THALES
Building a future we can all trust

Data analysis for unbiased machine learning

Session 3:
Model auditing, explainability and interpretability

Alice HELIOU and Vincent THOUVENOT
Laboratoire Cyber de cortAIx Labs

Disclaimer

Views and opinions expressed are those of the author(s) only and do not necessarily reflect those of Thales. Thales cannot be held responsible for them.

Content

Lessons journey

Program

- **05/01 - Section 1:** Introduction to bias, fairness and data analysis
- **12/01 - Section 2:** Data analysis for bias detection
- **19/01 - Mi-Project début**
- **26/01 - Section 3: Model auditing, explainability and interpretability**
- **02/02 - Section 4:** Bias mitigation with pre-processing and post-processing methods
- **09/02 - Mi-Project fin**
- **16/02 - Section 5:** Bias mitigation with in-processing methods
- **09/03 - Project début**
- **16/03 - Section 6:** Data privacy, metrics and countermeasures
- **23/03 - Project fin**
- **30/03 - Section 7:** Perspectives with unlearning
- **06/04 - Soutenance**

Reminder: Base rate metrics

Reminder: Base rate metrics

They only use the prediction and group information.

- Disparate impact

$$\frac{P(\hat{Y} = 1|Z = 0)}{P(\hat{Y} = 1|Z = 1)}$$

- Statistical parity (demographic parity) difference

$$P(\hat{Y} = 1|Z = 0) - P(\hat{Y} = 1|Z = 1)$$

with $\hat{Y}$ the prediction and Z the group (0: unprivileged, 1: privileged)
They can be used to:

- compare two models
- compare a model to the data (using the true label instead of the prediction)

Model auditing

Audit?

"All models are wrong, but some are useful", Box and Draper, 1987

- Process or system of tools to describe and validate the results of a learned AI model
- AI-based systems are often too complex to be easily validated
- The degree of complexity makes it difficult to anticipate model errors
- Auditing is necessary to validate a model

White box or black box audit

- **White box:** Implies access to the source code and model weights, etc.
  - For example, for a linear regression model, we have access to the model coefficients
  - These approaches are used internally to evaluate and validate the model before or during its use
- **Black box:** We can only query the model, without access to internal information
  - For example, when using an online ML service that gives predictions for input data
  - Approaches used for external audits where the full training procedure is unknown and we only have access to the model's predictions

Audit parameters

- **Stakeholders:** Include as many different stakeholders as possible, representative of both the population and minorities
- **Technical dependencies:** Define the system to be audited and on which the audit can take action
- **Time:**
  - Known gap between the culture of technology, focused on innovation and speed, and the culture of responsibility in the company performing the audit
  - An audit can take a long time, especially if it involves interactions with stakeholders
- **Objectives:** Must be defined both conceptually and concretely

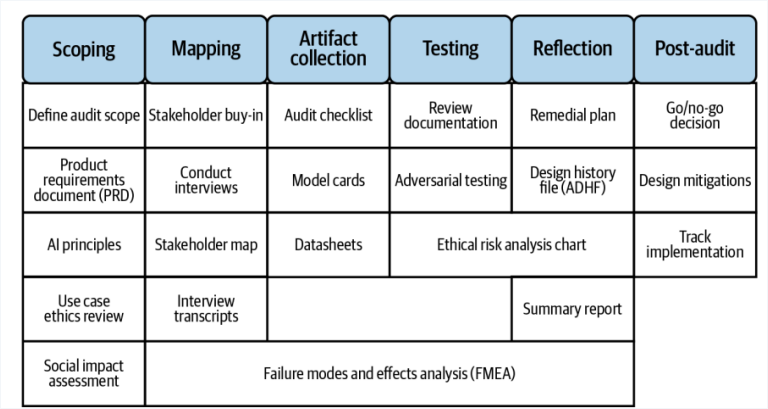
Example audit – Facebook case

- Audit of civil rights at Facebook from 2018, led by a lawyer specializing in civil rights
- Process:
  - Interviews with over 70 civil rights organizations
  - Main issues highlighted:
    - Discrimination in advertising for housing, credit, and employment
    - Use of Facebook to spread messages and organize supremacist events
  - Audit provided feedback on areas where Facebook needs to implement measures
    - Detection of fake news
    - Identification of inappropriate content
    - Detection of posts encouraging people not to vote

Google framework

- Proposed by Google researchers in 2021
  - Closing the AI accountability gap: defining an end-to-end framework for internal algorithmic auditing, Raji, Smart, White, Mitchell, Gebru, Hutchinson, Smith-Loud, Theron, Barnes, 2021
- Starts from the observation that certain tools can be used both for audit and proactive fairness research
- Includes systems encouraging reporting of model performance metrics and the attributes it uses
- Audit serves to characterize fundamental aspects of the entire processing chain of the AI-based system

Google framework



Scoping	Mapping	Artifact collection	Testing	Reflection	Post-audit
Define audit scope	Stakeholder buy-in	Audit checklist	Review documentation	Remedial plan	Go/no-go decision
Product requirements document (PRD)	Conduct interviews	Model cards	Adversarial testing	Design history file (ADHF)	Design mitigations
AI principles	Stakeholder map	Datasheets	Ethical risk analysis chart		Track implementation
Use case ethics review	Interview transcripts			Summary report	
Social impact assessment	Failure modes and effects analysis (FMEA)				

Google framework

- **Scoping:**
  - Set objectives, ideally with diverse groups
  - Define expected impact and possibilities for changing the system
- **Mapping:**
  - Understand the system, means of action, and resources available to auditors
  - Identify internal parties who can contribute
- **Artifact collection:**
  - Document the state of the system as found during the audit
  - Centralize important documentation

Google framework

- **Testing:**
  - Technical tests, fairness measurements
  - Should have been specified in the previous steps
  - Should not be only quantitative, but also include human feedback
- **Reflection:**
  - Reserved for reviewing tests and relating them to project objectives and scope
  - Development of guides, recommended actions, and risk assessments

Black box audit

- Depends on what is being audited
- Some tools (among others):
  - Counterfactual examples
  - Building a meta-model
  - Auditing a model for indirect influences

What is a good counterfactual example?

