dm25s1

Topic 06 : Data Modelling

Part 02 : Column Encoding Scaling

Dr Bernard Butler
Department of Computing and Mathematics, WIT.
(bernard.butler@setu.ie)

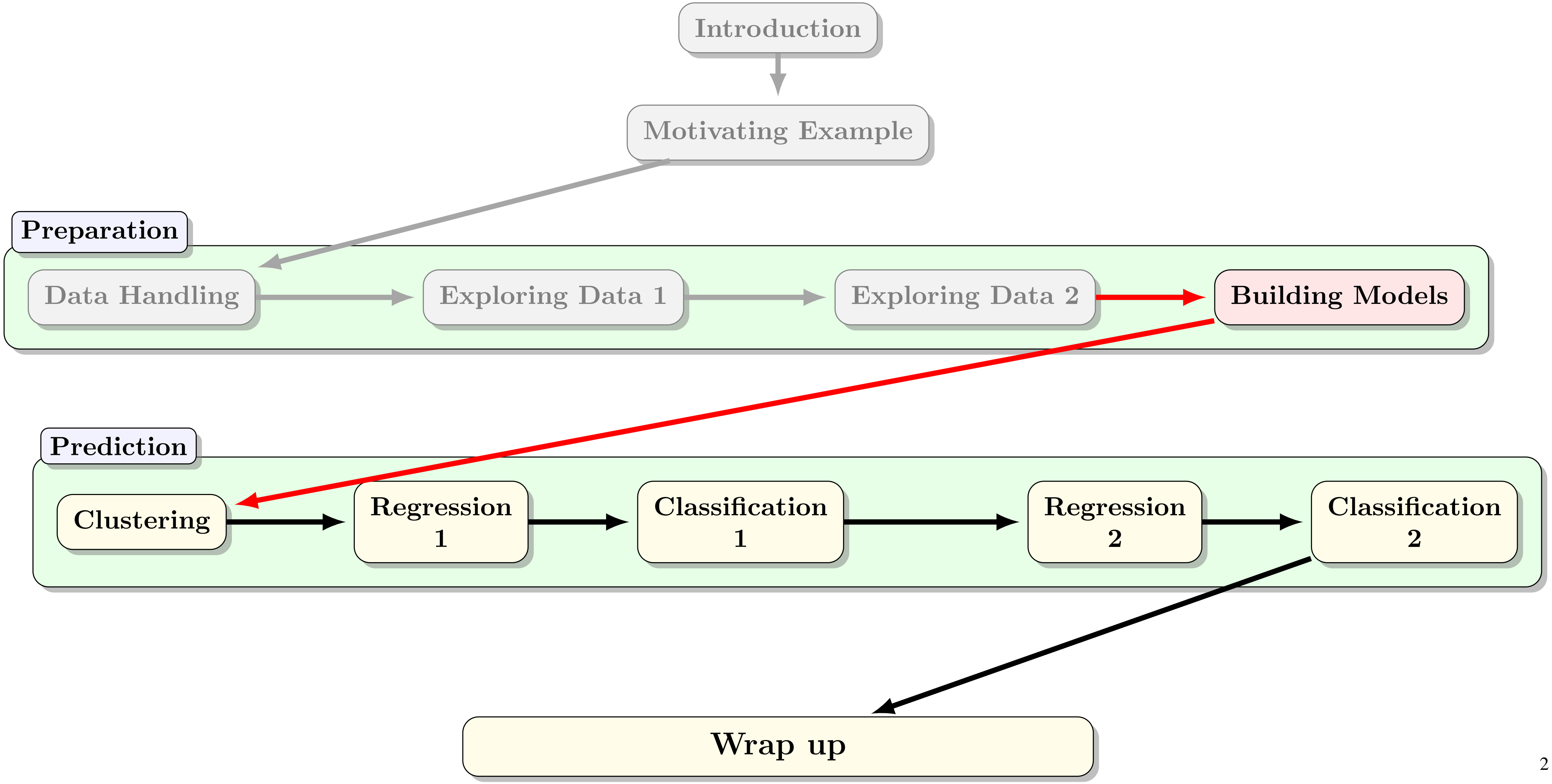
Autumn Semester, 2025

Outline

- Encoding categorical features, using pandas
- Scaling numeric features
- Looking ahead to feature preparation
- Overview of ML and what we have achieved

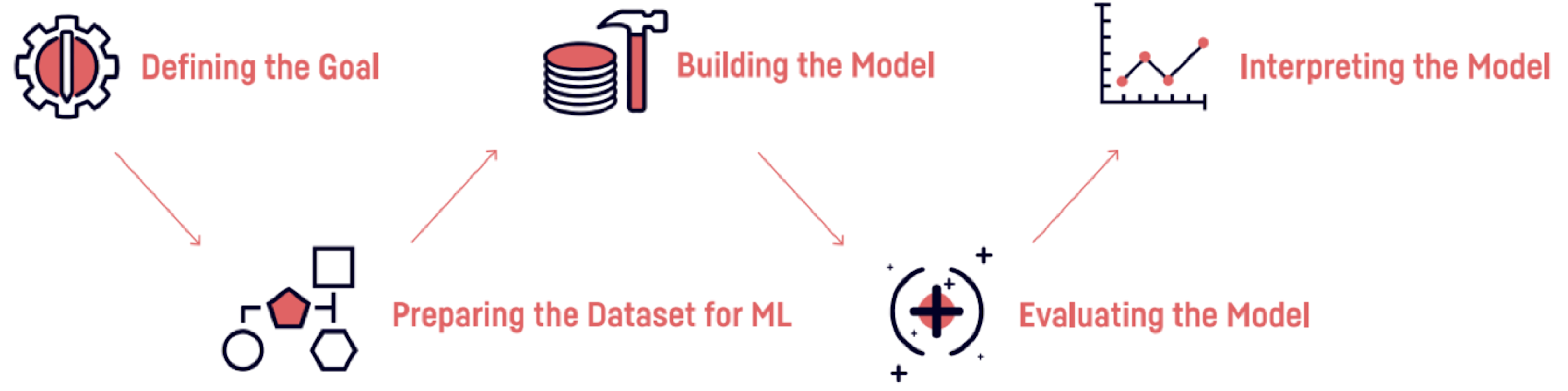
Wrap up

Data Mining (Week 6)



