Machine Learning for Industry

Lecture 1 - Introduction to ML and Regression

Mattias Villani

Department of Computer and Information Science
Linköping University

Department of Statistics
Stockholm University

AI COMPETENCE FOR SWEDEN

LINKÖPING UNIVERSITY

Course overview

- Topic 1 - Basic ML and Regularized regression
- Topic 2 - Classification and Unsupervised learning
- Topic 3 - Neural networks and Deep learning
- Topic 4 - Reinforcement learning

Topic 1 - Basic ML and regularized regression

- **Lecture block 1 - Basic ML and Regularized regression**
  - Intro to Machine Learning
  - Linear and Nonlinear Regression
  - Bias-Variance trade-off
  - Regularization
- **Lecture block 2 - Regression trees and extensions**
  - Regression trees
  - Random forest
  - Boosted trees

Topic 2 - Classification and unsupervised learning

- **Lecture block 3 - Classification**
  - Likelihood and Bayesian learning
  - Discriminative classification models
  - Generative classification models
- **Lecture block 4 - Unsupervised learning**
  - Clustering methods
  - Mixture models
  - Visualizing data using t-SNE

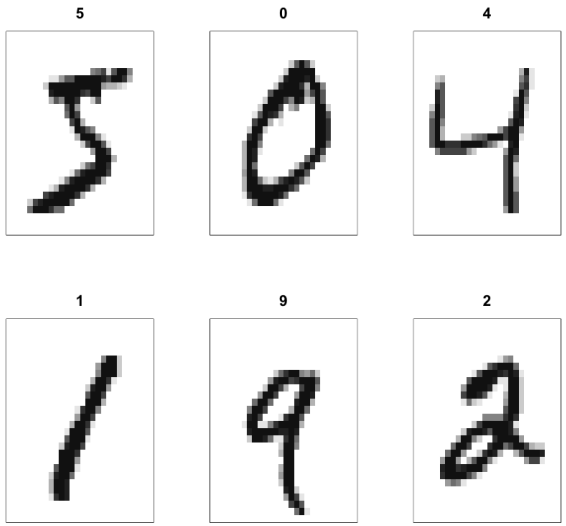
What is machine learning?

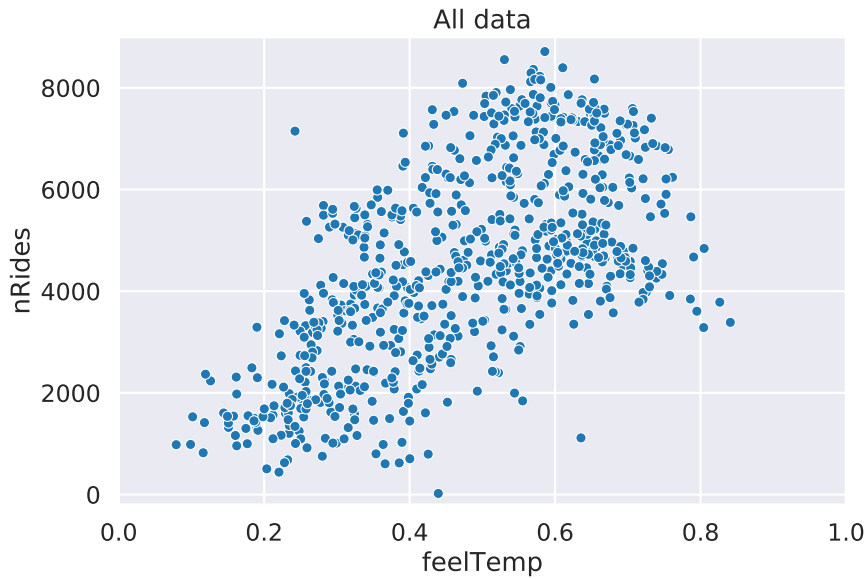
- **Machine learning** is the scientific study of algorithms and statistical models that computer systems use to perform a specific task without using explicit instructions, relying on patterns and inference instead.
- It is seen as a subset of **artificial intelligence**. Machine learning algorithms build a mathematical model based on sample data to make predictions or decisions
- Machine learning is closely related to **computational statistics**.
- **Data mining** is a field of study within machine learning, and focuses on exploratory data analysis through unsupervised learning.
- In its application across business problems, machine learning is also referred to as **predictive analytics**.
- **Data science**?

Wikipedia (Nov 2, 2019).

Labeling distinctions

- Availability of labels
  - **Supervised learning**: labeled training data.
  - **Unsupervised learning**: unlabeled training data.
  - **Semi-supervised learning**: labels for a subset of training data
  - **Reinforcement learning**: sequential learning with incremental rewards for good behavior.
- Types of labels
  - **Regression** - labels are real numbers
  - **Classification** - labels are discrete





Data distinctions

- **Regression** data
  - Real valued
  - Counts
  - Proportions
- **Categorical** data (binary, multi-class)
- **Time series** (sensor data over time)
- **Spatial data** (sensors at different locations, images)
- **Survival data** (lifetime of hard drives)
- **Longitudinal data** (one short time series for many objects)
- Specialized structure
  - **Images**
  - **Text**
  - **Sound**
  - ...

