Error-based Learning

1. Fundamentals
   - Simple Linear Regression
   - Measuring Error
2. Standard Approach: Multivariate Linear Regression
   - Gradient Descent
   - Learning Rates and Initialization
3. Interpreting Multivariable Linear Regression Models
4. Handling Categorical Descriptive Features and Categorical Target Features
5. Other Regression Models

Fundamentals

- A parameterized prediction model is initialized with random parameters, and an error function evaluates its initial performance on a training dataset.
- Parameters are iteratively adjusted based on the error function to improve model accuracy.

Simple Linear Regression

Table: The **office rentals dataset**: a dataset that includes office rental prices and a number of descriptive features for 10 Dublin city-centre offices.

ID	Size	Floor	Broadband Rate	Energy Rating	Rental Price
1	500	4	8	C	320
2	550	7	50	A	380
3	620	9	7	A	400
4	630	5	24	B	390
5	665	8	100	C	385
6	700	4	8	B	410
7	770	10	7	B	480
8	880	12	50	A	600
9	920	14	8	C	570
10	1,000	9	24	B	620

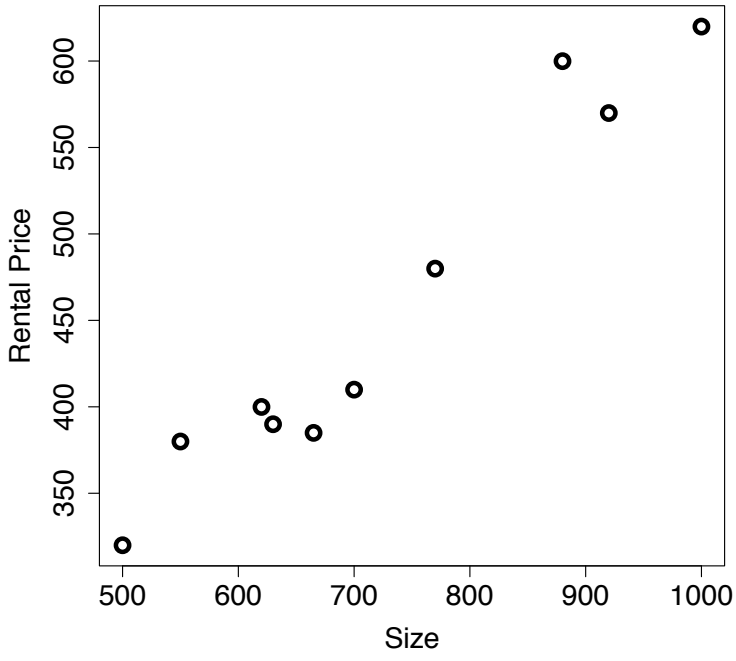


Figure: Dataset

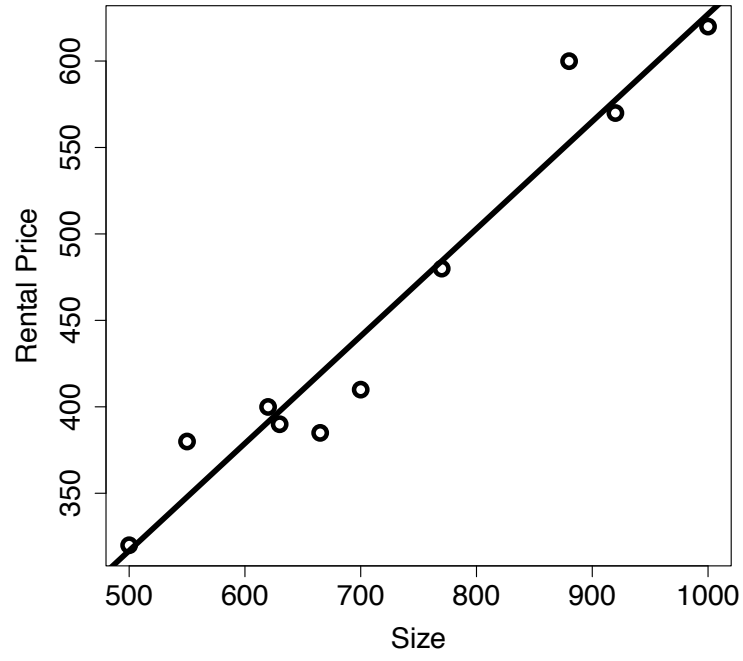


Figure: Final Model

- From the scatter plot it appears that there is a linear relationship between the SIZE and RENTAL PRICE.
- The equation of a line can be written as:

$$y = mx + b \tag{1}$$

Univariate Linear Regression

$$\mathbb{M}_{\mathbf{w}}(d) = \mathbf{w}[0] + \mathbf{w}[1] \times \mathbf{d}[1] \tag{2}$$

$$\text{RENTAL PRICE} = 6.47 + 0.62 \times \text{SIZE}$$

- Using this model determine the expected rental price of the 730 square foot office:

$$\begin{aligned} \text{RENTAL PRICE} &= 6.47 + 0.62 \times 730 \\ &= 459.07 \end{aligned}$$

Measuring Error

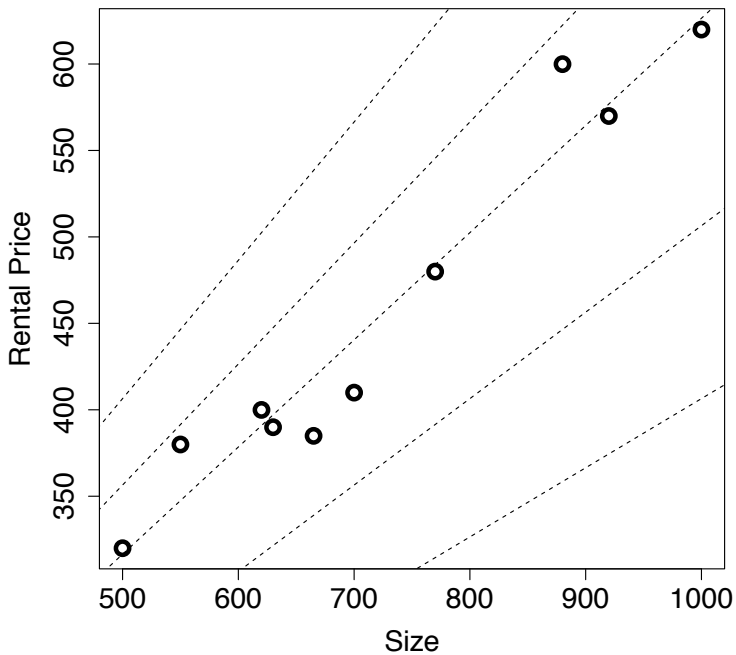


Figure: Different Models

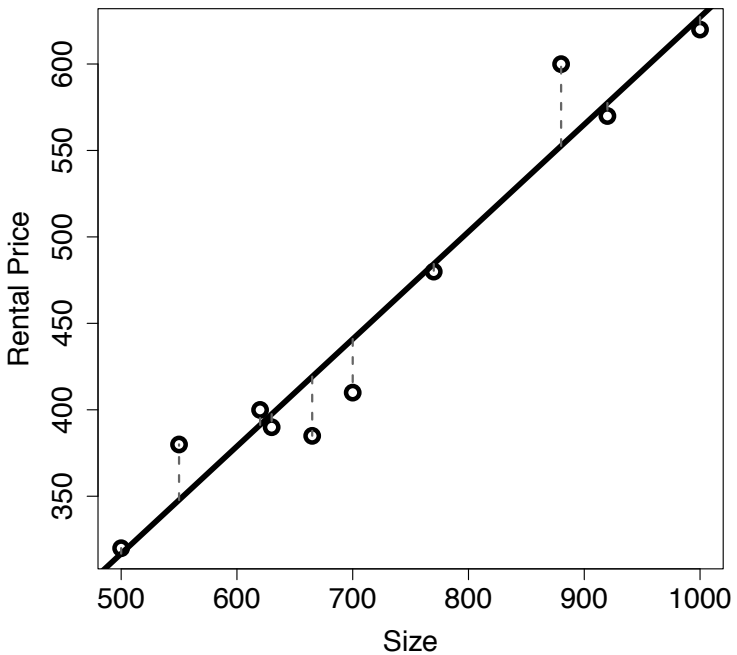


Figure: Resulting Errors

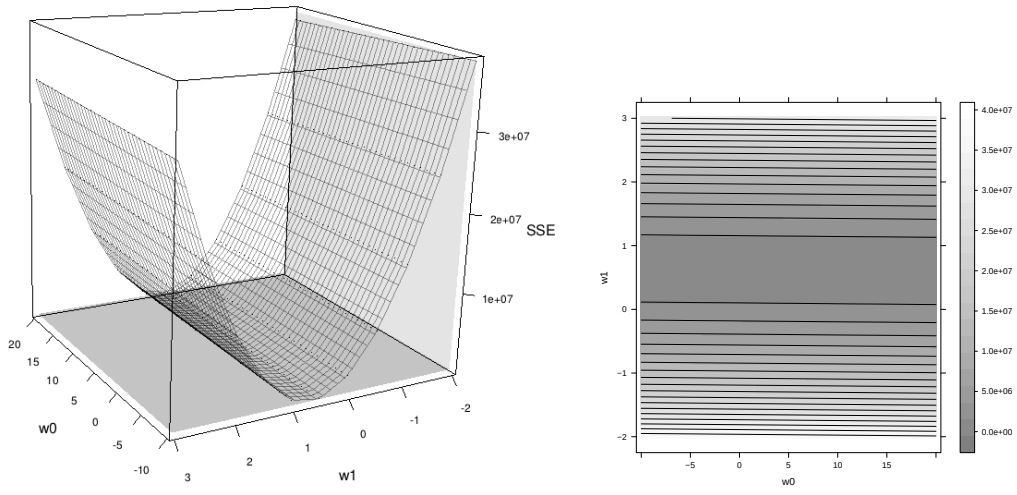
$$L_2(\mathbb{M}_{\mathbf{w}}, \mathcal{D}) = \frac{1}{2}\sum_{i=1}^{n}(t_i - \mathbb{M}_{\mathbf{w}}(\mathbf{d}_i[1]))^2 \quad (3)$$

$$= \frac{1}{2}\sum_{i=1}^{n}(t_i - (\mathbf{w}[0] + \mathbf{w}[1] \times \mathbf{d}_i[1]))^2 \quad (4)$$

