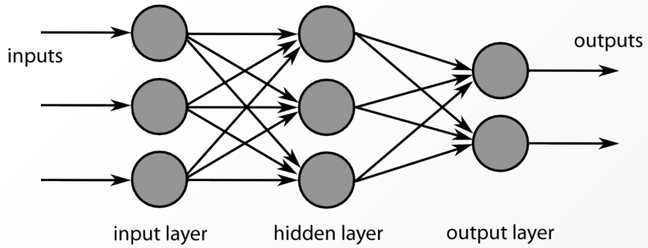




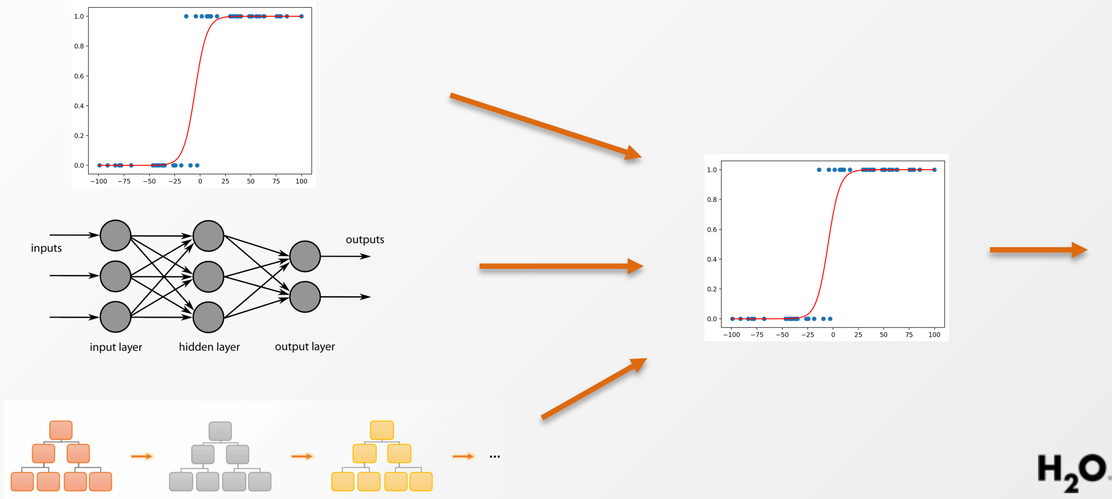
- Gradient Boosting Machine, XGBoost, Distributed Random Forest



- Deep Learning



- AutoML



H2O.ai

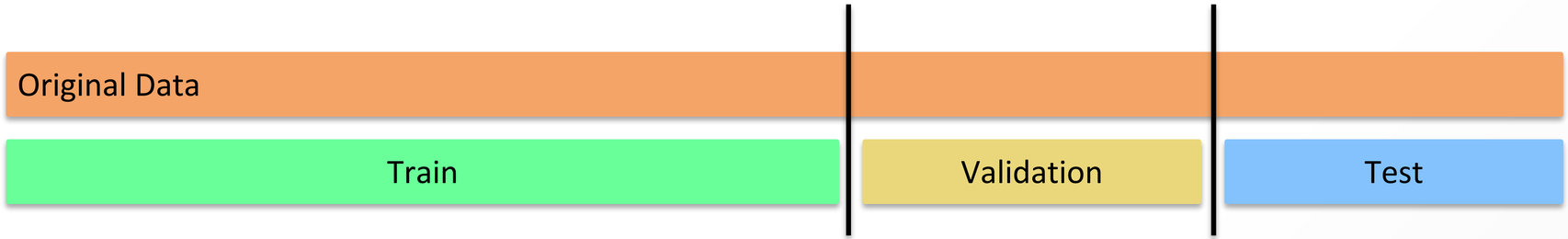
Spark Pipeline API

```
from pysparkling import *
from pysparkling.ml import H2OGBM
from pyspark.ml import Pipeline
from pyspark.ml.feature import SQLTransformer

h2o_gbm = H2OGBM(seed=1,
                 featuresCols=feature_cols,
                 predictionCol=target_col)

statement = "SELECT *, prediction_output.Value AS prediction FROM __THIS__"
rename_stage = SQLTransformer(statement=statement)
h2o_pipe = Pipeline(stages=[h2o_gbm, rename_stage])
h2o_model = h2o_pipe.fit(df_train_processed)
prediction = h2o_model.transform(df_val_processed)
```