Modeling distinctions

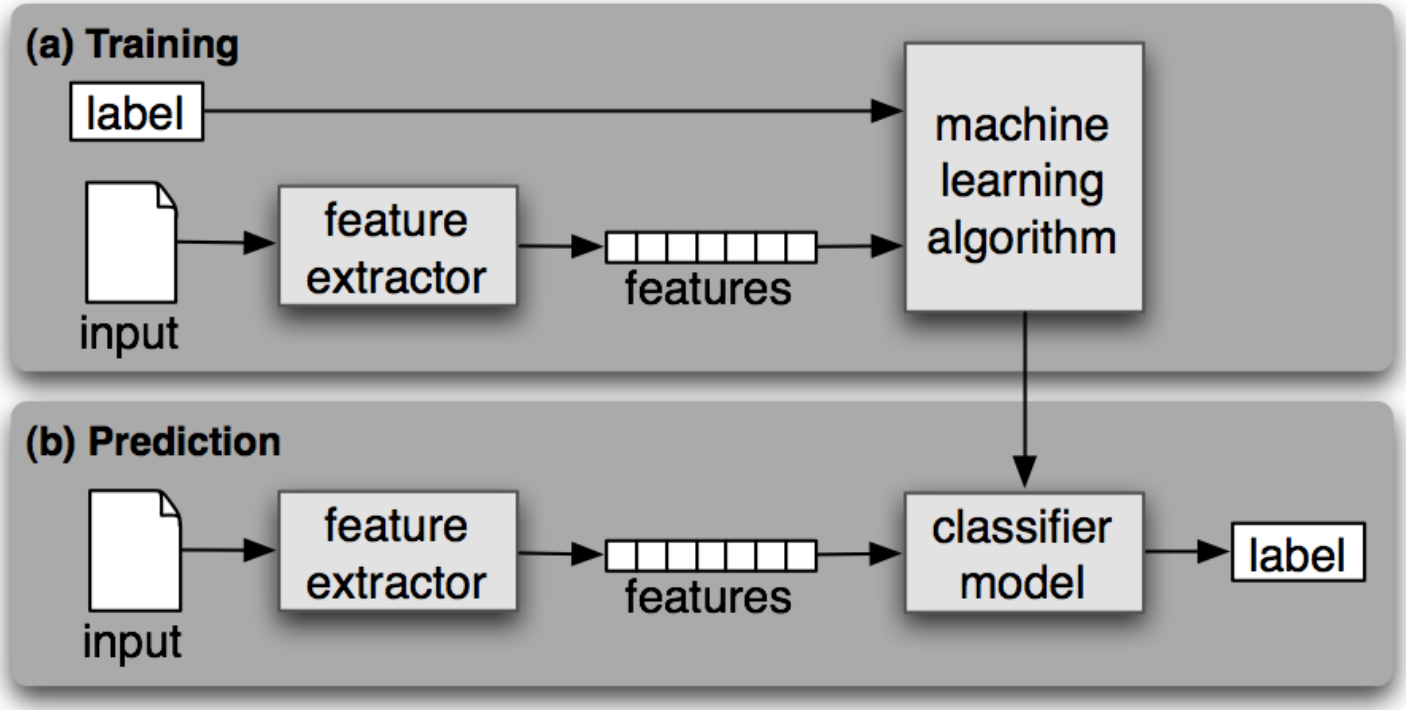
- Approach
  - **Probabilistic**
  - **Algorithmic**
- Type of probability model
  - **Generative**
  - **Discriminative**

Why probability models?

- Probability models and statistical inference is a **framework**.
- Principled **way to think** about any problem in ML.
- Can better **evaluated** and critiqued.
- **Quantify uncertainties**. Crucial for **Decision making**.

> *As robotics is now moving into the open world, the issue of **uncertainty** has become a major stumbling block for the design of capable robot systems. Managing uncertainty is possibly the most important step towards robust real-world robot systems.*
>
> *from the book Probabilistic Robotics by Thrun et al.*

The machine learning work flow



Regression

- **Continuous response** data: y.
- Explanatory variables, **features**, covariates: $x_1, ..., x_p$
- **Linear regression** with Gaussian (normal) errors

$$y = w_0 + w_1 x_1 + ... + w_p x_p + \varepsilon, \ \varepsilon \sim N(0, \sigma^2).$$

- For a sample of n data points

$$\underset{(n\times 1)}{\boldsymbol{y}} = \underset{(n\times p)(p\times 1)}{\boldsymbol{X}\ \boldsymbol{w}} + \underset{(n\times 1)}{\boldsymbol{\varepsilon}},$$

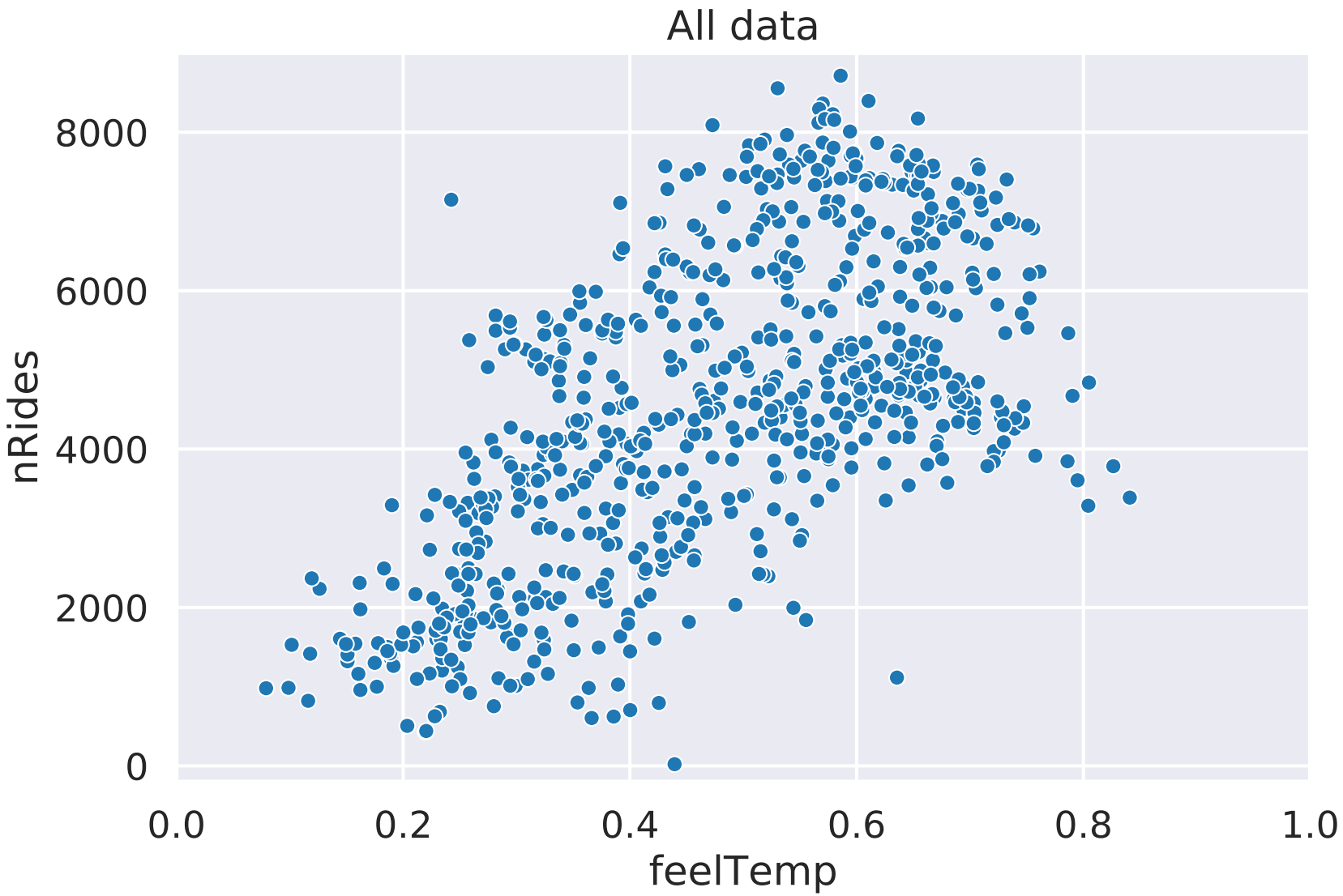
  where $\boldsymbol{y} = (y_1, \ldots, y_n)^T$, $\boldsymbol{w} = (w_0, w_1, \ldots, w_p)^T$ etc.
- The **Least Squares** (**LS**) estimator

