# IN-STK5000 Project 2

Fall 2023

Helene Bøsei Olsen, Sander Finnset Ørnes,
Even Tronstad, Andreas Christian Poole

# Agenda

- Data Leakage
- Reproducibility
- Pipeline
- Model evaluation - performance metrics
- Ranges of performance metrics
- Privacy

# Baseline model: Data leakage

We considered data leakage through all steps in our analysis:

**Scope reduction:**

Deleted samples based on gender and race

- Based on the whole data set. Danger!

**Train-Test split:**

- Done very early
- Seperate data sets through most of the process

**Data Analysis** Primarily performed only on training set

  - Outliers, correlations
- For missing data performed on whole data set
  - Should not cause data leakage

# Data leakage

**Data manipulation:**

- Do operations on training and test set in separate steps
- Separation of data to be corrected and data used to correct (train data)

Example:

```
train = handle_outliers(train, train_outlier_bounds)
test = handle_outliers(test, train_outlier_bounds)
```

**Feature selection:**

- Mostly used non data-driven criteria:
  - Common sense and cost of collection
- Temperature removed based on low variance (train data)

# Data leakage

**Classification:**

- Simple classifier with manually set hyperparameters.
  - Might have been unconsciously set based on info from test set
    Danger!

**Correction of data leakage issues:**

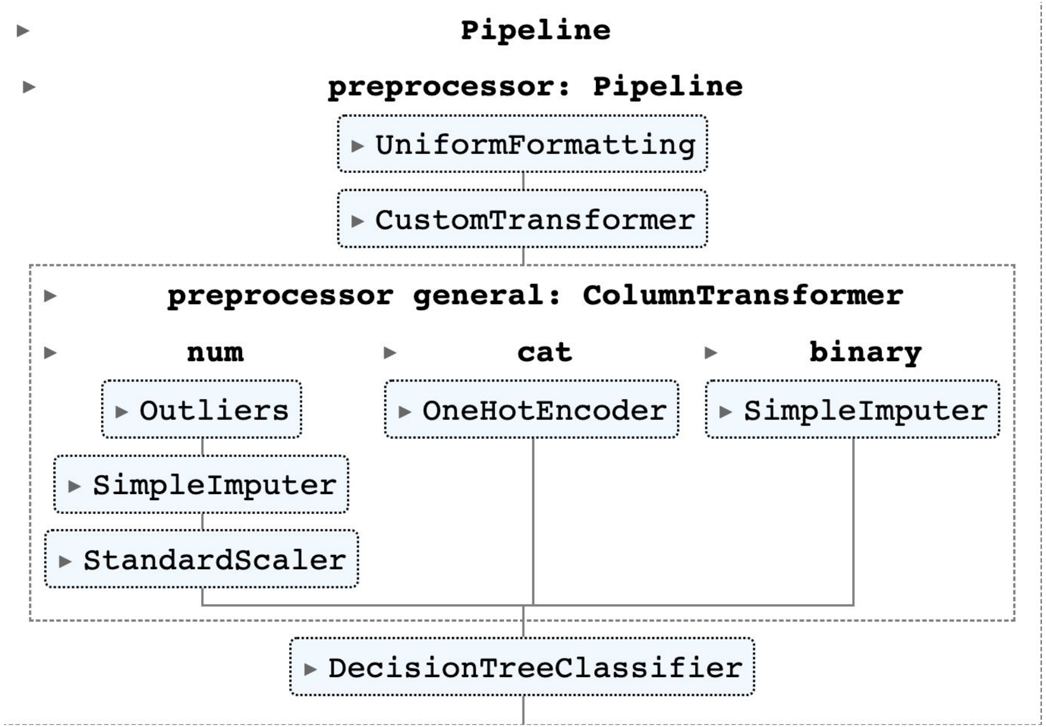
- Not delete non whites and females
- Hyperparameter tuning with grid search cross validation

# New implementation: Pipeline

- Never edits the actual data set.
  - all preprocessing and formatting in the pipeline.
- Preprocessing
  - imputes and scales without using test set by design.
- No scope reduction.
  - include gender and race (one hot encoder with ‘other’ if prevalence less than 0.1)
- Hyperparameter tuning with grid search cross validation.
  - on max_depth and complexity parameter ccp_alpha.

