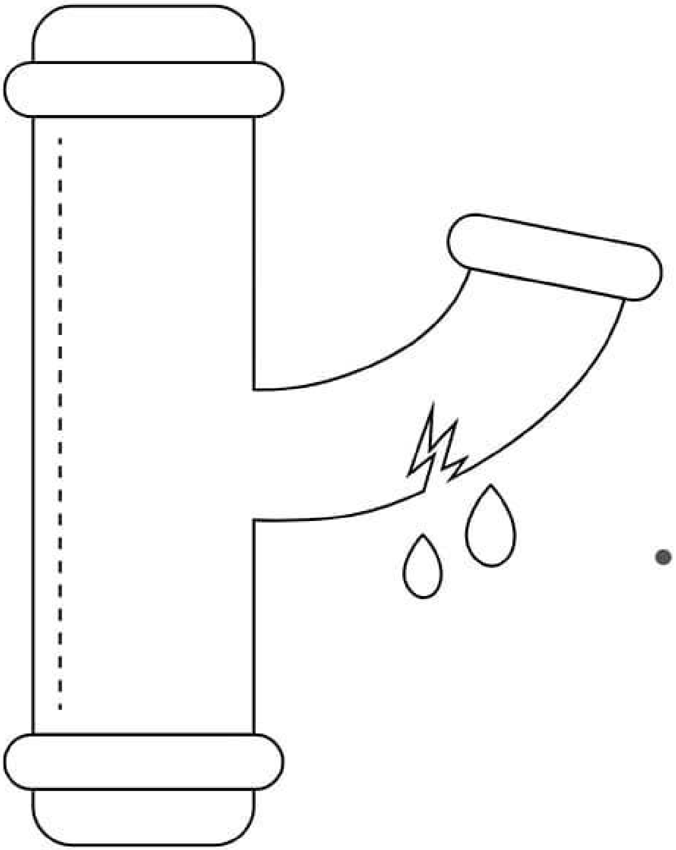
My **model** without **data leakage**

My **model** with **data leakage**

Swipe to learn about 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 performs well on train and test data, but not on real world?? this is what we call data leakage
- The goal of machine learning models is to learn on the data available and making prediction on unseen data.
- Data leakage is when information from outside the training dataset is used to create the model.



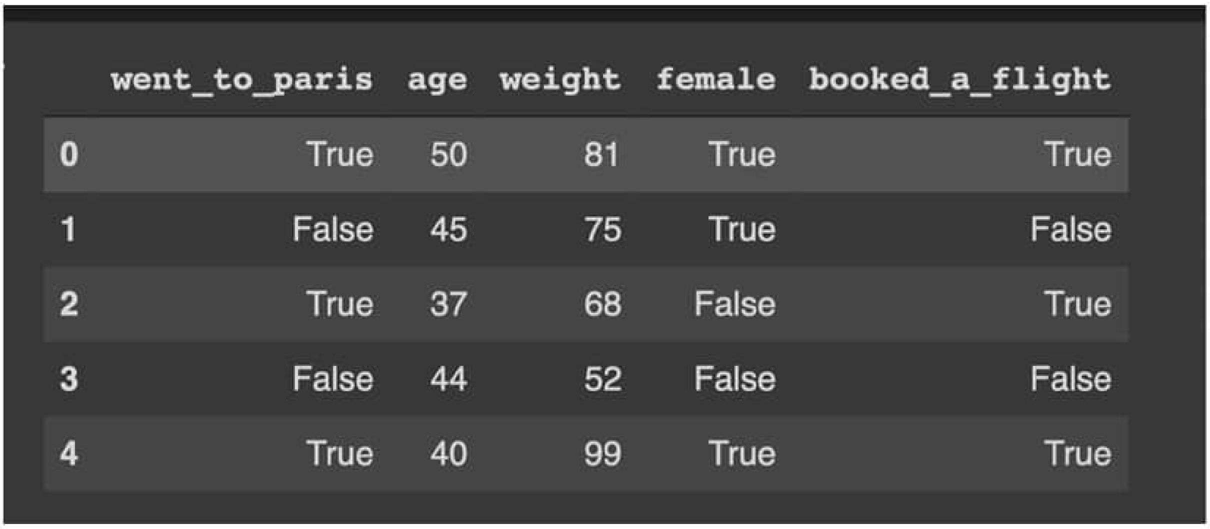
- Data leakage refers to when engineers accidentally shares information between train and test data.

