

**iscte**

UNIVERSITY INSTITUTE OF LISBON

# **Machine Learning Processing Pipeline**

**Processamento e  
Modelação de Big Data**

# Outline

1. Recalling ML Concepts
2. ML Processing Pipeline

**Recalling ML  
Concepts**

**1**

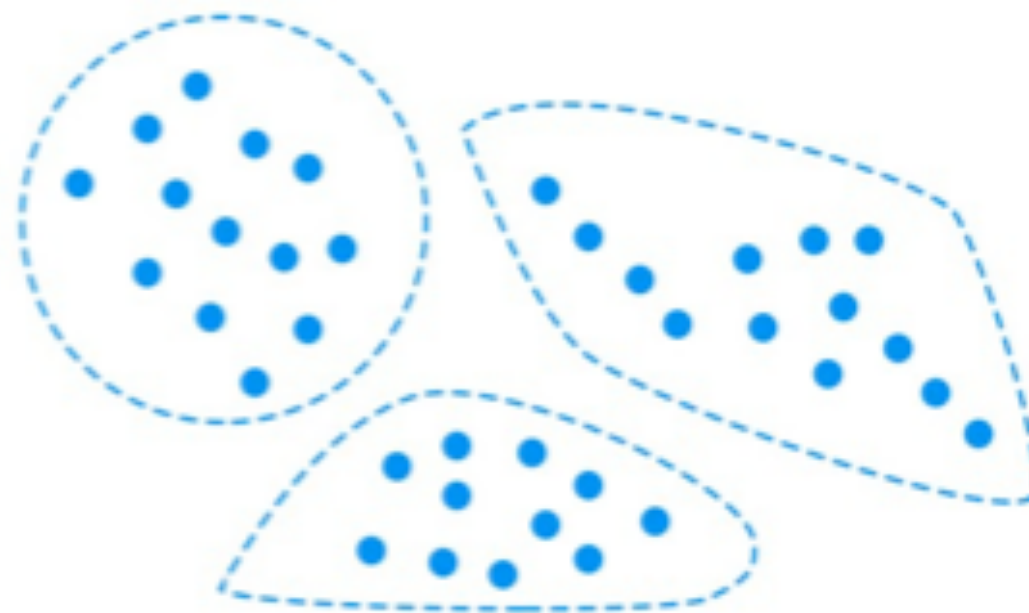
# Unsupervised learning

## Recalling

To uncover and create the labels itself

Clustering

Customer segmentation  
Targeted marketing  
Medical diagnostics  
...



Notice that it is important to profile the resulting clusters

# Supervised learning

## Recalling

Learn by identifying patterns in data that is already labeled

### Classification

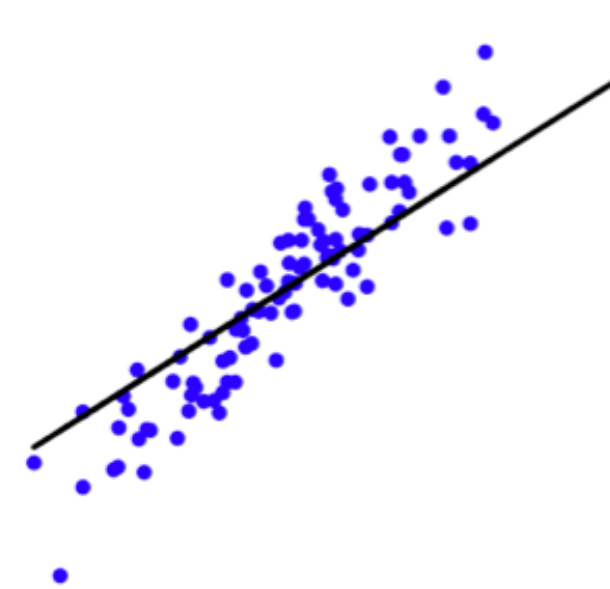
Binary [0, 1]

Fraud detection  
Image recognition  
Customer retention  
Medical diagnostics  
Personalised advertising  
...

Multi [0, 1, 2, ...]



### Regression



Product sales prediction  
Weather forecasting  
Market forecasting  
...

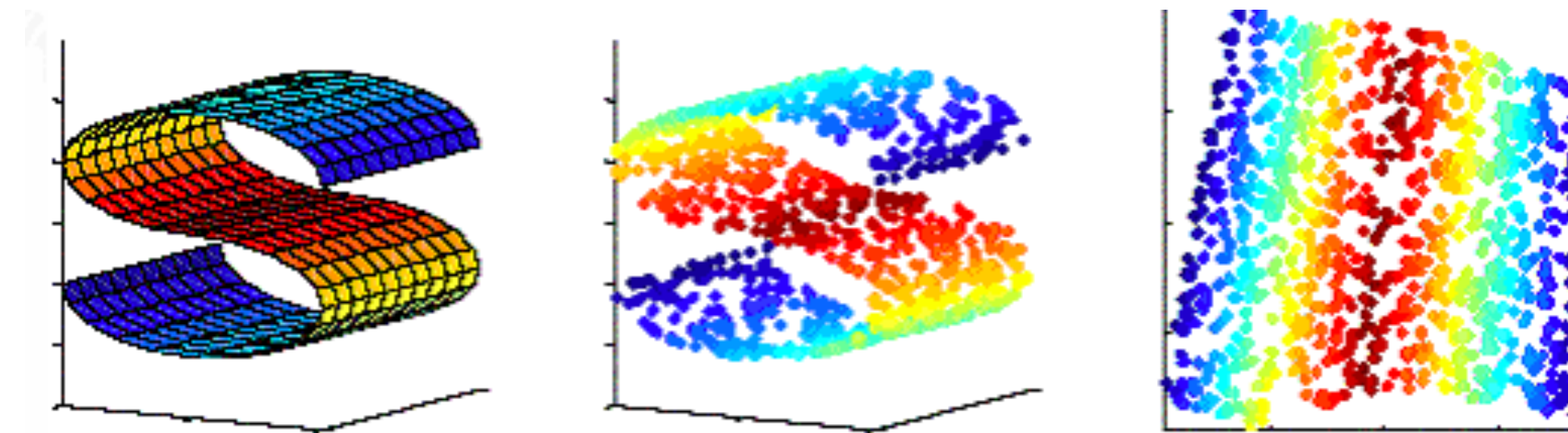
# Dimensionality reduction

## Recalling

Worth pointing out

Dimensionality  
reduction

Visualization  
Natural language processing  
Data structure discovery  
Gene sequencing  
...



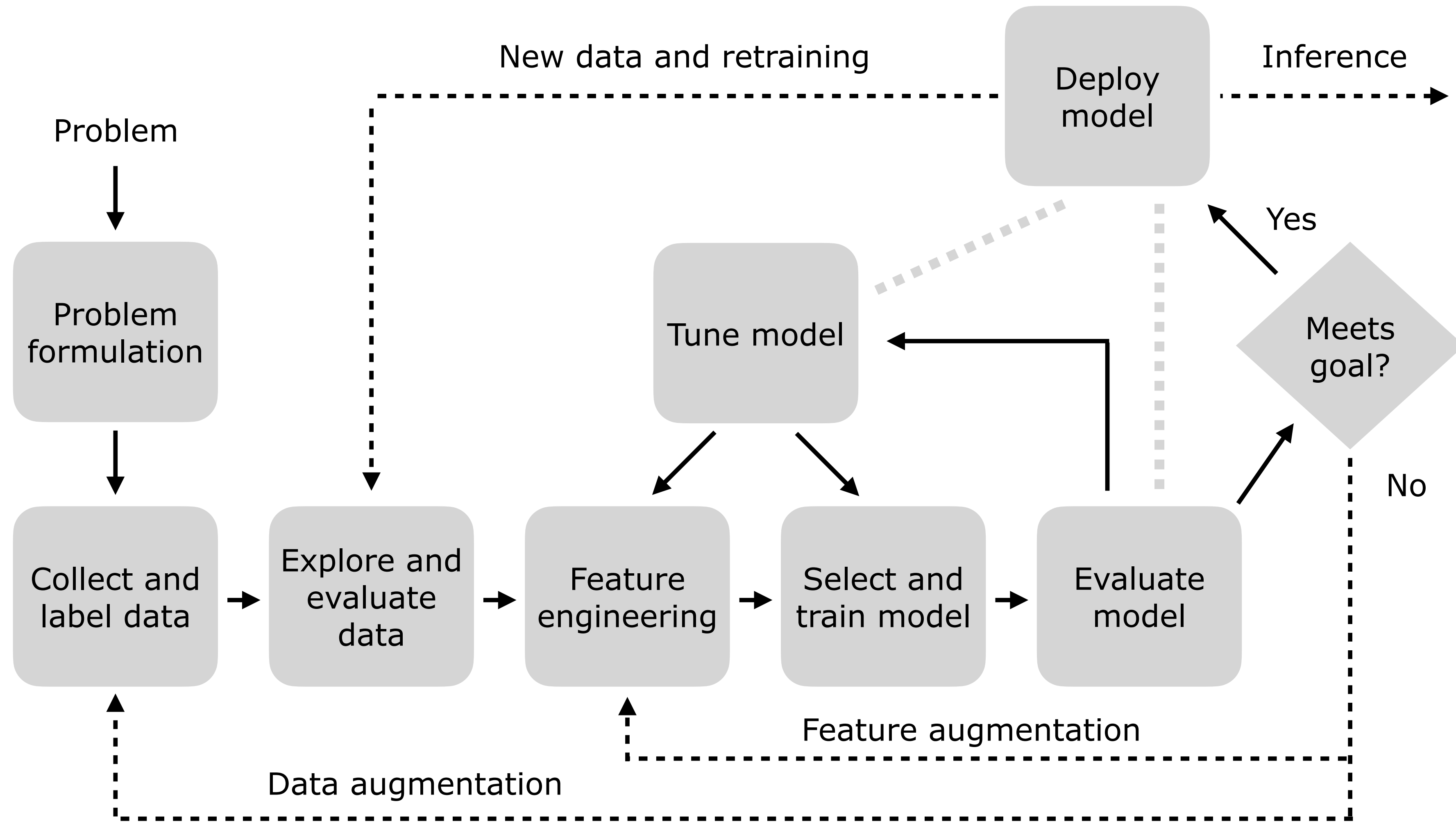
Using data features, the aim is to discover how to combine and reduce into fewer components

# ML Processing Pipeline

# 2

# ML pipeline

## Iterative process

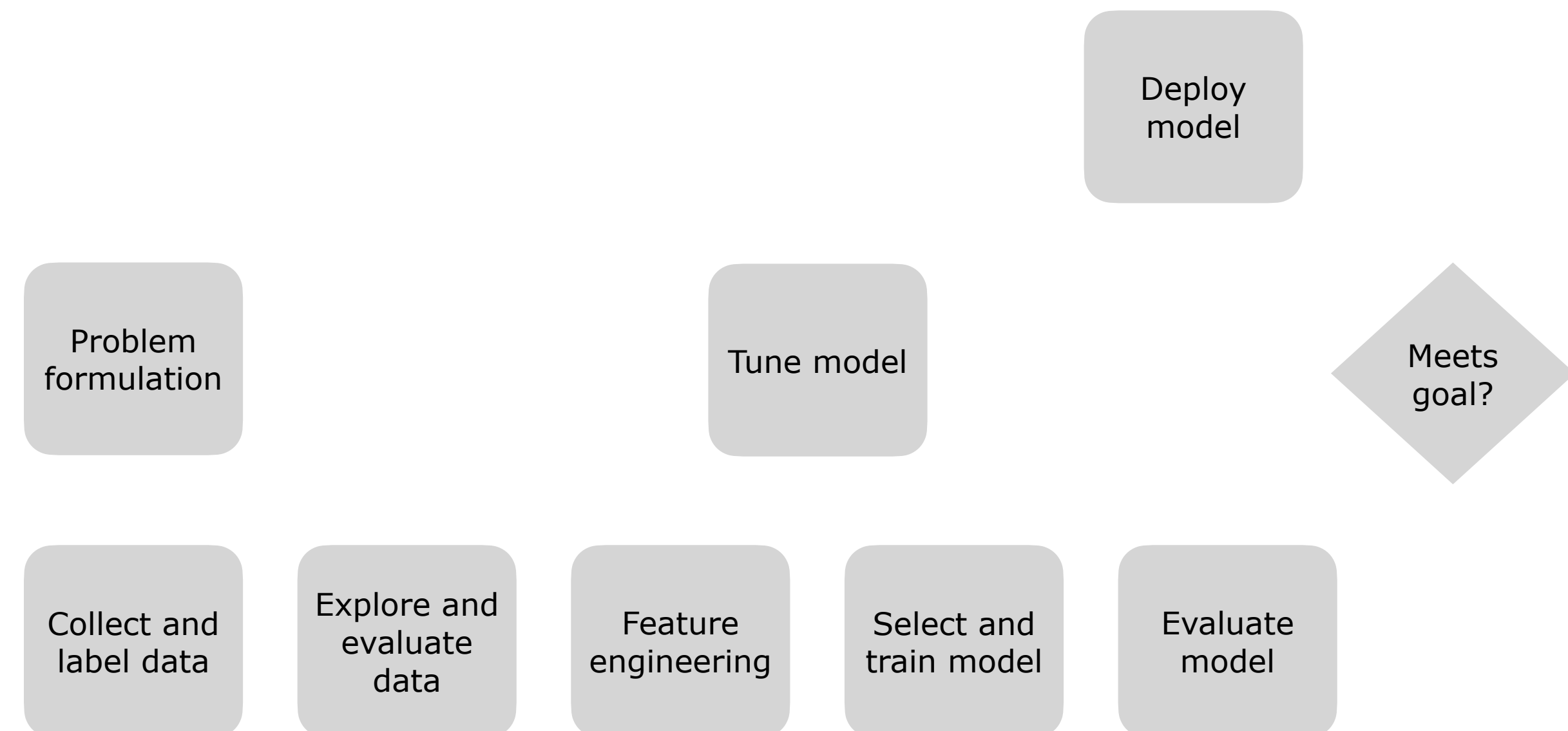




# ML pipeline

## Stages

- From a business/research requirement to a ML problem and datasets to use
- How to obtain and secure data for ML activities
- Tools and techniques for understanding data
- Preprocessing data so that it is ready to train a model
- Selecting and training an appropriate ML model
- Process of evaluating the performance of a ML model
- Selecting and training an appropriate ML model
- How to tuning the model
- How to deploy the model to infer possible outcomes



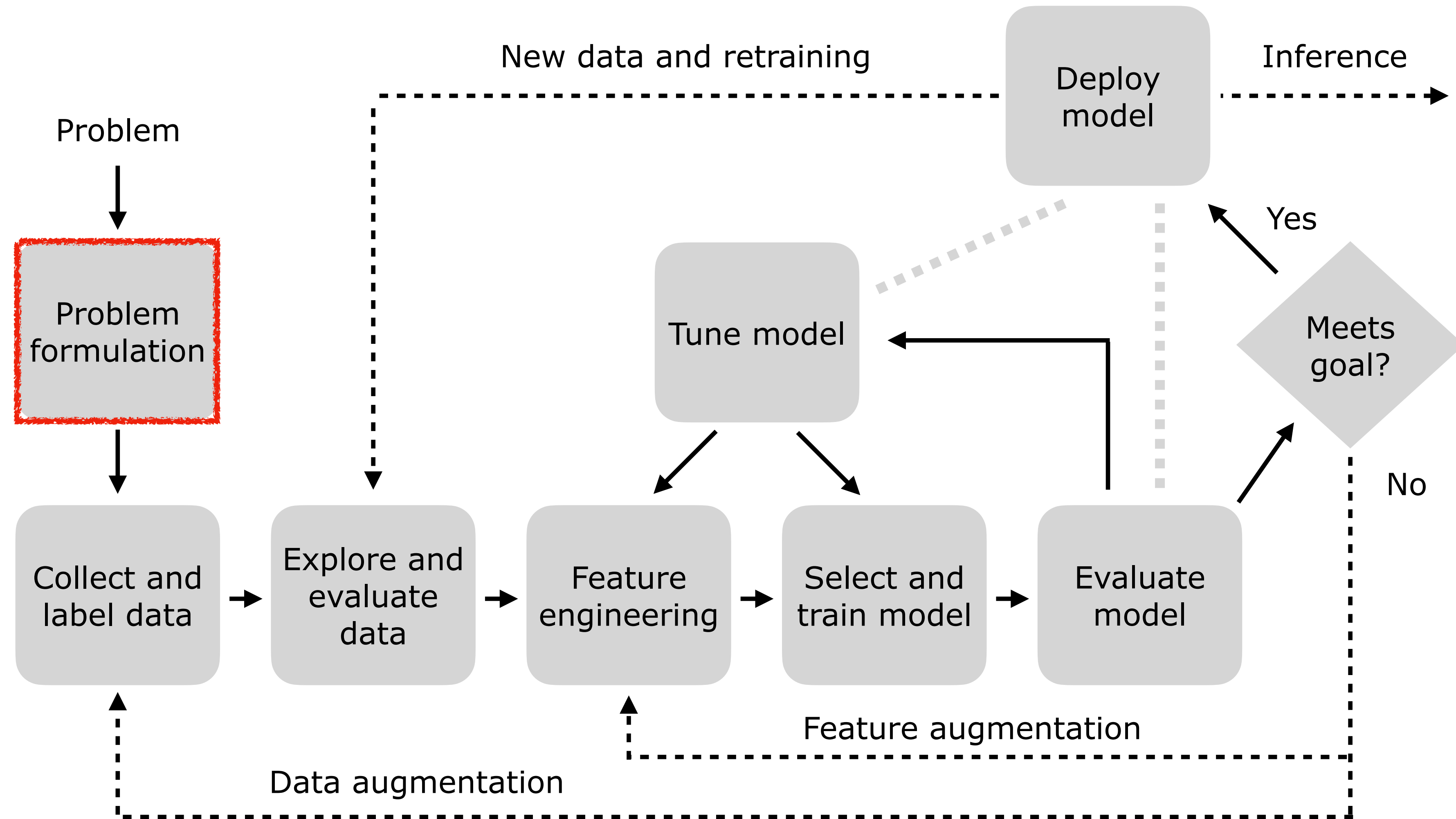
# ML Processing Pipeline

## Problem formulation



# ML pipeline

## Problem formulation



# Problem formulation

## Questions to formulate

- The goal
  - How things have been done in the past?
  - How to measure success?
  - How the solution is going to be used?
  - Are there other similar solutions?
  - Who are the domain experts?
  - ...
- Setting up the problem
  - Is the problem a ML problem?
  - Is it a problem supervised, unsupervised, or of more advanced type?
  - What is the target to predict?
  - What about access to data?
  - What is the minimum performance required?
  - ...