ID	Rental Price	Model Prediction	Error Error	Squared Error
1	320	316.79	3.21	10.32
2	380	347.82	32.18	1,035.62
3	400	391.26	8.74	76.32
4	390	397.47	-7.47	55.80
5	385	419.19	-34.19	1,169.13
6	410	440.91	-30.91	955.73
7	480	484.36	-4.36	19.01
8	600	552.63	47.37	2,243.90
9	570	577.46	-7.46	55.59
10	620	627.11	-7.11	50.51
			Sum	**5,671.64**
	Sum of squared errors (Sum/2)			**2,835.82**

Error Surfaces

- For every combination of **w**[0] and **w**[1], there corresponds a sum of squared errors value, forming a surface.
- The *x*-*y* plane is called **weight space**, and the surface is the **error surface**.
- The best-fitting model corresponds to the lowest point on the error surface.



(a) (b)

Figure: (a) A 3D surface plot and (b) a contour plot of the error surface, showing sum of squared errors for each combination of **w**[0] and **w**[1]

- We formally define this point on the error surface:

$$\frac{\partial}{\partial \mathbf{w}[0]} \frac{1}{2} \sum_{i=1}^{n} (t_i - (\mathbf{w}[0] + \mathbf{w}[1] \times \mathbf{d}_i[1]))^2 = 0 \tag{5}$$

and

$$\frac{\partial}{\partial \mathbf{w}[1]} \frac{1}{2} \sum_{i=1}^{n} (t_i - (\mathbf{w}[0] + \mathbf{w}[1] \times \mathbf{d}_i[1]))^2 = 0 \tag{6}$$

- There are a number of different ways to find this point.
- We will describe a **guided search** approach known as the **gradient descent** algorithm.

Standard Approach: Multivariate Linear Regression with Gradient Descent

ID	Size	Floor	Broadband Rate	Energy Rating	Rental Price
1	500	4	8	C	320
2	550	7	50	A	380
3	620	9	7	A	400
4	630	5	24	B	390
5	665	8	100	C	385
6	700	4	8	B	410
7	770	10	7	B	480
8	880	12	50	A	600
9	920	14	8	C	570
10	1,000	9	24	B	620

- We can define a multivariate linear regression model as:

$$\mathbb{M}_{\mathbf{w}}(\mathbf{d}) = \mathbf{w}[0] + \mathbf{w}[1] \times \mathbf{d}[1] + \cdots + \mathbf{w}[m] \times \mathbf{d}[m] \quad (7)$$

$$= \mathbf{w}[0] + \sum_{j=1}^{m} \mathbf{w}[j] \times \mathbf{d}[j] \quad (8)$$

- Inventing a dummy variable $\mathbf{d}[0]$ always equals to 1:

$$\begin{aligned}\mathbb{M}_{\mathbf{w}}(\mathbf{d}) &= \mathbf{w}[0] \times \mathbf{d}[0] + \mathbf{w}[1] \times \mathbf{d}[1] + \ldots + \mathbf{w}[m] \times \mathbf{d}[m] && (9)\\ &= \sum_{j=0}^{m} \mathbf{w}[j] \times \mathbf{d}[j] && (10)\\ &= \mathbf{w} \cdot \mathbf{d} && (11)\end{aligned}$$

- The sum of squared errors loss function L_2:

$$\begin{aligned}L_2(\mathbb{M}_{\mathbf{w}}, \mathcal{D}) &= \frac{1}{2}\sum_{i=1}^{n}(t_i - \mathbb{M}_{\mathbf{w}}(\mathbf{d}_i))^2 && (12)\\ &= \frac{1}{2}\sum_{i=1}^{n}(t_i - (\mathbf{w} \cdot \mathbf{d}_i))^2 && (13)\end{aligned}$$

- The resulting multivariate model equation:

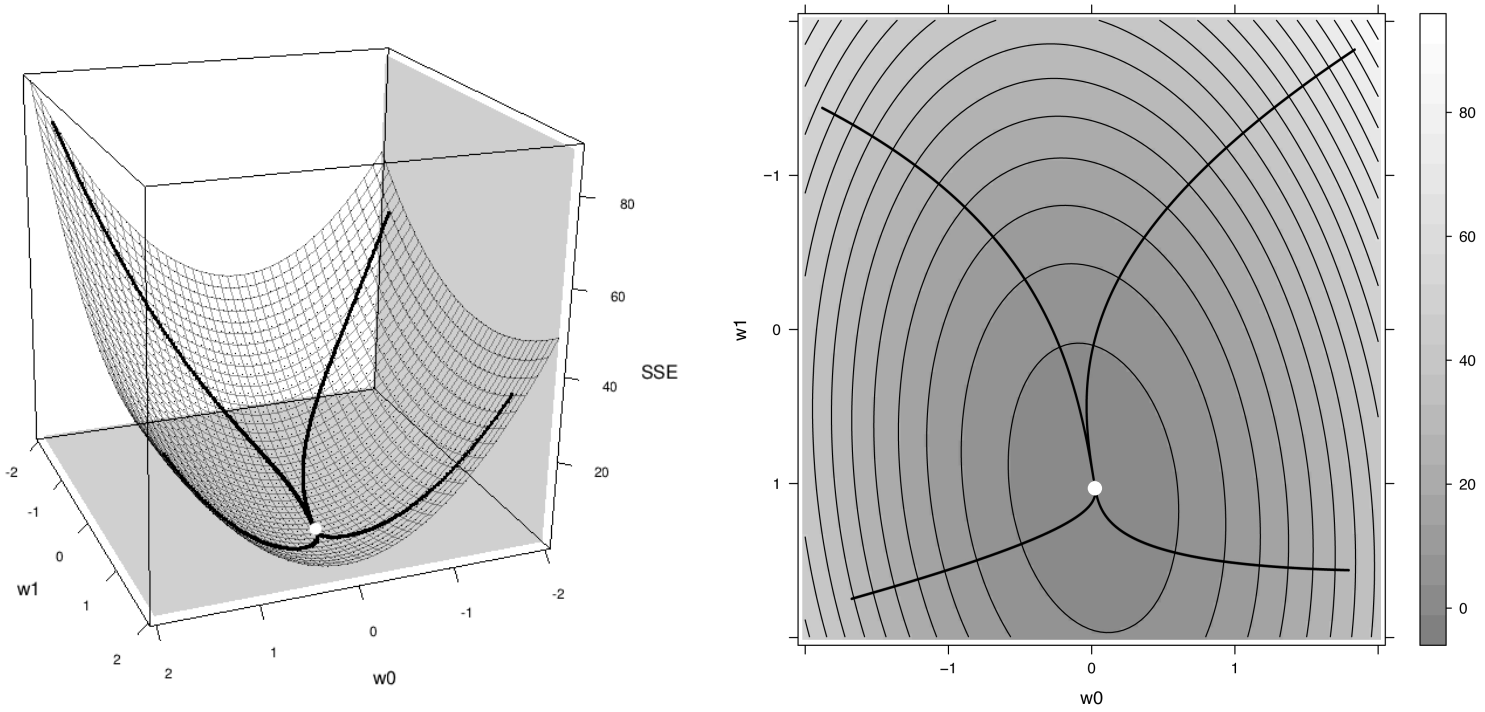
$$\begin{aligned}\textsc{Rental Price} = \mathbf{w}[0] &+ \mathbf{w}[1] \times \textsc{Size} + \mathbf{w}[2] \times \textsc{Floor}\\ &+ \mathbf{w}[3] \times \textsc{Broadband Rate}\end{aligned}$$

- Assume we know
  - $\mathbf{w}[0] = -0.1513$,
  - $\mathbf{w}[1] = 0.6270$,
  - $\mathbf{w}[2] = -0.1781$,
  - $\mathbf{w}[3] = 0.0714$.
- Using this model to predict the expected rental price of a 690 square foot office on the 11^{th} floor of a building with a broadband rate of 50 Mb per second as::

$$\begin{aligned}\text{RENTAL PRICE} = -0.1513 &+ 0.6270 \times \text{SIZE} \\ &- 0.1781 \times \text{FLOOR} \\ &+ 0.0714 \times \text{BROADBAND RATE}\end{aligned}$$

$$\begin{aligned}\text{RENTAL PRICE} &= -0.1513 + 0.6270 \times 690 \\ &\quad -0.1781 \times 11 + 0.0714 \times 50 \\ &= 434.0896\end{aligned}$$

Gradient Descent



(a) (b)

Figure: (a) A 3D surface plot and (b) a contour plot of the same error surface. The lines indicate the path that the gradient decent algorithm would take across this error surface from different starting positions to the global minimum - marked as the white dot in the centre.