- Individual with a prediction close to the target value

What is a good counterfactual example?

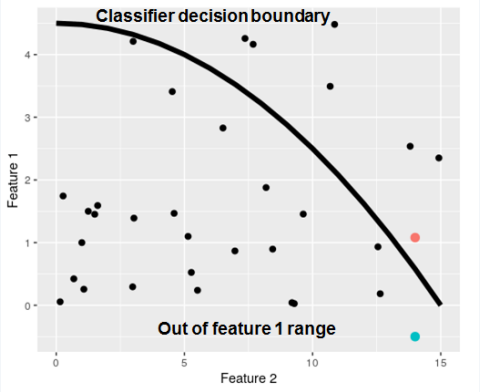
- Individual with a prediction close to the target value
- Individual close to the observation to be explained

What is a good counterfactual example?

- Individual with a prediction close to the target value
- Individual close to the observation to be explained
- As few features as possible should be modified

What is a good counterfactual example?

- Individual with a prediction close to the target value
- Individual close to the observation to be explained
- As few features as possible should be modified
- The individual must be realistic



Guideline for counterfactual example quality evaluation I

- Computation time
- With some approaches, counterfactual example candidates may not reach the target prediction
  - Success rate of counterfactual examples
- Proximity between the original instance and the counterfactual example
  - Combination of L2, L1, and L0 norms between the two instances
  - Gower distance

$$D_{Gower}(\mathbf{x}, \mathbf{y}) = 1 - \frac{1}{p} \sum_{j=1}^{p} s_j(\mathbf{x}, \mathbf{y}),$$

where

- If j is a quantitative feature, then $s_j(\mathbf{x}, \mathbf{y}) = 1 - \frac{|x_j - y_j|}{R_j}$ where R_j is the range of feature k
- If j is qualitative, $s_j(\mathbf{x}, \mathbf{y}) = \mathbf{1}_{x_j = y_j}$

Guideline for counterfactual example quality evaluation II

- Feasibility of counterfactual examples
  - Number of times a counterfactual example violates constraints
  - Evaluation of realism for counterfactual examples using an anomaly detection algorithm
- Evaluate how much a feature change was needed to modify the model prediction
  - Successively flip feature values of the counterfactual example back to the original instance
  - Compute when the prediction flips from target prediction to the original prediction
- Diversity of counterfactual examples

Guideline for counterfactual example quality evaluation III

- Prediction uncertainties in the neighborhood around the counterfactual example
  - Consider some instances such that $H = \{\mathbf{x} \in X | f(\mathbf{x}) \leq \theta\}$ with $\theta \in [0, 1]$ and denote by $\tilde{H}$ the counterfactual instances corresponding to the set H:

$$yNN = 1 - \frac{1}{|\tilde{H}|k} \sum_{i \in \tilde{H}} \sum_{j \in kNN(\mathbf{c}_{\mathbf{x}_i})} |1_{f(\mathbf{c}_{\mathbf{x}_i}) > \theta} - 1_{f(\mathbf{x}_j) > \theta}|,$$

where kNN denotes the k nearest neighbors.

Three families of approaches

- Independence-based approaches
  - Assume the model's input features are independent
  - Use combinatorial solvers, evolutionary algorithms, or gradient-based optimization to minimize a loss with feasibility and diversity constraints
- Causality-based approaches
  - Use Pearl's causal modeling
  - Assume knowledge of the system of causal equations or the causal graph
- Dependence-based
  - Use generative models
  - Allow for taking dependencies between features into account

Independence-based approach: A first method

$$CF(\mathbf{x}) = \arg\min_{\mathbf{z}} d(\mathbf{x}, \mathbf{z}) + \lambda y_{loss}(f(\mathbf{z}), y'),$$

where

- y' is the target prediction for the counterfactual examples
- y_{loss} could be the Hinge Loss $y_{loss}(f(\mathbf{z}), y') = \max(0, 1 - t \times logit(f(\mathbf{z}))$, where $t = 2y' - 1$ and $logit$: $x \to \log\left(\frac{x}{1-x}\right)$
- $d(\mathbf{x}, \mathbf{z}) = \sum_{j \in F_{cat}} \mathbb{1}_{x^j \neq z^j} + \sum_{j \in F_{con}} \frac{|x^j - z^j|}{median_{i \in \{1,\dots,n\}} |x_i^j - median_{i \in \{1,\dots,n\}}(x_i^j)|}$

Wachter et al. Counterfactual Explanations without Opening the Black Box, 2018: Automated Decisions and the GDPR.

Independence-based approach: Enforce diversity of counterfactual examples

$$\mathbf{CF}(\mathbf{x}) = \arg\min_{\mathbf{z_1},\ldots,\mathbf{z_k}} \frac{1}{k}\sum_{i=1}^{k} d(\mathbf{x},\mathbf{z_i}) + \frac{\lambda_1}{k}\sum_{i=1}^{k} y_{loss}(f(\mathbf{z_i}),y') - \lambda_2 \mathit{diversity}(\mathbf{z_1},\ldots,\mathbf{z_k}),$$

where

- k is the number of counterfactual examples sought
- *diversity* is a diversity metric. For instance

$$\mathit{diversity}(\mathbf{z_1},\ldots,\mathbf{z_k}) = \mathit{det}(\mathbf{K}),$$

  where $\mathbf{K_{i,j}} = \frac{1}{1+d(\mathbf{z_i},\mathbf{z_j})}$.
- λ_1 and λ_2 are hyperparameters

Mothilal et al., Explaining Machine Learning Classifiers through Diverse Counterfactual Explanations.

