ZELROS

AI for Augmented Insurers.

Detect, explain & correct data drift in a machine learning system.

November 2021

2021/11

Agenda

1. Introduction
2. Detect data drift
3. Explain data drift
4. Correct data drift
5. Demo
6. Conclusion

Introduction

Context

- We consider a ML system answering a binary classification problem where:
  - $Y \in \{0, 1\}$: target
  - $X \in \chi \subset R^d$: input

- Let $\hat{f}(x)$ a model which answers the binary classification problem and is already trained on some dataset.

  Some idea developed in the presentation requires that $\hat{f}$ is a tree-based model (Gradient Boosting (XGBoost, LightGBM), Random Forest, CART, etc.).

- Let $\mathcal{D}_1$: $\left(X_{i,1}, Y_{i,1}\right)_{i=1,\dots,n_1}$ (i.i.d.) and $\mathcal{D}_2$: $\left(X_{i,2}, Y_{i,2}\right)_{i=1,\dots,n_2}$ (i.i.d.) two datasets corresponding to the binary classification problem. Let P_{X_1,Y_1} and P_{X_2,Y_2} the corresponding distributions.

  Our goal is to **study the data drift between $\mathcal{D}_1$ and $\mathcal{D}_2$**

- In a typical situation $\mathcal{D}_1$ is the training/validation data and $\mathcal{D}_2$ is the production data.

  In practice depending on the use case, $\left(Y_{i,2}\right)_{i=1,\dots,n_2}$ may not be observed.

What is data drift ?

Definition: A data drift between $\mathcal{D}_1$ and $\mathcal{D}_2$ corresponds to the case where $P_{X_1,Y_1} \neq P_{X_2,Y_2}$

Two categories of data drift:

- Case 1: **Covariate shift**
  - the conditional distributions are the same: $\boldsymbol{P_{Y_1|X_1} = P_{Y_2|X_2}}$, but the distribution of inputs changes: $\boldsymbol{P_{X_1} \neq P_{X_2}}$
- Case 2: **Concept drift**
  - the conditional distributions are not the same: $\boldsymbol{P_{Y_1|X_1} \neq P_{Y_2|X_2}}$

Property: P_X and $P_{Y|X}$ characterize the distribution $P_{X,Y}$

proof: $dP_{X,Y}(x,y) = dP_{Y|X}(y|x) \cdot dP_X(x)$

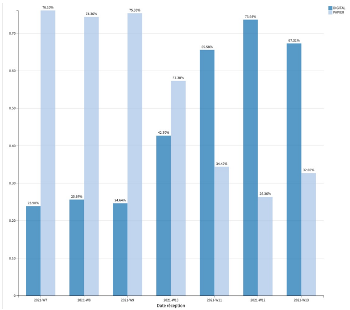
Property: These 2 cases cover all possible cases

proof: In covariate shift, if P_X is unchanged, then $P_{X,Y}$ is given by P_X and $P_{Y|X}$ which are both unchanged. Hence $P_{X,Y}$ is unchanged -> no data drift

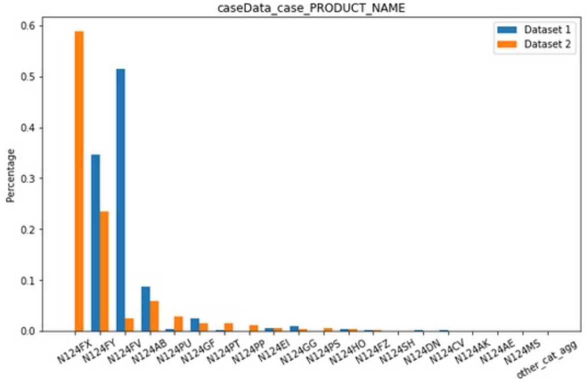
----

Case 1: Examples of covariate shift

- The distribution of some input variable that changes with time. (e.g. evolution of inbound channel repartition: more digital, less paper)
- Some input variable that may take a new value (e.g. new product, new option, new functionality, etc.)
- Hand-based input with a user error in it (e.g. you're expecting an input to be in M$, and it's in $ instead)



Evolution of inbound channel repartition



New product which modifies the input distribution

Case 2: Examples of concept drift

- **Censoring of some observations**
  - At training time, **we filter the data to keep only the samples for which we know the target Y**. Let $D \in \{0, 1\}$ be the censoring variable. Training data is (X_i, Y_i, D_i) with $Y_i = \infty$ if $D_i = 0$ (training data has the distribution $P_{X,Y|D=1}$).
    → Typical situation is the **presence of non stated cases** (accepted, rejected and unknown status).