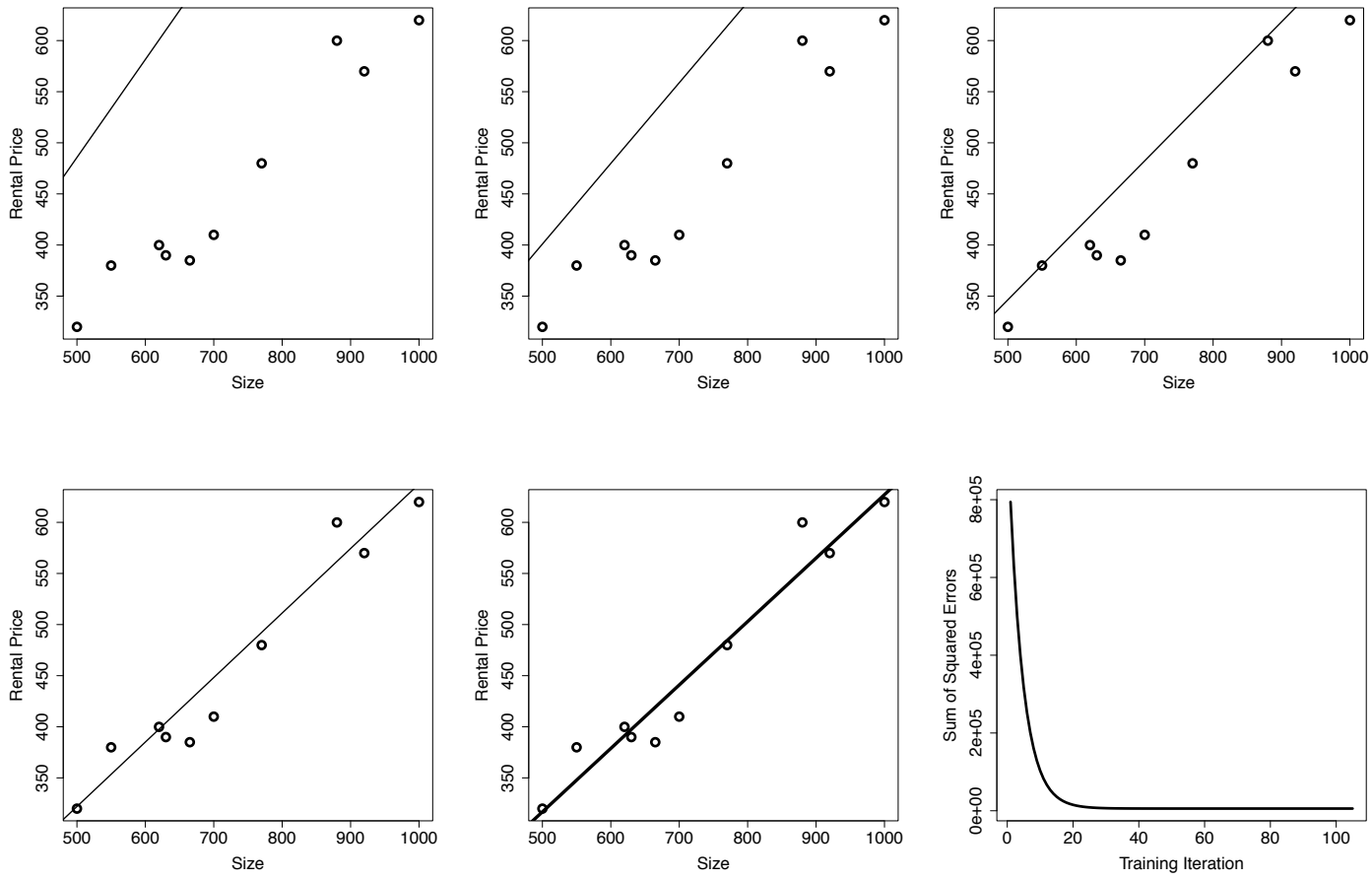


Figure: A selection of the simple linear regression models developed during the gradient descent process for the office rentals dataset. The final panel shows the sum of squared error values generated during the gradient descent process.

Require: training set $\mathcal{D}$
Require: learning rate α for convergence speed
Require: function **errorDelta** to adjust weight **w**[j] down the error surface slope
Require: convergence criterion for algorithm completion

1: **w** $\leftarrow$ random starting point in the weight space
2: **repeat**
3: **for** each **w**[j] in **w** **do**
4: **w**[j] $\leftarrow$ **w**[j] $- \alpha \times$ **errorDelta**($\mathcal{D}$, **w**[j])
5: **end for**
6: **until** convergence occurs

- Gradient descent algorithm for training multivariate linear regression models.
- Each weight **w**[j] is adjusted independently by adding a small **delta**.
- Adjustments ensure movement *downwards* on the error surface.

- Consider $\mathcal{D}$ has just one training example: $(\mathbf{d}, t)$.
- The gradient is the partial derivative of L_2 with respect to each weight, $\mathbf{w}[j]$:

$$\frac{\partial}{\partial \mathbf{w}[j]} L_2\left(\mathbb{M}_{\mathbf{w}}, \mathcal{D}\right) = \frac{\partial}{\partial \mathbf{w}[j]} \left(\frac{1}{2} \left(t - \mathbb{M}_{\mathbf{w}}\left(\mathbf{d}\right)\right)^2 \right) \tag{14}$$

$$= \left(t - \mathbb{M}_{\mathbf{w}}(\mathbf{d})\right) \times \frac{\partial}{\partial \mathbf{w}[j]} \left(t - \mathbb{M}_{\mathbf{w}}(\mathbf{d})\right) \tag{15}$$

$$= \left(t - \mathbb{M}_{\mathbf{w}}(\mathbf{d})\right) \times \frac{\partial}{\partial \mathbf{w}[j]} \left(t - (\mathbf{w} \cdot \mathbf{d})\right) \tag{16}$$

$$= \left(t - \mathbb{M}_{\mathbf{w}}(\mathbf{d})\right) \times -\mathbf{d}[j] \tag{17}$$

- Adjusting the calculation to take into account multiple training instances:

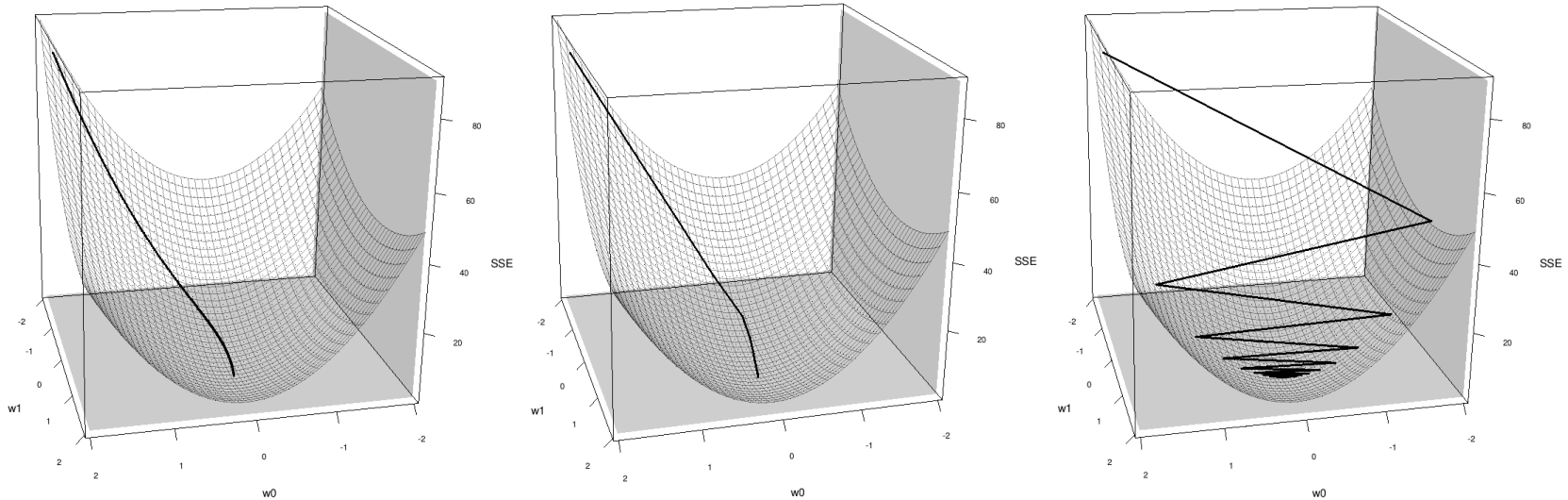
$$\frac{\partial}{\partial \mathbf{w}[j]} L_2(\mathbb{M}_{\mathbf{w}}, \mathcal{D}) = -\sum_{i=1}^{n} ((t_i - \mathbb{M}_{\mathbf{w}}(\mathbf{d}_i)) \times \mathbf{d}_i[j])$$

- We use this equation to define the **errorDelta** in our gradient descent algorithm.

$$\mathbf{w}[j] \leftarrow \mathbf{w}[j] + \alpha \underbrace{\sum_{i=1}^{n} ((t_i - \mathbb{M}_{\mathbf{w}}(\mathbf{d}_i)) \times d_i[j])}_{errorDelta(\mathcal{D},\mathbf{w}[j])}$$

- The learning rate, α, sets the adjustment size for each weight per step.
- Practitioners typically rely on heuristics and experimentation.

