Dr. Christian Tsoungui Obama

ML Cheatsheet #1

Your practical 8-step guide from business idea to a deployable model

A Foundational Guide for Aspiring Data Scientists

Based on
"Hands-On Machine Learning with Scikit-learn, Keras & TensorFlow"
by
Aurélien Géron

The Problem

Manually applying transformations to your training and test data is repetitive and prone to errors.

You might forget a step on the test set, leading to data leakage or a model that fails in production. This is a classic mistake!

The Solution: Pipelines!

A Scikit-Learn `Pipeline` chains together multiple steps, so you can apply the same sequence of transformations to your data with a single, reliable command.

How It Works

A pipeline is a list of `(key, value)` pairs, where the `key` is a name for the step, and the `value` is an estimator object.

```
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler

# Define the steps for the pipeline
num_pipeline = Pipeline([
    ('imputer', SimpleImputer(strategy="median")),
    ('std_scaler', StandardScaler()),
])

# Now you can fit and transform data in one go!
# prepared_data = num_pipeline.fit_transform(data)

```

Why Use Pipelines?

- **Simplicity:** Cleaner, more readable code.
- **Prevents Data Leakage:** Ensures you fit transformers on training data ONLY.
- **Easier Deployment:** Save the entire pipeline as a single object for production.
- **Grid Search Ready:** Tune hyperparameters of both transformers and models together.

Did you find this useful?

Follow me (Dr. Christian Tsoungui Obama) for the next cheatsheet in this series.

Like this post to help it reach more people.

Comment below: What's your favorite Scikit-Learn feature?

Let's Connect!

/in/christiantsoungui

@chrisTs