Data Leakage in Notebooks: Static Detection and Better Processes

Chenyang Yang, Rachel Brower-Sinning, Grace A. Lewis, Christian Kästner

Carnegie Mellon University
Why ML Models Fail in Production?

ML models

High test accuracy

Software systems

Low production accuracy
When is Test Accuracy not Reliable?

Non-representative test data

African Bush Elephant

North America Wild Horse

Low production accuracy
When is Test Accuracy not Reliable?

Data leakage: **leak test data** into model development through repeated evaluation, pre-processing, and dependency

We use **static analysis** to detect data leakage in ~**281k notebooks** ~**81k GitHub repositories** created in Sep. 2021
2 top Kaggle competitions
Principle of Independent Evaluation

Model development

Model training

Model selection

Training

Validation

Testing

(validation holdout sample)

(testing holdout sample)

Independent evaluation
Data Leakage #1: through Repeated Evaluation

Models overfit to test data after repeated evaluation

Independent evaluation

Inflated test accuracy!

Found by our tool in 
~18% notebooks
Data Leakage #2: through Preprocessing

Peeking at test data in competitions is common

| Training data | the | red | dog | cat | eats | food |
|---------------|-----|-----|-----|-----|------|------|
| 1. the red dog | 1   | 1   | 1   | 0   | 0    | 0    |
| 2. cat eats dog| 0   | 0   | 1   | 1   | 1    | 0    |
| 3. dog eats food| 0   | 0   | 1   | 0   | 1    | 1    |
| 4. red cat eats| 0   | 1   | 0   | 1   | 1    | 0    |

Test data:

- Different distribution
- Unknown words

Inflated test accuracy!

Found by our tool in ~12% notebooks
Data Leakage #3: through Dependency

Data augmentation could introduce dependency

Train/test dependency

Inflated test accuracy!

Found by our tool in ~6% notebooks
Data Leakage is Prevalent in Practice

~281k notebooks from GitHub and Kaggle

~30% GitHub notebooks have data leakage issues

- 33% assignments (keyword: ‘assignment’, ‘homework’)
- 20% popular notebooks (>=10 stars)
- 16% tutorials (keyword: ‘this tutorial’)

55% competition solutions leak through preprocessing
Leakage Exhibits Non-local Patterns

Leakage and training are often far apart
span >20% of the whole notebook in >50% cases

Hard for manual detection!
Could we statically detect data leakage?
Statically Detecting Data Leakage

Front-end
- Raw Python
- Python (SSA)
- Type Inference
- Datalog Facts

Back-end
- Datalog Facts
- API Specs
- Pointer Analysis
- Data-flow Analysis
- Related Data Analysis
- Data-Model Mappings
- Dataset Transformations
- Leakage Detection

flow-sensitive
2-call-site-sensitive
Walkthrough Example

```python
import pandas as pd
from sklearn.feature_selection import SelectPercentile, chi2
from sklearn.model_selection import LinearRegression, train_test_split
data = pd.read_csv('data.csv')
X_raw = data.drop('label', axis=1)
y = data['label']

select = SelectPercentile(chi2, percentile=50)
select.fit(X_raw)
X = select.transform(X_raw)

X_train, y_train, X_test, y_test = train_test_split(X, y)
lr = LinearRegression()
lr.fit(X_train, y_train)
lr_score = lr.score(X_test, y_test)
```

- Load data
- Feature selection
- Model training & evaluation
Test Data is Used for Feature Selection

```python
import pandas as pd
from sklearn.feature_selection import SelectPercentile, chi2
from sklearn.model_selection import LinearRegression, train_test_split
data = pd.read_csv('data.csv')
X_raw = data.drop('label', axis=1)
y = data['label']

select = SelectPercentile(chi2, percentile=50)
select.fit(X_raw)
X = select.transform(X_raw)

X_train, y_train, X_test, y_test = train_test_split(X, y)
lr = LinearRegression()
lr.fit(X_train, y_train)
lr_score = lr.score(X_test, y_test)
```
When is an Operation Leakage-inducing?

Computing across rows could lead to leakage
When is an Operation Leakage-inducing?

import pandas as pd
from sklearn.feature_selection import SelectPercentile, chi2
from sklearn.model_selection import LinearRegression, train_test_split

data = pd.read_csv('data.csv')
X_raw = data.drop('label', axis=1)
y = data['label']

select = SelectPercentile(chi2, percentile=50)
select.fit(X_raw)
X = select.transform(X_raw)

X_train, y_train, X_test, y_test = train_test_split(X, y)
lr = LinearRegression()
lr.fit(X_train, y_train)
lr_score = lr.score(X_test, y_test)

|   | col1 | col2 |   | col1 |
|---|------|------|---|------|
| 1 | 3    | 4    | 1 | 3    |
| 2 | 0    | 1    | 2 | 0    |
| 3 | 6    | 3    | 3 | 6    |
| 4 | -3   | 6    | 4 | -3   |
| 5 | 2    | 1    | 5 | 2    |