Learning Rates & Initial Weights



(a) (b) (c)

Plots of error surface journeys for different learning rates in the office rentals prediction problem: (a) small (0.002), (b) medium (0.08), (c) large (0.18).

- Typical learning rates range from $[0.00001, 10]$. Random initial weights from $[-0.2, 0.2]$ often perform well.
- **Learning rate decay** starts high and reduces over time based on a preset schedule $\alpha_\tau = \alpha_0 \frac{c}{c+\tau}$

$$\text{Rental Price} = \mathbf{w}[0] + \mathbf{w}[1] \times \text{Size} + \mathbf{w}[2] \times \text{Floor} + \mathbf{w}[3] \times \text{Broadband Rate}$$

ID	Size	Floor	Broadband Rate	Energy Rating	Rental Price
1	500	4	8	C	320
2	550	7	50	A	380
3	620	9	7	A	400
4	630	5	24	B	390
5	665	8	100	C	385
6	700	4	8	B	410
7	770	10	7	B	480
8	880	12	50	A	600
9	920	14	8	C	570
10	1,000	9	24	B	620

- $\alpha = 0.00000002$

Initial Weights

$\mathbf{w}[0]$:	-0.146	$\mathbf{w}[1]$:	0.185	$\mathbf{w}[2]$:	-0.044	$\mathbf{w}[3]$:	0.119

Iteration 1

ID	RENTAL PRICE	Pred.	Error	Squared Error	**errorDelta**($\mathcal{D}$, **w**[**i**]) **w**[0]	**w**[1]	**w**[2]	**w**[3]
1	320	93.26	226.74	51411.08	226.74	113370.05	906.96	1813.92
2	380	107.41	272.59	74307.70	272.59	149926.92	1908.16	13629.72
3	400	115.15	284.85	81138.96	284.85	176606.39	2563.64	1993.94
4	390	119.21	270.79	73327.67	270.79	170598.22	1353.95	6498.98
5	385	134.64	250.36	62682.22	250.36	166492.17	2002.91	25036.42
6	410	130.31	279.69	78226.32	279.69	195782.78	1118.76	2237.52
7	480	142.89	337.11	113639.88	337.11	259570.96	3371.05	2359.74
8	600	168.32	431.68	186348.45	431.68	379879.24	5180.17	21584.05
9	570	170.63	399.37	159499.37	399.37	367423.83	5591.23	3194.99
10	620	187.58	432.42	186989.95	432.42	432423.35	3891.81	10378.16
			Sum	1067571.59	3185.61	2412073.90	27888.65	88727.43
Sum of squared errors (Sum/2)				533785.80				

$$\mathbf{w}[j] \leftarrow \mathbf{w}[j] + \alpha \underbrace{\sum_{i=1}^{n} ((t_i - \mathbb{M}_{\mathbf{w}}(\mathbf{d}_i)) \times d_i[j])}_{errorDelta(\mathcal{D},\mathbf{w}[j])}$$

Initial Weights

w[0]:	-0.146	**w**[1]:	0.185	**w**[2]:	-0.044	**w**[3]:	0.119

$$\mathbf{w}[1] \leftarrow 0.185 + 0.00000002 \times 2,412,074 = 0.23324148$$

New Weights (Iteration 1)

w[0]:	-0.146	**w**[1]:	0.233	**w**[2]:	-0.043	**w**[3]:	0.121

Iteration 2

ID	Rental Price	Pred.	Error	Squared Error	**errorDelta**($\mathcal{D}$, **w**[**i**]) **w**[0]	**w**[1]	**w**[2]	**w**[3]
1	320	117.40	202.60	41047.92	202.60	101301.44	810.41	1620.82
2	380	134.03	245.97	60500.69	245.97	135282.89	1721.78	12298.44
3	400	145.08	254.92	64985.12	254.92	158051.51	2294.30	1784.45
4	390	149.65	240.35	57769.68	240.35	151422.55	1201.77	5768.48
5	385	166.90	218.10	47568.31	218.10	145037.57	1744.81	21810.16
6	410	164.10	245.90	60468.86	245.90	172132.91	983.62	1967.23
7	480	180.06	299.94	89964.69	299.94	230954.68	2999.41	2099.59
8	600	210.87	389.13	151424.47	389.13	342437.01	4669.60	19456.65
9	570	215.03	354.97	126003.34	354.97	326571.94	4969.57	2839.76
10	620	187.58	432.42	186989.95	432.42	432423.35	3891.81	10378.16
			Sum	886723.04	2884.32	2195615.84	25287.08	80023.74
Sum of squared errors (Sum/2)				443361.52				

$$\mathbf{w}[j] \leftarrow \mathbf{w}[j] + \alpha \underbrace{\sum_{i=1}^{n} ((t_i - \mathbb{M}_{\mathbf{w}}(\mathbf{d}_i)) \times d_i[j])}_{errorDelta(\mathcal{D},\mathbf{w}[j])}$$

Initial Weights (Iteration 2)

w[0]:	-0.146	**w**[1]:	0.233	**w**[2]:	-0.043	**w**[3]:	0.121

$$\mathbf{w}[1] \leftarrow -0.233 + 0.00000002 \times 2195616.08 = 0.27691232$$

New Weights (Iteration 2)

w[0]:	-0.145	**w**[1]:	0.277	**w**[2]:	-0.043	**w**[3]:	0.123

- The algorithm iteratively updates weights until converging to a stable set where model accuracy hardly improves.
- After 100 iterations, the final weight values are:
  - $\mathbf{w}[0] = -0.1513$,
  - $\mathbf{w}[1] = 0.6270$,
  - $\mathbf{w}[2] = -0.1781$
  - $\mathbf{w}[3] = 0.0714$
- which results in a sum of squared errors value of $2,913.5$

Interpreting Multivariable Linear Regression Models

- Linear regression weights, through **sign** and **magnitude**, show each feature's impact on model predictions.
- Weights imply feature importance, but direct comparisons can mislead.
- **Statistical significance tests**, like the ***t*-test**, better assess feature importance.
- The *t*-test checks if feature $\mathbf{d}[j]$ significantly impacts the model, using the *t*-statistic under a null hypothesis of no effect.

Table: Weights and standard errors for each feature in the office rentals model.

Descriptive Feature	Weight	Standard Error	*t*-statistic	*p*-value
SIZE	0.6270	0.0545	11.504	$<$0.0001
FLOOR	-0.1781	2.7042	-0.066	0.949
BROADBAND RATE	0.071396	0.2969	0.240	0.816

Hypothesis Testing: H_0 and H_1

- **Null Hypothesis (H_0)**:
  - Represents the default or no-effect scenario.
  - Assumes no relationship or no difference between groups.
  - Example: H_0: $\mu = \mu_0$ (the population mean is equal to a specified value).

- **Alternative Hypothesis (H_1 or H_A)**:
  - Represents the effect or difference we aim to detect.
  - Contradicts the null hypothesis.
  - Example: H_1: $\mu \neq \mu_0$ (the population mean is not equal to a specified value).

Decision Rule

- If the test statistic falls within the critical region, reject H_0.
- If the test statistic does not fall within the critical region, fail to reject H_0.

- The standard error for the overall model

$$se = \sqrt{\frac{\sum_{i=1}^{n} (t_i - \mathbb{M}_{\mathbf{w}}(\mathbf{d}_i))^2}{n-2}} \tag{18}$$

- A standard error calculation for a descriptive feature

$$se(\mathbf{d}[j]) = \frac{se}{\sqrt{\sum_{i=1}^{n} \left(\mathbf{d}_i[j] - \overline{\mathbf{d}[j]}\right)^2}} \tag{19}$$

- The t-statistic for this test is calculated as follows:

$$t = \frac{\mathbf{w}[j]}{se(\mathbf{d}[j])} \tag{20}$$

- Using a t-statistic table, we find the p-value for a two-tailed t-test. If the p-value is below 0.05, we reject the null hypothesis, indicating the feature significantly impacts the model; otherwise, it does not.

Handling Categorical Descriptive Features

- Categorical features are transformed into continuous values through one-hot encoding, turning each level of a categorical feature into a separate continuous feature.
- ENERGY RATING is split into three features.