The Pipeline Metaphor

Model Building Pipeline

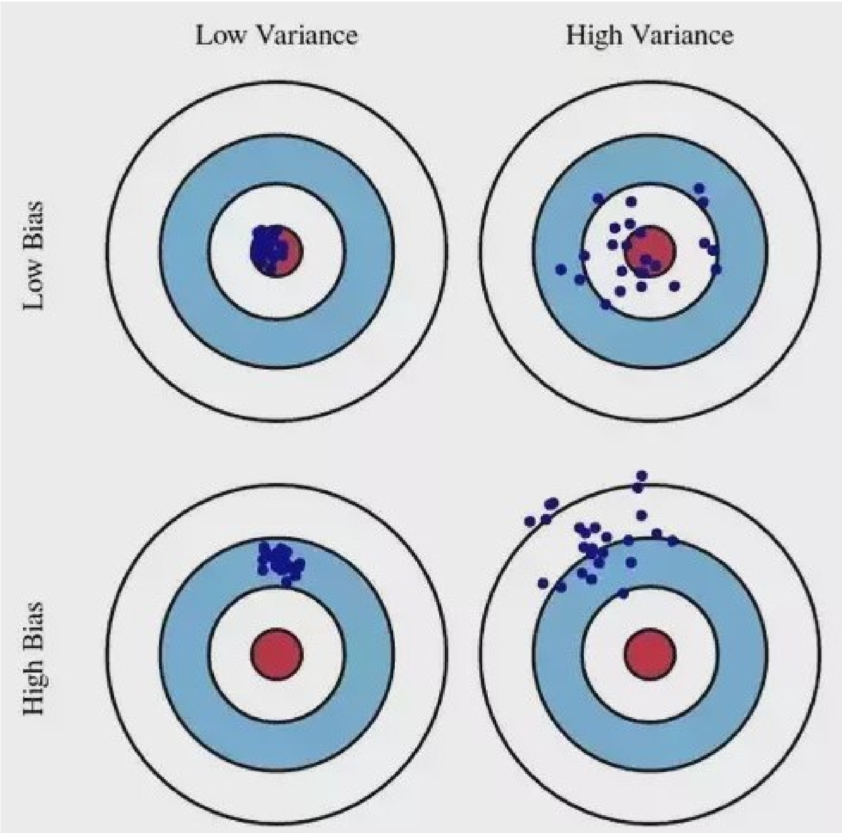


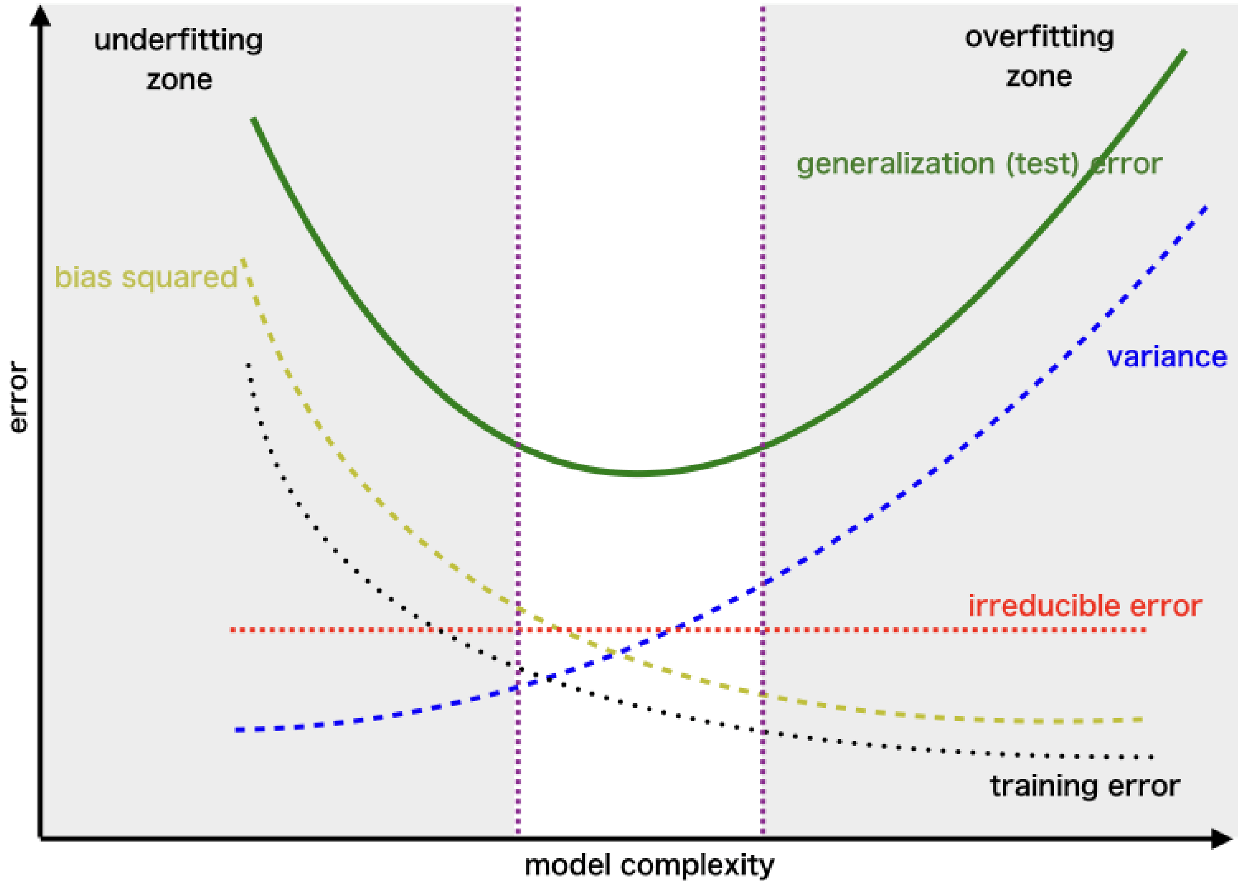
Source: Dataiku

Comments

- We saw the first two stages in previous weeks
- This week we look at the remaining stages
- Of course this pipeline is a simplification. In reality it is iterative.

Bias-Variance and Total Error





Look for parameters $\boldsymbol{a}$ that minimise the generalization error (estimated using the test set that was not used during training)

Example Model Types

Model	Applications	Concerns
Logistic Regression	X-ray classification	Regression with transformed variable
Fully connected networks	Classification	Classical ANN: choose encoding and size
Convolutional Neural Networks	Image processing	deep learning - choose segmentation
Recurrent Neural Networks	Voice recognition	ANN with feedback - how much?
Random Forest	Fraud Detection	Ensemble method - how many?
Reinforcement Learning	Learning by trial and error	Choose goal and penalties
Generative Models	Text, Image creation	Choose parameters
K-means	Segmentation	Choose distance function and k
k-Nearest Neighbors	Recommendation systems	Choose distance function and k
Bayesian Classifiers	Spam and noise filtering	Deal with imbalances

Using Categorical Features in (Logistic) Regression

How can Categorical-valued features participate in linear models?

Given the following fragment of a dataset, where the goal is to predict the salary of employees in a large organisation:

```
df = pd.read_csv('data/team.csv',\
        index_col="Name")
df
```

	Role	Skilled	Salary
Name			
Alice	Designer	Yes	40000
Bob	Programmer	No	25000
Carol	Tester	No	30000

How can this data be represented by a linear model, where all quantities must take numeric values?

Using pandas .getdummies() on a binary-valued column

```
dfSkilledDummies = pd.get_dummies(df['Skilled'],\
        prefix='Skilled',\
        dtype=int)
dfSkilledDummies
```

	Skilled_No	Skilled_Yes
Name		
Alice	0	1
Bob	1	0
Carol	1	0

Note that a binary-valued column becomes 2 dummy columns

Reducing redundancy (by 1) in 2 dummy columns

```
dfSkilledIndicators = pd.get_dummies(df['Skilled'],\
       prefix='Skilled',\