# Problem formulation

## Example of problem formulation

- Goal

To identify fraudulent credit card transactions so that we can stop the transaction before it processes



- Why?

To reduce the number of customers who end their membership because of fraud



- How to measure?

Move from qualitative statements to quantitative statements that can be measured

**5%** reduction in  
fraud claims in retail

- What kind of ML problem is and consequent model to use?

- Credit card transaction is either fraudulent or not. So it should be a binary classification problem

- Use of historical data of fraud reports to help defining the model

**Fraud**  
**Not fraud**

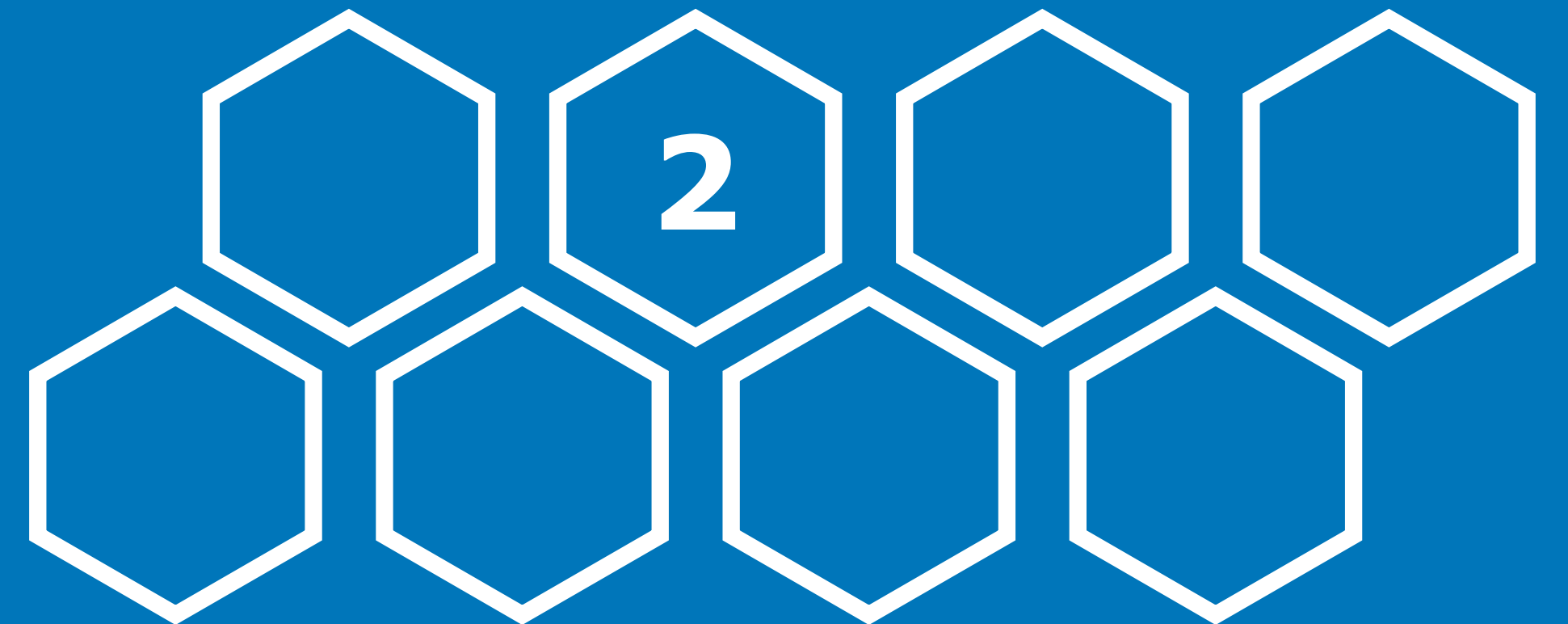
# Problem formulation

## Datasets

- Given the datasets available, can we answer the questions of interest?
- Examples, considering datasets available at UCI ML repository, ([archive.ics.uci.edu](http://archive.ics.uci.edu))
  - Based on the composition of the wine, can we predict the quality of the wine and its price?
  - Can we use a car's attributes to predict whether the car will be purchased or not?
  - Based on the bio-mechanical features, can we predict if a patient has an abnormality?
- Some data processing techniques that may be needed to apply
  - View statistics
  - Deal with outliers
  - Scale numeric data
  - Encode categorical data

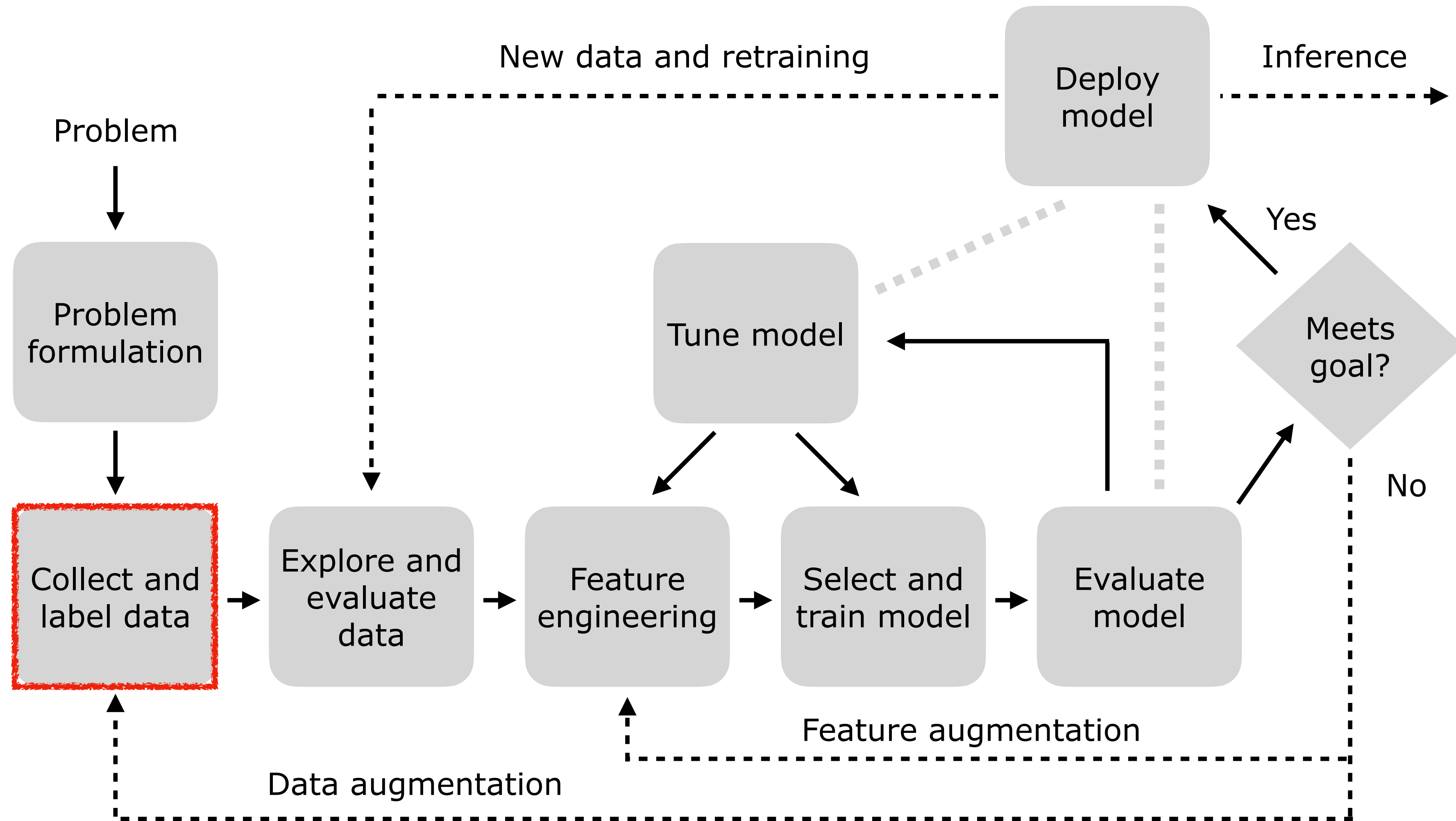
# ML Processing Pipeline

## Collect and label data



# ML pipeline

## Collect and label data





# Collect and label data

## What data is needed

- How much, where it is and access granted
- Domain experts to validate its representativity
- Sources
  - Private: created by people in general
  - Commercial: external providers
  - Open: publicly available, like from World Health Organisation, Kaggle, governamental entities, AWS, etc.

# Collect and label data

## Extract, transform, load (ETL)

- Extract

Pulling data from various sources to a single location

- Transform

During extraction, data might need to be modified, matching records might need to be combined, or other transformations might be necessary

- Load

Finally, data is loaded into a repository

## **Collect and label data**

## **Securing data**

- Identity and access management (control)
- Data encryption
- Compliance audits

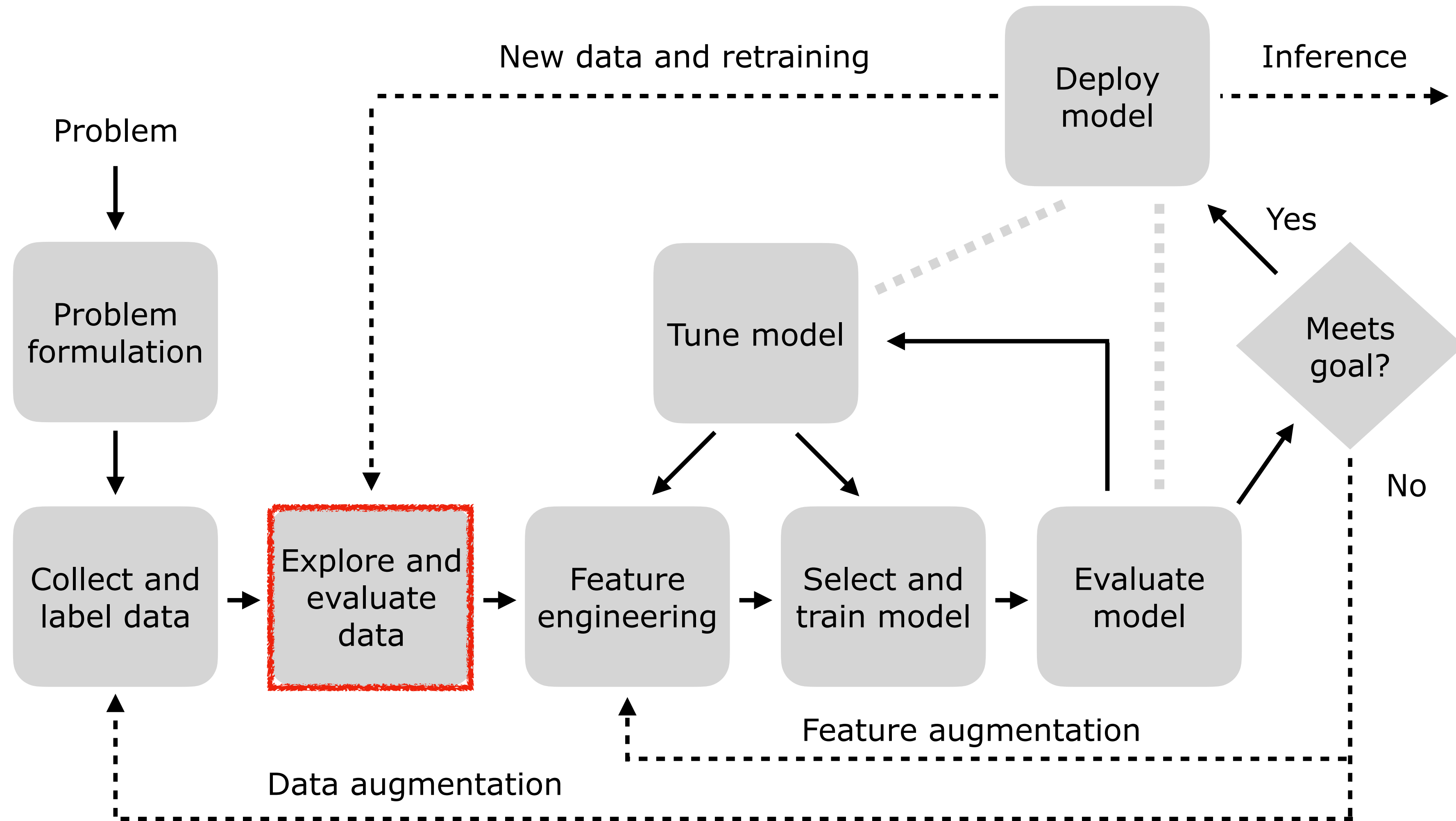
# ML Processing Pipeline

## Explore and evaluate data



# ML pipeline

## Explore and evaluate data



# Evaluate data

## Understanding data

- Before running statistics to better understand data, it has to be in the right format for analysis, e.g. csv, json
- Use of toolkits like Spark ML or Pandas library, based on the concept of DataFrame and its tabular schema
  - Rows - instances (or observations)
  - Columns - data attributes
- Use of correct data types
- Use of descriptive statistics to gain insight into data, prior to clean it, with focus on visualisation