- **Non-stationary environment**
  - Change of the regulation (e.g. credit agreement).
  - Important events that change behaviors (e.g. propensity to travel during Covid crisis).

- **Spatial drift**
  - Model trained on French insurance data and deployed in the Italian market.

Detect data drift

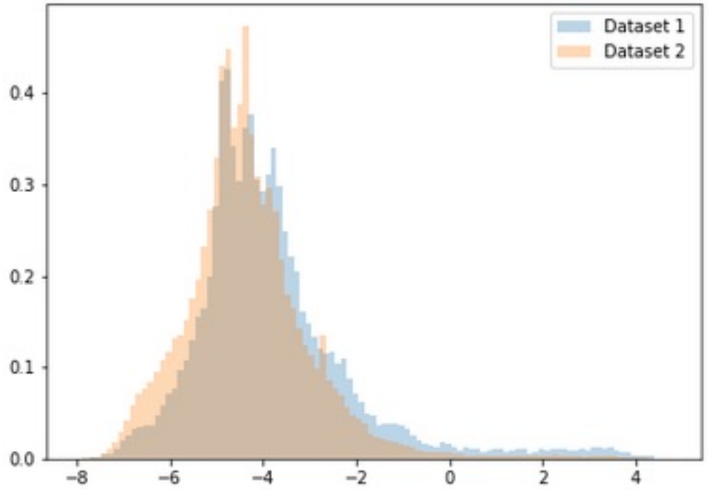
Measure of drift - Numerical variables

- Difference of means.
- Wasserstein distance

  $W(V_1, V_2) = \int_{-\infty}^{+\infty} |F_{V_1}(v) - F_{V_2}(v)| dv,$

  with $F_V(t) = P(V \leq t)$ the CDF of V.
- Kolmogorov-Smirnov 2 sample test.

Distribution of variable V




```
{'mean_difference': -0.514477079781603,
 'wasserstein': 0.5144829964200954,
 'kolmogorov_smirnov': KstestResult(statistic=0.13030577961608247, pvalue=0.0)}
```

Measure of drift - Categorical variables

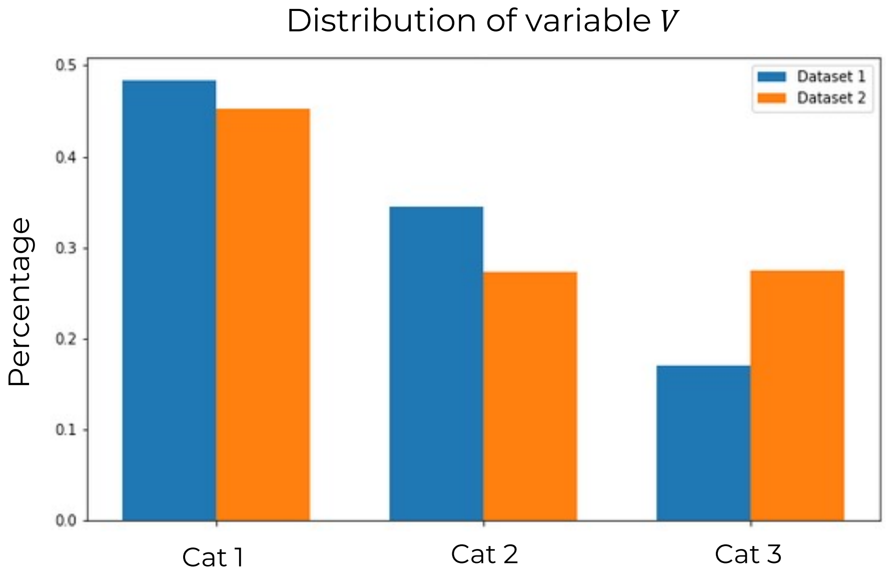
- Wasserstein distance (we assume distance between 2 categories is equal to 1*).

