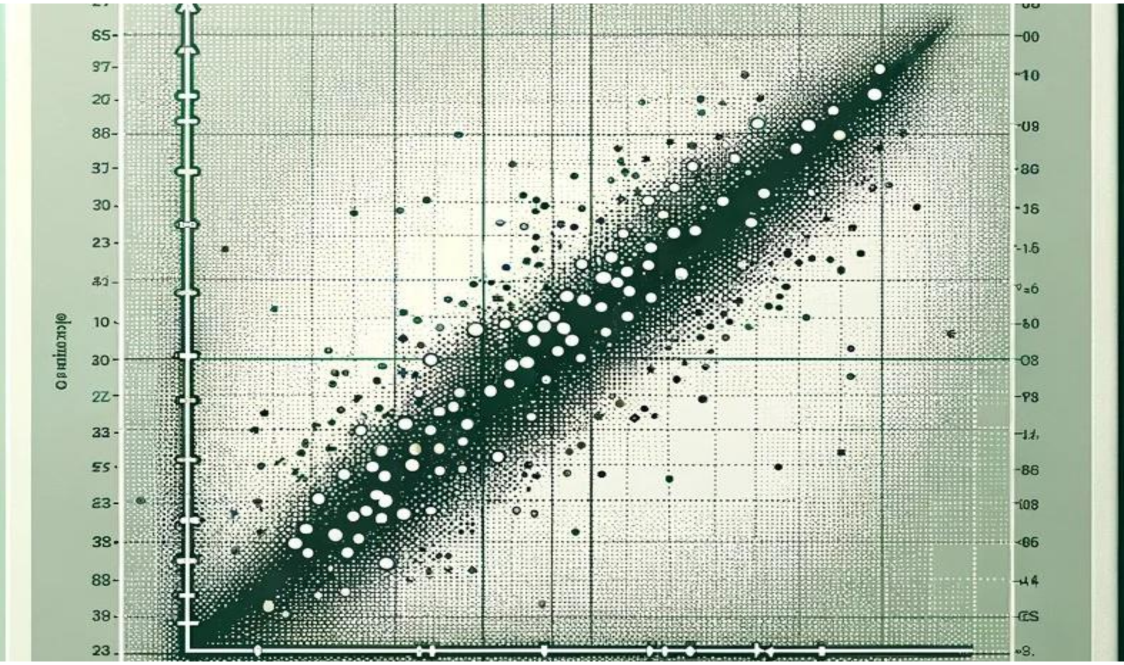
Hochschule Karlsruhe
University of
Applied Sciences

Fakultät für
Elektro- und
Informationstechnik

Linear Regression (2)



Source: DALL.E

Kawther Aboalam

Linear Regression

Multiple Linear Regression

+ So far, models with one feature have been discussed
+ The approach is generalized in the following to multiple features.

Linear Regression

Multiple Linear Regression – Matrix Notation

+ Assuming the number of training examples (training dataset) is n and the number of dependant variable (features) is m, the independent variable Y and the predicted independent variable $\hat{Y}$ can be represented as follows:

$$\begin{array}{cccccccc} & n\times 1 & & \text{n} \times \text{m+1} & & \text{m+1} \times 1 & & \text{n} \times 1 \\ Y = & \begin{bmatrix} y_1 \\ y_2 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ y_n \end{bmatrix} & = & \begin{bmatrix} 1 & x_{11} & \dots & x_{1m} \\ 1 & x_{21} & \dots & x_{2m} \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ 1 & x_{n1} & \dots & x_{nm} \end{bmatrix} & \cdot & \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_m \end{bmatrix} & + & \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \epsilon_n \end{bmatrix} \end{array}$$

$$\begin{array}{cccccc} & n\times 1 & & \text{n} \times \text{m+1} & & \text{m+1} \times 1 \\ \hat{Y} = & \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \hat{y}_n \end{bmatrix} & = & \begin{array}{r} \\ X_2 \\ \\ \\ \\ \\ \\ \end{array} \begin{bmatrix} \boxed{1 \quad x_{11} \quad \dots \quad x_{1m}} \\ \boxed{1 \quad x_{21} \quad \dots \quad x_{2m}} \\ \cdot \qquad \cdot \qquad\qquad \cdot \\ \cdot \qquad \cdot \qquad\qquad \cdot \\ \cdot \qquad \cdot \qquad\qquad \cdot \\ \cdot \qquad \cdot \qquad\qquad \cdot \\ 1 \quad x_{n1} \quad \dots \quad x_{nm} \end{bmatrix} \begin{array}{l} X_1 \\ \\ \\ \\ \\ \\ \\ \end{array} & \cdot & \begin{bmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 \\ \vdots \\ \hat{\beta}_m \end{bmatrix} \\ & \begin{array}{c}\text{Predicted} \\ \text{Target} \\ \hat{Y}\end{array} & & \begin{array}{c}\text{Features} \\ \text{Matrix} \\ \text{X}\end{array} & & \begin{array}{c}\text{Estimated} \\ \text{Coefficients} \\ \hat{\beta}\end{array} \end{array}$$

Linear Regression

Multiple Linear Regression – Matrix Notation

+ Assuming the number of training observations(training dataset) is n and the number of dependant variable is m, the independent variable Y and the predicted independent variable $\hat{Y}$ can be represented as follows:

$$Y = X\beta + \epsilon \qquad and \qquad \hat{Y} = X\hat{\beta}$$

  – Where:
    – $Y \in \mathbb{R}^{n\times 1}$: vector of true target values
    – $X \in \mathbb{R}^{n\times(m+1)}$: features matrix (including column of 1s)
    – $\beta \in \mathbb{R}^{(m+1)\times 1}$: model parameters (coefficients + intercept)
    – $\epsilon \in \mathbb{R}^{n\times 1}$: error vector
    – $\hat{Y} \in \mathbb{R}^{n\times 1}$: vector of predicted target values
    – $\hat{\beta} \in \mathbb{R}^{(m+1)\times 1}$: model estimated parameters (coefficients + intercept)

Linear Regression

Multiple Linear Regression – Model Training – Finding best $\hat{\beta}$

+ Approach: Least Squares to minimize the Sum of Squared Errors (SSE) (Loss Function L)

+ The goal is to find the optimal values of $\hat{\beta}$ such that minimizes

$$\mathrm{L} = \sum_{i=1}^{n}(y_i - \hat{y_i})^2$$
$$= \sum_{i=1}^{n}(y_i - X_i\hat{\beta}))^2$$

Where n is the number of the training examples, $y_i \in \mathbb{R}^{1\times 1},\ X_i\hat{\beta} \in \mathbb{R}^{1\times 1},\ X_i \in \mathbb{R}^{1\times(m+1)}\ and\ \hat{\beta} \in \mathbb{R}^{(m+1)\times 1}$

+ Taking partial derivatives of L with respect to $\hat{\beta}$ we obtain

$$\frac{\partial \mathrm{L}}{\partial \hat{\beta}} = -2\sum_{i=1}^{n} X_i^T(y_i - X_i\hat{\beta}) \overset{set}{=} 0 \qquad \sum_{i=1}^{n} X_i^T(y_i - X_i\hat{\beta}) = 0$$

Using matrix notation, we can rewrite the last equation as: $X^T(\mathrm{Y} - X\hat{\beta}) = 0$

Solving for $\hat{\beta}$: $\hat{\beta} = (X^TX)^{-1}X^TY$

Linear Regression

Multiple Linear Regression – Model Training – Finding best $\hat{\beta}$

+ The estimated values for $\hat{\beta}$ are the solution of the following equation

  $\hat{\beta} = (X^T X)^{-1} X^T Y$

+ This equation has a unique solution if $X^T X$ is invertible.
+ $X^T X$ is **not invertible** if:
  - rank(X) = rank($X^T X$) < its columns number (number of features+1)
  - Its determinant $= 0$ (singular matrix)
+ It is an indication that:
  - the columns of X are linearly dependent. Some columns (features) can be recreated by a linear combination of the others.

Linear Regression

Multiple Linear Regression – Nonlinear Effects

+ Conventionally, many statistical and machine learning models assume linearity. This simplifies the modeling process, allows for efficient computation and support the models interpretability.
+ However, in real-world problems, features often exhibit complex interactions that cannot be captured by linear relationships.
+ Feature Engineering for Nonlinear Effects:
  - Interaction Terms: Creating new features by multiplying existing features.
  - Polynomial Features: Introducing polynomial terms (e.g., squared, cubed, m-order features) allows models to capture curvature.

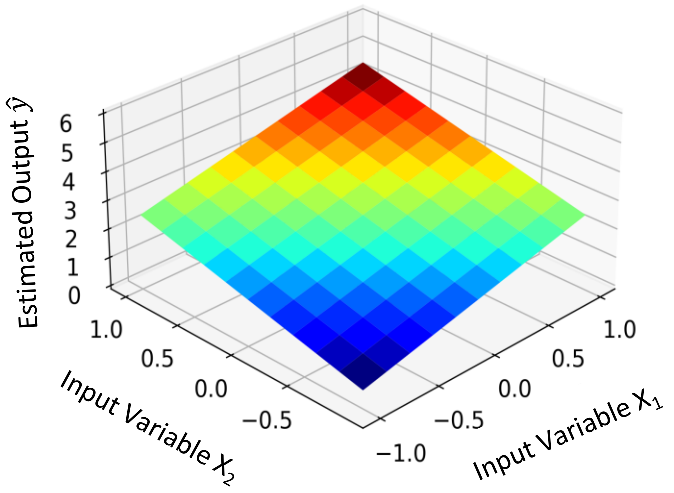
Linear Regression

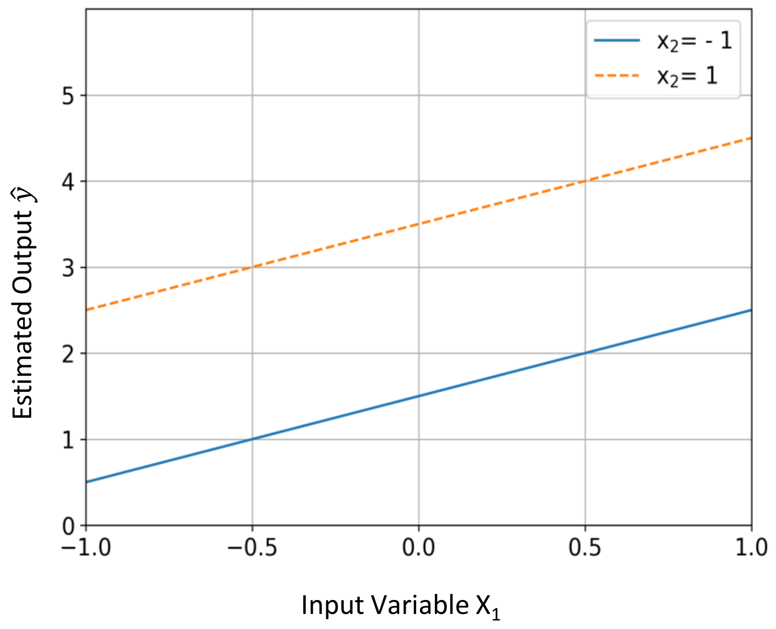
Multiple Linear Regression – Nonlinear Effects – Interaction Terms

+ A linear approach is characterized by the fact that the target variable is described via linear terms, the two input variables have no interaction

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 . x_1 + \hat{\beta}_2 . x_2$$

+ Along the axes, the coefficients $\hat{\beta}_1$ and $\hat{\beta}_2$ correspond to the slope of the surface, the target variable increases linearly with the variable x_1 regardless of the variable x_2
+ The relationship becomes particularly clear if the target variable is represented as a function of only one variable, in this case the variable x_1
+ Lines always have the same slope, they do not cross each other, parallel characteristics are characteristic for linear models without interaction





