Utrecht University

Lecture 2: Model Fit and Linear Regression

Utrecht Summer school on
Machine Learning with Python

July 2024

So far…

- Basic concepts of machine learning
- Types of learning
- Regression, classification
- Exploratory data analysis

In this lecture

- What is a model?
- Linear regression model
- Data splits
- Feature engineering
- Regularization

What is a model?

In supervised learning, there is a **dependent variable (Y)**, which is the one you are trying to explain/predict, and one or more **independent variables (X)** that are related to it.

You can express the relationship, such as:

$$Y = f(X) + \epsilon$$

What is a model?

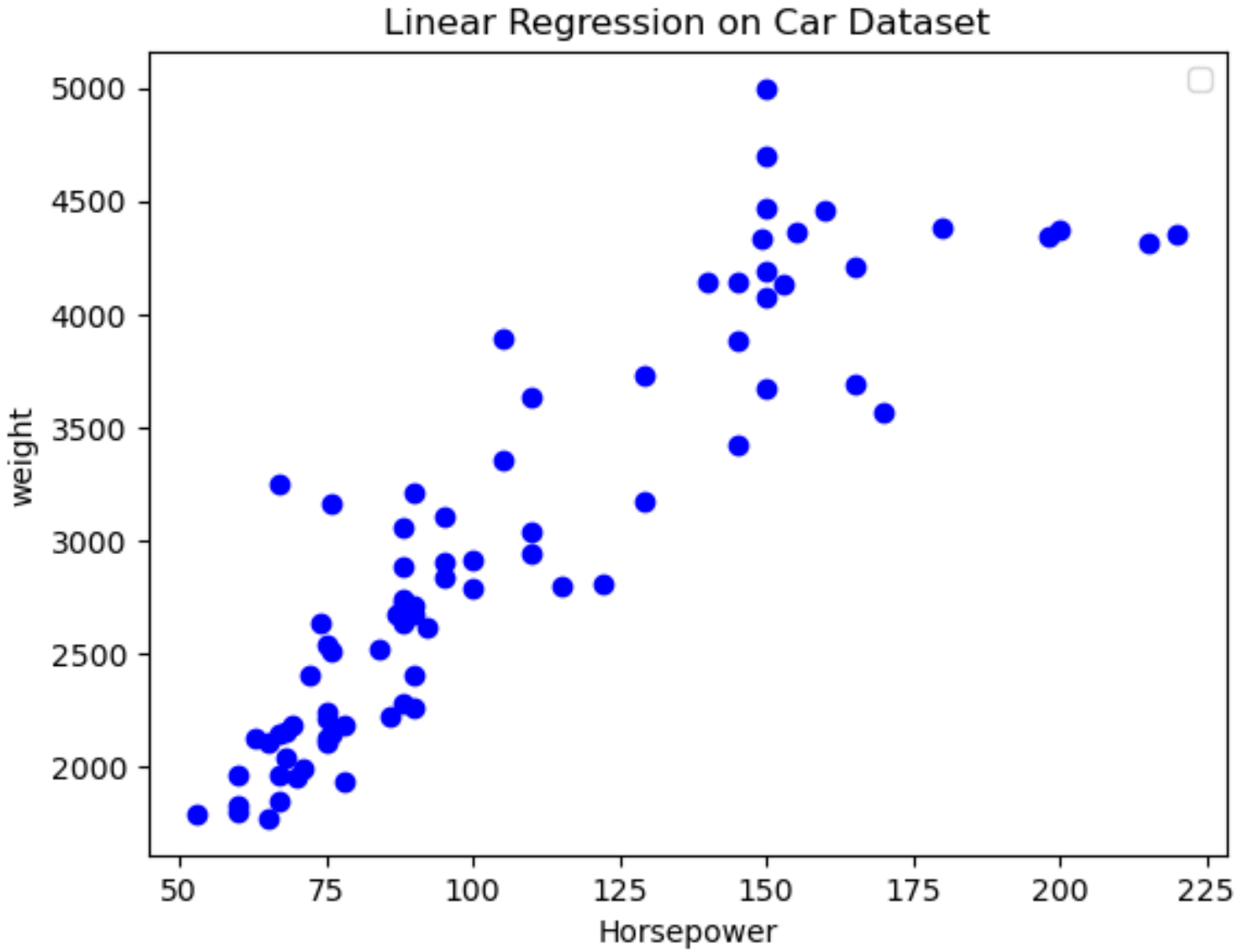
$$Y = f(X) + \epsilon$$

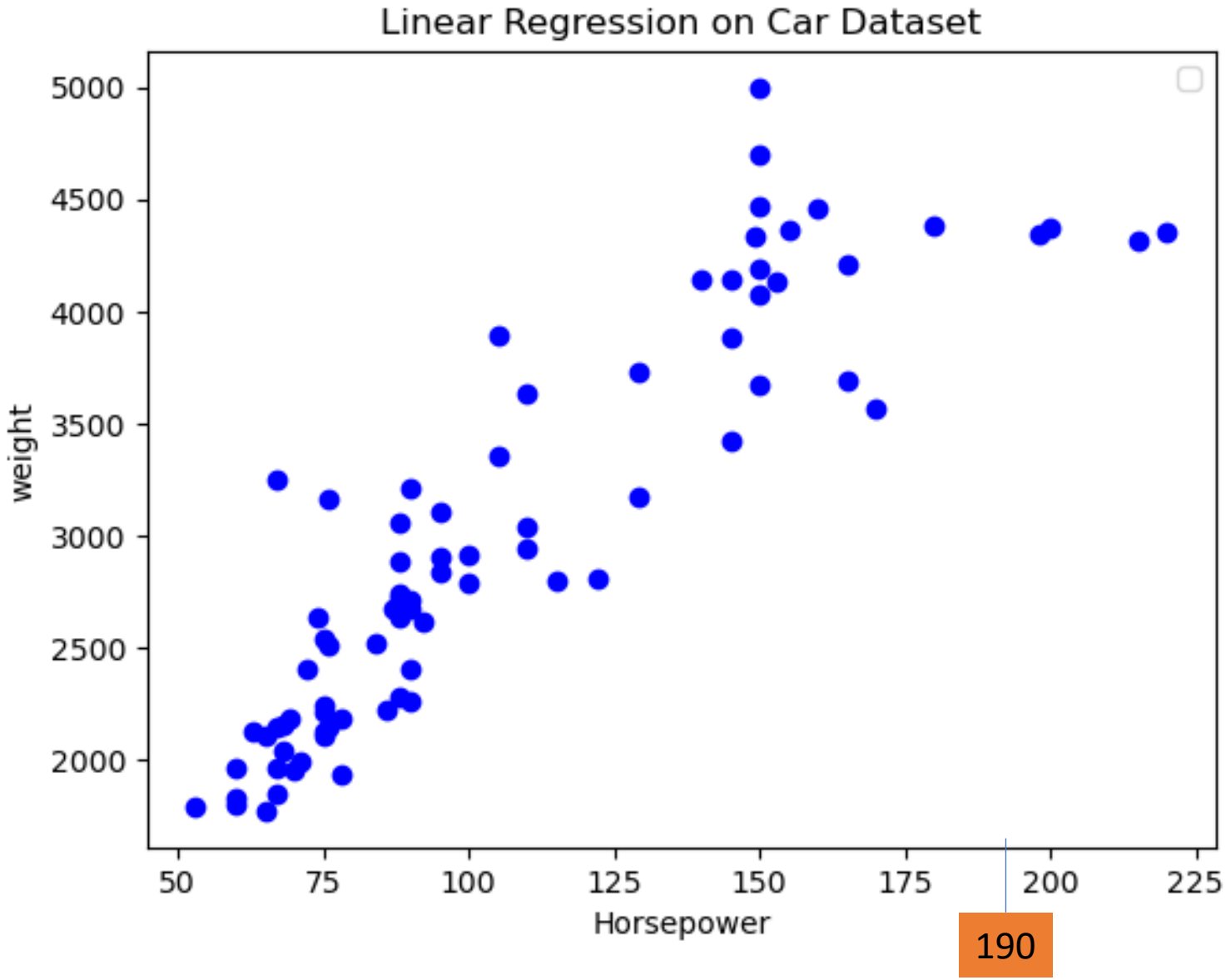
- Y: variable to predict (observed outcome)
- $f(X)$: a function of observed predictors (independent variables)
- ϵ is a random error with mean zero

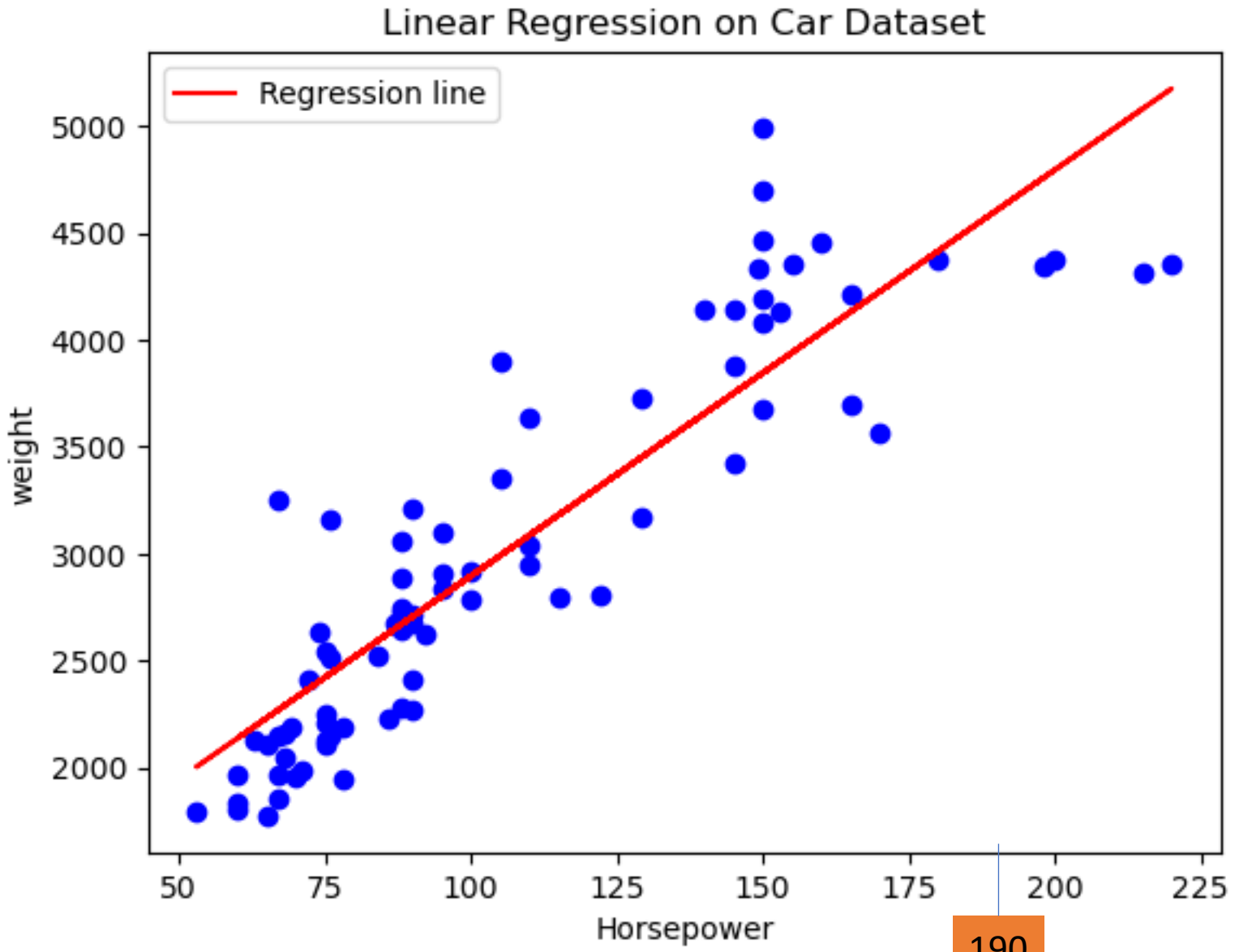
- Statistical learning: learns f using data