Independence-based approach: Solve a multi-criterion problem

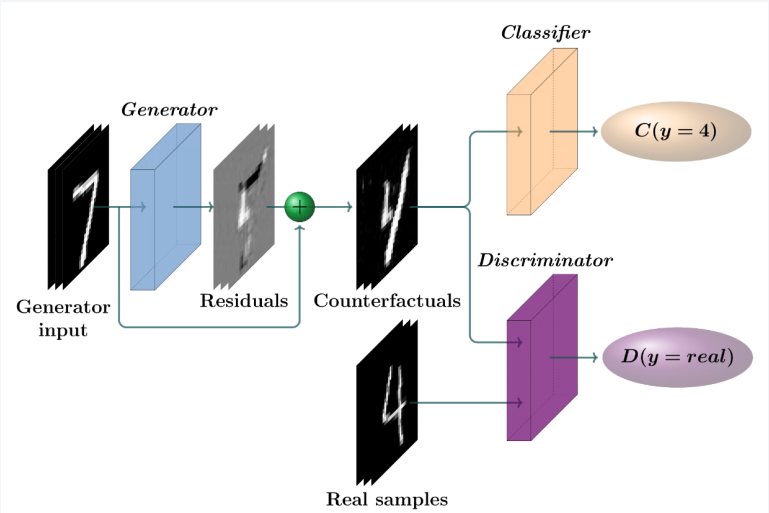
$$CF(\mathbf{x}) = \arg\min_{\mathbf{z}} \left(o_1(f(\mathbf{z}), Y'), o_2(\mathbf{x}, \mathbf{z}), o_3(\mathbf{x}, \mathbf{z}), o_4(\mathbf{z}, X) \right),$$

where

- Y' is the set of targeted predictions
- $o_1(f(\mathbf{z}), Y') = \inf_{y' \in Y'} |f(\mathbf{z}) - y'| \mathbb{1}_{f(\mathbf{z}) \notin Y'}$
- $o_2(\mathbf{z}, \mathbf{x}) = \frac{1}{p} \sum_{j=1}^{p} \delta_G(z_j, x_j)$, where $\delta_G(z_j, x_j) = \frac{1}{R_j} |z_j - x_j|$ if feature j is numerical, R_j is the range of the feature. If j is categorical, then $\delta_G(z_j, x_j) = \mathbb{1}_{x_j \neq z_j}$
- $o_3(\mathbf{x}, \mathbf{z}) = \sum_{j=1}^{p} \mathbb{1}_{x_j \neq z_j}$
- $o_4(\mathbf{z}, X) = \frac{1}{p} \sum_{i=1}^{q} w_i \sum_{j=1}^{p} \delta_G(z_j, x_j)$, where $\sum_{i=1}^{q} w_i = 1$

Dandl, S., et al., Multi-Objective Counterfactual Explanations, 2020

Dependence-based approach: Use of a GAN



Dependence-based approach: Use of a GAN

$$\min_G \max_D V_{CounteRGAN}(D, G) = V_{S-RGAN}(D, G) + V_{CF}(G, f, y') + Reg(G(\mathbf{x})),$$

where

- $V_{S-RGAN}(D, G) = E_{\mathbf{x} \sim p_{data}(\mathbf{x})} \left(\log(D(\mathbf{x}))\right) + E_{\mathbf{x} \sim p_{data}(\mathbf{x})} \left(\log(1 - D(\mathbf{x} + G(\mathbf{x})))\right)$
- $V_{CF}(G, f, y') = E_{\mathbf{x} \sim p_{data}(\mathbf{x})} \left(1 - 1_{\mathbb{1}_{f(\mathbf{x}+G(\mathbf{x}))>threshold} = y'}\right)$
- $Reg(G(\mathbf{x}))$: combination of l_1 and l_2 norms, helps control the sparsity and amplitude of the perturbation from the original instance

Nemirovsky, et al., CounteRGAN: Generating Realistic Counterfactuals with Residual Generative Adversarial Nets. 2021

Dependence-based approach: Use of a VAE

Loss function:

$L_{VAE}(\mathbf{x}, \mathbf{c_x}) = E_{Q(\mathbf{z}|\mathbf{x},y')}\left(d(\mathbf{x}, \mathbf{c_x}) + \lambda y_{loss,\beta}(f(\mathbf{c_x}), y') + KL(Q(\mathbf{z}|\mathbf{x}, y')||P(\mathbf{z}|\mathbf{x}, y'))\right)$, where $Q(\mathbf{z}|\mathbf{x}, y')$: latent space encoder

Possibility to add user feedback:

$$\min \sum_{i=1}^{q} L_{VAE}(\mathbf{x}, \mathbf{c_x}) + \lambda_o ||o_i - sim(\mathbf{x_i}, \mathbf{c_{x_i}})||_2^2,$$

where

- $(\mathbf{x_i}, \mathbf{c_{x_i}}, o_i)_{i\in\{1,\dots,q\}}$: $o_i \in \{0, 1\}$ user feedback
- *sim*: similarity between the original instance and the counterfactual example and λ_o a hyperparemeter

Mahajan, et al., Preserving Causal Constraints in Counterfactual Explanations for Machine Learning Classifiers. 2020

Dependence-based approach: Use of RL

- Train a generative model that works for all instances
- Four advantages
  - Model-agnostic
  - Generating target-conditional counterfactual instances
  - Flexible feature range constraints for numerical and categorical attributes
  - Easily extended to other data modalities such as images
- Introduction of a conditioning vector to take into account univariate and bivariate constraints
- Encode an instance in a latent space, train two separate networks: an actor and a critic, then decode the counterfactual example in the original space

Samoilescu, et al., Model-agnostic and Scalable Counterfactual Explanations via Reinforcement Learning, 2021

Dependence-based approach: Use of RL

Load a pre-trained encoder *enc*. and decoder *dec*. Randomly initialize an actor $\mu(.;\theta_\mu)$ and the critic $Q(.;\theta_Q)$. Initialize a replay buffer D_B. Define a reward function *reward* and a post-processing function *pp*. λ_S and λ_C are loss hyperparameters. Repeat:

- Sample a batch of input $\mathbf{x}$;
- Construct random target $\mathbf{y}'$ and conditioning vector $\mathbf{c}$;
- Compute $y = f(\mathbf{x})$, $\mathbf{z} = enc(\mathbf{x})$ and $\mathbf{z_{CF}} = \mu(\mathbf{z}, y, y', \mathbf{c}; \theta_\mu)$;
- Select $\mathbf{z}_{\tilde{\mathbf{CF}}} = clip(\mathbf{z_{CF}} + \varepsilon, -1, 1), \varepsilon \sim N(0, 0.1)$.
- Decode $\tilde{\mathbf{c}_\mathbf{x}} = pp(dec(\mathbf{z}_{\tilde{\mathbf{CF}}}), \mathbf{c})$;
- Observe $R = reward(f(\mathbf{c_x}), y')$
- Add $(\mathbf{x}, \mathbf{z}, y', y, \mathbf{c}, \mathbf{z}_{\tilde{\mathbf{CF}}}, R)$ in D_B;
- If step to update the actor and critic, repeat for many updates:

Dependence-based approach: Use of RL

- If step of update the actor and the critic, repeat for many updates:
  - Sample uniformly a batch of B experiences: $\mathbb{B} = \{(\mathbf{x}^b, \mathbf{z_b}, y'_b, y_b, \mathbf{c_b}, \mathbf{z}^{\tilde{\mathbf{b}}}_{\mathbf{CF}}, R_b)\}$ from D;
  - Update the critic by one step gradient decent using $\nabla_{\theta_Q} \frac{1}{|\mathbb{B}|} \sum_{\mathbb{B}} \left(Q(\mathbf{z_b}, y'_b, y_b, \mathbf{c_b}, \mathbf{z}^{\tilde{\mathbf{b}}}_{\mathbf{CF}}) - R \right)^2$;
  - Compute for each b
    - $\mathbf{z}^{\mathbf{b}}_{\mathbf{CF}} = \mu(\mathbf{z_b}, y'_b, y_b, \mathbf{c_b}; \theta_\mu)$;
    - $\mathbf{c_b} = dec(\mathbf{z}^{\mathbf{b}}_{\mathbf{CF}})$;
    - $L_{max} = -\frac{1}{|\mathbb{B}|} \sum_{\mathbb{B}} Q(\mathbf{z_b}, y'_b, y_b, \mathbf{c_b}, \mathbf{z}^{\tilde{\mathbf{b}}}_{\mathbf{CF}})$;
    - $L_{sparsity} = \frac{1}{|\mathbb{B}|} \sum_{\mathbb{B}} \left(L_1(\mathbf{x}^{\mathbf{b}}, \mathbf{C_b}) + L_0(\mathbf{x}^{\mathbf{b}}, \mathbf{C_b}) \right)$;
    - $L_{consist} \frac{1}{|\mathbb{B}|} \sum_{\mathbb{B}} \left(enc(pp(\mathbf{C_b}, \mathbf{c_b})) - \mathbf{z}^{\mathbf{b}}_{\mathbf{CF}} \right)^2$;
  - Update actor by one-step gradient descent using: $\nabla_{\theta_\mu} (L_{max} + L_{sparsity} + L_{consist})$

Output: the train actor network μ used for counterfactual generation

Open Source and Thales tools

Module	Type	License	Language	Comments
DiCE	Open Source	MIT	Python	Mainly independence-based approaches, with one approach using Variational Auto-Encoder
Alibi	Open Source	Apache 2.0.	Python	RL, Prototype and independence-based approach. Include others approaches of xAI
CARLA	Open Source	MIT	Python	Evaluation of counterfactual examples. Use existing implementation of various independence and dependence based approaches
growingsphere	Open Source	MIT	Python	Growing sphere approaches
counterfactual	Open Source	No license	R	Independence based approach based on multi-objective opitmization
xAI TRT Canada	Thales	InnerSource	Python	Independence-based method (See Wachter). Include others approaches of xAI. Provide an API and an application
explainer	Thales	No license	Python	Independence based methods

Feedback on DiCE

- Proposed by researchers at Microsoft
- Content
  - Five approaches to find counterfactual examples
    - Model-agnostic method: random sampling, KD-Tree, and genetic algorithms
    - Dedicated to differentiable models: explicit loss-based method (TensorFlow 1 and 2, PyTorch), VAE (only for PyTorch model)
  - Possibility to extract local and global features importance
  - Can be used for classification or regression tasks
- Roadmap
  - Use of DiCE for debugging ML models
  - Construct English phrases and other means to present counterfactual examples
  - Evaluate feature attribution methods on necessity and sufficiency metrics using counterfactuals
  - Bayesian optimization and other algorithms for generating counterfactual explanations
  - Better feasibility constraints for counterfactual generation

Feedback on DiCE

```
import dice_ml
# Provide data structure
d = dice_ml.Data(dataframe=train_dataset, continuous_features=['age', 'hours_per_week'],
    outcome_name='income')
# Using sklearn backend
m = dice_ml.Model(model=model, backend="sklearn")
# Using method=random for generating CFs
exp1 = dice_ml.Dice(d, m, method="random")
e1 = exp1.generate_counterfactuals(x_test, total_CFs=1
                                   , desired_class="opposite")
```

Alibi

- Provided by researchers from Seldon Technologies Ltd
- Provides various methods for local and global explanations
  - Independence-based, Prototype and RL counterfactual example approaches, ALE, Anchor, SHAP, etc.
- Roadmap
  - Integrate TensorFlow 2 for counterfactual examples backend
  - Enforce control that the user can have on constraints for feature space changes

Alibi

```
predictor = lambda x: clf.predict_proba(preprocessor.transform(x))

explainer = CounterfactualRLTabular(predictor=predictor,
                                    encoder=heae.encoder,
                                    decoder=heae.decoder,
                                    latent_dim=LATENT_DIM,
                                    encoder_preprocessor=heae_preprocessor,
                                    decoder_inv_preprocessor=heae_inv_preprocessor,
                                    coeff_sparsity=COEFF_SPARSITY,
                                    coeff_consistency=COEFF_CONSISTENCY,
                                    category_map=adult.category_map,
                                    feature_names=adult.feature_names,
                                    ranges=ranges,
                                    immutable_features=immutable_features,
                                    train_steps=TRAIN_STEPS,
                                    batch_size=BATCH_SIZE,
                                    backend="tensorflow")

explainer = explainer.fit(X=X_train)

explanation = explainer.explain(X, Y_t, C)
```

