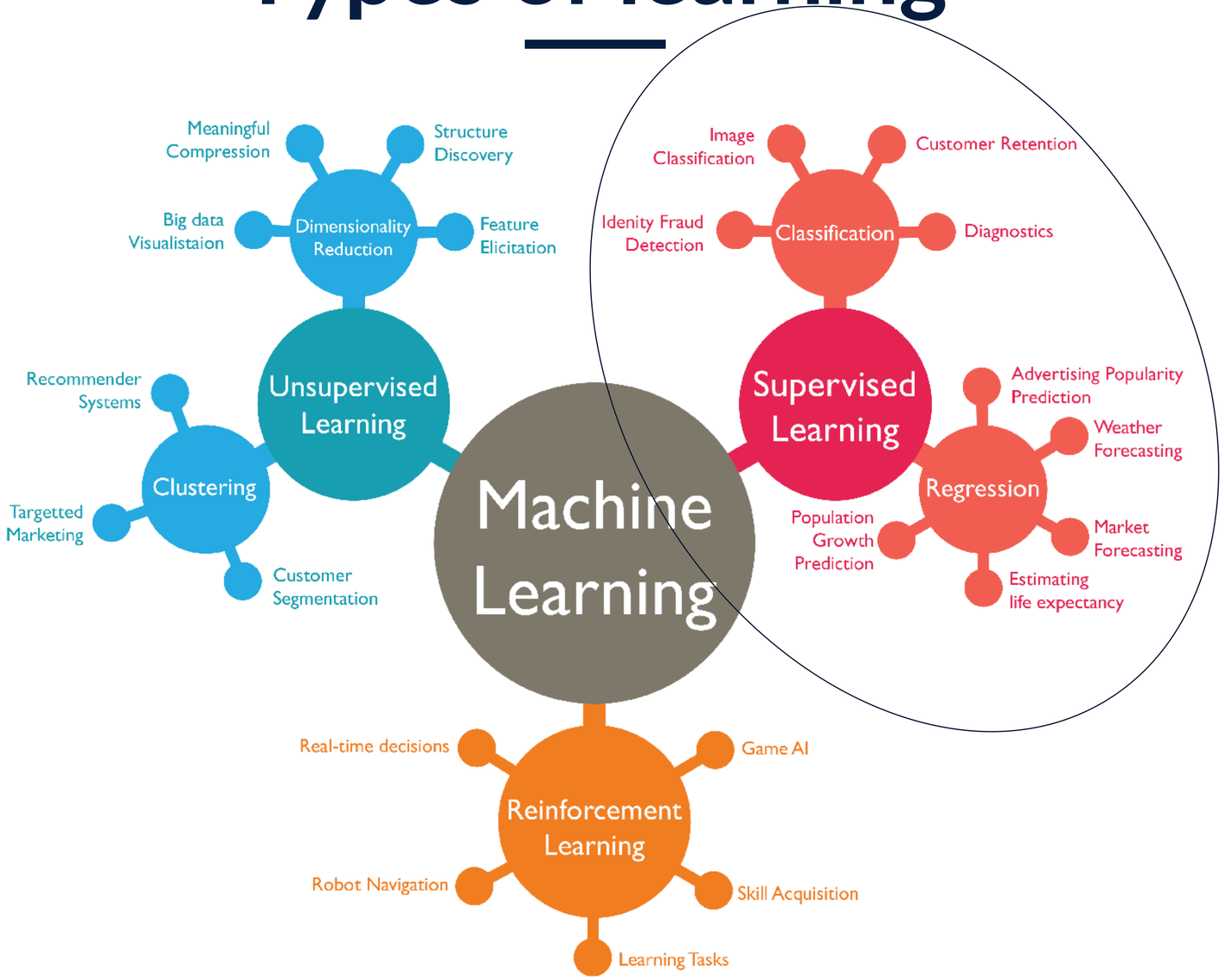
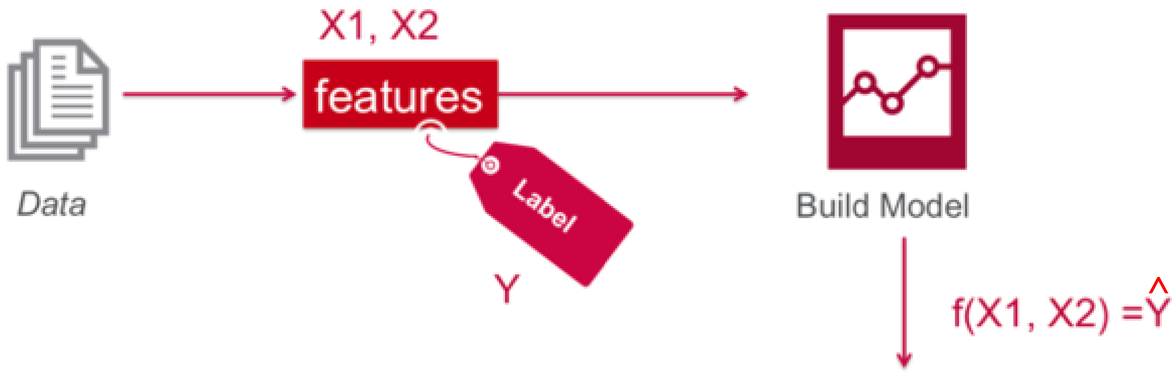
Machine learning
Regression