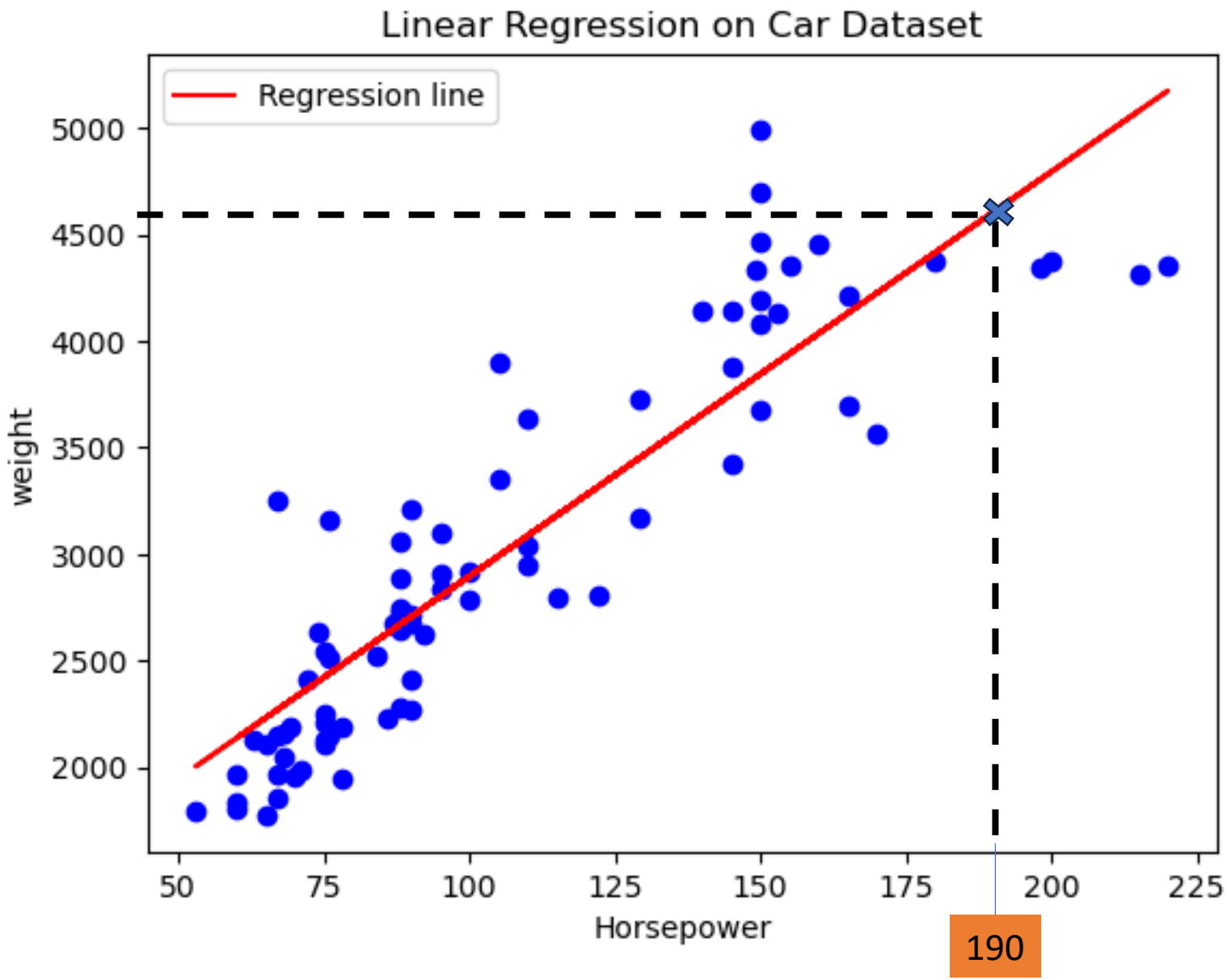
Why do we estimate f?

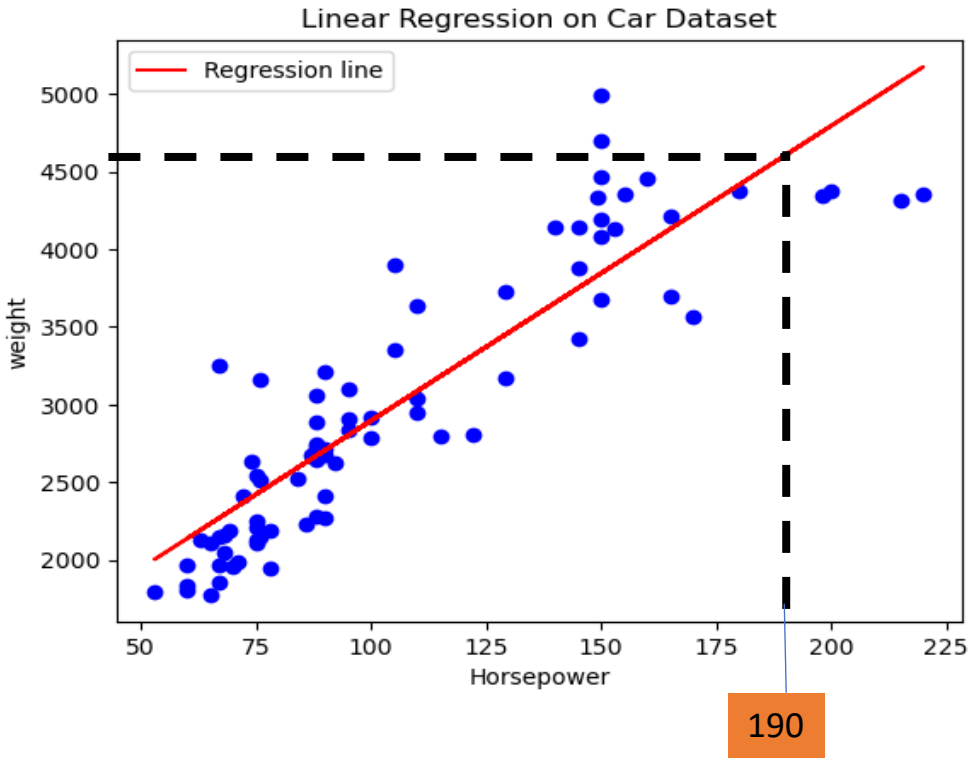
- Prediction: If we can produce a good estimate for f we can make accurate predictions for the response Y, based on a new value of X
- Inference: Alternatively, we may also be interested in how the X (the predictors) affect the Y (the outcome)
  - Which particular predictors actually affect the response?
  - Is the relationship positive or negative?
  - Is the relationship a simple linear one or is it more complicated etc.?


Linear Regression on Car Dataset
weight
Horsepower
5000
4500
4000
3500
3000
2500
2000
50
75
100
125
150
175
200
225


Linear Regression on Car Dataset
weight
Horsepower
5000
4500
4000
3500
3000
2500
2000
50
75
100
125
150
175
200
225
190


Linear Regression on Car Dataset
Regression line
weight
5000
4500
4000
3500
3000
2500
2000
50
75
100
125
150
175
200
225
Horsepower


Linear Regression on Car Dataset
Regression line
weight
5000
4500
4000
3500
3000
2500
2000
50
75
100
125
150
175
200
225
Horsepower
190



Supervised model

- Data has right answers (*Y*)
- y = ouput/target variable
- y is the car weight

The model learned the relationship from some input:

- x = input variables/predictors
- x is the horsepower

(x_i, y_i) = single training example

Model is *f(cars)* such that

Weight_car = f(cars) + ϵ

How do we estimate *f*?

- We will assume we have observed a set of training data. Training set contains input variables (features) *X* and one target variable *Y*
- $\{(X_1, Y_1), (X_2, Y_2), \ldots, (X_n, Y_n)\}$

- We use the training data and a statistical method to estimate *f*
- e.g. Weight_car = *f*(cars) + ϵ

- We have a set of test data to make predictions $\hat{y}$

How do we estimate *f*?

Statistical Learning Methods:

- **Parametric Methods**: Make some assumption about the functional form of *f*
- e.g. $f(cars) = \beta_0 + \beta_1 \cdot cars$
- **Non-parametric Methods:** They do not make explicit assumptions about the functional form of *f*
- e.g. The price of car is the average of the 3 points in our dataset with the closest number of Cars.

Linear Regression

Linear regression

- Fits a line, assumes linear relationship between the predictors and the outcome.

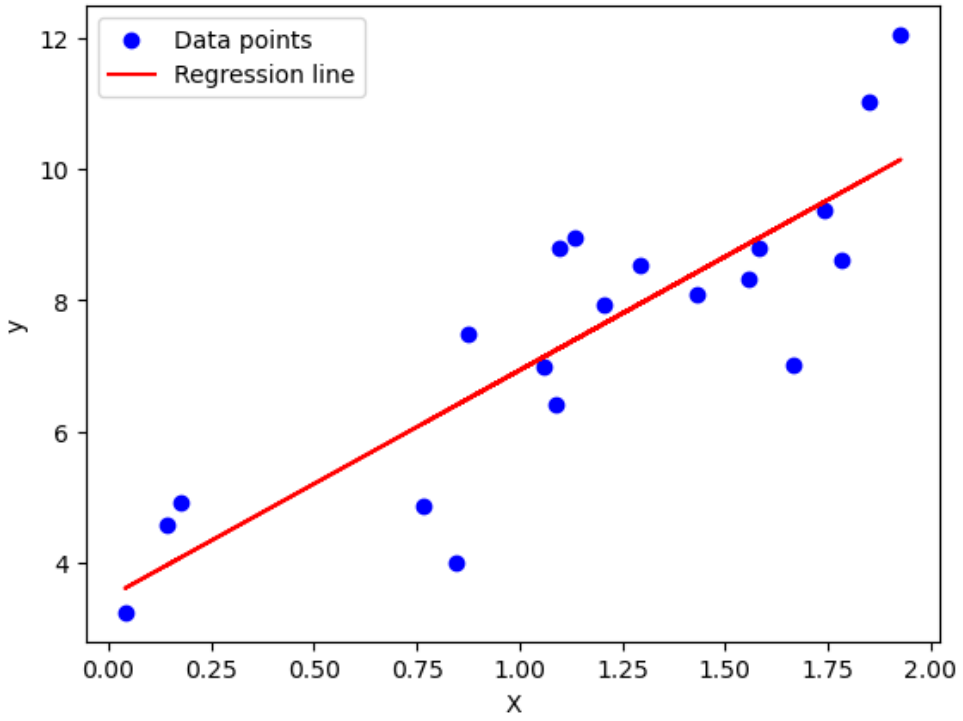
- $f = \beta_0 + \beta_1 \cdot X + \epsilon$

- β_0 in the intercept term -> the expected value of Y when X = 0,
- β_1 is the slope—the average increase in Y associated with a one-unit increase in X
- β_0, β_1 are also called parameters or weights

- The error term is for what we miss with this simple model: the true relationship is probably not linear, there may be other variables that cause variation in Y , and there may be measurement error.

How well the line fits the data?

How well this line fits the data?