# Pipeline Structure

- Custom transformers imputes Obesity and Polydipsia
- General preprocessing
- Decision tree as classifier
- Finally, grid search on tree depths and complexity parameter (not displayed)



# Reproducibility

Ensure reproducibility through the following steps:

- All code and documentation available on GitHub
  - Installation script for easy usage
- Set a global seed
  - Separate seed for privacy
- requirements.txt file available with versions of packages / libraries etc.
- Thorough instructions on how to run experiments
- Simplifying code with pipelines
- Pytest for reproducibility

# Model evaluation - performance metrics

Business case recap:

- Work for the Public health authorities
- Diagnosing and treatment handled by the Private health service
- Machine Learning system on website for evaluation of diabetes risk
- Goal: Get the right people tested

Accuracy:

- Only ~10% of population have diabetes
- Can achieve high accuracy by always predicting negative
- Flawed measure of how system help us achieve our goal

# Model evaluation - performance metrics

Precision:

- High precision tells us we are not advising too many people to go into the doctor's office

Recall:

- High recall tells us we do not miss many positive cases of diabetes

$F_1$-score:

- Balance between precision and recall
- High $F_1$ score tells us we are able to detect many cases of diabetes, without many negative tests
- Most important evaluation metric for how system help us achieve our goals

**Performance of our classifier:**

| | Test set |
|---|---|
| Accuracy | 92.7% |
| Precision | 95.4% |
| Recall | 92.5% |
| F1 | 93.9% |

# Ranges and methodology

Repeated Bootstrap

- Draw training set as bootstrap sample of the size of the entire data set
  - ~60% unique samples
- All datapoints not included in bootstrap sample is the test set
  - ~40% of total data set
- Train on training set
- Predict and evaluate performance on test set

Why this method?

- Simple and transparent
- Can generate any number of estimates for the metrics
  - Not limited by number of folds as CV

# Results - ranges

Ranges for 100 estimates of the chosen metrics

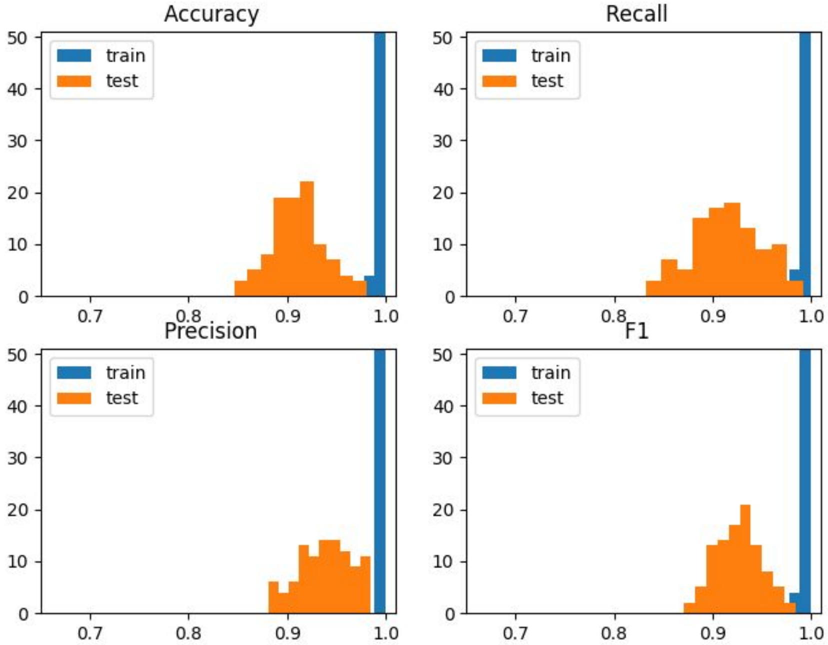
Training data almost always obtains perfect scores

Quite low spread of scores on test set

- Low standard deviations

F1-score has the smallest standard deviation of all metrics

Model seems robust, and performs well across metrics and test sets!