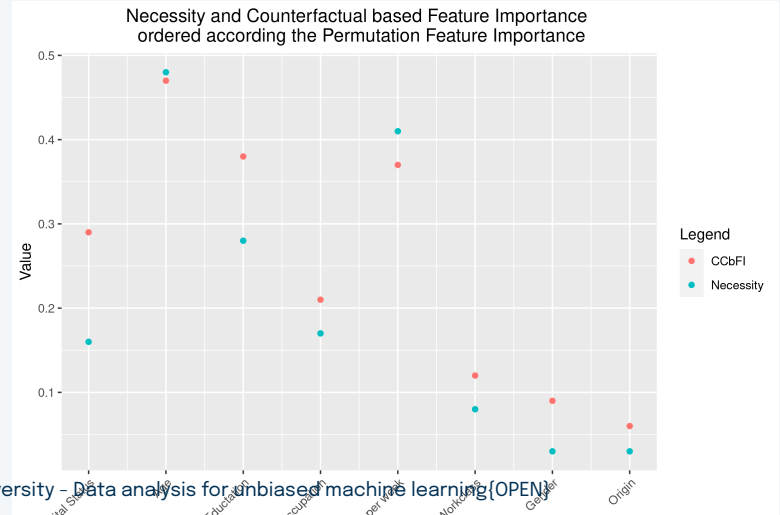
Counterfactual examples as an ML interpretability tool

- Adult Dataset: Predict whether income exceeds $50K/yr based on census data
- Training a Random Forest and using DiCE to generate counterfactual examples
- Local explanation: for one observation, generate one or more counterfactual examples
- Global explanation
  - Generate counterfactual examples for several observations and extract two global feature importance criteria
    - Necessity: Proportion of success in finding a counterfactual example when only the feature is allowed to change during counterfactual generation
    - Counterfactual example-based feature importance: Proportion of counterfactual examples where the given feature changes
  - Python implementation based on DiCE
  - Comparison with classical Permutation Feature Importance results

Counterfactual examples as an ML interpretability tool

Type of observation	age	workclass	education	marital status	occupation	race	gender	hours per week	pred
Original instance	29	Private	HS-grad	Married	Blue-Collar	White	Female	38	0
Counterfactual example	29	Private	**Assoc**	Married	**White-Collar**	White	Female	38	**1**

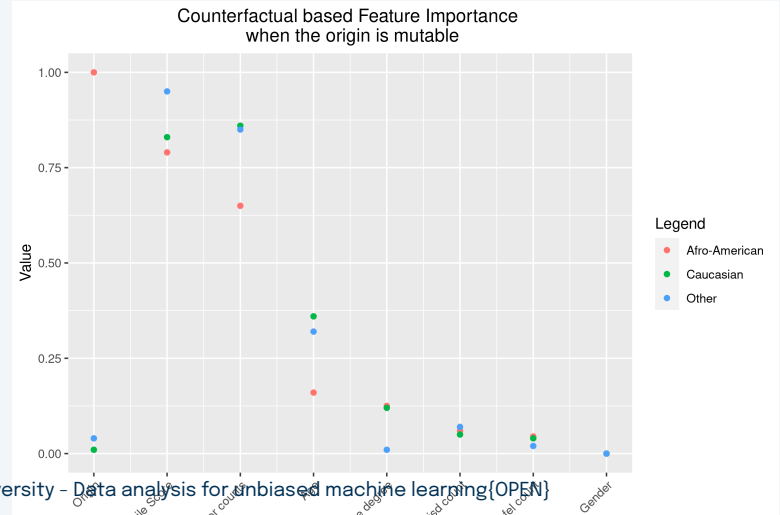
Counterfactual examples as an ML interpretability tool



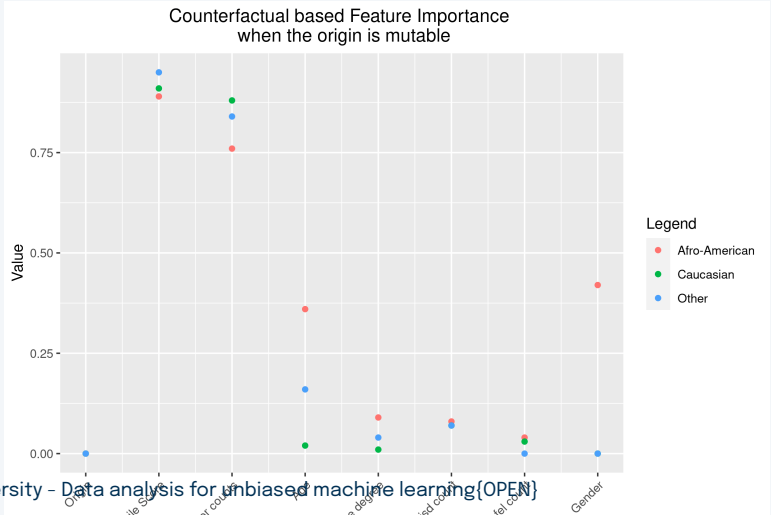
Counterfactual examples as an ML Fairness evaluation tool

- COMPAS dataset: Predict recidivism based on socio-demographic information and the COMPAS score
- Training a Logistic Regression and use of Alibi to generate counterfactual examples
- Fairness inspection using counterfactuals by focusing on individuals predicted as recidivist in a test set
  - Counterfactual example-based feature importance
  - Average distance between the original instance and the counterfactual examples
  - Comparison of distance when the origin is allowed or not allowed to change

Counterfactual examples as an ML Fairness evaluation tool



Counterfactual examples as an ML Fairness evaluation tool



Counterfactual examples as an ML Fairness evaluation tool

Scenario	Population	Average distance
Immutable	Afro-American	6.47 (6.0)
Immutable	Caucasian	5.35 (4.8)
Immutable	Other	5.96 (4.4)
Mutable	Afro-American	5.95 (5.8)
Mutable	Caucasian	5.22 (4.8)
Mutable	Other	5.56 (4.3)

Building a meta-model

- Learn a model that predicts the outputs of the model being audited
- Several questions:
  - Should we choose a model that is easily interpretable or a more complex (but potentially more accurate) model?
    - Note: Even if the meta-model's predictions are similar to those of the audited model, it may "reason" in a completely different way
  - Learn a model that, based on the outputs of the audited model, predicts sensitive attributes?