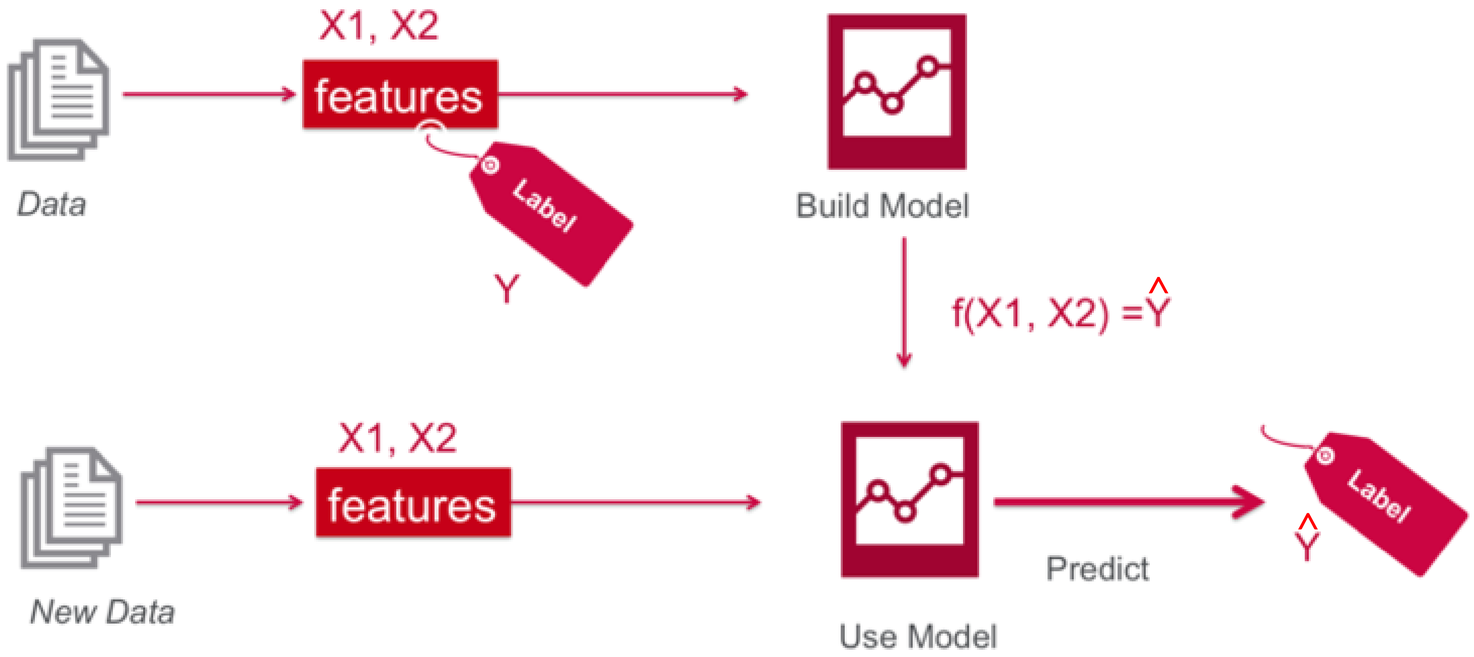
Types of learning



Supervised learning



Supervised learning



Univariate linear regression

Heart disease vs Biking to work

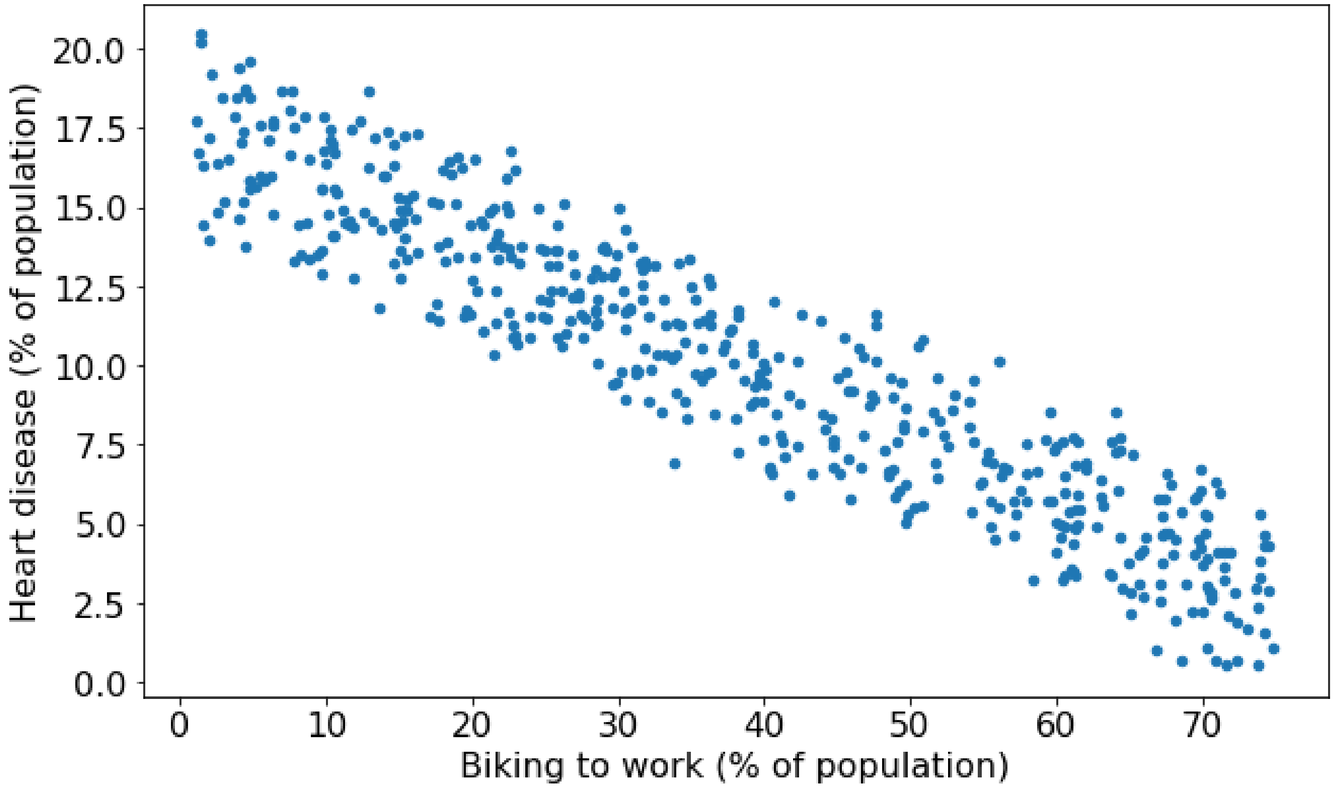

```
data = pd.read_csv(”heartData.csv")

data.head()
```

	biking	smoking	heart.disease
0	30.801246	10.896608	11.769423
1	65.129215	2.219563	2.854081