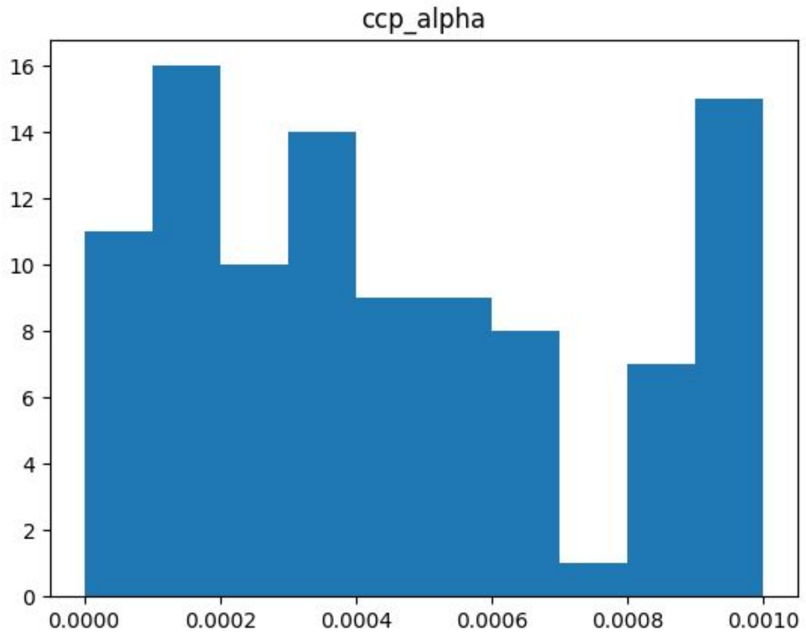

| | Train mean | Train stdev | Test mean | Test stdev |
|---|---|---|---|---|
| Accuracy | 99.7% | 0.003 | 91.1% | 0.027 |
| Precision | 99.9% | 0.002 | 93.8% | 0.026 |
| Recall | 99.6% | 0.005 | 91.6% | 0.036 |
| F1 | 99.8% | 0.003 | 92.6% | 0.023 |

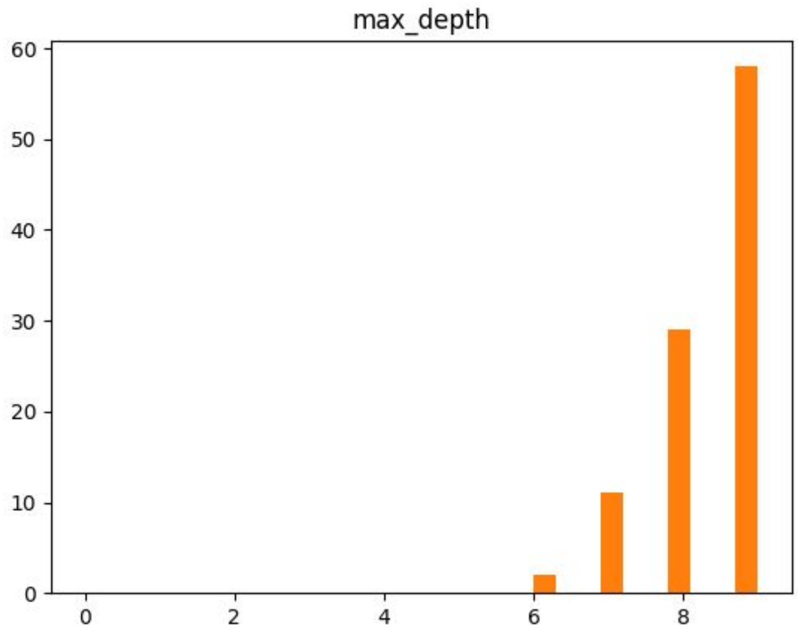
Table 1: Performance on original data

# Hyperparameters - plots

- Grid defined by tree depth and cost-complexity pruning (ccp)
- Optimal parameters found using cross-validation on grid

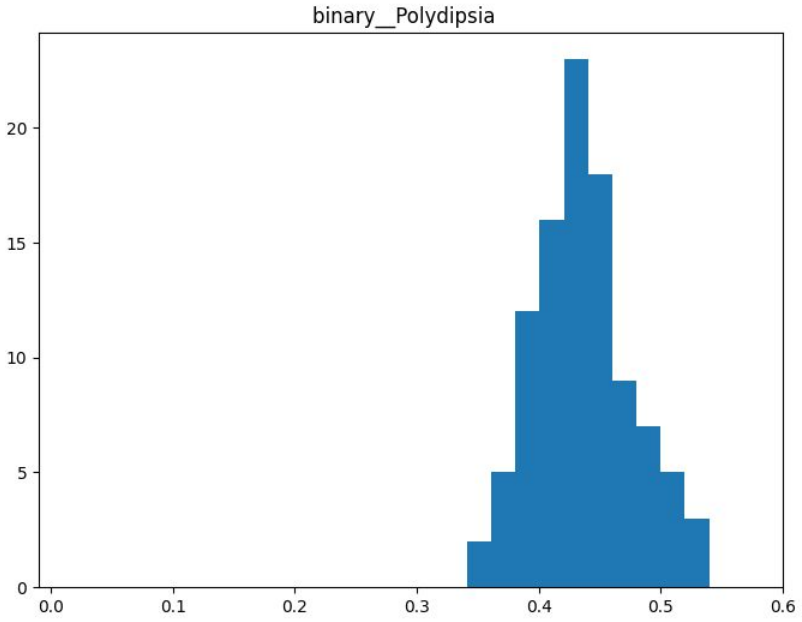
Figures show optimal parameters for each sample