       drop_first=True,\
       dtype=int)\
       .rename(columns={"Skilled_Yes": "IsSkilled"})
dfSkilledIndicators
```

	IsSkilled
Name	
Alice	1
Bob	0
Carol	0

A single indicator column can replace a group of 2 dummy columns

Using pandas .getdummies() on a multi-valued column

```
dfRoleDummies = pd.get_dummies(df['Role'],\
       prefix='Role',\
       dtype=int)
dfRoleDummies
```

	Role_Designer	Role_Programmer	Role_Tester
Name			
Alice	1	0	0
Bob	0	1	0
Carol	0	0	1

Note that an n-valued column becomes n dummy columns

Reducing redundancy (by 1) in n dummy columns

```
dfRoleIndicators = pd.get_dummies(df['Role'],\
        prefix='Role',\
        drop_first=True,\
        dtype=int)\
        .rename(columns={\
        "Role_Programmer": "IsProgrammer",\
        "Role_Tester": "IsTester"})
dfRoleIndicators
```

	IsProgrammer	IsTester
Name		
Alice	0	0
Bob	1	0
Carol	0	1

$n-1$ indicator columns can replace a group of n dummy columns

Deriving and using dummy/indicator features

- Identify potential categorical features in EDA Pass 1
- Identify whether each feature is (potentially) *usable* in EDA Pass 2
- Identify whether each feature is (potentially) *useful* in EDA Pass 3
- Add all potentially usable and useful features (regardless of type) to a list F
- For each categorical feature f_j in F having n levels
  - Derive $n-1$ indicator features $\tilde{f}_j^k$, where $k = 1, \ldots, n-1$
  - Replace the original categorical feature f_j in F with the derived indicator features $\tilde{f}_j^k$.
- Build the model using the features in F.

Analysis of pandas dummies

Pandas enables us to convert unordered categorical features into sets of (0,1)-valued numeric features

But what about...

1. Ordered categorical features - can we do better than treating them as unordered (and losing information)?
2. Categorical targets (whether ordered or not) - how should we handle these?
3. How do we handle data (that includes categocial columns) that is split into training and test?
4. How can we reverse the operation (i.e., return from (0,1)-valued columns to categorical columns)?

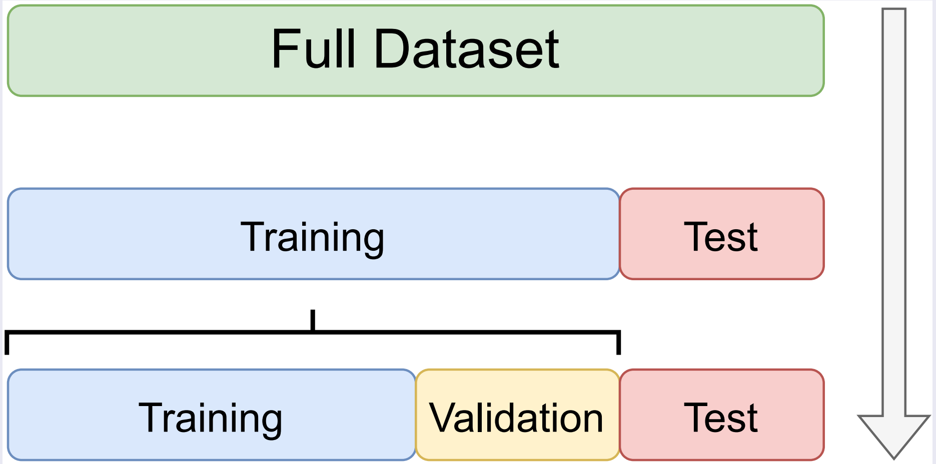
Training, test and valuation subsets: 3-way Holdout

Why Split?