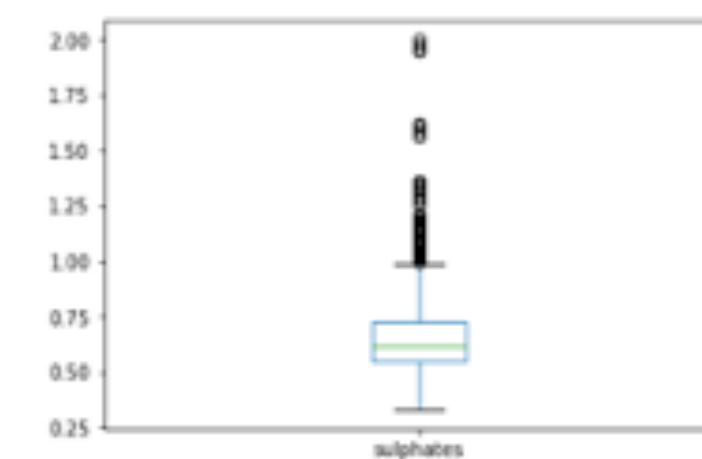
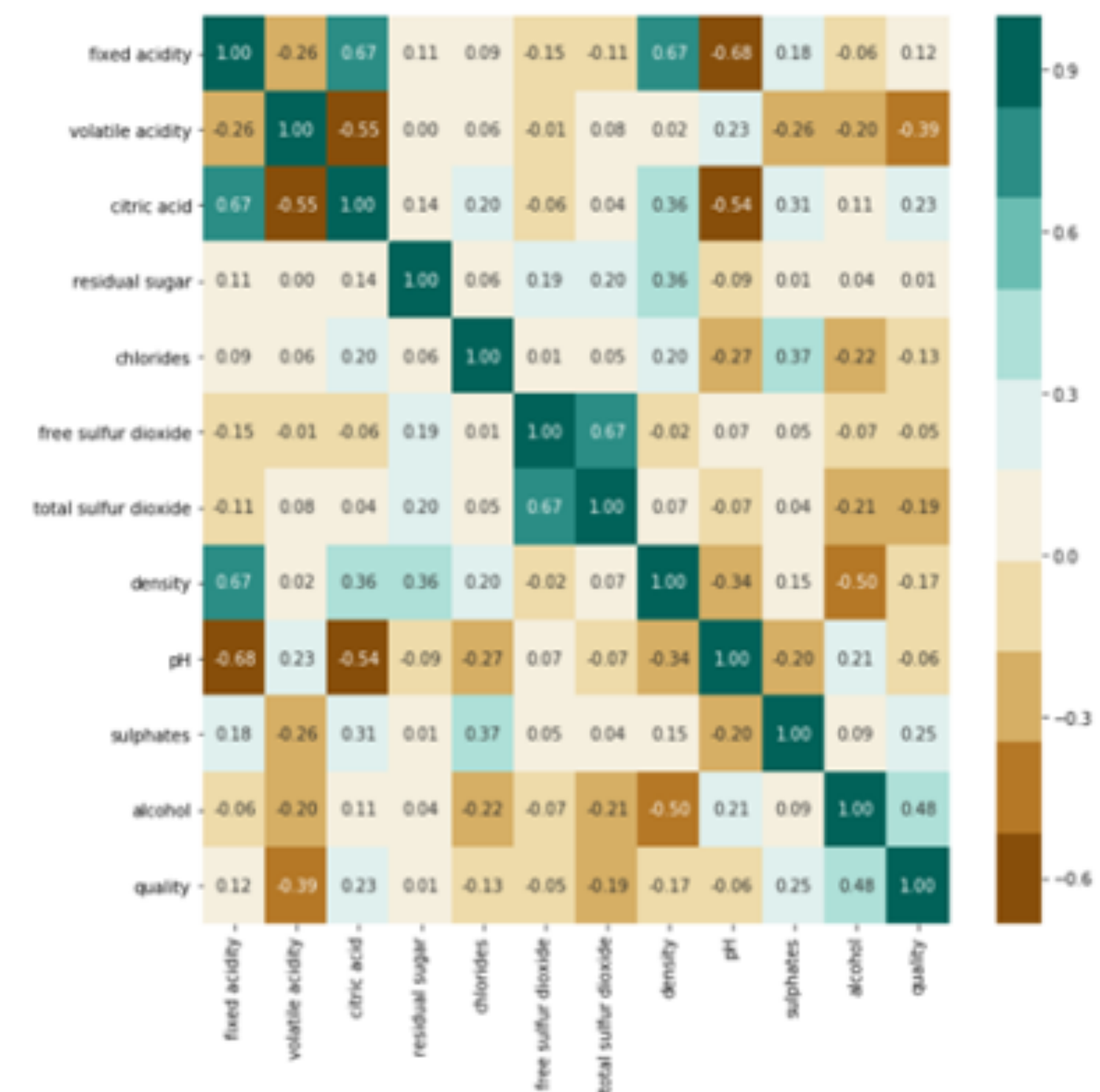
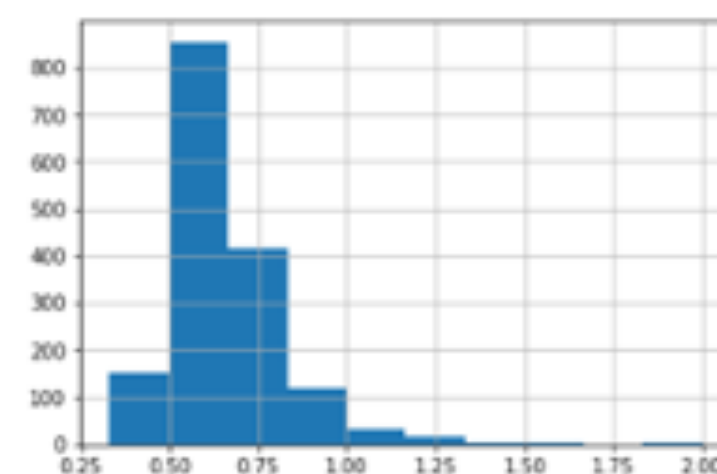
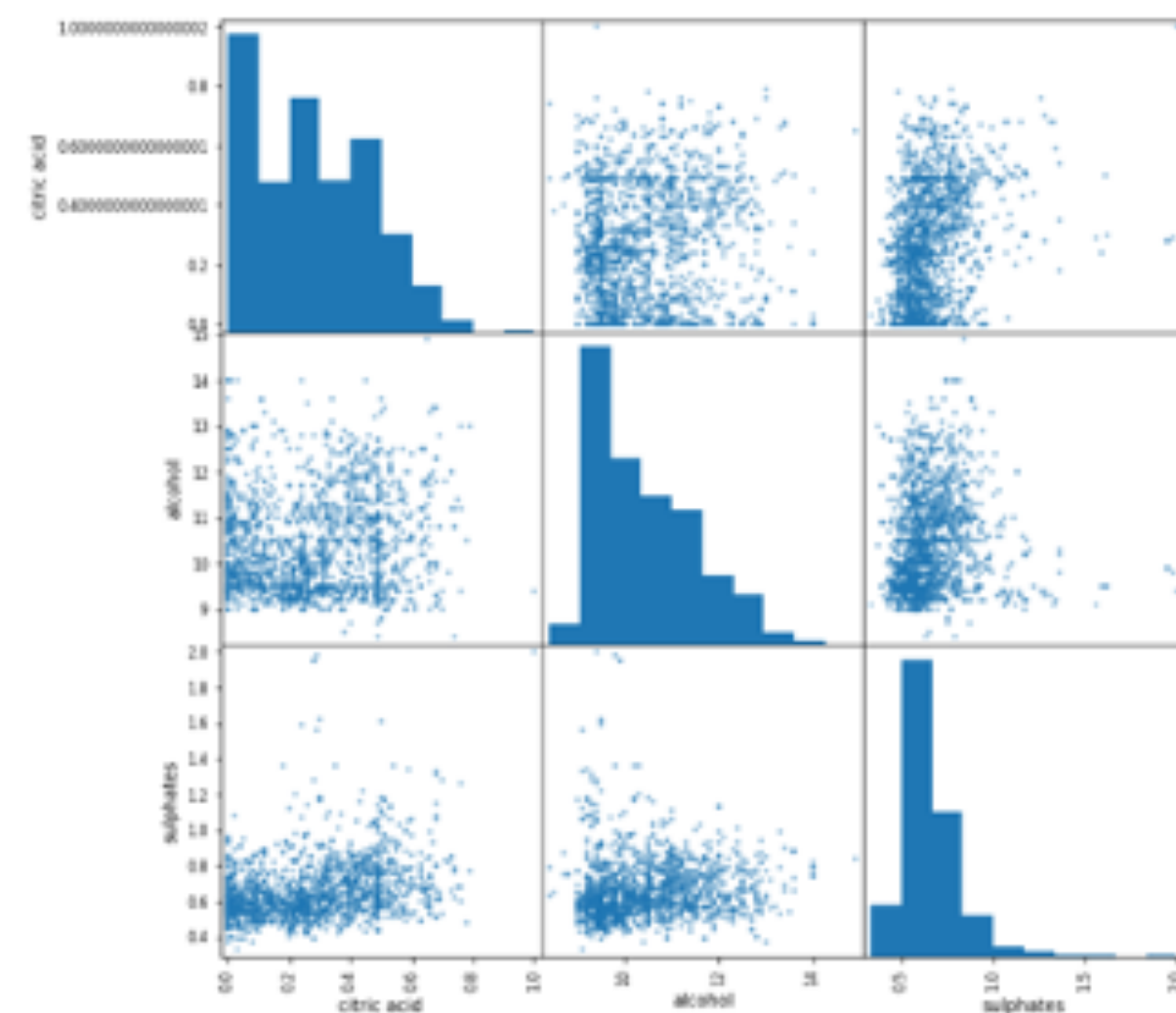
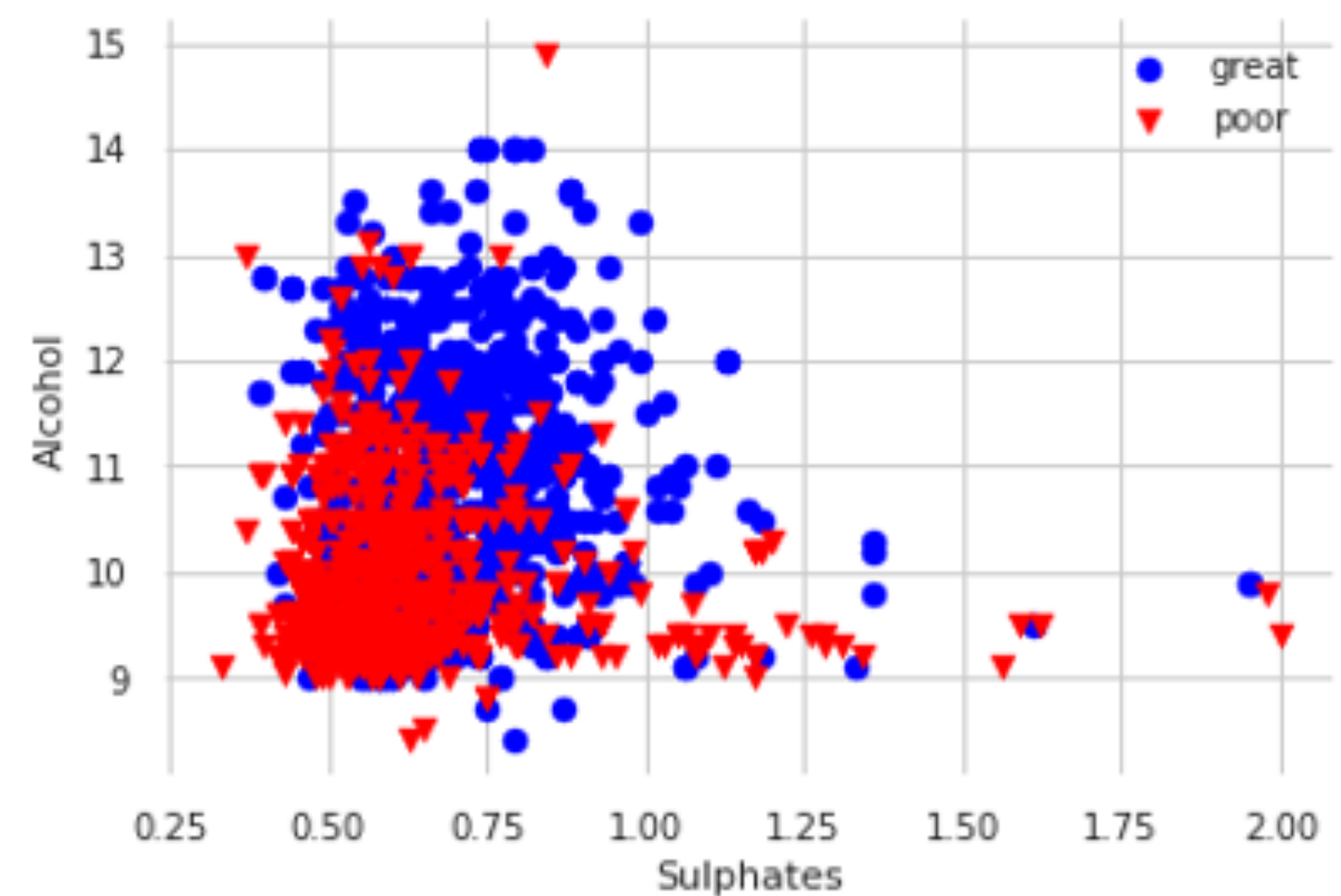
# Evaluate data

## Descriptive statistics

- Statistical characteristics
  - count, mean, std, min, max, etc.
- Categorical statistics
  - To identify frequency of values and class imbalances
- Plotting attribute statistics
  - Histograms, plot box, etc.
- Plotting multivariate statistics
  - Scatter plot, scatter matrix, correlation matrix, etc.

