DataSci 400

lesson 8: supervised learning

Seth Mottaghinejad

today's agenda

- **supervised learning**
- a simplified example
- everyone has their **jargon**
- supervised learning **algorithms**
- **historical data** vs **future data**
- a good model should **generalize** well
- splitting data into **training** and **test data**
- **underfitting** vs. **overfitting**

supervised learning

- also called **predictive modeling** or **inference**
- look at some current examples (**labeled data**) and find a model that can predict future examples (**unlabeled data**)
- the **target variable** or **label** is we want to predict
  - **regression** algorithms are used with a **numeric target**
  - **classification** algorithms are used with a **categorical target**
- by comparing predictions with the actual labels, we can evaluate our model's accuracy (hence the term *supervised*)

supervised learning simplified example

let's look at credit rating use case

- **data we have**: age, income, credit score (300-800)
- **algorithm choice**: linear regression
  `a + b * age + c * income + some error = credit score`
- during **training**, the algorithm **learns** `a`, `b`, and `c`
  `5.8 + 3.9 * age + 1.98 * income + error = credit score`
- during **scoring**, we apply it to get predictions
  `5.8 + 3.9 * age + 1.98 * income = predicted credit score`

in pseudo-code

- in Python we call `.fit()` to train and `.predict()` to score
- choose the algorithm:

```
lin_reg = LinearRegression(alpha = .005)
```

- train the algorithm on data

```
lin_reg.fit(X_train, y_train)
```

- predict on any new data

```
lin_reg.predict(X_test)
```

everyone has their jargon

- **rows** - observation, example, sample, record, data points, item, instance
- **columns** - variables, attributes, properties, features, fields, dimensions
  - **target** - label, response variables, dependent variable, outcome
  - **features** - explanatory variable, independent variable, predictors, covariates
    - **numeric features**: dates, counts, amounts, etc.
    - **categorical features**: grouping variables, identifiers

supervised learning algorithms

- **linear regression**: regression
- **logistic regression**: classification (even though it's called logistic *regression*)
- **tree-based algorithms**: more commonly for classification
- **support vector machines** (SVMs): binary classification
- **neural networks**: classification and regression
  - **deep neural networks** (deep learning)
    - image recognition and natural language processing (NLP)

