Target Leakage in ML

Yuriy Guts
Machine Learning Engineer
Automated ML, time series forecasting, NLP

Lecturer
AI, Machine Learning, Summer/Winter ML Schools

Compete sometimes
Currently hold an Expert rank, top 2% worldwide
Follow my presentation and code at:
https://github.com/YuriyGuts/odsc-target-leakage-workshop
Leakage in a Nutshell

- Data known at prediction time
- Data unknown at prediction time

contamination

prediction point
Leakage in a Nutshell

Training on **contaminated data** leads to overly optimistic expectations about model performance in production.
“But I always validate on random K-fold CV. I should be fine, right?”
They suspect nothing
Leakage can happen anywhere during the project lifecycle
Leakage in Data Collection
Where is the leakage?

<table>
<thead>
<tr>
<th>EmployeeID</th>
<th>Title</th>
<th>ExperienceYears</th>
<th>MonthlySalaryGBP</th>
<th>AnnualIncomeUSD</th>
</tr>
</thead>
<tbody>
<tr>
<td>315981</td>
<td>Data Scientist</td>
<td>3</td>
<td>5,000.00</td>
<td>78,895.44</td>
</tr>
<tr>
<td>4691</td>
<td>Data Scientist</td>
<td>4</td>
<td>5,500.00</td>
<td>86,784.98</td>
</tr>
<tr>
<td>23598</td>
<td>Data Scientist</td>
<td>5</td>
<td>6,200.00</td>
<td>97,830.35</td>
</tr>
</tbody>
</table>
Target is a function of another column

<table>
<thead>
<tr>
<th>EmployeeID</th>
<th>Title</th>
<th>ExperienceYears</th>
<th>MonthlySalaryGBP</th>
<th>AnnualIncomeUSD</th>
</tr>
</thead>
<tbody>
<tr>
<td>315981</td>
<td>Data Scientist</td>
<td>3</td>
<td>5,000.00</td>
<td>78,895.44</td>
</tr>
<tr>
<td>4691</td>
<td>Data Scientist</td>
<td>4</td>
<td>5,500.00</td>
<td>86,784.98</td>
</tr>
<tr>
<td>23598</td>
<td>Data Scientist</td>
<td>5</td>
<td>6,200.00</td>
<td>97,830.35</td>
</tr>
</tbody>
</table>

The target can have different formatting or measurement units in different columns. Forgetting to remove the copies will introduce target leakage.

Check out the example: example-01-data-collection.ipynb
## Where is the leakage?

<table>
<thead>
<tr>
<th>SubscriberID</th>
<th>Group</th>
<th>DailyVoiceUsage</th>
<th>DailySMSUsage</th>
<th>DailyDataUsage</th>
<th>Gender</th>
</tr>
</thead>
<tbody>
<tr>
<td>24092091</td>
<td>M18-25</td>
<td>15.31</td>
<td>25</td>
<td>135.10</td>
<td>0</td>
</tr>
<tr>
<td>4092034091</td>
<td>F40-60</td>
<td>35.81</td>
<td>3</td>
<td>5.01</td>
<td>1</td>
</tr>
<tr>
<td>329815</td>
<td>F25-40</td>
<td>13.09</td>
<td>32</td>
<td>128.52</td>
<td>1</td>
</tr>
<tr>
<td>94721835</td>
<td>M25-40</td>
<td>18.52</td>
<td>21</td>
<td>259.34</td>
<td>0</td>
</tr>
</tbody>
</table>
Feature is an aggregate of the target

<table>
<thead>
<tr>
<th>SubscriberID</th>
<th>Group</th>
<th>DailyVoiceUsage</th>
<th>DailySMSUsage</th>
<th>DailyDataUsage</th>
<th>Gender</th>
</tr>
</thead>
<tbody>
<tr>
<td>24092091</td>
<td>M18-25</td>
<td>15.31</td>
<td>25</td>
<td>135.10</td>
<td>0</td>
</tr>
<tr>
<td>4092034091</td>
<td>F40-60</td>
<td>35.81</td>
<td>3</td>
<td>5.01</td>
<td>1</td>
</tr>
<tr>
<td>329815</td>
<td>F25-40</td>
<td>13.09</td>
<td>32</td>
<td>128.52</td>
<td>1</td>
</tr>
<tr>
<td>94721835</td>
<td>M25-40</td>
<td>18.52</td>
<td>21</td>
<td>259.34</td>
<td>0</td>
</tr>
</tbody>
</table>