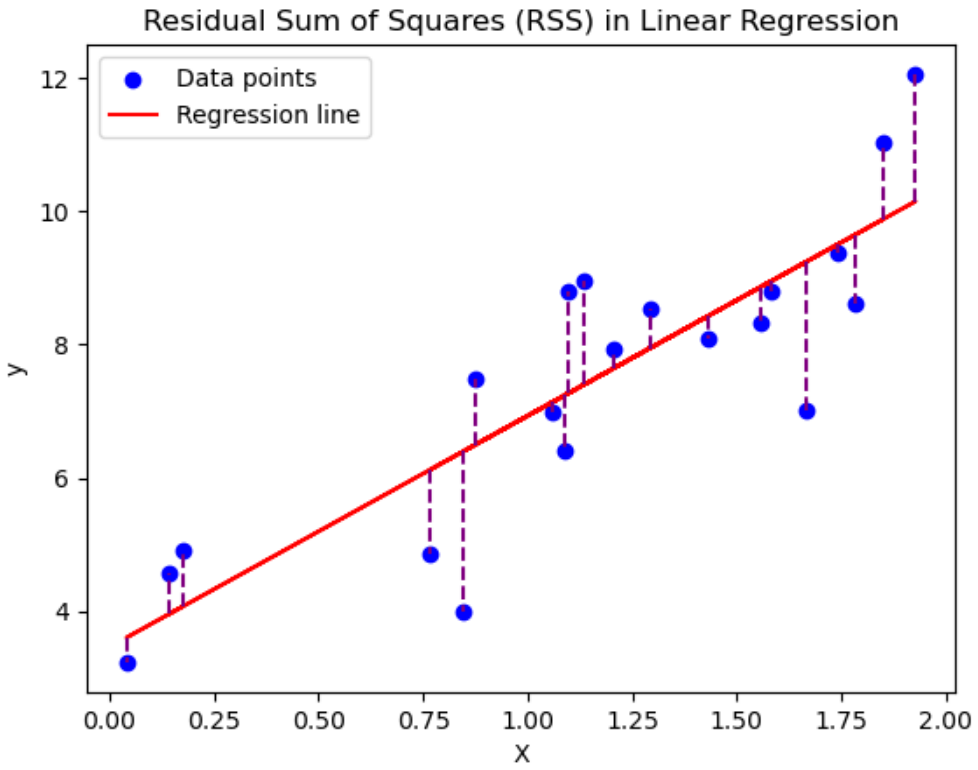
$f = \beta_0 + \beta_1 \cdot X + \epsilon$

$\hat{y}_i = f(x_i)$

We need to find β_0 *and* β_1 *such that* $\hat{y}$ is close to y_i for all (x_i, y_i)

How well the line fits the data?

How well the line fits the data?



$f = \beta_0 + \beta_1 \cdot X + \epsilon$

$\hat{y}_i = f(x_i)$

Find β_0 *and* β_1 *such that* $\hat{y}$ is close to y_i for all (x_i, y_i)

$(y_i - \hat{y}_i)^2$ is the error for one of the observations

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

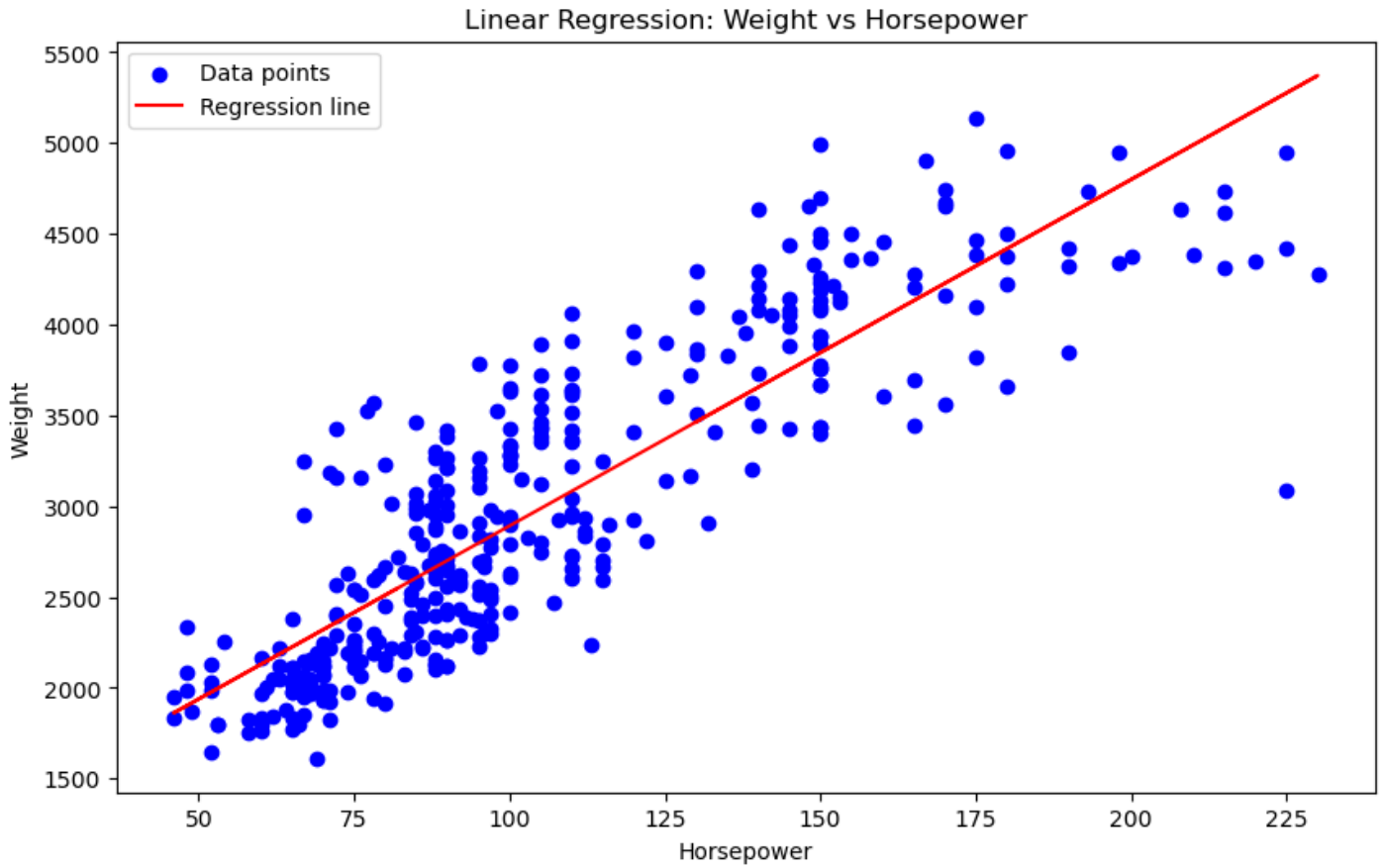
Cost function

- How well the line fits the data?
- Residual sum of squares

$$\mathrm{RSS} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2.$$

$$\mathrm{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{f}(x_i))^2,$$

Example



$f = \beta_0 + \beta_1 \cdot X$

$f = 984 + 19{,}07 *$ horsepower

Interpretation

- 0.747: This means that approximately 74.7% of the variance in the dependent variable (weight) can be explained by the independent variable (horsepower).
- An R-squared value closer to 1 indicates a better fit of the model.

OLS Regression Results

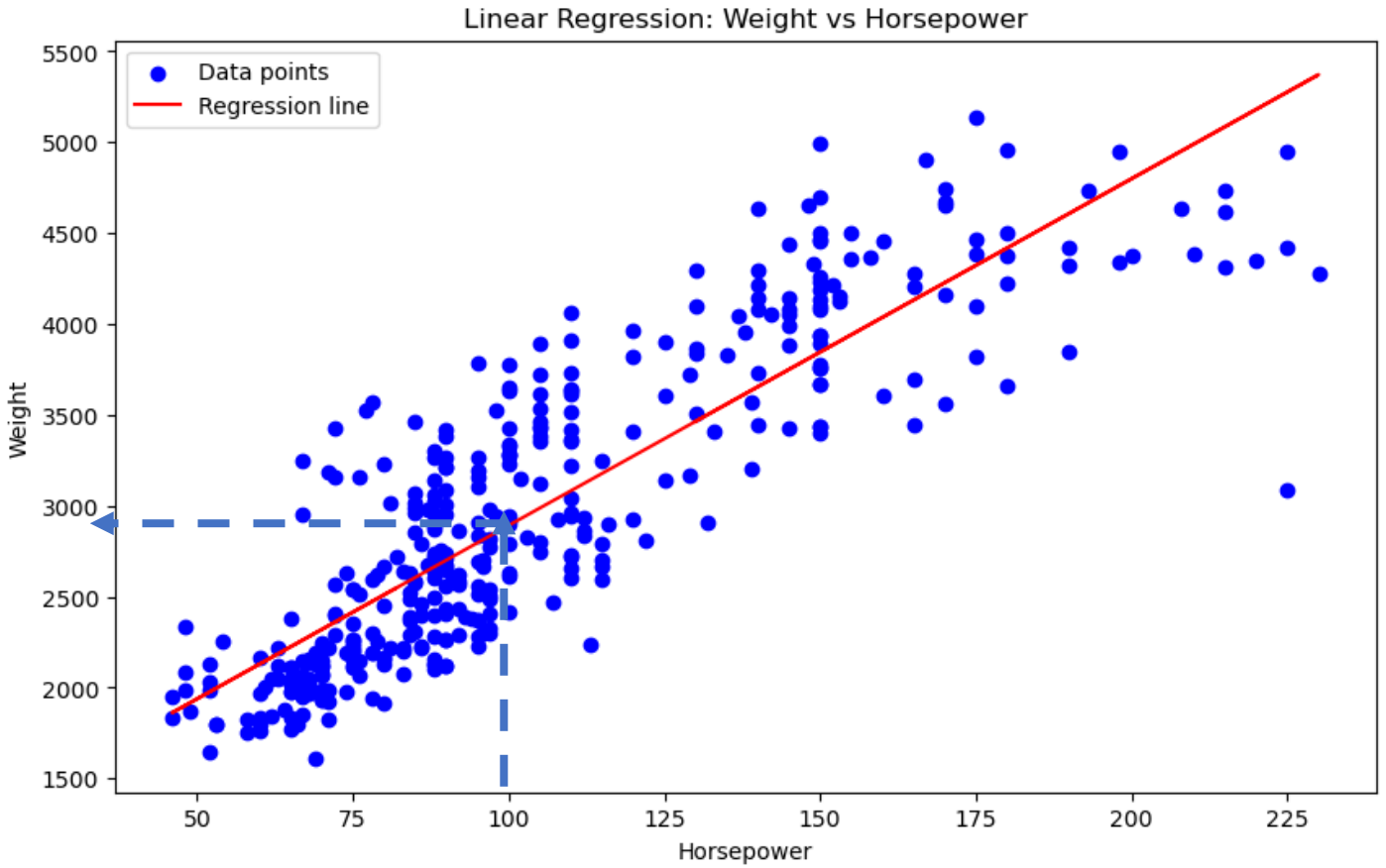
Dep. Variable:	weight	R-squared:	0.747
Model:	OLS	Adj. R-squared:	0.747
Method:	Least Squares	F-statistic:	1154.
Date:	Sun, 21 Jul 2024	Prob (F-statistic):	1.36e-118
Time:	17:39:53	Log-Likelihood:	-2929.9
No. Observations:	392	AIC:	5864.
Df Residuals:	390	BIC:	5872.
Df Model:	1		
Covariance Type:	nonrobust		

Interpretation

- const: The intercept of the regression line.
- Here, the intercept is 984, which means when horsepower is 0, the weight is expected to be 985 units.
- horsepower: The slope of the regression line. Here, the slope is 19.07, meaning that for each additional unit of horsepower, the weight increases by 19.07 units on average.