Hold back some data to check how the model is doing.

- Training data is sample used to fit the model parameters.
- Test data is sample used to test the final model fitted to the training data.
- Validation data is sample used to test each interim model while tuning it.

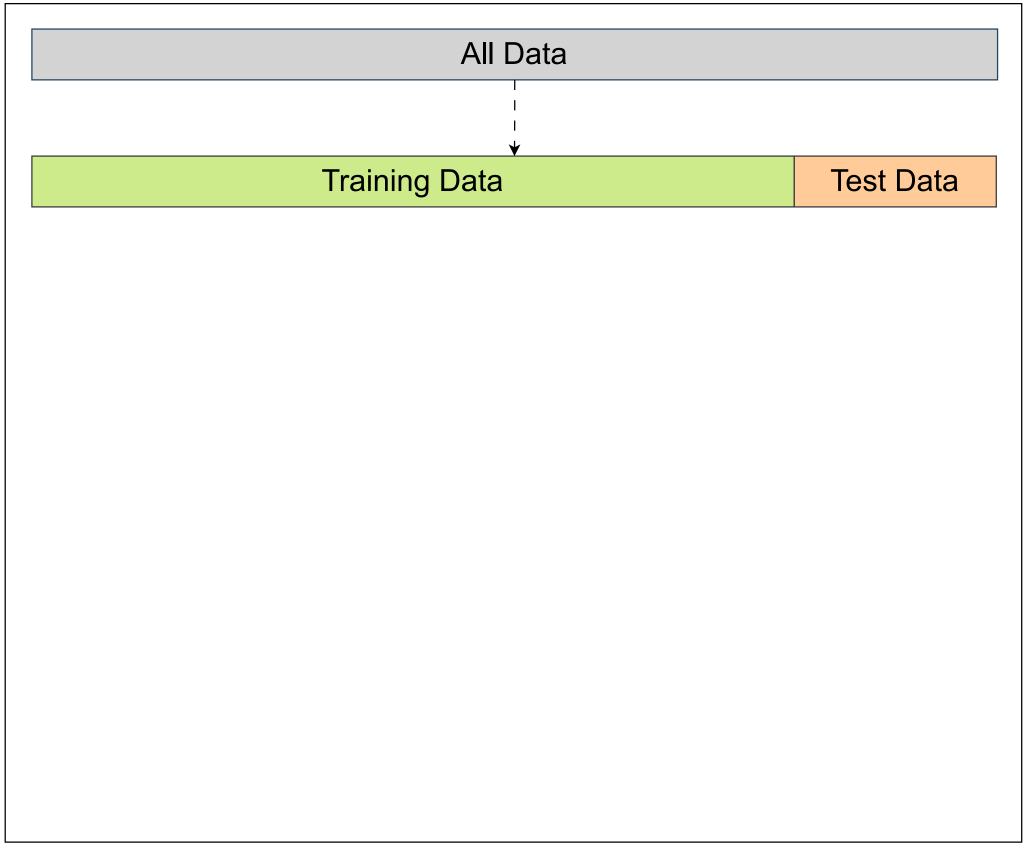
Typical Splits



sklearn example

```
from sklearn.model_selection import train_test_split
trainVal, test = train_test_split(df, test_size=0.2, seed=42)
train, validation = train_test_split(trainVal, test_size=0.1)
```