E.g., the data can have derived columns created after the fact for reporting purposes
**Where is the leakage?**

<table>
<thead>
<tr>
<th>Education</th>
<th>Married</th>
<th>AnnualIncome</th>
<th>Purpose</th>
<th>LatePaymentReminders</th>
<th>IsBadLoan</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Y</td>
<td>80k</td>
<td>Car Purchase</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>N</td>
<td>120k</td>
<td>Small Business</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>Y</td>
<td>85k</td>
<td>House Purchase</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>N</td>
<td>72k</td>
<td>Marriage</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Database records get overwritten as more facts become available. But these later facts won’t be available at prediction time.
Leakage in Feature Engineering
My model is sensitive to feature scaling...

- **Training Set**
- **StandardScaler**
- **Partition into CV/Holdout**
  - **Train**
  - **Evaluate**
My model is sensitive to feature scaling...

Training Set → StandardScaler → Partition into CV/Holdout

OOPS. WE'RE LEAKING THE TEST FEATURE DISTRIBUTION INFO INTO THE TRAINING SET

Check out the example: example-02-data-prep.ipynb
Removing leakage in feature engineering

Obtain feature engineering/transformation parameters only on the training set.
Apply them to transform the evaluation sets (CV, holdout, backtests, ...)

- Training Set
- Partition
- StandardScaler (fit) → Train
- StandardScaler (transform) → Evaluate
  - train
  - eval

Apply mean and std obtained on train
Learn DTM columns from the training set only, then transform the evaluation sets (avoid leaking possible out-of-vocabulary words into the training pipeline)

**Categoricals:**
Create mappings on the training set only, then transform the evaluation sets (avoid leaking cardinality/frequency info into the training pipeline)
Leakage in Partitioning
Our full paper on Deep Learning for pneumonia detection on Chest X-Rays.
@pranavrajpurkar @jeremy_irvin16 @mattlungrenMD  arxiv.org/abs/1711.05225
3. Data

3.1. Training

We use the ChestX-ray14 dataset released by Wang et al. (2017) which contains 112,120 frontal-view X-ray images of 30,805 unique patients. Wang et al. (2017) annotate each image with up to 14 different thoracic pathology labels using automatic extraction methods on radiology reports. We label images that have pneumonia as one of the annotated pathologies as positive examples and label all other images as negative examples for the pneumonia detection task. We randomly split the entire dataset into 80% training, and 20% validation.

Before inputting the images into the network, we downscale the images to $224 \times 224$ and normalize based on the mean and standard deviation of images in the ImageNet training set. We also augment the training data with random horizontal flipping.

https://twitter.com/AndrewYNg/status/931026446717296640
Group Leakage

Nick Roberts
@nizkroberts

Relying to @AndrewYNg @pranavrajpurkar and 2 others

Were you concerned that the network could memorize patient anatomy since patients cross train and validation? "ChestX-ray14 dataset contains 112,120 frontal-view X-ray images of 30,805 unique patients. We randomly split the entire dataset into 80% training, and 20% validation."

Oops. There are four times more unique images than patients
For the pneumonia detection task, we randomly split the dataset into training (28744 patients, 98637 images), validation (1672 patients, 6351 images), and test (389 patients, 420 images). There is no patient overlap between the sets.

<table>
<thead>
<tr>
<th>Pathology</th>
<th>Wang et al. (2017)</th>
<th>Yao et al. (2017)</th>
<th>CheXNet (ours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atelectasis</td>
<td>0.716</td>
<td>0.772</td>
<td>0.8209</td>
</tr>
<tr>
<td>Cardiomegaly</td>
<td>0.807</td>
<td>0.904</td>
<td>0.9048</td>
</tr>
<tr>
<td>Effusion</td>
<td>0.784</td>
<td>0.859</td>
<td>0.8831</td>
</tr>
<tr>
<td>Infiltration</td>
<td>0.609</td>
<td>0.695</td>
<td>0.7204</td>
</tr>
<tr>
<td>Mass</td>
<td>0.706</td>
<td>0.792</td>
<td>0.8618</td>
</tr>
<tr>
<td>Nodule</td>
<td>0.671</td>
<td>0.717</td>
<td>0.7766</td>
</tr>
<tr>
<td>Pneumonia</td>
<td>0.633</td>
<td>0.713</td>
<td>0.7632</td>
</tr>
<tr>
<td>Pneumothorax</td>
<td>0.806</td>
<td>0.841</td>
<td>0.8932</td>
</tr>
<tr>
<td>Consolidation</td>
<td>0.708</td>
<td>0.788</td>
<td>0.7939</td>
</tr>
<tr>
<td>Edema</td>
<td>0.835</td>
<td>0.882</td>
<td>0.8932</td>
</tr>
<tr>
<td>Emphysema</td>
<td>0.815</td>
<td>0.829</td>
<td>0.9260</td>
</tr>
<tr>
<td>Fibrosis</td>
<td>0.769</td>
<td>0.767</td>
<td>0.8044</td>
</tr>
<tr>
<td>Pleural Thickening</td>
<td>0.708</td>
<td>0.765</td>
<td>0.8138</td>
</tr>
<tr>
<td>Hernia</td>
<td>0.767</td>
<td>0.914</td>
<td>0.9387</td>
</tr>
</tbody>
</table>