$$\hat{\boldsymbol{w}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$$

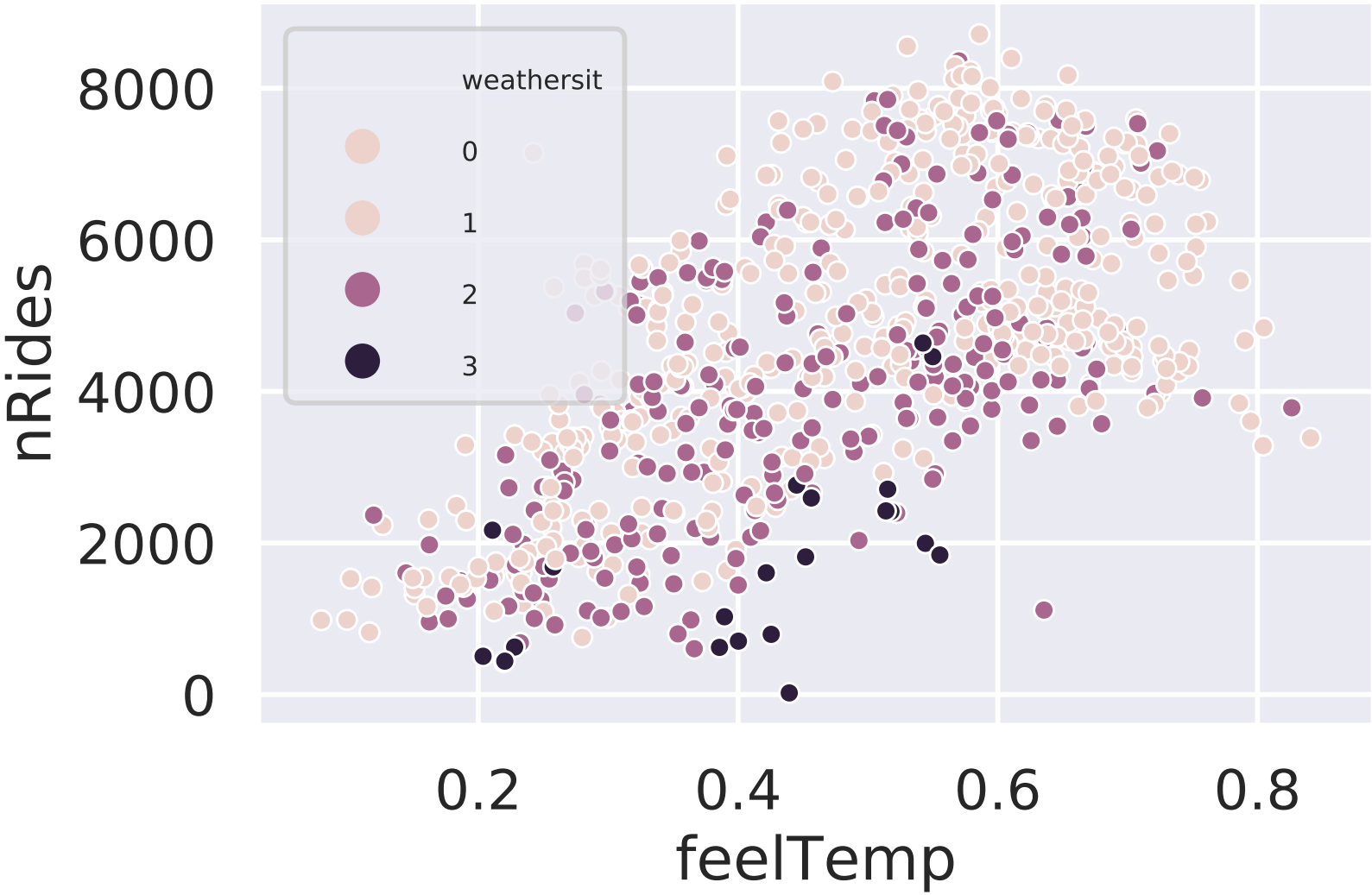
  minimizes the sum of squared errors (**SSE**)

$$\sum_{i=1}^{n}(y_i - \boldsymbol{x}_i^T\boldsymbol{w})^2 = (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w})^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}).$$