Computing each row independently is safe
When is an Operation Leakage-inducing?

```
import pandas as pd
from sklearn.feature_selection import SelectPercentile, chi2
from sklearn.model_selection import LinearRegression, train_test_split
data = pd.read_csv('data.csv')
X_raw = data.drop('label', axis=1)
y = data['label']

select = SelectPercentile(chi2, percentile=50)
select.fit(X_raw)
X = select.transform(X_raw)
X_train, y_train, X_test, y_test = train_test_split(X, y)
lr = LinearRegression()
lr.fit(X_train, y_train)
lr_score = lr.score(X_test, y_test)
```

Computing each row independently is safe
Reduce-like Operations could Lead to Leakage

| col1 | col2 |
|------|------|
| 1    | 3    | 4    |
| 2    | 0    | 1    |
| 3    | 6    | 3    |
| 4    | -3   | 6    |
| 5    | 2    | 1    |

reduce

| col1 | col2 |
|------|------|
| 1    | 3    | 4    |
| 2    | 0    | 1    |
| 3    | 6    | 3    |
| 4    | -3   | 6    |
| 5    | 2    | 1    |

map

| col1 | col2 |
|------|------|
| 1    | 3    |
| 2    | 0    |
| 3    | 6    |
| 4    | -3   |
| 5    | 2    |

filter

| col1 |
|------|
| 1    |
| 2    |
| 3    |

reduce, map, filter
Detecting Data Leakage with Data-flow

```python
import pandas as pd
from sklearn.feature_selection import SelectPercentile, chi2
from sklearn.model_selection import LinearRegression, train_test_split
data = pd.read_csv('data.csv')
X_raw = data.drop('label', axis=1)
y = data['label']

select = SelectPercentile(chi2, percentile=50)
select.fit(X_raw)
X = select.transform(X_raw)

X_train, y_train, X_test, y_test = train_test_split(X, y)
lr = LinearRegression()
lr.fit(X_train, y_train)
lr_score = lr.score(X_test, y_test)
```

*There are more subtleties in tracking data-flow and determining whether two datasets are related: see our paper for details.*
Implementation

Front-end
- Raw Python → Python (SSA) → Type Inference → Datalog Facts

Back-end
- Datalog Facts
- API Specs
- Pointer Analysis → Data-flow Analysis → Related Data Analysis
- Data-Model Mappings
- Dataset Transformations → Leakage Detection

Related Data Analysis

2-call-site-sensitive
Evaluation: Accuracy & Efficiency

93% accuracy from comparing results with 100 manually labeled sample notebooks

3 seconds (avg.) of analysis on a standard desktop with Intel Xeon CPU and 32GB memory
Recall: Data Leakage is Prevalent in Practice

~30% GitHub notebooks have data leakage issues
  33% assignments
  20% popular notebooks
  16% tutorials

55% competition solutions leaks through preprocessing
Could we avoid data leakage in practice?
Data Leakage: Better Processes

Static analysis as **warnings** in notebooks

```python
import pandas as pd
from sklearn.feature_selection import SelectPercentile, chi2
from sklearn.model_selection import LinearRegression, train_test_split
data = pd.read_csv('data.csv')
X_raw = data.drop('label', axis=1)
y = data['label']

select = SelectPercentile(chi2, percentile=50)
select.fit(X_raw)  # data leakage (preprocessing)
X = select.transform(X_raw)

X_train, y_train, X_test, y_test = train_test_split(X, y)
lr = LinearRegression()
lr.fit(X_train, y_train)  # train
lr_score = lr.score(X_test, y_test)  # test
```
Data Leakage: Better Processes

Limited access to test label/data

Do not share test data
Data Leakage: Better Processes

API Design to prevent leakage

```python
X_selected = SelectKBest(k=25).fit_transform(X, y)
X_train, X_test, y_train, y_test = train_test_split(
    X_selected, y, random_state=42)
gbc = GradientBoostingClassifier(random_state=1)
gbc.fit(X_train, y_train)
y_pred = gbc.predict(X_test)
accuracy_score(y_test, y_pred)
```

```python
from sklearn.pipeline import make_pipeline
X_train, X_test, y_train, y_test = train_test_split(
    X, y, random_state=42)
pipeline = make_pipeline(SelectKBest(k=25),
    GradientBoostingClassifier(random_state=1))
pipeline.fit(X_train, y_train)
y_pred = pipeline.predict(X_test)
accuracy_score(y_test, y_pred)
```
Takeaways

Data Leakage is **prevalent** in practice (in ~30% GitHub notebooks)

Static analysis and better process designs could help

Contact me & Read the paper!
Bonus: Practical Impact of Data Leakage

Often marginal accuracy differences

Data leakage makes models “learn” from random data

Data leakage leads to flawed experiments and wasted time

```python
1 import numpy as np
2 # generate random data
3 n_samples, n_features, n_classes = 200, 10000, 2
4 rng = np.random.RandomState(42)
5 X = rng.standard_normal((n_samples, n_features))
6 y = rng.choice(n_classes, n_samples)
7
8 # leak test data through feature selection
9 X_selected = SelectKBest(k=25).fit_transform(X, y)
10
11 X_train, X_test, y_train, y_test = train_test_split(
12    X_selected, y, random_state=42)
13 gbc = GradientBoostingClassifier(random_state=1)
14 gbc.fit(X_train, y_train)
15
16 y_pred = gbc.predict(X_test)
17 accuracy_score(y_test, y_pred)
18 # expected accuracy ~0.5; reported accuracy 0.76
```