	coef
const	984.5003
horsepower	19.0782

Example



$f = \beta_0 + \beta_1 \cdot X$

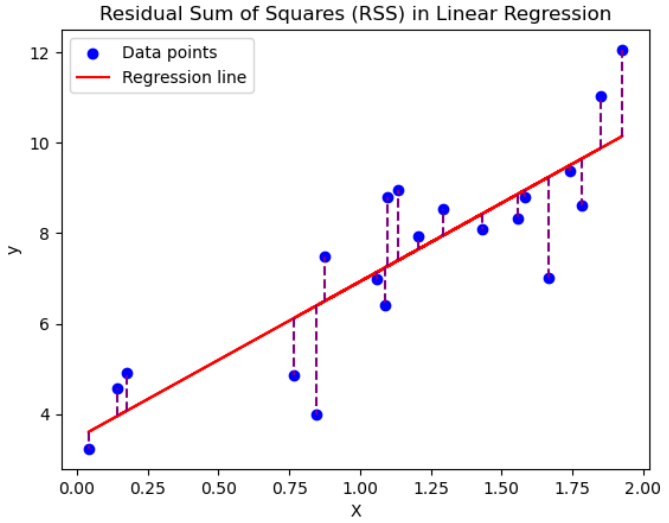
$f = 984 + 19{,}07 * horsepower$

If a new observation arrives with horsepower = 100:

$f(100) = 984 + 19{,}07 * 100 = 2891$

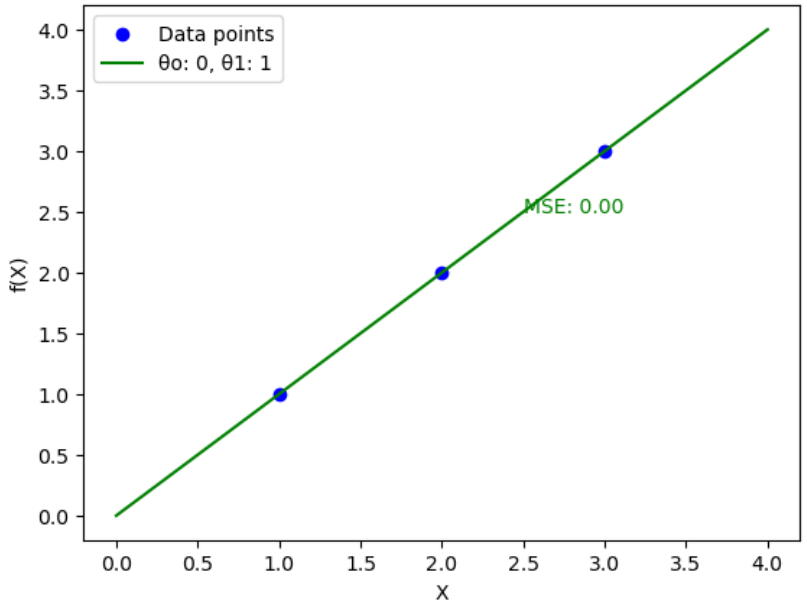
Least squares linear regression

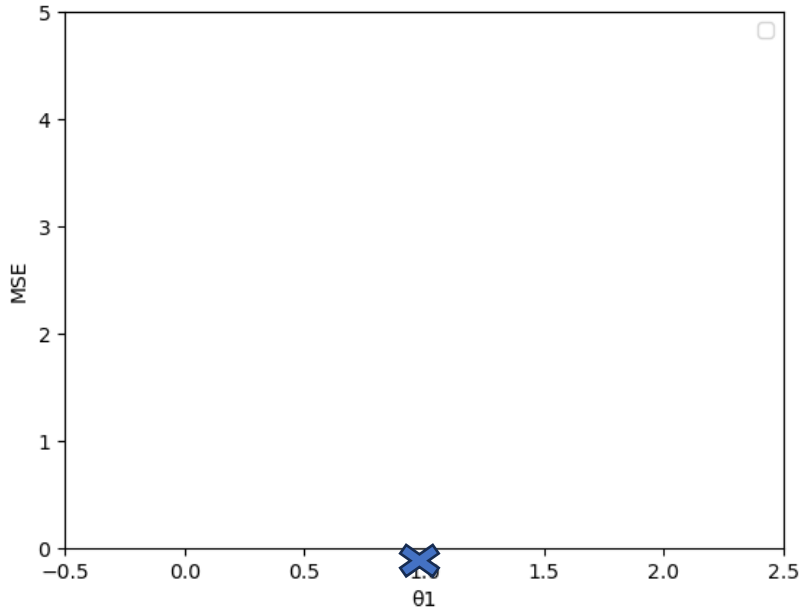
- Goal: Find the parameters ϑ *(same as* β*)* that minimise the loss/cost function/MSE
- *L(*ϑ*) = Cost(*ϑ*)=MSE(*ϑ*) =* $\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2,$



How to select ϑ?

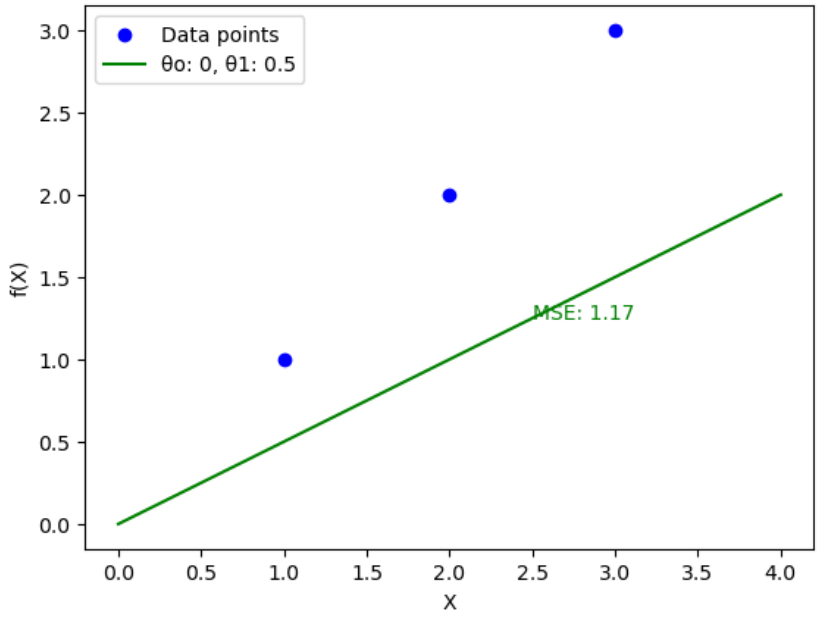
- Goal: Minimize MSE $L(\vartheta) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2,$
- Let's assume a simple model $f(x_i) = \vartheta_1 * x_i$ where intercept is 0
- $\vartheta_0 = 0, \vartheta_1 = 1$

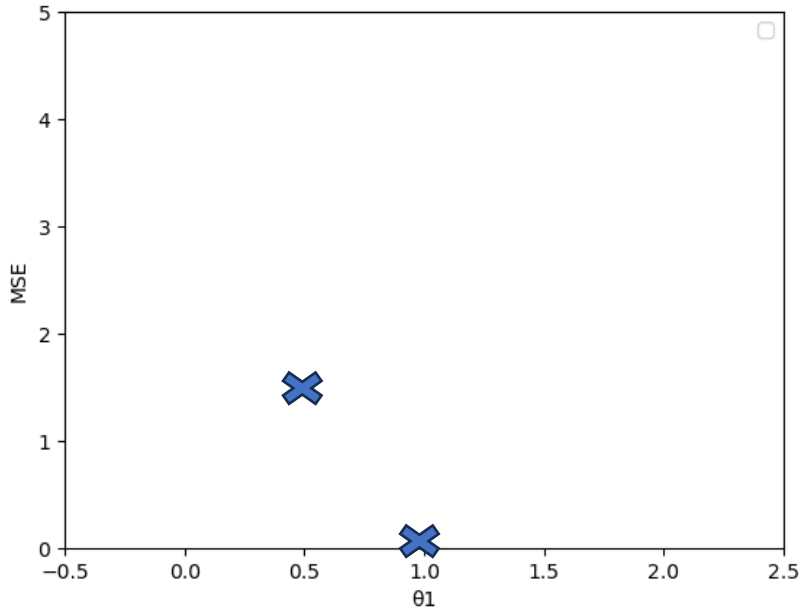




How to select ϑ?

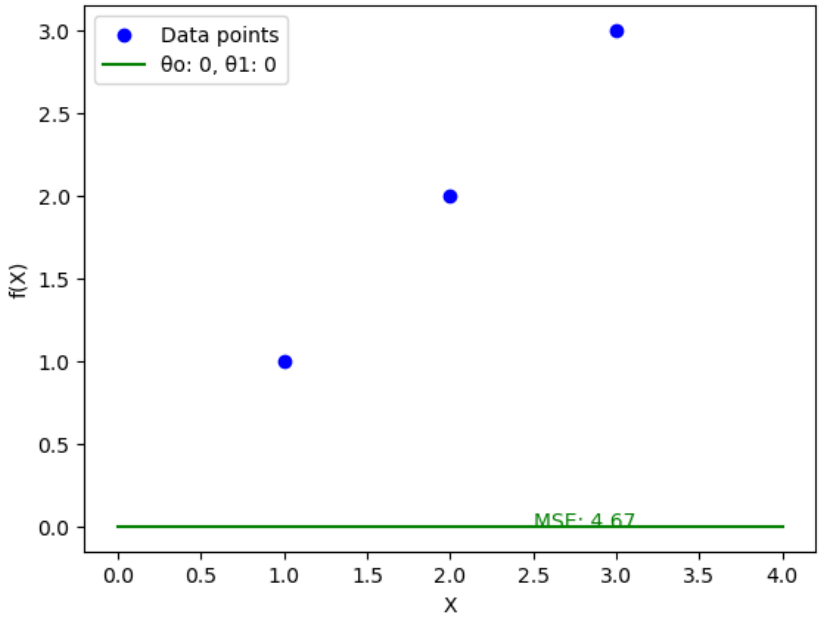
- Goal: Minimize MSE $L(\vartheta) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2,$
- Let's assume a simple model $f(x_i) = \vartheta_1 * x_i$ where intercept is 0
- $\vartheta_0 = 0, \vartheta_1 = 0.5$

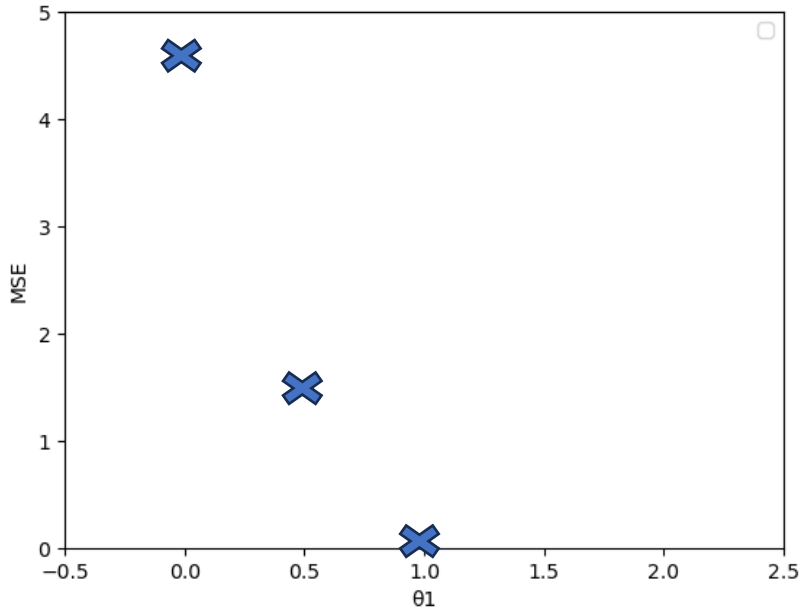




How to select ϑ?