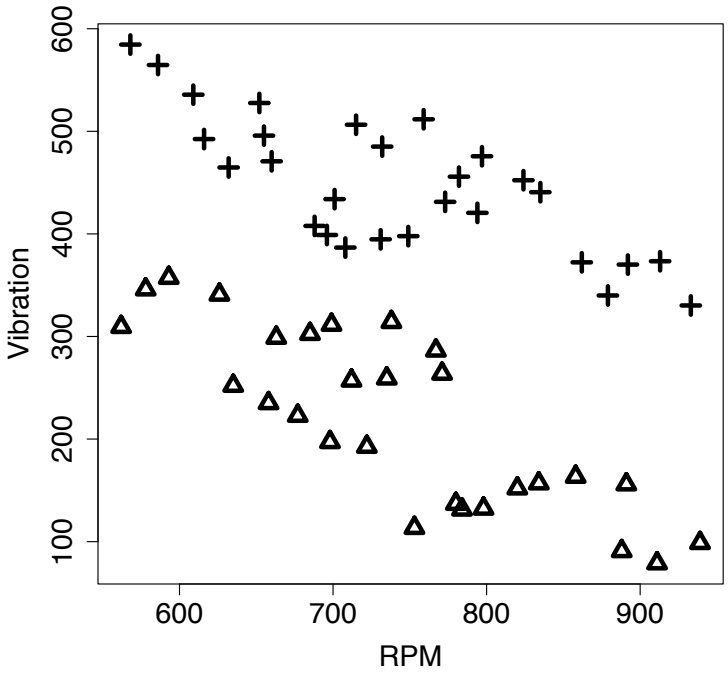
ID	SIZE	FLOOR	BROADBAND RATE	ENERGY RATING A	ENERGY RATING B	ENERGY RATING C	RENTAL PRICE
1	500	4	8	0	0	1	320
2	550	7	50	1	0	0	380
3	620	9	7	1	0	0	400
4	630	5	24	0	1	0	390
5	665	8	100	0	0	1	385
6	700	4	8	0	1	0	410
7	770	10	7	0	1	0	480
8	880	12	50	1	0	0	600
9	920	14	8	0	0	1	570
10	1 000	9	24	0	1	0	620

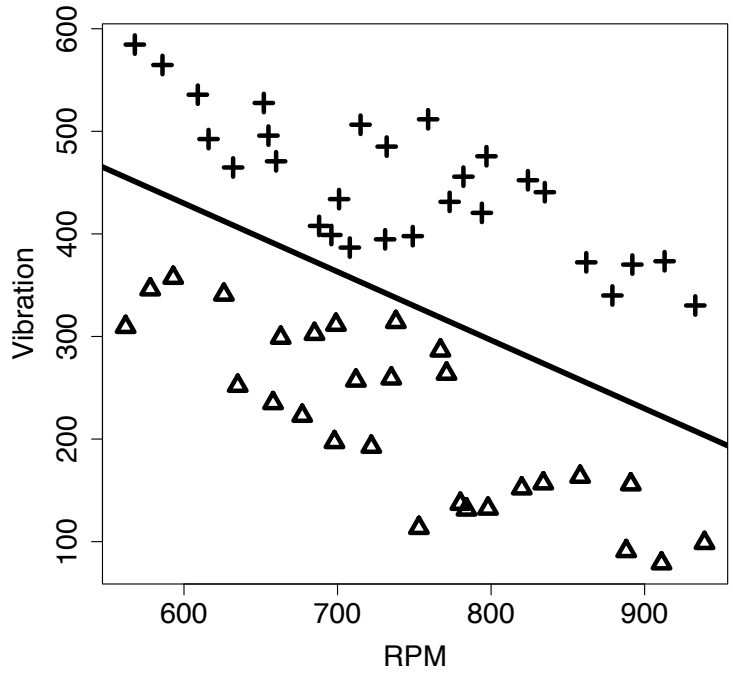
$$\begin{aligned} \text{RENTAL PRICE} = \mathbf{w}[0] &+ \mathbf{w}[1] \times \text{SIZE} + \mathbf{w}[2] \times \text{FLOOR} \\ &+ \mathbf{w}[3] \times \text{BROADBAND RATE} \\ &+ \mathbf{w}[4] \times \text{ENERGY RATING A} \\ &+ \mathbf{w}[5] \times \text{ENERGY RATING B} \\ &+ \mathbf{w}[6] \times \text{ENERGY RATING C} \end{aligned}$$

Handling Categorical Target Features: Logistic Regression

Table: A dataset listing features for a number of generators.

ID	RPM	VIBRATION	STATUS	ID	RPM	VIBRATION	STATUS
1	568	585	good	29	562	309	faulty
2	586	565	good	30	578	346	faulty
3	609	536	good	31	593	357	faulty
4	616	492	good	32	626	341	faulty
5	632	465	good	33	635	252	faulty
6	652	528	good	34	658	235	faulty
7	655	496	good	35	663	299	faulty
8	660	471	good	36	677	223	faulty
9	688	408	good	37	685	303	faulty
10	696	399	good	38	698	197	faulty
11	708	387	good	39	699	311	faulty
12	701	434	good	40	712	257	faulty
13	715	506	good	41	722	193	faulty
14	732	485	good	42	735	259	faulty
15	731	395	good	43	738	314	faulty
16	749	398	good	44	753	113	faulty
17	759	512	good	45	767	286	faulty
18	773	431	good	46	771	264	faulty
19	782	456	good	47	780	137	faulty
20	797	476	good	48	784	131	faulty
21	794	421	good	49	798	132	faulty
22	824	452	good	50	820	152	faulty
23	835	441	good	51	834	157	faulty
24	862	372	good	52	858	163	faulty
25	879	340	good	53	888	91	faulty
26	892	370	good	54	891	156	faulty
27	913	373	good	55	911	79	faulty
28	933	330	good	56	939	99	faulty





- As the boundary is a **linear separator**, it can be defined by:

$$\text{VIBRATION} = 830 - 0.667 \times \text{RPM} \tag{21}$$

or

$$830 - 0.667 \times \text{RPM} - \text{VIBRATION} = 0 \tag{22}$$

- For the instance RPM $= 810$, VIBRATION $= 495$, the result is above the decision boundary:

$$830 - 0.667 \times 810 - 495 = -205.27$$

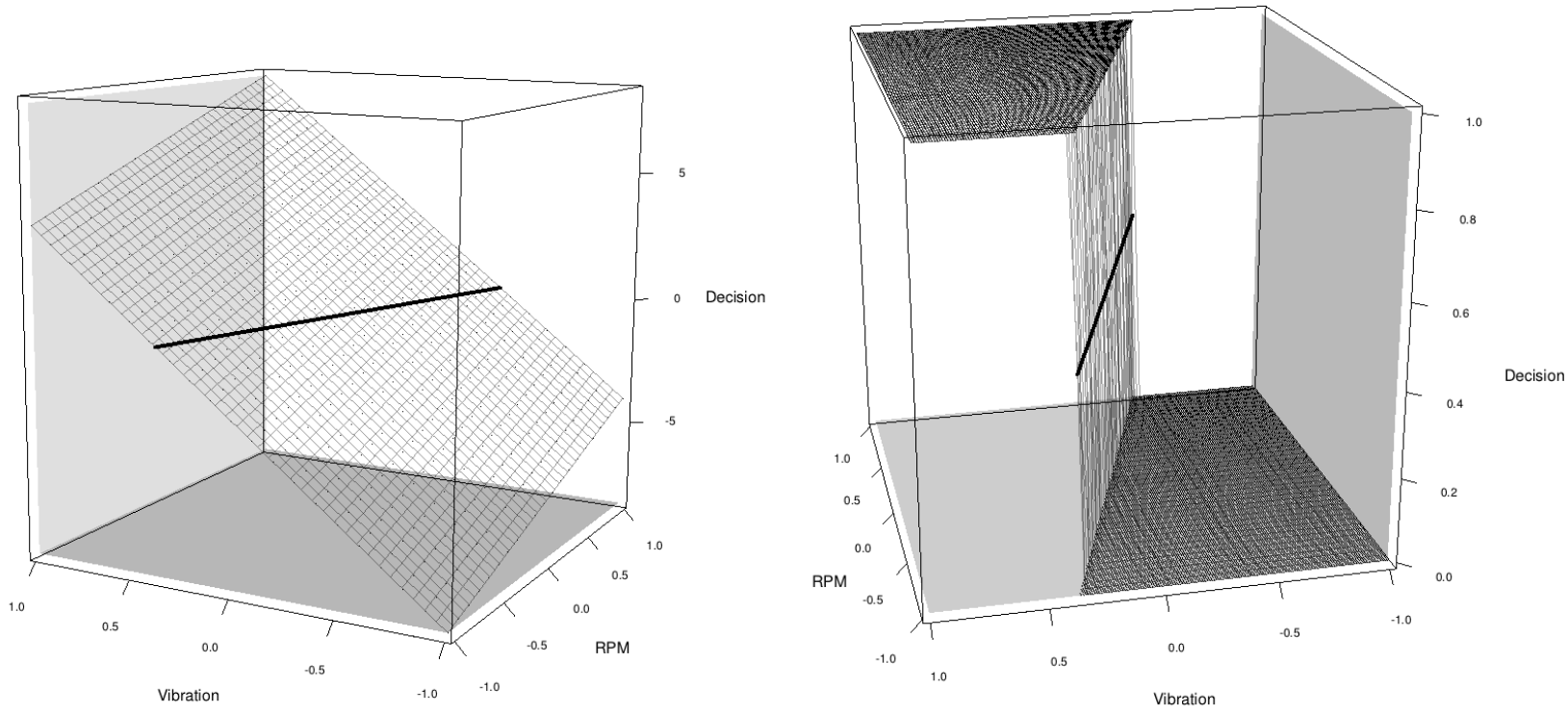
- For the instance RPM $= 650$ and VIBRATION $= 240$, the result is below the decision boundary:

$$830 - 0.667 \times 650 - 240 = 156.45$$

- Data points above the decision boundary yield a negative value, while those below yield a positive value.
- Then we have:

$$\mathbb{M}_{\mathbf{w}}(\mathbf{d}) = \begin{cases} 1 & \text{if } \mathbf{w} \cdot \mathbf{d} \geq 0 \\ 0 & \textit{otherwise} \end{cases} \tag{23}$$

- The surface is known as a **decision surface**.