# Evaluate data

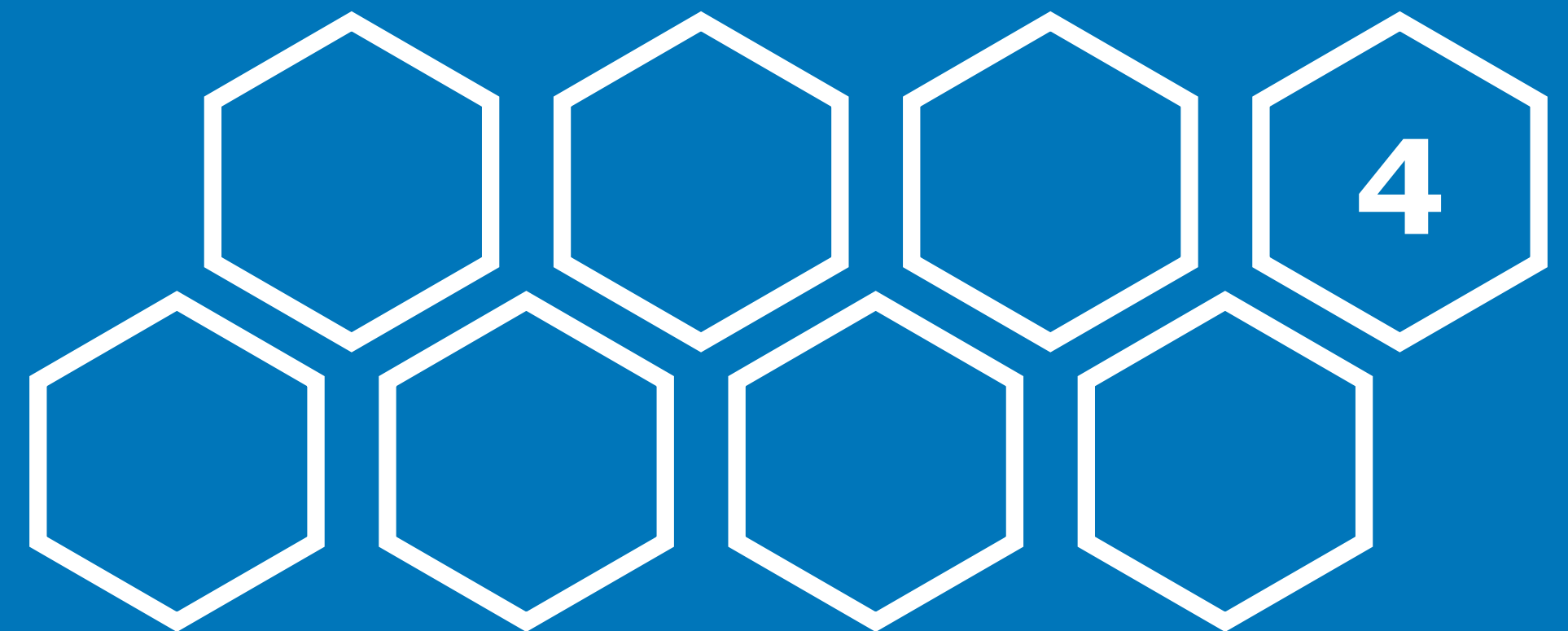
## Descriptive statistics





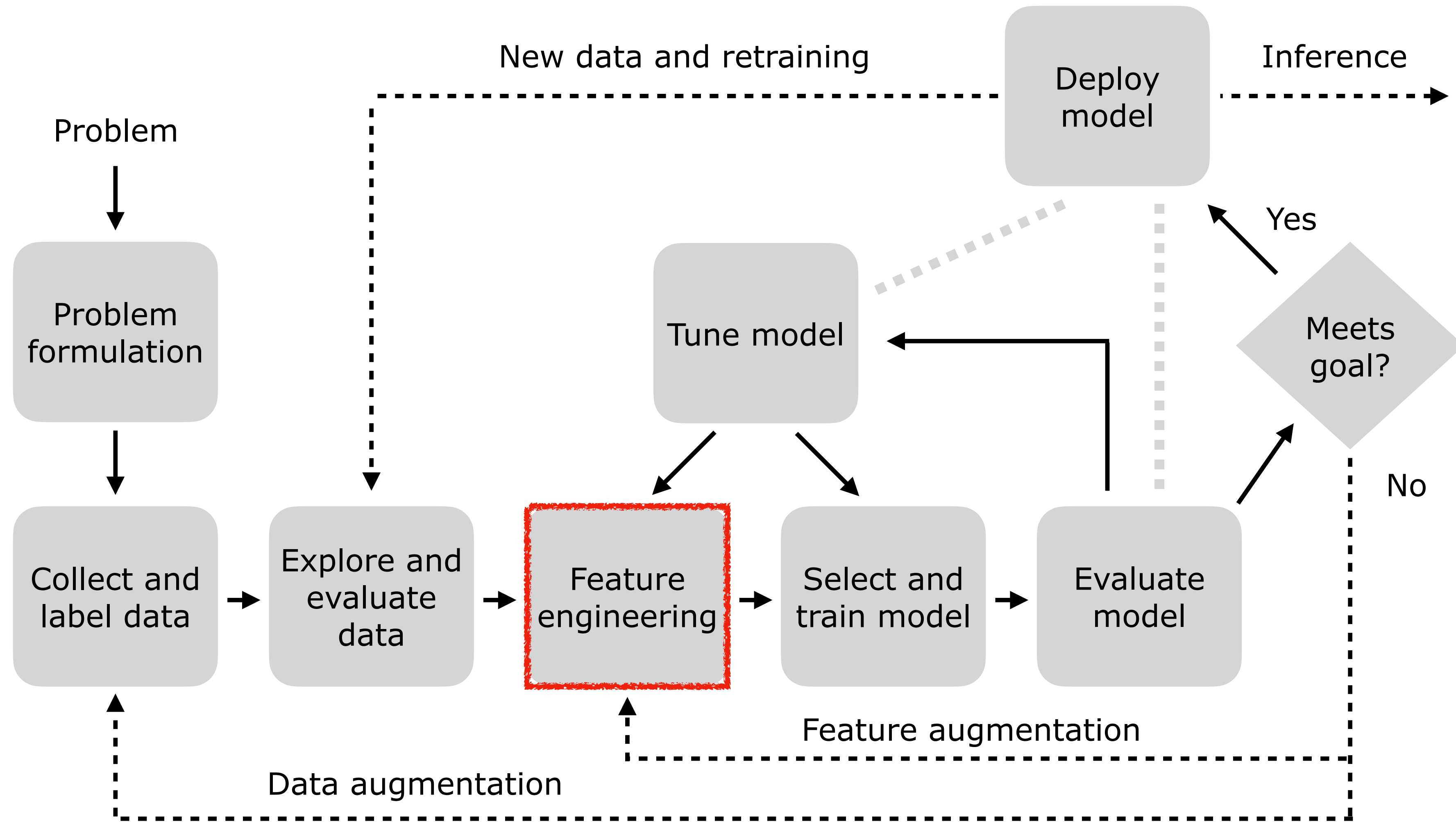
# ML Processing Pipeline

# Feature Engineering



# ML pipeline

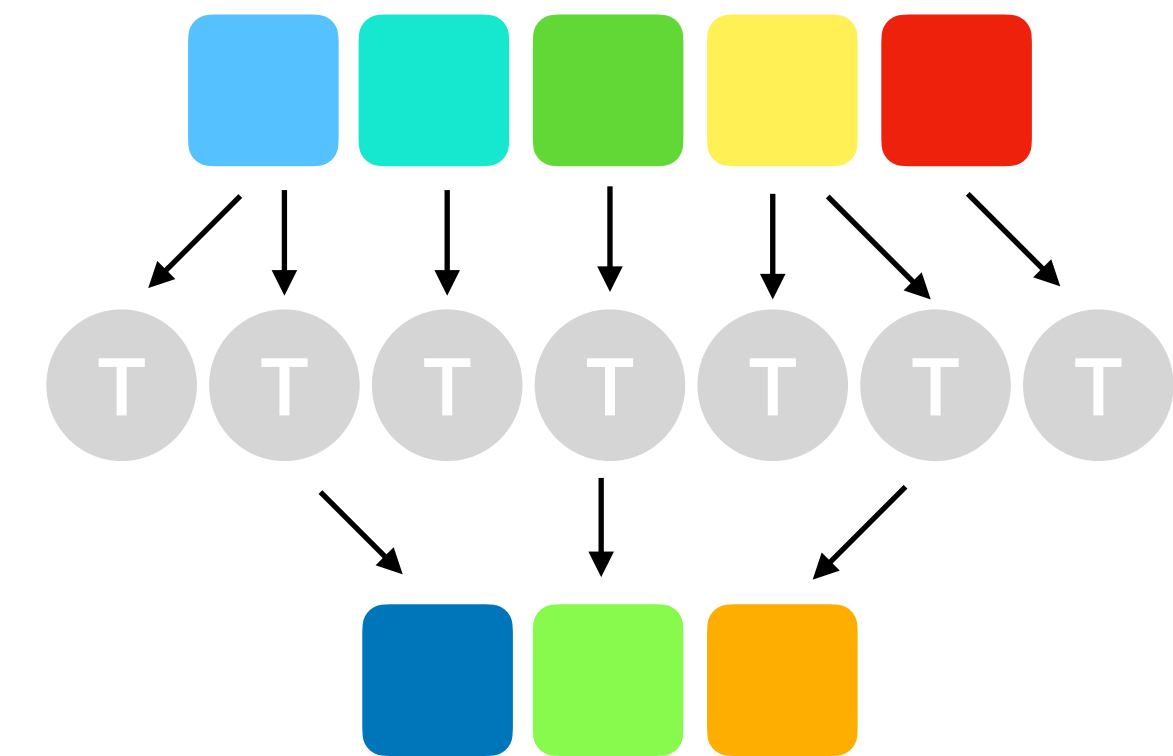
## Feature engineering



# Feature engineering

## Extraction and selection

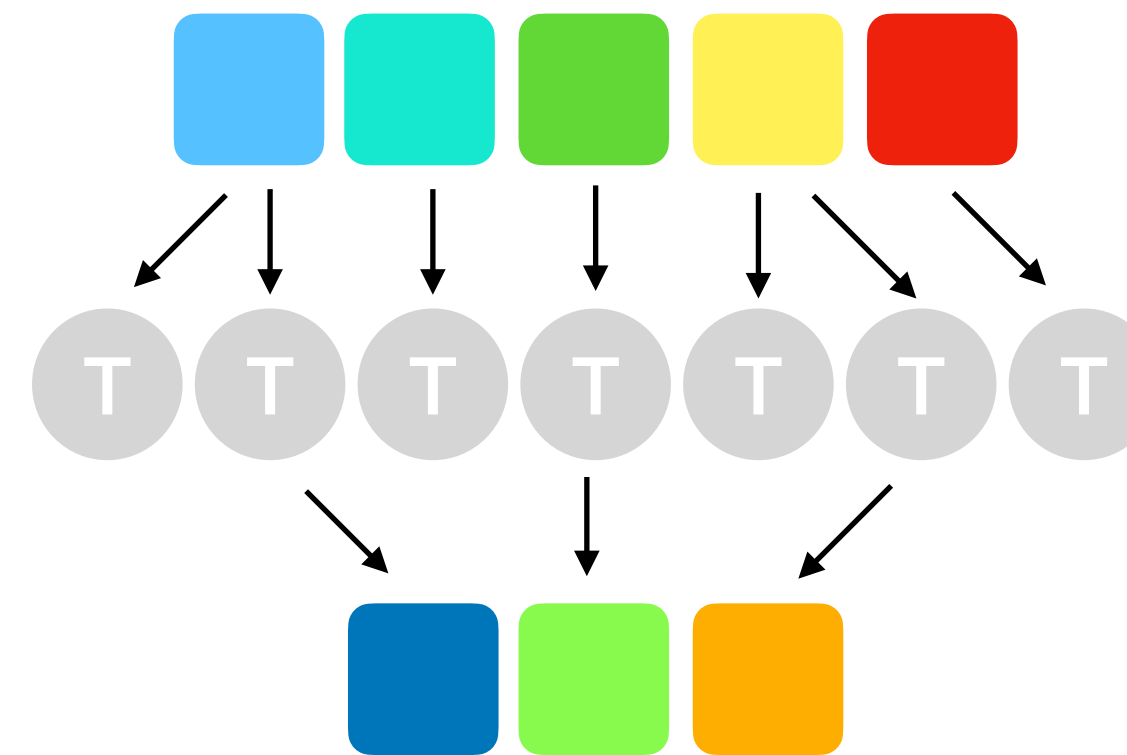
- Building up valuable information from raw data by reformatting, combining, and transforming primary features into new ones
- Selecting the columns of data that make the most impact in the model
  - To prevent either redundancy or irrelevance in the existing features
  - To get a limited number of features in order to prevent overfitting



# Feature engineering

## Somes tasks in feature extraction

- Invalid values
- Wrong formats
- Misspelling
- Duplicates
- Consistency
- Rescale
- Encode categories
- Remove/reassign outliers
- Bucketing
- Decomposition
- Aggregation
- Combination
- Transformation
- Normalisation
- Dimensionality reduction



Some more ML algorithms related:

- Text to numbers
- Outliers
- Potentially rescaling

# Feature engineering

## Data encoding - text to numbers

- Ordinal data

Categorical data is non-numeric but it has to be encoded to a numeric scale, as required by ML algorithms

Level	Encoding
Low	1
Medium	2
High	3
Very high	4

- Non-ordinal data

Encoded values still non-ordinal but broken into multiple categories holding numbers

...	Fuel
...	Gasoline
...	Diesel
...	Gasoline
...	Kerosene



...	Gasoline	Diesel	Kerosene
...	1	0	0
...	0	1	0
...	1	0	0
...	0	0	1

# Feature engineering

## Data cleaning - some situations

- Some problems to sort out
  - Strings not consistent
  - Variables not using consistent scales
  - Data items representing more than one variable so they should be split
  - Missing data for some variables
  - Outliers in the data
- Non-ordinal data
  - Encoded values still non-ordinal but broken into multiple categories that hold numbers

# Feature engineering

## Data cleaning - some situations

Problem to sort out	Example	Potential action
Strings not consistent	Med. vs Medium	Convert to standard text
Not using consistent scales	Cars: number of doors vs number of cars sold (e.g. thousands)	Normalize to a common scale
Columns representing more than one variable	Safety and maintenance features	Parse into multiple columns
Missing data	Missing columns of data	Delete rows or impute values
Outliers	Various	More complex and requires statistical analysis

# Feature engineering

## Missing data

- Context
  - It is not easy to interpret relationships in data if some is missing
  - Missing data can be due to undefined values, or errors related to collection or cleaning
- Action to take can be **dropping** or **imputing** (replacing) missing values.  
But first, one has to ask:
  - What mechanisms have caused the missing values?
  - Are missing values randomly spread?
  - Are there more rows or columns that we are not aware of?



# Feature engineering

## Missing data

- **Dropping**

- Example of column or row with large percentage of missing data
- Dropping all rows with missing data, or just drop specific values, from a subset

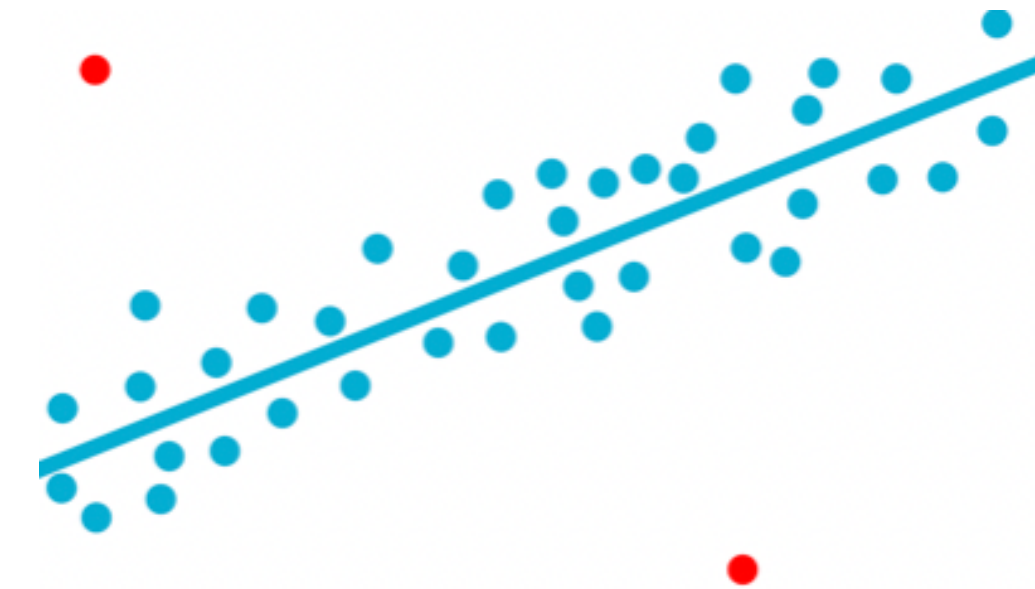
- **Imputing**

- Potential use if missing values are randomly spread
- It can be either a single row (univariate) or multiple rows (multivariate) of missing data
- For categorical variables, usually replacement is with the mean, median, or most frequent values
- For numerical or continuous variables, usually replacement is with mean or median
- There are some tools to help the task, even using more advanced methodologies

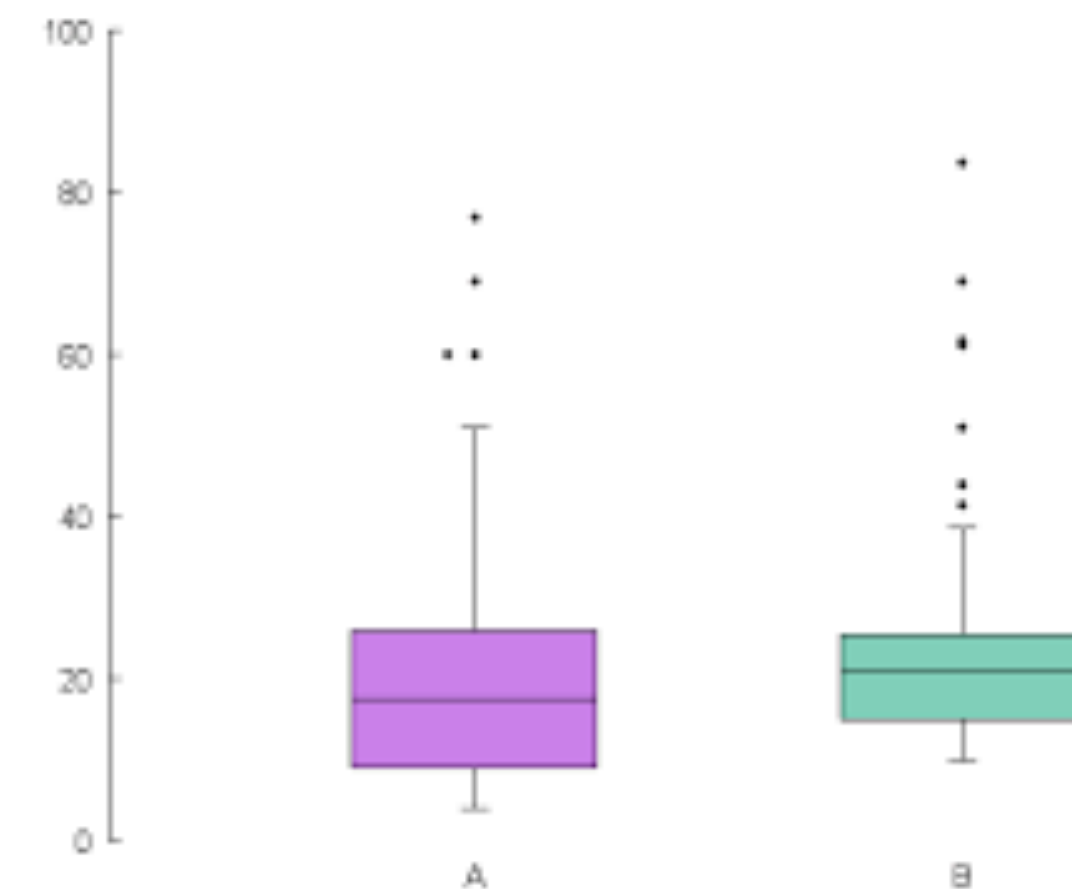
# Feature engineering

## Outliers

- Characterisation
  - Provide a broader picture of the data
  - Accurate predictions are more difficult because outliers skew values away from the ones more normal
  - We can have abnormal values for one or combination of various variables
  - They may suggest the need for more columns
  - Nonetheless, implied enrichment of data due to outliers cannot be discarded
- Discovery, with help of statistics and visualisation



Scatter plots depicting multivariate outliers



Box plot showing variation and distance from the mean

# Feature engineering

## Actions to dealing with outliers

- **Delete**

When the outlier is based on an artificial error e.g. data entered incorrectly

- **Transform**

To reduce the variation that the extreme value causes, so the outlier's influence, e.g. by using the natural log of the value

- **Impute**

To replace the outlier value, e.g. by using the mean of the feature (instead of simply deleting the outlier)

# Feature engineering

## Methods in feature selection

- **Filters**

Use of statistics to measure the relevance of features by their correlation with the target

- **Wrappers**

- Train the model
- Measure the usefulness of a subset by training a model on it and measuring its success

- **Embedded**

- Integrated with model
- Algorithm-specific and might use a combination of filters and wrappers



Fast and cheaper (no repeated training)



Usually get the best subset of features but with overfitting risk (as opposed to filter's subsets of features)

# Feature engineering

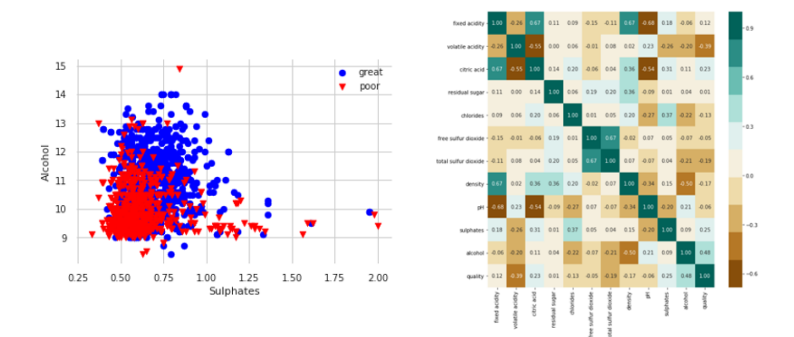
## Filter methods in feature selection

- Use a proxy measure instead of the actual model's performance
- Fast to compute, and still capturing the usefulness of the feature set
- Some measures
  - Pearson's correlation coefficient - Measures the statistical relationship between two continuous variables
  - Linear discriminant analysis (LDA) - Used to find a linear combination of features that separates two or more classes
  - Analysis of variance (ANOVA) - Used to analyse differences among group means in a sample
  - Chi-square - Single number that tells how much difference exists between observed and expected counts, if no relationship exists in the population

