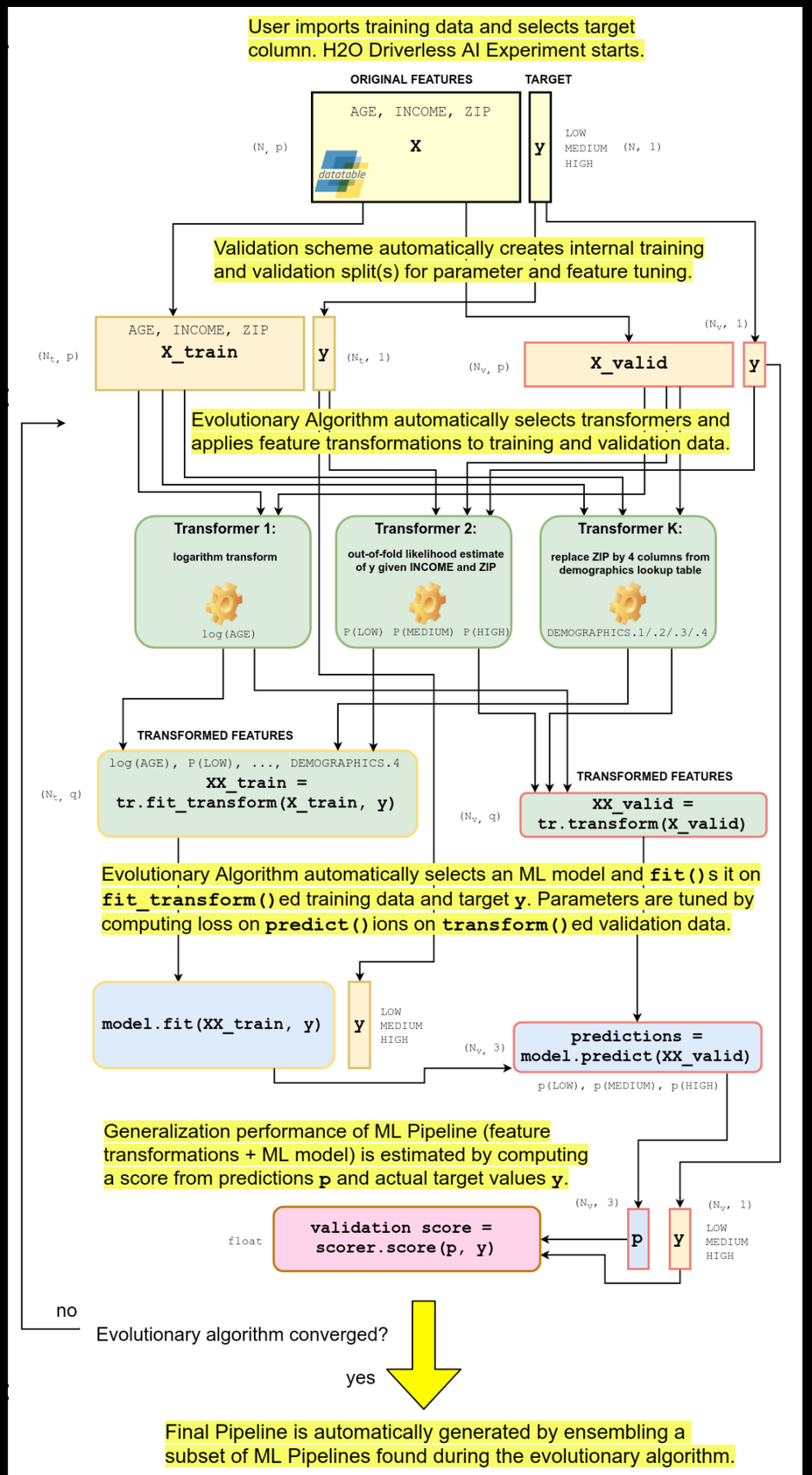
DAI Expert settings

Transformers

Transformer recipes = feature engineering

Feature engineering in DAI



H2O.ai
DRIVERLESSAI

User imports training data and selects target column. H2O Driverless AI Experiment starts.

ORIGINAL FEATURES

TARGET

AGE, INCOME, ZIP

X

(N, p)

y

LOW
MEDIUM
HIGH

(N, 1)

Validation scheme automatically creates internal training and validation split(s) for parameter and feature tuning.

AGE, INCOME, ZIP

X_train

(N$_t$, p)

y

(N$_t$, 1)

(N$_v$, 1)

(N$_v$, p)

X_valid

y

Evolutionary Algorithm automatically selects transformers and applies feature transformations to training and validation data.