(d) (e)

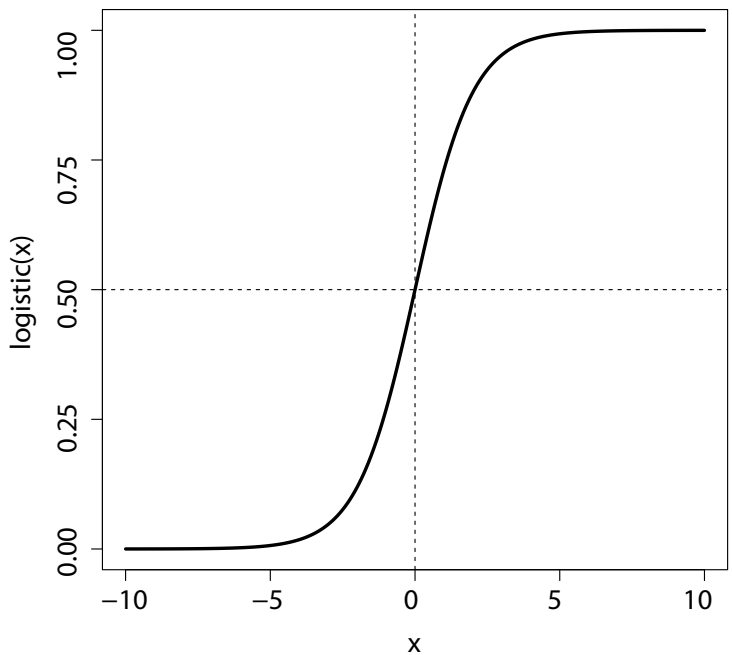
Figure: (d) A surface showing the predictive value for RPM and VIBRATION. (d) The same surface linearly thresholded at zero to operate as a predictor.

- The hard decision boundary is **discontinuous**, making it non-differentiable, so we can't calculate the error gradient.
- Additionally, the model always predicts 0 or 1 with complete confidence, lacking subtlety.
- We address these with the **logistic function**.

logistic function

$$Logistic(x) = \frac{1}{1 + e^{-x}} \tag{24}$$

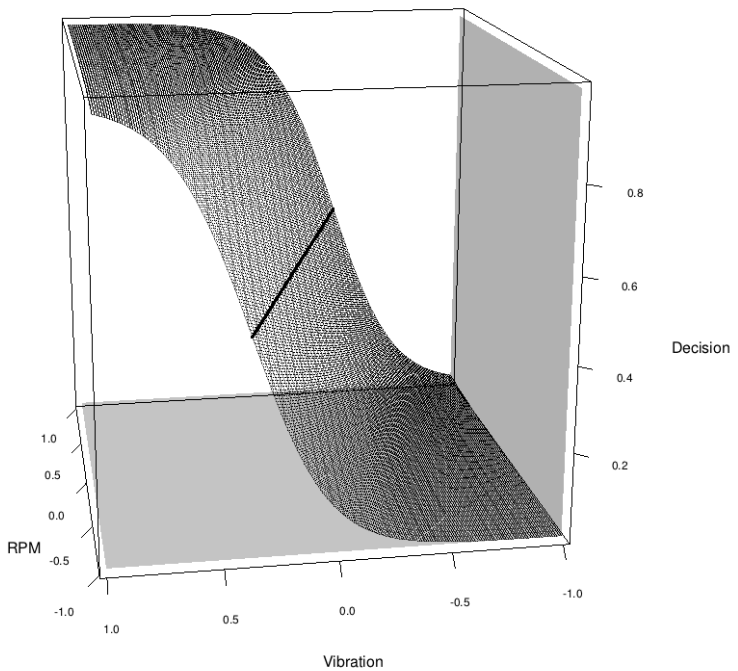
x is a numeric value and e is **Euler's number**.



$$\mathbb{M}_{\mathbf{w}}(\mathbf{d}) = \text{Logistic}(\mathbf{w} \cdot \mathbf{d}) = \frac{1}{1 + e^{-\mathbf{w} \cdot \mathbf{d}}} \tag{25}$$

1. Before training logistic regression, we map binary target levels to 0 or 1.
2. The model error for each instance is the difference between the target (0 or 1) and the prediction $[0, 1]$.

$$\mathbb{M}_{\mathbf{w}}(\langle \text{RPM}, \text{VIBRATION} \rangle) = \frac{1}{1 + e^{-(-0.4077 + 4.1697 \times \text{RPM} + 6.0460 \times \text{VIBRATION})}}$$



$$P(t = \textit{'faulty'} | \mathbf{d}) = \mathbb{M}_{\mathbf{w}}(\mathbf{d})$$

$$P(t = \textit{'good'} | \mathbf{d}) = 1 - \mathbb{M}_{\mathbf{w}}(\mathbf{d})$$

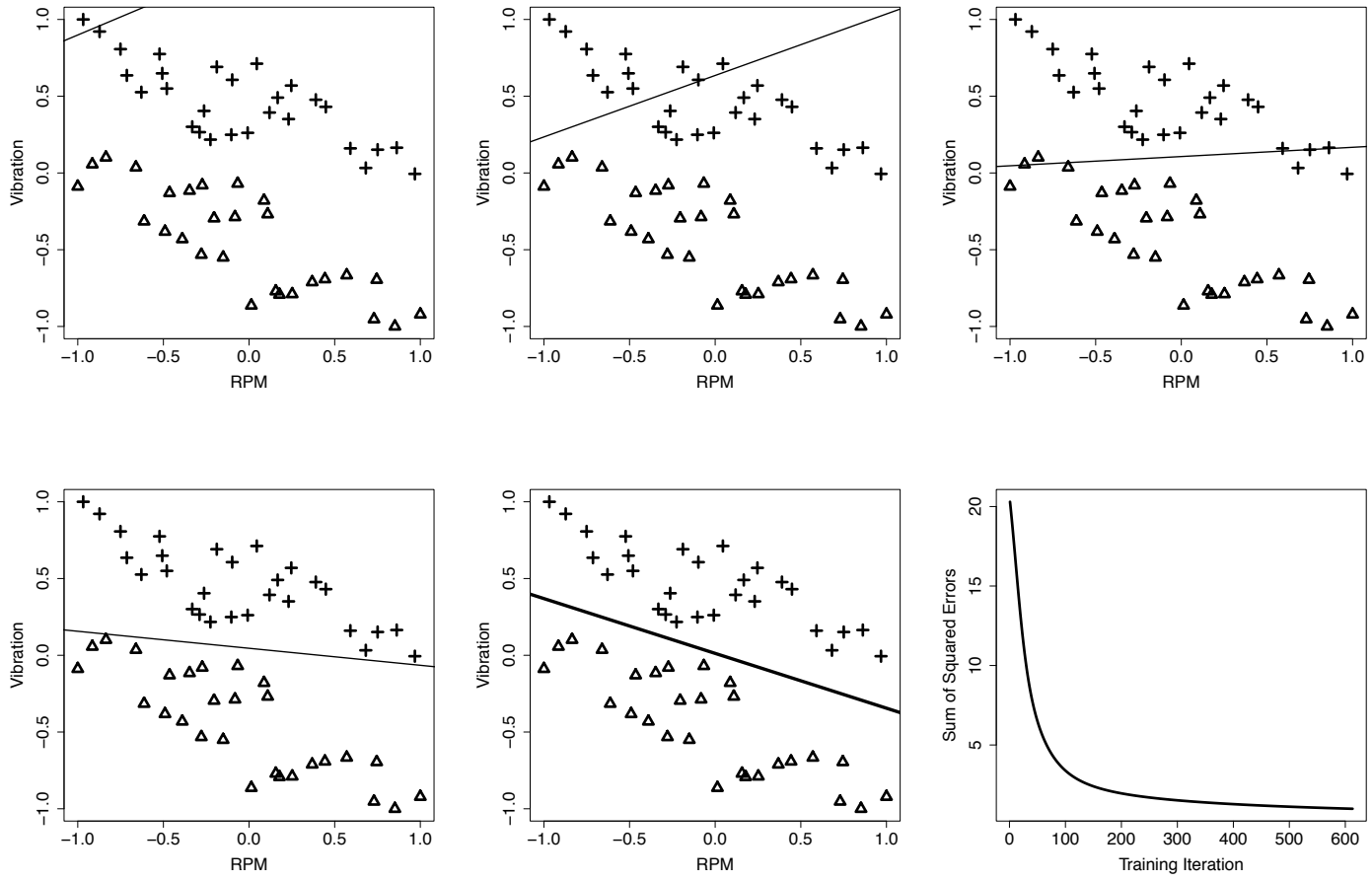


Figure: A selection of the logistic regression models developed during the gradient descent process for the machinery dataset. The bottom-right panel shows the sum of squared error values generated during the gradient descent process.