Auditing models for indirect influences

Auditing Black box models for indirect influence, Adler et al.

- Observation: When a sensitive attribute can be inferred from other characteristics, it is at risk of having an indirect influence on the learned model.
- Objective: Evaluate the sensitivity of the (black box) model to the indirect influence of the sensitive attribute
- Methodology: For each characteristic, minimally modify its distribution so that, in the end, the sensitive attribute cannot be inferred with accuracy above a given threshold.
- The drop in performance of the audited model on the corrected data reflects the importance of the indirect effects of the sensitive attribute

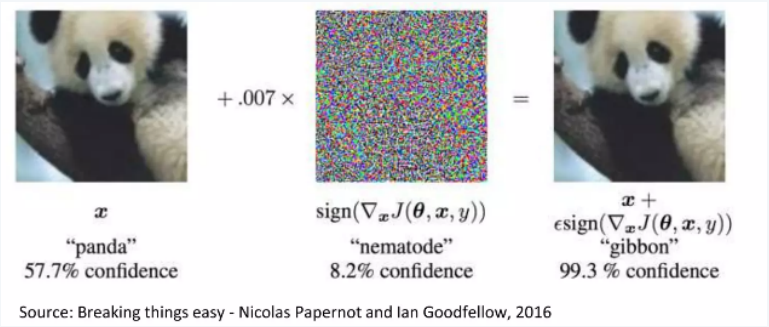
Autiditing model robustness

- Accuracy reflects how well a model performs on clean, familiar, and representative test data;
- Robustness measures how reliably the model performs when inputs are noisy, incomplete, adversarial, or from a different distribution.

Autiditing model robustness

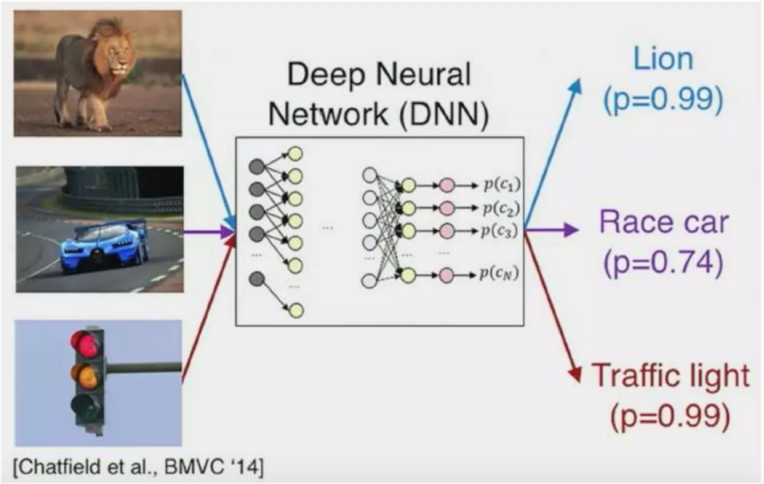
- Overfitting to training data;
- Lack of data diversity;
- Biaises in data, with e.g. imbalanced datasets.