2	1.959665	17.588331	17.177803
3	44.800196	2.802559	6.816647
4	69.428454	15.974505	4.062224

```
ax = data.plot.scatter('biking','heart.disease’)

ax.set_xlabel("Biking to work (% of population)")
ax.set_ylabel("Heart disease (% of population)")
```



Heart disease vs Smoking

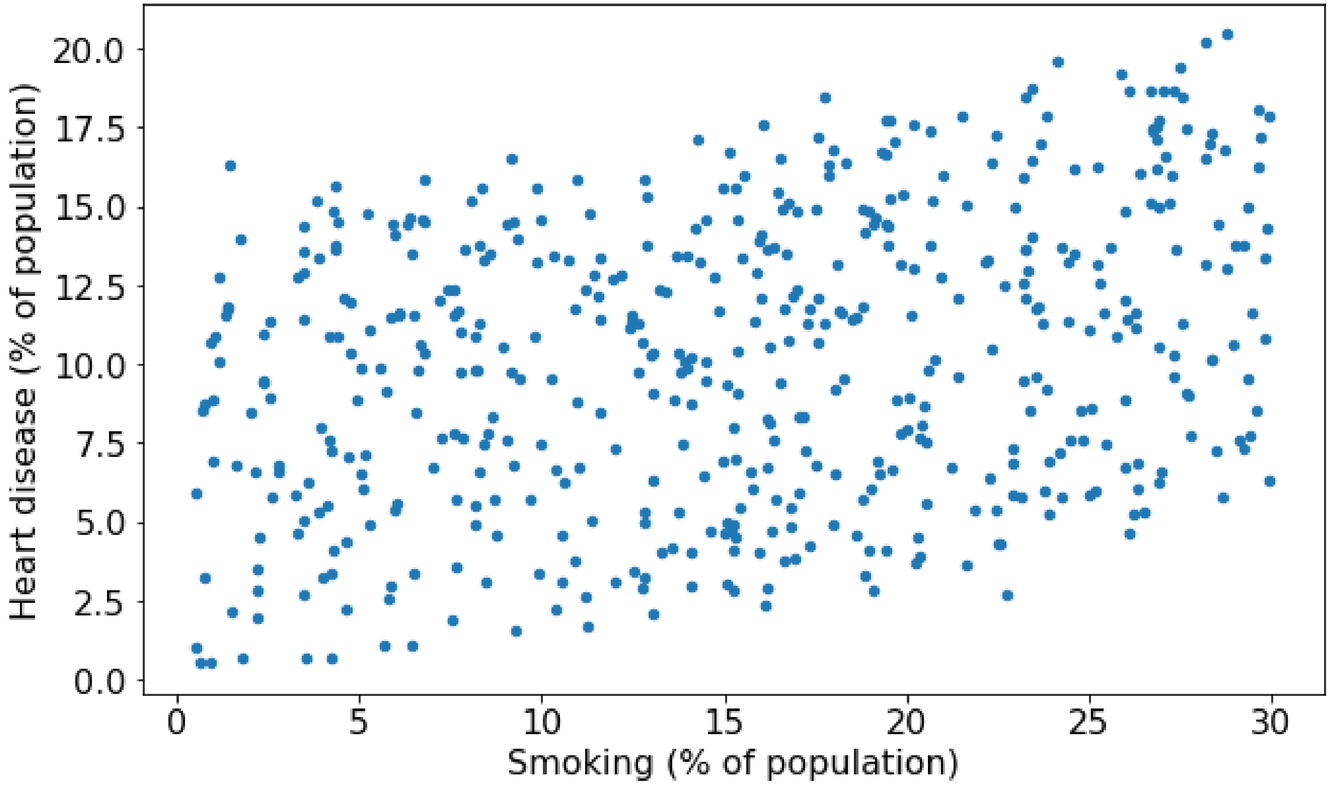

```
data = pd.read_csv(”heartData.csv")

data.head()
```