- Due to the above error you see a high performance on train and test dataset but doesn't work well on 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

- Target or Label leakage
- Train-Test contamination

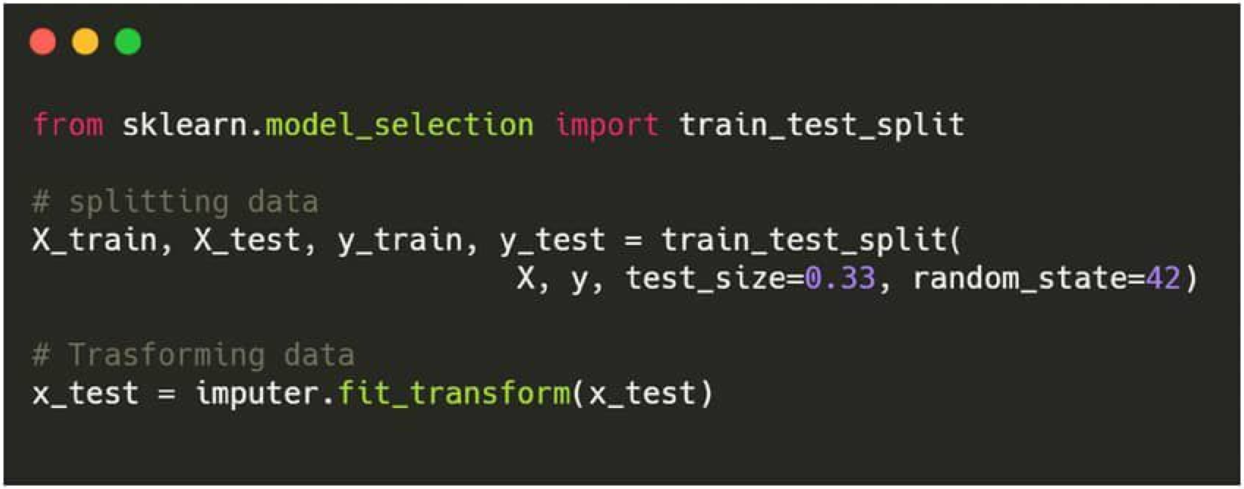
Target or Label leakage



	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 which is not available at the time predictions in real world.
- lets say went_to_paris is 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.

Train-Test contamination

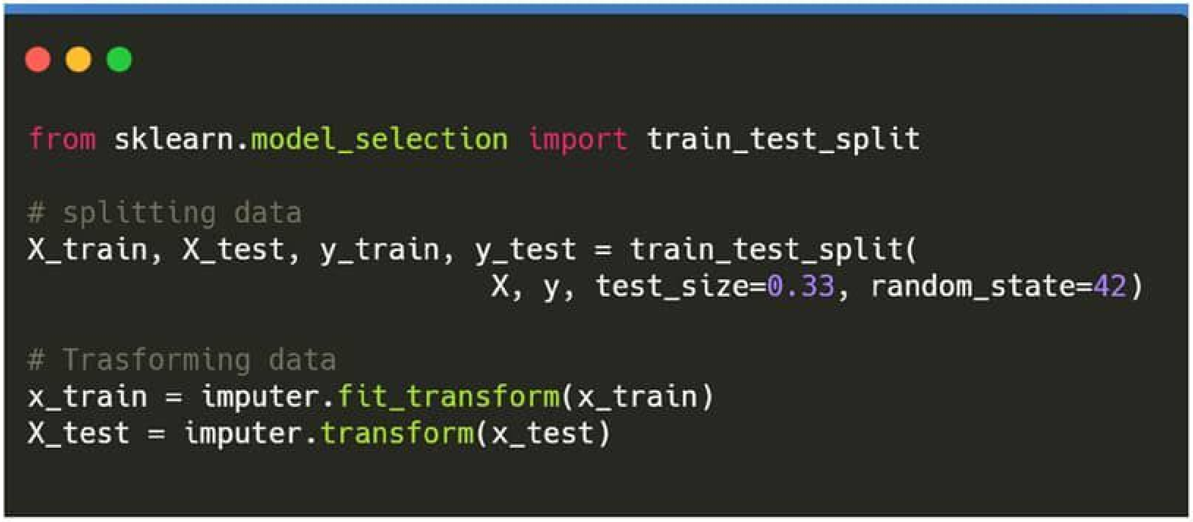


```
from sklearn.model_selection import train_test_split

# 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.



```
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)
```

- Here the correct pseudo code would be

Prevent Data Leakage

- Understanding the Dataset
- Cleaning Dataset for Duplicates
- Selecting Features with Regard to 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 Skepticism

5 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
Making Developers Awesome at Machine Learning