- Chi2 test

  Contingency table :

	Cat 1	Cat 2	Cat 3
X1	1152.0	821.0	407.0
X2	1909.0	1154.0	1163.0

* This corresponds to Wasserstein distance between dummy representations, with $\| \ \|_\infty$ norm

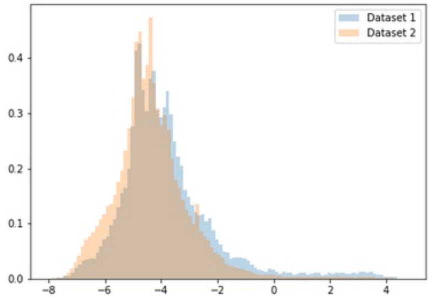



```
{'wasserstein': 0.10419273246449548,
 'chi2_test': {'chi2_stat': 99.2937, 'p_value': 2.74556e-22}}
```

What indicators do we track in the ML system ?

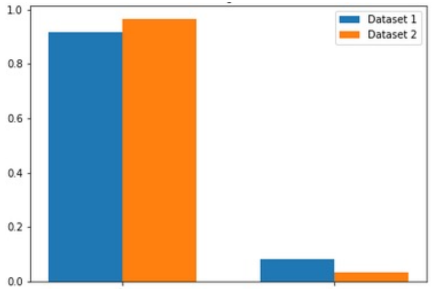
- Distribution of predictions of the model
- Distribution of the target
- Performance metrics

Distribution of predictions




```
{'mean_difference': -0.514477079781603,
 'wasserstein': 0.5144829964200954,
 'kolmogorov_smirnov': KstestResult(statistic=0.13030577961608247, pvalue=0.0)}
```

Distribution of the target




```
{'wasserstein': 0.04893272477192796,
 'chi2_test': {'chi2_stat': 1655.184463874216,
  'p_value': 0.0,
```

Performance metrics

```
log_loss valid: 0.17342663278191264
log_loss prod: 0.10822475472437297
AUC valid: 0.8950968405582416
AUC prod: 0.8393885465363519
```

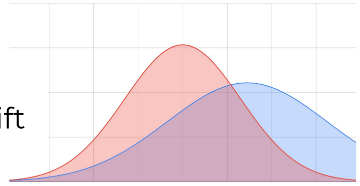
We focus on data drift that have an impact on the ML system (we don't track data drift on all inputs of the model)

Explain data drift

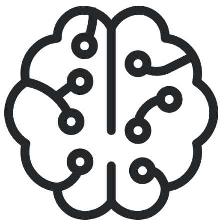
Approach 1: Model based approach

- Ideas:
  - What is the impact of the data drift of a given input on the ML system ?
    - Compute **drift values** of each input (i.e. contribution of the feature to the global data drift)
    - Importance of a data drift on some input of the model should be lowered if the given input is not important in the model
  - Need to study the data drift through the lens of the model
    - **Model specific approach** (here for tree-based model)

Input drift



ML model

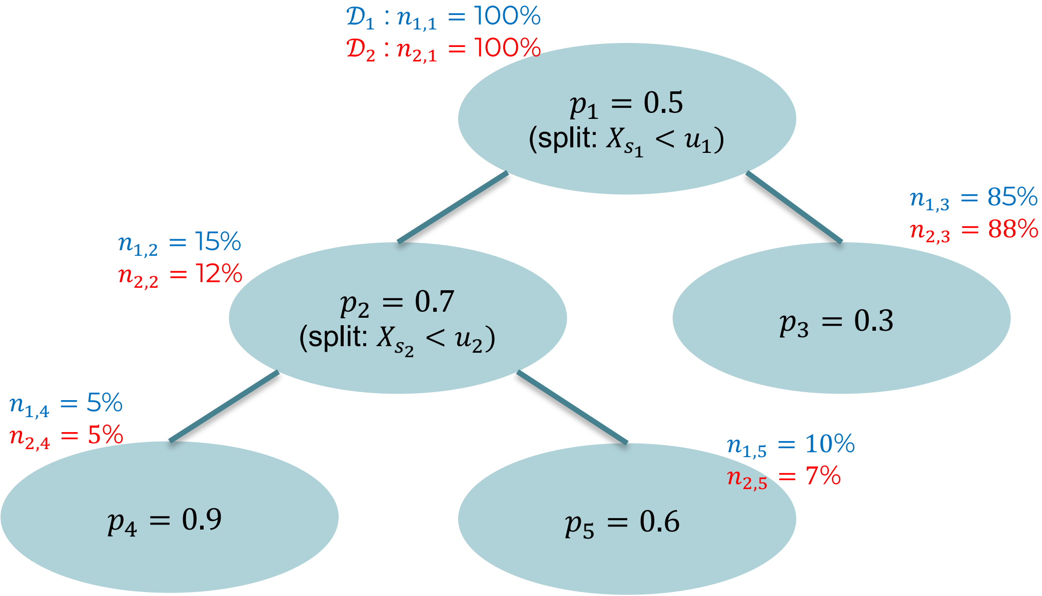


What impact on the ML system ?