Bike share data

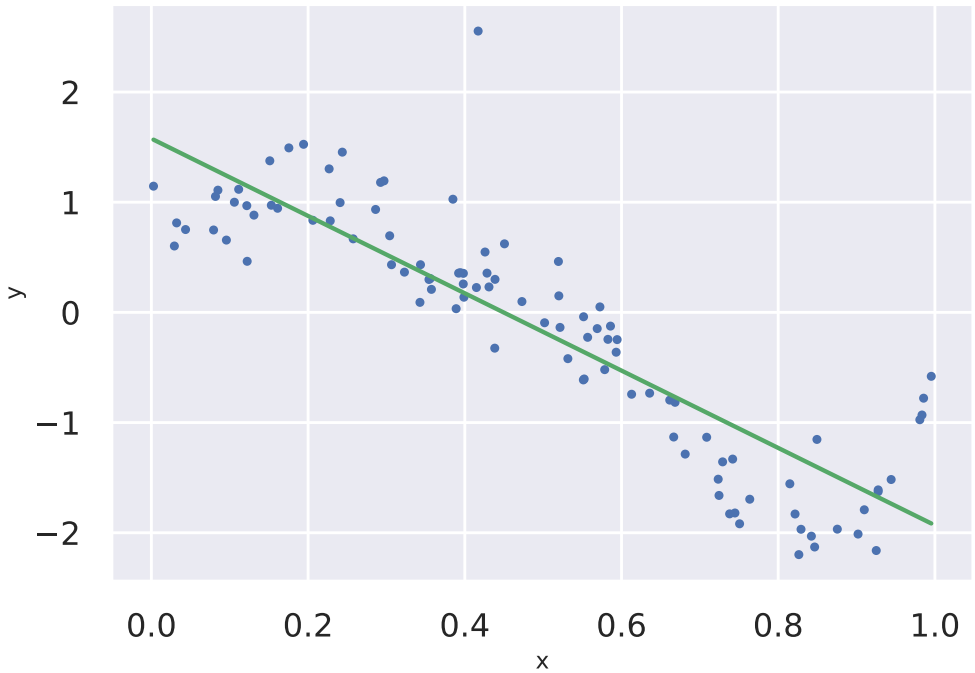


Bike share data



Linear regression in scikit-learn

```
from sklearn import linear_model
regModel = linear_model.LinearRegression()
regModel.fit(X = X, y = y)
print(regModel.coef_)
```



Prediction in Regression

- **Prediction** for a new set of features: $\boldsymbol{x}^\star = (x_1^\star, ..., x_p^\star)^T$

$$\hat{y}_i = \hat{w}_0 + \hat{w}_1 x_1^\star + ... + \hat{w}_p x_p^\star.$$

- Also possible to derive 95% **prediction intervals**

$$\hat{y}_i \pm 1.96 \cdot s_{\hat{y}_i},$$

where the formula for **prediction standard error**, $s_{\hat{y}_i}$, is given in standard statistics textbooks.

- In more complex models: use simulation to get intervals (see also Bayesian learning later in the course).
- Prediction method for `sklearn` model objects

```
regModel.predict(Xnew)
```

where `Xnew` is a $n_{\text{pred}} \times p$ matrix with n_{pred} new examples.

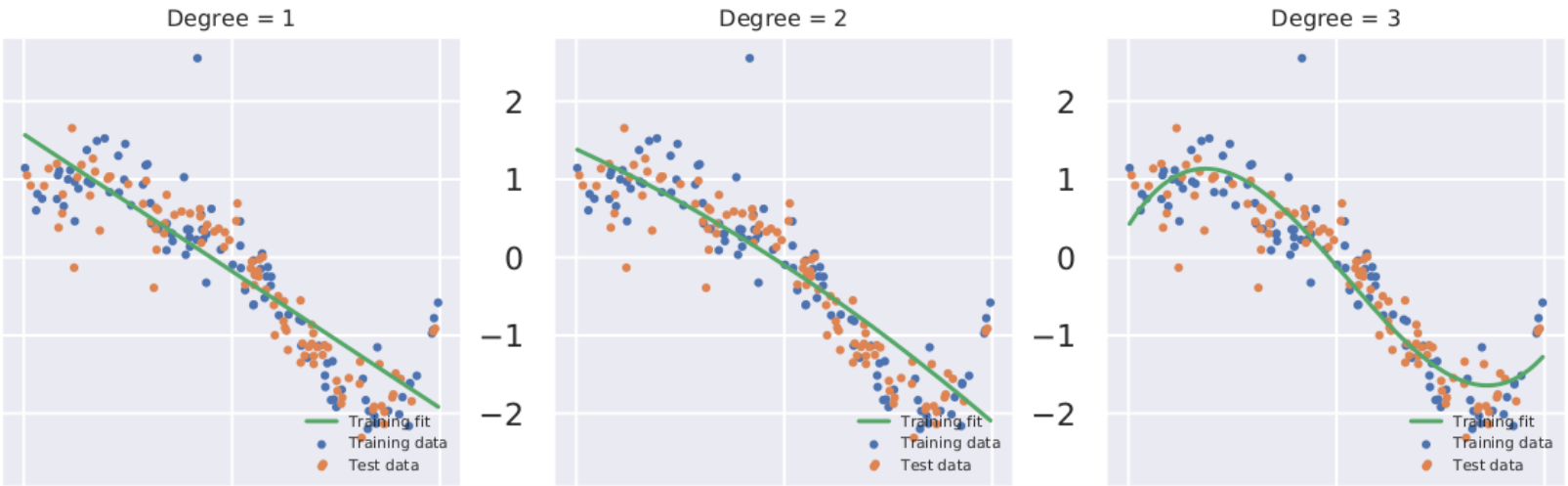
Polynomial regression in scikit-learn

- **Polynomial regression**

$$y = w_0 + w_1 x + w_2 x^2 + ... + w_p x^p + \varepsilon,\ \varepsilon \sim N(0, \sigma^2).$$

- **Linear** in the weights. LS applies. Features: $(1, x, x^2, ..., x^p)$.

```
from sklearn import linear_model
from sklearn.preprocessing import PolynomialFeatures
regModel = linear_model.LinearRegression()
poly = PolynomialFeatures(degree=3, include_bias=False)
XBasis = poly.fit_transform(X)
regModel.fit(X = XBasis, y = y)
```



Feature engineering

- Not all data comes as a nice $n \times p$ matrix.
- **Image**, **text**, **sound**.
- **Categorical variables**. 'man'/'woman'. seasons.
- `sklearn.preprocessing.OneHotEncoder()`
- **Basis functions**.
  - Polynomial $x^2, x^3, \ldots$.
  - Splines (local polynomials)
  - Interactions, $x_1 x_2$.
- **Transformations**.
  - $\log(y)$ for positive data
  - $\text{logit}(y) = \log(y/(1-y))$ för proportions.
- **Missing data**.
- See PDSH - Feature construction.