All



Statistics  
and  
correlation



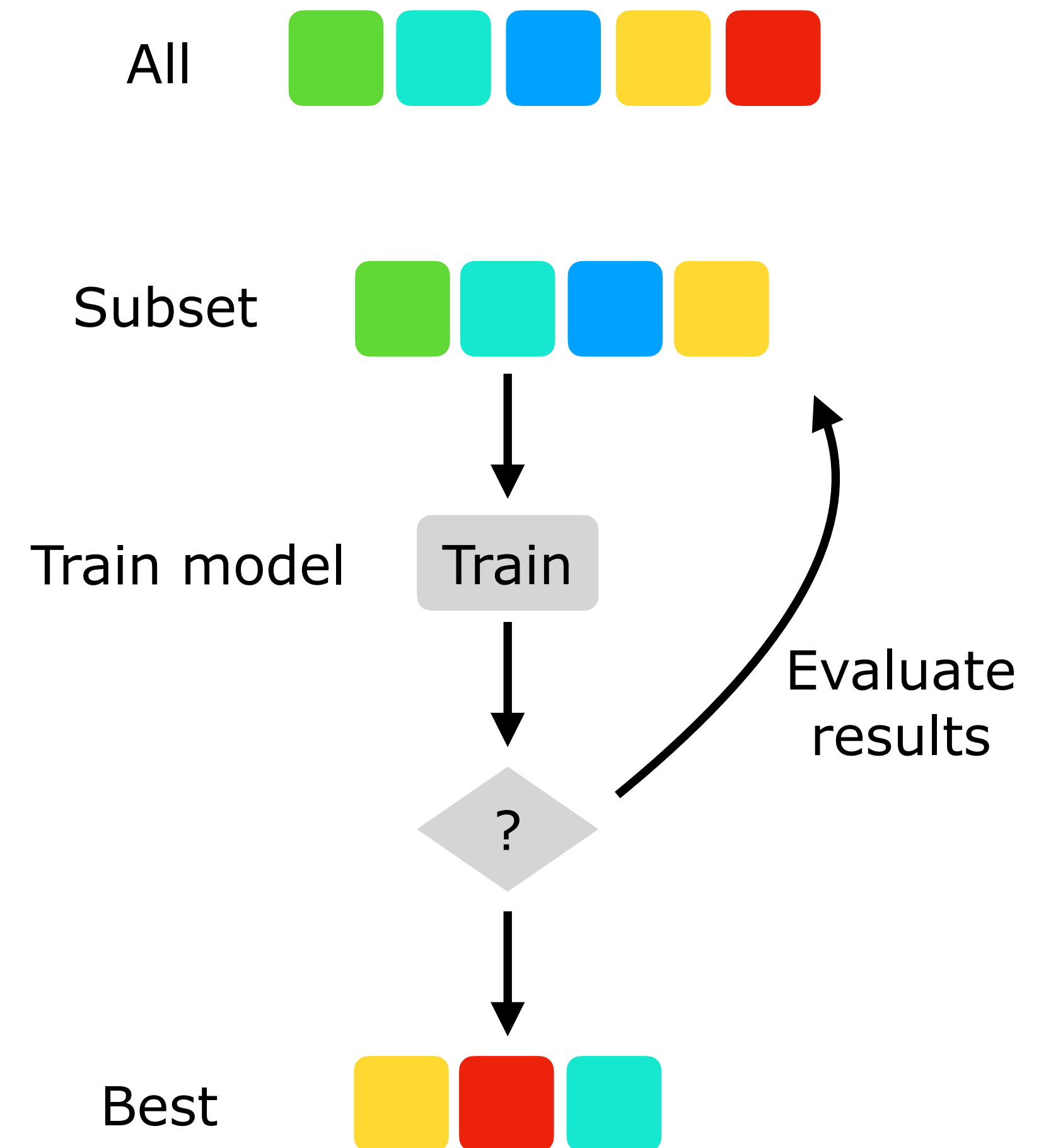
Best



# Feature engineering

## Wrapper methods in feature selection

- Predictive model to score feature subset
  - Each new subset is used to train a model and gets a score for that subset
  - Computationally intensive as it trains a new model for each subset
  - But they usually provide the best-performing feature set
- Selection can be
  - Forward (starts with no feature, then adds one at a time)
  - Backward (starts with all features, then drops one at a time)

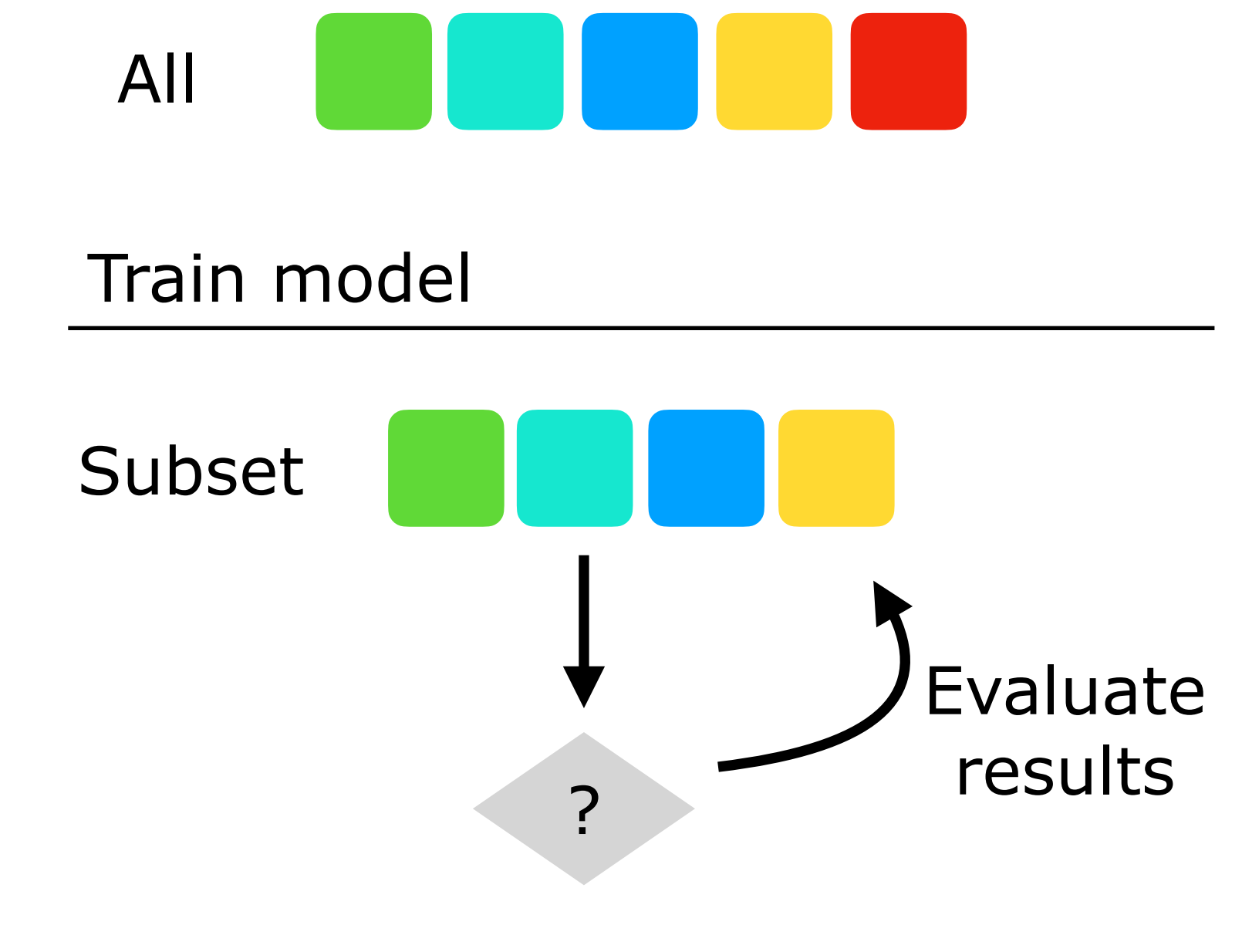


# Feature engineering

## Embedded methods in feature selection

- Merging qualities from filter and wrapper methods
- Examples

LASSO and RIDGE regression, with built-in penalisation functions to reduce overfitting



# ML Processing Pipeline

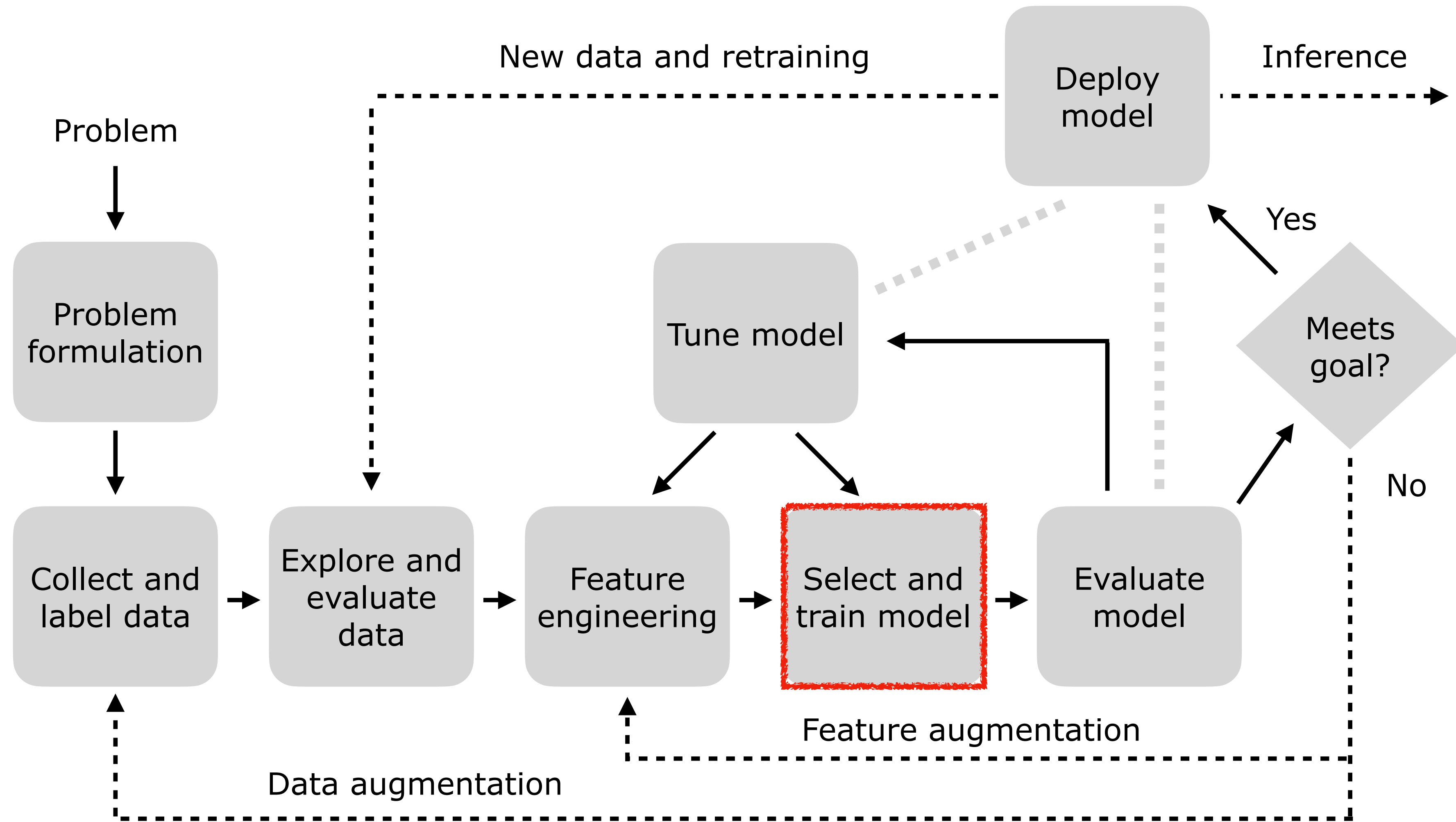
## Select and train model





# ML pipeline

## Select and train model



# Select and train model

## Formats

Although data has already been clean, and with a defined target and various features, ML algorithms may impose further constraints in relation to

- File formats, e.g. by not processing data frames as-is but text only
- Requiring numeric data instead of textual data so sort of mapping is needed

# Select and train model

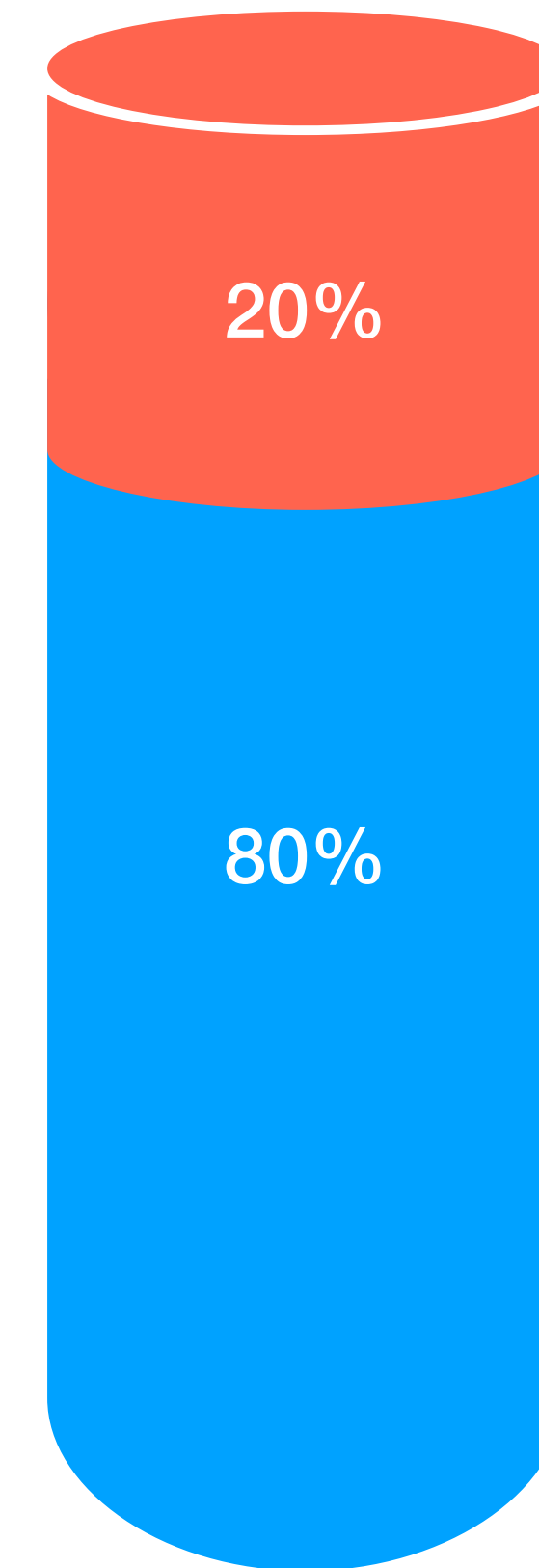
## Data splitting in classification tasks

- In order to cross validate later on, data will be split into
  - Training data - with features and labels, feeding the algorithm to produce the model
  - Validation data - with purpose of noticing things to tweak, tune and change
  - Testing data - with the features to finally predict labels. What we get here is what we expect to get in production
- Common split size
  - 80% for training, 10% for validation and 10% for testing
  - If data sets are very large, it may be 70%, 15%, 15% respectively

# Select and train model

## Basic scenario for data splitting

- But in a basic scenario, data is split into just train and test sets. After training the model with the train set, errors are evaluated on the test set
- Only then we might move to an enhanced cross validation process. This would be fairer as we would get metrics from a final test set that we have not made adjustments based upon it