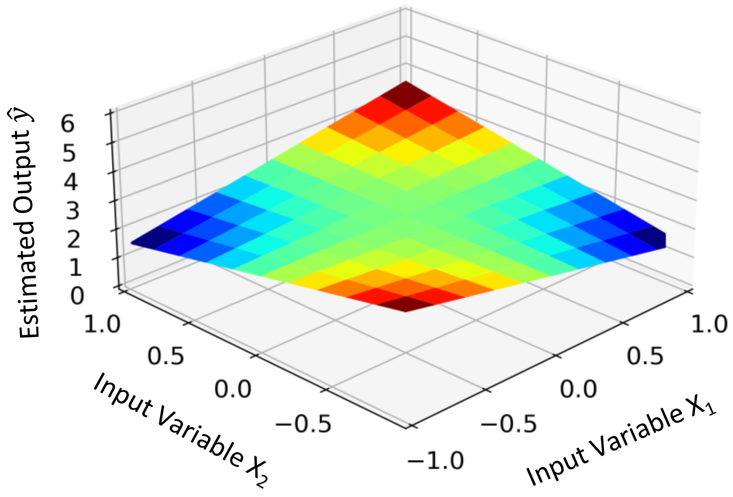
Linear Regression

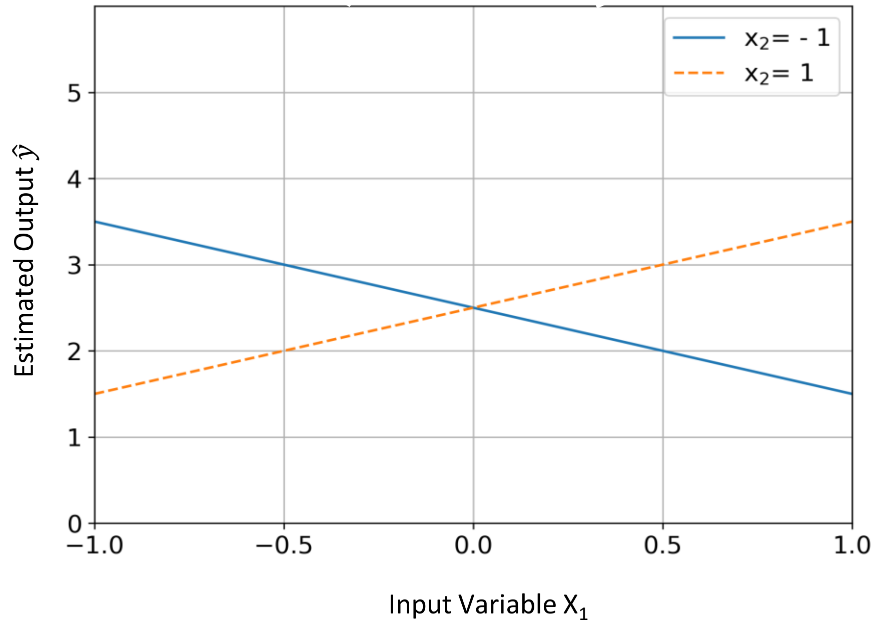
Multiple Linear Regression – Nonlinear Effects – Interaction Terms

+ The effect of the input variable x_1 depends on the input variable x_2 and vice versa

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_3 x_1 x_2$$

+ Depending on the variable x_2, the slope of the straight line $\hat{y}(x_1)$ in this example changes even if the sign changes





Linear Regression

Multiple Linear Regression – Nonlinear Effects – Polynomial Features

+ High-degree polynomials can capture more complex relationships between the input variable and the target variable. If the true relationship is indeed non-linear and complex, higher-degree terms might improve model performance.

+ Assuming a dataset that has two independent input variables (x_1, x_2),
  - Degree-2 polynomial (full quadratic representation) will be:

$$(1, x_1 , x_2 , x_1^2 , x_1 . x_2 , x_2^2)$$

  - Degree-3 polynomial $(1, x_1 , x_2 , x_1^2 , x_1 . x_2 , x_2^2 , x_1^3, x_1^2 . x_2 , x_1 . x_2^2 , x_2^3)$

  And so on.

+ Depending on your understanding of the machine learning problem, domain knowledge, and insights from exploratory data analysis, you might choose to omit some terms

X_1	X_2
0	5
2	2
1	3

Original Input Variables

1	x_1	x_2	x_1^2	$x_1 . x_2$	x_2^2
1	0	5	0	0	25
1	2	2	4	4	4
1	1	3	1	3	3

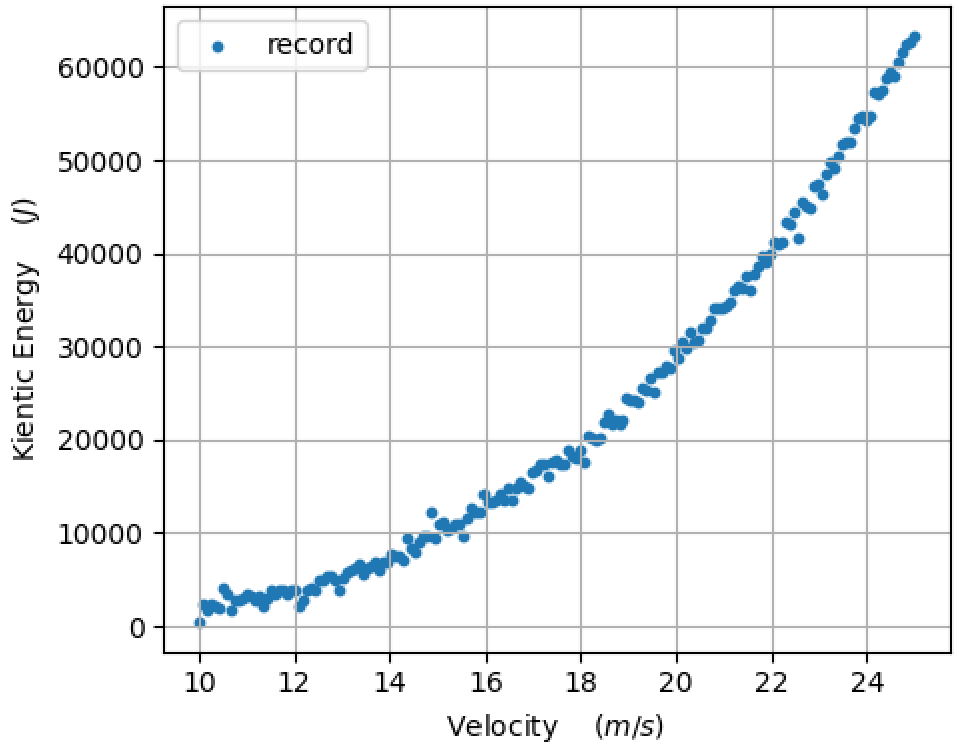
Full quadratic features representation (Degree-2 polynomial)

Linear Regression

Multiple Linear Regression – Polynomial Features – Example: Kinetic Energy

- Suppose that we have a dataset about estimating the kinetic energy
- The dataset contains two dependent input variables mass, M and velocity, V
- The sequence below is followed:
  1. Data visualization

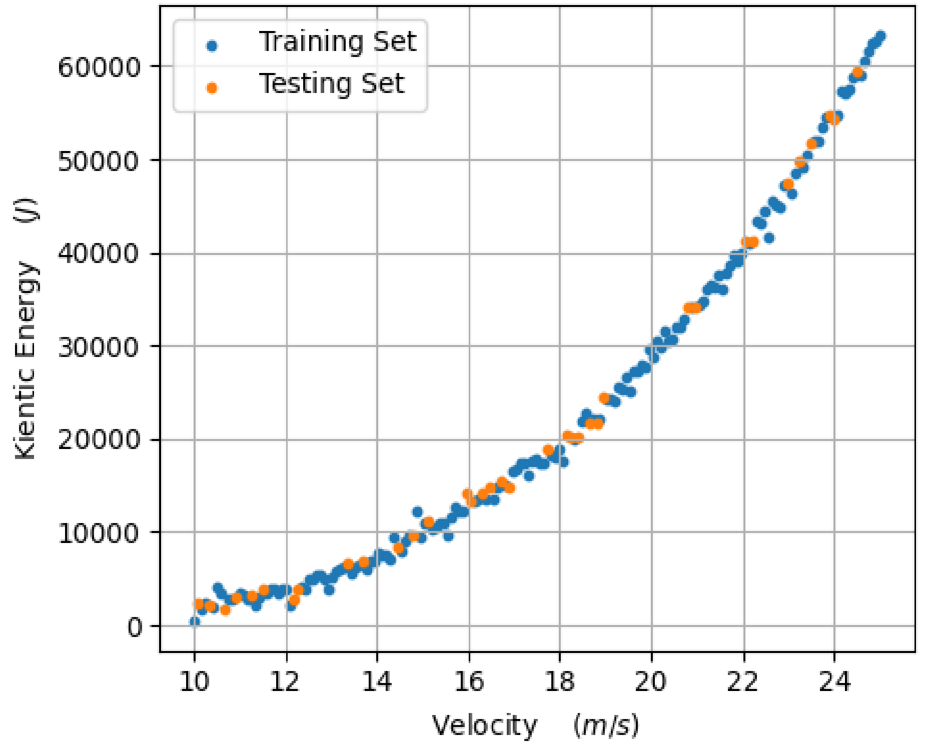
Index	KE	M	V
0	441.17	20	10.00
1	2433.20	21	10.08
2	1713.01	22	10.17
3	2455.48	23	10.25
4	2196.88	24	10.34



Linear Regression

Multiple Linear Regression – Polynomial Features – Example: Kinetic Energy

+ Suppose that we have a dataset about estimating the kinetic energy
+ The dataset contains two dependent input variables mass, M and velocity, V
+ The sequence below is followed:
  1. Data visualization by a scatter plot
  2. Train-Test-Split (80% for training and 20% for testing)