Bike share data - categorical variables

	dteday	weekday	workingday	weathersit	feelTemp	hum	windspeed	nRides
0	2011-01-01	6	0	2	0.363625	0.805833	0.160446	985
1	2011-01-02	0	0	2	0.353739	0.696087	0.248539	801
2	2011-01-03	1	1	1	0.189405	0.437273	0.248309	1349
3	2011-01-04	2	1	1	0.212122	0.590435	0.160296	1562
4	2011-01-05	3	1	1	0.229270	0.436957	0.186900	1600
5	2011-01-06	4	1	1	0.233209	0.518261	0.089565	1606
6	2011-01-07	5	1	2	0.208839	0.498696	0.168726	1510
7	2011-01-08	6	0	2	0.162254	0.535833	0.266804	959
8	2011-01-09	0	0	1	0.116175	0.434167	0.361950	822
9	2011-01-10	1	1	1	0.150888	0.482917	0.223267	1321

Bike share data - one-hot

	weekday_1	weekday_2	weekday_3	weekday_4	weekday_5	weekday_6	weathersit_2	weathersit_3	feelTemp	hum	windspeed	nRides
0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	0.363625	0.805833	0.160446	985
1	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.353739	0.696087	0.248539	801
2	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.189405	0.437273	0.248309	1349
3	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.212122	0.590435	0.160296	1562
4	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.229270	0.436957	0.186900	1600
5	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.233209	0.518261	0.089565	1606
6	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.208839	0.498696	0.168726	1510
7	0.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	0.162254	0.535833	0.266804	959
8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.116175	0.434167	0.361950	822
9	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.150888	0.482917	0.223267	1321

Generalization and Training-Test splits

- **Generalization**: how well will the model predict unseen data?
- Split the data examples $\mathcal{S} = \{1, \ldots, n\}$ into two parts:

$$\mathcal{S} = \mathcal{S}_{\text{training}} \cup \mathcal{S}_{\text{test}}$$

  - **Training data** to estimate the model parameters (e.g. $\boldsymbol{w}$)
  - **Test data** to estimate the generalization performance.

- **Evaluation metrics**:
  - Regression: $\text{RMSE}_{\text{test}} = \sqrt{n_{\text{training}}^{-1} \sum_{i \in \mathcal{S}_{\text{test}}} (y_i - \hat{y}_i)^2}$
  - Classification: Percentage of misclassified examples.

- How to make the split:
  - Randomly
  - Systematic (time series: most recent data in test set).

```
from sklearn.model_selection import train_test_split
xTrain, xTest, yTrain, yTest = train_test_split(x, y, test_size = 0.5)
```

Bias and Variance

- True relationship (ex. sin(x))

$$y = f(\boldsymbol{x}) + \varepsilon.$$

- Estimated model $\hat{f}(\boldsymbol{x})$ (ex. linear regression $\hat{f}(\boldsymbol{x}) = \boldsymbol{x}^T \hat{\boldsymbol{w}}$).
- **Bias**

$$\text{Bias}\hat{f}(\boldsymbol{x}) = \mathbb{E}\left[\hat{f}(\boldsymbol{x})\right] - f(\boldsymbol{x})$$

- Bias = how correct **on average**, over all possible datasets?
- **Variance**

$$\mathbb{V}\left[\hat{f}(\boldsymbol{x})\right] = \mathbb{E}\left[\left(\hat{f}(\boldsymbol{x}) - \mathbb{E}\left[\hat{f}(\boldsymbol{x})\right]\right)^2\right].$$

- **Mean Squared Error (MSE)**

$$\text{MSE} = \mathbb{E}\left[\left(\hat{f}(\boldsymbol{x}) - f(\boldsymbol{x})\right)^2\right] = \mathbb{V}\left[\hat{f}(\boldsymbol{x})\right] + \left(\text{Bias}\hat{f}(\boldsymbol{x})\right)^2$$

- MSE = expected squared deviation from $f(\boldsymbol{x})$, in a given dataset.

Bias-Variance trade-off

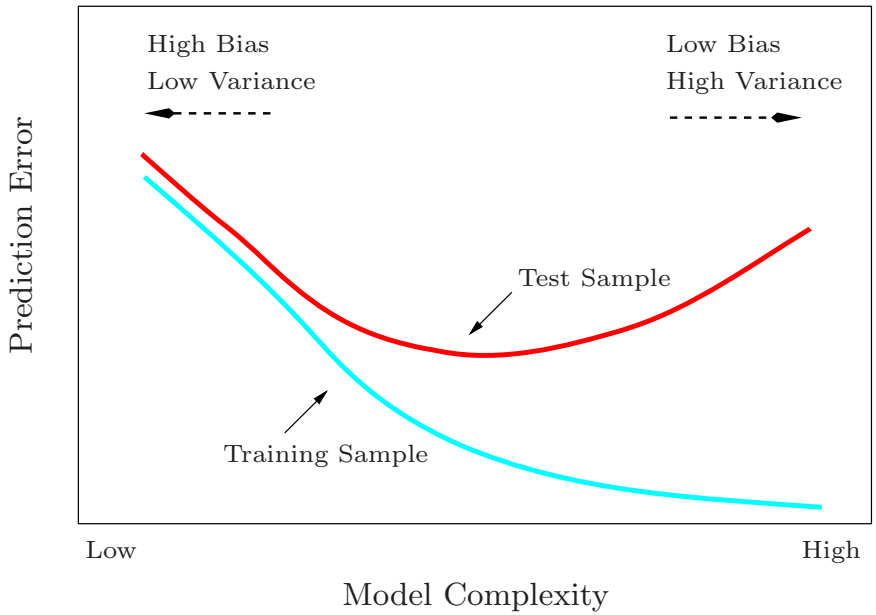
- More complex models always fit better:

$$\text{RMSE}_{\text{training}}(\text{simple model}) > \text{RMSE}_{\text{training}}(\text{complex model})$$

- But complex models may **overfit** the training data

$$\text{RMSE}_{\text{training}} \ll \text{RMSE}_{\text{test}}$$

- Too complex models **overfit** the data. **High variance**.
- Too simple models will **underfit** the data. **Large bias**.