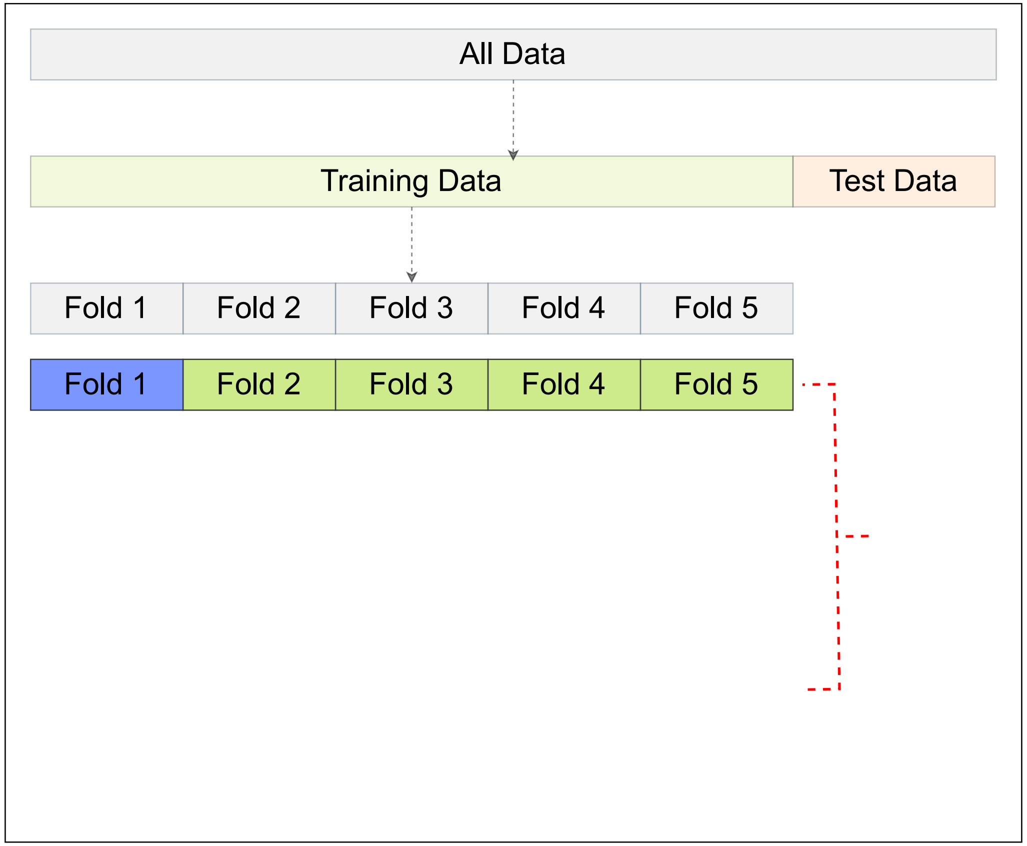
K-fold cross validation - preparation



Before Cross Validation

- Perform the train-test split
- Using the Training Set only
  - Perform 3-pass EDA
  - Encode the categorical features
  - Scale the numerical features
  - Other feature engineering steps…

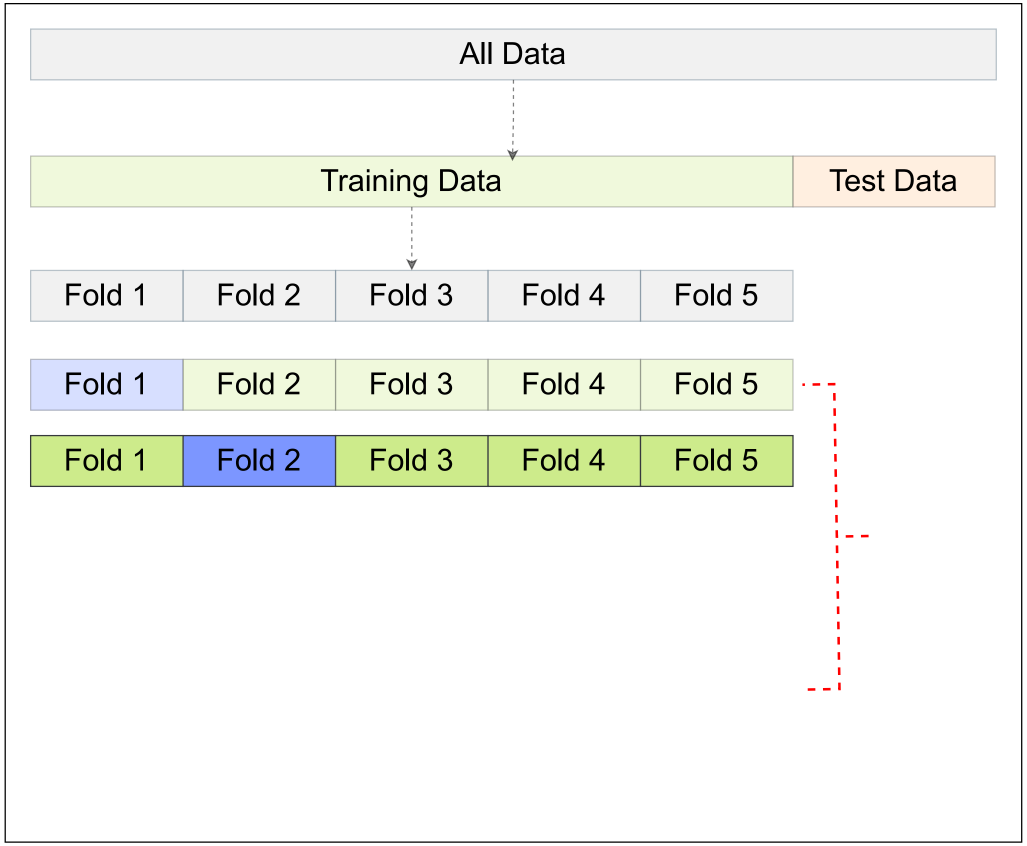
K-fold cross validation - initialise, pass 1



Initialise and perform first pass

- Randomly split the training set into K folds
- In the diagram, $K = 5$ but $K = 10$ is common
- In the first pass, set $i = 1$
- Fold $i = 1$ is held back for validation
- Folds $i = 2, \ldots, K$ are used to train a model, giving parameters $\boldsymbol{a}^{(1)}$.
- The model is evaluated against fold $i = 1$
- The distance between the actual and predicted targets is calculated and recorded as $d^{(1)}$.