Validation



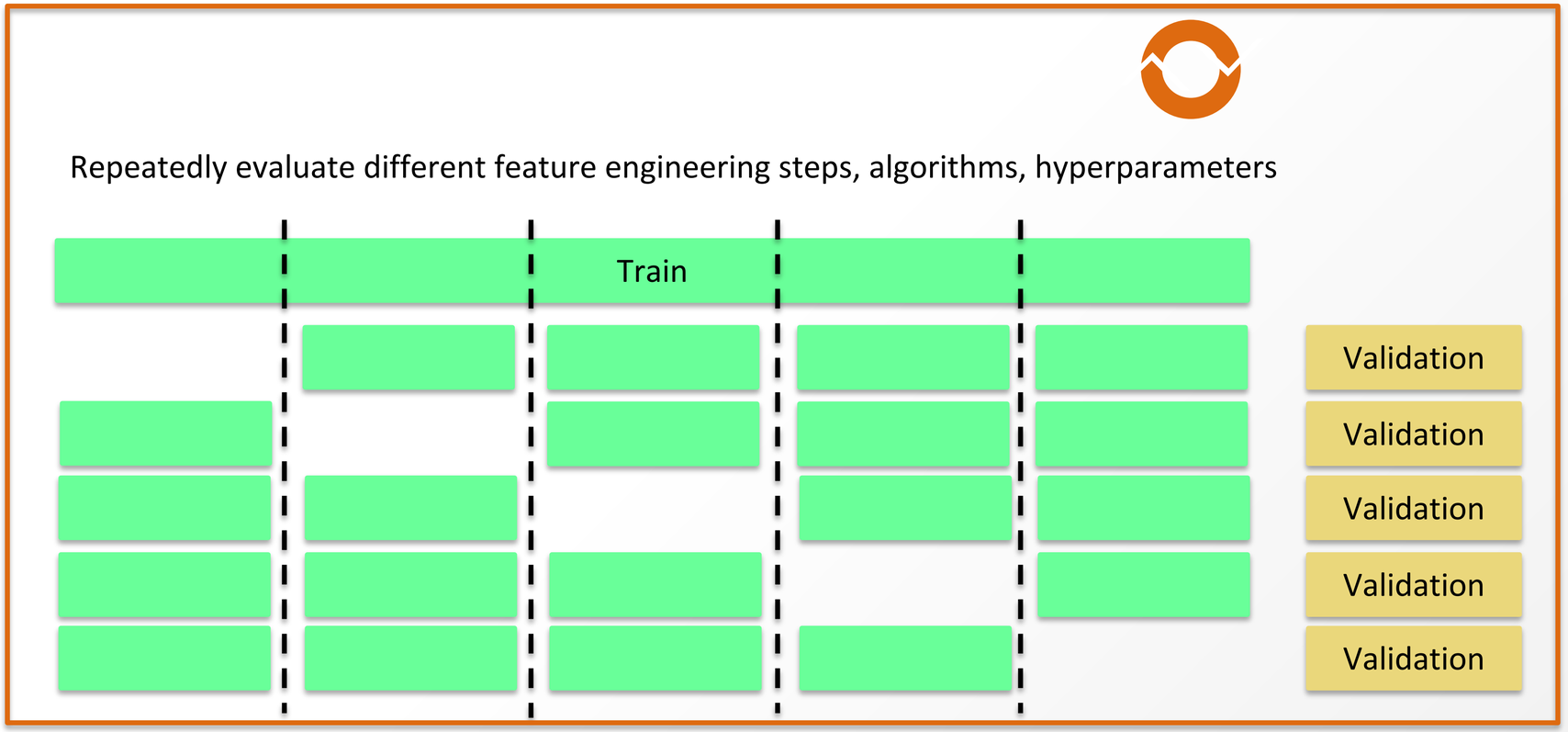
Repeatedly evaluate different feature engineering steps, algorithms, hyperparameters