	biking	smoking	heart.disease
0	30.801246	10.896608	11.769423
1	65.129215	2.219563	2.854081
2	1.959665	17.588331	17.177803
3	44.800196	2.802559	6.816647
4	69.428454	15.974505	4.062224

```
ax = data.plot.scatter(smoking','heart.disease’)

ax.set_xlabel(”Smoking (% of population)")
ax.set_ylabel("Heart disease (% of population)")
```



Univariate linear regression

- Supervised learning model
- Linear & Univariate, hence the model is a straight line

$$y = \underbrace{w_0 + w_1 x}_{\text{linear model}} + \underbrace{\epsilon}_{\text{error}}$$

- y is the Dependent Variable
- x is the Independent Variable
- w_1 is the Slope of the line
- w_0 is the Coefficient (or the y-intercept)

Univariate linear regression

$$y = \underbrace{w_0 + w_1 x}_{\hat{y}} + \qquad \epsilon$$

Univariate linear regression

$$y = \underbrace{w_0 + w_1 x}_{\hat{y}} + \underbrace{\epsilon}_{\sim \mathcal{N}(0, \sigma^2)}$$

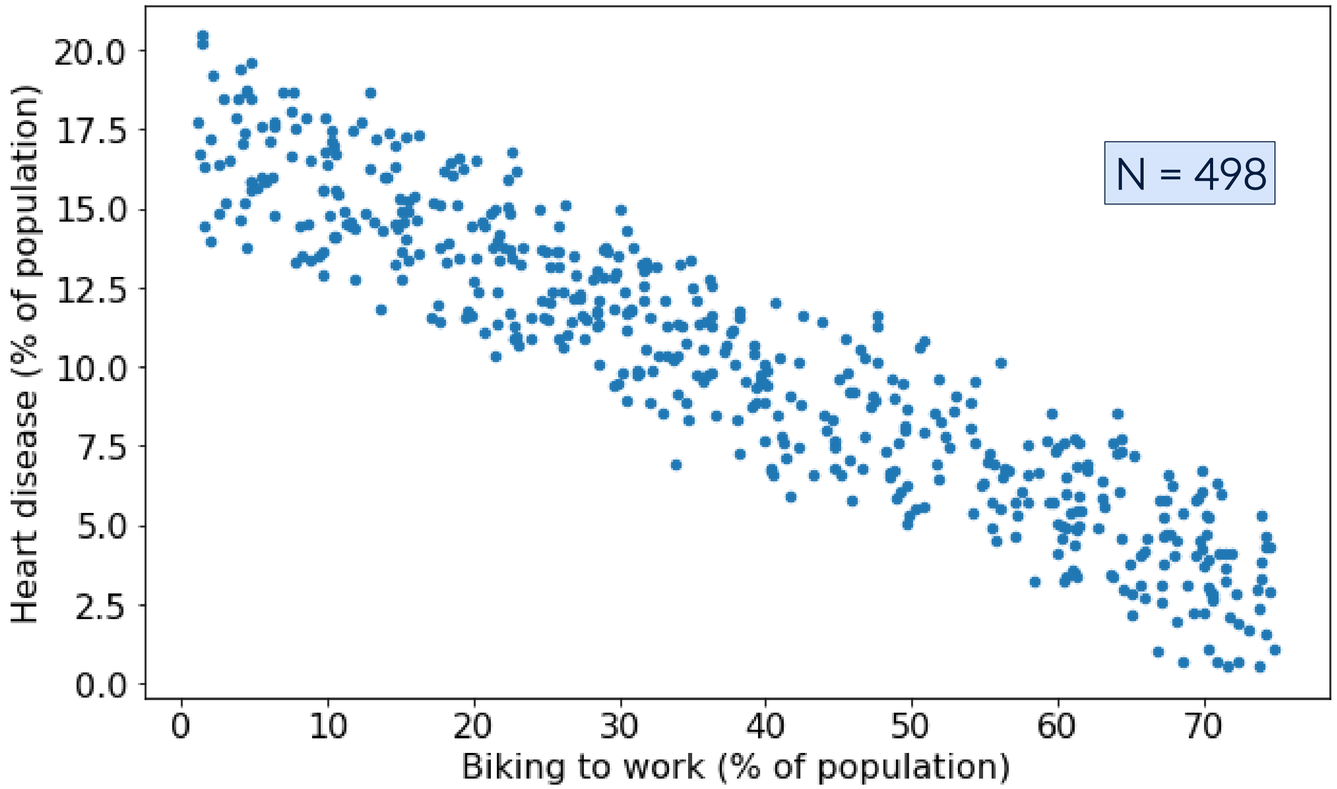
Univariate linear regression

$$y = \underbrace{w_0 + w_1 x}_{\hat{y}} + \underbrace{\epsilon}_{\sim \mathcal{N}(0, \sigma^2)}$$