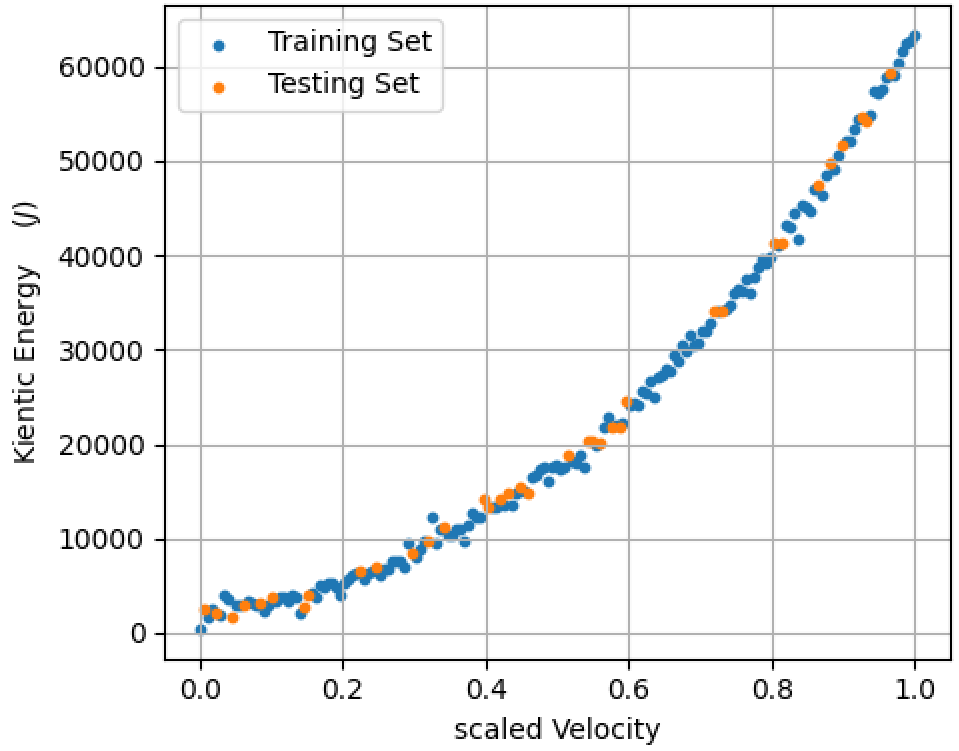


Linear Regression

Multiple Linear Regression – Polynomial Features – Example: Kinetic Energy

- Suppose that we have a dataset about estimating the kinetic energy
- The dataset contains two dependent input variables mass, M and velocity, V
- The sequence below is followed:
  1. Data visualization by a scatter plot
  2. Train-Test-Split (80% for training and 20% for testing)
  3. Feature scaling using Max-Min Normalization (or standardization)
     - Why should we split the data before scaling the features?
     - Scaling process involves calculating statistical parameters (e.g., minimum, maximum, mean, standard deviation) on the training set only (fit-transform) and then applying those parameters to normalize the testing set (transform). This ensures that the testing set remains separate from the training process (unseen).

```
scaler = MinMaxScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

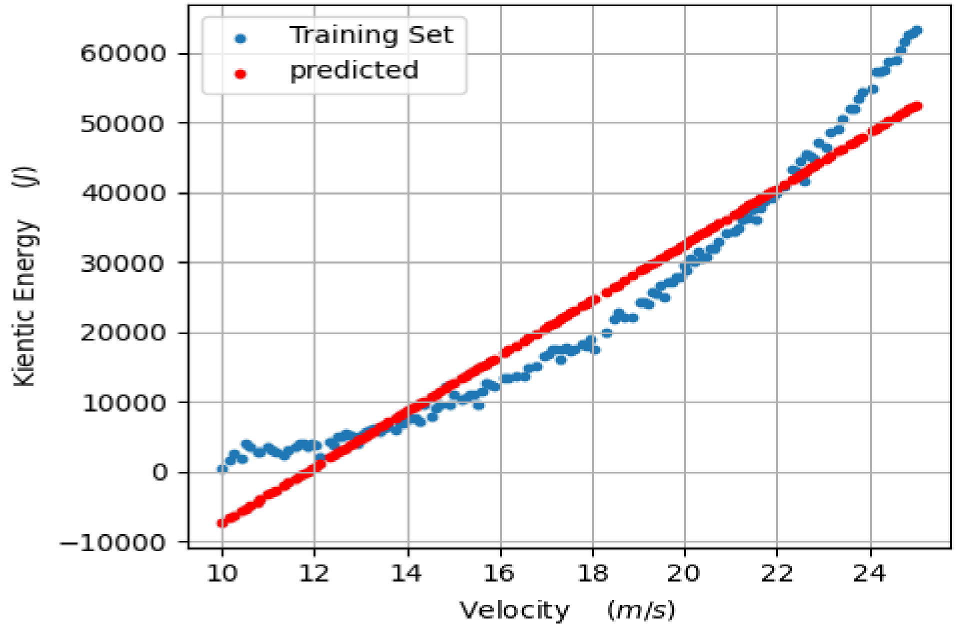


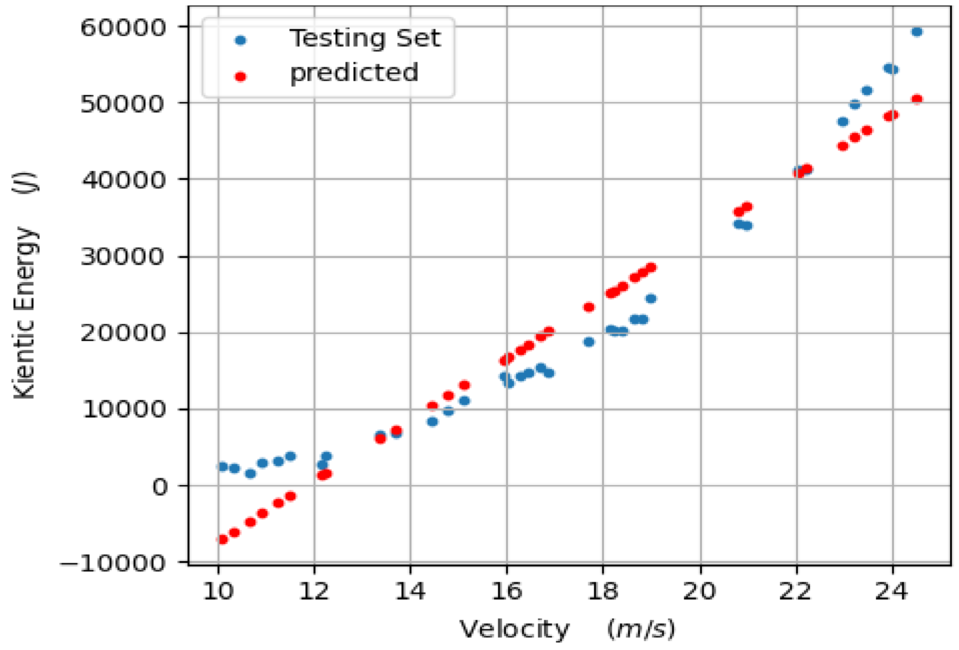
Linear Regression

Multiple Linear Regression – Polynomial Features – Example: Kinetic Energy

+ Suppose that we have a dataset about estimating the kinetic energy
+ The dataset contains two dependent input variables mass, M and velocity, V
+ The sequence below is followed:
  1. Data visualization by a scatter plot
  2. Train-Test-Split (80% for training and 20% for testing)
  3. Feature scaling using Max-Min Normalization (or standardization)
  4. Train a linear regression model (solve for the model coefficients)
  5. Model evaluation on both the training and testing sets.

	R^2	MSE	MAE
Train	0.94	20861938.11	3829.80
Test	0.93	22450539.66	4119.19





Linear Regression

Multiple Linear Regression – Polynomial Features – Example: Kinetic Energy

+ Kinetic energy of a body is equal to one-half the product of its mass, M, and the square of its velocity, V, ($1/2mv^2$)
+ Adding a new feature as the squared velocity can improve the performance of the model

Index	KE	M	V
0	441.17	20	10.00
1	2433.20	21	10.08
2	1713.01	22	10.17
3	2455.48	23	10.25
4	2196.88	24	10.34

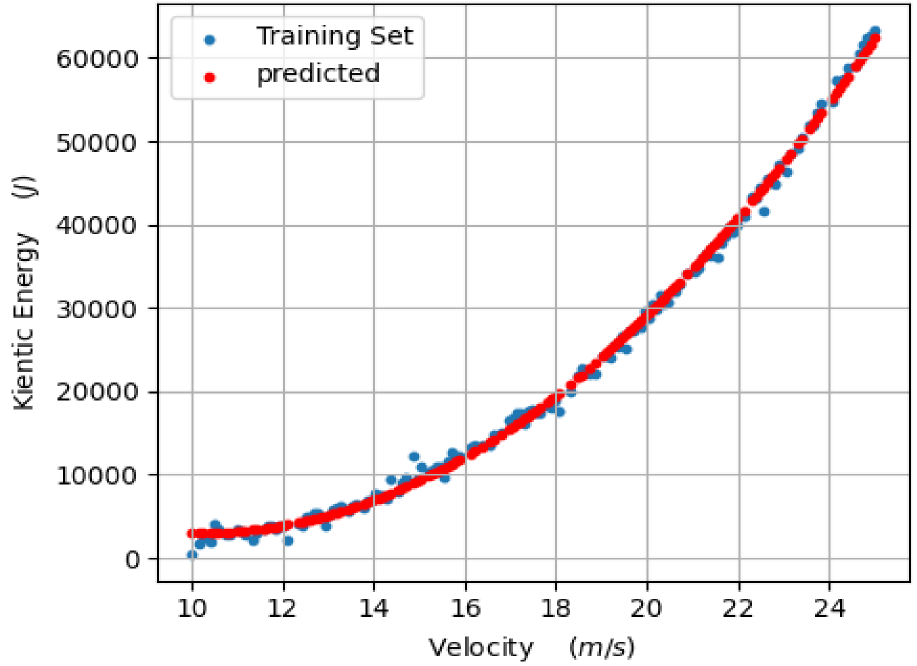
Index	KE	M	V	Vsquare
0	441.17	20	10.00	100.00
1	2433.20	21	10.08	101.68
2	1713.01	22	10.17	103.38
3	2455.48	23	10.25	105.09
4	2196.88	24	10.34	106.82

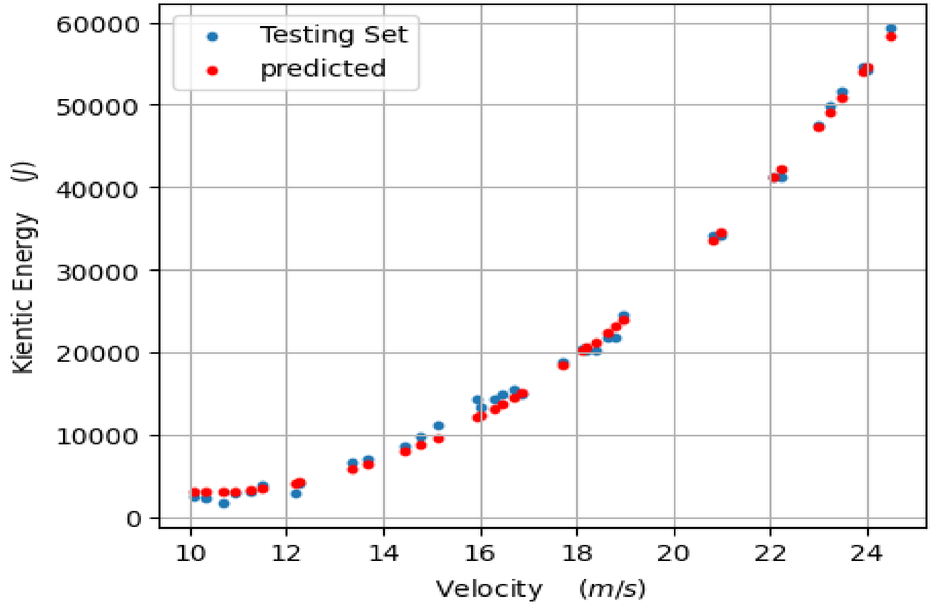
Linear Regression

Multiple Linear Regression – Polynomial Features – Example: Kinetic Energy

- Kinetic energy of a body is equal to one-half the product of its mass, M, and the square of its velocity, V, ($1/2mv^2$)
- Adding a new feature as the squared velocity can improve the performance of the model
- The same sequence is followed but starting with data splitting:
  - Train-Test-Split
  - Feature scaling using Max-Min Normalization
  - Train a linear regression model (solve for the model coefficients)
  - Model evaluation on both the training and testing sets.

	R²	MSE	MAE
Train	1.00	743260.54	660.01
Test	1.00	756358.20	735.24

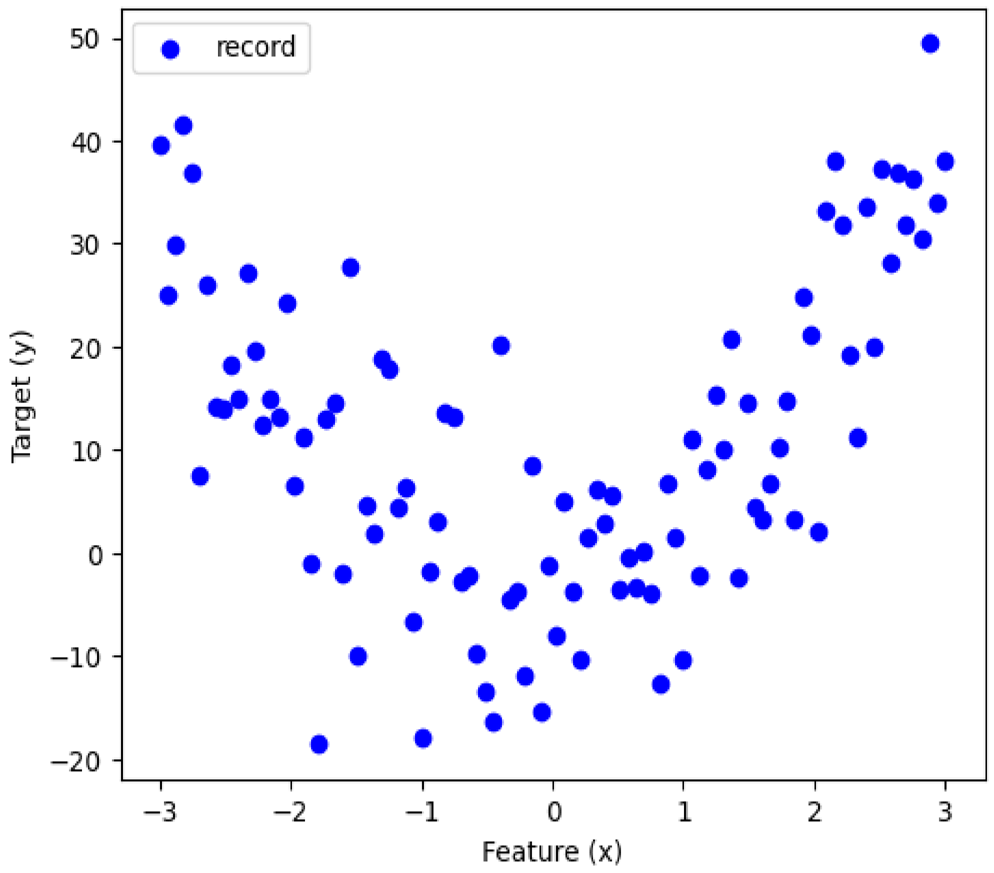




Linear Regression

Multiple Linear Regression – Model Complexity Versus Training and Testing Error

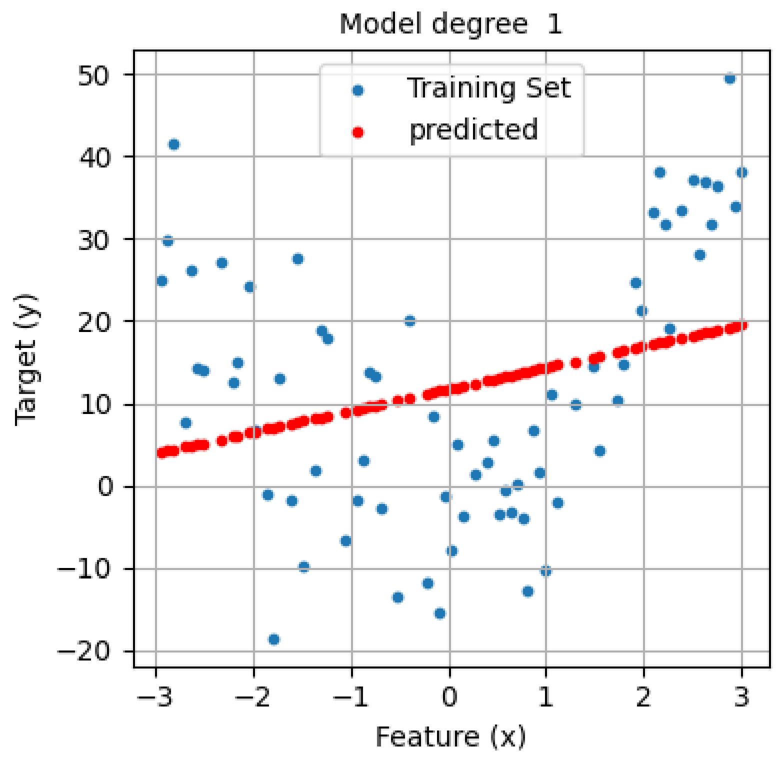