Train | Validation

Final Evaluation of your model

Test

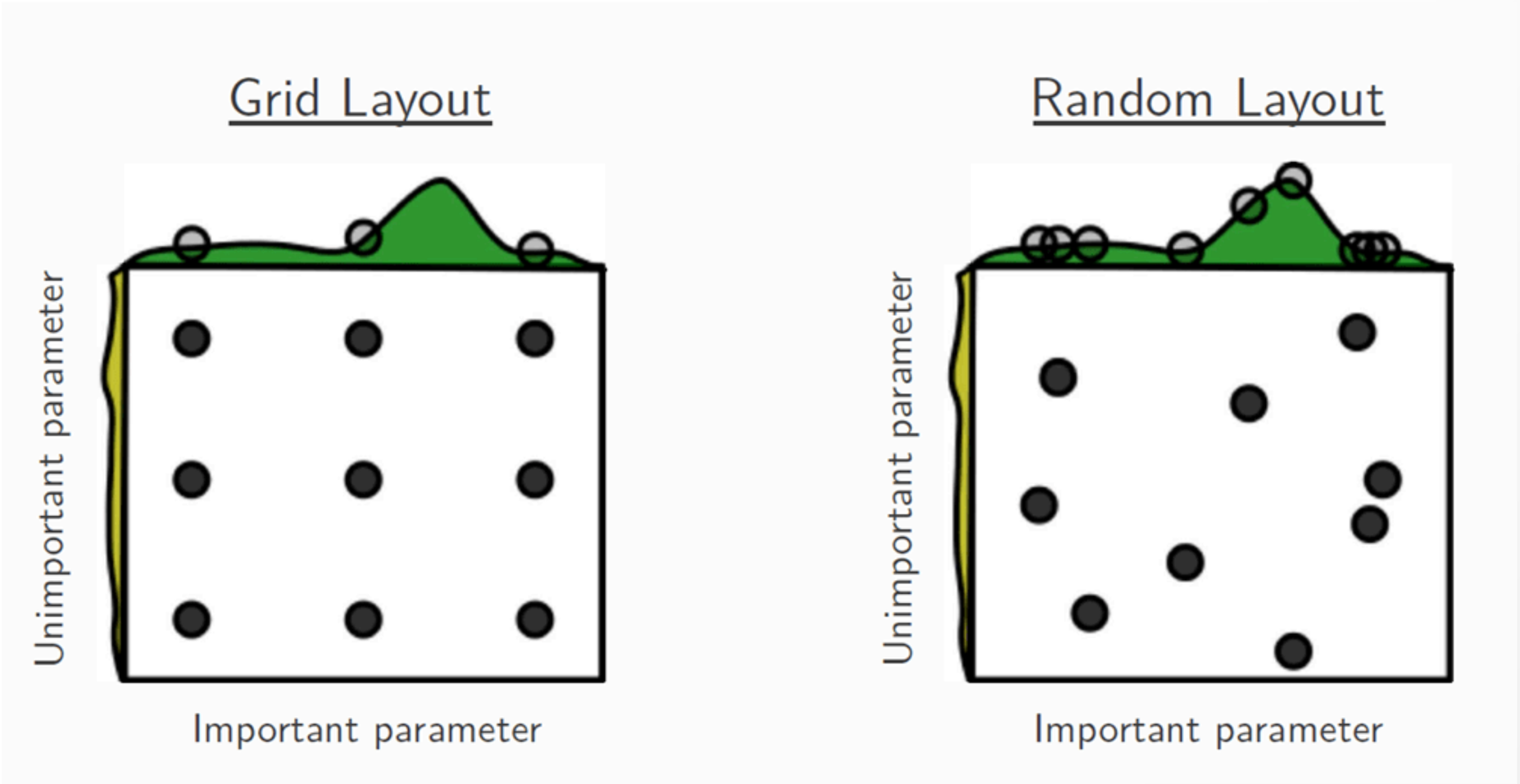
Cross Validation



Final Evaluation of your model

Test

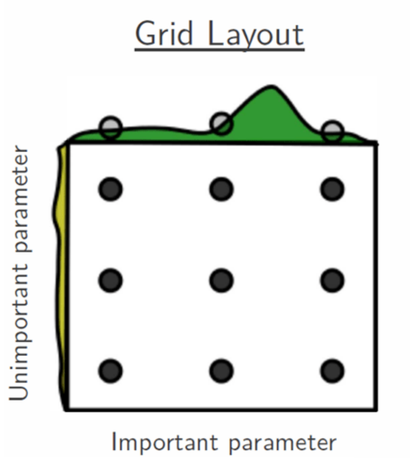
Grid Search


Grid Layout
Unimportant parameter
Important parameter
Random Layout
Unimportant parameter
Important parameter

Hyper-parameter tuning

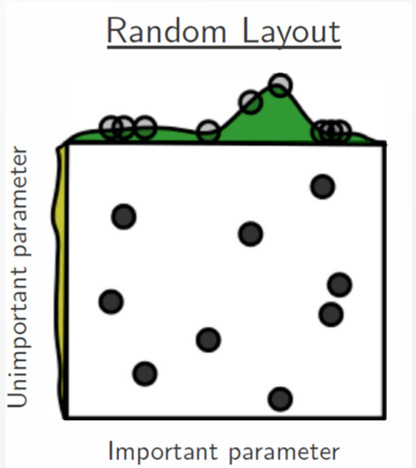
Apache Spark

- CrossValidator
  - **cartesian** grid search + cross validation
- TrainValidationSplit
  - **cartesian** grid search on train/val split



Sparkling Water

- Grid Search
  - Limit execution time by max models/runtime/stopping criterion
  - **Cartesian / Random**
  - Cross validation or train/val split (ratio)



H2O.ai

Hyper-parameter tuning

```
from pysparkling import *
from pysparkling.ml import H2OGridSearch, H2OGBM
from pyspark.ml import Pipeline
from pyspark.ml.feature import SQLTransformer

gbm_params = {'_learn_rate': [i * 0.01 for i in range(1, 11)],
              '_max_depth': list(range(2, 100,4)),
              '_sample_rate': [i * 0.1 for i in range(5, 11)],
              '_col_sample_rate': [i * 0.1 for i in range(1, 11)],
              '_ntrees' : [10**i for i in range(4)]}

grid_search = H2OGridSearch(algo=H2OGBM(),
                            ratio=1.0,
                            nfolds=5,
                            hyperParameters=gbm_params,
                            predictionCol=target_col,
                            strategy='RandomDiscrete',
                            maxModels=10,
                            selectBestModelBy='RMSE',
                            selectBestModelDecreasing=True)

statement = "SELECT *, prediction_output.Value AS prediction FROM __THIS__"
rename_stage = SQLTransformer(statement=statement)
h2o_grid_search_pipe = Pipeline(stages=[grid_search, rename_stage])
h2o_grid_search_model = h2o_grid_search_pipe.fit(df_train_processed)
```

