My ML Model without "Data Leakage"

My ML Model with "Data Leakage"

Learn more >>>

What is Data leakage

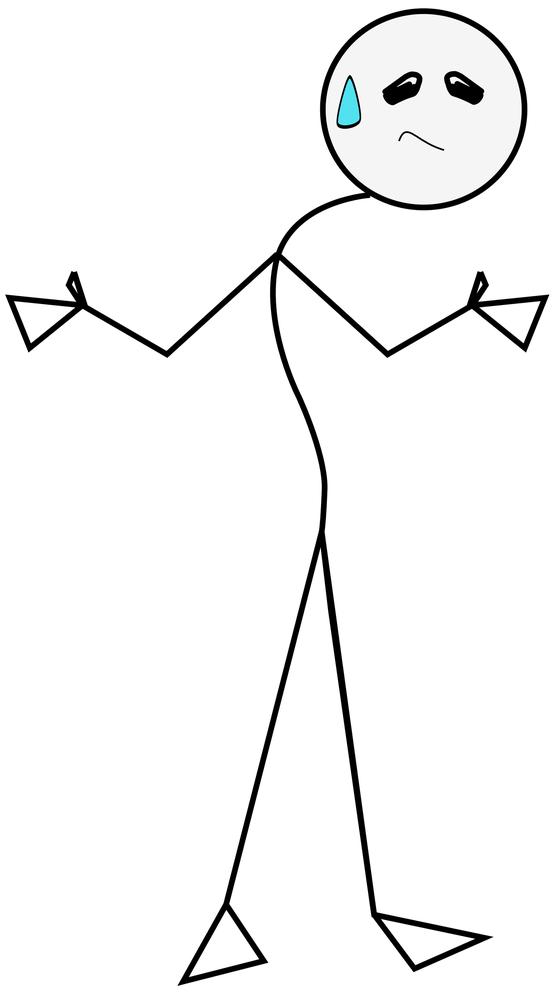
Data leakage is a big problem in machine learning models when it is exposed to real-world data.

Have you ever encountered, your model performing well on train and test data, but not in real-world?? this is what we call data leakage

The goal of machine learning models is to learn about the data available and make predictions on unseen data.

Data leakage is when information from outside the training dataset is used to create the model.

What is Data leakage



Data leakage refers to when engineers accidentally share information between train and test data.

Due to the above error, you see a high performance on the train and test dataset but doesn't work well in the real world. Because it is already aware of the test data.

In simple terms When the data you are using to train a machine learning algorithm happens to have the information, you are trying to predict.

Types of Data leakage

Target or Label leakage

	went_to_paris	age	weight	female	booked_a_flight
0	True	50	81	True	True
1	False	45	75	True	False
2	True	37	68	False	True
3	False	44	52	False	False
4	True	40	99	True	True

Consider this example where we are leaking information that is not available at the time predictions in the real world.

lets say went_ to_paris is the target variable and if you see that variable and booked_a_flight are highly correlated and this variable is not available at the time. This is where you leaked the data while training your model.

Types of Data leakage

Train-Test contamination

```python
from sklearn.model_selection import train_test_split

# splitting data
X_train, X_test, y_train, y_test = train_test_split(
                                X, y, test_size=0.33, random_state=42)

# Trasforming data
x_test = imputer.fit_transform(x_test)
```