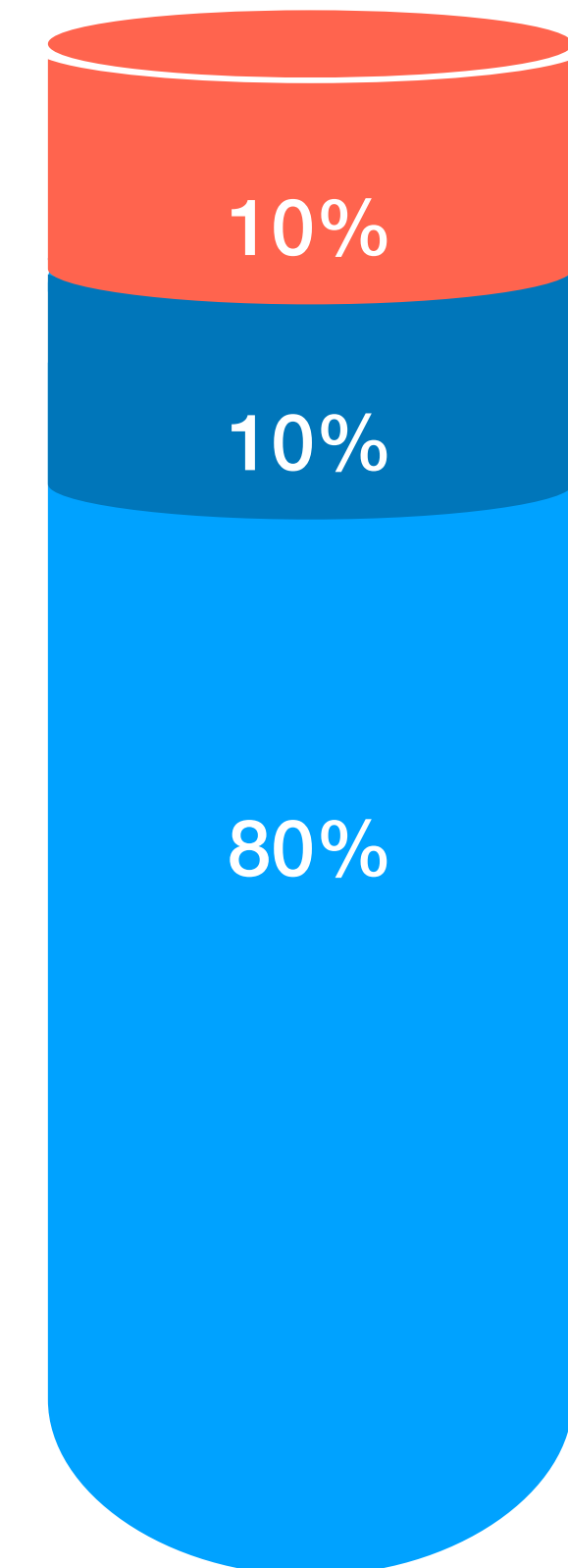
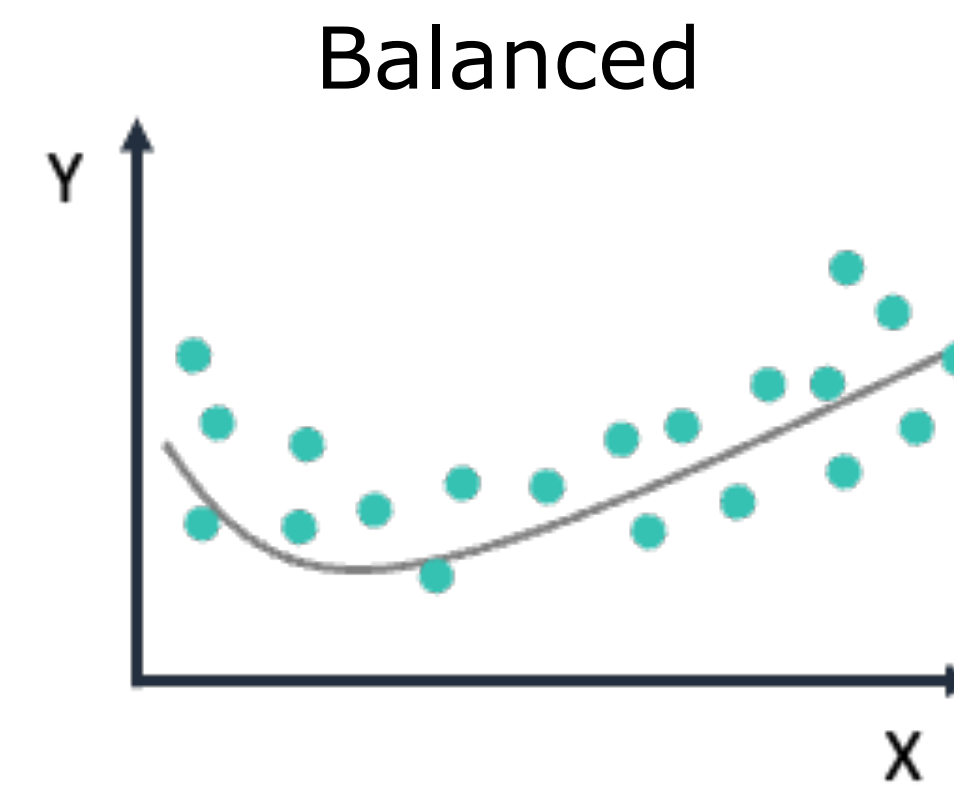
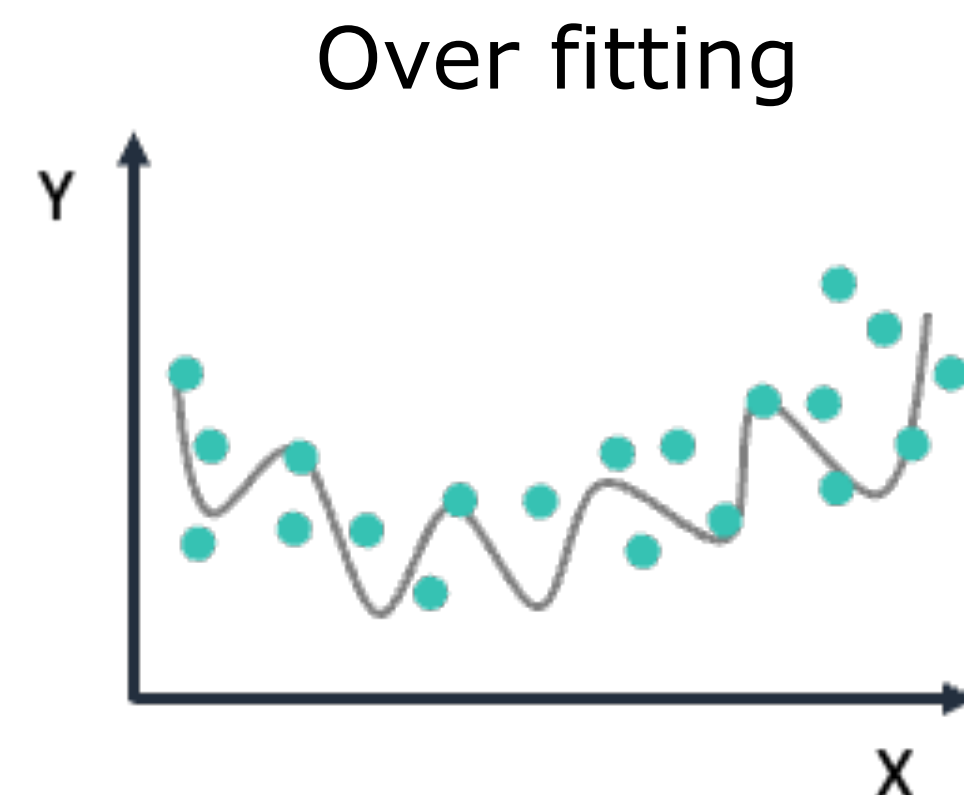
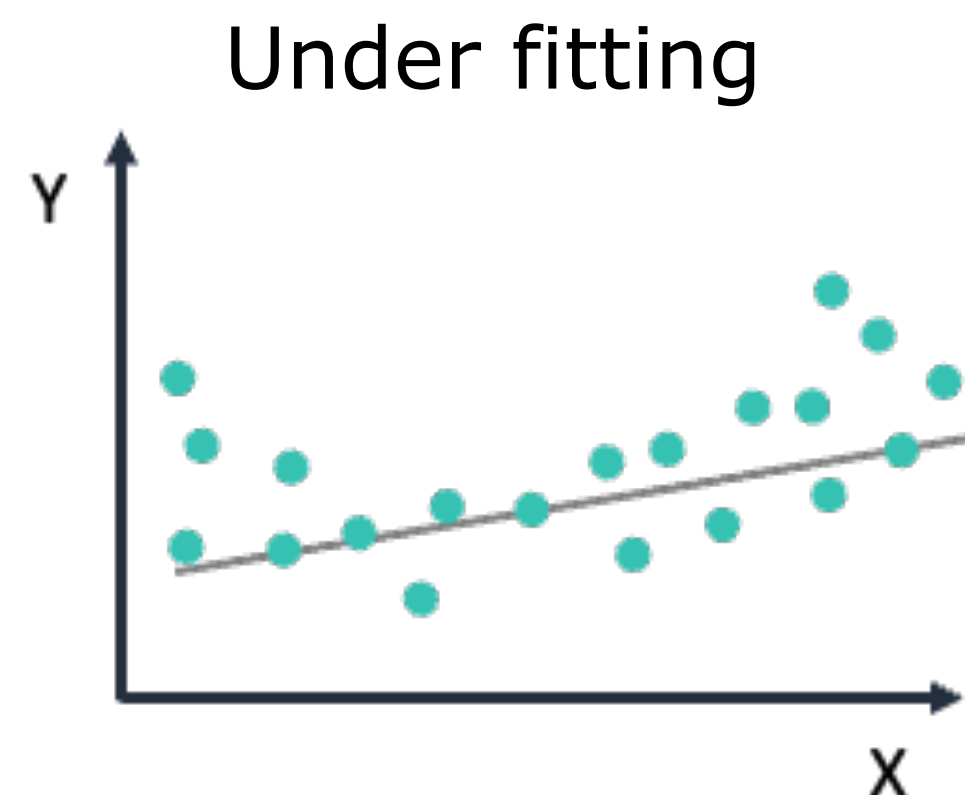
# Select and train model

## Data splitting size affects model fitness

- Data splitting size will affect model fitness

If too much training data, it memorizes and it may fail later on...

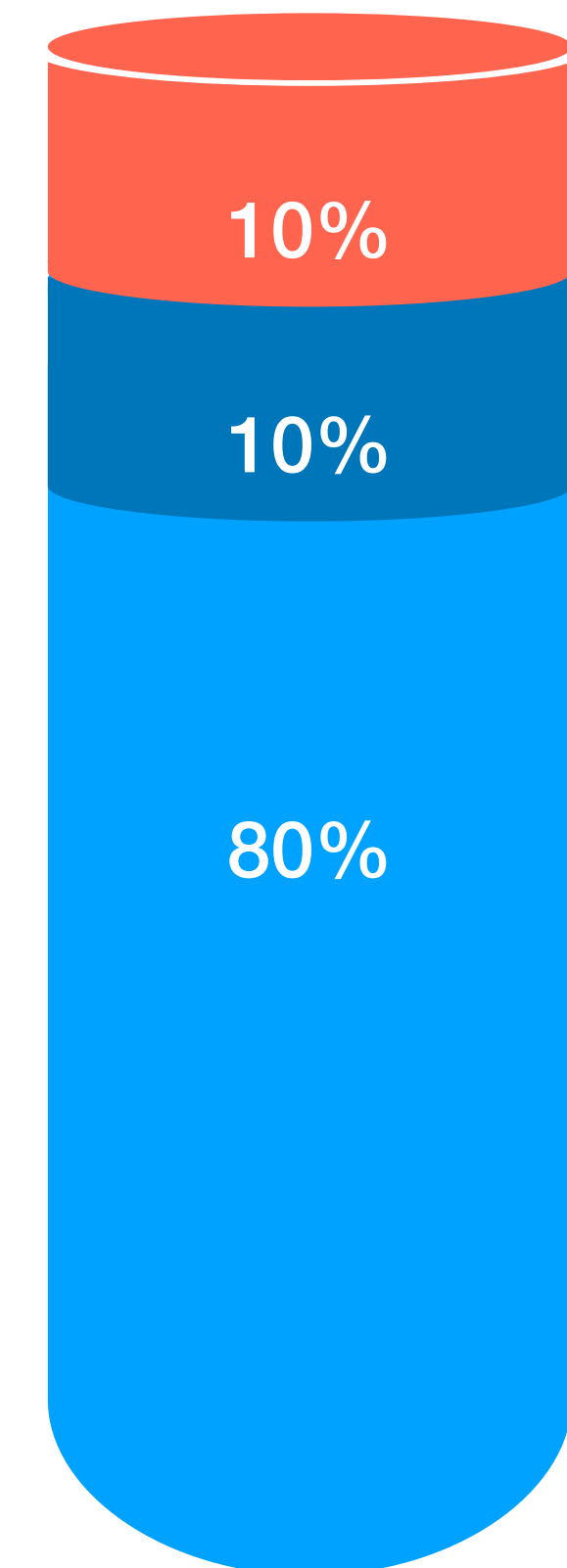
- For small data sets, we can perform k-fold cross validation.  
(Randomly partitioning data into k segments and then checking various outcomes, and with different models)
- The goal is to pick up the best solution for training data/model



# Select and train model

## Data shuffling

- If data is somehow ordered, this pattern when applied to test data biases the model. Therefore, we have to randomise the data set before splitting
- With small datasets, we can go further by using stratified sampling. It ensures both train and test parts will have similar percentage of samples of each target as in the case of the full data set



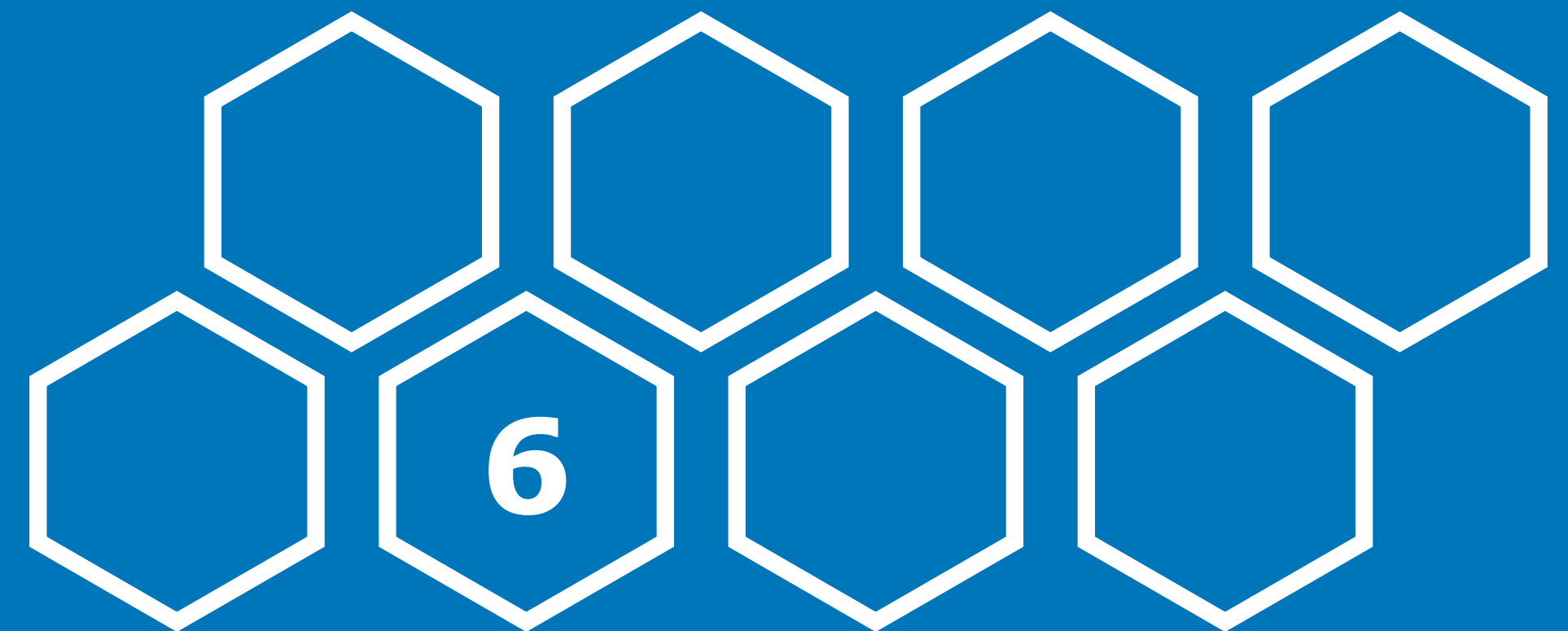
# Select and train model

## Popular algorithms

- K-means, SVM, XGBoost, ...
- We can consider many popular algorithms!