Drift values: Calculus ½ (individual tree)

Definition:

- $n_{i,j}$ $(i = 1, 2\ ;\ j = 1, \dots, K)$: proportion of samples of dataset i in node j
- p_j $(j = 1, \dots, K)$: predicted value associated to node j
- s_j, u_j $(j = 1, \dots, K)$: feature index and value used for split j ($s_j, u_j = -1$ if node j is a terminal leaf)
- l_j, r_j $(j = 1, \dots, K)$: indexes of the left and right child nodes of node j ($l_j, r_j = -1$ if node j is a terminal leaf)



Drift values: Calculus 2/2

- Different measure of **split contribution** $S_j (j = 1, \ldots, K)$ to the data drift:
  - node size: $S_j = \left| \frac{n_{2,l_j}}{n_{2,j}} - \frac{n_{1,l_j}}{n_{1,j}} \right| * \min(n_{1,j}, n_{2,j})$
  - mean: $S_j = c_{2,j} - c_{1,j}$, with $c_{i,j} = (n_{i,l_j} * p_{l_j} + n_{i,r_j} * p_{r_j} - n_{i,j} * p_j)$
  - mean with normalization: $S_j = \left(\frac{c_{2,j}}{n_{2,j}} - \frac{c_{1,j}}{n_{1,j}} \right) * \min(n_{1,j}, n_{2,j})$

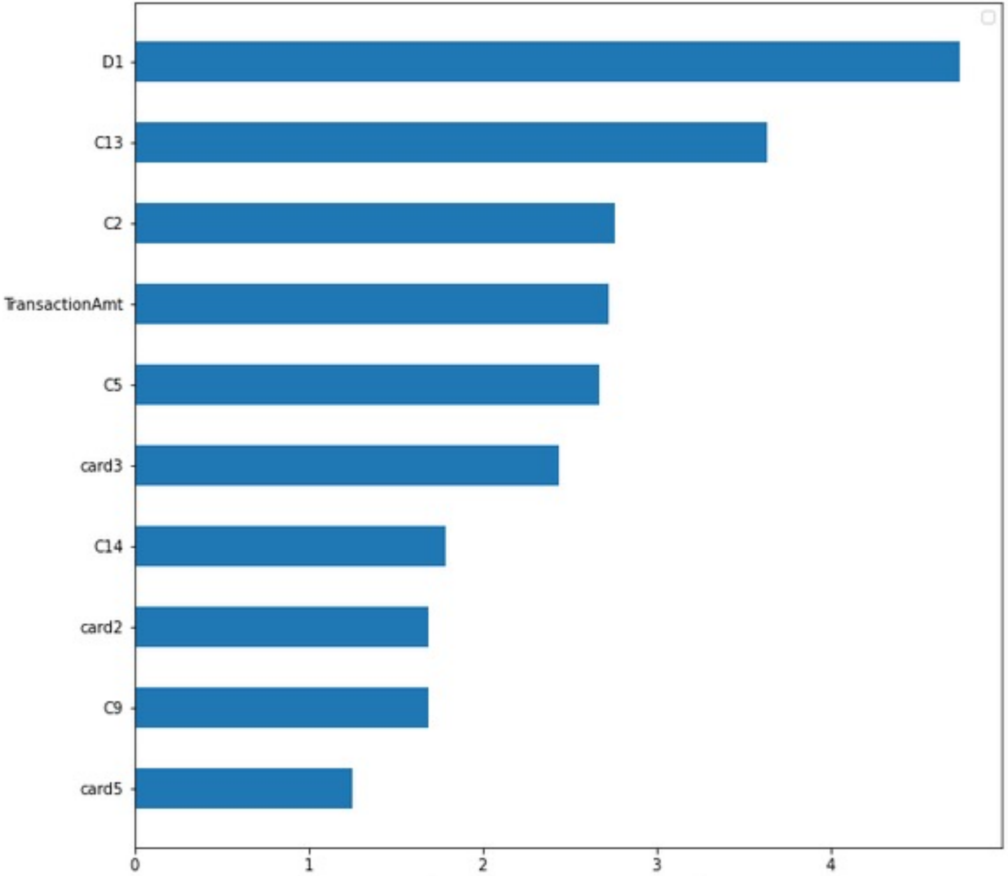
(in all case we set $S_j = 0$ if $l_j = -1$)

