**Data Leakage:**

Data leakage is when some data from the training set includes information about the very thing you’re trying to predict. This is a problem as when the model is actually deployed it won’t be able to have access to this leaked data and will result in poor real-world performance.

Obvious examples:

* Including the label to be predicted as a feature.
* Including test data with training data.

Often if your model’s performance is too good to be true its probably due to data leakage.

Data leakage leads to suboptimal model training which in turn leads to a model that will not generalize well to new data instances as it would have learned the wrong patterns.

**Subtle Examples of Data Leakage:**

Training data includes **information about the future**. E.g. in time series analysis this could lead to huge improvements. E.g. predicting if a customer will open an account, the data set might have the customers email online for instances when they do open an account.

**Leakage in Training Data:**

* **Performing data pre-processing using parameters or results from analysing the entire dataset**: Normalizing and rescaling, detecting and removing outliers, estimating missing values, feature selection.
* **Time-series datasets**: using records from the future when computing feature for the current prediction.
* **Errors in data values/gathering** or missing variable indicators (e.g. the special value 999) can encode information about missing data that reveals information about the future.

**Leakage in Features:**

* Removing variables that are not legitimate without also removing variables that encode the same or related information (e.g. diagnosis info may still exist in patient ID).
* Reversing of intentional randomization or anonymization that reveals specific information about e.g. users not legitimately available in actual use.

**Detecting Data Leakage:**

**Before building the model**:

* Do exploratory data analysis to find surprises in the data.
* Check to see if there are features very highly correlated with target values.

**After building the model:**

* Look for surprising feature behaviour in the fitted model.
* Are there features with very high weights, or high information gain?
* Simple rule-based models like decision trees can help with features like account numbers, patient ID’s
* Is overall model performance surprisingly good compared to known results on eh same dataset, or for similar problems on similar datasets?

**Limited real-world deployment of the trained model:**

* Potentially expensive in terms of development time, but more realistic.
* Is the trained model generalizing well to new data?

**How to Minimize Data Leakage:**

**Perform data preparation within each cross-validation fold separately**

* Scale/normalize data, perform feature selection, etc. within each fold separately, not using the entire dataset.
* For any such parameters estimated on the training data, you must use those same parameters to prepare data on the corresponding held-out test fold.

**With time series data, use a timestamp cut-off**

* The cut-off value is set to the specific time point where prediction is to occur using current and past records.
* Using a cut-off time will make sure you aren’t accessing any data records that were gathered after the prediction time.

**Before any work with a new dataset, split off a final test validation dataset**

* Only if you have enough data.
* Use this final test dataset as the very last step in your validation.
* Helps to check the true generalization performance of any trained models.