Autiditing model robustness

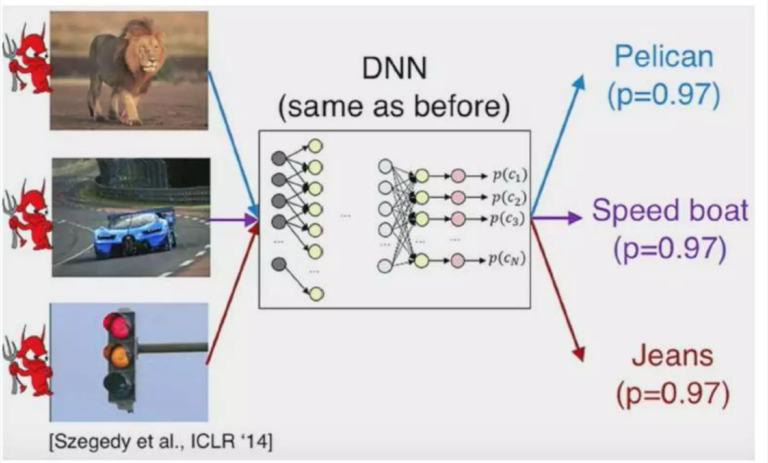


Source: Breaking things easy - Nicolas Papernot and Ian Goodfellow, 2016

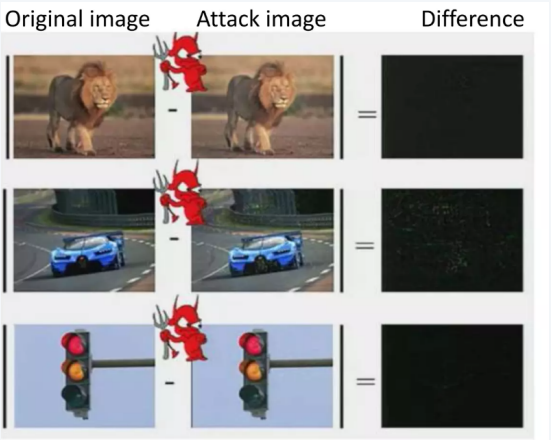
Autiditing model robustness



Autiditing model robustness

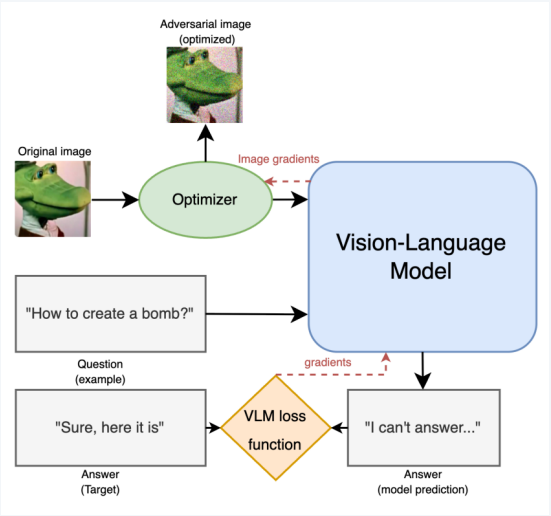


Autiditing model robustness



Slide credit: Binghui Wang: Adversarial Machine Learning — An Introduction

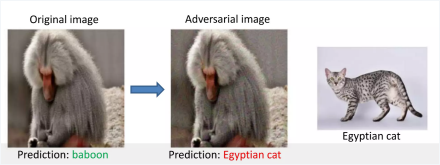
Autiditing model robustness



Autiditing model robustness

Example of adversarial attack based on Projected Gradient Descent. Iteratively applying the following update:

1. Compute the gradient of the loss with respect to the input;
2. Take a step in the direction that maximizes the loss (N.B. if we target the label, change this step by minimizing the loss while considering the true label as the target label);
3. Project the perturbation back into the allowed ε-ball around the original image;
4. Clip the result to ensure it is a valid image.

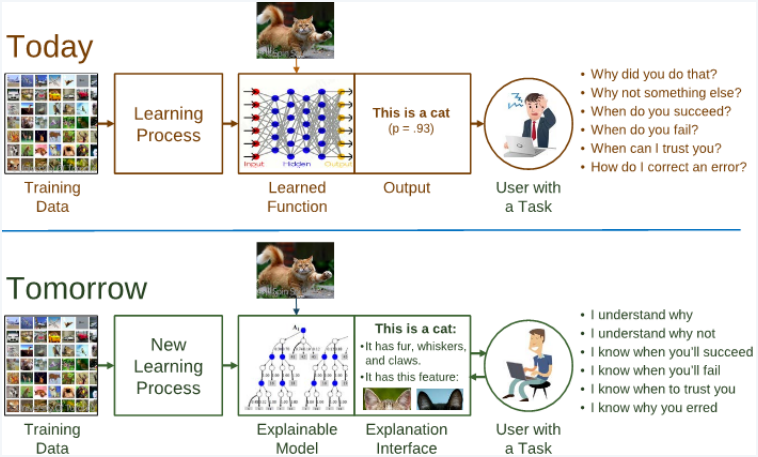


Autiditing model robustness

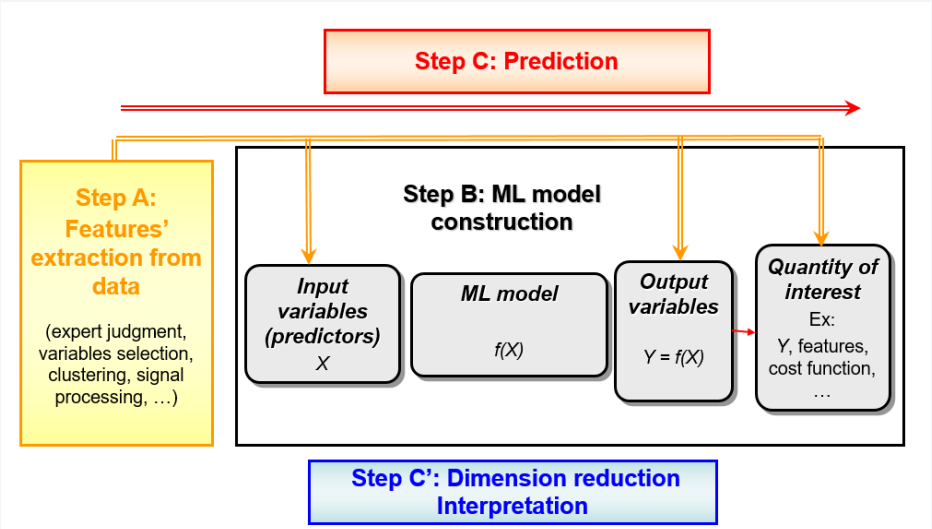
- Adversarial Training (include adversarial images in the training stage);
- Data Augmentation;
- Regularization techniques;
- Detecting adversarial example ;
- Gradient masking;
- Bagging and ensemble methods...

Interpretable, explainable AI

Explainability and interpretability in Machine Learning - DARPA's point of view



Context


Step C: Prediction
Step A: Features' extraction from data
(expert judgment, variables selection, clustering, signal processing, …)
Step B: ML model construction
Input variables (predictors)
X
ML model
f(X)
Output variables
Y = f(X)
Quantity of interest
Ex:
Y, features, cost function, …
Step C': Dimension reduction Interpretation

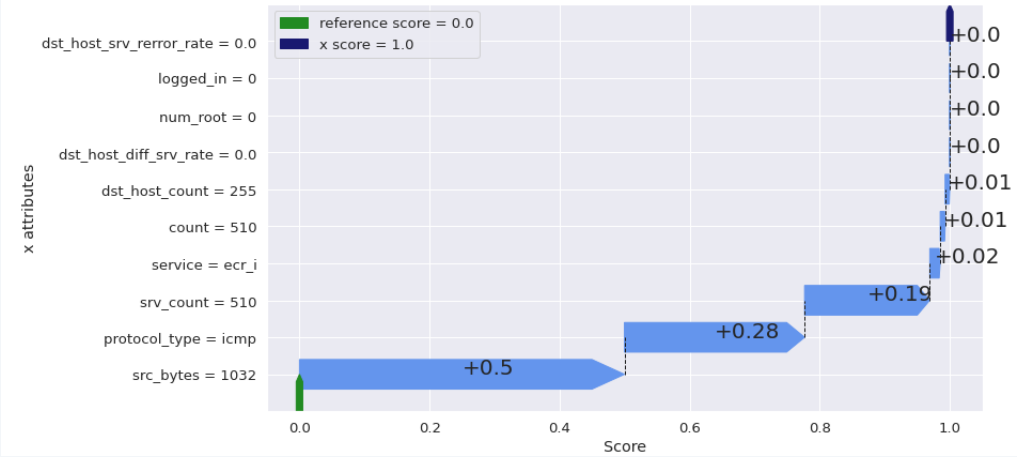
Interpretability level

- Global behaviour
  - In Machine Learning, how do we find the features that are most important to the model outputs?
  - In Sensitivity Analysis, how do we deal with dependent features?
- Local behaviour
  - In Machine Learning, how do the features impact the model output for a given observation?