- Based on the split contributions, we can compute **drift values**. Let T be number of trees in the model and s_j^t, K_j^t, S_j^t be the above values for the tree t. The feature contribution of the feature k $(k = 1, \ldots, d)$ is defined as:

$$F_k = \sum_{t=1}^{T} F_k^t \text{, with } F_k^t = \sum_{j=1,\ldots K^t / s_j^t = k} S_j^t$$

Drift values: Example

Drift values (mode = “node size”)



Approach 2: Adversarial approach

- Consider $(X_{i,1})_{i=1,\dots,n_1}$ and $(X_{i,2})_{i=1,\dots,n_2}$ and let:
  - $L_{i,1} = 0, w_{i,1} = n_2/n_1$
  - $L_{i,2} = 1, w_{i,2} = 1$

- The **adversarial approach** consists in a building an **adversarial classifier $\hat{c}$** with target L and covariates X based on the concatenation of datasets $(X_{i,1}, L_{i,1}, w_{i,1})_{i=1,\dots,n_1}$ and $(X_{i,2}, L_{i,2}, w_{i,2})_{i=1,\dots,n_2}$

- **Drift values** can be calculated by considering the **feature importance of the adversarial classifier**

X	L	w
$X_{1,1}$	0	n_2/n_1
$X_{2,1}$	0	n_2/n_1
…	…	…

concatenate

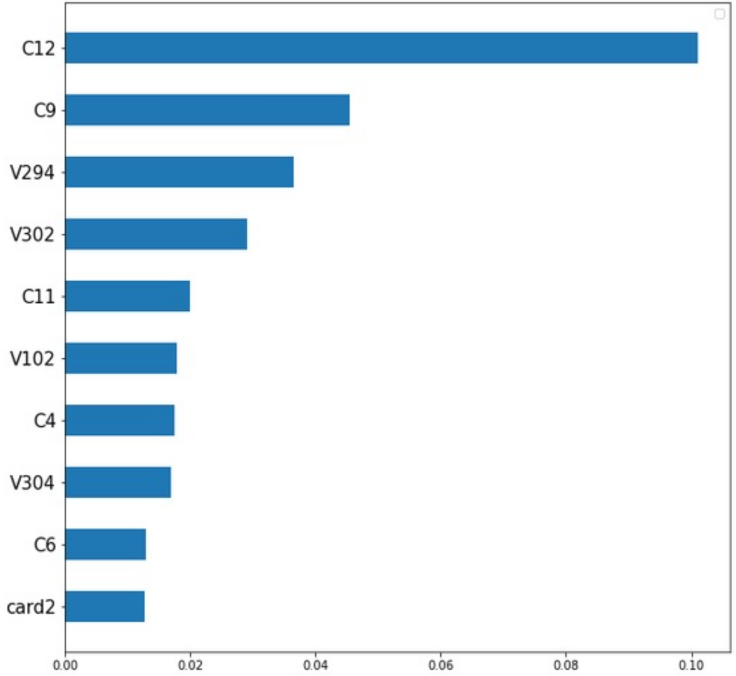
X	L	w
$X_{1,2}$	1	1
$X_{2,2}$	1	1
…	…	…

Adversarial classifier $\hat{c}$

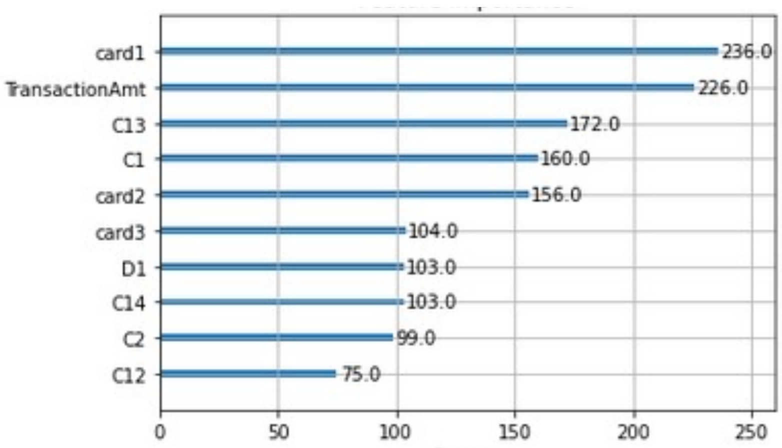
Concatenate $\mathcal{D}_1$ and $\mathcal{D}_2$