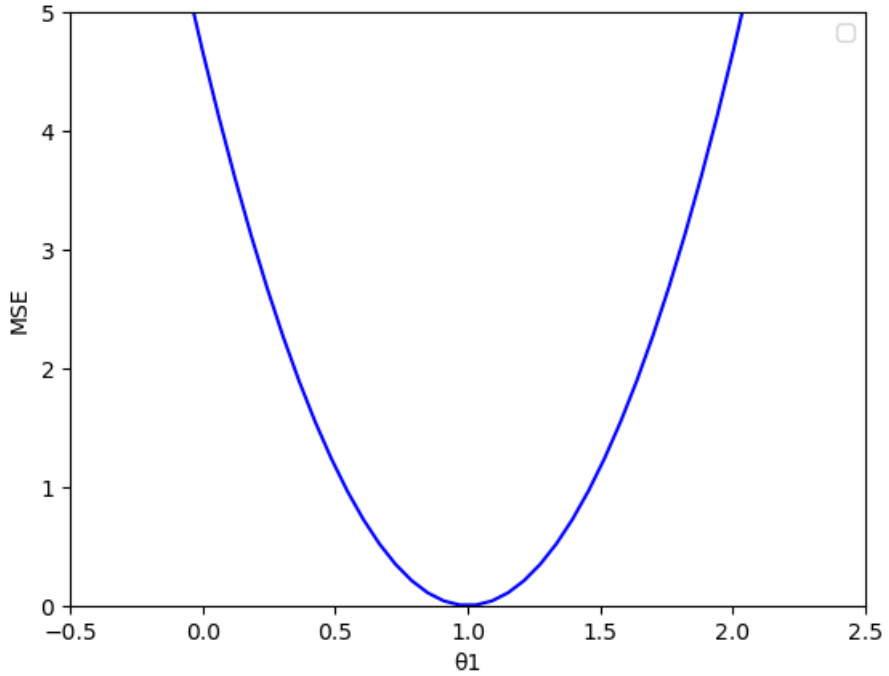
- Goal: Minimize MSE $L(\vartheta) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2,$
- Let's assume a simple model $f(x_i) = \vartheta_1 * x_i$ where intercept is 0
- $\vartheta_0 = 0, \vartheta_1 = 0$



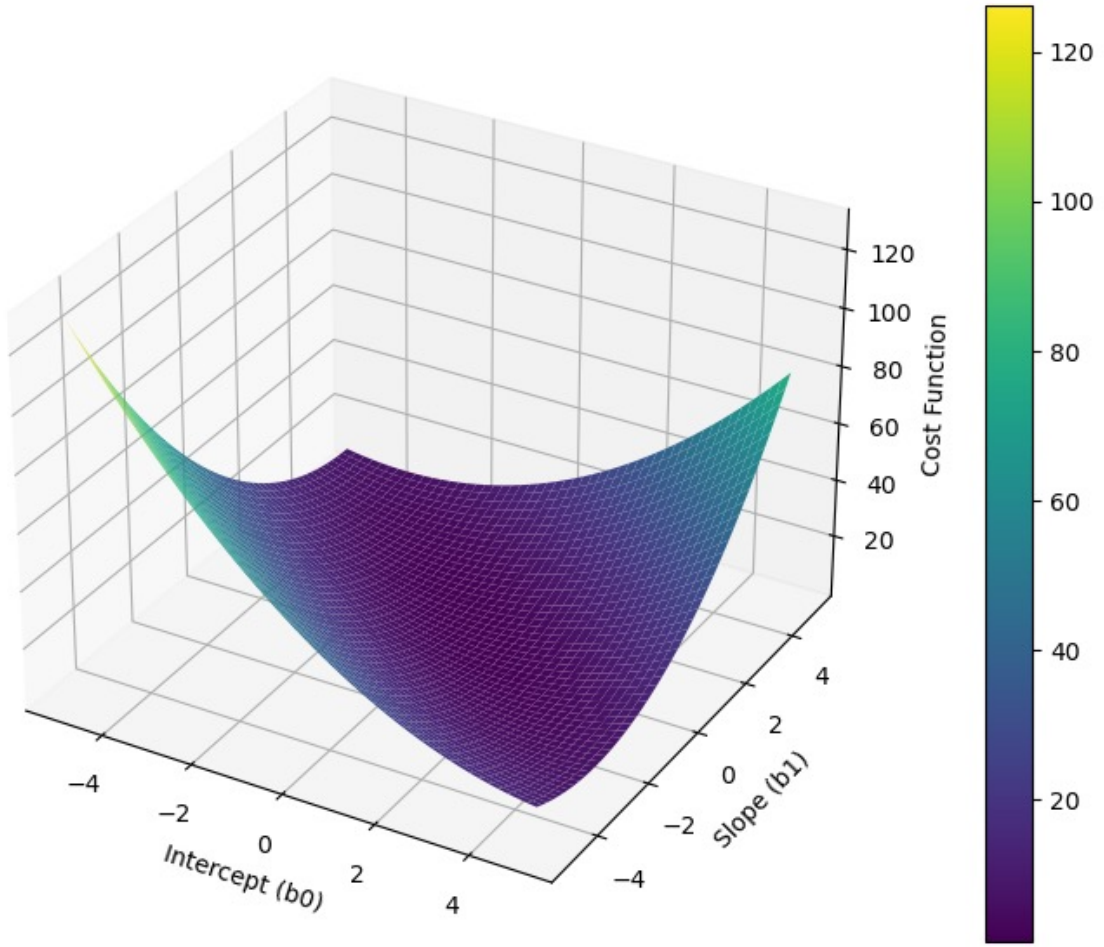


Gradient descent

- Goal: Minimize MSE *L(ϑ)* = $\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2,$
- *Let's assume a simple model f(xi) = ϑ1 * xi where intercept is 0*



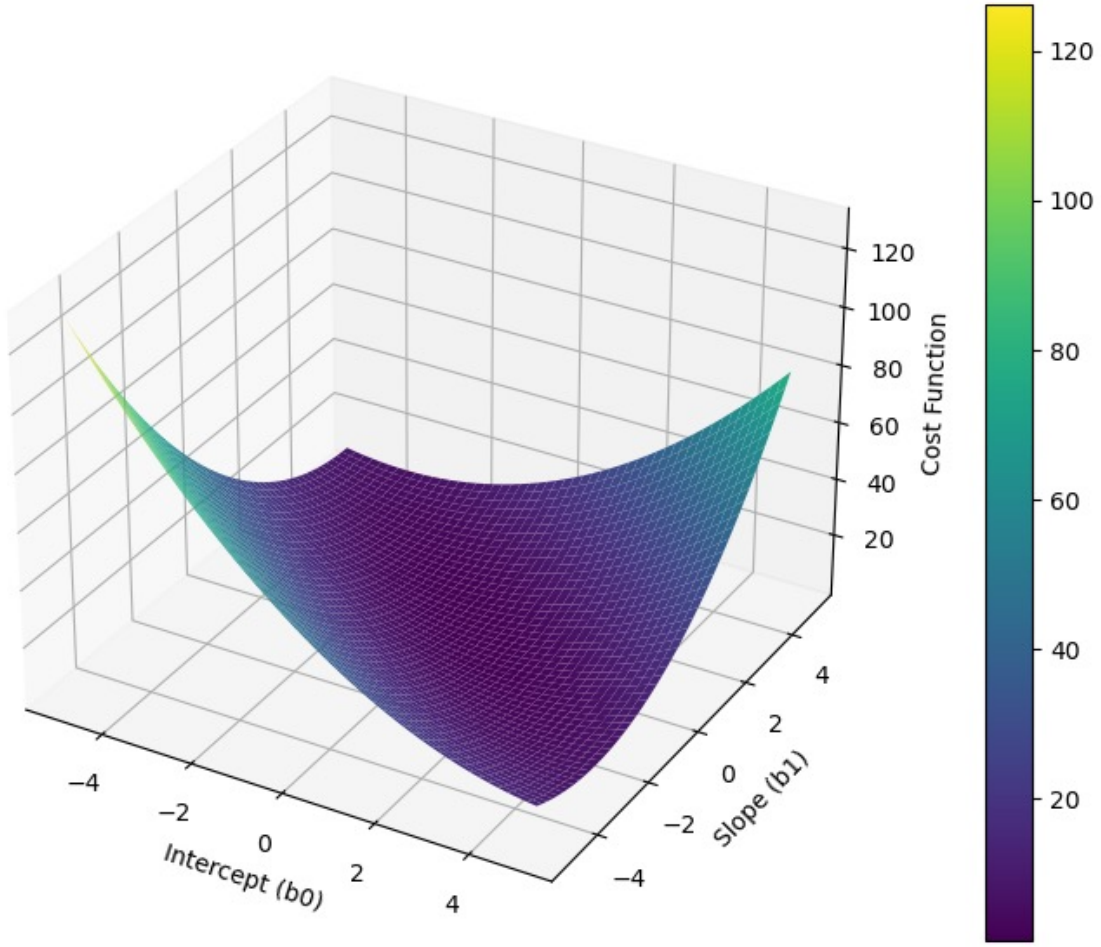
The cost function is convex



How to find the optimal θ?

- Find the ones that minimize the cost function
- Use gradient descent -> optimization technique used in many machine learning methods

Drop a marble in the curve

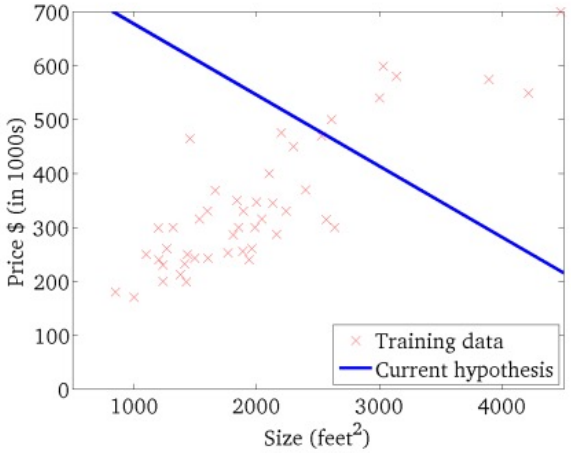


Cost function

- Initiate parameters ϑ randomly
- Predict $h\vartheta$ (x) (i.e., $\hat{y}$) and calculate $L(\vartheta)$
- Change ϑ using a small value to move to a lower point in the curve
- Repeat until near minimum

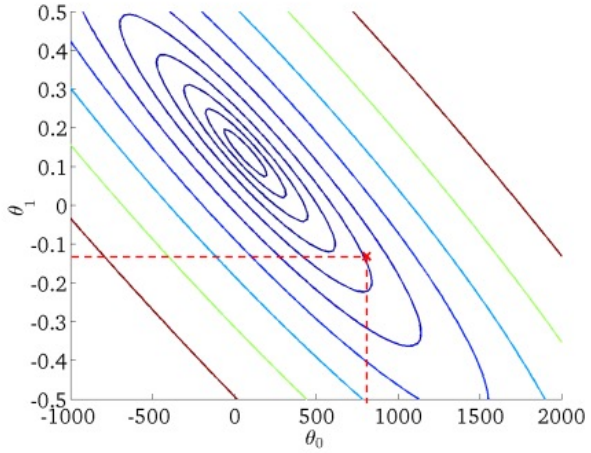
$h_{\boldsymbol{\theta}}(x)$

(for fixed θ_0, θ_1, this is a function of x)