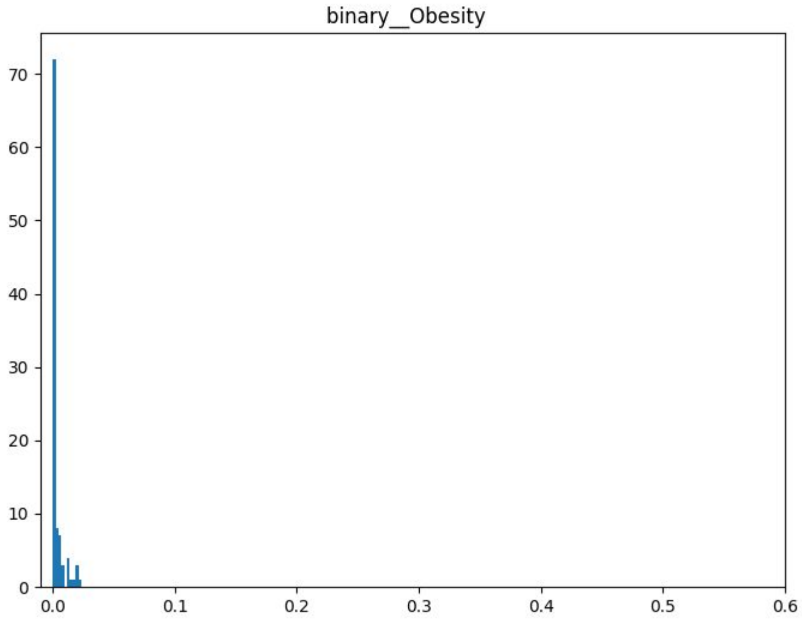


# Feature importance

Built in method of sklearn
- Measures Gini importance
- Higher Gini importance means more important variable



# Privacy

Dataset: Personally identifiable health information

Anonymize the binary data (Majority of data)

- Sensitive data:
  - genital thrush, obesity, gender etc.

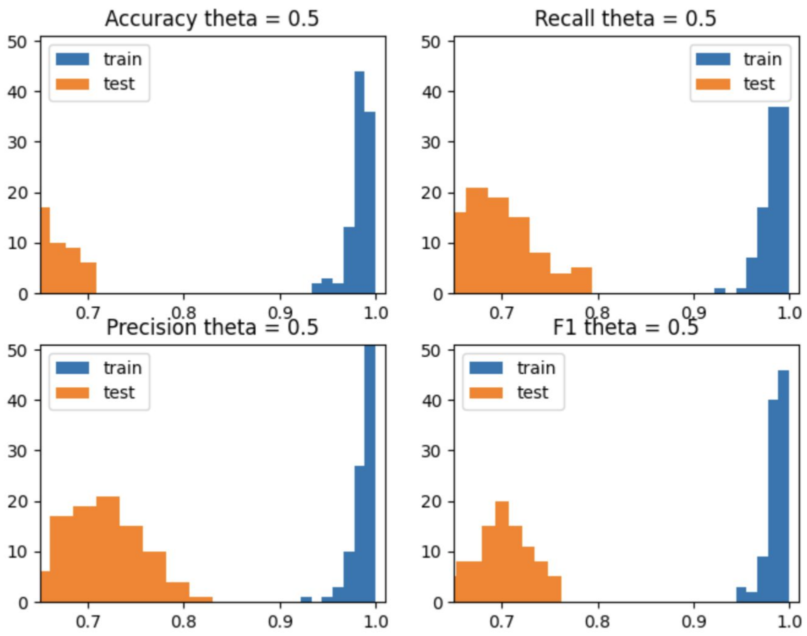
Adding controlled noise through Differential privacy (ε-differential privacy)