Model complexity

- **Fitted values** for linear models

$$\hat{\boldsymbol{y}} = \boldsymbol{X}\,\hat{\boldsymbol{w}} = \boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y} = \boldsymbol{H}\boldsymbol{y}.$$

- **Hat matrix**

$$\underset{n\times n}{\boldsymbol{H}} = \boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T.$$

- The ith row of $\boldsymbol{H}$: how $\hat{y}_i$ depends on the n data points.
- A **linear smoother** is any fitting method of the form

$$\hat{\boldsymbol{y}} = \boldsymbol{H}\boldsymbol{y}.$$

- Ex: poly reg, splines, nearest neighbor, ridge regression ...
- **Degrees of freedom,** $\mathrm{tr}\boldsymbol{H}$, measures complexity of smoother.
- Sanity check: linear regression with p features: $\mathrm{tr}\boldsymbol{H} = p$.

Ridge regression

- Many features $\Rightarrow \hat{\boldsymbol{w}}_{LS} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$ has high variance.
- $\hat{\boldsymbol{y}} = \boldsymbol{X}\hat{\boldsymbol{w}}_{LS}$ can **overfit** the data. Poor prediction on test data.
- **Regularization**: "soft restrictions" on the estimated weights.
- Ridge estimator $\hat{\boldsymbol{w}}_{\text{ridge}}(\lambda)$ minimizes **L_2-penalized SSE**

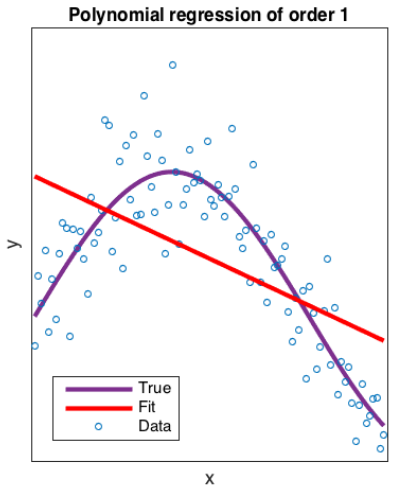
$$(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w})^T (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}) + \lambda \|\boldsymbol{w}\|_2^2$$

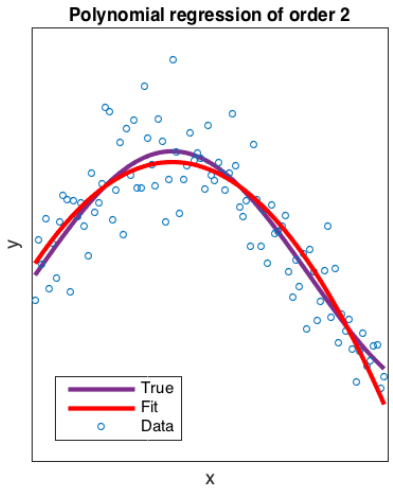
  where $\|\boldsymbol{w}\|_2^2 = \boldsymbol{w}^T\boldsymbol{w} = \sum_{j=1}^{p} w_j^2$ is **L_2-regularization**.
- Ridge regression estimator

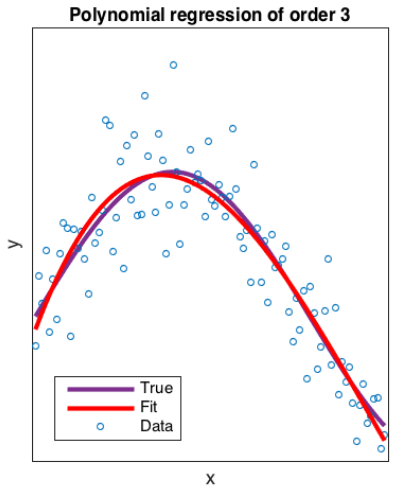
$$\hat{\boldsymbol{w}}_{\text{ridge}}(\lambda) = (\boldsymbol{X}^T\boldsymbol{X} + \lambda I_n)^{-1}\boldsymbol{X}^T\boldsymbol{y}.$$

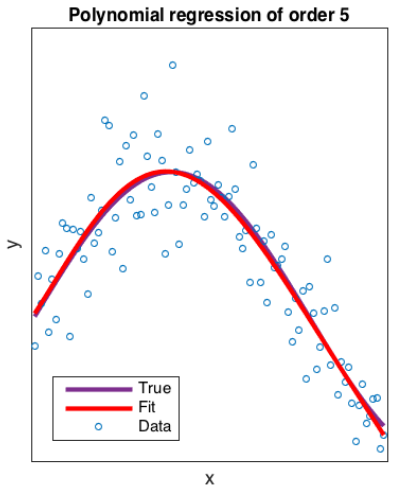
- $\hat{\boldsymbol{w}}_{\text{ridge}}(\lambda)$ **shrinks** $\hat{\boldsymbol{w}}$ toward zero as $\lambda \to \infty$.
- Trades **bias** against variance.

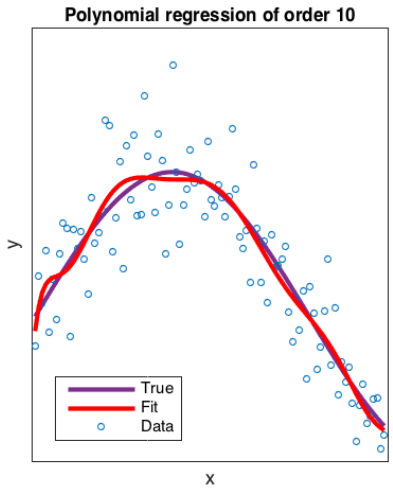
Polynomial regression without regularization