$\mathcal{L}(\boldsymbol{\theta}_0, \boldsymbol{\theta}_1)$

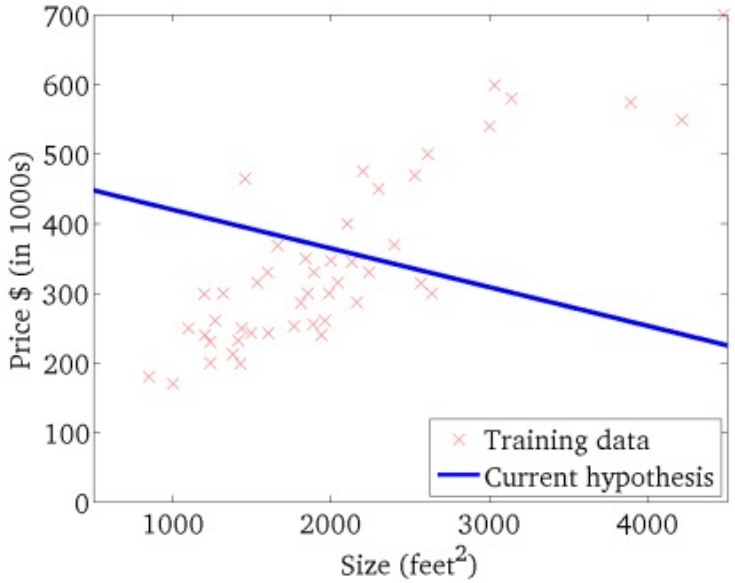
(function of the parameters θ_0, θ_1)



Cost function

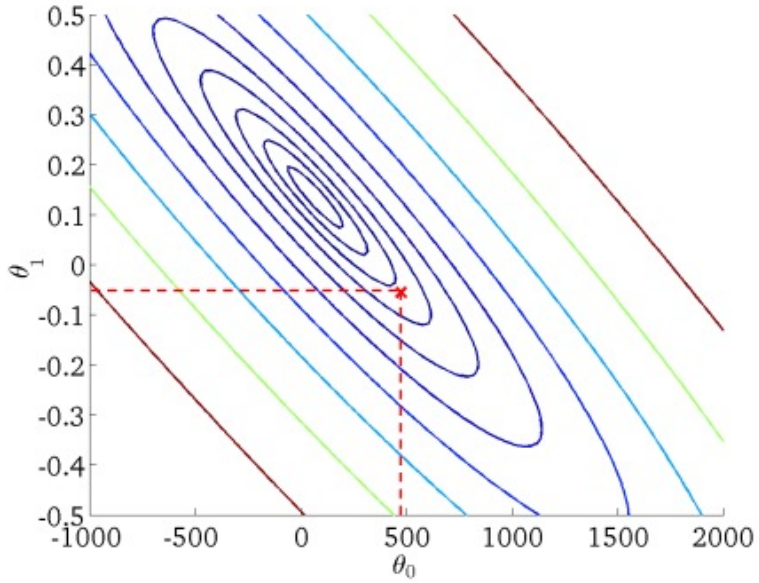
$h_{\boldsymbol{\theta}}(x)$

(for fixed θ_0, θ_1, this is a function of x)



$\mathcal{L}(\boldsymbol{\theta}_0, \boldsymbol{\theta}_1)$

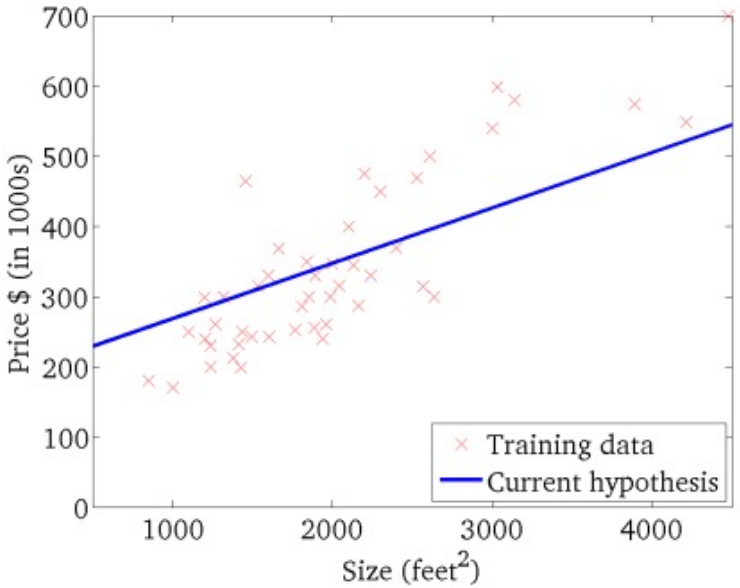
(function of the parameters θ_0, θ_1)



Cost function

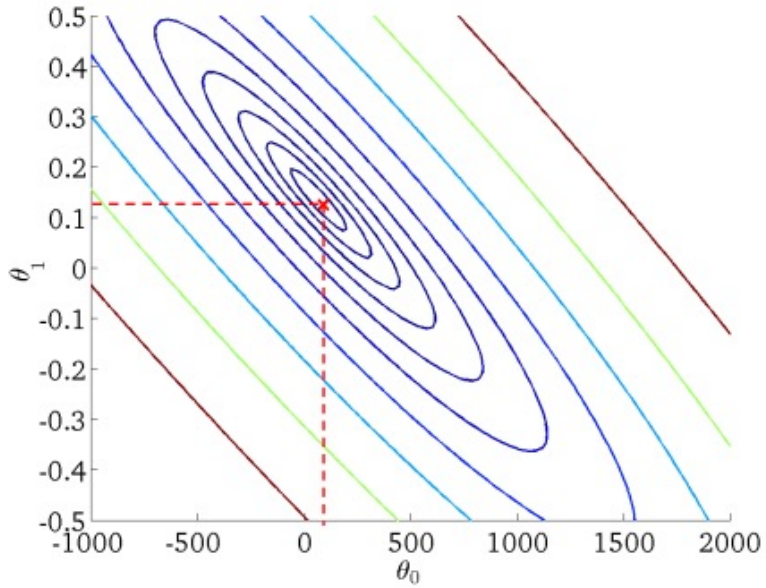
$h_{\boldsymbol{\theta}}(x)$

(for fixed θ_0, θ_1, this is a function of x)



$\mathcal{L}(\boldsymbol{\theta}_0, \boldsymbol{\theta}_1)$

(function of the parameters θ_0, θ_1)

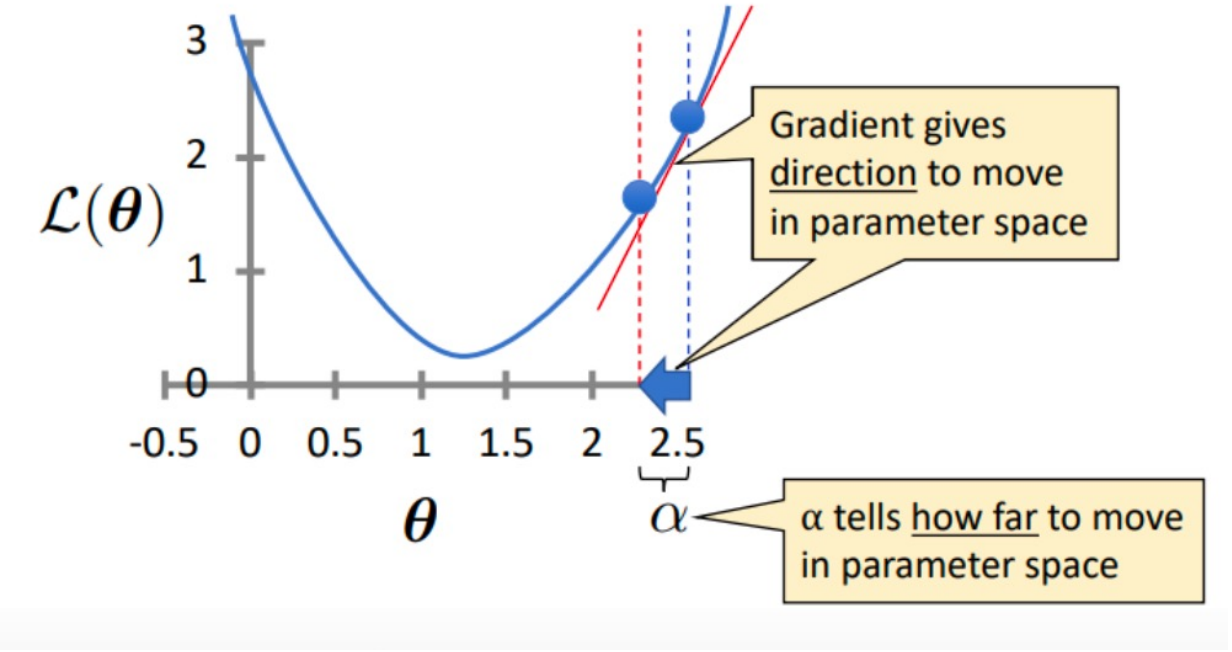


How do we do this?

- Calculus!
  - We want to know where is “down”in the curve, i.e. the gradient of the curve
- In practice: For each coefficient in the model, ϑj
- New $\vartheta j = \vartheta j - \alpha \frac{dL(\vartheta)}{d\vartheta_j}$

α is positive = learning rate(a small value, can be adaptive)

How do we do this?



New $\vartheta j = \vartheta j - \alpha \frac{d\mathrm{L}(\vartheta)}{d\vartheta_j}$

if $\frac{d\mathrm{L}(\vartheta)}{d\vartheta_j} > 0$, then

New $\vartheta j = \vartheta j$ – <positive number>

ϑj will decrease

if $\frac{d\mathrm{L}(\vartheta)}{d\vartheta_j} < 0$, then

New $\vartheta j = \vartheta j$ – <negative number>

ϑj will increase

10 minutes break

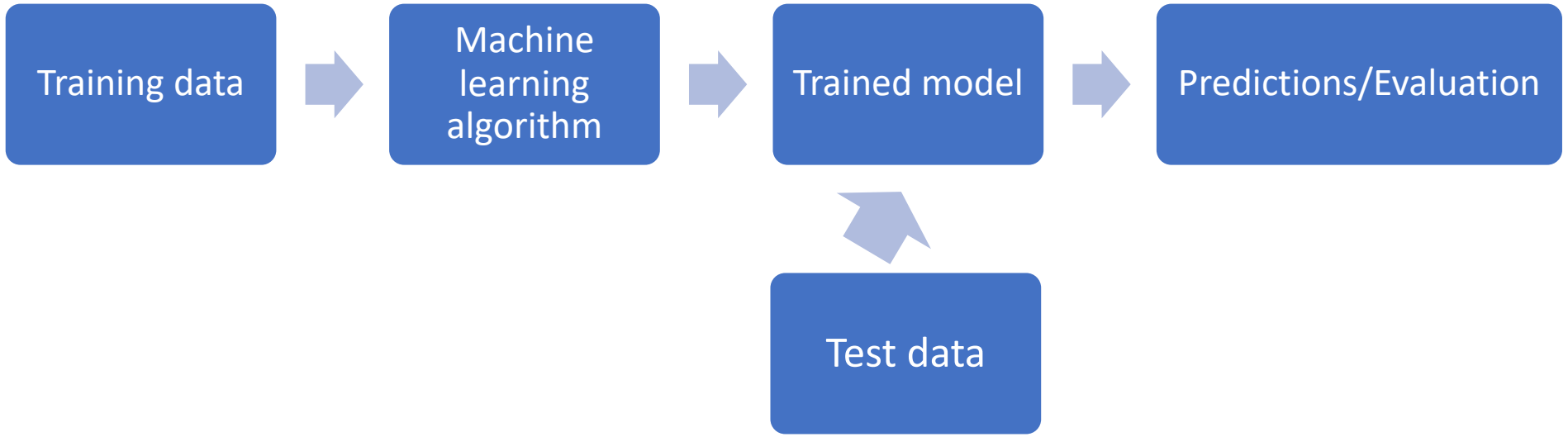
Data splits and cross validation

Data splits

- Okay so we learn this function on our data.
- Are we sure that this is going to be effective on future data?

- We need to split data to evaluate the performance of the model.

Supervised Learning – General Workflow



Why splitting data?

- **We cannot use the same data to evaluate the generalization error**
- If we use some data to compare between models, the MSE of the best model is not an unbiased estimate of the generalization error. We need a new dataset (the test data) to estimate the generalization error.