Adversarial approach: Example

Drift values with the adversarial method



Feature importance of the model in production



- The drift values obtained with the adversarial approach may be combined with the feature importance of the model. **But it is hard to interpret !**

Correct data drift

Correct covariate shift with importance weighting

Property: Assume there is covariate shift between (X_1, Y_1) and (X_2, Y_2) and that $W(x) = \frac{dP_{X_2}(x)}{dP_{X_1}(x)}$ is defined.

Then for any function g:

$$E[W(X_1)g(X_1, Y_1)] = E[g(X_2, Y_2)]$$

proof: $E[W(X_1)g(X_1, Y_1)] = \iint W(x)g(x,y)dP_{Y_1|X_1}(y|x)\, dP_{X_1}(x) = \iint g(x,y)dP_{Y_2|X_2}(y|x)\, dP_{X_2}(x) = E[g(X_2, Y_2)]$

Morally, thanks to importance weights $W(x)$, it is possible to estimate the distribution (X_2, Y_2) based on the distribution (X_1, Y_1).

- Idea
  - Given the dataset $(X_{i,1}, Y_{i,1})_{i=1,\dots,n_1}$ and $(X_{i,2}, Y_{i,2})_{i=1,\dots,n_2}$, compute sample weights $W_{i,1}$ so that the distribution of $(X_{1,i}, Y_{1,i}, W_{i,1})_{i=1,\dots,n_1}$ approximates the distribution of $(X_{i,2}, Y_{i,2})_{i=1,\dots,n_2}$.
  - Then train, validate, and select model with the dataset $(X_{1,i}, Y_{1,i}, W_{i,1})_{i=1,\dots,n_1}$ should result in better performance on $(X_{i,2}, Y_{i,2})_{i=1,\dots,n_2}$

Proposed procedure to correct data drift

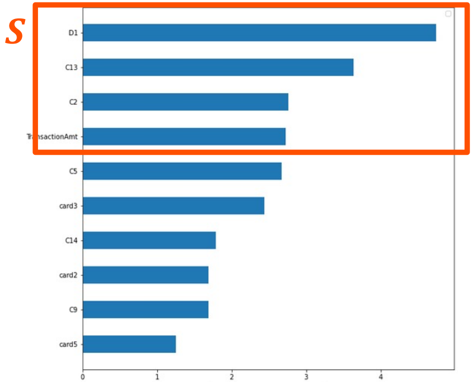
1) Based on model based drift values, **select a subset of input variables $S \subset \{1, \dots, d\}$** for which we will correct the drift.

2) Use the adversarial approach to **build an adversarial classifier $\hat{c}$ only based on feature S**. Let X^S the restriction of X to the variables in S.

3) **Compute weights** $W_{i,1} = \frac{\hat{c}(X_{i,1}^S)}{1 - \hat{c}(X_{i,1}^S)}$:

- Note that $W(x) = \frac{dP_{X_2}(x)}{dP_{X_1}(x)} = \frac{P(L=1|X=x)}{P(L=0|X=x)} = \frac{P(L=1|X=x)}{1-P(L=1|X=x)}$ and thus $W(x)$ can be estimated by $\frac{\hat{c}(x)}{1 - \hat{c}(x)}$
- Cross validation is used in order to compute weights $W_{i,1}$

4) The dataset $(X_{1,i}, Y_{1,i}, W_{i,1})_{i=1,\dots,n_1}$ can then be use for model training, model selection.