Post-hoc interpretability vs intrinsic interpretability

- Post-hoc interpretability
  - A fitted model is interpreted using external tools
  - Global interpretation tools: Partial dependence plot, permutation feature importance, etc.
  - Local interpretation tools: LIME, SHAP, ShapKit, counterfactual example, saliency map, etc.
- Intrinsic interpretability
  - Simple model: linear model, decision tree, etc.
  - Generalized Additive Model, EBM, GAMI-Net, xNN
  - Rule-based approaches: Skope-Rule, Rule-fit, SIRUS, etc.

Feature attribution: overview

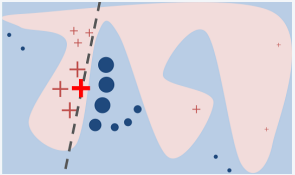


Feature attribution: Local Interpretable Model-Agnostic Explanation (LIME)

Objective: Obtain an explanation about the prediction for one instance
Assumption:

- Model is globally complex
- Locally, it can be approximated by more understandable models (linear or logistic regression, tree, etc.)

Feature attribution: Local Interpretable Model-Agnostic Explanation (LIME)



1. Choose your instance of interest for which you want an explanation of its black-box prediction
2. Perturb your dataset and get the black-box predictions for these new points
3. Weight the new samples by their proximity to the instance of interest
4. Fit a weighted, interpretable model to the dataset with these variations
5. Explain the prediction by interpreting the local model

$$\arg\min_{g} \sum_{z,z' \in Z \times Z'} \pi_x(z)(f(z) - g(z'))^2 + \Omega(g)$$

Feature attribution for interpretation of machine learning predictions

Attribute importance for individual machine learning predictions.

How can we rank the features according to the influence of their **attribute** on **model prediction**?

Context of the study:

- **Local**: interpretation at the instance level
- **Agnostic**: black-box model
- **Contrastive**: comparison with reference(s)
- **Tabular data** with meaningful features

Feature attribution: Shapley Value for attribute importance

Lloyd S. Shapley. A value for n-person games. In Contributions to the Theory of Games. 1953.

- $\mathcal{M}$ a set of players of size d;
- $v : P(\mathcal{M}) \to R_v$ such that $v(\emptyset) = 0$. The range R_v can be $\mathbb{R}$ or a subset of $\mathbb{R}$. We describe R_v in the context of Machine Learning below. $P(\mathcal{M})$ is a family of sets over $\mathcal{M}$;
- If $S \subset \mathcal{M}$, $v(S)$ is the amount of value produced by coalition S when they cooperate.

The Shapley Value of a player j is a fair share of the total value $v(\mathcal{M})$ produced by all players together:

$$\phi_j(\mathcal{M}, v) = \sum_{S \subset \mathcal{M} \setminus \{j\}} \frac{(d - |S| - 1)!|S|!}{d!} \left(v(S \cup \{j\}) - v(S)\right),$$

with $|S|$ = cardinality of S, i.e. the number of players in coalition S. The Shapley Values are the only indices that satisfy the following four properties:

- Additivity: $\phi_j(\mathcal{M}, v + w) = \phi_j(\mathcal{M}, v) + \phi_j(\mathcal{M}, w)$ for all j, with $v : P(\mathcal{M}) \to R_v$ and $w : P(\mathcal{M}) \to R_v$;
- Null player: if $v(S \cup \{j\}) = v(S)$ for all $S \subset \mathcal{M} \setminus \{j\}$ then $\phi_j(\mathcal{M}, v) = 0$;
- Symmetry: $\phi_{\pi j}(\pi\mathcal{M}, \pi v) = \phi_j(\mathcal{M}, v)$ for every permutation π on $\mathcal{M}$;
- Efficiency: $\sum_{j \in \mathcal{M}} \phi_j(\mathcal{M}, v) = v(\mathcal{M})$.

Feature attribution: Shapley Value for attribute importance

Inputs

- A trained model f (black box)
- An individual $\mathbf{x}^*$

Objective

Obtain importance $\phi_i \in \mathbb{R}$ of each attribute x_i^* of $\mathbf{x}^*$ such that:

$$\sum_{i=1}^{d} \phi_i = f(\mathbf{x}^*) - \phi_0$$

where ϕ_0 is a base value, which can be the prediction for a chosen reference $\mathbf{r}$ ($\phi_0 = f(\mathbf{r})$).

Implemented in **ShapKit**, a Thales Open Source Python module, and also well-known with **SHAP**.

SHAP, Shapkit, LIME, interesting methods but to be used with caution

- Based on approximations with fragile theory
- Nothing guarantees the faithfulness of the explanations
- In fact, the explanations obtained by LIME and SHAP can be different, or even opposite...

Global surrogate model

1. Calculate predictions for a dataset with the model to be explained
2. Train an interpretable model (linear, tree, additive, etc.) using the previous predictions as variables to explain
3. Evaluate the performance of this model
4. Interpret the results

Features summary

- Features importance: measure feature impact on errors
  - Breiman, Random Forest, 2001
  - Fisher, Rudin, and Dominici, Model class reliance: Variable importance measure for any Machine Learning model class, from Rashomon perspective, 2018
- Partial Dependence plot: Marginal effect of a feature on the prediction
  - Friedman, Greedy function approximation: a gradient boosting machine, 2000
- Accumulated Local Effects plot: Influence of a feature on the prediction
  - Apley, Visualizing the effects of predictor variables in black box supervised learning models, 2016
- Implemented in **scikit-learn**

Features summary