Why splitting data?

- Why? To avoid overfitting (model that is only working well on particular data and can’t generalise)
- Suppose that you’re a teacher writing an exam for some students [models]. If you want to evaluate their skills, will you give them exercises [observations] that they have already seen and solved [train set], and that they still have on their desks, or new exercises, inspired by what they learned, but different from them? [different set] (jpl, stackoverflow)

Why splitting data?

- Compare statistical methods (e.g., linear regression vs knn regression) to find which fits the best on my data
- Compare models with different predictors included (e.g. linear regression including predictors $[X_1, X_2]$ vs $[X_1, X_2, X_3]$)
- Comparing models with different hyperparameters (e.g. KNN regression using the closest 3 vs 10 neighbours) to find the optimal parameters

Train/dev/test

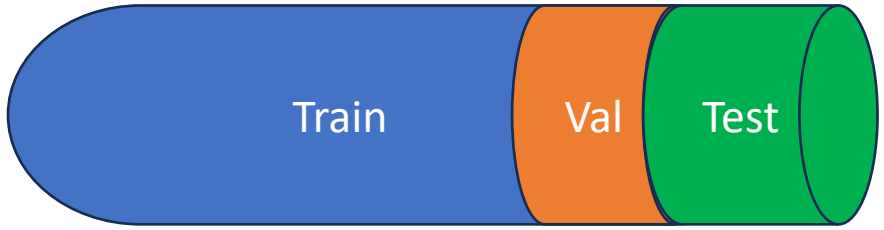
General framework

- Training dataset: To train the models
- Validation dataset: To select the best model
- Test dataset: To estimate the generalization error of the best model

- Often, the terms “validation set”, “dev set”, “test set” are used interchangeably

Train/dev/test

- Training data: Observations used to train ("fit", "estimate") $f(\boldsymbol{x})$
- Validation data (or "dev" data): New observations from the same source as training data
- Used several times to select model complexity
- Test data: New observations from the intended prediction situation
- A very common split is 70% training set, 10% validation, 20% test



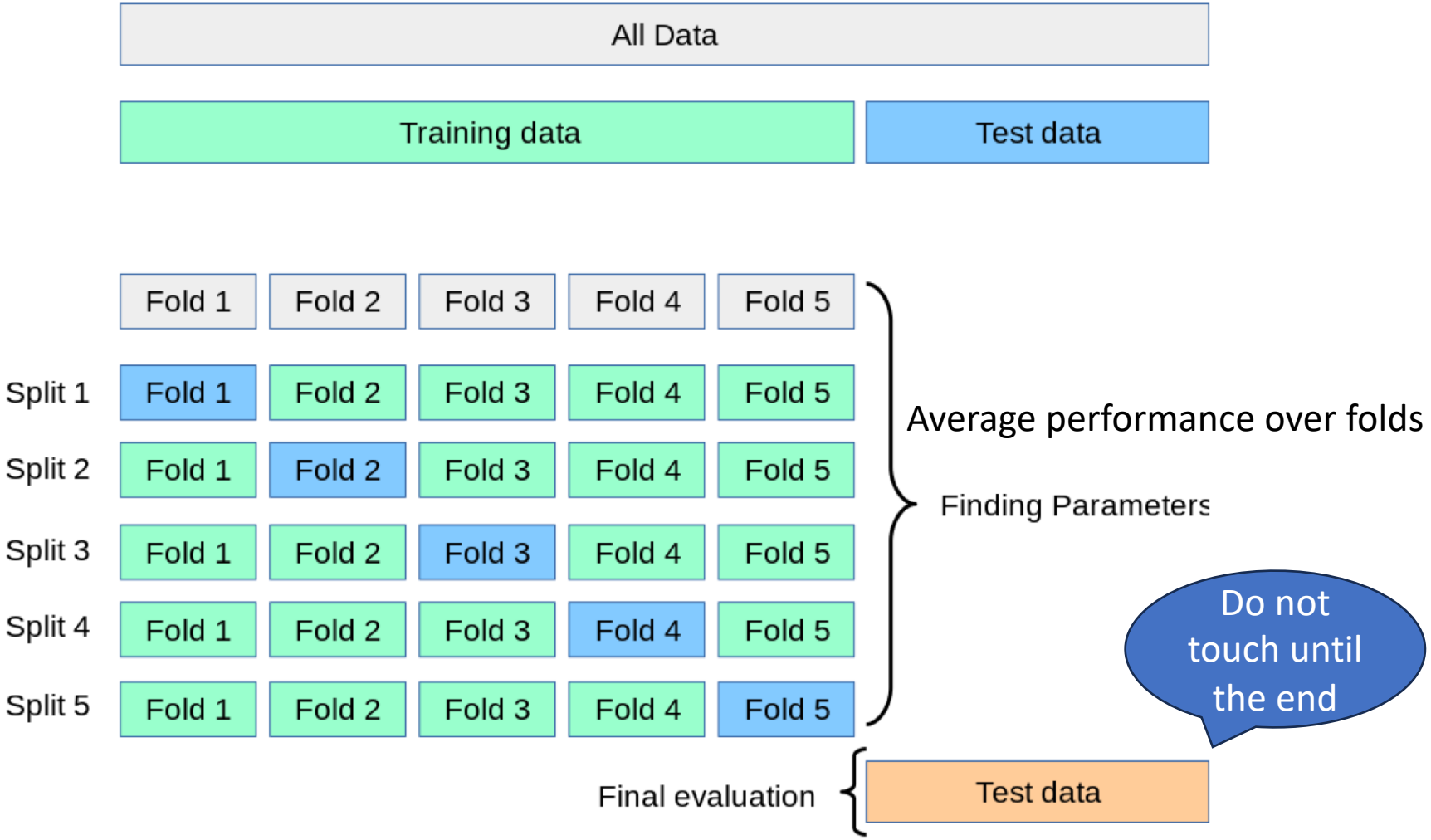
Drawbacks of train/dev/test

- **Fixed Split**: The performance of the model might heavily depend on how the data is split. If the split is not representative, the model may perform poorly in practice
- **Overfitting to Validation Set**: Hyperparameter tuning using the validation set can lead to overfitting to this specific set, reducing the model's ability to generalize to new data
- **Reduced Training Data**: Splitting the data into train, validation, and test sets reduces the amount of data available for training. This can be particularly problematic for small datasets, where having more training data can significantly improve model performance

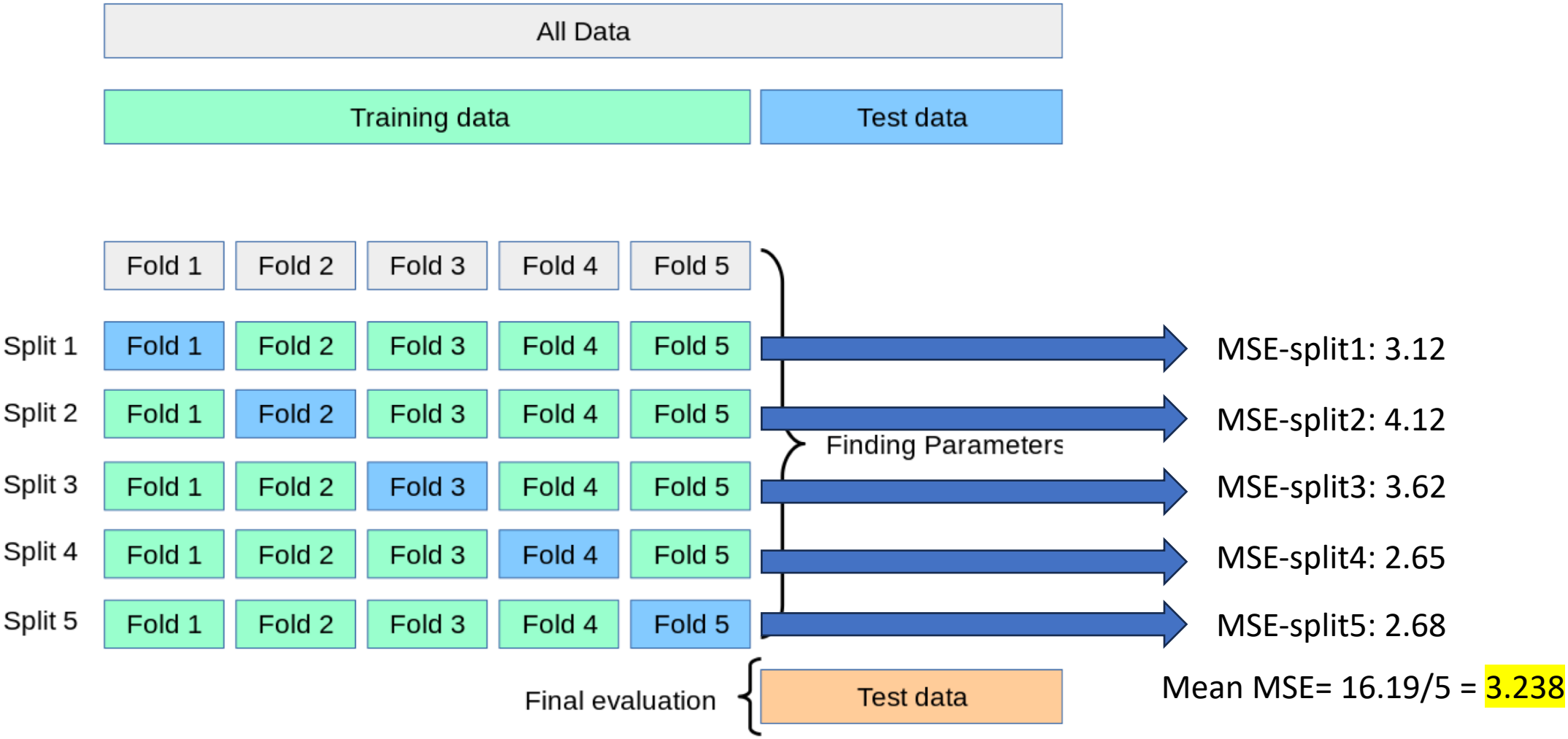
K-fold cross validation

- “Cross-validation” often used to replace single dev set approach;
- Perform the train/dev split several times, and average the result
- When K = 1, “leave-one-out”;
- Usually K = 5 or K = 10