⚠ $(x, y) = \{(x^i, y^i)\}_{i=1}^N$

Heart disease vs Biking to work

$$\text{heart.disease} = \underbrace{w_0 + w_1 \, \text{biking}}_{\text{heart.disease}} + \epsilon$$

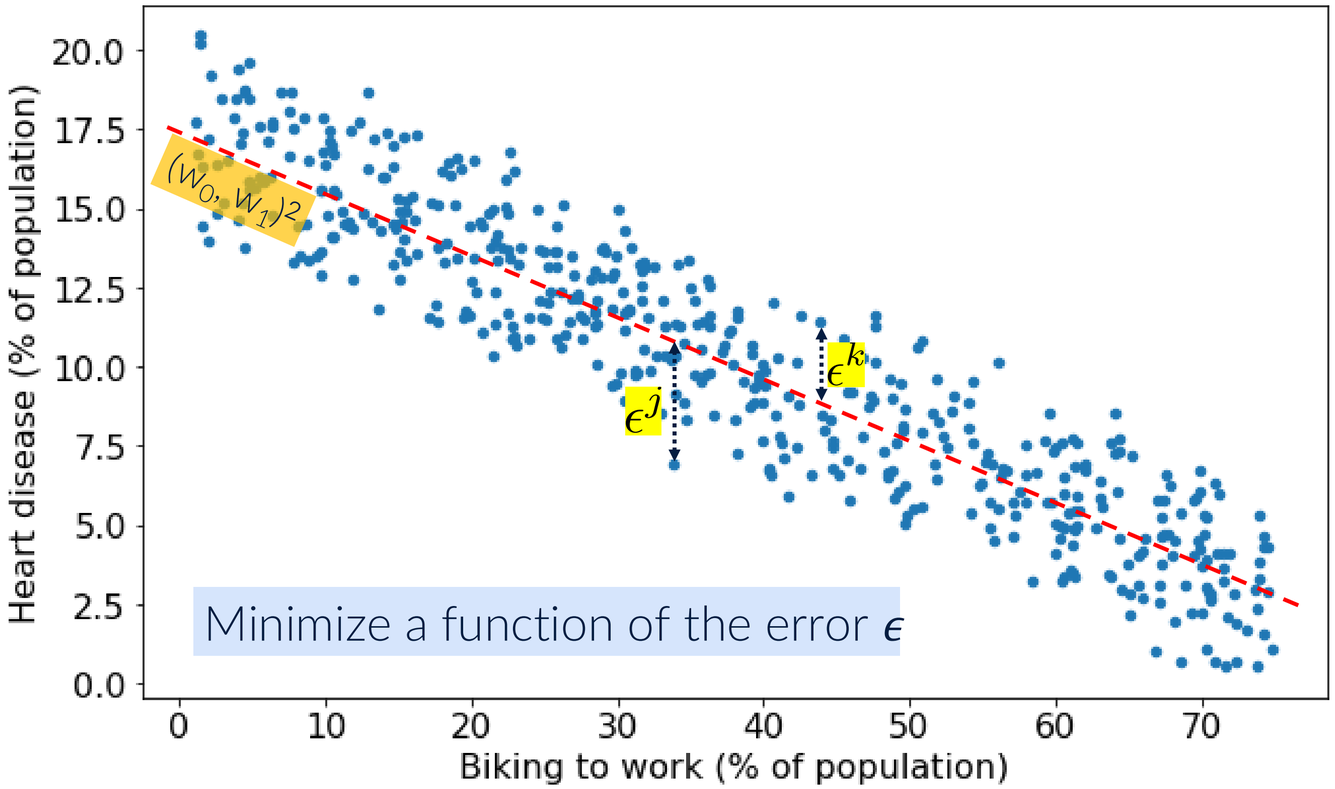


Heart disease vs Biking to work

$$\text{heart.disease} = \underbrace{w_0 + w_1 \, \text{biking}}_{\text{heart.disease}} + \epsilon$$

Heart disease vs Biking to work

$$\text{heart.disease} = \underbrace{w_0 + w_1 \, \text{biking}}_{\text{heart.disease}} + \epsilon$$



Error (or cost, loss) function

$$\text{heart.disease} = \underbrace{w_0 + w_1 \, \text{biking}}_{\widehat{\text{heart.disease}}} + \epsilon$$

$$\epsilon = \text{heart.disease} - \widehat{\text{heart.disease}}$$

Where,

heart.disease - Actual output

$\widehat{\text{heart.disease}}$ - Predicted output

Common cost functions for regression

Mean squared error (MSE)

$$\frac{1}{N}\sum_{i=1}^{N}\left(\widehat{\text{heart.disease}}^{i} - \text{heart.disease}^{i}\right)^2$$

Mean absolute error (MAE)

$$\frac{1}{N}\sum_{i=1}^{N}\left|\widehat{\text{heart.disease}}^{i} - \text{heart.disease}^{i}\right|$$

Where, $\widehat{\text{heart.disease}}^{i} = w_0 + w_1 \text{biking}^{i}$

The best model in the least squares

Ordinary least squares (OLS)

$$y = \underbrace{w_0 1 + w_1 x}_{\hat{y}} + \epsilon$$

$$\hat{y} = \tilde{\mathbf{X}}\mathbf{w} \text{ where, } \mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix} \text{ and } \tilde{\mathbf{X}} = \begin{bmatrix} 1 & x^1 \\ 1 & x^2 \\ \vdots & \vdots \\ 1 & x^N \end{bmatrix}$$

We seek to find $\mathbf{w}^*$ that minimizes the MSE.