Transformer 1:

logarithm transform

log(AGE)

Transformer 2:

out-of-fold likelihood estimate of y given INCOME and ZIP

P(LOW) P(MEDIUM) P(HIGH)

Transformer K:

replace ZIP by 4 columns from demographics lookup table

DEMOGRAPHICS.1/.2/.3/.4

TRANSFORMED FEATURES

log(AGE), P(LOW), ..., DEMOGRAPHICS.4

XX_train = tr.fit_transform(X_train, y)

(N$_t$, q)

(N$_v$, q)

TRANSFORMED FEATURES

XX_valid = tr.transform(X_valid)

H$_2$O.ai DRIVERLESSAI

Evolutionary Algorithm automatically selects an ML model and **`fit()`**s it on **`fit_transform()`**ed training data and target **`y`**. Parameters are tuned by computing loss on **`predict()`**ions on **`transform()`**ed validation data.

`model.fit(XX_train, y)`

`y`

LOW
MEDIUM
HIGH

$(N_v, 3)$

`predictions = model.predict(XX_valid)`

p(LOW), p(MEDIUM), p(HIGH)

Generalization performance of ML Pipeline (feature transformations + ML model) is estimated by computing a score from predictions **`p`** and actual target values **`y`**.

$(N_v, 3)$ $(N_v, 1)$

float **`validation score = scorer.score(p, y)`**

`p` **`y`**

LOW
MEDIUM
HIGH

no

Evolutionary algorithm converged?

yes

Final Pipeline is automatically generated by ensembling a subset of ML Pipelines found during the evolutionary algorithm.

Demo : Expert settings

Transformers included in DAI

http://docs.h2o.ai/driverless-ai/latest-stable/docs/userguide/transformations.html

1. Numeric (int, real, binary)

2. Time series

3. Categorical (string)

4. Text (string)

5. Time (date, time)

Transformers included in DAI

http://docs.h2o.ai/driverless-ai/latest-stable/docs/userguide/transformations.html

1. Numeric (int, real, binary)
2. Time series
3. Categorical (string)
4. Text (string)
5. Time (date, time)

- Original
- Interactions
- ClusterDist
- NumCatTE
- NumToCatTE
- ClusterTE
- NumToCatWoE
- NumToCatWoEMonotonic
- TruncSVDNum

Transformers included in DAI

http://docs.h2o.ai/driverless-ai/latest-stable/docs/userguide/transformations.html

1. Numeric (int, real, binary)
2. Time series
3. Categorical (string)
4. Text (string)
5. Time (date, time)

- Original
- Interactions
- ClusterDist
- NumCatTE
- NumToCatTE
- ClusterTE
- NumToCatWoE
- NumToCatWoEMonotonic
- TruncSVDNum

- Interactions

f : +, -, x, /

y	x1	x2	x3
[0,100]	<str>	<int>	<int>
[0,100]	<str>	<int>	<int>
[0,100]	<str>	<int>	<int>
[0,100]	<str>	<int>	<int>
[0,100]	<str>	<int>	<int>

y	x1	x2	x3	f(2,3)
[0,100]	<str>	<int>	<int>	<int>
[0,100]	<str>	<int>	<int>	<int>
[0,100]	<str>	<int>	<int>	<int>
[0,100]	<str>	<int>	<int>	<int>
[0,100]	<str>	<int>	<int>	<int>

Name of variable f(2,3) : Interaction_x2#subtract#x3

Transformers included in DAI

http://docs.h2o.ai/driverless-ai/latest-stable/docs/userguide/transformations.html

1. Numeric (int, real, binary)
2. Time series
3. Categorical (string)
4. Text (string)
5. Time (date, time)

- Original
- Interactions
- ClusterDist
- NumCatTE
- NumToCatTE
- ClusterTE
- NumToCatWoE
- NumToCatWoEMonotonic
- TruncSVDNum

- ClusterDist

f : 2,3 distance from cluster n (total k clusters, k-Means)

y	x1	x2	x3
[0,100]	<str>	<int>	<int>
[0,100]	<str>	<int>	<int>
[0,100]	<str>	<int>	<int>
[0,100]	<str>	<int>	<int>
[0,100]	<str>	<int>	<int>

y	x1	x2	x3	f(2,3)
[0,100]	<str>	<int>	<int>	<int>
[0,100]	<str>	<int>	<int>	<int>
[0,100]	<str>	<int>	<int>	<int>
[0,100]	<str>	<int>	<int>	<int>
[0,100]	<str>	<int>	<int>	<int>

Name of variable f(2,3) : ClusterDist_k_x2_x3_n

Transformers included in DAI

http://docs.h2o.ai/driverless-ai/latest-stable/docs/userguide/transformations.html

1. Numeric (int, real, binary)
2. Time series
3. Categorical (string)
4. Text (string)
5. Time (date, time)

- Original
- Interactions
- ClusterDist
- NumCatTE
- NumToCatTE
- ClusterTE
- NumToCatWoE
- NumToCatWoEMonotonic
- TruncSVDNum

- NumCatTE

Categorical variable **x1** holds m categories

y	x1	x2	x3
[0,100]	<str>	<int>	<int>
[0,100]	<str>	<int>	<int>

Training dataset

y	x1	x2	x3	*f*(1,y)
[0,100]	<str>	<int>	<int>	(0,100)
[0,100]	<str>	<int>	<int>	(0,100)

f :Mean of out-of-sample target for category n of variable **x1**

y	x1	x2	x3
[0,100]	<str>	<int>	<int>
[0,100]	<str>	<int>	<int>

Validation dataset (k-folds)