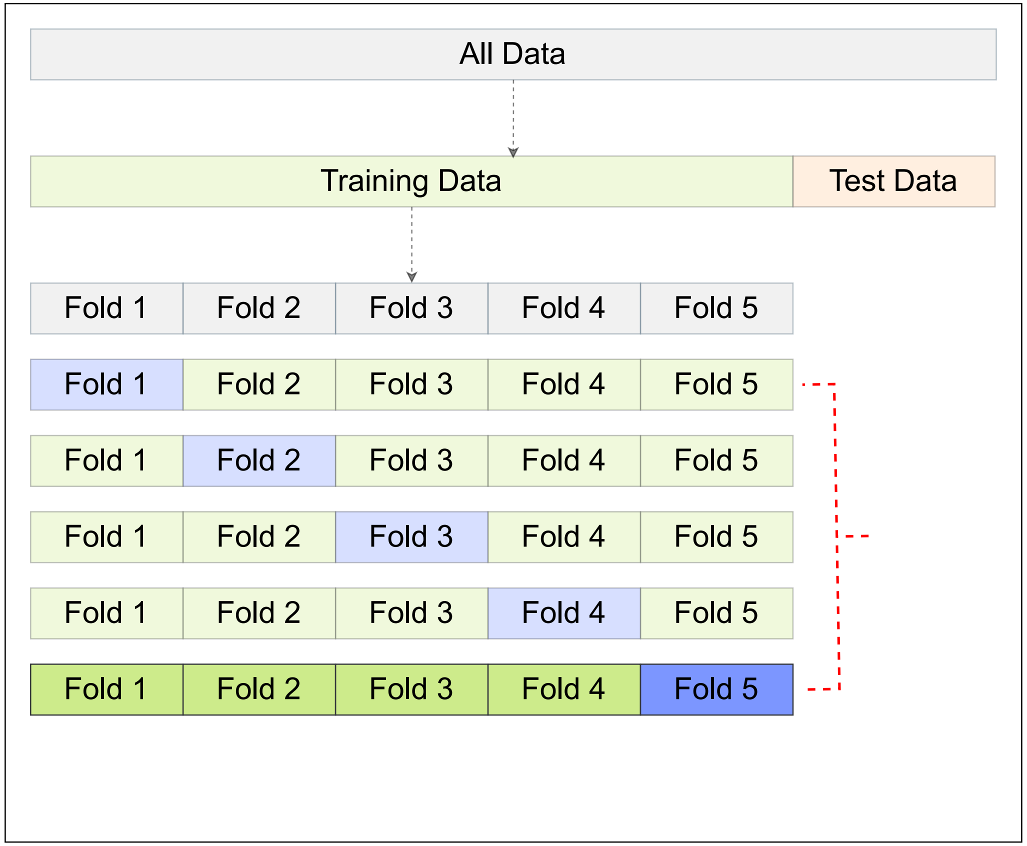
K-fold cross validation - initialise, pass 2



Perform second pass

- In the second pass, set $i = 2$
- Fold $i = 2$ is held back for validation
- Folds $i = 1, 3, \ldots, K$ are used to train a model, giving parameters $\boldsymbol{a}^{(2)}$.
- The model is evaluated against fold $i = 2$
- The distance between the actual and predicted targets is calculated and recorded as $d^{(2)}$.
- The parameters $\boldsymbol{a}^{(2)}$ and distance $d^{(2)}$ are recorded.

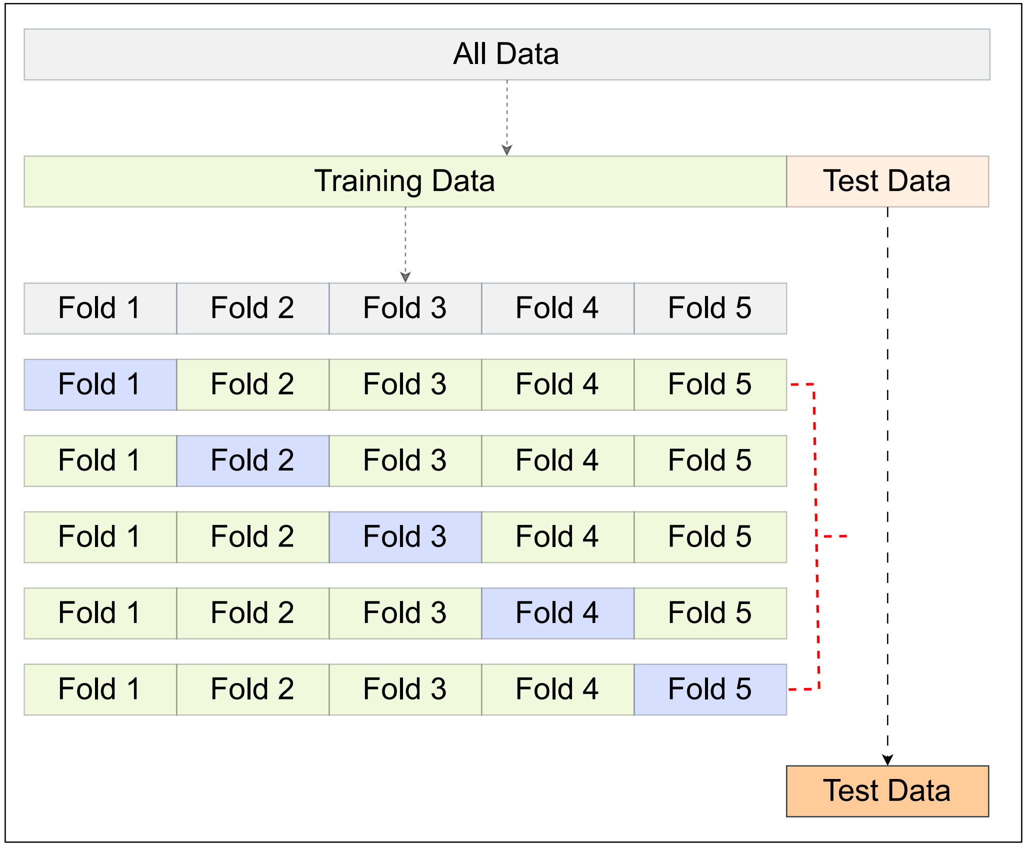
K-fold cross validation - initialise, pass $K = 5$



Perform last pass

- In the last pass, set $i = K$
- Fold $i = K$ is held back for validation
- Folds $i = 1, \ldots, K - 1$ are used to train a model, giving parameters $\boldsymbol{a}^{(K)}$.
- The model is evaluated against fold $i = K$
- The distance between the actual and predicted targets is calculated and recorded as $d^{(K)}$.
- The entire training set was used to learn the "average" parameters $\boldsymbol{a}$
- We also have an estimate of the distribution of the prediction error d, so can compare hyperparameters

K-fold cross validation - using the results



Finding and using the best model

- Typically cross validation is used to compare effects of hyperparameters
- Use the setting(s) with the lowest cross-validation error
- Given this “best” model, we can predict the test targets
- We can then compute the prediction error on the test set, as before

K-fold cross validation

sklearn example

```
from sklearn.model_selection import cross_val_score
# clf is some classifier, X and y are the features and target of the training set
scores = cross_val_score(clf, X, y, cv=5)
```

- `scores` is a $k = 5$ element array, can be used to estimate the prediction error (or other score) while building a model
- Details of cross validation are hidden...