H2O.ai

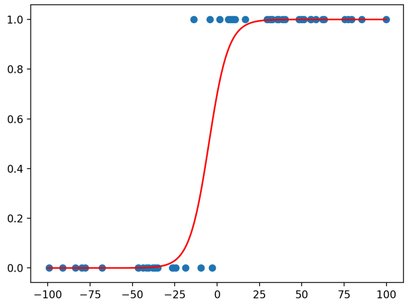
Hyper-parameter tuning

Mojo Model ID	_ntrees	_sample_rate	_col_sample_rate	_learn_rate	_max_depth	RMSE
GBM_mojoModel_ad0231a6b825	100	0.9	0.6000000000000001	0.04	74	0.469885
GBM_mojoModel_b17a317a2cfe	100	0.7000000000000001	0.6000000000000001	0.02	22	0.470060
GBM_mojoModel_1a067f5c6438	100	0.7000000000000001	0.5	0.07	78	0.478073
GBM_mojoModel_6313fb7586c6	10	0.9	0.30000000000000004	0.1	70	0.484843
GBM_mojoModel_32447a085acb	1000	0.5	0.8	0.05	14	0.487212
GBM_mojoModel_752193111635	1000	0.9	0.30000000000000004	0.06	70	0.487293
GBM_mojoModel_a71549ad61b7	1000	0.5	0.6000000000000001	0.05	70	0.489884
GBM_mojoModel_5a513838921b	10	0.6000000000000001	0.30000000000000004	0.05	22	0.495684
GBM_mojoModel_d85b078d9a93	10	0.7000000000000001	0.1	0.08	2	0.512300
GBM_mojoModel_d7164d3e8231	1	1.0	0.9	0.09	14	0.512508

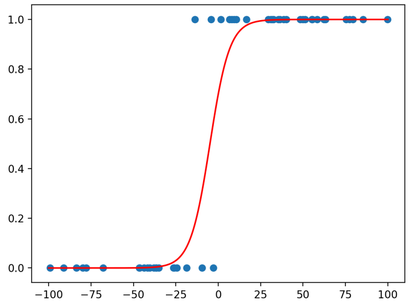
Stacking (Setting up)

- Specify a list of L base models (different algorithms/parameters)

Parameters 1



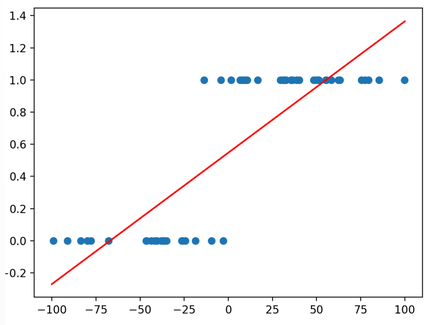
Parameters 2



Parameters 3

- Specify metalearning model

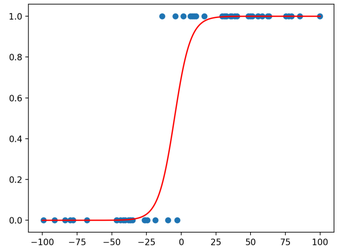
Elastic Net



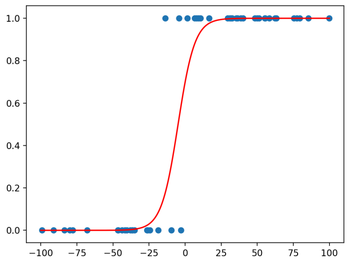
H2O.ai

Stacking (Train Ensemble)