Name of variable *f*(1,y) : CV_TE_x1

Target Encoder Transformers in a nutshell

Encoder name	Variables in training dataset…	Response in validation dataset…	Name of output variable
NumCatTE	…are identified as categorical	…is averaged for each category	CV_TE_<str>
NumToCatTE	…are binned to become categorical	…is averaged for each category	CV_TE_<num>
ClusterTE	…are clustered	…is averaged for each cluster	CV_TE_<num>_<num>_…

Transformers included in DAI

http://docs.h2o.ai/driverless-ai/latest-stable/docs/userguide/transformations.html

1. Numeric (int, real, binary)
2. Time series
3. Categorical (string)
4. Text (string)
5. Time (date, time)

- Original
- Interactions
- ClusterDist
- NumCatTE
- NumToCatTE
- ClusterTE
- NumToCatWoE
- NumToCatWoEMonotonic
- TruncSVDNum

Weight of Evidence (WoE) in a nutshell

High cardinality categorical variables mapped to binary outcomes

Category	Number of observations	Distribution of outcome Y	Distribution of outcome N	WoE
A	$Y_A + N_A$	$D^Y_A = Y_A / Y$	$D^N_A = N_A / N$	$\ln(D^Y_A / D^N_A)$
B	$Y_B + N_B$	$D^Y_B = Y_B / Y$	$D^N_B = N_B / N$	$\ln(D^Y_B / D^N_B)$
C	$Y_C + N_C$	$D^Y_C = Y_C / Y$	$D^N_C = N_C / N$	$\ln(D^Y_C / D^N_C)$
D	$Y_D + N_D$	$D^Y_D = Y_D / Y$	$D^N_D = N_D / N$	$\ln(D^Y_D / D^N_D)$
All	Y + N	Y	N	

Example:
Categories: Education level bins
Outcomes: Good loan status (Y), Defaulted (N)

- NumToCatWoE

f : (1) binning(x) -> categorical_x
(2) WoE(categorical_x)

y	x1	x2	x3
[0,100]	<str>	<int>	<int>
[0,100]	<str>	<int>	<int>
[0,100]	<str>	<int>	<int>
[0,100]	<str>	<int>	<int>
[0,100]	<str>	<int>	<int>

y	x1	x2	x3	f(3)
[0,100]	<str>	<int>	<int>	<int>
[0,100]	<str>	<int>	<int>	<int>
[0,100]	<str>	<int>	<int>	<int>
[0,100]	<str>	<int>	<int>	<int>
[0,100]	<str>	<int>	<int>	<int>

Transformers included in DAI

http://docs.h2o.ai/driverless-ai/latest-stable/docs/userguide/transformations.html

1. Numeric (int, real, binary)
2. Time series
3. Categorical (string)
4. Text (string)
5. Time (date, time)

- Original
- Interactions
- ClusterDist
- NumCatTE
- NumToCatTE
- ClusterTE
- NumToCatWoE
- NumToCatWoEMonotonic
- TruncSVDNum

- TruncSVDNum

$$A = U \cdot \Sigma \cdot V^T$$

Σ (=singular values): diagonal matrix with ordered values

f : (1) calculate U, Σ, V^T from original data (A)
(2) Truncate Σ to the top k singular values (Σ_k)
(3) Use Σ_k and V_k^T to calculate new matrix B

k = 2

y	x1	x2	x3	x4
[0,100]	<str>	<int>	<int>	<int>
[0,100]	<str>	<int>	<int>	<int>
[0,100]	<str>	<int>	<int>	<int>
[0,100]	<str>	<int>	<int>	<int>

A

y	x1	x'1	x'2
[0,100]	<str>	<int>	<int>
[0,100]	<str>	<int>	<int>
[0,100]	<str>	<int>	<int>
[0,100]	<str>	<int>	<int>

B

Name of variable f(1,y) : TruncSVD_x2_x3_x4_k(n^{th})

DAI, custom recipes

```python
"""Adds together 3 or more numeric features"""
from h2oaicore.transformer_utils import CustomTransformer
import datatable as dt
import numpy as np


class SumTransformer(CustomTransformer):

    _regression = True
    _binary = True
    _multiclass = True
    _numeric_output = True
    _is_reproducible = True
    _included_model_classes = None  # List[str]
    _excluded_model_classes = None  # List[str]

    @staticmethod
    def is_enabled():
        return True

    @staticmethod
    def do_acceptance_test():
        return True

    @staticmethod
    def get_default_properties():
        return dict(col_type="numeric", min_cols=3, max_cols="all", relative_importance=1)

    def fit_transform(self, X: dt.Frame, y: np.array = None):
        return self.transform(X)

    def transform(self, X: dt.Frame):
        return X[:, dt.sum([dt.f[x] for x in range(X.ncols)])]
```

Demo : Custom recipes