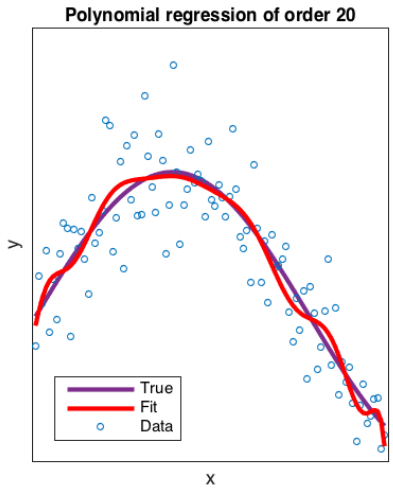




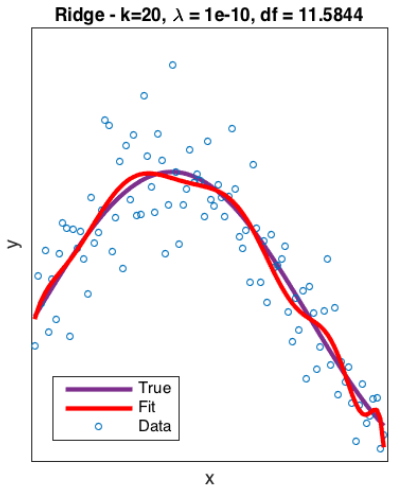


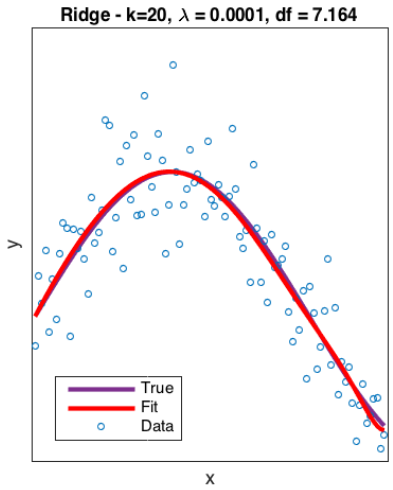


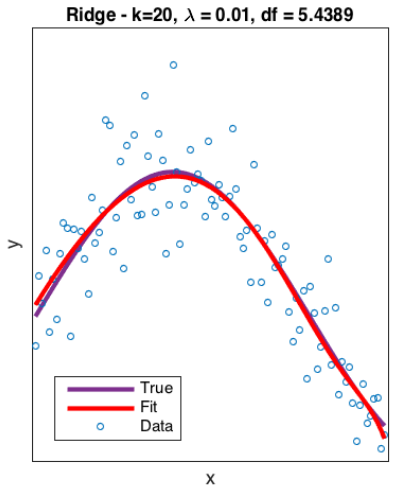


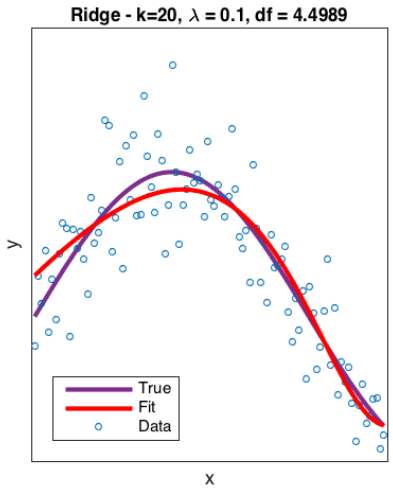


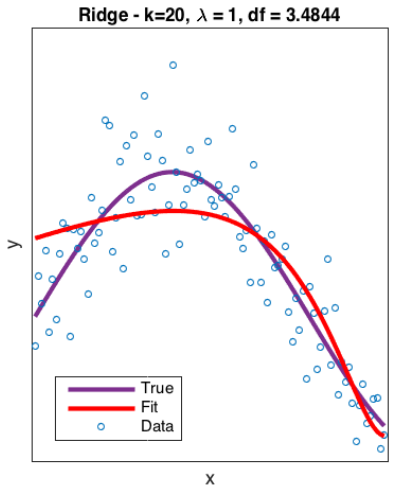
Polynomial regression with L_2-regularization

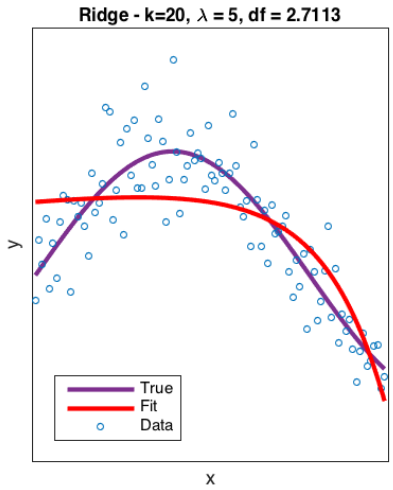












Lasso regression

- Lasso estimator $\hat{\boldsymbol{w}}_{\text{lasso}}(\alpha)$ minimizes L_1-penalized SSE

$$(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w})^T (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}) + \alpha \|\boldsymbol{w}\|_1$$