- A synthetic dataset that includes one dependent variable *x* and a target *y*, where *y* has a true underlying relationship with *x* (e.g., a polynomial relationship) plus some noise

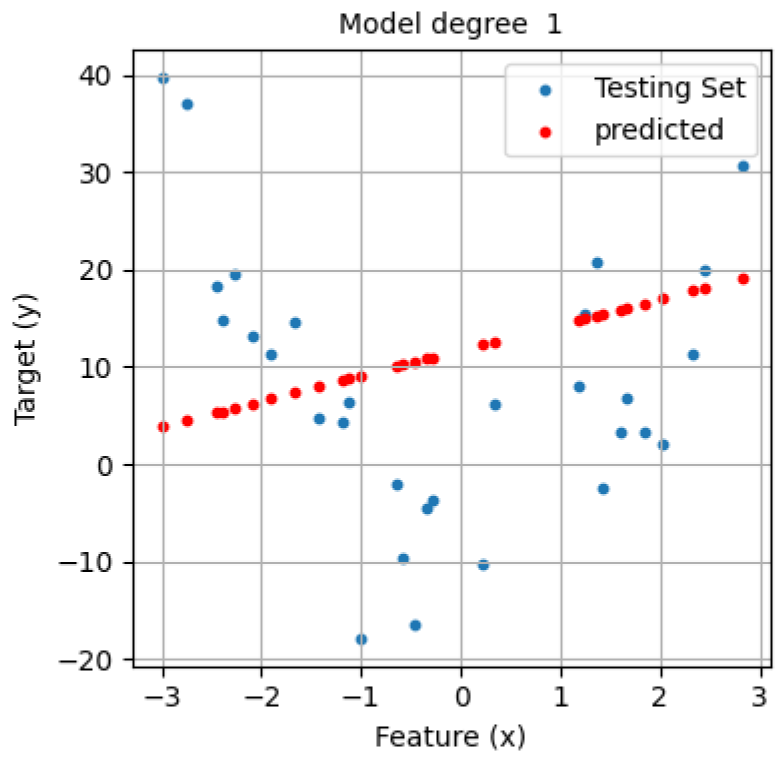


Linear Regression

Multiple Linear Regression – Model Complexity Versus Training And Testing Error

- Fit multiple models of increasing complexity (e.g., linear, quadratic, cubic, etc.) to the training data
- Evaluate each model's performance on both the training and testing sets
- Plot the training and testing errors as a function of model complexity.

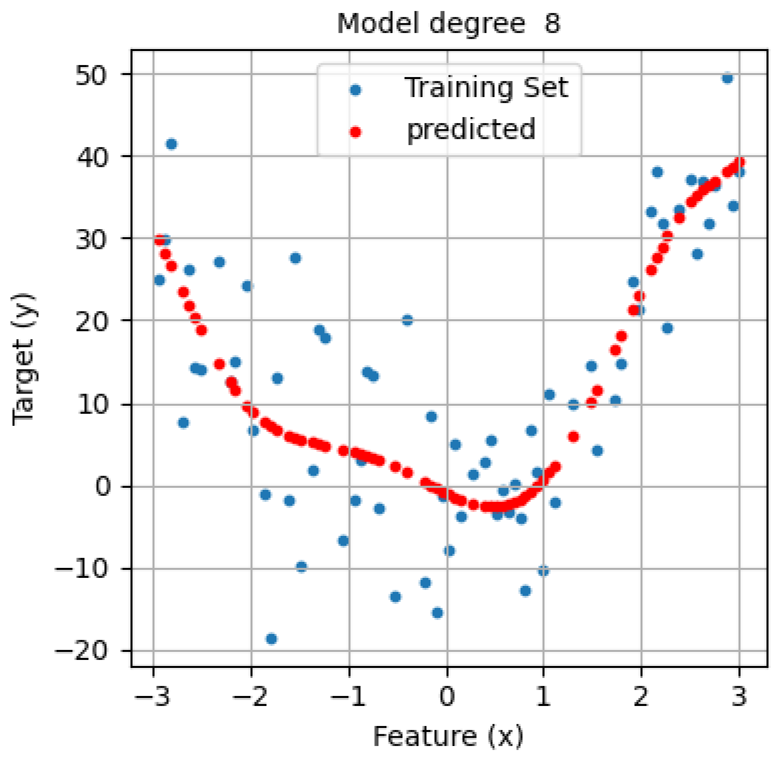


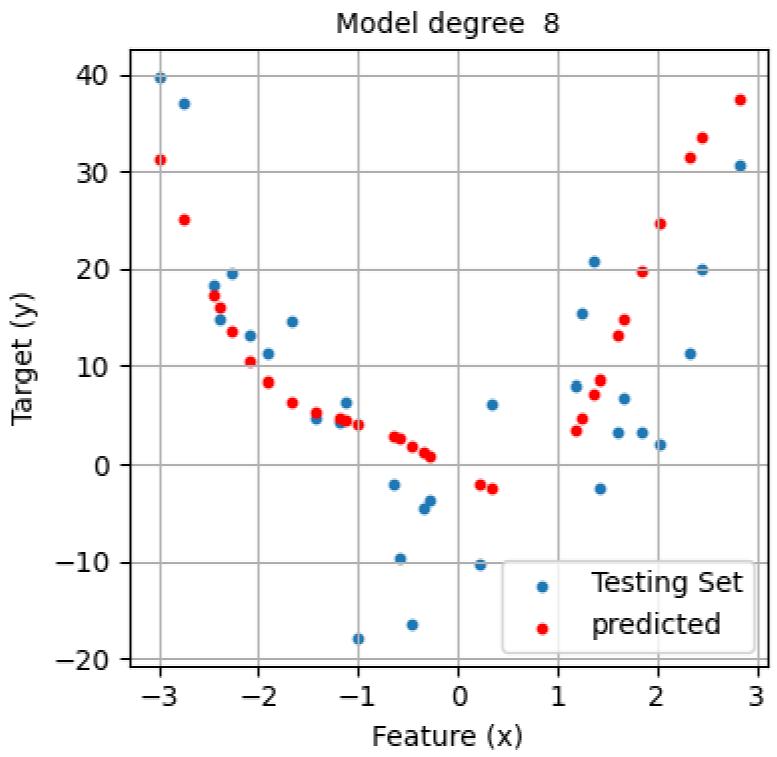


Linear Regression

Multiple Linear Regression – Model Complexity Versus Training And Testing Error

- Fit multiple models of increasing complexity (e.g., linear, quadratic, cubic, etc.) to the training data
- Evaluate each model's performance on both the training and testing sets
- Plot the training and testing errors as a function of model complexity.

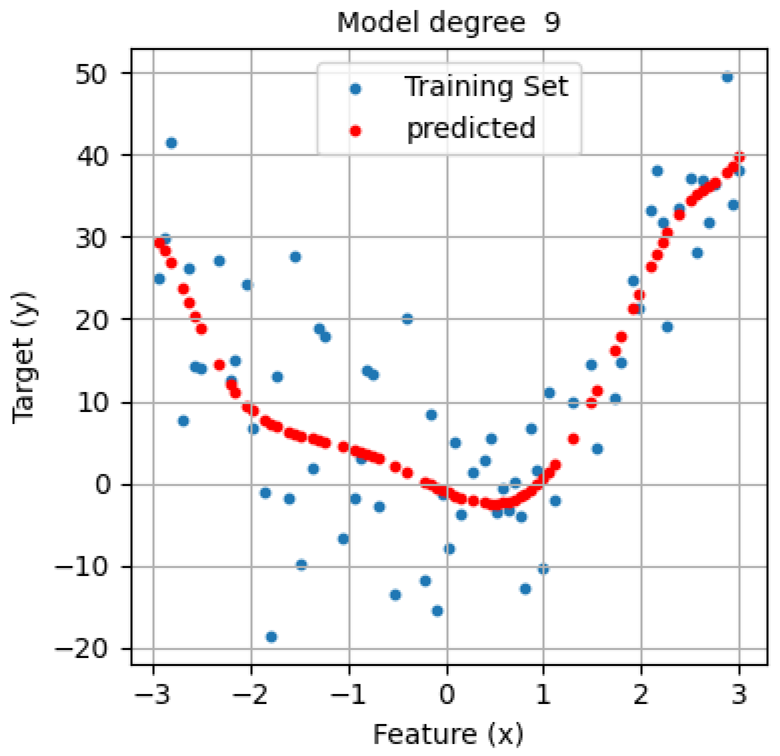


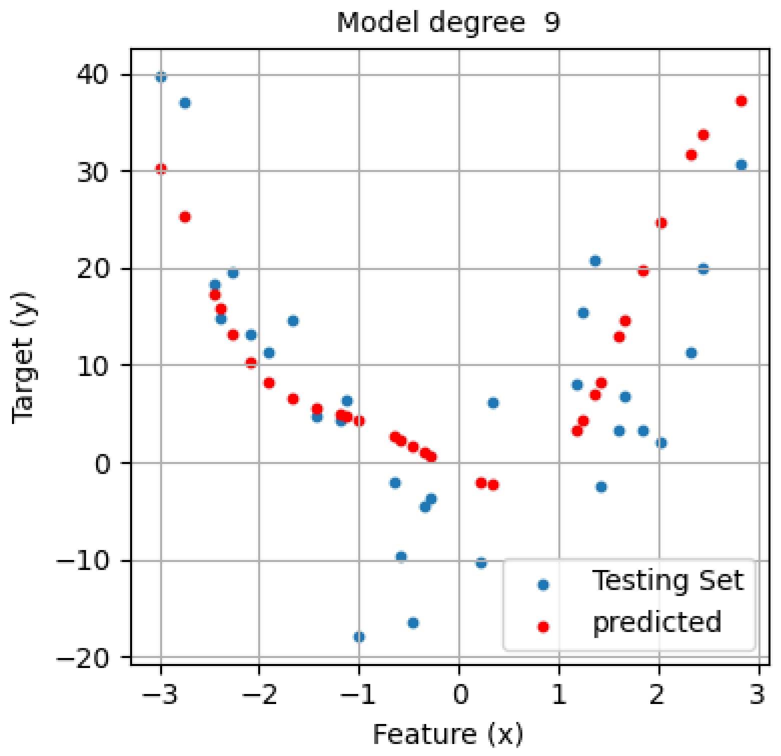


Linear Regression

Multiple Linear Regression – Model Complexity Versus Training And Testing Error

+ Fit multiple models of increasing complexity (e.g., linear, quadratic, cubic, etc.) to the training data
+ A higher-order polynomial function is more complex than a lower-order polynomial function, since the higher-order polynomial has more parameters and the model function's selection range is wider.
+ Evaluate each model's performance on both the training and testing sets.
+ Plot the training and testing errors as a function of model complexity.

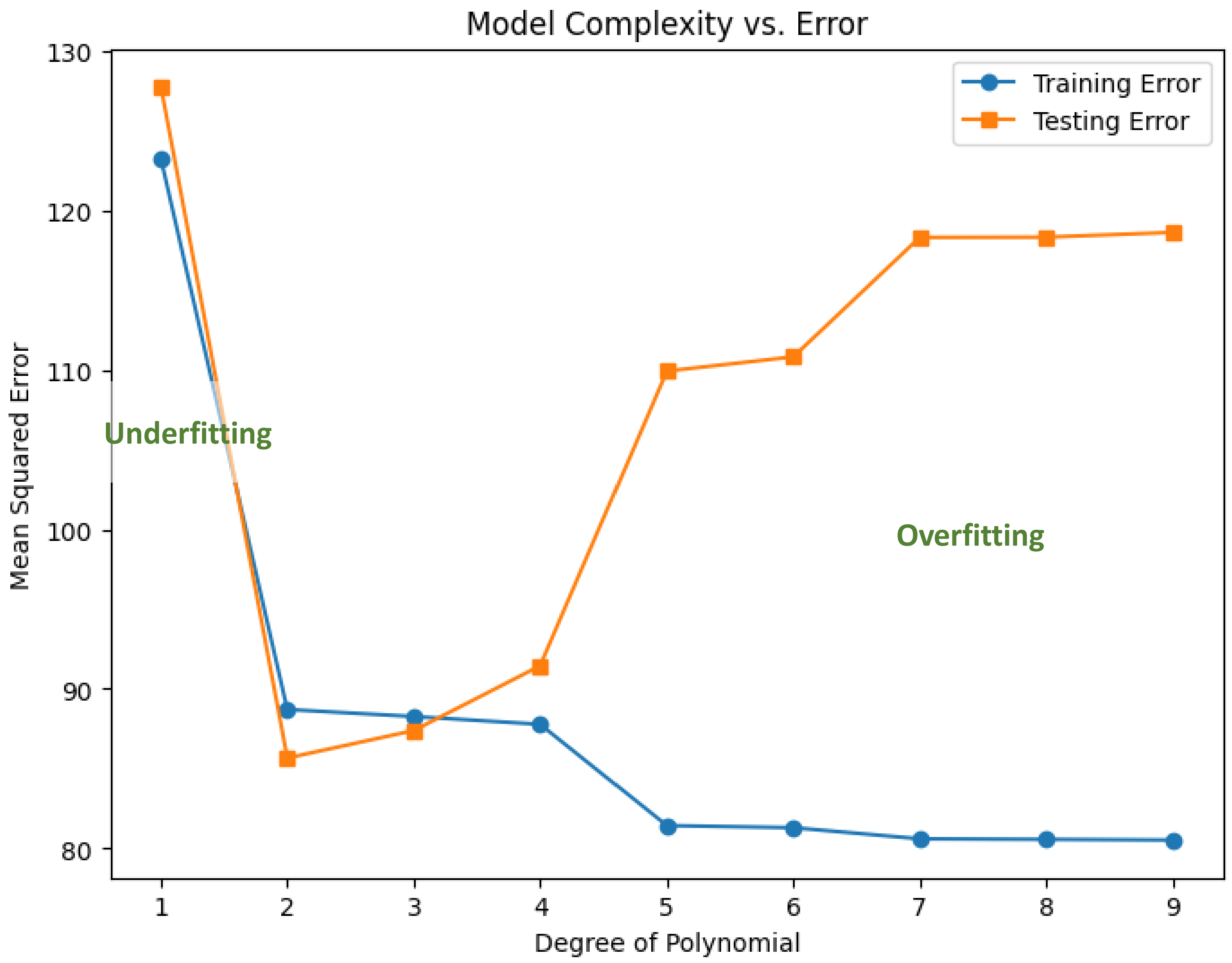




Linear Regression

Multiple Linear Regression – Model Complexity Versus Training And Testing Error

- A regression model with degree 1 maps the dataset with a simple line, may failing to capture the underlying pattern—this leads to underfitting.
- In contrast, a high-degree polynomial model may capture noise in the training data, resulting in overfitting.
- The Training Error decreases consistently as the model becomes more complex.
- However, testing error follows a U-shaped trend: it decreases at first—indicating better generalization—but starts increasing past a certain point, signaling overfitting.

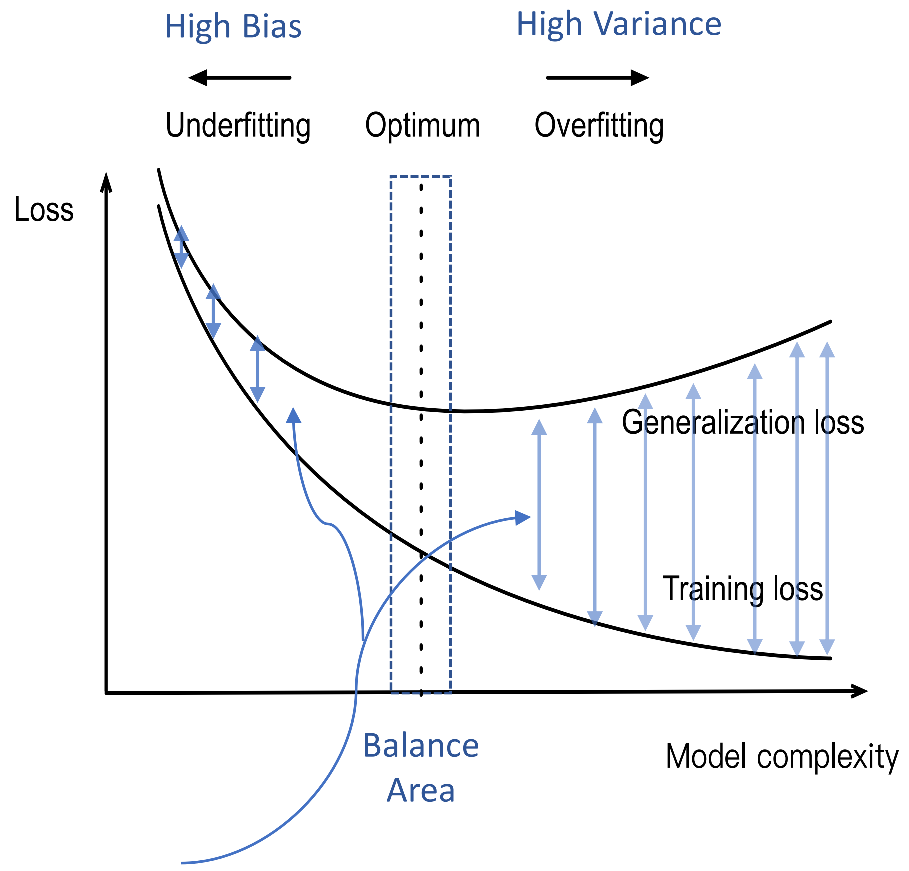


Linear Regression

Overfitting and Underfitting

- Overfitting
  - Overfitting occurs when a model represents training data very well, but unknown test data is inadequately represented.
  - The reason for this is an elaborate and detailed mapping of the training dataset with all individual disturbances, which are, however, typical only for the training data
  - For unknown test data, however, this level of detail is inappropriate because the test data set will have different individual perturbations
- Underfitting
  - Underfitting occurs if the model is unable to reduce the training error, that could mean that our model is too simple (i.e., insufficiently expressive) to capture the pattern that we are trying to model.

- The graph shows the influence of model complexity on underfitting and overfitting