K-fold cross validation

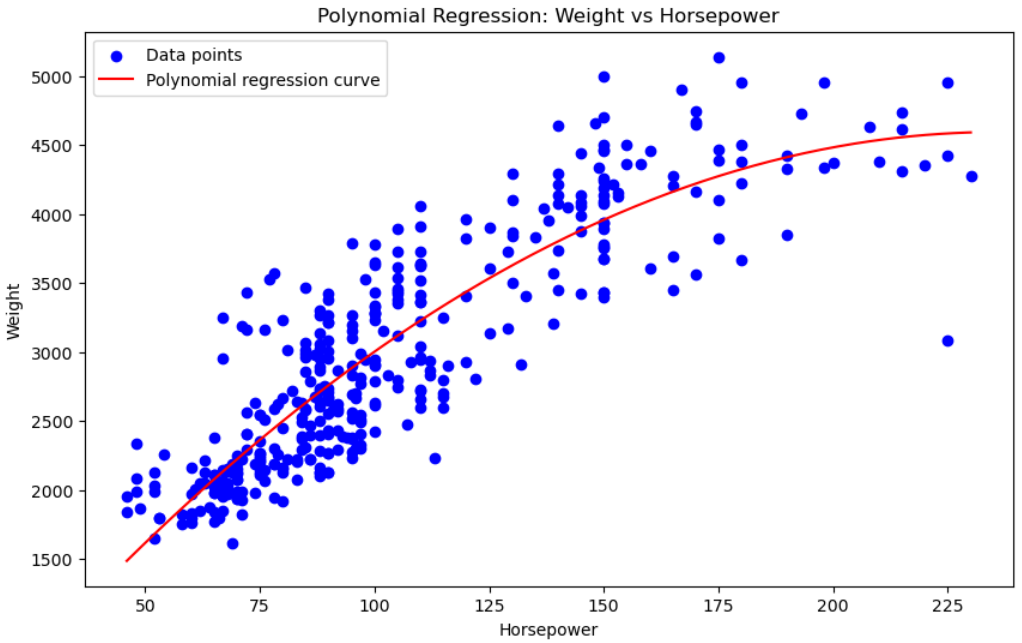


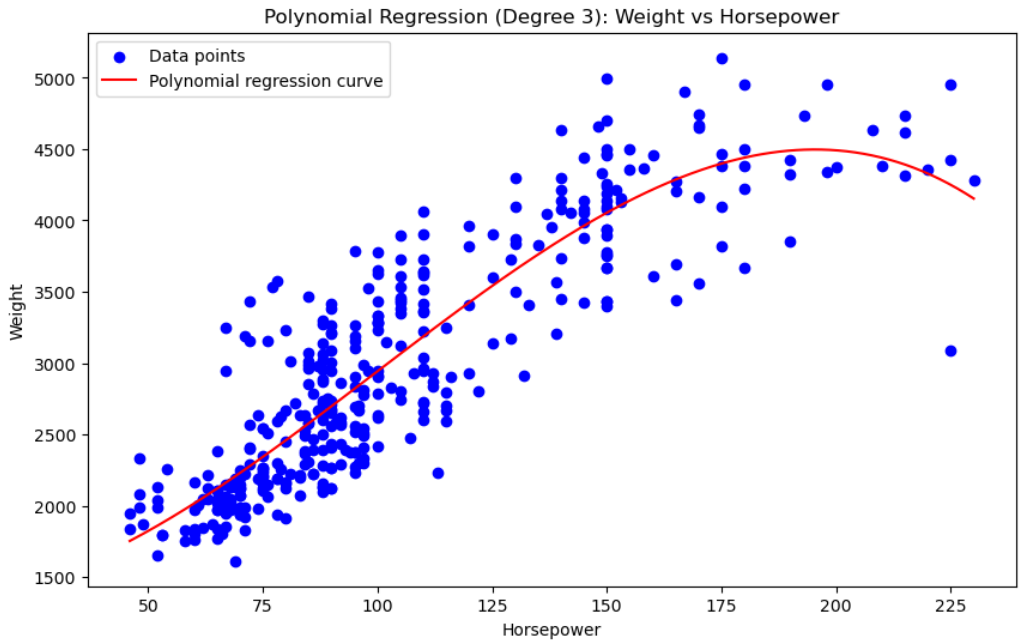
K-fold cross validation



Feature Selection

Polynomials





weight= $\beta_0 + \beta_1 \cdot$ horsepower $+ \beta_2 \cdot$ horsepower^2

weight= $\beta_0 + \beta_1 \cdot$ horsepower $+ \beta_2 \cdot$ horsepower$^2 + \beta_3 \cdot$ horsepower^3

More features

- What if I decide to include more predictors in the model?

- weight=$\beta_0 + \beta_1 \cdot$ horsepower $+ \beta_2 \cdot$ displacement
- weight=$\beta_0 + \beta_1 \cdot$ horsepower $+ \beta_2 \cdot$ displacement $+ \beta_3 \cdot$ acceleration
- …

Features

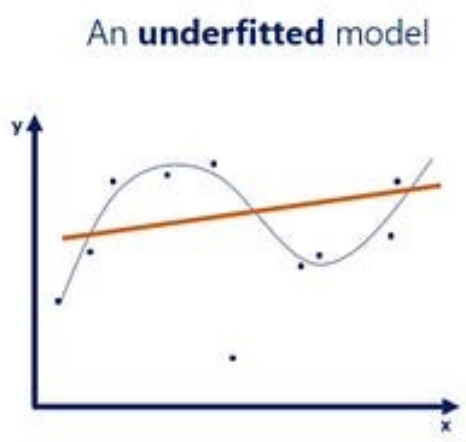
- Choosing the right features has a great impact on the models' performance
- $f(x) = \beta_0 + \beta_1 x_1 + \beta_2 x_2$
- $f(x) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1 x_2$

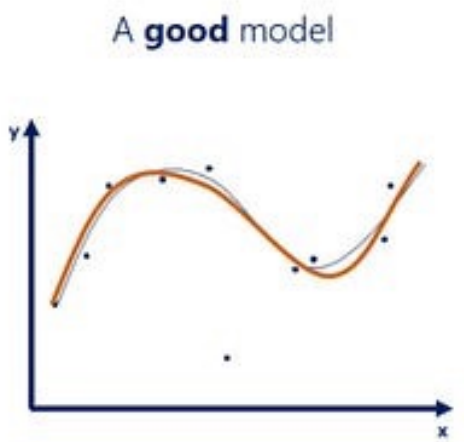
	mpg	cylinders	displacement	horsepower	weight	acceleration	\
0	18.0	8	307.0	130.0	3504	12.0	
1	15.0	8	350.0	165.0	3693	11.5	
2	18.0	8	318.0	150.0	3436	11.0	
3	16.0	8	304.0	150.0	3433	12.0	
4	17.0	8	302.0	140.0	3449	10.5	

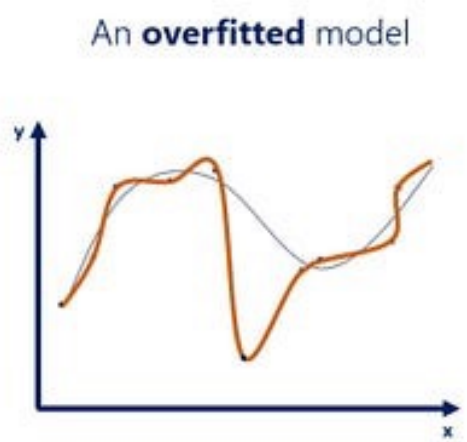
	model_year	origin	name
0	70	usa	chevrolet chevelle malibu
1	70	usa	buick skylark 320
2	70	usa	plymouth satellite
3	70	usa	amc rebel sst
4	70	usa	ford torino

How many variables?

- Too few variables -> Underfitting
- Too many variables -> Overfitting







Feature selection I: Best subset selection

- Try all the different combinations of predictors and choose the one with the best performance
- Computationally not feasible to check all the different combinations of features

if x1, x2, x3 –then try

- f(x1), f(x2), f(x3),
- f(x1, x2), f(x2, x3), f(x1, x3),
- f(x1, x2, x3)

Feature selection II: Forward selection

- We begin with the null model—a model that contains an intercept but no predictors
- We then fit *p* simple linear regressions (*p* is the number of features)
- Add to the null model the variable that results in the lowest RSS.
- We then add to that model the variable that results in the lowest RSS for the new two-variable model.
- This approach is continued until some stopping rule is satisfied

Feature selection II: Forward selection

if x1, x2, x3 –then try :

- f(x1), f(x2), f(x3),
- Find the variable with the lowest RSS (e.g., x1)

- Keep x1 and add one more: f(x1, x2), f(x1, x3),
- Find the model with the lowest RSS (e.g., x1, x2)
- f(x1, x2, x3)

- Until all remaining variables to consider have a p-value larger than some specified threshold, if added to the model.

Feature selection III: Backward selection

- We start with all variables in the model
- Remove the variable with the largest *p-value*—that is, the variable that is the least statistically significant
- The new (p − 1)-variable model is fit, and the variable with the largest *p-value* is removed.
- This procedure continues until a stopping rule is reached.
- For instance, we may stop when all remaining variables have a *p-value* below some threshold.

Feature selection - number of models

- Number of models fitted at each step, example for a dataset with 20 predictors:

Method	Step 1	Step 2	Step 3	...
Best subset	$k = 1$ $\binom{20}{1} = 20$ models	$k = 2$ $\binom{20}{2} = 190$ models	$k = 3$ $\binom{20}{3} = 1140$ models	... ...
Forward	$k = 0$ 20 - 0 = 20 models	$k = 1$ 20 - 1 = 19 models	$k = 2$ 20 - 2 = 18 models	... ...
Backward	$k = 20$ 20 models	$k = 19$ 19 models	$k = 18$ 18 models	... ...

Regularization

Regularization

- A very flexible model (one with many coefficients) is like a kid in candyshop with a platinum credit card: It goes around buying all the coefficients it wants and never stops.
- Idea: Tell the model not to go overboard with the complexity. We set up the correct complexity as the one that minimizes MSE in the validation data.

Penalized (regularized) regression: buying coefficients on a budget

- We want to fit the training data (estimate the weights of the coefficients)
- Make the model behave ‘regularly’ by penalizing the purchase of ‘too many’ coefficients
- Extremely efficient way to approximately solve the best subset problem: Variable selection + regression in one step
- Often yields very good results
- If you are interested in prediction and not inference (i.e. if identifying the relevant features is not a primary goal of the analysis), regularization will usually be better

Regularization: buying coefficients on a budget

- Usually, find the ϑj (or sometimes we use the notation βj) that minimizes
- $L(\vartheta) = MSE = \frac{1}{n}\sum_{i=1}^{n}(yi - h_{\vartheta}(x_i))^2$

- Now, find the ϑ_j that minimizes $L(\vartheta) = MSE + \lambda \cdot Penalty$

where the penalty is:

- $\sum_{j>0} |\vartheta_j|$-> (L1, Lasso) -> Tends to set some coefficients to zero (great for interpretability)
- $\sum_{j>0} \vartheta_j^2$-> (L2, Ridge) -> Tends to keep all coefficients

How to select λ – option 1

Option 1:

- Divide the data into train/val/test
- Create models using different λ, fit them using the train data.
- Calculate MSE in the validation data and select the best model.
- Estimate generalization error for the best model in the test dataset.

How to select λ – option 2

Option 2 (better):

- Divide the data into train/test
- Use cross-validation, for each k split of train –> train/val:
- - Fit models using different λ.
- - Calculate MSE in the validation dataset
- Select the model with the minimum average MSE.
- Estimate generalization error in the test dataset

Regularization in Python

- *alpha* in scikit-learn is equivalent to *lambda*

```
from sklearn.linear_model import Lasso, Ridge

# Fit Lasso and Ridge regression models
lasso = Lasso(alpha=10.0)
lasso.fit(X_train, y_train)

ridge = Ridge(alpha=10.0)
ridge.fit(X_train, y_train)

# Predict using both models
y_pred_lasso = lasso.predict(X_test)
y_pred_ridge = ridge.predict(X_test)
```

Regularization in Python

- *alpha* in scikit-learn is equivalent to *lambda*

```
# Prepare LassoCV with a range of alphas for cross-validation
alphas=np.logspace(-4, 4, 100)
lasso_model = LassoCV(alphas=alphas, cv=5)
```

```
# Prepare RidgeCV with a range of alphas for cross-validation
ridge_model = RidgeCV(alphas = alphas, cv=5)

# Fit the RidgeCV model to find the best alpha
ridge_model.fit(X_train_scaled, y_train)

# Extract the best alpha value
optimal_alpha_ridge = ridge_model.alpha_

print("Optimal Alpha for Ridge:", optimal_alpha_ridge)
```

Summary

Conclusion

- Linear regression is a parametric model, and assumes linear relationship between predictors and target variable
- Split data or use cross validation in your problem
- It is better to standardize the predictors

Conclusion

- By using feature selection or regularization, we can obtain better prediction accuracy and model interpretability
- Feature selection includes best subset, forward and backward selection
- Best subset selection performs best, but it comes at a prize
- Regularization includes LASSO and Ridge
- LASSO shrinks unimportant parameters to truly zero, while Ridge shrinks them to small values

Practical 2