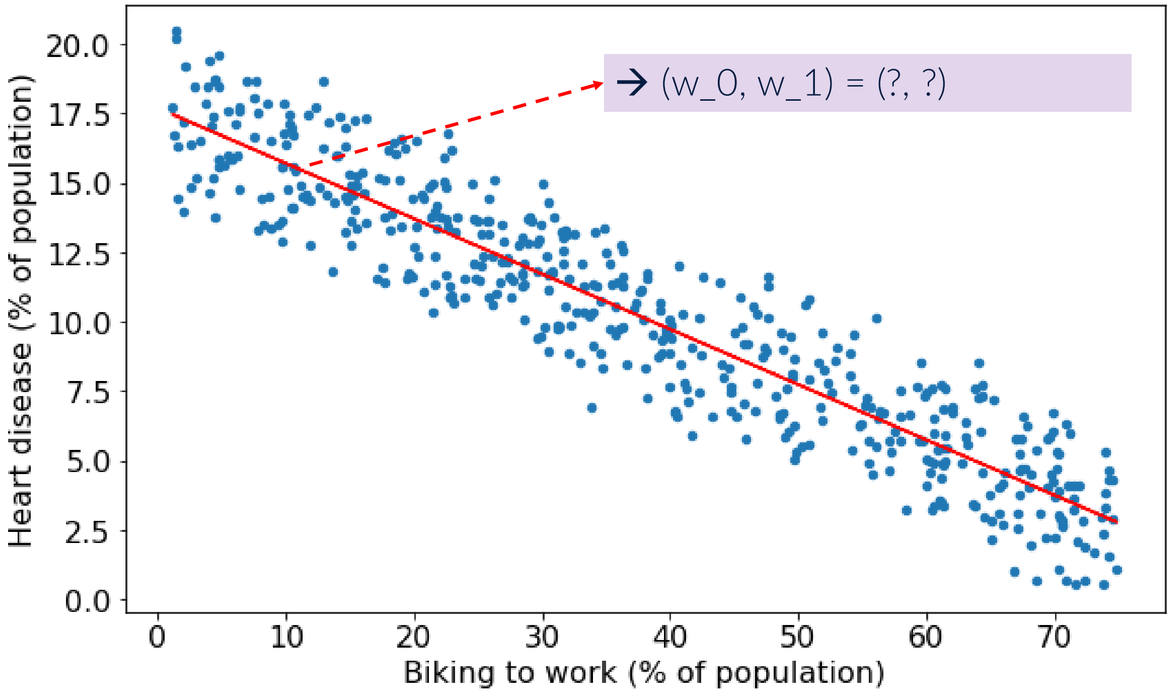
$$\mathbf{w}^* = \left(\tilde{\mathbf{X}}^\mathbf{T}\tilde{\mathbf{X}}\right)^{-1}\tilde{\mathbf{X}}^\mathbf{T}\mathbf{y}$$

cf. cours d'Algèbre Linéaire - Chapitre 12 de S. Friedli (page 191

An example in (pure) Python

Exercise: Find the best line in the least squares sense

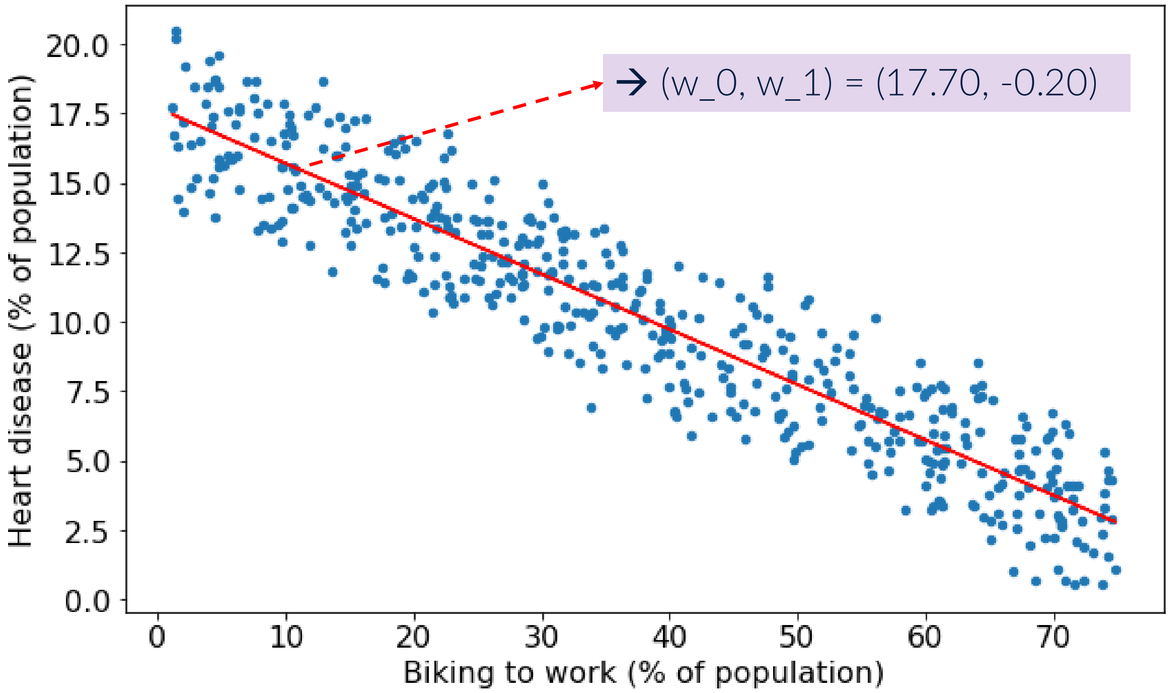
using the formulation of the OLS solution from the previous slide