Paper v1 (AUC)  

Paper v3 (AUC)
The Cold Start Problem

Observe:
- Illinois
- Oklahoma
- Texas

Predict:
- North Carolina
Group Partitioning, Out-of-Group Validation

Fold 1
- Illinois
- Oklahoma

Fold 2
- Illinois
- Texas

Fold 3
- Oklahoma
- Texas

Training:

Validation:
Leakage in oversampling / augmentation

Original Data → Oversampling or Augmentation → Random Partitioning → Train, Evaluate
Leakage in oversampling / augmentation

Original Data

Oversampling or Augmentation

Random Partitioning

Train

Evaluate

Pairwise Data (features for A and B)

Augmentation (AB → BA)

Random Partitioning

Train

Evaluate
Leakage in oversampling / augmentation

**Original Data**

**Oversampling or Augmentation**

**Pairwise Data** (features for A and B)

**Augmentation** (AB → BA)

**Random Partitioning**

**Train**

**Evaluate**

**OOPS. WE MAY GET COPIES SPLIT BETWEEN TRAINING AND EVALUATION**
Leakage in oversampling / augmentation

Original Data

Oversampling or Augmentation

Pairwise Data (features for A and B)

Augmentation (AB → BA)

Random Partitioning

Train

Evaluate

Random Partitioning

Train

Evaluate

First partition, then augment the training data.
Random Partitioning for Time-Aware Models
Random Partitioning for Time-Aware Models

Oops. We’re training on the future to predict the past.
Out-of-Time Validation (OTV)

Backtest 2

Backtest 1

Holdout
Leakage in Training & Tuning
Reusing a CV split for multiple tasks

Fold 1

Fold 2

Fold 3

Fold 4

Fold 5

Feature selection, hyperparameter tuning, model selection...
Reusing a CV split for multiple tasks

Fold 1: V
Fold 2: V
Fold 3: V
Fold 4: V
Fold 5: V

Feature selection, hyperparameter tuning, model selection...

OOPS. CAN OVERFIT VALIDATION FOLDS
BETTER USE DIFFERENT SPLITS FOR DIFFERENT TASKS
Model stacking on in-sample predictions

Model 1
- Train
- Predict

Model 2
- Train
- Predict

2nd Level Model
- Train
Model stacking on in-sample predictions

OOPS. WILL LEAK THE TARGET IN THE META-FEATURES
Better way to stack

Model 1
Train
Predict

Model 2
Train
Predict

2nd Level Model
Train

Compute all meta-features only out-of-fold
Leakage in Competitive ML
Case Studies

- Removing customer/user IDs does not necessarily mean data anonymization

- Anonymizing feature names does not mean anonymization either
  (Kaggle: Santander Value Prediction competition, 2018)

- Target can sometimes be recovered using side channels or external datasets
  (Kaggle: Dato “ Truly Native?” competition, 2015)

- Overrepresented minority class opens possibilities for reverse engineering
  (Kaggle: Quora Question Pairs competition, 2017)
Leakage prevention checklist (not exhaustive!)

- Split the holdout away immediately and do not preprocess it in any way before final model evaluation.
- Make sure you have a data dictionary and understand the meaning of every column, as well as unusual values (e.g. negative sales) or outliers.
- For every column in the final feature set, try answering the question: “Will I have this feature at prediction time in my workflow? What values can it have?”
- Figure out preprocessing parameters on the training subset, freeze them elsewhere.
- Treat feature selection, model tuning, model selection as separate “machine learning models” that need to be validated separately.
- Make sure your validation setup represents the problem you need to solve with the model.
- Check feature importance and prediction explanations: do top features make sense?
Thank you!

yuriy.guts@gmail.com

linkedin.com/in/yuriyguts