# ML Processing Pipeline

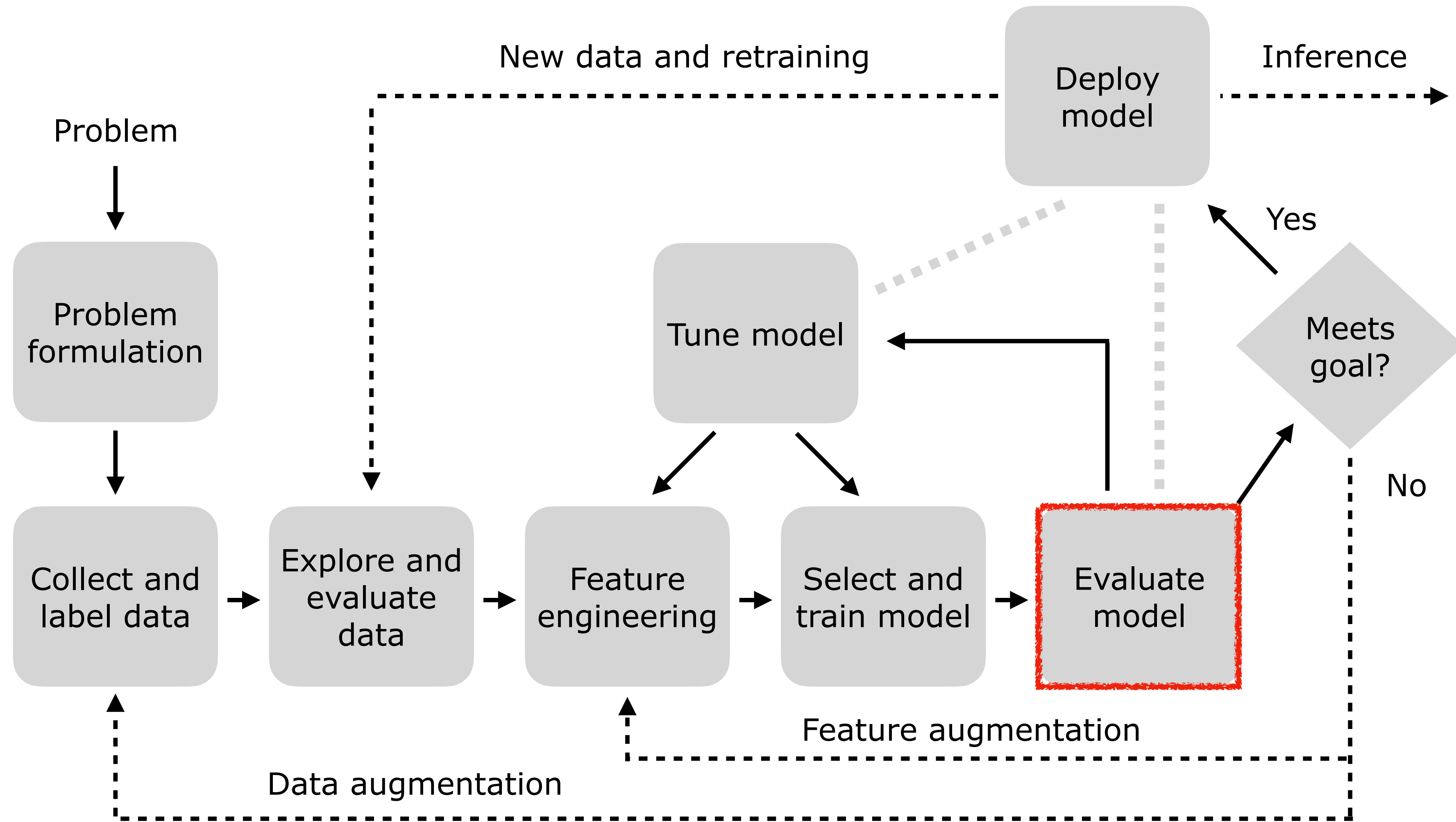
## Evaluate model





# ML pipeline

## Evaluate model



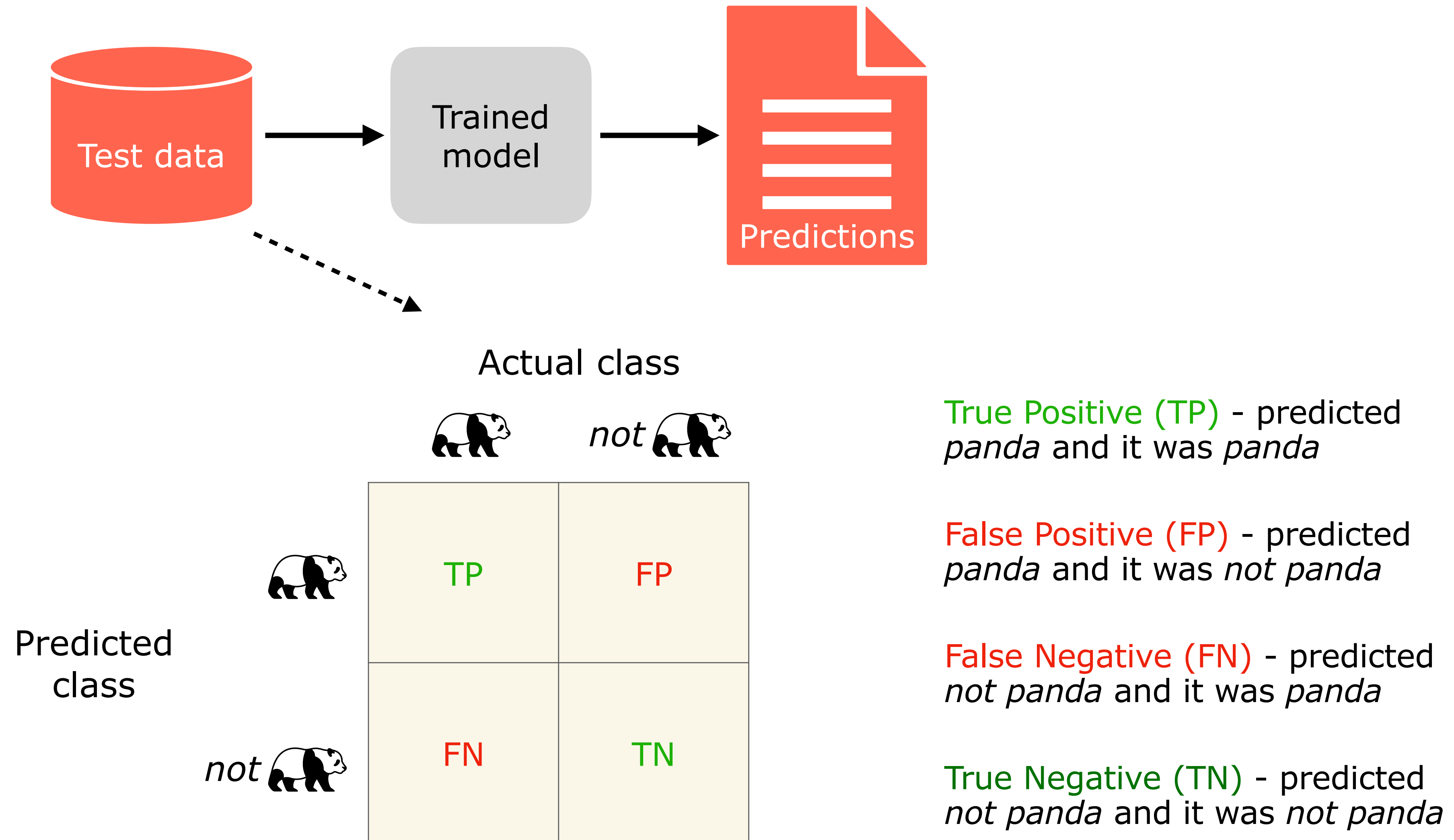
# Evaluate model

## Model accuracy

- Metrics to evaluate success in respect to the problem leads to a correlated metric for the model
- There are various metrics used in classification and regression
- But importantly, metrics used for model evaluation should be aligned with the problem goal set in advance

# Evaluate model

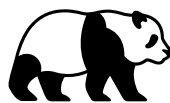
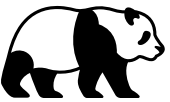
## Confusion matrix



# Evaluate model

## Confusion matrix to figure out best models

- Which is the best model is not the correct question. It depends on the purpose, e.g. identifying pandas in a website may be a totally different scenario from identifying patients with heart diseases!
- Metrics
  - **Sensitivity** (or recall, hit rate, true positive rate): What percentage of pandas were correctly identified? Good for the case of diseases mentioned above
  - **Specificity** (or selectivity, true negative rate): What percentage of not pandas were correctly identified?

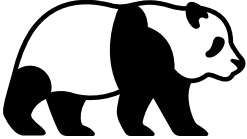
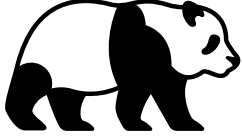
	not 
TP	FP
FN	TN

$$\text{sensitivity (recall)} = \frac{\# TP}{\# TP + \# FN}$$

$$\text{specifity (selectivity)} = \frac{\# TN}{\# TN + \# FP}$$

# Evaluate model

## Classification

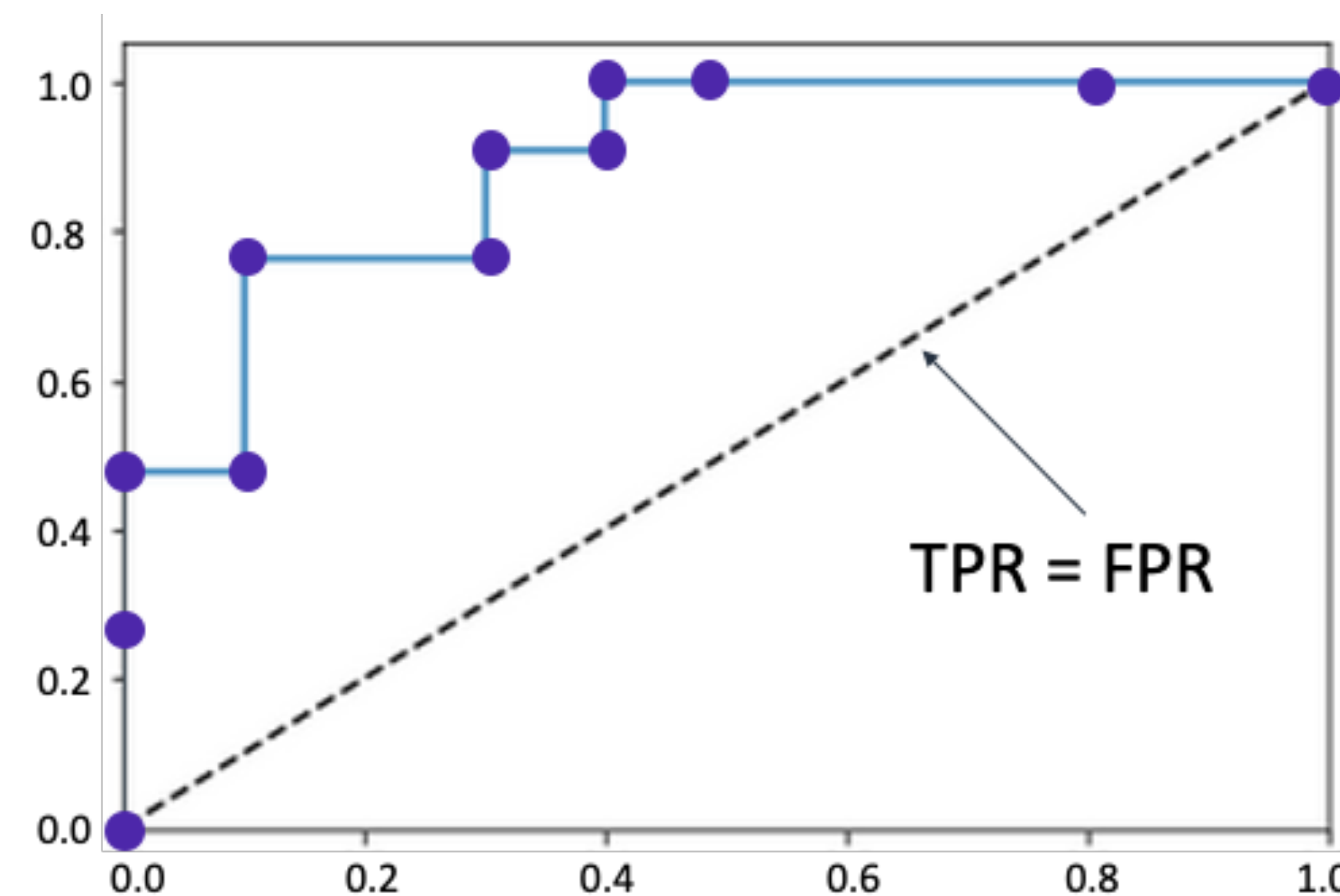
- As models return probabilities for predictions, e.g. within  $[0,1]$ , if we change the threshold to use then we may change the outcome of the classification
- Recall that, for example,
  - if predicted value  $\geq$  threshold  $\rightarrow$  
  - if predicted value  $<$  threshold  $\rightarrow$  *not* 

# Evaluate model

## Classification

- Receiver operating characteristic (ROC)
- Plot points for each threshold value
- Select threshold based on the problem case

Sensitivity  
TP rate



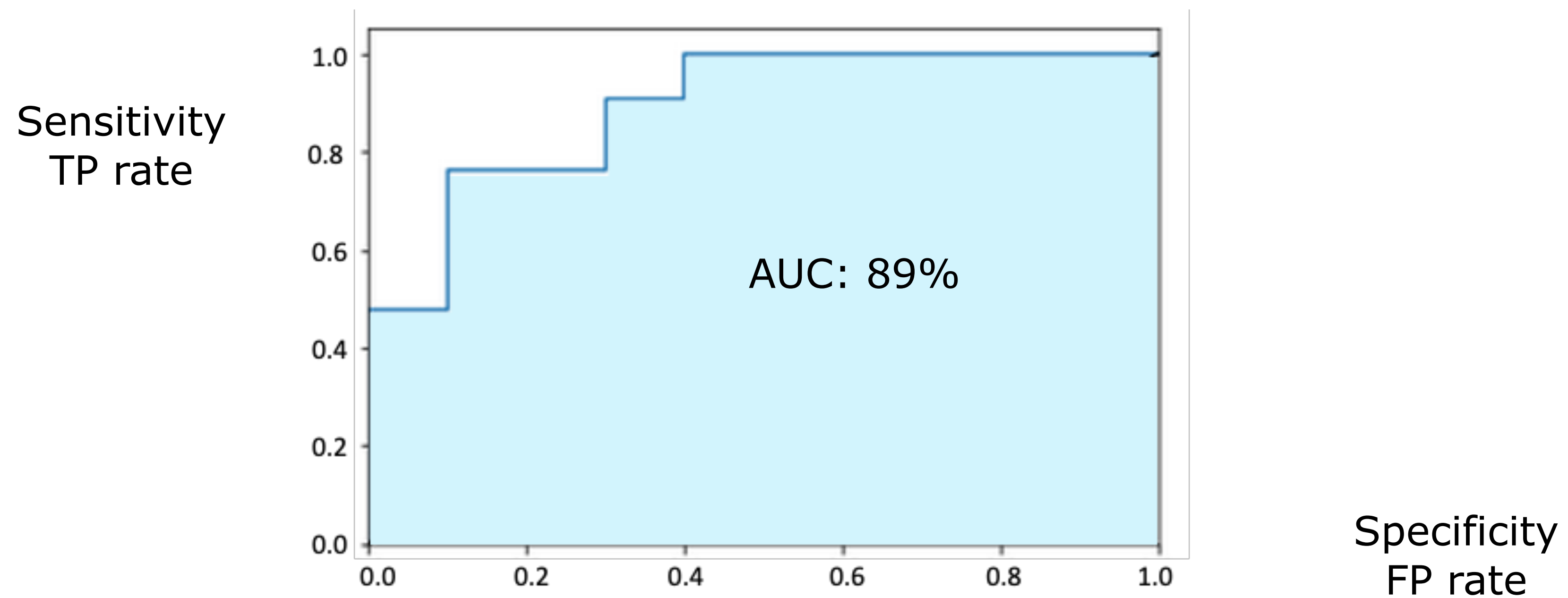
Specificity  
FP rate

# Evaluate model

## Classification

Area under the curve of receiver operating characteristic (AUC-ROC)



It easily compares one model to another



# Evaluate model

## Other metrics for classification

- **Accuracy** (or score): widely used but not so good when data has many TN cases (only gets well not panda!)
- **Precision**: Proportion of positive results that were correctly identified
  - Good for identifying email spam as it removes NP from consideration
  - Not so good for identifying seriousness as it is not considering FN. (e.g. we want to identify illness when actually there is one)
- **F1 score**: combines precision and sensitivity
  - Provides overall performance
  - Good for class imbalance but it should preserve equality between precision and sensitivity

	not 
TP	FP
FN	TN

$$accuracy = \frac{\# TP + \# TN}{\# TP + \# TN + \# FP + \# FN}$$

$$precision = \frac{\# TP}{\# TP + \# FP}$$

$$F1\ score = 2 \times \frac{precision \times sensitivity}{precision + sensitivity}$$