- To adapt gradient descent for logistic regression, only the weight update rule needs to change.

$$\mathbf{w}[j] \leftarrow \mathbf{w}[j] + \alpha \times \sum_{i=1}^{n} ((t - \mathbb{M}_{\mathbf{w}}(\mathbf{d}_i)) \times \mathbb{M}_{\mathbf{w}}(\mathbf{d}_i) \times (1 - \mathbb{M}_{\mathbf{w}}(\mathbf{d}_i)) \times \mathbf{d}_i[j])$$

ID	RPM	VIBRATION	STATUS
1	498	604	faulty
2	517	594	faulty
3	541	574	faulty
4	555	587	faulty
5	572	537	faulty
6	600	553	faulty
7	621	482	faulty
8	632	539	faulty
9	656	476	faulty
10	653	554	faulty
11	679	516	faulty
12	688	524	faulty
13	684	450	faulty
14	699	512	faulty
15	703	505	faulty
16	717	377	faulty
17	740	377	faulty
18	749	501	faulty
19	756	492	faulty
20	752	381	faulty
21	762	508	faulty
22	781	474	faulty
23	781	480	faulty
24	804	460	faulty
25	828	346	faulty
26	830	366	faulty
27	864	344	faulty
28	882	403	faulty
29	891	338	faulty
30	921	362	faulty
31	941	301	faulty
32	965	336	faulty
33	976	297	faulty
34	994	287	faulty
35	501	463	good
36	526	443	good
37	536	412	good
38	564	394	good
39	584	398	good
40	602	398	good
41	610	428	good
42	638	389	good
43	652	394	good
44	659	336	good
45	662	364	good
46	672	308	good
47	691	248	good
48	694	401	good
49	718	313	good
50	720	410	good
51	723	389	good
52	744	227	good
53	741	397	good
54	770	200	good
55	764	370	good
56	790	248	good
57	786	344	good
58	792	290	good
59	818	268	good
60	845	232	good
61	867	195	good
62	878	168	good
63	895	218	good
64	916	221	good
65	950	156	good
66	956	174	good
67	973	134	good
68	1002	121	good

- First normalize descriptive features to $[-1, 1]$, and set $\alpha = 0.02$

Initial Weights

w[0]:	-2.9465	**w**[1]:	-1.0147	**w**[2]:	-2.1610

Iteration 1

ID	TARGET LEVEL	Pred.	Error	Squared Error	**errorDelta**$(\mathcal{D}, \mathbf{w}[\mathbf{i}])$ **w**[0]	**w**[1]	**w**[2]
1	1	0.5570	0.4430	0.1963	0.1093	-0.1093	0.1093
2	1	0.5168	0.4832	0.2335	0.1207	-0.1116	0.1159
3	1	0.4469	0.5531	0.3059	0.1367	-0.1134	0.1197
4	1	0.4629	0.5371	0.2885	0.1335	-0.1033	0.1244
				...			
65	0	0.0037	-0.0037	0.0000	0.0000	0.0000	0.0000
66	0	0.0042	-0.0042	0.0000	0.0000	0.0000	0.0000
67	0	0.0028	-0.0028	0.0000	0.0000	0.0000	0.0000
68	0	0.0022	-0.0022	0.0000	0.0000	0.0000	0.0000
			Sum	24.4738	2.7031	-0.7015	1.6493
Sum of squared errors (Sum/2)				12.2369			

$$\mathbf{w}[j] \leftarrow \mathbf{w}[j] + \alpha \times \sum_{i=1}^{n} ((t_i - \mathbb{M}_{\mathbf{w}}(\mathbf{d}_i)) \times \mathbb{M}_{\mathbf{w}}(\mathbf{d}_i) \times (1 - \mathbb{M}_{\mathbf{w}}(\mathbf{d}_i)) \times \mathbf{d}_i[j])$$

New Weights (after Iteration 1)

w[0]:	-2.8924	**w**[1]:	-1.0287	**w**[2]:	-2.1940

- The final model found is:

$$\mathbb{M}_{\mathbf{w}}(\langle \text{RPM}, \text{VIBRATION} \rangle) = \frac{1}{1 + e^{-(-0.4077 + 4.1697 \times \text{RPM} + 6.0460 \times \text{VIBRATION})}}$$

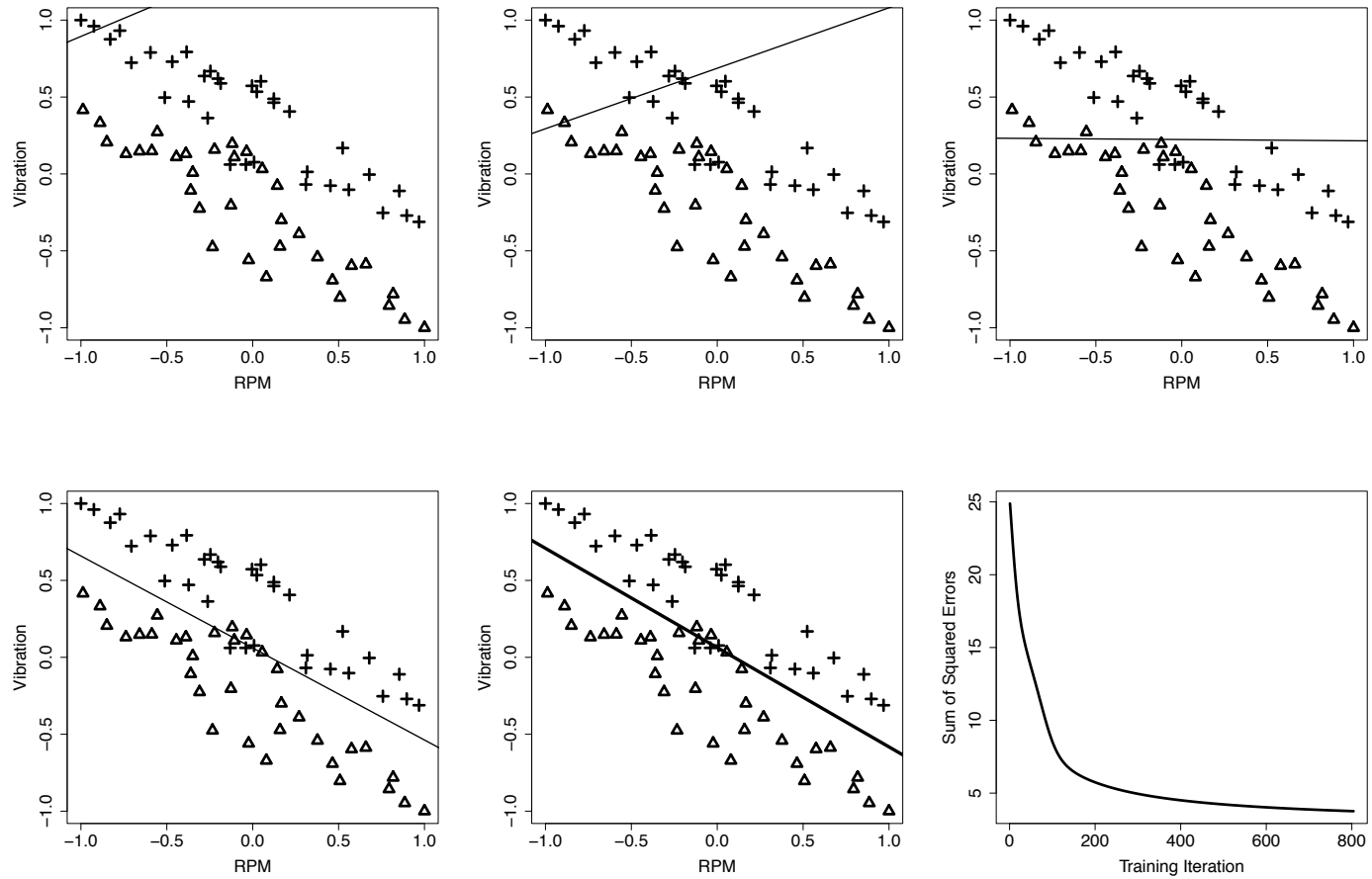


Figure: A selection of the logistic regression models with gradient descent process. The bottom-right shows the sum of squared error values.

Other Non-linear Regression Models

Table: A dataset describing grass growth on Irish farms during July 2012.

ID	RAIN	GROWTH	ID	RAIN	GROWTH	ID	RAIN	GROWTH
1	2.153	14.016	12	3.754	11.420	23	3.960	10.307
2	3.933	10.834	13	2.809	13.847	24	3.592	12.069
3	1.699	13.026	14	1.809	13.757	25	3.451	12.335
4	1.164	11.019	15	4.114	9.101	26	1.197	10.806
5	4.793	4.162	16	2.834	13.923	27	0.723	7.822
6	2.690	14.167	17	3.872	10.795	28	1.958	14.010
7	3.982	10.190	18	2.174	14.307	29	2.366	14.088
8	3.333	13.525	19	4.353	8.059	30	1.530	12.701
9	1.942	13.899	20	3.684	12.041	31	0.847	9.012
10	2.876	13.949	21	2.140	14.641	32	3.843	10.885
11	4.277	8.643	22	2.783	14.138	33	0.976	9.876