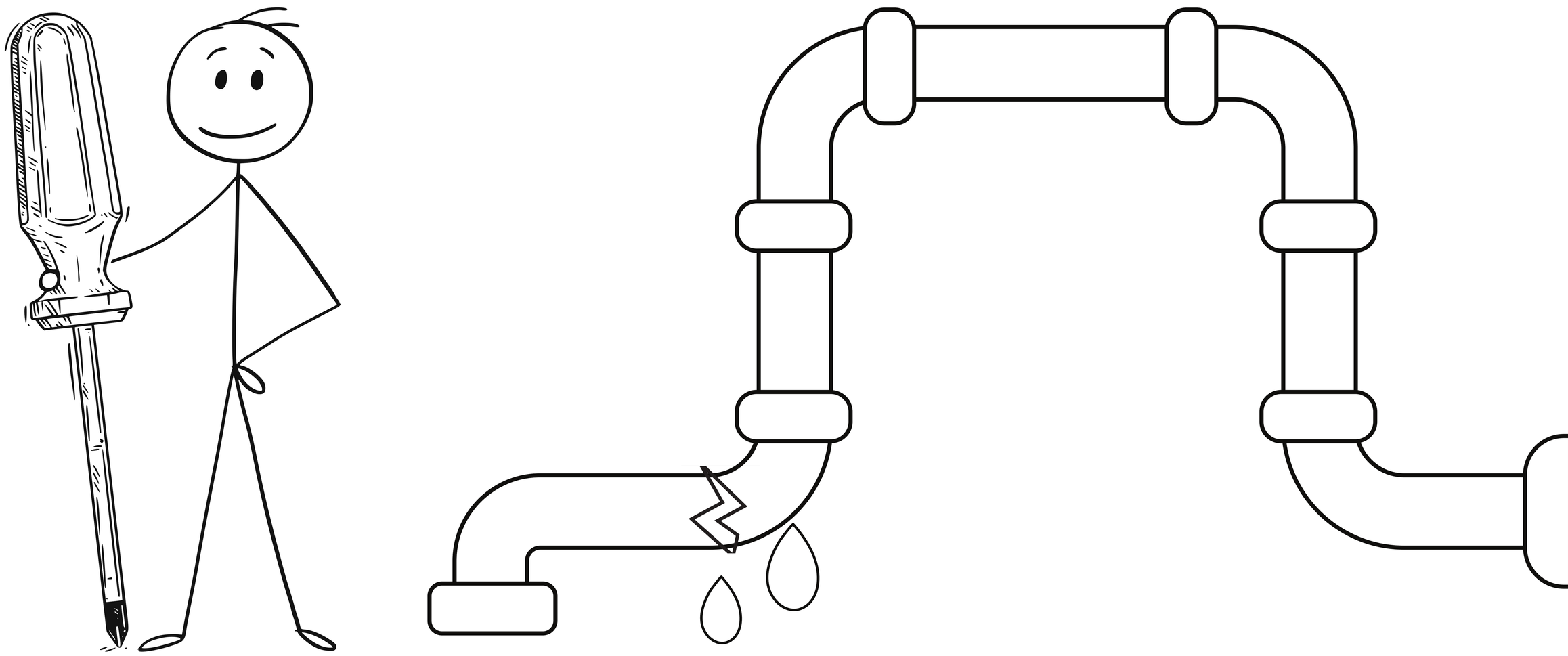
We shouldn't call fit_transform methods on our test set but only on our training dataset.

```python
from sklearn.model_selection import train_test_split

# splitting data
X_train, X_test, y_train, y_test = train_test_split(
                                X, y, test_size=0.33, random_state=42)

# Trasforming data
x_train = imputer.fit_transform(x_train)
X_test = imputer.transform(x_test)
```

Prevent Data leakage



- Understanding the Dataset
- Cleaning Dataset for Duplicates
- Selecting Features about Target Variable correlation and Temporal Ordering
- Splitting Dataset into Train, Validation, and Test groups
- Normalizing After Splitting, BUT Before Cross Validation
- Assessing Model Performance with a Healthy scepticism

Tips to combat Data leakage

- Temporal Cutoff: Remove all data just prior to the event of interest, focusing on the time you learned about a fact or observation rather than the time the observation occurred.
- Add Noise: Add random noise to input data to try and smooth out the effects of possibly leaking variables.
- Remove Leaky Variables: Evaluate simple rule-based models line OneR using variables like account numbers and IDs and the like to see if these variables are leaky, and if so, remove them. If you suspect a variable is leaky, consider removing it.
- Use Pipelines: Heavily use pipeline architectures that allow a sequence of data preparation steps to be performed within cross-validation folds, such as the caret package in R and Pipelines in scikit-learn.
- Use a Holdout Dataset: Hold back an unseen validation dataset as a final sanity check of your model before you use it.

This slide source (machine learning mastery)

Turn on post notifications

Which topic do you want me to cover next?

Like

Save