break time

historical data vs future data

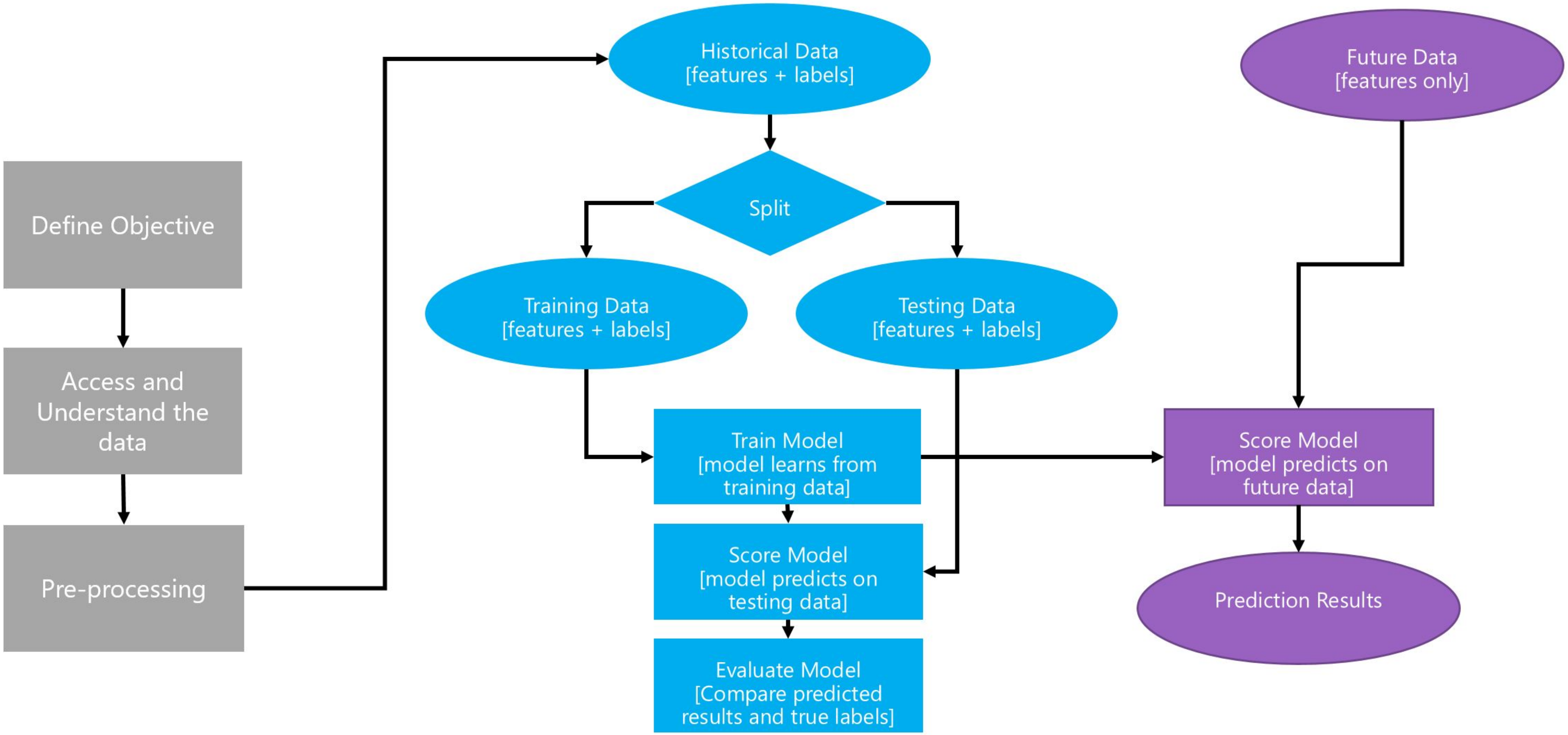
- data we use for modeling is a snapshot, e.g. the last two years
  - we refer to this as **historical data**, and it is labeled
- but we keep collecting data after we train the model
  - we refer to that as **future data**, and it may or may not be labled
- use historical data to create a model
- deploy the model to get predictions on future data, called **scoring**
- but if future data is unlabeled, how do we know if predictions are any good?

a good model should generalize well

- I repeat: if future data is unlabeled, how do we know if predictions are any good?
- the premise of the question is this: **a model's performance should be measured on data that it hasn't seen during training**
- we say a good model should **generalize** or **extrapolate** to out-of-sample data (data not used for training)
- at training time we try to find parameters that minimize error **on the training data**, but there's no guarantee that this will also minimize error on out-of-sample data