- The best linear model we can learn for this data is:

$$\text{GROWTH} = 13.510 + -0.667 \times \text{RAIN}$$

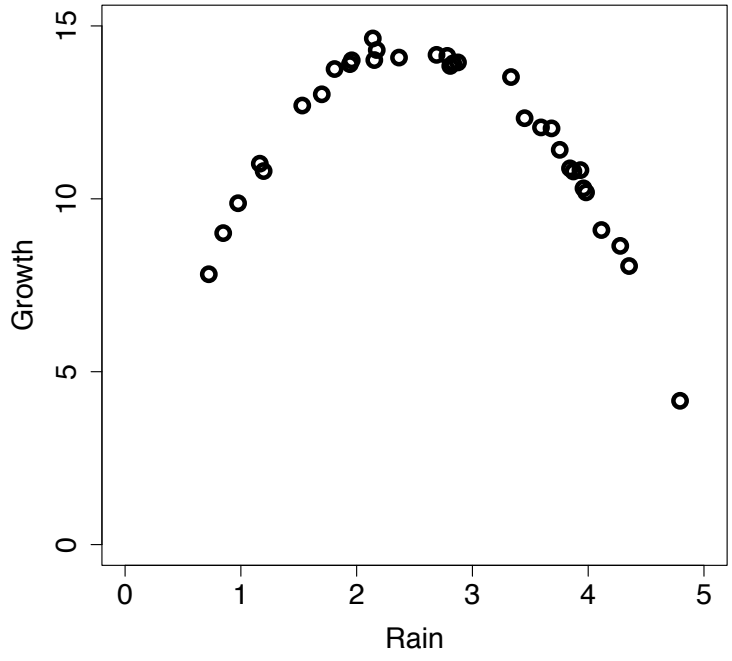


Figure: Grass Growth Dataset

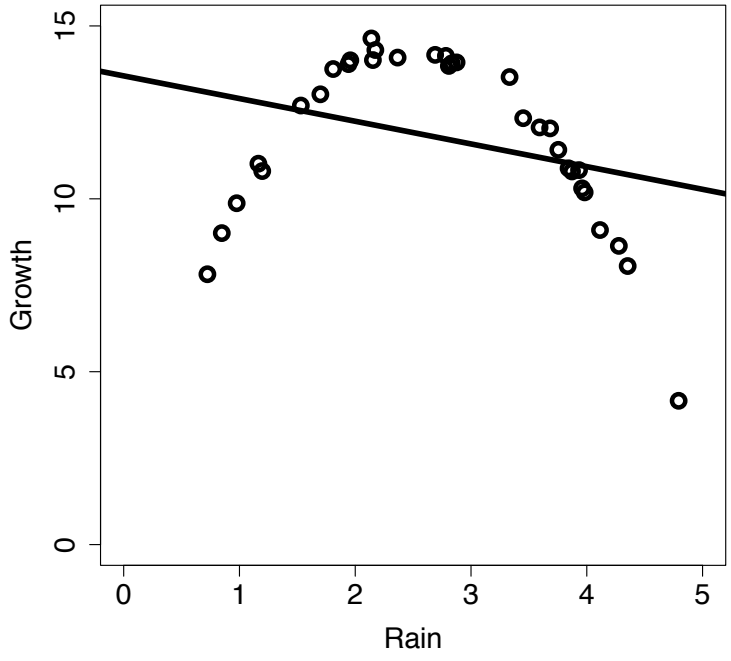


Figure: Linear Model

- To handle non-linear relationships, we transform the data using basis functions:

$$\mathbb{M}_{\mathbf{w}}(\mathbf{d}) = \sum_{k=0}^{b} \mathbf{w}[k]\phi_k(\mathbf{d}) \tag{26}$$

  where ϕ_0 to ϕ_b are basis functions that transform $\mathbf{d}$.
- This approach requires no other changes to the existing method.
- The relationship between rainfall and grass growth can be modeled as a second-order polynomial:

$$\textsc{Growth} = \mathbf{w}[0]\phi_0(\textsc{Rain}) + \mathbf{w}[1]\phi_1(\textsc{Rain}) + \mathbf{w}[2]\phi_2(\textsc{Rain})$$

  where

$$\phi_0(\textsc{Rain}) = 1, \quad \phi_1(\textsc{Rain}) = \textsc{Rain}, \quad \phi_2(\textsc{Rain}) = \textsc{Rain}^2$$

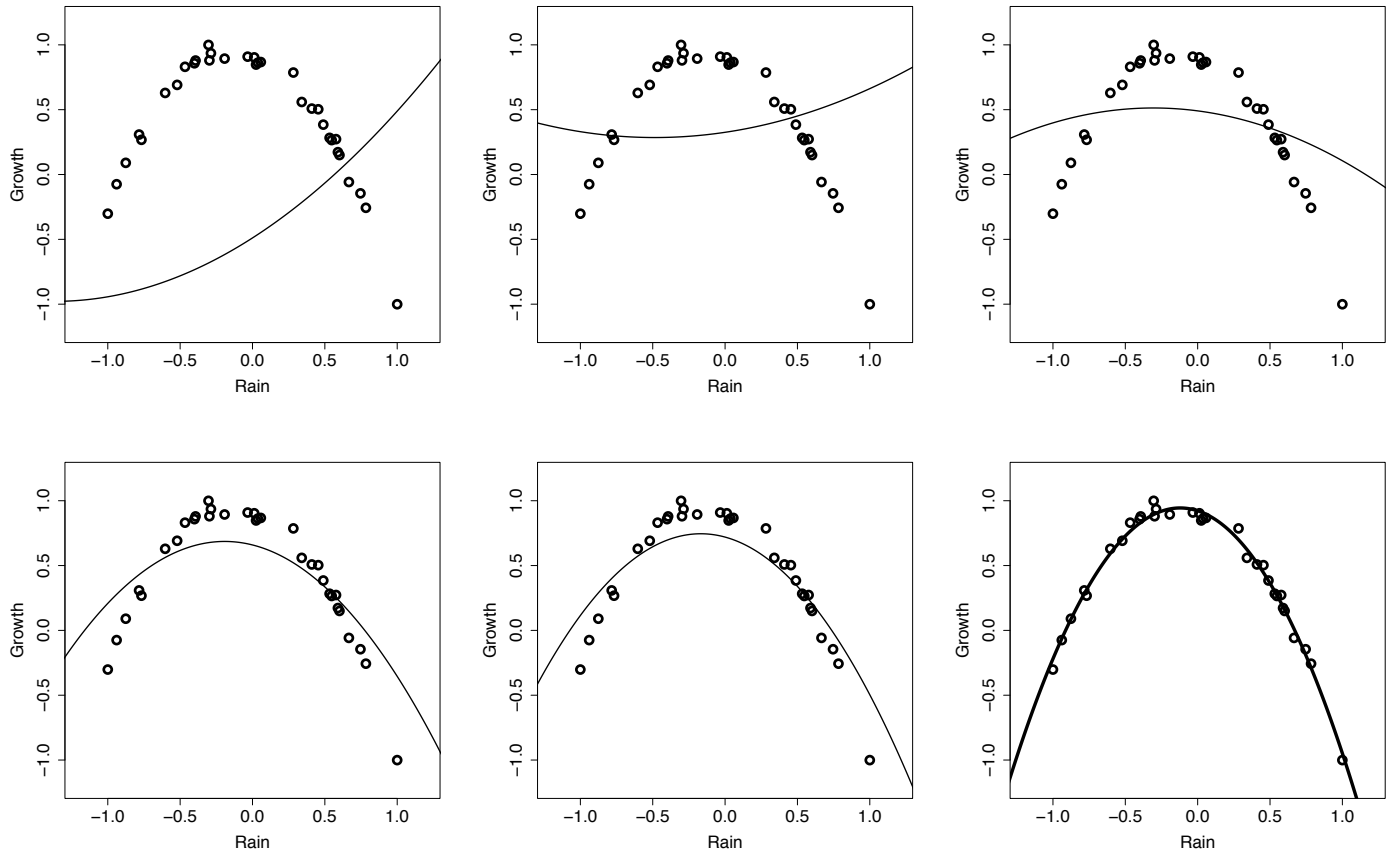


Figure: A selection of the models developed with gradient descent.

$$\text{GROWTH} = 0.3707 \times \phi_0(\text{RAIN}) + 0.8475 \times \phi_1(\text{RAIN}) + -1.717 \times \phi_2(\text{RAIN})$$

Ridge Regression

- **Objective**: Minimize the sum of squared residuals with a penalty on the size of coefficients.
- **Formula**:

$$\min_{\mathbf{w}} \left(\sum_{i=1}^{n} (y_i - \mathbf{w}^T \mathbf{x}_i)^2 + \lambda \sum_{j=1}^{p} w_j^2 \right) \tag{27}$$

where λ is the regularization parameter.

Benefits of Ridge Regression

- **Reduces Overfitting**: By adding a penalty on the size of coefficients, it shrinks large weights, thus reducing model complexity.
- **Handles Multicollinearity**: It can handle multicollinearity better than standard linear regression by imposing a constraint on the coefficients.
- **Stability**: Makes the model more stable and less sensitive to the training data.

Lasso Regression

- **Objective**: Minimize the sum of squared residuals with a penalty on the absolute value of coefficients.
- **Formula**:

$$\min_{\mathbf{w}} \left(\sum_{i=1}^{n} (y_i - \mathbf{w}^T \mathbf{x}_i)^2 + \lambda \sum_{j=1}^{p} |w_j| \right) \tag{28}$$

where λ is the regularization parameter.

Benefits of Lasso Regression

- **Feature Selection**: Lasso can shrink some coefficients exactly to zero, effectively performing feature selection.
- **Reduces Overfitting**: By adding a penalty, it reduces the risk of overfitting the model to the training data.
- **Simpler Models**: Produces simpler models that are easier to interpret due to fewer non-zero coefficients.