Generalization Gap: difference between training and generalization errors

Linear Regression

Bias-Variance Tradeoff

- Bias$[\hat{y}] = \mathrm{E}(\hat{y}) - y$ and Var$[\hat{y}] = \mathrm{E}((\mathrm{E}(\hat{y}) - \hat{y})^2)$, where $\mathrm{E}(\hat{y})$ is the expectation value of $\hat{y}$
- $$\begin{aligned} E\left((\hat{y} - y)^2\right) &= E\left((\hat{y} - \mathrm{E}(\hat{y}) + \mathrm{E}(\hat{y}) - y)^2\right) \\ &= E\left((\hat{y} - \mathrm{E}(\hat{y}))^2 + 2 \cdot (\hat{y} - \mathrm{E}(\hat{y})) \cdot (\mathrm{E}(\hat{y}) - y) + (\mathrm{E}(\hat{y}) - y)^2\right) \\ &= E\left((\hat{y} - \mathrm{E}(\hat{y}))^2\right) + 2 \cdot E\left((\hat{y} - \mathrm{E}(\hat{y})) \cdot (\mathrm{E}(\hat{y}) - y)\right) + E\left((\mathrm{E}(\hat{y}) - y)^2\right) \end{aligned}$$
- Simplifications lead to

$$E\left((\hat{y} - y)^2\right) = E\left((\hat{y} - \mathrm{E}(\hat{y}))^2\right) + E\left((\mathrm{E}(\hat{y}) - y)^2\right)$$

$$= E\left((\mathrm{E}(\hat{y}) - y)^2\right) + E\left((\hat{y} - \mathrm{E}(\hat{y}))^2\right)$$

$$MSE = Bias^2 + Variance$$

- A good model has balance between bias and variance, mean squared error becomes minimal

$$\mathrm{MSE} = \mathrm{Variance} + \mathrm{Bias}^2 \overset{!}{=} \mathrm{Min.}$$

Linear Regression

How to Avoid Underfitting and Overfitting?

Underfitting (High Bias)

+ Increase model complexity – Use a more complex model or add more relevant features to increase the input space's dimensionality, allowing the model to learn more complex patterns
+ Reduce noise in the data

Overfitting (High Variance)

+ Simplify the model – Choose a simpler model with fewer parameters.
+ Train with more data – Having more data can help the model to improve its performance and generalizability, reducing the chances of overfitting
+ Regularization – Techniques like L1 and L2 regularization add a penalty on the magnitude of model parameters. This discourages the model from becoming too complex.

Linear Regression

Regularization

+ A linear model can be represented as:

$$\hat{Y} = X\hat{\beta}$$

where X is the feature matrix and $\hat{\beta}$ is the coefficients vector

+ For a given feature x_i , a large coefficient $\hat{\beta}_i$ means that the predicted output is highly sensitive to x_i. Therefore, a small variation in x_i can lead to a significant change in the predicted value, which often signifies an overfitting problem.

+ Regularization adds the norm of the coefficients vector (L1 norm or L2 norm) multiplied by a hyperparameter α to the loss function, so that the coefficients $\hat{\beta}$ should be as small as possible.

+ The two most common types of regularization are L1 regularization (Lasso) and L2 regularization (Ridge) regularization

Linear Regression

Regularization – L2 Regularization – Ridge Regression

+ The regularization term in Ridge regression is the L2 norm of the coefficients, squared.

$$L = \sum_{i=1}^{n}(y_i - X_i\hat{\beta}))^2 + \alpha \sum_{i=1}^{m+1} \hat{\beta}^2,$$

$$L = RSS + \alpha \sum_{i=1}^{m+1} \hat{\beta}^2$$

where m is the number of features, n is the number of training examples, and RSS is the residual sum of squares.