Select subset of features S based on drift values

Demo

CinnaMon, a Python library for the monitoring of machine learning systems that focus on data drift: https://github.com/zelros/cinnamon

Conclusion

Conclusion

- Main ideas
  - Monitoring of the data drift
    - Focus on the data drift that has an impact on the ML system ?
      - Detection: Monitor indicators that makes sense from this perspective
      - Explain: Introduce drift values with a Model based approach
  - Correction of the data drift
    - Proposed methodology to correct covariate shift (using weights computed with adversarial approach) -> need more investigation

- Future work:
  - Closer next steps
    - Write an article about CinnaMon
    - CinnaMon library (add test, add documentation, etc.)
  - Research directions
    - Extend the model approach to other specific models
    - Model agnostic method to compute drift values (only relying on ”model.predict” call)
    - Deal with streaming data (add .update() on top of .fit() method). In fact the question is how do you define D1 and D2)
    - Design a live alerting system (with robust alerts) based on the 3 data drift indicators
    - Benchmark the different ways to compute tree-based drift values

Thank you for your attention !

Bibliography

1) Sugiyama, Masashi, and Motoaki Kawanabe. *Machine learning in non-stationary environments: Introduction to covariate shift adaptation*. MIT press, 2012.
2) Sethi, Tegjyot Singh, and Mehmed Kantardzic. "On the reliable detection of concept drift from streaming unlabeled data." *Expert Systems with Applications* 82 (2017): 77-99.

ZELROS

Zelros / *[Z EH L R AO S]*

Word invented by a recurrent neural network trained on 130k tech company names from crunchbase.com

198 Avenue de France
75013 Paris
France

Balanstraße 73/Haus 10
81541 Munich
Germany

Corso di Porta Romana, 61
20122 – Milan
Italy

4 Place Ville-Marie, 2e et 3e étages,
Montréal, QC H3B 2E7
Canada

Drift values: illustration with tree

- Here split contributions are computed with « node size » mode

$n_{1,1}, n_{2,1}$

Name of the splitting feature

S_1

```
C2 (1.0, 1.0) - [0.102]
├── C14 (0.596, 0.494) - [0.03]
│   ├── V314 (0.508, 0.451) - [0.03]
│   │   ├── C6 (0.063, 0.086) - [0.002]
│   │   │   ├── (0.012, 0.014)
│   │   │   └── (0.051, 0.072)
│   │   └── card3 (0.445, 0.365) - [0.022]
│   │       ├── (0.065, 0.031)
│   │       └── (0.38, 0.334)
│   └── card3 (0.088, 0.043) - [0.003]
│       ├── C1 (0.017, 0.006) - [0.0]
│       │   ├── (0.001, 0.0)
│       │   └── (0.016, 0.005)
│       └── C5 (0.071, 0.038) - [0.0]
│           ├── (0.012, 0.006)
│           └── (0.059, 0.032)
└── D1 (0.404, 0.506) - [0.086]
    ├── V317 (0.19, 0.345) - [0.001]
    │   ├── C7 (0.159, 0.292) - [0.001]
    │   │   ├── (0.002, 0.001)
    │   │   └── (0.157, 0.291)
    │   └── card2 (0.031, 0.053) - [0.001]
    │       ├── (0.002, 0.005)
    │       └── (0.029, 0.048)
    └── card3 (0.214, 0.161) - [0.009]
        ├── C5 (0.194, 0.155) - [0.03]
        │   ├── (0.042, 0.064)
        │   └── (0.152, 0.091)
        └── TransactionAmt (0.02, 0.006) - [0.0]
            ├── (0.0, 0.0)
            └── (0.02, 0.006)
```

Adversarial approach (maths)

- Let (X, L) the random vector given by:
    - $L \sim Bernouilli(1/2)$
    - X follows the mixture distribution with:
        - $X|L = 0 \sim X_1$
        - $X|L = 1 \sim X_2$

- Let $(X_i, L_i, w_i)_{i=1,\dots,n_1+n_2}$ the concatenation of the datasets:
    - $\left(X_{i,1}, L_{i,1} = 0, w_{i,1}\right)_{i=1,\dots,n_1}$ with $w_{i,1} = n_2/n_1$
    - $\left(X_{i,2}, L_{i,2} = 1, w_{i,2}\right)_{i=1,\dots,n_2}$ with $w_{i,2} = 1$

    Then $(X_i, L_i, w_i)_{i=1,\dots,n_1+n_2}$ follows the same distribution as (X, L)

- The **adversarial approach** consists in a building a **discrimination model** $\hat{c}$ with target L and covariates X(based on data $(X_i, L_i, w_i)_{i=1,\dots,n_1+n_2}$)

- Feature (drift) contributions can be calculated by considering the **feature importance of the discrimination model**