Now using Scikit-Learn

Data preparation

```
X = data.iloc[:,0:1].values
y = data.iloc[:,-1].values
```



w* via OLS method using Scikit-Learn

```
from sklearn.linear_model import LinearRegression

model = LinearRegression()
model.fit(X, y)

print("(w_0, w_1) = ({:2.2f}, {:2.2f})".format(model.intercept_, model.coef_[0]))
```

Multivariate linear regression

Extension of the univariate model

$$y = \underbrace{w_0 1 + w_1 x_1 + w_2 x_2 + \ldots + w_m x_m}_{\hat{y}} + \epsilon$$

$$\hat{y} = \tilde{\mathbf{X}}\mathbf{w} \text{ where, } \mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_m \end{bmatrix} \text{ and } \tilde{\mathbf{X}} = \begin{bmatrix} 1 & x_1^1 & x_2^1 & \ldots & x_m^1 \\ 1 & x_1^2 & x_2^2 & \ldots & x_m^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_1^N & x_2^N & \ldots & x_m^N \end{bmatrix}$$

OLS solution: $\mathbf{w}^* = \left(\tilde{\mathbf{X}}^\mathbf{T}\tilde{\mathbf{X}}\right)^{-1}\tilde{\mathbf{X}}^\mathbf{T}\mathbf{y}$

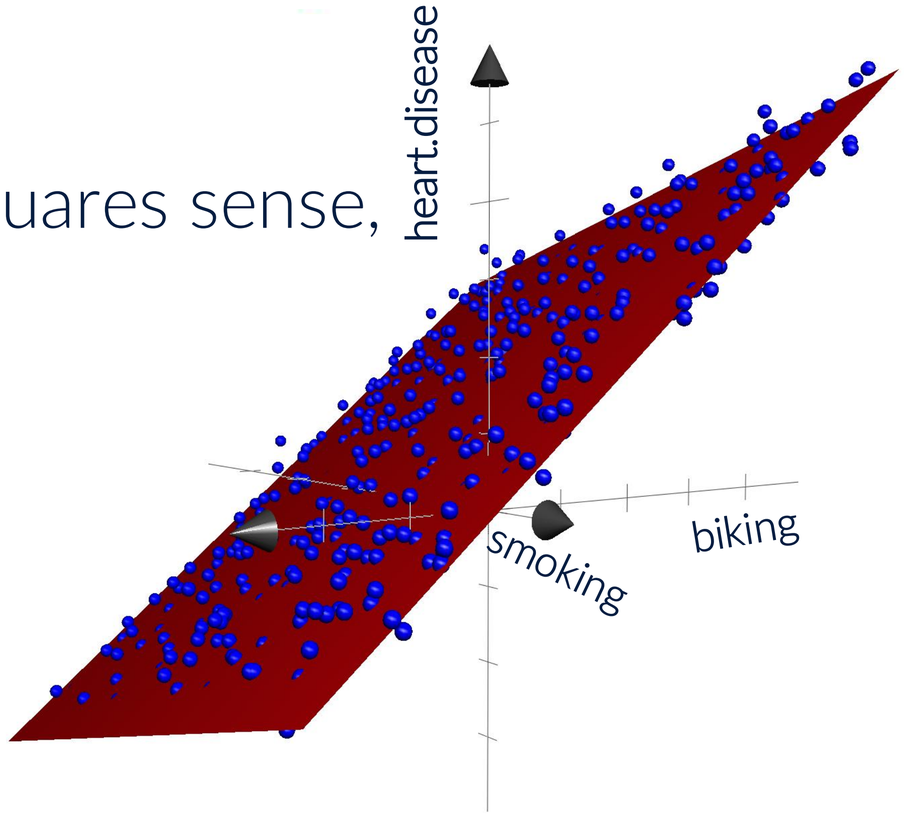
cf. cours d'Algèbre Linéaire - Chapitre 13 de S. Friedli (page 203

Back to our example

Using both features / independent variables

$$\text{heart.disease} = \underbrace{w_0 1 + w_1 \text{biking} + w_2 \text{smoking}}_{\widehat{\text{heart.disease}}} + \epsilon$$

Exercise: Find the best (hyper)plane in the least squares sense, and comment on the results.