+ Least squares optimization approach leads to the matrix equation:

$$\hat{\beta} = (X^T X + \alpha I)^{-1} X^T Y$$

+ Adding α on the main diagonal decreases the values of the coefficients.
+ A compromise between decreasing the summation of residuals squared and decreasing summation of the squared coefficients has to be made when choosing the hyperparameter α.

Linear Regression

Regularization – L1 Regularization – Lasso Regression

+ Lasso refers to Least Absolute Shrinkage and Selection Operator regression
+ The regularization term in Lasso regression is the L1 norm of the coefficients

$$L = \sum_{i=1}^{n} (y_i - X_i \hat{\beta})^2 + \alpha \sum_{i=1}^{m+1} \left| \hat{\beta}_i \right|$$

+ Comparing the loss function of Ridge and lasso, we see that the lasso and ridge regression have similar formulations
+ The only difference is that the $\hat{\beta}^2$ term in the ridge regression penalty has been replaced by $\left| \hat{\beta}_i \right|$
+ As with ridge regression, the lasso shrinks the coefficient estimates towards zero. However, in the case of the lasso, the L1 penalty has the effect of forcing some of the coefficient estimates to be exactly equal to zero when the tuning hyperparameter α is sufficiently large.
+ A compromise between decreasing the summation of residuals squared and decreasing summation of the absolute values of the coefficients has to be made when choosing the hyperparameter α

Linear Regression

Regularization – L1 Regularization – Lasso Regression

+ One can show that the lasso and ridge regression solve the problem by

  choosing $\hat{\beta}$ that minimize $\{RSS\}$ with subject to $\sum_{i=1}^{m+1}|\hat{\beta}_i| \leq s$ (s is the constraint region)

region)

  and

  choosing $\hat{\beta}$ that minimize $\{RSS\}$ with subject to $\sum_{i=1}^{m+1}\hat{\beta}_i^{\ 2} \leq s$, respectively.

+ For every value of α, there is s, where the above equations will give the same estimated coefficients by minimizing the loss functions of Lasso and Ridge regression, respectively.

+ The relationship between *s* and α is essentially inverse. A larger value of α corresponds to a stronger penalty. But a larger value *s* allows for a larger sum of the absolute values or squared values of the coefficients, leading to a less strict penalty.

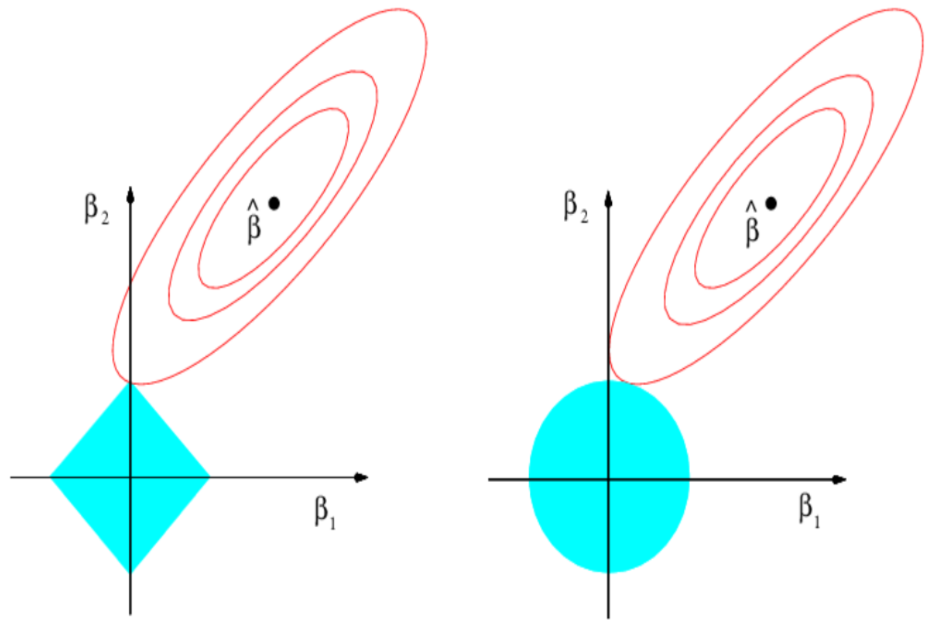
+ When m+1 = 2, then indicates that the lasso regression has the smallest RSS out of all points that lie within the diamond defined by $|\beta_1|+|\beta_2| \leq s$. Similarly, the ridge regression has the smallest RSS out of all points that lie within the circle defined by $\beta_1^2 + \beta_2^2 \leq s$.

https://hastie.su.domains/ISLP/ISLP_website.pdf.download.html

Linear Regression

Regularization – L1 Regularization – Lasso Regression

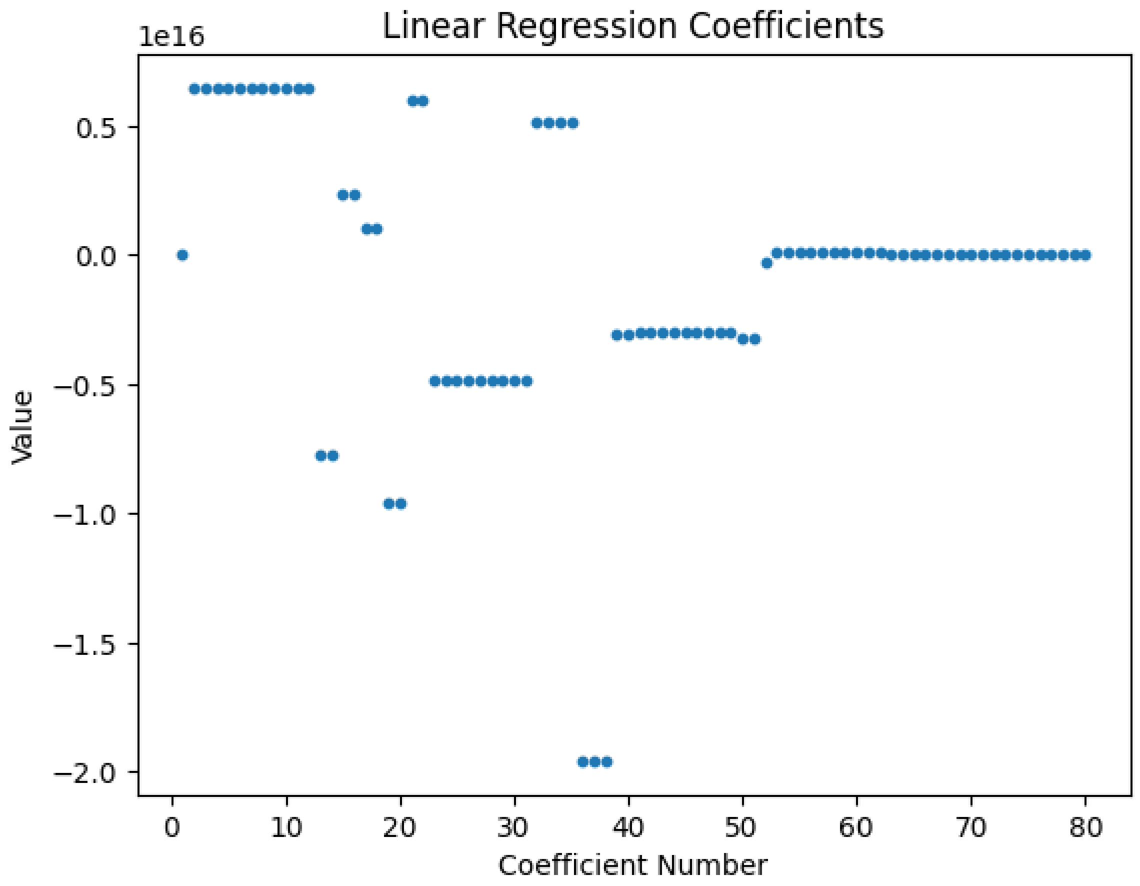
- Each of the ellipses centered around $\hat{\beta}$ represents a contour: this means contour that all of the points on a particular ellipse have the same RSS value
- As the ellipses expand away from the least squares coefficient estimates, the RSS increases