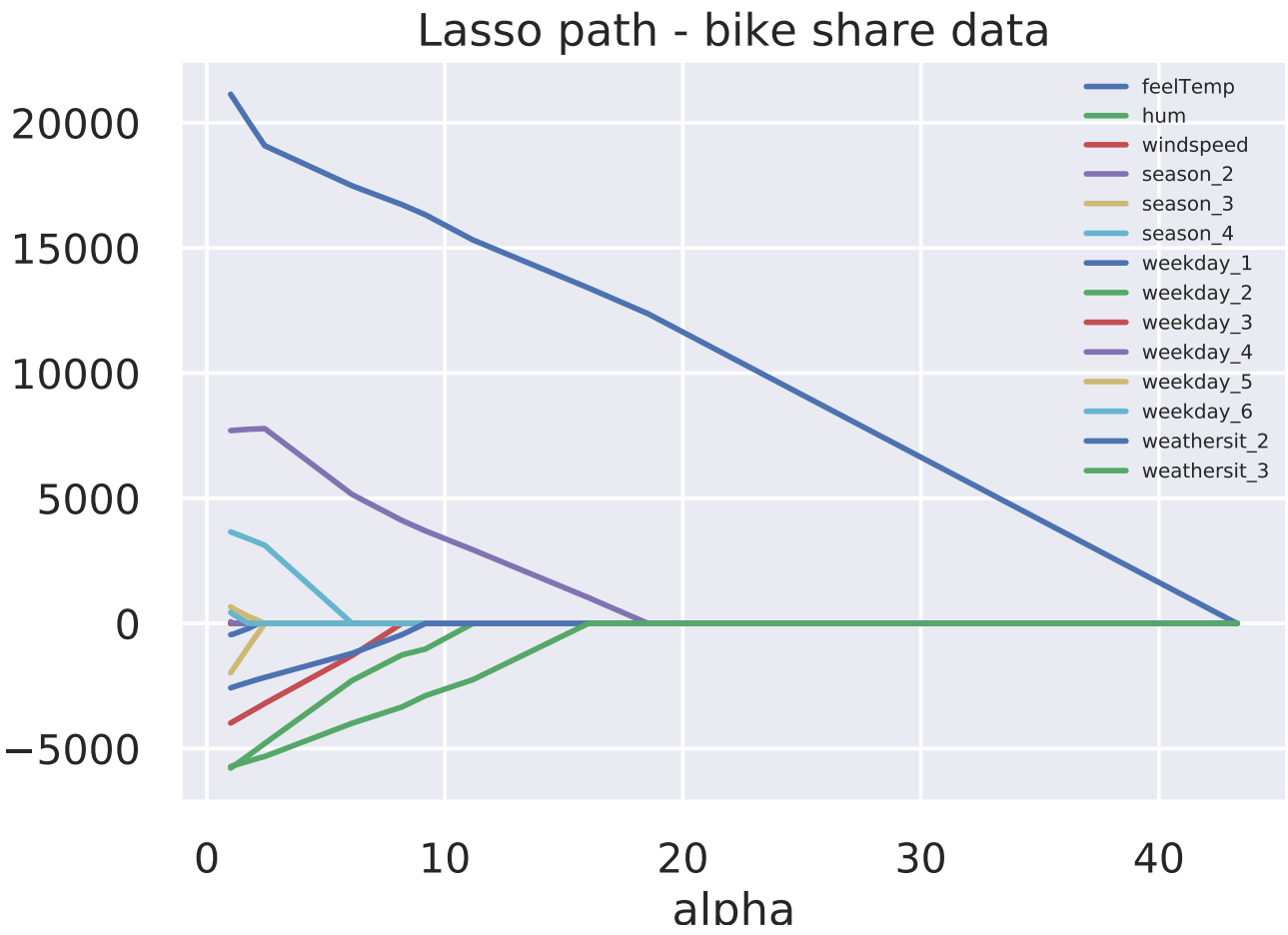
  where $\|\boldsymbol{w}\|_1 = \sum_{j=1}^{p} |w_j|$ is L_1-regularization.

- Lasso can shrink weights all the way to zero $\Rightarrow$variable selection.

- No explicit formula for $\hat{\boldsymbol{w}}_{\text{lasso}}(\alpha)$.

- LARS is a super-fast algorithm for computing the entire Lasso path.

Lasso regression - bike sharing data



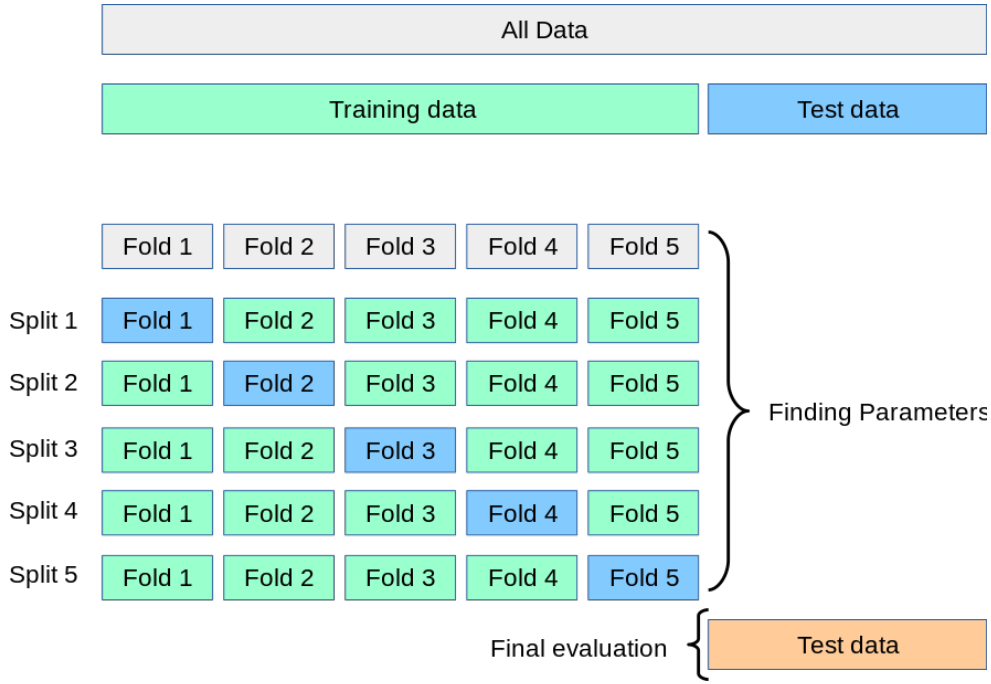
Local-Global Regularization

- Ideal shrinkage:
  - hard shrinkage of weights on noise features
  - leave weights on signal features untouched
- Ridge: shrinks all weights similarly.
- Lasso: can allow some signal features to have larger weights.
- Both Ridge and Lasso apply **global shrinkage**.

- Better variable selection with **global-local shrinkages**
  - **Global shrinkage**, α, is baseline shrinkage for all features
  - **Local shrinkage**, $\tau_1, \ldots, \tau_p$, for each feature.
  - Total shrinkage on jth feature: $\alpha\tau_j$.
- Example: **Horseshoe regularization**.
- More on this in the Bayesian learning part of Lecture Block 2.

Hyperparameter learning by Cross-validation

- How to estimate the **hyperparameter** α in Lasso regression?
- Bad idea: Find α that minimizes prediction errors on test data.
- **Cross-validation**: estimate generalization from training data.




```
from sklearn.model_selection import cross_val_score
score = cross_val_score(regModel, X = XTrain, y = yTrain, cv=5, scoring='neg_mean_squared_error')
```

- 5-10 folds is common, but depends on model complexity.
- **Leave-one-out CV** uses n_{training} folds.