# Evaluate model

## Regression

Mean absolute error, mean square error, root mean square error and root-squared metrics

$$MAE = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|$$

But will not punish large errors

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

But units are squared, different from y

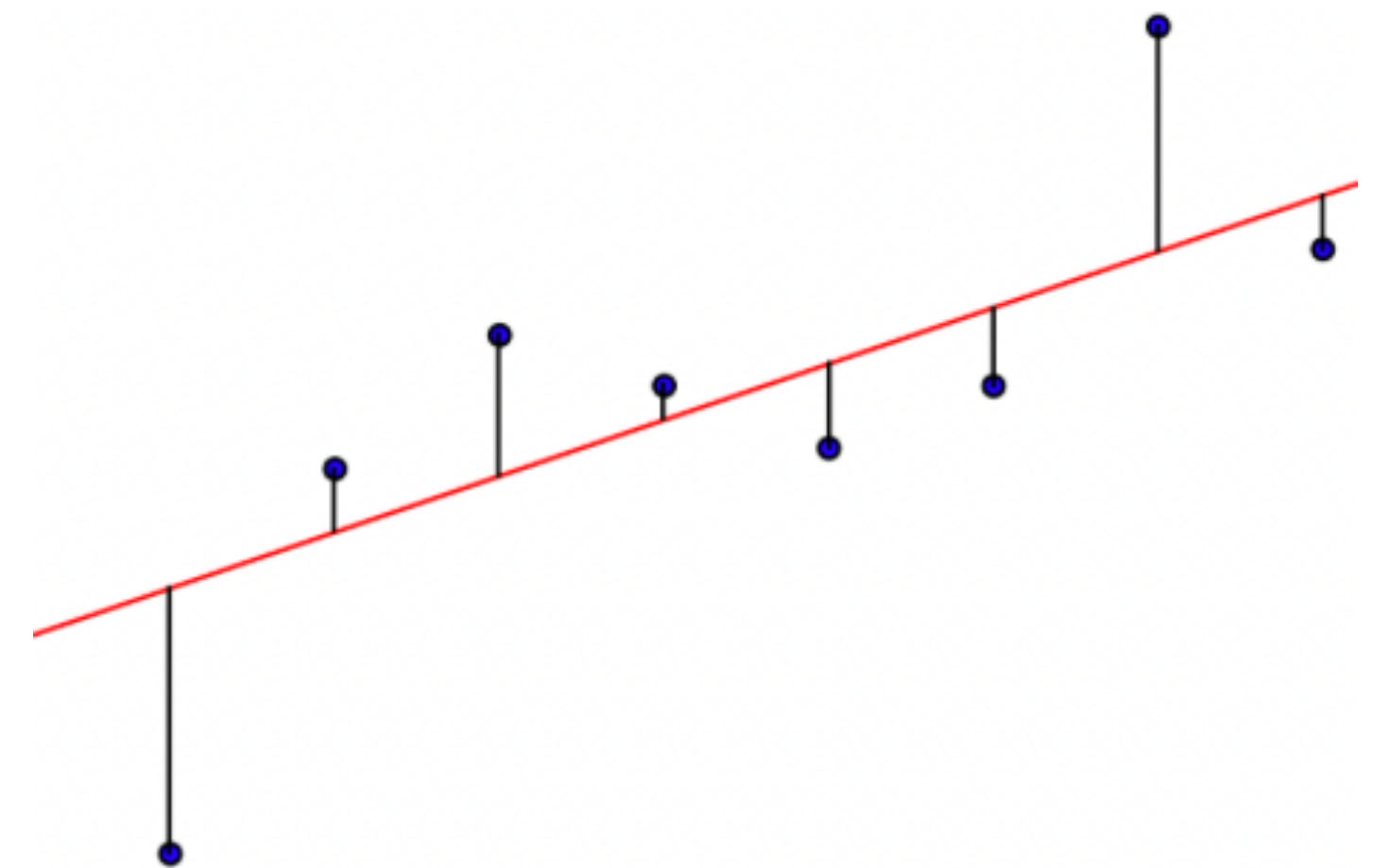
$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2}$$

Popular, units as from y

$$R^2 = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$$

Popular, value between 0 and 1; tells how much variation can be explained

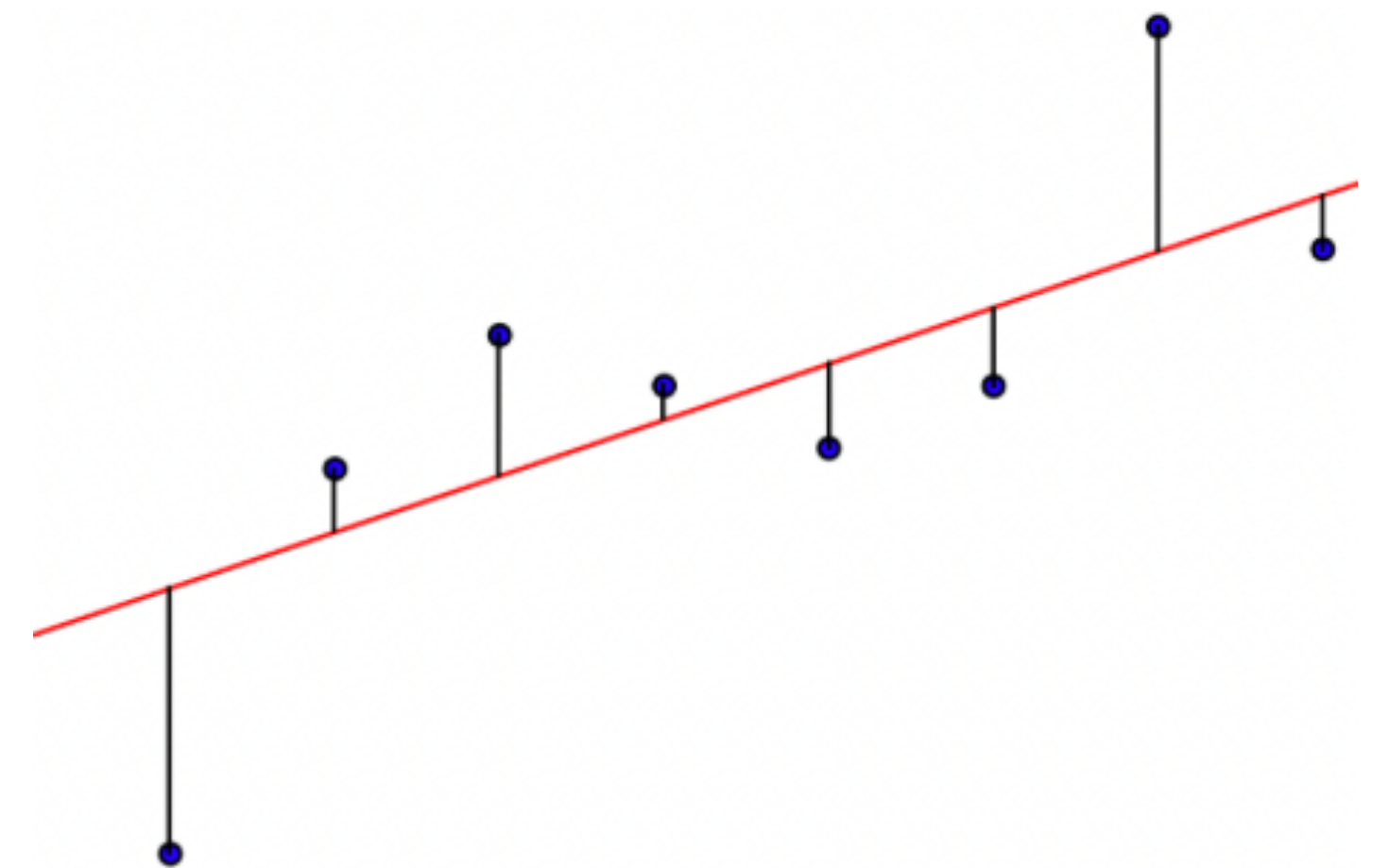
$\hat{y}_i$  is the predicted value



# Evaluate model

## Regression

- What is a good value for RMSE?
- Context is everything. For example:
  - Comparing the RMSE value with the average value of  $y$  to figure it out
  - Use of domain knowledge (suppose a small RMSE but in medicine dosage!)
- Alternatively, we can evaluate separately the residuals. Questions like “are residuals random and close to a normal distribution?” have to be put forward



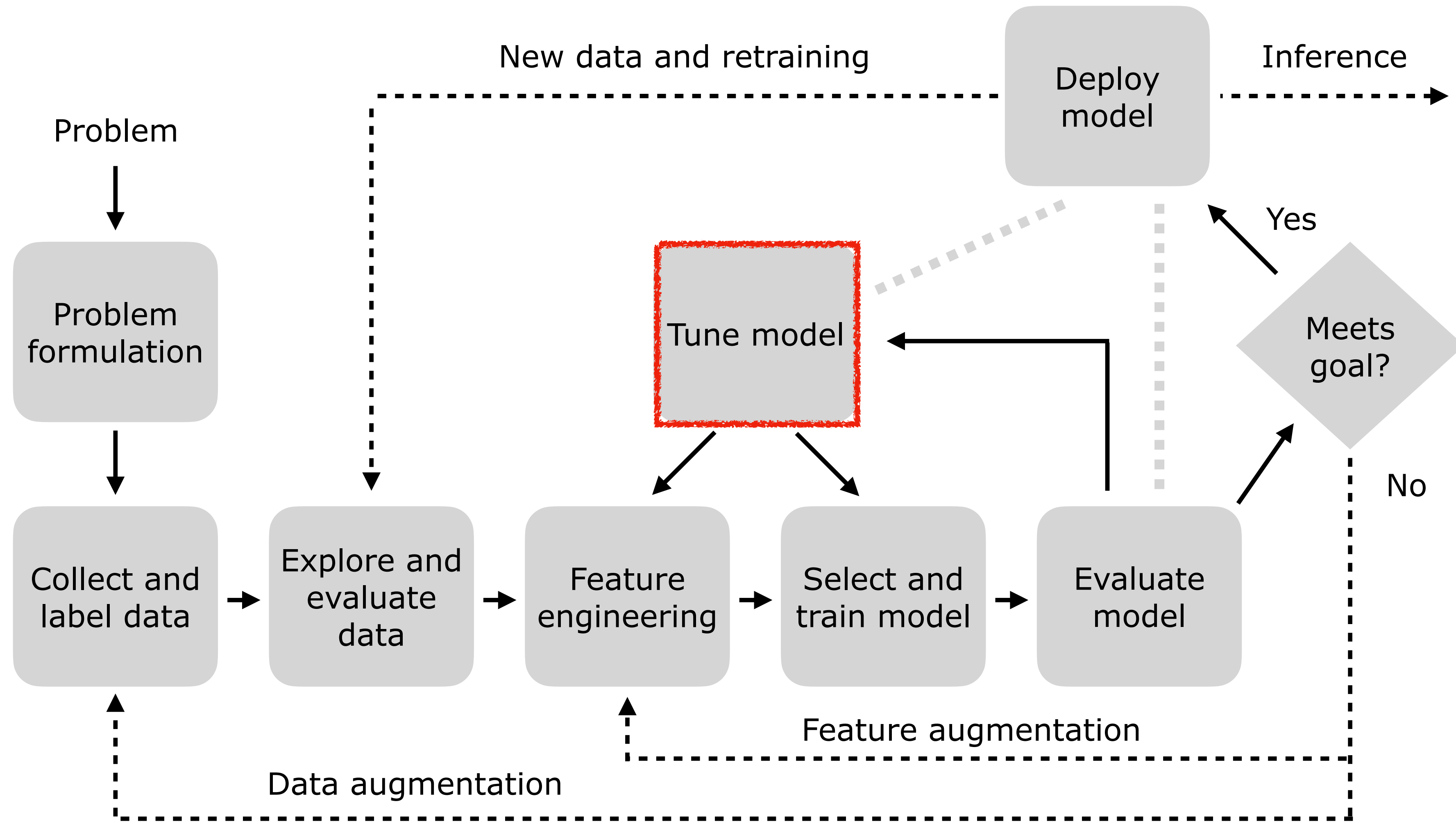
# ML Processing Pipeline

## Tune model



# ML pipeline

## Tune model



# Tune model

- Use of metrics obtained during testing to help tuning the model to find the best solution

But we cannot guarantee that is going to happen

- Examples of changes prior to training again
  - Selecting different set of features
  - Use of different data
  - Tune parameters of the model

# Tune model

## Model, optimiser and data parametrisation

- Model parametrisation

Helps to define the model, mostly related to the algorithms themselves (SVM, neural-networks, etc.)

- Optimiser parametrisation

Relates to how the model learns patterns on data (e.g. via stochastic gradient descent)

- Data parametrisation

Relates to attributes of data, which can be useful for small or homogeneous datasets

- The tuning process can involve

- Manual approach, following intuition and experience

- In general, it is not the most effective approach

- Better to attempt tuning just a few parameters, carefully, and within a limited range of values

- Automatic approach, looking for best tuning values given the target metric, e.g. AUC and ranges of values

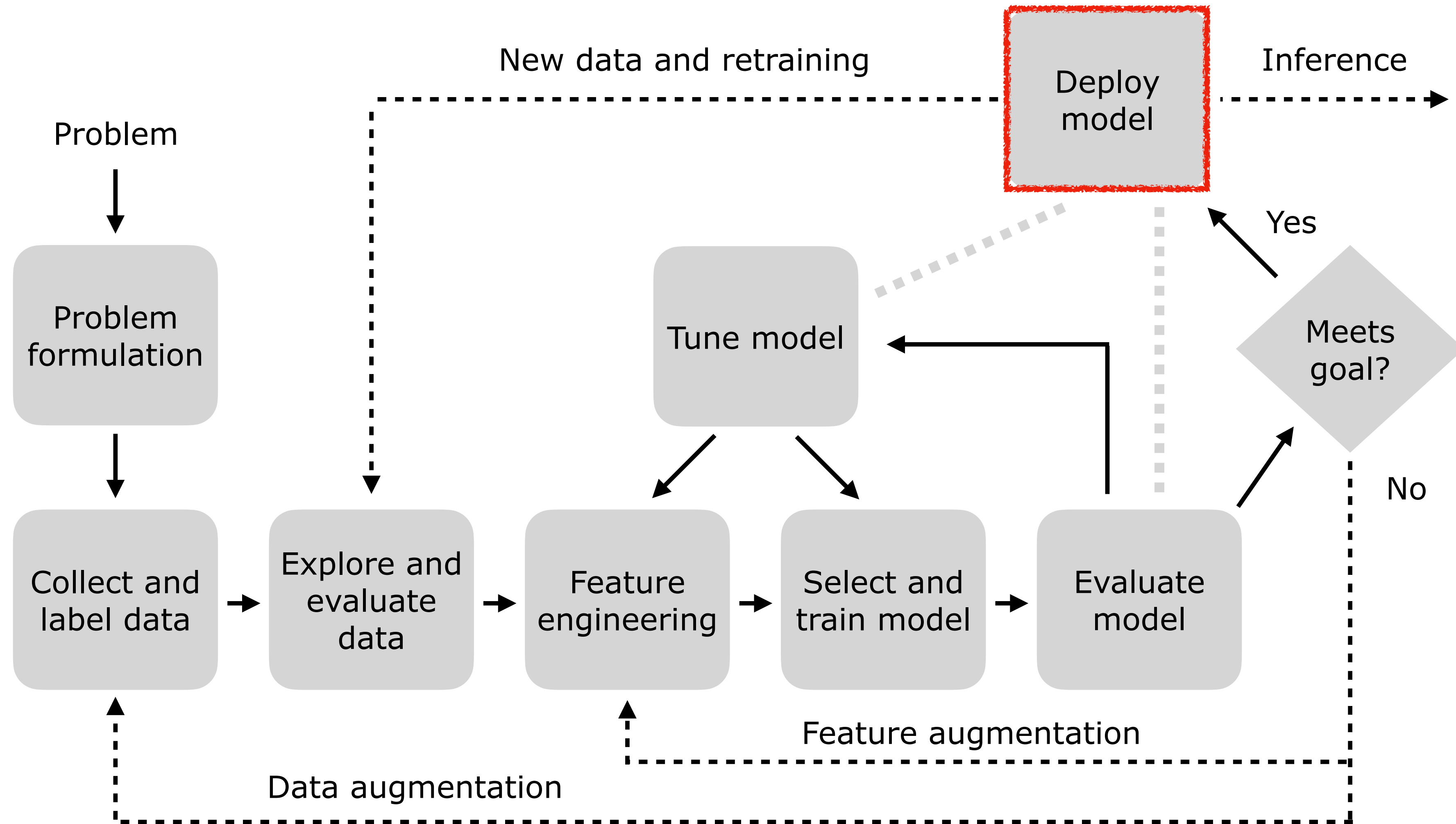
# ML Processing Pipeline

## Deploy model



# ML pipeline

## Deploy model





# Deploy model

## Goals

- Model ready to deploy after trained, tuned, and tested
- Testing and production environments
  - As a computational application, testing and obtaining performance metrics usually implies deployment. So order in the pipeline is not straightforward
  - Mechanics in both environments are the same
- Managed production environment
  - Hosting models that deliver inference both securely and with low latency
  - Production data has to be monitored and accumulated over time
  - It is a continuous process

## Deploy model

### Inference pipeline in production

- To achieve consistency, predictions on new data must rely on the same processing steps as in training. Otherwise we might get wrong predictions
- Inference pipeline reuses the same code that is applied in training
  - Helps with accuracy of each inference request and reduces development costs