- The lasso and ridge regression coefficient estimates are given by the first point at which an ellipse contacts the constraint region.
- Since ridge regression has a circular constraint with no sharp points, this intersection will not generally occur on an axis, and so the ridge regression coefficient estimates will be exclusively non-zero.
- However, the lasso constraint has corners at each of the axes, and so the ellipse will often intersect the constraint region at an axis. When this occurs, one of the coefficients will equal zero.
- In higher dimensions, many of the coefficient estimates may equal zero simultaneously.



Contours of the error and constraint functions for the lasso (left) and ridge regression (right). The solid blue areas are the constraint regions, $|\beta_1|+|\beta_2| \leq s$ and $\beta_1^2 + \beta_2^2 \leq s$, while the red ellipses are the contours of the RSS

Linear Regression

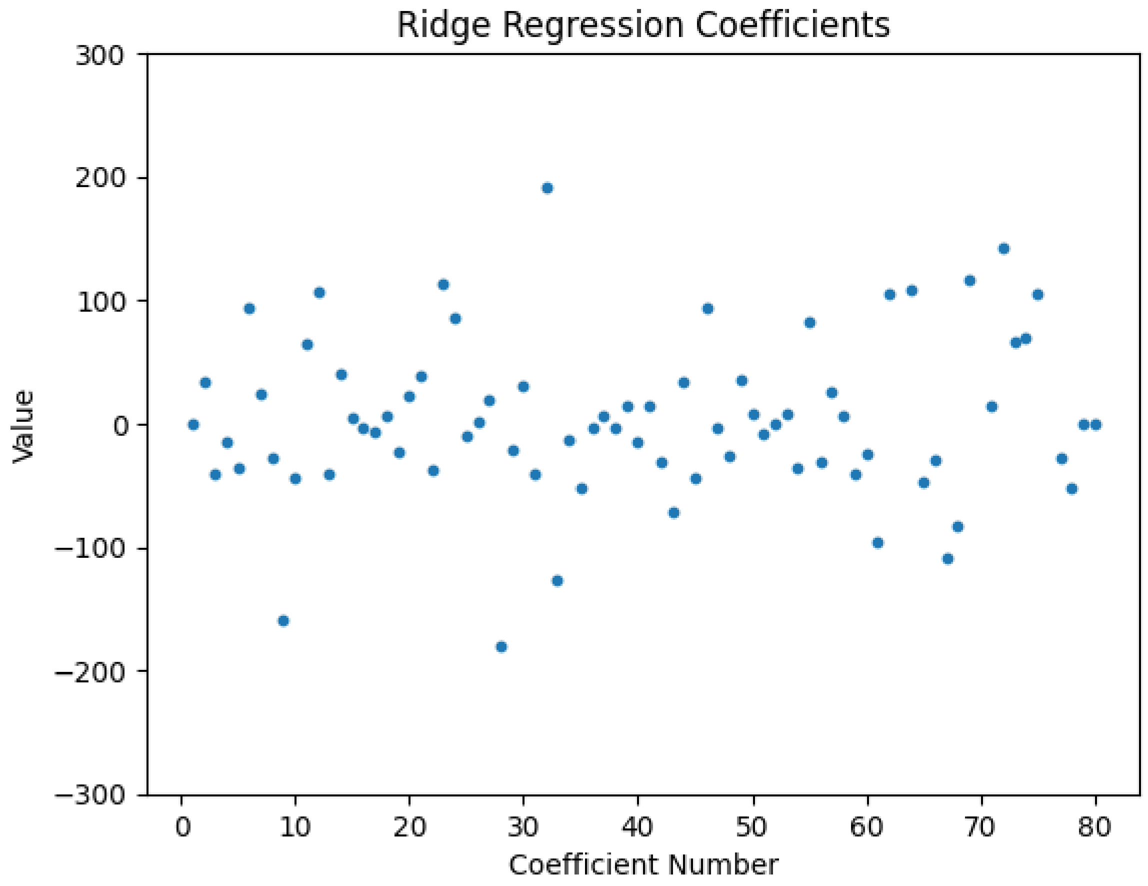
Regularization – Example: Immoscout24 Dataset



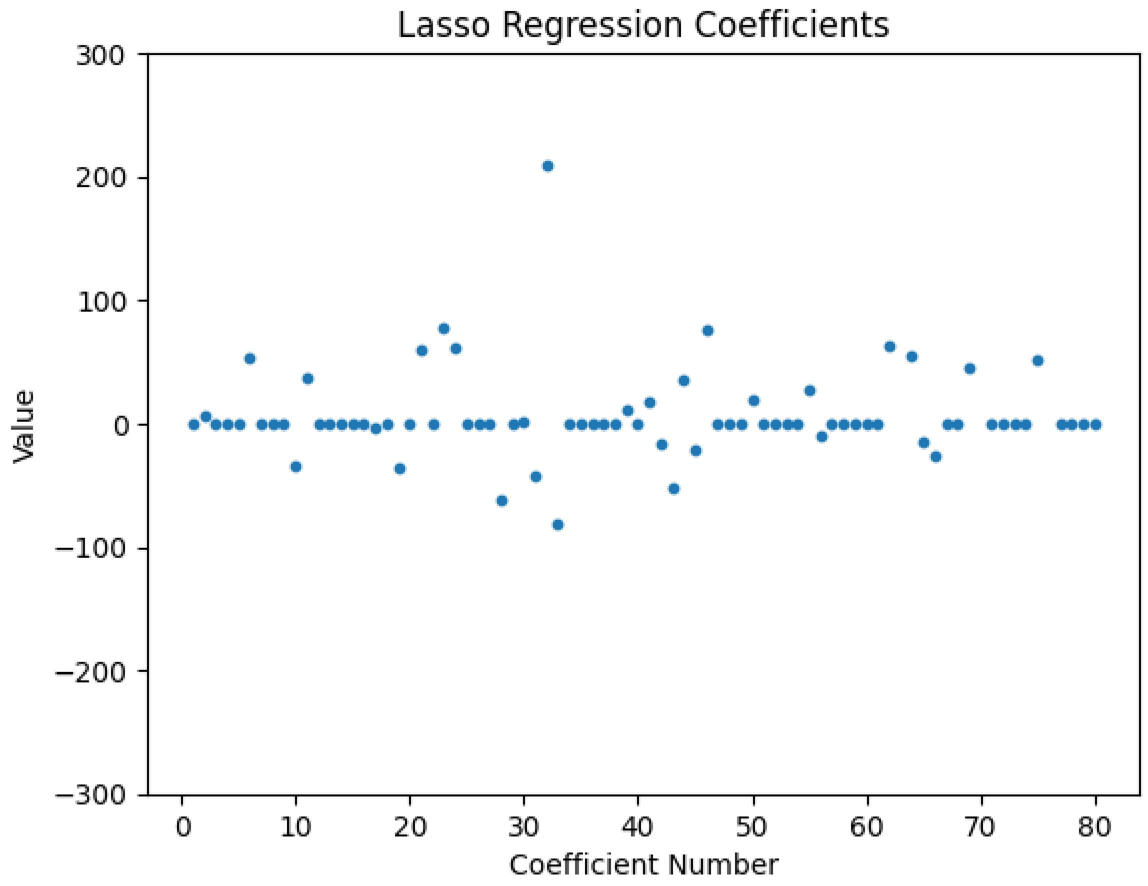
	R²	MSE	MAE
Train	0.77	35021.49	124.50
Test	-745267522613395658375168.00	Large Value	Large Val

Linear Regression

Regularization – Example: Immoscout24 Dataset



	R²	MSE	MAE
Train	0.76	35543.74	123.8
Test	0.72	37255.30	141.81



	R²	MSE	MAE
Train	0.75	37443.95	128.30
Test	0.72	37027.00	143.92

Linear Regression

Hyperparameter Tuning

+ What is a hyperparameter?
    - A **hyperparameter** is a parameter whose value is set before the training phase begins, and unlike model parameters, hyperparameters are not computed during training.
    - Examples: α is a hyperparameter of the regularization term of Lasso or Ridge regression and order of a regression polynomial
+ How to select a hyperparameter?
    - The selected hyperparameter should enhance the model performance. The objective is to find the best α that ensures a minimum mean squared error (good performance) on unseen data and avoid the overfitting problem. It is an optimization problem.
+ Optimization methods:
    - Grid Search Method
    - Random Search
    - Evolutionary Algorithms

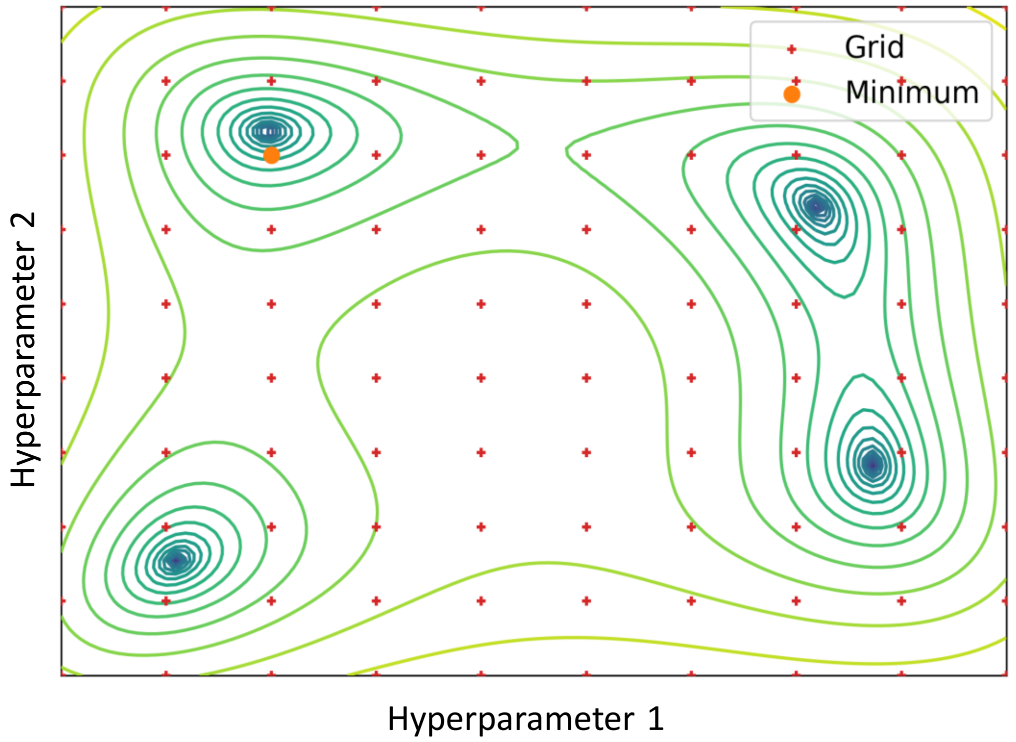
Linear Regression

Hyperparameter Tuning – Grid Search Method

- Grid Search method is similar to a simple search loop
  - Possible and reasonable values are defined for each hyperparameter
  - The result is a grid of hyperparameter combinations, which is called a grid.

- Each combination will be used to build a different model. How to evaluate these models?

- Could the test subset be used to evaluate these models to choose the best combination?
- The test subset should be unseen till the final evaluation before deploying the model in the production.
- Another subset that is not for training or for final testing is required. It is the validation subset.

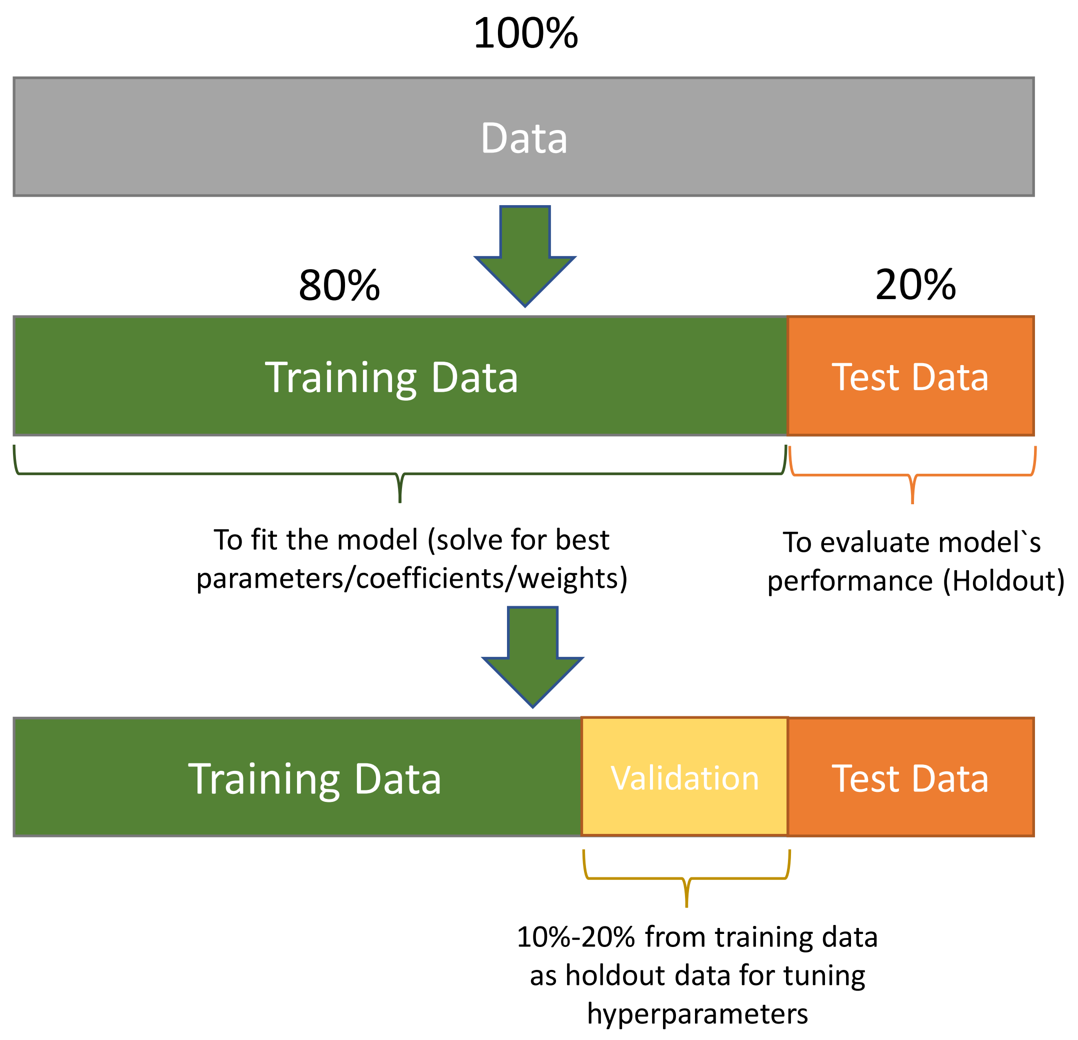


Linear Regression

Hyperparameter Tuning– Data Split Into Training, Validation and Testing Subsets

+ During Hyperparameters tuning, the test set should remain unseen
+ To validate the supposed models for various sets of the hyperparameters, the available training data is divided into two subsets: one for training the models and another for validating their performance.