- Allow us to train a machine learning model on the data without compromising information on specific individuals
- Avoids de-anonymization with possible future datasets.
  - In contrast, K-anonymity gives no such guarantee

# Privacy procedure

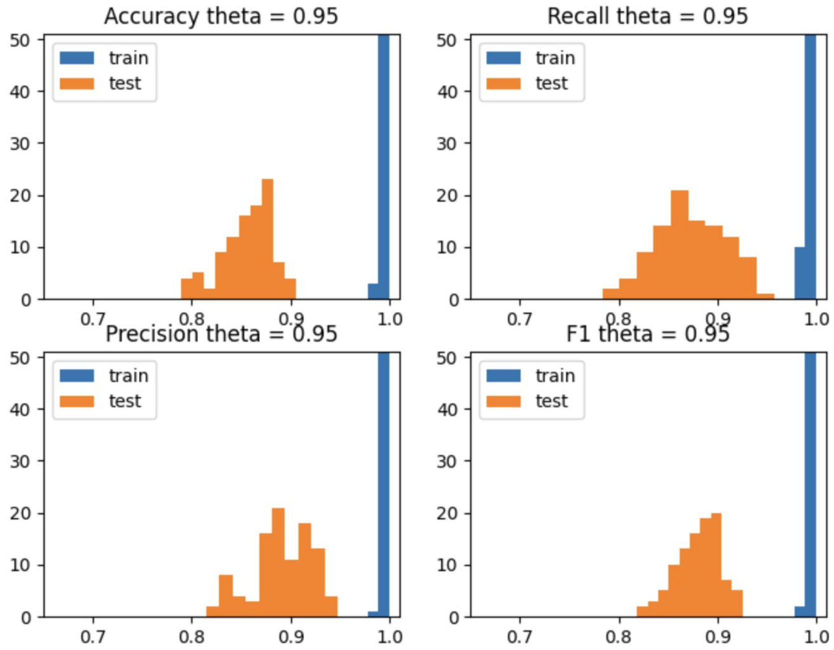
- **Randomized response:**
  - Coin flipping with a probability of answering truthfully = theta
- Column wise:
  - Calculated independently for each column
  - measure privacy guarantee- Calculate epsilon for each column
- Set a separate random seed for anonymization
- Result in new dataset - for reproducing experiments
- Not randomized target
  - Small dataset
  - Less meaningful performance metrics when comparing noisy target
- We experiment with two different cases: theta = 0.5 and theta = 0.95
  - epsilon = 1.09 and epsilon = 3.66

# Experiments with different theta




| | Train mean | Train stdev | Test mean | Test stdev |
|---|---|---|---|---|
| Accuracy | 98.3% | 0.012 | 63.4% | 0.037 |
| Precision | 98.9% | 0.012 | 70.9% | 0.046 |
| Recall | 98.4% | 0.012 | 68.9% | 0.044 |
| F1 | 98.6% | 0.010 | 69.7% | 0.032 |

Table 3: Theta = 0.5




| | | | | |
|---|---|---|---|---|
| Accuracy | 99.8% | 0.003 | **85.7%** | 0.026 |
| Precision | 99.9% | 0.002 | **89.1%** | 0.031 |
| Recall | 99.7% | 0.005 | **87.3%** | 0.036 |
| F1 | 99.8% | 0.003 | **88.1%** | 0.022 |

Table 5: Theta = 0.95

# Privacy is not free

| | Original data | | Anonymized data | |
|---|---|---|---|---|
| | Test mean | Test stdev | Test mean | Test stdev |
| Accuracy | **91.1%** | 0.027 | 85.7% | 0.026 |
| Precision | **93.8%** | 0.026 | 89.1% | 0.031 |
| Recall | **91.6%** | 0.036 | 87.3% | 0.036 |
| F1 | **92.6%** | 0.023 | 88.1% | 0.022 |

Table 7: Comparing results on test set for original data and anonymized data with theta = 0.95

# Privacy: future work

- Differential privacy to proof against current and future datasets
  - Only applied to binary features
- Possible improvement:
  - combine differential privacy with other methods such as k-anonymity to increase the level of privacy.
  - or combine with Laplace for continuous data
  - Anonymise target