Featuring engineering 1: Scaling of numerical variables

Scaling - what it does

- If numeric features have different scales, e.g. [-0.005, -0.003] and [10000, 10001] some terms dominate, others are "lost"
- Better: transfer the scaling from the feature to the model parameter
- A min-max scaling is often a good choice:

$$\tilde{X} = \frac{X - X_{\mathrm{min}}}{X_{\mathrm{max}} - X_{\mathrm{min}}}$$

- Note that X is in the range $[X_{\mathrm{min}}, X_{\mathrm{max}}]$ but $\tilde{X}$ is in the range $[0, 1]$.
- Other options include StandardScaler (subtract mean and divide by standard deviation) and a max-abs scaler (scales to [-1,1])

sklearn example

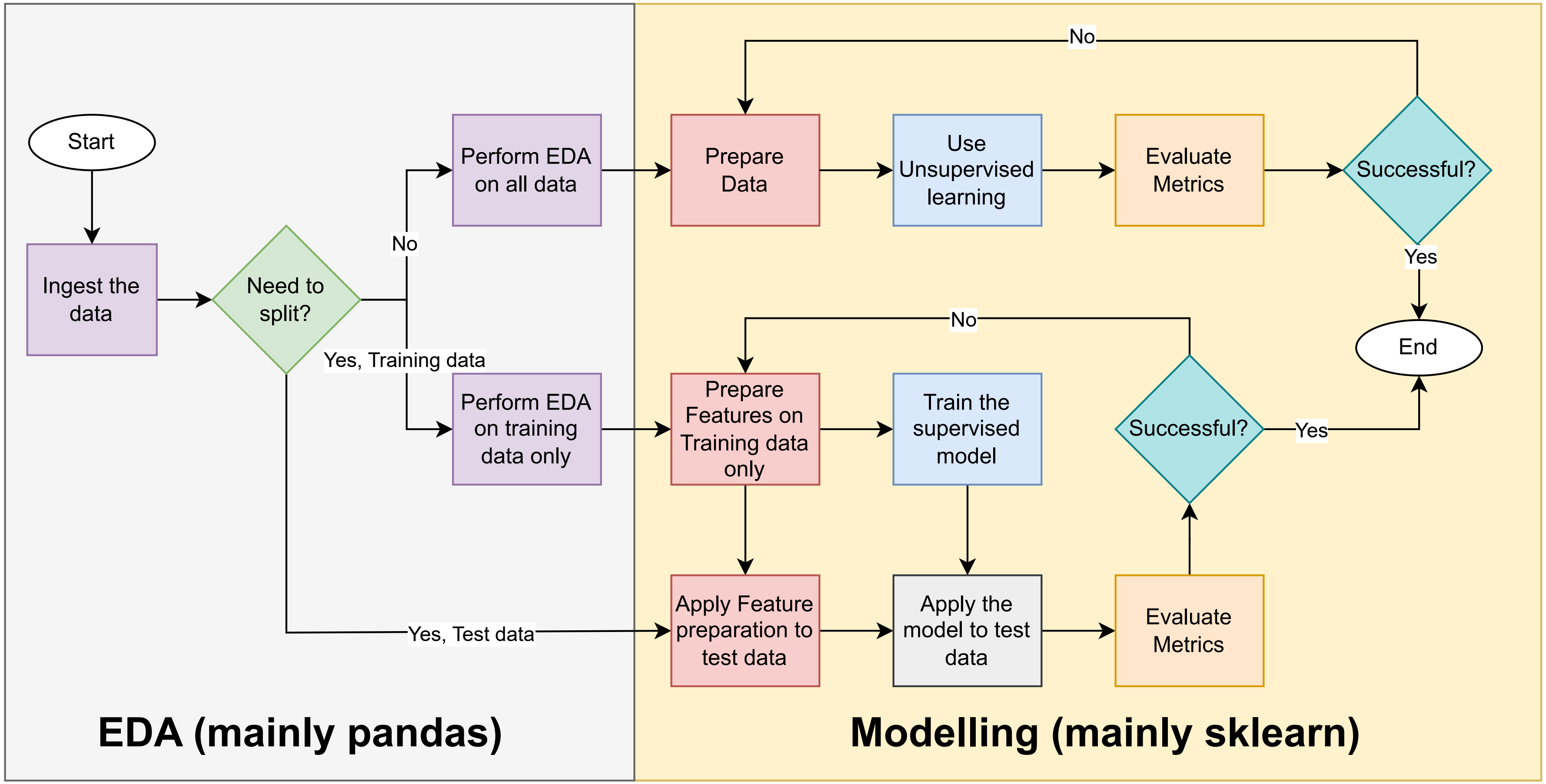
```
from sklearn.preprocessing import MinMaxScaler
# df is a dataframe with numeric features
scaler = MinMaxScaler()
dfScaled = scaler.fit(df))
```

`dfScaled` can be used instead of `df` with the advantage that the fitted parameters are more accurate.