+ The model does not learn from validation data. Its performance on the validation set is used to gauge how well it might perform on new, unseen data.
+ The validation set is between 10% to 20% of the training set depending on how large the dataset is.

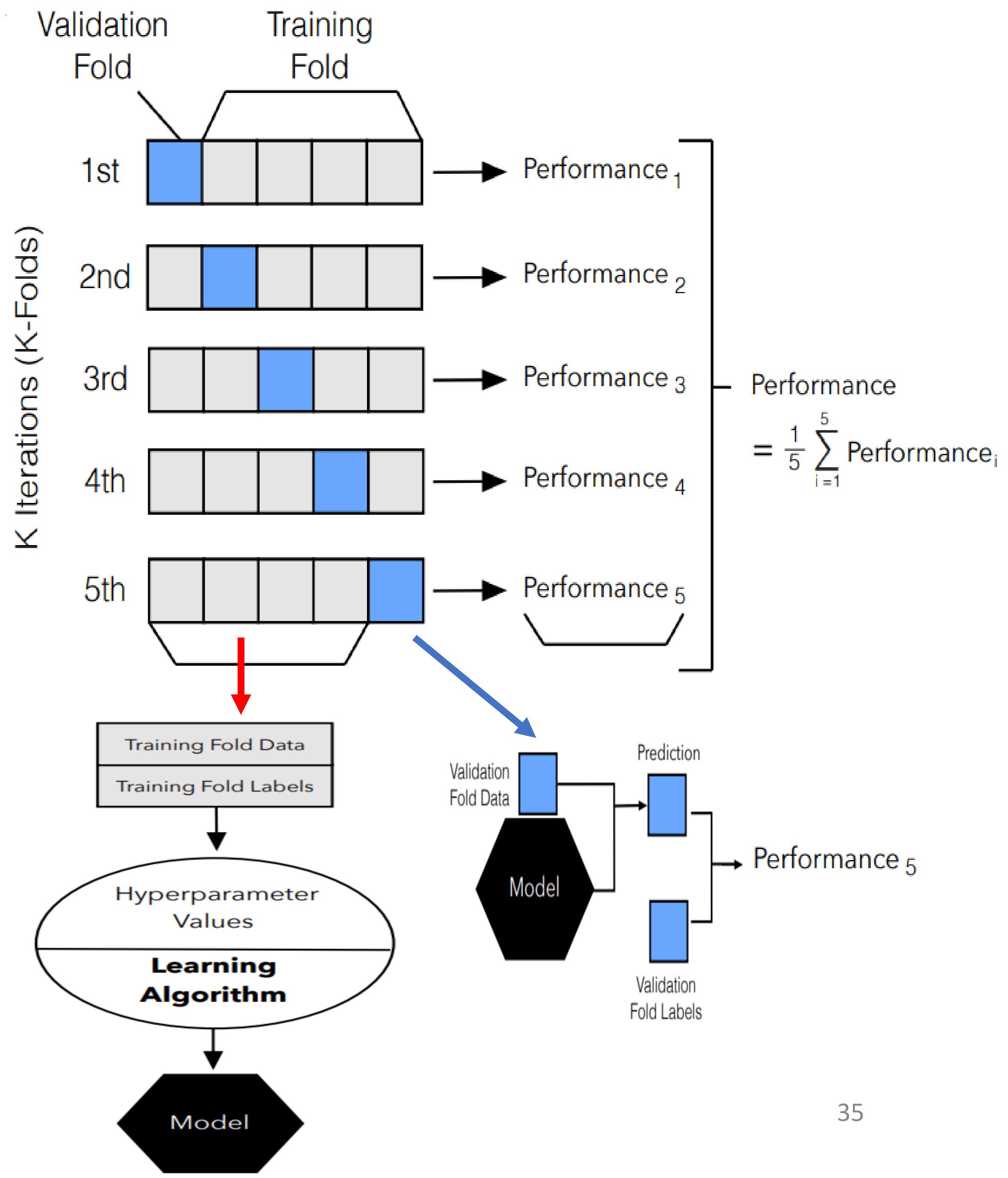
+ Is it enough to ensure that the model will perform good on unseen data (test subset or later in the production)?
+ What about if the validation subset does not represent the distribution of the whole dataset.
+ What about if we have a limited dataset.


100%
Data
80%
20%
Training Data
Test Data
To fit the model (solve for best parameters/coefficients/weights)
To evaluate model`s performance (Holdout)
Training Data
Validation
Test Data
10%-20% from training data as holdout data for tuning hyperparameters

Linear Regression

K-Fold Cross Validation in general

- It is most common technique for model evaluation and model selection in machine learning practice
- The main idea behind cross-validation is that each sample in our dataset has the opportunity of being tested. K-fold cross-validation is a special case of cross-validation where we iterate over a dataset k times.
- In each round, we split the dataset into k parts: one part is used for validation, and the remaining k – 1 parts are merged into a training subset
- Procedure is repeated K times, so that validation has taken place once with each part

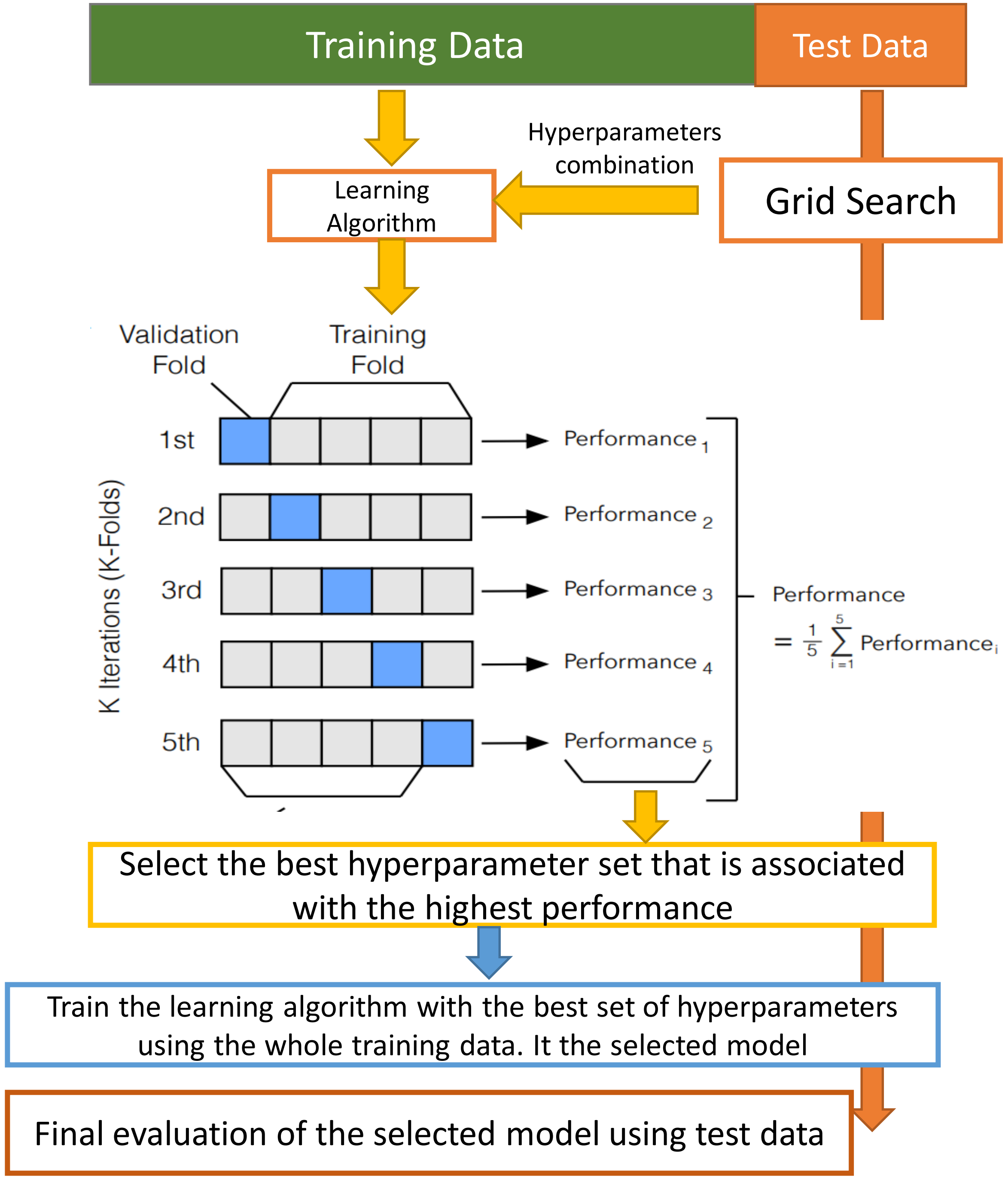


[Source: https://arxiv.org/pdf/1811.12808.pdf](https://arxiv.org/pdf/1811.12808.pdf)

Linear Regression

Grid Search Optimization with k-Fold Cross-Validation

+ Grid search combined with k-fold cross-validation is a robust method for hyperparameter tuning.
+ Define the grid with different hyperparameter values
+ For each combination of hyperparameters in the grid, apply k-fold cross-validation approach to validate its average performance
+ Selection: The combination of hyperparameters that gives the best average performance across the k-folds is selected as the best set
+ The selected model: Train the learning algorithm with the best set of hyperparameters using the whole training data and then evaluate it finally using the test data
+ Grid search combined with k-fold cross-validation is computationally expensive
+ Grid-search searches over a specified set of values, meaning it won't find a better solution if it lies between the grid points.



Training Data
Test Data
Hyperparameters combination
Learning Algorithm
Grid Search
Validation Fold
Training Fold
K Iterations (K-Folds)
1st
2nd
3rd
4th
5th
Performance $_1$
Performance $_2$
Performance $_3$
Performance $_4$
Performance $_5$
Performance $= \frac{1}{5}\sum_{i=1}^{5} \text{Performance}_i$
Select the best hyperparameter set that is associated with the highest performance
Train the learning algorithm with the best set of hyperparameters using the whole training data. It the selected model
Final evaluation of the selected model using test data

Source: https://arxiv.org/pdf/1811.12808.pdf

Hochschule Karlsruhe
University of
Applied Sciences

Fakultät für
Elektro- und
Informationstechnik

www.h-ka.de