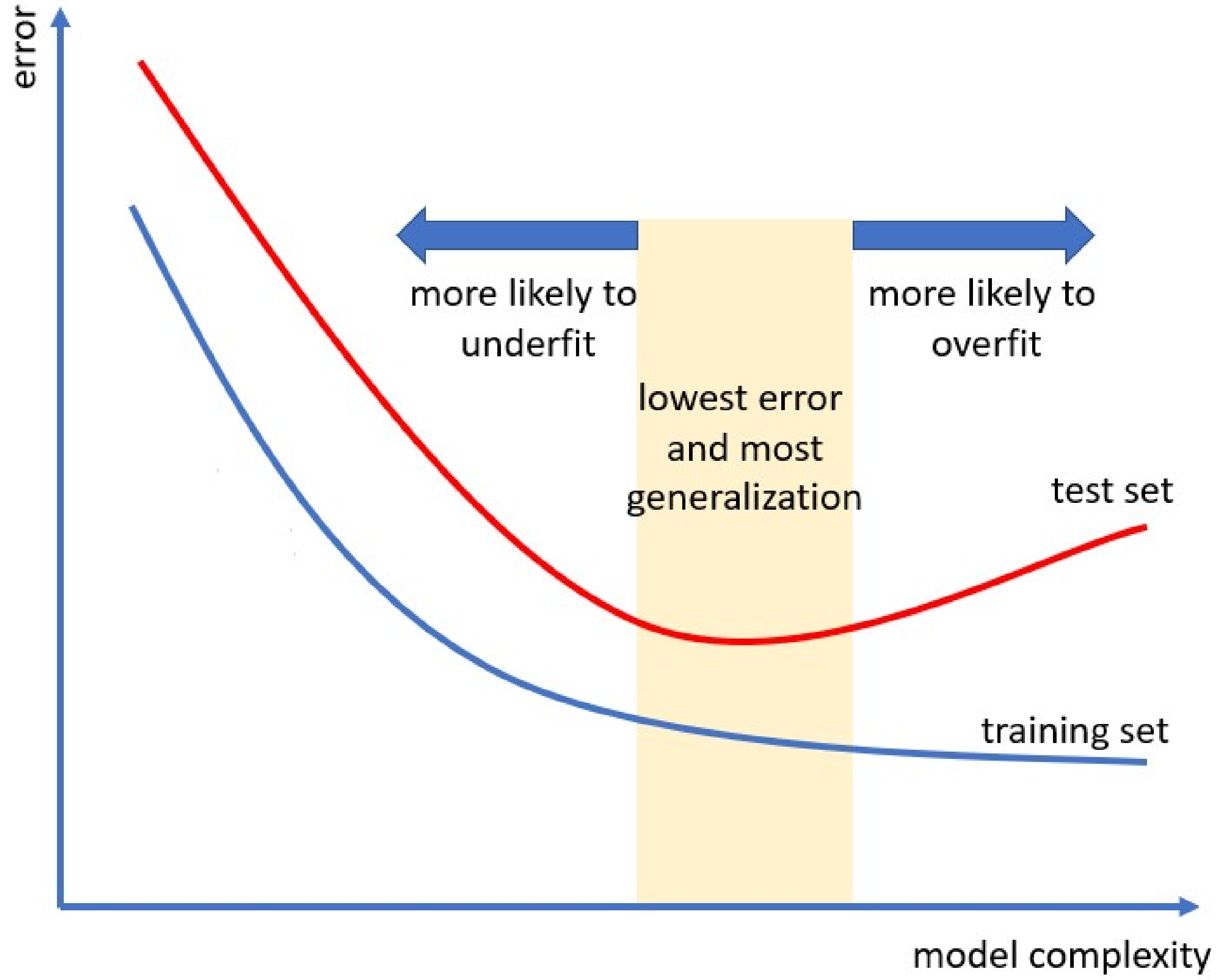
training and test data

- set aside a small **random** portion of historical data and **pretend it's future data**, we call this **test data**
- the remaining portion is called **training data**
- unlike future data, test data is **labeled**, so we can compare predictions with observed, also called **ground truth**
- so we use the training data to train a model
- then we **evaluate** the trained model's performance on the test data (data that the model didn't see at training time)


Define Objective
Access and Understand the data
Pre-processing
Historical Data [features + labels]
Split
Training Data [features + labels]
Testing Data [features + labels]
Train Model [model learns from training data]
Score Model [model predicts on testing data]
Evaluate Model [Compare predicted results and true labels]
Future Data [features only]
Score Model [model predicts on future data]
Prediction Results

underfitting vs. overfitting

- *fitting* means learning: when we call the `.fit()` method
- if a model performs poorly on the training data, then it almost certainly will perform poorly on the test data as well: we say the model is **underfitting** (not learning enough)
- if a model performs well on the training data, but poorly on the test data: we say the model is **overfitting** (it's learning the signal but also "learning" the noise in the training data, and hence fails to generalize)
- a good model is one that neither underfits nor overfits


error
more likely to
underfit
more likely to
overfit
lowest error
and most
generalization
test set
training set
model complexity

the end