Feature Engineering 2: Choice of Features

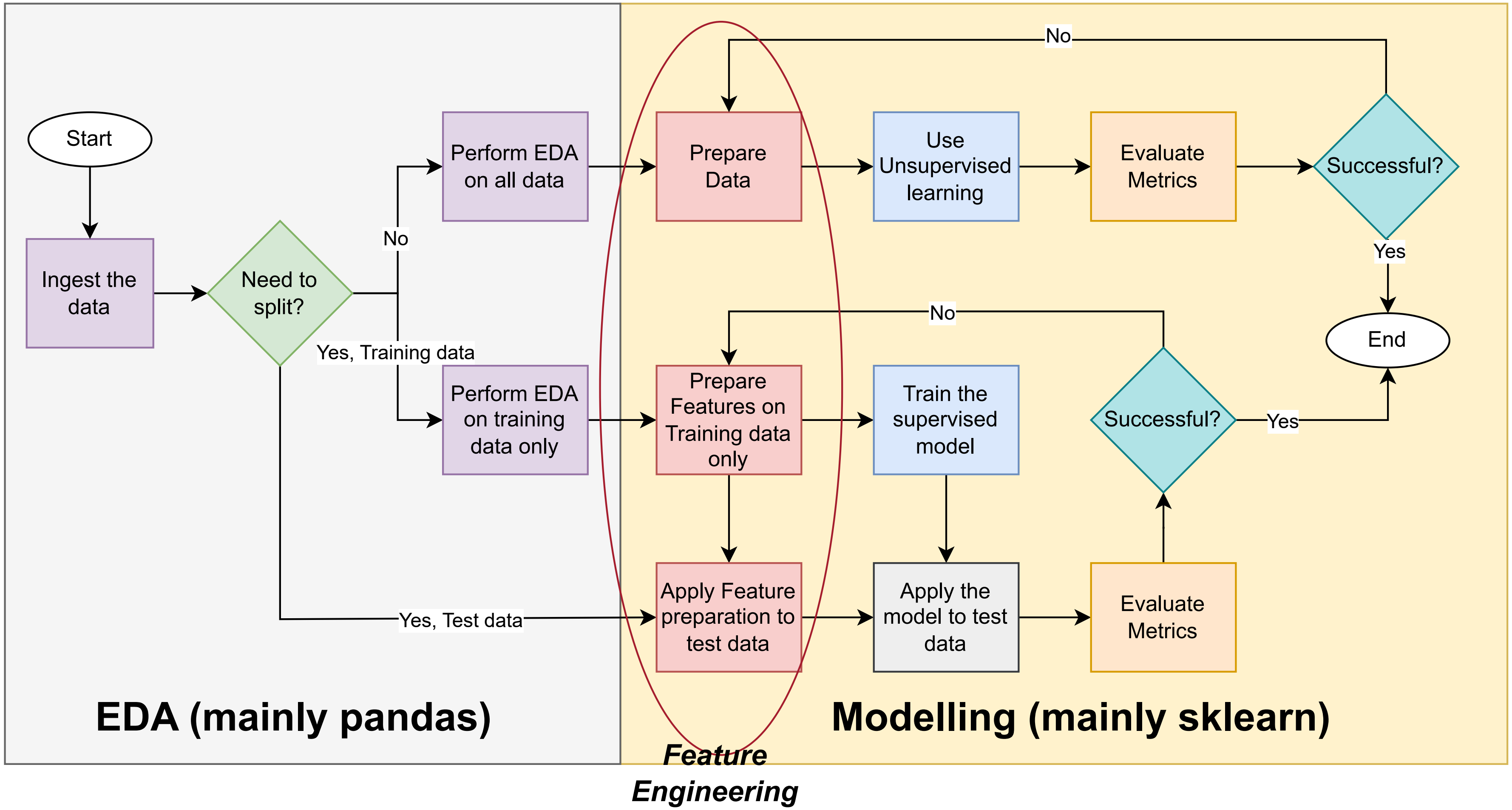
- How many to include? Use metrics to decide. Will see some when considering regression and classification.
- How do we handle different feature types? Need to encode categorical variables.
- Can we derive new numeric features? Yes, $f' = \log(f)$ etc. is possible

An overview of Machine Learning



On first glance, this might seem overwhelming, but note that the boxes are colour-coded, so that related operations share the same colour.

Where feature preparation fits. . .



NB: Feature preparation is informed by EDA *but is not part of EDA*. So, not part of your EDA assignment!

Feature Engineering more generally

- Use of pandas `getdummies()` requires care and is not always suitable
- Scikit-learn, imported as the `sklearn` package, supports a variety of column transformations
- Categorical columns can be features or targets, ordered or unordered
- Can also rescale numerical columns, or encode more exotic columns (other datatypes, computed columns, . . .)
- As seen in the schematic, if an ML procedure is unsuccessful, more feature engineering should be considered - it can help.

Summary

- We have reviewed different types of models and considered their general form.
- We looked at the goals of modelling: minimise predictive error.
- We considered how feature engineering can help.
  - Scaling numerical features, so that variation is treated fairly between features.
  - Choosing a subset of features (more to come in future weeks...), looking for the sweet spot between under- and over-fitting.
  - Encoding categorical features as numerical dummy features (more to come in future weeks...), so they can participate in linear models
- In subsequent weeks we will put this theory into practice.

Resources

- **A Summary of the Basic Machine Learning Models**
  towardsdatascience.com/a-summary-of-the-basic-machine-learning-models-e0a65627ecbe

- **Train-Test Split for Evaluating Machine Learning Algorithms**
  https://machinelearningmastery.com/
  train-test-split-for-evaluating-machine-learning-algorithms
  This week I have focused on the theory rather than its (python) implementation. This is a nice article that covers the implementation side of things.

- **Cross-Validation: Estimator Evaluator**
  medium.com/swlh/cross-validation-estimator-evaluator-897d28afb4ff
  Nice article that covers cross-validation in a lot more detail — we will be using many of these variants in later weeks, especially k-fold stratified.