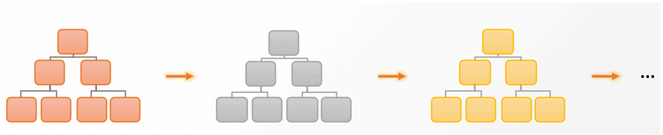
Parameters 1



Parameters 2

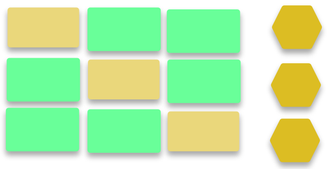


Parameters 3

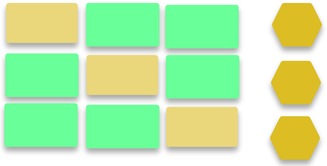


K-fold cross validation

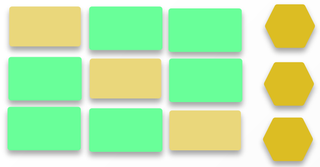
Out of fold predictions



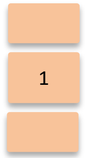
Model 1

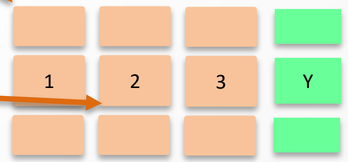


Model 2

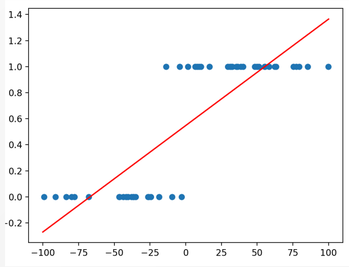


Model 3





Elastic Net

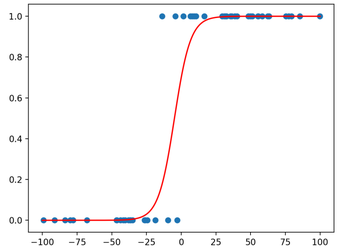


Metamodel

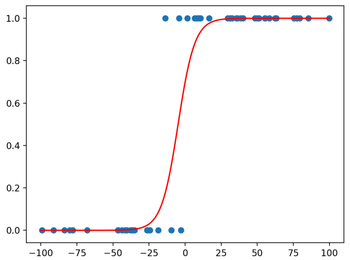
H2O.ai

Stacking (Predict)

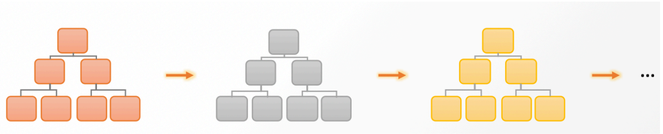
Parameters 1



Parameters 2



Parameters 3



Model 1

Model 2

Model 3

$\hat{Y}_1$

$\hat{Y}_2$

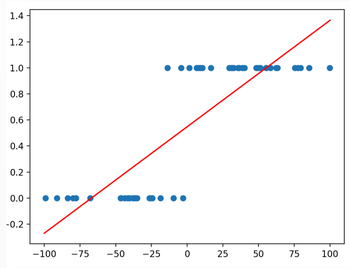
$\hat{Y}_3$

$\hat{Y}_1$ $\hat{Y}_2$ $\hat{Y}_3$

Metamodel

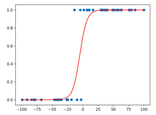
$\hat{Y}$

Elastic Net



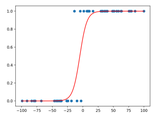
H2O AutoML

GLM 1



AUC 0.79

GLM 2



AUC 0.75

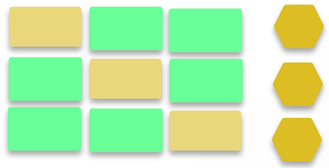
GBM 1

AUC 0.81

GBM2

AUC 0.84

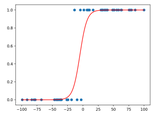
K-fold cross validation



Out of fold predictions

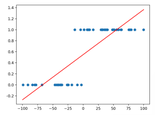
Model

GLM 1



GBM2

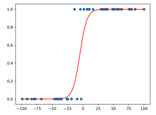
Elastic Net 1



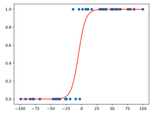
AUC 0.85

$\hat{Y}_1 \cdots \hat{Y}_N \rightarrow$ Metamodel $\rightarrow \hat{Y}$

GLM 1



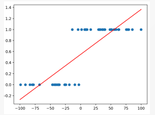
GLM 2



GBM 1

Elastic Net 2

Elastic Net 1



AUC 0.86

H2O.ai

AutoML

```
from pysparkling import *

from pysparkling.ml import H2OAutoML

from pyspark.ml import Pipeline

from pyspark.ml.feature import SQLTransformer


h2o_automl = H2OAutoML(seed=1,

                       predictionCol=target_col,

                       maxRuntimeSecs=6000,

                       ignoredColumns=[id_col],

                       keepCrossValidationPredictions=False,

                       keepCrossValidationModels=False,

                       maxModels = 10)

statement = "SELECT *, prediction_output.Value AS prediction FROM __THIS__”

rename_stage = SQLTransformer(statement=statement)

h2o_automl_pipe = Pipeline(stages=[h2o_automl, rename_stage])

h2o_automl_model = h2o_automl_pipe.fit(df_train_processed)  # Trained Model

automl_lb = h2o_automl.leaderboard().toPandas()  # AutoML Leader Board
```

Thank You!