Performance evaluation

Regression

Two common metrics

- Mean Absolute Error $$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$$
- (Root) Mean Squared Error $$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$
  - Since the error is squared, any prediction error is being heavily penalized

Linear Regression as Predictive Model

- An example –
  - Let's take our **multivariate** model
  - Can we predict the proportion of **heart disearse** for (**biking** = 30%, **smoking** = 5.5%)?
  - Note the tuple (**biking** = 30%, **smoking** = 5.5%) does **not** exist in the original dataset

```
X_pred     = [[30.0, 5.5]]

prediction = model_multivariate.predict(X_pred)

print("»» Prédiction maladie cardiaque: {:2.1f}%".format(prediction[0]))
```

```
>>> Prédiction maladie cardiaque: 10.0%
```

How well do we do though?

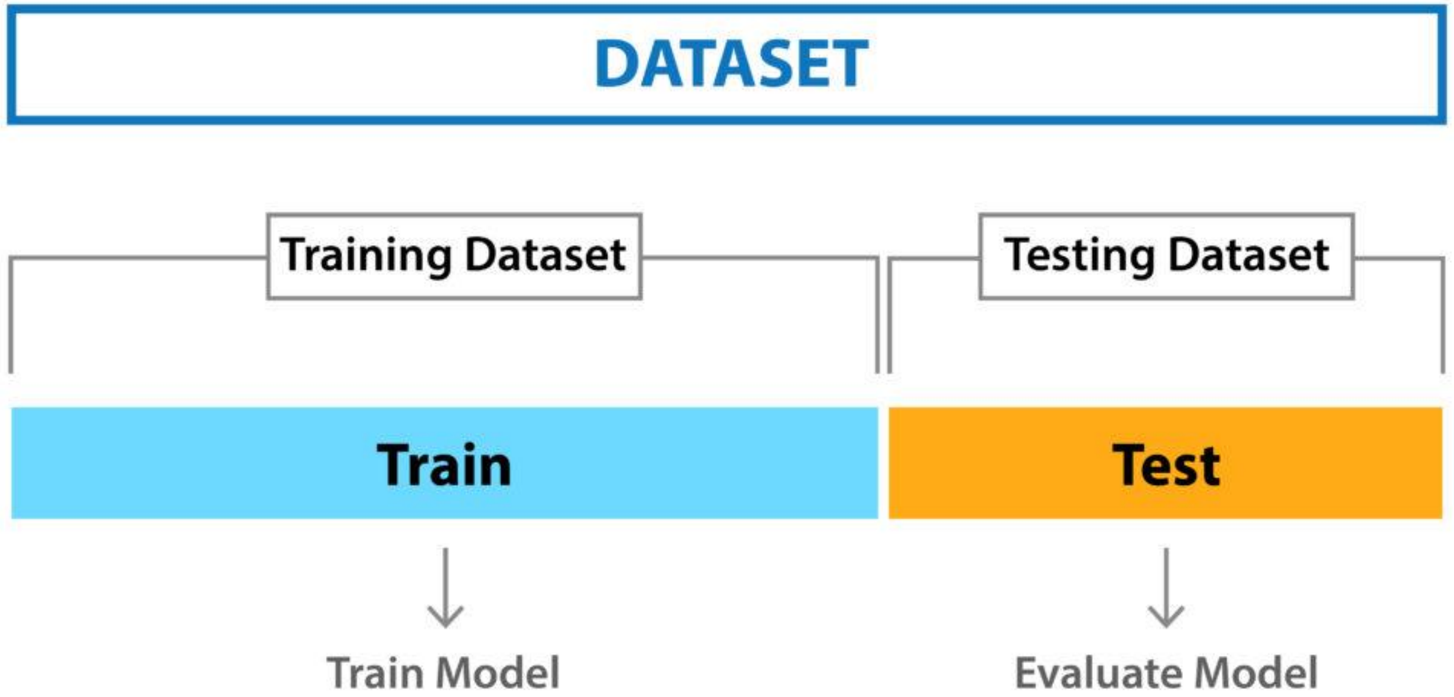
- The train-test split procedure is used to estimate the performance of machine learning algorithms when they are used to make predictions on data not used to train the model

    - **Train Dataset**: Used to fit the machine learning model
    - **Test Dataset**: Used to evaluate the fit machine learning model

    Common split percentages include:

    - Train: 80%, Test: 20%
    - Train: 67%, Test: 33%
    - Train: 50%, Test: 50%

The train-test split procedure




```
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

What did I learn?

- Supervised vs unsupervised learning
- Linear regression as supervised learning model
- Univariate vs multivariate linear regression
- Error / cost / loss function
- OLS solution: method for choosing the unknown parameters in a linear regression model by the principle of least squares (not least absolutes!)
- Performance evaluation of a regression model
  - Common metrics: R^2, MSE, MAE
- Train-test split procedure

QUESTIONS ?