The Six Steps

to build a preprocessor

Before starting!
****Know your columns****

What are the dtypes? -> df.dtypes
What are the values? -> df[col].value_counts()

01

Are you missing any values?

Imports

```
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OneHotEncoder
from sklearn.pipeline import make_pipeline
from sklearn.compose import
make_column_transformer
```

For EACH column where you're missing data, but NOT the ones that aren't, instantiate an imputer

- imputer = SimpleImputer(strategy = ' ')
- strategies: 'mean', 'median', 'most_frequent', 'constant' (fill_value = ' ')
- use argument add_indicator = True to negate effect of imputation

02

Does it need to be one-hot encoded?

- Nominal categorical columns should be one-hot encoded
  - ohe = OneHotEncoder()
- If the column has too many unique values relative to the length of the dataset, consider dropping the column instead of one-hot encoding

03

Does it need to be scaled?

- Numeric columns whose values do NOT fall between 0 and 1 should be scaled for MOST models
  - scaler = StandardScaler()
- Numeric columns whose values fall between 0 and 1 are already scaled

04

Collect steps into pipelines

- You should have one pipeline for each group of columns that are receiving different treatments
- ex) group1_pipe = make_pipeline(mean_imputer, scaler)
- ex) group2_pipe = make_pipeline(constant_imputer, ohe)

05

Create tuples

Combine pipelines and columns together in tuples

- ex) group1_tuple = (group1_pipe, group1_columns)
- ex) group2_tuple = (group2_pipe, group2_columns)

Create the preprocessor

- preprocessor = make_column_transformer(group1_tuple, group2_tuple, remainder = ' ')
- Use remainder parameter option 'drop' or 'passthrough'

06

You have a preprocessor!

Congratulations!